

TECHNISCHE UNIVERSITEIT EINDHOVEN
Department of Mathematics and Computer Science

**Simulation of Atrial Fibrillation
using the Cellular Bidomain Model**

By
R. J. Rijken

Supervisors:

Prof. Dr. P.A.J. Hilbers
Ir. N.H.L. Kuijpers

Eindhoven, August 2007

TABLE OF CONTENTS

Appendices	Page
1. INTRODUCTION	1
1.1 Motivation and problem statement	1
1.2 Aim of the thesis	1
1.3 Lay-out of the thesis	1
2. MODELLING INTRODUCTION	3
2.1 Anatomy and function of the heart	3
2.2 Cardiac cell models	3
3. CELL MODELLING	6
3.1 Polynomial model	6
3.2 Electrophysiological Modelling	7
4. TISSUE MODELLING	13
4.1 Bidomain model	13
4.2 Cellular Bidomain Model	15
4.3 Numerical aspects of the Cellular Bidomain Model	19
4.4 Implementing the Cellular Bidomain Model for a triangular mesh	19
5. PROCESSES UNDERLYING CARDIAC ARRHYTHMIA	23
5.1 Introduction	23
5.2 Reentry characteristics	23
5.3 Model	23
5.4 Numerical solution	24
5.5 Simulation results	25
6. ESTABLISHING PARAMETERS	29
6.1 Default parameters	29
6.2 Simulation results	30

Appendices	Page
6.3 Conclusion	35
7. SIMULATING ATRIAL TISSUE	37
7.1 Simulation results	37
7.2 Conclusion	39
8. GENERAL-PURPOSE COMPUTATION ON GRAPHICS HARDWARE	48
8.1 Introduction	48
8.2 Graphics processing unit	48
8.3 Reaction-diffusion model	50
8.4 Results	50
8.5 Limitations	51
9. CONCLUSION	53
9.1 Simulations	53
9.2 Real-time simulations	54
BIBLIOGRAPHY	56
APPENDIX	
A: REACTION-DIFFUSION GPGPU	62

Abstract

Atrial fibrillation (AF) is the most common cardiac arrhythmia. The risk of developing AF rises with the increase of age. Almost 10% of the people aged 80 and older suffer from AF. Structural changes of cardiac tissue, such as electrophysiological and structural remodelling, caused by diseases such as heart attacks and fibrosis, play a major role in initiating and sustaining of AF. However, next to these disease related cases of AF, there are cases where AF is reported to occur in healthy tissue. As the electrical activity of the heart is the result of ionic processes, it is necessary to study these processes in depth to get insight into the properties of cardiac fibrillation. Mathematical modelling may provide data and realistic simulations, which may provide insight into otherwise hard to study mechanisms.

In this study we investigate characteristics of AF using the Cellular Bidomain Model. It is believed that ionic processes are key factors in the initiation and sustaining of AF. Therefore, these processes are studied in detail. In particular, we study the effect of stretch-activated channels and electrophysiological remodelling on AF. For our simulations we use the Human Atrial Action Potential model, which is extended with a stretch-activated current and a so-called remodelling factor. Furthermore, we use the Cellular Bidomain Model implemented on a mesh model of the atria, created from MRI data. With our simulations we study the main characteristics of AF and investigate the effects of stretch and remodelling of ionic currents. The results indicate that the remodelling of ionic currents is necessary to initiate and sustain AF. Furthermore, we show that stretch may lead to atrial arrhythmia, which could progress to AF.

As the used models are fairly complex, time consumed by an average simulation is extensive. To address this issue, we study the implementation of reaction-diffusion equations, which are the basis of the cardiac models, on graphical hardware. Our results show a large decrease in computational time when simulations are performed on the graphical hardware.

We conclude that the Cellular Bidomain Model is well suited to study AF and investigate the role of the underlying ionic processes. Our simulations indicate that there is a direct relation between the duration of the action potential and AF and that stretch can cause the initiation of tachycardia such as AF. Moreover, we demonstrate the potential of the graphical hardware to speedup cardiac simulations. Even though the limitations of today's graphical hardware prevent widespread application for scientific use, its potential and growth are factors that have to be taken into account.

Samenvatting

Boezemfibrilleren (BF) is de meest voorkomende hartritmestoornis. Met het stijgen van de leeftijd neemt de kans op het verkrijgen van BF toe. Bijna 10% van de mensen die ouder zijn dan 80 jaar lijdt aan BF. Structurele veranderingen van de eigenschappen van het weefsel, veroorzaakt door ziekten zoals hartaanvallen, fibrosis en veroudering, spelen een grote rol bij het ontstaan en in stand houden van BF. Maar naast deze door ziekten veroorzaakte gevallen van BF, zijn er gevallen van BF bekend in het gezonde hart. Aangezien ionenuitwisseling ten grondslag ligt aan de elektrische activiteit van het hart, is het noodzakelijk deze processen in detail te bestuderen om inzicht te krijgen in de eigenschappen van BF. Het gebruik van wiskundige modellen kan helpen bij het verkrijgen van inzicht in anders moeilijk te bestuderen gebieden.

In deze studie onderzoeken we de eigenschappen van BF gebruikmakend van het Cellular Bidomain Model. Ionenuitwisselingen spelen een belangrijke rol bij het ontstaan en in stand houden van BF. Deze processen worden daarom in detail bestudeerd. In het bijzonder analyseren we het effect van stretch-activated channels en electrophysiologisch remodeleren op BF. Voor onze simulaties maken we gebruik van het Human Atrial Action Potential model dat uitgebreid is met stretch-activated channels en een zogenoemde remodelleer factor. Verder gebruiken we het Cellular Bidomain Model toegepast op een driedimensionaal model van het atrium, geconstrueerd van MRI data. Met onze simulaties bestuderen we de algemene kenmerken van BF en onderzoeken de effecten van mechanische belasting en het remodeleren van het weefsel. De resultaten wijzen uit dat remodeleren een belangrijke factor speelt bij het ontstaan en ondersteunen van BF. Verder laten we zien dat mechanische belasting van het hart kan leiden tot hartritmestoornissen die kunnen ontwikkelen tot BF.

De complexiteit van de gebruikte modellen resulteert in lange simulatietijden, wat de hoeveelheid uit te voeren simulaties aanzienlijk beperkt. Het gebruik van een grafische kaart voor de uitvoering van simulaties zou deze tijd kunnen verminderen. Hiertoe onderzoeken we de simulatie van reactie-diffusie processen op de grafische kaart. De resultaten tonen aan dat de grafische kaart simulatie tijden behoorlijk kan verkorten.

Concluderend kunnen we zeggen dat het Cellular Bidomain Model zeer geschikt is om BF te onderzoeken en de rol van onderliggende processen in kaart te brengen. Onze simulaties geven aan dat er een directe relatie is tussen de duur van de actiepotentiaal en BF en dat mechanische belasting van het hart BF kan veroorzaken. Verder laten we de mogelijkheden van grafische hardware zien om simulaties van het hart te optimaliseren. Ook al bemoeilijken beperkingen van de huidige grafische kaarten een breed toepassingsgebied, de mogelijkheden en toekomstige ontwikkelingen zijn factoren waarmee rekening dient te worden gehouden.

CHAPTER 1 INTRODUCTION

1.1 Motivation and problem statement

1.1.1 Atrial Fibrillation

Atrial fibrillation (AF) is an abnormal heart rhythm, or cardiac arrhythmia, of the two small, upper heart chambers. A normal heart beat starts at the sinoatrial (SA) node from which an electrical pulse spreads over the heart and causes the heart-tissue to contract and pump blood. During AF, disorganized, rapid pulses replace the normal electrical pulse and cause a highly irregular contraction of the atria.

Atrial fibrillation is the most common cardiac arrhythmia. The risk of developing AF rises with the increase of age. Almost 10% of the people aged 80 and older suffer from AF [26]. Even though AF is sometimes asymptomatic, it may lead to symptoms of palpitations (awareness of the beating of the heart), fainting, chest pain or even heart failure. Furthermore, the erratic beating of the heart leads to stasis (blood stagnation) in the heart which increases the risk of the forming of blood clots.

The electrical activity of the heart is the result of the interaction of billions of cells forming a heterogeneous system. On an even smaller scale, the electrical activity of a cardiac cell is the result of the interaction of many ionic processes. The result of this complexity is the difficulty to fully understand the functional behavior of the organ. The fact that AF is still not fully understood is a direct consequence of this complexity. To gain insight into the mechanisms and characteristics of AF, integration is needed of the empirical results from clinical and experimental research, and theoretical knowledge provided by scientific studies. A helpful tool to support this integration is mathematical modelling. Mathematical modelling may provide data and insight into certain mechanisms which would be otherwise hard or impossible to study. Furthermore, modelling can provide realistic simulations which could predict events in the future. Even though the modelling process is a valuable tool, it poses new challenges. In general, the biological processes that are simulated by a model are complex. Due to this complexity the model has to approximate this process for computational reasons. Hence, the results provided by the model should be interpreted and related to the underlying problem to be of real value.

1.2 Aim of the thesis

The aim of this thesis is to study the characteristics of AF using the Cellular Bidomain Model [25]. In particular, we investigate the role of ionic processes, such as stretch activated channels and electrophysiological remodelling, and their influence on AF. In particular we hypothesize that stretch, applied to the atrium, can cause AF.

1.3 Lay-out of the thesis

The thesis is organized as follows: In Chapter 2, the basic anatomy and physiology of the heart is introduced. Furthermore, a basic explanation of cellular electrophysiology is given to provide a background for the following chapters. In Chapter 3, the theory of cell modelling is explored. Starting with simple functional models of the cardiac electrical activity, we end with a complex physiological model. In Chapter 4 we introduce tissue modelling with the Bidomain Model and conclude with a discrete model, the Cellular Bidomain Model. This model will be used for our simulations and is studied in somewhat greater detail. Atrial Fibrillation and its characteristics are introduced in Chapter 5. Using a simplified model, we introduce the mechanisms of reentry and its

relation to AF. Chapter 6 presents the choice of parameters and specifies the used model for our experiments. Chapter 7 presents seven simulations of the atrium meshmodel, which progress from a single heart beat to AF. Chapter 8 explores the use of graphical hardware for scientific computing and investigates the possibility of realtime cardiac simulations. Finally, we discuss our used models, parameters and simulation results, which lead to the conclusion in Chapter 10.

CHAPTER 2 MODELLING INTRODUCTION

2.1 Anatomy and function of the heart

Before we can proceed with the explanation of cardiac electrophysiology, we need to get a thorough understanding of the area of interest on a larger scale. The heart is roughly situated in the center of the chest, between the right and left lung. The heart can be divided into four compartments. These four compartments consist of two large chambers, the right and left ventricle, responsible for pumping blood into the lungs and to the rest of the body. The smaller chambers, the right and left atrium, are responsible for pumping blood into the ventricles. Besides size, the chambers differ in thickness. The ventricles have a thick wall structure compared to the relative thin walls of the atria. Focussing on our object of interest, the atria, one notices some geometrical structures such as the superior vena cava, the tricuspid valve and the mitral valve. (See Figure 2.1). These structures form obstacles that may block propagation of electrical pulses and may play a major role in AF. [41]

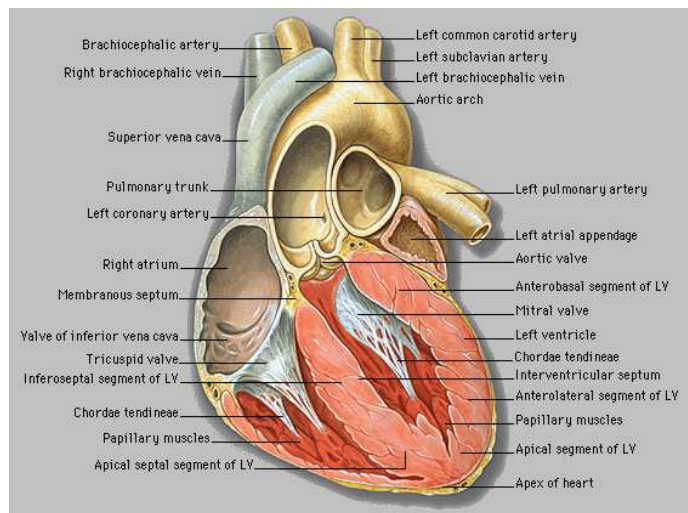


Figure 2.1. Anatomy of the heart, from [3]

2.2 Cardiac cell models

To provide a background for the following chapters and explain the electrical nature of cardiac cells, we give a basic description of cardiac electrophysiology. The cell is surrounded by a cell membrane which consists of phospholipids. This membrane acts as a barrier and isolates the interior of the cell from the outside. This barrier divides the space into extracellular space, denoted by a subscript e and an intracellular space, denoted by a subscript i . In the membrane, protein-lined pores are situated, called channels, which allow passage of specific molecules. In most cases these ion-channels actively select which ions can pass and which ions are blocked. This mechanism is called selective permeability. This mechanism causes an imbalance of ions such as Na^+ , Ca^{2+} and K^+ and creates a potential difference between extracellular and intracellular space.

The two forces acting on an ion in extracellular and intracellular space are an electrical and a chemical force. Due to the concentration gradient of ions there is a natural flow of ions down this gradient. As the chemical gradient differs for the different ions, a charge buildup on either side

of the cell membrane is created. This charge causes an electrical potential gradient opposing the chemical gradient. The potential at which equilibrium is reached, for ion x , is called the reversal potential and can be calculated with the Nernst equation:

$$E_x = \frac{RT}{z_x F} \log_e \frac{[x]_e}{[x]_i}, \quad (2.1)$$

Where R denotes the universal gas constant, z_x denotes the valence of the ion, T is the absolute temperature and F is Faraday's constant. $[x]_e$ denotes the concentration of the ion in extracellular space and $[x]_i$ denotes the concentration of the ion in intracellular space.

Knowing what causes the electrical potential across the membrane of a cell is only the begin of constructing a model of the electrical potential of a tissue. Many complex processes operate inside a cell, and modelling all of these processes would create a model which is computational intractable. To create a realistic model which is computational feasible to simulate the cardiac electrophysiology, one needs to model only the key aspects contributing to the electrical characteristics of a cell. Following Jacquemet [22] one can discern the following key aspects:

- excitability of the cell;
- action potential shape;
- propagation;
- restitution;
- geometry;

2.2.1 Excitability of the cell

Electrical fluctuations of neighboring cells and external electrical pulses cause disturbances in the resting state of the cardiac cell. If these disturbances are small the resting state is maintained. If these disturbances, however, are greater than a certain threshold an action potential (AP) is triggered. See figure 2.2

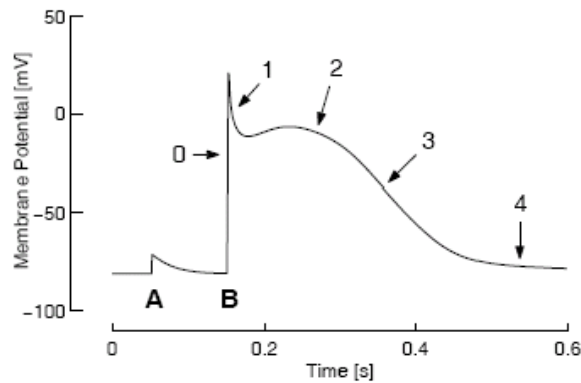


Figure 2.2. Action Potential shape

The action potential can be divided into five phases: 0. upstroke, 1 excited, 2. refractory, 3. repolarization phase and 4. resting phase.

- Phase 0, upstroke.** This phase is characterized by the fast depolarization of the cell membrane. This depolarization is caused by opening of the fast Na^+ channels. The opening of these channels causes an increase in conductance of the membrane for Na^+ ions. The concentration increase of the Na^+ ions results in a rise of positive charge. If the cell is stimulated when its membrane potential is above the resting potential of $\pm 82\text{mV}$, depolarization will be slower. At the higher potential not all Na^+ channels will be closed and will not be responsive to a new stimulation. This results in a smaller conductance and a slower depolarization of the cell membrane.
- Phase 1, excited.** When the membrane potential reaches its maximum, $\pm 20\text{mV}$ the Na^+ channels close and the K^+ channels begin to open, releasing K^+ ions to the cell's environment. This change of ion currents is responsible for the small repolarization of the cell and is called "the notch".
- Phase 2, refractory.** During the refractory period there is a balance between the inward current of Ca^{2+} ions and the outward current of K^+ ions. This balance maintains the membrane potential at a constant level and ensures the cell is not responsive to new stimulations just after depolarization.
- Phase 3, repolarization.** When the Ca^{2+} channels close and the K^+ channels remain open there will be a net outward current of positive ions, causing the cell membrane potential to decrease and start the repolarization phase of the cell.
- Phase 4, resting.** When the potential of the membrane is returned to its resting state, the cell is ready for a new stimulation to start the whole cycle from the beginning.[22]

2.2.2 Action potential shape

As will be explained in Chapter 5, the shape of the action potential plays an important role in the mechanics of AF. Hence, it is necessary to accurately model this aspect.

In 1952 Hodgkin and Huxley [19] studied dynamic ionic conductances that generate a nerve action potential. Using a voltage clamp experiment, they were able to separate the total ionic current into its ionic parts, and lay the foundation for modelling the electrical behavior of the cardiac cell. Later models, such as the LuoRudy(ventricle) model and the Courmanche(atrium) model, included more and more ionic currents into their system of equations. This resulted in an approximation of the AP shape, though at an increased computation cost.

2.2.3 Propagation

The intracellular space of cells is connected via channels, called gap junctions, which differ functionally from the ion-channels described previously. The gap junctions are permeable to large charged molecules and provide a low resistance pathway between adjacent cells. See Figure 2.3 for a schematic view. This non-linear diffusion process can be described by the following reaction-diffusion equation:

$$\frac{\partial u}{\partial t} = D \frac{\partial^2 u}{\partial x^2} + f(u), \quad (2.2)$$

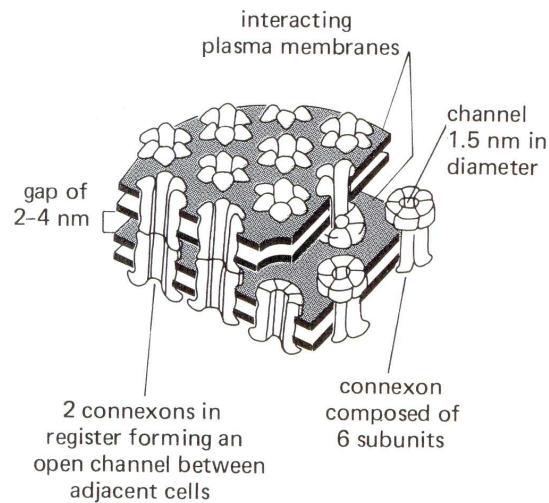


Figure 2.3. Gap junctions, from [23]

where D is the diffusion coefficient and $f(u)$ denotes the non-linear, reaction part.

2.2.4 Repolarization

Circular and spiral waves are closely related to chaotic phenomena, such as AF. The formation and propagation of these waves are significantly influenced by the duration of the repolarization of the cell. Hence, the modelling of this phase of the cardiac cell is a key aspect of the propagation of the AP. The process of repolarization of the cardiac tissue may appear similar as the depolarization wave but is fundamentally different. Where the depolarization is caused by propagation of the electrical potential from cell to cell, repolarization is merely caused by the finite duration of the AP. Closely related to this phase is the term restitution. Restitution refers to the fact that the velocity of the AP, as well as the duration of the AP, depend on the diastolic interval (DI), which is the period between the end of the repolarization and the new excitation [14].

2.2.5 Geometry

The previous described aspects were all characteristics of the cell and its ionic processes. An aspect of a different order is the structure formed by the individual cells, the tissue. Geometric aspects, such as non-conducting parts, in- and outgoing vanes and colliding layers of cardiac tissue, form obstacles for the propagation of the AP. These physical objects can play an important role for many phenomena regarding AF. Non-conducting parts cause perturbations in the wavefront causing the wave to break and increasing the chance for reentry significantly. The Superior Vena Cava forms a large non-conducting obstacle to which wavefronts tend to attach. (See Chapter 5). If this attachment is combined with a unidirectional conduction, a "rotor-wave" may form. [34]

CHAPTER 3 CELL MODELLING

Most of the cardiac cell models are based on the model of Hodgkin and Huxley [19], mentioned in the previous chapter. The level of complexity of the models has increased significantly since then. The electrical behavior of cardiac cells is modelled by modelling the ionic currents across the membrane of the cell. If one is only interested in the functional characteristics of the AP, however, a simpler model could suffice. Such models provide a simple, less computational demanding, representation of the AP, which can provide insights into processes on a larger scale, for which a model, based on sub-cellular processes is not necessary.

3.1 Polynomial model

Following [41] we introduce the polynomial model. This model, described by Hunter, McNaughton and Noble (1975) is a function of a single variable. It only models the depolarisation of the cell and not the repolarisation. Due to this simplification, it is easily solvable and hence, suitable for large-scale simulations. The obvious drawback is the inability to simulate processes for which the repolarisation is crucial, like reentry problems.

An example of a polynomial model is the cubic equation:

$$\frac{dV_m}{dt} = g \left[V_m \left(1 - \frac{V_m}{V_{th}} \right) \left(1 - \frac{V_m}{V_p} \right) \right], \quad (3.1)$$

where V_m is the membrane potential, V_{th} is the threshold potential, V_p is the plateau potential and g is the membrane current. The shape of the AP of this model is depicted in Figure 3.1.

