ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA

SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA BIOMEDICA

A 2-DIMENSIONAL COMPUTATIONAL MODEL TO ANALYZE THE EFFECTS OF CELLULAR HETEROGEINITY ON CARDIAC PACEMAKING

Tesi in:

Bioingegneria molecolare e cellulare LM

Relatore:

Prof. **STEFANO SEVERI**

Correlatore:

Prof. **ERIC SOBIE**

Presentata da:

CHIARA CAMPANA

Sessione III

Anno Accademico 2013 - 2014

Alla mia famiglia

Contents

Abstract

L'azione meccanica del cuore è possibile grazie al verificarsi di eventi elettrici che interessano le cellule cardiache, proprietà che classifica il tessuto cardiaco tra i tessuti eccitabili. L'evento elettrico è il segnale che scatena la contrazione meccanica, inducendo un temporaneo incremento di calcio intracellulare, che, a sua volta, reca un messaggio di contrazione alle proteine contrattili della cellula. Per queste ragioni il processo che combina l'eccitazione elettrica alla funzione meccanica è definito accoppiamento eccitazione-contrazione. Il sistema di conduzione atriale comprende il nodo seno-atriale (SAN) posizionato nel lato superiore destro del cuore e in grado di generare spontaneamente un segnale elettrico periodico ad una frequenza di 60-100 battiti al minuto. In seguito le fibre intra-nodali conducono l'impulso al nodo atrioventricolare, che rappresenta l'unica connessione elettrica tra atri e ventricoli. Il segnale elettrico si propaga attraverso il tessuto cardiaco via gap junctions tra cardiomiociti e in ciascuno di questi induce un processo denominato potenziale d'azione (AP). La morfologia del potenziale d'azione cardiaco mostra un'elevata variabilità all'interno del cuore. Le cellule del SAN non possiedono un vero e proprio potenziale di riposo, bensì generano regolarmente potenziali d'azione spontanei. A differenza dei potenziali d'azione non-pacemaker nel cuore, la corrente depolarizzante proviene principalmente da ioni Ca^{2+} piuttosto che da correnti di Na^{+}. Non vi sono infatti canali veloci di Na^{+} e relative correnti operanti nelle cellule nodali SA. Ciò si traduce in un più lento potenziale d'azione in termini di rapidità con cui le cellule si depolarizzano.
I meccanismi citati possono essere descritti e studiati sfruttando i principi della modellazione matematica, introdotta in ambito cardiaco in seguito al lavoro di Hodgkin e Huxley, i quali hanno presentato la descrizione matematica di correnti ioniche generanti AP nell'assone gigante di calamaro. Le equazioni di Hodgkin-Huxley (HH) costituiscono ancora oggi parte della

modellazione di AP cardiaci. Seguendo questo formalismo, la cellula è rappresentata come un circuito elettrico, in cui la membrana è descritta come una capacità e i canali ionici voltaggio-dipendenti come conduttanze elettriche. Da allora, il numero di modelli formulati è cresciuto rapidamente. In questa tesi sono presi in considerazione i due più recenti modelli di singola cellula del nodo seno-atriale di coniglio (modello Severi (2012) e modello Maltsev (2009)).
Nella formulazione di modelli di singola cellula si considerano solamente le proprietà medie del tessuto studiato, invece, per fornire una descrizione più realistica, sarebbe utile considerare la variabilità normalmente presente all'interno del tessuto. Questo è particolarmente vero nel caso del SAN, che ha una struttura molto complessa che mostra eterogeneità anatomica e funzionale. Come sarà discusso più avanti, questa variabilità può dipendere da molteplici cause. È possibile allora considerare modelli di cellule accoppiate utilizzando un array o una matrice e introducendo differenze nel comportamento di ogni singola cellula. In tal caso il comportamento di ciascuna cellula è influenzato da quella vicina, in particolare ogni cellula riceverà un contributo in corrente dalle cellule adiacenti, che dipende dalla loro differenza di tensione e dalla resistenza di accoppiamento attraverso la legge di Ohm. Tale ragionamento è noto come cable theory e può essere esteso ad una propagazione multidimensionale, considerando un modello tissutale.
Obiettivo principale del mio progetto è stato l'implementazione in CUDA (acronimo di Compute Unified Device Architecture, un'architettura hardware per l'elaborazione parallela creata da NVIDIA) di un modello tissutale del nodo seno-atriale di coniglio attraverso il quale valutare l'eterogeneità della sua struttura e come tale variabilità influenzi il comportamento delle cellule. In particolare ogni cellula possiede una frequenza di scarica intrinseca, dunque diversa da quella di ogni altra cellula del tessuto ed è quindi interessante studiare il processo di sincronizzazione delle cellule e quale sia la frequenza ultima di scarica qualora queste risultino sincronizzate.

- Il primo passo è stato realizzato utilizzando MATLAB per implementare un modello monodimensionale del SAN di coniglio, descrivendo ogni cellula attraverso il modello Maltsev o Severi. Considerando un gruppo di cellule in fila, se ciascuna cellula è regolata dalle stesse equazioni differenziali ed ha gli stessi valori per tutti i parametri, in ogni momento le cellule possiedono lo stesso potenziale di membrana e quindi hanno tutte la stessa frequenza di scarica. Ciò si traduce in

un comportamento identico a quello della singola cellula.

- Assegnare ad ogni cellula diverse condizioni iniziali ha in seguito permesso di valutare l'effetto dell'accoppiamento tra cellule sulla velocità di conduzione.

- Il resto del lavoro è stato effettuato utilizzando CUDA e visualizzando poi i risultati in MATLAB. Grazie ai vantaggi introdotti dallo sfruttamento di unità di elaborazione grafiche (GPU) in termini di tempo di esecuzione, è stato possibile creare un modello 2D del SAN di coniglio.

- Tale modello è stato infine utilizzato per esaminare la sincronizzazione e l'influenza reciproca tra le cellule. Dopo aver eseguito la randomizzazione di tutte le conduttanze massime presenti nel modello, sono stati valutati gli effetti sulla lunghezza del ciclo e l'ampiezza dei potenziali d'azione. Diversi livelli di accoppiamento resistivo tra cellule e di variabilità intercellulare delle conduttanze sono stati testati.

Le simulazioni effettuate utilizzando il modello realizzato suggeriscono che le cellule sincronizzano la loro frequenza di scarica. In particolare il valore ultimo della frequenza di scarica cresce al crescere della resistenza di accoppiamento e della variabilità intercellulare testate. L'ampiezza dei potenziali d'azione presenta un comportamento simile a quello della lunghezza del ciclo, diminuendo nel caso di aumento di resistenza di accoppiamento e variabilità intercellulare, anche se in maniera meno marcata nell'ultimo caso.
Il primo capitolo tratta il sistema di conduzione cardiaco, le caratteristiche del nodo seno-atriale e il potenziale d'azione delle cellule che lo costituiscono per poi riportare i principi della modellazione cardiaca, con particolare riferimento alla descrizione della propagazione elettrica intercellulare. Nel secondo capitolo si discute l'implementazione dei modelli utilizzati in questo lavoro di tesi e sono descritte in dettaglio le simulazioni effettuate. L'ultimo capitolo riporta infine i risultati ottenuti dalle simulazioni, dedicando particolare attenzione agli esiti delle simulazioni 2D, obiettivo primario del lavoro.

Introduction

The mechanical action of the heart is made possible in response to electrical events that involve the cardiac cells, a property that classifies the heart tissue between the excitable tissues. At the cellular level, the electrical event is the signal that triggers the mechanical contraction, inducing a transient increase in intracellular calcium which, in turn, carries the message of contraction to the contractile proteins of the cell. For these reasons, the process that combines the electrical excitation to the mechanical function is called *excitation-contraction coupling*. The atrial conduction system includes the *sinoatrial node (SAN)* placed in the upper right corner of the heart able to spontaneously generate a periodic electrical signal at a frequency of 60-100 bpm; then intra-nodal pathways lead the impulse to the atrioventricular node, which is the only electrical connection between atria and ventricles. The electrical signal propagates in the cardiac tissue via gap junctions between cardiac myocytes and in each myocyte it initiates a process named *action potential (AP)*. The morphology of the cardiac action potential shows a high variability within of the heart. SAN cells are characterized as having no true resting potential, but instead generate regular, spontaneous action potentials. Unlike non-pacemaker action potentials in the heart, the depolarizing current is carried into the cell primarily by relatively slow Ca^{2+} currents instead of by fast Na^{+} currents. There are, in fact, no fast Na^{+} channels and currents operating in SA nodal cells. This results in a slower action potentials in terms of how rapidly they depolarize.

The mentioned mechanisms can be usefully treated within the mathematical modeling. The first step towards mathematical modeling of cardiac cells was made after Hodgkin and Huxley presented the mathematical description of ion currents generating APs in the squid giant axon. The Hodgkin-Huxley (H-H) equations were later introduced to the field of cardiac APs, and the H-H formalism is still used as a part of nowadays cardiac

AP modeling. Following this formalism, the cell is represented as an electrical circuit, where the membrane is represented as a capacitance and voltage gated ion channel are represented as electrical conductances. Since that time, the number of different cell models has grown rapidly. In this thesis the two most recent rabbit SAN cell models (Severi model (2012), Maltsev model (2009)) are taken into account.
In formulating models of single cell, only the average properties of the tissue under inspection can be considered, instead to have a more realistic description it would be helpful to consider the variability normally present within the tissue. This is especially true in the case of the SAN, that has a very complex structure showing anatomical and functional heterogeneity. As discussed later, this variability may depend on multiple causes. It is possible to consider models of coupled cells using an array or a matrix of cells and introducing differences in the behavior of each single cell. In that case the behavior of each cell is influenced by her neighboring, in particular each cell will receive a contribution in current from its neighbors, which depends from their voltage difference and from the coupling resistance through the Ohm's law. This is known as the cable theory and it can be extended to a multidimensional propagation, considering a tissue model.

The primary goal of my project was to implement in *CUDA (Compute Unified Device Architecture*, an hardware architecture for parallel processing created by NVIDIA) a tissue model of the rabbit sinoatrial node to evaluate the heterogeneity of its structure and how that variability influences the behavior of the cells. In particular, each cell has an intrinsic discharge frequency, thus different from that of every other cell of the tissue and it is interesting to study the process of synchronization of the cells and look at the value of the last discharge frequency if they synchronized.

- The first step has been made using MATLAB to implement a 1-dimensional model of the rabbit sinoatrial node, describing each cell through the Maltsev or Severi model. Considering a cable of cells, if each cell is governed by the same differential equations and has the same values for all the parameters, at any moment the cells will be at the same potential and therefore have all the same rate. This results in a behavior of the cable which is identical to the behavior of the single cell.

- Giving each cell different initial conditions has made it possible to

evaluate the effect of the inter-cellular coupling on the conduction velocity.

- The rest of the work was carried out using CUDA and then displaying the results in MATLAB. Thanks to the advantages of CUDA in terms of execution time it was possible to create a 2-dimensional model of the rabbit SAN.

- This model was finally used to evaluate the synchronization and the mutual influence between cells. After having performed the randomization of all maximal conductances, we evaluated the effects on cycle length and action potential amplitude. Different levels of resistive coupling between cells and inter-cellular variability of conductances were tested.

The simulations made using the tissue model show how cells synchronize their discharge frequency, and in particular the ultimate value of discharge frequency increases with the coupling resistance and the inter-cellular variability. The action potential amplitude has a behavior similar to that of the cycle length, decreasing as coupling resistance and inter-cellular variability increase, although less markedly in the last case.

The first chapter describes the cardiac conduction system, the properties of the sinoatrial node and the action potential of his cells and then the principles of the cardiac modeling, with particular reference to the inter-cellular electrical propagation, are explained. The second chapter discusses the implementation of the models used in this thesis and the performed simulations. The last chapter finally outlines the results obtained from the simulations with particular attention to 2D simulations, primary goal of this work.

Chapter 1

Sinoatrial node: physiology and mathematical modeling

In this first chapter we introduce the background knowledge which has been necessary to carrying out the work of thesis. We start with a brief description of the cardiac conduction system, focusing on the sinoatrial node, its physiological role and its anatomy. Then, talking about the cardiac electrical activity, we describe the cardiac action potential, making distinction between different cell types and focusing on the pacemaking activity and on the relation between action potential features and heart rhythm and the problem of cardiac arrhythmias. The second part explains the bases of the mathematical modeling in cardiac field, giving particular attention to the models studied for our project and to the mathematical representation of the anisotropic electrical propagation in cardiac physiology.

1.1 The cardiac conduction system

It was two centuries ago when for the first time Galvani and Volta proved spontaneous heart contractions to be related to electrical phenomena. These last events which take place within the heart give rise to the normal cardiac contraction and their alterations can cause severe cardiac rhythm disorders. The nervous system can command several heart properties, like its frequency and its force of contraction but heart functionality does not rely on its innervation. A denervated and transplanted heart is still able to work and adapt itself to different circumstances and this ability is due to some cardiac tissue intrinsic properties, its *automaticity*, i.e. the ability to autonomously initiate the heart beat and its *rhythmicity*, i.e. regularity of this autonomous activity [24].

The electrical signal coordinate the mechanical activity of the heart, a four-

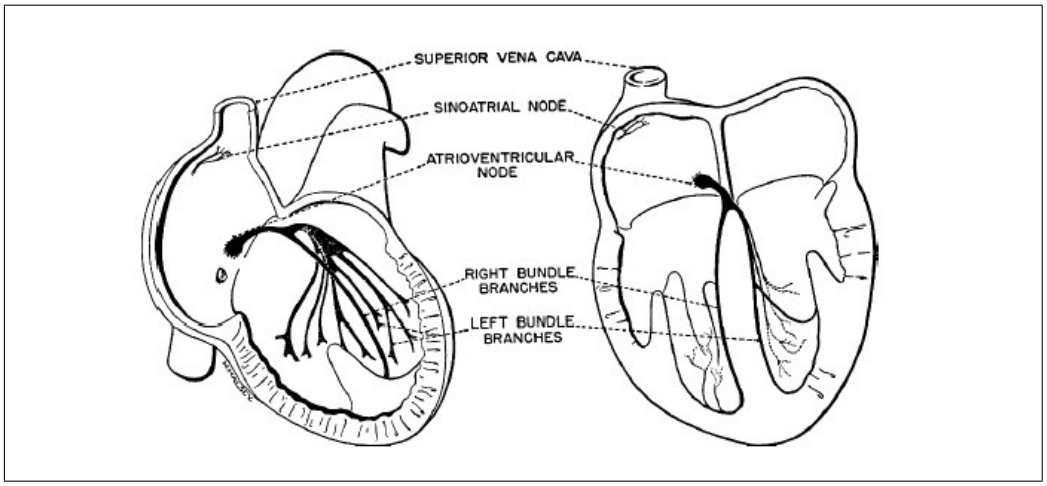

Figure 1.1: Scheme of the cardiac conduction system [20].

chambered organ, consisting of two pumps: the right heart, which drives blood through the lungs and then back to the heart, forming the pulmonary circulation, and the left heart, that is responsible for the systemic circulation, driving the oxygenated blood around the body.

Cardiac tissue is a syncytium of cardiac muscle cells, each of which has a contractile capability similar to that of the skeletal muscle. Besides being contractile, cardiac cells are excitable, enabling action potentials to propagate, and the action potential causes the cells to contract, thereby enabling

the pumping of blood. We will discuss in detail later the features of cardiac action potential while for now we focus on the pathway of propagation of the electrical activity of the heart, shown in Fig. 1.1. It is initiated in a collection of cells known as the *sinoatrial node* (SAN) located just below the superior vena cava on the right atrium. The cells in the SAN are autonomous oscillators and the action potential generated by these cells is propagated through the atria by the atrial cells. The conduction of action potentials between atria and ventricles is normally prevented by a septum composed of non excitable cells and the action potential can continue its propagation only through a group of cells, known as the atrioventricular node (AVN) and located at the base of the atria. After having passed quite slowly through the AVN, the action potential propagates through the bundle of HIS composed by a specialized collection of fibers, named Purkinje fibers which spread via tree-like branching into the left and right bundle branches throughout the interior of the ventricles, ending on the endocardial surface of the ventricles. At this point the action potentials activate the ventricular muscle and propagate through the ventricular wall outward to the epicardial surface. The process whereby an electrical stimulus is converted into muscle contraction in ventricular cardiomyocytes is called *Excitation-contraction (EC) coupling* and its fundamental steps are shown in Fig. 1.2. L-type Ca^{2+} channels open as a result of the depolarization of the T-tubule by the action potential and they lead to an inward flow of Ca^{2+} ions (I_{Ca}). The amount of Ca^{2+} entered the cell induces the sarcoplasmic reticulum (SR) to release additional Ca^{2+} through ryanodine receptors (RyR). This process is named *Ca^{2+}-induced Ca^{2+} release (CICR)*. The released amount of Ca^{2+} diffuses through the myoplasm, binds to the myofilaments and causes contraction. It is then eventually removed from the myoplasm by ATPases, which pump the Ca^{2+} into the SR (sarcoplasmic reticulum Ca^{++}-ATPase (SERCA)) or out of the cell, or by the Na^{+}-Ca^{2+} exchanger (NCX), which transfers Ca^{2+} to the outside of the cell. Phospholamban (PLB) is a protein and the major substrate for the cAMP-dependent protein kinase (PKA) in cardiac muscle. In the unphosphorylated state it works as an inhibitor of the sarcoplasmatic calcium pump SERCA, while when phosphorylated by PKA its ability to inhibit SERCA is lost. When phospholamban is not phosphorylated, contractility and rate of muscle relaxation are decreased and this lead to decreasing stroke volume and heart rate, respectively. On the contrary, in case of sympathetic stimulation, for example, activators

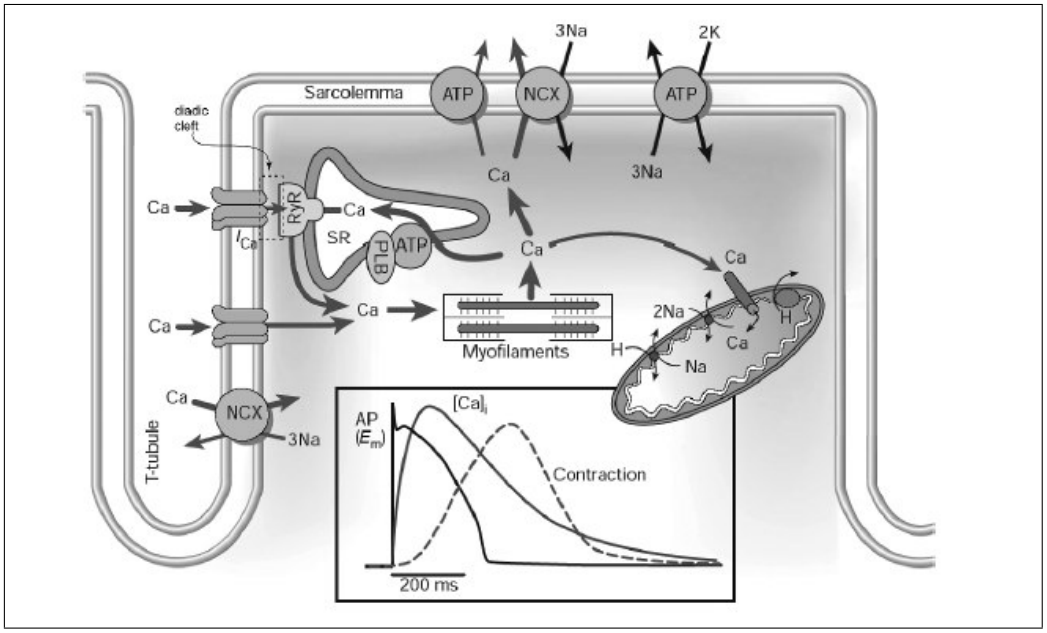

Figure 1.2: Scheme of the major Ca^{2+} fluxes underlying Excitation-contraction coupling in ventricular cardiomyocytes. The inset shows the time courses of the action potential (AP), the Ca^{2+} transient, and the contraction. AP happens first, followed by the Ca^{2+} transient and then by contraction [20].

of PKA, such as the beta-adrenergic agonist epinephrine can be released and this fact may enhance the rate of cardiac myocyte relaxation. In addition, since SERCA is more active, the next action potential will cause an increased release of calcium, resulting in increased contraction (positive inotropic effect) [20].

1.1.1 Anatomy and functions of the SAN

As mentioned before, the sinoatrial node, located in the right atrium, serves as the primary site for initiation of the normal heartbeat (sinus rhythm). It was discovered over a century ago by Arthur Keith and Martin Flack as an anatomically defined tissue at the junction of the superior vena cava and right atria and today is recognized to be a distributed and heterogeneous complex adjacent to the crista terminalis with distinct regions defined by unique electrophysiological and structural properties. The SAN has a cres-

cent shape, it is 15 mm in length and 5 mm in width. It can be divided in two parts, one larger placed in the upper part of the right atrium and named head and one thinner, named tail and placed in the inferior right part [13]. In large animals, including the human, the node is functionally

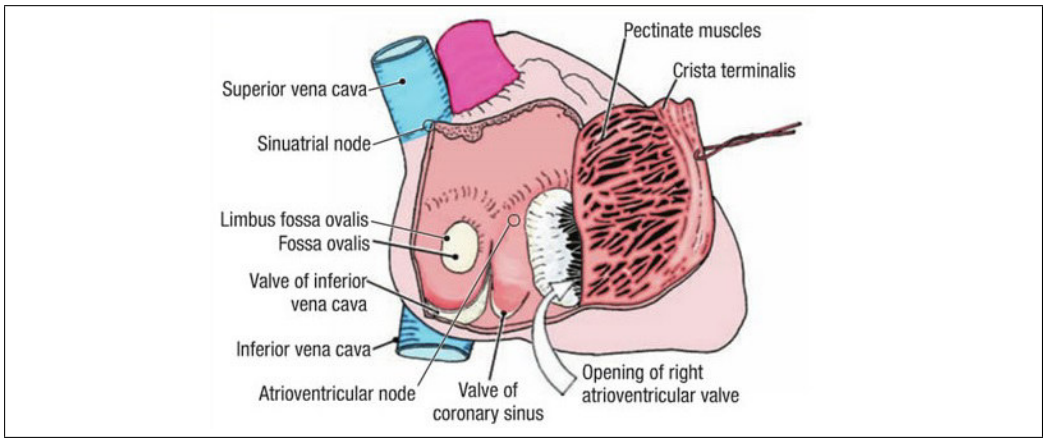

Figure 1.3: Anatomy of the right atrium [3].

isolated with the exception of exit pathways that allow for communication between the SAN and atrial tissue. This insulation is made possible through the presence of a connective tissue, which is able to insulate the pacemaker's automaticity from the hyperpolarizing electrical activity of the atrial myocardium. Another important feature of SAN found in multiple species is the presence of a centrally located artery around which the SAN cells are organized. These cells, weakly coupled between them, are heterogenous, in fact they comprise both pacemaker cells and non-pacemaker cells, such atrial myocytes, adipocytes and fibroblasts. The functional consequences of this great heterogeneity and its importance in our study are further discussed later in this chapter, while here we focus on the morphology of SAN cells. Pacemaker cells found in the sinus node vary by size and electrophysiological properties and can be divided into three different groups:

- *elongated spindle shaped cells*: cells which extend up to 80 μm in length and have a slightly striated cell body with one or more nuclei;
- *spindle cells*: cells which have a similar shape to that of elongated spindle cells, but are shorter in length, extending up to 40 μm and are predominantly mono-nucleated;

- *spider cells*: cells which have irregularly shaped branches with blunt ends.

There is no clear understanding of the different cell types distribution, but experimental studies have shown that none of the three cell types have been found exclusively in a specific SAN area. In particular in the case of the rabbit SAN, which is our case of study, a uniform distribution of all the three pacemaker cell types has been observed in the central area.

Fig. 1.4 shows a schematic representation of these three types of cells. Also a representation of an atrial cell is illustrated, in fact in the crista terminalis region, atrial cells are the predominant cell type, together with a smaller

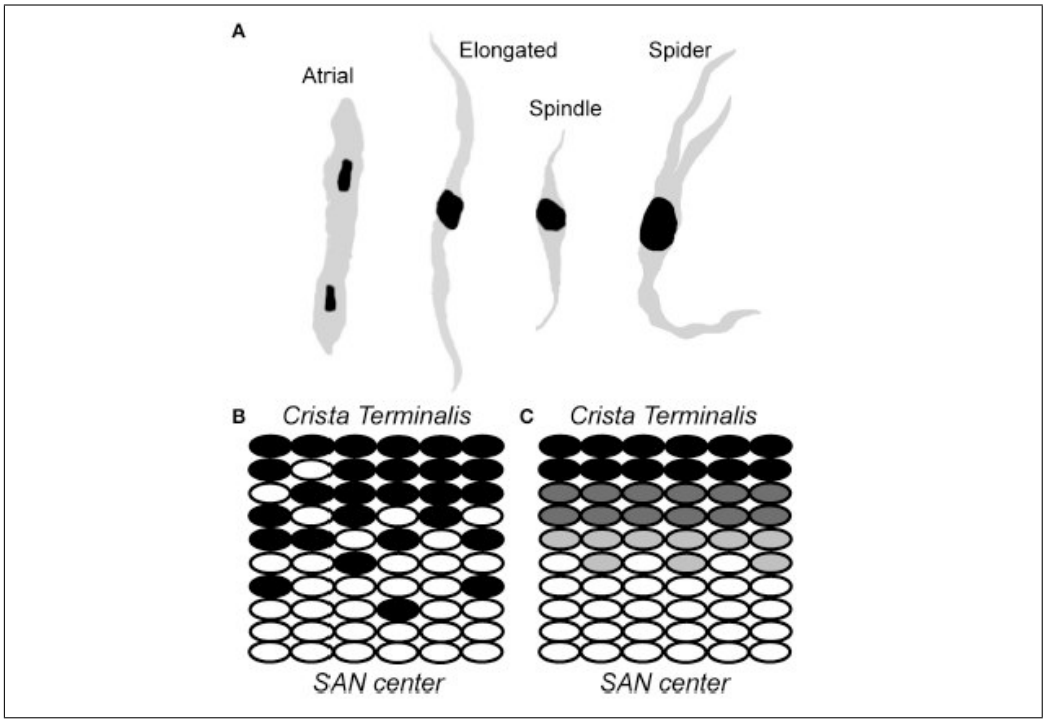

Figure 1.4: (A) Scheme on different types of SAN cells; (B)-(C) Schematic representation of two different models proposed to describe the transition from typical/central SAN (white ovals) to atrial (black ovals) cells: in (C) a gradual transition of intermediate cells (gray) from SAN center to crista terminalis is taken into account while in (B) a mosaic model is illustrated, the latter involves gradually decreasing ratio of SAN to atrial cell densities from SAN center to crista terminalis [32].

percentage of spindle nodal cells. Indeed the septal area of SAN is mostly composed of atrial cells and an almost uniform amount of the other three types of cells is interspersed between them. In the figure we can also see two different models proposed to describe the distribution of these different types of cells within the SAN. We will further discuss these two models later in this chapter [32].

1.1.2 The cardiac action potential

The electrical signal whose propagation has been briefly described before is named action potential and its general shape and phases for a ventricular cell are illustrated in Fig. 1.5. It is represented as the membrane potential waveform and it can be simply defined as a momentary change in electrical potential on the surface of a cell, especially of a nerve or muscle cell, that occurs when it is stimulated, resulting in the transmission of an electrical impulse. As in other cells, the cardiac action potential is a short-lasting event in which the difference of potential between the interior and the exterior of each cardiac cell rises and falls following a consistent trajectory, created by a sequence of ion fluxes through specialized channels in the membrane of cardiomyocytes that leads to cardiac contraction. While there are some differences in the action potentials of various types of cardiac tissue, discussed below, the following model is most commonly used for education purposes. Looking at the figure, we can distinguish five different stages:

- phase 4: *resting phase*. The typical resting potential in a cardiomyocyte is -90 mV, in this condition Na^+ and Ca^{2+} channels are closed.

- phase 0: *depolarization*. When an action potential triggered in a neighbouring cardiomyocyte or pacemaker cell causes the potential to rise above -90 mV, fast Na^+ channels start to open and Na^+ can enter the cell, further raising the membrane potential. The large Na^+ current rapidly depolarizes the potential to 0 mV and slightly above 0 mV for a transient period of time called the *overshoot*, then fast Na^+ channels close. Another type of channels playing during this phase is L-type (long-lasting) Ca^{2+} channels which open when the membrane potential is greater than -40 mV and cause a small but steady influx of Ca^{2+}.
  After completion of depolarization, the cell begins to repolarize, or

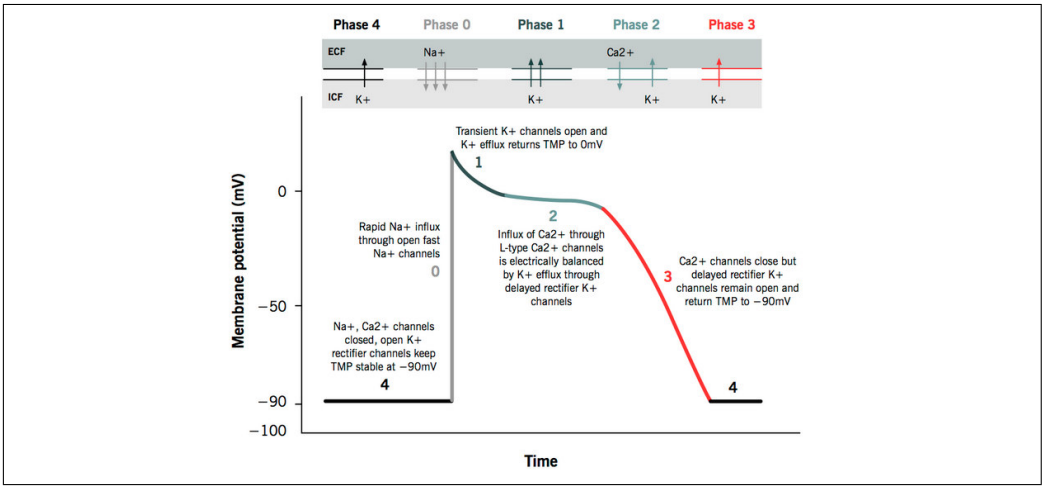

Figure 1.5: Action potential waveform in a ventricular cell, (TMP=transmembrane potential) [18].

return to its original resting state. The cell can not depolarize again until this happens. Phases 1-3 are the repolarization phases and coincide with the time that the cell is refractory and can not respond to a new stimulus.

- phase 1: *early repolarization*. At the beginning of this phase the membrane potential is slightly positive, then some K^+ channels open briefly and an outward flow of K^+ returns the potential to approximately 0 mV.

- phase 2: *plateau phase*. This is the distinguishing phase of the cardiac AP and it is cause of its long lasting. During this stage there is an equilibrium between inward and outward currents, in fact L-type Ca^{2+} channels are still open and there is a small, constant inward current of Ca^{2+}, and different types of K^+ outward currents. The inward and outward currents are electrically balanced, so that the membrane potential is maintained at a plateau just below 0 mV throughout phase 2.

- phase 3: *repolarization*. This phase starts with the gradual inactivation of Ca^{2+}. The outflow of K^+ instead is still present and now, exceeding Ca^{2+} inflow, brings the membrane potential back towards

resting potential of -90 mV to prepare the cell for a new cycle of depolarization.

The described ionic flows change the normal transmembrane ionic concentration gradients, that must be restored, especially by returning Na^+ and Ca^{2+} ions to the extracellular environment, and K^+ ions inside of the cell. This is made primarily through the sarcolemmal Na^+-Ca^{2+} exchanger, Ca^{2+}-ATPase and Na^+-K^+-ATPase [15].

