FINAL REPORT

Simulating action potentials in cardiac cells in order to determine the link between atrial fibrillation and ectopic foci in the pulmonary vein

Girish Gupta
School of Physics and Astronomy
University of Manchester

Supervised by Henggui Zhang and Oleg Aslanidi.
Carried out in collaboration with Lucy Foster.

September 2009 – May 2010

The formation of spiral waves, and so atrial fibrillation, has been shown to occur by the interaction of stimuli and ectopic beats in the pulmonary vein in a computer model. Models were made of cardiac cells and their action potentials to achieve this. These then formed the basis for two- and then three-dimensional models incorporating a cylindrical pulmonary vein. The models were stimulated from various points and the geometry modified to more closely resemble actual anatomy. The 3D model was used to simulate several excitation wave patterns within the atria: (1) propagation of normal periodic action potentials onto the pulmonary vein; (2) emergence of an ectopic wave source due to spontaneous action potential firing within the pulmonary vein; (3) interaction of re-entrant spiral waves with the pulmonary vein. Our 3D atrial tissue model provides mechanistic insights into the role of the PV in the genesis of AF, primarily conditions for the initiation of ectopic action potentials breaking down the normal periodic waves, and the development of a fibrillatory state.

Contents

Introduction

A computer model of the heart would aid clinicians hugely. Rather than resorting to invasive surgery which would cause the patient much trauma and unnecessary risk, a model would allow clinicians to test their ideas (to a limit of course) with no physical harm caused. As well as surgical techniques, the modelled heart would be allowed to become addled with drugs so clinicians could test the heart's and body's response to them, again with no danger to the patient.

On a prophylactic level, computer models allow scientists to determine the causes of disease in order to combat them more efficiently. It is this branch of the wider project that this report aims to undertake in determining the link between atrial fibrillation and ectopic foci in and around the pulmonary vein.

Our work began as an extension of work carried out by Aslanidi et. al. (2008[i]). Their work concentrated on clarifying and confirming their models of single cardiac cells and their action potentials. Our work attempts to extrapolate this to two-dimensional tissues and then three-dimensional models that show a pulmonary vein entering the left atrium and its cardiac tissue.

Our aim was to create a geometrically-accurate framework with which to model various stimuli across the atria and pulmonary vein in order to discover the link between atrial fibrillation and ectopic foci in and around the pulmonary vein. We also hoped to visualise atrial fibrillation without forcing it artificially, rather allowing a more natural sequence to occur in reaching it.

Anatomy and Physiology of the Heart

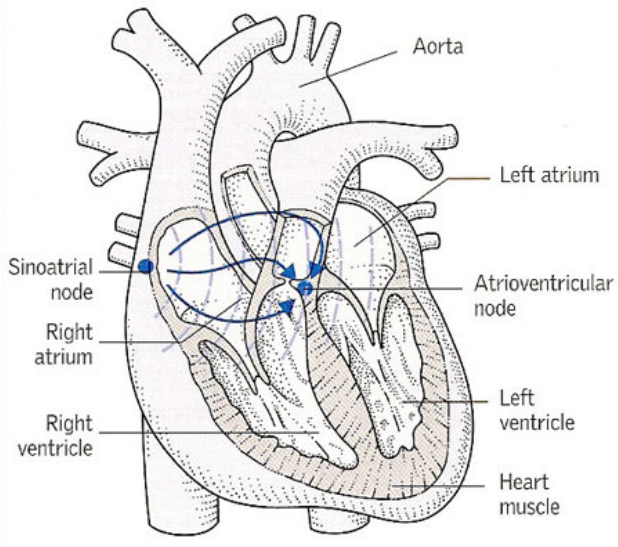

Figure 1 – The human heart[ii].

The electrophysiology that leads to the heart's pacemaking is primarily initiated at the sinoatrial node, as seen in Figure 1. The sinoatrial node's cells form the 'slow response' action potential which can be seen as the uppermost action potential in Figure 3. The 'fast response' action potentials are detailed below the sinoatrial node's in Figure 3. The different action potentials will be discussed later.

The electrocardiogram (ECG), while not relevant to this project directly, will aid in the explanation of the processes involved in a typical heartbeat. An ECG readout can be found at the bottom of Figure 3. It is essentially a scalar average of the potentials created by the heart's electrical activity that are found at the body's surface. Understanding the causes of each part of it allows clinicians to diagnose problems when faced with an atypical ECG.

The sinoatrial node will begin the process with its 'slow' action potential. The impulse will pass through the internodal tracts and Bachmann's bundle (BB, an interatrial tract) to depolarise first the right and then the left atrium. As can be seen by the progression in Figure 3, the atrial muscle then begins to contract. It is this that causes the P wave of the ECG. A lack of P wave indicates a lack of atrial contraction and so is an indicator of atrial fibrillation[iv, p. 150].

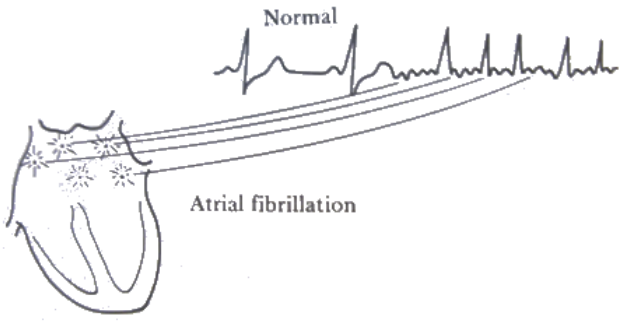

Figure 2 – Atrial fibrillation is indicated on an ECG readout by a lack of P waves[iii].

The potential will then arrive at the atrioventricular node before continuing into the bundle of His and onto the Purjinke network to depolarise the ventricular cells. This causes the ventricles to contract to pump blood through the aorta and pulmonary artery to the rest of the body. This is indicated by the QRS complex of the ECG. An extension in the P-Q or P-R interval, therefore, indicates that the potential is taking too long in transit from the atria to ventricles.

The ECG's T wave is produced by ventricular repolarisation. Atrial repolarisation is masked by the QRS complex[iv, pp. 141, 146]

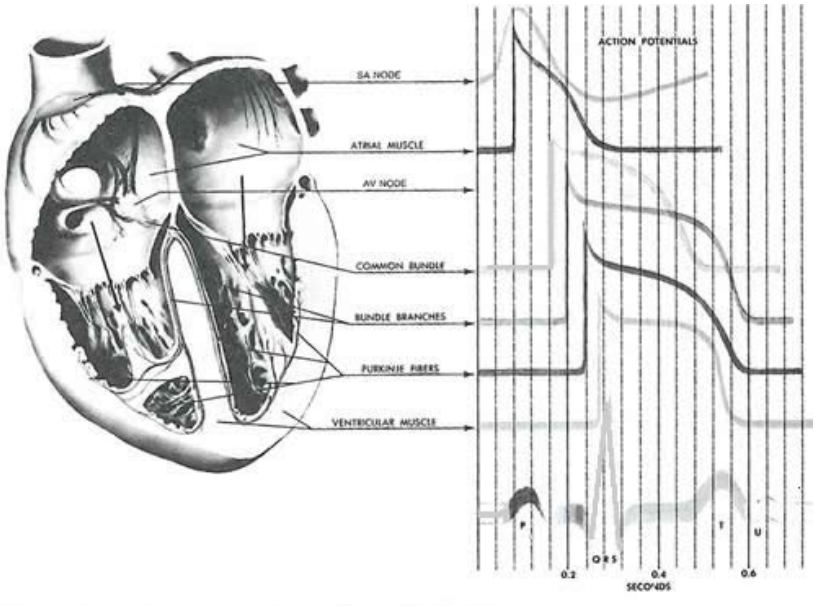

Figure 3 – The human heart and its action potentials, as well as an electrocardiogram readout at the bottom which enables us to ascertain the stage within the heart's cycle that each action potential is 'fired'[iv, p. 143].

Each has a slightly different shape of action potential which shall be described further along.

Cell Type	Acronym
Left atrium	LA
Right atrium	RA
Pulmonary vein	PV
Coronary sinus	CS
Bachmann's bundle	BB
Cristae terminalis	CT
Pectinate muscle	PM

Table 1 – The different cardiac cell types and their acronyms used throughout this report.

The Cell Membrane

The cell membrane is a phospholipid bilayer that separates the contents of a cell from its exterior. It is across this phospholipid bilayer that action potentials will form due to the variability in permeability of the membrane, to be discussed in *Action Potentials*. This variability is down to electrical and concentration gradients as well as the presence of ion channels and active-transport pumps.

Ion channels are large protein molecules that span the cell membrane. They are specific to ions and allow these through, down their concentration gradient. The channels' permeability to their specific ion is controlled by 'gates'; these open and close when prompted by either

the presence of a potential or chemical, i.e., whether their state is open or closed to specific ions will be controlled by either the presence of a potential greater than some threshold or the presence of a chemical stimulus[v].

Sodium-potassium and calcium pumps are also important in the permeability of the membrane. Unlike ion channels which allow an essentially passive processes down a concentration gradient, these pumps actively transport ions against their concentration gradient in order to maintain the resting potential.

For example, in a single procedure, the sodium-potassium pump uses one molecule of adenosine triphosphate (ATP) to pump three sodium ions out of the cell and two potassium ions in[vi, p. 161].

Despite their huge underlying importance, it is not necessary to go into the specifics of the mechanics of the ion channels and pumps. Suffice to say that they play a large role in controlling the permeability of the cell membrane and therefore a large role in the formation and conduction of action potentials through the heart. More in depth work on ion channels formed the basis of the previous work of Aslanidi et. al. that we are extending[i].

Being a means of separating charge, the cell membrane can be modelled as a capacitor circuit, as shown in Figure 4.

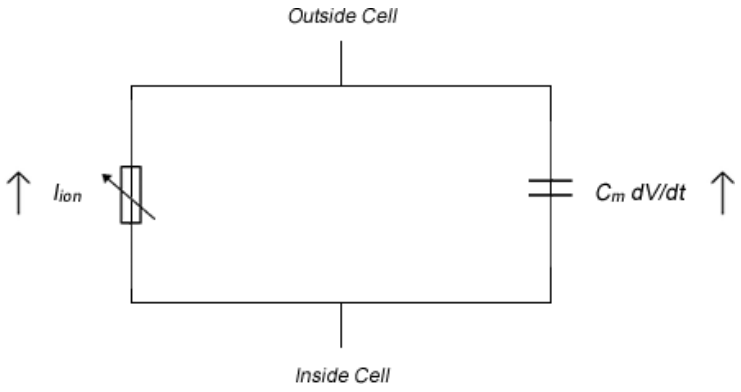

Figure 4 – Electrical circuit model of the cell membrane. Adapted from Keener & Sneyd (1998[x, p. 57]). The Hodgkin-Huxley model is an extension of this bringing in all ion types.