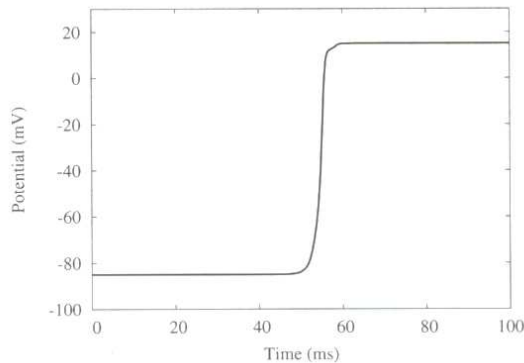


Figure 3.1. Cubic model. A stimulus of -50 mV is applied at 50 ms. From [41].

Another well-known functional model is the FitzHugh-Nagumo model (FitzHugh 1961, Nagumo et al. 1962). This model is also a cubic polynomial and uses an extra variable to model the repolarisation of the AP. The shape of the AP, generated by this model, is depicted in Figure 3.2.

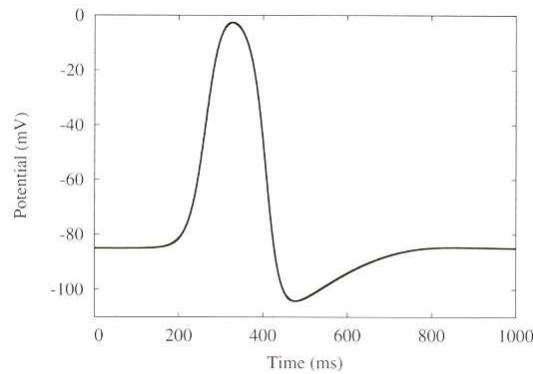


Figure 3.2. FitzHugh-Nagumo model. Note that the potential reaches -100mV during repolarization, which is lower than the resting potential. This is typical for a nerve cell, but not for a cardiac cell. From [41].

3.2 Electrophysiological Modelling

However useful these functional models are, they lack the possibility to simulate the effects of fine grained processes on the electrical behavior of the cell. To fully unlock the possibilities of simulation experiments, one needs to model processes on a cellular and sub-cellular level.

3.2.1 Models for ionic flux

The Nernst equation describes the equilibrium transmembrane current (See chapter 2). If the membrane potential is not in an equilibrium state, a different model is used. In this case the cell membrane is modelled as a resistive component and a capacitive component in parallel. See [23]. In Figure 3.3 this is depicted graphically.

The parallel component is a consequence of the isolating characteristic of the membrane (See Chapter 1). The resistive component is dependent on the voltage difference between intracellular and extracellular space, and will be different for each ion species. The voltage sources E_{Na} and E_k represent the equilibrium potential states. The following equation:

$$I_m = C_m \frac{\partial V_m}{\partial t} + I_{ion} \quad (3.2)$$

models the cell membrane, where I_m is the transmembrane current, C_m is the capacitance of the membrane, V_m is the transmembrane potential and I_{ion} is the total ionic current. In general we know that the ionic current has to obey Ohm's Law, $V = IR$, which can be adapted for the cell membrane into, $I = g_{ion}(v - v_{eq})$, where g_{ion} is the permeability of the membrane for a particular ion and v_{eq} is the reversal potential for that particular ion. Depending on the ionic current g_{ion} is modelled as a constant, depending on time, or depending on voltage and ionic concentrations.

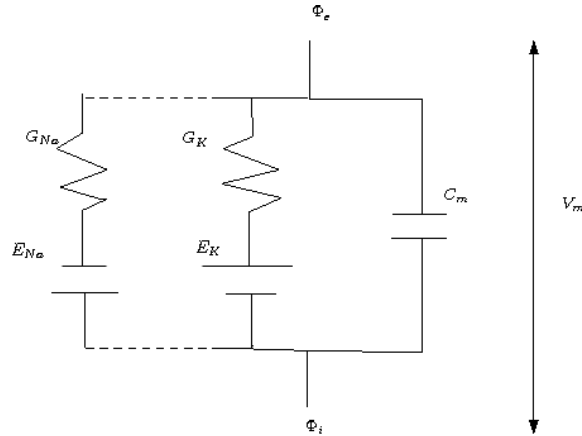


Figure 3.3. Modelling the cell membrane. C_m is the cell membrane capacitance. V_m is the membrane potential. Φ_e is the extracellular potential. Φ_i is the intracellular potential. E_{Na} and E_K are the equilibrium potentials and G_{Na} and G_K are the conductances.

3.2.2 Channel gating

Ion channels play an important role in maintaining an electric potential difference across the cell membrane. From experiments [19] it is known that the channels open and close in response to an electric stimulus. An open channel allows the flow of ions through the membrane and controls the conductance of these ions. Hodgkin and Huxley proposed the following model for this gating mechanism: For an arbitrary ion the conductance is described by [19]

$$g_{ion} = G_{ion}y, \quad (3.3)$$

where g_{ion} is the conductance of a particular ion, G_{ion} is the maximal conductance of that ion and y is the gating variable. The following differential equation describes the time dependency of the portion of channels in the open position

$$\frac{dy}{dt} = \alpha_v(1 - y) - \beta_v y, \quad (3.4)$$

where α_v and β_v are voltage dependent rate constants. The value of the gating variable is in the range $[0, 1]$ where 0 indicates that the gate is fully closed and 1 indicates that the gate is fully open.

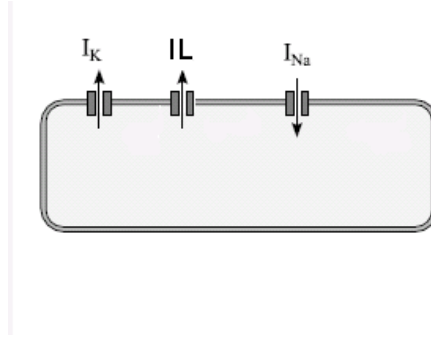


Figure 3.4. Hodgkin-Huxley model

3.2.3 Hodgkin and Huxley

As the ions sodium, potassium and calcium are very important for the shape of the action potential of a cell, one could construct a rudimentary model with only these ions. The Hodgkin-Huxley model [19], published in 1952, is an example of such a model. Instead of an ion current for calcium however, they used an unspecified leakage current.

The current for each ion is described by equations of the form 3.3 and substituted into 3.2 to create the following model for a single cell

$$I_{Na} = G_{Na} m^3 h (V - V_{Na}) \quad (3.5)$$

$$I_K = G_K n^4 (V - V_K) \quad (3.6)$$

$$I_L = G_L (V - V_L) \quad (3.7)$$

where $V = E - E_R$, $V_{Na} = E_{Na} - E_R$, $V_K = E_K - E_R$, $V_L = E_L - E_R$, E_{Na} , E_K , E_L are the equilibrium potentials and E_R is the absolute value of the resting potential. So V , V_{Na} , V_K and V_L can be seen as displacements from their respective reversal potentials.

The total ionic current is the sum of the three ionic currents

$$I_{ion} = I_{Na} + I_K + I_L, \quad (3.8)$$

and this may be substituted in Equation 3.2 to complete the model. The resulting shape of the action potential is depicted in Figure 3.5.

3.2.4 Luo-Rudy

More recent models, such as the Luo-Rudy phase two model 1994 [30], incorporate more ionic currents to model the electrophysiological functions with more detail. The increased detail provides the model with more flexibility to simulate experimentally observed mechanisms, such as changes in electrical behavior when the cell is stretched. The Luo-Rudy model describes in total twelve membrane currents, including pumps and exchangers. These currents are graphically depicted in Figure 3.6

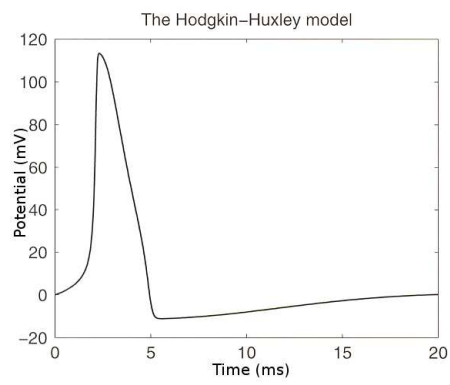


Figure 3.5. Hodgkin-Huxley model, from [41]

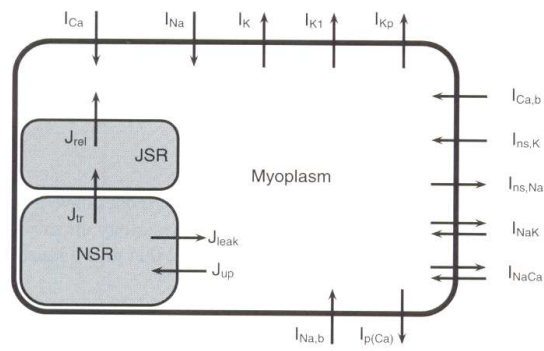


Figure 3.6. Luo-Rudy model, from [3]

3.2.5 Courtemanche

Based upon the Luo-Rudy model is the Courtemanche model. This model describes in total twelve membrane currents. A schematic figure of the currents is depicted below

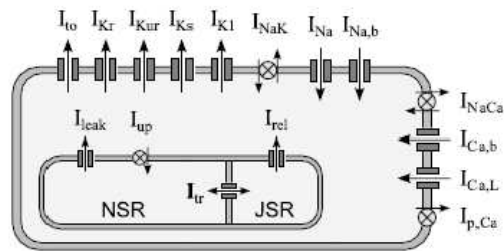


Figure 3.7. Courtemanche ion model, from [41]

CHAPTER 4 TISSUE MODELLING

After exploring the modelling of the cell, we introduce the modelling of tissue in this chapter. Starting with a volume averaging model, the Bidomain Model, we conclude this chapter with a discrete model, the Cellular Bidomain model, that we will be using for our simulations.

4.1 Bidomain model

This model divides the cardiac tissue into two domains, the intracellular space and the extracellular space. The extracellular space is assumed to be continuous. Due to gapjunctions, which connect the intracellular space of the individual cells together, the intracellular space is also considered to be continuous. The division into these two domains results in two potentials for each point lying in the domain, an intracellular potential and an extracellular potential. The cell membrane is considered to be the physical obstacle separating the two domains from each other. Furthermore, the transmembrane potential is defined as the difference between the intracellular potential and the extracellular potential.

The intra-cellular and extracellular current must adhere to Ohm's law

$$J_i = -M_i \nabla u_i, \quad (4.1)$$

$$J_e = -M_e \nabla u_e, \quad (4.2)$$

where M_i and M_e are the intra- and extracellular conductances, J is the current and u is the potential. The Bidomain model assumes that there is no build up of charge in any point of the domain.

$$\frac{\partial}{\partial t}(q_i + q_e) = 0, \quad (4.3)$$

where q_i and q_e are the charge in the intracellular and extracellular space, respectively. Furthermore, the total current into a point must equal the current leaving that point and the accumulation of charge in that point.

$$-\nabla \cdot J_i = \frac{\partial q_i}{\partial t} + \chi I_{ion}, \quad (4.4)$$

$$-\nabla \cdot J_e = \frac{\partial q_e}{\partial t} - \chi I_{ion}, \quad (4.5)$$

where χ is the membrane area to volume ratio. The current from intracellular space into the extracellular space is defined as positive, hence the negative sign in (4.5). Adding (4.5) and using (4.3) results in

$$\nabla \cdot J_i + \nabla \cdot J_e = 0. \quad (4.6)$$

Substituting (4.2) in (4.6) gives

$$\nabla \cdot (M_i \nabla u_i) + \nabla \cdot (M_e \nabla u_e) = 0. \quad (4.7)$$

Next, we proceed using the fact that the membrane is modelled as a capacitor in parallel with a resistor. The voltage over a capacitive membrane is written as

$$v = \frac{q}{\chi C_m}. \quad (4.8)$$

Where q is defined as

$$q = \frac{1}{2}(q_i - q_e). \quad (4.9)$$

Combining (4.8) and (4.9) and taking the time derivative results in

$$\chi C_m \frac{\partial v}{\partial t} = \frac{1}{2} \frac{\partial (q_i - q_e)}{\partial t}. \quad (4.10)$$

Using (4.3) we know that

$$\frac{\partial q_i}{\partial t} = -\frac{\partial q_e}{\partial t} = \chi C_m \frac{\partial v}{\partial t}. \quad (4.11)$$

Substituting this result in (4.5) gives us

$$-\nabla \cdot J_i = \chi C_m \frac{\partial v}{\partial t} + \chi I_{ion}. \quad (4.12)$$

Now we substitute (4.2) into the above equation to get rid of J_i

$$\nabla \cdot (M_i \nabla u_i) = \chi C_m \frac{\partial v}{\partial t} + \chi I_{ion}. \quad (4.13)$$

Finally, we use the definition of the transmembrane potential and rewrite this as follows $u_i = v + u_e$. Substituting this in (4.13) and (4.7) eliminates u_i and results in the definition of the Bidomain model

$$\nabla \cdot (M_i \nabla (u_e + v)) = \chi C_m \frac{\partial v}{\partial t} + \chi I_{ion}, \quad (4.14)$$

$$\nabla \cdot (M_i \nabla (u_e + v)) + \nabla \cdot (M_e \nabla (u_e)) = 0. \quad (4.15)$$

The first equation is a reaction-diffusion equation describing the transmembrane potential, where the ionic currents represents the non-linear reaction term. The second equation describes the extracellular potential field resulting from the transmembrane potential distribution. [41]

4.2 Cellular Bidomain Model

As the Bidomain equations of the previous section are quite complex and non-linear, one uses numerical approximation schemes to solve the system. Often used numerical integration schemes are the finite difference method and the finite element method. The fast, non-linear, upstroke of the depolarization current, forces the approximation schemes to use small time steps (0.01 s) and small spatial steps (0.1 mm) [8]. Due to these small steps the computation of the equations takes a large amount of time.

The Cellular Bidomain Model approximates the domain of interest in a different way. This model constructs a simulation graph $\mathcal{G}(\mathcal{N}, \mathcal{E})$, where \mathcal{N} is a set of nodes and \mathcal{E} is a set of undirected edges. A node represents a segment, which consists of one or more cells and an edge represents the gap junctions between two neighboring segments. Each node has a state, consisting of the intracellular potential, extracellular potential and the state of the membrane, which is modelled by gating variables and ionic concentrations. As each node has its own membrane state, different membrane behavior for different places of the tissue can be modelled. Furthermore, the simulation graph provides the model with the flexibility to model any geometry and structure.

The edges model the conductance between the segments, denoted by σ_{int} and σ_{ext} for the intracellular and extracellular currents. See Figure 4.1 for a graphical description of the simulation graph.

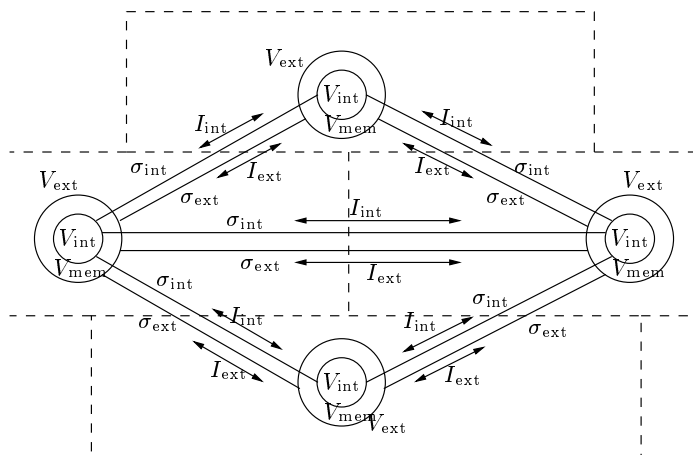


Figure 4.1. Cellular Bidomain Model. From [25]

For each edge $(n, m) \in \mathcal{E}$ it holds that $\sigma_{int}^{(n,m)} > 0$ and $\sigma_{ext}^{(n,m)} > 0$. For the intracellular and extracellular currents from node n to node m Ohm's law is applied

$$I_{int}^{n \rightarrow m} = (V_{int}^n - V_{int}^m) \sigma_{int}^{(n,m)}, \quad (4.16)$$

$$I_{ext}^{n \rightarrow m} = (V_{ext}^n - V_{ext}^m) \sigma_{ext}^{(n,m)}. \quad (4.17)$$

Current entering the node from surrounding nodes is described by

$$I_{int}^n = \sum_{(a,n) \in \mathcal{E}} I_{int}^{a \rightarrow n}, \quad (4.18)$$

$$I_{ext}^n = \sum_{(a,n) \in \mathcal{E}} I_{ext}^{a \rightarrow n}, \quad (4.19)$$

for the intracellular and extracellular current. Furthermore, for each node Kirchoff's current law must apply, hence

$$I_{trans}^n = I_{int}^n = -I_{ext}^n. \quad (4.20)$$

Following the Bidomain model, the transmembrane current equals

$$I_{trans}^n = C_m^n \frac{dV_m^n}{dt} + S_m^n I_{ion}, \quad (4.21)$$

where C_m^n denotes the membrane capacitance of node n in μF and S_m^n denotes the membrane surface in cm^2 . The ionic currents are modelled according to a modified version of the Human Atrial Action Potential Model (HAAP) of Courtemanche *et al.* [36]. I_{ion} is defined as

$$I_{ion} = I_{Na} + I_{K1} + I_{to} + I_{Kur} + I_{Kr} + I_{Ks} + I_{Ca,L} + I_{p,Ca} + I_{NaK} + I_{NaCa} + I_{b,Na} + I_{b,Ca}, \quad (4.22)$$

where I_{Na} is the fast inward Na^+ current, $I_{K1}, I_{to}, I_{Kur}, I_{Kr}, I_{Ks}$ are potassium currents and $I_{Ca,L}$ is the calcium current. I_{NaCa} is the $\text{Na}^+/\text{Ca}^{2+}$ exchanger current, I_{NaK} is the $\text{Na}^+ - \text{K}^+$ pump current, and $I_{b,Na}$ and $I_{b,Ca}$ are the background Na^+ and Ca^{2+} currents, respectively.

4.2.1 Stretch-activated current

To include the effect of stretch on the AP, Kuijpers *et al.* [27] proceed as follows. Studies have reported a relation between conduction slowing and shortening of the refractory period in acutely dilated atria [7] [21] [38]. Eijsbouts *et al.* [13] [12] found, in addition to conduction slowing, an increased occurrence of intra-atrial block. Hu and Sachs [20] and Kohl and Sachs [24] hypothesize that stretch-induced changes in electrophysiological behavior can be explained by stretch-activated channels (SACs). To incorporate SACs, the model of Courtemanche *et al.* is extended with a stretch-activated current: I_{sac} . The total ionic current is now given as

$$I_{ion} = I_{Na} + I_{K1} + I_{to} + I_{Kur} + I_{Kr} + I_{Ks} + I_{Ca,L} + I_{p,Ca} + I_{NaK} + I_{NaCa} + I_{b,Na} + I_{b,Ca} + I_{sac}, \quad (4.23)$$

where I_{sac} is the stretch-activated current. I_{sac} is defined in the following way

$$I_{sac} = I_{sac,Na} + I_{sac,K} + I_{sac,Ca}, \quad (4.24)$$

where $I_{\text{sac,Na}}$, $I_{\text{sac,K}}$ and $I_{\text{sac,Ca}}$ represent the Na^+ , K^+ and Ca^{2+} contributions, respectively, to I_{sac} . These currents are defined by the constant-field Goldman-Hodgkin-Katz current equation [23]

$$I_{\text{sac,Na}} = P_{\text{Na}} g_{\text{sac}} \frac{z_{\text{Na}}^2 F^2 V_{\text{mem}}}{RT} \frac{[\text{Na}^+]_i - [\text{Na}^+]_o \exp(-\frac{z_{\text{Na}} F V_{\text{mem}}}{RT})}{1 - \exp(-\frac{z_{\text{Na}} F V_{\text{mem}}}{RT})}, \quad (4.25)$$

$$(4.26)$$

$$I_{\text{sac,K}} = P_{\text{K}} g_{\text{sac}} \frac{z_{\text{K}}^2 F^2 V_{\text{mem}}}{RT} \frac{[\text{K}^+]_i - [\text{K}^+]_o \exp(-\frac{z_{\text{K}} F V_{\text{mem}}}{RT})}{1 - \exp(-\frac{z_{\text{K}} F V_{\text{mem}}}{RT})}, \quad (4.27)$$

$$(4.28)$$

$$I_{\text{sac,Ca}} = P_{\text{Ca}} g_{\text{sac}} \frac{z_{\text{Ca}}^2 F^2 V_{\text{mem}}}{RT} \frac{[\text{Ca}^{2+}]_i - [\text{Ca}^{2+}]_o \exp(-\frac{z_{\text{Ca}} F V_{\text{mem}}}{RT})}{1 - \exp(-\frac{z_{\text{Ca}} F V_{\text{mem}}}{RT})}, \quad (4.29)$$

where P_{Na} , P_{K} and P_{Ca} denote the relative permeabilities to Na^+ , K^+ and Ca^{2+} , z_{Na} , z_{K} and z_{Ca} represent the ion valences, F is Faraday's constant, R is the universal gas constant, and T is temperature (310 K) [9].

