Computer Physics Communications 210 (2017) 72–78

Contents lists available at ScienceDirect

Computer Physics Communications

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

High-performance GPU parallel solver for 3D modeling of electron transfer during ion–surface interaction

I.K. Gainullin

Faculty of Physics, Moscow State University, Leninskie gory 1 # 2, Moscow, 119992, Russia

ARTICLE INFO

Article history:
Received 23 June 2016
Received in revised form
25 September 2016
Accepted 29 September 2016
Available online 8 October 2016

Keywords:
Resonant charge transfer
3D modeling
TDSE
FDM
GPU
High-performance calculations

ABSTRACT

This study presents parallelized time-dependent Schrödinger equation solver (GPU TDSE Solver) for 3D modeling of resonant electron transfer (RCT) during ion–surface interactions and atomic collisions. The computer modeling of RCT process is based on the numerical one-electron TDSE solution in relatively large spatial domain (about 10^3–10^4 nm^3). Due to the numerical complexity of direct 3D TDSE solution in such domains, most of RCT calculations use approximations that reduce problem to 2D calculations (e.g. cylindrical symmetry). Last years the TDSE Solver was developed for 3D RCT modeling in large-scale nanosystems (Gainullin and Sonkin, 2015). It was shown to have rather good performance due to the effective parallel implementation of simple numerical scheme on GPUs (explicit finite-difference method in Cartesian coordinates). Note, that usage of FDM in Cartesian coordinates requires ∼100 times greater numerical grid, comparing to the finite-element or finite-volume methods. For the majority of RCT problems the calculations transfer to the cylindrical coordinates decreases the size of numerical grid by an order of magnitude without loss of the calculations precision. The main problem in the transfer to the cylindrical coordinates is that explicit numerical schemes for parabolic equations, including TDSE, are unstable near the axis $\rho = 0$. This study presents hybrid numerical scheme, which eliminates this instability and preserves the effective parallelization on GPUs. The performance of new version of the GPU TDSE Solver, based on the hybrid numerical scheme, was found to be 6 times greater comparing to the previous version. Such performance gain is stipulated by less discrete points, required for the FDM implementation in cylindrical coordinates. Due to reduction of required GPU memory, new version of GPU TDSE Solver can handle spatial domains about 10^3 nm^3 using an ordinary personal computer (equipped with modern GPU, e.g. Tesla k20 or better) or up to 10^5 nm^3 using supercomputers. GPU TDSE Solver was applied to the calculation of the resonant charge transfer in nanosystems. The calculated neutralization probability for Li+ ions impinging on the Ag(100) surface shows a good quantitative agreement with the experimental data.

1. Introduction

The investigation of electron transfer during the interaction between atomic particles and solid surfaces is of fundamental and practical importance in several branches of physics and chemistry. For fundamental science the electron transfer is of interest due to numerous phenomena such as scattering, sputtering, adsorption, and molecular dissociation [1–3]. The practical importance of the electron transfer is stipulated by such applications as semiconductor miniaturization via thin-film deposition, reactive ion etching, catalysis, surface analysis tools and modification [2,4–6]. An important application is solid body diagnostics by ion beams, where such properties, like chemical composition, crystallographic structure, adsorbates, etc., are determined by analyzing spectra of scattered or sputtered ions. Because the most of experimental equipment is capable to register charged atoms (ions), the atomic neutralization or ionization processes will significantly affect the registered spectra. Therefore, the electron exchange calculations are especially important for the quantitative diagnostics by ion beams.

Since the electron affinity level of atomic anions and ionization level of some alkali ions are significantly higher than the typical Fermi energy of most metallic surfaces, the transfer of electron from anion to the surface has single-electron nature [7]. Thus, the resonant charge transfer (RCT), i.e. energy-conserving one-electron tunneling through the potential barrier between atomic particle and surface, can be considered as a basic electron transfer mechanism for such systems [8].

E-mail address: Ivan.Gainullin@gmail.com.

http://dx.doi.org/10.1016/j.cpc.2016.09.021
0010-4655/

The computer modeling of RCT process between atomic particle and metallic surface is based on the numerical one-electron TDSE solution in relatively large spatial domain (about 10^3–10^4 nm^3). The moving atomic particle and fixed metallic surface are described by independent pseudopotentials. The effect of image charge is considered. The perturbation effects are usually neglected, because RCT process is modeled for the rather large distances (>2 Å). Typically theoretical studies consider atomic particle to be one-active-electron atom/ion (e.g. H^0/H^-, Li^0/Li^+, Na^+, K^+), that can be modeled by analytical pseudopotential [9]. By means of density functional theory (DFT) the pseudopotentials have been calculated for two types of bulk metal surfaces: free-electron metals ("jellium" model) [10] and metals with projected band gap [11].

The existing theory in many cases gives quantitative agreement with experimental data [3,6,7,12–21]. Due to the numerical complexity of direct 3D TDSE solution in large spatial domains, the most of the described researches were performed for the 2D systems (3D with cylindrical or spherical symmetry). The first time the 3D RCT calculations were mentioned in 2000 [22], but the reported system size (10^7 points) was too small to solve real 3D RCT problems. In addition, 3D RCT calculations were reported in 2009, but their details and complexity are omitted [18]. To model 3D experiments by means of 2D calculation method, the electron transfer probability in many studies is calculated by means of the rate equation which can be applied under the adiabatic conditions (adiabatic approximation) [8,12,14,16,23]. The rate equation assumes that electron transition rate at each moment of time is proportional to the atomic particle level width Γ, which depends on the distance to the surface, but not on the projectile's velocity. In case of grazing scattering trajectories the electron translation factor is usually accounted [3,24]. It should be mentioned that last time the alternative approach has been developed: (i) the electron transition rate is calculated as function of ion–surface distance by means of 3D DFT computations; (ii) electron transition probability is calculated by integrating the transition rate along the ions trajectory [24–26].

