GPU Load Balancing Using Sparse Cartesian Grids: Making Interactive WebGL Simulations of Complex Ionic Models Even Faster on 3D Heart Structures

Abouzar Kaboudian[1], Elizabeth M Cherry[2], Flavio H Fenton[3]

[1] Division of Biomedical Physics, US Food & Drug Administration, USA
[2] School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA, USA
[3] School of Physics, Georgia Institute of Technology, Atlanta, GA, USA

Abstract

Cardiac modeling on supercomputers has limited arrhythmia studies to groups with specialized access and expertise. We previously showed that WebGL programs can simulate complex ionic models in 2D and 3D cardiac geometries in real time without the need for a supercomputer by utilizing parallel hardware through the graphics processing unit (GPU). In this work, we use sparse Cartesian grids to balance GPU load, conserve memory, and avoid unnecessary read/write operations, speeding up 3D simulations by up to a factor of 20. We also present a simple mapping technique to compress sparse data structures into compact structures, which allows us to access texture memory efficiently during the time-stepping portion of the computation. As examples, we present the implementation of phenomenological models for 3D atrial and ventricular simulations, as well as the 41-state-variable OVVR human ventricular cell model on 3D ventricular human anatomical structures. We show how our programs can be used to initiate and terminate scroll waves in 3D interactively.

1. Introduction

Cardiac arrhythmias are a major cause of death and disability worldwide [1]. Atrial fibrillation (AF) and ventricular fibrillation (VF) are two of the most common types of cardiac arrhythmias [1]. VF can usually be treated with an implantable cardioverter defibrillator (ICD) in at-risk patients. However, there is no widely effective and long-lasting treatment for atrial fibrillation (AF). Radiofrequency catheter ablation can interrupt abnormal and repetitive electrical activity in the heart, but it often requires follow-up treatments. Ablation is effective for about 60% of people with paroxysmal AF, but less than 30% of people with persistent AF, and it is also not advisable for all patients [2].

Numerical simulations of cardiac dynamics are becoming increasingly useful for developing patient-specific treatments for AF, ventricular tachycardia (VT), and VF [3, 4]. These simulations have been facilitated by the continuous growth in computer power and new methods allowing simulations of the highly computationally demanding problem of solving the voltage dynamics in cardiac tissue.

Over the years, multiple groups have developed software to perform numerical simulations of cardiac arrhythmias using complex models in 2D and 3D structures, such as CARP [5], Chaste [6], and Beatbox [7]. While these programs are robust, they all require downloading and installation, as well as access to parallel supercomputers and multiple CPUs. Less complex but interactive codes have also been developed for simulations ranging from single-cell dynamics, such as Labheart [8] for ion channels and calcium transport in rabbit ventricular cells, and Myokit [9] for a variety of cells.

Complex software can simulate realistic cardiac cell models in anatomically accurate structures, but it requires supercomputers and specialized knowledge. It also lacks interactivity, so results are only available after the simulation has finished. In contrast, most available interactive software can visualize the simulation as it occurs, but the models used are simpler and less detailed. However, recent advances in graphics processing unit (GPU) computing using WebGL have enabled the development of fast, interactive, easy-to-use, and robust computational tools that are independent of the operating system and the device [10–12]. Modern GPUs are equipped with thousands of powerful computational cores at affordable prices, sometimes as low as a few hundred US dollars. Moreover, GPUs enable computational acceleration on personal computers and even cell phones, rather than requiring remote computer clusters [10, 11].

To achieve optimal GPU performance, the computational load on the GPU cores must be balanced [13]. However, when a Cartesian grid is used to represent the heart geometry, only a fraction of the grid points represent the tissue. In our analysis of a virtual cohort of 24 patients [14], ventricular and atrial structures occupied $3.42 \pm 0.72\%$ and $10.39 \pm 2.81\%$ of the grid, respectively. This sparse grid complicates load balancing and workload distribution on the GPU. Moreover, the empty space may increase the memory requirements for a GPU-accelerated code.

Here, we present a method to compress the sparse grid structure to create a balanced GPU code, thereby ensuring that the memory requirements are associated only with the tissue points, and all GPU cores are always utilized to update the PDE on the tissue points. The resulting programs are up to 20 times faster than when the grid is not compressed.