The conductance (g_{sac}) is defined as follows

$$g_{\text{sac}} = \frac{G_{\text{sac}}}{1 + K_{\text{sac}} \exp[-\alpha_{\text{sac}}(\lambda - 1)]}, \quad (4.30)$$

where $G_{\text{sac}} = 0.015 \mu\text{m/s}$ is the maximum membrane conductance, $K_{\text{sac}} = 100$ is a parameter to define the amount of current when the cell is not stretched, λ denotes the stretch ratio, and $\alpha_{\text{sac}} = 3$ is a parameter to describe the sensitivity to stretch. K_{sac} and α_{sac} are from Zabel et al. [46].

The reversal potential (E_{sac}) can be obtained by solving the following equation for V_{mem} : $I_{\text{sac,Na}} + I_{\text{sac,K}} + I_{\text{sac,Ca}} = 0$. For our simulations, we consider the following case: $P_{\text{Na}} : P_{\text{K}} : P_{\text{Ca}} = 1 : 1 : 1$, with $E_{\text{sac}} = -0.2$ mV. Figure 4.2 illustrates the current-voltage relation for the stretch-activated current (I_{sac}).

4.2.2 Remodelling

In clinical experiments it is determined that AF can be divided into different classes. One class of AF is called 'lone Atrial Fibrillation'. This is a form of AF that is not caused by heart diseases and other cardiac tissue abnormalities. In most cases this form of AF is not life threatening. Other classes of AF are 'persistent' and 'permanent AF'. In these cases the cardiac tissue is structurally altered by diseases, for example dilatation and fibrosis. This adaptation of the atria is called 'remodelling'. Remodelling develops due to changing load conditions, which may be caused by changes in pressure or volume load, or caused by diseases, such as a heart infarct. Remodelling primarily affects the excitability and electrical activity of the atrial myocytes. There are three types of remodelling: structural remodelling (changes in the size of the tissue and changes in properties of gap junctions), electrical remodelling (adaptation of ionic current properties) and mechanical remodelling (abnormal contraction or relaxation of the cardiac tissue). For our experiments we simulate electrical remodelling by changing $I_{\text{Ca,L}}$ and I_{to} . One way to incorporate remodelling of $I_{\text{Ca,L}}$ into the Cellular Bidomain is the following method, used by Model Kuijpers *et al.*[28]:

In the Courtemanche et al. model, $I_{\text{Ca,L}}$ is defined by

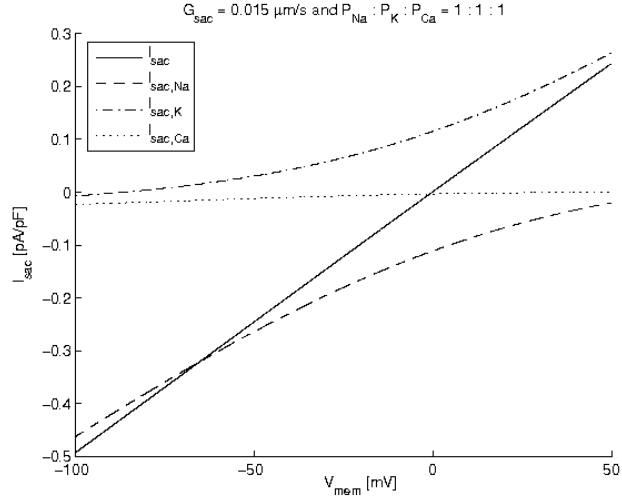


Figure 4.2. Stretch activated current.

$$I_{Ca,L} = g_{Ca,L} d f f_{Ca}(V_{mem} - 65), \quad (4.31)$$

where $g_{Ca,L}$ is the maximum $I_{Ca,L}$ conductance, d is the activation variable, f is the voltage-dependent inactivation gating variable, and f_{Ca} the Ca^{2+} -dependent inactivation gating variable [9]. The dynamics of the activation variable d is defined by

$$\frac{dd}{dt} = \frac{d_{\infty} - d}{\tau_d}, \quad (4.32)$$

where steady-state value d_{∞} and time constant τ_d are defined

$$d_{\infty} = \frac{1}{1 + \exp\left(-\frac{V_{mem} + V_{shift,ref}}{8}\right)}, \quad (4.33)$$

$$\tau_d = \frac{1 - \exp\left(-\frac{V_{mem} + V_{shift,ref}}{6.24}\right)}{0.035(V_{mem} + V_{shift})(1 + \exp\left(-\frac{V_{mem} + V_{shift,ref}}{6.24}\right))}, \quad (4.34)$$

where $V_{shift,ref} = 10$ mV.

Plotnikov et al. [35] observed a more positive $I_{Ca,L}$ activation threshold and slower inactivation in cardiac myocytes compared to control myocytes. Based on these observations, $I_{Ca,L}$ activation in the Cellular Bidomain Model is shifted with respect to V_{mem} by

$$V_{shift} = V_{shift,ref}(\rho + 1), \quad (4.35)$$

where ρ is the remodelling parameter, which ranges from -1.0 to 1.0.

Another method used to remodel $I_{Ca,L}$ and I_{to} is to vary the density of these currents, as is mentioned by Courtemanche et al. [9]. Figure 4.3 shows the effect of varying $I_{Ca,L}$ and I_{to} on the AP.

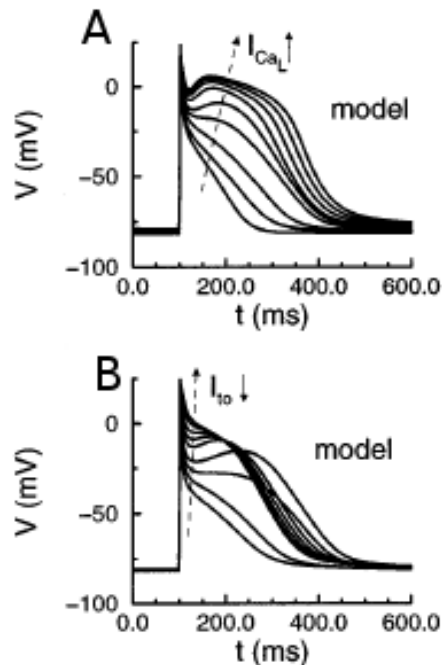


Figure 4.3. The effect of varying $I_{Ca,L}$ (A) and I_{to} (B), conductances (10, 25, 50, 75, 100, 150, 200, 300% of control) on action potential during stimulation at 1 Hz. From [9].

4.3 Numerical aspects of the Cellular Bidomain Model

For the computation of the Cellular Bidomain Model the equations are translated to a matrix notation to facilitate the computation. The intracellular, extracellular and membrane potentials are represented by vectors and the intracellular and extracellular conductivities are represented by connectivity matrices. Using an iterative scheme, the solution of the set of equations is approximated for each time step. See Kuijpers et al. [26] for more detail. During the simulation each node is assigned a high, medium or low accuracy level. Depending on this level, the time step to update the membrane state is varied.

4.4 Implementing the Cellular Bidomain Model for a triangular mesh

For our simulations, we implement the Cellular Bidomain Model on a triangular mesh, which represents a model of the atrium constructed from data of an MRI scan, provided by Peter van Dam

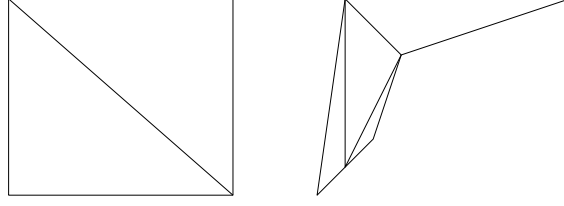


Figure 4.4. Triangular mesh

[43]. A triangular mesh is a geometric shape which is constructed from triangles. The triangles can have any size and shape, see Figure 4.4.

To implement the Cellular Bidomain Model on such a mesh, we have to translate the triangular structure to a simulation graph. This is done as follows. Consider a mesh, consisting of equilateral triangles representing an isotropic tissue with height h . Each node of the mesh is added to the set of nodes of the simulation graph and each edge of the mesh is added to the set of edges of the simulation graph. Each node $n \in \mathcal{N}$ represents a segment of cardiac tissue. Take two such nodes, x and y , which represent segment $s1$ and segment $s2$. Let L denote the distance of the segment-edge pq , which is shared by $s1$ and $s2$. Let d denote the distance between node x and node y . Assume pq intersects the edge between x and y in the center. See Figure 4.5 and Figure 4.6. As the triangles are equilateral, $\angle pxy = \frac{\pi}{6}$. The area A , that connects the two adjacent segments is defined by

$$A = L \times h. \quad (4.36)$$

The intracellular and extracellular conductivities between two adjacent segments are defined by

$$\sigma_{int} = g_{int} \frac{A}{D}, \quad (4.37)$$

$$\sigma_{ext} = g_{ext} \frac{A}{D}, \quad (4.38)$$

where g_{int} and g_{ext} are the intracellular and extracellular conductivities, expressed in mS/cm.

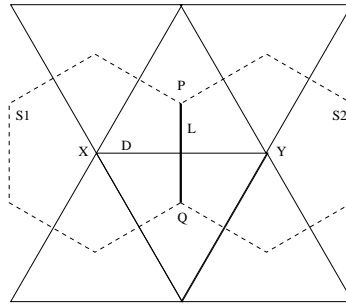


Figure 4.5. Two adjacent segments.

As the resulting graph should represent a cardiac tissue as accurate as possible, it is necessary to investigate the influence of the mesh shape and size on the propagation of the AP. In an ideal mesh, the time it takes for the AP to propagate from one node to an adjacent node, should be related

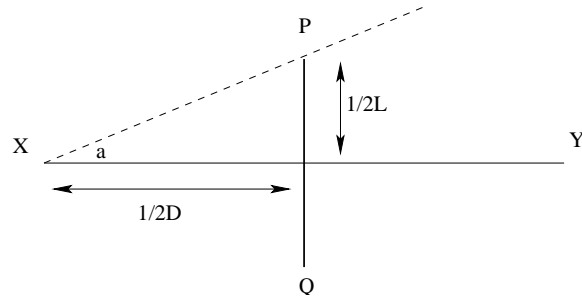


Figure 4.6. Calculating the area between two adjacent segments. As $\angle pxy = \frac{\pi}{6}$, $\tan(\frac{\pi}{6}) = \frac{\frac{1}{2}L}{\frac{1}{2}D} = \frac{L}{D}$, so $L = \tan(\frac{\pi}{6}) \times D$.

to the distance between these nodes and the conductivity of the edge connecting these nodes. A mesh structure which would influence the propagation due to the placement of edges would be undesirable. Figure 4.7 shows an example of such an undesirable mesh.

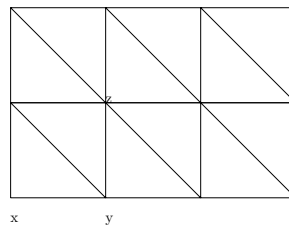


Figure 4.7. Undesirable mesh structure

Assume that the triangles of Figure 4.7 are not equilateral. Furthermore, note that even though segment x and z are adjacent in the cardiac tissue, there is no edge between x and z in the triangular mesh. This results in a greater time for the AP to propagate from node x to z , in the simulation graph, compared to the propagation time in the cardiac tissue. If the triangles were equilateral, however, the mesh would have the structure as the polygon in figure 4.8. In this case, the

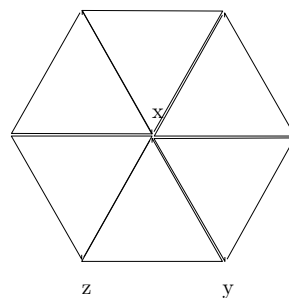


Figure 4.8. Desirable mesh structure

propagation time of the simulation graph equals the propagation time of the cardiac tissue. Hence, to minimize the error made due to the structure of the mesh, the triangles of the mesh should be equilateral, or equally, the mesh should contain equilength edges. In the mesh structure (gatria),

used for our experiments, the average edge length equals 0.27 cm and the standard deviation of the edges equals 0.07 cm, which we consider a good approximation of equilateral.

Another factor influencing our simulations is the distance between the nodes. As we are using the model of Courtemanche et al., that assumes a distance of 0.01 cm between adjacent nodes, the distance between two nodes in our simulation graph should approximate 0.01 cm. The average distance between two nodes in guatria equals 0.27 cm. To decrease the distance between two nodes, every triangle in guatria is divided in smaller triangles. The simulation program defines three levels and for each level the original triangle is divided into smaller triangles. In Figure 4.9 this is graphically depicted

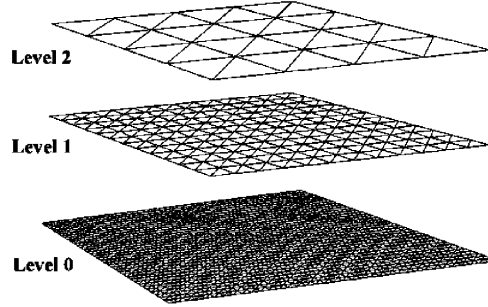


Figure 4.9. Mesh levels

The number of times a triangle is divided is controlled by the "division frequency" parameter. For instance, using a frequency of three combined with three levels, will decrease the distance between nodes nine times. After the refinement, executed by our simulation program, the average distance between two nodes equals 0.03 cm. Ideally we would like to refine even more, but this would increase the computation time and used memory even more.

Due to the large increase in the number of nodes, when one uses a high level and division frequency, the computation time increases dramatically. This increase in time may be undesirable and one may be forced to use a lower level and frequency. To compensate for the error made during these simulations, an extra parameter is introduced: the membrane current factor α . To calculate the ionic current I_{ion} , each node $n \in N$ uses a fraction of the average current of the adjacent nodes. The fraction is determined by the membrane current factor. In mathematical terms

$$I_{\text{ion}}^n = (1 - \alpha)I_{\text{ion}}^n + \alpha \frac{\sum_{(n,m) \in \mathcal{E}} I_{\text{ion}}^m}{\#\{(n,m) \in \mathcal{E}\}}. \quad (4.39)$$

CHAPTER 5 PROCESSES UNDERLYING CARDIAC ARRHYTHMIA

5.1 Introduction

In this chapter we study the general aspects of AF. As mentioned in Chapter 1, AF is described as highly irregular electrical activity of the cardiac tissue. The mechanisms that disturb the normal, regular electrical cardiac behavior are still unclear. Even though the recordings of an electrical cardiogram (ECG) reveal the chaotic electrical activity during AF, it does not give any insight into the propagation and detailed mechanisms of the electrical waves. An intuitive reason for AF would be irregularities of the cardiac tissue, which could cause the mentioned disturbances. These irregularities, however, could be either microscopic or macroscopic and have to be studied separately. In this chapter we focus on the general characteristics of excitable media and large, geometric heterogeneities in the cardiac tissue.

To isolate the area of interest even more, we abstract away from the physiological models introduced in the previous chapters, and focus on a simple reaction-diffusion model. This model enables us to compute the propagation of waves in excitable media in real time and provides insight into mathematical and geometric aspects of AF.

5.2 Reentry characteristics

Excitable media, such as cardiac tissue, have a few characteristics in common which are crucial to the chaotic behavior. First of all, elements of the media are excitable. This means that a cell, exposed to an electrical potential larger than a certain threshold, will depolarize. This depolarization is a fast, non-linear process. Once the cell reaches this state it enters a refractory phase, during which it will not respond to any stimulation. After the refractory phase, the cell resets itself during the repolarization phase. This phase is a relatively slow process compared to the depolarization phase. Once the cell has finished this cycle, it is responsive again for external stimulations. The shape of the electrical propagation wave is constant as energy is consumed during the different phases of the cell. These cell characteristics have the following results for the wave propagating over the cardiac tissue. First, two colliding wavefronts will cancel each other out. Second, a wavefront encountering refractory cells, will be blocked. Third, a wavefront encountering repolarized cells will continue, as these cells are stimulated again. This last effect, called reentry, is thought to be one of the major factors contributing to AF.

5.3 Model

To study the characteristics of AF, we use the simplified model from Dwight Barkley [4]. This model is the following two variable reaction-diffusion system,

$$\frac{\partial u}{\partial t} = f(u, v) + \nabla^2 u, \tag{5.1}$$

$$\frac{\partial v}{\partial t} = g(u, v). \tag{5.2}$$

The local kinetics of u and v , expressed by the functions $f(u, v)$ and $g(u, v)$ is defined as follows:

$$f(u, v) = \epsilon^{-1}u(1 - u)[u - u_{th}(v)], \quad (5.3)$$

$$g(u, v) = u - v, \quad (5.4)$$

where $u_{th}(v) = (v + b)/a$, and a, b and ϵ are parameters. In this system the variable u denotes the fast I_{Na} ion current causing the fast depolarization of the cell. The variable v denotes the slower I_K ion current, causing the repolarization of the cell. Figure 5.1 depicts the local dynamics of the system. One of the noticeable points is the excitation point. This point is a stable point from which a cell can be excited if it is stimulated with a potential larger than u_{th} . The dynamics of the variable u are fast, which should be taken into account for the numerical integration scheme, to keep the approximation stable. The parameter values, used in our experiments, are: $a = 0.3$, $b = 0.01$, $\epsilon^{-1} = 200$. The range of the variables u and v is $[0, 1]$.

5.4 Numerical solution

For the solution of the system of equations, (5.1) and (5.2), we follow [4]. To approximate the local dynamics of our model, the functions $f(u, v)$ and $g(u, v)$ we use an explicit-Euler scheme. To approximate the diffusion term of the model, we use central differences and assume that the nodes are placed uniformly on a square grid with size $P \times Q$. Hence $\Delta x = \Delta y$. To rewrite the laplacian for node (i, j) , $0 < i < P$ and $0 < j < Q$, we use follow the following approach:

$$\frac{\partial^2 u_{i,j}}{\partial x^2} \approx \frac{(u_{i-1,j}^n + u_{i+1,j}^n - 2u_{i,j}^n)}{(\Delta x^2)}, \quad (5.5)$$

and

$$\frac{\partial^2 u_{i,j}}{\partial y^2} \approx \frac{(u_{i,j+1}^n + u_{i,j-1}^n - 2u_{i,j}^n)}{(\Delta y^2)}. \quad (5.6)$$

Adding these two equations and using $\Delta x = \Delta y = h$ results in:

$$\nabla^2 u \approx \frac{(u_{i-1,j}^n + u_{i+1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - 4u_{i,j}^n)}{h^2}. \quad (5.7)$$

Hence, we get the following explicit scheme to calculate u and v for time $n + 1$:

$$u_{i,j}^{n+1} = u_{i,j}^n + \Delta t[f(u_{i,j}^n, v_{i,j}^n) + \frac{(u_{i-1,j}^n + u_{i+1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - 4u_{i,j}^n)}{h^2}], \quad (5.8)$$

$$v_{i,j}^{n+1} = v_{i,j}^n + \Delta t[g(u_{i,j}^n, v_{i,j}^n)]. \quad (5.9)$$

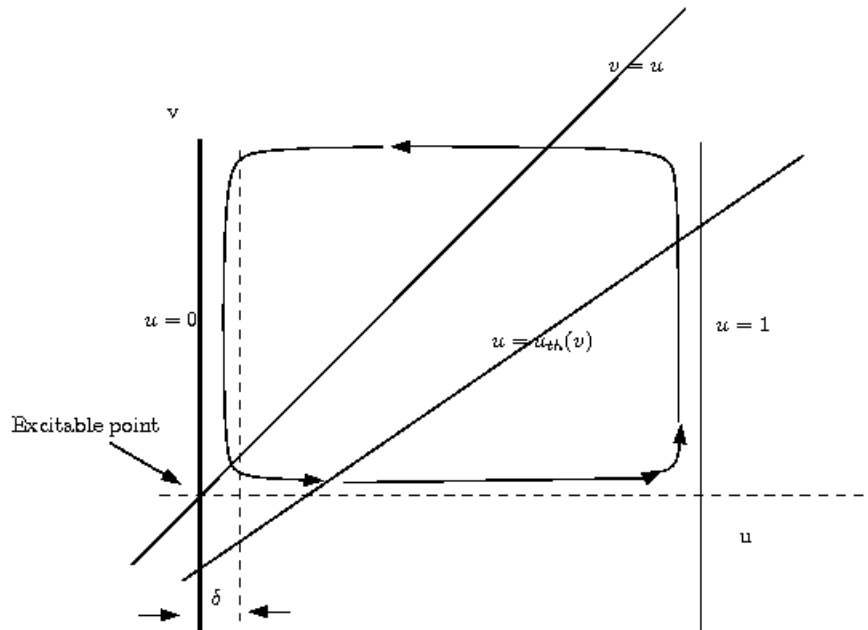


Figure 5.1. Local dynamics. δ denotes a small "boundary layer". If the system is outside this boundary layer, it is excited, otherwise it is recovering. A typical value for δ is 0.001.

5.5 Simulation results

With this model from the previous two sections, we investigate the different aspects of the reentry of the electrical propagation wave. Following [17] we define the following occurrences of reentry:

5.5.1 One-dimensional reentry

For this type of reentry we construct a string of single cardiac cells and connect the tail of the string to the head of the string. Let part of the string be subjected to unidirectional conductance, so that the electrical propagation wave can only propagate in one direction, say clockwise. Stimulate the string in a single cell to initiate the wave. Figure 5.2 depicts the situation.

Let R denote the radius of the string cells, let C denote the propagation speed of the wave and let P denote the perimeter of the string cells. The period (T) of the wave is now defined as $T = \frac{P}{C}$. Let L denote the length of the excited region, which is often defined as the wavelength. The wavelength is now equal to $L = APD \times C$. If $L > P$ it is obvious that the wavefront will be blocked and that reentry cannot occur.

