Real-Time Interactive Simulations of Complex Ionic Cardiac Cell Models in 2D and 3D Heart Structures with GPUs on Personal Computers

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

[1] School of Physics, Georgia Institute of Technology, Atlanta, GA, USA
[2] School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA, USA

Abstract

Cardiac modeling in heart structures to study arrhythmia mechanisms has required running software on supercomputers, limiting such studies to groups with cluster access and skilled personnel. We present WebGL programs to run and visualize simulations of complex ionic models in 2D and 3D cardiac geometries, in real time, interactively using the multi-core GPU of a single computer. We use `Abubu.js`*, a library we developed for solving partial differential equations such as those describing crystal growth and fluid flow to simulate complex ionic cell models. By combining this library with JavaScript, we allow direct real-time interactions with simulations. We implemented: 1) modification of model parameters and equations at any time, with direct access to the code while it runs; 2) electrode stimulation by mouse click anywhere in the 2D/3D tissue; 3) saving the system state at any time to re-initiate dynamics from a saved state; and 4) rotation/visualization of 3D structures at any angle. As examples of this modeling platform, we implemented phenomenological and human ventricular cell models (OVVR, 41 variables). In 2D we illustrate dynamics in an annulus, disk, and square tissue; in 2D and 3D porcine ventricles, we show functional/anatomical reentry, spiral wave dynamics in different regimes, early afterdepolarizations (EADs) and real-time model parameter variation effects.*

Keywords—**Interactive 2D 3D, simulations cardiac tissue.**

1. Introduction

The first theoretical study of the complex electrical impulse transmission in cardiac tissue in 1946 [1], followed by the first numerical simulation of cardiac arrhythmias in 1964 [2], clearly illustrated the emergent behavior and possibly irregular dynamics of cells when connected in tissue. Whereas single cells have only limited oscillatory or excitable dynamics, when connected in 1D, 2D and 3D tissue, new behavior emerges, such as propagating plane waves, reentrant (spiral) waves, discordant alternans and scroll waves that can create deadly arrhythmias. Investigation of arrhythmias with computer simulations has become increasingly important, with 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 including CARP [3], Chaste [4], Beatbox [5] and Cardioid [6]. While these programs are robust, all require downloading/installation and access to parallel supercomputers and multiple CPUs. Less complex but interactive codes have been developed for simulations ranging from single-cell dynamics, such as Labheart [7] for ion channels and Ca transport in rabbit ventricular cells and Myokit [8] for a variety of cells, to Xspiral [9] (originally created under Unix X-windows, then translated to Java applets), which includes a suite of various cell models from single cell, 1D and 2D, to ezspiral/ezscroll [10] codes in openGL to simulate the Barkley model in 2D and 3D.

While complex software can simulate physiologically detailed cardiac cell models in anatomically realistic structures, they require access to supercomputers and a strong knowledge of programming and modeling, and they do not have interactivity capabilities. Results are obtained after the simulation has finished and the data are processed for analysis and visualization. In contrast, interactive software can visualize the simulation as it occurs and interactivity with the parameters and the system is possible. However, the models used are relatively simple and far less detailed.

We present a new type of interactive code for simulations of simple and complex ionic cell models in 2D and 3D anatomically accurate structures. The codes are developed using WebGL, a shader language that allows access to the GPU of a computer to render interactive 2D and 3D graphics from a web browser. As the simulation structures are accessible by the many GPU processors, it is possi-

ble to solve the many ODEs/PDEs of cardiac models in parallel. Our approach [11] represents the first real-time simulation of complex models in physiologically accurate structures **with direct interaction with the system and the code itself while running** on a single desktop computer. There is no need to download any packages or plug-ins or for any knowledge of coding or modeling. The models run independent of operating system and device and on the GPU of the computer that opens the link to the codes, with faster GPUs running the codes more quickly.

