AM5016

NUMERICAL METHODS IN BIOMEDICAL ENGINEERING

PROJECT REPORT

ON

"Comparative Analysis of Numerical methods for cardiovascular Action potential modelling"

ARUNACHALAM U AM23M020

DINESH KUMAR M AM23M022

KAVIYARASAN P R AM23M025

POOJA V AM23D060

CONTENTS

CHAPTER 1

INTRODUCTION

Cardiac action potential modelling is a vital area of study within the field of cardiology and computational biology. It plays a central role in comprehending the electrical behaviour of the heart, offering critical insights into its physiology, pathophysiology, and clinical applications. The action potential of cardiac cells represents the electrical events responsible for the initiation and propagation of heartbeats. Understanding this complex waveform is essential for unravelling the mechanisms that underlie cardiac rhythms, arrhythmias, and other cardiac disorders. Accurate modelling of cardiac action potentials provides a valuable framework for investigating the effects of various drugs, genetic mutations, and disease conditions on cardiac electrophysiology.

Computational models of cardiac action potentials have become indispensable tools in cardiac research and clinical practice. These models help predict the effects of drugs and interventions, aid in the design of medical devices such as pacemakers and defibrillators, and support the development of novel anti-arrhythmic therapies. However, cardiac action potential modelling is not without its challenges. The heart's electrophysiology is highly nonlinear, and the action potential waveform is influenced by various factors, including ion channel kinetics, membrane potential, and the cell's state. Numerical methods are essential for simulating these dynamics accurately.

The choice of numerical methods for modelling cardiac action potentials is a critical decision, as it affects the accuracy and computational cost of simulations. Researchers often choose between Implicit and Explicit methods, each with its own advantages and limitations. Implicit methods are known for their stability but can be computationally demanding, while Explicit methods are computationally more efficient but may require smaller time steps for

stability in certain scenarios. This project seeks to address the challenges of modelling cardiac action potentials by developing a detailed Simulink model and implementing both Implicit and Explicit methods, specifically using the Runge-Kutta 4 (RK4) and Euler algorithms. The goal is to explore the strengths and weaknesses of these numerical methods in accurately capturing the dynamics of cardiac action potentials.

By comparing and analyzing the performance of these methods, we aim to provide valuable insights for researchers and clinicians in the field of cardiology. The results of this study will help inform the selection of numerical methods for more precise and efficient simulations of cardiac action potentials, ultimately contributing to advancements in cardiac research, drug development, and clinical treatment strategies.

1.1 OBJECTIVES

- Create a detailed cardiovascular model using Simulink that accurately represents the action potential of cardiac cells.
- Implement and validate the model for action potential generation and propagation under various conditions and scenarios.
- Develop numerical solvers for both Implicit and Explicit methods, incorporating the Explicit RK4, Implicit RK4, Forward Euler, Backward Euler and Livermore Solver for ordinary differential equations (LSODA) algorithms, to simulate the action potential.
- Investigate the efficiency and accuracy of each numerical method in capturing the dynamic behaviour of the cardiovascular system.
- Analyze and compare the computational resources required for each numerical method, including simulation time and stability.
- Present the results using graphical representations, highlighting the differences in action potential waveforms and computational performance.

- Discuss the implications of the findings on the choice of numerical methods for cardiovascular modelling and its impact on research and clinical applications.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we delve into the existing body of knowledge and research relevant to the topics of cardiovascular modelling and numerical methods for simulating cardiac action potentials. A thorough literature review is essential to understand the current state of the field and to identify gaps, trends, and key findings. This chapter is organized as follows:

- Historical Perspective on Cardiac Action Potential Modelling: This section provides a historical context for cardiac action potential modelling. Early pioneers in the field, such as Hodgkin and Huxley, laid the groundwork for understanding the electrical behaviour of cardiac cells. Their groundbreaking research in the mid-20th century led to the formulation of the Hodgkin-Huxley model, a cornerstone in the study of action potentials. Subsequent developments, including the advent of computers and advanced experimentation techniques, have contributed to the evolution of our knowledge in cardiac electrophysiology.
- Cellular Electrophysiology of Cardiac Action Potentials: In this section, we delve into the intricate cellular processes governing cardiac action potentials. The action potential waveform consists of distinct phases, each with specific ion channels and currents involved. The depolarization phase, for example, is primarily driven by the sodium current, while the repolarization phase is influenced by potassium currents. Understanding the role of each ionic current and their interactions is crucial for accurate action potential modelling.
- Computational Models of Cardiac Action Potentials: Cardiac action potentials can be simulated using computational models that range from simple single-cell models to complex whole-heart models. These models incorporate equations that describe ion channel kinetics, membrane

potential, and cell geometry. Single-cell models focus on the behaviour of individual cardiac cells, while tissue and whole-heart models extend this understanding to tissue-level and organ-level interactions. Researchers choose the appropriate level of complexity based on their research objectives.

- Numerical Methods in Cardiac Electrophysiology: Numerical methods play a central role in solving the differential equations that describe cardiac electrophysiology. These methods are essential for simulating the dynamic behaviour of action potentials over time. Researchers leverage numerical techniques to approximate solutions and study how electrical impulses propagate through cardiac tissues. The choice of numerical methods has a significant impact on the accuracy and efficiency of simulations.
- Implicit and Explicit Methods in Computational Cardiology : In computational cardiology, two broad categories of numerical methods are commonly used: Implicit and Explicit methods. Implicit methods are known for their stability and are particularly useful for stiff differential equations commonly encountered in cardiac modelling. In contrast, Explicit methods are computationally efficient but may require smaller time steps to maintain stability. The choice between these methods depends on the specific modelling objectives and trade-offs between stability and computational cost.
- Simulation Software and Tools : Simulation software and tools are essential for creating and running cardiac action potential models. Software platforms like Simulink, NEURON, and others provide the infrastructure for building and simulating models. These tools offer user-friendly interfaces and powerful numerical solvers that enable researchers to experiment with different scenarios and study the behavior of action potentials.

- Gaps and Challenges in Current Research : Current research in cardiac electrophysiology faces several challenges. Models, though sophisticated, may still lack accuracy in certain conditions or require extensive computational resources. Scaling these models to represent larger tissues or entire hearts can be challenging. Additionally, the incorporation of patient-specific data and individualized modelling remains an ongoing challenge in the field.