Assuming this capacitive nature of the cell membrane, it is possible to derive an equation that governs the potential across a membrane in terms of the ions involved and the capacitance (as opposed to the Nernst Equation and its derivative, Equation 2, which deals with ionic concentration). It is this equation, Equation 1, that is used as the basis for all of our models.

$$C_m \frac{dV}{dt} + I_{ion} = C_m(\nabla . \mathbf{D})(\nabla \mathbf{V})$$

Equation 1 – The membrane potential is governed by this equation. C_m is the membrane capacitance; V is the voltage across it; I_{ion} is the sum of the ions detailed in Table 3; D is the diffusion coefficient[x, p. 56].

The differing concentrations of various ions inside and outside the cell given in Table 2 cause a potential to exist across the membrane. As described in *Phases of the Action Potential*, the diffusional and electric forces acting on the ions will balance. The potential across the membrane at this steady state is given by Equation 2.

$$E = \frac{RT}{F} \ln \frac{P_K[K]_o + P_{Na}[Na]_o + P_{Cl}[Cl]_i}{P_K[K]_i + P_{Na}[Na]_i + P_{Cl}[Cl]_o}$$

Equation 2 – The equation giving the potential difference across a membrane with concentration $[X]_o$ outside the cell and $[X]_i$ inside the cell. P_x is the permeability of the membrane to the particular ion x. R is the molar gas constant; T is the absolute temperate; F is the Faraday constant.

Equation 2 is an extension of the Nernst equation by Goldman (1943) and later modified by Hodgkin and Katz (1949)[iv, p. 124]. It gives the potential across a membrane as a function of the permeability of the membrane to particular ions as well as the concentrations of those ions.

Table 2 lists the intra- and extra-cellular concentrations of the most important ions in forming the action potential. It is these figures, when coupled with the permeability of the membrane to them, that are used in Equation 2 to give the resting potential.

Ionic Species	Extracellular concentration (mmol)	Intracellular concentration (mmol)
Na^+	145	20
K^+	4	150
Cl^-	2.5	0.0001

Table 2 – The concentrations of the ionic species both inside and outside the cell. Variations in these concentrations give rise to action potentials[vii].

For reference, the ions involved in our models and calculations are detailed in Table 3. As our work is not directly using these for the two- and three-dimensional modelling, they are not mentioned frequently in this report. They are, however, very important in the basis of our work and that of our predecessors.

Total ionic current	$\boldsymbol{I_{ion}}$
Transient outward potassium current	I_{to}
Long lasting calcium current	I_{CaL}
Transient calcium current	I_{CaT}
Sodium current	I_{Na}
Inward rectifier potassium current	I_{K1}
Delayed rectifier potassium current (rapid)	I_{Kr}
Delayed rectifier potassium current (slow)	I_{Ks}

Table 3 – The different ionic currents modelled. Their potentials are summed to give the action potential[viii].

Action Potentials

There are two types of action potential found in the heart: the fast and slow responses[ix, p. 185]. The slow response occurs solely in the sinoatrial and atrioventricular nodes. These are essentially the pacemakers of the heart, however, the action potentials we shall be concentrating on are the fast responses shown in Figure 5. These exist in the majority of cardiac cells as shown later in the interim report.

In the early half of the last century, much work was conducted to measure and later quantify the electric signal propagation along giant squid axons (giant for the size of the axon, *not* the squid). Alan Hodgkin and Andrew Huxley developed the first quantitative model of that action potential in 1952, and soon realised its significance in further papers when the word 'squid' was excluded from their titles. It was their work that brought about Equation 1[x, p. 117], [xi]. Beeler and Reuter continued the work and applied it to ventricular cells in 1977[x, p. 149].

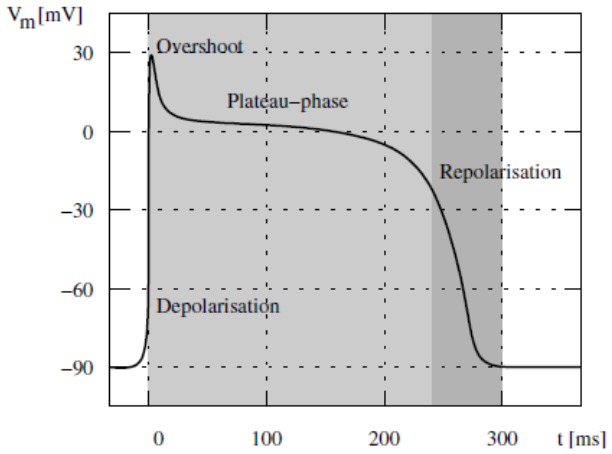

Figure 5 – A typical cardiac action potential with its resting phase at -90 mV, rapid depolarisation, overshoot, plateau phase and repolarisation[vi, p. 159].

The models demonstrate a typical cardiac action potential, as shown in Figure 5. This shows the potential difference across the membrane of an excitable cell for the period of an action potential. The action potential is caused by ionic currents through the cell membrane, the carriers of which are allowed through by the aforementioned ion channels. The summation of the potentials created by the different ionic currents, shown in Table 3, is the shape shown in Figure 5.

The ions involved are primarily sodium, potassium and calcium. To generate the shape in Figure 5, the membrane permeability varies as a function of time, as do the transmembrane concentration and electrical potential differences.

Following the initiation of the action potential, the cardiac cell will experience a refractory period during which it is unable to initiate another action potential with the ease of the first. This is down to inactivation of the Na^+ channels. The refractory period is split into an absolute period and a relative one. During the absolute refractory period, an action potential cannot be initiated no matter how high the stimulus. This lasts for a short period before the onset of the relative refractory period during which a much higher stimulus than the usual threshold is required to initiate an action potential[iv, p. 126].

This refractory period is important in conduction of stimuli. It prevents the electrical impulse going backwards along a pathway as the state (open or closed) of the previous ion channel will not be altered by the potential in and around the current ion channel until the refractory period is over, by which time the stimulus will have moved on.

Phases of the Action Potential[xii]

The resting potential is governed primarily by potassium and sodium ion concentrations. Two opposing forces lead to an equilibrium voltage difference caused by the differing potassium ion concentrations. One is the chemical force, based on the concentration gradient; this leads to an outward current (where current is conventionally positive and so this implies an outward K^+ flow, for example). The other is the electrostatic force which leads to an inward current as there is a significant concentration of negative ions left residing within the cell; this leads to an inward current.

On inserting the measured values of these concentrations in Equation 2 and taking into account the other ions flowing through the membrane, the value for the resting potential is roughly -90mV.

The following should be read with reference to Figure 5.

On the application of a stimulus to a threshold level of roughly -65mV, an action potential is produced. This is begun with a rapid depolarisation of the potential. This is due almost solely to an influx of sodium ions through the cell membrane; this is the I_{Na} of Table 3. Many of the drugs used to treat some types of cardiac arrhythmias work by blocking the 'fast' Na^+ channels that allow this depolarisation.

There is a small notch—labelled 'overshoot' in Figure 5—showing the beginnings of repolarisation. This is caused by a cessation of I_{Na} and the activation of the transient outward potassium current, I_{to}. Liu et. al. (1993[xiii]) demonstrated the link between the size of the notch to both the type of cardiac cell in question as well as the basic cycle length (BCL). The greater the BCL, the larger the notch and the further inside the heart the cell, the smaller the notch, i.e., endocardial cells have smaller repolarisation phases. This is demonstrated in Figure 6 and will be used later to confirm the accuracy our own work.

A plateau then occurs. There is a dual effect from the influx of Ca^{2+} and the efflux of K^+, described by I_{CaL} and I_{Ks} respectively. When the efflux of potassium exceeds the influx of calcium, the cell finally begins to repolarise back to its resting potential. The transient outward potassium current (I_{to}) and delayed rectifier potassium (I_{Kr} and I_{Ks}) currents help initiate this phase.

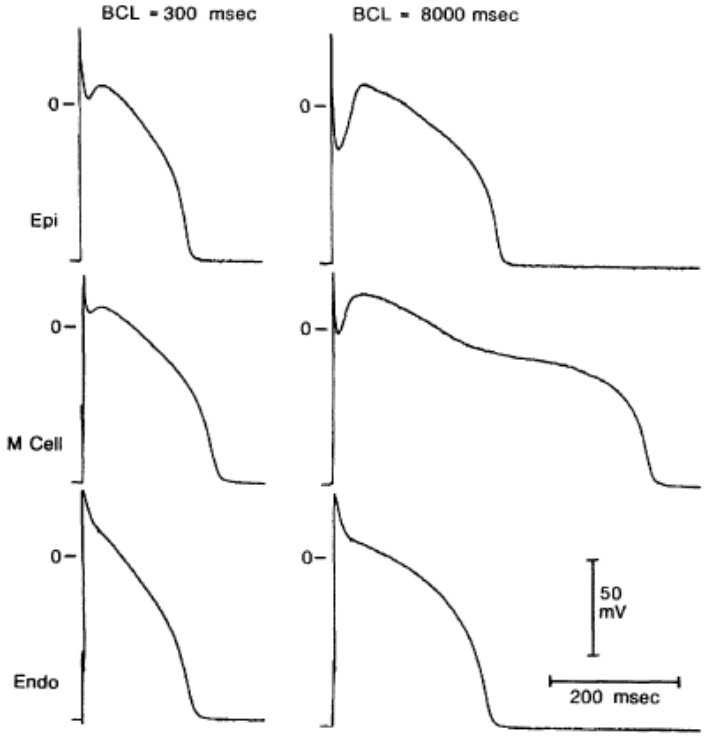

Figure 6 – Liu et. al. (1993[xiii]) showed experimentally that as BCL increased, the size of the notch showing the beginnings of repolarisation (left vs. right of figure). They also showed that the notch decreased in size as action potentials 'progressed' into the heart, i.e., epicardiac cells have a larger notch than endocardial cells (bottom to top of figure). (M stands for midmyocardial.)

Atrial Fibrillation

Atrial fibrillation is the most common form of cardiac arrhythmia, though is poorly understood[xiv,xv,xvi]. Chard and Tabrizchi (2009[xiv]) wrote of the link between the pulmonary vein and atrial fibrillation physiology following the discovery of the phenomenon by Haissaguerre et. al. just over a decade earlier[xvi].

Haissaguerre et. al. studied 45 patients with atrial fibrillation and found that 94 per cent of ectopic foci were located in the pulmonary vein (varying between the left and right superior and inferior). Their conclusion was that the pulmonary vein is "an important source of ectopic beats [which would lead to] atrial fibrillation"[xvi]. The results are shown graphically in Figure 7.

