Draft

JEX, Thomas Pearce

November 2024

1 Introduction

The human heart is a remarkable organ, whose primary function is to pump blood rhythmically and efficiently throughout the body. This rhythmic contraction is coordinated by a complex electrical signaling system that propagates through the heart's tissue, ensuring that each beat is precisely timed. In a healthy heart, these electrical waves follow well-defined pathways that maintain the heart's regular rhythm. However, under certain external conditions, this orderly rhythm can be disrupted, leading to the emergence of spiral waves in the heart that 're-enter' and circulate continuously around the organ. These spirals represent self-sustaining patterns of electrical activity that when combined with the heart's natural periodic rhythm can be associated with life-threatening arrhythmias.

The most important method for analyzing these cardiac arrhythmia-causing spirals is undoubtedly with mathematical modeling. The FitzHugh-Nagumo (FHN) equation is effective for modeling this medical area, as it provides a powerful framework for studying wave propagation, including the formation and dynamics of spiral waves in cardiac tissue. With its dynamic excitation and recovery parameters, the FHN model allows us to simulate the mechanisms underlying spiral formation and their implications for heart health.

In this dissertation, I will first paint a picture of how a normal, healthy heart's electrical activity should be, followed by an extensive exploration of spiral wave phenomena, emphasizing their detrimental impact on cardiac function. This will be done using ordinary and partial differential equations, incorporating spatial diffusion, a critical factor for simulating the spread of electrical signals across cardiac tissue. In addition, these chapters include visualizations and animations to demonstrate how perturbations can lead to the emergence of spirals.

Beyond deterministic models, I'll then shift focus to the role of randomness and noise in cardiac dynamics. Nature is never perfect, and heart cells are no exception, as realistic cardiac behavior is influenced by a variety of stochastic processes, including random fluctuations at the cellular level and external perturbations. This will consist of incorporating white and colored noise via the Ornstein–Uhlenbeck process, along with Chaos Theory to study the impact of randomness on wave propagation and spiral dynamics. These stochastic perspectives bridge the gap between idealized perfect models and the variability observed in biological systems, providing an incredibly realistic depiction of how this phenomenon could occur in nature.

In summary, this dissertation combines mathematical modeling, computational simulations, and stochastic analysis to study the behavior of spiral waves in the heart. By utilizing the FitzHugh-Nagumo equation, it aims to deepen our understanding of the mechanisms driving cardiac arrhythmias and provide a foundation for future research into more realistic and biologically informed models.

2 But what is a healthy heart?

While the basis of this project is focused solely on the damaging effects of spirals and biological randomness on the heart, I find it helpful to illustrate an idea of what the heart should look like.

2.1 Normal Electrical Activity in the Heart

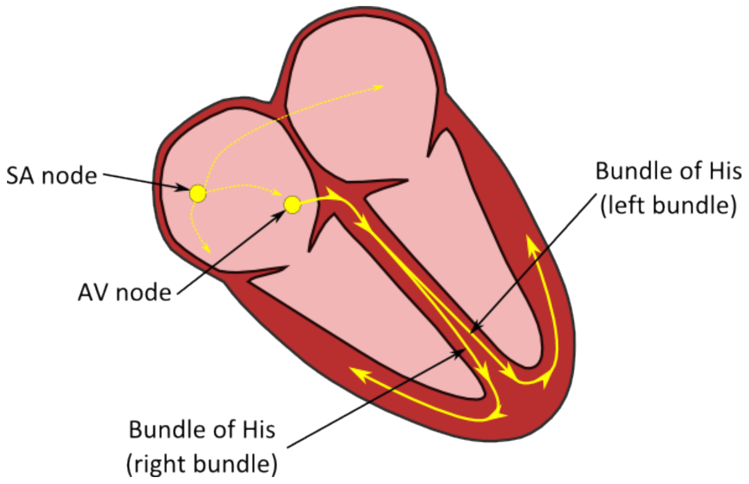
Normal electrical activity in the heart is highly organized and complicated, and one could write up an entire 40 page project on it. Since anatomy is not this report's focus, I will simplify it slightly. Here's a brief description of how it works:

2.1.1 Sinoatrial (SA) Node - The Heart's Natural Pacemaker

The *sinoatrial (SA) node*, located in the top left atrium, generates the initial electrical impulse. This impulse is often called the *"pacemaker signal"* and occurs at regular intervals (periodic).
The SA node's rhythmic firing sets the pace for the entire heart, controlling its overall rate and rhythm.