2. Methods

We solve a simplified minimal atrial ionic model [15] and a complex human ventricular model [16] in 3D human atrial and ventricular structures from the online database [14], respectively, using uniform Cartesian grids with the same discretization of 0.026 cm. The grids were converted from VTK to JSON using our in-house conversion toolkit. The `WebGL` numerical implementations rely on the `Abubu.js` library [10, 11], which is available along with training resources for implementing `WebGL` programs using it at https://www.abubujs.org/

Cable equation. The simulations are carried out by solving the cable equation [17]:

$$\partial_t V = D\nabla^2 V - \sum I_i / C_m, \tag{1}$$

where V represents the membrane potential, D is the diffusion coefficient, I_{ion} are ionic currents described by either a minimal model [15] or the 41-variable OVVR model [16], and C_m is the membrane capacitance.

Using 2D textures to depict 3D grids. Texture memory is a GPU data structure that can store physiological information, such as state variables (e.g., membrane potential or ionic concentrations), on each grid point. It consists of pixels with four color channels: red, green, blue, and alpha. Each channel can represent a physical variable. Textures can serve as input or output for shader programs [11].

To represent a 3D grid as a 2D texture, we generate an array of sub-images by slicing the 3D grid in the third dimension and placing the sub-images on a 2D grid of sub-images stored in the 2D texture. A simple utility function can retrieve the data corresponding to each point in the 3D space from the 2D texture.

To differentiate tissue points from empty spaces on the 3D grid, we store binary values of zero and one on a texture representing the anatomical structure. This domain representation results in a sparse data structure, with many pixels corresponding to non-tissue regions. Compression can alleviate this storage inefficiency.

Compressing and uncompressing 2D textures For compression, tissue pixels are stored in the smallest square texture that can fit them. If the original (uncompressed) texture consists of $W \times H$ pixels and the compressed texture consists of $w \times w$ pixels, two auxiliary integer textures with pixel sizes of $W \times H$ (compressed-index-texture) and $w \times w$ (full-index-texture) are used to store the mappings between the compressed and uncompressed textures.

An index $\tilde{I}$ of a pixel on the original texture is expressed as

$$\tilde{I} \in \{(i, j) | i, j \in Z, 0 \leq i \leq W, 0 \leq j \leq H\}, \tag{2}$$

and an index $\tilde{i}$ on the compressed texture is expressed as

$$\tilde{i} \in \{(i, j) | i, j \in Z, 0 \leq i, j \leq w\}, \tag{3}$$

where Z represents the set of integers.

Pixels in the compressed data structure are assigned by traversing the pixels of the uncompressed data structure in order, noting the index ($\tilde{I}$) of each tissue point and assigning the next available pixel in the compressed texture, with index ($\tilde{i}$). Index $\tilde{I}$ of the compressed-index-texture stores the value of $\tilde{i}$, and index $\tilde{i}$ of the full-index-texture stores the value of $\tilde{I}$.

For each pixel on the uncompressed data structure that does not represent tissue, index $\tilde{I}$ of the compressed-index-texture stores the value of $\tilde{i} = (w, w)$ so that all non-tissue grid points map to the last pixel of the compressed data structure. Index $\tilde{i} = (w, w)$ of the full-index-texture stores the sentinel value $(-1, -1)$ to indicate that this point does not map to tissue.

Using these mappings, all physiological information can be stored in compressed textures of size $w \times w$ pixels, and mappings in both directions are available for all tissue points, with sensible handling of non-tissue points.

Applying zero-flux boundary conditions. Zero-flux boundary conditions can be applied using the mirroring technique, which uses ghost nodes [18]. When using the central-differencing scheme, the required neighboring cell of the stencil is used as is if it is available. However, if the required neighbor in the computational cell is outside the tissue, the neighbor's value is mirrored with respect to the center of the computation cell. This approach is equivalent to setting the first derivative of the membrane potential equal to zero and applying a central differences scheme around the central node along the normal to the tissue interface.

Visualizing the simulation results. The simulation results of the electrical excitations on the 3D tissue are visualized using the built-in methods in the `Abubu.js` library. These methods allow for transparency and cut planes to inspect the excitations deep within the tissue. All visualization and simulation occur concurrently.

Interactivity. Combining `WebGL` with `Abubu.js` allows JavaScript event handling and interaction with the WebGL component. User interactions can be performed through the GUI or mouse and touch interactions. The GUI allows setting model and visualization parameters; activating the code editor; and starting, stopping, and initializing simulations. Interactions can occur during simulations to see the effect of changing parameters.

3. Numerical Results

We converted the atrial and ventricular sections of the first human subject from the publicly available online cohort of 24 patients [14]. We used our in-house conversion tool, Carp2Cartesian, to convert the tetrahedral finite element mesh in VTK format to a uniform Cartesian grid. This conversion kit is available for use at https://github.com/dbp-osel/Carp2Cartesian. The simulations were generated using the WGLPackedCartesian software package and can be downloaded from here: https://github.com/dbp-osel/WGLPackedCartesian. Alternatively, the programs can run in the browser without installation here: https://dbp-osel.github.io/WGLPackedCartesian.

