Computer simulations of arrhythmia and ECG

David Casas Vidal* and Ángel García Gómez†
Departament de Física Aplicada, Universitat Politécnica de Catalunya,
EPSEB, 44-50 Av. Dr. Marañón, 08028 Barcelona, Spain

Abstract: Having computer simulations of biophysical systems is an important step for checking the validity of our models and serve as a first testing ground for further applications. This article describes a faster implementation of a known approach to the heart's electrophysiology, as well as taking a look at the simulation results and how well do they match known heart conditions. In the end our application shows an speedup of up to 115 while leading to results that agree with experimental observation. Hence we believe our work to be a useful improvement over other implementations and in addition a similar improvement could be brought upon other similar simulations.

Keywords: cardiology simulations, reentrant ventricular arrhythmia, alternans, heart electrophysiology, ventricular fibrillation, ventricular tachycardia, computational biophysics, nvidia CUDA, parallel simulation,

INTRODUCTION

The aim of this article is to showcase the results obtained with an up-to-date program we developed which models the electrophysiology of heart tissue. In order to have an understanding of such results, some biology related concepts are introduced under these lines.

Electrophysiology

Contraction in the heart's muscular tissue is triggered by an electrical impulse called *action potential*. The action potential is a wave sourced in one point (the sinoatrial node) that propagates through the heart. When the action potential propagates abnormally, it affects the heart's beat and may lead to different types of arrhythmia.

A method for analyzing how this wave travels through the heart is electrocardiography or *ECG*. Propagation is considered normal when the wavefront advances onwards steadily and periodically, leading to a regular heartbeat and an ECG like the one shown in Figure 1.

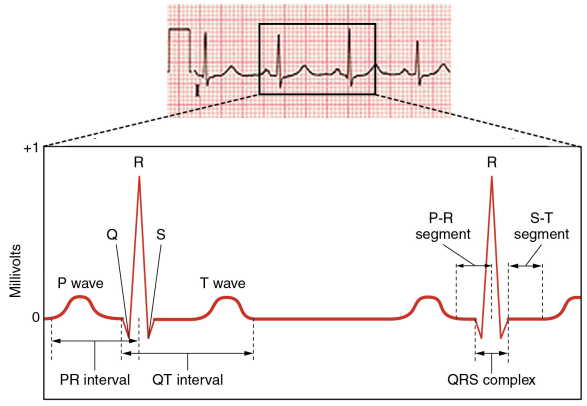

FIG. 1. At the top, a fragment of a normal ECG. Below, a labeled diagram showing its different regions.

There is an abnormality in the wave's propagation of great relevance called *reentry*. Due to a variety of causes like the existence of dead, unexcitable tissue or simply a difference in the tissue properties, a wave may travel back in a circular fashion. Often enough this will lead to a self-sustaining cycle, effectively becoming a signal generator. When such is the case, the heart beats at exceedingly high rates, developing *ventricular tachycardia*. More than one reentry foci can exist at a given moment, causing what is known as *polymorphic* ventricular tachycardia (as opposed to *monomorphic* ventricular tachycardia).

On more extreme cases, the wave will break apart and change into a self-sustaining rapid-changing chaotic pattern of stimuli. Under such circumstances, the ventricle cannot contract coherently: the heart enters *ventricular fibrillation* and becomes unable to pump blood.

There are several mechanisms for VT to become a fibrillation, one such appears when the pulses' duration changes between each heartbeat, leading to an ECG where the T wave is alternating between two different positions and/or amplitudes. This is called *alternans* and it is a factor that increases the chances of the aforementioned tachycardia becoming a fibrillation, thus being recognized as a risk factor of suffering *sudden death*.

