BME 525 Final Project:
Python Hybrid Automaton Implementation of Cardiac Cells

Alexander Kyu

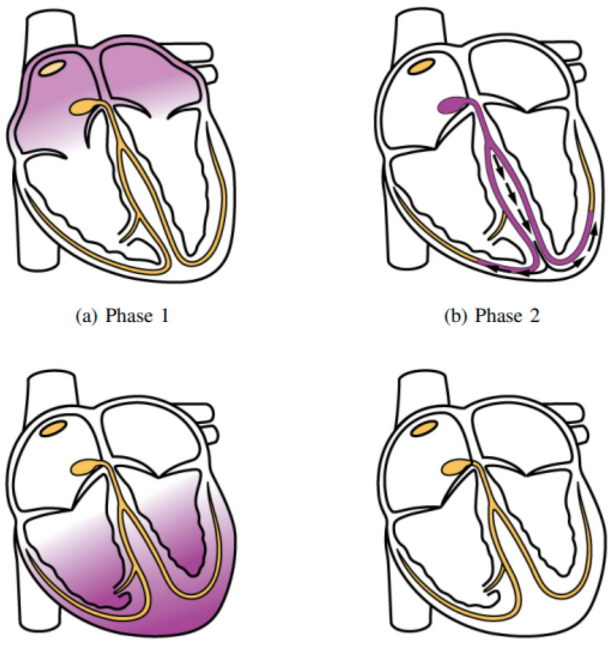
I. Background and Rationale

The cardiovascular system is one of the most complex and most important systems within the human body and at the center of that is the heart, an organ and muscle that is responsible for creating that pumping action that pushes the blood throughout the cardiovascular system. The cardiac pacemaker cells play an incredibly important role in this function, consistently stimulating the heart with an action potential, starting the cascade that eventually creates a heartbeat. However, patients can develop abnormal heart behaviors that can alter and even compromise heart functionality—such as bradycardia or a slower heart rate. In these cases, a pacemaker can be critical to saving the patients life by continuously monitoring the patient's heart rate and delivering electrical stimuli when needed.

These medical devices must be certified by various regulatory bodies and standards that often have strict guidelines and requirements such as IEC 60601-1. This often involves clinical trials in which the pacemaker is tested in patients, giving the regulating body an idea of how the device will perform when released as a product. However, with product development cost, safety cost, and other costs clinical trials can be very expensive and very time inefficient, a major hurdle for medical devices. These hurdles include small patient populations, patient compliance, misrepresentation of the patient population, patient risk, and difficulty in collecting consistent and meaningful data from patients.

In the case of pacemakers specifically, a virtual, fully simulated heart could be a good way of doing closed-loop testing of pacemakers with a heart. These heart models can react to outputted shocks from the pacemaker in real-time, allowing for real-time reactions from the pacemaker and a more easily analyzable heart. This is also a rather cheap method of testing the device and therefore can also be done earlier in the product development cycle, reducing the likelihood of having to re-evaluate the device late in the development cycle—which can be very costly and time consuming.

Before we can emulate the heart in a simulated model, we must first understand both the cardiac cycle and the Cardiac Action Potential. The cardiac cycle can be described in four different phases. In phase one, the Sinoatrial Node generates the electrical stimulus that eventually propagates throughout the heart. This propagation first reaches the right and left atriums, causing them to contract, pumping blood into the right and left ventricles. In phase two the action potential reaches the atrioventricular node and propagates down the septum, eventually reaching and stimulating the ventricles after a short delay, allowing the atria to contract before the ventricles. In phase three, the stimulus has reached the apex of the ventricles and quickly propagates out into the fast Purkinje fibers, causing the ventricles to contract, pushing blood out of the heart and into the body. In phase four, the ventricles relax, resetting the heart before another stimulus is triggered. All of these phases can be summed up in Figure 1a-d for phases one, two, three, and four, respectively.



(a) Phase 1 (b) Phase 2 (c) Phase 3 (d) Phase 4

Figure 1: Phases of the Cardiac Cycle from [1] which was adapted from http://philschatz.com/anatomy-book/contents/m46664.html#sinoatrial-sa-node.

Now that we have established the macrolevel of the heart, we can start to talk about the units that make up the heart and more specifically the electrical pathway of the heart—no mechanical mechanisms are discussed in this paper. The heart is made up of cardiac cells with many different properties that shape their respective action potentials. However, all cardiac cell (cardiomyocyte) action potentials can be broken into 5 phases and all pacemaker cells can be broken down into 4 phases (it skips phase 1).

In cardiomyocytes, the cell starts in phase 4 or in the resting phase where the cell is largely inactive. When a neighboring cell or external source stimulates it, the cardiomyocyte enters the stimulated phase. If the cell passes the voltages threshold (VT), then it enters the upstroke phase (phase 0) where the cell depolarizes rapidly to reach an overshoot voltage (Vo). It then enters phase 1 which is characterized by a sudden repolarization before entering phase 2, which is characterized by a plateau. Finally, the cell will move into phase 3 when the cell continues to repolarize and once it reaches the resting voltage (VR), the cell finally relaxes and re-enters phase 4.