5.5.2 Two-dimensional reentry

The one-dimensional case can be extended to the two-dimensional case by decreasing the radius R . In this way we construct a ring with two radii, an inner radius R_i and an outer radius

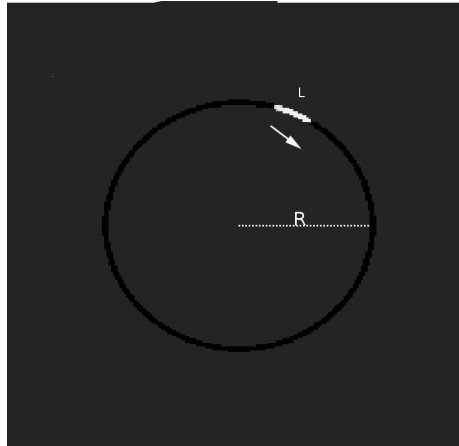


Figure 5.2. One-dimensional reentry. A black color indicates excitable, repolarized tissue, and a white color indicates excited, depolarized tissue.

R_o . Again we have a part of the ring which is unidirectionally conducting, so upon stimulation a wavefront travels clockwise around the ring. See figure 5.3. This type of reentry is called "anatomical reentry" [17].

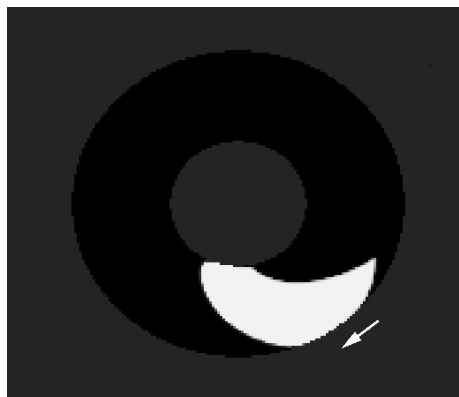


Figure 5.3. Two-dimensional reentry

Note that, as R_i is smaller than R_o and the propagation speed C is equal for every node of the tissue, the wavefront deforms. This deformation increases as R_i decreases.

If we decrease R_i even more, we notice an interesting phenomena. The wavefront cannot remain attached to the non-conducting object and it starts rotating around itself. See Figure 5.5. This type of reentry is called "functional reentry". The central point, around which the wave seems to rotate is called "the core" of the wave. If one defines $\phi(\bar{x}, t)$ as the phase of the wave at $\bar{x} = (x, y)$ the core is the point where the phase is undefined. It is a phase singularity and may be calculated through the topological charge n_t , which is defined as follows:

$$n_t = \frac{1}{2\pi} \oint_c \nabla \phi \cdot \bar{dl}, \quad (5.10)$$



Figure 5.4. Two-dimensional reentry, smaller inner radius.

where the line integral is taken over the path \bar{l} on a closed curve c around the singularity. We refer to [6] for a detailed explanation.



Figure 5.5. Functional reentry

Another typical reentry characteristic is the breakup of the normal, stable wavefront. This breakup can be caused by multiple factors such as conduction block, two wavefronts colliding, or stimulation just behind a passing wavefront. This last factor is shown in Figure 5.6, where a passing wavefront interferes with a new stimulation causing this new wavefront to break and enter in a so called "figure of eight" reentry.

Finally, in Figure 5.7 we show a reentry that developed into a chaotic situation, where there are multiple singularities causing rapid stimulation of each segment in the tissue. This chaotic situation developed over time, from a single wave break due to a figure of eight reentry.



Figure 5.6. Figure eight reentry

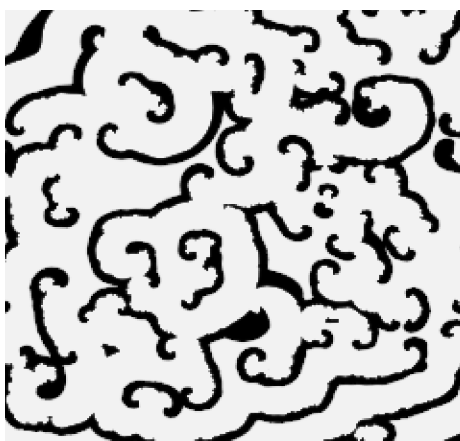


Figure 5.7. Multiple reentry

CHAPTER 6 ESTABLISHING PARAMETERS

In the last few chapters we have seen how one can mathematically model the electrical aspects of the atrial cell. Furthermore, we described the mechanism behind reentry and stated that it is an important factor contributing to AF. In this chapter, we discuss our results in simulating abnormal behavior and AF with a triangular mesh using the Cellular Bidomain Model. Before any simulations can be made, we have to establish a set of parameters to approximate the observed behavior of real cardiac tissue. When this set of parameters is established, small parametric changes will be induced to the model to simulate different states of the tissue, such as mechanical stretch, aging of the tissue and ectopic beats.

6.1 Default parameters

As our experiments should resemble real cardiac tissue, we establish a set of parameters which approximate the electrophysiological properties of cardiac cells as close as possible. Furthermore, we develop a notion of the reactions of the model to small changes of underlying ionic processes, such as stretch-activated channels and the remodelling of L-type Ca^{2+} current. These two ionic processes influence the behavior of the APD significantly and are thought to be of great influence on the forming of cardiac arrhythmia. In this study, we only use the effects of these processes, for a detailed study we refer to Kuijpers et al. [28][27]. The following properties are considered:

- conduction properties;
- anisotropy;
- geometry;
- action potential duration;
- stretch-activated channels;
- remodelling.

To establish working parameters for our simulations, we conducted a series of tests, on an isotropic homogeneous, rectangular piece of tissue. The triangular mesh representing the tissue consists of equilateral triangles. The tissue has a size of 7.5×7.5 cm and consists of 900 nodes and 1682 triangles. The geometry is analyzed by comparing the shape of the circular propagation wave front to a circle. As the modelled tissue is isotropic, the propagation should be equal in all directions. A depolarization wave was initiated by stimulating the central segments of the tissue with a 100 pA/pF current until the membrane was depolarized. We model a depolarization wave front speed of 0.9 m/s based on the thesis of Jacquemet [22]. Our default parameters were chosen to obtain this speed. The series of test were conducted over a period of 100 ms and have the parameters defined in Table 6.1, if not stated otherwise.

Parameter	Definition	Value
g_{int}	Intracellular conductivity	6.25 mS/cm
g_{ext}	Extracellular conductivity	6.25 mS/cm
C_{mem}	Membrane Capacitance	$1.0 \mu\text{F}/\text{cm}^2$
χ	Surface-to-volume ratio	2000 /cm
G_{Na}	Maximum I_{Na} conductance	16 nS/pF

Table 6.1. Tissue parameters

6.2 Simulation results

6.2.1 Levels and frequency and membrane current factor

Four tests are performed to investigate the effect of the spatial distance between nodes and the membrane current factor. We use a piece of tissue with an average spatial distance between nodes of 2.5 mm. This tissue approximates the spatial structure of the MRI model guatria. The tests were performed over a period of 100 ms with varying parameters for the division frequency of the triangles, the number of levels used in the simulation and the membrane current. The first two tests use two levels and a division frequency of two for each level. The first test uses a membrane current factor of 0.0 and the second test uses a membrane current factor of 0.3. On the lowest level of the simulation graph the average spatial distance between two nodes, for test one and two, equals 1.3 mm. The third and fourth tests use three levels and a division frequency of three for each level. The third test uses a membrane current factor of 0.0 and the fourth test uses a membrane current factor of 0.3. The third test uses three levels and a division frequency of three for each level. On the lowest level of the simulation graph, the average spatial distance between two nodes for test three and four equals 0.27 mm. Figure 6.1 shows the result for the first test. The average speed of the wavefront equals 0.48 m/s. The circular shape of the wavefront indicates an isotropic propagation. Figure 6.2 shows the result for the second test. The membrane current factor of 0.3 results in an increase of the conduction velocity to 0.78 m/s, but also results in a deformation of the wavefront shape. The non-circular shape indicates that the propagation is not isotropic. Figure 6.3 shows the result for the third test. Using a higher level and frequency, the activation time pattern has a circular shape. The average velocity of the wave front is 0.80 m/s. Figure 6.4 shows the result for the fourth test. Using the parameters of the third test, the membrane current factor is increased to 0.3. This results in a higher average velocity of 0.87 m/s, but has no influence on the shape of the wave front pattern. To get a feeling for the error in geometry, we define the following error function:

$$E = \sqrt{\frac{1}{N} \sum (e_i - \bar{e})^2} \quad (6.1)$$

$$e_i = t_{act}^i - t_{exp}^i \quad (6.2)$$

$$t_{exp}^i = \sqrt{\frac{x^2}{v^2} + \frac{y^2}{v^2}} \quad (6.3)$$

where N equals the number of elements and e_i is the error of element i . This error is defined as the difference of the activation time of element i (t_{act}^i) and the expected activation time of element i (t_{exp}^i) using the average AP velocity. \bar{e} is defined as the average error. Intuitively, this error measures how much the AP wavefront resembles a circle. The results for this error function are listed in Table 6.2.

Test			Error
division frequency	levels	memcurfactor	
2	2	0.0	0.0015 s
2	2	0.3	0.0011 s
3	3	0.0	0.0004 s
3	3	0.3	0.0004 s

Table 6.2. Geometry error.

Table 6.2 suggests that using a division frequency of three and three levels results in the smallest geometric error made. It also suggests that a memcurfactor of 0.3 results in a smaller geometric error for a division frequency of two and two levels. If the error in speed is taken into account as well, it is clear that a memcurfactor of 0.3 has to be preferred for a division frequency of two and two levels. Compared to a division frequency of three and three levels, a memcurfactor of 0.0 differs by 0.32 m/s, whereas a memcurfactor of 0.3 differs by 0.02 m/s.

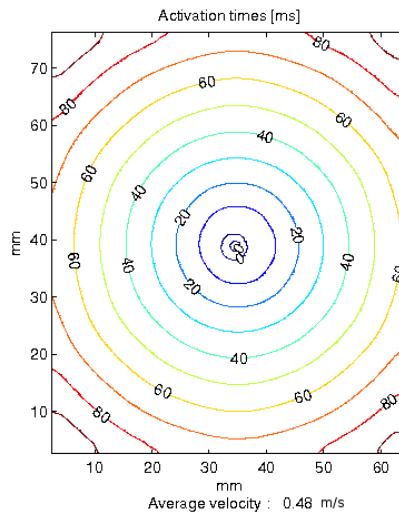


Figure 6.1. Activation times and average velocity for division frequency of two and two levels, using a membrane current factor of 0.0.

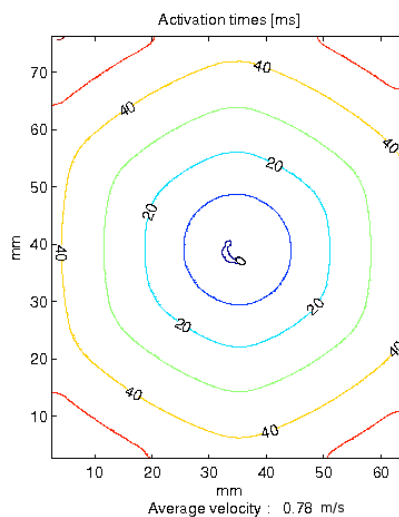


Figure 6.2. Activation times and average velocity for division frequency of two and two levels, using a membrane current factor of 0.3.

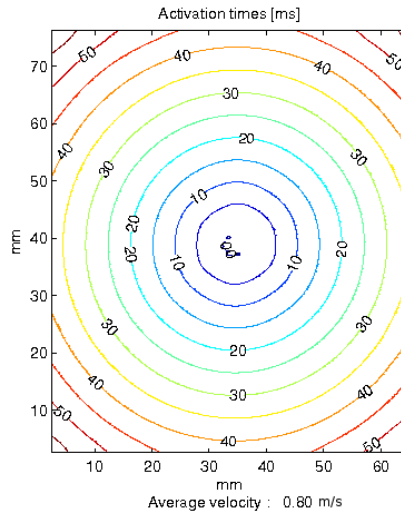


Figure 6.3. Activation times and average velocity for division frequency three and three levels, using a membrane current factor of 0.0.

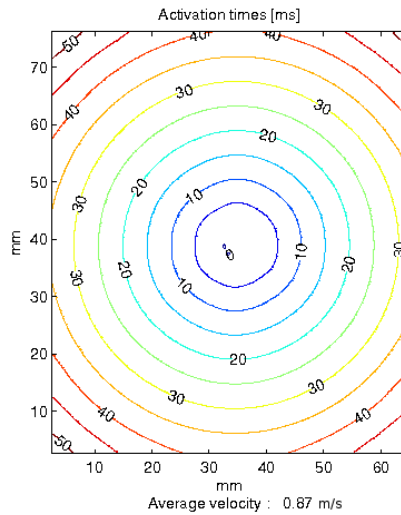


Figure 6.4. Activation times and average velocity for division frequency three and three levels, using a membrane current factor of 0.3.

6.2.2 Diastolic interval

As mentioned in Chapter 5, the action potential duration (APD) is an important factor in the mechanics of AF. Studies show that the APD decreases significantly when the diastolic interval (DI) becomes small (< 200 ms). For the sake of completeness we include the results of Xie et al. [45] to show the influence of the DI on the APD. The stimulation protocol used for these experiments is

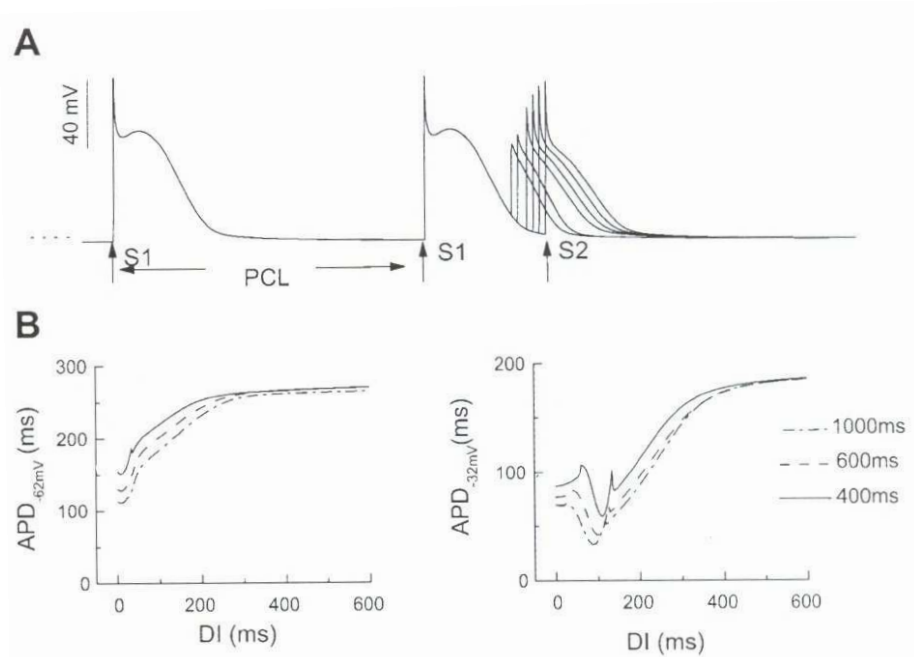


Figure 6.5. Single cell APD restitution as a function of pacing cycle length (PCL). A: S1S2 protocol for measuring APD restitution. APs are superimposed as the S1S2 coupling interval is progressively shortened. B: APD_{-62mV} (left) and APD_{-32mV} (right) of the S2 beats versus the DI, showing APD restitution curves for PCL=400 ms (solid line), 600 ms (dashed line) and 1000 ms (dashed-dotted line). From [45].

S1S2 protocol. (See Figure 6.5.)

6.2.3 Burst pacing

A different stimulation protocol used in many simulations, is the burst pacing protocol. Courtemanche et al. [37] show that, in their model, the APD adapts to the stimulation rate of the tissue, which is in agreement with experimental observations [2]. Another example is an experiment of Alessie et al. [2]. These canine, experimental observations show that rapid, short bursts of stimuli (2 s, interval 20 ms) result in sustained AF. To investigate these findings for the chosen parameter set, we conduct a series of 5 test with stimulation rates of 1, 5, 10, 15 and 20 Hz. over a period of 5 s. The rectangular tissue has a size of 2×2 mm, the spatial distance between nodes equals 0.1 mm. An AP is initiated by stimulating the upper left corner segments of the tissue. The nodes from which the membrane potential data is collected are situated in the lower right corner.

Figure 6.6 illustrates the influence of the stimulation frequency on the shape and duration of the AP. During a 1 Hz stimulation, the APD is approximately 375 ms and the segments are fully repolarized before the next AP arrives. During a 5 Hz stimulus, one notices that roughly every other pulse an AP is formed. The APD is hardly influenced. The 10 Hz stimulus results in a deformed AP. For example, at 2.3 s the AP increases after the initial depolarization and repolarizes faster than normal. The APD is approximately 325 ms. The APs of a 15 Hz stimulation resemble the APs of a 5 Hz stimulation. The APs of a 20 Hz stimulation, however, show unusual behavior. The AP around 2.2 s does not repolarize completely when a new depolarization is initiated at 2.6 s.

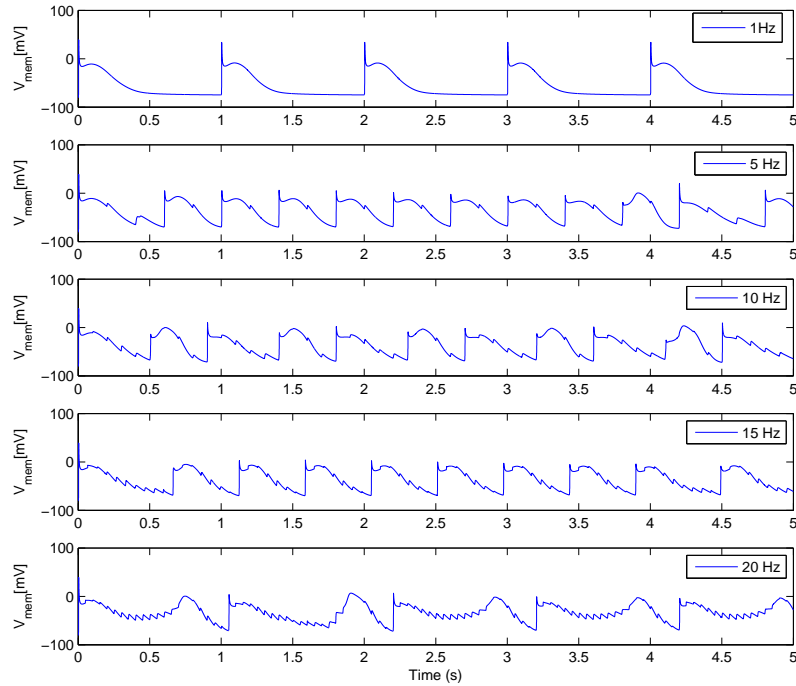


Figure 6.6. Action Potential shape of tissue stimulated with 1, 5, 10, 15 and 20 Hz, respectively.

6.2.4 Stretch Activated Channels

To investigate the effect of stretch on the AP we conduct tests with constant stretch applied to the tissue. The applied stretch equals a factor of 1.05, 1.1 and 1.2 for the consecutive tests. The rectangular tissue has a size of 2×2 mm, the spatial distance between the nodes equals 0.1 mm. An AP is initiated by stimulating the upper left corner segments of the tissue. The nodes from which the membrane potential data is collected are situated in the lower right corner. Figure 6.7 illustrates the influence of stretch on the shape of the AP. The increase of stretch causes a prolongation of the repolarization period and an increase of the resting potential. The prolongation of the repolarization of the cell decreases the possibility of a reentry, due to the increase in wavelength. (See Section 5.5.1.)

6.2.5 Remodelling

To investigate the effect of the remodelling parameter ρ , see Kuijpers et al. [28], we conducted a series of five tests, during which the parameter was varied between -1.0, 0.0 and 1.0. The rectangular tissue has a size of 2×2 mm and the spatial distance between nodes equals 0.1 mm. An AP is initiated by stimulating the upper left corner segments of the tissue. The nodes from which the membrane potential data is collected are situated in the lower right corner. The influence of the remodelling parameter is illustrated in Figure 6.8. A negative remodelling parameter shows a

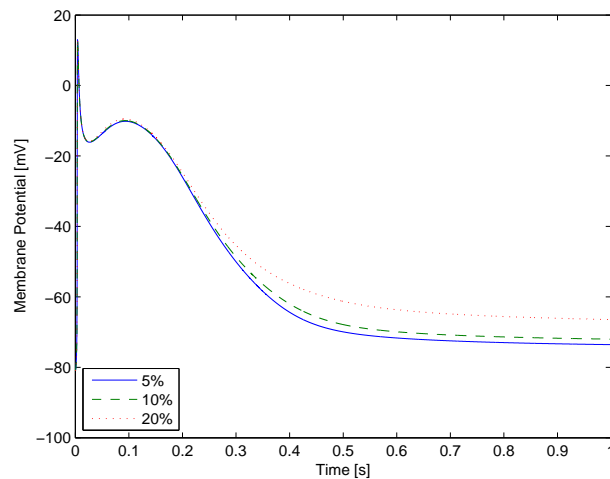


Figure 6.7. The influence of stretch on the shape of the Action Potential. The figure depicts the influence of 5%, 10% and 20% stretch, respectively.