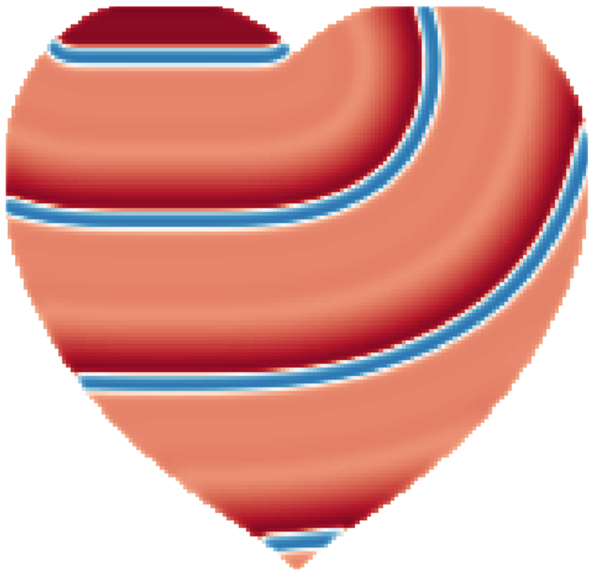
2.1.2 Wave Propagation Through the Atria

The electrical impulse spreads across the atria (upper chambers), causing them to contract and push blood into the ventricles (lower chambers). For the rest of this project, I will be more general, focusing more on the heart as a whole.



2.2 Modeling Normal Heart Behavior

The heart, being a three-dimensional organ, is computationally challenging to model. To ensure my computer does not break down, I will simplify the computational complexity by using a two-dimensional vertical slice of the heart to represent the electrical waves propagating through that specific section.



For a better idea of what this looks like visually, here is the animation of the normal heart model animated over time: Click Here:.

2.3 From Normal to Disrupted: Setting the Stage

This animation represents the normal behavior of a healthy heart, with no disruptions such as perturbations, dead cell regions, or external noise. From this point onward, the focus of this project will shift to exploring how various external and internal influences can disrupt this normal behavior, leading to phenomena such as spiral waves.
At the core of this exploration lies the FitzHugh-Nagumo model, a mathematical framework that enables us to simulate and analyze these disruptions and their effects on cardiac dynamics.

3 An Introduction to the FitzHugh-Nagumo Equation

3.1 Historical Context and Application to Spiral Waves

Created by Richard FitzHugh and Jinichi Nagumo, the FitzHugh-Nagumo model was first proposed in the early 1960s as a simplified alternative to the Hodgkin-Huxley equations for neural excitability. While originally developed to progress work on neural systems, the FHN equation has been recently adapted to model spiral wave phenomena in the heart. It's selling point is how the model's simplicity allows us to analyze spiral wave formation with limited computational requirements.

3.2 Simplified FHN Equation and Variable Definition:

$$\frac{du}{dt} = \frac{1}{\epsilon}\left(u^* - \frac{1}{3}(u^*)^3 - v^*\right), \tag{1}$$

$$\frac{dv}{dt} = \epsilon\left(u^* + \beta - \gamma v^*\right). \tag{2}$$

The FitzHugh-Nagumo (FHN) equation focuses on two state variables: the excitatory variable u, and the recovery variable v. These variables represent the electrical excitatory potential, u, and the recovery of the system, v.

3.3 ϵ, β, and γ: The 3 Parameters

The FHN equation has three important parameters, ϵ, β, and γ, each of them influencing the system:

- ϵ: Controls the timescale separation between the fast excitation of u and the slower recovery dynamics of v. A smaller ϵ leads to a higher change of u compared to v.
- β: Determines the threshold for excitation. A lower β means that it's easier for the system (ex. a heart cell) to transition from rest to excited. The higher it is the harder.
- γ: Influencing how the recovery variable v dampens the excitatory behavior.

Scientists typically choose $\epsilon = 0.3$, $\beta = 0.7$, and $\gamma = 0.5$ for the FHN equation, as these specific variables ensure that a formed spiral is as simple as possible (ex. no tip movement). A simple spiral from these parameters ensure that we can carefully analyze the system as successfully as possible.

4 Modelling The FitzHugh-Nagumo ODE

Starting with an ordinary differential equation (ODE), the FHN model is defined as a system of coupled ODEs describing the dynamics of two state variables, u- (excitatory variable) and v (recovery variable).

These ODEs represent the rate of change of these variables over time and they are a good representation of the underlying processes driving excitability and oscillations in heart tissue. While spatial coordinates (locations) are not a factor here, we can still determine equilibrium points, study the stability of the system, and explore how perturbations evolve. We will begin with the steps necessary to construct a plot of this ODE system over time.

