An Adaptive Time-Step Method for Cardiac Cell Simulation based on Multi-GPU

Haiyi Ye, Ching-Hsing Luo*, Xinji Chen

Sun Yat-sen University, School of Data and Computer Science, Guangzhou, China

luojinx5@mail.sysu.edu.cn

*Corresponding author

Keywords: Computer Simulation, Ventricular Cell, Adaptive Time-step Method, Graphics Processing Unit, High Performance Computing

Abstract. Cardiac electrophysiological simulation is a very complex computational process. Running on Graphics processing unit (GPU) is an effective method for cardiac electrophysiological simulation. In addition, the use of adaptive time-step can also effectively speed up the simulation of heart cells. However, the previous works running on GPU could not get apparent speedup (only 1.5 times). This paper implements adaptive time-step methods on multi-GPU to run Luo-Rudy dynamic (LRd) ventricular action potential model incorporating a Markov Sodium Channel model. For adaptive time-step methods, we use Traditional Hybrid Method(THM) and Chen-Chen-Luo's quadratic adaptive algorithm (CCL). As LRd is solved by THM or CCL in a single GPU, the acceleration is 17.5 times(17.5x) and 43x respectively compared with the fixed time-step under Mix Root Mean Square error lower than 5%. In 2 GPUs, the acceleration of THM and CCL is 9.7x and 33.7x separately. As there are 4 GPUs, the acceleration of THM is only 5.7 x while acceleration is 25.6x for CCL. In addition, compared with the fixed time-step in CPU, THM and CCL accelerated to 1861x and 5054x in a single GPU, 1885x and 6652x in 2 GPUs, 2093x and 8024x in 4 GPUs. In conclusion, CCL is much better than THM to save computation cost on multi-GPU, while the speedup is not lineally increased with the number of GPU.

1. Introduction

The heart is one of the important organs of human beings, and cardiovascular disease is one of the three major diseases of human beings, so it is of great significance for the study of the heart. Over recent decades, a variety of cardiac virtual models have been used to explore the causes of various diseases of the human heart. Cardiac electrophysiological processes are of extreme complication, and computation models have become valuable tools for studying and understanding such complex phenomena because they allow lots of information from different physical scales and experiments to be combined to achieve better simulation of the entire system function. This is the conversion of highly complex biophysical processes into complex mathematical and computational models. The modern heart model is described by a nonlinear system coupled to a Partial Differential Equation (PDE) of Nonlinear Ordinary Differential Equations (ODE).

Since the whole cardiac model includes billions of cells containing multiple state variables per cell, it takes a lot of computational resources to simulate a whole heart. This requires researchers to focus on how to speed up the simulation process. Most of the current acceleration methods are using GPU or using the adaptive time-step method.

GPU is an effective acceleration tool, giving up-to 200x for a single GPU applied on cardiac electrophysiological simulation [1-6]. The acceleration was farther improved in the multi-GPU. However, it is still impossible to complete the simulation within an acceptable time frame (hours).

Papers [1-6] has not used the adaptive time-step method on the GPU, while the adaptive time-step method can effectively accelerate cardiac electrophysiology simulation.

At present, some researchers used simple adaptive algorithms to accelerate in the GPU. Victor M. Garcia[7] and Rafael Sachetto Oliveira[8] combined GPU and adaptive time-step method in order to speed up the simulation of cardiac electrophysiology, but it was only 1.5x faster than the fixed time-step in GPU.

This paper accelerates cardiac cell simulation using CCL and THM in multi-GPU to check if the apparent speedup is realized or not.

2. Methods

2.1 Cardiac Ventricular Cell Model

The form of reaction diffusion equation describing cardiac electrophysiological model is as follows:

$$\nabla\left(\sigma_i \bullet \nabla V_i\right) = A_m(C_m \frac{\partial V_m}{\partial t} + I_{ion} + I_{stim}) \tag{1}$$

Where ∇ is gradient operator, σ_i is intracellular conductivity, V_i is intracellular potential, A_m is the ratio of cell surface area to volume, C_m is the capacitance of unit membrane area, V_m is cross-membrane potential, I_{ion} is the sum of transmembrane ion Current, I_{stim} is the stimulation current, t represents time. The goal of the equation is to solve the V_m, so equation (1) can be converted into:

$$\frac{\partial V_m}{\partial t} = -\frac{I_{ion} + I_{stim}}{C_m} + \frac{\sigma_i}{A_m C_m}\nabla^2 V_m \tag{2}$$

LetD_diff= σ _i/(A_m C_m)，we have：

$$\frac{\partial V_m}{\partial t} = -\frac{I_{ion} + I_{stim}}{C_m} + D_{diff}\nabla^2 V_m \tag{3}$$

Where D_diff is the diffusion coefficient. $-\frac{I_{ion} + I_{stim}}{C_m}$ is reaction term and $D_{diff}\nabla^2 V_m$ is diffusion term.