Note, that rate equation approximation is not valid for low energy ions, especially when the surface contains nanosystems (e.g. thin films) [27]. Moreover, some experiments, e.g. azimuthal dependence of RCT probability, requires direct 3D RCT modeling and 2D methods fail to describe them [3,8,17]. These facts stipulate the actuality of TDSE Solver for the 3D RCT modeling.

Last years the TDSE Solver was developed for 3D RCT modeling in large-scale nanosystems [28]. It was shown to have rather good performance due to the effective parallel implementation of simple numerical scheme on GPUs (explicit finite-difference method in Cartesian coordinates). The serious weakness of GPU TDSE Solver is the usage of regular numerical grid in Cartesian coordinates. This demands ∼100 times greater numerical grid comparing to the FVM/FEM. Such dramatic difference occurs because regular Cartesian numerical grid approximates the central region of calculation domain and the periphery (halo) with the same precision. While in typical FVM/FEM calculations the numerical grid is adapted to the physical problem under study, and the volume of element in central region is an order of magnitude smaller, in comparison to the periphery. For the majority of RCT problems the calculations transfer to the cylindrical coordinates decrease the size of numerical grid (and hence increase performance) by an order of magnitude without loss of the calculations precision. Note, that cylindrical as well as Cartesian are full 3D coordinates. The main problem in the transfer to the cylindrical coordinates is that explicit numerical schemes for parabolic equations, including TDSE, are unstable near the axe $\rho = 0$.

This paper presents hybrid numerical scheme, which eliminates TDSE's instability in cylindrical coordinates and preserves the effective parallelization on GPUs (Section 2). In Section 3 the parallel performance and scalability of new version of GPU TDSE Solver are discussed; its performance is compared to other known TDSE solvers. Section 4 shows the application of the GPU TDSE Solver to the calculation of the RCT in nanosystems; the numerical results are compared to the experimental data.

2. Numerical method

2.1. Hybrid numerical scheme for TDSE in cylindrical coordinates

The explicit propagation scheme for solving TDSE, its precision and stability are described in detail in Ref. [28]. The novelty of present study is the hybrid numerical scheme, which is used to solve TDSE in cylindrical coordinates explicitly. The derivation of the hybrid numerical scheme is given in Eqs. (3)–(10) and calculation algorithm is described at the end of this section.

At a glance, we consider TDSE in form:

$$i\frac{d\psi(\vec{r},t)}{dt} = \left(-\frac{\Delta}{2} + U(\vec{r},t)\right)\psi(\vec{r},t), \tag{1}$$

where $U(\vec{r},t) = V_{ion}(\vec{r},t) + V_{e-surf}(\vec{r})$ is time-dependent potential. Then we use following time-evolution scheme with absorbing image potential $V(\vec{r},t)$:

$$\begin{aligned}\phi^{n+1} &= -2i\tau\left[\left(-\frac{\Delta}{2} + U(\vec{r},t)\right)\phi^n\right] + \phi^{n-1} \\ &= -2i\tau e^{-i\tau V}\left[\left(-\frac{\Delta}{2} + U(\vec{r},t)\right)\phi^n\right] + e^{-2i\tau V}\phi^{n-1}.\end{aligned} \tag{2}$$

An important feature of GPU TDSE Solver implementation in cylindrical coordinates is the choice of boundary conditions for ρ and θ coordinates. To describe the boundary conditions we will use the following discretization notation $\phi_{k,l,m} = \phi(x = x_k;\ \rho = \rho_l;\ \theta = \theta_m)$.

Of course, the periodical boundary conditions should be chosen for θ coordinate: $\phi_{k,l,0} = \phi_{k,l,M}$; $\phi_{k,l,M+1} = \phi_{k,l,1}$, where $M = N_\theta$ is the number of discrete points along θ coordinate.

A tricky problem is definition of boundary conditions for $\rho \to 0$. To solve it we use numerical grid starting from value $\rho_1 = 0.5 \cdot d\rho$. This allows us to write the boundary conditions for $\rho \to 0$ as $\phi_{k,0,m} = \phi_{k,1,m^*}$, where the point $k,\ 1,\ m^*$ is a point of space symmetrical to the point $k,\ 1,\ m$ relatively the axe $\rho = 0$. Note that due to such scheme the number of discrete points along θ coordinate should be even.

Due to the features of the GPU TDSE Solver architecture (MPI decomposition along x-coordinate) the usage of cylindrical coordinates $x,\ \rho,\ \theta$ is convenient, that can be transformed to Cartesian coordinates as $y = \rho \cdot \cos(\theta);\ z = \rho \cdot \sin(\theta)$.

Considering TDSE (1) Eq. (2) without absorbing potential can be rewritten in the form:

$$\phi^{n+1} = \phi^{n-1} + 2i\tau\left[\frac{\Delta}{2}\phi^n - U\phi^n\right], \tag{3}$$

where Laplacian in cylindrical coordinates $x,\ \rho,\ \theta$ is equal to:

$$\Delta = \frac{1}{\rho}\frac{\partial}{\partial\rho}\left(\rho\frac{\partial}{\partial\rho}\right) + \frac{\partial^2}{\partial x^2} + \frac{1}{\rho^2}\frac{\partial^2}{\partial\theta^2}. \tag{4}$$