In summary, this chapter has provided a comprehensive review of the literature related to cardiac action potential modelling, starting from its historical origins and progressing to the present. Understanding the cellular electrophysiology, computational models, numerical methods, and challenges in the field is fundamental to the context of this project and its objectives. The literature review serves as the foundation upon which this research is built.

CHAPTER 3

NOBLE MODEL

The early development of mathematical models for cardiac cells is essential in the context of cardiac electrophysiology. One notable model, the 1962 model by Denis Noble, stands as a pioneering example in this field. It is worth noting that this model was an adaptation of the original Hodgkin-Huxley model, originally designed for the squid axon.

Noble's Model (1962): Noble's 1962 model successfully described the long-lasting action and pace-maker potentials observed in Purkinje fibers of the heart. This model was instrumental in bridging the gap between the Hodgkin-Huxley model and the electrophysiological behaviour of cardiac cells.

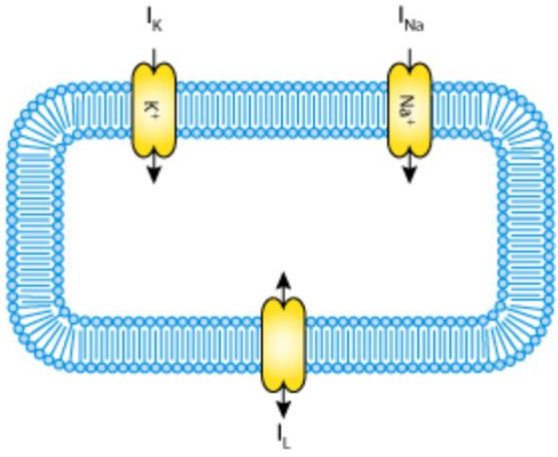

Fig 3.1 Noble's model

3.1 Noble's Model

- A significant limitation of Noble's model is that it excluded calcium currents, an essential component in the cardiac action potential. It should be noted that at the time of this model's development, calcium currents had not yet been discovered. Consequently, they were omitted from the model.

- Within the model, an intriguing observation was made: in order to make the model work effectively, it was necessary to substantially extend the voltage range of the sodium current. This adjustment allowed the sodium current to fulfill the dual role of representing both sodium and calcium channels during the plateau phase of the action potential.

3.2 Sodium Current Adaptation

In Noble's model, the sodium current was adjusted to serve as a surrogate for both sodium and calcium channels during the plateau phase of the action potential. This adaptation allowed the model to function more effectively in the absence of explicitly defined calcium currents.

Experimental Predictions: Noble's model generated two crucial experimental predictions:

- Sodium channels in the heart may exhibit quantitative differences from their counterparts in neurons.
- In addition to sodium channels, other inward current-carrying channels, including calcium channels, must exist in cardiac cells.

The Noble (1962) model represents a significant milestone in the evolution of cardiac electrophysiology models. It demonstrated the importance of adapting the Hodgkin-Huxley framework to cardiac tissue while highlighting the limitations associated with the omission of calcium currents.

Subsequent experimental research has validated both of Noble's predictions, leading to a more comprehensive understanding of the electrophysiological behaviour of cardiac cells.

3.3 NOBEL MODEL FOR CARDIAC ACTION POTENTIAL

The Hodgkin-Huxley (HH) model, originally developed for the squid giant axon, has been adapted and extended to describe the electrical behavior of cardiac myocytes, specifically ventricular myocytes. Cardiac myocytes have some distinct ion channels and characteristics compared to neurons, so the HH model for cardiac myocytes includes modifications to reflect these differences. Here's an overview of the HH model adapted for cardiac myocytes:

- Membrane Potential (V): This represents the electrical potential across the cell membrane and is described by an ordinary differential equation (ODE).

$$\frac{\mathrm{d}}{\mathrm{d\,time}}(V) = \frac{-(\mathrm{i\_Na}+\mathrm{i\_K}+\mathrm{i\_Leak})}{\mathrm{Cm}}$$

- Sodium Current (I_Na): Sodium channels are described by the following equations:

$$\mathrm{g\_Na} = m^3 h \mathrm{g\_Na\_max}$$

$$\mathrm{i\_Na} = (\mathrm{g\_Na} + 140)(V - \mathrm{E\_Na})$$

- The activation (m) and inactivation (h) gates of sodium channels are voltage-dependent and follow ODEs. α and β functions are voltage-dependent rate constants.

$$\mathrm{alpha\_h} = 170 e^{\frac{-V-90}{20}}$$

$$\mathrm{alpha\_m} = \frac{100(-V-48)}{e^{\frac{-V-48}{15}}-1}$$

$$\mathrm{beta\_h} = \frac{1000}{1+e^{\frac{-V-42}{10}}}$$

$$\mathrm{beta\_m} = \frac{120(V+8)}{e^{\frac{V+8}{5}}-1}$$

$$\frac{\mathrm{d}}{\mathrm{d\,time}}(h) = \mathrm{alpha\_h}(1-h) - (\mathrm{beta\_h}h)$$

$$\frac{\mathrm{d}}{\mathrm{d\,time}}(m) = \mathrm{alpha\_m}(1-m) - (\mathrm{beta\_m}m)$$

- Potassium Currents (I_K, I_to, I_K1, I_Kr, I_Ks): Different types of potassium channels are represented, each with its own activation and inactivation gates.
- Capacitance (Cm): The membrane capacitance is used to relate changes in voltage to current.

The ODEs for the activation and inactivation gates (m, h, n, etc.) follow voltage-dependent rate equations similar to those used in the original model but are adjusted to reflect the specific ion channels found in cardiac myocytes. The ionic currents are described using Ohm's law and depend on the conductance of the channels, the voltage, and the ion equilibrium potentials.

The model for cardiac myocytes is highly detailed and can include additional ion channels and currents to accurately represent the behavior of ventricular, atrial, or specialized cardiac cells. These models have been instrumental in understanding the electrophysiology of the heart and are used for simulating and studying various cardiac phenomena, including action potentials, arrhythmias, and the effects of drugs or mutations on cardiac function.

CHAPTER 4

SIMULATING CARDIAC ACTION POTENTIAL

The Hodgkin-Huxley (HH) model, originally developed to describe the electrical behavior of neurons, has been adapted for use in cardiac myocytes to explain the generation of action potentials in the heart. This model provides a detailed description of the ionic currents involved in cardiac cell excitation. Here's a brief overview of the HH model for cardiac myocytes:

4.1 Membrane Potential Dynamics

The HH model focuses on the changes in the membrane potential (V) of a cardiac myocyte over time, explaining how it depolarizes and repolarizes to produce an action potential.