2. Methods

We solve a simplified ionic model [12] and a complex human ventricular model [13] in 2D and a 3D porcine ventricle structure using uniform Cartesian grids with the same discretization of 0.02‘cm. Grid points are marked as domain points or empty space using binary values. The `WebGL` numerical implementations rely on the `Abubu.js` library [11, 14], which is available along with training resources for implementing `WebGL` programs using it at https://www.abubujs.org/

Numerical methods. The simulations are carried out by solving the cable equation [3, 5, 6, 9]:

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

where V represents the membrane potential, D is the diffusion coefficient, I_{ion} are ionic currents described by either the 3-variable model [12] or the 41-variable OVVR model [13] and C_m is the membrane capacitance. The small discretization steps required in both space and time impose high computational costs, which are traditionally addressed using parallel clusters. Modern GPUs provide an affordable alternative to clusters. While there are multiple languages and approaches to achieve GPU parallelization, we chose `WebGL`, which provides superior performance[15] as well as portability and interactivity [11, 14].

We use 2D texture memory to store the domain information both for 2D and 3D. In 3D, the third dimension is implemented using a sub-grid implementation where each sub-grid is a slice in the third dimension. Each texture can store four color channels and each channel is assigned to a physical variable. The textures are manipulated using `Abubu.js` solvers, which are JavaScript objects that run the shader code to color one or multiple textures at a time. Sequential runs of the solvers create the time-stepping algorithm while each solver carries out solutions of the ODE gating variables using a semi-implicit method. Solvers use the fragment and shader codes to receive their instructions for texture manipulations and use a single-instruction, multiple-data approach to achieve parallelization. More details can be found at https://www.abubujs.org/learning.

Zero-flux boundary conditions. The Laplacian term in eq. (1) is discretized using a second-order centered difference scheme. Zero-flux boundary conditions are handled with a phase-field approach [11] for large domains, and the ghost/fictitious method widely used in fluid flow [16] and solid media wave propagation is applied in the normal direction for thin boundaries.

Interactivity. Combining `WebGL` with `Abubu.js` enables us to utilize the power of the JavaScript language for event-handling and interacting with the `WebGL` part of the program. User interactions can be carried out through the graphical user interface (GUI) menus or by mouse and screen touch interactions.

Altering parameters. Parameters whose values can be changed during the simulation can be defined as uniforms in the shaders and solvers. `Abubu.js` facilitates updating the values of these uniforms during the simulation. We have added a number of these parameters to the GUI. When a value is changed in the GUI, a `change` event is triggered. Callback functions are designed in each program to update the corresponding uniforms in the solvers when the change event is triggered. The changes can be applied in the middle of time-stepping; where the time-stepping is interrupted, the uniform value is updated and then time-stepping is resumed using the updated values.

Excitation/obstacle addition/removal. To excite the tissue, a stimulating current is injected into the target region using the mouse; the size and intensity can be varied in the menu. Interactions are connected to a special solver that creates the desired manipulations to the membrane potential or the texture defining the tissue (domain) points. If the domain boundaries are changed, the texel (texture element) indices are updated with the ghost method.

Visualizing the 3D structure. The geometry is created on the fly by visualizing voxels (3D pixels) of information. Each voxel is displayed as a cube. To avoid sending a huge number of coordinates and connectivity information for generating all voxels, the GPU program uses the appropriate number of triangles required to visualize all the voxels without directly sending node data. Instead, the coordinates of a unit cube are saved in the vertex shader. The voxel center coordinates can be obtained from their indices in the uniform Cartesian 3D grid. Subsequently, the coordinates of the voxel center are used to translate the cube that is already scaled to the appropriate size. Coloring is decided based on the membrane potential and the light. A Lambertian model is used for lighting, which requires calculation of the direction of light an the surface's normal.

Mouse interactions in 3D space. 3D interactions with the mouse fall into two categories: manipulating the viewpoint and creating excitation regions in 3D space. The viewpoint is set by manipulating the camera matrix and updating the visualizer solvers with the new matrix values.

Detecting the excitation coordinates in the tissue structure by the mouse in 3D, referred to as picking, is more involved. To pick the appropriate coordinate given a click location, the coordinates of the visible points of the structure are projected using the same view and projection matrices used for visualization to a texture off screen. The aspect ratio of this texture must match that of the canvas used for visualization. The click point on the canvas is used to read the projected coordinate texture to find the coordinates of the target point for stimulation. These coordinates then can be used to create the excitation region.