4.1 Step 1: Finding the Equilibrium Points

Nullclines are curves in the phase plane where the derivative of a variable in a dynamical system equals zero, representing the system's fixed points. Spiral wave behavior often occurs near these Nullcline fixed points where the dynamics are not purely stable or unstable but oscillatory.

1. **u-nullcline**: The curve where $\frac{du}{dt} = 0$.
2. **v-nullcline**: The curve where $\frac{dv}{dt} = 0$.

The intersection points of the u- and v-nullclines correspond to the equilibrium points of the system, where the system is stationary ($\frac{du}{dt} = 0$ and $\frac{dv}{dt} = 0$).

$$0 = \frac{1}{\epsilon}\left(u^* - \frac{1}{3}(u^*)^3 - v^*\right), \tag{3}$$

$$0 = \epsilon\left(u^* + \beta - \gamma v^*\right). \tag{4}$$

Given Parameters:

- $\epsilon = 0.3$
- $\beta = 0.7$
- $\gamma = 0.5$

You might be curious as to why those values of $\epsilon = 0.3$, $\beta = 0.7$, $\gamma = 0.5$ have been chosen. Researchers often choose a range of plausible values from the literature. For the FitzHugh-Nagumo model, the parameter values above. are standard because they produce a balance of timescales and interaction strengths known to generate realistic dynamics.

Solve equation (4) for v^*:

$$u^* + 0.7 - 0.5v^* = 0 \quad \Rightarrow \quad v^* = 2u^* + 1.4.$$

Substitute v^* into equation (3):

$$u^* - \frac{1}{3}(u^*)^3 - (2u^* + 1.4) = 0 \quad \Rightarrow \quad (u^*)^3 + 3u^* + 4.2 = 0.$$

Numerical Solution of the Cubic Equation: Using the Newton-Raphson method for $u^3 + 3u + 4.2 = 0$:

- Initial guess: $u_0 = -1.0$
- Iterative (time consuming) steps yield: $u^* \approx -1.0328$
- Calculating v^* using the obtained u^*: $v^* = 2(-1.0328) + 1.4 = -0.6658$.

Final Equilibrium Point:

$$(u^*, v^*) \approx (-1.0328, -0.6658)$$

We can verify this by substituting $u^* = -1.0328$ and $v^* = -0.6658$ back into the original equations:

$$\frac{1}{0.3}\left(-1.0328 - \frac{1}{3}(-1.0328)^3 - (-0.6658)\right) \approx 0,$$
$$0.3\,(-1.0328 + 0.7 - 0.5(-0.6658)) = 0.$$

4.2 Step 2: The Nullcline Equations

After setting $\frac{du}{dt} = 0$ and $\frac{dv}{dt} = 0$ as above, we can rearrange the terms in eq. (1) and (2) to get simplified, concise versions of:

- The u-nullcline ($\frac{du}{dt} = 0$):
$$v = u - \frac{u^3}{3}.$$
  This is the **nonlinear nullcline**, represented as a cubic curve in the u-v plane.
- The v-nullcline ($\frac{dv}{dt} = 0$):
$$v = \frac{u + \beta}{\gamma}.$$
  This is the **linear nullcline**, represented as a straight line in the u-v plane.

4.3 Step 3: Starting Initial Conditions

Our starting initial conditions are the equilibrium points. $(u, v) = (u^* + 0.29, v^*)$

4.3.1 Why add a perturbation to the Equilibrium Point?

The equilibrium point of the system, denoted as (u^*, v^*), is the point where the system is stationary, i.e., $\frac{du}{dt} = 0$ and $\frac{dv}{dt} = 0$. While it is mathematically significant, starting the simulation exactly at (u^*, v^*) would result in no motion in the phase plane since the system would remain at rest.

To observe the dynamic behavior of the system, I will add a small perturbation to the equilibrium point. I'll pick 0.29, so

$$u = u^* + 0.29,$$

where 0.29 is a small deviation from u^*. Without this perturbation, the trajectory in the phase plane would remain static, and no insights into the dynamics near the equilibrium could be obtained.