4.2 Ion Channels

The HH model includes descriptions of various ion channels that play a crucial role in cardiac cell electrophysiology, particularly sodium (Na+), potassium (K+), and calcium (Ca2+) channels. These ion channels have different conductances, voltage-dependent gating properties, and ion reversal potentials.

4.3 Gating Variables

To account for the voltage-dependent behavior of ion channels, the HH model introduces gating variables (m, h, and n for Na+ and m for K+) that represent the fraction of channels in their open or closed states. These gating variables change over time based on the membrane potential.

4.4 Differential Equations

The HH model uses a set of differential equations to describe the dynamics of the membrane potential and gating variables. These equations specify how the ion currents are affected by the gating variables and the membrane potential.

4.5 Action Potential Phases

The HH model can explain the phases of the cardiac action potential, including rapid depolarization due to Na+ influx (Phase 0), a plateau phase characterized by Ca2+ influx (Phase 2), and repolarization due to K+ efflux (Phase 3). The resting membrane potential corresponds to Phase 4.

4.6 Simulation and Analysis

To simulate the action potential in cardiac myocytes, you can numerically solve the differential equations using numerical integration methods. This allows you to observe the changes in membrane potential and gating variables over time.

4.7 Adaptations for Cardiac Myocytes

The HH model has been adapted and extended to account for the unique features of cardiac myocytes, including various ionic currents specific to the heart and different cell types (atrial, ventricular, or pacemaker cells). Different forms of the model, such as the "cardiac" HH model, have been developed to better capture the behavior of cardiac cells.

The Hodgkin-Huxley model for cardiac myocytes is a fundamental framework for understanding the electrical activity of the heart. It has been instrumental in advancing our knowledge of cardiac electrophysiology and has contributed to the development of treatments for cardiac arrhythmias and other heart-related conditions.

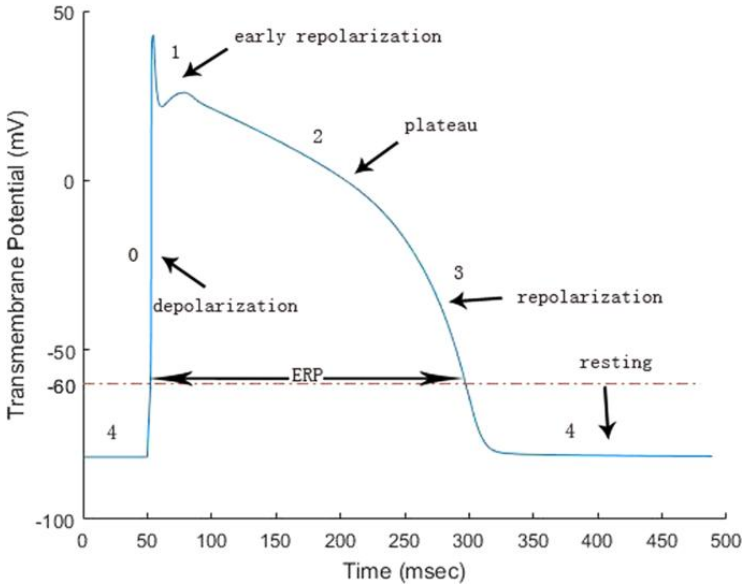

Fig 4.1 Cardiac myocytes action potential

The phases of cardiac action potential:

- Phase 0: Rapid depolarization
- Phase 1: Initial repolarization
- Phase 2: Plateau phase
- Phase 3: Rapid repolarization
- Phase 4: Resting potential.

4.8 SIMULATION

1. Set Up Simulink:
   - Open MATLAB and then open Simulink.
   - Start a new model or open an existing one.
2. Create Subsystems for Each Ionic Current: Each type of ion channel (Na+, K+, Ca2+, etc.) can be represented by a subsystem block that models its dynamics:
   - Use the "Subsystem" block to encapsulate the equations governing each ionic current.

- Inside each subsystem, implement the equations using "Function" blocks, "Gain" blocks, "Integrator" blocks, etc., depending on what's needed for the mathematical representation.

3. Implement the Differential Equations: The action potential involves solving a set of differential equations:

- Use "Integrator" blocks to represent the time evolution of the membrane potential.
- Model the voltage-gated ion channels with dynamic equations using "Function" blocks that change their conductance based on the voltage and time.

4. Add Blocks for Membrane Potential: Create a block or a subsystem that calculates the overall membrane potential by summing the contributions from all the ion channels and capacitive currents.
5. Configure Parameters: Set the parameters for each block to match the physiological data from cardiac cells, such as the maximal conductance, reversal potentials, and rate constants.
6. Input Stimulus: Create an input block to simulate the external stimulus that triggers the action potential, if necessary. This could be a "Pulse Generator" or "Step" function.
7. Connect the Blocks: Wire all the blocks together to reflect the interactions between the different ionic currents and the membrane potential.
8. Run the Simulation: After setting up the blocks, run the simulation to observe the action potential. Adjust the model parameters and initial conditions as necessary to ensure the simulation results align with physiological data.
9. Analyze the Results: Use scopes or data display blocks to visualize the action potential waveform and other relevant variables over time.