The term $\frac{1}{\rho^2}\frac{\partial^2}{\partial\theta^2}$ leads to the numerical instability of explicit leap-frog scheme near the axe $\rho = 0$. Let us rewrite Laplacian as the sum of 2 terms:

$$\Delta = \Delta_0 + \Delta_1;\ \Delta_0 = \frac{1}{\rho}\frac{\partial}{\partial\rho}\left(\rho\frac{\partial}{\partial\rho}\right) + \frac{\partial^2}{\partial x^2};\ \Delta_1 = \frac{1}{\rho^2}\frac{\partial^2}{\partial\theta^2}. \tag{5}$$

The wave function ϕ^{n+1} at the "next step" (6) can be treated as sum of main term ϕ_0^{n+1} and correction ϕ_1^{n+1}:

$$\phi^{n+1} = \phi^{n-1} + 2i\tau \left[\frac{\Delta_0}{2}\phi^n + \frac{\Delta_1}{2}\phi^n - U\phi^n \right] = \phi_0^{n+1} + \phi_1^{n+1}, \tag{6}$$

where

$$\phi_0^{n+1} = \phi^{n-1} + 2i\tau \left[\frac{\Delta_0}{2}\phi^n - U\phi^n \right];$$

$$\phi_1^{n+1} = 2i\tau \frac{\Delta_1}{2}\phi^n = \frac{i\tau}{\rho^2}\frac{\partial^2}{\partial\theta^2}\phi^n. \tag{7}$$

The main term ϕ_0^{n+1} at each step can be calculated easily using explicit leap-frog scheme; its does not exhibit numerical instability. The correction ϕ_1^{n+1} is non-zero only near the axe $\rho = 0$, but the explicit scheme cannot be applied to its calculation. Therefore, using Eqs. (6) and (7) the following semi-implicit scheme for the correction calculation can be derived:

$$\phi_1^{n+1} = \phi^{n+1} - \phi_0^{n+1} = \frac{i\tau}{\rho^2}\frac{\partial^2}{\partial\theta^2}\phi^n$$

$$= \frac{i\tau}{\rho^2}\frac{1}{\alpha+\beta}\frac{\partial^2}{\partial\theta^2}\left[\alpha\phi^{n+1} + \beta\phi^n\right]. \tag{8}$$

When $\alpha = 1$ and $\beta = 0$ the above scheme is completely implicit; when $\alpha = 0$ and $\beta = 1$ it is completely explicit (and hence unstable).

Of note, Eq. (9) does not contain the correction ϕ_1^{n+1} and the main term ϕ_0^{n+1} is already known from Eq. (7). Therefore, for each fixed value of z and ρ we get 1D differential equation on ϕ^{n+1}.

Now let us redefine some variables: $h = \phi^{n+1}$; $f = \phi_0^{n+1}$; $g = \phi^n$. Of note, f and g are known functions. Then Eq. (11) can be rewritten as:

$$\frac{i(\rho\, d\theta)^2}{\tau}(\alpha+\beta)(f_k - h_k) = \alpha(h_{k+1} + h_{k-1} - 2h_k) + \beta(g_{k+1} + g_{k-1} - 2g_k), \tag{9}$$

where $d\theta$ is a discretization step along θ-coordinate, and the subscript denotes discretization along θ-coordinate with periodical boundary conditions (for $k = 1$ $h_{k-1} = h_M$ and for $k = M$ $h_{k+1} = h_1$).

Also the absorbing image potential $V(\vec{r})$ should be taken into account:

$$\frac{i(\rho\, d\theta)^2}{\tau}e^{-i\tau V}(\alpha+\beta)(f_k - h_k) = \alpha(h_{k+1} + h_{k-1} - 2h_k) + \beta(g_{k+1} + g_{k-1} - 2g_k), \tag{10}$$

where the transformation from Eqs. (9) to (10) could be easily done by insertion of absorbing term from Eq. (2) into Eq. (3) and appropriate modification of Eqs. (6)–(8).

Eq. (10) is tridiagonal system with periodic boundary conditions. It could be solved by means of Thomas algorithm in combination with Sherman–Morrison formula [29].

Let us describe the algorithm in detail. Eq. (10) can be expressed in the matrix form as $[A] \times [h] = [d]$:

$$\begin{bmatrix} b_1 & c_1 & & & a_1 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & . & \\ & & . & . & c_{n-1} \\ c_n & & & a_n & b_n \end{bmatrix} \times \begin{bmatrix} h_1 \\ h_2 \\ . \\ . \\ h_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ . \\ . \\ d_n \end{bmatrix}, \tag{11}$$

where $a_k = c_k = \alpha$, $b_k = \frac{i(\rho\, d\theta)^2}{\tau}e^{-i\tau V}(\alpha+\beta) - 2\alpha$,

$$d_k = \frac{i(\rho\, d\theta)^2}{\tau}e^{-i\tau V}(\alpha+\beta)f_k - \beta(g_{k+1} + g_{k-1} - 2g_k).$$

Eq. (11) can be rewritten as:

$$\left[B + u \times v^T\right] \times [h] = [d], \tag{12}$$

where $u^T = [-b_1\, 0\, 0 \ldots c_n]$; $v^T = [1\, 0\, 0 \ldots 0 - a_1/b_1]$, and $[B] = \left[A - u \times v^T\right]$.

Then unknown vector $[h]$ is computed as:

$$h = y - q\frac{(v^T \times y)}{1 + (v^T \times y)}, \tag{13}$$

where y and q are solutions of tridiagonal systems $[B] \times [y] = [d]$ and $[B] \times [q] = [u]$ respectively.

The solution of the above tridiagonal system

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & . & \\ & & . & . & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ . \\ . \\ d_n \end{bmatrix}, \tag{14}$$

can be obtained by means of Thomas algorithm, consisting of forward elimination and backward substitution phases.