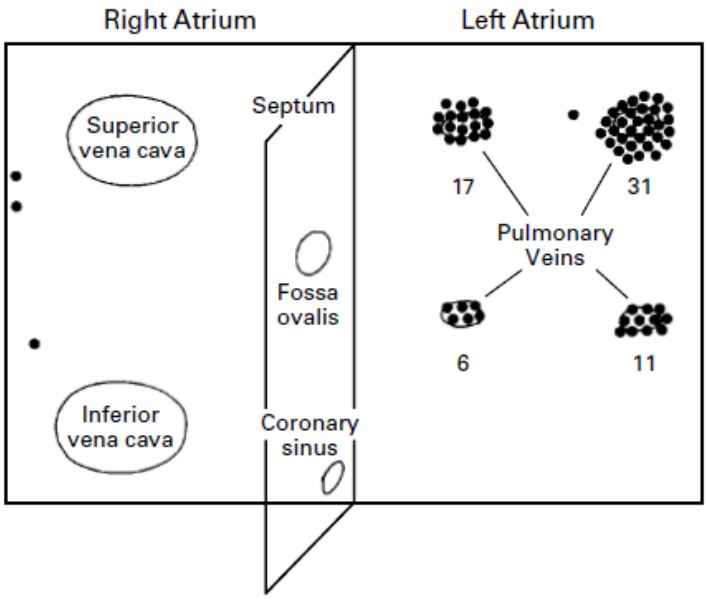

Figure 7 – "Diagram of the sites of 69 foci triggering atrial fibrillation in 45 patients. Note the clustering in the pulmonary veins, particularly in both superior pulmonary veins. Numbers indicate the distribution of foci in the pulmonary veins."—Haissaguerre et. al., 1998[xvi].

Their idea was that pacemaking cells within the pulmonary vein can produce electrical signals which interfere with those propagating from the sinoatrial node. This interference leads to so-called 're-entry'.

Re-entry is a re-excitation of an area through which an electrical impulse has already passed. There are two types of re-entrant circuits described by Berne and Levy (2000[ix, pp. 195-196]): When the circuit is fixed in one region of the heart, the loop can lead to tachycardia—raised heartbeat. When the loop, however, consists of "multiple, highly irregular continuously moving re-entrant circuits," this can lead to atrial (or ventricular) fibrillation.

Spiral waves form within the area experiencing fibrillation. These do not require a repeating source to exist, as they are typically self-sustained. This is a reason that they are generally only found in pathophysiological situations, i.e., abnormal and unhealthy situations[x, p. 305].

Atrial fibrillation can be present in a patient for many years with no serious side-effects barring an occasional shortness of breath. However, due to its irregular nature, patients are susceptible to the formation of blood clots in or around the re-entry loops. If these clots make their way through the carotid artery up to the brain, this could lead to stroke, a rapid loss of brain function due to a lack of blood to the brain which could cause permanent neurological damage and death. Ventricular fibrillation, incidentally, cannot be left for a sustained period without treatment as its presence will lead to a cessation of blood pumping around the body. Defibrillation would then be immediately essential.

Methods

Computer Modelling

Computer modelling was based upon a single-cell model by Aslanidi, et. al. (2008[i]). Written in *C*, the model outputted datasets demonstrating the different action potentials of the various cardiac cell types. The code for one of these, the pulmonary vein, is replicated in full at the back of this report (*PV_Cell.h*).

The other cell types have similar code bar certain parameters, so it is not necessary to copy all code to this report. Another file that is shown is *Atria_Variables.h* which is a collection of variables called upon by the main file, as its name suggests.

The most important file is *3D_model.c* which contains the main function and much of the code that was constantly updated and tweaked during the course of this project. It was in this file that the stimuli were set and timed to coincide with self-oscillations from the pulmonary vein.

The three dimensional model required a matrix, created by our code, which indicated the cell type in a particular region. This geometry can be seen in Figure 8. The code simply creates the large array of numbers corresponding to our anatomy (*3Dgeo.c*). The matrix shows the locations of the difference types of cell indicated by the action potentials shown in the interim report. This geometry allowed for the cells in the cardiac tissue as in our two-dimensional model to be arranged as a surface out of which the pulmonary vein emanated.

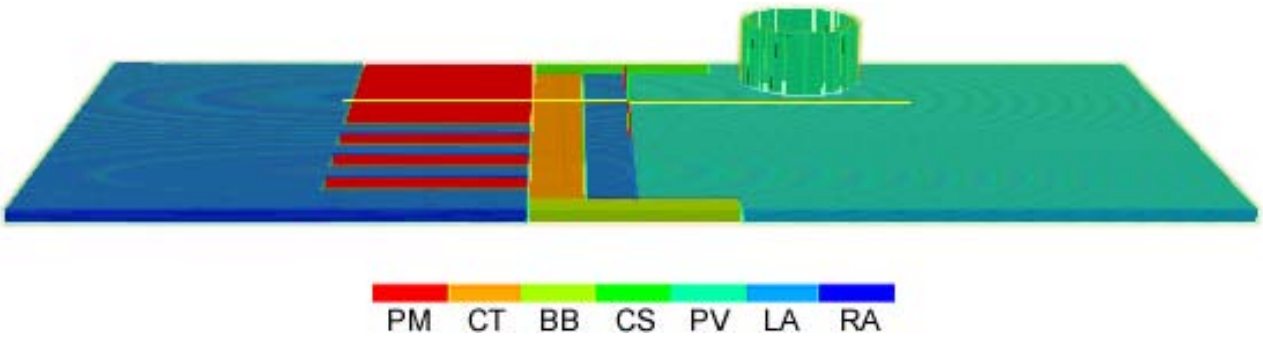

Figure 8 – The geometry of our three-dimensional model. The key shows the cell type indicated by the associated colour; refer to Table 1 for explanation of the acronyms. The cells' action potentials can be seen in the interim report.

Note: The pulmonary vein appears broken. This is due solely to the effects of scaling the geometry file down to speed up visualisation for creation of this image. The actual visualisations of the code do not contain a broken pulmonary vein.

Self Oscillation and the Ectopic Focus

Following the interim report which confirmed that the 3D model was able to handle and show spiral waves by placing a resting phase into the atria to induce them, the next task was to have a more natural progression towards atrial fibrillation. This would involve stimulating the sinoatrial node correctly (i.e., not from the base of the atria but at a smaller area shown in Figure 9) but more importantly would require the self-oscillations that were seen in our preliminary work. These self-oscillations in the pulmonary vein are induced by initial stimulations across the cardiac tissue.

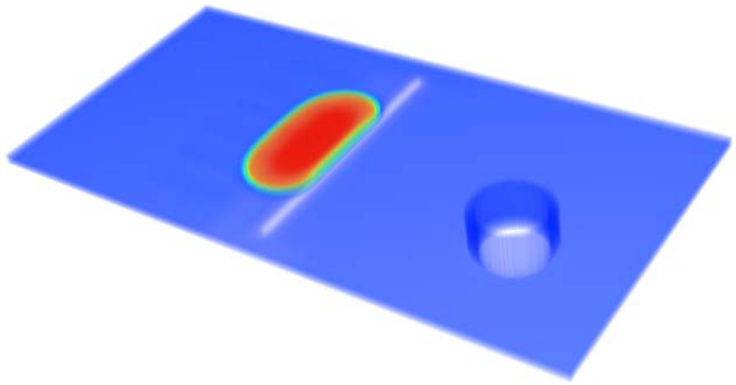

Figure 9 – The later geometry where the sinoatrial node stimulation was not at the base of the atria but at a more anatomically accurate position.

When our own stimulations, sited at the sinoatrial node, interact with these self-oscillations—without the resting phase introduced to force spiral waves—we know that we have created a much more natural model of atrial fibrillation.

Breakdown of method

Creating a more accurate geometry was the first task for the second phase of this project. Thanks to our earlier work in creating a geometry file (*3Dgeo.c*), this was not too difficult a task as all we had to do was tweak it. The pulmonary vein was given a smaller diameter, its walls slightly thinned and, more importantly, the sinoatrial stimulus was repositioned.

Figure 10 (from *3D_model.c*) gives the geometry to be stimulated—the top of the pulmonary vein—at the beginning of the programme, and stimulates it.

```
for (x=100; x<=200; x++)
 for (y=325; y<=350; y++)
  for (z=0; z<=10; z++)
   if (h[x][y][z] > 0)
   {
       V[x][y][z] = 20.0;
       new_V[x][y][z] = 20.0;
   }
```

Figure 10 – This code gives the area to be stimulated and stimulates it.

Originally, a resting phase was introduced to induce spiral waves without a natural progression. This is shown in Figure 14 and explained above it. The code used to do this is shown in Figure 11, part of *3D_model.c*. It was commented out for the most important part of the second phase of this report, however, is here for completeness and to demonstrate its necessity in earlier work.

```
if (t>=0.04 && t<(0.04+HT))
      for (x = BLOCK_LOW(rank, size, X+1); x <= BLOCK_HIGH(rank, size, X+1); x++)
      {
    for (y=0; y<=200; y++)
              for (z=0; z<=10; z++)
       {
                 V[x][y][z] = -73.8;
                 new_V[x][y][z] = -73.8;

              }
        for (y=400; y<=Y; y++)
              for (z=0; z<=10; z++)
       {
                 V[x][y][z] = -73.8;
                 new_V[x][y][z] = -73.8;
              }
       }
```

Figure 11 – This code was commented out for the most important part of this project, however, did demonstrate beforehand that spiral waves could be shown in the programme, a vital step.

Further work was carried out on the model described in the interim report. This was stimulated from the base of the atria as well as from the top of the pulmonary vein. Initially these were timed to coincide and the forcing of spiral waves cut, as in Figure 11. This was repeated with the repositioning of the sinoatrial node stimulus, as was all work in this report's methodology.

Our next task was to work on self-oscillation. Self-oscillation had been achieved in the initial 2D models, though was not apparent in our current work. Much time was spent working on

the reasons for this when eventually it was decided that we would re-index our original 2D code—which we confirmed did self-oscillate—to a 3D version. Another important factor was the running-time of our programme. This was initially three hours. Over the course of the project, this was extended first to six and then nine hours. The self-oscillation did occur but very late on in the visualisation, so the longer it ran, the more true and relevant a picture we saw.

Once an ectopic beat was found to occur, our stimulus—be it at the base of the atria or at the more realistic sinoatrial node—was repeated to coincide with it in an attempt to force an interaction between the two propagating waves that led to spiral waves and so atrial fibrillation. This is much more realistic to the heart's function. In the below code sample (from *3D_model.c*), the ectopic beat is found 900 ms (the `0.90` in the code) into the running of the programme. Figure 12 gives the co-ordinates of the geometry to stimulate and does so.

```
if (t>=0.90 && t<(0.90+HT))
{
        for (x=100; x<=200; x++)
         for (y=325; y<=350; y++)
          for (z=0; z<=10; z++)
           if (h[x][y][z] > 0)
           {
                V[x][y][z] = 20.0;
                new_V[x][y][z] = 20.0;
           }
}
```

Figure 12 – The code stimulates the upper pulmonary vein at 900 ms.