In pacemaker cells such as the SA and AV Nodes, their action potential can be better characterized by 4 phases. The cell starts in phase 4 or the resting phase where it is slowly depolarizing over time. When the cell has reached the voltage threshold (VT), it rapidly depolarizes during phase 0 before entering phase 2 characterized by a rather steep plateau and early repolarization. Once the voltage has dropped below VR, the cell enters phase 3 or fast repolarization before re-entering the resting phase. Both action potentials can be seen in Figure 2.

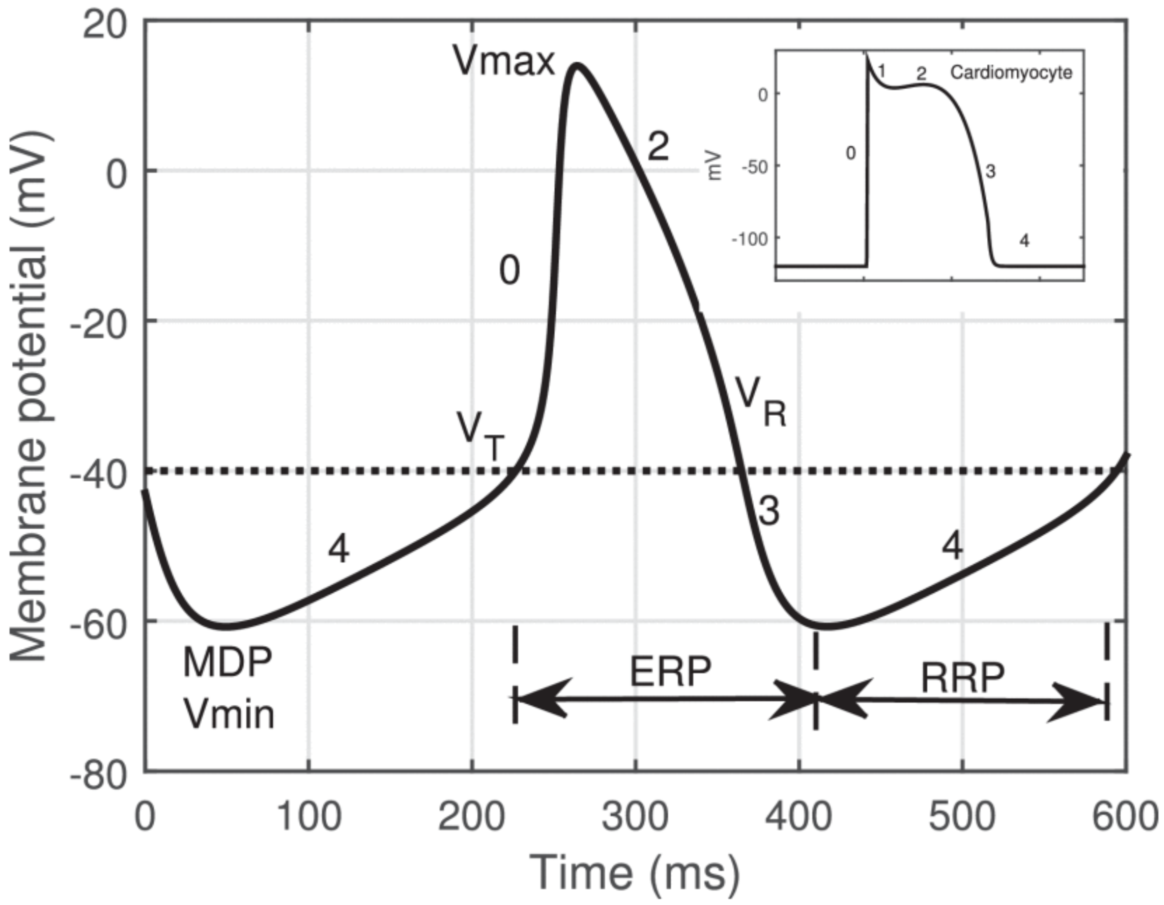


Figure 2: Action Potential of Pacemaker Cells (main graph) and Cardiomyocytes (shown in the top right). MDP Vmin is the Maximal Diastolic Potential. VT is the threshold potential. VR is the resting potential. ERP is the effective or absolute refractory period. RRP is the relative refractory period. This figure is from [2].