During forward elimination phase, the original coefficients are replaced as:

$$\left.\begin{aligned} b'_k &= b_k - c_{k-1}\frac{a_k}{b_{k-1}} \\ d'_k &= d_k - d_{k-1}\frac{a_k}{b_{k-1}} \end{aligned}\right\} \quad k = 2, \ldots, M. \tag{15}$$

During backward substitution phase we iteratively calculate the unknown vector values:

$$h_k = \frac{d_k - c_k h_{k+1}}{b_k}, \quad k = M-1, \ldots, 1, \tag{16}$$

where $h_M = d_n/b_n$.

It should be noted, that for the problem under the study (TDSE for RCT between ion and surface) the matrix $[B]$ is diagonally dominant, i.e. $|b_k| > |a_k| + |c_k|$. Hence, Thomas algorithm is suitable.

Finally, the hybrid numerical scheme for TDSE solution in cylindrical coordinates can be formulated as following algorithm for ϕ^{n+1} calculation:

1. For $\rho \geq \rho 0$ we use the ordinary explicit leap-frog numerical scheme, described in Eq. (6).
2. If $\rho < \rho 0$, then ϕ^{n+1} is calculated in two steps:
   - 2.1 Initially we calculate the "main term" ϕ_0^{n+1} by means of explicit leap-frog numerical scheme from Eq. (7).
   - 2.2 Then we find ϕ^{n+1} from Eq. (10) using algorithm described in Eqs. (11)–(16).

It should be stressed that numerical complexity of the "implicit" part of the hybrid numerical scheme is small comparing to its "explicit" part and the "implicit" part can be easily parallelized. Therefore, the performance degradation of hybrid numerical scheme in comparison to explicit leap-frog scheme should not be large.

Note, that despite the proposed hybrid numerical scheme was developed for the GPU TDSE Solver, it could be also applied to other parabolic PDEs in cylindrical coordinates regardless of the used calculation equipment (CPU or GPU).

2.2. Parallel GPU implementation

The GPU version of the program is written in C programming language with CUDA extension. Message Passing Interface (MPI) [30] technology is used [31] for multi-GPU parallel implementation.

The MPI implementation is the same as in "Cartesian" version of GPU TDSE Solver [28]. Briefly, each parallel process calculates its own part of a calculation domain (block). After each time-step the synchronization with other processes occurs.

Two main points of GPU realization are: (i) memory access optimization (effective usage of shared memory and coalesced global memory access); (ii) asynchronous data transfer and computation. The GPU realization for 1 MPI process is described below in detail.

Concerning the memory access optimization we follow close to the FMD realization described in [31]. The core idea is to use GPUs shared memory in order to reduce the memory accesses. According to FDM time evolution scheme (Eq. (2)) to compute the wave function value at the next time moment $\phi^{n+1}_{k,l,m}$ we have to know current wave function value $\phi^{n}_{k,l,m}$ and wave function values at the neighbor discrete points $\phi^{n}_{k\pm1,l\pm1,m\pm1}$. Therefore, for the effective usage of shared memory the block of threads should handle the compact subdomain of the calculation domain. The subdomain volume (number of discrete FDM points in the domain) corresponds to the single-read wave functions values, the subdomain surface corresponds to the additional memory read from GPU global memory. Therefore, the volume/surface ratio should be maximized for the memory access optimization. But, due to the memory limitations (16–64 kB of shared memory per multiprocessor), we cannot store large 3D subdomains in the shared memory. Instead of this, we store in shared memory only 2D subdomain (slice in ρ–θ space) and iterate over x-coordinate to handle 3D subdomain.

The calculation domain for 1 MPI process is divided into 3D spatial-blocks over ρ–θ coordinates (Fig. 1). Each spatial-block is handled by single block of CUDA threads (CUDA-block). This means that we use 2D CUDA grid. Inside the selected block, each CUDA thread handles its own line, iterating along x-coordinate. During the each iteration, the block of CUDA threads reads 2D ρ–θ slice of wave function values into the shared memory (Fig. 2). So the shared memory is used to access the neighbor values along ρ–θ coordinates (excluding the spatial-block surface). The peripheral values of each spatial-block are accessed twice (by threads of own spatial-block and by threads of neighbor spatial-blocks). To optimize the neighbor values access along x coordinate, during each iteration of "for" cycle we read "next" value of wave function and translate it as "current" and "previous" values for the subsequent iterations. Using the described method, the wave function values inside 3D spatial-block are accessed only once. The once read wave function values are stored in shared memory and are reused for the calculations of other points. This minimizes the number of global memory accesses and significantly increases the performance.

The optimal size of spatial-block has been defined from the performance measurements as $(N_x - 2) \cdot (1 + 2) \cdot (128 + 2)$, that correspond to the 2D shared memory array size $(1 + 2) \cdot (128+2)$; 3 for ρ and 130 for θ. Of note, in most of our calculations we use $N_\theta = 128$, but extra two points in 2D shared memory array are necessary to store boundary conditions. Theoretically, the allocation of greater shared memory should increase the performance. However, in practice we should take into account the hardware restrictions—amount of shared memory available per block of processes. The above-given optimal 2D shared memory array size was experimentally defined for Tesla k20 and Tesla M2090 GPUS, it could be larger for some more modern GPUs. To

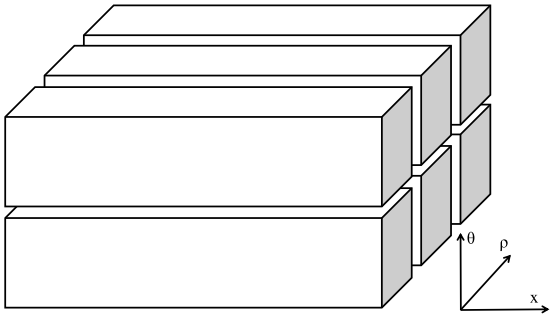

Fig. 1. Illustration of decomposition of calculation domain into 3D blocks.