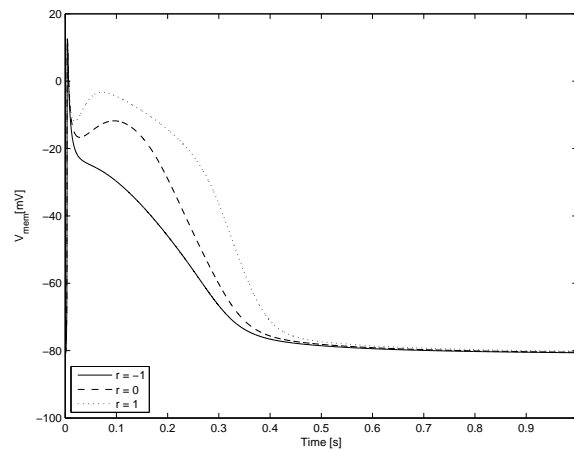


Figure 6.8. The influence of remodelling on the shape of the Action Potential. The figure depicts the influence of the remodelling parameter ρ .

decrease of the refractory period. This decrease results in a smaller wavelength which increases the chance of reentry.

For the varying of the density of the currents $I_{Ca,L}$ and I_{to} we mainly refer to Courtemanche et al. [9]. For our experiments we use the values listed in Table 6.3, and Figure 6.9 illustrates the effect of these current changes on the APD. From Figure 6.9 it is clear that the remodelled values results in a decrease of the refractory period, which results in a smaller wavelength and an increase in the chance of reentry.

6.3 Conclusion

In this chapter we have determined the model parameters and the effect of stretch-activated currents and remodelling on the shape of the AP. Furthermore, we have investigated the error made

Parameter	Default Value	Remodelled Value
$g_{Ca,L}$	0.1652 nS/pF	0.0826 nS/pF
g_{to}	0.1238 nS/pF	0.02476 nS/pF

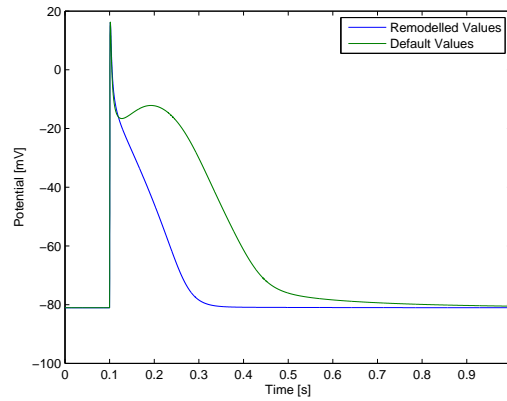
Table 6.3. Values for $I_{Ca,L}$ and I_{to} .

Figure 6.9. The effect of varying current density of $I_{Ca,L}$ and I_{to} on the APD. The default value and the remodelled value for the two currents.

using different division frequencies and levels for the tissue. Our default parameters give us an AP speed of 0.87 m/s and, ideally, we should use a division frequency of three and three levels to minimize the computation error.

CHAPTER 7 SIMULATING ATRIAL TISSUE

In the previous chapter, we investigated the influence of various model parameters on the propagation velocity and shape of the AP. We also saw the relation between rapid stimulation and the adaptation of the APD. In this section, we study the simulation of AF on a triangular mesh, using the Cellular Bidomain Model. For the simulation we use a homogeneous, isotropic tissue with an intracellular and extracellular conductance of 6.25 mS/cm and a maximum G_{Na} conductance of 7.8 nS/pF. The lower G_{Na} conductance results in a wavefront velocity of around 0.5 m/s. This lower velocity reduces the wavelength, which facilitates AF. For experiments with similar results we refer to [40]. The used geometry is obtained from an MRI scan, provided by Peter van Dam and Adriaan van Oosterom [43], with a total of 3800 nodes, and 7446 triangles. We initiate a wave front by stimulating a node that coincides with the SA node. For the simulation graph we used two levels and a division frequency of two.

7.1 Simulation results

7.1.1 Single pulse

This simulation is used as a reference for the parameters. The default parameters, specified in the previous section, should result in a depolarization wave front that resembles a normal heart beat. For this simulation G_{Na} is increased to 16 nS/pF to reach a wavefront velocity of around 0.9 m/s. For the simulation graph we use three levels and a division frequency of three.

Figure 7.2 illustrates our results for the single pulse simulation. The AP pulse is initiated at the approximate location of the SA node at $t=0$ ms. The wave front progresses in a circular pattern and reaches the end of the atrium in $t=110$ ms. Due to the large refractory period, the wave front is cancelled out and the tissue is almost fully repolarized at $t = 250$ ms. The results are in accordance with well-known physiology text books [18].

7.1.2 Burst pacing with remodelling

To induce cardiac arrhythmia, we started with a so-called Burst-pacing protocol [22]. This simulation is conducted by stimulating a single segment of tissue with a constant interval. In experiments this kind of stimulation is often used to induce AF. For example Akira et al. show that prolonged left atrial pacing results in AF [1]. The stimulation frequency in our simulations was 10 Hz. To increase the susceptibility of the tissue to arrhythmia we used a remodelling factor $\rho = -1$.

Figure 7.3 illustrates the results of the burst pacing simulation. The pacing frequency of 10 Hz results in a 100 ms pacing interval. As this interval is smaller than the repolarization period, most pulses will not initiate a new AP. This is shown in the figure at 102 and 202 ms. At 402 ms, the tissue around the SA node is partly repolarized and the stimulus results in a new AP. The resulting wavefront, however, is partly blocked and progresses to one side. This sequence is repeated for a few seconds and is depicted in Figure 7.3. The results of the simulation indicate cardiac arrhythmia due to the burst pacing protocol, but do not show a progression towards AF. This is mainly due to the large refractory period, which prevents a reentry.

7.1.3 Stretch

Atrial tissue is nonhomogeneous of nature [31]. We incorporated this fact into the simulation by varying the thickness of the simulation tissue. Inspired by Peter van Dam [42], we calculated

the shortest path, from the SA node to every other node in the simulation graph, using Dijkstra's algorithm [11]. Next, we counted the number of times a node is used in a shortest path. The more often a node is used in a shortest path the thicker the tissue is modelled at that position. With this method we simulate tissue fibres running over the atria. For a graphical illustration of the result, see Figure 7.1. The simulation is conducted to investigate the effect of stretch on the propagation of the AP. From our previous experiments, we know that stretch induced on the tissue of the atria results in a prolongation of the repolarization period of the cells. In general, this prolongation results in a greater wavelength and impedes reentry. Kuijpers et al. [27], however, show that thicker tissue diminishes the effect of stretch and hence we expect the simulated fibres to facilitate conduction.

The stimulation protocol used is a so-called programmed stimulation protocol (S1-S2-S3) [22]. Three stimulation segments are used: a first stimulus is applied at segment S1, followed by subsequent stimulations at segments S2 and S3. The interval between the different stimulations is arbitrary and two stimulations may coincide. The amount of stretch ratio, applied over three simulations, is 5%, 10% and 20% respectively. The figures 7.4, 7.5 and 7.6 depict the results of stretch applied to nonhomogeneous tissue. Note that tissue cells, which are modelled thicker, repolarize faster than thinner modelled tissue cells. When a stretch factor of 1.05 is applied to the tissue, the cardiac cells are largely repolarized when a second stimulus is applied at 560 ms. The new stimulus causes the cells to depolarize and a normal wavefront travels across the atrium. When a stretch factor of 1.1 is applied, however, the difference in thickness of the cardiac cells plays a role. Thicker tissue, such as the fibres, is repolarized while thinner tissue is still depolarized. The effect is a unidirectional block and causes the wavefront to break. If we increase the stretch factor even more to 1.2, all tissue cells are still depolarized and a new stimulus has no effect.

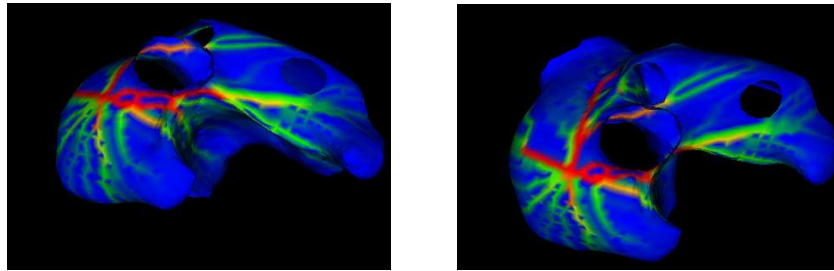


Figure 7.1. Shortest path

7.1.4 Simulation of atrial flutter

Geometric structures can cause perturbations in the AP wavefront, which may lead to circular and spiral waves. In this simulation the Superior Vena Cava, in combination with unidirectional block, is used to create a sustaining flutter. The first stimulation at $t=0$ ms is situated at the SA node and is used to create a repolarization wavefront at $t=325$ ms. The second stimulation, at $t=370$ ms, is again situated at the SA node. The third stimulation at $t=380$ ms, consisting of three simultaneous stimulations, is situated just after the repolarization front. For this simulation we use a maximum G_{Na} of 7.8 nS/pF and a remodelling factor $\rho = -1$. Figure 7.7 illustrates our strategy.

The first stimulus produces a wavefront, which progresses normally over the atrium. The second stimulus produces the wavefront which is to be blocked. The third stimulus is positioned just after the repolarization wave and ahead of the wavefront of stimulus two. This results in a unidirectional block of the second wavefront and causes this wavefront to attach to the Superior

Vena Cava. Due to the remodelling factor $\rho = -1$ the wavelength is small enough to allow a reentry at the SA node, which results in a stable rotor wavefront.

7.1.5 Simulation of AF with adjusted I_{Na} , I_{Ca} and I_{to}

Even though our previous simulations showed cardiac arrhythmia and even a stable rotor wavefront, we did not achieve a situation where sustained AF was induced. In most cases we either observed conduction block due to a large refractory period or we needed geometric structures to induce anatomical reentry. For functional reentry, however, we need to alter the properties of our model, such that a relative slow wavefront velocity combined with a reduced refractory period result in spiral waves.

For the initiation of the wavebreak we use an S1-S2 protocol (See Figure 7.9). The initial stimulus is initiated at $t=0$ ms at the SA-node. The second stimulus, which has a rectangular shape, is initiated at $t=300$ ms and is timed just behind the first wavefront to create a unidirectional block. The unidirectional block and the rectangular shape of the second stimulus result in a figure of eight reentry at $t=500$ ms. This figure of eight reentry is continued until $t=1600$ ms, when a spiral wave is formed.

7.2 Conclusion

Our simulations show that acute dilatation of cardiac tissue results in block of the AP, which coincides with simulations of inhomogeneous fibres by Kuijpers et al. [27]. The block of the AP results in a wavebreak and forms a figure of eight. The wavebreak does not progress towards a reentry due to the duration of the AP, which is prolonged by the applied stretch. Furthermore, our results show that a stable rotor wavefront can be created with a wavelength which is normally too large to allow a functional reentry. This rotor wavefront was initiated with remodelling and a decrease of the maximum I_{Na} . Note that even though we used remodelling the wavelength was still too large to enable a functional reentry. Only if we remodel $I_{Ca,L}$, I_{to} and I_{Na} the wavelength is small enough to allow occurrence of functional reentry. Finally, our simulations of rapid pacing show a break of the wavefront, that does not progress towards AF. Literature [2] shows a relative ease in developing of AF during rapid pacing, one could hypothesize that the change of the APD, due to the diastolic interval, must be the cause of this kind of AF. Further experiments are needed to shed more light on this problem. Instead of simulating rapid stimulation with a fixed interval, one would need to simulate different diastolic intervals precisely. The need for precise intervals, however, could indicate a weakness of the underlying model as this does not coincide with the reported ease of initiating AF with rapid stimulation.

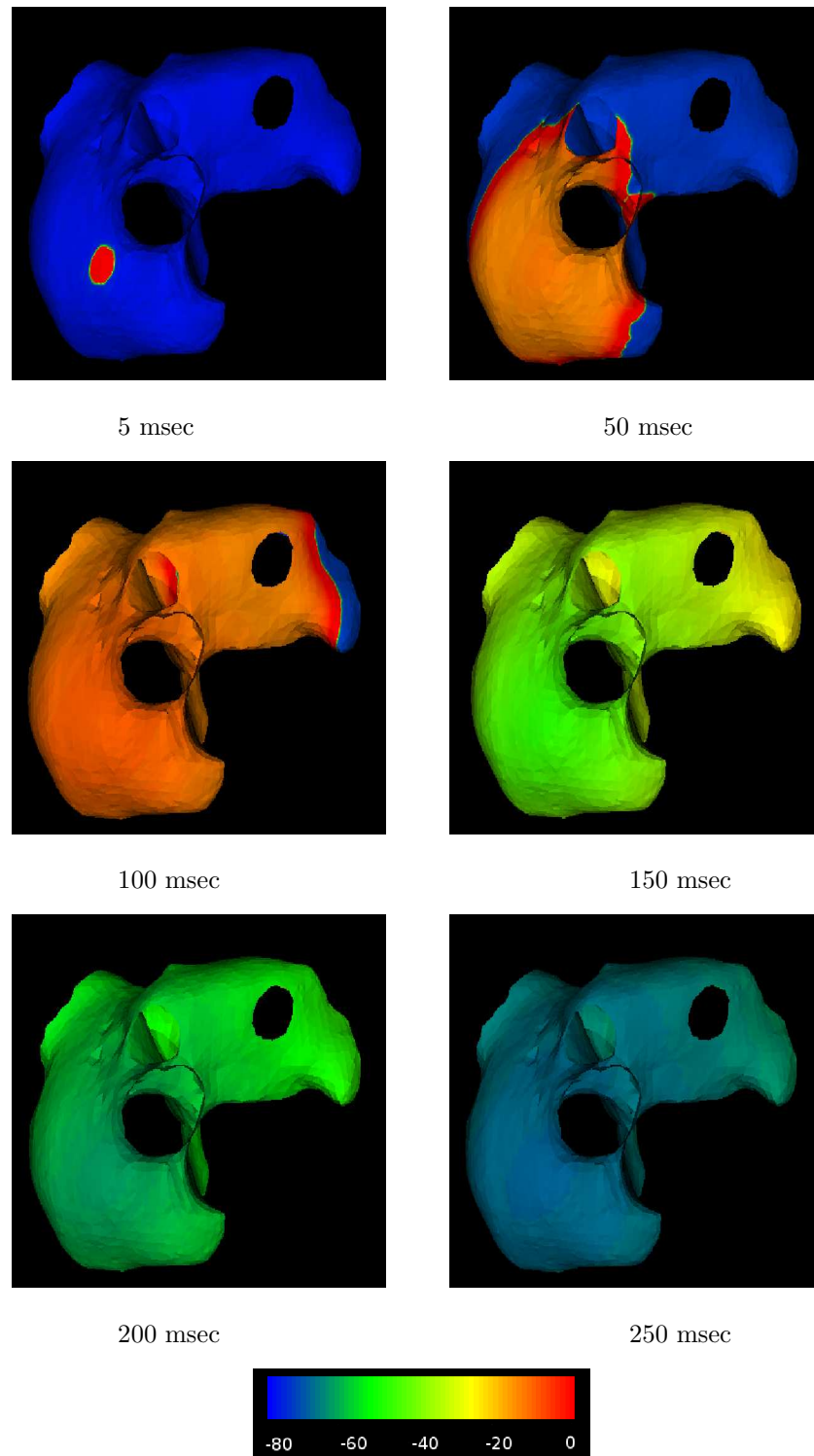
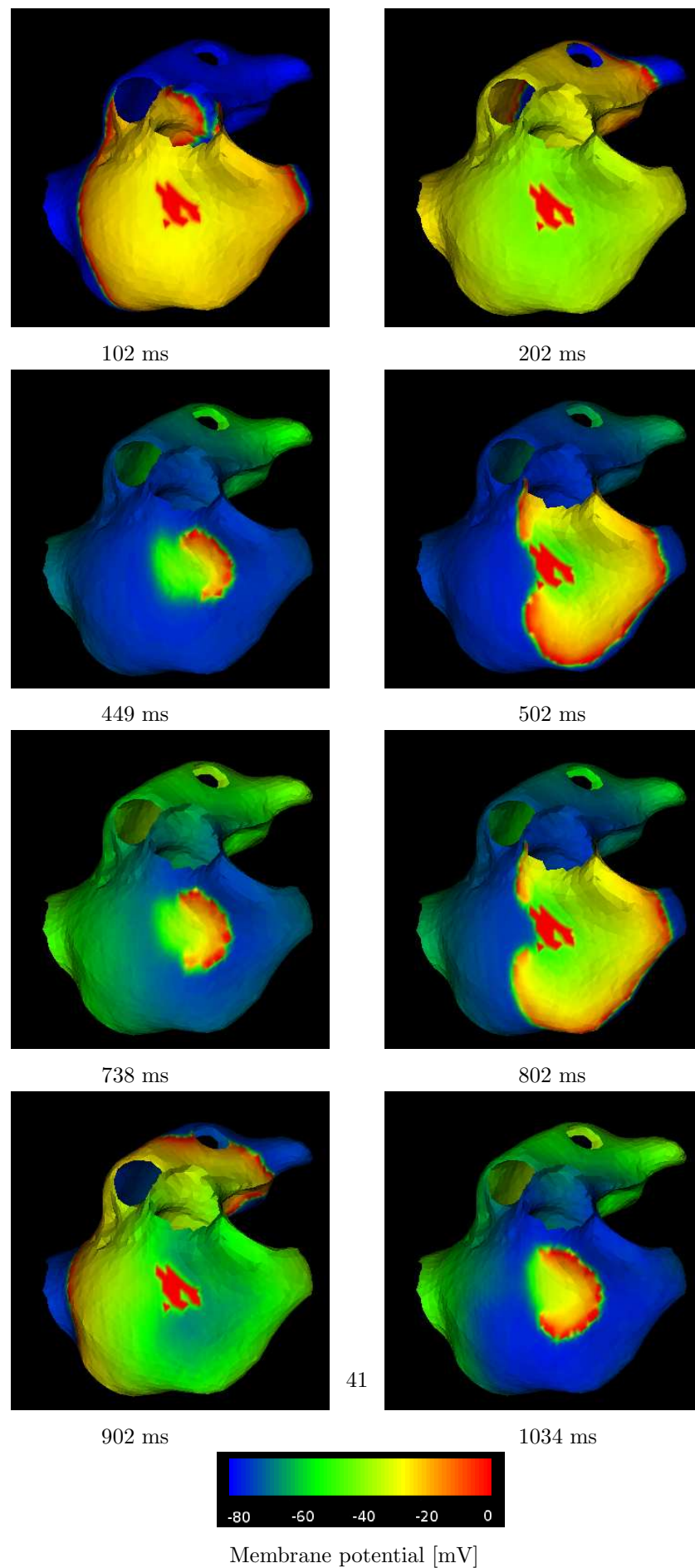


Figure 7.2. Single pulse action potential.

Figure 7.3. 100 ms interval pacing, with remodeling factor $r = -1$.

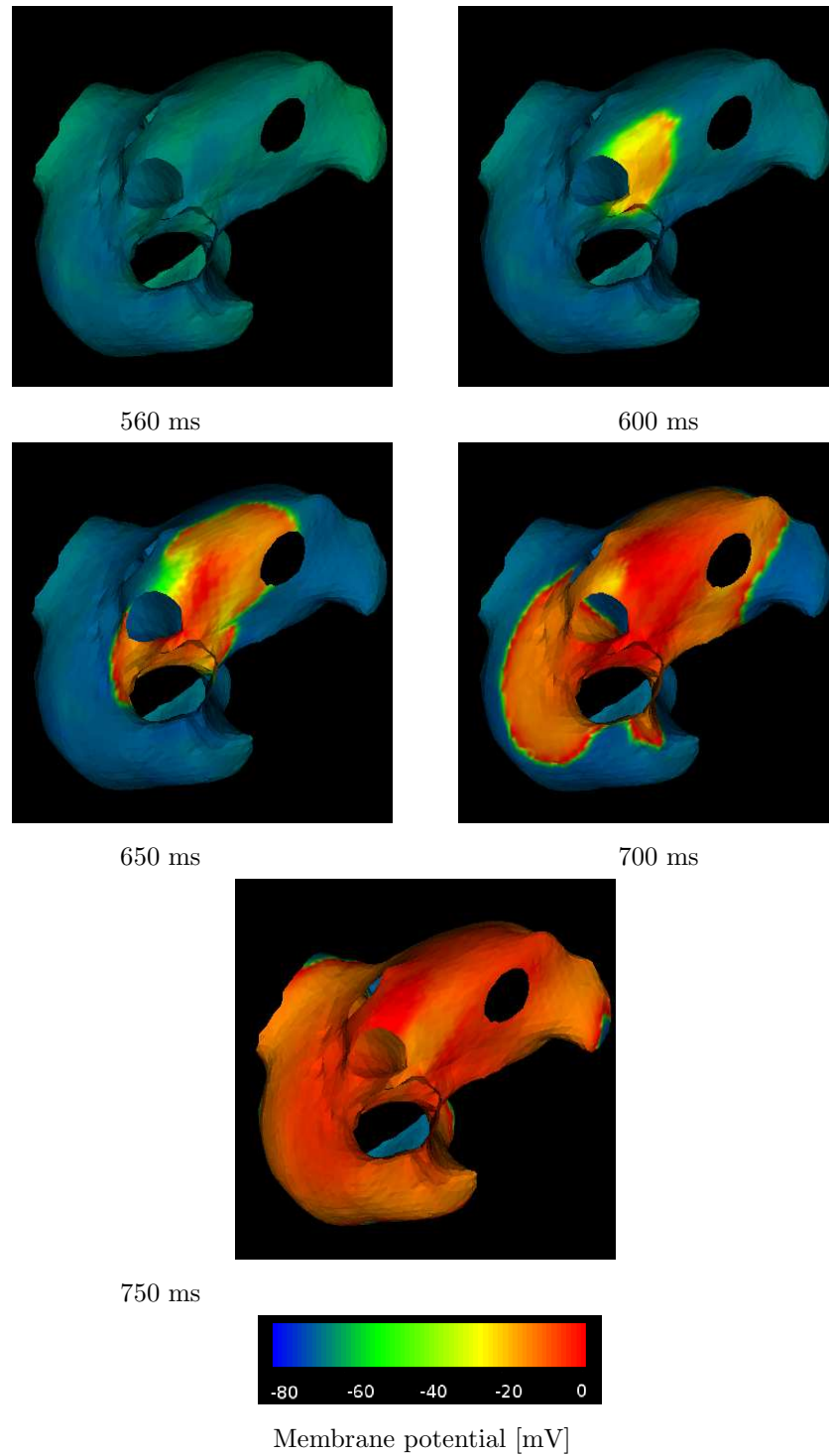


Figure 7.4. 5% Stretch, with non-homogeneous tissue.