METHODS

Physical Model

We use the model explained in [1], where the electrical cardiac wave propagation is modelled with an ionic model composed of three variables: the membrane potential $u(x,t)$ and the inactivation gates of the fast and slow inward currents, respectively $h(x,t)$ and $f(x,t)$. These variables obey the kinetics:

$$\frac{\partial u}{\partial t} = D\nabla^2 V - J_{ion}$$

$$\frac{\partial h}{\partial t} = \frac{h_\infty(u) - h}{\tau_h(u)}$$
$$\frac{\partial f}{\partial t} = \frac{f_\infty(u) - h}{\tau_f(u)}$$

Subject to the boundary condition $\hat{n} \cdot \nabla V = 0$ and where J_{ion} is the total membrane scaled current, the sum of the fast inward Na current J_{fi}, the slow outward K current J_{so} and the slow inward Ca current J_{si}:

$$J_{fi} = hm_\infty(u - 1.3)/\tau_{fi}$$
$$J_{si} = fd_\infty(u - 1.4)/\tau_{si}$$
$$J_{so} = (1 - e^{-4u})/\tau_{so}$$

The various functions are:

$$m_\infty(u) = \begin{cases} \frac{(5u)^6}{1+(5u)^6} & \text{if } u > 0 \\ 0 & \text{if } u < 0 \end{cases}$$
$$h_\infty(u) = \begin{cases} \frac{1}{1+(10u)^6} & \text{if } u > 0 \\ 1 & \text{if } u < 0 \end{cases}$$
$$d_\infty(u) = \begin{cases} \frac{(2.5u)^4}{1+(2.5u)^4} & \text{if } u > 0 \\ 0 & \text{if } u < 0 \end{cases}$$
$$f_\infty(u) = \begin{cases} \frac{1}{1+(10u)^4} & \text{if } u > 0 \\ 1 & \text{if } u < 0 \end{cases}$$
$$\tau_h(u) = \tau_{h_1} + \tau_{h_2} e^{-20(u-0.1)^2}$$
$$\tau_f(u) = \tau_{f_2} + (\tau_{f_1} - \tau_{f_2})u^3$$

The exterior potential used to compute the ECGs follows the equation:

$$\phi(t) = \int_S \frac{\nabla^2 u(r,t)}{\sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2}} dr$$

Where S is the heart's surface, and (x_0, y_0, z_0) is the position of the electrode. Actually, this is not the real value of the exterior potential, since it uses the adimensional membrane potential u, but it is proportional to the correct value.

Numerical Method

The heart tissue is modelled with a 2-dimensional grid, in which each point corresponds to a cell in the tissue, and therefore has a value for the potential and the inactivation gates. Using this, the 2-dimensional laplacian of every inner cell can be computed with the following finite differences approximation:

$$\nabla^2_{1/3} = \frac{2}{3}\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} + \frac{1}{3}\begin{pmatrix} 1/2 & 0 & 1/2 \\ 0 & -2 & 0 \\ 1/2 & 0 & 1/2 \end{pmatrix}$$

This approximation is the one that gives the best approximation of rotational symmetry[2]. Then, also for each cell in the grid, the time dependent kinetic equations are solved numerically using Euler's method. Finally, the ECGs are obtained with a numerical integral, using the trapezoidal rule.

Code and Optimization

Besides the model's correctness, another concern we had was that of speed. Since some parameters have to be fine-tuned to achieve the desired simulation results, many runs are needed to get a single result. Waiting for hours for each run to complete was not acceptable and hence making the application more efficient became a capital point too.

Sequential code. Initial tests aside, our first application was a sequential implementation of the simulation. For every time step to be simulated, we looped over each cell and solved the equation there, as well as add its contribution to the ECG.

First parallel code. Due to the nature of our simulation, we reckoned paralelization as the best way to improve the applications' speed and therefore we decided to take advantage of GPU parallel computing. A first naive, direct translation of the previous code into its parallel version was made. While a great speedup was indeed obtained, we felt the improvement wasn't quite as great as expected, so we looked closer into it, identifying some faults: *a*) using *atomicAdd* to accumulate the ECG breaks the advantage of parallelism since all threads have to wait each other anyway; *b*) copying memory between host (cpu) and device (gpu) is slower than expected; and *c*) the cpu spends most of its time idle, waiting for the gpu to finish.