Another area to investigate was repeated stimuli, as in a heartbeat. These were easy to code by modifying the above simply to repeat. The aim here was to discover whether there was an intermediate time period at which one stimulus could not follow another. This would indicate whether the pulmonary vein as a structure was an obstacle to propagation of stimuli.

Results

Results were given in the interim report of the 1D work, the action potentials produced, confirmed and then applied to a 2D programme. This was in turn used as a basis for a 3D model.

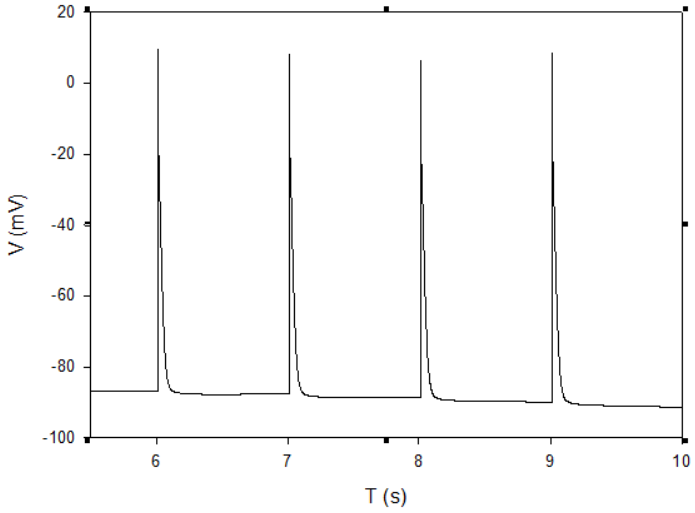

Figure 13 – Self-oscillating in the pulmonary vein, in looking at a single cell.

Figure 13 shows self-oscillation in a single cell. It was this which was so important to achieving our aim and whose code would be re-indexed to three dimensions.

Three-Dimensional Model

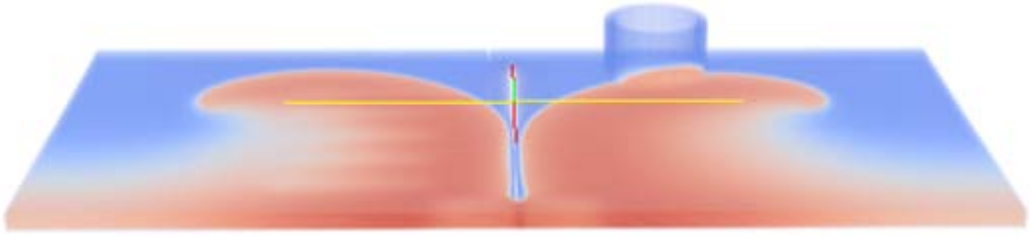

Figure 14 – Previous work showed spiral waves that had been forced by causing certain areas of the geometry to relax at certain points in time. The aim of further work was to create the same spiral waves but more naturally.

Figure 14 shows the conclusion of the work discussed in the interim report. It shows a stimulus from the base of the atria that has propagated through the cardiac tissue and then been forced into spiral waves by resting at a particular point in its progress in both atria (by the code in Figure 11. As stated, the next task was to create spiral waves without this forced resting.

Figure 15 shows the same stimulus from the base of the atria propagating through the cardiac tissue towards the pulmonary vein without this forced resting. Figure 16 then shows the self-oscillation that is so important to our work. Creating another stimulus identical to that shown in Figure 15 at the appropriate time to interact with the ectopic stimulation in Figure 16 gives the spiral waves associated with atrial fibrillation.

This important result is seen in Figure 17.

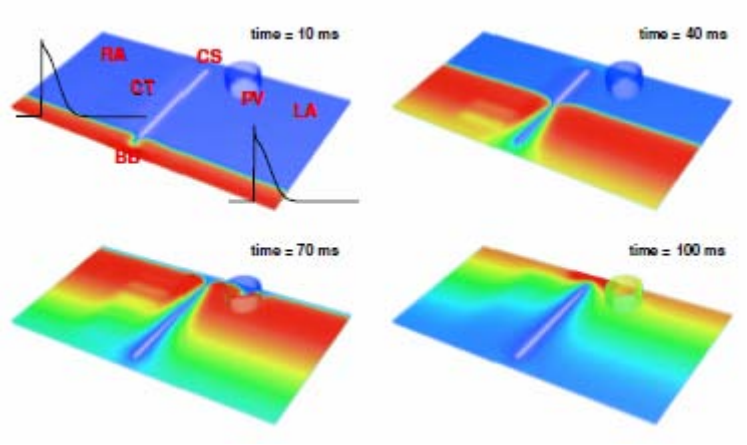

Figure 15 – Initiation and propagation of action potentials through the atria towards the pulmonary vein.

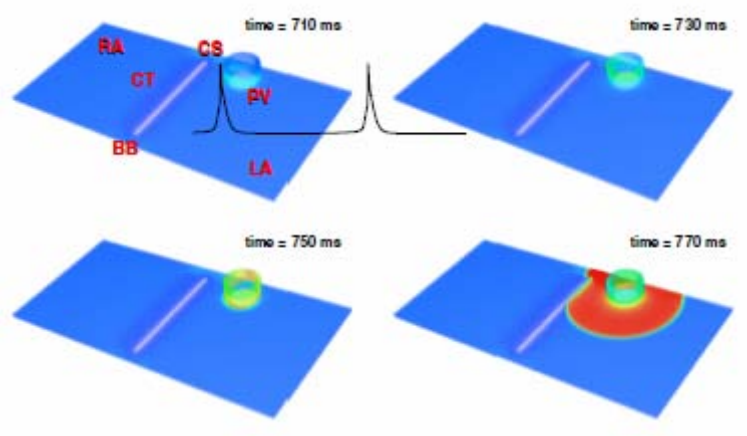

Figure 16 – Initiation and propagation of an ectopic action potential from the pulmonary vein. This is the self-oscillation.

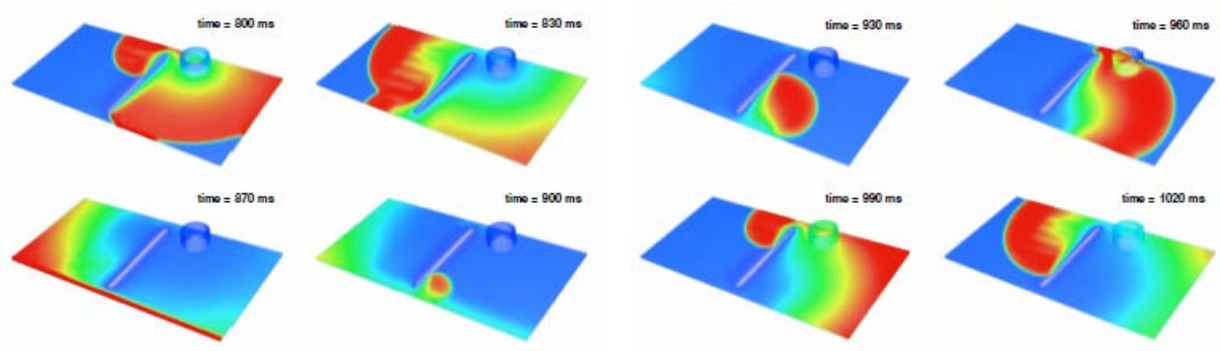

Figure 17 – Interaction of the ectopic stimulation from Figure 16 and the initial stimulus from Figure 15 to give a spiral wave.

The apex of the spiral wave normally moves in a random trajectory, be it a circle, straight line or more complicated figure. When it runs into an obstacle, it is possible for the trajectory to pin to it or not.

Cardiac ablation is a medical technique whereby a catheter is guided to the point where an ectopic stimulus is thought to be produced. Energy is then applied to the area in the hope that it will cull it, much like defibrillation. When the catheter is guided to the pulmonary vein in cases of atrial fibrillation, it is found to be helpful implying some sort of pinning[xvii].

Figure 18 shows no evidence of pinning as the trajectory of the spiral waves is not in any way linked to the obstacle of the pulmonary vein.

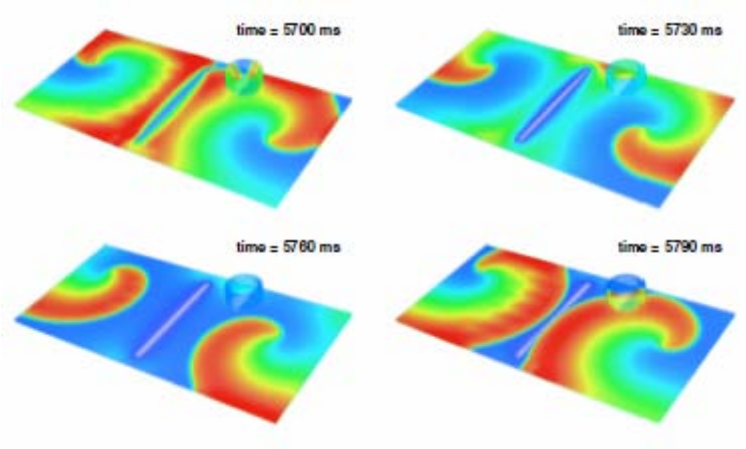

Figure 18 – Our research has not found evidence that the spiral wave is pinned to the pulmonary vein.

Figure 19 shows that a normal heart beat suppresses ectopic beats in the pulmonary vein. This simply demonstrates that those with a normal heartbeat will not suffer atrial fibrillation. If there is not a regular stimulus, i.e. regular heartbeat, an ectopic stimulus is formed; this corresponds to the previous figures.

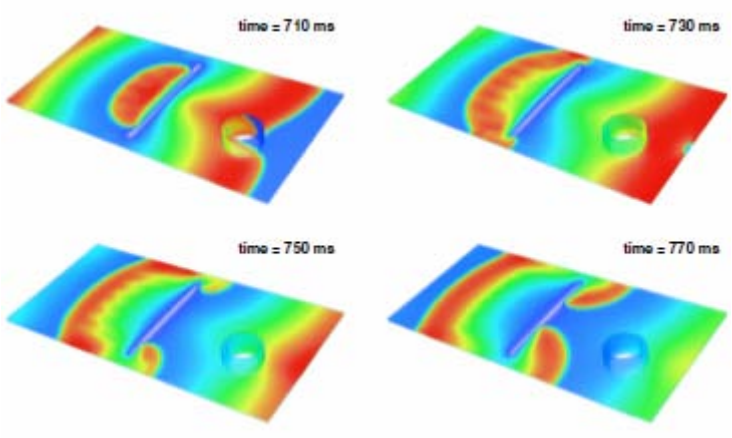

Figure 19 – Fast, periodic pacing, suppresses ectopic foci in the pulmonary vein.