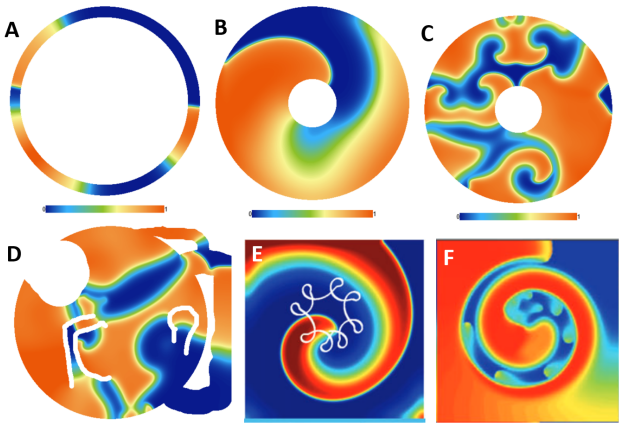

Figure 1. Examples of simulations with the 2D codes. (A) Narrow annulus with 3 propagating waves. (B) Annulus and anatomical spiral wave with alternans. (C) annulus and spiral-wave breakup. (D) Adding obstacles and new connections. (E) OVVR model with decreased I_{Na} current. (F) Spiral wave with EADs along the wave back.

Editing source code during simulations. While the main parameters of a model, such as conductances, can be included in a menu to be changed easily during a simulation, complex models have hundreds of parameters and dozens of equations that it may be useful to modify. With `WebGL`, the ACE editor can be used to open the shader's source code in the same web page that is running the code, allowing the code to be modified while running! Whenever the source code changes, the source code for the appropriate solvers is updated and the `Abubu.js` library automatically re-compiles the new code and immediately uses it. Therefore, **any** parameter or equation in the model can be modified at any time during a simulation.

Saving and reloading data using JSON. The JSON format is an easy and efficient process to save data in JavaScript. Texture data can be saved by passing the `.value` attribute of the object to a JSON object to be saved. Once a JSON file is loaded, the same data can be assigned to the `.value` attribute of the texture. The `Abubu.js` library automatically uploads the value to the GPU to be used by the application. Modeling parameters can be passed to the JSON object in the same way to be saved. Once reloaded, the corresponding solvers and the GUI are updated with the imported values.

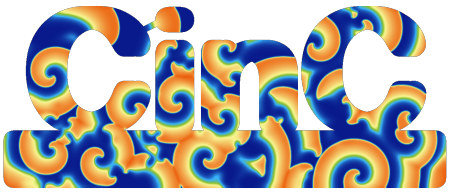

Figure 2. Charge conservation at irregular domain boundaries. The CinC structure is simulated with parameter set 6 of Ref. [12].

3. 2D Results

Example of simulations possible with the 2D codes (see link to codes following Conclusions) are shown in Fig. 1. Codes will start as soon as the link to the page is open; to initiate a simulation, click on the start/pause button near the bottom of the menu interactions options. The mouse-click settings allow the user to stimulate the tissue; add/draw obstacles, walls and channels of any size; or add new tissue with the mouse (Fig. 1A-F, Fig. 2). Stimulation can initiate waves in the tissue and model parameters can be changed by the "model parameters" menu of the GUI ***(or by using the edit source code option and making changes directly to the code as it runs)***. Fig. 1B-C show dynamics of the 3V model for parameter sets 3 and 5 of Ref. [12]. Fig. 1E-F use the OVVR model [13] with doubled capacitance g_{Ito} and the time constant τ_d multiplied by five, resulting in meandering dynamics and EADs emanating from the wave back. Any initial condition or structure can be saved or loaded with the save/reload menu.

