Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 171 (2020) 312–321

Procedia Computer Science

www.elsevier.com/locate/procedia

Third International Conference on Computing and Network Communications (CoCoNet'19)

Solving the 3D Schrödinger Equation on a GPU

Qingjun Liu[a,b,*], Fang Liu[c], Chaofeng Hou[d]

[a]*Department of Mathematics and Physics, Beijing Institute of Petro-chemical Technology, Beijing 102617, China*
[b]*Institute of Nano-photoelectronics and High Energy Physics, Beijing Institute of Petro-chemical Technology, Beijing 102617, China*
[c]*Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China*
[d]*Institute of Process Engineering, Chinese Academy of Sciences, Beijing 100190, China*

Abstract

Solving the 3D Schrödinger equation is of great interests in many branches of physical sciences and is usually a computing intensive task. Modern graphics processing units (GPUs) are extensively employed in the general purpose computing domain due to their high performance parallel processing capability. In this paper, an algorithm for solving the time-independent 3D Schrödinger equation with a finite difference time domain (FDTD) method is presented, together with an implementation of the algorithm by using CUDA (Compute Unified Device architecture) C. The GPU-based solver is validated in four cases: 3D Coulomb potential, 3D harmonic oscillator, three coupled anharmonic oscillators, and H_2^+. Additionally, a CPU-based solver that is a serial program is developed according to the same FDTD method for the purpose of testing the effectiveness of the GPU-based solver. Relative to the CPU-based solver that employs one core of a multi-core CPU, the GPU-based solver can achieve a speed-up of more than 90X, 60X, 40X, and 90X in the case of 3D Coulomb potential, 3D harmonic oscillator, three coupled anharmonic oscillators, and H_2^+, respectively. The GPU-based solver can accelerate the solution of the 3D Schrödinger equation.

Keywords: 3D Schrödinger equation; Finite difference time domain method; Graphic processing unit; Compute unified device architecture

1. Introduction

It is well-known that solving Schrödinger equation is one of the essential ways for exploring the properties of a quantum system. For example, the charge transfer and charge exchange in nanosystems [1, 2], high-order harmonic generation via short pulse laser interaction with atoms and molecules [3, 4, 5, 6], the Rydberg state excitation and dissociation of atoms or molecules [7, 8, 9], the quarkonium production in ultra-relativistic heavy-ion collisions [10, 11], may all be theoretically modeled based on solution of the Schrödinger equation. Ever since the founding of quantum mechanics, it has been realized that for most of the cases, the Schrödinger equation can only be solved

* Corresponding author. Tel.: +86-10-81292320; fax: +86-10-81292176.
E-mail address: liuqingjun@bipt.edu.cn

10.1016/j.procs.2020.04.032

numerically. Being computing intensive, some numerical solutions [12, 13, 14, 15, 16] usually necessitate using parallel processing software technologies involving openMP [17] or the Message Passing Interface (MPI) [18] or Compute Unified Device Architecture (CUDA) [19], even a combination of them, together with clustered modern hardware like many-core GPUs (graphic processing units) and multi-core CPUs.

The finite difference time domain (FDTD) method [20, 21, 22, 23, 24, 25], spectral method [26, 27, 28], integration method [29] and other methods [30, 31] have been advocated for solving the Schrödinger equation. Based on the FDTD method described in [23], a parallel algorithm for solving the time-independent 3D Schrödinger is presented in [12]. The method and the implementation of that algorithm [12] enabled insightful studies [10, 11] on quarkonium production in ultra-relativistic heavy-ion collisions at the Relativistic Heavy-Ion Collider (RHIC) [32] and the Large Hadron Collider (LHC) [33]. The heavy quarkonium production at both LHC and RHIC is one of the observables that are of great help to enrich our understanding of nature, revealing the properties of the very hot and highly densed nuclear matter generated through energetic collisions of heavy ions [34, 35, 36, 37, 38]. The solver out of that implementation runs on a computer cluster consisting of several multi-core CPUs, and hence it can deal well with problems that are memory demanding. However it is purely CPU-based and thus cannot take good advantages that GPUs can offer.

A modern GPU is capable of running thousands of concurrent threads. These threads perform the same instruction over different data elements under the Single-Instruction-Multiple-Thread (SIMT) execution model. Due to their significant parallelism and high performance per watt, GPUs have been widely used for building general-purpose applications in both science and engineering [1, 2, 13, 14, 16, 39, 40]. Given the high performance of GPUs for general purpose computing in addition to graphic processing, it is interesting to develop a GPU-accelerated solver for the time-independent 3D Schrödinger equation, on the basis of [12, 23, 35]. The GPU-based solver introduced in this research runs on one GPU. It is demonstrated that the solver provides an efficient and alternative way of solving the 3D Schrödinger equation.