Final parallel code. A final parallel code was developed that tried to fix the issues we just mentioned: *a*) the atomic operation was replaced with a parallel reduction [3]; *b*) memory transfers were overlapped with other operations to the extent it was possible; and *c*) the generation of the images is now done while the gpu is solving.We believe that this way the computer's resources are used as efficiently as possible.

Parallelism aside, there was another way to speed up our code whose approach was entirely different.The model requires the use of many exponentials, which is a very slow operation. Therefore, the application may be speeded if we use *precalculated tables* instead —this leads to a precission loss, which we dismissed as not critical. In the end, the different applications can be executed either using tables or not.

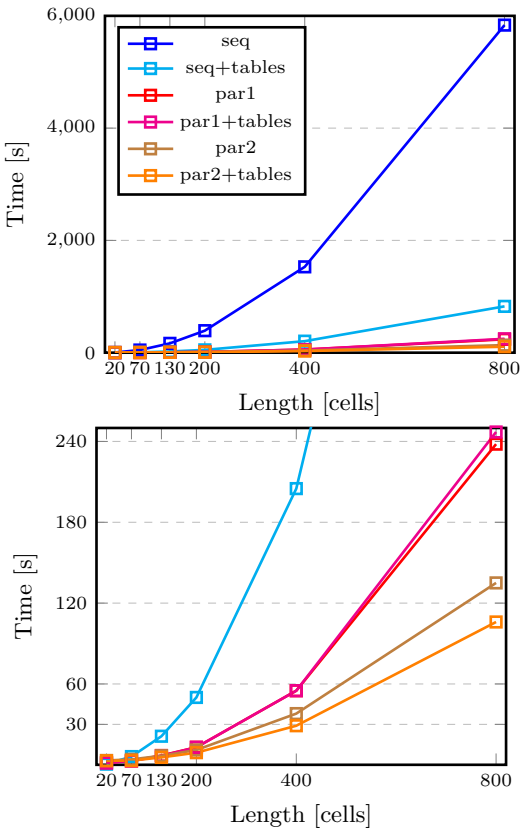

FIG. 2. Graph showing the total time each version needed to finish simulations of different sizes. On the left, a comparison between the two sequential versions and any parallel is provided. On the right parallel versions are compared, with some points of sequential+tables as a reference. Since the distance between the first points can't be fully appreciated, Table I is provided containing the actual values.

size	reference	seq	par1	par2	table-seq	table-par1	table-par2
20	0	3.6	1.6	3.2	0.5	1.3	3
70	0	47	2.9	4	6.1	2.8	3.1
130	0	166	6.9	6.7	21.2	6.6	5.5
200	0	392	13	11	50	13	9
400	0	1,529	55	38	205	55	29
800	12,222	5,835	238	135	825	247	106

TABLE I. Table showing the runtime in seconds for each code to complete a simulation of each one of the shown sizes. Part of this data is plotted in Figure 2. The reference column shows the time for the reference code to complete the simulation, with zeroes in those sizes we did't run it for. The seq columns hold the runtimes for the sequential code, par-1 those of the first naive parallel program and par-2 those of the final application.

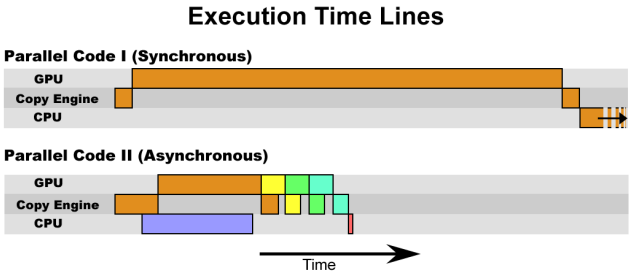

FIG. 3. Diagram showing the execution timeline of each parallel version when running each loop of the 800x800 size simulation. Blocks of the same colour share the same stream (ie. they are synchronous among themselves). The timeline is also on scale: blocks twice as wide take twice as long to finish. The last block in Parallel 1 is about 2.5 times larger than the whole time span showed, but occurs only one out of fifty iterations.