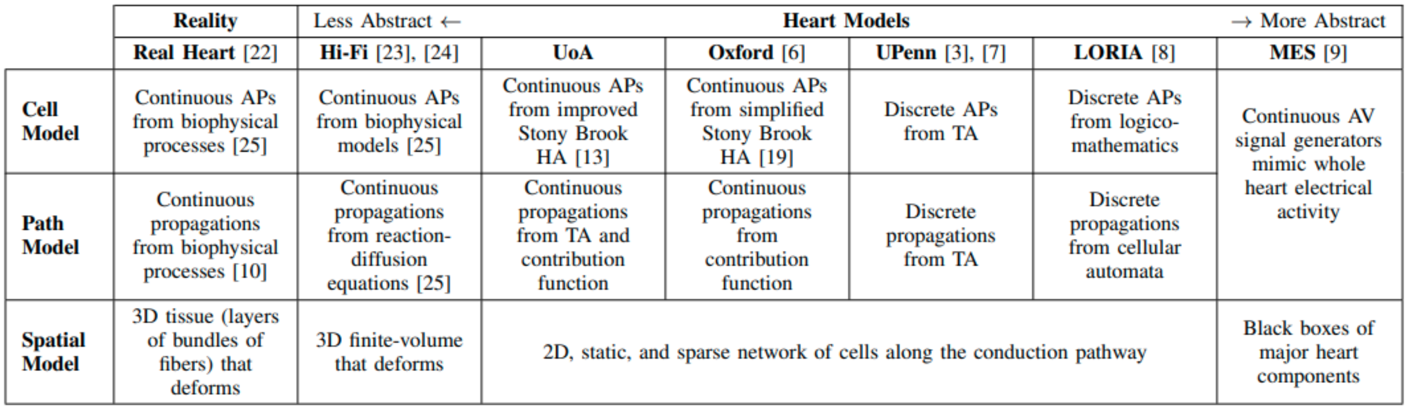
II. Rationale

When selecting the model for this project, several high-level models were considered during the peer review process. First a physiological model of the heart can be split into three different submodels. First, the cell model describes how the cell itself reacts over time with various external stimuli. The second submodel is the path model, which describes how an action potential travels from one cell to another. Finally, the third submodel is the spatial model, which describes how this collection of cells and paths interact within space.

Yip et al [1] did a thorough review of several different models. When looking at the real heart, the signals are continuous, and these signals can continuously be propagated between cells. This allows for 3-Dimensional tissues to develop with bundling of these connected fibers and cells that can be deformed. Of course, there are lots of equations and rather robust models that can model the heart quite accurately. These models, however, are quite computationally expensive despite being accurate and useful. More abstract models are much more computationally cheaper while still allowing for accurate closed-loop testing for pacemaker devices. These observations can be summed up in Table I:

Table I

Comparison of Heart Models designed for testing medical devices. AP = Action Potential. HA = Hybrid Automata. TA = Timed Automata. Table is from [1].



	Reality	Less Abstract ←		**Heart Models**			→ More Abstract
	Real Heart [22]	**Hi-Fi** [23], [24]	**UoA**	**Oxford** [6]	**UPenn** [3], [7]	**LORIA** [8]	**MES** [9]
Cell Model	Continuous APs from biophysical processes [25]	Continuous APs from biophysical models [25]	Continuous APs from improved Stony Brook HA [13]	Continuous APs from simplified Stony Brook HA [19]	Discrete APs from TA	Discrete APs from logico-mathematics	Continuous AV signal generators mimic whole heart electrical activity
Path Model	Continuous propagations from biophysical processes [10]	Continuous propagations from reaction-diffusion equations [25]	Continuous propagations from TA and contribution function	Continuous propagations from contribution function	Discrete propagations from TA	Discrete propagations from cellular automata	
Spatial Model	3D tissue (layers of bundles of fibers) that deforms	3D finite-volume that deforms	2D, static, and sparse network of cells along the conduction pathway				Black boxes of major heart components

Because the goal of this project was to simulate cardiac and pacemaker cells in a more computationally efficient way, the chosen models that were implemented were the University of Auckland's (UoA) model described by Yip et al [1] for cardiomyocytes, the pacemaker cells, and path model described by Ai et al [2]. These are all Hybrid Automata Models (HA) which are mathematical models for describing a system in which a digital computational process can interact with an analog process. More specifically, these are finite state machines (FSMs) with continuous variables that are described by differential equations. This allows us to break each action potential and the path model into discrete phases and describe each one with an empirically derived differential equation or set of differential equations.

III. Methods

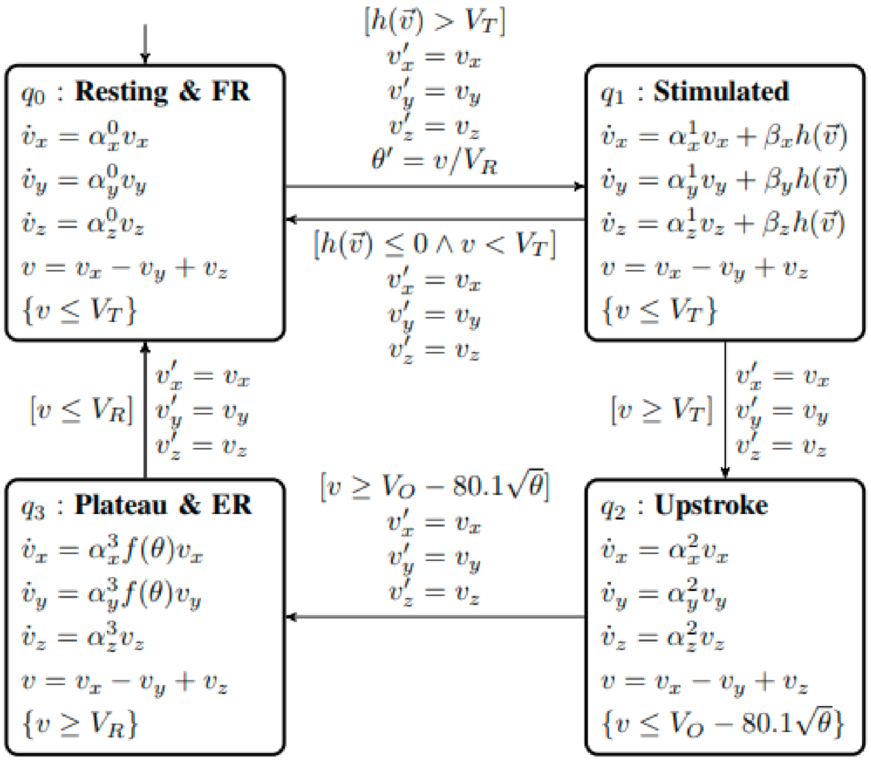
As stated in the section above, all of the models are HA models which allow each of the models to be broken down into FSMs, each with their own set of differential equations to describe them. Overall, a total of four HA models were implemented in this project: a cardiomyocyte model, nodal pacemaker cell model, subsidiary pacemaker cell model, and a path model.