4.3.2 Defining the first Variables:

```
#Starting Parameters:
epsilon = 0.3
beta = 0.7
gam = 0.5
ustar = -1.032789870  # The fixed points found before.
vstar = -0.6655797400 #'star' means asterik *

# Initial conditions
u = ustar + 0.29
v = vstar

#The Nullclines
nonlinearline = u - u**3 / 3.0  # du/dt = 0 (nonlinear)
linearline = (u + beta) / gam    # dv/dt = 0 (linear)
```

4.4 Step 4: Numerical Time Evolution of the System

To analyze the behavior of the FHN (Fitzhugh-Nagumo) system over **time**, we can use a numerical integration scheme based on an explicit forward method. Starting with the initial conditions $(u_0, v_0) = (u^* + 0.29, v^*)$, the equations for u and v are updated at each step using the following formulas:

4.4.1 Finite Increment Approximation (Euler's Method)

The derivative $\frac{du}{dt}$ tells how u changes per unit of time. Over a small time interval Δt, which we will call $dt = 0.01$, the change in u is approximately:

$$u_{t+\Delta t} \approx u_t + \Delta t \cdot \frac{du}{dt}.$$

The same applies to v:

$$v_{t+\Delta t} \approx v_t + \Delta t \cdot \frac{dv}{dt}.$$

Thus, substituting in our previous equations for $\frac{du}{dt}$ and $\frac{dv}{dt}$:

$$u_{t+\Delta t} = u_t + \Delta t \cdot \frac{1}{\epsilon}\left(u_t - \frac{u_t^3}{3} - v_t\right), \tag{5}$$
$$v_{t+\Delta t} = v_t + \Delta t \cdot \epsilon\left(u_t + \beta - \gamma v_t\right). \tag{6}$$

- $\Delta t = 0.01$: Time step size.
- $\epsilon = 0.3, \beta = 0.7, \gamma = 0.5$: System parameters.

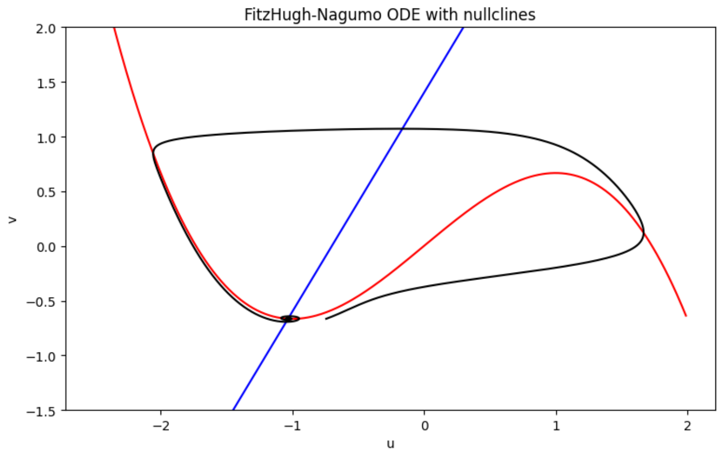
Iterating this over a for loop in python in the u-v phase plane, starting from the perturbed initial conditions of $(u_0, v_0) = (u^* + 0.29, v^*)$, will give us a good representation of what this system will look like overtime.

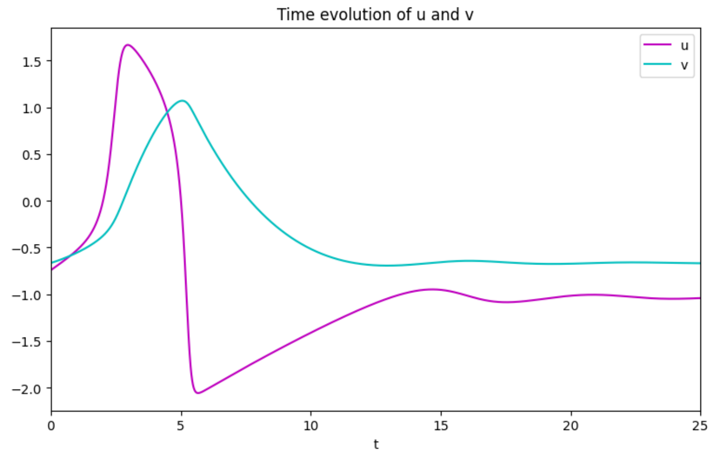
4.5 Full Code:

Listing 1: FULL CODE of the Fitzhugh-Nagumo ODE with Nullclines:

```
#initial parameters given
epsilon=0.3
beta=0.7
gam=0.5
ustar=-1.032789870 #"star", meaning asterik
vstar=-0.6655797400
loops=2500 # how many iterations performed
dt=0.01 # time step size
t=0.0 # initial time
# initial conditions given (ustar + 0.29, vstar)
u=ustar+0.29
v=vstar
# stores values of u,v, and t as they evolve over time
ulist=[u]
vlist=[v]
tlist=[t]
for loop in range(0,loops):
    ut=(u-u**3/3.0-v)/epsilon #du/dt
    vt=epsilon*(u+beta-gam*v) #dv/dt
    t=t+dt #time updater
    u=u+ut*dt #un+1
    v=v+vt*dt #vn+1
    tlist.append(t)
    ulist.append(u)
    vlist.append(v)
#nullcline plotting
np.arange(start, stop, step), good for handling tiny steps
uu=np.arange(-2.5,2.0,0.01) #Range of u values
nonlinearline=uu-uu**3/3.0 #when du/dt = 0
linearline=(uu+beta)/gam #when dv/dt = 0
```