4.8.1 NOBLE MODEL

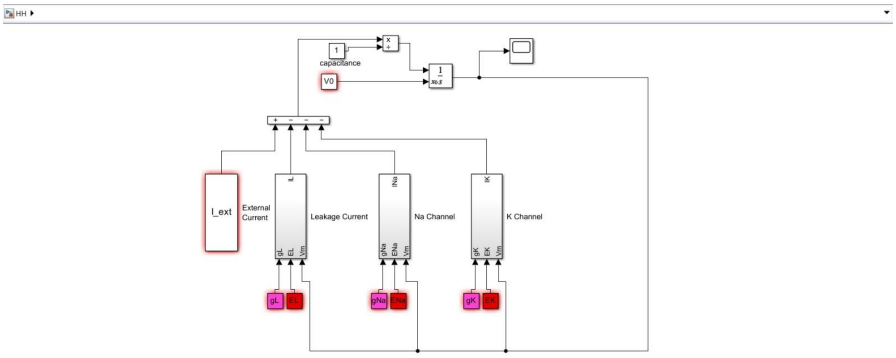

Fig 4.2 Modified HH model

4.8.2 Na Channel

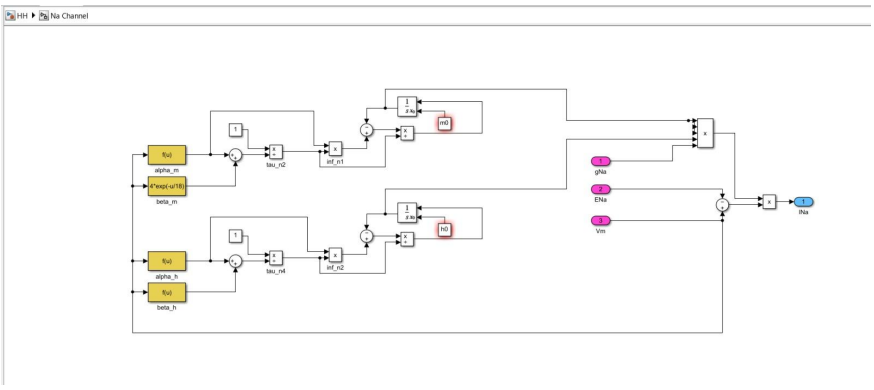

Fig 4.3 Na channel block diagram

4.8.3 K Channel

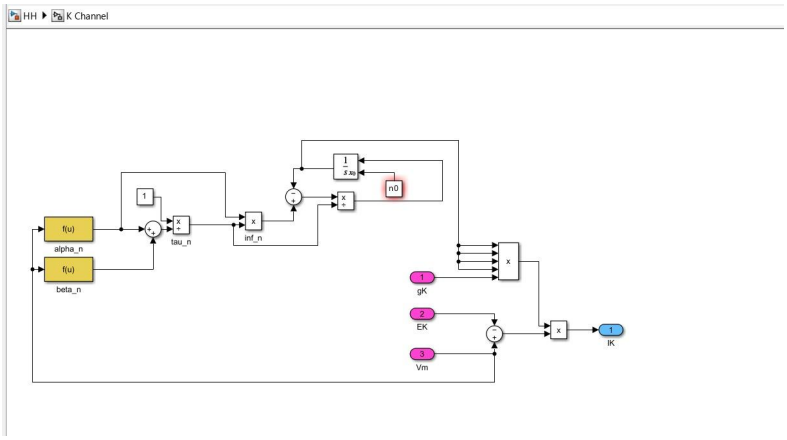

Fig 4.4 K Channel

4.8.4 Leakage current

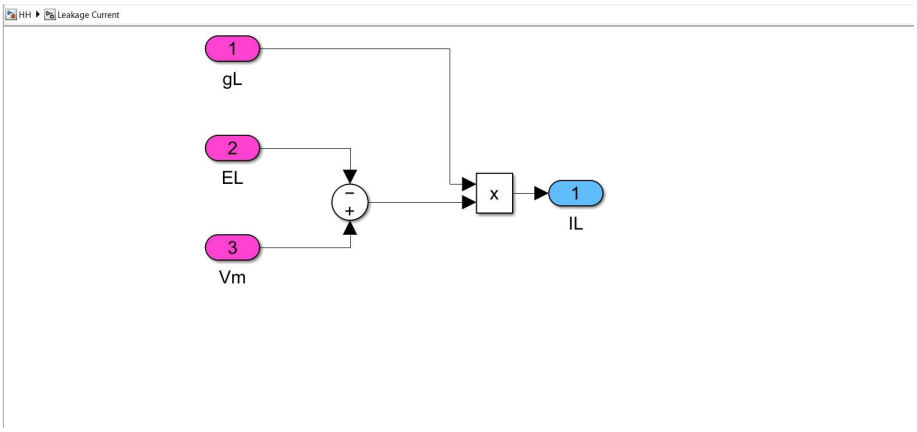

Fig 4.5 Leakage current

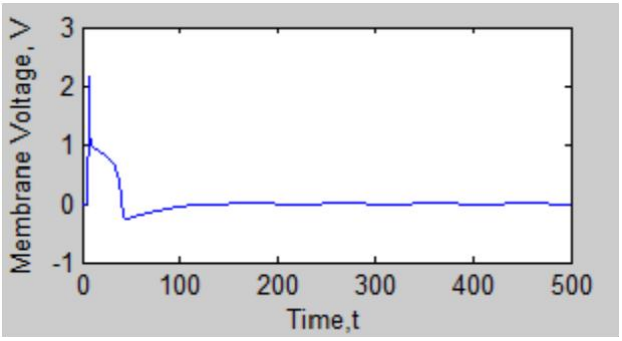

Fig 4.6 Simulated graph

CHAPTER 5

METHODOLOGY

5.1 FORWARD EULER'S METHOD

Euler's method is essentially a first-order approximation of the solution. It can be viewed as a simple application of the first-order Taylor series expansion. In this method, the derivative at the current point approximates the change over a small step size, providing an incremental estimate of the solution.

$$y_{n+1} = y_n + h \cdot f(x_n, y_n)$$

5.2 BACKWARD EULER'S METHOD

The Backward Euler method is a numerical technique for solving ordinary differential equations (ODEs). It is an implicit method, meaning that it computes the stage values indirectly by solving algebraic equations at each time step. This method is widely used in numerical simulations, especially for stiff ODEs.

$$\frac{y_n - y_{n-1}}{\Delta t} = f(t_n, y_n)$$

or

$$\frac{y_{n+1} - y_n}{\Delta t} = f(t_{n+1}, y_{n+1})$$

5.3 IMPLICIT RK4 METHOD

Implicit Runge-Kutta 4 (IRK4) is a numerical method for solving ordinary differential equations (ODEs). It is an implicit method, meaning that it uses an iterative approach to compute stage values at each time step. IRK4 is a

fourth-order method, which means it offers relatively high accuracy in approximating the solution of ODEs. IRK4 is particularly useful for solving stiff ODEs, where explicit methods may require very small time steps to maintain stability. IRK4 can handle stiff systems more efficiently.