The first model, the cardiomyocyte model, is an extension of the Stony Brook Cell model [3]. This model is made of up four distinct phases in Figure 3. Each phase describes a different set of three differential equations for v_x, v_y, and v_z. All α and β constants are from the Stony Brook Model. These are then summed up as described in the phases to the overall membrane voltage v. Furthermore, external voltages are accounted for by $h_k(\vec{v})$ [Equation (1)] where Γ_{ik} is the cross-sectional area (units of mm2) from cell i to k, σ_{ik} is the electrical conductivity (units of mS/mm) from cell i to k, A_m is cell k's surface area to volume (units of mm^{-2}), C_m is cell k's specific membrane capacitance (units of $\mu F/mm^2$), and v_i^{out} is cell i's membrane potential after propagating along the path.

$$\mathrm{h_k}(\vec{v}) = \sum_{n=1}^{n} \left(\frac{\Gamma_{ik}\sigma_{ik}}{A_m C_m} (v_i^{out} - v_k) \right) \qquad (1)$$

The improvements that Yip et al [1] implemented was the addition of the f(θ) in the derivative of v_x in q3 and the addition of a maximum f(θ) value [Equation (1)]. The addition of f(θ) for the derivative of v_x in q3 solved the issue in the Stony Brook model that created progressively shortening action potentials. Furthermore, the imposed maximum for f(θ) further prevents this shortening of action potentials over time.

$$\mathrm{f}(\theta) = \begin{cases} 0.29 * e^{62.89*\theta} + e^{-10.99*\theta}, & if\ \theta < 0.04 \\ 4.0395, & if\ \theta \geq 0.04 \end{cases} \qquad (2)$$



$[h(\vec{v}) > V_T]$
$v_x' = v_x$
$v_y' = v_y$
$v_z' = v_z$
$\theta' = v/V_R$
q_0 : Resting & FR
$\dot{v}_x = \alpha_x^0 v_x$
$\dot{v}_y = \alpha_y^0 v_y$
$\dot{v}_z = \alpha_z^0 v_z$
$v = v_x - v_y + v_z$
$\{v \leq V_T\}$
q_1 : Stimulated
$\dot{v}_x = \alpha_x^1 v_x + \beta_x h(\vec{v})$
$\dot{v}_y = \alpha_y^1 v_y + \beta_y h(\vec{v})$
$\dot{v}_z = \alpha_z^1 v_z + \beta_z h(\vec{v})$
$v = v_x - v_y + v_z$
$\{v \leq V_T\}$
$[h(\vec{v}) \leq 0 \wedge v < V_T]$
$v_x' = v_x$
$v_y' = v_y$
$v_z' = v_z$
$[v \leq V_R]$
$v_x' = v_x$
$v_y' = v_y$
$v_z' = v_z$
$[v \geq V_T]$
$v_x' = v_x$
$v_y' = v_y$
$v_z' = v_z$
$[v \geq V_O - 80.1\sqrt{\theta}]$
$v_x' = v_x$
$v_y' = v_y$
$v_z' = v_z$
q_3 : Plateau & ER
$\dot{v}_x = \alpha_x^3 f(\theta) v_x$
$\dot{v}_y = \alpha_y^3 f(\theta) v_y$
$\dot{v}_z = \alpha_z^3 v_z$
$v = v_x - v_y + v_z$
$\{v \geq V_R\}$
q_2 : Upstroke
$\dot{v}_x = \alpha_x^2 v_x$
$\dot{v}_y = \alpha_y^2 v_y$
$\dot{v}_z = \alpha_z^2 v_z$
$v = v_x - v_y + v_z$
$\{v \leq V_O - 80.1\sqrt{\theta}\}$

Figure 3: UoA Model which is an improvement on the Stony Brook Model. Figure is from [1].