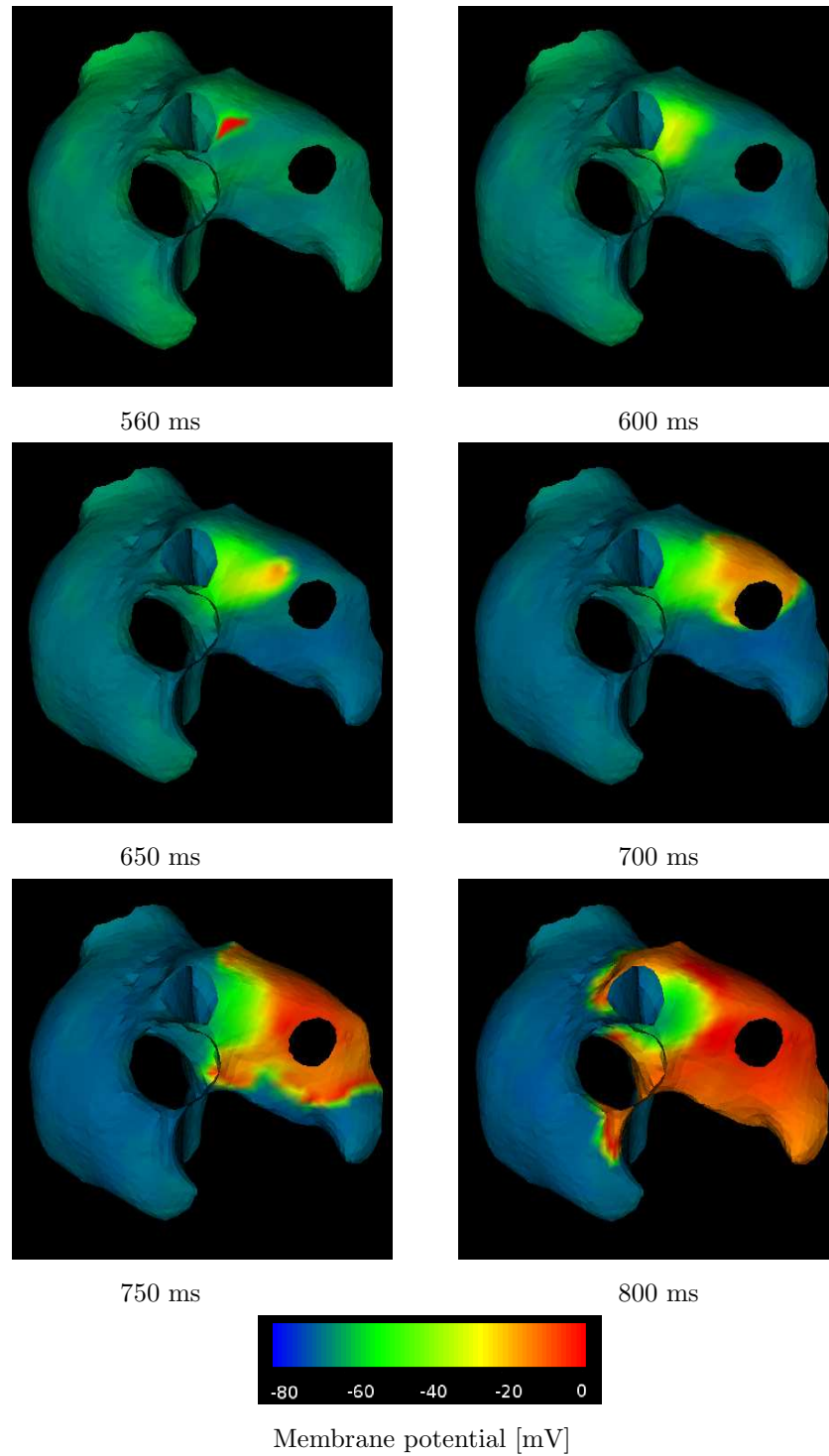


Figure 7.5. 10% Stretch, with non-homogeneous tissue.

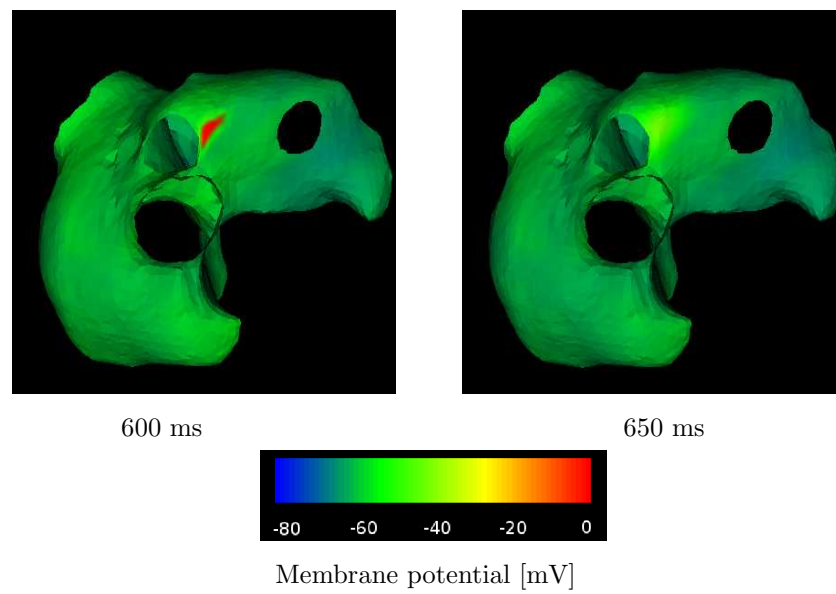


Figure 7.6. 20% Stretch, with non-homogeneous tissue.

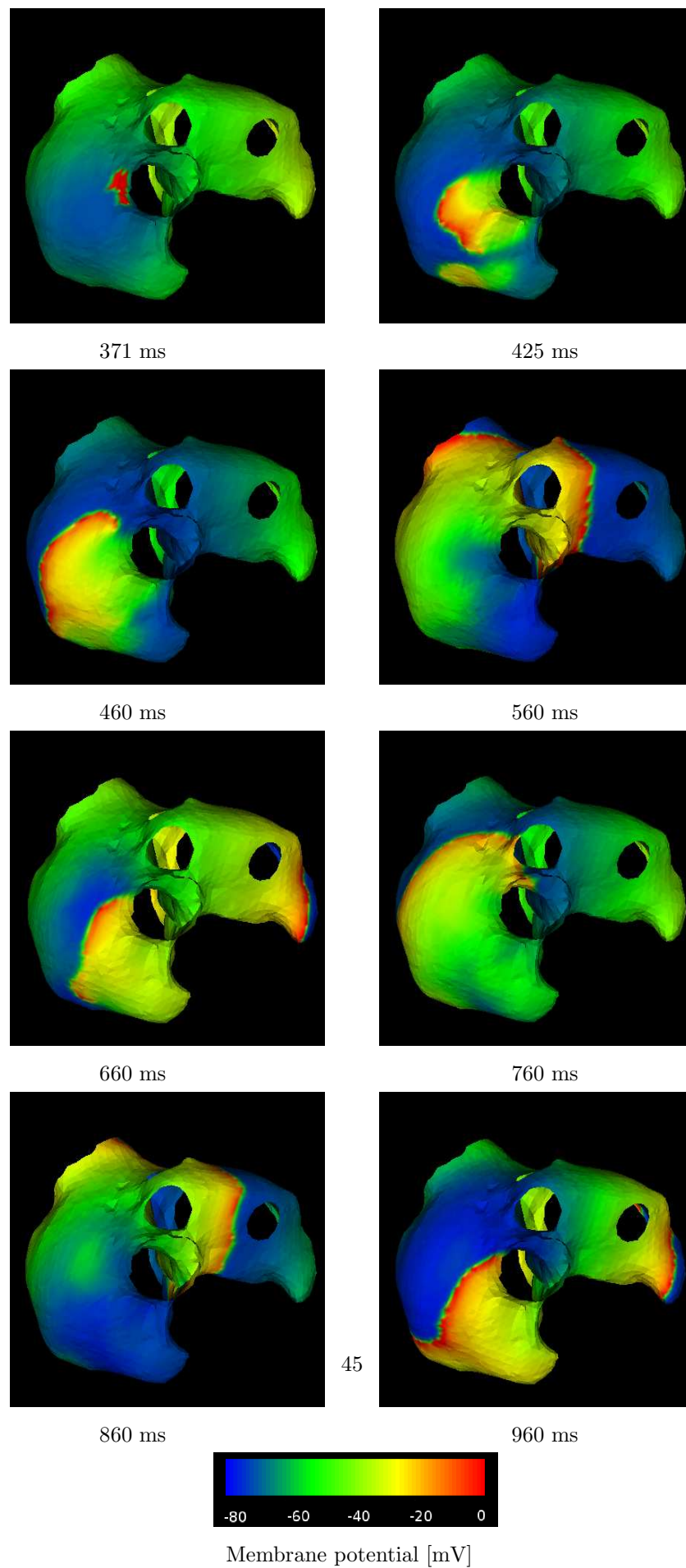


Figure 7.7. S1-S2-S3 combined with remodelling.

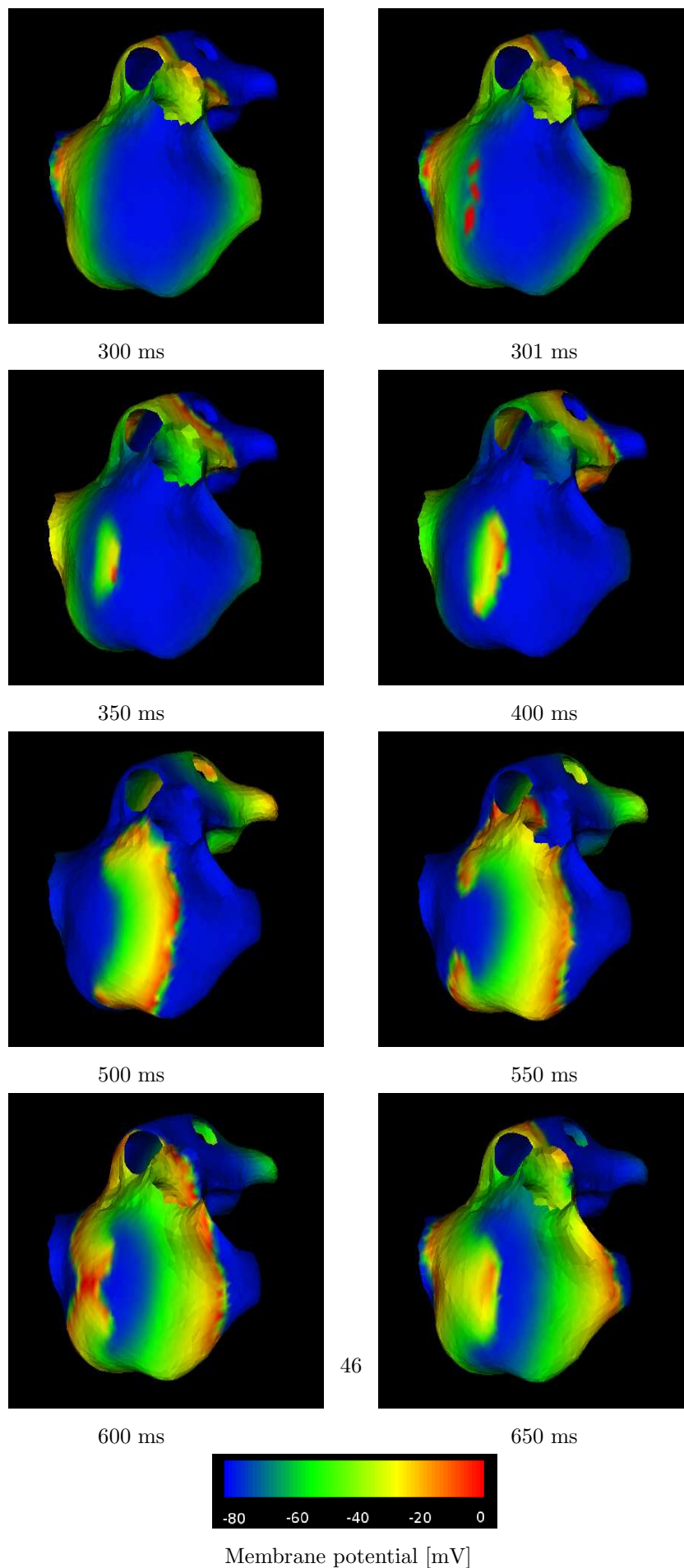
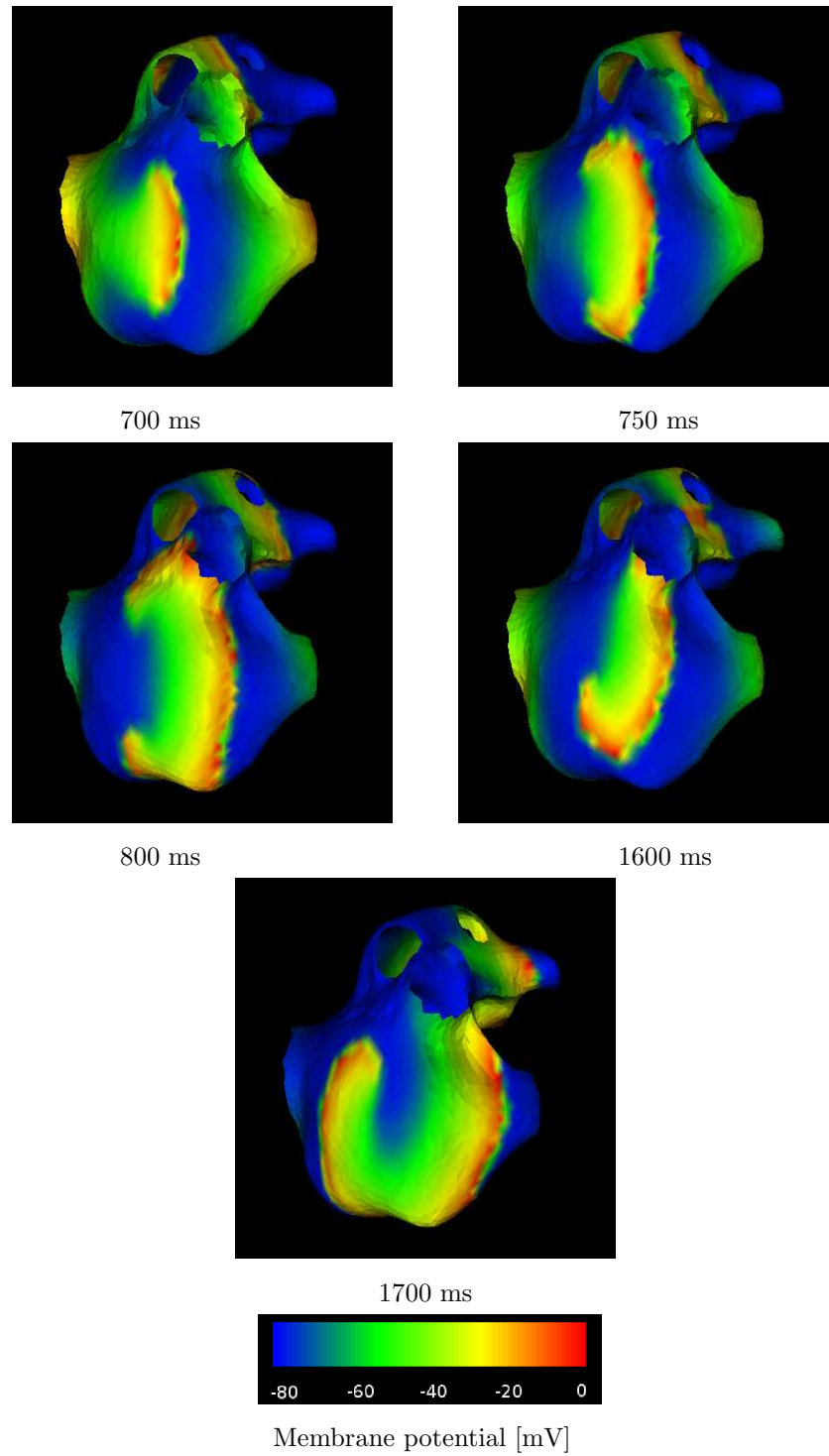


Figure 7.8. Simulating AF with adjusted I_{Na} , I_{Ca} and I_{to} .

Figure 7.9. Simulating AF with adjusted I_{Na} , I_{Ca} and I_{to} (continued)

CHAPTER 8 GENERAL-PURPOSE COMPUTATION ON GRAPHICS HARDWARE

8.1 Introduction

Simulations conducted in the previous chapter consumed an enormous amount of time. For the simulation of a few seconds of real time, a week or more of computation time is needed. In general, the complexity of numerical computations of a set of equations often requires a considerable amount of time and resources. For this reason, parallel solutions have been studied extensively. An alternative solution might be the implementation on graphics hardware. Recent development of graphics processing units (GPU's) show an increase in computation power, which outperforms the CPU on many areas. The fundamental difference between the CPU and the GPU is the architecture. Where the former is optimized for the computation of sequential code, the latter is optimized for problems suitable for parallel solutions. A consequence of this architectural difference is the growth of performance, of CPU versus GPU. In general, the performance growth of the CPU is a factor of 1.4 per year, this rate of growth is overshadowed by the growth of GPUs (See figure 8.1) [33]. In the last few years, the programmability of the GPU has increased, with the latest trend towards a programming environment similar to C or C++. In this chapter, we introduce general purpose computation on the GPU (GPGPU) and present a solution to the reaction-diffusion equations of the Dwight-Barkley model (See Chapter 5).

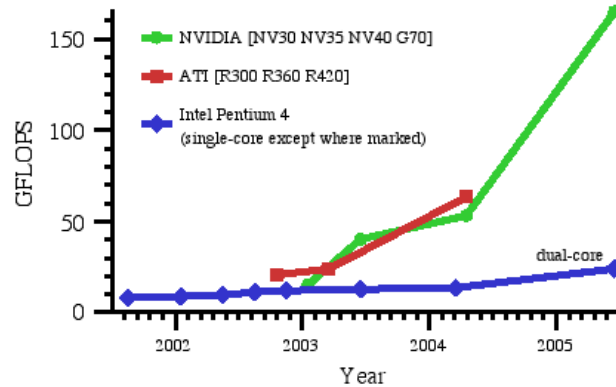


Figure 8.1. Programmable floating-point performance of GPUs. (GFLOPS = one billion floating-point operations per second.) From [33]

8.2 Graphics processing unit

The GPU's architecture is optimized for the handling of graphics primitives. Hence, a lot of terminology is rooted in graphics processing. For the sake of brevity we assume a basic understanding of the graphics terminology; for a description of the used terms we refer to [39]. The architecture of the GPU is often referred to as a pipeline and is depicted in Figure 8.2. The pipeline can contain hundreds of processors, which operate parallel on the incoming data. The processors are divided into two groups: fragment processors and vertex processors.