The primary cardiac cell types are: nodal cells (SAN cells and AVN cells),

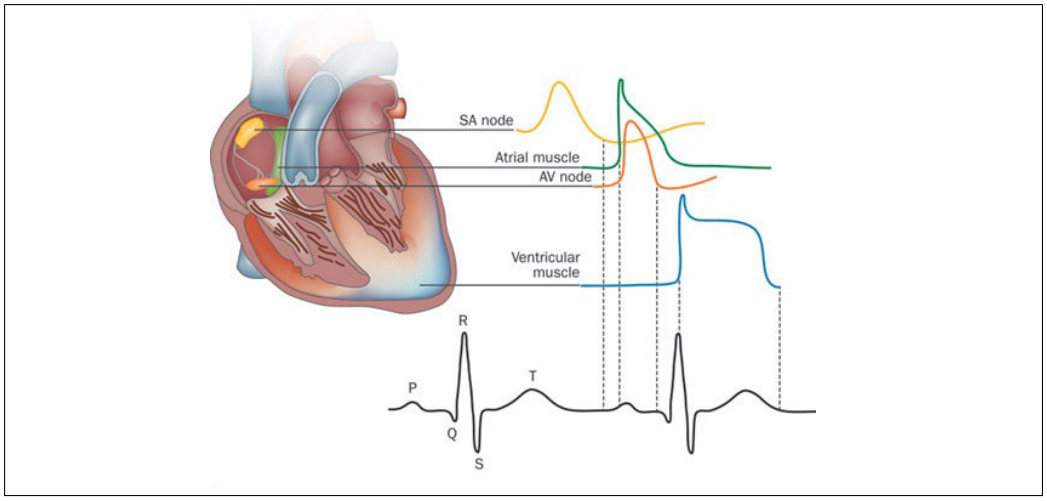

Figure 1.6: Action potential waveform throughout the heart [15].

Purkinje fiber cells, and atrial and myocardial cells. Each cell type has a slightly different function, we already know that the primary function of SAN cells is to provide a pacemaker signal for the rest of the heart, the AVN have to transmit the electrical signal from atria to ventricles with a delay, Purkinje fiber cells are responsible for fast conduction, for the activation of the myocardium, and finally myocardial cells, both atrial and ventricular, are muscle cells, so that they are both contractile and excitable. The different functions of different cardiac cell types lead these cell to have different action potential shapes, however all are noticeably different than the neural action potential, in particular they have a long plateau phase which facilitates and controls muscular contraction and cannot be found in the neural cells. Fig. 1.6 shows typical action potentials for several cell types.

The action potential for SAN cells is the shortest, while both Purkinje fiber

cells and myocardial cells have substantially prolonged action potentials (300 - 400 ms compared to 3 ms for the squid axon). Even within a single cell type, there can be substantial variation. For example, in the ventricles, epicardial, midmyocardial, and endocardial cells have noticeable differences in action potential duration [20].

1.1.3 Pacemaker action potential

Cells of the sinoatrial node, on which we focus, have the property of automaticity thanks to their unique electrophysiological profile, that is distinct from that in atrial or ventricular cells. These latter are characterized as having a stable rest potential, while the SAN AP lacks a true resting potential due in large part to lack of the inward rectifier K^+ channel I_{K1}. The SAN AP reaches a maximum diastolic potential (MDP) of about -60 mV, followed by a spontaneous depolarization that eventually reaches threshold to generate another AP, therefore the SAN is able to generate regular, spontaneous action potentials. Unlike most other cells that elicit action potentials (e.g., nerve cells, muscle cells), the depolarizing current is carried primarily by a relatively slow, inward Ca^{2+} current instead of by fast Na^+ currents. In fact pacemaker cells have fewer inward rectifier K^+ channels than do other cardiomyocytes, so their membrane potential is never lower than -60 mV. As fast Na^+ channels need a transmembrane potential of -90 mV to reconfigure into an active state, they are permanently inactivated in pacemaker cells so there is no rapid depolarization phase. Pacemaker cells have an unstable membrane potential and their action potential is not usually divided into the same defined phases seen before.

Fig. 1.7 shows a typical AP of a SAN cell, the different phases and the involved currents are indicated. The sequence of events for pacemaker action potential are:

- phase 4: spontaneous depolarization, which leads the membrane potential to overcome the threshold level. At the end of the repolarization, when the membrane potential is really negative, i.e. the *maximum diastolic potential* (MDP=the lowest membrane potential reached by the cell)) is about -60 mV, the *funny current* (I_f) is activated. The latter is a distinguishing current of the nodal tissue, it is carried both by Na^+ and K^+ ions and it is activated by repolarization/iperpolarization, which starts at about -50 mV. The reversal

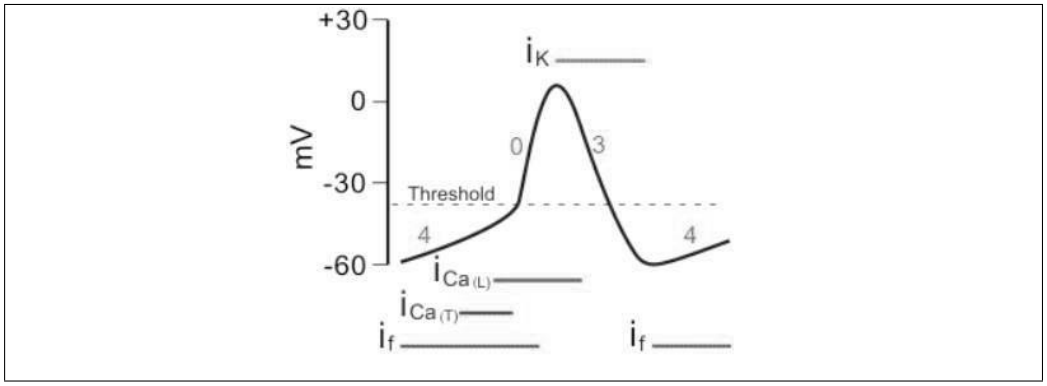

Figure 1.7: SAN cell Action Potential and currents involved in the different phases [21].

potential of the involved ionic channels is -10/20 mV, due to the their mixed permeability. The activation of this current, together with the increased activity of the sodium-calcium exchanger is responsible of the *diastolic depolarization* (DD). In particular this phase can be divided into two stages, the first one, called *early diastolic depolarization* primarily relies on the presence of the funny current, while the exchanger is more active in the second part, *late diastolic depolarization*, immediately before systole. At the end of this phase the membrane potential is closer to the threshold, so almost ready for the upstroke;

- phase 0: depolarization of the AP (*upstroke*). Once the membrane potential has reached the threshold level (about -50/-40 mV), a rapid depolarization can start and move the membrane potential to a peak of about +20 mV. During this phase the conductivity of calcium channels increases and we have two different types of inward currents, I_{CaL}, which is the largest contributor to the upstroke, and I_{CaT}, which is the first one to be activated (already at the end of phase 4). So the primary current during this stage is carried by Ca^{2+} flowing through L-type channels, for this reason the slope of phase 0, and so the speed of depolarization, is lower than the one found in the other cardiomyocytes;

- phase 3: repolarization. Once the membrane potential has reached positive values, K^+ channels start to open and we have two types of

currents: I_{Kr} and I_{Ks}, which, together with the inactivation of I_{CaL}, return the membrane potential to negative values.

1.1.4 Heart rhythm and cardiac arrhythmias

Cardiac arrhythmias are disruptions of the normal cardiac electrical cycle and they are generally of two types. There are temporal disruptions, which occur when cells act out of sequence, either by firing autonomously or by refusing to respond to a stimulus from other cells, as in AVN block or a bundle branch block. A collection of cells that fires autonomously is called an ectopic focus. These arrhythmias cause little disruption to the ability of the heart muscle to pump blood, and so if they do not initiate some other kind of arrhythmia, are generally not life-threatening.
The second class of arrhythmias are those that are *reentrant* in nature and can occur only because of the spatial distribution of cardiac tissue. If they occur in the ventricles, reentrant arrhythmias are of serious concern and life-threatening, as the ability of the heart to pump blood is greatly diminished. Reentrant arrhythmias on the atria are less dangerous, since the pumping activity of the atrial muscle is not necessary to normal function with minimal physical activity, although long-lived atrial reentrant arrhythmias are known to increase the chance of strokes. A reentrant arrhythmia is a self-sustained pattern of action potential propagation that circulates around a closed path, reentering and reexiting tissue as it goes. A classic example of a one-dimensional reentrant rhythm of clinical relevance is one in which an action potential circulates continuously between the atria and the ventricles through a loop, exiting the atria through the AV node and reentering the atria through an accessory pathway (or vice versa). Reentrant patterns which are not constrained to a one-dimensional pathway are much more problematic. The two primary reentrant arrhythmias of this type are tachycardia and fibrillation. Both of these can occur on the atria (atrial tachycardia and atrial fibrillation) or in the ventricles (ventricular tachycardia and ventricular fibrillation). When they occur on the atria, for the reasons mentioned before, they are not immediately life-threatening, while when they occur on the ventricles, they are life-threatening. Ventricular fibrillation is fatal if it is not terminated quickly. Tachycardia is often classified as being either monomorphic or polymorphic, depending on the assumed morphology of the activation pattern. Monomorphic tachy-

cardia is identified as having a simple periodic ECG, while polymorphic tachycardia is usually quasi-periodic, apparently the superposition of more than one periodic oscillation. A typical example of a polymorphic tachycardia is called *torsades de pointes*, and appears on the ECG as a rapid oscillation with slowly varying amplitude (Fig. 1.8). Stable monomorphic ventricular tachycardia is rare, as most reentrant tachycardias degenerate into fibrillation [20].

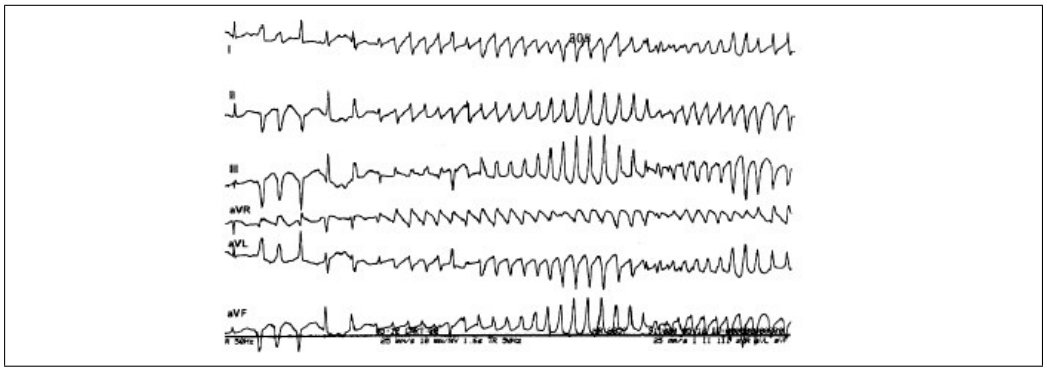

Figure 1.8: A six-lead ECG recording of torsades de pointes [20].

1.2 Mathematical modeling of cardiac AP

Most part of the work made in cardiac modeling field derives from the experiments performed by Hodking and Huxley in the early '50s. They were working on the squid giant axon using for the first time a technique, named *patch-clamp*, which then became crucial during the following years. With this technique the cells and their ionic channels are given particular voltage or current trajectory and then the resulting electrical activity is recorded. We can use different protocols, depending on the shape or type of trajectory used to stimulate the cell. The most common examples are: voltage clamp, current clamp and AP clamp. The latter does not use a fixed value of voltage or current but the cells are stimulated by a voltage profile reconstructed on the basis of a normal action potential. In their pioneering work, Hodking and Huxley recorded multiple electrical currents across the membrane and they identified an electric analogy to describe the cell membrane. From the circuit shown in Fig. 1.9 it derives that the

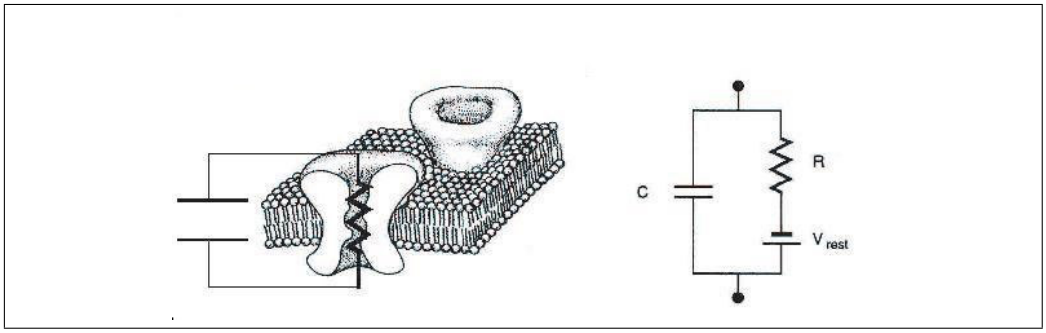

Figure 1.9: Electric analogy used to represent the cell membrane, the phospholipid bilayer is described as an electric capacitance C and the ionic channels through an electric resistance R [4].

currents playing a role in the action potential generation can be described mathematically by the following equation:

$$C_m * \frac{dV_m}{dt} = -I_m \tag{1.1}$$

where C_m is the electric capacitance of the cell, V_m is the voltage difference across the cell membrane and I_m is the sum of all the currents flowing across the cell membrane. In turn each one of these currents is described mathematically by the following equation:

$$I_{ion} = x * g_{ion} * (V_m - E_{ion}) \tag{1.2}$$

where:

- x is the gating variable, i.e. an adimensional value in the range $0 \div 1$, which represents the ratio of open channels at a certain instant in time. We could have more than one gating variable to represent a single channel. The mathematical description of a gating variable is discussed later;

- g_{ion} is the electrical conductivity of the ionic channel;

- V_m-E_{ion} is called *driving force*, E_{ion} is the reversal potential of the considered ion (also known as the Nernst potential), i.e. the membrane potential at which there is no net flow of that particular ion from one

side of the membrane to the other. E_{ion} is determined by the Nernst equation:

$$E_{ion} = \frac{R * T}{Z * F} * log \frac{[ion]_i}{[ion]_o} \tag{1.3}$$

where:

- ion is the considered specific ion;
- R=8.314472 J K^{-1} mol^{-1}, universal gas constant;
- T temperature in Kelvin;
- Z valency of the element;
- F=96485.3399 C mol^{-1}, Faraday constant;
- $[ion]_i$ intracellular ion concentration;
- $[ion]_e$ extracellular ion concentration.

Gating variables are described by differential equations. Conductivity values reached by some channels at fixed potential values are determined by performing clamp experiments, until the maximal conductivity is reached. These found values are then expressed in relative terms, giving a value to the gating variable for each one of the tested potentials. The resulting parameter is named x_∞ and it is defined by a voltage dependent equation. The ionic channels require a certain amount of time to reach the x_∞ value (*steady state* value), so that we can define a time constant τ_x, which is still voltage dependent. The equations for both parameters come from an interpolation of experimental data and they are the following:

$$x_\infty = \frac{\alpha}{\alpha + \beta} \tag{1.4}$$

$$\tau_x = \frac{1}{\alpha + \beta} \tag{1.5}$$

where α and β decribe the relationship with voltage, usually in an exponential form. Finally the expression for the voltage gating is the following:

$$\frac{dx}{dt} = \alpha * (1 - x) - \beta * x \tag{1.6}$$

An equal alternative way is:

$$\frac{dx}{dt} = \frac{x_\infty - x}{\tau_x} \tag{1.7}$$

During the years a great number of cardiac cell models have been formulated, in this work we focus upon rabbit SAN cell models.

1.2.1 Single cell model of the rabbit sinoatrial node

The first sinoatrial node cell model was published in 1980 by Yanagihara, Noma and Irisawa. The model uses a Hodgkin-Huxley formulation and includes five trans-membrane currents: the Na^+, slow inward (Ca^{2+}), delayed rectifier K^+, hyperpolarization-activated, and time-independent leak (background) currents. In this model, the slow inward current is responsible for the rising phase of the action potential and the plateau, determined by both slow inward current inactivation and activation of the dynamic K^+ current [19].
After this first one, many other models have been proposed with the aim of including always more details. We carefully describe two of the most recent rabbit SAN cell models: *Maltsev* and *Severi* models, because they are the ones from which we start the implementation of our models and also because they describe the diastolic depolarization phase through two different hypotheses: *membrane clock* and *calcium clock*.
The properties of funny channel seem specifically apt to generate the diastolic depolarization phase of the action potential, which is the phase responsible for normal spontaneous activity. Because the membrane ionic channels open and close according to the membrane potential, this process is referred to as a membrane voltage clock. As mentioned before, I_f activates upon hyperpolarization, one of the unusual features which at the time of its discovery made the current deserve the attribute funny, at a threshold of about -40/-50 mV, and is fully activated at about -100 mV. In its range of activation, which quite properly comprises the voltage range of diastolic depolarization, the current is inward, its reversal occurring at about -10/-20 mV. This is due to the mixed Na^+/K^+ permeability. The channels are encoded by the hyperpolarization-activated, cyclic nucleotide-gated (HCN) channel gene family. Of the four known HCN subunits, HCN4 is the most highly expressed in the mammalian SAN. The major role of I_f has been

reinforced by the fact that mutations in the I_f channel are associated with a reduced baseline heart rate, and drugs, which block I_f (such as ivabradine) do the same. Funny channels are so a successful target of specific heart-rate-reducing agents like ivabradine, and is known that funny current plays a key role on the pacemaking activity of heart cells.

The second hypothesis relies on the recent finding of spontaneous Ca^{2+}

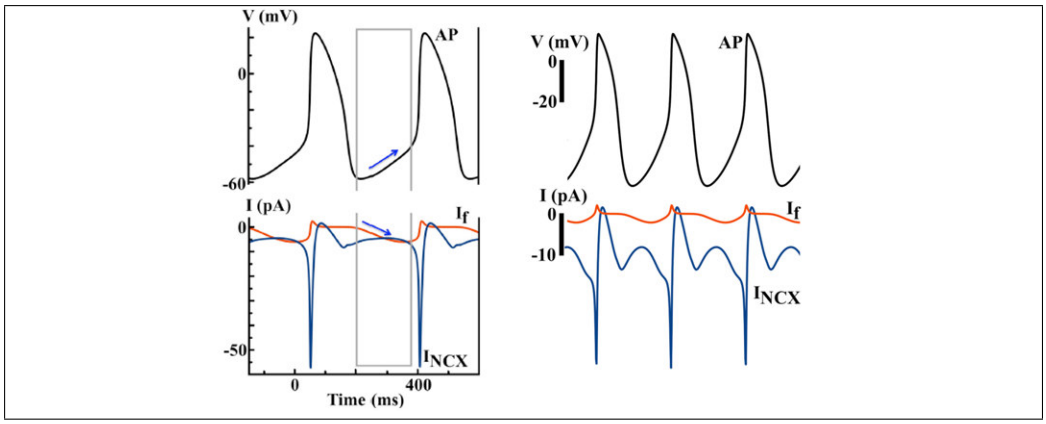

Figure 1.10: In the left part of the figure: membrane clock (I_f) hypothesis: Simulated AP using the Severi model. In the right part of the figure: calcium clock hypothesis: Simulated AP using the Maltsev model [29].

release from the sarcoplasmic reticulum (SR), which has been suggested to be the mechanism for sinus rhythm generation. When the SR is full, the probability of spontaneous Ca^{2+} release increases. On the other hand, when the SR is empty, the chances for spontaneous Ca^{2+} release decrease. The rhythmic alteration of SR Ca^{2+} release is referred to as the Ca^{2+} clock. Because the SR Ca^{2+} content is controlled in part by the membrane voltage, it is important to recognize that the activation of the Ca^{2+} clock and the membrane ionic clock are interdependent. Based on evidence from isolated SAN myocytes, late diastolic Ca_i elevation (LDCAE) relative to the action potential upstroke is a key signature of pacemaking by the Ca^{2+} clock. On these bases a new modeling approach starts to be considered, in which rate regulation is governed by intracellular Ca^{2+} cycling and its coupling to the surface membrane. These new models differ from classical pacemaker models, which attribute the rate regulation of SAN cells mainly to the I_f activation state and its relation to the early DD slope. The rate and amplitude of SR Ca^{2+} cycling of the most recent models is controlled

by the amount of free Ca^{2+} in the system, the SR Ca^{2+} pumping rate and the numbers of activated Ryanodine Receptors (RyR) [14].

1.2.2 Maltsev model

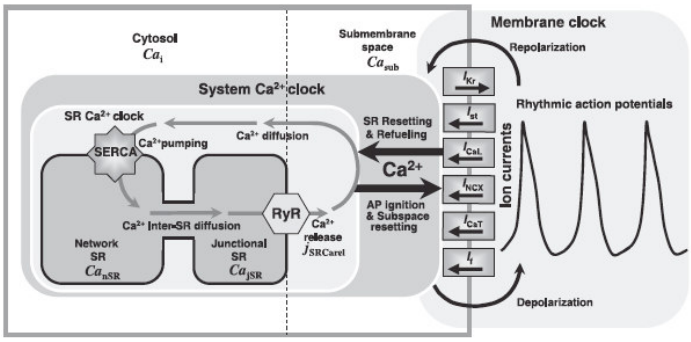

Figure 1.11: Schematic diagram of the interacting Ca^{2+} clock and membrane clock in the Maltsev model of rabbit sinoatrial node cells [29].

In contrast to earlier membrane-delimited models of the SAN cell, in 2009, Maltsev and Lakatta first developed a model of a coupled membrane- and Ca^{2+}-clock to address the ionic mechanism of cardiac pacemaking. Importantly, they quantitatively described the contribution of spontaneous Ca^{2+} release during late DD to SAN pacemaking, by triggering an inward Na^{+}-Ca^{2+} exchange current (I_{NCX}). They concluded that only a coupled system of membrane- and Ca^{2+}-clocks offers both the robustness and flexibility that are required to maintain normal pacemaking function. The Maltsev model predicts that the most important pacemaker current during late DD is the inward I_{NCX}, instead of I_f. I_{NCX} is the leading current during the DD, and is secondary to dynamic changes in $[Ca]_i$ and the Ca^{2+}-clock. Without the Ca^{2+}-clock, the membrane-clock alone is not capable of maintaining normal pacemaking function.
This model is a system of 29 first-order differential equations. To avoid a lengthy transitional process, they set initial conditions for membrane clock gating variables to *ready-to-fire* status.

1.2.3 Severi model

In 2012, Severi et al. developed an updated model of a rabbit SAN cell based on the most recent experimental data, with an updated representation of intracellular Ca^{2+} dynamics. I_f remains the major pacemaker current in the Severi model, and it is formulated using a similar Hodgkin-Huxley scheme. As predicted by the Severi model, I_f and the Na^+-Ca^{2+} exchange current I_{NCX} are of similar size during DD. Yet I_f gradually increases to promote depolarization, while the inward I_{NCX} decays slightly over time. In addition, the model predicts that complete blockade of I_f will lead to cessation of cell automaticity. In fact an outstanding improvement with respect to the Maltsev model, is its ability in reproducing the experimental effects of the I_f reduction, in particular the AP rate decrease, experimentally observed in response to a partial I_f blockade. The Severi model reproduces an AP rate decrease of 22% in case of a 66% I_f blockade; on the contrary, Maltsev model is able to reproduce only a small reduction of the AP rate (about 5%) if I_f is completely blocked. I_f is described as composed of two relatively independent Na^+ and K^+ components, I_{fNa} and I_{fK}, whose contributions to the total conductance at normal Na^+ and K^+ concentrations, are similar. I_f is also modulated by the extracellular potassium concentration K_o.
This model is a system of 34 first-order differential equations.

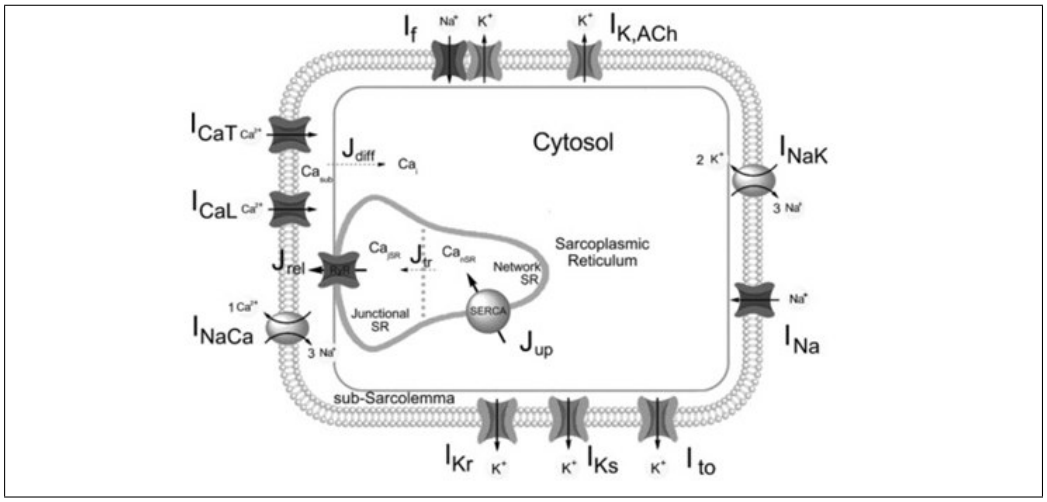

Figure 1.12: Schematic diagram of the Severi model [31].

1.2.4 Electrical propagation and cable theory

In this work of thesis the single cell model is only a starting point, then we will work on systems of coupled cells, either along a line or in a matrix. For this reason in this section we want briefly explain how to mathematically represent an action potential propagating down a uniform cable. First of all the cable must be divided into discrete segments so that we can analyze the cable as coupled equivalent circuits (Fig. 1.13). In the illustration we

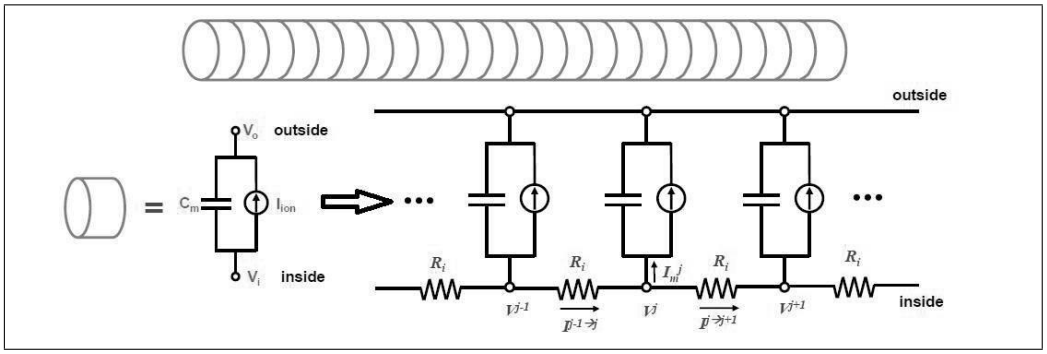

Figure 1.13: One dimensional cable theory [34].

can see the mathematical circuit used to represent a single segment of the cable, which is the same we already described in the previous section. Then this circuit is replicated and coupled to other equivalent circuits to describe all the cable. These circuits are connected between them by a resistance R_i, which corresponds to the intra-cellular resistance (for simplicity we assume extra-cellular resistance R_e=0), while V^j represents the voltage at the jth element of the cable. Said that and using the Ohm's law and the Kirchoff's current law, we can write the equations that describe the jth element of the cable:

$$I^{j-1\mapsto j} = (V^{j-1} - V^j)/R_i \tag{1.8}$$

$$I^{j\mapsto j+1} = (V^j - V^{j+1})/R_i \tag{1.9}$$

$$I^{j-1\mapsto j} = I^{j\mapsto j+1} + AI_m^j \tag{1.10}$$

where I_m is a current density and A is the surface area of the jth element and $I^{j-1\mapsto j}$ is the current flowing from the (j-1)th element to the jth element. Analyzing the equivalent circuit the membrane current (normalized per unit

area) can be written as follows:

$$I_m^j = C_m \frac{dV^j}{dt} + I_{ion}^j \tag{1.11}$$

Now, putting the equations together, we obtain:

$$(V^{j-1} - V^j)/R_i = (V^j - V^{j+1})/R_i + A[Cm\frac{dV^j}{dt} + I_{ion}^j] \tag{1.12}$$

and, rearranging the fields:

$$C_m \frac{dV^j}{dt} = (V^{j-1} - 2V^j) + V^{j+1}/AR_i - I_{ion}^j \tag{1.13}$$

The parameter R_i can be also related to cable geometry, in fact defined ρ_i the intracellular resistivity, a the radius of the transversal section and Δ_x the length of each segment of the cable, we have: $R_i = \rho_i \Delta_x / \pi a^2$.
The approach of the cable theory to describe AP propagation can be easily extended to a multidimensional propagation. In this last case we should consider that transverse propagation means encountering more gap junctions per unit length and thus the resistance in this direction should be greater than the longitudinal one. The resistance value in fact is determined both by cardiac structure and gap junctions, although the cytoplasmic resistance is relatively low compared with the resistance encountered at gap junctions [34].

1.3 Heterogeneity in the SAN

A fundamental limit of single cell models is that they are unable to describe the interaction between cells and the variability normally existent between cells of they same organ. In fact each parameter present in the model is given an average value, which cannot take into account slight differences within the tissue. In this section we describe the great morphological and functional heterogeneity of the rabbit sinoatrial node, pointing out the most relevant aspects to this work of thesis.
Measurements from intact rabbit SAN have shown heterogeneity of electrophysiological properties from the center to the border of the atrium including gradual morphological changes in action potential, a decrease in maximum diastolic potential, an increase in peak overshoot potential (POP), an

increase in upstroke velocity (UV) and a decrease in pacemaker potential slope. On the basis of these results, as described in the work of Oren et al. [28], two distinct hypotheses have arisen to explain intact SAN heterogeneity. The first is that the SAN has two specific cell types, central cells and peripheral cells, each with distinct electrophysiological characteristics. The second hypothesis suggests that all observed heterogeneity in the intact SAN results from electrotonic coupling effects, i.e. cells in the SAN near the atria will be strongly affected and modified by the atrium. However what Oren et al. concluded from their tissue model simulations is that atrial electrotonic effects is plausible to account for SAN heterogeneity, sequence, and rate of propagation. They also have studied the effect of fibroblasts, concluding that they can act as obstacles, current sinks or shunts to conduction in the SAN depending on their orientation, density, and coupling. Some of the most important hypotheses proposed to explain the observed heterogeneity of the rabbit sinoatrial node are summarised below:

- SAN tissue has fibroblasts interspersed in islands that occupy about 50% of SAN volume and they can act as obstacles, current sinks or shunts to conduction in the SAN depending on their orientation, density and coupling;
- Verheijck et al. [11] have observed atrial cells interspersed in the SAN and suggested a mosaic model of SAN and atrial cells for SAN organization;
- Gap junction density and conductance increase moving from the center to the pheriphery of the SAN;
- SAN has two specific cell types, central cells and peripheral cells, each with distinct electrophysiological characteristics;
- All observed heterogeneity in the intact SAN results from electrotonic coupling effects, cells in the SAN near the atria will be strongly affected and modified by the atrium.

Chapter 2

Materials and Methods

Starting from the single cell models for the rabbit sinoatrial node: *Severi model* and *Maltsev model*, first a 1D model both in MATLAB and CUDA and after a 2D model in CUDA were implemented. The 1D model, or cable model consists of an array of cells placed along a line and connected between them and it was used to perform a comparison between the execution time in MATLAB and CUDA, running the same simulations with both softwares. It was also handled to study and better understand the relationship between the cells coupling and the conduction velocity within a cable of cells. The 2D model, or tissue model consists instead of a matrix of cells and each one, except the ones on the boundary, is connected to four other cells. This last model has been useful in introducing in this study the heterogeneity of the SAN through a randomization of all the conductances existent in the Maltsev model, so that each cell had a different behavior and it was possible to evaluate how each one influences the others.
Therefore in this chapter we expose the steps taken for the implementation of the instruments and the last section briefly describes the performed simulations.

2.1 1-dimensional model implementation in MATLAB

The Maltsev and Severi single cell models for the rabbit sinoatrial node, described in details in the previous chapter, were the starting point of my work. In the cable model we consider a group of cells placed along a line and connected between them through gap junctions, therefore, to pass from a single cell model to a cable model, each differential equation existent in the model and describing the evolution in time of a state variable must be replicated for each cell of the cable. In this thesis we have looked at cables of both 25 and 50 cells. The differential equation describing the evolution of the cell's membrane potential was modified according to the cable theory to include the contribution in current each cell gives to the two neighboring cells. No-flux boundary conditions were considered.