Discussion

The project began with work on the different cell types given in Table 1, ensuring that their action potentials were both accurate and able to be worked into a multi-dimensional model. Also, it was necessary to determine whether self-oscillation would take place specifically in pulmonary vein cells.

The next task was to essentially extrapolate this to a 2D model. The full method behind this is discussed in our interim report, as are its failings. The model, despite only being two-dimensional, was abstract and complex in its construction, requiring intricate boundary conditions between the cardiac atrial tissue and the cylindrical pulmonary vein which had been flattened to maintain two-dimensionality.

Unfortunately, too long was spent on this model before embarking on the 3D model that has been used since. This model utilised a geometry file which was relatively easy to code. This then was visualised with no boundary conditions necessary, as propagation was as automatic as it had been in simpler models.

Our work since the interim report has been productive and certainly achieved the aims of the project. We began this phase with a 3D model which did show spiral waves and so atrial fibrillation, however, they were forced rather than occurring in a more natural sense. This was never meant to demonstrate a link between ectopic foci and initial stimuli. However, it did show that spiral waves were capable of being shown in our visualisations.

The most important task was to confirm that ectopic foci were forming following a stimulation. This proved to take longer than expected, partly due to the length of time in running our programme (nine hours at the end). We eventually decided to return to the 2D

model which we knew and could show to self-oscillate, and re-index it to three dimensions. In hindsight, had this been done earlier, much time would have been saved.

When this was done, ectopic foci were found to occur in the pulmonary vein and so we were simply able to time our own stimulus, that of the sinoatrial node—now moved from the base of the atria to the more anatomically correct position—to coincide with the ectopic beat. This would interact to form spiral waves and so achieve our aim of demonstrating the cause of atrial fibrillation, as shown in Figure 17.

Conclusion

The aims of our interim report were:

- Refine the geometry file to become more anatomically accurate.
- Stimulate the anatomically correct points.
- Self-excitation in the pulmonary vein itself.
- This and further work will establish the link between ectopic foci in the pulmonary vein and atrial fibrillation, as per our aim.

I am happy that we have fully achieved the aims set out both for this entire project and then by our own interim report.

However, further work needs to be done before this becomes useful to the medical sciences. Ideally, the model needs to be made as anatomically accurate as possible. This is a simple task in theory, however, as we discovered when upgrading from two to three dimensions, it will no doubt be incredibly tricky.

Not only geometrical parameters, but parameters in terms of the voltages applied at stimuli, their strength and exact area need to be carefully implemented into the code. These may then demonstrate pinning, something we have failed to establish.

However, I am very pleased in that our aims have been achieved and our work has been an important stepping stone in what is a vital project.

Appendix

Here I include three of the main files used in this project. *3D_model.c* contains the main function and links into the separate header files of which I have included two: *Atria_Variables.h*, which contains and defines many of the variables used, and *PV_Cell.h*, the code relevant to the pulmonary vein. This is similar to the code for all the different cell types.

Also included is *3Dgeo.c* which created the matrix for the geometry file.

3D_model.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <mpi.h>
#include "LA_cell.h"
#include "RA_cell.h"
#include "PM_cell.h"
#include "BB_cell.h"
#include "CS_cell.h"
#include "PV_cell.h"
#include "CT_cell.h"
#include "Atria_Variables.h"

#define BLOCK_LOW(id,p,n) ((id)*(n)/(p))
#define BLOCK_HIGH(id,p,n) (BLOCK_LOW((id+1),p,n)-1)
#define BLOCK_SIZE(id,p,n) (BLOCK_LOW((id+1),p,n)-BLOCK_LOW(id,p,n))


int main(int argc, char **argv)
{
   float new_V[X+1][Y+1][Z+1];
   float global_V[X+1][Y+1][Z+1]; // added this line 17th dec as variable wasn't defined
   float dvdt[X+1][Y+1][Z+1];

   float dvdx2, dvdy2, dvdz2;
   float t=0.0;

   int x, y, z;
   int cnt = 0;
   int cnt2 = 0;
   int dd;

   int girishcounter = 1; //renamed from n as n is something else

   int isbound;
   int i, j, k;
   int num = 0;
   int gg[27];

   float root2 = sqrt(2.0);
   float root3 = sqrt(3.0);
   float ic, ir, il, imax;
   float tflt;

   FILE *in, *out;
   char *str;

   RTONF=Temp*0.08554;

   ECl=RTONF*log(30./132.);
   EbCl=ECl-0.49*(ECl+30.59);

   Vi=0.0126;
   VCa=0.005884;
   Vc=0.0025;
   Vup=0.0003969;
   Vrel=0.000044;
```

```
 int rank = 0;
 int size = 1;

 int *recv_cnts, *recv_disp ;
 MPI_Status status ;

 MPI_Init(&argc, &argv) ;
 MPI_Comm_size(MPI_COMM_WORLD, &size);
 MPI_Comm_rank(MPI_COMM_WORLD, &rank);

 recv_cnts = calloc (size, sizeof(int)) ;
 recv_disp = calloc (size, sizeof(int)) ;

 for (int i = 0 ; i < size ; i++) {
   recv_cnts[i] = BLOCK_SIZE(i, size, X+1)*(Y+1)*(Z+1);
   recv_disp[i] = BLOCK_LOW(i, size, X+1)*(Y+1)*(Z+1);
 }

if (!rank) {

  in = fopen ("geometry.c", "r");
      for (z = 0; z <= Z; z++) {
    for (y = 0; y <= Y; y++) {
              for (x = 0; x <= X; x++) {
                fscanf(in, "%d ", &dd);
        h[x][y][z] = dd;
        if (dd > 0)
                     g[x][y][z] = 1;
              else
          g[x][y][z] = 0;
              }
      fscanf(in, "\n");
    }
    fscanf(in, "\n");
  }
  fclose (in);

  num = 1;
  for (x = 1; x < X; x++)
    for (y = 1; y < Y; y++)
      for (z = 1; z < Z; z++)
        if (g[x][y][z] > 0) {
          g[x][y][z] = num;
          num++;
               }

     for (x = 1; x < X; x++)
   for (y = 1; y < Y; y++)
    for (z = 1; z < Z; z++) {
      gg[1] = g[x - 1][y - 1][z - 1];
      gg[2] = g[x - 1][y - 1][z];
      gg[3] = g[x - 1][y - 1][z+1];
      gg[4] = g[x - 1][y][z - 1];
      gg[5] = g[x - 1][y][z];
      gg[6] = g[x - 1][y][z + 1];
      gg[7] = g[x - 1][y + 1][z - 1];
      gg[8] = g[x - 1][y + 1][z];
      gg[9] = g[x - 1][y + 1][z + 1];

      gg[10] = g[x][y - 1][z - 1];
      gg[11] = g[x][y - 1][z];
      gg[12] = g[x][y - 1][z + 1];
      gg[13] = g[x][y][z - 1];
      gg[14] = g[x][y][z + 1];
      gg[15] = g[x][y + 1][z - 1];
      gg[16] = g[x][y + 1][z];
      gg[17] = g[x][y + 1][z + 1];

      gg[18] = g[x + 1][y - 1][z - 1];
      gg[19] = g[x + 1][y - 1][z];
      gg[20] = g[x + 1][y - 1][z + 1];
      gg[21] = g[x + 1][y][z - 1];
      gg[22] = g[x + 1][y][z];
      gg[23] = g[x + 1][y][z + 1];
      gg[24] = g[x + 1][y + 1][z - 1];
```

```
      gg[25] = g[x + 1][y + 1][z];
      gg[26] = g[x + 1][y + 1][z + 1];

      isbound = 0;
      for(i = 1; i <= 26; i++) {
        if (gg[i] > 0) {gg[i] = 1; isbound++;}
        else gg[i] = 0;
      }

      if (g[x][y][z] == 0 && isbound > 0) {
        ic = (gg[3]/root3) - (gg[1]/root3) + (gg[6]/root2) +
             (gg[9]/root3) - (gg[7]/root3) - (gg[4]/root2) +
             (gg[12]/root2) - (gg[10]/root2) + gg[14] +
             (gg[17]/root2) - (gg[15]/root2) - gg[13] +
             (gg[20]/root3) - (gg[18]/root3) + (gg[23]/root2) +
             (gg[26]/root3) - (gg[24]/root3) - (gg[21]/root2);

        ir = (gg[9]/root3) - (gg[2]/root2) - (gg[3]/root3) -
             (gg[1]/root3) + (gg[8]/root2) + (gg[7]/root3) +
             (gg[17]/root2) - gg[11] - (gg[12]/root2) -
             (gg[10]/root2) + gg[16] + (gg[15]/root2) +
             (gg[26]/root3) - (gg[19]/root2) - (gg[20]/root3) -
             (gg[18]/root3) + (gg[25]/root2) + (gg[24]/root3);

        il = (gg[18]/root3) + (gg[19]/root2) + (gg[20]/root3) +
             (gg[21]/root2) + gg[22] + (gg[23]/root2) +
             (gg[24]/root3) + (gg[25]/root2) + (gg[26]/root3) -
             (gg[1]/root3) - (gg[2]/root2) - (gg[3]/root3) -
             (gg[4]/root2) - gg[5] - (gg[6]/root2) -
             (gg[7]/root3) - (gg[8]/root2) - (gg[9]/root3);

        imax = fabs(ic);
        if (fabs(ir) > imax) imax = fabs(ir);
        if (fabs(il) > imax) imax = fabs(il);

        i = 0; j = 0; k = 0;

        tflt = ir / fabs(imax);
        if (tflt <= 0.5 && tflt >= -0.5) i = 0;
        else if (tflt > 0.5) i = 1;
        else if (tflt < -0.5) i = -1;

        tflt = ic / fabs(imax);
        if (tflt <= 0.5 && tflt >= -0.5) j = 0;
        else if (tflt > 0.5) j = 1;
        else if (tflt < -0.5) j = -1;

        tflt = il / fabs(imax);
        if (tflt <= 0.5 && tflt >= -0.5) k = 0;
        else if (tflt > 0.5) k = 1;
        else if (tflt < -0.5) k = -1;

        if (imax == 0) { i = 0; j = 0; k = 0; }

        if (g[x + k][y + i][z + j] > 0)
          g[x][y][z] = -1 * g[x + k][y + i][z + j];
        else
          g[x][y][z] = g[x + k][y + i][z + j];
       }
    }
}

MPI_Bcast (g, (X+1)*(Y+1)*(Z+1), MPI_INT, 0, MPI_COMM_WORLD);
MPI_Bcast (h, (X+1)*(Y+1)*(Z+1), MPI_INT, 0, MPI_COMM_WORLD);