We organize this paper as follows. In Section 2, we outline the basic theory and the FDTD-based scheme for solving the 3D Schrödinger equation. The GPU-based solver as a result of a parallel implementation for one GPU is described in section 3. In Section 4, we validate the implementation, then the performance of the GPU-based solver is estimated for four use cases. In the last section, we present conclusions and discussions.

2. Outline of Theory and FDTD

On the basis of [12, 23, 35], a FDTD method, together with the quantum theory necessary to understand the FDTD approach for solving the time-independent Schrödinger equation is introduced below.

2.1. The Theory

The goal is to solve the time-independent Schrödinger equation for a particle of mass m:

$$\hat{H}\psi_n(\vec{r}) = E_n\psi_n(\vec{r}), \tag{1}$$

where $\psi_n(\vec{r})$ and E_n are the eigenfunction and the energy eigenvalue that solve this equation, with $\hat{H} = -\frac{\hbar^2}{2m}\nabla^2 + V(\vec{r})$ being the Hamiltonian operator and $V(\vec{r}, t) = V(\vec{r})$ is the 3D potential.

In quantum mechanics, a solution to the time-dependent Schrödinger equation

$$i\hbar\frac{\partial}{\partial t}\Psi(\vec{r}, t) = \hat{H}\Psi(\vec{r}, t) = \left[-\frac{\hbar^2}{2m}\nabla^2 + V(\vec{r})\right]\Psi(\vec{r}, t), \tag{2}$$

is formulated as

$$\Psi(\vec{r}, t) = \sum_{n=0}^{\infty} a_n\psi_n(\vec{r})\exp^{-iE_nt/\hbar}, \tag{3}$$

where a_n is the expansion coefficient corresponding to the nth state, with $n = 0$ representing the ground state, and $n = 1$ the first excited state, etc..

Setting $\hbar = 1$ and $\tau = it$ [12, 23, 35], Eq. 2 becomes

$$\frac{\partial}{\partial t}\Psi(\vec{r},\tau) = \left[\frac{1}{2m}\nabla^2 - V(\vec{r})\right]\Psi(\vec{r},\tau), \tag{4}$$

and its solution is

$$\Psi(\vec{r},\tau) = \sum_{n=0}^{\infty} a_n\psi_n(\vec{r})\exp^{-E_n\tau}. \tag{5}$$

Since $E_0 < E_1 < E_2 < ...$, the ground state wave function term $a_0\psi_0(\vec{r})\exp^{-E_0\tau}$ dominates in the limit when τ goes to infinity. Hence one may arrive at

$$\lim_{\tau\to\infty}\Psi(\vec{r},\tau) \approx a_0\psi_0(\vec{r})\exp^{-E_0\tau}. \tag{6}$$

Equation Eq. 6 indicates that evolving Eq. 4 to a pretty large τ one may obtain a good approximation of the ground state wave function $\psi_0(\vec{r})$ and then the ground state energy expectation value E_0 is calculated according to

$$E_0 = \frac{\langle\psi_0(\vec{r})\hat{H}\psi_0(\vec{r})\rangle}{\langle\psi_0(\vec{r})\psi_0(\vec{r})\rangle} = \frac{\int\psi_0(\vec{r})^*\hat{H}\psi_0(\vec{r})d^3\vec{r}}{\int\psi_0(\vec{r})^*\psi_0(\vec{r})d^3\vec{r}}. \tag{7}$$

2.2. FDTD-based numerical scheme

The FDTD method first of all involves discretization of Eq. 4 in order to get it numerically solved. The discretization is realized by using the finite difference method. Let δx, δy, δz be the spacing of the grid points in the 3D computation domain and $\delta\tau$ the step size to evolve $\Psi(\vec{r},\tau)$ in τ, and set $\delta x = \delta y = \delta z = a$, then the left hand side of Eq. 4 may be written as

$$\frac{\partial}{\partial\tau}\Psi(\vec{r},\tau) \approx \frac{\Psi(x,y,z,\tau+\delta\tau) - \Psi(x,y,z,\tau)}{\delta\tau}, \tag{8}$$

and from the the right hand side of Eq. 4 we have

$$\begin{aligned}\left[\frac{1}{2m}\nabla^2 - V(\vec{r})\right]\Psi(\vec{r},\tau) \approx{}& \frac{1}{2ma^2}\left[\Psi(x+a,y,z,\tau) - 2\Psi\left(\vec{r},\tau\right) + \Psi(x-a,y,z,\tau)\right]\\ &+ \frac{1}{2ma^2}\left[\Psi(x,y+a,z,\tau) - 2\Psi\left(\vec{r},\tau\right) + \Psi(x,y-a,z,\tau)\right]\\ &+ \frac{1}{2ma^2}\left[\Psi(x,y,z+a,\tau) - 2\Psi\left(\vec{r},\tau\right) + \Psi(x,y,z-a,\tau)\right]\\ &- \frac{1}{2}V\left(\vec{r}\right)\left[\Psi\left(\vec{r},\tau+\delta\tau\right) + \Psi\left(\vec{r},\tau\right)\right].\end{aligned} \tag{9}$$

