GPU-Accelerated Matrix Exponent for Solving 1D Time-Dependent Schrödinger Equation

Yea Rem Choi[1(✉)] and Vladimir Stegailov[1,2]

[1] HSE University, Moscow, Russia
e.choj@hse.ru
[2] Joint Institute for High Temperatures of RAS, Moscow, Russia
stegailov@jiht.ru

Abstract. Non-adiabatic electron-ion quantum dynamics is still an area of many unresolved problems even for such simple systems as the H_2^+ molecular ion. Mathematical modelling based on time-dependent Schrödinger equation (TDSE) is an important method that can provide better understanding of these phenomena. In this work, we present TDSE solution for 1D TDSE that describes non-adiabatic electron-ion quantum dynamics for the simplified H_2^+ model. For solving TDSE, we use the real-space representation and the matrix exponent method that is quite computationally expensive but is free from usual symmetry-based simplifications. For this purpose, we make use of the very high performance of modern Nvidia V100/A100 GPUs and deploy our parallel multi-GPU matrix multiplication algorithm.

Keywords: TDSE · Non-adiabatic quantum dynamics · Hydrogen molecular ion · Soft-core Coulomb potential · Parallel computing

1 Introduction

Despite the significant progress in the understanding of the quantum processes many phenomena even in such simple systems as molecular hydrogen are still objects of active experimental and theoretical studies, e.g. the dynamics of attosecond photoionization of H_2^+ [1] and H_2 [2]. Despite the simplicity, the models of electronic structure of such molecules still have such approximations as, for example, the use of the soft-core Coulomb potential [3]. The description of the non-adiabatic electron-ion quantum dynamics of these molecules is beyond the capabilities of the analytical solutions of TDSE and requires the high performance computing methods. Since we consider the time evolution of isolated systems the finite-difference time domain (FDTD) approach for solving TDSE seems to be a reasonable choice. For example, the FDTD approach was successfully applied for solving TDSE of quantum dots [4].

In this work, we report our attempt to use the FDTD method based on the matrix exponent for solving TDSE. The matrix exponent approach is one of the most generic methods for solving differential equations. The calculation of

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
V. Voevodin et al. (Eds.): RuSCDays 2023, LNCS 14388, pp. 100–113, 2023.
https://doi.org/10.1007/978-3-031-49432-1_8

the matrix exponent has been a computationally hard problem for a long time. Various approaches of acceleration for particular matrix were developed [5,6]. The development of the corresponding computational tools provided new appeal to the matrix exponent-based FDTD calculations [7].

2 Related Works

TDSE for simple systems have been considered in the context of the interaction of an atom or a molecule with ultrashort atomic pulses. In the paper of Lugovskoy and Bray an almost sudden perturbation of a quantum system by a ultrashort pulse has been considered [8]: the analytical theory has been compared with numerical solution of TDSE. Different scenarios of He ionization have been considered by numerically solving 1D TDSE by Yu and Madsen [9,10]. Non-adiabatic quantum dynamics of H_2^+ and HD^+ excited by single one-cycle laser pulses linearly polarized along the molecular axis have been studied within a three-dimensional model by Paramonov et al. [11]. Strong laser field interactions with He, H_2^+, and H_2 have been modelled by Majorosi et al. [12].

In several papers, the authors used the simplified 1D approximation with soft-core Coulomb potentials [8–10,12]. This type of electron-ion potentials is used in different atomic models (e.g., see the recent paper of Truong et al. [13]). The most accurate 3D model of molecular hydrogen based on the exponential split operator method [14] was used by Paramonov et al. [11]. The method presented in this work is free from the any symmetry assumptions and can be used with any form of potentials. The method is extendable to 3D geometry too. With our computational algorithm we have obtained the first results on the non-adiabatic vibronic energy transfer of energy from moving ions to electron subsystem: to the best of our knowledge this process was not considered before at the TDSE level of theory.