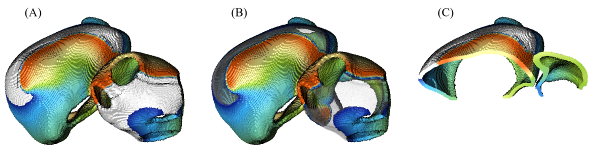

Figure 1. Three views of an instance of functional reentry on a human atrial structure: (A) View of the scroll wave on opaque atria, (B) view of the scroll wave on transparent atria, (C) cross-sectional view of the atrial structure.

Figure 1 shows three views of functional reentry on a human atrial structure. We used the P1 fitting of the minimal model [15] as the cell model for this example. We initiated the functional reentry by interactively creating an activation wave using the computer mouse. Then, we created another activation in the refractory region of the first excitation wave. The non-empty space of the atrial structure for this subject was 3.24% of the cubical bounding box. The GPU memory requirement for this simulation was less than 4% of the uncompressed version of the program. More importantly, the simulations were more than 20 times faster than their uncompressed counterparts, which allowed for the real-time calculation of the reentry on this structure.

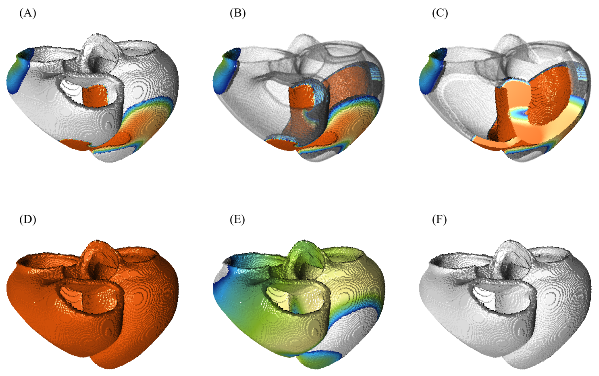

Figure 2. Simulation and visualization of tachycardia using the OVVR model. (A-C): three views of the same moment during tachycardia. (D-F): termination of the scroll wave by a single far-field stimulus over 400 ms

Figure 2(A-C) show three views of the same instant during an episode of tachycardia, including the scroll wave on (A) opaque and (B) transparent ventricles and (C) cross-sectional view of the ventricular structure with two scroll waves. Note that the scroll wave may not appear in view (A); however, it can be clearly seen on the septum in the other views at the same time. Panels (D) to (F) show termination of the scroll wave with the application of a single far-field stimulus: (D) the instant when the large far-field stimulus is applied, (E) 150 ms after the stimulus is delivered, and (F) termination of the scroll-waves 400 ms after the delivery of the stimulus. For this example, we used the 41-variable OVVR model [16] as the cell model. Reentry was initiated interactively, similar to the previous example. The non-empty space of this ventricular structure was 11.7% of the cubical bounding box. The GPU requirement for this simulation using compression was less than 13% of, and the speed was more than 7 times faster than, the uncompressed version of the program.

4. Conclusion

In this article, we have shown that a compression algorithm can significantly speed up simulations of electrical activity on realistic anatomical structures. Previous advances in GPU and WebGL computing have already made such simulations possible on personal computers and cell phones, representing a significant improvement over traditional parallel processing. The compression algorithm presented here speeds up simulations by up to a factor of 20, which is an order-of-magnitude improvement. This technique allows the use of higher-resolution grids and significantly conserves memory.

Disclaimer

The mention of commercial products, their sources, or their use in connection with material reported herein is not to be construed as either an actual or implied endorsement of such products by the Department of Health and Human Services.

The findings and conclusions in this article have not been formally disseminated by the Food and Drug Administration and should not be construed to represent any Agency determination or policy.

This article reflects the views of the authors and should not be construed to represent FDA's views or policies.

Acknowledgments

We acknowledge support from the Division of Biomedical Physics, Office of Science and Engineering Laboratories, Center for Devices and Radiological Health, US Food and Drug Administration. We also acknowledge support for this study from NSF grant and CNS-2028677 and from NIH grant 1R01HL143450.

References

[1] Benjamin EJ, Blaha MJ, Chiuve SE, Cushman M, Das SR, Deo R, Floyd J, Fornage M, Gillespie C, Isasi C, et al. Heart disease and stroke statistics-2017 update: a report from the American Heart Association. Circulation 2017; 135(10):e146–e603.