The voltage equation that appears in the single cell model as:

$$dV_m = -i_{tot}/C_m \tag{2.1}$$

is then converted to the following:

$$dV_m = -(i_{tot}/C_m) + V_{net}/(R_{gap} * C_m) \tag{2.2}$$

In the previous equations as usual C_m represents the cell's membrane capacitance and it is equal to 32 pF, V_m is the membrane potential, expressed in mV and i_{tot}, expressed in pA, is the sum of all the different currents used in the model, respectively for the Maltsev and Severi model:

$$i_{tot} = i_{CaL}+i_{CaT}+i_f+i_{st}+i_{Kr}+i_{Ks}+i_{to}+i_{sus}+i_{NaK}+i_{NaCa}+i_{b_{Ca}}+i_{b_{Na}} \tag{2.3}$$

$$i_{tot} = i_f+i_{b_{Na}}+i_{b_{Ca}}+i_{Kr}+i_{Ks}+i_{to}+i_{NaK}+i_{NaCa}+i_{Na}+i_{CaL}+i_{CaT}+i_{b_K}+i_{KACh} \tag{2.4}$$

The new terms introduced with the cable model are:

- R_{gap}: this parameter takes into account the coupling strength between the cells. It has the units of measurements of an electrical resistance, increasing its value the current contribution between the cells is decreased according to the Ohm's law. Its importance in this study will be further discussed later in this chapter and in the next one;

- V_{net}: this parameter takes into account the difference in voltage between the cells, therefore it is expressed in mV and it is obtained as follows:

$$V_{net} = V_m([2 : end, end]) - 2 * V_m + V_m([1, 1 : end - 1]) \tag{2.5}$$

The resulting models are composed of systems of *ordinary differential equations* (ODE). MATLAB has a number of tools for numerically solving ODE, in our case we used the built-in function *ode15s*, which is one of the solvers designed for stiff problems, namely when ODE are such that the numerical errors compound dramatically over time. In general in that cases it is necessary to take considerably smaller steps in time to solve the equations, and this can lengthen the time to solution dramatically. Often, solutions can be computed more efficiently using one of the solvers specific for stiff problems.

I first considered the cells governed by the same differential equations and with the same values for the parameters and the initial conditions of the state variables. The cells, having at each moment the same voltage, do not exert any influence on each other, in fact the contribution in current deriving from the Ohm's law is equal to zero, the value given to R_{gap} does not have any effect and the cells behave as a single one.

Figures 2.1 and 2.2 show the results obtained in this condition using the two models, obviously the results are the same considering a cable of either 25 or 50 cells.

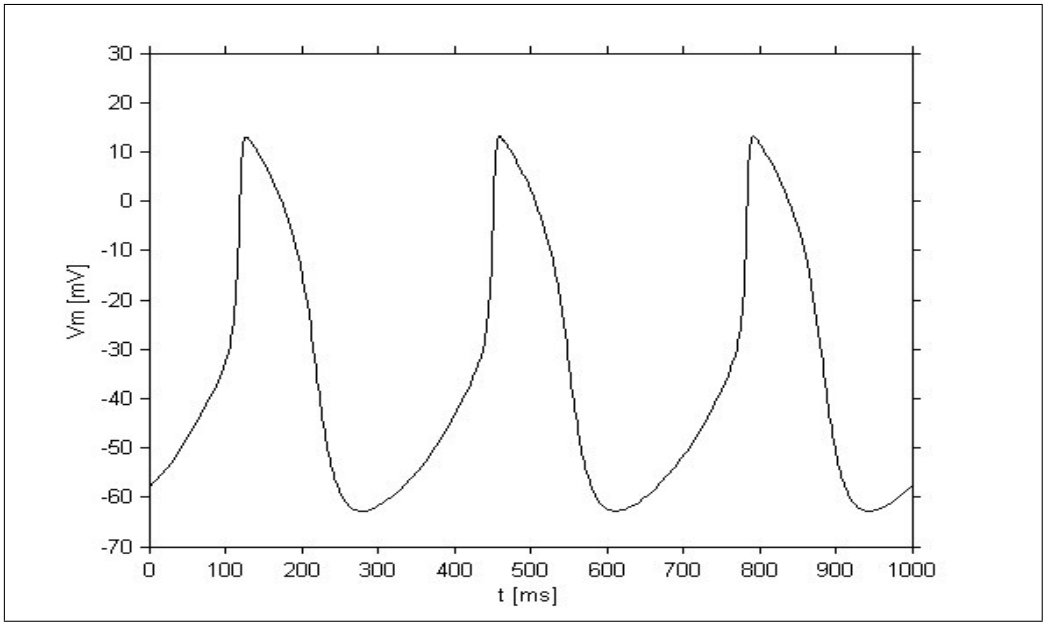

Figure 2.1: Simulated AP using the Maltsev cable model

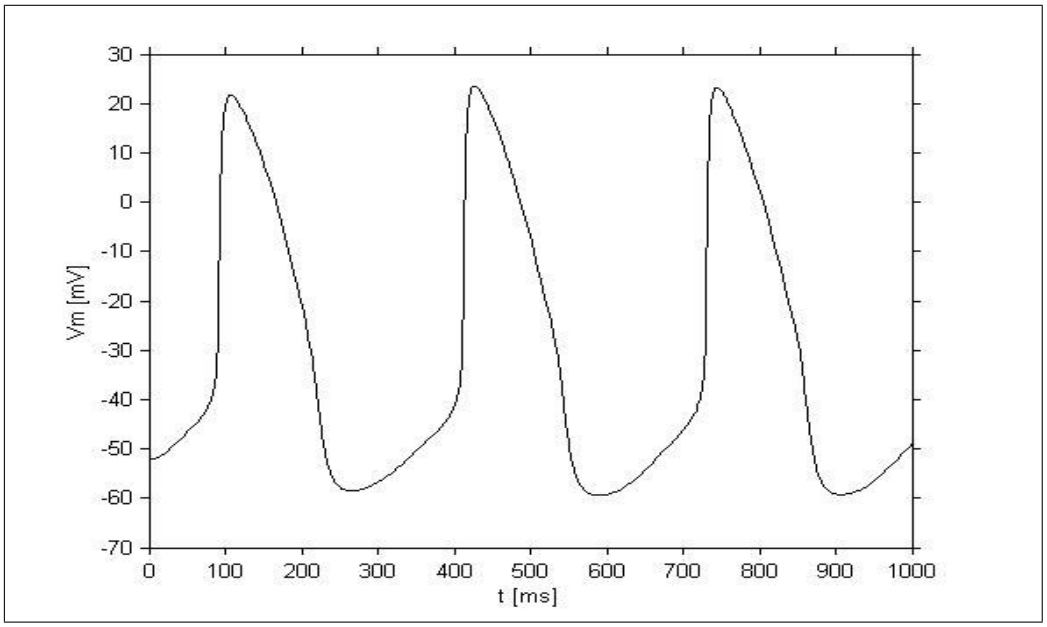

Figure 2.2: Simulated AP using the Severi cable model

Once verified the correct implementation of the models, the initial conditions have subsequently been changed to evaluate the effect of different values of R_{gap} over the conduction velocity within the cable. Each cell was given different initial conditions for the state variables according to hypotheses made referring to the estimated physiological values of conduction velocity in the rabbit SAN. Realistic values for the initial conditions of the models state variables can be determined by using the usual relation between time, displacement and velocity (velocity=displacement/time). In our case the displacement is the distance between two neighboring cells within the cable and, using the relation above, it results that, considering average values for velocity and distance equal respectively to 10 cm/s and 100 μm, the time taken by the impulse to pass from one cell to the next is roughly equal to 1 ms. On the basis of these considerations, the initial conditions were in practice computed running simulations of the single cell models. To assign the initial conditions to the first cell of the line, a 50 s simulation was performed, the final values of the state variables become the initial conditions for this cell and they are stored in the first line of a matrix. Starting with these values as initial conditions, another simulation was performed, but this time the length of the simulation corresponds to the time delay between the two cells, so it is equal to 1 ms. As before the ending values of the state variables become the initial conditions for this cell and the conditions from which to start for the subsequent simulation. Repeating these same steps for the total number of the cells, all the initial conditions are assigned.
The simulations conducted using these instruments and their results are presented later.

2.2 Implementation of the models in CUDA

Although the implementation of the cable model has been an important tool for learning the mechanism of connection between cells and the necessary steps to take to pass from single cell models to models of coupled cells, the primary purpose of my project was the implementation of a tissue model of the SAN. In order to do that we made use of CUDA, that enables huge improvements in computing performances thanks to the power of the graphic processing units (GPU). GPU's parallel structure allows a very effective manipulation of many parallel tasks rather than serial tasks. The code written in CUDA implements both a cable and a tissue model of the SAN, even if the 1D model has only been used to evaluate the advantages introduced in term of execution time comparing to MATLAB. The tissue model in CUDA has been realized considering only the Maltsev model to describe the cells, this choice has been supported by practical time reasons, but the same study could be made considering the Severi model, indeed it would be interesting to compare the results obtained in the two situations.

2.2.1 A look inside the CUDA Programming Model

First of all the program is divided into several phases that are executed on either the host (CPU) or the device (GPU) depending on the amount of data parallelism that they exhibit. Both host and device code are enclosed by the same source code, which is then compiled using the *NVIDIA C Compiler (NVCC)*. At this point the two codes are separated, the host code is compiled with the host's standard C compilers while the device code is further compiled by the NVCC and executed on a GPU device.
Therefore in our case the code, named *Maltsev.cu*, will be compiled from the command line by typing the following sequence:

```
nvcc -arch=sm_13 -o a Maltsev.cu
```

The executable file is then executed through the line:

```
a.exe
```

The device code uses keywords for labeling data-parallel functions, called *kernels*, and their associated data structures. In particular a kernel, defined

as a *computationally intensive function that is performed by threads in parallel*, generates thousands of *threads* to exploit data parallelism.

As described in fig.2.3 the execution of a typical CUDA program starts

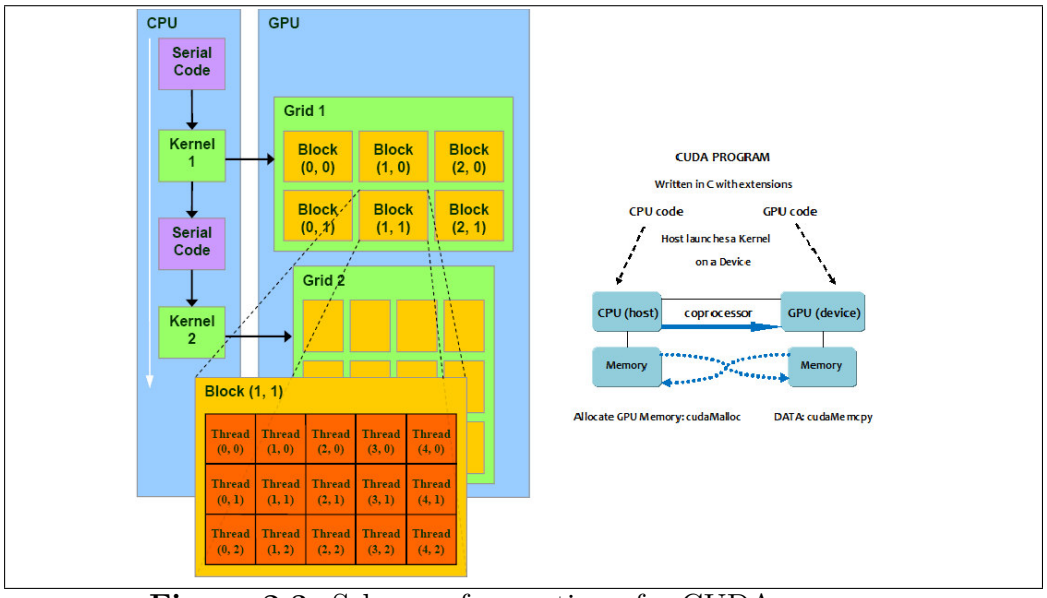

Figure 2.3: Scheme of execution of a CUDA program

with host (CPU serial code) execution, then, if a kernel function is invoked, the execution is moved to the device (GPU parallel kernel). A kernel, during an invocation, as said before, generates a large number of threads and all these collectively form a *grid*. When all threads of a kernel complete their execution, the corresponding grid terminates and the execution continues on the host until another kernel is invoked. Each kernel is involved in the performance of an individual part of the process and inside it specifies the code to be executed by all threads of a parallel phase. The following example shows a kernel code used in our program:

__global__ void initialConditions(double* vars, int num_param, int num_cells, int cells_per_thread) {

double Vm = -57.9639 ;
... other 27 variables are defined ...
double RI = 0.31195 ;

Within each test, the variables are divided as follows Vm(cell1),

```
Vm(cell2), Vm(cell3) ... Vm(cellLast), m(cell1), m(cell2), ... m(cellLast)
... for all 29 parameters
  int idx = threadIdx.x*cells_per_thread;
  int simulations = blockIdx.x;
  int limit = idx+cells_per_thread;
    for (;idx<limit;idx++) {
    vars[(simulations*num_param*num_cells) + idx +(0*num_cells)] = Vm;
    ... other 27 values are assigned ...
    vars[(simulations*num_param*num_cells) + idx +(28*num_cells)] = RI;
    }
}
```

The syntax is ANSI C with some extensions, in fact there is a CUDA specific keyword *__global__*, which starts the code and the declaration of each kernel in general and it indicates that that particular function is a kernel and so it can be called from a host function to generate a grid of threads. Other extensions used in CUDA and shown in the example are the keywords *threadIdx.x* and *blockIdx.x*, that refer, respectively, to the indices of a thread and of a block in which threads are grouped together. These subdivisions and indices allow each thread to work only on the particular part of the data structure for which it is designated. After calculating the starting position in the input vector based on unique block and thread coordinates the computation is then iterated through a loop, in which the indices are given by the thread's coordinate and each element of the vector can be computed in a separate thread. Although in the examined code only one dimension (x) is considered, the threads have a multi-dimensional organization. Briefly threads in a grid are organized into a two-level hierarchy, each grid is formed by one or more thread blocks and all blocks have the same number of threads and are identified through a two dimensional coordinate given by the CUDA specific keywords blockIdx.x and blockIdx.y. Each thread block is in turn structured as a three dimensional array of threads (maximum 512 threads per block), whose coordinates are given by the following indices: threadIdx.x, threadIdx.y, and threadIdx.z.
The following line describes how the kernel function of the previous example is invoked or called in the host code:

```
initialConditions<<<simulations,(num_cells/cells_per_thread)>>>
```

(dev_vars,num_param,num_cells, cells_per_thread);

The two parameters: *simulations* and *num_cells/cells_per_thread*, between the CUDA symbols for the syntax of the call function, set, respectively, the dimensions of grids in terms of number of blocks and the dimensions of blocks in terms of number of threads. In this particular case the number of threads is related to the number of SAN cells in the model. The structure of the kernel functions implemented in the code used in this study and their execution will be discussed in detail in the following section.

Fig. 2.3 also indicates that host and device have separate memory spaces, therefore, in order to execute a kernel on a device, it is necessary to allocate memory on the device and transfer the pertinent data from the host memory to the allocated device memory. Similarly, after device execution, the result data can be transferred from device back to the host and the device memory, no longer needed, can be freed up. The CUDA device memory model comprises *Global Memory* and *Constant Memory*, that the host code can write to and read from. The activities of allocating and de-allocating device Global Memory can be performed using the *Application Programming Interface (API)* functions provided by the CUDA runtime system, between whom the most important are: *cudaMalloc()* and *cudaFree()*. The first one can be called from the host code to allocate a block of memory on the GPU for an array or any other object. In order to use this function, two parameters must be specified: the address of a pointer to the allocated object and the size of the object to be allocated:

cudaMalloc(&dev_vars, sizeof(double)*size);

The function cudaFree() is called after the computation is terminated to free the storage space for the allocated object from the Global Memory and it uses as parameter in input the same pointer used before in cudaMalloc():

cudaFree(dev_vars);

Thanks to one of the API functions provided by CUDA for data transfer between memories, called *cudaMemcpy()*, it is then possible to transfer the desired data from the host to the block of memory allocated on the GPU. In order to use this function, two pointers must be declared: one points to the source data object to be copied and the other one points to the destination

location for the copied object. The other two required parameters state, respectively, the number of bytes to be copied and the types of memory involved in the copy operation, in fact cudaMemcpy() can be utilized both to copy data from a location of the device (or host) memory to another one in the same memory and to transfer data from host memory to device memory and vice versa. We report an example of this function used in our code, in which host_conductances represents a vector of conductances obtained by simulations performed in MATLAB:

```
cudaMemcpy(dev_conductances, host_conductances, size*sizeof(double),
                                    cudaMemcpyHostToDevice);
```

The fourth parameter is specified through a symbolic constant predefined in the CUDA environment. In this example the host code calls the function to copy the vector of conductances from the host memory to the device memory, but simply reversing the order of the pointers and using as fourth parameter the symbolic constant: *cudaMemcpyDeviceToHost*, the considered data can be copied from the device memory to the host memory. The latter modality is useful when the outputs of interest must be read from the GPU to the CPU so they can be available to main() (host code) to be printed or to be exported and then visualized and analyzed.
The code has been written using Visual Studio 2010 and working on Dell Precision T5500 8 Core Workstation, whose features are: Software: Microsoft Windows 7 Professional 64 Bit; Processor: Intel Quad Core Xeon E5504 2.00 GHz CPU Processor - Qty 2 - Total of 8 Cores; Graphics: Nvidia Quadro FX 1800 768MB Video Memory.

2.2.2 Maltsev tissue model

After having discussed the general structure of a CUDA program we want to describe in detail the kernel functions used in our program Maltsev.cu, whose complete code is shown in Appendix A. As said, a kernel function specifies the statements that are executed by each individual thread created when the kernel is launched at run-time. In our case each thread corresponds to a single cell in the model, so that all computations are evaluated in parallel for each cell. When the execution of a kernel is terminated the results computed launching that kernel are updated for each cell and the execution of the following kernel can start.

In the first kernel, that we can find in the code, *initialConditions*, an array of dimension *num_cells*num_param* is assigned to the initial conditions of the state variables. Once executed this kernel, the array, named *vars*, is structured as illustrated in fig. 2.4 and contains in sequence the values of the initial conditions of each state variable for all the cells.

After having initialized the array of the state variables, these latter must be

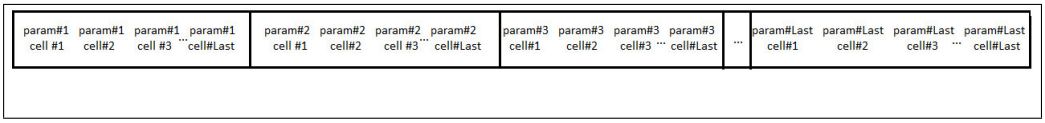

Figure 2.4: Structure of the array initialized with the Initial Conditions

updated for every time step and this is made possible through the execution of the next kernels. The second kernel *computeState*, as the name suggests, calculates the updated values for each of the 28 parameters, while the membrane voltage will be managed by a separate kernel. Through this kernel first the variables are indexed within the array, then all the constant values used in the model are specified and finally the differential equations describing the state variables are integrated using the *Euler Forward Method*. This latter is a first-order numerical procedure for solving ODEs with a given initial value. The following differential equation $\frac{dy}{dt}$=f(t,y) is satisfied by a group of functions, a unique initial condition y(t=0)=y_0 identifies only one of these functions, which is the *Initial Value Problem* (IVP) solution. Then we assume that a unique solution exists and denote that solution by y^e(t), while y(t) refers to the numerically computed solution, which can only be an approximation of the exact one. Denoted the time at the *nth* time-step by t_n and the computed solution at the *nth* time-step by y_n, i.e., y_n ≡y(t=t_n), the constant step size h is then given by h=t_n-t_{n-1}. Given (t_n,y_n), the forward Euler method computes y_{n+1} as:

$$y_{n+1} = y_n + hf(t_n, y_n) \tag{2.6}$$

Therefore we estimate the solution by considering the tangent in (t_n,y_n), approximation of the solution in t_{n+1} is the value in t_{n+1} of the tangent to the approximated solution curve in (t_n,y_n). All this is possible by proceeding iteratively from the initial f_0=f(t_0,y_0) which is note.

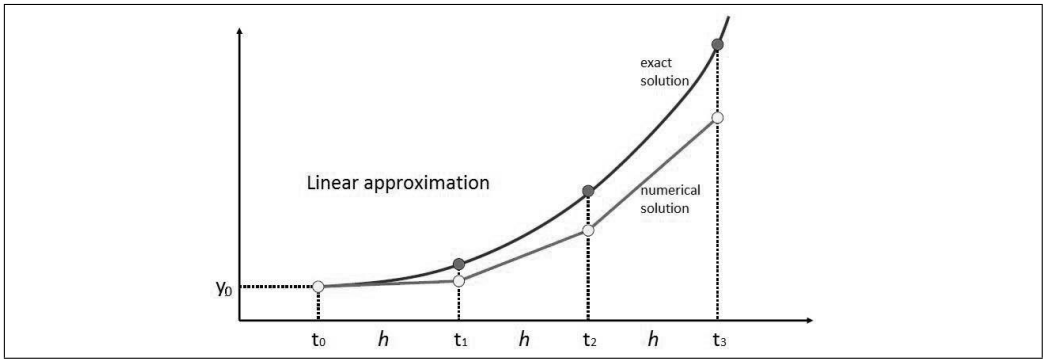

Figure 2.5: Scheme of forward Euler integration

At the end of this function, all the state variables are updated into a temporary array and the next step is to copy the updated variables at this time step in another array, which at the end of the simulation will contain all the values for the state variables along all the time of simulation (kernel *updateState*).
The subsequent kernel computes the new values for the membrane voltage. While the value of each of the other state variables at a certain instant of time and for a certain cell is not affected by the value assumed by the state variables in the other cells, the value of one cell's potential, as we know, is affected by that of neighboring cells. For this reason it is necessary to have a separate kernel for handling the membrane potentials. Within the kernel *compute_voltage* two different implementations are shown, first the case of a cable of cells and then the case of a tissue. As usual the cells are connected between them through gap junctions, but this time the resistance is expressed referring to geometrical parameters of the cells, and so using a resistivity and the dimensions of the cell. The relation between ρ and R_{gap} is given by the second Ohm's law: R=ρ*l/S, i.e. the resistance R of a homogeneous conductor of constant transversal section is directly proportional to its length (l) and is inversely proportional to the area of its transversal section (S). As it was for the resistance, also for the resistivity there is not a fixed value in literature, in our simulations different values within a wide range have been tested. In the cable model the equation used to update the voltage is the same shown before while describing the implementation in one dimension in MATLAB, the only difference is the method of integration. The following illustration and the lines below describe how the

voltage is updated in the case of the tissue model, where no more only two neighboring cells must be considered when calculating one cell's potential, but 4 neighboring cells must be taken into account, as the cells are placed as in a matrix and no more along a line.

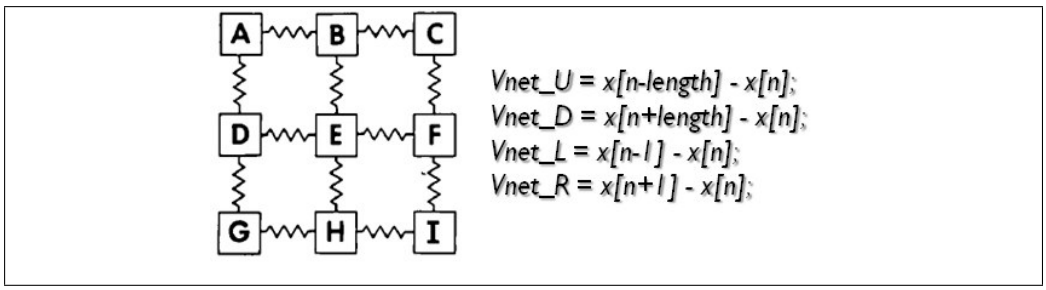

Figure 2.6: Scheme of connection between cells in the tissue model

$$V_m[m] = (x[n])+(step)*((rad^2*\frac{\pi}{(\rho * C_m * deltx)})*(V_{net})-\frac{(I_{ion}[n]+stim)}{C_m}); \tag{2.7}$$

where the following parameters are utilized:

- $V_m[m]$ is the updated membrane potential of the cell whose position in the voltage array is denoted by the index m (m=0,1...num_cell). After having written into V_m the updated values for each cell, these are copied into another array x, so that V_m is only a temporary vector;

- x is the state variable array, which contains the updated values of all the state variables. At the first time step, it is filled with the initial conditions, then the updated values of the state variables are copied from two temporary arrays into it. In particular voltage values occupy the first num_cells positions of the array and n, like m, is an index which denotes the number of the cell (n=0,1...num_cell). For each step, m and n denote the same cell, just in two different vectors and x[n] represents the value assumed by V_m[m] at the previous time step;

- V_{net} is the sum of the contributions in voltage deriving from the four neighbornig cells: V_{net}=V_{net_R}+V_{net_L}+V_{net_U}+V_{net_D};

- *step* is the time step, imposed equal to 0.005 ms after having verified that it returns no appreciable difference in results compared to what obtained with a 0.002 ms step;

- *rad* is the cell radius imposed equal to 4 μm;

- ρ is the resistivity and its value, as discussed in details later, is changed within a range of 10^4;

- C_m as usual is the membrane capacitance, whose value is equal to 32 pF;

- *deltx* is the cell length imposed equal to 70 μm;

- I_{ion} is the sum of all the different ionic currents;

- *stim* is an eventual stimulus current, not normally present in a model of SAN cells, but it could be useful in our simulations as a tool to impose different initial conditions to the cells, giving the stimulus only to few of them.

As before another kernel (*update_voltage*) is implicated in the storing of the updated values for the voltage. These are saved in the first cell_num positions of the vector used previously to save the other updated state variables.
Once terminated the execution of each kernel, the updated values for every time step of the simulation are available to be exported and then visualized and further used to measure outputs of interest, such as cycle length and action potential amplitude. In order to be able to do that in MATLAB, cell membrane voltage values for all time points are written into a file named *Maltsev GPU Voltage* using the *fopen()* function in the write mode. These data are structured such that each row of the matrix contains the values of each cell's voltage at the same time point. Therefore the file can be opened in MATLAB environment through the built in function *dlmread()*, similarly the function *dlmwrite()* is used when it is necessary to import in CUDA an array created in MATLAB and then saved as a file.txt and opened in the program using the *fopen()* function in the read mode.

2.2.3 Parameter randomization

First the program *Maltsev.cu* was tested to see if it returned the same results of the one implemented in MATLAB, therefore the case of a cable model with the same initial conditions and same parameters for every cell was simulated. Fig. 2.7 shows the correspondences between the results obtained by the two programs in the same condition.

Another assessment was carried out before starting to use the model with

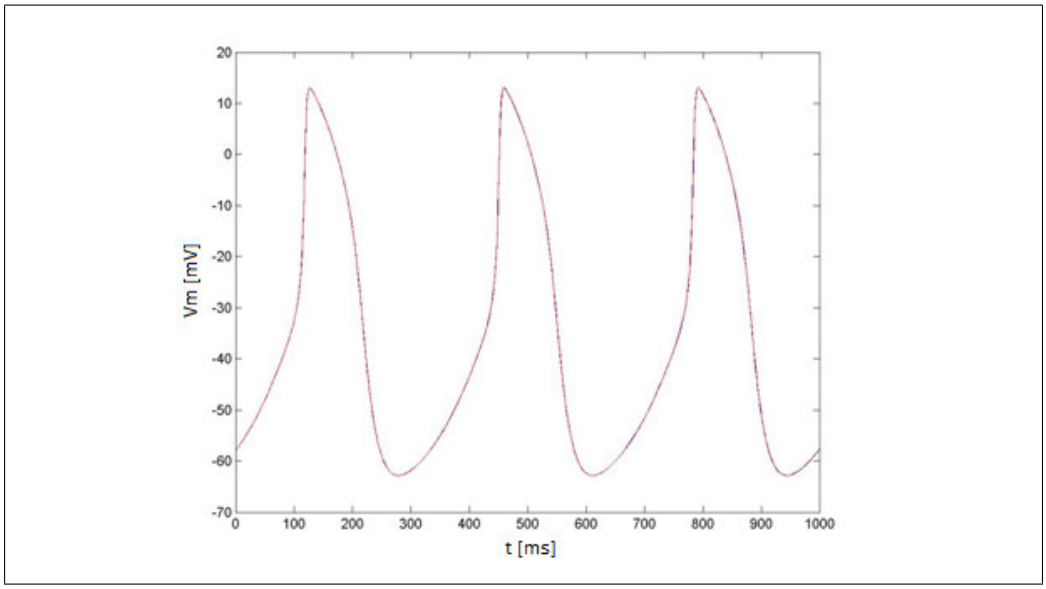

Figure 2.7: Comparison between MATLAB and CUDA simulations in the same condition. The two curves are superimposed.

its ultimate purpose. We wanted to see how the tissue model behaves when we have a population of different initial conditions for the cells. In MATLAB a script was realized to store in a matrix, then converted to a vector, all the initial conditions to be assigned each cell in the CUDA program. To define the initial conditions of the state variables, a similar strategy to that of the cable case was used, the Pythagoras's theorem was taken into account to establish the time delay between cells placed on the diagonal of the matrix, in fact the distance between these cells is greater than the one between cells along a row or a column. In the script we start by running a 50 s single cell model simulation, the values of the state variables at the end of the simulation become the initial conditions for the cell placed in the

upper left corner of the matrix of cells. Starting from these values for the variables, two types of simulation are run, the first type lasts 1 ms and is used to compute the initial conditions for the cells placed in the first column and row, each time using as initial conditions the final values of the previous cell, the second type lasts $\sqrt{2}$ ms and is used to compute the initial conditions for the cell placed on the diagonal next to the first cell. The process is then repeated running the same types of simulations but using as initial conditions the final values of the cell on the diagonal and filling in this way the second row and column and the third cell on the diagonal, from which to start the next group of simulations.

Fig. 2.8 shows at the top the pattern of the membrane potential obtained applying the described approach to a matrix of cells (function *imagesc()* in MATLAB). In (b) the evolution of the membrane potential over time is represented and two marks are highlighted at the point where respectively first and last cell cross a potential value between -20 and -10 mV. The difference between these two points will correspond to a certain time interval and 10 time points are chosen within this time interval. The panels in (c) depict in a color map (still using the *imagesc()* function) the values assumed by the membrane potential of each cell at that instant of time. Warm colors represent cells that are crossing or have passed the threshold to unleash the action potential while cool colors indicate cells that still have very negative potential.