Secondly, with the FDTD method $\Psi(\vec{r},\tau)$ evolves in τ according to an update equation. From Eq. 8 and Eq. 9, the update equation is as follows:

$$\Psi(x,y,z,\tau+\delta\tau) = b\Psi(x,y,z,\tau) + c\sum_{i=1}^{3}\left(A\times B^T\right)_i, \tag{10}$$

where the definition of b and c are

$$b = \frac{2-\delta\tau V(\vec{r})}{2+\delta\tau V(\vec{r})}, \tag{11}$$

$$c = \frac{\delta\tau}{ma^2\left[2+\delta\tau V(\vec{r})\right]}, \tag{12}$$

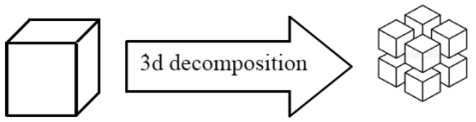

Fig. 1. A sketch of a 3D domain decomposed into 3D sub-domains.

and the definition of vector B and matrix A are

$$B = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}, \tag{13}$$

$$A = \begin{bmatrix} \Psi(x+a,y,z,\tau) & \Psi(\vec{r},\tau) & \Psi(x-a,y,z,\tau) \\ \Psi(x,y+a,z,\tau) & \Psi(\vec{r},\tau) & \Psi(x,y-a,z,\tau) \\ \Psi(x,y,z+a,\tau) & \Psi(\vec{r},\tau) & \Psi(x,y,z-a,\tau) \end{bmatrix}. \tag{14}$$

After a number of updates of the wave function, the ground state energy is estimated according to the discrete form of Eq. 7. For example, after kth update, we arrive at $\tau = k\delta\tau$, then the energy of the ground state can be estimated via

$$E[\tau] = \frac{\sum_i \Psi(\vec{r_i},\tau)\left[\frac{1}{2m}\sum_{i=1}^{3}\left(A \times B^T\right)_i - V(\vec{r})\Psi(\vec{r},\tau)\right]}{\sum_i |\Psi(\vec{r_i},\tau)|^2}. \tag{15}$$

The estimation converges to the ground state energy E_0 when a pre-set tolerance is satisfied. At this moment, the update terminates.

3. Implementation on a GPU

There are three domain decomposition strategies that are usually adopted to divide the continuous three-dimensional (3D) computation domain into discrete grid of points. The first one is one-dimensional (1D) decomposition, also called a slab decomposition. The second one is two-dimensional (2D) decomposition, also known as a pencil decomposition. The third is 3D decomposition, as is shown in Fig. 1. Here we adopt the 3D decomposition. Specifically, we first decompose the computation domain that is a 3D volume, into 3D sub-domains. Then, each of the sub-domain is treated as a 3D lattice. Along each dimension of the lattice, the spacing of neighbouring grid points is equally set. Depending on the amount of available memory and also the expected precision of the calculation, one may choose the spacing appropriately for given size of the 3D domain. After the decomposition, the 3D computation domain contains $(N_x + 1) * (N_y + 1) * (N_z + 1)$ grid points. These grid points are uniformly distributed in $N_x * N_y * N_z$ 3D sub-domains. We use Cartesian coordinate system, and the origin is at the center of the computation domain.

The GPU-based solver is mainly written in programming language C with CUDA extension, called CUDA C [19]. Here is a brief introduction of CUDA programming model and for more detailed description about CUDA, the interested readers are referred to [19]. CUDA C extends C, allowing a programmer to define C functions called kernels, which can be executed N times in parallel by N different CUDA threads. A number of threads are grouped to form a block of threads, called a thread block. A kernel can also be executed in parallel by multiple thread blocks. Thread blocks are organized into 1D, or 2D, or 3D grid of thread blocks. A grid of many thread blocks are utilized in order to hide latency of accessing global memory, when the size of the data being processed greatly exceeds the number of streaming multi-processor (SM) within a GPU. Thus CUDA provides a natural way to invoke parallel computation at space points in sub-domains that is from a domain decomposition, when the calculation is independent of space points. As one can see from Eq. 10, the independence is evident for calculating the value for the wave function at space points when updating the wave function. Examining Eq. 15, it can also be seen that for both the estimation of state energy and the normalization of the wave function one can invoke parallel computation at space points in the computation domains.