The second model, the nodal pacemaker cell, was developed by Ai et al. [2] and is described in Figure 4. This model, like the UoA model, is broken up into four different states each with a scaling factor, d4, d0, d3, and d2 for each phase respectively, and a non-linear functions $f_3(\theta)$ and $f_2(\theta)$. The transition between q4 and q0 has two different pathways. The first pathway is the bottom path which is through the natural depolarization of the cell over time that eventually triggers the Action Potential once achieving the threshold voltage (V_T). The second pathway is the top pathway which describes the upstroke phase being triggered by external stimuli, described by the $g(\vec{v})$ function [Equation (3)] where D_{ik} is a decay factor of voltage contribution from cell i to cell k:

$$g(\vec{v}) = \sum_{n=1}^{n} (D_{ik} * v_i^{out} - v_k) \qquad (3)$$

The other variables help keep track of the state of the model as well as modify conditions based on the environment. θ keeps track of the frequency of external stimuli applied to the cell model. If the cell has been stimulated while in its relative refractory period, then θ is closer to 1. Otherwise, its closer to 0. The i variable is a counter for the beats. n and w(n) capture the effect of repetitive overdrive stimulations to the cell defined in equation 4. Equations 5-8 show f1-f4(θ).

$$w(n) = \begin{cases} 0 & if\ n < 1 \\ \dfrac{1}{1 + e^{-h_r*(n-h_s)}} & if\ n \geq 1\ and\ 1 \leq \dfrac{5}{h_r} + h_s \\ \dfrac{1}{1 + e^{-5}} & if\ n > \dfrac{5}{h_r} + h_s \end{cases} \tag{4}$$

$$f_1(\theta) = e^{(-h*\theta)} \tag{5}$$

$$f_2(\theta) = e^{(-f*\theta)} \tag{6}$$

$$f_3(\theta) = \frac{1}{w(n) * m * \theta^s + 1} \tag{7}$$

$$f_4(\theta) = e^{(j*(\theta-1\_)} \tag{8}$$

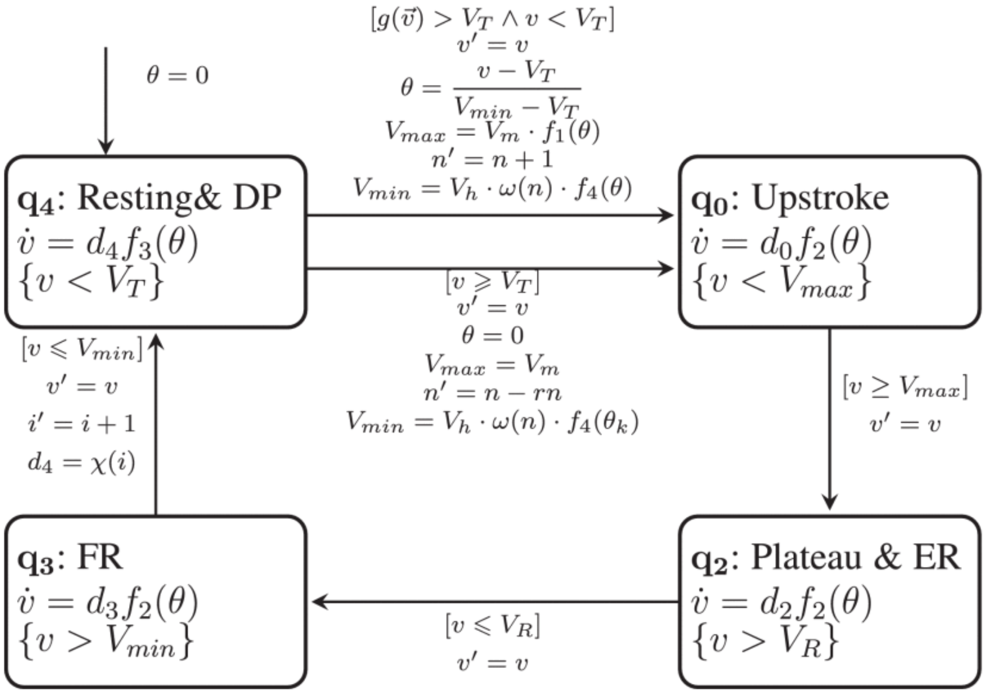


Figure 4: HA model for the Nodal Pacemaker Cell Model. Figure and model from [2].

Finally, when the model transitions back from q3 to q4, d4 is updated using χ(i) which is described in equations 9 and 10. This equation looks at the power spectrum of the variation in bpm to create a new d4 which determines the new slope and therefore how long it takes before the threshold voltage is reached again. This creates a variation in timing of heart beats seen in realistic heart models. In equation 9, BCL(i) gives the time between two action potentials that occurred in succession based on the power spectrum found by McSharry et al [4].

$$\chi(i) = \frac{V_T - V_{min}}{BCL(i) - C} \tag{9}$$

$$C = \frac{V_{max} - V_R}{|d_2|} + \frac{V_R - V_{min}}{|d_3|} + \frac{V_{max} - V_T}{d_0} \tag{10}$$

The third model, the subsidiary pacemaker cell, is a combination of the first two models linked together using the fourth model described below (the path model). These cells behave similarly to cardiomyocytes but retain the automaticity that the nodal pacemaker cells—the His-Purkinje cells are an example of this. Figure 5 shows this combination connecting the two models with a zero-distance path with the cardiomyocyte cell being the external interfacing cell. This allows the encapsulated nodal cell to act as a consistent stimulator while maintaining the properties of cardiomyocyte cells.