3 1D TDSE Model of a H_2^+ Molecular Ion

In this work we consider 1D TDSE for a single electron

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

where i is the imaginary unit, $\hbar$ is the Planck constant, ψ is the wave function, and $H(t)$ is the Hamiltonian matrix that can depend on time and has kinetic (T) and potential (V) parts as follows

$$H = T + V = -\frac{\hbar}{2m}\frac{\partial^2}{\partial r^2} + V(r,t). \tag{2}$$

In the spatial form TDSE can be approximated by the second order finite-difference scheme as [15]

$$i\hbar\frac{\partial}{\partial t}\psi_i = H\psi_i = -\frac{\hbar^2}{2m}\left(\frac{\psi_{i+1} - 2\psi_i + \psi_{i-1}}{\Delta r^2}\right) + V(r_i)\psi_i, \tag{3}$$

that can be written in the matrix form as

$$H = -\frac{\hbar^2}{2m\Delta r^2}\begin{pmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & -2 & 1 & \\ & & & 1 & -2 & \end{pmatrix} + \begin{pmatrix} V(r_0) & & & & & \\ & \ddots & & & & \\ & & V(r_i) & & & \\ & & & \ddots & & \\ & & & & V(r_N) & \end{pmatrix}. \quad (4)$$

The finite-difference solution of TDSE (1) can be expressed using the matrix exponent as

$$\psi(r, t + \Delta t) = \exp\left[iH\Delta t\right]\psi(r, t), \quad (5)$$

where the exponent of a matrix A is defined as the infinite series

$$\exp A = I + \frac{1}{1!}A + \frac{1}{2!}A^2 + \frac{1}{3!}A^3 + \ldots. \quad (6)$$

In (3) and (4) the Planck constant ($\hbar$) and the mass (m) are defined as 1 in our calculations to avoid the work with huge or tiny numbers. The absorbing boundary conditions are implemented.

The 1D hydrogen models with a soft-core Coulomb of electron-nucleus interaction is considered

$$V(r) = \frac{-Z}{\sqrt{r^2 + a}}. \quad (7)$$

To mimic the ionic vibration we have implemented the movement of the ion centers

$$V(r, t) = \frac{-Z}{\sqrt{(r - \alpha \sin(\beta t))^2 + a}}, \quad (8)$$

where Z is the soft-core Coulomb interaction strength, a is the softening parameter [9], α and β are some constants.

The stationary state of the wave function can be found by the time-independent equation

$$H\psi(r) = E\psi(r). \quad (9)$$

Here E is a constant value, so ψ in stationary state is the eigenvector of Hamiltonian matrix H and E is the eigen energy of the corresponding state (see Fig. 1).

4 Remarks About the Matrix Exponent Calculation

The matrix exponent algorithm used in this work for solving of TDSE (5) was developed based on our multi-GPU GEMM algorithm [16] that shows high performance for rather big matrices. So we took the attempt to build the program which does not use any assumptions about the matrix structure and calculates the matrix exponent for comparatively big matrix sizes.

Our multi-GPU GEMM algorithm uses only GPU devices for computation and data store, because it was aimed to the platforms with high speed links

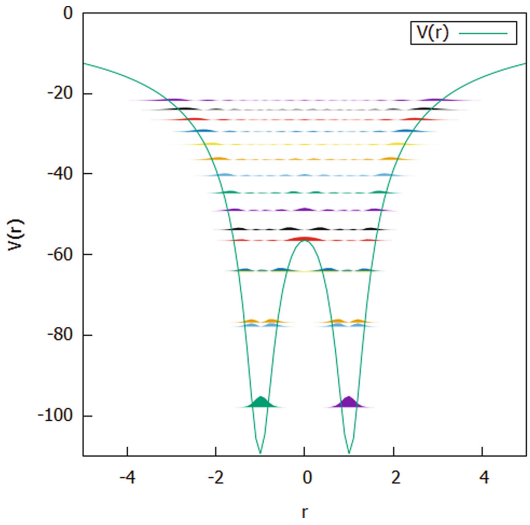

Fig. 1. The example of electron density distributions $|\psi_n(r)|^2$ for several bound states on the corresponding energy level positions in the soft-core Coulomb potential of two ions (7) ($Z = 30, a = 0.1$).

between GPUs (e.g., NVLink). The algorithm works asynchronously and reaches its high performance by overlapping computation and communication. It shows the performance rate rather close to the theoretical peak performance when it works with big matrices.

However, the proposed algorithm for solving TDSE has parts executed on CPU as well. In the beginning, the initial state ψ_0 is set in CPU memory and sent to GPUs. Also, the time-dependent potential $V(r, t)$ on each time step is defined in CPU memory then supplied to GPUs. The output operation of storing ψ at the specified moments of time into data files goes through CPU as well.

Yet, the most part of the algorithm is executed on GPU devices. The potential V in the Hamiltonian matrix is reinitialized each time step. The real and imaginary parts of matrices are stored in GPUs memory. The following objects are stored in GPU memory: the initialized complex matrix, the k-th term, and the resulted matrix to which each $(k+1)$-th term is added. To find the next term of the complex matrix 4 matrix multiplications are used: two for the real part $(Re * Re - Im * Im)$ and two for the imaginary part $(Re * Im + Im * Re)$. Also, the matrix-to-vector multiplication $(\exp H\psi)$ is done on GPUs, identically, two operations for the real part and two operations for the imaginary part at each time step. The algorithm scheme is shown at Algorithm 1.