4. 3D Results

Fig. 3A-B shows a 3D porcine ventricular structure [11] with the 3-variable [12] and the 41-variable OVVR [13] models. Multiple-spiral and single-spiral cases initiated by the S1-S2 protocol using the mouse are shown. The structure can be rotated and visualized at any angle, and all parameters of the models can be modified in real time to investigate the dynamics of the waves. Periodic stimulation can be induced at any point with any frequency and size using the menu options. Additionally, the voltage signal (or any current or gate variable) from any point in the tissue can be displayed as a function of time in a moving window next to the 3D simulation (Fig. 3C). Simulation with the 3V model on the porcine structure using an NVIDIA-2080TI GPU card ran faster than real time, while

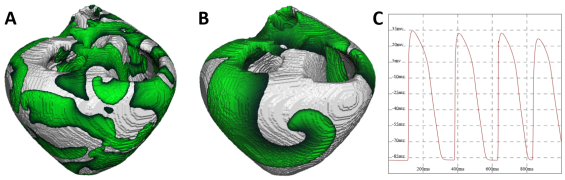

Figure 3. 3D simulations in porcine ventricles. (A) Multiple spiral waves (3V model [12]). (B) Single spiral wave with the OVVR model [13]. (C) Screen shot of voltage signal in time from one point in the domain.

with the OVVR model it runs 3 times slower than real time at maximum output.

5. Conclusions

In this paper we greatly extend our previous work [11, 14] by creating a set of four programs, two for 2D and two for 3D simulations, for two cardiac cell models, a phenomenological 3-ODE model [12] and the 41-ODE OVVR model [13], which is currently considered the standard to study arrhythmias by the FDA and CiPA initiative. Because these codes do not require a computer cluster, programming knowledge, or any downloads and compilation, they can be accessed and used by a large community of students and researchers. Along with their speed from GPU acceleration, they provide the ability to interact with the simulation in real time and to change any model parameter using a menu or by directly by modifying the code while it runs. Extensions of these codes to other heart structures and models is relatively simple, so we expect that more codes will be developed by us and others in the near future and that this framework will be valuable for future studies of arrhythmia mechanisms and control. The `WebGL` codes can be accessed and run by visiting https://kaboudian.github.io/CinC-2021-Competition/.

Acknowledgments

We acknowledge support from: NIH 1R01HL143450-01, NSF-FDA-2037894, NSF CNS-2028677, CNS-1446675 and 1762553.

References

[1] Wiener N, Rosenblueth A. A conduction of impulses in cardiac muscle. Arch Inst Cardiol Mex 1946;16:205–265.

[2] Moe GK, Rheinboldt WC, Abildskov J. A computer model of atrial fibrillation. American Heart Journal 1964; 67(2):200–220.

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

[4] 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.

[5] 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.

[6] Richards DF, Glosli JN, Draeger EW, Mirin AA, Chan ea. Towards real-time simulation of cardiac electrophysiology in a human heart at high resolution. Comp Meth in Biomech and Biomed Eng 2013;16(7):802–805.

[7] 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.

[8] 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.

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

[10] Dowle M, Martin Mantel R, Barkley D. Fast simulations of waves in three-dimensional excitable media. International Journal of Bifurcation and Chaos 1997;7(11):2529–2545.

[11] 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.

[12] Fenton FH, Cherry EM, Hastings HM, Evans SJ. Multiple mechanisms of spiral wave breakup in a model of cardiac electrical activity. Chaos An Interdisciplinary Journal of Nonlinear Science 2002;12(3):852–892.

[13] 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.

[14] 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.

[15] Kaboudian A, Velasco-Perez HA, Iravanian S, Shiferaw Y, Cherry EM, Fenton FH. A comprehensive comparison of gpu implementations of cardiac electrophysiology models. In From Reactive Systems to Cyber-Physical Systems. Springer, 2019; 9–34.

[16] Fedkiw RP, Aslam T, Merriman B, Osher S. A non-oscillatory eulerian approach to interfaces in multimaterial flows. J Comp Phys 1999;152(2):457–492.

Address for correspondence:

Flavio H. Fenton

flavio.fenton@physics.gatech.edu

School of Physics, Georgia Institute of Technology