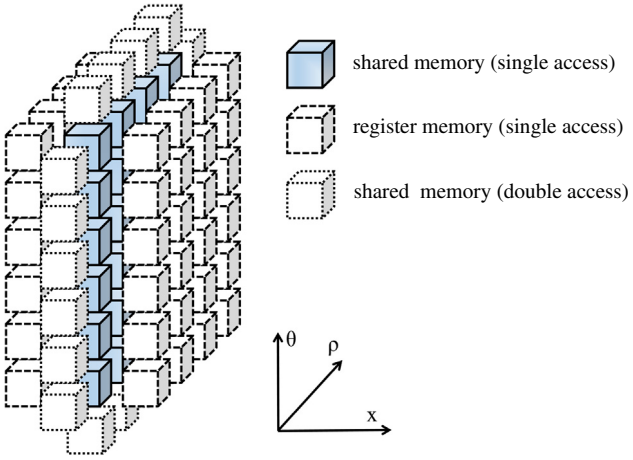

Fig. 2. Illustration of shared-memory usage.

calculate the next values of the wave function we also should use halo, so the number of memory accesses is not less than $N_x \cdot (1 + 2) \cdot (128 + 2)$. On each step of "for" a cycle block of threads reads $3 \cdot 130$ current wave function values in order to calculate $1 \cdot 128$ next wave-function values.

The 3D matrices for wave function and potentials are stored in a 1D array (in GPU global memory). According to the "CUDA C Best Practices Guide" the coalescing significantly increases the speed of global memory access [32]. A coalesced memory access is realized when consecutive threads access consecutive memory addresses. To provide the coalesced memory access we have to use the following index mapping rule

$$index_{1D} = index_x \cdot N_\rho \cdot N_\theta + index_\rho \cdot N_\theta + index_\theta. \quad (17)$$

The given mapping rule and above described CUDA grid fulfill the coalesced memory access conditions, i.e. the neighbor CUDA threads read the neighbor global memory addresses. It should be stressed that the combination of the 3D->1D mapping and CUDA grid described above corresponds to the optimal memory access, which is critical for the performance, e.g. if one changes the 3D->1D mapping to $index_{1D} = index_\theta \cdot N_\rho \cdot N_x + index_\rho \cdot N_x + index_x$ the performance will reduce by an order of magnitude.

Another major feature of GPU TDSE Solver implementation is parallel asynchronous transfer of the halo data and calculations in the main region. Each calculation step consists of two stages. Initially wave function ϕ^{n+1} is calculated in the halo, this requires the calculation of $2 \cdot N_\rho \cdot N_\theta$ points for each MPI-process. Then in parallel starts the asynchronous MPI data transfer ($2 \cdot N_\rho \cdot N_\theta$ points) and ϕ^{n+1} calculation in the main region ($N_x \cdot N_\rho \cdot N_\theta$ points for each MPI-process).

Table 1
Performance comparison of different versions of GPU TDSE Solver.

TDSE Solver	Calculations facilities, used number of processing units and its price	Problem size, 10^9 points	Time for 1 step (s)	Performance (Gpnts/(s))	Normalized calculation time (s)	Normalized calculation time without price (s)
TDSE GPU Solver ("cylindrical" version)	GSRV MSU, 1 T k20m * 2700 USD [33]	0.0108	0.0073	1.48	0.26	0.70
TDSE GPU Solver ("cylindrical" version)	GSRV MSU, 1 GeForce Titan * 1000 USD [34]	0.0108	0.0120	0.90	0.16	0.16
TDSE GPU Solver ("cylindrical" version)	Lomonosov MSU, 1 T M2090 * 1600 USD [35]	0.0108	0.0136	0.80	0.29	0.46
TDSE GPU Solver ("Cartesian" version [28])	GSRV MSU, 1 T k20m * 2700 USD [33]	0.016	0.0087	1.84	1.47	3.96

2.3. GPU TDSE Solver verification, optimal parameters and precision

For the convenience the atomic system of units (with $m_e = e = \hbar = 1$; 1 a.u. of distance is equal to 0.53 Å, 1 a.u. of time is equal to $2.419 \cdot 10^{-17}$ s, and 1 a.u. of velocity is equal to $2.188 \cdot 10^8$ cm/s) is used in this paper.

Like the previous version, the new version of GPU TDSE Solver was verified in several stages, including: (i) finding the optimal parameters; (ii) TDSE solution comparison with known solutions; (iii) calculation of energy eigenstates of the hydrogen atom and affinity level of H^- ion (for details see Ref. [28]). These entire tests have been successfully passed by new version of GPU TDSE Solver, the GPU TDSE Solver calculation results in Cartesian and cylindrical coordinates were identical.

For the problem of RCT between atomic particle and nanosystems it was found that for parameters $dx = d\rho = 0.2$ a.u., $d\theta = 2\pi/128$; $d\tau = 0.005$ a.u.; $\rho_0 = 1$ a.u.; $\alpha = 1$; $\beta = 0$, the numerical scheme converges to the TDSE solution. The decrease of the parameters does not change the solution, but doubling the dx, $d\rho$, $d\theta$ parameters leads to a wrong TDSE solution and the doubling of τ leads to numerical instability. Notably, that the above parameters guarantee that the space discretization in region $\rho < 20$ a.u. is more precise comparing to discretization of Cartesian coordinates with the same step $dx = dy = dz = 0.2$ a.u.

As used propagation scheme is only the SOD scheme, the important question is its accuracy. The numerical error was estimated as 10^{-8} per time-step. Thus for the time propagation of 10^6 steps the numerical error does not succeed 1%. Such precision is acceptable for the RCT problems.