Same considerations have been made in the case of random initial conditions for the state variables. These were computed choosing random scale factors for conductances from a log-normal distribution with a median value of 1, and with σ equal to 0.1873. This last parameter specifies the standard deviation of the distribution of log-transformed variables and so controls the extent to which parameters vary. The choice of this value derives from its assessment as a good approximation to reproduce the physiological variability in a population of cardiac cells, based on previous studies on ventricular cells behavior investigation using linear regression [14]. Simulations with a population of cells with random values for the conductances were run (time of simulation = 1 s) and the obtained results for the state variables were stored in a vector of initial conditions then used in the CUDA program.
Examples of the results of the simulations carried out in these conditions are shown in fig. 2.9.
Once verified the correct implementation, we were able to use the program

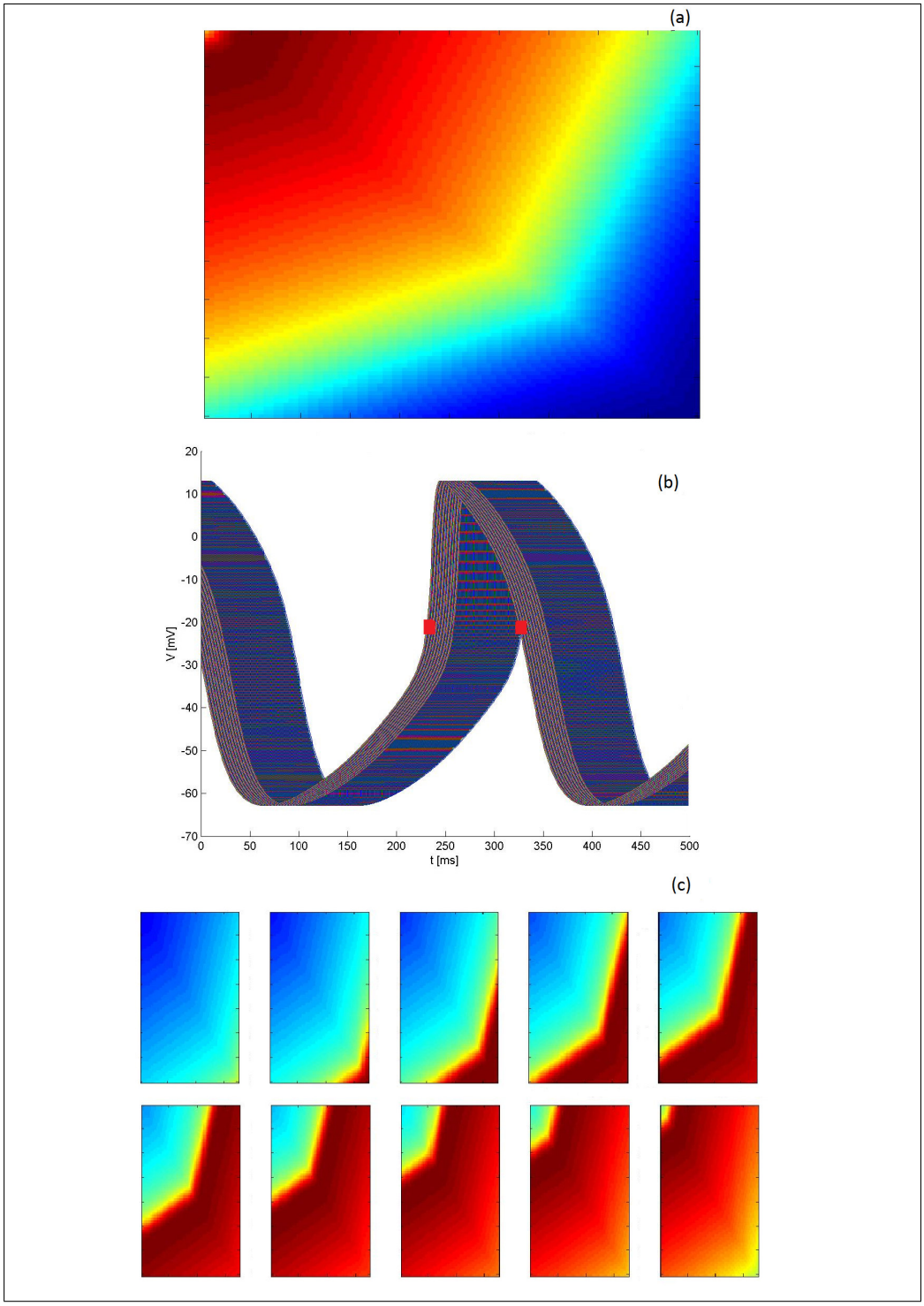

Figure 2.8: (a) Illustration of the initial conditions for the membrane potential, the level of membrane potential is encoded by pixel color; (b) Evolution of the membrane potential over time; (c) Illustration of the membrane potential at 10 different time point values.

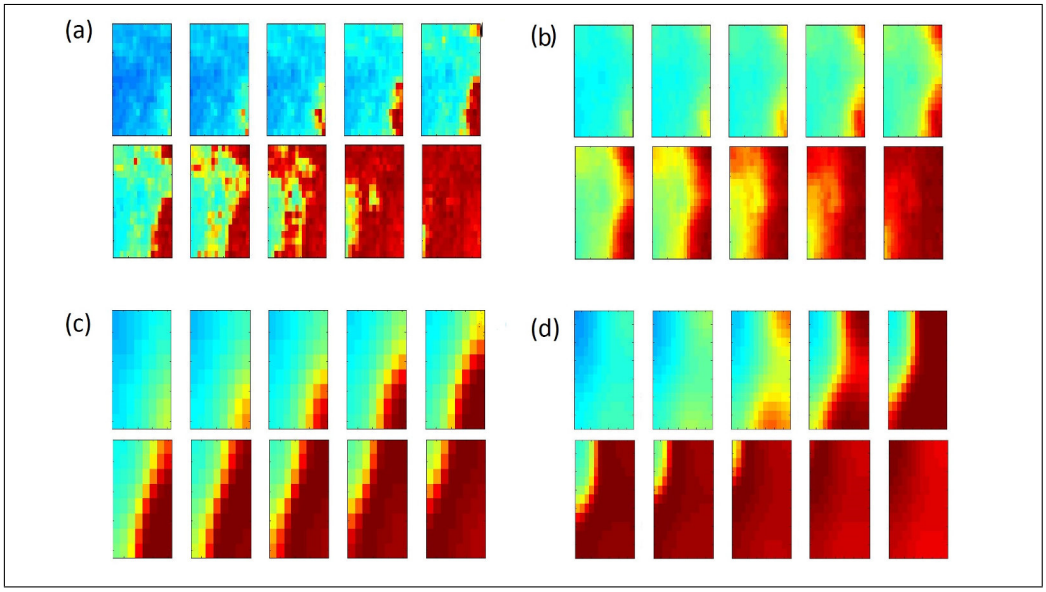

Figure 2.9: Examples of color maps of voltage membrane obtained running Maltsev.cu with random initial conditions for the SAN cells in the tissue: (a)-(b) tissue 15x15 cells at different intervals of time during the simulations; (c) tissue 8x8 cells; (d) tissue 12x12 cells

with its main scope, that is, to evaluate the effect of heterogeneity within the tissue. For this purpose a script in MATLAB was formulated to randomly vary the values of the maximal conductances used in the model, so that each cell has a different value for each of the conductances.
The parameters concerned and their base values are the following:

- double $g_{sus} = 0.02$ *nS/pF (in AP sensitive currents)*;
- double $g_{to} = 0.252$ *nS/pF (in AP sensitive currents)*;
- double $g_{CaT} = 0.1832$ *nS/pF (in i_{CaT})*;
- double $g_{Kr} = 0.08113973$ *nS/pF (in i_{Kr})*;
- double $g_{Ks} = 0.0259$ *nS/pF (in i_{Ks})*;
- double $g_{CaL} = 0.464$ *nS/pF (in i_{CaL})*;
- double $g_{b\_Ca} = 0.0006$ *nS/pF (in i_{b_Ca})*;

- double g $_{b\_Na}$ = 0.00486 *nS/pF (in i_{b_Na})*;
- double g_{if} = 0.15 *nS/pF (in i_f)*;
- double g_{st} = 0.003 *nS/pF (in i_{st})*.

The conductances values for each cell were computed as follows:

conductances = [g_{sus};g_{to};g_{CaL};g_{CaT};g_{Kr};g_{Ks};g_{b_Ca};g_{b_Na};g_{if};g_{st}];
n_conductances = length(conductances);
σ = 0.05;%0.1;%0.1873;%0.3;%0.4; *% standard deviation of variation of parameters*
for ii=1:variations
scaling = exp(σ*randn(n_conductances,1)); *%randn: standard normal distribution*
newparams = scaling.*conductances ;
...
end

Where:

- *scaling* is the array of the conductance scaling factors;
- σ is the standard variation arbitrarily chosen to represent the physiological variation in a population of cells, 5 different values are studied (σ=0.05; 0.1; 0.1873; 0.3; 0.4);
- *randn(N)* is a MATLAB built-in function that returns an N-by-N matrix containing pseudo-random values drawn from the standard normal distribution.
- *newparams* are the resulting computed conductances.

In this way random scale factors for conductances are chosen from a log-normal distribution with a median value 1. As more clearly shown in Fig. 2.10 log-normally distributed random numbers are obtained from the normally-distributed random numbers generated using the MATLAB function randn through the transformation l=e^n. This was done to exclude non-physiological conditions such as negative conductances [33].

Conductances are then stored in a vector to be loaded in the GPU and used to perform Maltsev.cu 2D simulations.

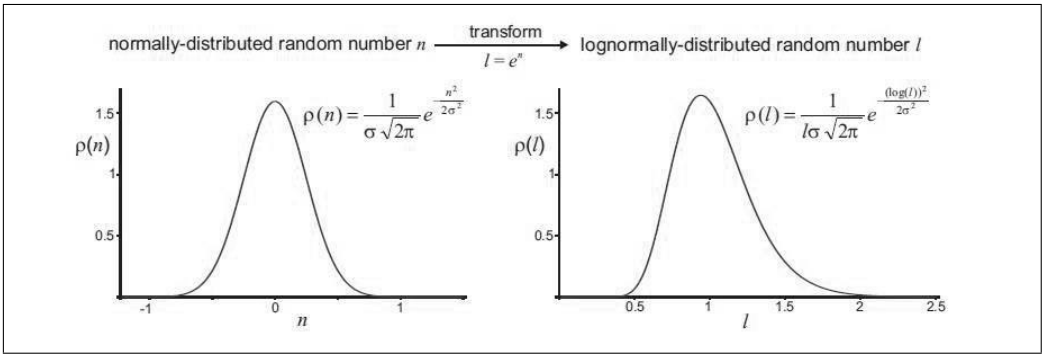

Figure 2.10: The two plots are the probability density functions of the normal (left) and log-normal (right) distributions, calculated for σ=0.25. The equations show the transformation from normally-distributed random numbers to log-normally distributed random numbers

2.3 Simulations

The performed simulations can be subdivided into two macro groups: 1D simulations and 2D simulations.

- 1D simulations enclose simulations carried out in MATLAB using both Severi and Maltsev Cable Model to evaluate how vary the resistive coupling within a wide range (three values have been tested: R_{gap}=133,1330,13300 MΩ) affects the conduction velocity (CV) and so the ability of synchronization of a group of either 25 or 50 cells. CV is computed by dividing the distance (cell's length replicated for the cells number) between first and last cell for the time delay between the two. In turn the time delay is the difference between the time points at which the two cells membrane voltages have the maximum value of first derivative (dV_{max} = maximum upstroke velocity). Simulations performed in this study last 50 s.
  The Maltsev cable model was then also used to evaluate the difference in terms of execution time between CUDA and MATLAB, testing different numbers of cells within the range 5 - 400 cells.

- 2D simulations were mainly directed to the evaluation of the behavior of a population of coupled SAN cells with an heterogeneous distribution of maximal conductances. Simulations performed in this study last 20 s and different values of inter-cellular variability (σ=0.05, 0.1,

0.1873, 0.3, 0.4) and resistive coupling (ρ=1, 10, 10^2, 10^3, 10^4 MΩ m) have been tested. In particular the effects over *cycle length* (CL) and *action potential amplitude* (APA) were evaluated.
The two outputs were computed as follows:

- *CL*: time between two consecutive crossings of 0 mV on the upstroke of the AP;
- *APA*: difference between the most positive voltage recorded during the AP (OS, peak voltage) and the most negative one (MDP, maximum diastolic potential).

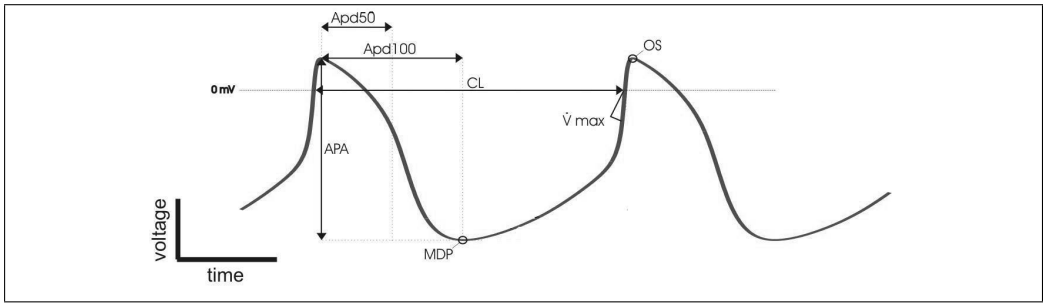

Figure 2.11: Scheme of an AP and its outputs of interest

The main question to be addressed through this group of simulations was if the cells synchronize their spontaneous beating and at what rate. In order to do that the behavior of the isolated cells was compared with the behavior of the same cells put together and connected between them in the Maltsev Tissue Model.

The results of these simulations are discussed in the next chapter.

Chapter 3

Results

In this chapter we discuss the results of the simulations carried out with the models implemented in this work. The results, as done previously with the description of the simulations, have been subdivided into two groups. Through the first section we briefly explore the effects of the resistive coupling over the synchronization of a cable of cells. We also include a demonstration of the clear benefits observed in terms of speed of execution while conducting simulations in CUDA rather than in MATLAB. In the second part of the chapter, where 2D results are described, we finally answer the main question to be addressed with this work. Results of comparisons between isolated heterogeneous cells and the same cells connected to form a tissue are reported, so that in this way the ability and manner of synchronization of said cells are evaluated.

3.1 1-D model results

In this section two different analyses are considered and three models were used to obtain the reported results. First *Maltsev cable model* and *Severi cable model* are compared in the study of how the resistive coupling between cells influences their ability of action potentials synchronization. Afterwards *Maltsev cable model* is compared to *Maltsev.cu* considering the basal case in which all cells have the same characterization.

3.1.1 Conduction velocity and gap junctions

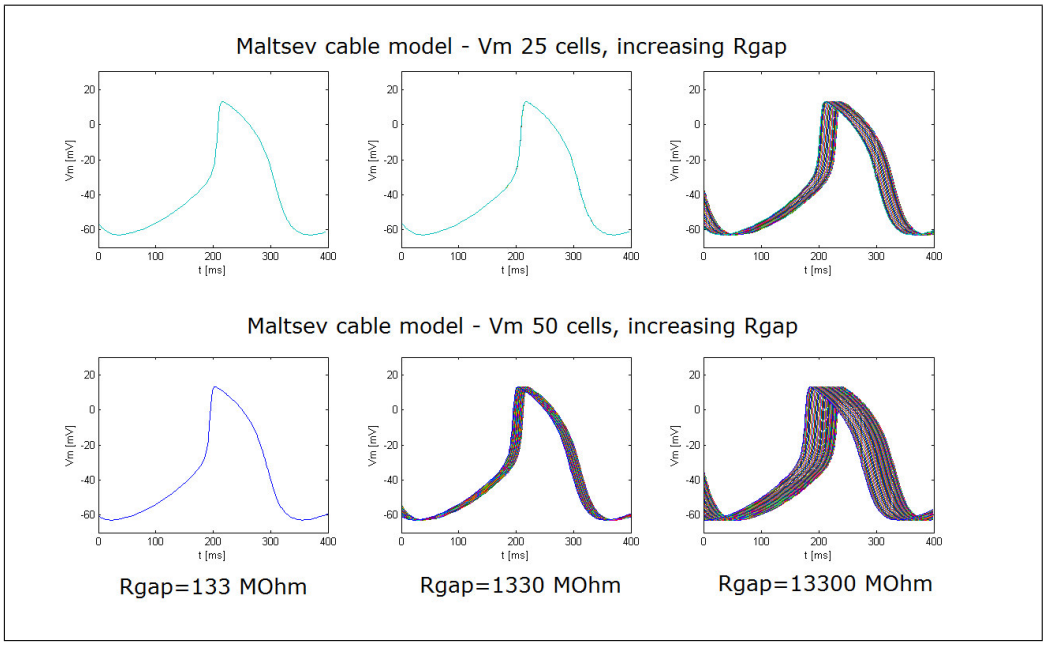

Figure 3.1: Simulated action potentials in different resistive coupling conditions using Maltsev Cable Model

The two models, *Maltsev* and *Severi Cable Models*, have been used to perform simulations of 50 s considering both 25 and 50 cells. To be able to carry on the simulations in MATLAB, especially in the 50 cells case, we had to use a strategy to save and store the state variables values only for the last part of the simulation (e.g., the last 500-1000 ms) to avoid memory problem (*out of memory* error). The results obtained by testing

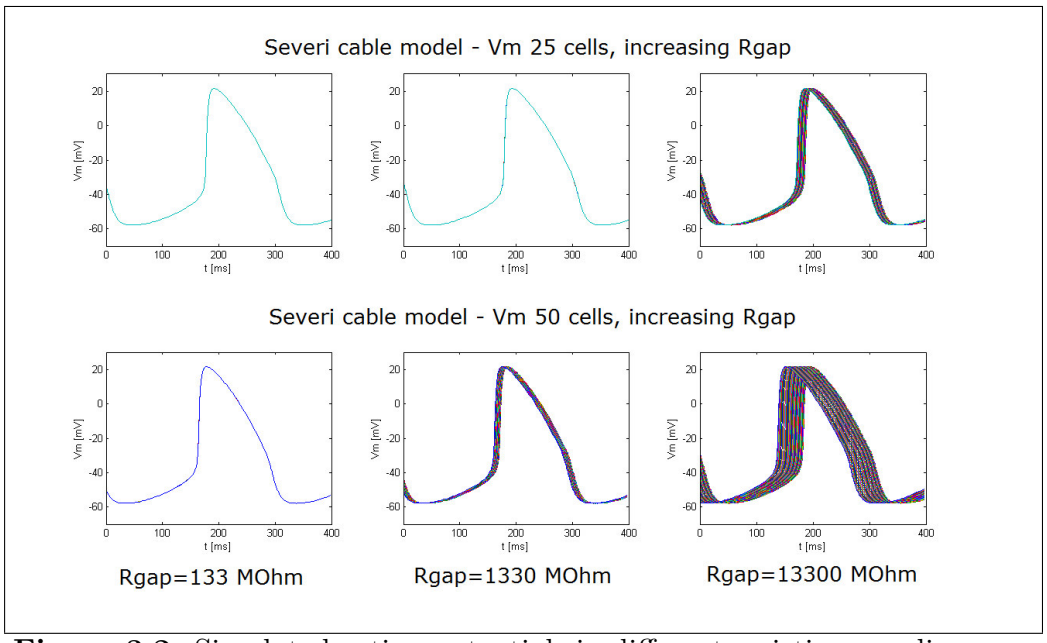

Figure 3.2: Simulated action potentials in different resistive coupling conditions using Severi Cable Model

three different values for the coupling resistance are shown in Fig. 3.1 and Fig. 3.2, the last action potential of each simulation is plotted. The starting value of 133 MΩ is derived from a reference value found in the literature [28], but it is only an indicative value because also in that study a fixed value for the coupling resistance between cells was not considered.
The values of conduction velocity achieved by our simulations, especially when considering lower resistance's case, are greater when compared to the values found in the literature for conduction velocity measured in the intact rabbit SAN [6] [16] [17] [30]. CV values reported in these studies fall in the range 1÷30 cm/s and we are able to obtain them in our models only when we consider a value of R_{gap} greater than 10^3 MΩ, in fact when considering R_{gap}=13300 MΩ we achieve a value of 5.79 cm/s using Maltsev Cable Model and a value of 9.64 cm/s whit Severi Cable Model. These values are computed as mean of CV values obtained considering 25 and 50 cells, slightly different between them. When considering lower values of R_{gap} in case of 25 cells, the time delay between first and last cell is too small to be able to use it to compute the CV. We are still able to do that only in the

case of 50 cells with R_{gap}=1330 MΩ, obtaining CV=17.5 cm/s with Maltsev model and CV=35.44 cm/s with Severi model. As discussed in the first chapter, the SAN has an extremely heterogeneous structure, for example, it is rich of fibroblasts that influence and alter the conduction velocity through the SAN. The SAN is also surrounded by the atrium, which still can act as a current sink. Finally, a distinction should be made between values of CV measured along fibres direction and the ones measured along the transversal direction; indeed, as far as we know, in the models found in the literature the scientists do not start from a fixed value of coupling resistance, but a CV target value is set and then the coupling value is calculated as the one necessary to reach that CV.
These considerations provide a starting point to making the model more detailed (e.g., by introducing models of atrial cells around the SAN cell models or fibroblasts dispersed within them) in order to be able to make a more truthful comparison with the values found experimentally, but that does not fall within this thesis direct goals.
We also performed a comparison between results obtained in the previous condition with R_{gap}=13300 MΩ and the ones obtained after having assigned randomly the initial conditions of the cells. This was made giving the cells random conductances and then running a 1 s simulation, the final values of the state variables become the initial conditions for the cable model simulation. The APs obtained with Severi and Maltsev models at the end of 50 s simulation are shown in Fig. 3.3.

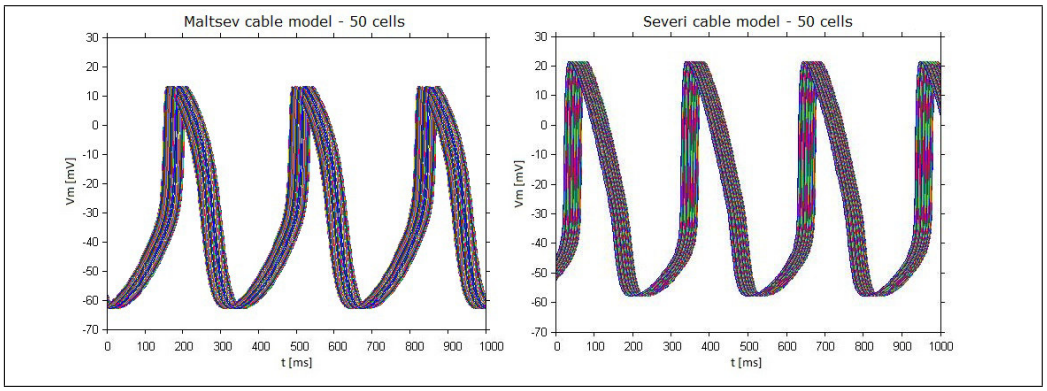

Figure 3.3: Simulated action potential using Maltsev and Severi cable models. Simulations are performed for 50 cells, with R_{gap}=13300 MΩ and random initial conditions.

3.1.2 Execution Time in MATLAB and CUDA

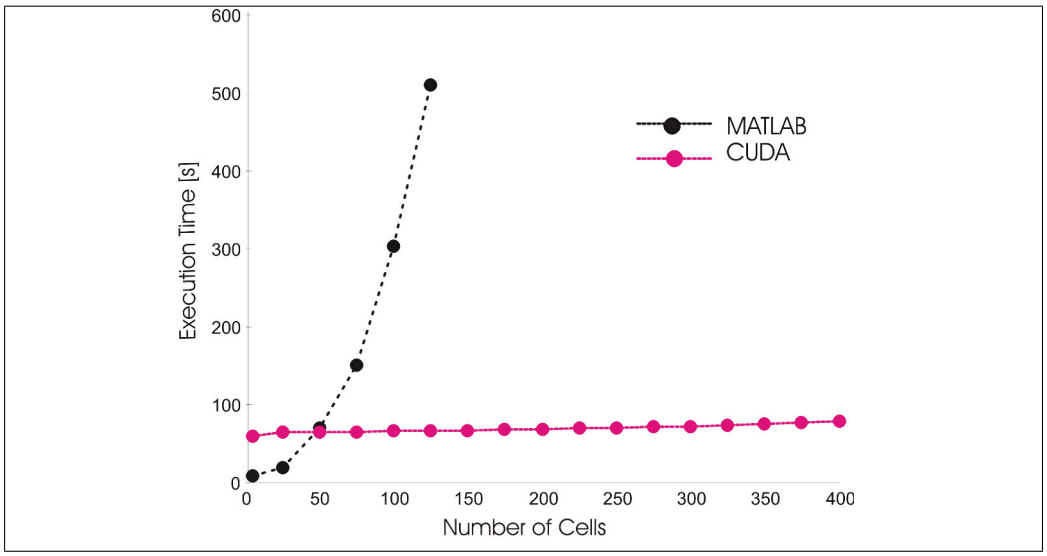

Figure 3.4: Execution Time in MATLAB and CUDA, 5 - 400 cells, Simulation Time = 20 s

Fig.3.4 clearly shows the advantages in terms of speed of execution obtained when the same simulations performed before in MATLAB are then repeated in CUDA.
The Execution Time of a 20 s 1D simulation in MATLAB rapidly grows with the number of simulated cells, passing from a value of 7.96 s when considering 5 cells to one of 510.46 s when considering 125 cells; beyond this value a problem of memory, mentioned before, occurs in MATLAB, even if we try to save only the last 500 ms of simulation. On the contrary the Execution Time of the same simulation in CUDA varies to a lesser extent varying the number of cells, passing from a value of 58.58 s when considering 5 cells to one of 77.26 s when considering 400 cells.
This test was designed only to provide a proof of what apparently occurred during simulations.

3.2 2-D model results

This section presents the results obtained by simulating a heterogeneous population of cells in the sinoatrial node. The heterogeneity is achieved in practice through a randomization of the maximal conductances in the currents expressions of the Maltsev model. The purpose of this analysis is to evaluate the difference in behavior between a population of isolated cells and the same cells connected together to form a matrix.

3.2.1 Effect of heterogenity on Cycle Length and Action Potential Amplitude

Fig. 3.5 to Fig. 3.9 show the conductances distributions for each tested value of σ. These results come from a simulation of the Maltsev single cell model considering 400 different combinations of conductances, whose scaling factors were picked, as described before, from a log-normal distribution. Therefore we have 400 cells still not connected between them and each one described by a unique model characterized by having a unique combination of conductances values. Together with the conductances distributions, also Cycle Length and Action Potential Amplitude distributions, computed for each cell as previously explained, are displayed.
Since the standard deviation σ of log-transformed variables controls the extent to which parameters vary, the smaller the σ value, the narrower are the parameters distributions. In particular we can notice that, for sigma values of 0.1873 and above, a large number of cells have CL and APA equal to zero, reaching a percentage of 40% in the case of σ=0.4. This means that with particular conductances combinations, cells are not excitable, so that they do not exhibit any action potential and it is reasonable that this happens more easily in case of great values of σ, when the values of conductances most different from the basal ones are taken into account.

Fig. 3.10 should better point out the difference in the amplitude of parameters distributions when considering different values of σ. All cases plots are superposed, so that we use the same scale for the parameters values and it is easier to appreciate the difference between distributions amplitude in the five cases.
Fig. 3.11 to Fig. 3.15 show the results obtained after having loaded in the CUDA program the conductances vector and having run 20 s simula-

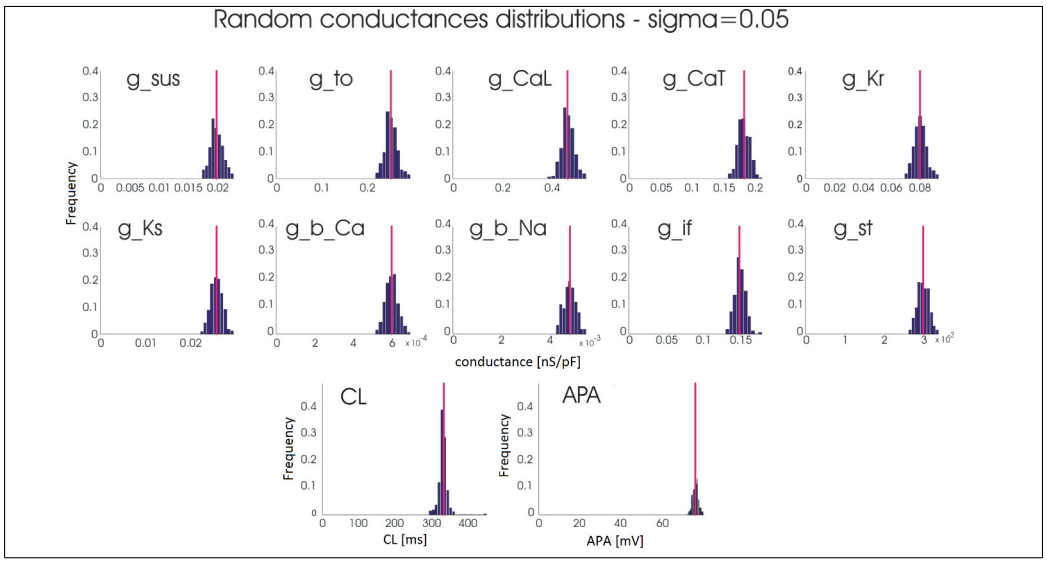

Figure 3.5: Conductances, CL and APA distributions when considering σ=0.05. The basal values of conductances, APA (75.87 mV) and CL (332.50 ms) are shown in magenta line.

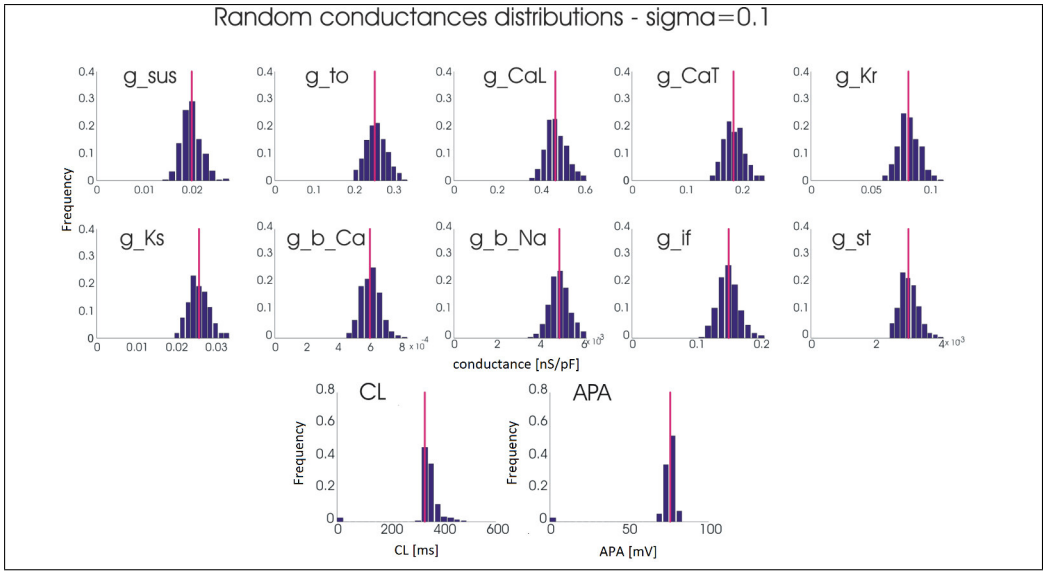

Figure 3.6: Conductances, CL and APA distributions when considering σ=0.1. The basal values of conductances, APA (75.87 mV) and CL (332.50 ms) are shown in magenta line.

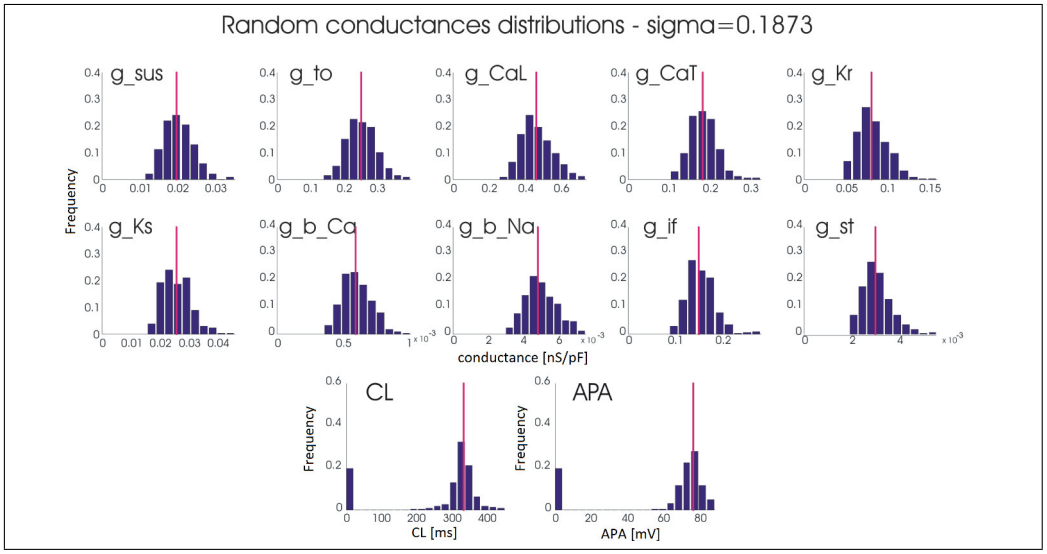

Figure 3.7: Conductances, CL and APA distributions when considering σ=0.1873. The basal values of conductances, APA (75.87 mV) and CL (332.50 ms) are shown in magenta line.