The matrix exponent series (6) converges exponentially fast [17], so we have the question, when the convergence is reached and we can stop the computing and adding the subsequent terms due their small impact on the result. One method is to check the matrix elements if they are lesser than some fixed ϵ (this method may be applied in next version of the program). Currently, the selection

Algorithm 1. The algorithm scheme executed in GPUs during one time step, ψ is the wave function, H is the generated Hamiltonian matrix, $\exp A$ is the resulted matrix exponent. All operations are done separately for real and imaginary matrices. The communications between GPUs are not shown.

receive (potential V);
build Hamiltonian matrix ($H = T + V$), $A = iH\Delta t$;
$\exp A = I + A$;
for $k = 2$ to rank **do**
  $\alpha = 1/k$;
  Multi-GPU GEMM ($A^k = \alpha A A^{k-1}$);
  $\exp A = \exp A + A^k$;
end for
GEMV ($\psi_j = \exp A\ \psi_{j-1}$);
send (ψ_j); //for output in file

of the number of terms in the series is determined by the biggest row (column) of the matrix to which we apply the exponential. The matrix (4) is symmetric, so the biggest row (column) determines the slowest converging element of the series.

From (3) and (4) we see that the matrix exponent in TDSE has only an imaginary part, so the series can be transformed as

$$\exp iA = \left(I + \frac{i^2}{2!}A^2 + \ldots\right) + \left(\frac{i}{1!}A + \frac{i^3}{3!}A^3 + \ldots\right) = \cos A + i \sin A. \quad (10)$$

In such way we can make the algorithm simpler by separating the real and imaginary parts (till this moment the program has been developed for the general case). For optimization we take into some features of trigonometric matrices [18]. Since we have to get trigonometric matrices it could be anticipated that the elements of the resulted matrix should be in the range of $[-1, 1]$. As the matrix is symmetric (4), if the absolute value of the element is bigger than a natural k, then all terms till the k-th would be definitely bigger than 1 ($(a_{ij})^k/k!, a_{ij} > k$). Then, if we suppose that there are not much elements with comparably big values, the $k+1$ term and the following terms would decrease exponentially fast. Such "hump" phenomenon is known to cause some problems [6]. Accordingly, the margin of error increases dependent on the maximal value of the summing floating points. So, if the elements of the original matrix are much bigger than 1, then after the computation we can receive the margin of error comparable to 1. Here is one of the restrictions on the time step value (the smaller is the time step Δt, the smaller are the values of the matrix elements). Thus, if we would use a small time step, then, firstly, the series would start decreasing faster, and, secondly, we would have better accuracy, but, it would be required to compute more iteration for the same time period.

Talking about the distance step Δr, reducing it causes increase of kinetic energy part of the matrix (4). In other words, scaling up can matter the issues

mentioned above as well, reasoning the necessity of the time step decrease as $\Delta r \sim \Delta t^2$.

There are some issues with allocating GPU memory. In GPUs we need to allocate memories of four complex matrices: the initial one, the k-th term, the $(k+1)$-th term and the resulted one. The real and imaginary parts can be stored separately in different devices, so they can be spread up to eight GPUs. Memory allocation size of one matrix is needed on devices with the resulted matrices (real and imaginary) to receive the data of k-th term and to do summation. Storing complex wave functions (ψ_j and ψ_{j+1}) and the vector of potential energy (V) requires much less memory. Additionally, during the multi-GPU GEMM execution there are needed temporary memory dependent on size of tile matrices [16]. This volume would be smaller for small tile size, but, it is recommended to choose optimal one dependent on the platform parameters [19,20].