The reaction term is incorporated into the system of ODEs and the diffusion term is incorporated into the system of PDEs. To solve the involved system of ODEs, we have adopted Rush-Larsen method [9] for gating variable and Forward-Euler. Finite differences and Forward-Euler temporal discretization are applied to the purely diffusive PDE raised from splitting.

2.2 LRd model + Markov Sodium Channel Model

LRd [10] was a dynamic ventricular action potential model proposed by Luo and Rudy in 1994. Many ion channels were included in LRd model with the absorption and release of calcium ions in the sarcoplasmic reticulum (SR). This was the first time that SR was added to a cardiac cell model. The main function of SR in cardiac cells is to regulate the concentration of ions in the cytoplasm. LRd model was created for the mammalian ventricular action potential is based mostly on the guinea pig ventricular cell. The following processes are formulated: Ca^{2+} current through the L-type channel (I_{Ca}), the Na^+-Ca^{2+} exchanger, Ca^{2+} release and uptake by the SR, buffering of Ca^{2+} in the SR and in the myoplasm, a Ca^{2+} pump in the sarcolemma, the Na^+-K^+ pump, and a nonspecific Ca^{2+}-activated membrane current. Activation of I_{Ca} is an order of magnitude faster than in previous models. Inactivation of I_{Ca} depends on both the membrane voltage and $[Ca^{2+}]_i$. SR is divided into two subcompartments, a network SR (NSR) and a junctional SR (JSR). Functionally, Ca^{2+} enters NSR and translocates to JSR following a monoexponential function. Release of Ca^{2+} occurs at JSR and can be triggered by two different mechanisms, Ca^{2+}-induced Ca^{2+} release and spontaneous release. The model provides the basis for the study of arrhythmogenic activity of the single myocyte including afterdepolarizations and triggered activity. It can simulate cellular responses under different degrees of Ca^{2+} overload.

In order to study the performance of LRd model with Markovian ion channel[11] mentioned as LRdm, I_{Na} channel of Huxley-Hogkin model in LRd is replaced with the wild-type Markov models which consists of one conducting open state (O), three closed states (C1, C2, C3), two closed-inactivation states (IC3, IC2), one fast inactivation state (IF), and two intermediate inactivation states (IM1, IM2).

2.3 Time Adaptive Methods

The adaptive time-step method can have an effective acceleration for cardiac cell simulation. The basic idea is to select different time-step according to the inconsistent voltage change rate. As the voltage rate changes rapidly, in order to maintain the stability of the cell, it is necessary to give the cell a small time-step. On the contrary, As the voltage change rate is slow, a large time-step can be set to speed up the simulation.

2.3.1 Traditional Hybrid Method

Traditional Hybrid Method (THM) was proposed by Victorri in 1985 [12]. It computes time-step through the deviant of voltage, which limit in $\Delta V_{min} \leq \Delta V \leq \Delta V_{max}$. The change rate of voltage dV/dt determines Δt. As $dV/dt < \Delta V_{min}$, the time-step becomes larger. As $dV/dt > \Delta V_{max}$, the time-step becomes smaller. A range of time-step should be set in order to avoid too small or too large time-step for high computation cost or instability.

If $\Delta V < \Delta V_{min}$, then: $\Delta t = \Delta V_{max}/(dV/dt)$
If $\Delta V > \Delta V_{max}$, then: $\Delta t = \Delta V_{min}/(dV/dt)$
If $\Delta V_{min} \leq \Delta V \leq \Delta V_{max}$, then: $\Delta t = \Delta V$

THM can effectively speed up the heart cell simulation, but it has shortage at local maximum and minimum zones. As small ΔV creates unwanted large time-step, Chen-Chen-Luo's quadratic adaptive algorithm(CCL) [13] was proposed to solve this problem.

2.3.2 CCL Method

CCL is an effective time-step method proposed by Chen et al [13]. to improve traditional Hybrid method. The main idea of CCL is to take d^2V/dt^2 into account, and use d^2V/dt^2 to deal with local extremum problems.

Consider the second order Taylor expansion of $V(t_{n+1})$:

$$V(t_{n+1}) \cong V(t_n) + V'(t_n)(t_{n+1} - t_n) + \frac{1}{2}V''(t_n)(t_{n+1} - t_n)^2 \tag{4}$$

Where $V'(t_n) = \frac{V(t_{n+1}) - V(t_n)}{t_{n+1} - t_n}$, $V''(t_n) = \frac{V'(t_{n+1}) - V'(t_n)}{(t_{n+1} - t_n)}$.