4.6 Output of the Visual Static FHN ODE





4.6.1 A Brief Analysis:

The red u-nullcline ($\frac{du}{dt} = 0$), and the blue v-nullcline ($\frac{dv}{dt} = 0$), equilibria intersect at approximately $(-1.0328, -0.6658)$ as expected. We can examine the system's oscillatory and stabilizing behavior with the black trajectory line which illustrates the system's time evolution. This spirals around and eventually stabilizes near the equilibrium point.

4.7 Animation

To see the time evolution animated visually, here is the animation of the FitzHugh-Nagumo model ODE over time: Click Here:.

5 Modelling The FitzHugh-Nagumo PDE

Instead of just time based behaviour (ODEs), the FHN model can be extended to partial differential equations (PDEs) to study the spatial behavior. By incorporating spatial diffusion (which we will talk about later), essentially, **this model allows us to simulate how perturbations evolve over space and time**.

5.1 Step 1: Defining the Spatial Domain

The first step in solving the PDE is defining a discrete spatial grid area to represent the system. The spatial domain is set up as follows:

1. **Length of the Domain** (L):
   - While this will change in future models, the total length of the domain here is $L = 100$, spanning from $x = -50$ to $x = 50$.
2. **Number of Grid Points** (m):
   - A total of 201 grid points ($m = 201$) is chosen. This 'odd' choice of a number is so that the center of the grid ($x = 0$) aligns with a discrete grid point and not in between two.
3. **Spatial Step Size** (Δx):
   - The distance between adjacent grid points is calculated as $\Delta x = 0.5$, providing a fine resolution for accurate numerical differentiation.
4. **Grid Array** (x):
   - The grid points are represented by a numpy array, $x = \text{np.linspace}(-50, 50, 201)$, allowing the system to evaluate the u and v-values at each point in space.

```python
# spatial domain
L = 100.0  #length
m = 201
dx = 0.5
x = np.linspace(-50, 50, m)  #spatial grid points -50 to 50
dt = 0.01
```

5.2 Step 2: A Gaussian Perturbation

It's all good setting up this spatial grid, but if all the points are initially at equilibrium, nothing would happen over time. Which is exactly where the Gaussian perturbation comes in. Like before, it will affect the excitatory u-variable, representing a localized disturbance at t=0. The perturbation is set up as follows:

1. **Perturbation Parameters:**

   - Amplitude (a): $a = 0.5$
   - Width (b): $b = 0.05$

   These parameters determine the height ('strength') and spread of the Gaussian perturbation.

2. **Initial Conditions for u and v:**

   - u is perturbed by a Gaussian centered at $x = 0$:

     $$u(x, t = 0) = u^* + ae^{-bx^2},$$

     where u^* is the equilibrium value of u, and ae^{-bx^2} represents the localized perturbation.
   - v remains uniform across the domain: $v(x, t = 0) = v^*$, where v^* is the equilibrium value of v.

```
a = 0.5
b = 0.05
u = ustar + a * np.exp(-b * x**2)
v = np.ones(m) * vstar
```

5.3 Step 3: Adding the Laplacian

The most important part of the PDE: The Laplacian represents the second spatial derivative and accounts for how u diffuses across the spatial grid. I'll implement this with a numerical approximation below:

1. **Second Derivative Approximation:**

   $$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}.$$

   Here, u_i represents the value of u at grid point i, and Δx is the spatial step size. $u_i + 1$ represents the value of u at the grid point 'after' i, and $u_i - 1$ is the one before it. Essentially this looks at the grid point before, the point it's on, and the point after divided by step size to determine how u diffuses across the grid.

```
    d2u_dx2 = np.zeros(m)  #to store second derivative
    d2u_dx2[1:-1] = (u[2:] - 2*u[1:-1] + u[:-2]) / dx**2
```

Notice as well that the boundary points are excluded to avoid errors, as the second derivative is not well-defined there (there's no point before/after to analyze per say).