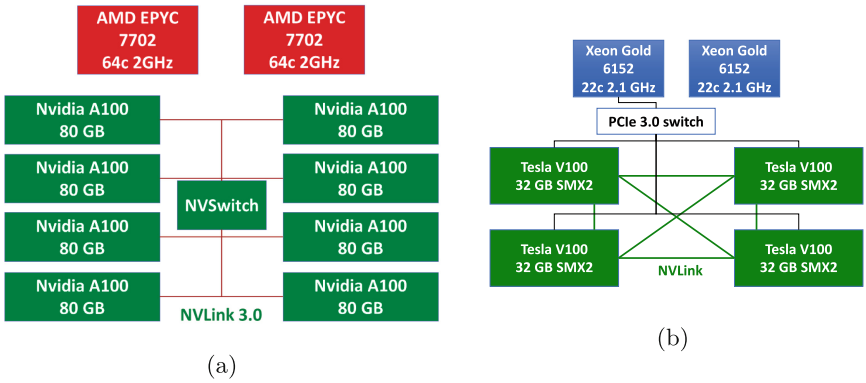

Fig. 2. The topology of the A100-equipped node of the cHARISMa supercomputer with two CPUs and eight Nvidia A100 GPUs by NVLink 3.0 (a) and the V100-equipped node of the cHARISMa supercomputer with two CPUs and four Nvidia V100 GPUs by NVLink 2.0, configuration K (type A, B) (b).

5 Testing Platforms

The results reported in this study are obtained on the nodes of the cHARISMa supercomputer at HSE University [21,22]. The first kind of nodes are based on the 8x Nvidia A100 GPU "type E" platform with NVSwitch (Fig. 2a). Each GPU has 80 Gb of HBM2 memory, and the eight GPUs are connected by NVLINK 3.0 via NVSwitch.

The second kind of nodes are based on the 4x Nvidia V100 GPU "type A, B, C" platform (Fig. 2b). Each GPU has 32 Gb of HBM2 memory, and the four GPUs are connected by NVLINK 2.0. Between GPUs there are no differences for configuration K (types A and B) and configuration M (type C).

The benchmarking studies were carried out using the standard HPC software stack based on CentOS Linux release 7.9.2009, GNU compilers 8.3.0, and CUDA Version 11.7.64 with the driver ver. 515.43.04.

6 Analysis of Numerical Experiments

The computations are performed with single precision numbers (FP32). One of the reason for this choice is that while using the tensor core technology (TF32) the performance rate increases greatly in comparison to double precision (TF64), but certain problem is the accuracy. As it is described in Sect. 4, we limit the maximal value of matrices by 8 (and better by 1) by regulating the time step. Then in the case of 8 the last 30-th ($k = 30$) term would be $\max a_{ij}^k \lesssim 8^{30}/30! \approx 4.7 * 10^{-6}$ and in the case of 1 the last 10-th ($k = 10$) term would be $\max a_{ij}^k \lesssim 1^{10}/10! \approx 2.8 * 10^{-7}$. The simple non-matrix example of the sequence behavior is illustrated in Fig. 3. The sequence converges to a number in range $[-1, 1]$, but has a "hump" due the first terms which increases the error rate.

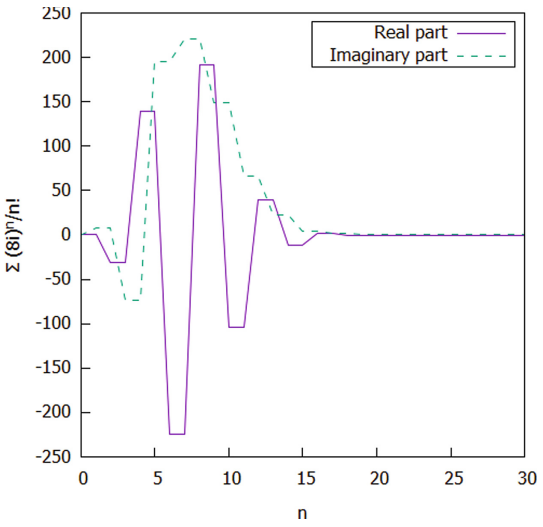

Fig. 3. The behavior of function $\sum_n \frac{(8i)^n}{n!}$ that illustrates the typical convergence of the matrix exponent elements.

In the experiments the number of distance points were set to $N = 8192$, consequently the matrices have $N^2 = 8192^2$ elements. For this size of matrices the GPUs do not show the best performance with tensor cores, that to share the workload between all GPUs we have to take the size of tiles $N_i = 2048$ or less for four V100 GPUs, and $N_i = 1024$ or less for eight A100 GPUs. This issue would go away with sufficient increase of the matrices size (e.g., in 2D

or 3D geometries), but, as we have also pointed out the memory requirements pose their limitations. A possible further development of the algorithm is to execute several multiplications in parallel. It means that the GPUs can be split into groups (of two or four) where each group would compute the different real and image parts combinations of complex multiplication. In such way the bigger tile size can be used of sharing the workload between all devices in the group. Now with the studied size of matrices, the GPUs show about 70% of utilization on four V100 GPUs and 10% of utilization on eight A100 GPUs which is not efficient enough.