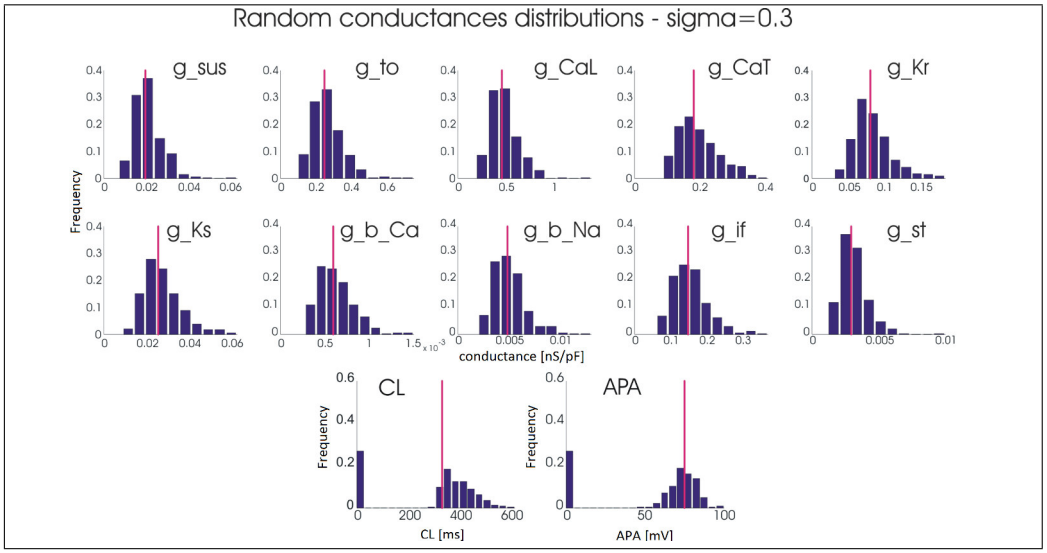

Figure 3.8: Conductances, CL and APA distributions when considering σ=0.3. The basal values of conductances, APA (75.87 mV) and CL (332.50 ms) are shown in magenta line.

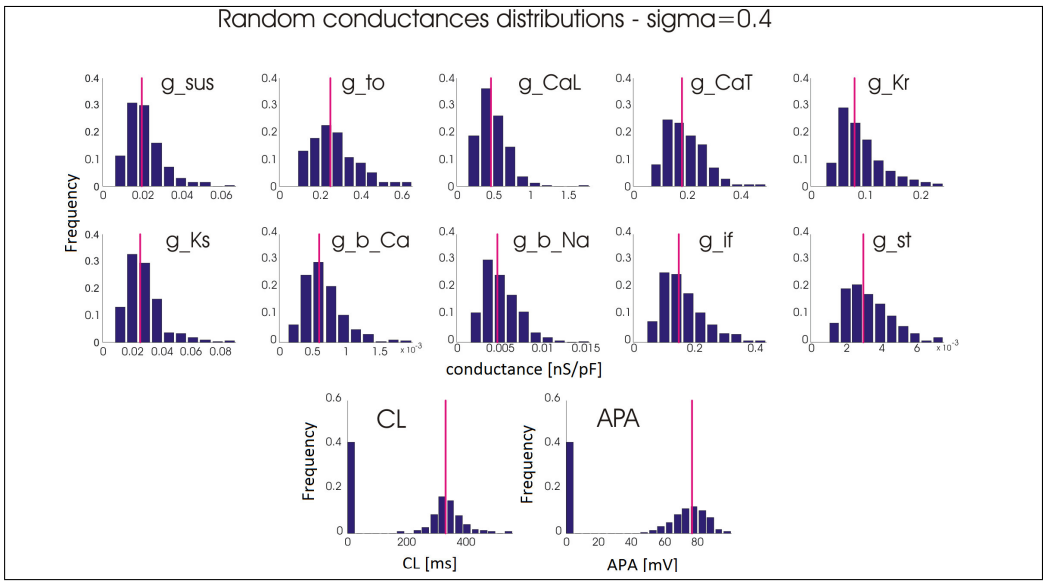

Figure 3.9: Conductances, CL and APA distributions when considering σ=0.4. The basal values of conductances, APA (75.87 mV) and CL (332.50 ms) are shown in magenta line.

tions with the Maltsev Tissue model. Each cell is still characterized by a unique combination of conductances, but now the cells are connected between them, so that they influence each other, in particular through these illustrations we can see the synchronization effect on CL.

Starting considering the first case, σ=0.05, we see that at the end of the simulations the cells have synchronized their discharge frequency, this can be clearly noticed both from the colour maps and the histograms. The most favourable case for the cells to synchronize their behaviour is when the inter-cellular variability, expressed in our simulations as the extent of conductances variability, is slight and when the resistivity ρ has a small value, therefore they have a strong coupling between them. In fact when ρ is equal to the smallest value which has been tested (ρ=1 MΩ m) all cells show the same cycle length, moreover this last is equal to the control value ($CL_{control}$=332.5 ms), i.e. when all conductances have their basal values. Looking at the worst case, i.e. when ρ is equal to the greatest value (ρ=10^4 MΩ m), we find out that cells are still synchronized, there are only two different values of cycle length (CL=331.5 ms and CL=332 ms)) which are

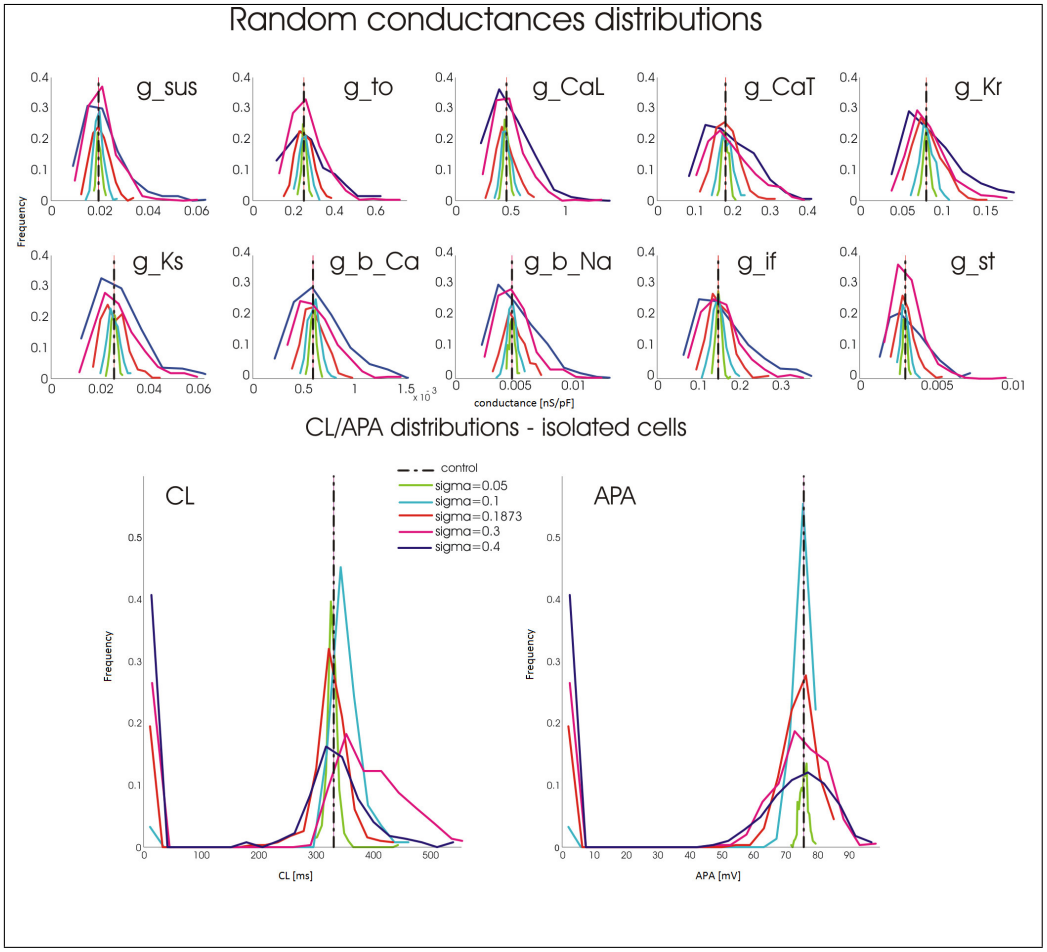

Figure 3.10: Conductances, CL and APA distributions are shown using different colours for each value of σ to point out the different extent of variation of parameters

really close to the control value, even if slightly lower. This result is in accordance with the fact that we were evaluating a low degree of variability.

When σ=0.1 and ρ has the minimum value, the cells have a final uniform value of CL equal to 332 ms. As anticipated, increasing the value of resistivity and thus resistance, the ability of the cells to synchronize toward a unique value of CL should decrease, in fact in the case of ρ maximum CL varies within a range of 3.5 ms (CL_{min}=327.5 ms, CL_{max}=331 ms).

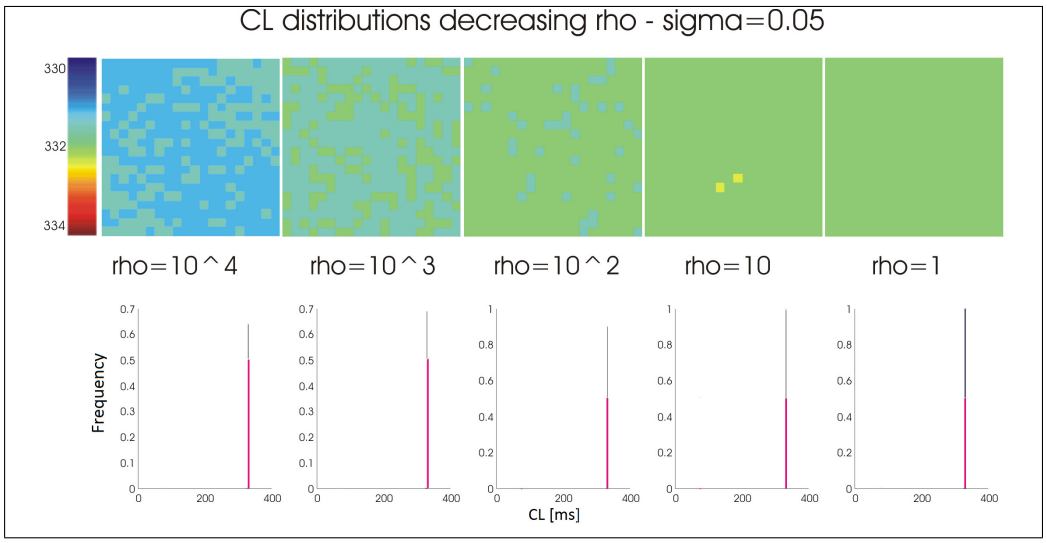

Figure 3.11: At the top colour maps of CL at the end of 20 s simulations are shown. At the bottom we can see histograms for the same data.

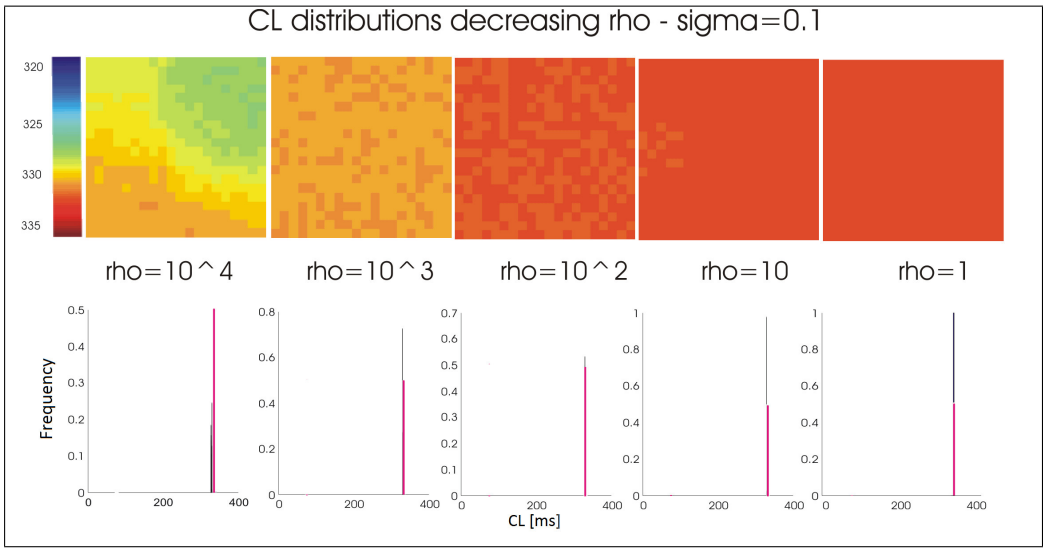

Figure 3.12: At the top colour maps of CL at the end of 20 s simulations are shown. At the bottom we can see histograms for the same data.

What we expect is that, increasing the inter-cellular variability, CL final value will vary within a wider range, especially when considering cells less strongly connected between them. Looking at the case σ=0.1873, shown in

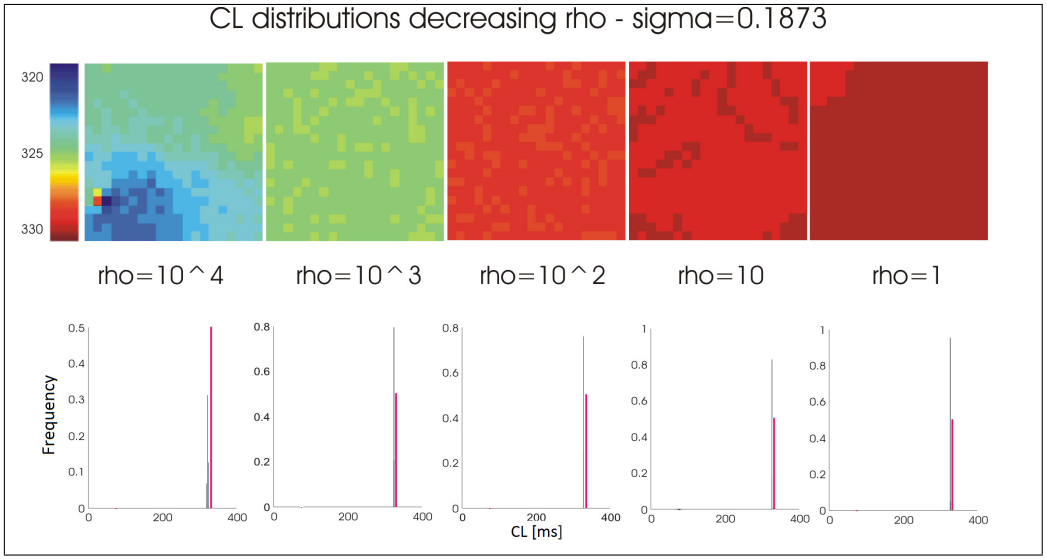

Figure 3.13: At the top colour maps of CL at the end of 20 s simulations are shown. At the bottom we can see histograms for the same data.

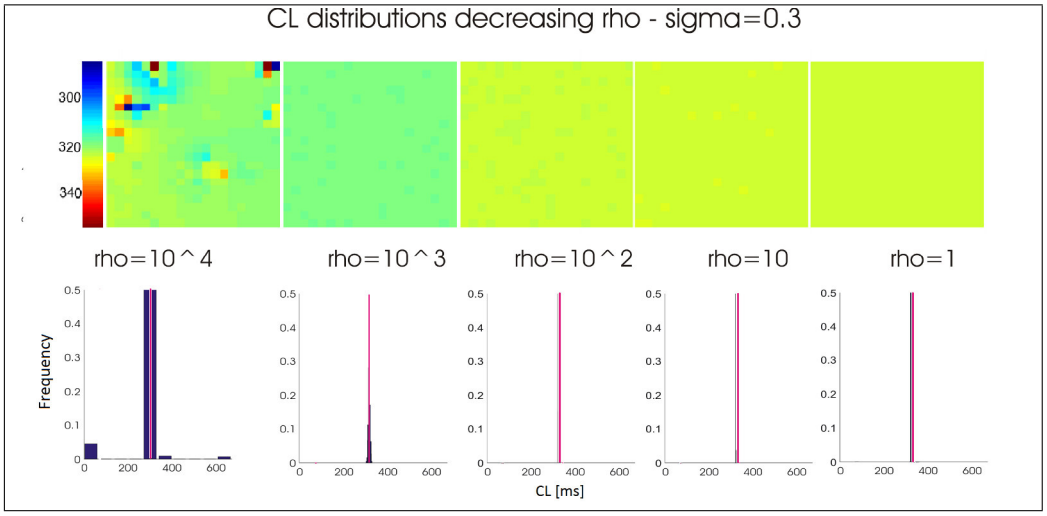

Figure 3.14: At the top colour maps of CL at the end of 20 s simulations are shown. At the bottom we can see histograms for the same data.

Fig. 3.13, we find evidence for this hypothesis in the case of ρ=10^4 MΩ m when CL varies within a range of 8 ms (CL_{min}=320 ms, CL_{max}=328 ms), while when considering ρ equal to the minimum value the cells are still able to synchronize their rhythm. CL in fact assumes only two alternative close

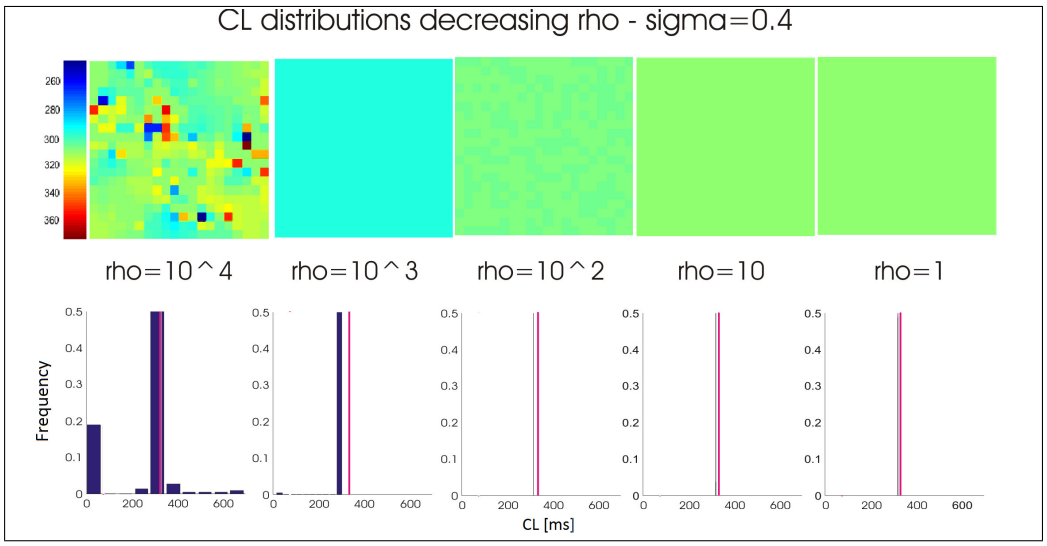

Figure 3.15: At the top colour maps of CL at the end of 20 s simulations are shown. At the bottom we can see histograms for the same data.

values: 329 ms and 329.5 ms and this consideration is still true even when referring to ρ=10^3 MΩ m, but this time the two values are lower (CL=325 ms and CL=325.5 ms) and this fact recalls what we have seen less markedly also before.
The last two conditions: σ=0.3 and σ=0.4 are the most extremely ones, in fact, as said when we were talking about isolated cells, they generate a large percentage of non excitable cells. Moreover CL and APA for the remaining cells vary within a very large range. Having said that, it is interesting find out that with both values of σ, when considering values of ρ in the range ρ=1÷10^3, the cells are still able to synchronize their frequency. The CL final values considering ρ=1 MΩ m are CL=325.5 ms and CL=321÷321.5 ms respectively for σ=0.3 and σ=0.4. These two values are lower than the ones reported for the previous σ conditions (with same rho) and make us note that CL not only decreases with the increasing coupling resistance but it also decreases with the increase of inter-cellular variability. At the top of Fig. 3.14 and Fig. 3.15 we report the colour map of CL, but in the case ρ=10^4 MΩ m in the upper left part of the figures, we consider only cells with a value of CL comprised between 200 ms and 400 ms, however the histograms still consider all the cells.

Previous considerations are summarized in the right section of Fig. 3.21

where all situations of inter-cellular variability and resistivity are illustrated. The mean values of CL are plotted both versus ρ and σ and in both conditions it is evident the decreasing trend.

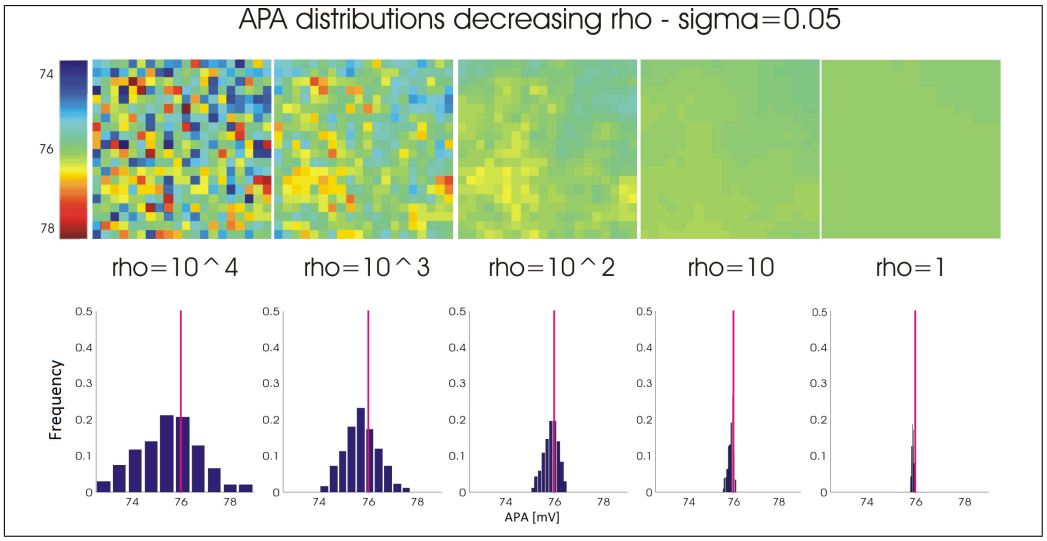

Figure 3.16: At the top colour maps of APA at the end of 20 s simulations are shown. At the bottom we can see histograms for the same data.

The same analysis was carried out to study the effects on the Action Potential Amplitude and Fig. 3.16 to Fig. 3.20 show the obtained results. Starting again from the condition σ=0.05 we can note that, unlike what was seen for the CL, the cells are less likely to synchronize their APA, in fact the final APA value is almost the same for all cells (APA_{min}=75.80 mV, APA_{max}=75.94 mV) when ρ assumes the lowest value but it varies within a range of about 6 mV (APA_{min}=72.51 mV, APA_{max}=78.98 mV) when considering ρ=10^4 MΩ m, while before all cells had the same CL value even in the latter case. As can be seen in Fig. 3.16, both in the colour maps and the histograms, the range within which APA varies increases progressively with ρ. This fact reoccurs also in the other conditions of sigma, moreover, as it was for the CL, the range in which APA varies increases with σ and this is more appreciable when considering a great value for ρ. The trend of APA mean values with σ and ρ is similar to the one of the CL and it is shown in the right part of Fig. 3.21.

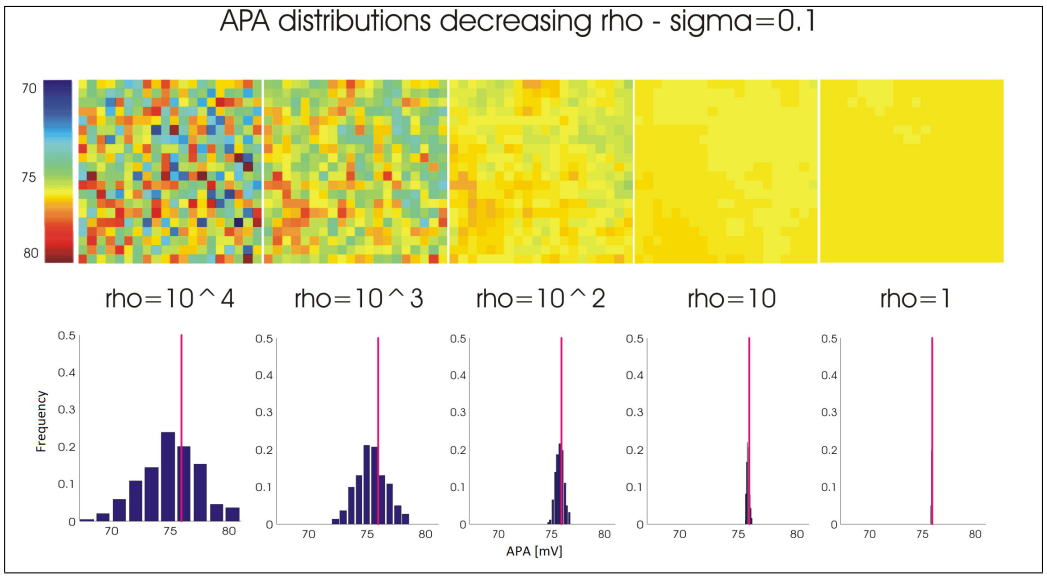

Figure 3.17: At the top colour maps of APA at the end of 20 s simulations are shown. At the bottom we can see histograms for the same data.

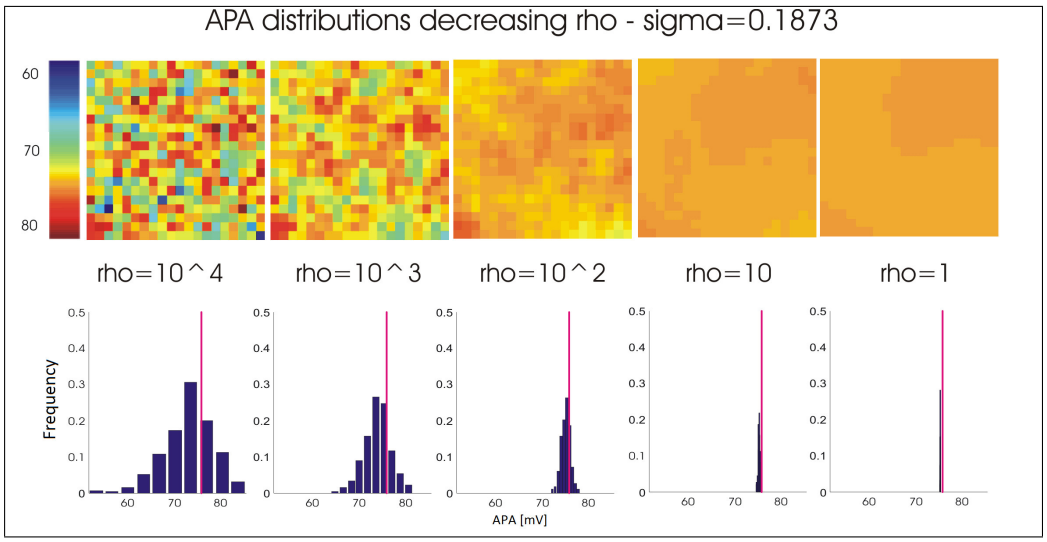

Figure 3.18: At the top colour maps of APA at the end of 20 s simulations are shown. At the bottom we can see histograms for the same data.

3.2.2 Control + non-excitable cells model

After having analyzed the previous results, we decided to perform another group of simulations to better understand the bases of coupled cells be-

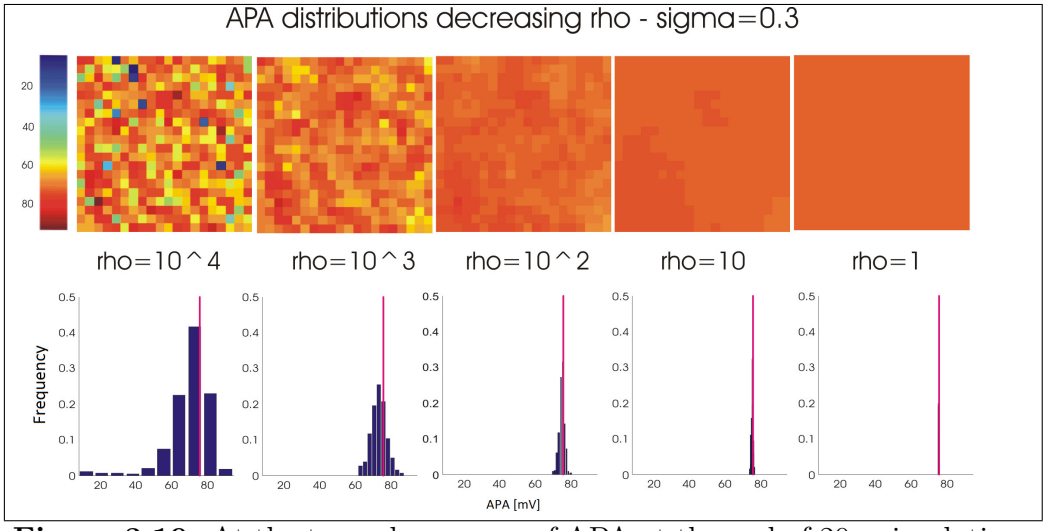

Figure 3.19: At the top colour maps of APA at the end of 20 s simulations are shown. At the bottom we can see histograms for the same data.

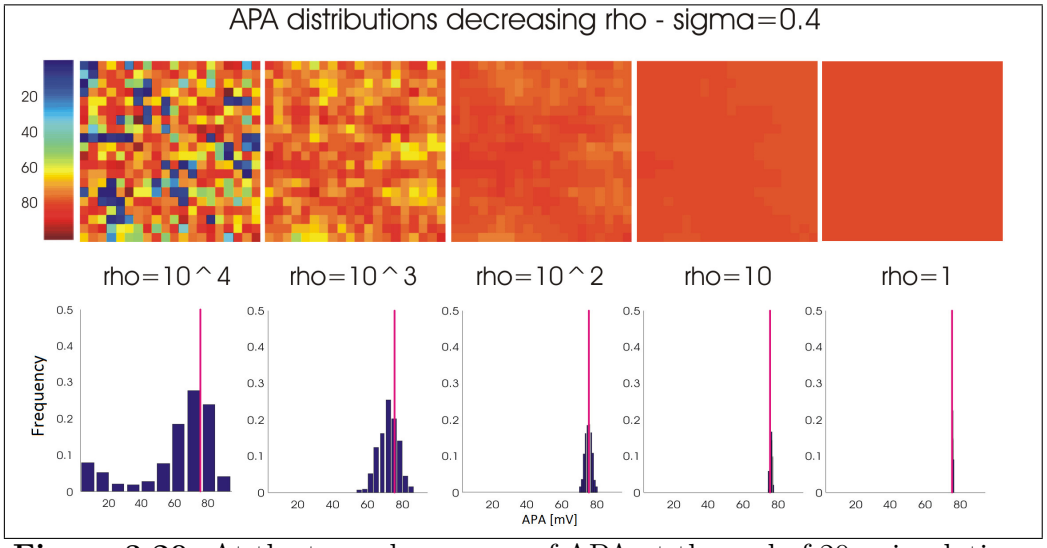

Figure 3.20: At the top colour maps of APA at the end of 20 s simulations are shown. At the bottom we can see histograms for the same data.

haviour and in particular to try to explain the decreasing trend of APA and CL with ρ and σ.
The coupling strength between cells plays an important role in their synchronization process, indeed the discrepancy in term of discharge frequency

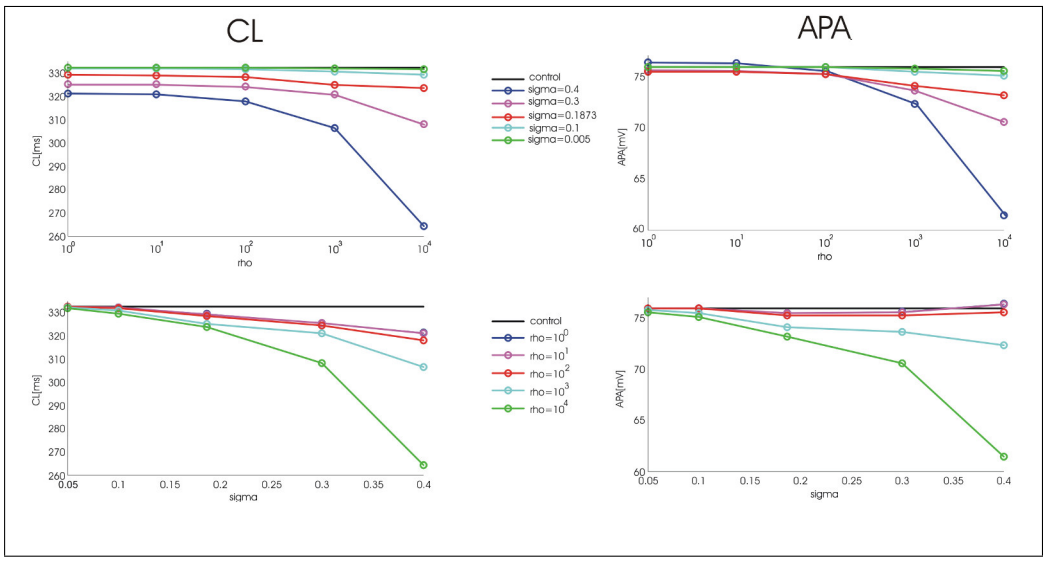

Figure 3.21: In the left part of the figure we plot the decreasing trend of CL both versus rho (at the top) and versus sigma (at the bottom). In the right part of the figure we do the same in the case of APA. Mean values of CL and APA obtained by 20 s simulations of Maltsev Tissue Model are plotted.

between SA node cells derives from their differences in the slow diastolic depolarization phase. When this last is slower, cells will have a reduced discharge frequency, vice versa cells with a less slower depolarization will be the fastest ones, i.e. they will have the greatest discharge frequency. When ρ has a small value, so that cells are more strongly connected, the very small currents playing during this phase will be sufficient to make up for the slight cells discrepancies during the diastolic phase and then to achieve their synchronization. On the contrary, if ρ assumes a greater value, the reduced coupling strength will prevent little current flows from synchronizing cells rhythm and ideally the fastest cell would predominate and it would impose its rhythm to the others. In order to maintain its fastest rhythm it should achieve the upstroke phase of the AP before being slowed too much by the influence of neighboring cells during the diastolic phase and, as said, this is possible only when coupling strength between cells is weak. Otherwise, with a stronger coupling strength the cells would be able to reach an almost uniform rhythm, slower than the fastest one.