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i,$$

where

$$k_i = f\left(t_n + c_i h,\ y_n + h\sum_{j=1}^{s} a_{ij} k_j\right), \quad i = 1, \ldots, s.$$ [20]

5.4 EXPLICIT RK4 METHOD

Explicit Runge-Kutta 4 (ERK4) is a numerical method for solving ordinary differential equations (ODEs). It belongs to the family of explicit methods, meaning that it computes stage values directly at each time step. ERK4 is a fourth-order method, which provides relatively high accuracy in approximating the solution of ODEs.

$$\begin{aligned}
k_1 &= f(t_n, y_n), \\
k_2 &= f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right), \\
k_3 &= f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right), \\
k_4 &= f(t_n + h, y_n + hk_3).
\end{aligned}$$

5.5 LIVERMORE SOLVER FOR ORDINARY DIFFERENTIAL EQUATIONS (LSODA)

LSODA is based on the Adams and BDF (Backward Differentiation Formula) methods for solving ODEs. The method automatically switches between these two approaches to optimize the solution based on the stiffness of the problem. The main idea behind LSODA is to combine the strengths of both methods, ensuring accuracy and efficiency.

CHAPTER 6

RESULTS AND DISCUSSION

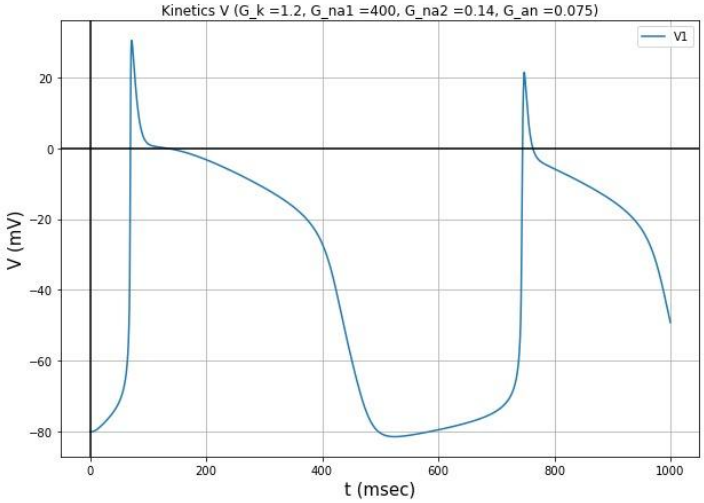

Fig 6.1 Approximate Analytical Plot

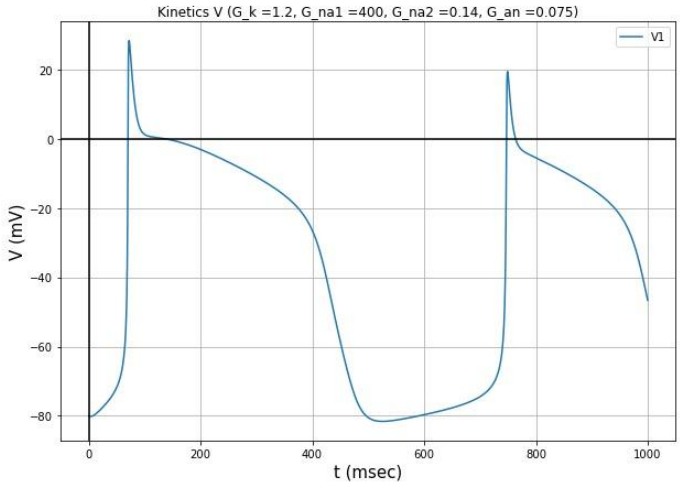

Fig 6.2 Forward Euler Method

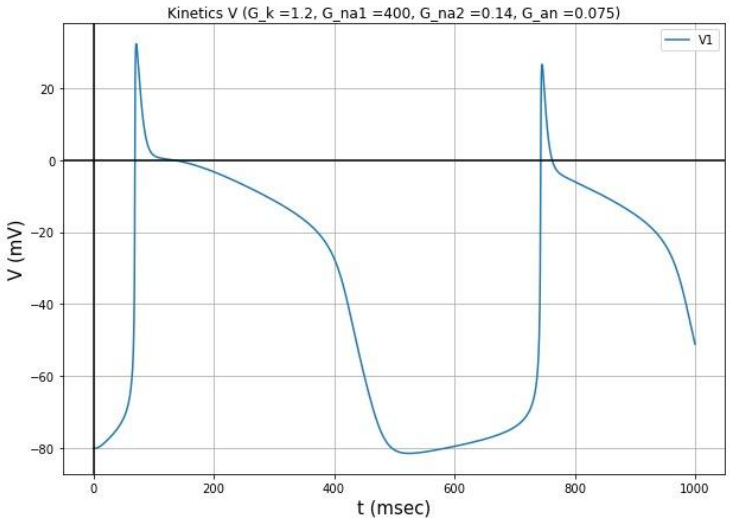

Fig 6.3 Backward Euler Method

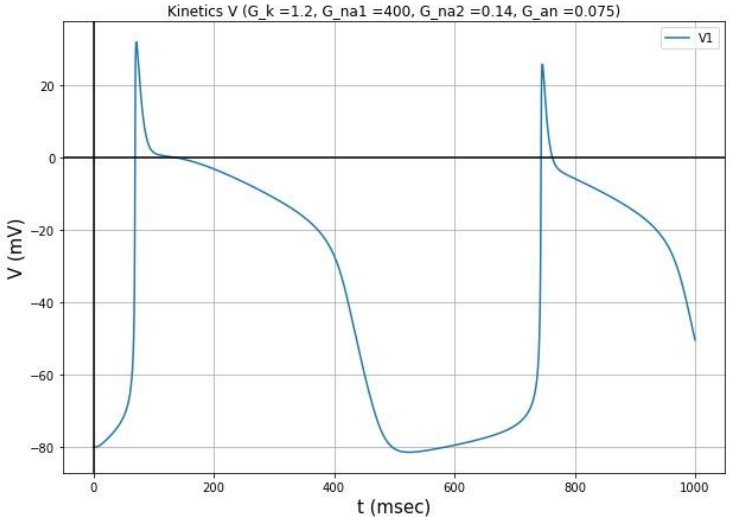

Fig 6.4 Implicit RK4 Method

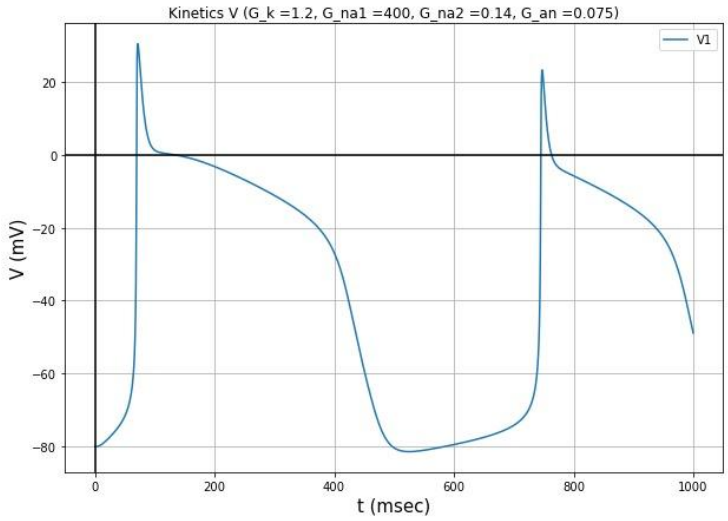

Fig 6.5 Explicit RK4 Method

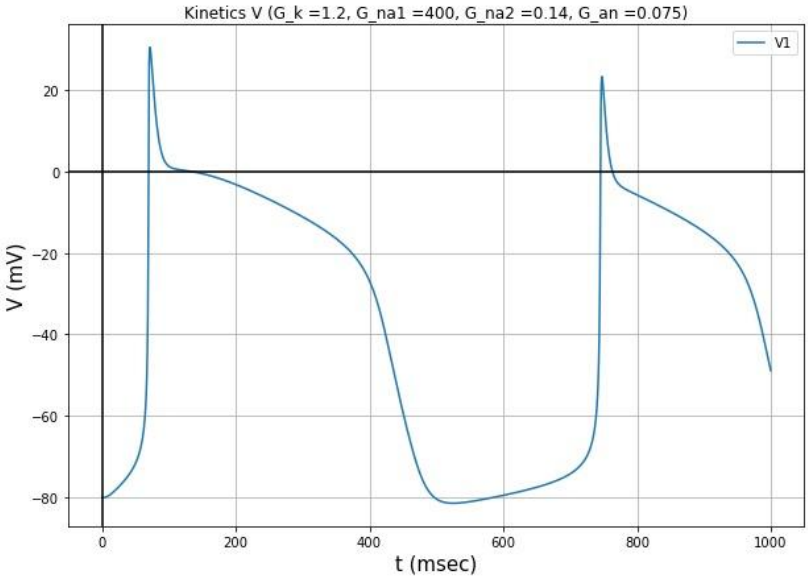

Fig 6.6 LDOSA Algorithm

The action potential (V) is plotted for the cardiac myocytes. The graph shows the potential over time, and it seems to exhibit the expected behavior of an action potential with depolarization followed by repolarization. The plot provides a clear visual comparison of the action potential for this specific method.

- Phase 0 - Rapid Depolarization: The initial upstroke of the action potential is primarily due to the opening of fast voltage-gated Na+ channels, similar to the HH model's depiction of neuronal depolarization. In cardiac cells, this corresponds to rapid depolarization as Na+ ions enter the cell.
- Phase 1 - Initial Repolarization: In cardiac cells, this corresponds to a brief partial repolarization. It is mainly due to the inactivation of the Na+ channels and the opening of transient outward K+ channels.
- Phase 2 - Plateau Phase: Unlike neurons, cardiac cells have a plateau phase due to the balance between slow inward Na+ current and outward K+ current. This is where the cardiac-specific modifications to the HH model are most apparent.