The approximate average execution time for 15000 time steps ($\Delta t = 0.002$) of series with 10 terms ($k = 10$) are shown in Table 1. For the same number of GPUs, A100 works faster than V100. But, with 8 GPUs the performance of A100 falls down. This is because between 8 GPUs the data transfer mapping becomes more complicated, but computation load is still low for our multi-GPU GEMM algorithm. Then, the necessary time of data transfer increases, thus, overall time increases as well.

Table 1. Average execution times on different platforms.

Number of GPUs	4xV100	4xA100	8xA100
Time (sec)	24700	17550	44850
Time per one iteration (sec)	1.65	1.17	2.99

The execution time dependence on the matrix size for A100 and V100 GPUs are illustrated in Fig. 4. Because of the memory limitations in the case of V100 GPUs it was impossible to launch the size of $N = 65536$.

In Fig. 5 the profiles by Nsight Systems are illustrated for $N = 16384$. In scaled up Fig. 5b there is shown that the multi-GPU GEMM algorithm works in best form, but, the barriers make some intervals between the kernel calls. However, these intervals would be smaller for bigger matrix sizes.

7 Results

Figure 6 and Fig. 7 illustrate the numerical results obtained with our TDSE model for the H_2^+ ion with $Z = 30, a = 0.1$. The ground state wave function is chosen to be the initial condition for TDSE. One can notice (see Fig. 1) that there are two nearly degenerate states with the lowest energy: one is in the left Coulomb potential well and other is in the right well.

In Fig. 6 the evolution of the wave is shown for the fixed distance between the nuclei. In this case, the Hamiltonian matrix does not change in time. It is the confirmation that the initial condition is indeed the stationary solution and the corresponding probability density does not depend on time.

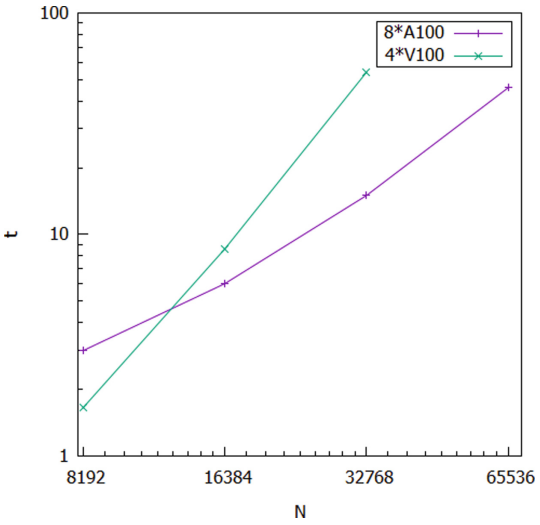

Fig. 4. The execution time (sec) per one time iteration versus the matrix size N.

Figure 7 shows the results for the model with two moving nuclei (8). In this case, the Hamiltonian matrix does change in time and our TDSE GPU-accelerated algorithm brings its benefit in high-speed recalculation of $\exp(iH\Delta t)$.

The change of $V(r,t)$ due to the nuclear motion with time displaces the wave function from the stationary state. Two effects are clearly visible. The electron probability becomes localized at both nuclei and the wave function obtains a node (i.e. the point with zero probability). It means that we observe the non-adiabatic process of electron excitation due to energy transfer from the moving nuclei to the electron subsystem.

8 Discussion

The proposed method for solving TDSE with the usage of the GPU-accelerated matrix exponent can be generalized to 2D and 3D geometries since there are no assumptions on the structure of Hamiltonian matrix. There are no assumptions on the symmetry of the model that open a way to study, for example, the collisions of a single proton on the hydrogen molecular ion. The method can be generalized for 2 electrons (with opposite spins) as well. The major challenge for such modification is the increase of memory for storing the Hamiltonian matrix. Several optimizations options are not been used at this moment: the use of the memory of all GPUs involved in the computation, the saving one half of the total memory taking into account the symmetry of the Hamiltonian matrix. Moreover, with the development of the APUs with unified memory (e.g., such as AMD MI300 or Nvidia Grace Hopper Superchip) one can expect that larger volumes of memory will be directly available for GPU-accelerated algorithms

Fig. 5. The profiles on 4 V100 GPUs. The number of distance elements is $N = 16384$ and the tile size is $N_i = 2048$. The blue bars are the computation kernels in GPUs, and the brown bars are the peer-to-peer data transfer operations between GPUs. In (a) the computation during one iteration is marked by green. By the long vertical blue line we show the wave function data supply to CPU for output operation. In (b) the scaled up fragment is shown. (Color figure online)