2. **Time Evolution:** Like in the ODE, we will use Euler's method for time-stepping, the u- and v-values are updated at each time step:

$$u_i^{n+1} = u_i^n + \Delta t \left[\frac{1}{\epsilon} \left(u - \frac{u^3}{3} - v \right) + \frac{\partial^2 u}{\partial x^2} \right],$$

The corresponding Python code is:

```
    ut = (u - u**3 / 3.0 - v) / epsilon #Reaction for u
    vt = epsilon * (u + beta - gam * v) #Reaction for v

    # Update interior points using Euler's method
    u[1:-1] = u[1:-1] + dt * (ut[1:-1] + d2u_dx2[1:-1])
    v[1:-1] = v[1:-1] + dt * vt[1:-1]
```

3. **Boundary Conditions:** As mentioned before, the Laplacian doesn't exactly work at the borders. But it's essential to handle them with boundary conditions, which we will be using Dirichlet to enforce fixed values of $u*$ and $v*$ at the edges of the spatial grid:

```
    u[0], u[-1] = ustar, ustar  # Fixed boundary for u
    v[0], v[-1] = vstar, vstar  # Fixed boundary for v
```

5.4 Step 4: Plotting the PDE

Below is the relevant code which produces a static image of the PDE at t = 0 and at t = 20 with the Gaussian Perturbation.

```
a = 0.5
b = 0.05
epsilon=0.3
beta=0.7
gam=0.5
ustar=-1.032789870
vstar=-0.6655797400

# spatial domain
L = 100.0
m = 201
dx = 0.5
#201 points / 0.5 over 100~ grid points
```

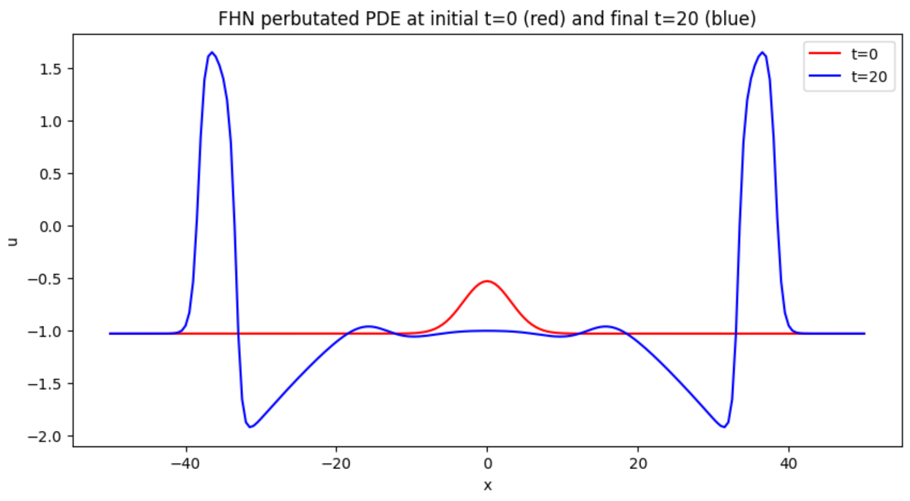
```python
x = np.linspace(-50, 50, m)
dt = 0.01
loops = 2000

#gauss perturbation
u = ustar + a * np.exp(-b * x**2)
v = np.ones(m) * vstar

atinitialtime = u.copy() #for t = 0 snapshot

for loop in range(loops):
    d2u_dx2 = np.zeros(m)
    d2u_dx2[1:-1] = (u[2:] - 2*u[1:-1] + u[:-2]) / dx**2
    ut = (u - u**3/3.0 - v) / epsilon #as part of FHN
    vt = epsilon * (u + beta - gam * v)
    u[1:-1] = u[1:-1] + dt * (ut[1:-1] + d2u_dx2[1:-1])
    v[1:-1] = v[1:-1] + dt * vt[1:-1]
    u[0], u[-1] = ustar, ustar
    v[0], v[-1] = vstar, vstar

atfinaltime = u.copy() #for t = 20 snapshot
```