// ** Initialise all variables at all co-ordinates ** //
 for (x=0; x<=X; x++)
   for (y=0; y<=Y; y++)
        for (z=0; z<=Z; z++) {
     V[x][y][z]=-100; // changed to -100 so that we can isolate it in paraview
        new_V[x][y][z]=-73.8;
     Pa[x][y][z]=0.000028;
     Pi[x][y][z]=0.933953;
     n[x][y][z]=0.00925;
     r1[x][y][z]=0.0016;
     s1[x][y][z]=0.5753;
     s2[x][y][z]=0.5300;
```

```
       s3[x][y][z]=0.5844;
       m[x][y][z]=0.007788;
       h1[x][y][z]=0.87277;
       h2[x][y][z]=0.84549;
       dL[x][y][z]=0.00016;
       fL1[x][y][z]=0.9981;
       fL2[x][y][z]=0.9981;
       y2[x][y][z]=0.0927;
       fca[x][y][z]=0.7755;
       dT[x][y][z]=0.00046;
       fT[x][y][z]=0.30752;
       F1[x][y][z]=0.288039;
       F2[x][y][z]=0.002262;
       F3[x][y][z]=0.612697;
       Cai[x][y][z]=0.000073;
       Caup[x][y][z]=0.730866;
       Carel[x][y][z]=0.726776;
       Kc[x][y][z]=5.0;
       Ki[x][y][z]=140.0;
       Nai[x][y][z]=8.4;
           fac[x][y][z]=0.029108;
       faTc[x][y][z]=0.014071;
       faTmgc[x][y][z]=0.214036;
       faTmgm[x][y][z]=0.693565;
       faCalse[x][y][z]=0.465921;
      }

// ** End Initialisation of variables ** //

// ** Set the stimulus sites ** //
//this is the line in the geometry-creating C file that relates to CT cells: if((y>=325 && y<=
350) && (x>=25 && x<=275))
   for (x=100; x<=200; x++)
         for (y=325; y<=350; y++)
           for (z=0; z<=10; z++)
                if (h[x][y][z] > 0) {
           V[x][y][z] = 20.0;
                  new_V[x][y][z] = 20.0;
                }

// ** Begin voltage calculation ** //

   while (t < 50.0) {
        t+=HT;
     cnt++;

 if (t>=0.90 && t<(0.90+HT))
        {
   for (x=100; x<=200; x++)
         for (y=325; y<=350; y++)
           for (z=0; z<=10; z++)
                if (h[x][y][z] > 0) {
           V[x][y][z] = 20.0;
                  new_V[x][y][z] = 20.0;
                }
           }

 //this next bit does the repeating stimulus.  leave it out for this one
  /* if (t>=(girishcounter*0.1) && t<((girishcounter*0.1)+HT))
          {
   for (x=100; x<=200; x++)
         for (y=325; y<=350; y++)
           for (z=0; z<=10; z++)
                if (h[x][y][z] > 0) {
           V[x][y][z] = 20.0;
                  new_V[x][y][z] = 20.0;
                }
                girishcounter++;
           }*/



        for (x = BLOCK_LOW(rank, size, X+1); x <= BLOCK_HIGH(rank, size, X+1); x++) {
      if (x == 0 || x == X) continue;

      for (y = 1; y < Y; y++)
               for (z = 1; z < Z; z++)
```

```
                if (g[x][y][z] < 0)
            for (i = -1; i <= 1; i++)
              for (j = -1; j <= 1; j++)
                for (k = -1; k <= 1; k++)
                                if (g[x][y][z] == -g[x + i][y + j][z + k])
                    V[x][y][z] = V[x + i][y + j][z + k];
        }

     for (x = BLOCK_LOW(rank, size, X+1); x <= BLOCK_HIGH(rank, size, X+1); x++) {
      if (x == 0 || x == X) continue;

      for (y=1; y<Y; y++)
       for (z=1; z<Z; z++)
            if (h[x][y][z] > 0) {

             Ek=RTONF*log(Kc[x][y][z]/Ki[x][y][z]);
          ENa=RTONF*log(140./Nai[x][y][z]);
          ECa=(RTONF/2.)*log(2.5/Cai[x][y][z]);

// ** Define the differential approximation ** //
             dvdx2 = (V[x+1][y][z]+V[x-1][y][z]-2*V[x][y][z])/(HX*HX);
             dvdy2 = (V[x][y+1][z]+V[x][y-1][z]-2*V[x][y][z])/(HX*HX);
             dvdz2 = (V[x][y][z-1]+V[x][y][z+1]-2*V[x][y][z])/(HX*HX);

         switch (h[x][y][z]) {
       case 1: dvdt[x][y][z]=D*(dvdx2+dvdy2+dvdz2)-I_tot_PM(x, y, z); break;
       case 2: dvdt[x][y][z]=D*(dvdx2+dvdy2+dvdz2)-I_tot_CT(x, y, z); break;
       case 3: dvdt[x][y][z]=D*(dvdx2+dvdy2+dvdz2)-I_tot_BB(x, y, z); break;
       case 4: dvdt[x][y][z]=D*(dvdx2+dvdy2+dvdz2)-I_tot_CS(x, y, z); break;
       case 5: dvdt[x][y][z]=0.5*D*(dvdx2+dvdy2+dvdz2)-I_tot_PV(x, y, z); break; //D is the
diffusion coefficient, notice smaller for PV.  now halved to 0.25
       case 6: dvdt[x][y][z]=D*(dvdx2+dvdy2+dvdz2)-I_tot_LA(x, y, z); break;
       case 7: dvdt[x][y][z]=D*(dvdx2+dvdy2+dvdz2)-I_tot_RA(x, y, z); break;
//     default: dvdt[x][y][z]=D*(dvdx2+dvdy2)-I_tot_RA(x, y, z);
        }

       new_V[x][y][z]=V[x][y][z]+HT*dvdt[x][y][z];
          }
   }


// ** Replace the old Voltage values with the new ones ** //
  for (x = BLOCK_LOW(rank, size, X+1); x <= BLOCK_HIGH(rank, size, X+1); x++)
    for (y=0; y<=Y; y++)
         for (z=0; z<=Z; z++)
           if (h[x][y][z] > 0) {
                V[x][y][z]=new_V[x][y][z];
              }
/*
   if (t>=0.04 && t<(0.04+HT))
        for (x = BLOCK_LOW(rank, size, X+1); x <= BLOCK_HIGH(rank, size, X+1); x++)
        {
       for (y=0; y<=200; y++)
                for (z=0; z<=10; z++)
         {
                  V[x][y][z] = -73.8;
                  new_V[x][y][z] = -73.8;

                }
          for (y=400; y<=Y; y++)
                for (z=0; z<=10; z++)
         {
                  V[x][y][z] = -73.8;
                  new_V[x][y][z] = -73.8;
                }
     }
        */
        if (rank < (size-1))
       MPI_Send(V[BLOCK_HIGH(rank, size, X+1)], (Y+1)*(Z+1), MPI_FLOAT, rank+1, 0,
MPI_COMM_WORLD);
     if (rank > 0)
       MPI_Recv(V[BLOCK_HIGH((rank-1), size, X+1)], (Y+1)*(Z+1), MPI_FLOAT, rank-1, 0,
MPI_COMM_WORLD, &status);
     if (rank > 0)
       MPI_Send(V[BLOCK_LOW(rank, size, X+1)], (Y+1)*(Z+1), MPI_FLOAT, rank-1, 1,
MPI_COMM_WORLD);
     if (rank < (size-1))
```

```
       MPI_Recv(V[BLOCK_LOW((rank+1), size, X+1)], (Y+1)*(Z+1), MPI_FLOAT, rank+1, 1,
MPI_COMM_WORLD, &status);

     if (cnt % 1000 == 0) {     //took off a zero
       MPI_Gatherv (&(((float *)V)[recv_disp[rank]]), recv_cnts[rank], MPI_FLOAT,
&global_V[0][0][0], recv_cnts, recv_disp, MPI_FLOAT, 0, MPI_COMM_WORLD);
     }

     if (!rank)

                if (cnt == 1000) //ditto
         {
             cnt=0;

             str = malloc (8*sizeof(char));
             sprintf (str, "a%d.vtk", cnt2++);
         out = fopen (str, "wt");

         fprintf (out, "# vtk DataFile Version 3.0\n");
         fprintf (out, "vtk output\n");
         fprintf (out, "ASCII\n");
         fprintf (out, "DATASET STRUCTURED_POINTS\n");
         fprintf (out, "DIMENSIONS %d %d %d\n", X+1, Y+1, Z+1);
         fprintf (out, "SPACING 1 1 1\n");
         fprintf (out, "ORIGIN 0 0 0\n");
         fprintf (out, "POINT_DATA %d\n", (X+1)*(Y+1)*(Z+1));
         fprintf (out, "SCALARS ImageFile float 1\n");
         fprintf (out, "LOOKUP_TABLE default\n");

                for (z=0; z<=Z; z++)
                {
              for (y=0; y<=Y; y++)
           {
             for (x=0; x<=X; x++)
                          fprintf (out, "%2.2f ", global_V[x][y][z]);

                 fprintf (out, "\n");
           }
           fprintf (out, "\n");
                }
            fclose (out);
            free (str);
         }
   }
   return 0;
}
```

Atria_Variables.h

```
#include <stdlib.h>

#define X   300
#define Y   600
#define Z   50
#define D   100

#define HX  0.1
#define HT  0.00001

#define Temp    308
#define Cm      0.05
#define Faraday 96487
#define R       8314

float Ek,ENa,ECa,EbCl,ECl;
float Vcell,Vi,VCa,Vc,Vup,Vrel;

int g[X+1][Y+1][Z+1];
int h[X+1][Y+1][Z+1];

float V[X+1][Y+1][Z+1];
float Pa[X+1][Y+1][Z+1];
float Pi[X+1][Y+1][Z+1];
float n[X+1][Y+1][Z+1];
float r1[X+1][Y+1][Z+1];
float s1[X+1][Y+1][Z+1];
```