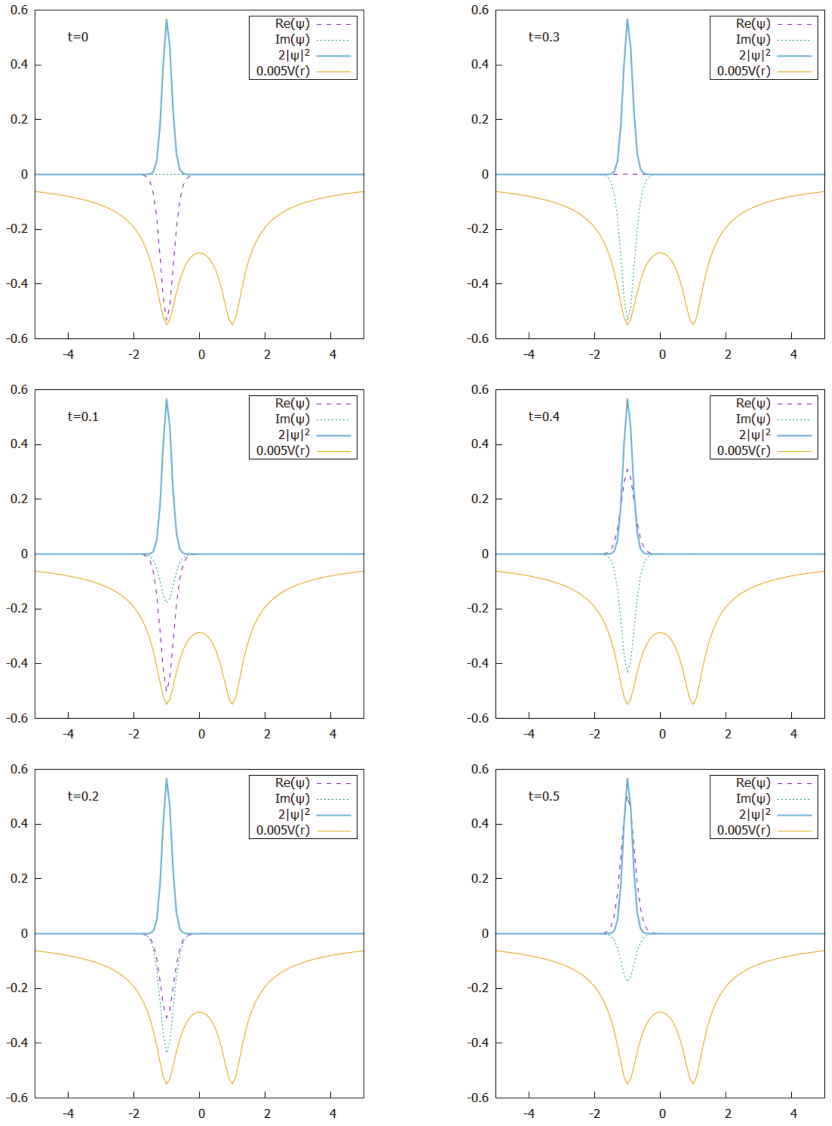

Fig. 6. The behavior of complex wave function (ψ) in double well soft-core Coulomb potential (7) (the distance between wells is 2, $Z = 30, a = 0.1$, $\Delta t = 0.002$, $\Delta r = 0.1$, $k = 10$). The time moments are 0, 0.1, 0.2 in the left block, and 0.3, 0.4, 0.5 in the right block. The results show that the probability density remains stationary in time for the initial ground state wave function.