This decreasing trend with ρ, however, is more significant when considering situations of great values for σ and so great inter-cellular variability. As already emphasised, in these cases we are handling borderline situations where conductances can assume values within a wide range and so cells behaviour is really different from the basal one. Moreover this decreasing trend of CL is shown also with σ and APA mean values show the same trend both with ρ and σ.
All these considerations could make us wonder if this decreasing trend is only due to the heterogeneity within cells or the large number of cells having no true pacemaking activity (and so resulting in a large number of CL and APA equal to zero) could play an important role too. In order to address this question we decided to carry out simulations using a new model, obtained combining a *control model*, in which cells are described using the basal conductances in the current's equations, and a *non-excitable cell model*. We started considering the array of conductances obtained in condition of σ=0.4, because it is in this condition that we can see the greatest percentage of non excitable-cells and we kept all the combinations of conductances causing non-excitable isolated cells while we replaced the other ones with the basal conductances. In this way we came up with a vector of conductances which generates the same percentage of non-excitable cells as before, but the remaining cells are described by the control model and so they exhibit a basal behaviour.
The following illustrations describe the results obtained by a comparison between simulations performed using this last model and simulations carried out before in the condition σ=0.4 with Maltsev Tissue Model.

In our analyses we have always computed both CL and APA, yet that which, for the most part concerns us is the study of the cycle length which has a greater value as a clinical parameter rather than the amplitude of the AP. Looking at Fig. 3.22 we can notice that in both cases, *control model + non-excitable cells model* and *sigma=0.4 model*, with ρ varying in the range $1 \div 10^3$, cells have synchronized their discharge frequency when the simulation is completed. What is more interesting is that in case of ρ equal to 10^4 MOhm m the percentage of non excitable cells is approximately 20% in the case σ=0.4 model while is only 10% considering the other model. Moreover when we look at the upper part of Fig. 3.24 we can see that for all values of ρ the mean values of CL in the pure σ=0.4 model are always smaller than the ones obtained with the new model. Since in the two mod-

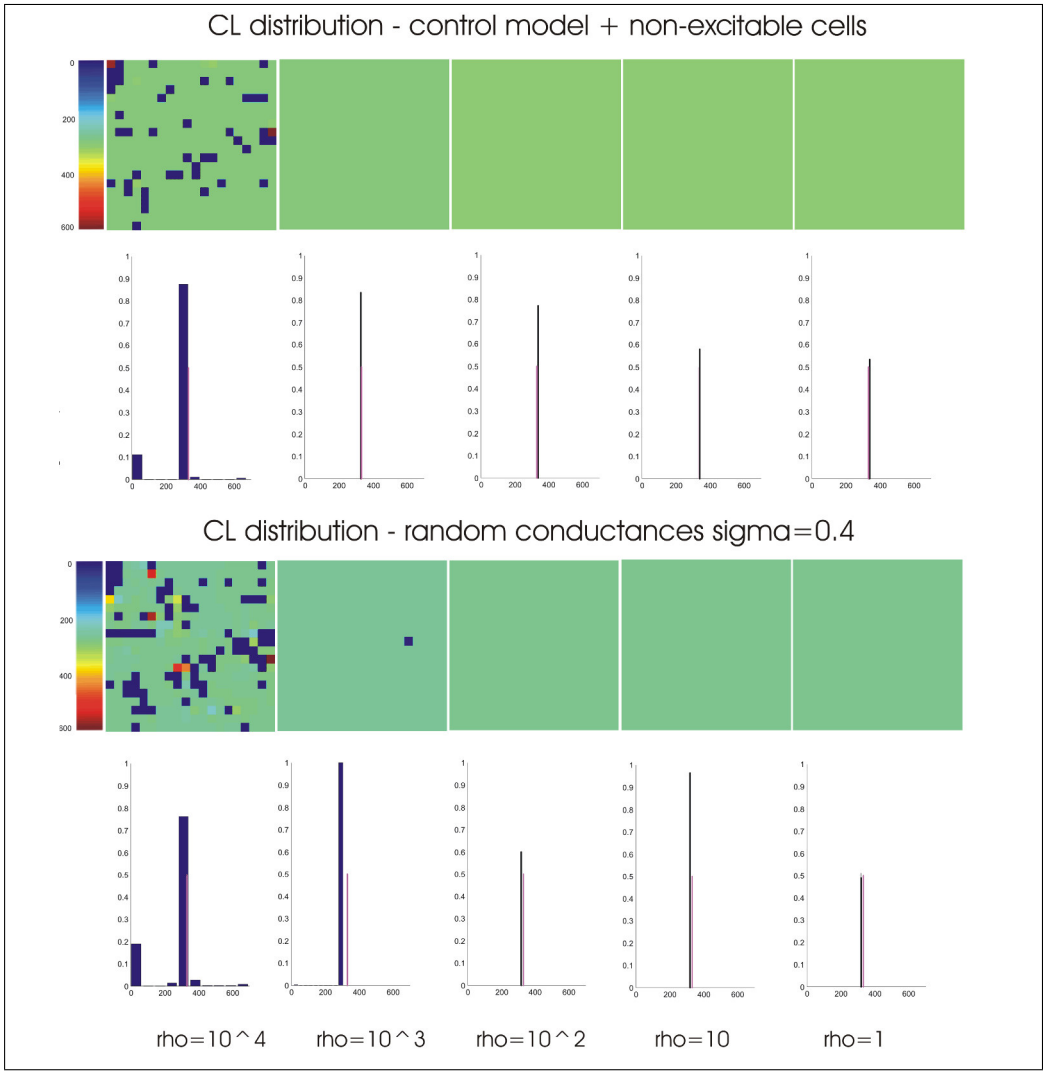

Figure 3.22: CL distributions obtained by 20 s simulations with Maltsev Tissue Model. At the top we can see the results obtained starting from a percentage of non-excitable cells equal to 40% and the remaining cells described using basal conductances values, at the bottom we illustrate again the results obtained with σ=0.4 (without excluding cells with CL=0) to make the comparison easier.

els there is the same number of non-excitable cells and moreover they are placed in the same positions within the matrix and they are characterized

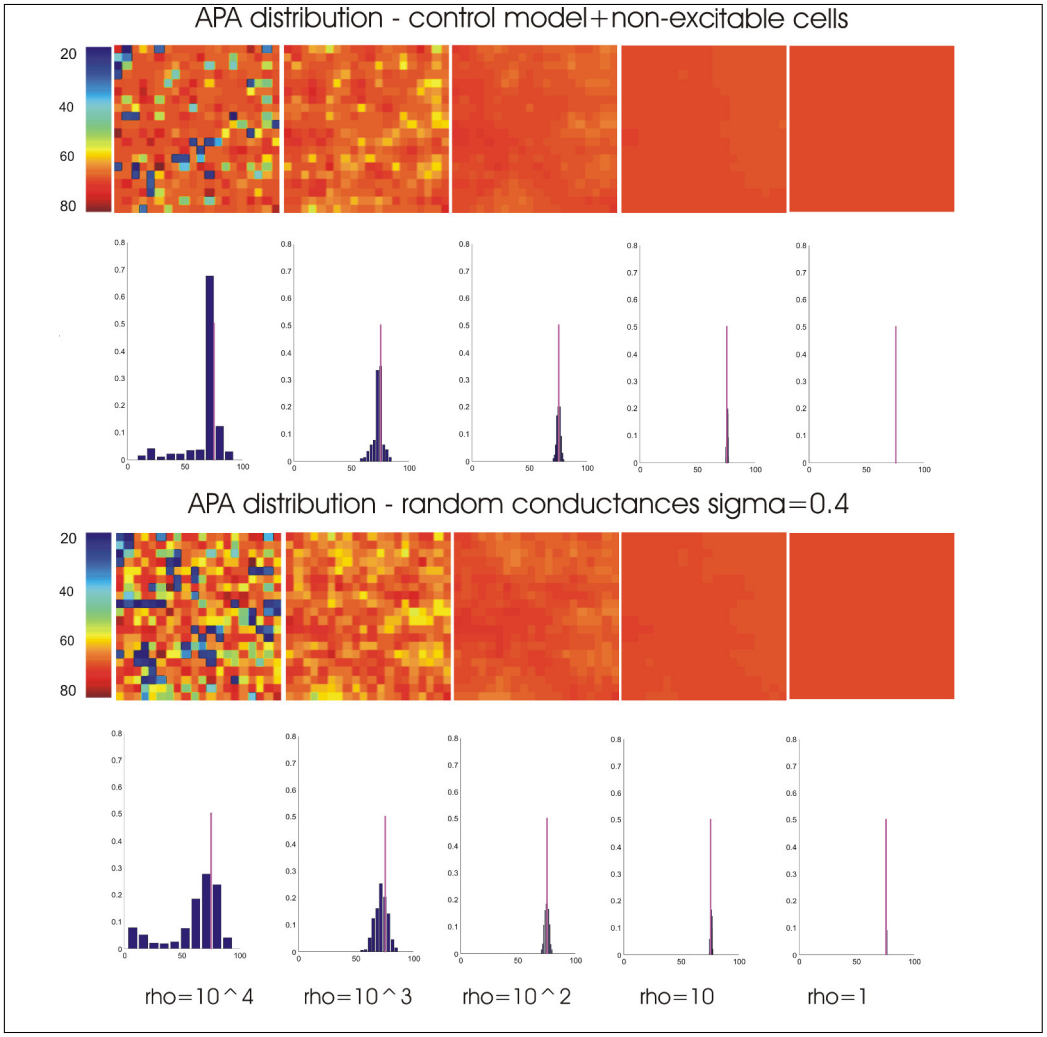

Figure 3.23: APA distributions obtained by 20 s simulations with Maltsev Tissue Model. At the top we can see the results obtained starting from 40% non-excitable cells and the remaining cells described using basal conductances values, at the bottom we illustrate again the results obtained with σ=0.4 to make the comparison easier.

by the same combinations of conductances, we can emphasize that the heterogeneity has effect on the cycle length synchronization and on the final discharge frequency of the SAN cells. Furthermore from these results we can conclude that the decreasing trend of CL mean values with ρ found in our

simulations, i.e. the tendency of the discharge frequency to move toward the fastest one, is not to be attributed to the great percentage of non-excitable cells, but the heterogeneity within the tissue plays a fundamental role.

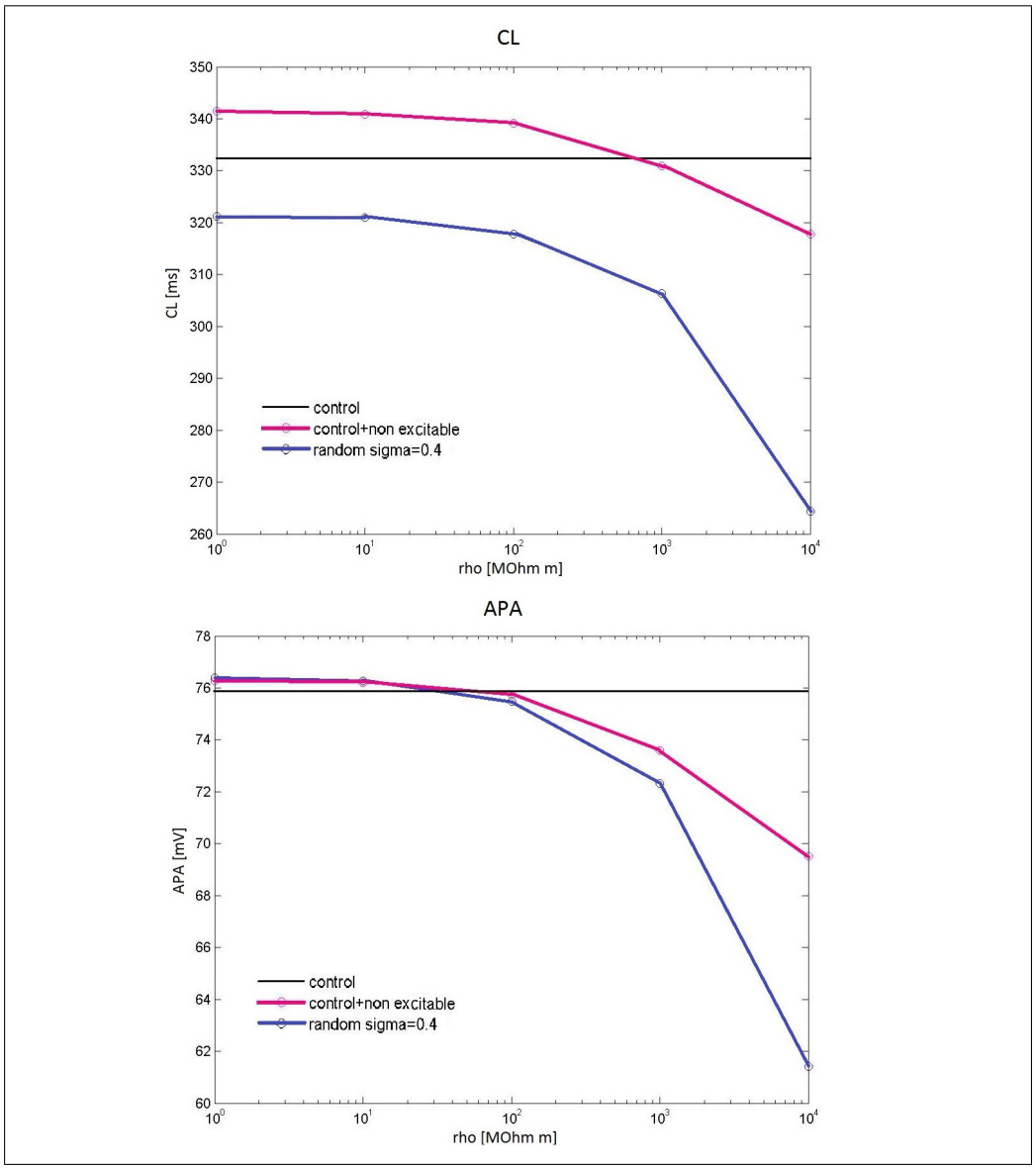

Figure 3.24: Decreasing trend of CL and APA with ρ. Mean values of CL and APA obtained from the two models are plotted.

Conclusions

Cardiac diseases remain the number one cause of death in the Western world, this is probably one of the reasons why the heart is one of the most studied organs of the body. Computational modeling, combined with state-of-the-art experimental techniques helps to gaining deeper insights into the regulation of normal heart function as well as how this regulation may be altered in pathological states. During the years many mathematical models of myocardial cells have been proposed, each one trying to better represent the real behaviour of the cells. Even if we can increase a lot the level of details in these models, they will never give an understanding of the mechanisms of interactions between cells and since they use a unique value for the parameters they cannot take into account the variability normally present within a population of cells of the same organ. Since the heart is structured as a syncytium of cells which can interact and influence each other to an extent that could completely change the behaviour of the same isolated cells, single cell model could predict a behaviour completely different from the real one.
In our study we are primarily interested in the understanding of how tissue heterogeneity can act on a specific cardiac physiological process, that is the genesis of the cardiac rhythm within the specialized tissue of the sinoatrial node. Since the morphological and functional heterogeneity of the SAN cells is a well known fact, during the years several hypotheses have been formulated to achieve a better comprehension of the SAN structure and so to be able to realize more detailed and realistic models of the SAN. Some of these models consider two types of SAN cells or they introduce an increase of gap junction density and conductance from the center to the pheriphery of the SAN. We have implemented a 2-dimensional computational model of the sinoatrial node starting from a single cell model and we have performed a randomization of the maximal values of conductances used in the

model. Each cell in our case has a different discharge frequency and we can compare the behaviour of isolated cells to the one of the same cells put together to form a tissue. The fundamental question we wanted to address through this study was how the cells influence each other's spontaneous beating. In particular, we wanted to find out if the cells synchronize their beating and if they do so, if the fastest cell predominates, becoming the leading pacemaker and imposing its frequency to the other cells or if the ultimate frequency is fairly equal to the average rate of the isolated cells. What we have concluded from our simulations, testing different conditions of inter-cellular variability and coupling resistance is that the population synchronizes on a rate slightly higher than the one of the isolated cells with the basal values of conductances, moreover the rate moderately increases with resistive coupling and with inter-cellular variability, but it does not equal the rate of the fastest cell.
Other questions could be addressed in the future by utilising our model, we could in fact consider the variations of other parameters and not only of the conductances, for example we could study the effect of variability in only one parameter (e.g. in funny current conductance or in calcium clock fluxes) on the heart rate. Moreover, to better describe the SAN heterogeneity we could add details in our model, for example by introducing atrial cell or fibroblast models interspersed within the matrix of SAN cells.
In conclusion, this work provides the starting point for a more comprehensive and realistic approach to the computational analysis of the cardiac pacemaking. This will likely help in addressing several open questions about heart rate and its modulation in physiological and pathological conditions.