```
float s2[X+1][Y+1][Z+1];
float s3[X+1][Y+1][Z+1];
float m[X+1][Y+1][Z+1];
float h1[X+1][Y+1][Z+1];
float h2[X+1][Y+1][Z+1];
float dL[X+1][Y+1][Z+1];
float fL1[X+1][Y+1][Z+1];
float fL2[X+1][Y+1][Z+1];
float y2[X+1][Y+1][Z+1];
float fca[X+1][Y+1][Z+1];
float dT[X+1][Y+1][Z+1];
float fT[X+1][Y+1][Z+1];
float F1[X+1][Y+1][Z+1];
float F2[X+1][Y+1][Z+1];
float F3[X+1][Y+1][Z+1];
float dF1[X+1][Y+1][Z+1];
float dF2[X+1][Y+1][Z+1];
float dF3[X+1][Y+1][Z+1];
float Kc[X+1][Y+1][Z+1];
float Ki[X+1][Y+1][Z+1];
float Nai[X+1][Y+1][Z+1];

float Cai[X+1][Y+1][Z+1];
float Caup[X+1][Y+1][Z+1];
float Carel[X+1][Y+1][Z+1];
float fac[X+1][Y+1][Z+1];
float faTc[X+1][Y+1][Z+1];
float faTmgc[X+1][Y+1][Z+1];
float faTmgm[X+1][Y+1][Z+1];
float faCalse[X+1][Y+1][Z+1];

float RTONF;
```

PV_Cell.h

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "Atria_Variables.h"

float I_tot_PV (int x, int y, int z)
{
   float IKf,IKs,IK,Ik1,Ito,IbNa,Ip,INaCa,IbCa,INa, ICap,ICaL,ICaT,Isus,IfNa,IfK,IKH_K,
IKH_Na, IKH,Iup,Itr,Irel;
   float ract, rinact,dfac,dfaTc,dfaTmgc,dfaTmgm,dfaCalse,dfab;

   float Ay, By;
   float ym, ty;
   float Apa, Bpa;
   float pam, tpa;
   float Api, Bpi;
   float pim, tpi;
   float An, Bn;
   float nm, tn;
   float Ar, Br;
   float rm, tr;
   float s1m, ts1;
   float s2m, ts2;
   float s3m, ts3;
   float xsusm;
   float Am, Bm;
   float Ah, Bh;
   float hm;
   float th1, th2;
   float Adl, Bdl;
   float dlm, tdl;
   float Afl, Bfl;
   float flm, tfl1, tfl2;
   float dfca;
   float Adt, Bdt;
   float dtm, tdt;
   float Aft, Bft;
   float ftm, tft;
   float E0=V[x][y][z]-Ek+3.6;
```

```
   Ay = exp(-(2.9 + (0.04*V[x][y][z])));
   By  = exp((3.6 + (0.11*V[x][y][z])));

   ym = 1/(1 + exp((V[x][y][z]+100)/20));
   ty = 0.4/(Ay+By);

   y2[x][y][z] += HT*(ym-y2[x][y][z])/ty;

   IKH_K= 0.1*ym*(V[x][y][z]-Ek);
   IKH_Na = 0.008*ym*(V[x][y][z]-ENa);
   IKH = (IKH_K + IKH_Na);

   IKf=4*0.0035*Pa[x][y][z]*Pi[x][y][z]*(V[x][y][z]-Ek);

   Apa=9.0*exp(V[x][y][z]/25.371);
   Bpa=1.3*exp(-V[x][y][z]/13.026);
   pam=1./(1+exp(-(V[x][y][z]+5.1)/7.4));
   tpa=1./(Apa+Bpa);
   Pa[x][y][z]+=HT*(pam-Pa[x][y][z])/tpa;

   Api=100.*exp(-V[x][y][z]/54.645);
   Bpi=656.*exp(V[x][y][z]/106.157);
   pim=1./(1+exp((V[x][y][z]+47.3921)/18.6603));
   tpi=1.0/(Api+Bpi);
   Pi[x][y][z]+=HT*(pim-Pi[x][y][z])/tpi;

   IKs=2.5*n[x][y][z]*(V[x][y][z]-Ek);

   An=1.66*exp(V[x][y][z]/69.452);
   Bn=0.3*exp(-V[x][y][z]/21.826);
   nm=1.0/(1+exp(-(V[x][y][z]-0.9)/13.8));
   tn=1./(An+Bn)+0.06;
   n[x][y][z]+=HT*(nm-n[x][y][z])/tn;

   IK=IKf+IKs;

   Ik1=(0.9*5.08*pow(5.0/(5.0+0.59),3)/(1+exp(1.393*(V[x][y][z]-Ek-
12)/RTONF))+0.05)*(V[x][y][z]-Ek-12);


Ito=0.45*0.515*50.02*r1[x][y][z]*(0.8*pow(s1[x][y][z],3)+0.2*pow(s2[x][y][z],3))*/*(0.600*pow(
s3[x][y][z],6)+0.4)*/(V[x][y][z]-Ek);     // CT - 0.2, PM - 0.35
   Isus = 0.46*(V[x][y][z]+85)+0.0*(V[x][y][z]-5);

   Ito+=Isus;

   Ar=386.6*exp(V[x][y][z]/12.0);
   Br=8.011*exp(-V[x][y][z]/7.2);
   rm=1/(1+exp(-(V[x][y][z]+15.0)/5.633));
   tr=1./(Ar+Br)+0.0004;
   r1[x][y][z]+=HT*(rm-r1[x][y][z])/tr;

   s1m=1./(1+exp((V[x][y][z]+29.4)/5.1));
   ts1=0.1155/(1+exp((V[x][y][z]+32.8)/0.1))+0.0147; // t1_r-t1_d / (Exp) + t1_d
   s1[x][y][z]+=HT*(s1m-s1[x][y][z])/ts1;

   s2m= 1./(1+exp((V[x][y][z]+29.4)/5.1));
   ts2=3.0967/(1+exp((V[x][y][z]+32.8)/0.1))+0.0926; // t2_r+t2_d / (Exp) + t2_d
   s2[x][y][z]+=HT*(s2m-s2[x][y][z])/ts2;

   s3m=((1./(1+exp((V[x][y][z]+50.67)/27.38)))+0.666)/1.666;
   ts3=(7.5/(1+exp((V[x][y][z]+23.0)/0.5)))+0.5;
   s3[x][y][z]+=HT*(s3m-s3[x][y][z])/ts3;


Ip=64.41*Kc[x][y][z]/(Kc[x][y][z]+1)*(pow(Nai[x][y][z],1.5)/(pow(Nai[x][y][z],1.5)+pow(11,1.5)
))*(1.6/(1.5+exp(-(V[x][y][z]+60)/40.)));

//Regular AP
//INaCa=0.02*(pow(Nai[x][y][z],3)*2.5*exp(0.450*V[x][y][z]/RTONF)-
pow(140,3)*Cai[x][y][z]*exp(V[x][y][z]*(0.45-
1)/RTONF))/(1+0.0003*(Cai[x][y][z]*pow(140,3)+2.5*pow(Nai[x][y][z],3)));

//SAP
INaCa = 1.5*(0.02*(pow(Nai[x][y][z],3)*7.5*exp(0.35*V[x][y][z]/RTONF)-
pow(140,3)*Cai[x][y][z]*exp(V[x][y][z]*(0.45-
1)/RTONF))/(1+0.0003*(Cai[x][y][z]*pow(140,3)+2.5*pow(Nai[x][y][z],3)))); //1.5
```

```
   if(fabs(V[x][y][z]+44.4) < 0.0001)
      Am=460.*12.673;
   else
     Am=-460*(V[x][y][z]+44.4)/(exp(-(V[x][y][z]+44.4)/12.673)-1);
   Bm=18400.0*exp(-(V[x][y][z]+44.4)/12.673);
   m[x][y][z] = Am/(Am+Bm);
//   m[x][y][z]+=HT*(Am*(1-m[x][y][z])-Bm*m[x][y][z]);

   Ah=44.9*exp(-(V[x][y][z]+66.9)/5.57);
   Bh=1491.0/(1+323.3*exp(-(V[x][y][z]+94.6)/12.9));

   th1=0.03/(1+exp((V[x][y][z]+40)/6.0))+0.00015;  //0.00035 - Lindblad
   th2=0.12/(1+exp((V[x][y][z]+60)/2.0))+0.00045;  //0.00295 - Lindblab

   hm=Ah/(Ah+Bh);
   h1[x][y][z]+=HT*(hm-h1[x][y][z])/th1;
   h2[x][y][z]+=HT*(hm-h2[x][y][z])/th2;

   if(fabs(V[x][y][z]) > 0.0001)

INa=0.75*0.0014*pow(m[x][y][z],3)*(0.635*h1[x][y][z]+0.365*h2[x][y][z])*140*V[x][y][z]*(Farada
y/RTONF)*(exp((V[x][y][z]-ENa)/RTONF)-1)/(exp(V[x][y][z]/RTONF)-1); // multiply by 0.75 for
instant activation
   else
INa=0.75*0.0014*pow(m[x][y][z],3)*(0.635*h1[x][y][z]+0.365*h2[x][y][z])*140*Faraday*(exp((V[x]
[y][z]-ENa)/RTONF)-1);

   ICap=9.509*(Cai[x][y][z]/(Cai[x][y][z]+0.0002));

   Adl=-16.72*(V[x][y][z]+45)/(exp(-(V[x][y][z]+45)/2.5)-1)-50.0*(V[x][y][z]+10)/(exp(-
(V[x][y][z]+10)/4.808)-1);
   if(fabs(V[x][y][z]+45) < 0.0001)
     Adl=16.72*2.5-50.0*(V[x][y][z]+10)/(exp(-(V[x][y][z]+10)/4.808)-1);
   if(fabs(V[x][y][z]+10) < 0.0001)
     Adl=-16.72*(V[x][y][z]+45)/(exp(-(V[x][y][z]+45)/2.5)-1)+50.0*4.808;

   if(fabs(V[x][y][z]+5.) < 0.0001)
     Bdl=4.48*2.5;
   else
     Bdl=4.48*(V[x][y][z]+5)/(exp((V[x][y][z]+5)/2.5)-1.);
   tdl=1/(Adl+Bdl);

   dlm=1./(1+exp(-(V[x][y][z]+7.5)/5.5));
   dL[x][y][z]+=HT*(dlm-dL[x][y][z])/tdl;


  flm=1./(1+exp((V[x][y][z]+23.4)/5.2));
  tfl1=(0.0336/(1+exp((V[x][y][z]+18)/4.0)))+0.0142;
  tfl2=(3.6908/(1+exp((V[x][y][z]+18)/4)))+0.0534;
  fL1[x][y][z]+=HT*(flm-fL1[x][y][z])/tfl1;
  fL2[x][y][z]+=HT*(flm-fL2[x][y][z])/tfl2;

  dfca=1.0/(1+Cai[x][y][z]/0.00001);  //0.00025
  fca[x][y][z]+=HT*(dfca-fca[x][y][z])/0.04;

//   ICaL=18*(dL[x][y][z]*((0.80*fL1[x][y][z])+(0.2*fL2[x][y][z])))*fca*(V[x][y][z]-61.4);
     ICaL=25*dL[x][y][z]*fL1[x][y][z]*fca[x][y][z]*(V[x][y][z]-58.3);

   Adt=674.173*exp((V[x][y][z]+23.3)/30.0);
   Bdt=674.173*exp(-(V[x][y][z]+23.3)/30.0); //23.3
   tdt=1/(Adt+Bdt);
   dtm=1./(1+exp(-(V[x][y][z]+23.0)/5.1)); //23 6.1
   dT[x][y][z]+=HT*(dtm-dT[x][y][z])/tdt;

   Aft=9.637*exp(-(V[x][y][z]+75)/83.3);// 75
   Bft=9.637*exp((V[x][y][z]+75)/16.3);// 15.38
   tft=1.0/(Aft+Bft);
   ftm=Aft/(Aft+Bft);
   fT[x][y][z]+=HT*(ftm-fT[x][y][z])/tft;

   ICaT=1.65*8.0*dT[x][y][z]*fT[x][y][z]*(V[x][y][z]-38.0);

   IbNa=0.02*(V[x][y][z]-ENa);  // 0.02 - CT, 0.03 - PM

   IbCa=0.02*(V[x][y][z]-ECa);  // 0.02 - CT, 0.03 - PM
```

```
   Nai[x][y][z]+=HT*(-3*Ip-3*INaCa-IbNa-INa)/(Faraday*Vi);

   Iup=3*2800.0*(((Cai[x][y][z]/0.0003)-
(0.4*0.4*Caup[x][y][z]/0.5))/(((Cai[x][y][z]+0.0003)/0.0003)+(0.4*(Caup[x][y][z]+0.5)/0.5)));

   Irel=4*200000.0*pow(F2[x][y][z]/(F2[x][y][z]+0.25),2)*(Carel[x][y][z]-Cai[x][y][z]);

   Itr=(Caup[x][y][z]-Carel[x][y][z])*2.*Faraday*Vup/0.01;

   dfac=200000.0*Cai[x][y][z]*(1-fac[x][y][z])-476*fac[x][y][z];
   dfaTc=78400*Cai[x][y][z]*(1-faTc[x][y][z])-392*faTc[x][y][z];
   dfaTmgc=200000*Cai[x][y][z]*(1-faTmgc[x][y][z]-faTmgm[x][y][z])-6.6*faTmgc[x][y][z];
   dfaTmgm=2000*2.5*(1-faTmgc[x][y][z]-faTmgm[x][y][z])-666*faTmgm[x][y][z];
   dfaCalse=480*Carel[x][y][z]*(1-faCalse[x][y][z])-400*faCalse[x][y][z];
   dfab=0.08*dfaTc+0.16*dfaTmgc+0.045*dfac;

   Caup[x][y][z]+=HT*(Iup-Itr)/(2.*Faraday*Vup);
   Carel[x][y][z]+=HT*((Itr-Irel)/(2.*Faraday*Vrel)-31.0*(480*Carel[x][y][z]*(1-
faCalse[x][y][z])-400*faCalse[x][y][z]));
   Cai[x][y][z]+=HT*((2.*INaCa-ICaL-ICaT-ICap-IbCa-Iup+Irel)/(2.*VCa*Faraday)-dfab);

   fac[x][y][z]+=HT*dfac;
   faTc[x][y][z]+=HT*dfaTc;
   faTmgc[x][y][z]+=HT*dfaTmgc;
   faTmgm[x][y][z]+=HT*dfaTmgm;
   faCalse[x][y][z]+=HT*dfaCalse;

   Kc[x][y][z]+=HT*(-2.*Ip+IKf+IKs+Ito+Ik1)/(Faraday*Vc);
   Ki[x][y][z]+=HT*(2.*Ip-IKf-IKs-Ito-Ik1)/(Faraday*Vi);

   ract=240.0*exp((V[x][y][z]-20.)/12.5)+203.8*pow(Cai[x][y][z]/(Cai[x][y][z]+0.0003),4);
   rinact=33.96+339.6*pow(Cai[x][y][z]/(Cai[x][y][z]+0.0003),4);

   dF1[x][y][z]=0.815*F3[x][y][z]-ract*F1[x][y][z];
   dF2[x][y][z]=ract*F1[x][y][z]-rinact*F2[x][y][z];
   dF3[x][y][z]=rinact*F2[x][y][z]-0.815*F3[x][y][z];

   F1[x][y][z]+=HT*dF1[x][y][z];
   F2[x][y][z]+=HT*dF2[x][y][z];
   F3[x][y][z]+=HT*dF3[x][y][z];


   return ((IKf+Ik1+Ito+Ip+INaCa+INa+IbNa+ICap+ICaL+ICaT+IbCa+IKH)/Cm);

}
```