The GPU-based solver mainly consists of three modules, named initializeWavefunction, updateWavefunction, and calculateEnergy. These modules consisting of CUDA kernels, initialize and update the wave function, and calculate the energy of state, respectively. The kernel that initializes the wave function does its job on the device. That kernel,

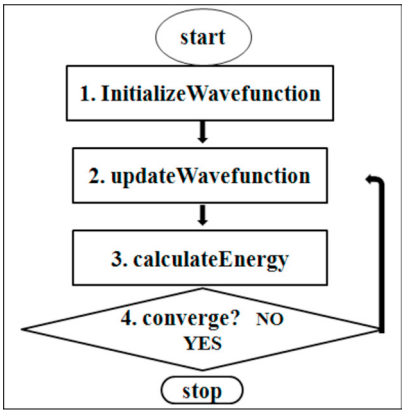

Fig. 2. The flowchart of the GPU-based solver, which mainly consists of three modules.

together with the two kernels that calculate the value of the energy and wave function at each space point, is executed by a 3D grid of thread blocks, and each of the thread blocks contains 3D threads. After the values of the energy and the wave function for all of the space points are ready, kernels that employ a parallel reduction procedure are used to get normalization factor and the energy of state. The flowchart of the main modules of the parallel program is shown in Fig. 2. Under our 3D decomposition strategy, we get a number of sub-domains. Through these kernels, each of the sub-domains is handled by one thread block. One may hence note that here the number of thread blocks is equal to the number of the sub-domains.

At the beginning when $\tau = 0$, the wave function is assumed a finite constant number for all of the grid points, and at the boundary the wave function is set to zero all the time. By design, the values of $\Psi(x, y, z, \tau)$ for all the grid points are stored in global memory. In so doing, seldomly there is a need for the normalization of wave function, the calculation of energy, and the update of wave function to access host memory. After each update, normalization of the wave function is carried out, avoiding numerical underflow that may happen because the norm of the wave function is decreasing.

4. Results

In this section, results about the ground state energy corresponding to four 3D potentials are presented. These results show the correctness of the GPU-based many-core solver. Additionally, benchmarks are presented that show the effectiveness of the solver. In order to have an idea about the effectiveness of the GPU-based solver, we have written a CPU-based solver, which is a serial solver utilizing one core of the multi-core CPU, following the aforementioned FDTD-based scheme. Speed-up of the parallel solver against the serial solver is calculated, according to the time it takes for the two solvers to complete a large number of update of the wave function when solving the 3D Schrödinger equation. Variation of initial conditions would introduce run-by-run fluctuations in the time it takes for the running the program. Therefore to avoid the fluctuations we used the same initial condition in every run of the two solvers. To achieve stability, spacing a and time step $\delta\tau$ are chosen to make the relation $\delta\tau < a^2/3$ hold true. According to the size of the global memory and the size of the computing domain, the 3D block dimension and 3D grid dimension values are set to: 4x4x8 and 64x64x32, respectively. Additionally, atomic units are used in the calculation.

The hardware configuration and a few software info are as follows.

- CPU: Xeon E5-2680v2@2.80GHz
- GPU card: Nvidia Tesla K20m
- Operating system: Linux kernel-2.6.32-358.el6.x86_64
- System main memory: 128GB
- Command line for compilation is nvcc -arch=sm_35 -O2 -o main main.cu
- CUDA Toolkit version: V6.0.1

4.1. Coulomb potential benchmarks

The following Coulomb potential is used.

$$V(r) = \begin{cases} 0 & r < a \\ -\frac{1}{r} + \frac{1}{a} & r \geqslant a, \end{cases} \tag{16}$$

where a is the lattice spacing in units of the Bohr radius and r is the distance between a grid point at (x,y,z) and the center of the 3D computation domain. We take this form from [12], and as stated in [12], the additional $1/a$ term ensures that the potential is continuous at $r = a$ and that it shifts the entire potential by a constant does not affect the binding energy. The exact solution of the 3D Schrödinger equation with this 3D potential is known, with the binding energy of the nth state being $E_{n,binding} = -1/(2(n+1)^2)$ when the particle mass m is set to 1. Therefore $E_{binding,0} = -0.5$ is the binding energy for the ground state.

The binding energy for the ground state is estimated at kth update of the wave function via

$$E_{binding}\,[k\delta\tau] = E[k\delta\tau] - V_\infty, \qquad V_\infty = \lim_{r\to\infty} V\left(\vec{r}\right) = 1/a, \tag{17}$$

where $r = \sqrt{x^2 + y^2 + z^2}$.

In table. 1, we tabulated the calculated binding energy and the number of updates of the wave function. Since $\tau = k\delta\tau$, where k is the number of updates. Table. 1 demonstrates that the binding energy from the the GPU-based solver converges toward ground state binding energy -0.5 when τ takes a fair large value. When the number of updates k is 15403, the preset convergence tolerance 1.0×10^{-7} is satisfied. At this moment, the binding energy calculated via Eq. 17 converges to -0.496378. This result, with the relative error being about 0.72%, validates the correctness of the GPU-based solver.