Appendix A

Code used to implement Maltsev Tissue Model:

```
    #include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <curand_kernel.h>
# define M_PI 3.14159265358979323846 /* pi */
__global__ void initialConditions(double* vars, int num_param, int num_cells, int cells_per_thread) {

double Vm = -57.9639; double q = 0.69424;double r = 0.0055813;double fCMi = 0.059488;double fCMs = 0.054381;double fCQ = 0.27321;double fTC = 0.029132;double fTMC = 0.43269;double fTMM = 0.50105;double Ca_jsr = 0.31676;double Ca_nsr = 1.4935;double Ca_sub = 0.00013811;double Cai = 0.00015002;double dL = 0.00058455;double fCa = 0.7114;double fL = 0.86238;double dT = 0.0050439;double fT = 0.42076;double paF = 0.14476;double paS = 0.4531;double pi_ = 0.84941;double n = 0.02646;double y = 0.11364;double qa = 0.4238;double qi = 0.44729;double I = 7.8618e-008;double O = 1.734e-007;double R1 = 0.68805;double RI = 0.31195;

    // Within each test, the variables are divided as follows: V(cell1), V(cell2), V(cell3) ... V(cellLast), m(cell1), m(cell2), ... m(cellLast) ... for all 29 parameters

    int idx = threadIdx.x*cells_per_thread;
int simulations = blockIdx.x;
int limit = idx+cells_per_thread;
for (;idx<limit;idx++) {

vars[(simulations*num_param*num_cells) + idx +(0*num_cells)] = Vm;
vars[(simulations*num_param*num_cells) + idx +(1*num_cells)] = q;
vars[(simulations*num_param*num_cells) + idx +(2*num_cells)] = r;
vars[(simulations*num_param*num_cells) + idx +(3*num_cells)] = fCMi;
vars[(simulations*num_param*num_cells) + idx +(4*num_cells)] = fCMs;
vars[(simulations*num_param*num_cells) + idx +(5*num_cells)] = fCQ;
vars[(simulations*num_param*num_cells) + idx +(6*num_cells)] = fTC;
vars[(simulations*num_param*num_cells) + idx +(7*num_cells)] = fTMC;
vars[(simulations*num_param*num_cells) + idx +(8*num_cells)] = fTMM;
vars[(simulations*num_param*num_cells) + idx +(9*num_cells)] = Ca_jsr;
vars[(simulations*num_param*num_cells) + idx +(10*num_cells)] = Ca_nsr;
vars[(simulations*num_param*num_cells) + idx +(11*num_cells)] = Ca_sub;
vars[(simulations*num_param*num_cells) + idx +(12*num_cells)] = Cai;
vars[(simulations*num_param*num_cells) + idx +(13*num_cells)] = dL;
vars[(simulations*num_param*num_cells) + idx +(14*num_cells)] = fCa;
vars[(simulations*num_param*num_cells) + idx +(15*num_cells)] = fL;
vars[(simulations*num_param*num_cells) + idx +(16*num_cells)] = dT;
vars[(simulations*num_param*num_cells) + idx +(17*num_cells)] = fT;
vars[(simulations*num_param*num_cells) + idx +(18*num_cells)] = paF;
vars[(simulations*num_param*num_cells) + idx +(19*num_cells)] = paS;
vars[(simulations*num_param*num_cells) + idx +(20*num_cells)] = pi_;
vars[(simulations*num_param*num_cells) + idx +(21*num_cells)] = n;
vars[(simulations*num_param*num_cells) + idx +(22*num_cells)] = y;
vars[(simulations*num_param*num_cells) + idx +(23*num_cells)] = qa;
vars[(simulations*num_param*num_cells) + idx +(24*num_cells)] = qi;
vars[(simulations*num_param*num_cells) + idx +(25*num_cells)] = I;
vars[(simulations*num_param*num_cells) + idx +(26*num_cells)] = O;
vars[(simulations*num_param*num_cells) + idx +(27*num_cells)] = R1;
vars[(simulations*num_param*num_cells) + idx +(28*num_cells)] = RI;

    }
}
```

```
__global__ void computeState(double* x, float* conductances, double* ion_current, int total_cells, double step, double*
randNums, int variations, double* x_temp, int num_changing_vars, int cells_per_thread) {

int idx = cells_per_thread*threadIdx.x;
int cell_num;
int limit = idx+cells_per_thread;
for (;idx¡limit;idx++) {

cell_num = (blockIdx.x*total_cells*29) + idx;

      //Index Variables to make life easier: Array is categorized by blocks of size=total_cells, each block contains the Vms
of one parameter across the cells

      int Vm_i = 0*total_cells ;
int q_i = 1*total_cells ;
int r_i = 2*total_cells ;
int fCMi_i = 3*total_cells ;
int fCMs_i = 4*total_cells ;
int fCQ_i = 5*total_cells ;
int fTC_i = 6*total_cells ;
int fTMC_i = 7*total_cells ;
int fTMM_i = 8*total_cells ;
int Ca_jsr_i = 9*total_cells ;
int Ca_nsr_i = 10*total_cells ;
int Ca_sub_i = 11*total_cells ;
int Cai_i = 12*total_cells ;
int dL_i = 13*total_cells ;
int fCa_i = 14*total_cells ;
int fL_i = 15*total_cells ;
int dT_i = 16*total_cells ;
int fT_i = 17*total_cells ;
int paF_i = 18*total_cells ;
int paS_i = 19*total_cells ;
int pi__i = 20*total_cells ;
int n_i = 21*total_cells ;
int y_i = 22*total_cells ;
int qa_i = 23*total_cells ;
int qi_i = 24*total_cells ;
int I_i = 25*total_cells ;
int O_i = 26*total_cells ;
int R1_i = 27*total_cells ;
int RI_i = 28*total_cells ;

double E_K, i_to, i_sus, q_infinity, tau_q
; double r_infinity, tau_r, i_CaL, i_CaT, E_Na, i_f_Na, i_f_K, i_f;
double i_st, i_Kr, E_Ks, i_Ks, i_NaK ;
double RTOnF, k32, k43, di, k14, k12, k41, k34, x2, Do, k21, k23, x1, x3, x4, i_NaCa;
double i_b_Ca, i_b_Na, delta_fTC, delta_fTMC, delta_fTMM, delta_fCMi;
double delta_fCMs, delta_fCQ, j_Ca_dif;
double V_sub, j_up, V_cell, V_nsr, V_in, j_SRCarel, V_jsr, j_tr, dL_infinity;
double adVm, alpha_dL, bdVm, beta_dL, tau_dL, fCa_infinity, tau_fCa;
double fL_infinity, tau_fL, dT_infinity, tau_dT, fT_infinity, tau_fT;
double pa_infinity, tau_paS, tau_paF, pi_infinity, tau_pi;
double alpha_n, beta_n, n_infinity, tau_n;
double y_infinity, tau_y, qa_infinity, alpha_qa, beta_qa, tau_qa;
double alpha_qi, beta_qi, qi_infinity, tau_qi;
double kCaSR, koSRCa, kiSRCa, Iion;

// conductances control values /*double g_sus = 0.02;
double g_to = 0.252;
double g_CaT = 0.1832;
double g_Kr = 0.08113973;
double g_Ks = 0.0259;
double g_CaL = 0.464;
double g_b_Ca = 0.0006;
double g_b_Na = 0.00486;
double g_if = 0.15;
double g_st = 0.003; */
double g_sus = conductances[cell_num+0*total_cells] ;
double g_to = conductances[cell_num+1*total_cells] ;
double g_CaL = conductances[cell_num+2*total_cells] ;
double g_CaT = conductances[cell_num+3*total_cells] ;
double g_Kr = conductances[cell_num+4*total_cells] ;
double g_Ks =conductances[cell_num+5*total_cells] ;
double g_b_Ca = conductances[cell_num+6*total_cells] ;
double g_b_Na = conductances[cell_num+7*total_cells] ;
```

```
double g_if = conductances[cell_num+8*total_cells] ;
double g_st = conductances[cell_num+9*total_cells] ;

double Cm=32;double CM_tot = 0.045;double CQ_tot = 10.0;double TC_tot = 0.031;double TMC_tot = 0.062;double
kb_CM = 0.542;double kb_CQ = 0.445;double kb_TC = 0.446;double kb_TMC = 0.00751;double kb_TMM = 0.751;dou-
ble kf_CM = 227.7;double kf_CQ = 0.534;double kf_TC = 88.8;double kf_TMC = 227.7; double kf_TMM = 2.277;double
Km_fCa = 0.00035;double alpha_fCa = 0.021;double E_CaL = 45.0;double E_CaT = 45.0;double K1ni = 395.3;double
K1no = 1628.0;double K2ni = 2.289;double K2no = 561.4;double K3ni = 26.44; double K3no = 4.663;double Kci =
0.0207;double Kcni = 26.44;double Kco = 3.663;double Qci = 0.1369;double Qco = 0.0;double Qn = 0.4315;double
kNaCa = 187.5;double Km_Kp = 1.4;double Km_Nap = 14.0;double i_NaK_max = 2.88;double VIf_half = -64.0;double
E_st = 37.4;

double K_up = 0.0006;double P_up = 0.012;double tau_dif_Ca = 0.04;double tau_tr = 40.0;double EC50_SR = 0.45;double
HSR = 2.5;double MaxSR = 15.0;double MinSR = 1.0;double kiCa = 0.5;double kim = 0.005;double koCa = 10.0;double
kom = 0.06;double ks = 250000.0;double Cao = 2.0;double F = 96485.0;double Ki = 140.0;double Ko = 5.4;double L_cell
= 70.0;double L_sub = 0.02;double Mgi = 2.5;double Nai = 10.0;double Nao = 140.0;double R2 = 8314.4;double R_cell
= 4.0;double T = 310.15;double V_in_part = 0.46;double V_jsr_part = 0.0012;double V_nsr_part = 0.0116;

/*statevar_i = [q ,r ,Vm ,fCMi ,fCMs ,fCQ ,fTC ,fTMC ,fTMM ,Ca_jsr ,Ca_nsr ,Ca_sub ,Cai , dL ,fCa ,fL ,dT ,fT ,paF
,paS ,pi_ ,n ,y ,qa ,qi ,I ,O ,R1 ,RI ];*/

E_K = R2*T/F*log(Ko/Ki) ;
i_to = Cm*g_to*(x[cell_num+Vm_i]-E_K)*(x[cell_num+q_i])*(x[cell_num+r_i]);
i_sus = Cm*g_sus*(x[cell_num+Vm_i]-E_K)*(x[cell_num+r_i]);

q_infinity = 1/(1+exp((x[cell_num+Vm_i]+49)/13));
tau_q=6.06+39.102/(0.57*exp(-0.08*(x[cell_num+Vm_i]+44))+0.065*exp(0.1*(x[cell_num+Vm_i]+45.93)));

r_infinity = 1/(1+exp(-(x[cell_num+Vm_i]-19.3)/15));
tau_r = 2.75352+14.40516/(1.037*exp(0.09*(x[cell_num+Vm_i]+30.61))+0.369*exp(-0.12*( x[cell_num+Vm_i]+23.84)));

i_CaL = Cm*g_CaL*(x[cell_num+Vm_i]-E_CaL)*x[cell_num+dL_i]* x[cell_num+fL_i]* x[cell_num+fCa_i];
i_CaT = Cm*g_CaT*(x[cell_num+Vm_i]-E_CaT)* x[cell_num+dT_i]* x[cell_num+fT_i];

E_Na = R2*T/F*log(Nao/Nai);
i_f_Na = Cm*0.3833*g_if*( x[cell_num+Vm_i]-E_Na)*pow(x[cell_num+y_i],2);
i_f_K = Cm*0.6167*g_if*( x[cell_num+Vm_i]-E_K)* pow(x[cell_num+y_i],2);
i_f = i_f_Na+i_f_K;

i_st = Cm*g_st*( x[cell_num+Vm_i]-E_st) *x[cell_num+qa_i] * x[cell_num+qi_i];

i_Kr = Cm*g_Kr*( x[cell_num+Vm_i]-E_K) *(0.6* x[cell_num+paF_i]+0.4* x[cell_num+paS_i]) * x[cell_num+pi__i];

E_Ks = R2*T/F*log((Ko+0.12*Nao)/(Ki+0.12*Nai));
i_Ks = Cm*g_Ks*( x[cell_num+Vm_i]-E_Ks) *pow(x[cell_num+n_i],2);

i_NaK = Cm*i_NaK_max/((1+pow((Km_Kp/Ko),1.2))*(1+pow((Km_Nap/Nai),1.3))*(1+exp(-(x[cell_num+Vm_i]
-E_Na+120.0) /30.0)));

RTOnF = R2*T/F;
k32 = exp(Qn* x[cell_num+Vm_i]/(2*RTOnF));
k43 = Nai/(K3ni+Nai);
di=1+x[cell_num+Ca_sub_i]/Kci*(1+exp(-Qci*x[cell_num+Vm_i]/RTOnF)+Nai/Kcni)+Nai/K1ni*
(1+Nai/K2ni*(1+Nai/K3ni));
k14 = Nai/K1ni*Nai/K2ni*(1+Nai/K3ni) *exp(Qn* x[cell_num+Vm_i]/(2*RTOnF)) /di;
k12 = x[cell_num+Ca_sub_i] /Kci*exp(-Qci* x[cell_num+Vm_i]/RTOnF) /di;
k41 = exp(-Qn* x[cell_num+Vm_i]/(2*RTOnF));
k34 = Nao/(K3no+Nao);
x2 = k32*k43*(k14+k12)+k41*k12*(k34+k32);
Do = 1+Cao/Kco*(1+exp(Qco* x[cell_num+Vm_i]/RTOnF))+Nao/K1no*(1+Nao/K2no*(1+Nao/K3no));
k21 = Cao/Kco*exp(Qco* x[cell_num+Vm_i]/RTOnF) /Do;
k23 = Nao/K1no*Nao/K2no*(1+Nao/K3no) *exp(-Qn* x[cell_num+Vm_i]/(2*RTOnF))/Do;
x1 = k41*k34*(k23+k21)+k21*k32*(k43+k41);
x3 = k14*k43*(k23+k21)+k12*k23*(k43+k41);
x4 = k23*k34*(k14+k12)+k14*k21*(k34+k32);
i_NaCa = Cm*kNaCa*(x2*k21-x1*k12)/(x1+x2+x3+x4);

i_b_Ca = Cm*g_b_Ca*( x[cell_num+Vm_i]-E_CaL);
i_b_Na = Cm*g_b_Na*( x[cell_num+Vm_i]-E_Na);

delta_fTC = kf_TC* x[cell_num+Cai_i] *(1- x[cell_num+fTC_i])-kb_TC* x[cell_num+fTC_i];
delta_fTMC = kf_TMC* x[cell_num+Cai_i] *(1-( x[cell_num+fTMC_i] + x[cell_num+fTMM_i]))-kb_TMC*
x[cell_num+fTMC_i];
delta_fTMM = kf_TMM*Mgi*(1-( x[cell_num+fTMC_i]+ x[cell_num+fTMM_i]))-kb_TMM*x[cell_num+fTMM_i];
delta_fCMi = kf_CM* x[cell_num+Cai_i] *(1- x[cell_num+fCMi_i])-kb_CM*x[cell_num+fCMi_i];
delta_fCMs = kf_CM* x[cell_num+Ca_sub_i]*(1- x[cell_num+fCMs_i])-kb_CM* x[cell_num+fCMs_i];
```

```
delta_fCQ = kf_CQ* x[cell_num+Ca_jsr_i]*(1- x[cell_num+fCQ_i])-kb_CQ* x[cell_num+fCQ_i];

j_Ca_dif = (x[cell_num+Ca_sub_i]- x[cell_num+ Cai_i])/tau_dif_Ca;

V_sub = 0.001*2*M_PI *L_sub*(R_cell-L_sub/2)*L_cell;//pi=M_PI
j_up = P_up/(1+K_up/ x[cell_num+ Cai_i]);
V_cell = 0.001*M_PI *pow(R_cell,2)*L_cell;
V_nsr = V_nsr_part*V_cell;
V_in = V_in_part*V_cell-V_sub;

j_SRCarel = ks* x[cell_num+O_i]*( x[cell_num+Ca_jsr_i]- x[cell_num+Ca_sub_i]);
V_jsr = V_jsr_part*V_cell;
j_tr = (x[cell_num+Ca_nsr_i] - x[cell_num+Ca_jsr_i])/tau_tr;
dL_infinity = 1/(1+exp(-(x[cell_num+Vm_i] +13.5)/6));

if (x[cell_num+Vm_i] == -35) {
adVm = -35.00001;
}
else if (x[cell_num+Vm_i] == 0) {
adVm = 0.00001;
}
else {
adVm = x[cell_num+Vm_i];
}

alpha_dL = -0.02839*(adVm+35)/(exp(-(adVm+35)/2.5)-1)-0.0849*adVm/(exp(-adVm/4.8)-1);
if (x[cell_num+Vm_i] == 5) {
bdVm = 5.00001;
}
else {
bdVm = x[cell_num+Vm_i];
}
beta_dL = 0.01143*(bdVm-5)/(exp((bdVm-5)/2.5)-1);
tau_dL = 1/(alpha_dL+beta_dL);
fCa_infinity = Km_fCa/(Km_fCa+ x[cell_num+Ca_sub_i]);
tau_fCa = fCa_infinity/alpha_fCa;
fL_infinity = 1/(1+exp((x[cell_num+Vm_i]+35)/7.3));
tau_fL = 44.3+257.1*exp(-pow(((x[cell_num+Vm_i]+32.5)/13.9),2));

dT_infinity = 1.0/(1+exp(-(x[cell_num+Vm_i]+26.3)/6));
tau_dT = 1/(1.068*exp((x[cell1_num+Vm_i]+26.3)/30)+1.068*exp(-(x[cell_num+Vm_i]+26.3)/30));
fT_infinity = 1/(1+exp((x[cell_num+Vm_i]+61.7)/5.6));
tau_fT = 1/(0.0153*exp(-(x[cell_num+Vm_i]+61.7)/83.3)+0.015*exp((x[cell_num+Vm_i]+61.7)/15.38));

pa_infinity = 1/(1+exp(-(x[cell_num+Vm_i]+23.2)/10.6));
tau_paS = 0.84655354/(0.0042*exp(x[cell_num+Vm_i]/17)+0.00015*exp(-x[cell_num+Vm_i]/21.6));
tau_paF = 0.84655354/(0.0372*exp(x[cell_num+Vm_i]/15.9)+0.00096*exp(-x[cell_num+Vm_i]/22.5));
pi_infinity = 1/(1+exp((x[cell_num+Vm_i]+28.6)/17.1));
tau_pi = 1/(0.1*exp(-x[cell_num+Vm_i]/54.645)+0.656*exp(x[cell_num+Vm_i]/106.157));

alpha_n = 0.014/(1+exp(-(x[cell_num+Vm_i]-40)/9));
beta_n = 0.001*exp(-x[cell_num+Vm_i]/45);
n_infinity = alpha_n/(alpha_n+beta_n);
tau_n = 1/(alpha_n+beta_n);

y_infinity = 1/(1+exp((x[cell_num+Vm_i]-VIf_half)/13.5));
tau_y = 0.7166529/(exp(-(x[cell_num+Vm_i]+386.9)/45.302)+exp((x[cell_num+Vm_i]-73.08)/19.231));

qa_infinity = 1/(1+exp(-(x[cell_num+Vm_i]+57)/5));
alpha_qa = 1/(0.15*exp(-x[cell_num+Vm_i]/11)+0.2*exp(-x[cell_num+Vm_i]/700));
beta_qa = 1/(16*exp(x[cell_num+Vm_i]/8)+15*exp(x[cell_num+Vm_i]/50));
tau_qa = 1/(alpha_qa+beta_qa);
alpha_qi = 1/(3100*exp(x[cell_num+Vm_i]/13)+700*exp(x[cell_num+Vm_i]/70));
beta_qi = 1/(95*exp(-x[cell_num+Vm_i]/10)+50*exp(-x[cell_num+Vm_i]/700))+0.000229/(1+exp(-x[cell_num+Vm_i]/5));
qi_infinity = alpha_qi/(alpha_qi+beta_qi);
tau_qi = 6.65/(alpha_qi+beta_qi);
kCaSR = MaxSR-(MaxSR-MinSR)/(1+pow((EC50_SR/x[cell_num+Ca_jsr_i]),HSR));
koSRCa = koCa/kCaSR;
kiSRCa = kiCa*kCaSR;

Iion=i_CaL+i_CaT+i_f+i_st+i_Kr+i_Ks+i_to+i_sus+i_NaK+i_NaCa+i_b_Ca+i_b_Na; ion_current[cell_num] = Iion;

// differential equations

if ( !isinf(x[cell_num+q_i] + step*((q_infinity-x[cell_num+q_i])/tau_q)) && !isnan(x[cell_num+q_i] +
step*((q_infinity-x[cell_num+q_i])/tau_q))) {
```

```
x_temp[cell_num+q_i] = x[cell_num+q_i] + step*((q_infinity-x[cell_num+q_i])/tau_q);
}
else { x_temp[cell_num+q_i] = x[cell_num+q_i]; }

if ( !isinf(x[cell_num+r_i] + step*( (r_infinity-x[cell_num+r_i])/tau_r)) && !isnan(x[cell_num+r_i]
+ step*( (r_infinity-x[cell_num+r_i])/tau_r))) {
x_temp[cell_num+r_i] = x[cell_num+r_i] + step*( (r_infinity-x[cell_num+r_i])/tau_r);
}
else { x_temp[cell_num+r_i] = x[cell_num+r_i]; }

if ( !isinf( x[cell_num+fCMi_i] + step*(delta_fCMi) ) && !isnan( x[cell_num+fCMi_i] + step*(delta_fCMi))) {
x_temp[cell_num+fCMi_i] = x[cell_num+fCMi_i] + step*(delta_fCMi) ;
}
else { x_temp[cell_num+fCMi_i] = x[cell_num+fCMi_i]; }

if ( !isinf( x[cell_num+fCMs_i] + step*(delta_fCMs) ) && !isnan( x[cell_num+fCMs_i] +
step*(delta_fCMs))) {
x_temp[cell_num+fCMs_i] = x[cell_num+fCMs_i] + step*(delta_fCMs);
}
else { x_temp[cell_num+fCMs_i] = x[cell_num+fCMs_i]; }

if ( !isinf( x[cell_num+fCQ_i] + step*(delta_fCQ) ) && !isnan( x[cell_num+fCQ_i] + step*(delta_fCQ))) {
x_temp[cell_num+fCQ_i] = x[cell_num+fCQ_i] + step*( delta_fCQ ) ;
}
else { x_temp[cell_num+fCQ_i] = x[cell_num+fCQ_i]; }


      if ( !isinf( x[cell_num+fTC_i] + step*( (delta_fTC))) && !isnan( x[cell_num+dL_i] + step*( (delta_fTC)))) {
x_temp[cell_num+fTC_i] = x[cell_num+fTC_i] + step*( delta_fTC) ;
}
else { x_temp[cell_num+fTC_i] = x[cell_num+fTC_i]; }

if ( !isinf( x[cell_num+fTMC_i] + step*(delta_fTMC) )&& !isnan( x[cell_num+fTMC_i] +
step*(delta_fTMC)) ) {
x_temp[cell_num+fTMC_i] = x[cell_num+fTMC_i] + step*( delta_fTMC );
}
else { x_temp[cell_num+fTMC_i] = x[cell_num+fTMC_i]; }


      if ( !isinf( x[cell_num+fTMM_i] + step*(delta_fTMM) ) && !isnan( x[cell_num+fTMM_i] +
step*(delta_fTMM) )) {
x_temp[cell_num+fTMM_i] = x[cell_num+fTMM_i] + step*( delta_fTMM) ;
}
else { x_temp[cell_num+fTMM_i] = x[cell_num+fTMM_i]; }

if ( !isinf(x[cell_num+Ca_jsr_i] + step*( j_tr-(j_SRCarel+CQ_tot*delta_fCQ) )) &&
!isnan(x[cell_num+Ca_jsr_i] + step*( j_tr-(j_SRCarel+CQ_tot*delta_fCQ) ))){
x_temp[cell_num+Ca_jsr_i] = x[cell_num+Ca_jsr_i] + step*( j_tr-(j_SRCarel+CQ_tot*delta_fCQ) );
}
else { x_temp[cell_num+Ca_jsr_i] = x[cell_num+Ca_jsr_i]; }

if ( !isinf(x[cell_num+Ca_nsr_i] + step*( j_up-j_tr*V_jsr/V_nsr )) && !isnan( x[cell_num+Ca_nsr_i] + step*( j_up-j_tr*V_jsr
/V_nsr ))){
x_temp[cell_num+Ca_nsr_i] = x[cell_num+Ca_nsr_i] + step*( j_up-j_tr*V_jsr/V_nsr );
}
else { x_temp[cell_num+Ca_nsr_i] = x[cell_num+Ca_nsr_i]; }

if ( !isinf( x[cell_num+Ca_sub_i] + step*( (j_SRCarel*V_jsr/V_sub-((i_CaL+i_CaT+i_b_Ca-
2*i_NaCa)/(2*F*V_sub)+j_Ca_dif+CM_tot*delta_fCMs)) ))
&& !isnan( x[cell_num+Ca_sub_i] + step*(
(j_SRCarel*V_jsr/V_sub-((i_CaL+i_CaT+i_b_Ca-2*i_NaCa)
/(2*F*V_sub)+j_Ca_dif+CM_tot*delta_fCMs)) ))){
x_temp[cell_num+Ca_sub_i] = x[cell_num+Ca_sub_i] + step*( (j_SRCarel*V_jsr/V_sub-((i_CaL+i_CaT+i_b_Ca-2*i_NaCa)/
(2*F*V_sub)+j_Ca_dif+CM_tot*delta_fCMs)) ) ; }
else { x_temp[cell_num+Ca_sub_i] = x[cell_num+Ca_sub_i]; }


      if ( !isinf( x[cell_num+Cai_i] + step*( (j_Ca_dif*V_sub-j_up*V_nsr)/V_in-(CM_tot*delta_fCMi+TC_tot*delta_fTC
+TMC_tot*delta_fTMC) )) && !isnan( x[cell_num+Cai_i] + step*( (j_Ca_dif*V_sub-j_up*V_nsr)/V_in-(CM_tot*delta_fCMi+
TC_tot*delta_fTC
+TMC_tot*delta_fTMC) ))) {
x_temp[cell_num+Cai_i] = x[cell_num+Cai_i] + step*( (j_Ca_dif*V_sub-j_up*V_nsr)/V_in-(CM_tot*delta_fCMi
+TC_tot*delta_fTC+TMC_tot*delta_fTMC) ) ; }
else { x_temp[cell_num+Cai_i] = x[cell_num+Cai_i]; }

if ( !isinf(x[cell_num+dL_i] + step*( (dL_infinity-x[cell_num+dL_i])/tau_dL )) && !isnan(
```

```
x[cell_num+dL_i] + step*( (dL_infinity-x[cell_num+dL_i])/tau_dL ))) {
x_temp[cell_num+dL_i] = x[cell_num+dL_i] + step*( (dL_infinity-x[cell_num+dL_i])/tau_dL ) ;
}
else { x_temp[cell_num+dL_i] = x[cell_num+dL_i]; }

if ( !isinf( x[cell_num+fCa_i] + step*( (fCa_infinity-x[cell_num+fCa_i])/tau_fCa )) && !isnan( x[cell_num+fCa_i] + step*(
(fCa_infinity-x[cell_num+fCa_i])/tau_fCa ))) {
x_temp[cell_num+fCa_i] = x[cell_num+fCa_i] + step*( (fCa_infinity-x[cell_num+fCa_i])/tau_fCa ) ; }
else { x_temp[cell_num+fCa_i] = x[cell_num+fCa_i]; }
if ( !isinf( x[cell_num+fL_i] + step*( (fL_infinity-x[cell_num+fL_i])/tau_fL )) && !isnan( x[cell_num+fL_i] + step*(
(fL_infinity-x[cell_num+fL_i])/tau_fL ))) {
x_temp[cell_num+fL_i] = x[cell_num+fL_i] + step*( (fL_infinity-x[cell_num+fL_i])/tau_fL ) ; }
else { x_temp[cell_num+fL_i] = x[cell_num+fL_i]; }

if ( !isinf( x[cell_num+dT_i] + step*( (dT_infinity-x[cell_num+dT_i])/tau_dT )) && !isnan( x[cell_num+dT_i] + step*(
(dT_infinity-x[cell_num+dT_i])/tau_dT))) {
x_temp[cell_num+dT_i] = x[cell_num+dT_i] + step*( (dT_infinity-x[cell_num+dT_i])/tau_dT) ; }
else { x_temp[cell_num+dT_i] = x[cell_num+dT_i]; }

if ( !isinf( x[cell_num+fT_i] + step*( (fT_infinity-x[cell_num+fT_i])/tau_fT)) && !isnan( x[cell_num+fT_i] + step*(
(fT_infinity-x[cell_num+fT_i])/tau_fT))) {
x_temp[cell_num+fT_i] = x[cell_num+fT_i] + step*( (fT_infinity-x[cell_num+fT_i])/tau_fT) ; }
else { x_temp[cell_num+fT_i] = x[cell_num+fT_i]; }

if ( !isinf( x[cell_num+paF_i] + step*( (pa_infinity-x[cell_num+paF_i])/tau_paF)) && !isnan( x[cell_num+paF_i] + step*(
(pa_infinity-x[cell_num+paF_i])/tau_paF))) {
x_temp[cell_num+paF_i] = x[cell_num+paF_i] + step*( (pa_infinity-x[cell_num+paF_i])/tau_paF) ;
}
else { x_temp[cell_num+paF_i] = x[cell_num+paF_i]; }

if ( !isinf( x[cell_num+paS_i] + step*( (pa_infinity-x[cell_num+paS_i])/tau_paS)) && !isnan( x[cell_num+paS_i] + step*(
(pa_infinity-x[cell_num+paS_i])/tau_paS))) {
x_temp[cell_num+paS_i] = x[cell_num+paS_i] + step*( (pa_infinity-x[cell_num+paS_i])/tau_paS);
}
else {
x_temp[cell_num+paS_i] = x[cell_num+paS_i]; }

if ( !isinf( x[cell_num+pi__i] + step*( (pi_infinity-x[cell_num+pi__i])/tau_pi)) && !isnan( x[cell_num+pi__i] + step*(
(pi_infinity-x[cell_num+pi__i])/tau_pi))) {
x_temp[cell_num+pi__i] = x[cell_num+pi__i] + step*( (pi_infinity-x[cell_num+pi__i])/tau_pi) ;
}
else { x_temp[cell_num+pi__i] = x[cell_num+pi__i]; }

if ( !isinf( x[cell_num+n_i] + step*( (n_infinity-x[cell_num+n_i])/tau_n)) && !isnan( x[cell_num+n_i] + step*( (n_infinity-
x[cell_num+n_i])/tau_n))) {
x_temp[cell_num+n_i] = x[cell_num+n_i] + step*( (n_infinity-x[cell_num+n_i])/tau_n) ;
}
else { x_temp[cell_num+n_i] = x[cell_num+n_i]; }

if ( !isinf( x[cell_num+y_i] + step*( (y_infinity-x[cell_num+y_i])/tau_y )) && !isnan( x[cell_num+y_i] + step*( (y_infinity-
x[cell_num+y_i])/tau_y ))) {
x_temp[cell_num+y_i] = x[cell_num+y_i] + step*( (y_infinity-x[cell_num+y_i])/tau_y) ;
}
else { x_temp[cell_num+y_i] = x[cell_num+y_i]; }

if ( !isinf( x[cell_num+qa_i] + step*( (qa_infinity-x[cell_num+qa_i])/tau_qa )) && !isnan(
x[cell_num+qa_i] + step*( (qa_infinity-x[cell_num+qa_i])/tau_qa ))) {
x_temp[cell_num+qa_i] = x[cell_num+qa_i] + step*( (qa_infinity-x[cell_num+qa_i])/tau_qa ) ;
}
else { x_temp[cell_num+qa_i] = x[cell_num+qa_i]; }

if ( !isinf( x[cell_num+qi_i] + step*( (qi_infinity-x[cell_num+qi_i])/tau_qi )) && !isnan(
x[cell_num+qi_i] + step*( (qi_infinity-x[cell_num+qi_i])/tau_qi ))) {
x_temp[cell_num+qi_i] = x[cell_num+qi_i] + step*( (qi_infinity-x[cell_num+qi_i])/tau_qi) ;
}
else { x_temp[cell_num+qi_i] = x[cell_num+qi_i]; }

if ( !isinf(x[cell_num+I_i] + step*( kiSRCa*x[cell_num+Ca_sub_i]*x[cell_num+O_i]-kim*x[cell_num+I_i]-(kom*x[cell_num+I_i]-
koSRCa*pow(x[cell_num+Ca_sub_i],2)*x[cell_num+RI_i]) )) && !isnan(x[cell_num+I_i] + step*( kiSRCa*x[cell_num+Ca_sub_i]
x[cell_num+O_i]-kim*x[cell_num+I_i]-(kom*x[cell_num+I_i]-koSRCa*pow(x[cell_num+Ca_sub_i],2)*x[cell_num+RI_i]) ))){
x_temp[cell_num+I_i] = x[cell_num+I_i] + step*( kiSRCa*x[cell_num+Ca_sub_i]*x[cell_num+O_i]-kim*x[cell_num+I_i]-
(kom*x[cell_num+I_i]-koSRCa*pow(x[cell_num+Ca_sub_i,2)*x[cell_num+RI_i]) );
}
else { x_temp[cell_num+I_i] = x[cell_num+I_i]; }

if ( !isinf(x[cell_num+O_i] + step*( koSRCa*pow(x[cell_num+Ca_sub_i],2)*x[cell_num+R1_i]-
```

```
kom*x[cell_num+O_i]-(kiSRCa*x[cell_num+Ca_sub_i]*x[cell_num+O_i]-kim*x[cell_num+I_i]) )) && !isnan(x[cell_num+O_i]
+
step*( koSRCa*pow(x[cell_num+Ca_sub_i],2)
x[cell_num+R1_i]-kom*x[cell_num+O_i]-(kiSRCa
x[cell_num+Ca_sub_i]*x[cell_num+O_i]-kim*x[cell_num+I_i]))) ){
x_temp[cell_num+O_i] = x[cell_num+O_i] + step*( koSRCa*pow(x[cell_num+Ca_sub_i],2)*x[cell_num+R1_i]-kom*x[cell_num+O_i]-
(kiSRCa*x[cell_num+Ca_sub_i]*x[cell_num+O_i]-kim*x[cell_num+I_i]) );
}
else { x_temp[cell_num+O_i] = x[cell_num+O_i]; }

      if ( !isinf(x[cell_num+R1_i] + step*(
kim*x[cell_num+RI_i]-kiSRCa*x[cell_num+Ca_sub_i]*x[cell_num+R1_i]-(koSRCa*pow(x[cell_num+Ca_sub_i],2)*x[cell_num+R1_i]-
kom*x[cell_num+O_i]) )) && !isnan(x[cell_num+R1_i] + step*( kim*x[cell_num+RI_i]-kiSRCa*x[cell_num+Ca_sub_i]*x[cell_num+R1_i]-
(koSRCa*pow(x[cell_num+Ca_sub_i],2)*x[cell_num+R1_i]-kom*x[cell_num+O_i]) ))){
x_temp[cell_num+R1_i] = x[cell_num+R1_i] + step*(
kim*x[cell_num+RI_i]-kiSRCa*x[cell_num+Ca_sub_i]*x[cell_num+R1_i]-(koSRCa*pow(x[cell_num+Ca_sub_i],2)*x[cell_num+R1_i]-
kom*x[cell_num+O_i]) );
}
else { x_temp[cell_num+R1_i] = x[cell_num+R1_i]; }

      if ( !isinf(x[cell_num+RI_i] + step*( kom*x[cell_num+I_i]-
koSRCa*pow(x[cell_num+Ca_sub_i],2)*x[cell_num+RI_i]-(kim*x[cell_num+RI_i]-kiSRCa*x[cell_num+Ca_sub_i]*x[cell_num+R1_i])
))
&& !isnan(x[cell_num+RI_i] + step*( kom*x[cell_num+I_i]-
koSRCa*pow(x[cell_num+Ca_sub_i],2)*x[cell_num+RI_i]-(kim*x[cell_num+RI_i]-kiSRCa*x[cell_num+Ca_sub_i]*x[cell_num+R1_i])
))){
x_temp[cell_num+RI_i] = x[cell_num+RI_i] + step*( kom*x[cell_num+I_i]-koSRCa*pow(x[cell_num+Ca_sub_i],2)*x[cell_num+RI_i]-
(kim*x[cell_num+RI_i]-kiSRCa*x[cell_num+Ca_sub_i]*x[cell_num+R1_i]) ); }
else { x_temp[cell_num+RI_i] = x[cell_num+RI_i]; }
}
}

      __global__ void updateState(double* x, double* x_temp, int num_cells, int cells_per_thread) {
int i;
int idx = cells_per_thread*threadIdx.x;
int variations = blockIdx.x;

      int limit = idx+cells_per_thread;
for (;idx<limit;idx++) {

      for (i=1;i<29;i++) {
x[(variations*29*num_cells) + idx +(i*num_cells)] = x_temp[(variations*29*num_cells) + idx
+(i*num_cells)];
}
}

      }

__global__ void compute_voltage(double* x, double* Vm, double* Iion, double step, double* randNums, int variations,
int length, int width, int num_changing_vars, int time, double stimDur, double stimAmp, int tstim, double* s2time, int
cells_per_thread, bool s2_analysis, int s2_loc) {

int num_cells = length*width;
int m;
int n;
double stim = 0.0;
double Istim1 = 0.0;
double Istim2 = 0.0;
double Vnet_R, Vnet_L, Vnet_U, Vnet_D;
double rad = 4 ;
double deltx = 70 ;
double rho;
double Cm=32;
double Rmyo;
double gj;
int tstim2;

int idx = cells_per_thread*threadIdx.x;
int limit = idx+cells_per_thread;

for (;idx<limit;idx++) {
m = (blockIdx.x * num_cells) + idx;
n = (blockIdx.x * num_cells*29) + idx;
if (num_changing_vars==0) {
gj = 1.27 ;
Rmyo = 526;
```

```
      }
else {
gj = 1.27*randNums[(blockIdx.x*num_changing_vars)+15] ;
Rmyo = 526*randNums[(blockIdx.x*num_changing_vars)+16];
}

rho = 10000; // total resistivity

if (s2_analysis) {
tstim2 = s2time[blockIdx.x]/step;
}

      if ( time%tstim > (stimDur/step)) {Istim1 = 0.0;}
else { Istim1 = stimAmp;}

      if (s2_analysis) {
if ( time>=tstim2 && time<=(stimDur/step)+tstim2) {Istim2 = -150;}
else {Istim2 = 0.0;}
}

// Cable Model

if (width==1) {
if(idx==0) {
if ( !isinf( (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*((x[n+1]-x[n])) -
(Iion[n]+Istim1) /Cm )) && !isnan( (x[n]) + (step)*(
(1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*((x[n+1]-x[n])) - (Iion[n]+Istim1) /Cm )) ) {
Vm[m] = (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*((x[n+1]-x[n])) -
(Iion[n]+Istim1) /Cm ) ;
//V[m] = (x[n]) + (step)*( rad/(2*rho*Cm*deltx*deltx)*(x[n+1]-2*x[n] + x[n+length-1]) -
(Iion[n]+Istim1) /Cm ) ; // loop
}
else { Vm[m] = x[n];}
}
else if(idx==num_cells-1){ //last if ( !isinf( (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(-x[n] + x[n-1])
- (Iion[n]) /Cm
)) && !isnan( (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(-x[n] + x[n-1]) -
(Iion[n]) /Cm )) ) {
Vm[m] = (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(-x[n] + x[n-1]) - (Iion[n]) /Cm
); //V[m] = (x[n]) + (step)*( rad/(2*rho*Cm*deltx*deltx)*(x[n+1-length] - 2*x[n] + x[n-1]) - (Iion[n]) /Cm ); // loop
}
else { Vm[m] = x[n]; }
}
else if(idx==1) { if ( !isinf( (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(x[n+1]-2*x[n]+x[n-1]) -
(Iion[n]+Istim1)/ Cm)) && !isnan( (x[n]) + (step)*(
(1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(x[n+1]-2*x[n]+x[n-1]) - (Iion[n]+Istim1)/ Cm)) ) {
Vm[m] = (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(x[n+1]-2*x[n]+x[n-1]) -
(Iion[n]+Istim1)/ Cm);
}
else { Vm[m] = x[n]; }
}
else if(idx==2) {
if ( !isinf( (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(x[n+1]-2*x[n]+x[n-1]) -
(Iion[n]+Istim1)/ Cm)) && !isnan( (x[n]) + (step)*(
(1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(x[n+1]-2*x[n]+x[n-1]) - (Iion[n]+Istim1)/ Cm)) ) {
Vm[m] = (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(x[n+1]-2*x[n]+x[n-1]) -
(Iion[n]+Istim1)/ Cm);
}
else { Vm[m] = x[n]; }
}
//stim2 else if (s2_analysis && s2_loc == idx) {
if ( !isinf( (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(x[n+1]-2*x[n]+x[n-1]) -
(Iion[n]+Istim2)/ Cm)) && !isnan( (x[n]) + (step)*(
(1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(x[n+1]-2*x[n]+x[n-1]) - (Iion[n]+Istim2)/ Cm)) ) {
Vm[m] = (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(x[n+1]-2*x[n]+x[n-1]) -
(Iion[n]+Istim2)/ Cm);
}
else { Vm[m] = x[n]; }
}
else if (s2_analysis && s2_loc+1 == idx) {
if ( !isinf( (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(x[n+1]-2*x[n]+x[n-1]) -
(Iion[n]+Istim2)/ Cm)) && !isnan( (x[n]) + (step)*(
(1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(x[n+1]-2*x[n]+x[n-1]) - (Iion[n]+Istim2)/ Cm)) ) {
Vm[m] = (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(x[n+1]-2*x[n]+x[n-1]) -
(Iion[n]+Istim2)/ Cm);
```

```
}
else { Vm[m] = x[n]; }
}
else if (s2_analysis && s2_loc-1 == idx) {
if ( !isinf( (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(x[n+1]-2*x[n]+x[n-1]) -
(Iion[n]+Istim2)/ Cm)) && !isnan( (x[n]) + (step)*(
(1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(x[n+1]-2*x[n]+x[n-1]) - (Iion[n]+Istim2)/ Cm)) ) {
Vm[m] = (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(x[n+1]-2*x[n]+x[n-1]) -
(Iion[n]+Istim2)/ Cm);
}
else { Vm[m] = x[n]; }
}
else {
if ( !isinf( (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*( (x[n-1]-x[n]) + (x[n+1]-x[n]) )
- (Iion[n]) /Cm ) ) && !isnan ((x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*( (x[n-1]-x[n]) +
(x[n+1]-x[n]) ) - (Iion[n]) /Cm )) ) {
Vm[m] = (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*( (x[n-1]-x[n]) + (x[n+1]-x[n]) ) - (Iion[n]) /Cm );
}
else { Vm[m] = x[n]; }
}
}

//Tissue Model

else {
// set which cells will get a stimulus
if (idx==0 —— idx==1) { stim=Istim1; }
if (idx==2 —— idx==0+length) { stim=Istim1; }
if (idx==1+length —— idx==2+length) { stim=Istim1; }
if (idx==0+2*length —— idx==1+2*length) { stim=Istim1; }
if (idx==2+2*length) { stim=Istim1; }


      if ( threadIdx.x¿=0 && threadIdx.x<=length-1 ) { // Top Edge
Vnet_U = 0.0;
}
else {
Vnet_U = x[n-length] - x[n];
}

if ( threadIdx.x>=((width*length)-length) && threadIdx.x<=((width*length)-1) ) { // Bottom Edge
Vnet_D = 0.0;
}
else {
Vnet_D = x[n+length] - x[n];
}


      if ( threadIdx.x%length==0 ) { // Left Edge
Vnet_L = 0.0;
//Vnet_L = x[n+length-1] - x[n]; // tissue loop
}
else {
Vnet_L = x[n-1] - x[n];
}

if ( threadIdx.x%length==(length-1) ) { // Right Edge
Vnet_R = 0.0;
//Vnet_R = x[n+1-length] - x[n]; // tissue loop
}
else {
Vnet_R = x[n+1] - x[n];
}

if ( !isinf( (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(Vnet_R+Vnet_L+Vnet_U+Vnet_D) -
(Iion[n]+stim) /Cm )) && !isnan( (x[n]) + (step)*(
(1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(Vnet_R+Vnet_L+Vnet_U+Vnet_D) - (Iion[n]+stim) /Cm )) ) {
Vm[m] = (x[n]) + (step)*( (1e3*pow(rad,2)*M_PI/(rho*Cm*deltx))*(Vnet_R+Vnet_L+Vnet_U+Vnet_D) -
(Iion[n]+stim) /Cm ) ;
}
else { Vm[m] = x[n]; }

      }

      }
}
```

```
__global__ void update_voltage(double* x, double* Vm, int total_cells, int cells_per_thread) {

int idx = cells_per_thread*threadIdx.x;

int limit = idx+cells_per_thread;
for (;idx¡limit;idx++) {
int m = (blockIdx.x * total_cells) + idx;
int n = (blockIdx.x * total_cells*29) + idx;
x[n] = Vm[m];
}
}
...

      int main( int argc, const char* argv[] ) {

      int i, ii;
int time = 0;
FILE *fV = fopen("Maltsev GPU Voltage", "w");
FILE *ft = fopen("Maltsev GPU Time", "w");
FILE *output = fopen("Maltsev GPU Sensitivity Analysis", "w");
FILE *s2output = fopen("Maltsev GPU s2 Analysis", "w");
int index=0;
double* host_vars;
double* dev_vars;
double* dev_ion_currents;
double* dev_x_temp;
double* host_Vtemp;
double* dev_Vtemp;
double* V_array;
double* t_array;
double* dev_V_array;
double* dev_vel;
double* vel;
double* s2_times;
double* s2_times_dev;
double* percent_excited;
double* dev_percent_excited;
cudaEvent_t start,stop;
float elapsedTime;
curandState *rndState;
int size;
double begin_time;
double end_time;
double test_interval;
//Number of Parameters in the Model
int num_param = 29;
// Assume only running 1 simulation initially
int simulations = 1;

// Time Step Variables
double step = 0.005;
double tend = 20000;
int iterations = tend/step;
double skip_time_value = 0.5; //ms
int skip_timept = skip_time_value/step; // skipping time points in voltage array & time array
int total_timepts = iterations/skip_timept;

// Number of Cells
int length = 20;
int width = 20;
int num_cells = length*width;
int cells_per_thread =1; // for cell numbers > 500, one thread may need to work on more than one cell

//Stimulus Variables
double stimDur = 0.0;
double stimAmp = 0.0;
double stimInterval = 0.0;
int tstim = stimInterval/step;

// Sensitivity Analysis?
int num_changing_vars = 0;

// S2 Analysis?
bool s2_analysis = false;


      float* host_conductances;
```

```
float* conductances;

host_conductances = (float *)malloc(sizeof(float)*num_cells*10);

int j;
FILE *f3=fopen("conductances_vector01873_2.txt","r");
//FILE *f3=fopen("conductances_vector.txt","r");
if((f3=fopen("conductances_vector01873_2.txt","r"))==NULL)
//if((f3=fopen("conductances_vector.txt","r"))==NULL)
{ printf("\nfailed to open\n");
}

      for(j=0;j<num_cells*10;j++){
fscanf(f3,"%f \n ",&host_conductances[j]);
printf("%f \n", host_conductances[j]);
//fprintf(prova2, "%f \n",host_conductances[j]);
}

      cudaMalloc( &conductances, sizeof(float)*num_cells*10);
cudaMemcpy(conductances,host_conductances,num_cells*10*sizeof(float), cudaMemcpyHostToDevice);


      cudaEventCreate(&start);
cudaEventCreate(&stop);
cudaEventRecord(start,0);
...
// vars array contains voltage&state vartiables for all cells across all simulations
host_vars = (double *)malloc(sizeof(double)*size);
cudaMalloc( &dev_vars, sizeof(double)*size);

// results of the computeState kernel cudaMalloc( &dev_ion_currents, sizeof(double)*num_cells*simulations);
cudaMalloc( &dev_x_temp, sizeof(double)*size);

// result of the computeVoltage kernel
host_Vtemp = (double*)malloc(sizeof(double)*num_cells*simulations);
cudaMalloc( &dev_Vtemp, sizeof(double)*num_cells*simulations);

V_array = (double*)malloc(sizeof(double)*(total_timepts*num_cells*simulations));
t_array = (double*)malloc(sizeof(double)*(total_timepts*simulations));

//fprintf(fV, "V = [ \n"); to be able to open Maltsev GPU Voltage in Matlab through dlmread


      // Initialize vars array with initial conditions
initialConditions<<<simulations,(num_cells/cells_per_thread)>>>(dev_vars,num_param,num_cells,
cells_per_thread);

      cudaMemcpy(host_vars, dev_vars, size*sizeof(double), cudaMemcpyDeviceToHost);
for(i=0;i<num_cells*num_param*simulations;i++){
printf("%f\n",host_vars[i]);
}

while (time<iterations) {
linebreak computeState<<<simulations,(num_cells/cells_per_thread)>>>(dev_vars, conductances, dev_ion_currents, num_cells,
step, dev_randNums, simulations, dev_x_temp, num_changing_vars, cells_per_thread); updateState <<<simulations,
(num_cells/cells_per_thread)>>>(dev_vars, dev_x_temp, num_cells, cells_per_thread);

compute_voltage<<<simulations,(num_cells/cells_per_thread)>>>(dev_vars, dev_Vtemp,
dev_ion_currents, step, dev_randNums, simulations, length, width, num_changing_vars, time, stimDur, stimAmp,
tstim, s2_times_dev, cells_per_thread, s2_analysis, s2_loc);
update_voltage<<<simulations,(num_cells/cells_per_thread)>>>(dev_vars, dev_Vtemp,
num_cells, cells_per_thread);

//update Voltage and time arrays and write data to file
cudaMemcpy(host_Vtemp, dev_Vtemp, num_cells*simulations*sizeof(double), cudaMemcpyDeviceToHost);
if (time%skip_timept == 0) {
for (i=0;i<num_cells*simulations;i++) {
V_array[ (i*(iterations/skip_timept)) +index] = host_Vtemp[i];
fprintf(fV, "%f ", host_Vtemp[i]);
}
fprintf(fV, "\n");
fprintf(ft, "%f \n", time*step);
for (i=0;i<simulations;i++) {
t_array[(index*simulations)+i] = time*step;
}
index++;
```

```
}
time++;
}

      //fprintf(fV, "]; \n"); to be able to open Maltsev GPU Voltage in Matlab through dlmread


      /* The Model Computations are Finished – This last section of code is only writing data to file(s) and
cleaning up the memory*/

cudaEventRecord(stop,0);
cudaEventSynchronize(stop);
cudaEventElapsedTime(&elapsedTime,start,stop);

free(host_vars);
cudaFree(dev_vars);
cudaFree(dev_ion_currents);
cudaFree(dev_x_temp);
free(host_Vtemp);
cudaFree(dev_Vtemp);

printf("Elapsed Time = %f s \n",elapsedTime/1000);
printf("\n");
printf("Calculating Simulation outputs...\n");
printf("\n");


      ...

      free(V_array);
cudaFree(dev_V_array);


      printf("Program is Done\n");

}
```