The vertex processor is a programmable unit that is responsible for the handling of vertices of a polygon to be drawn. The calculations of this stage are mostly transformations and calculations regarding lighting. The fragment processor is a programmable unit responsible for the handling of pixels and their final values. This last stage of the pipeline is the most important part for general

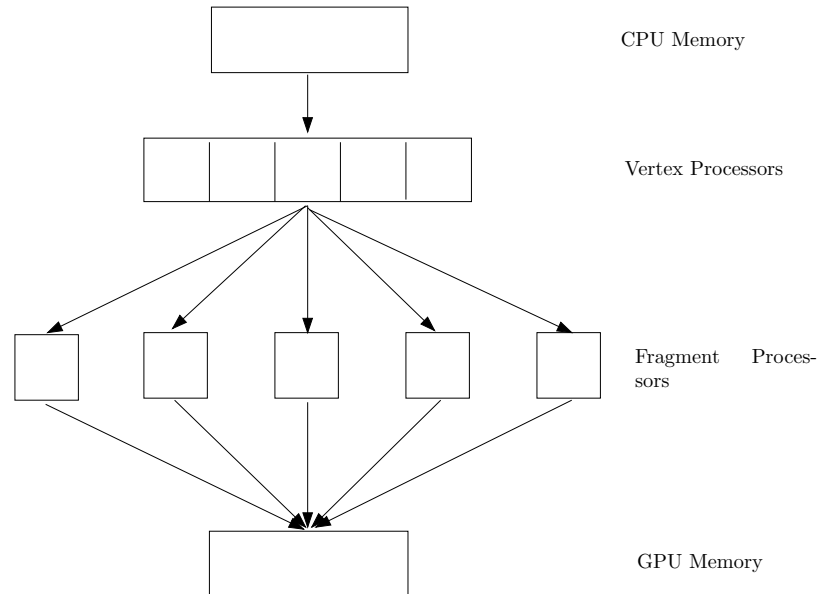


Figure 8.2. GPU pipeline

purpose computing, as this is the place where the textures are applied. We will cover this in more detail in the next section. After the data has been processed by the processors, the result is written to GPU memory, the framebuffer. This framebuffer can be displayed to the screen, used in a subsequent calculation or transferred to the CPU memory. The vertex and fragment processor are build up from several processors which operate in parallel on the incoming data. The programming of the processors is done through so called "shaders". A shader is a piece of code that runs on all processors and is applied parallel on the incoming data.

Following [33] one can discern the following general steps for the programming of the GPU:

1. First, one has to divide the problem into independent subproblems, which can be calculated in parallel. The program, which is applied to all the independent subproblems, is written in a language optimized for the GPU and is called a shader. The input of the shader is provided as an array of data by means of one or more textures. Multiple textures can be used to read from, but only a single texture can be used to write to and will contain the results.
2. To perform a calculation and apply the shader to the data, one has to perform a drawing operation on the GPU. This drawing operation can result in an output to the screen, but can also be done to the framebuffer. A typical method is the drawing of a graphic primitive such as a quadrilateral.
3. To perform the calculation, the GPU divides the data into separate, independent parts and applies the shader to each of these segments. If the problem size is too great to handle in a single pass through the pipeline, one can designate subdomains by means of texture coordinates, and perform multiple cycles to compute the problem.

The results of the computation is written to GPU memory in the form of a single texture, which can be reused for a next calculation, displayed or transferred to CPU memory.

8.3 Reaction-diffusion model

As an example of an GPGPU programming we will implement the Barkley model [4] from chapter 5 on the graphics hardware.

For the implementation we start with the algorithm presented in section 5.4 for the solution of the model. For each subsequent time step the following algorithm must be solved:

$$u_{i,j}^{n+1} = u_{i,j}^n + \Delta t [f(u_{i,j}^n, v_{i,j}^n) + \frac{(u_{i-1,j}^n + u_{i+1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - 4u_{i,j}^n)}{h^2}], \quad (8.1)$$

$$v_{i,j}^{n+1} = v_{i,j}^n + \Delta t [g(u_{i,j}^n, v_{i,j}^n)]. \quad (8.2)$$

The division of this problem into independent, separate subproblems is actually not too difficult. The approximation of the Laplacian of 8.1 can be visualized by a "pictorial operator" used in [10].

$$\nabla^2 u \approx \frac{1}{h^2} \begin{Bmatrix} & & 1 & & \\ & 1 & -4 & 1 & \\ & & & & \end{Bmatrix} u_{i,j}, \quad (8.3)$$

which indicates that the approximation to the laplacian can be calculated for an element $u_{i,j}$ of the domain of interest by adding the values of its neighbors to $u_{i,j}$, subtracting $4u_{i,j}$ and dividing by h^2 . So, by implementing this simple scheme we can calculate each element $u_{i,j}$ independently as long as the values of the neighboring elements are available.

For the implementation of equation (8.1) on the GPU, we proceed in the following way. First we create two shader programs which will update the u and v variable respectively. The two programs are a direct translation of the algorithm (8.1). Furthermore, we construct two data arrays (textures) which hold the values for u and v . Both textures are passed to the shaders as input variables. For each step of the computation, we first update the values of v after which the values of u are updated.

The complete implementation is presented in Appendix A . Note that we use OpenGL and GLUT as graphic programming environments to facilitate the programming of the GPU. We refer to the tutorial of Dominik Goddeke [16] for the specific technical aspects.

8.4 Results

To investigate the effectiveness of the GPU, we performed 6x3 simulations, using an ATI x1650 card (12 fragment processors), a NVIDIA GForce FX 5200 card (4 fragment processors) and the CPU as reference. To vary the calculation load we computed the Barkley model for 32×32 , 64×64 , 128×128 , 256×256 , 512×512 and 1024×1024 elements. As a measure of performance we used the number of displayed frames the program produces per second (FPS). Figure 8.3 shows the results of the simulation.

From 8.3 it is clear that the speedup factor increases with the increase of used elements. For example, with 64×64 elements the ATI has a speedup factor of 3.5, the NVIDIA has a speedup factor of 1.4, compared to the CPU. Using 512×512 elements, the ATI has a speedup factor of 81 and the the NVIDIA has a speedup factor of 6.4, compared to the CPU. This result emphasizes the effectiveness of the parallel architecture of the GPU.

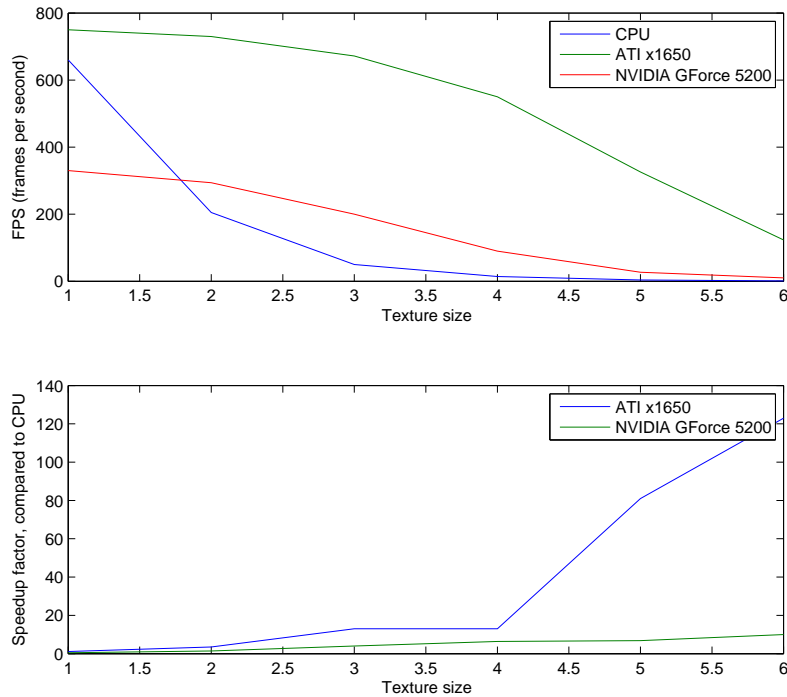


Figure 8.3. GPGPU results. Note that the used textures are coded as follows: 1 = 32×32 , 2 = 64×64 , 3 = 128×128 , 4 = 256×256 , 5 = 512×512 and 6 = 1024×1024 .

8.5 Limitations

The increasing possibilities of GPUs and the resourcefulness of programmers have enabled many applications, outside the original GPU tasks, to be designed. Even though the programming of the GPU can be a bit awkward, the enormous performance increase has tempted many programmers to transform general algorithms to graphic hardware. Even though the performance increase of the GPU can be tremendous, its arithmetic structure is optimized for the handling of graphic primitives, which restricts the area of usage. Hence, in its present form, the GPU is not suitable for many applications.

One of the major limitations of the GPU is its programming environment. This environment has been developed from a low-level, machine language like structure, into 'high-level' languages such as Cg, Brooke and Sh, to name a few. Even though some languages, such as Brooke, hide the graphic oriented, low-level structure better than others it remains necessary to cast non-graphical structures into graphical terms. Besides the programming environment, the maximum precision of calculations remains a problem. Recent graphic cards support a 32-bit floating point precision, but 64-bit double precision remains something to look forward to. Finally, we mention some memory limitations of the GPU. The used memory model allows only read-only operations during computations and write-only operations at the end of a computation. Disk access is not permitted and there is no provision for local memory.

Concluding this section, one can say that the programming of the graphic hardware in the scientific area, is restricted by the extended computer graphic knowledge needed and programming

limitations. But the computational power and growth curve of the GPU make this area very interesting.

CHAPTER 9 CONCLUSION

For the simulations of cardiac arrhythmia we use the well accepted Human Atrium Action Potential model of Courtemanche et al. [9]. Often, cardiac tissue is modelled using the Bidomain equations, introduced in chapter 4. For our experiments, however, we used the Cellular Bidomain Model which has some distinct advantages. The division of the tissue into finite segments, and constructing a simulation graph of these segments, provides a means to simulate cardiac tissue by modelling cellular membrane behavior. In this way, the natural anisotropic and nonhomogeneous nature of cardiac tissue can be modelled by defining individual properties for the segments of the graph. For example, pathological cardiac tissue, such as tissue that is decoupled due to fibrosis, is modelled by edge properties in the graph. Another advantage of the Cellular Bidomain Model is the possibility of modelling complex three dimensional geometric structures. As these geometric structures can play a significant role causing reentry and other arrhythmia, they should be incorporated into a model that simulates cardiac tissue.

As each model is an approximation of reality, accuracy is a very important factor. The model must be accurate enough such that the results from the experiments have real value, but irrelevant details which have no significant meaning must be abstracted from. Atrial fibrillation is a difficult field of study, due to the diversity of processes that play a role. On a macroscopic, phenomenological level, fibrillation can be studied by concentrating on patterns and waves. On a microscopic level, ionic processes influence the macroscopic processes in such a way that one cannot disregard them.

As we are interested in the influence of ionic processes on the forming of AF, we have to take both into account. Hence, our simulations must be accurate enough to sufficiently model the ionic processes on a cellular level, and at the same time provide pattern and wave information in relative short time.

9.1 Simulations

In section 6.2.1 the effect of spatial distance and a correction factor (memcurfactor) on the propagation of the AP are studied. Due to the strong non-linear behavior of depolarization we know that a small spatial step of 0.1 mm is required [8]. As stated, this requirement is mainly imposed to accurately model the depolarization of the cell membrane. Our main interest is the influence of ionic processes on AF and we could allow a less accurate modelling of the depolarization of the membrane as long as our main processes of study are not influenced too much. We know that the speed of the potential wavefront depends on the depolarization rate [5], which influences the wavelength, see chapter 5. Hence, we have to correct for the error made due to less accurate modelling of the depolarization of the membrane. Our experiments show a substantial decrease in propagation velocity if the spatial distance is increased, see Figure 6.1 and 6.3. To compensate this decrease in velocity we use a memcurfactor of 0.3. The result of this correction is a speed that approximates the speed of higher accuracy levels, but induces a small deformation of the propagation wavefront, which is considered negligible.

Before exploring the realm of arrhythmia's and AF one needs to establish a starting point from which a comparison can be made. This starting point is naturally a beating heart under average conditions. In this case the AP wavefront encompasses the atria within 100 ms [22] and the tissue is not subject to abnormal deformations or other pathological conditions. Our experiments show that the atria, in this state, have a large resilience for arrhythmia. The large APD, combined with a speed of 0.9 m/s, results in a large wavelength which causes any perturbation to be cancelled out. This finding corresponds to the relative small number of arrhythmia found in healthy hearts [44].

Our following simulations indicate the key factors contributing to AF. From Chapter 5 we have seen that the APD and the wavefront velocity are crucial factors in AF. To initiate AF, a perturbation of the wavefront is necessary. Our 'Burst pacing with remodelling' simulation shows a

method of creating such a perturbation. Besides the frequency of 10 Hz, other frequencies were tried, but resulted in either a normal AP or premature stimulation, which had no observable influence. Even though we used a remodelling factor $\rho = -1$, the velocity of the AP was large enough to result in a relative large wavelength, cancelling out any existing perturbation. In Chapter 7 we mentioned that experiments report AF after prolonged stimulation. It is known that cardiac cells adapt to this kind of stimulation and that remodelling must be the main factor in these experiments. It is also known, however, that relative short intervals (2 s) of rapid stimulation cause AF [1]. As the interval is too short to induce remodelling, a decrease of the APD due to a small diastolic interval could be a reason. Our experiments, though, fail to reproduce this kind of AF, which might be explained by the relative large wavefront velocity in our experiments.

Next, with the help of a geometric structure, we show that a wavefront with a relatively long APD can reenter. By attaching the wavefront to the Superior Vena Cava the distance travelled is increased which enables the wavefront to reenter. The precise timings and precise geometrical placement of the S1S2S3 protocol required many trial and error experiments and a substantial amount of time to succeed. As there are many geometric obstacles situated in the atria, many anatomical reentrant circuits are possible. Even scenarios in which two reentrant circuits are attached to two different holes, are imaginable. Another geometric aspect, not included in this experiment, is the thickness of the atria. There is evidence that three dimensional propagation is a factor which cannot be ruled out, even though the tissue of the atria is fairly thin [17].

In search of an easier AF initiation, and to induce functional reentry, we had to remodel the I_{Ca} , I_{to} and I_{Na} currents to decrease the APD and decrease the propagation velocity of the wavefront. During this simulation, the initiation of AF was fairly easy and progressed for a large period of time. Even though our effort to initiate AF during this simulation may not be directly related to the natural chance of occurrence, it might be an indication that the chance of AF occurrence increases with the decrease of the APD. During our experiments we noticed that perturbations, caused by external stimuli, wavefront breakage, geometrical obstacles or any other means, were cancelled out if the wavelength was large enough. The moment, however, that this was not the case, the AP wavefront became highly unstable. In agreement with literature [32], we conclude from our experiments that the remodelling of the mentioned ionic currents are a main factor in AF.

From experiments it is known that the cardiac cell adapts during rapid, sustained stimulation [32]. This kind of remodelling, called tachycardia remodelling, is an explanation for sustained AF. What remains is the search for a mechanism for the initiation of AF. Commonly it is believed that diseases, such as fibrosis or heart attacks, play a prominent role. Another factor that may stimulate the occurrence AF could be stretch. Experiments show that nonpenetrating blows to the chest (commotio cordis) may induce fibrillation of many types, which may even cause a sudden death [15]. Other simulation studies indicate that stretch-activated channels may be responsible for the termination of ventricular fibrillation during a precordial thump [29]. Our 'stretch' simulations show that stretch can cause a block of the AP wavefront. If the stretch is acute, and of short duration, the block of the wavefront could be followed by a reentry. In our simulations this reentry does not occur due to the prolonged period of stretch applied to the tissue, which causes an enlarged APD. In the case of commotio cordis, however, the deformation of the heart lasts only for the short duration of the blow and hence reentry could occur.

9.2 Real-time simulations

During our simulations some questions were answered but many were left open. In fact, the number of questions rising to the surface, due to the simulations, surpassed our initial wondering. As each simulation takes a large amount of time, it is not possible to perform a large amount of simulations. Proposing the GPU as an alternative for the CPU to conduct cardiac simulations, we indicated a rising technology that could prove to support scientific research. Even though our

simulation of reaction-diffusion equations on the GPU show a large performance gain compared to the CPU, it is far from a full simulation of a physiological model. Our initial efforts to implement a simple physiological model on a three dimensional mesh were frustrated by the limited memory model of the GPU. Redesigning our algorithms and data structures, however, should solve these problems and future work may support our initial positive findings concerning the GPU.

BIBLIOGRAPHY

-
- [1] H. Akira, C.H.E.M. Chang, S. Zhou, C.C. Chou, J. YI, Y. Miyauchi, Y. Okuyama, M.C. Fishbein, H.S. Karagueuzian, and P. Chen. Induction of atrial fibrillation and nerve sprouting by prolonged left atrial pacing in dogs. *Pacing and Clinical Electrophysiology*, 26(12):2247–2252, 2003.
- [2] M. Allessie, C. Kirchhof, GJ Scheffer, F. Chorro, and J. Brugada. Regional control of atrial fibrillation by rapid pacing in conscious dogs. *Circulation*, 84(4):1689–1697, 1991.
- [3] Leo K Cheng Andrew J Pullan and Martin L Buist. *MATHEMATICALLY MODELLING THE ELECTRICAL ACTIVITY OF THE HEART From Cell to Body Surface and Back Again*. World Scientific Publishing, 2005.
- [4] D. Barkley. A model for fast computer simulation of waves in excitable media. *Physica D*, 49(1-2):61–70, 1991.
- [5] P Bovendeerd. *Cardiac Modeling, lecture notes*. faculteit biomedische technologie, 2006.
- [6] M.A. Bray and JP Wikswo. Use of topological charge to determine filament location and dynamics in a numerical model of scroll wave activity. *Biomedical Engineering, IEEE Transactions on*, 49(10):1086–1093, 2002.
- [7] F.J. Chorro, S. Egea, L. Mainar, J. Canoves, J. Sanchis, E. Llavador, V. Lopez-Merino, and L. Such. Acute changes in wavelength of the process of auricular activation induced by stretching. Experimental study. *Rev Esp Cardiol*, 51(11):874–83, 1998.
- [8] M. Courtemanche. Complex spiral wave dynamics in a spatially distributed ionic model of cardiac electrical activity. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 6:579, 1996.
- [9] M. Courtemanche, R.J. Ramirez, and S. Nattel. Ionic mechanisms underlying human atrial action potential properties: insights from a mathematical model. *American Journal of Physiology-Heart and Circulatory Physiology*, 275(1):301–321, 1998.
- [10] F. Curtis and P.O. Wheatley. *Applied numerical analysis*. Addison-Wesley, 1984.
- [11] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.
- [12] S.C.M. Eijsbouts, R.P.M. Houben, Y. Blaauw, U. Schotten, and M.A. Allessie. Synergistic Action of Atrial Dilatation and Sodium Channel Blockade on Conduction in Rabbit Atria. *Journal of Cardiovascular Electrophysiology*, 15(12):1453–1461, 2004.
- [13] S.C.M. Eijsbouts, M. Majidi, M. van Zandvoort, and M.A. Allessie. Effects of acute atrial dilatation on heterogeneity in conduction in the isolated rabbit heart. *J Cardiovasc Electrophysiol*, 14(3):269, 2003.
- [14] A. Garfinkel, Y.H. Kim, O. Voroshilovsky, Z. Qu, J.R. Kil, M.H. Lee, H.S. Karagueuzian, J.N. Weiss, and P.S. Chen. Preventing ventricular fibrillation by flattening cardiac restitution, 2000.

-
- [15] L.A. Geddes and R.A. Roeder. Evolution of our knowledge of sudden death due to commotio cordis. *American Journal of Emergency Medicine*, 23(1):67–75, 2005.
- [16] D. Goddeke. Gpgpu–basic math tutorial. *Ergebnisberichte des Instituts fur Angewandte Mathematik, Nummer*, 300, 2005.
- [17] R.A. Gray and J. Jalife. Ventricular fibrillation and atrial fibrillation are two different beasts. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 8:65, 1998.
- [18] A.C. Guyton and J.E. Hall. *Medical Physiology. Philadelphia: WB Saunders Co*, 1996.
- [19] A.L. Hodgkin and A.F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Bulletin of Mathematical Biology*, 52(1):25–71, 1990.
- [20] H. Hu and F. Sachs. Stretch-Activated Ion Channels in the Heart. *Journal of Molecular and Cellular Cardiology*, 29(6):1511–1523, 1997.
- [21] J.L. Huang, C.T. Tai, J.T. Chen, C.T. Ting, Y.T. Chen, M.S. Chang, and S.A. Chen. Effect of atrial dilatation on electrophysiologic properties and inducibility of atrial fibrillation. *Basic Research in Cardiology*, 98(1):16–24, 2003.
- [22] V. Jacquemet. *A biophysical model of atrial fibrillation and electrograms: formulation, validation and applications*. PhD thesis, University of Lausanne, 2004.
- [23] J. Keener and J. Sneyd. *Mathematical Physiology*. Springer, 1998.
- [24] P. Kohl. Mechanoelectric feedback in cardiac cells. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 359(1783):1173–1185, 2001.
- [25] N.H.L. Kuijpers, R.H. Keldermann, T. Arts, and P.A.J. Hilbers. Computer simulations of successful defibrillation in decoupled and non-uniform cardiac tissue. *Europace*, 7(s2):S166, 2005.
- [26] N.H.L. Kuijpers, R.H. Keldermann, H.M.M. Ten Eikelder, T. Arts, and P.A.J. Hilbers. The role of the hyperpolarization-activated inward current i_f in arrhythmogenesis: A computer model study. *Biomedical Engineering, IEEE Transactions on*, 53(8):1499–1511, 2006.
- [27] N.H.L. Kuijpers, H.M.M. Ten Eikelder, P.H.M. Bovendeerd, S. Verheule, T. Arts, and P.A.J. Hilbers. Mechano-electric feedback leads to conduction slowing and block in acutely dilated atria: a modeling study of cardiac electromechanics. *Am J Physiol Heart Circ Physiol*, 2007.
- [28] N.H.L. Kuijpers, H.M.M. Ten Eikelder, P.H.M. Bovendeerd, S. Verheule, T. Arts, and P.A.J. Hilbers. Remodeling of l-type ca^{2+} current leads to homogeneous fiber shortening: a modeling study of cardiac electromechanics and adaptation. 2007.
- [29] W. Li, P. Kohl, and N. Trayanova. Myocardial ischemia lowers precordial thump efficacy: An inquiry into mechanisms using three-dimensional simulations. *Heart Rhythm*, 3(2):179–186, 2006.