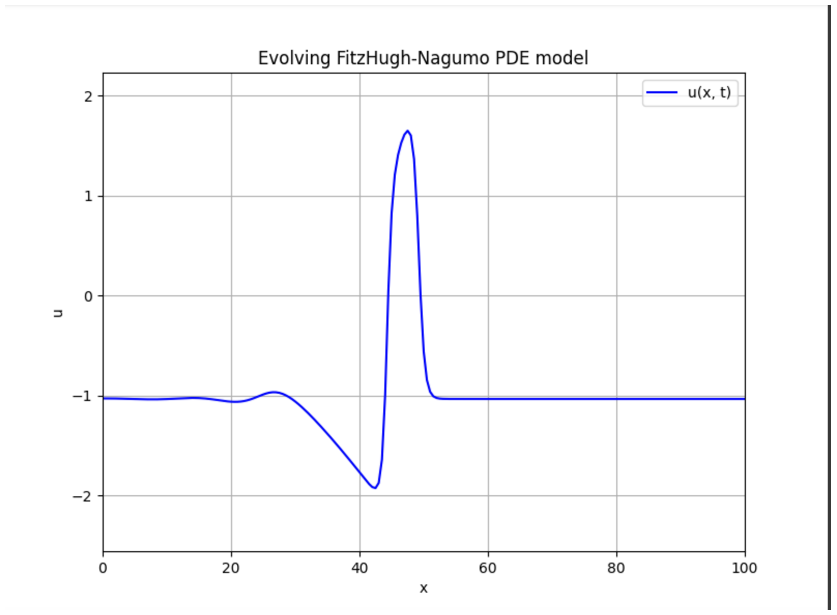


5.5 Animation + A Brief Analysis:

To see the time evolution animated visually: Click Here.
Over time, the initial localized disturbance propagates and interacts with the system's dynamics, leading to the formation of distinct spatial patterns.

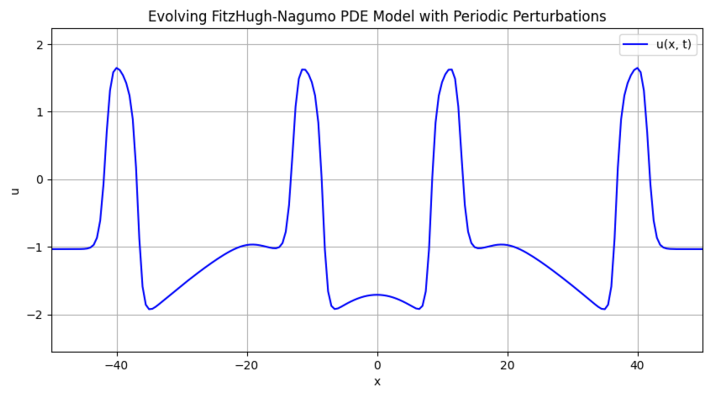
5.6 Other Relevant Models/Animations:

5.6.1 Restricted to Positive Spatial Points:



Notice how this resembles a heartbeat! To see the time evolution animated visually: Click Here:.

5.6.2 Periodic Perturbations:



To see the time evolution animated visually: Click Here:.

6 Modeling Spiral Waves in 2D

While modeling the FitzHugh-Nagumo system in one dimension provides insights into wave propagation, a 2D simulation is essential to observe the emergence and dynamics of spiral waves. Modeling spiral waves in two dimensions allows us to explore how localized perturbations can lead to harmful oscillatory patterns that propagate through a vertical slice of the heart as we extracted earlier in chapter 1.

6.1 Step 1: Setting Up the 2D Grid

To simulate the propagation of electrical signals in two dimensions, we first define a discrete spatial grid. The grid is constructed using a square domain, where each point represents a location in the 2D medium. The spatial domain is divided into evenly spaced intervals with grid points determined by the step size resolution parameter $\Delta x = 0.75$.
With the Python `numpy` import, the amazing `np.linspace` function will be used to set up the grid with:

```
np.linspace('starting point', 'ending point', 'number of points
that'll fit inside')
```

Then, it will be combined together with `np.meshgrid`, to form the Points x Points grid. Slice of code below:

```
points = 201
dx = 0.75
x = np.linspace(0, 50, points)
y = np.linspace(0, 50, points)
X, Y = np.meshgrid(x, y)
```

The parameters for the grid are as follows:

- **Domain Size:** A square domain spanning $[0, 50]$ in both x- and y-directions.
- **Number of Grid Points:** 201×201, the `points` variable.
- **Spatial Step Size (Δx):** 0.75, written as `dx`

6.2 Step 2: Adjusting the Gaussian Perturbation for 2D

Unlike the 1D case, where the perturbation is applied along a single spatial axis, in the 2D model, the Gaussian perturbation needs to account for spatial variation across both dimensions. This introduces a localized disturbance that combined with all the grid points being concurrently applied with the FitzHugh-Nagumo equation, will form a spiral at the location specified.

The functional form of the perturbation is given as:

$$u = u^* + 10 \cdot \exp\left(-\frac{(X-10)^2}{100}\right), \quad v = v^* + 10 \cdot \exp\left(-\frac{(Y-10)^2}{100}\right),$$

where:

- X and Y are the grid coordinates from `np.meshgrid`.
- u^* and v^* are the unchanged equilibrium points.
- Where $X = 10$ and $Y = 10$, indicate the location of this perturbation.
- The factor 10 determines the strength of the initial perturbation.
- The denominator 100 controls the spread of the initial perturbation. (don't be mistaken, this is only initially, as the Laplacian handles the rate of diffusion (spread) after).

```
u = ustar + 10 * np.exp(-((X - 10) ** 2) / 100)
v = vstar + 10 * np.exp(-((Y - 10) ** 2) / 100)
```

Do note that from here on out, I will be calling this 2D Gaussian Perturbation the `stim` variable.