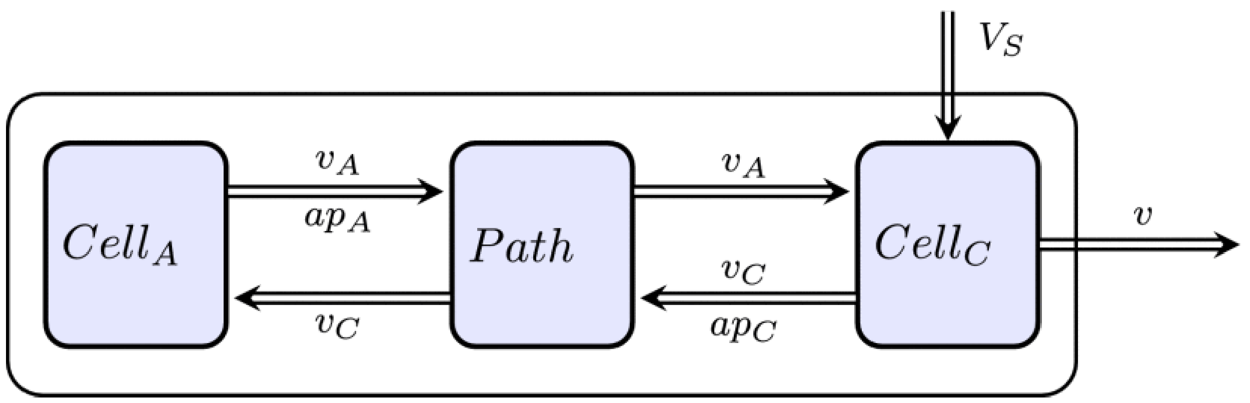
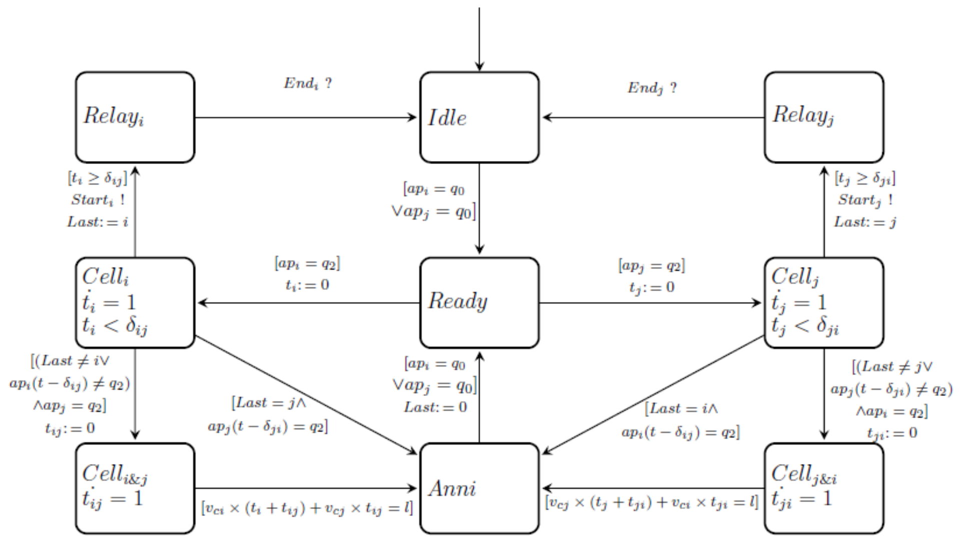
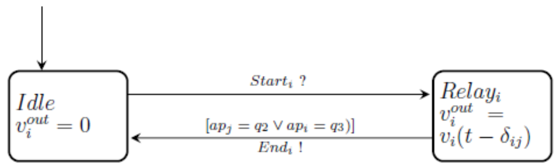


Figure 5: Subsidiary Pacemaker Cell Model. This shows the encapsulation of the other two models being connected via a 0 length path. Cell C is the cardiomyocyte model and is the interface with external cells. If no external cell stimulates Cell C, then Cell A (the internal nodal cell) will stimulate it, creating automaticity. Figure from [2].

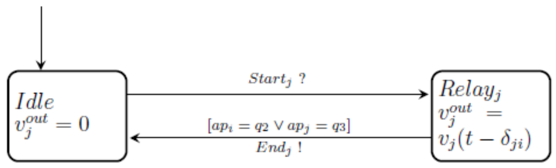
The final model implemented is the path model from Yip et al. (Figure 6) [2]. This model simulates the connection between two cells—cell i and cell j as shown in the figure. This model takes into account conduction velocity of each cell and the distance between them. Furthermore, this model also takes into account signals annihilating each other if two signals travel along the same path against each other.


$Relay_i$
End_i ?
$Idle$
End_j ?
$Relay_j$
$[t_i \geq \delta_{ij}]$ $Start_i$! $Last := i$
$[ap_i = q_0 \vee ap_j = q_0]$
$[t_j \geq \delta_{ji}]$ $Start_j$! $Last := j$
$Cell_i$ $\dot{t}_i = 1$ $t_i < \delta_{ij}$
$[ap_i = q_2]$ $t_i := 0$
$Ready$
$[ap_j = q_2]$ $t_j := 0$
$Cell_j$ $\dot{t}_j = 1$ $t_j < \delta_{ji}$
$[(Last \neq i \vee ap_i(t - \delta_{ij}) \neq q_2) \wedge ap_j = q_2]$ $t_{ij} := 0$
$[Last = j \wedge ap_j(t - \delta_{ji}) = q_2]$
$[ap_i = q_0 \vee ap_j = q_0]$ $Last := 0$
$[Last = i \wedge ap_i(t - \delta_{ij}) = q_2]$
$[(Last \neq j \vee ap_j(t - \delta_{ji}) \neq q_2) \wedge ap_i = q_2]$ $t_{ji} := 0$
$Cell_{i\&j}$ $\dot{t}_{ij} = 1$
$[v_{ci} \times (t_i + t_{ij}) + v_{cj} \times t_{ij} = l]$
$Anni$
$[v_{cj} \times (t_j + t_{ji}) + v_{ci} \times t_{ji} = l]$
$Cell_{j\&i}$ $\dot{t}_{ji} = 1$

(a) The HA model of the path.