Get:

$$\Delta V \cong \frac{dV}{dt} * \Delta t + \frac{1}{2}\frac{d^2V}{dt^2} * \Delta t^2 \tag{5}$$

solve the second-order equation:

$$\Delta t = \frac{-dV/dt \pm \sqrt{(dV/dt)^2 + 2*d^2V/dt^2*\Delta V}}{d^2V/dt^2} \tag{6}$$

At $\sqrt{(dV/dt)^2 + 2 * d^2V/dt^2 * \Delta V} \geq 0$, formula (6) can be solved. If $(dV/dt)^2 + 2 * d^2V/dt^2 * \Delta V < 0$, formula (6) can't be solved, so Δt is computed by the follow formula:

$$\Delta t = -\frac{dV/dt}{d^2V/dt^2} \tag{7}$$

Meantime, in order to ensure voltage stable, Δt can't be changed rapidly. So $\Delta t$is limited at 2x faster or slower than last time-step. It means that Δt can't be higher 2x or lower 2x than the last time-step:

$$1/2 * \Delta t_n < \Delta t_{n+1} < 2 * \Delta t_n$$

Also Δt should be in the range of $\Delta t_{min} \leq \Delta t_n \leq \Delta t_{max}$.

2.4 GPU implementation

2.4.1 single GPU

In GPU, we arrange large matrices (potentials, currents, gated variables, etc.) in rows and organize them into one-dimensional arrays. Because large matrices are much larger than the amount of on-chip memory, they can only be loaded into off-chip global memory. For constant data (temperature, conductivity, cell volume, etc.) that are shared by all cells, place it in constant memory. Constant memory, although off-chip, is an efficient use of memory bandwidth because of its combined access (a single read is available to all threads that need this data). For small-scale data, we put it into shared memory to make efficient use of bandwidth. For the boundary, we consider the third boundary condition: extending the outer cell by another layer.

The pseudo-code for the process is shown in Figure 1A.

```
Initialization data;
t = 0;        // record running time
while (t < simulated time){
Boundary treatment;
Compute ODE for  Δt;
Compute PDE for  Δt;
if (t % output-time){
Output potential;
t = t +  Δt;
}
Output t;
```

A

```
Initialization data;
t = 0;        // record running time
while (t < simulated time){
Boundary treatment;
Compute ODE for  Δt;
Data exchange
Compute PDE for  Δt;
if (t % output-time){
Output potential;
t = t +  Δt;
}
Output t;
```

B

Figure 1. Pseudo-code. A is the pseudo-code of single GPU and B is the pseudo-code of multi-GPUs

2.4.2 multi-GPU

In multi-GPUs, we first divide the data into different GPUs on average. Data exchange before PDE computation on different GPUs (Figure 1B). In data exchange, we define transition zone and fill area. The transition zone stores the data that needs to be transferred to other GPUs, and the fill area stores the data that needs to be transferred from other GPUs. Figure 2 shows the data exchange process, where the data of the transition zone is put into the fill area.

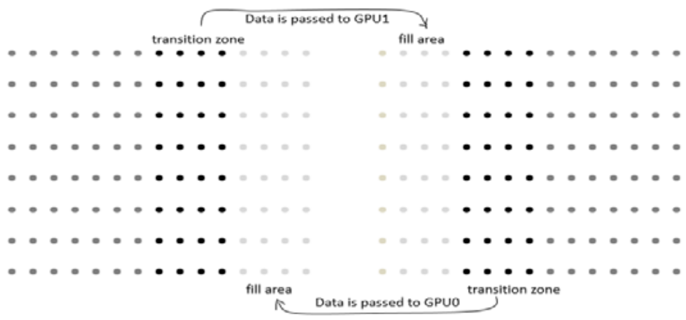

Figure 2. Data exchange. Light gray indicates the fill area and black indicates the transition zone

2.5 Mix Root Mean Square error

The MRMSe is proposed by Kevin R [14] and defines by:

$$[\mathrm{MEMSe}] = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{\widehat{w}_i - w_i}{1 + |\widehat{w}_i|}\right)^2}$$

Where w_i and $\hat{w}_i$ denote the numerical and the reference solutions of scalar quantity w at time point i.

2.6 Experimental conditions

The CPU we used was Intel core i7 7700, with main frequency 3.6 GHz, and 16 GB host memory. The GPU used is NVIDIA GeForce GTX 1080 TI with a frequency of 1582 MHz and video memory of 12 GB. In addition, the simulation time was 600 ms, and the stimulation started at 10 ms and lasted 0.5 ms. The simulation was carried out with fixed time-step, THM and CCL respectively.