3. GPU TDSE Solver performance

The useful measure for FDM performance is a number of spatial points processed per second within one step of time evolution [31]. So, we will use "Gpnts/s" to measure GPU TDSE Solver absolute performance and scalability. However, the above-defined "Gpnts/s" is not useful for relative performance measure, e.g. GPU TDSE Solver performance comparison with other solvers, because many of them are based on FVM/FEM and use CPU calculations. In Ref. [28] the measure of relative performance – "normalized calculation time" (NCT) – was introduced $NCT = t \cdot \frac{Price}{GridFactor \cdot ProblemSize}$. This measure accounts the number of points/elements necessary for the space discretization (so-called "grid factor") and the price of computational resources (useful for the CPU vs. GPU comparison). The typical values of "Grid factor" are not more than 200 for finite volumes/elements method and about 10 for FDM in cylindrical coordinates (see [28] for details). Note, that when two programs are compared on the same equipment, their NCTs ratio does not depend on the used equipment price. For this reason, we also use metric "normalized calculation time without price" in Table 1.

The absolute performance of "cylindrical" version GPU TDSE Solver measured on the single Tesla k20m GPU is about 1.48 Gpnts/s, that is about 25% less than for "Cartesian" version (1.83 Gpnts/s). This occurs because the FDM calculations in the cylindrical coordinates themselves and instability elimination in the hybrid numerical scheme are more expensive. For the Tesla M2090 GPU the GPU TDSE Solver's performance is 0.80 Gpnts/s for "cylindrical" version and 1.06 Gpnts/s for "Cartesian". The surprising fact is that calculations performance on graphical card GeForce Titan (not Titan X or Titan Z) is 0.90 Gpnts/s, that is comparable with the performance of more expensive GPU Tesla M2090.

The major characteristic of the parallel code is its scalability. The scaling of GPU TDSE Solver in cylindrical coordinates is very similar to its Cartesian version [28]. GPU TDSE Solver shows linear weak scaling up to the 32 GPUs. The strong scaling is limited to ~4 GPU for the typical problem.

An important consequence of the FDM calculation's transfer to the cylindrical coordinates is that the required computational grid (the number of discrete points) is ~10 times less comparing to the Cartesian coordinates. This leads to the ~10 times reduction of the minimal-required computational resources, especially the amount of GPU memory. As a result, the numerical simulation of the one-electron processes in large-scale nanosystems (up to 10^3 nm^3) can be performed on ordinary personal computer, equipped with modern GPU (Tesla k20 or higher). For the supercomputer calculations, the significant advantage is much better scalability: the GPU TDSE Solver shows linear scalability up to the 32 GPUs, that corresponds to the calculation domain about 3×10^4 nm^3 (compare to 3×10^3 nm^3 for the "Cartesian" version).

The essential point for the new TDSE Solver is the comparison of its performance with state-of-the-art technology. In Ref. [28] the performance of "Cartesian" version of GPU TDSE Solver was compared to other TDSE Solver's performance using NCT. The main conclusion is that "Cartesian" version of GPU TDSE Solver is 3 times faster. Also good GPU/CPU acceleration was shown comparing to other TDSE solvers. In this article, we will emphasize on the performance comparison between "Cartesian" and "cylindrical" versions of GPU TDSE Solver. The results of the comparison are shown in Table 1.

The NCT depends on the absolute performance ("Gpnts/s") as well as on the "Grid factor". According to Ref. [28] the lower estimation of "Grid factor" in cylindrical coordinates is 7. We can see that the performance of the GPU TDSE Solver in cylindrical coordinates ($NCT = 0.26$ s for Tesla k20m) is ~6 times better than performance of its "Cartesian" version ($NCT = 1.5$ s for Tesla k20m). Additionally it is ~20 times better than the performance of the best CPU TDSE Solver ($NCT = 4.5$ s, see Ref. [28]). This means that despite non-optimal FDM computational grid ($GridFactor = 200$ for FVM/FEM vs. $GridFactor = 7$ for cylindrical FDM), the efficient GPU parallelization of the FDM-based numerical scheme provides great benefit in performance.

The NCT of GPU TDSE Solver was also measured for different graphical devices. The NCT for GPUs Tesla k20m and Tesla M2090

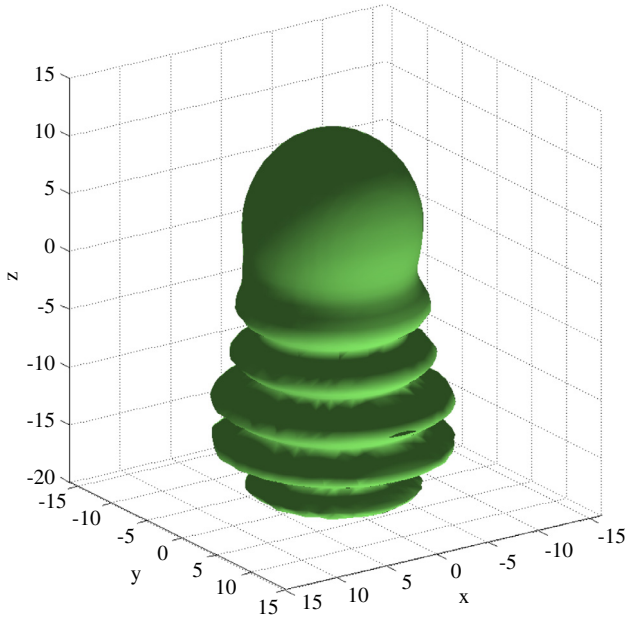

Fig. 3. Isosurface of electron density for the moment of time 50 a.u.

are found to be very similar, whereas calculations on graphics card GeForce Titan are ∼1.75 times more profitable comparing to GPU calculations. Note, that these measurements were done for the "oldest" GeForce Titan card (GTX TITAN 6 GB GDDR5 384 bit), double precision calculations on other cards (Titan X or Titan Z) could be restricted. In addition, it should be noted, that code scalability and parallel performance were not measured for multiple GeForce Titan cards.