- Phase 3 - Rapid Repolarization: This phase is due to the closure of Na+ channels and the opening of rectifying K+ channels that increase K+ efflux.
- Phase 4 - Resting Potential: The membrane potential returns to and stabilizes at the resting level due to the high permeability of K+ channels relative to Na+ channels, similar to the resting potential explained by the model for neurons.

6.1 ODE System and Parameters

The ODE system describes the kinetics of ion channels and the membrane potential. It involves four variables: V, m, h, and n. The parameters are G_k, G_na1, G_na2, and G_an.

6.2 Integration Methods

The code implements five different integration methods to solve the ODE system:

- Euler's method (euler)
- Explicit Runge-Kutta (RK4) method (explicit_rk4)
- Backward Euler method (backward_euler)
- Implicit Runge-Kutta (IRK4) method (implicit_rk4)
- SciPy's odeint solver (odeint)

The comparative analysis of integration methods suggests that each method has its strengths and weaknesses. For this specific ODE system, the implicit Runge-Kutta (IRK4) method seems to perform well in terms of accuracy and may be a suitable choice for solving stiff ODEs. However, the choice of the best method depends on the specific characteristics of the ODE system being modeled.

6.3 Performance Measurements

The code measures the performance of these integration methods by recording the execution time for each method's simulation. It also calculates and displays the Root Mean Square Error (RMSE) for the voltage variable (V) and a global RMSE for all state variables combined

Method	RMSE	Global RMSE	Computation Time
FE	2.44221725799268	1.2211879859439723	0.1702265739440918
ERK4	0.6542742988205605	0.32716176808032277	0.7009162902832031
LSODA	0.6568167013526719	0.3284331031988431	0.20209431648254395
BE	1.9694941562646897	0.9848080864642528	1.7819700241088867
IRK4	1.2719308371407967	0.6360071176979982	9.581271171569824

Fig 6.7 COMPARATATIVE STUDY

6.4 Jacobian Matrix for Sparsity pattern

The code calculates and displays the sparsity pattern of the Jacobian matrix. The Jacobian matrix is crucial in solving stiff ODEs as it relates changes in the state variables to their derivatives.

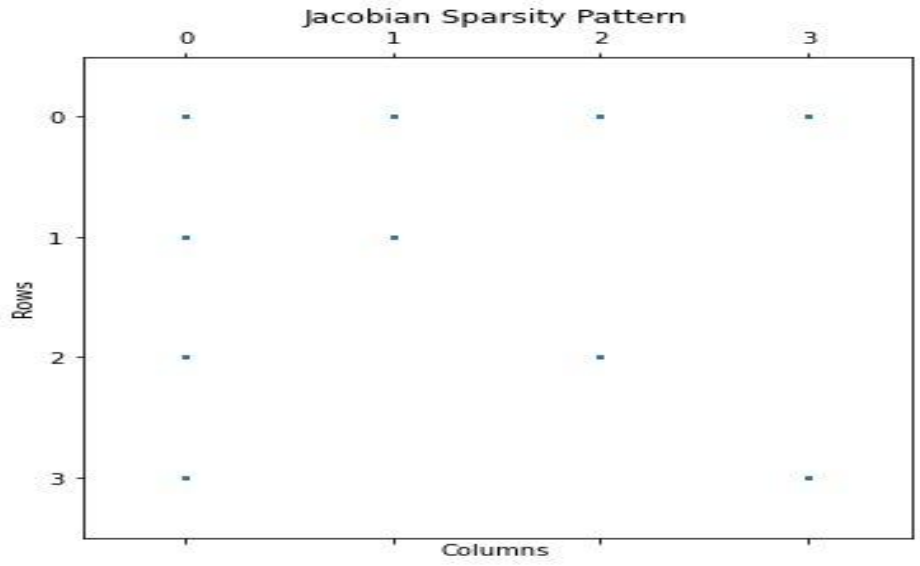

Fig 6.8 Sparsity Pattern

The sparsity pattern of the Jacobian matrix is useful for identifying which state variables are strongly coupled and understanding the stiffness of the ODE system. A sparse Jacobian matrix can be indicative of a less stiff system, while a dense matrix suggests stiffness.