RESULTS

Code speed comparisons

We ran all of the codes we explained (sequential, first parallel and second parallel) both using tables and not, obtaining the results shown in both Table I and Figure 2. We also ran a reference code we were provided which implements the same numerical methods (Euler's method with the same Laplacian approximation), whose time is shown in Table I too.

The results show that for small system sizes ($\lesssim 100$) Parallel 1 is faster than Parallel 2, most likely due to the overhead introduced to improve its scalability. It is also evident that the use of tables is a definite improvement, being most useful in the sequential case in which it consistently serves an speedup of about 700% or higher, and less so in Parallel 1 where the effect is close to unnoticeable. Overall, all methods show an improvement with respect to the reference code, being by far the best (faster, escales better) Parallel 2 with tables.

A more detailed time analysis has also been done on the two parallel codes to make sure they behave as expected, ie. to make sure that Parallel 2 indeed overlaps different operations. The result is illustrated in Figure 3.

Simulation Results

Figure 4 shows the results obtained with different simulations, all of them with an 800x800 cells grid ($dx = 0.025$cm) and spanning 4 seconds.

The normal case is obtained stimulating the first ten rows with -0.4 for 1ms at a period of 250 ms, with constants: $\tau_{so} = 15, \tau_{fi} = 3, \tau_{h1} = 4, \tau_{f1} = 200, \tau_{f2} = 100, \tau_{si} = 6, \tau_{h2} = 1$. In order to obtain the ECG shown here, the probe is placed at (400,400,100), that is, hovering about 100 cells' width over the center of the square. As we can see, all the typical structures remain identifiable in the resulting ECG, even though it looks different than what we are used to, mostly due to the fact that we are simulating a square of tissue instead of a 3D heart and computing the ECG from a single electrode instead