4. GPU TDSE Solver application for the RCT calculations

The GPU TDSE Solver was applied to the computer modeling of real experimental data. In Ref. [16] Li^+ neutralization probability during interaction with Ag(100) surface was measured as function of ion's energy. In experiment the ion beam is oriented at 45^0 to the surface normal and the scattered Li atoms / Li^+ ions are detected along the normal to the surface. The ion beam energy varies from 0.2 to 2 keV.

To simulate the above experiment we have used several assumptions. First, we considered RCT only on outgoing part of ionic trajectory. Such approximation is valid because the influence of the initial charge state of the particle was studied in several experiments and for typical atom/ion–surface combinations no memory effect of the initial charge state was found [12,13,36]. Also we used the time reversal symmetry to investigate the problem of electron loss by neutral atom Li on the ongoing trajectory, instead of problem of electron capture by positive Li^+ ion on the outgoing trajectory [16].

The one-active-electron pseudo-potential for Li ion/atom is described in Ref. [9]. Because Ag(100) is a surface with projected band gap, the pseudo-potential from Ref. [11] was chosen. Fig. 3 illustrates the RCT between Li and Ag(100) surface. We see that electron density distribution inside the metal exhibits maxima along z-coordinate. This happens due to the presence of the projected band gap. Contrary to the case of "jellium" metal, when electron freely propagates deep into the bulk, in the case of projected band gap electron motion is confined along the normal to the surface and electron is forced to propagate parallel to the surface. The calculated neutralization probability for Li^+ ions impinging on the Ag(100) surface shows a good quantitative agreement with the experimental data (Fig. 4).

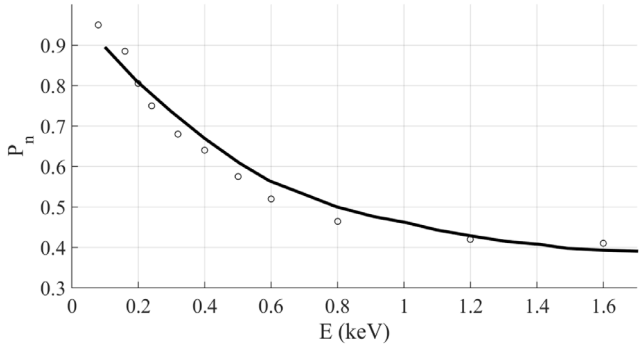

Fig. 4. Li^+ neutralization probability as function of ion energy. Solid line represent the numerical modeling results and open dots—experimental data [36].

Note, that due to the cylindrical symmetry, the above experimental data do not contain "3D effects" and can be modeled by means of 2D calculation method [16]. Of course, numerous RCT experiments exhibit 3D effects, e.g. azimuthal (orientation) dependence of RCT probability during H^- ion grazing scattering on Cu(110) surface [3,8]. For computer modeling of these experiments we have to use 3D calculations as well as 3D pseudo-potentials. However, for the current moment only 1D pseudopotentials (that depend on z-coordinate) for the surface description are well developed [10,11]. Thus, the further development of the GPU TDSE Solver will be the construction of the 3D pseudopotentials in order to describe such 3D effects like RCT orientation dependence.

5. Conclusion

The new version of GPU TDSE Solver for 3D modeling of resonant electron transfer (RCT) during ion–surface interactions and atomic collisions is presented in this study. Note, that the key requirement to the TDSE Solver's for 3D RCT modeling is capability to handle large spatial domains (about 10^3–10^4 nm^3). Contrary to the previous version, the new version of GPU TDSE Solver performs the FMD explicit scheme in cylindrical coordinates, instead of Cartesian. The calculations transfer to the cylindrical coordinates significantly decreases the size of numerical grid. The main problem in the transfer to the cylindrical coordinates is that explicit numerical schemes for parabolic equations, including TDSE, are unstable near the axe $\rho = 0$. In the present study, the hybrid numerical scheme for parabolic PDEs (including TDSE) in cylindrical coordinates was proposed, which eliminates instability near axe $\rho = 0$ and preserves effective parallelization of the explicit leap-frog numerical scheme.

The performance of new version of the GPU TDSE Solver, based on the hybrid numerical scheme, was found to be 6 times greater comparing to the previous version and 20 times greater comparing to other known TDSE solvers. Such performance gain is obtained because in cylindrical coordinates the required computational grid (the number of discrete points) is ∼10 times less comparing to the Cartesian coordinates. The less computational grid also leads to the 10 times reduction of the minimal required computational resources, especially the amount of GPU memory. As a result, the numerical simulation of the one-electron processes in large-scale nanosystems (up to 10^3 nm^3) can be performed on ordinary personal computer, equipped with modern GPU (Tesla k20 or higher). For the supercomputer calculations, the significant advantage is much better scalability: the GPU TDSE Solver shows linear scalability up to the 32 GPUs, that corresponds to the calculation domain about 3×10^4 nm^3 (compare to the 3×10^3 nm^3 for the previous version).

The GPU TDSE Solver was applied to the calculation of the resonant charge transfer (RCT) in nanosystems. The calculated neutralization probability for Li^+ ions impinging on the Ag(100) surface

shows a good quantitative agreement with the experimental data. The further development of the GPU TDSE Solver consists in the construction of the 3D pseudopotentials in order to describe such 3D effects as RCT orientation dependence. After that, we are going to implement the new module for the calculation of the short laser pulses interaction with atomic/molecular electronic system.