6.5 Parameter Sensitivity

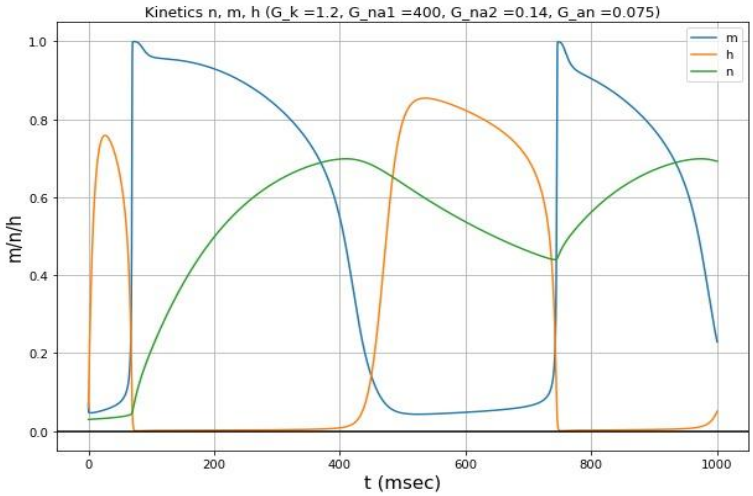

Fig 6.9 Influence of Parameters

The accuracy of the simulations may be sensitive to the choice of parameters (G_k, G_na1, G_na2, G_an). Adjusting these parameters could lead to different simulation results, which may have biological or physical implications.

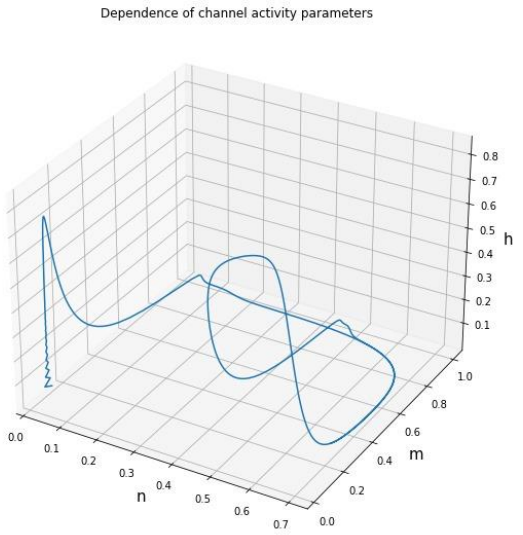

Fig 6.10 Dependence of Channel Activity Parameters

The relationships and interactions among the different gating variables determines the conductance of ion channels in a neuron or other excitable cell. These gating variables, such as m, h, and n are associated with specific ion channels, and they play a crucial role in controlling the flow of ions across the cell membrane.

APPENDIX A

CODE DOCOUMENTATION

```
from scipy.integrate import odeint
import time
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from scipy.optimize import newton
from scipy.optimize import fsolve
from scipy.integrate import ode

def func(w, t):
    V, m, h, n = w
    dm = lambda m, t, V: (0.1*(-V-48)/(np.exp((-V-48)/15)-1))*(1-m) - 0.12*(V+8)*m/(np.exp((V+8)/5)-1)
    dh = lambda h, t, V: 0.17*np.exp((-V-90)/20)*(1-h) - h/(np.exp((-V-42)/10)+1)
    dn = lambda n, t, V: (0.0001*(-V-50)/(np.exp((-V-50)/10)-1))*(1-n) - 0.002*np.exp((-V-90)/80)*n

    dV = lambda V, t, m, h, n: -((G_na1*(m**3)*h + G_na2)*(V - 40)
                + (G_k*np.exp((-V-90)/50)+0.015*np.exp((V+90)/60)+ G_k*n**4)*(V + 100)
                + G_an*(V+60))/12
    return np.array([dV(V,t,m,h,n),dm(m,t,V),dh(h,t,V),dn(n,t,V)])
```