Appendix B

Examples of some particular cases obtained performing Maltsev Tissue Model simulations in condition of σ=0.4, which lead to conductances values really far from the nominal ones. Fig. 3.25 shows a colour map of CL values computed for 400 cells at the end of a 20 s simulation with ρ=10^4 MΩ m. Some positions in the matrix corresponding to cells with unusual CL values are indicated by an arrow.

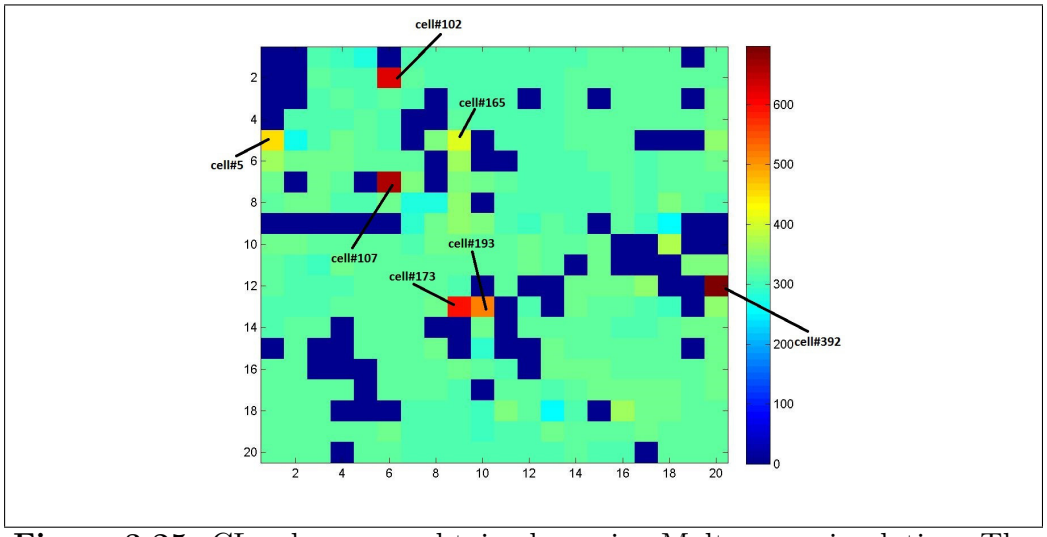

Figure 3.25: CL colour map obtained running Maltsev.cu simulation. The CL values of 400 cells at the end of 20 s simulation with ρ=10^4 MΩ m are plotted, as usual warm colours represent greater values of CL while cold ones represent lower values of CL, i.e. fastest cells.

The APs of some of the cells highlighted in the previous illustration are plotted in the following figures.

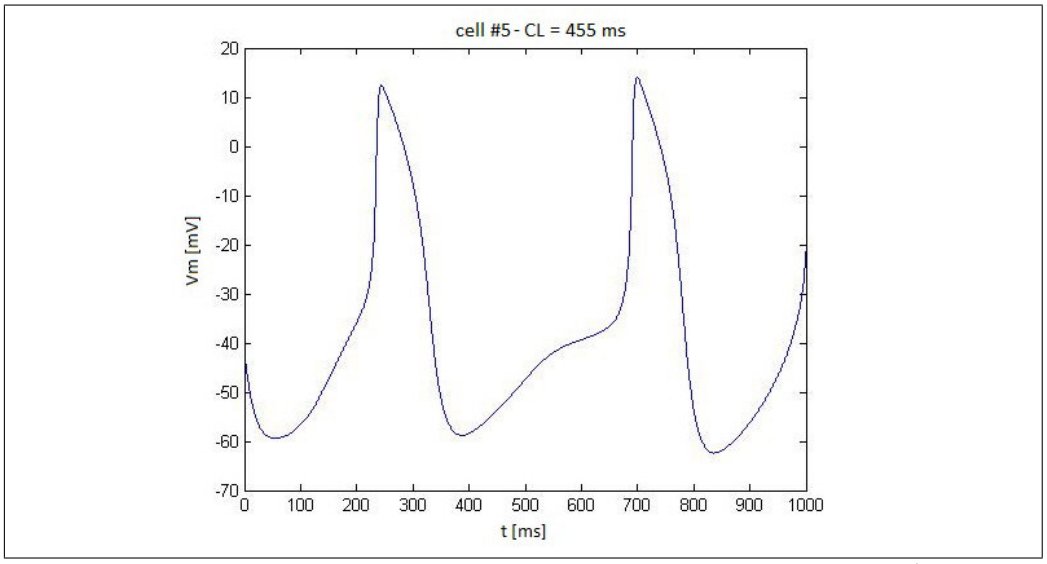

Figure 3.26: AP of cell n.5, tissue model simulation with ρ=10^4 MΩ m

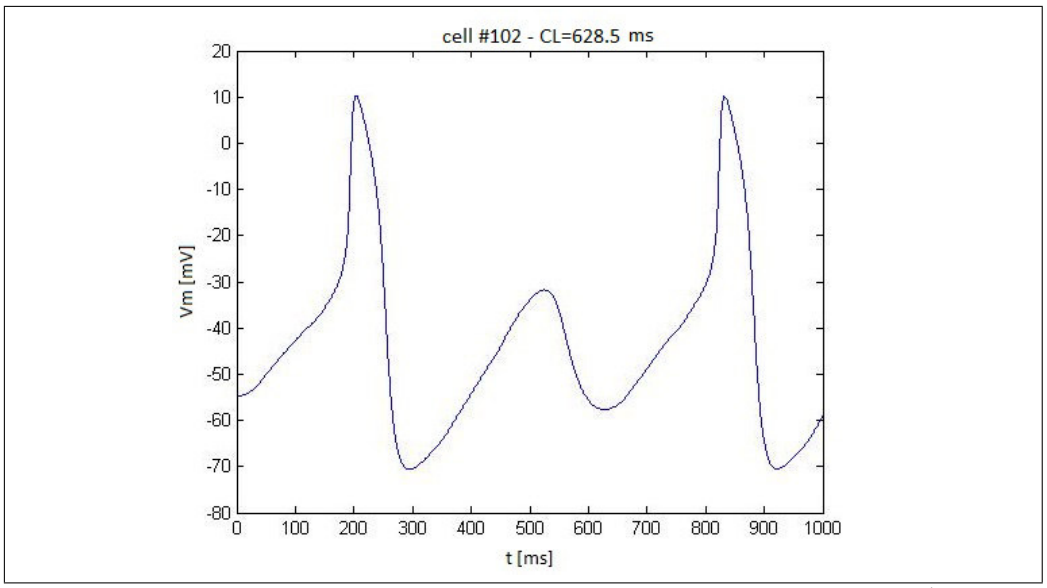

Figure 3.27: AP of cell n.102, tissue model simulation with ρ=10^4 MΩ m

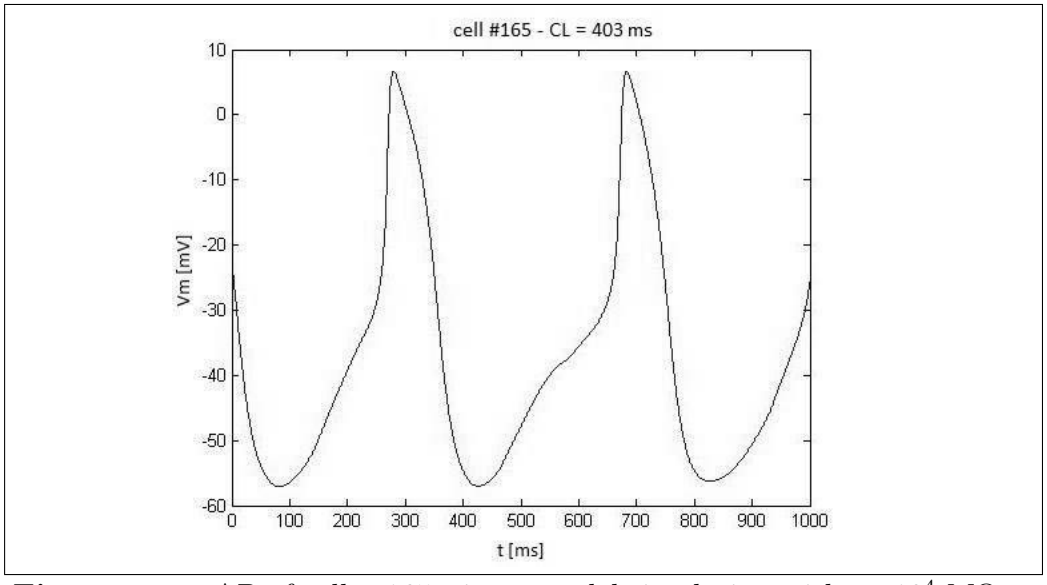

Figure 3.28: AP of cell n.165, tissue model simulation with ρ=10^4 MΩ m

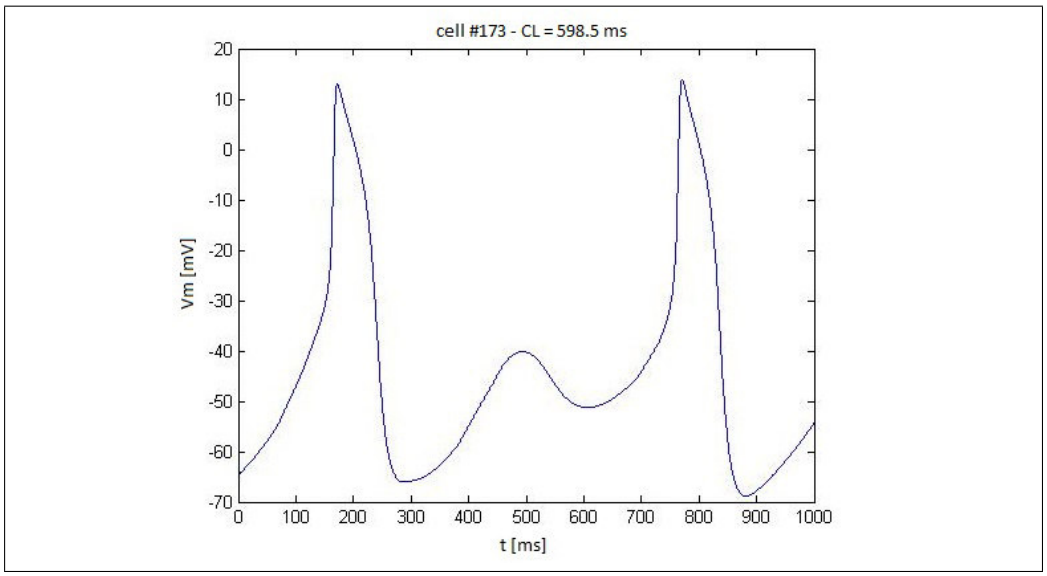

Figure 3.29: AP of cell n.173, tissue model simulation with ρ=10^4 MΩ m

In the summary figures reported in the third chapter mean values of CL and APA of 400 cells are plotted alternatively versus ρ and σ. To consider the mean value or the median one does not make great difference when we

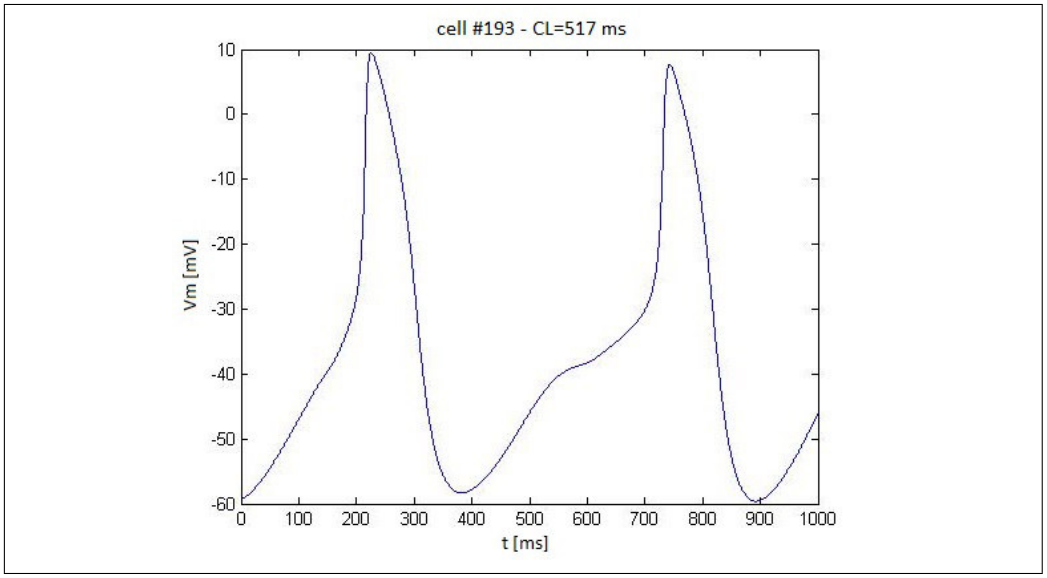

Figure 3.30: AP of cell n.193, tissue model simulation with ρ=10^4 MΩ m

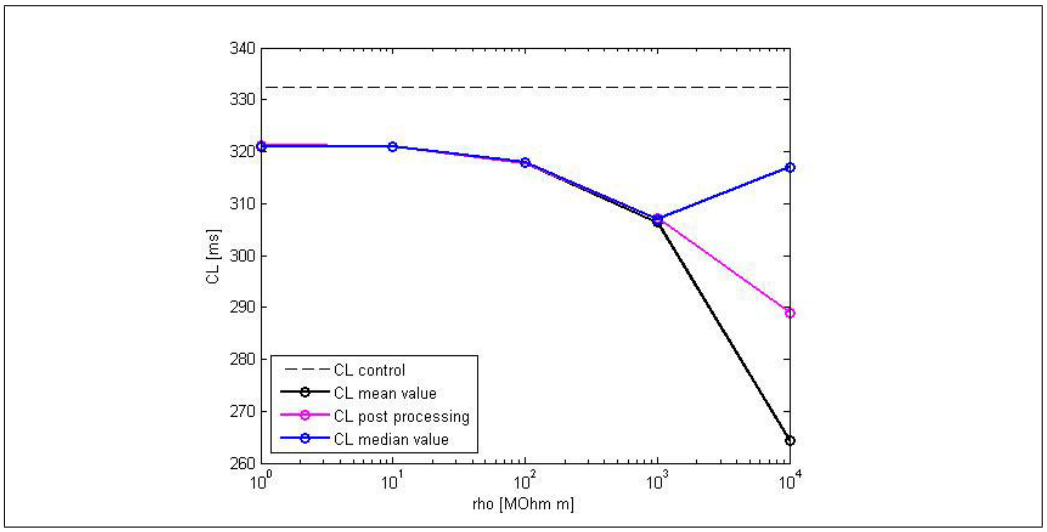

Figure 3.31: Mean, median and post-processing values of CL versus ρ. The condition σ=0.4 is considered. In the condition ρ=10^4 MΩ m, before modifying the CL array, it contained 75 CL values equal to 0 ms and 7 CL values greater than 400 ms. The CL mean and median value (without post-precessing) with ρ=10^4 MΩ m are respectively 264.32 ms and 317 ms.

deal with reasonable values of σ, instead when we stress the level on inter-cellular variability, i.e. using σ=0.3 and σ=0.4 there is some difference between the two values. In particular we find the greatest discrepancy when we consider the extreme values both for ρ and σ. This can be seen in Fig. 3.31 where three different curves are compared: CL mean values, CL median values, and CL mean values computed after having substituted the extreme CL values, i.e. CL=0 ms and CL>400 ms with the mean value calculated for the remaining cells.

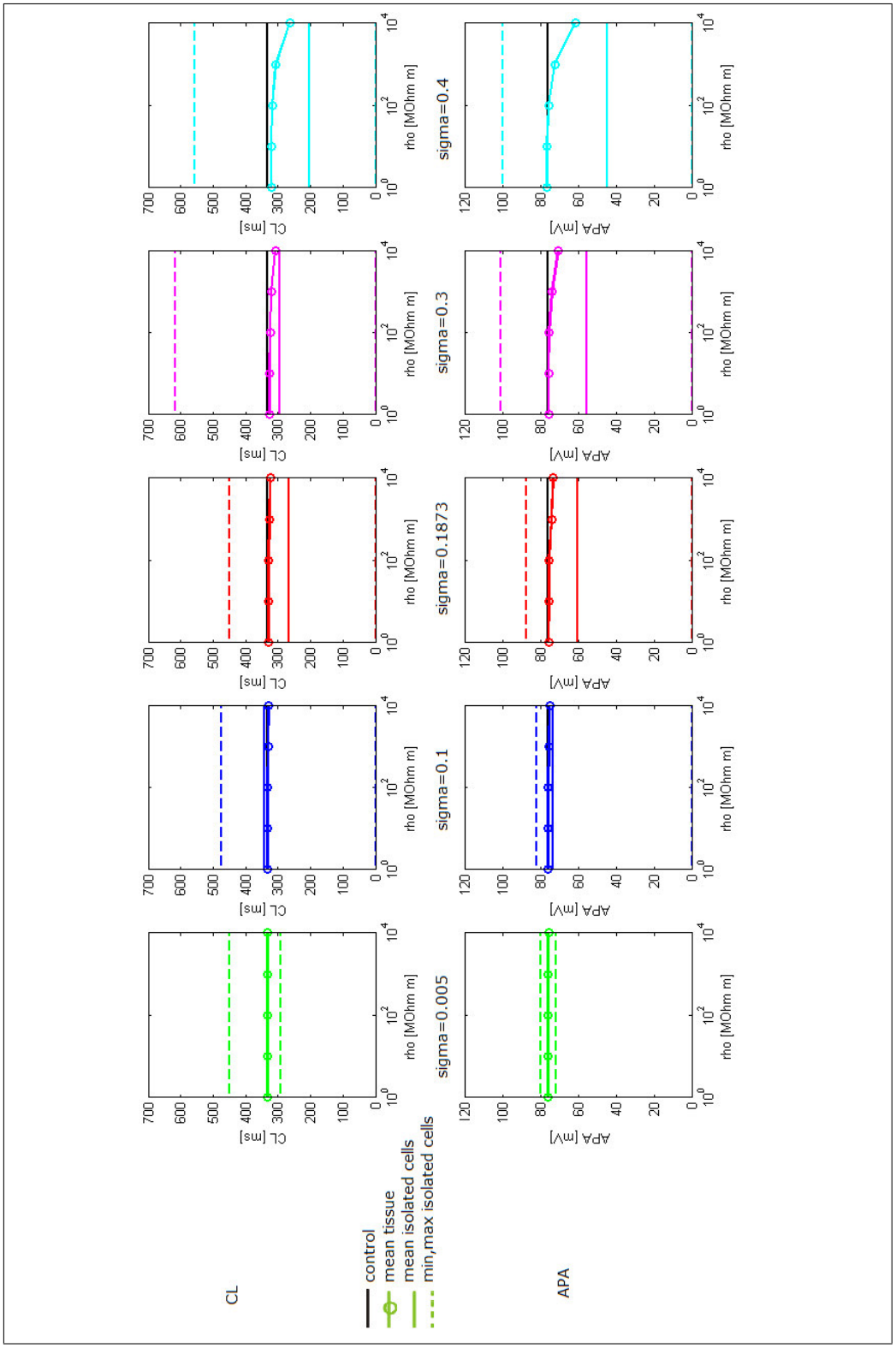

Figure 3.32: Comparison between mean values of CL and APA in conditions of isolated cells or connected cells.

Acknowledgments

I would like to express my sincere gratitude to Prof. Stefano Severi and Prof. Eric Sobie for having offered me this exciting opportunity, for having encouraged me during my work and most of all for having transmitted to me enthusiasm for research and for what I have been doing for my project. Thanks to Alan for his advices and for having lightened the worries of the last days.

Thanks to my family for being always with me even when they have to wait until 3 pm to wish me a good day.

Finally thanks to everyone I met in these six months, in particular Marta and Yasi, my favorite New Yorkers!

Bibliography

[1] The cellml project. http://www.cellml.org/.

[2] Cuda programming guide. https://developer.nvidia.com/cuda-zone.

[3] Right atrium. http://successimg.com/right-atrial-anatomy/.

[4] *Ionic Channels of Excitable Menbranes*. Sinauer Associetes, 1992.

[5] *CUDA by Example -An Introduction to General-Purpose GPU Programming*. Addison - Wesley, 2010.

[6] N. H. Lovell A. A. Abed, T. Guo and S. Dokos. Optimisation of ionic models to fit tissue action potentials: Application to 3d atrial modelling. *Computational and Mathematical Methods in Medicine*, 2013.

[7] T. Guo N.H. Lovell S. Dokos A.D. Bradd, A.A. Abed. Study of cardiac pacemaker excitation using generic ionic models and realistic cell distribution. In *34th Annual International Conference of the IEEE EMBS San Diego, California USA*.

[8] NVIDIA Corporation. Cuda c/c++ basics.

[9] E.P Matyas D.C Michaels and J. Jalife. Dynamic interactions and mutual synchronization of sinoatrial node pacemaker cells. a mathematical model. *Circulation Research*, 1986.

[10] E.P Matyas D.C Michaels and J.Jalife. Mechanisms of sinoatrial pacemaker synchronization: a new hypothesis. *Circulation Research*, 1987.

[11] A.C. van Ginneken J. Bourier M.W. Markman et al. E.E. Verheijck, A. Wessels. Distribution of atrial and nodal cells within the rabbit sinoatrial node: models of sinoatrial transition. *Circulation*, 1998.

[12] A.C. van Ginneken J. Bourier W.M. Markman L.M. Vermeulen J.M. de Bakker W.H. Lamers T. Opthof L.N. Bouman E.E. Verheijck, A. Wessels. Distribution of atrial and nodal cells within the rabbit sinoatrial node. *Circulation*, 1998.

[13] M. Fantini. Un nuovo modello di potenziale d'azione del nodo senoatriale. Bachelor's thesis, Biomedical Engineering, 2009.

[14] A. Giovannini. Parametric sensitivity analysis of the most recent computational models of rabbit cardiac pacemaking. Bachelor's thesis, Biomedical Engineering, 2012.

[15] J.R. Giudicessi and M.J. Ackerman. Potassium channel mutations and cardiac arrhythmias diagnosis and therapy. *Nature Reviews Cardiology*, 2012.

[16] Y. Okada H. Masumiya, Y. Oku. Inhomogeneous distribution of action potential characteristics in the rabbit sino-atrial node revealed by voltage imaging. *Journal of Physiological Sciences*, 59:227–241, March 2009.

[17] I. Kodama H. Honjo M. Lei T. Varghese H. Zhang, A. V. Holden and M. R. Boyett. Mathematical models of action potentials in the periphery and center of the rabbit sinoatrial node. *Am J Physiol Heart Circ Physiol*, 279:397–421, 2000.

[18] R.S. Kass J.M. Nerbonne. Molecular physiology of cardiac repolarization. *Physiological Reviews*, 2005.

[19] H. Irisawa K. Yanagihara, A. Noma. Reconstruction of sino-atrial node pacemaker potential based on the voltage clamp experiments. *Jpn J Physiol*, 1980.

[20] J. Keener and J. Sneyd. *Mathematical Physiology*. Springer, ii edition, 2009.

[21] R.E. Klabunde. Cardiovascular physiology concepts,2011.

[22] E.G. Lakatta and D. DiFrancesco. What keeps us ticking: a funny current, a calcium clock, or both? *Journal of Molecular and Cellular Cardiology.*, 2009.

[23] D. Difrancesco M. Baruscotti, A. Bucchi. Physiology and pharmacology of the cardiac pacemaker ("funny") current. *Pharmacology and Therapeutics*, 2005.

[24] Robert M.Berne, Matthew N.Levy, Bruce M.Kocppen, and Bruce A.Stanton. *Fisiologia*. Casa Editrice Ambrosiana, iv edition, 2010.

[25] W. Ying M.L. Hubbard and C.S. Henriquez. Effect of gap junction distribution on impulse propagation in a monolayer of myocytes: a model study. *Europace*, 2007.

[26] I. Kodamab M.R. Boyetta, H. Honjob. The sinoatrial node, a heterogeneous pacemaker structure. *Cardiovascular Research*, 2000.

[27] M.D. Munther Homoud. Introduction to cardiovascular pathophysiology. lectures,.

[28] R. Oren and C. Clancy. Determinants of heterogeneity, excitation and conduction in the sinoatrial node: A model study. *PLoS Computational Biology*, 6, December 2010.

[29] M.M. Maleckar P. Li, T.G. Lines and A. Tveito. Mathematical models of cardiac pacemaking function. *Frontiers in Physics*, 2013.

[30] N. H. Lovell S. L. Cloherty, S. Dokos. A comparison of 1-d models of cardiac pacemaker heterogeneity. *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, 53(2), February 2006.

[31] L.A. Charawi S. Severi, M. Fantini and D. DiFrancesco. An updated computational model of rabbit sinoatrial action potential to investigate the mechanisms of heart rate modulation. *The Journal of Physiology*, 2012.

[32] R.M. Wolf S.D. Unudurthi1 and T.J. Hund. Role of sinoatrial node architecture in maintaining a balanced source-sink relationship and synchronous cardiac pacemaking. *Frontiers in Physiology*, November 2014.

[33] E. A. Sobie. Parameter sensitivity analysis in electrophysiological models using multivariable regression. *Biophysical Journal*, 96:1264–1274, February 2009.

[34] E.A. Sobie. Anisotropic electrical propagation in heart.

[35] E.G. Lakatta V.A. Maltsev. Synergism of coupled subsarcolemmal ca2+ clocks and sarcolemmal voltage clocks confers robust and flexible pacemaker function in a novel pacemaker cell model. *Am J Physiol Heart Circ Physiol.*, 2009.