-
- [30] C.H. Luo and Y. Rudy. A model of the ventricular cardiac action potential. Depolarization, repolarization, and their interaction. *Circulation Research*, 68(6):1501–1526, 1991.
- [31] S.J. Melby, A. Zierer, S.P. Kaiser, R.B. Schuessler, and R.J. Damiano. Epicardial microwave ablation on the beating heart for atrial fibrillation: The dependency of lesion depth on cardiac output. *The Journal of Thoracic and Cardiovascular Surgery*, 132(2):355–360, 2006.
- [32] S. Nattel. Atrial Fibrillation: Experimental and Theoretical Developments. *Cardiac Electrophysiology Review*, 5(2):263–267, 2001.
- [33] J.D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Kruger, A.E. Lefohn, and T.J. Purcell. A survey of general-purpose computation on graphics hardware. *Eurographics 2005, State of the Art Reports*, pages 21–51, 2005.
- [34] A.V. Panfilov. Three-dimensional organization of electrical turbulence in the heart. *Physical Review E*, 59(6):6251–6254, 1999.
- [35] A.N. Plotnikov, H. Yu, J.C. Geller, R.Z. Gainullin, P. Chandra, K.W. Patberg, S. Friezema, P. Danilo, I.S. Cohen, S.J. Feinmark, et al. Role of L-Type Calcium Channels in Pacing-Induced Short-Term and Long-Term Cardiac Memory in Canine Heart, 2003.
- [36] R.J. Ramirez, S. Nattel, and M. Courtemanche. Mathematical analysis of canine atrial action potentials: rate, regional factors, and electrical remodeling. *American Journal of Physiology-Heart and Circulatory Physiology*, 279(4):1767–1785, 2000.
- [37] R.J. Ramirez, S. Nattel, and M. Courtemanche. Mathematical analysis of canine atrial action potentials: rate, regional factors, and electrical remodeling. *American Journal of Physiology-Heart and Circulatory Physiology*, 279(4):1767–1785, 2000.
- [38] F. Ravelli and M. Allessie. Effects of Atrial Dilatation on Refractory Period and Vulnerability to Atrial Fibrillation in the Isolated Langendorff-Perfused Rabbit Heart. *Circulation*, 96(5):1686–1695, 1997.
- [39] D. Shreiner, M. Woo, J. Neider, and T. Davis. *Opengl programming guide: The official guide to learning opengl (red book)*.
- [40] M.S. Spach and J.P. Boineau. Microfibrosis produces electrical load variations due to loss of side-to-side cell connections: a major mechanism of structural heart disease arrhythmias. *Pacing Clin Electrophysiol*, 20(2 Pt 2):397–413, 1997.
- [41] J. Sundnes. *Computing the Electrical Activity in the Heart*. Springer, 2006.
- [42] P.M. van Dam and A. van Oosterom. Atrial excitation assuming uniform propagation. *J Cardiovasc Electrophysiol*, 14(10 Suppl):S166–71, 2003.
- [43] P.M. van Dam and A. van Oosterom. Volume conductor effects involved in the genesis of the P wave. *Europace*, 7(s2):S21–29, 2005.

-
- [44] I.C. Van Gelder and M.E.W. Hemels. The progressive nature of atrial fibrillation: a rationale for early restoration and maintenance of sinus rhythm. *Europace*, 8(11):943, 2006.
- [45] F. Xie, Z. Qu, A. Garfinkel, and J.N. Weiss. Electrical refractory period restitution and spiral wave reentry in simulated cardiac tissue. *American Journal of Physiology- Heart and Circulatory Physiology*, 283(1):448–460, 2002.
- [46] M. Zabel, BS Koller, F. Sachs, and MR Franz. Stretch-induced voltage changes in the isolated beating heart: importance of the timing of stretch and implications for stretch-activated ion channels. *Cardiovasc Res*, 32(1):120–30, 1996.

APPENDIX

APPENDIX A: REACTION-DIFFUSION GPGPU

```

//This program demonstrates an implementation of a reaction diffusion equation
//on the GPU

#include <stdio.h>
#include <assert.h>
#include <stdlib.h>
#include <GL/glut.h>
#include <Cg/cgGL.h>
#include <stdio.h>
#include <cmath>
#include <iostream>

/* Defines */
#define rows 256
#define columns 256
#define windowWidth 1024
#define windowHeight 1024

#define GLEW_STATIC 1

#ifndef max
#define max(a,b)      (((a) > (b)) ? (a) : (b))
#define min(a,b)      (((a) < (b)) ? (a) : (b))
#endif

// forward declarations
class Barkley;
// globals
CGcontext  g_cgContext;
CGprofile  g_cgProfile;
Barkley *barkley;

// This shader performs u update
static const char *uFragSource =
"double4 u_update(double2 coords : TEX0, \n"
"                uniform sampler2D texture_u, uniform sampler2D texture_v):COLOR  \n"
"{                                                                 \n"
"    static const double offset = 1.0 / 256.0;                    \n"
"    static const double dt = 0.01;                               \n"
"    static const double one_o_e = 200;                           \n"
"    static const double b_var = 0.01;                            \n"
"    static const double a_var = 0.78;                            \n"
"    double tot ;                                                 \n"
"    double delta = 0.0001;                                       \n"
"    double4 v = tex2D(texture_v, coords);                          \n"
"    double4 c = tex2D(texture_u, coords);                          \n"
"    double4 l = tex2D(texture_u, coords + double2(-offset,      0)); \n"
"    double4 t = tex2D(texture_u, coords + double2(      0,  offset)); \n"
"    double4 r = tex2D(texture_u, coords + double2( offset,      0)); \n"

```

```

"    double4 b = tex2D(texture_u, coords + double2(      0, -offset));           \n"
"    double test = c.x;                                                         \n"
"    tot = c + 2*c*(1.0-c)*(c - (v+ b_var)/a_var)+ 0.04*(r+l+t+b-4*c);         \n"
"    if (tot > 0.95) tot = 0.95;                                               \n"
"    if(test < delta) tot = 0.04*(r+l+t+b-4*c);                               \n"
"                                                                              \n"
"    return tot ;                                                               \n"
"}                                                                              \n";

// This shader performs v update
static const char *vFragSource =
"double4 v_update(double2 coords : TEX0,                                       \n"
"    uniform sampler2D texture_u, uniform sampler2D texture_v) : COLOR        \n"
"{                                                                              \n"
"    static const float dt = 0.01;                                           \n"
"    double tot ;                                                             \n"
"    double delta = 0.0001;                                                  \n"
"    double4 u = tex2D(texture_u, coords);                                    \n"
"    double4 v = tex2D(texture_v, coords);                                    \n"
"    double test = u.x;                                                       \n"
"    tot = (v + dt*(u-v));                                                    \n"
"    if (test < delta) tot = (1-50*dt)*v;                                     \n"
"    return tot;                                                              \n"
"}                                                                              \n";

// This class encapsulates all of the GPGPU functionality.
class Barkley
{
public: // methods
    Barkley(int w, int h)
        : _iWidth(w),
          _iHeight(h), time(0)
    {
        // Create the textures for the u, v1, v2 variables.
        // For now we use a v2 variable to hold the old value of v.
        // After the updates of v1 and u, v1 is copied to v2. This can
        // be optimized by passing v1 and v2 as a parameter to the v update shader
        // and alternating the update of v1 and v2. So one pass v1 contains the old value of
        // and the next pass v2 contains the old value of v.

// Initialize the values for u and v variable
valuesMake();

        // Initialize the textures for u, v1 and v2
        glGenTextures(1, &_iTexture_u);
        glBindTexture(GL_TEXTURE_2D, _iTexture_u);
        glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
        glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
        glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
        glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
        glTexImage2D(GL_TEXTURE_2D, 0, 3, _iWidth, _iHeight,

```

```

        0, GL_RGB, GL_FLOAT, u_value);

glGenTextures(1, &_iTexture_v1);
    glBindTexture(GL_TEXTURE_2D, _iTexture_v1);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
    glTexImage2D(GL_TEXTURE_2D, 0, 3, _iWidth, _iHeight,
        0, GL_RGB, GL_FLOAT, v_value);

glGenTextures(1, &_iTexture_v2);
    glBindTexture(GL_TEXTURE_2D, _iTexture_v2);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
    glTexImage2D(GL_TEXTURE_2D, 0, 3, _iWidth, _iHeight,
        0, GL_RGB, GL_FLOAT, v_value);

    // Create the u update fragment program
    _fragmentProgram_u = cgCreateProgram(g_cgContext, CG_SOURCE,
        uFragSource, g_cgProfile,
        "u_update", NULL);

// Create the v update fragment program
    _fragmentProgram_v = cgCreateProgram(g_cgContext, CG_SOURCE,
        vFragSource, g_cgProfile,
        "v_update", NULL);

}

~Barkley()
{
    cgDestroyProgram(_fragmentProgram_u);
    cgDestroyProgram(_fragmentProgram_v);
}

// This method updates the textures to be drawn for each draw-loop.
void update()
{
    // store the window viewport dimensions so we can reset them,
    // and set the viewport to the dimensions of our texture
    int vp[4];
    glGetIntegerv(GL_VIEWPORT, vp);

// Set the correct viewport for a one-to-one mapping of texture to screen
    glViewport(0, 0, _iWidth, _iHeight);

// ***** UPDATE V1 *****

```

```

// load the shader for v
if(_fragmentProgram_v != NULL)
{
    cgGLLoadProgram(_fragmentProgram_v);
}

// setup the input parameters for the shader
_textureParam_u = cgGetNamedParameter(_fragmentProgram_v, "texture_u");
_textureParam_v = cgGetNamedParameter(_fragmentProgram_v, "texture_v");

glBindTexture(GL_TEXTURE_2D, _iTexture_u);
glBindTexture(GL_TEXTURE_2D, _iTexture_v1);

    cgGLBindProgram(_fragmentProgram_v);
    cgGLEnableProfile(g_cgProfile);

    // bind the u and v1 texture as input parameters to the filter
    cgGLSetTextureParameter(_textureParam_u, _iTexture_u);
    cgGLEnableTextureParameter(_textureParam_u);

cgGLSetTextureParameter(_textureParam_v, _iTexture_v1);
    cgGLEnableTextureParameter(_textureParam_v);

    // In order to calculate the next values of v1, we draw the scene to the framebuffer.
    glBegin(GL_QUADS);
    {
        glTexCoord2f(0, 0); glVertex3f(-1, -1, -0.5f);
        glTexCoord2f(1, 0); glVertex3f( 1, -1, -0.5f);
        glTexCoord2f(1, 1); glVertex3f( 1,  1, -0.5f);
        glTexCoord2f(0, 1); glVertex3f(-1,  1, -0.5f);
    }
    glEnd();

    // disable the filter
    cgGLDisableTextureParameter(_textureParam_u);
    cgGLDisableTextureParameter(_textureParam_v);
    cgGLDisableProfile(g_cgProfile);

    // Copy the next values of v1 to the texture.
    glBindTexture(GL_TEXTURE_2D, _iTexture_v1);
    glCopyTexSubImage2D(GL_TEXTURE_2D, 0, 0, 0, 0, 0, 0, _iWidth, _iHeight);

// ***** UPDATE U *****

// load the shader for u
    if(_fragmentProgram_u != NULL)
    {
        cgGLLoadProgram(_fragmentProgram_u);
    }
}

```

```

// setup the input parameters for the shader
_textureParam_u = cgGetNamedParameter(_fragmentProgram_u, "texture_u");
_textureParam_v = cgGetNamedParameter(_fragmentProgram_u, "texture_v");

glBindTexture(GL_TEXTURE_2D, _iTexture_u);
glBindTexture(GL_TEXTURE_2D, _iTexture_v2);

    cgGLBindProgram(_fragmentProgram_u);
    cgGLEnableProfile(g_cgProfile);

    // bind the u and v2 texture as input parameters to the filter
    cgGLSetTextureParameter(_textureParam_u, _iTexture_u);
    cgGLEnableTextureParameter(_textureParam_u);

cgGLSetTextureParameter(_textureParam_v, _iTexture_v2);
    cgGLEnableTextureParameter(_textureParam_v);

    // In order to calculate the next values of u, we draw the scene to the buffer.
    glBegin(GL_QUADS);
    {
        glTexCoord2f(0, 0); glVertex3f(-1, -1, -0.5f);
        glTexCoord2f(1, 0); glVertex3f( 1, -1, -0.5f);
        glTexCoord2f(1, 1); glVertex3f( 1,  1, -0.5f);
        glTexCoord2f(0, 1); glVertex3f(-1,  1, -0.5f);
    }
    glEnd();

    // disable the filter
    cgGLDisableTextureParameter(_textureParam_u);
cgGLDisableTextureParameter(_textureParam_v);
    cgGLDisableProfile(g_cgProfile);

    // Copy the new values of u to the texture.
    // update the texture again, this time with the filtered scene
    glBindTexture(GL_TEXTURE_2D, _iTexture_u);
    glCopyTexSubImage2D(GL_TEXTURE_2D, 0, 0, 0, 0, 0, 0, _iWidth, _iHeight);

// ***** COPY V1 TO V2 *****
// The textures are copied by drawing v1 to the frame buffer and reading the
// buffer values into the texture for v2.

//glClear(GL_COLOR_BUFFER_BIT);
    glBindTexture(GL_TEXTURE_2D, _iTexture_v1);
glEnable(GL_TEXTURE_2D);

    glBegin(GL_QUADS);
    {
        glTexCoord2f(0, 0); glVertex3f(-1, -1, -0.5f);
        glTexCoord2f(1, 0); glVertex3f( 1, -1, -0.5f);
        glTexCoord2f(1, 1); glVertex3f( 1,  1, -0.5f);
    }

```

```

        glTexCoord2f(0, 1); glVertex3f(-1, 1, -0.5f);
    }
    glEnd();

// copy buffer to v2
glBindTexture(GL_TEXTURE_2D, _iTexture_v2);
glCopyTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, 0, 0, _iWidth, _iHeight, 0);

// restore the stored viewport dimensions
glViewport(vp[0], vp[1], vp[2], vp[3]);
}

void display()
{
    // Bind the filtered texture
    glBindTexture(GL_TEXTURE_2D, _iTexture_u);
    glEnable(GL_TEXTURE_2D);

    // render a full-screen textured quad.
    glBegin(GL_QUADS);
    {
        glTexCoord2f(0, 0); glVertex3f(-1, -1, -0.5f);
        glTexCoord2f(1, 0); glVertex3f( 1, -1, -0.5f);
        glTexCoord2f(1, 1); glVertex3f( 1,  1, -0.5f);
        glTexCoord2f(0, 1); glVertex3f(-1,  1, -0.5f);
    }
    glEnd();

    glDisable(GL_TEXTURE_2D);

    // update time
    time++;
    if (time == 100000) time = 0;
}

// initialize data
void valuesMake(void)
{
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++)
        {
            u_value[i][j][0] = (GLfloat) 0;
            u_value[i][j][1] = (GLfloat) 0;
            u_value[i][j][2] = (GLfloat) 0;

            v_value[i][j][0] = (GLfloat) 0.0;
            v_value[i][j][1] = (GLfloat) 0.0;
            v_value[i][j][2] = (GLfloat) 0.0;
        }
    }
}

```

```

}

for (int i = 0; i < rows; i++)
{
    //set initial stim node (0,0) to 0.15
    u_value[i][100][0] = (GLfloat) 0.15;
    u_value[i][100][1] = (GLfloat) 0.15;
    u_value[i][100][2] = (GLfloat) 0.15;
}
}

// apply an external stimulus to the value u
void stimS2(int x, int y, int b)
{
    glBindTexture(GL_TEXTURE_2D, _iTexture_u);
    glGetTexImage(GL_TEXTURE_2D, 0, GL_RGB, GL_FLOAT, u_value);

    // stimulus initiated by time
    if(x == -1)
    {
        for (int i = 0; i < (int)rows/2; i++)
        {
            //set initial stim node (0,0) to 0.15
            u_value[i][(int)columns/2][0] = (GLfloat) 0.15;
            u_value[i][(int)columns/2][1] = (GLfloat) 0.15;
            u_value[i][(int)columns/2][2] = (GLfloat) 0.15;
        }

        glBindTexture(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
        glBindTexture(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
        glBindTexture(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
        glBindTexture(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
        glBindTexture(GL_TEXTURE_2D, 0, 3, _iWidth, _iHeight,
        0, GL_RGB, GL_FLOAT, u_value);
    }

    // stimulus initiated by mouse. b indicates the button: left, middle or right button.
    if(b == 0)
    {
        u_value[rows-y][x][0] = (GLfloat) 0.15;
        u_value[rows-y][x][1] = (GLfloat) 0.15;
        u_value[rows-y][x][2] = (GLfloat) 0.15;

        glBindTexture(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
        glBindTexture(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
        glBindTexture(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
        glBindTexture(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
        glBindTexture(GL_TEXTURE_2D, 0, 3, _iWidth, _iHeight,
        0, GL_RGB, GL_FLOAT, u_value);
    }
}

```

```

}
if(b == 1)
{

for (int i = max(0,rows-(y+(int)rows/10)); i < min(rows,rows-(y-(int)rows/10)); i++)
{
//set initial stim node (0,0) to 0.15
u_value[i][x][0] = (GLfloat) 0.15;
u_value[i][x][1] = (GLfloat) 0.15;
u_value[i][x][2] = (GLfloat) 0.15;
}

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
glTexImage2D(GL_TEXTURE_2D, 0, 3, _iWidth, _iHeight,
0, GL_RGB, GL_FLOAT, u_value);

}

if(b == 2)
{
for (int i = 0; i < rows; i++)
{
for (int j = 0; j < columns; j++)
{
//set initial stim node (0,0) to 0.15
u_value[i][j][0] = (GLfloat) 0.15;
u_value[i][j][1] = (GLfloat) 0.15;
u_value[i][j][2] = (GLfloat) 0.15;
}
}

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
glTexImage2D(GL_TEXTURE_2D, 0, 3, _iWidth, _iHeight,
0, GL_RGB, GL_FLOAT, u_value);

}
}

protected: // data
    int _iWidth, _iHeight; // The dimensions of our array

// data
GLfloat u_value[rows][columns][3];
GLfloat v_value[rows][columns][3];

```

```

    unsigned int  _iTexture_u;          // The textures used as a data array
unsigned int  _iTexture_v1;
unsigned int  _iTexture_v2;

    CGprogram    _fragmentProgram_u; // the fragment program used to update u
CGprogram    _fragmentProgram_v; // the fragment program used to update v
    CGparameter  _textureParam_u;    // parameters to the fragment program
CGparameter  _textureParam_v;

int time; // Time is implemented as a counter to initiate a stimulation
};

// GLUT idle function
void idle()
{
    glutPostRedisplay();
}

// GLUT display function
void display()
{
    barkley->update(); // update the scene and run the edge detect filter
    barkley->display(); // display the results
    glutSwapBuffers();
}

// GLUT reshape function
void reshape(int w, int h)
{
    if (h == 0) h = 1;

    glViewport(0, 0, w, h);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-1, 1, -1, 1);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}

// Called when Cg detects an error
void cgErrorCallback()
{
    CGError lastError = cgGetError();

    if(lastError)
    {
        printf("%s\n\n", cgGetErrorString(lastError));
        printf("%s\n", cgGetLastListing(g_cgContext));
        printf("Cg error!\n");
    }
}

```

```

    }
}

// Called at startup
void initialize()
{
    // Setup Cg
    cgSetErrorCallback(cgErrorCallback);
    g_cgContext = cgCreateContext();

    // get the best profile for this hardware
    g_cgProfile = cgGLGetLatestProfile(CG_GL_FRAGMENT);
    assert(g_cgProfile != CG_PROFILE_UNKNOWN);
    cgGLSetOptimalOptions(g_cgProfile);

    // Create the example object
    barkley = new Barkley(rows, columns);
}

void processNormalKeys(unsigned char key, int x, int y)
{
    if (key == 27)
        exit(0);
}

void processMouse(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON)
    {
        barkley->stimS2((int)(x*columns/windowWidth), (int)(y*rows/windowHeight),0);
    }
    if (button == GLUT_RIGHT_BUTTON)
    {
        barkley->stimS2((int)(x*columns/windowWidth), (int)(y*rows/windowHeight),1);
    }
    if (button == GLUT_MIDDLE_BUTTON)
    {
        barkley->stimS2((int)(x*columns/windowWidth), (int)(y*rows/windowHeight),2);
    }
}

// The main function
int main(int argc, char **argv)
{
    // init glut
    glutInit( &argc, argv );
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(windowWidth, windowHeight);
    glutCreateWindow("Hello, GPGPU!");
}

```

```
// define event functions
    glutIdleFunc(idle);
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
glutKeyboardFunc(processNormalKeys);
glutMouseFunc(processMouse);

    initialize();

// main loop, calls the "display" method
    glutMainLoop();

    return 0;
}

%\end{lstlisting}
```