Table 1. The binding energy $E_{binding}$ as a function of the number of updates k, for the case of Coulomb potential.

$\mathbf{k}(\times 10^2)$	66	88	110	132	154
$\mathbf{E}_{binding}(-a.u.)$	0.390	0.464	0.488	0.495	0.496

We have run the two solvers 12 times for several cases. In each case, the number of updates is fixed. These runs report computing time, from which we then calculated the speed-up. The speed-up as a function of the number of updates is shown in Fig. 3, which reveals that the speed-up tends to saturate when the number of updates gets fairly high, i.e. $k \geqslant 1800$. The speed-up is about 93 ± 3 when the number of updates is 1800. Therefore the GPU-based solver is very efficient compared to the CPU-based solver.

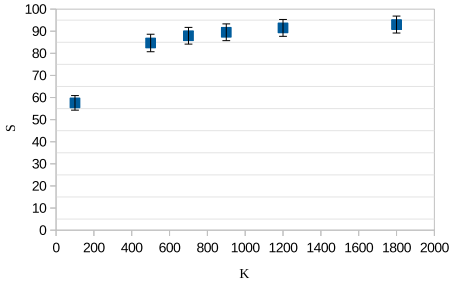

Fig. 3. The speed-up as a function of the number of updates for the case of Coulomb potential.

For this benchmark, we used a constant lattice spacing of $a = 0.08$, and a constant step size of $\delta\tau = 1.0 \times 10^{-3}$.

4.2. 3D harmonic oscillator benchmarks

In the case of 3D harmonic oscillator, the following 3D potential is used

$$V(r) = \frac{1}{2}r^2, \tag{18}$$

where r is the distance between grid point (x,y,z) and the center of the 3D computation domain. The exact solution of the 3D Schrödinger equation with this 3D potential is known, with state energy of the nth state being $E_n = (n + 3/2)$, where n is the quantum number that marks the energy levels and $n \geq 0$. Analytically the energy for the ground state is $E_0 = 1.5$ when $n = 0$.

In Table. 2, we tabulated the state energy and the number of updates of the wave function. Table. 2 reveals that the energy calculated by using the GPU-based solver, converges toward the exact value of the ground state energy 1.5 as the number of updates gets large enough. This is consistent with the implication of Eq. 6, demonstrating that the FDTD-based solver developed for running on GPU is applicable to the calculation of the ground state energy. When the number of updates k is 5004, the preset convergence tolerance 1.0×10^{-7} is satisfied. At this moment, the state energy is 1.500825, a good estimation of the ground state energy. The relative error is 0.055%. With this result, the correctness of the GPU-based solver is validated.

Table 2. The state energy E_0 estimated via $E[k\delta\tau]$ as a function of the number of updates k, for the case of 3D harmonic oscillator.

$\mathbf{k}(\times 10^3)$	1	2	3	4	5
$\mathbf{E}_0(a.u.)$	2.256	1.627	1.525	1.505	1.501

For estimation of the effectiveness of the GPU-based solver, we calculated the speed-up of the GPU-based solver relative to the CPU-based solver. By using the computing time out of 12 runs of the two solvers, the speed-up for calculating the ground state energy is estimated to be 67 ± 1. This result validates the effectiveness of the GPU-based solver, and shows that it is more efficient than the CPU-based solver.

For this benchmark, a constant lattice spacing of $a = 0.04$, and a constant imaginary time step of $\delta\tau = 4.0 \times 10^{-4}$ are used in the calculation.

4.3. Benchmarks for the three coupled anharmonic oscillators

A typical 3D example is the three coupled anharmonic oscillators, for which the potential is as follows.

$$V(x, y, z) = V(x) + V(y) + V(z) + xy + yz + zx, \tag{19}$$

where

$$V(x) = 0.5x^2 + 2x^4 + 0.5x^6, \qquad V(y) = 0.5y^2 + 2y^4 + 0.5y^6, \qquad V(z) = 0.5z^2 + 2z^4 + 0.5z^6, \tag{20}$$

Table 3. The state energy E_0 estimated via $E[k\delta\tau]$ as a function of the number of updates k, for the three coupled anharmonic oscillators.

$\mathbf{k}(\times 10^2)$	9	30	60	80	95.41
$\mathbf{E}_0(a.u.)$	5.146	3.12	2.982	2.978	2.978

In Table. 3, we tabulated the calculated ground state energy and the number of updates of the wave function. Table. 3 reveals that the energy calculated by using the GPU-based solver, converges toward the pretty accurate value [31] of 2.978303 as the number of updates gets large enough. This is consistent with the implication of Eq. 6, demonstrating that the FDTD-based solver developed for running on GPU is applicable to the calculation of the ground state energy. When the number of updates k is 9541, the preset convergence tolerance 1.0×10^{-7} is satisfied. At this moment, the

ground state energy is estimated to be 2.977861, which corresponds to a relative error of 0.014% relative to the highly accurate value 2.978303 reported in [31]. With this result the correctness of the GPU-based solver is validated.