Acknowledgment

This work was partially financed by the Russian Foundation of Basic Research (16-02-00478).

References

[1] R. Brako, D.M. Newns, Rep. Progr. Phys. 52 (1989) 655.
[2] H. Shao, D.C. Langreth, P. Nordlander, in: J.W. Rabalais (Ed.), Low Energy Ion-Surface Interactions, Wiley, New York, 1994.
[3] T. Hecht, H. Winter, A.G. Borisov, J.P. Gauyacq, A.K. Kazansky, Phys. Rev. Lett. 84 (2000) 2517.
[4] J.P. Gauyacq, A.G. Borisov, D. Teillet-Billy, in: V.A. Esaulov (Ed.), Formation/Destruction of Negative Ions in Heavy Particle-Surface Collisions, Cambridge University Press, Cambridge, England, 1996.
[5] J.J.C. Geerlings, J. Los, Phys. Rep. 190 (1990) 133.
[6] H. Chakraborty, T. Niederhausen, U. Thumm, Phys. Rev. A 70 (2004) 052903.
[7] H. Chakraborty, T. Niederhausen, U. Thumm, Phys. Rev. A 69 (2004) 052901.
[8] T. Hetch, H. Winter, A.G. Borisov, J.P. Gauyacq, A.K. Kazansky, Faraday Discuss. 117 (2000) 27.
[9] J.N. Bardsley, Case Stud. At. Phys. 4 (1974) 299.
[10] P.J. Jennings, R.O. Jones, M. Weinert, Phys. Rev. B 37 (1988) 6113.
[11] E.V. Chulkov, V.M. Silkin, P.M. Echenique, Surf. Sci. 437 (1999) 330.
[12] M. Maazouz, A.G. Borisov, V.A. Esaulov, J.P. Gauyacq, L. Guillemot, S. Lacombe, D. Teillet-Billy, Phys. Rev. B 55 (1997) 13 869.
[13] L. Guillemot, V.A. Esaulov, Phys. Rev. Lett. 82 (1999) 4552.
[14] J. Sjakste, A.G. Borisov, J.P. Gauyacq, Nucl. Instrum. Methods B 203 (2003) 49.
[15] K. Niedfeldt, E.A. Carter, P. Nordlander, J. Chem. Phys. 121 (2004) 375.
[16] A.R. Canario, A.G. Borisov, J.P. Gauyacq, V.A. Esaulov, Phys. Rev. B 71 (2005) 121401(R).
[17] B. Bahrim, B. Makarenko, J.W. Rabalais, Surf. Sci. 594 (2005) 62.
[18] B. Bahrim, S. Yu, B. Makarenko, J.W. Rabalais, Surf. Sci. 603 (2009) 703.
[19] R.A. Vidal, F. Bonetto, J. Ferron, M.A. Romero, E.A. Garsia, E.C. Goldberg, Surf. Sci. 605 (2011) 18.
[20] H. Zhou, L. Chen, D. Feng, Y. Guo, M. Ji, G. Wang, W. Zhou, Y. Li, L. Zhao, X. Chen, Phys. Rev. A 85 (2012) 014901.
[21] L. Chen, B. Ding, Y. Li, S. Qiu, F. Xiong, H. Zhou, Y. Guo, X. Chen, Phys. Rev. A 88 (2013) 044901.
[22] U. Thumm, https://jrm.phys.ksu.edu/Research/Meetings/DOE-AMOP/ut.pdf [26.03.2015].
[23] I.K. Gainullin, I.F. Urazgildin, Izv. RAN, Ser. Fiz. 70 (2006) 897 (in Russian). Bulletin of the Russian Academy of Sciences, Physics 70 (2006) 1024.
[24] B. Obreshkov, U. Thumm, Phys. Rev. A 74 (2006) 012901.
[25] B. Obreshkov, U. Thumm, Phys. Rev. A 83 (2011) 062902.
[26] B. Obreshkov, U. Thumm, Phys. Rev. A 87 (2013) 022903.
[27] I.K. Gainullin, M.A. Sonkin, Phys. Rev. A 92 (2015) 022710.
[28] I.K. Gainullin, M.A. Sonkin, Comput. Phys. Comm. 188 (2015) 68–75.
[29] L.H. Thomas, Elliptic Problems in Linear Differential Equations over a Network, Watson Sci. Comput. Lab Report Columbia University, New York, 1949;
S.D. Conte, C. deBoor, Elementary Numerical Analysis, McGraw-Hill, New York, 1972;
http://www.cfd-online.com/Wiki/Tridiagonal_matrix_algorithm_-_TDMA_(Thomas_algorithm).
[30] I. Foster, Designing and Building Parallel Programs, Chapter 8 Message Passing Interface, Addison-Wesley, 1995.
[31] P. Micikevicius, Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units, ACM, New York, NY, USA, 2009.
[32] http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/;
https://devblogs.nvidia.com/parallelforall/how-access-global-memory-efficiently-cuda-c-kernels/.
[33] http://www.amazon.com/NVIDIA-Tesla-K20-Accelerator-900-22081-2220-000/dp/B00AA2C1DC/ref=pd_sim_sbs_pc_1?ie=UTF8&refRID=1HGHVMN66D1685ZB6YWE (10.09.2014).
[34] http://www.pcmag.com/article2/0,2817,2415630,00.asp (23.06.2016).
[35] http://www.amazon.com/nVidia-M2090-Processing-Computing-Module/dp/B007ED62NU/ref=pd_sim_sbs_pc_4?ie=UTF8&refRID=1HGHVMN66D1685ZB6YWE (10.09.2014).
[36] A.G. Borisov, D. Teillet-Billy, J.P. Gauyacq, Phys. Rev. Lett. 68 (1992) 2842.