$Idle$ $v_i^{out} = 0$
$Start_i$?
$[ap_j = q_2 \vee ap_i = q_3)]$ End_i !
$Relay_i$ $v_i^{out} = v_i(t - \delta_{ij})$

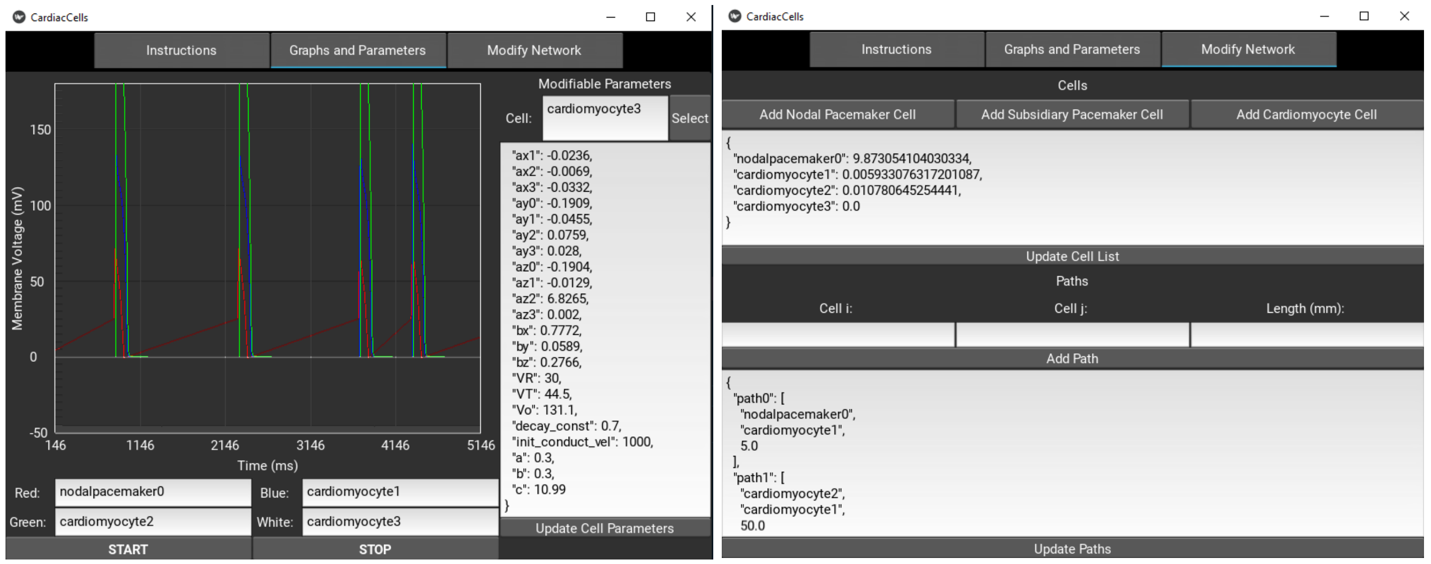
(b) HA to relay action potential of cell i to cell j.


$Idle$ $v_j^{out} = 0$
$Start_j$?
$[ap_i = q_2 \vee ap_j = q_3]$ End_j !
$Relay_j$ $v_j^{out} = v_j(t - \delta_{ji})$

(c) HA to relay action potential of cell j to cell i.

Figure 6: Path Hybrid Automata model. Model and figure from [2] supplementary material.

All of these models were implemented in Python 3.7 using the numpy library. A GUI was also developed using kivy and kivy_garden to compliment these models and to make it easier to change parameters (Figure 7).



```
"ax1": -0.0236,
"ax2": -0.0069,
"ax3": -0.0332,
"ay0": -0.1909,
"ay1": -0.0455,
"ay2": 0.0759,
"ay3": 0.028,
"az0": -0.1904,
"az1": -0.0129,
"az2": 6.8265,
"az3": 0.002,
"bx": 0.7772,
"by": 0.0589,
"bz": 0.2766,
"VR": 30,
"VT": 44.5,
"Vo": 131.1,
"decay_const": 0.7,
"init_conduct_vel": 1000,
"a": 0.3,
"b": 0.3,
"c": 10.99
}
```

```
{
 "nodalpacemaker0": 9.873054104030334,
 "cardiomyocyte1": 0.005933076317201087,
 "cardiomyocyte2": 0.010780645254441,
 "cardiomyocyte3": 0.0
}
```

```
{
 "path0": [
  "nodalpacemaker0",
  "cardiomyocyte1",
  5.0
 ],
 "path1": [
  "cardiomyocyte2",
  "cardiomyocyte1",
  50.0
```

Figure 7: Image of the GUI developed, showing the ability of the developed app. On the left is one tab that allows the plotting of the simulated cells and the editable parameters of one of the cells. On the right is a second tab that allows for the addition and modification of cells and the paths between them.