Both the GPU-based and the CPU-based solvers were executed 12 times. These runs reported computing time for obtaining the ground state energy. Dividing the computing time from running the CPU-based solver by that from the GPU-based solver, we obtained the speed-up, and the result is 47 ± 3. This result demonstrates that running on GPU the parallel solver is quite efficient, relative to the serial program running on CPU.

For this benchmark, a constant lattice spacing of $a = 0.03$, and a constant imaginary time step of $\delta\tau = 1.0 \times 10^{-4}$ are used in the calculation.

4.4. H_2^+ benchmarks

H_2^+ stands for an ionized hydrogen molecule H_2. In H_2^+, there are two protons and one electron. The following 3D potential is considered:

$$V(r) = -\frac{1}{r_1} - \frac{1}{r_2}, \tag{21}$$

where r_1 and r_2 are the distance between the electron and the two protons, respectively:

$$r_1 = \sqrt{r^2 + 2z + 1}, r_2 = \sqrt{r^2 - 2z + 1}, \tag{22}$$

where r is the distance between the electron at the grid point (x,y,z) and the mid-point between the two protons. This mid-point is at the center of the 3D computation domain.

In Table. 4, we tabulated the calculated electron state energy as a function of the number of updates of the wave function. Table. 4 reveals that the energy calculated by using the GPU-based solver, converges toward the highly accurate electron ground state energy [41] of -1.10263 as the number of updates gets larger. This is consistent with the implication of Eq. 6, demonstrating that the FDTD-based solver developed for running on GPU mat be used for calculating the ground state energy. When the number of updates k is 21611, the preset convergence tolerance 1.0×10^{-7} is satisfied. At this moment, the electron ground state energy is estimated to be -1.100022. This result is in good agreement with the highly accurate value -1.10263 reported in [41]. With this result the correctness of the GPU-based solver is validated. We have run each of the two solvers, i.e. the GPU-based and the CPU-based solver, 12

Table 4. The electron state energy E_0 estimated via $E[k\delta\tau]$ as a function of the number of updates k, for H_2^+.

$\mathbf{k}(\times 10^3)$	4	8	13	16.69	21.611
$\mathbf{E}_0(-a.u.)$	0.902	1.076	1.098	1.099	1.1

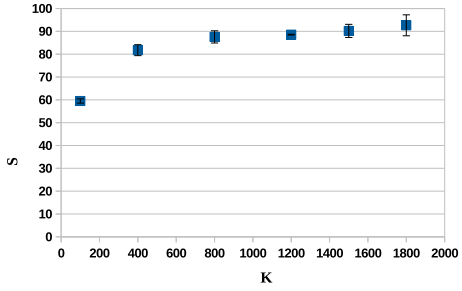

Fig. 4. The speed-up as a function of the number of updates for the case of H_2^+.

times for several cases. In each case, the number of updates is fixed. These runs report computing time, from which we

then calculated the speed-up. The speed-up as a function of the number of updates is shown in Fig. 4, which reveals that the speed-up tends to saturate when the number of updates gets fairly high, i.e. $k \geqslant 1800$. The speed-up is 93 ± 5 when the number of updates is 1800. This result shows that when getting the electron ground state energy for H_2^+, running on a many-core GPU the parallel solver is quite efficient compared with the serial program that utilizes one core of the multi-core CPU.

For this benchmark, a constant lattice spacing of $a = 0.03$, and a constant imaginary time step of $\delta\tau = 2.8 \times 10^{-4}$ are used in the calculation. The separation between the two protons is 2.

5. Conclusions and Discussions

A GPU-based solver for the 3D Schrödinger equation was presented. We fisrt introduced an algorthm that is based on a FDTD method for solving the 3D Schrödinger equation. Then an implementation of the algorithm by using CUDA C was described. Although FDTD methods are well-known and many FDTD-based solvers have been developed, in this paper we presented the first GPU-accelerated solver that is based on the specific FDTD method reported in [23].

The solver is validated by using four well-known potentials, i.e. 3D harmonic oscillator, 3D Coulomb potential, three coupled anharmonic oscillators, and H_2^+. It would be interesting to test the GPU-based solver by using more complicated potentials, such as the one formulated in [11, 35, 38]. We will do the sophisticated and important test in the near future.