 # Perform Euler method

```
def euler(func, w0, t):
    # Initialize arrays to store the solutions
    num_points = len(t)
    w = np.zeros((num_points, len(w0)))
    w[0] = w0
```

```
    # Perform RK4 integration
    for i in range(num_points - 1):
        h = t[i + 1] - t[i]
        k1 = h * func(w[i], t[i])
        w[i + 1] = w[i] + k1
    return w
    # Perform Explicit

def explicit_rk4(func, w0, t):
    # Initialize arrays to store the solutions
    num_points = len(t)
    w = np.zeros((num_points, len(w0)))
    w[0] = w0
    for i in range(num_points - 1):
        h = t[i + 1] - t[i]
        k1 = h * func(w[i], t[i])
        k2 = h * func(w[i] + 0.5 * k1, t[i] + 0.5 * h)
        k3 = h * func(w[i] + 0.5 * k2, t[i] + 0.5 * h)
        k4 = h * func(w[i] + k3, t[i + 1])
        w[i + 1] = w[i] + (k1 + 2 * k2 + 2 * k3 + k4) / 6

    return w

  # Perform Backward Euler
def backward_euler(func, w0, t):
    # Initialize arrays to store the solutions
    num_points = len(t)
    w = np.zeros((num_points, len(w0)))
    w[0] = w0
    for i in range(num_points - 1):
        h = t[i + 1] - t[i]
        equation = lambda x: x-h * func(x, t[i + 1]) - w[i]

        w[i + 1] = fsolve(equation, w[i])
```

```
    return w
  # Perform Implicit
def implicit_rk4(func, y0, t):
    n = len(t)
    m = len(y0)
    y = np.zeros((n, m))
    y[0] = y0

    for i in range(n - 1):
        h = t[i + 1] - t[i]
        k1 = func(y[i], t[i])

        # Define a function for Newton's method to find the new state
        def equation(y_new):
            k2 = func(y_new, t[i] + h / 2)
            k3 = func(y_new, t[i] + h / 2)
            k4 = func(y_new, t[i] + h)
            y_next = y[i] + (k1 + 2 * k2 + 2 * k3 + k4) * h / 6
            return y[i + 1] - y_new + y_next

        # Use Newton's method to find the new state
        y_new = newton(equation, y[i],maxiter=200)
        y[i + 1] = y_new

    return y

G_k = 1.2
G_na1 = 400
G_na2 = 0.14
G_an = 0.075
initial = (-80,0.07,0.06,0.03)
t = np.linspace(0, 1000, 5000)
start_time = time.time()
```

```
sol = euler(func,initial, t)
FE_time = time.time() - start_time

start_time = time.time()
sol1 = explicit_rk4(func, initial, t)
ERK4_time = time.time() - start_time

start_time = time.time()
sol2 = odeint(func,initial, t)
LSODA_time = time.time() - start_time

start_time = time.time()
sol3 = backward_euler(func, initial, t)
BE_time = time.time() - start_time

start_time = time.time()
sol4 = implicit_rk4(func,initial, t)
IRK4_time = time.time() - start_time

ref = np.mean((sol,sol1,sol2,sol3,sol4),axis=0)

#Action potential plot
fig = plt.figure(figsize = (10,7))
# plt.plot(t, sol4[:, 0], label = 'V')
plt.plot(t, ref[:,0], label = 'V1')
plt.legend()
plt.xlabel('t (msec)', fontsize=15)
plt.ylabel('V (mV)', fontsize=15)
plt.axhline(0,color='black')
plt.axvline(0,color='black')
plt.title(f'Kinetics V (G_k ={G_k}, G_na1 ={G_na1}, G_na2 ={G_na2}, G_an ={G_an})')
plt.grid()
plt.show()
```

#Conductance plot

```
fig = plt.figure(figsize = (10,7))

plt.plot(t, sol1[:, 1], label = 'm')
plt.plot(t, sol1[:, 2], label = 'h')
plt.plot(t, sol1[:, 3], label = 'n')
plt.legend()

plt.xlabel('t (msec)', fontsize=15)
plt.ylabel('m/n/h', fontsize=15)

plt.axhline(0,color='black')
plt.title(f'Kinetics n, m, h (G_k ={G_k}, G_na1 ={G_na1}, G_na2 ={G_na2}, G_an ={G_an})')
plt.grid()
plt.show()

#3D view
fig = plt.figure(figsize = (7, 7))
ax = Axes3D(fig)
ax.plot(sol[:,3], sol[:,1], sol[:,2])
ax.set_xlabel('n', fontsize=15)
ax.set_ylabel('m', fontsize=15)
ax.set_zlabel('h', fontsize=15)
plt.title('Dependence of channel activity parameters')
plt.show()
```

 # Error computation

```
simulated_data = [sol, sol1, sol2, sol3, sol4]
method_names = ['FE', 'ERK4', 'LSODA', 'BE', 'IRK4']
rmse_results = []
global_rmse_results = []

for method, data in zip(method_names, simulated_data):
```

```
    rmse = np.sqrt(np.mean((data[:, 0] - ref[:, 0]) ** 2))  # For V
    global_rmse = np.sqrt(np.mean((data - ref) ** 2))  # For all state variables combined
    rmse_results.append(rmse)
    global_rmse_results.append(global_rmse)

time = [FE_time, ERK4_time, LSODA_time, BE_time, IRK4_time]
# Create a table
print("{:<10} {:<20} {:<20} {:<20}".format("Method", "RMSE", "Global RMSE", "Computation Time"))
print("="*80)
for method, rmse, global_rmse, ti in zip(method_names, rmse_results, global_rmse_results, time):
    print("{:<10} {:<20} {:<20} {:<20}".format(method, rmse, global_rmse, ti))

#sparsity matrix
r = ode(func).set_integrator('vode', method='bdf')
initial = np.array(initial)
r.set_initial_value(initial, t[0])

# Number of variables in the system
n = len(initial)

# Initialize the Jacobian matrix with zeros
jacobian_matrix = np.zeros((n, n))

# Perturbation size for finite differences
epsilon = 1e-6

t = np.linspace(0,1000,5000)

# Loop over time points and compute the Jacobian matrix
for i in range(len(t)):
    if not r.successful():
        break
    w = r.integrate(t[i])
    for j in range(n):
```

```
    w_plus = w.copy()
    w_minus = w.copy()
    w_plus[j] += epsilon
    w_minus[j] -= epsilon
    # Explicitly copy the state variable and apply perturbations
    perturbed_w_plus = np.copy(w)
    perturbed_w_minus = np.copy(w)
    perturbed_w_plus[j] = w_plus[j]
    perturbed_w_minus[j] = w_minus[j]
    derivative = (func(perturbed_w_plus, t[i]) - func(perturbed_w_minus,t[i])) / (2 * epsilon)
    jacobian_matrix[:, j] = derivative
```

 # Plot the sparsity pattern

```
plt.figure(figsize=(8, 6))
plt.spy(jacobian_matrix, markersize=2)
plt.title('Jacobian Sparsity Pattern')
plt.xlabel('Columns')
plt.ylabel('Rows')
plt.show()
```

APPENDIX B

REFERENCES

1) Noble D. (1962). A modification of the Hodgkin-Huxley equations applicable to Purkinje fibre action and pace-maker potentials. Journal of Physiology 160, 317-52.

2) Belhamadia, Y., 2010. Recent numerical methods in electrocardiology. New Development in Biomedical Engineering, pp.151-162.

3) A Historical Perspective on the Development of Models of Rhythm in the Heart Heart Rate and Rhythm, 2011 ISBN : 978-3-642-17574-9.

4) Spear, J.F., Horowitz, L.N., Hodess, A.B., MacVaugh 3rd, H. and Moore, E.N., 1979. Cellular electrophysiology of human myocardial infarction. 1. Abnormalities of cellular activation. Circulation, 59(2), pp.247-256.

5) Göktepe, S. and Kuhl, E., 2009. Computational modeling of cardiac electrophysiology: a novel finite element approach. International journal for numerical methods in engineering, 79(2), pp.156-178.