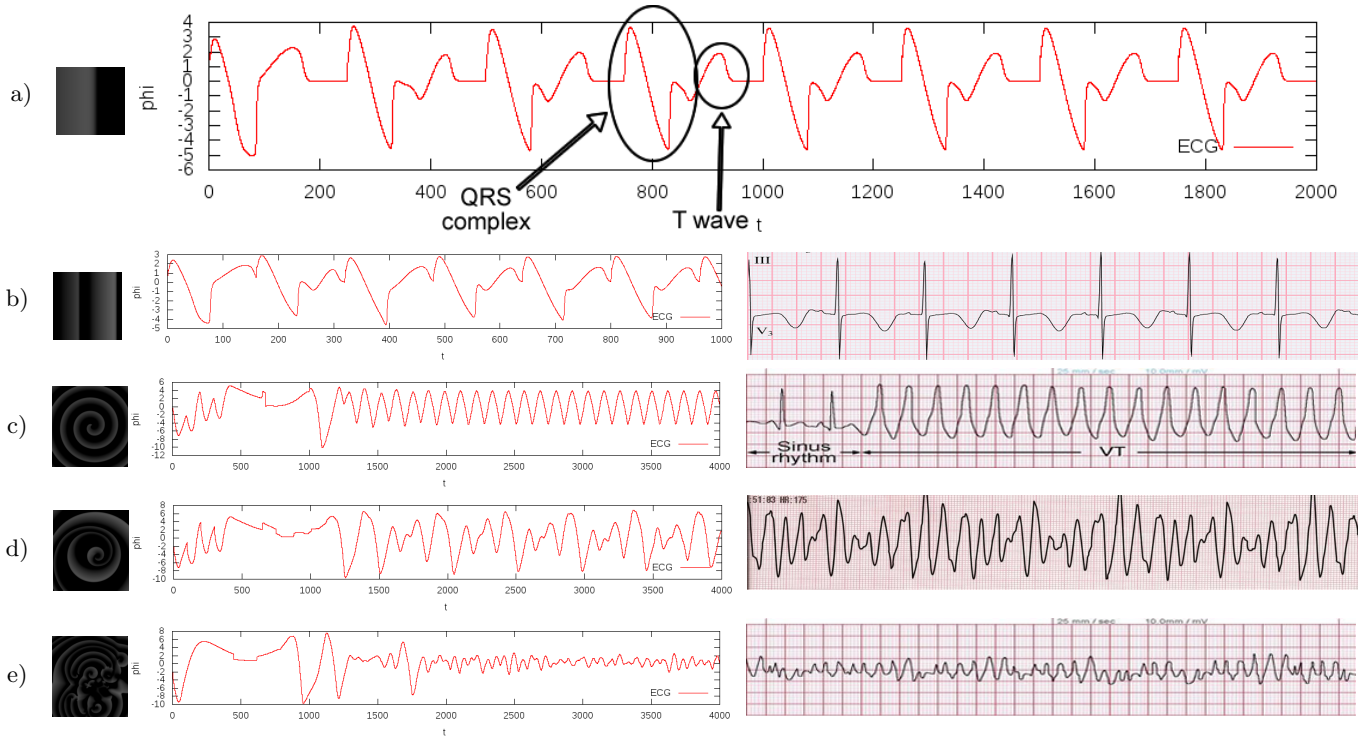

FIG. 4. For all subfigures (except from the first one, which has no "right" element) : on the left, a miniature with a heatmap of the potential (the lighter the colour, the higher the potential); on the middle, our generated ECG; on the right a real-case ECG from a patient that suffered that condition. Where each subfigure is a) normal heart; b) alternans[b]; c) monorphic ventricular tachycardia; d) polymorphic ventricular tachycardia; and e) ventricular fibrillation.

of using the difference between different probes.

To obtain alternans, we decreased the stimulation period to 160 ms. We can see how in both our simulation as well as the experimental ECG the T wave moves up and down at every heartbeat.

Tachycardia is obtained by setting the potential and the gates everywhere but the rectangular region between (400,0) and (457,400) to zero at the 820 ms mark. Then the wave suffers from reentry, starting the formation of spirals. Monomorphic tachycardia is obtained by introducing a region of unexcitable tissue whose location is closse to the reentry point. As a result the spiral gets anchored and rotates at a constant frequency. The ECG obtained here had the probe placed at (0,400,100) due to the symmetry existing in the previous position. Another simulation is made without the region of unexcitable tissue. Without this obstacle, the tip of the spiral starts moving in circles, leading to a meandering spiral which, due to Doppler's effect, produces uneven waves, bunching them in front of the tip and widening those behind it. This results in a modulation of the ECG, also known as polymorphic ventricular tachycardia. Both of them look fairly similar to their experimental counterparts.

Changing some parameters ($\tau_{si} = 5, \tau_{h2} = 2$) as well as the rectangular region that doesn't get grounded to (400,0),(800,400); the previous spiral breaks apart giving rise to the emergence of many other rotors that interfere with each other, resulting in a very disordered, probably spatio-temporally chaotic, state. This is the fibrillation case and we can see that its ECG is quite similar to that from experimental data.

CONCLUSIONS

All in all, we have developed an application that successfully simulates the behaviour of an action potential travelling through heart tissue, obtaining ECGs that fit with those obtained experimentally. Our implementation is more than 115 times faster than previous reference codes by making use of well-known optimizations that could be generalized to other biophysical simulations.

* Also at Facultat d'Informàtica de Barcelona, UPC.
† Also at Facultat de Matemàtiques i Estadística, UPC.

[1] B.Echebarría and A.Karma, Eur. Phys. J. Special Topics **146**, 217 (2007).
[2] T. Lindeberg, IEEE Transactions of Pattern Analysis and Machine Intelligence **12**, 234 (1990).
[3] M. Harris, "Optimizing parallel reduction in CUDA," (2007).