IV. Results

The implementation of the model seemed to work. The cells when linked together through the path model were able to stimulate each other and the resulting action potentials resembled their real-life counterparts. However, the adjustable parameters of each cell seemed to have a rather large effect. For example, the decay factor of signals propagated between cells had a major effect on the maximum voltage of the subsidiary pacemaker and cardiomyocyte cells. Furthermore, the time steps taken within the simulation had an effect on runtime. Smaller time steps resulted in a slower simulation but resulted in more accurate ranges of voltages.

Furthermore, there seemed to be a few bugs that need to be solved in the GUI implementation. Variability in order of input and order of running the cells versus the paths in updates for each model step resulted in different results. This variability and its possible sources are discussed below.

It should also be noted that the current implementation ran slower than a real-time model, running at about 1/18th the speed required for a real-time simulation in a simple 3-cell setup. Furthermore, more cells and paths added to the network resulted in a slower model. This is further discussed in the discussion and future direction.

V. Discussion

As stated, these models and implementation seem like an accurate and dynamic emulation of the heart. It should be noted that all of the action potentials stimulated here have all been shifted to a baseline of 0, so that would need to be adjusted if this was actually used for pacemaker device testing. Furthermore, at the time of this paper submission, the parameters were not well adjusted for many of the cells, meaning that the interaction between cells was also not well tested. Ai et al. in [2] suggest a wide array of parameters for different sections of the heart and the corresponding cells within those sections in their supplemental material that should be tested.

On a similar note, the decay constant that determined the effect of one cell to another was not easily changeable, unique, and its implementation would need to change to be path dependent instead of cell dependent to be more dynamic. This one parameter created “spikes” in the upstroke phase of some cardiomyocytes and subsidiary pacemakers. Furthermore, this makes sense because the derivatives of each component, v_x, v_y, and v_z, in the q1 phase largely depended on the external voltages applied to the cell, so any decay constant that was not well-tuned could drastically change that upstroke and therefore, the entire action potential.

In the results, the effect of the time step length had a large variability in the model. As stated above, the larger time steps resulted in a faster, but less accurate model. This makes sense as larger time steps would mean that each voltage change between iteration of the model would be larger. In essence, the resolution of the model would be decreased and therefore, less accurate results would be achieved. Ai et al. in [2] mentioned they used time steps of 0.01 ms in their implementation. Upon testing of this model, it seems that around 1 ms of time resolution seemed to be sufficient in reducing large inaccuracies that could arise from model implementation.

The variability of output due to input order and general usage in the GUI implementation seems to be an issue related to errors within the implementation itself. At the time of this paper submission, these bugs are likely still present. However, it is possible that some of these variations are due to the variability of the Nodal Pacemaker Cell Model implementation that uses the power spectrum of normal heart rates to create natural variations seen in [4]. This is unlikely to be the reason behind all of the strange and uncharacterized outputs but could be a potential source for some.

As mentioned above, this implementation of these models ran somewhat slow and could not run in real-time. Part of this issue is due to the lack of multicore abilities in Python. This is due to Python's Global Interpreter Lock (GIL) which means that Python is not "thread safe." The result of this is that Python only runs a single program on a single core, which means that while multithread applications may get improvements in runtime, the improvement is quite marginal. A future implementation of these models in a multicore multithread-capable language like C++ might significantly improve runtime with the ability to generate multiple threads that can run on multiple cores to update cell and path states.

Finally, for a future direction besides a more multithread-friendly implementation, the effect of the order of updating cells and paths should be investigated. In the current implementation, all of the cells are updated followed by all of the paths in an arbitrary fashion. The order has a rather small effect theoretically in the short-term simulations. However, it can be speculated that these small changes due to order may build up over time.

VI. Addendum: Code

See the attached Project_Code.zip file containing of the code written for this project. A virtual environment with the required libraries should be included and should be used to run the "main.py" script that runs the program.

References

[1] *Towards the Emulation of the Cardiac Conduction System for Pacemaker Validation | ACM Transactions on Cyber-Physical Systems*. (2018). ACM Transactions on Cyber-Physical Systems. https://dl.acm.org/doi/10.1145/3134845

[2] Ai, W., Patel, N. D., Roop, P. S., Malik, A., Andalam, S., Yip, E., Allen, N., & Trew, M. L. (2018). A Parametric Computational Model of the Action Potential of Pacemaker Cells. *IEEE Transactions on Biomedical Engineering*, *65*(1), 123–130. https://doi.org/10.1109/tbme.2017.2695537

[3] P. Ye, E. Entcheva, S. A. Smolka, and R. Grosu, "Modelling Excitable Cells Using Cycle-Linear Hybrid Automata," IET Systems Biology, vol. 2, no. 1, pp. 24 – 32, Jan. 2008.

[4] McSharry, P. E., Clifford, G. D., Tarassenko, L., & Smith, L. A. (2003). A dynamical model for generating synthetic electrocardiogram signals. *IEEE Transactions on Biomedical Engineering*, *50*(3), 289–294. https://doi.org/10.1109/tbme.2003.808805