[2] Scherr D, Khairy P, Miyazaki S, Aurillac-Lavignolle V, Pascale P, Wilton SB, Ramoul K, Komatsu Y, Roten L, Jadidi A, et al. Five-year outcome of catheter ablation of persistent atrial fibrillation using termination of atrial fibrillation as a procedural endpoint. Circulation Arrhythmia and Electrophysiology 2015;8(1):18–24.

[3] Zahid S, Whyte KN, Schwarz EL, Blake III RC, Boyle PM, Chrispin J, Prakosa A, Ipek EG, Pashakhanloo F, Halperin HR, et al. Feasibility of using patient-specific models and the "minimum cut" algorithm to predict optimal ablation targets for left atrial flutter. Heart Rhythm 2016; 13(8):1687–1698.

[4] Galappaththige S, Gray RA, Costa CM, Niederer S, Pathmanathan P. Credibility assessment of patient-specific computational modeling using patient-specific cardiac modeling as an exemplar. PLoS computational biology 2022; 18(10):e1010541.

[5] Vigmond EJ, Hughes M, Plank G, Leon LJ. Computational tools for modeling electrical activity in cardiac tissue. Journal of Electrocardiology 2003;36:69–74.

[6] Mirams GR, Arthurs CJ, Bernabeu MO, Bordas R, Cooper J, Corrias A, Davit Y, Dunn SJ, Fletcher AG, Harvey DG, Marsh ME, Osborne JM, Pathmanathan P, Pitt-Francis J, Southern J, Zemzemi N, Gavaghan DJ. Chaste: An open source C++ library for computational physiology and biology. PLOS Computational Biology March 2013; 9(3):e1002970.

[7] Antonioletti M, Biktashev VN, Jackson A, Kharche SR, Stary T, Biktasheva IV. Beatbox—hpc simulation environment for biophysically and anatomically realistic cardiac electrophysiology. PloS One 2017;12(5):e0172292.

[8] Puglisi JL, Bers DM. Labheart: an interactive computer model of rabbit ventricular myocyte ion channels and ca transport. AJP Cell Phys 2001;281(6):C2049–C2060.

[9] Clerx M, Collins P, de Lange E, Volders PG. Myokit: a simple interface to cardiac cellular electrophysiology. Prog in Biophy and Mol Bio 2016;120(1-3):100–114.

[10] Kaboudian A, Cherry EM, Fenton FH. Real-time interactive simulations of large-scale systems on personal computers and cell phones. Sci Adv 2019;5(3):eaav6019.

[11] Kaboudian A, Cherry EM, Fenton FH. Large-scale interactive numerical experiments of chaos, solitons and fractals in real time via gpu in a web browser. Chaos Solitons and Fractals 2019;121:6–29.

[12] Kaboudian A, Cherry EM, Fenton FH. Real-time interactive simulations of complex ionic cardiac cell models in 2d and 3d heart structures with gpus on personal computers. In 2021 Computing in Cardiology (CinC), volume 48. 2021; 1–4.

[13] Cederman D, Tsigas P. On sorting and load balancing on gpus. ACM SIGARCH Computer Architecture News 2009; 36(5):11–18.

[14] Strocchi M, Augustin CM, Gsell MA, Karabelas E, Neic A, Gillette K, Razeghi O, Prassl AJ, Vigmond EJ, Behar JM, et al. A publicly available virtual cohort of four-chamber heart meshes for cardiac electro-mechanics simulations. PloS one 2020;15(6):e0235145.

[15] Lombardo DM, Fenton FH, Narayan SM, Rappel WJ. Comparison of detailed and simplified models of human atrial myocytes to recapitulate patient specific properties. PLoS computational biology 2016;12(8):e1005060.

[16] O'Hara T, Virág L, Varró A, Rudy Y. Simulation of the undiseased human cardiac ventricular action potential: model formulation and experimental validation. PLoS Comput Biol 2011;7(5):e1002061.

[17] Fenton FH, Cherry EM, Hastings HM, Evans SJ. Real-time computer simulations of excitable media. Biosystems 2002; 64(1-3):73–96.

[18] Chai M, Luo K, Wang H, Zheng S, Fan J. Imposing mixed dirichlet-neumann-robin boundary conditions on irregular domains in a level set/ghost fluid based finite difference framework. Computers Fluids 2021;214:104772.

Address for correspondence:

Abouzar Kaboudian
abouzar.kaboudian@fda.hhs.gov
Division of Biomedical Physics
Office of Science and Engineering Laboratories
Center for Devices and Radiological Health
US Food & Drug Admisnistration