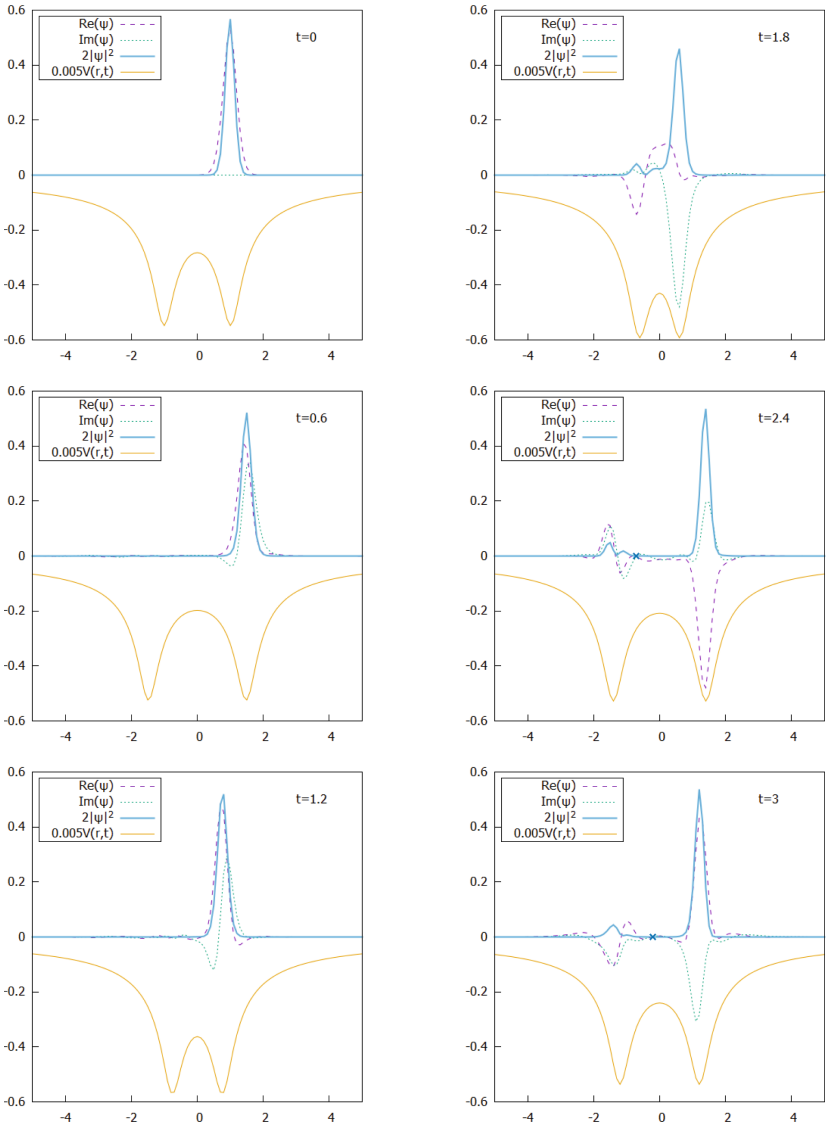

Fig. 7. The behavior of complex wave function (ψ) in the sinusoidally moving double well soft-core Coulomb potential (8) (the same parameters as in Fig. 6 with $\alpha = 0.5$ and $\beta = 3$). The cross shows the node of the wave function that appears after the excitation from the ground state.

that will make easier the deployment of the algorithms like the one considered in this work. In principle, the presented multi-GPU accelerated matrix exponent approach can be applicable not only for solving TDSE but other types of differential equations as well.

9 Conclusions

In this work, we propose a brute-force approach for solving TDSE that is based on GPU-accelerated calculation of the matrix exponent. The main motivation is that floating point operations are very "cheap" on modern GPUs and the matrix multiplication operations can be performed very quickly for large matrix sizes. We show the application of this brute-force approach for 1D hydrogen molecular ion. It is important to note that we use no assumptions on the structure of the Hamiltonian matrix and on the system symmetry that is why this approach can be generalized for 2D and 3D geometries.

The high speed recalculation of the Hamiltonian exponent $\exp(iH\Delta t)$ opens the possibility to study the processes that include the coupled electron-ion quantum dynamics. Using this high-performance computational tool we modelled (for the first time as we are aware of) the non-adiabatic (vibronic) energy transfer from moving nuclei to the single electron excitation of H_2^+.

Acknowledgments. The article was prepared within the framework of the HSE University Basic Research Program. This research was supported in part through resources of supercomputer facilities provided by HSE University. This research was supported in part through computational resources of HPC facilities at HSE University [22].

References

1. Staudte, A., et al.: Attosecond strobing of two-surface population dynamics in dissociating H_2^+. Phys. Rev. Lett. **98**(7), 073003 (2007)
2. Bello, R.Y.,et al.: Reconstruction of the time-dependent electronic wave packet arising from molecular autoionization. Sci. Adv. **4**(8), eaat3962 (2018)
3. Li, C.: Exact analytical ground state solution of 1d H_2^+ with soft coulomb potential. J. Math. Chem. **60**(1), 184–194 (2022)
4. Sullivan, D., Citrin, D.: Time-domain simulation of two electrons in a quantum dot. J. Appl. Phys. **89**(7), 3841–3846 (2001)
5. Moler, C., Van Loan, C.: Nineteen dubious ways to compute the exponential of a matrix. SIAM Rev. **20**(4), 801–836 (1978)
6. Moler, C., Van Loan, C.: Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. SIAM Rev. **45**(1), 3–49 (2003)
7. Ramadan, O.: Unified matrix-exponential FDTD formulations for modeling electrically and magnetically dispersive materials. Comput. Phys. Commun. **183**(5), 1101–1106 (2012)
8. Lugovskoy, A., Bray, I.: Almost sudden perturbation of a quantum system with ultrashort electric pulses. Phys. Rev. A **77**(2), 023420 (2008)