3. Results

LRdm is run at the fixed time-step as reference solution in comparison to THM and CCL for computation cost at CPU and multi-GPU.

3.1 Fixed time-step method

Table 1. Computation time in a fixed time-step. For 2000*2000, CPU cannot be run due to insufficient memory.

model	grid	dt(ms)	time(s)			
			CPU	1 GPU	2 GPU	4 GPU
LRdm	1000*1000	0.001	1023954.0	6710.3	3841.4	2138.6
	1500*1500	0.001	2062320.1	14984.5	8344.1	4768.0
	2000*2000	0.001	/	27519.0	15188.3	8564.8

Using GPU to perform cardiac electrophysiological simulation is an effective acceleration method. Table 1 shows the computation time in a fixed time-step and apparently GPU has higher acceleration efficiency than CPU. It has 150x speedup using GPU in comparison to CPU in LRdm. With the higher the speedup and the larger the number of GPU, the acceleration reaches up to 270x for 2 GPUs and up to 450x for 4 GPUs.

3.2 THM acceleration

Table 2. Computation time with THM method

model	grid	dt(ms)	time(s)			
			CPU	1 GPU	2 GPU	4 GPU
LRdm	1000*1000	0.001-0.1	49185.7	695.1	681.5	650.7
	1500*1500	0.001-0.1	94506.9	1108.7	1094.0	985.4
	2000*2000	0.001-0.1	139386.1	1571.8	1566.8	1500.4

In Table 2 THM in CPU has 20x acceleration compared to the fixed time-step. In a single GPU, its acceleration reaches 17.5x than the fixed time-step. However, it is found that the acceleration decreases as the number of GPU increases, in which acceleration is down to 9.7x in 2 GPUs, and further down to 5.7x in 4 GPUs.

3.3 CCL acceleration

Table 3. Computation time with CCL method

model	grid	dt(ms)	time(s)			
			CPU	1 GPU	2 GPU	4 GPU
LRdm	1000*1000	0.001-0.1	11453.5	228.1	177.1	141.7
	1500*1500	0.001-0.1	27029.4	408.3	310.7	257.0
	2000*2000	0.001-0.1	43325.8	635.9	450.3	334.5

In Table 3, CCL has 90x speedup than the fixed time-step method in CPU, and there is 40x speedup in single GPU, 30x in 2 GPUs, and only 20x in 4 GPUs. Similar to THM, the acceleration of CCL decreases as the number of GPU increases. However, CCL can still accelerate to over 20x in 4 GPUs while it is only 5.7x for THM.

3.4 Acceleration ratio for THM and CCL

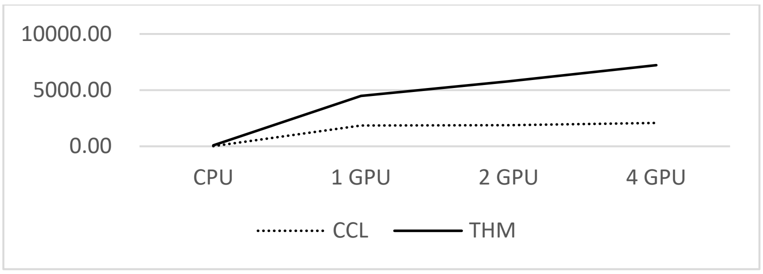

Figure 3. Acceleration ratio for THM and CCL

Figure 3 shows the acceleration of CCL and THM compared to the fixed time-step in CPU and multi-GPU. Obviously, CCL is better than THM. As the number of GPUs increases, THM acceleration increases hardly, while CCL acceleration increases apparently. In 4 GPUs, CCL accelerates to 7000x while THM only increases to 2000x. It indicates that CCL has better acceleration than THM.

According to section 2.3, the computation of CCL is obviously more complicated than THM, but it is true that CCL is faster than THM in simulation. In fact, CCL has a larger granularity for time-step partitioning than THM and therefore it has a greater acceleration. CCL maintained a large time-step before and after stimulation as THM maintained a relatively small time-step. Both CCL and THM maintained small time-step during stimulation. At the local extreme point, CCL maintains a small time-step less than THM. Thus CCL has faster acceleration than THM.