3Dgeo.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

        const int X = 300; // one side of either left or right atrium
        const int Y = 600; // left and right atrium length of one side
        const int Z = 50; // height of the whole lot
        const int pv_radius = 30; // radius of pulmonary vein
        const int pv_thickness = 5;
        const int offset_x = 250;
        const int offset_y = 200;

        int garray[X+1][Y+1][Z+1]; // the array that will later be outputted to a file

int main()
{
        int x, y, z;

        FILE *in;
        in = fopen("geometry.c", "w");

        for (z=0; z<=10; z++) // layer number one-10, mostly 1s but also with a hole
        {
                for (y=0; y<=Y; y++)
                {
                        for (x=0; x<=X; x++)
                        {
```

```
                                if ((((x-offset_x)*(x-offset_x)+(y-offset_y)*(y-
offset_y))>=pv_radius*pv_radius))
                                {

                                        //LA/RA
                                        if (y<300)
                                        {
                                                garray[x][y][z]=6;
                                        }
                                        else
                                        {
                                                garray[x][y][z]=7;
                                        }

                                        //PM
                                        if ((y>350 && y<= 450) && ((x >= 40 && x < 60) || (x >=
80 && x <100) || (x >= 120 && x < 140) || (x >= 160 && x < 180) || (x >= 200 && x <220 ) || (x
>= 240 && x < 260)))
                                        {
                                                garray[x][y][z] = 1;
                        }

                                        //CT
                                        if((y>=325 && y<= 350) && (x>=25 && x<=275))
                                        {
                                                garray[x][y][z] = 2;
                        }

                                        //BB
                                        if((y>=250 && y<= 350) && (x>=0 && x<25))
                                        {
                                                garray[x][y][z] = 3;
                                        }

                                        //CS
                                        if((y>=250 && y<= 350) && (x>=275 && x<X))
                                        {
                                                garray[x][y][z] = 4;
                                        }

                                }
                                else
                                {
                                        garray[x][y][z] = 0;
                                }
                        }
                }
        }

        for (z=11; z<=Z; z++) // layer number 11-50, mostly 0s but also with a pul vein
        {
                for (y=0; y<=Y; y++)
                {
                        for (x=0; x<=X; x++)
                        {
                                if ((((x-offset_x)*(x-offset_x)+(y-offset_y)*(y-
offset_y))>=pv_radius*pv_radius)&&(((x-offset_x)*(x-offset_x)+(y-offset_y)*(y-
offset_y))<=(pv_radius+pv_thickness)*(pv_radius+pv_thickness)))
                                {
                                        garray[x][y][z] = 5; // 5 being pul vein
                                }
                                else
                                {
                                        garray[x][y][z] = 0;
                                }
                        }
                }
        }

        for (z=0; z<=Z; z++) // all edges are zero
        {
                for (y=0; y<=Y; y++)
                {
                        for (x=0; x<=X; x++)
                        {
                                if (x==0 || x==X || y==0 || y==Y || z==0 || z==Z || x==1 ||
x==X-1 || y==1 || y==Y-1 || z==1 || z==Z-1)
```

```
                                {
                                        garray[x][y][z] = 0;
                                }
                        }
                }
        }

        for (z=0; z<=Z; z++) // septum is zero
        {
                for (y=0; y<=Y; y++)
                {
                        for (x=0; x<=X; x++)
                        {
                                if (y==300)
                                {
                                        if (garray[x][y][z]!=4 && garray[x][y][z]!=3) // to make
sure septum does not extend to CS and BB cells
                                        {
                                                garray[x][y][z] = 0;
                                        }
                                }
                        }
                }
        }


    for (z=0; z<=Z; z++)
        {
                for (y=0; y<=Y; y++) // printing it all
                {
                        for (x=0; x<=X; x++)
                        {
                                fprintf(in, "%d ", garray[x][y][z]);
                        }
                        fprintf(in, "\n");
                }
                fprintf(in, "\n");
        }


 fclose(in);
 return 0;
}
```

References

[i] Aslanidi, O.V., Dewey, R.S., Morgan, A.R., Boyett, M.R., Zhang, H. 2008, 'Regional differences in rabbit atrial action potential properties: mechanisms, consequences and pharmacological implications', *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, pp. 141-144.

[ii] Darling, D. <http://www.daviddarling.info/encyclopedia/S/sinoatrial_node.html>. (Image used with permission.)

[iii] Phibbs, B. 1975, *The Human Heart*, Mosby, St Louis.

[iv] Webster, J.G. 1998, *Medical Instrumentation*, John Wiley & Sons, New York.

[v] Seeley, R.R., Stephens, T.D., Tate 1995, *Anatomy & Physiology*, Mosby, St Louis, p. 267.

[vi] Sachse, F. 2004, *Cardiac Electrophysiology*, Springer, Berlin, p. 159.

[vii] Klabunde, R.E. <http://www.cvphysiology.com/Arrhythmias/A007.htm>.

[viii] Lindblad, D.S., Murphey, C.R., Clark, W., Giles, W.R. 1966, 'A model of the action potential and underlying membrane currents in a rabbit atrial cell', *Sp. Comm. Am. Physiol. Soc.*, pp. 1666-1696.

[ix] Berne, R.M, Levy, M.N. 2000, *Principles of Physiology*, Modsby, Missouri, pp. 195-196.

[x] Keener, J., Sneyd, J. 1998, *Mathematical Physiology*, Springer, New York.

[xi] Hodkin, A., Huxley, A 1952, 'A quantitative description of membrane current and its application to conduction and excitation in nerve', *J. Physiol.*, no. 117, pp. 500-544.

[xii] Berne, R.M., Levy, M.N. 1998, *Physiology*, Mosby, Missouri, pp. 330-338.

[xiii] Liu, D.W., Gintant, G.A., Antzelevitch, C. 1993, 'Ionic bases for electrophysiological distinctions among epicardial, midmyocardial, and myocytes from the free wall of the canine left ventricle', *Circ. Res.*, vol. 72, pp. 671-687.

[xiv] Chard, M., Tabrizchi, R. 2009, 'The role of pulmonary veins in atrial fibrillation: A complex yet simple story', *Pharm. & Ther.*, no. 124, pp. 207-218.

[xv] Atienza, F., Jalife, J. 2007, 'Reentry and atrial fibrillation', *Heart Rhym. Soc.*

[xvi] Haissaguerre, M et. al. 1998, 'Spontaneous initiation of atrial fibrillation by ectopic beats originating in the pulmonary veins, *New England Journ. of Med.*, vol. 339, no. 10, pp. 659-666.

[xvii] British Heart Foundation <http://www.bhf.org.uk/living_with_a_heart_condition/treatment/ablation.aspx>.