The GPU-based solver is very effective. Against the serial solver, we arrived at a speed-up of about 90X, 60X, 40X, and 90X in the case of 3D Coulomb potential, 3D harmonic oscillator, three coupled anharmonic oscillators, and H_2^+, respectively. Both the serial solver and the GPU-based solver are parts of the main program. The main program defines the context in which the serial solver runs after the GPU-based solver terminates.The two solvers were compiled at the same optimization level, which is O2. Even when all the cores of the CPU are utilized in parallel with the aid of openMP, since the CPU we used has 10 cores, the speed-ups are still encouraging, at least approximately between 4 and 9 corresponding to the four cases.

Given the effectiveness in terms of the significant speed up, the solver that employs one GPU is helpful for solving the 3D Schrödinger equation. It promises to avoid using of a small computer cluster when the potential is simple and memory usage is not very demanding. Certainly, solving Schrödinger equation means not only to get the wave function and the energy of the ground state, but also means to get those of the excited states. As pointed out in [12, 23], the excited states can be processed basically in the same way that the ground state is handled. Therefore the GPU-based solver can not only correctly but also efficiently solve the 3D time-independent Schrödinger equation.

Acknowledgements

This work was supported in part by Beijing Natural Science Foundation grant no. 1132017, Beijing Municipal Education Commission's major scientific project grant no. KZ201910017019, and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry. We thank the Computer Network Information Center and the Institute of Process Engineering, the Chinese Academy of Sciences, for providing computing support.

References

[1] I.K. Gainullin. (2017) "High-performance GPU parallel solver for 3D modelling of electron transfer during ion-surface interaction." J. Comput. Phys. 210: 72–78.

[2] I. K. Gainullin. (2017) "Three-dimensional modelling of resonant charge transfer between ion beams and metallic surfaces." Phys. Rev. A 95(5): article ID. 052705, 16 pages.

[3] S. Yue, H. Du, H. Wu, J. Li, B. Hu. (2017) "Wavelength dependence of high-harmonic yield in stretched molecules." Chem. Phys. 494: 56–60.

[4] Q. L. Guo, P. C. Li, X. X. Zhou, S. I. Chu. (2018) "Efficient enhancement of below-threshold harmonic generation by laser-driven excited states of Cs atom." Opt. Commun. 410: 262–268.

[5] Y. He, L. He, P. Lan, B. Wang, L. Li , X. Zhu, W. Cao , and P. Lu.(2019) "Direct imaging of molecular rotation with high-order-harmonic generation." Phys. Rev. A 99(5): article ID. 053419, 8 pages.

[6] J. Zhang, H. Du, H.F. Liu, J. Guo, and X.S. Liu. (2016) "Multichannel molecular high-order harmonic generation from HeH_2^+ with the combination of chirped laser and a unipolar pulse." Opt. Commun. 366: 457–461.