3.5 Mix Root Mean Square error

Table 4. MRMSe, the benchmark is made with a fixed time-step of 0.0001ms

model	method	dt(ms)	MRMSe		
			1 GPU	2 GPU	4 GPU
LRdm	fix time	0.001	0.5%	0.5%	0.5%
	CCL	0.001-0.1	1.9%	1.9%	1.9%
	THM	0.001-0.1	3.0%	3.0%	3.0%

In Table 4, the MRMSe of the fixed time-step (0.001ms) is only 0.5%, it is 1.91% for CCL, and it is 3.06% for THM. It is obvious that the accuracy of CCL is better than that of THM.

4. Conclusion

Neither GPU nor adaptive time-step methods alone can achieve good acceleration. This study combines GPU with CCL or THM to achieve thousands of times of acceleration in a single GPU. In the previous study, only 498x acceleration was achieved by using the adaptive time-step method different from CCL and THM in a single GPU [8]. It shows that THM and CCL have good acceleration in GPU and CCL has better acceleration than THM. In a single GPU, CCL is 2.4x faster than THM, and it is 3.5x or 4.5x faster in 2 or 4 GPUs separately. The MRMSe of CCL was only 1.9% in comparison to 3.0% of THM. Therefore, CCL is better than THM in terms of acceleration and accuracy. However, in multi-GPU as the number of GPUs increases to 4, CCL only accelerates by 1.6x while THM hardly accelerates at all. CCL acceleration increases from 4000x in a single GPU to 7000x in 4 GPUs while it increases from 1800x to 2000x only for THM. In future researches, it should be paid attention to the linear acceleration of adaptive time methods in multi-GPUs.

References

[1]B.M.Rocha, F.O.Campos, R.M.Amorim et al. Accelerating cardiac excitation spread simulations using graphics processing units. Concurrency and Computation-Practice & Experience 2011(23) 708-720.

[2]Guillermo Vigueras, Ishani Roy, Andrew Cookson et al. Toward GPGPU accelerated human electromechanical cardiac simulations. International Journal for Numerical Methods in Biomedical Engineering. 2014(30) 117-134.

[3]Wei Wang, Lifan Xu, John Cavazos et al. Fast Acceleration of 2D Wave Propagation Simulations Using Modern Computational Accelerators. PLoS ONE. 2014(9) e86484.

[4]Jun Chai, Mei Wen, Nan Wu et al. simulating cardiac electrophysiology in the era of GPU-cluster computing. IEICE Transactions on Information and Systems. 2013(12) 2587-2595.

[5]Aurel Neic, Manfred Liebmann, Elena Hoetzl et al. Accelerating Cardiac Bidomain Simulations Using Graphics Processing Units. IEEE Trans Biomed Eng. 2012(59) 2281-2290.

[6]Yong Xia, Kuanquan Wang, Henggui Zhang. Parallel Optimization of 3D Cardiac Electrophysiological Model Using GPU. Computational and Mathematical Methods in Medicine. 2015 862735.

[7]Victor M. Garcia, A. Liberos, A.M. Climent et al. An Adaptive Step Size GPU ODE Solver for Simulating the Electric Cardiac Activity. Computing in Cardiology. 2011

[8]Rafael Sachetto Oliveira, Bernardo Martins Rocha, Denise Burgarelli et al. Performance evaluation of GPU parallelization, space-time adaptive algorithms, and their combination for simulating cardiac electrophysiology. International Journal for Numerical Methods in Biomedical Engineering. 2018(34) e2913.

[9]Stanley Rush, Hugh Larsen. A Practical Algorithm for Solving Dynamic Membrane Equations. IEEE Trans Biomed Eng. 1978(25) 389-392.

[10] CH Luo, Y Rudy. A Dynamic Model of the Cardiac Ventricular Action Potential II Afterdepolarizations, Triggered Activity, and Potentiation. Circulation Research. 1994(74) 1097–1113.

[11] Xing-Ji Chen, Ching-Hsing Luo, Min-Hung Chen et al. Combination of “quadratic adaptive algorithm” and “hybrid operator splitting” or uniformization algorithms for stability against acceleration in the Markov model of sodium ion channels in the ventricular cell model. Medical & Biological Engineering & Computing. 2019(57) 1367-1379.

[12] Bernard Victorri, Alain Vinet, Fernand A et al. Numerical Integration in the Reconstruction of Cardiac Action Potentials Using Hodgkin-Huxley-Type Models. Computers and Biomedical Research. 1985(18) 10-23.

[13] Min-Hung Chen, Po-Yuan Chen, Ching-Hsing Luo. Quadratic adaptive algorithm for solving cardiac action potential models. Computers in Biology and Medicine. 2016(77) 261-273.

[14] Kevin R. Green, Raymond J. Spiteri. Gating-enhanced IMEX splitting methods for cardiac monodomain simulation. Numerical Algorithms. 2019.