9. Yu, C., Madsen, L.B.: Sequential and nonsequential double ionization of helium by intense xuv laser pulses: revealing ac stark shifts from joint energy spectra. Phys. Rev. A **94**(5), 053424 (2016)
10. Yu, C., Madsen, L.B.: Above-threshold ionization of helium in the long-wavelength regime: examining the single-active-electron approximation and the two-electron strong-field approximation. Phys. Rev. A **95**(6), 063407 (2017)
11. Paramonov, G.K., Klamroth, T., Lu, H., Bandrauk, A.D.: Quantum dynamics, isotope effects, and power spectra of H_2^+ and HD^+ excited to the continuum by strong one-cycle laser pulses: Three-dimensional non-born-oppenheimer simulations. Phys. Rev. A **98**(6), 063431 (2018)
12. Majorosi, S., Benedict, M.G., Bogár, F., Paragi, G., Czirják, A.: Density-based one-dimensional model potentials for strong-field simulations in he, h 2+, and h 2. Phys. Rev. A **101**(2), 023405 (2020)
13. Truong, T.D., et al.: Soft parameters in coulomb potential of noble atoms for non-sequential double ionization: classical ensemble model and simulations. Comput. Phys. Commun. **276**, 108372 (2022)
14. Bandrauk, A.D., Shen, H.: Exponential split operator methods for solving coupled time-dependent schrödinger equations. J. Chem. Phys. **99**(2), 1185–1193 (1993)
15. Viklund, L., Augustsson, L., Melander, J.: Numerical approaches to solving the time-dependent schrödinger equation with different potentials (2016)
16. Choi, Y.R., Nikolskiy, V., Stegailov, V.: Matrix-matrix multiplication using multiple GPUs connected by NVLink. In: 2020 Global Smart Industry Conference (GloSIC), pp. 354–361. IEEE (2020)
17. Anthonisse, J.M., Tijms, H.: Exponential convergence of products of stochastic matrices. J. Math. Anal. Appl. **59**(2), 360–364 (1977)
18. Alonso, P., Ibáñez, J., Sastre, J., Peinado, J., Defez, E.: Efficient and accurate algorithms for computing matrix trigonometric functions. J. Comput. Appl. Math. **309**, 325–332 (2017)
19. Choi, Y.R., Nikolskiy, V., Stegailov, V.: Tuning of a matrix-matrix multiplication algorithm for several GPUS connected by fast communication links. In: Sokolinsky, L., Zymbler, M. (eds.) Parallel Computational Technologies: 16th International Conference, PCT 2022, Dubna, Russia, March 29–31, 2022, Revised Selected Papers, pp. 158–171. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-11623-0_12
20. Choi, Y.R., Stegailov, V.: Multi-GPU GEMM algorithm performance analysis for Nvidia and AMD GPUs connected by NVLink and PCIe. In: Balandin, D., Barkalov, K., Meyerov, I. (eds.) Mathematical Modeling and Supercomputer Technologies: 22nd International Conference, MMST 2022, Nizhny Novgorod, Russia, November 14–17, 2022, Revised Selected Papers, pp. 281–292. Springer Nature Switzerland, Cham (2022). https://doi.org/10.1007/978-3-031-24145-1_23
21. Kondratyuk, N., et al.: Performance and scalability of materials science and machine learning codes on the state-of-art hybrid supercomputer architecture. In: Voevodin, V., Sobolev, S. (eds.) Supercomputing: 5th Russian Supercomputing Days, RuSCDays 2019, Moscow, Russia, September 23–24, 2019, Revised Selected Papers, pp. 597–609. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-36592-9_49
22. Kostenetskiy, P.S., Chulkevich, R.A., Kozyrev, V.I.: HPC resources of the higher school of economics. J. Phys.: Conf. Series **1740**, 012050 (Jan 2021). https://doi.org/10.1088/1742-6596/1740/1/012050