[7] W. Zhang et al.. (2019) "Electron-nuclear correlated multiphoton-route to Rydberg fragments of molecules." Nat. Commun. 10: article ID. 757, 8 pages.
[8] X.Y. You and F. He. (2014) "Dissociation of H_2^+ into selected electronic states in strong laser fields." Phys. Rev. A 89(6): article ID. 063405, 7 pages.
[9] P. Lu et al.. (2018) "High-order above-threshold dissociation of molecules." Proc. Natl. Acad. Sci. USA 115(9):2049–2053.
[10] M. Strickland. (2011) "Thermal Υ (1s) and χ (b1) suppression in $\sqrt{s_{NN}}$ = 2.76 TeV Pb-Pb collisions at the LHC." Phys. Rev. Lett. 107(13): article ID. 132301, 4 pages.
[11] B. Krouppa, A.Rothkopf, and M. Strickland. (2018) "Bottomonium suppression using a lattice QCD vetted potential." Phys. Rev. D 97(1): article ID. 016017, 13 pages.
[12] M. Strickland, and D. Yager-Elorriaga. (2010) "A parallel algorithm for solving the 3D Schrödinger equation." J. Comput. Phys. 229(17): 6015–6026.
[13] I.K. Gainullin and M.A. Sonkin. (2015) "High-performance parallel solver for 3D time-dependent Schrödinger equation for large-scale nanosystems." Comput. Phys. Comm. 188: 68–75.
[14] C. Ó Broin and L.A.A. Nikolopoulos. (2014) "A GPGPU based program to solve the TDSE in intense laser fields through the finite difference approach." Comput. Phys. Comm. 185(6): 1791–1807.
[15] Y. Fu, J. Zeng, and J. Yuan. (2017) "PCTDSE: A parallel Cartesian-grid-based TDSE solver for modelling laser-atom interactions." Comput. Phys. Comm. 210: 181–192.
[16] J.P. Wilson. (2019) "Generalized finite-difference time domain method with absorbing boundary conditions for solving the nonlinear equation on a GPU." Comput. Phys. Comm. 235: 279–292.
[17] OpenMP Architecture Review Board. "OpenMP." https://www.openmp.org/.
[18] MPI Forum. "MPI." https://www.mpi-forum.org/.
[19] NVIDIA Corporation. "CUDA toolkit." https://developer.nvidia.com/cuda-toolkit-60.
[20] A. Wadehra, A.K. Roy, and B.M. Deb. (2003) "Ground and excited states of one-dimensional self-interacting nonlinear oscillators through time-dependent quantum mechanics." Int. J. Quantum Chem. 91(5): 597–606.
[21] D.M. Sullivan and D.S. Citrin. (2005) "Determining quantum eigenfunctions in three-dimensional nanoscale structures." J. Appl. Phys. 97(10): article ID. 104305, 6 pages.
[22] A.K. Roy, A.J. Thakkar, and B.M. Deb. (2005) "Low-lying states of two-dimensional double-well potentials." J. Phys. A: Math. Theor. 38(10): 2189–2200.
[23] I. W. Sudiarta and D. J. W. Geldart. (2007) "Solving the Schrödinger equation using the finite difference time domain method." J. Phys. A: Math. Theor. 40(8): 1885–1896.
[24] F.I. Moxley III, T. Byrnes, F. Fujiwara, and W. Dai. (2012) "A generalized finite-difference time-domain quantum method for the N-body interacting Hamiltonian." Comput. Phys. Comm. 183(11): 2434–2440.
[25] J. Shen, W.E. Sha, Z. Huang, M. Chen, and X. Wu. (2013) "High-order symplectic fdtd scheme for solving a time-dependent Schrödinger equation." Comput. Phys. Comm. 184(3): 480–492.
[26] M.D. Feit, J. Fleck, and A. Steiger. (1982) "Solution of the Schrödinger equation by a spectral method." J. Comput. Phys. 47(3): 412–433.
[27] P.A. Milewski and E.G. Tabak. (1999) "A pseudo-spectral algorithm for the solution of nonlinear wave equations with examples from free-surface flows." SIAM J. Sci. Comput. 21(3): 1102–1114.
[28] J. Lo and B.D. Shizgal. (2008) "Pseudospectral methods of solution of the Schrödinger equation." J. Math. Chem. 44(3): 787–801.
[29] A. Askar, A.S. Cakmak. (1978) "Explicit integration method for the time-dependent Schrödinger equation for collision problems." J. Chem. Phys. 68(6): 2794–2798.
[30] Q.J. Liu and W.Q. Zhao. (2010) "Iterative solution for ground state of H_2^+ ion." Commun. Theor. Phys. 53(1): 57–62.
[31] M. Kaluza.(1994) "Analytical Lanczos method: quantum eigenstates of anharmonic oscillators in one or more dimensions." Comput. Phys. Comm. 79(3): 425–446.
[32] BNL (Brookhaven National Laboratory). "RHIC." https://www.bnl.gov/RHIC/.
[33] CERN (European Organization for Nuclear Research). "LHC." https://home.cern/science/accelerators/large-hadron-collider.
[34] T. Matsui and H. Satz. (1986) "J/Ψ suppression by quark-gluon plasma formation." Phys. Lett. B 178(4): 416–422.
[35] A. Dumitru, Y. Guo, A. Mocsy, and M. Strickland. (2009) "Quarkonium states in an anisotropic QCD plasma." Phys. Rev. D 79(5): article ID. 054019, 10 pages.
[36] PHENIX Collaboration, A. Adare, et al.. (2007) "J/Ψ production vs centrality, transverse momentum, and rapidity in Au + Au collisions at $\sqrt{s_{NN}}$ = 200 GeV." Phys. Rev. Lett. 98: article ID. 232301, 6 pages; (2007) Erratum Phys. Rev. Lett. 98: article ID. 249902.
[37] P. Dillenseger for ALICE Collaboration. (2018) "Quarkonium measurements in nucleus-nucleus collisions." Nucl. Phys. A 982: 703–706.
[38] M. Abu-Shady, H. M. Mansour, and A. I. Ahmadov. (2019) "Dissociation of quarkonium in hot and dense media in an anisotropic plasma in the nonrelativistic quark Model." Adv. High Energy Phys. 2019: article ID. 4785615, 10 pages.
[39] Q.J. Liu, W.Q. Zhao, F. Liu, N.M. Nie, and C.B. Zhou. (2016) "GPU-accelerated parton cascade in heavy-ion collisions." Int. J. Comput. Theor. Eng. 8(6): 439–443.
[40] NVIDIA Corporation. "GPU Applications catalog." https://www.nvidia.com/en-us/data-center/gpu-accelerated-applications/catalog/.
[41] H. Wind. (1965) "Electron energy for H_2^+ in the ground state." J. Chem. Phys. 42(7): 2371–2374.