6.3 Step 3: Updates to the Iterative Loop and Boundary Conditions

After defining the 2D grid and introducing the Gaussian perturbation, the next step involves altering the FitzHugh-Nagumo model equations over time to incorporate this new layout. A short summary of the new notation that follows:

- $u^n_{i+1,j}$ is the neighboring grid point in the x-direction to the right.
- $u^n_{i-1,j}$ is the neighboring grid point in the x-direction to the left.
- $u^n_{i,j+1}$ is the neighboring grid point directly above in the y-direction.
- $u^n_{i,j-1}$ is the neighboring grid point directly below in the y-direction.

6.3.1 Iterative Loop for Time Evolution

Not much directly is changed here, besides being able to handle 2D grid points

$$u^{n+1}_{i,j} = u^n_{i,j} + \Delta t \left[\frac{1}{\epsilon} \left(u^n_{i,j} - \frac{(u^n_{i,j})^3}{3} - v^n_{i,j} \right) + \nabla^2 u^n_{i,j} \right],$$

$$v^{n+1}_{i,j} = v^n_{i,j} + \Delta t \cdot \epsilon \left(u^n_{i,j} + \beta - \gamma v^n_{i,j} \right),$$

```
ut = (u[1:-1, 1:-1] - (u[1:-1, 1:-1] ** 3)/ 3 - v[1:-1, 1:-1])
/ epsilon
vt = epsilon * (u[1:-1, 1:-1] + beta - gam * v[1:-1, 1:-1])
```

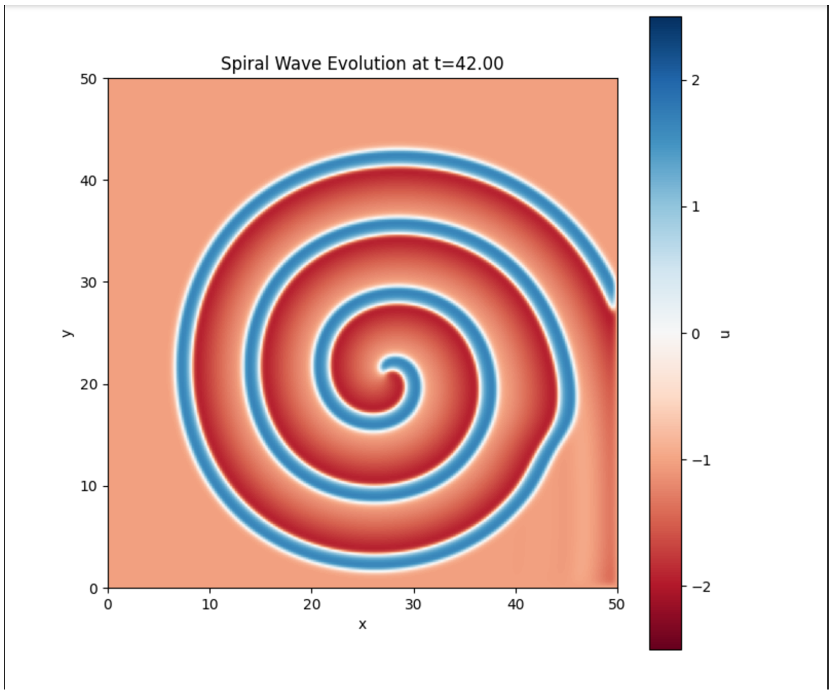
where $\nabla^2 u^n_{i,j}$ is the discrete Laplacian operator, and since it also needs to account for the diffusion of u in both x- and y-directions here, the Laplacian $\nabla^2 u$ will now be computed as:

$$\nabla^2 u^n_{i,j} = \frac{u^n_{i+1,j} - 2u^n_{i,j} + u^n_{i-1,j}}{\Delta x^2} + \frac{u^n_{i,j+1} - 2u^n_{i,j} + u^n_{i,j-1}}{\Delta x^2}.$$

```python
u2x = (u[2:, 1:-1] - 2 * u[1:-1, 1:-1] + u[:-2, 1:-1]) /dx**2
u2y = (u[1:-1, 2:] - 2 * u[1:-1, 1:-1] + u[1:-1, :-2]) /dx**2
u2xy = u2x + u2y #adding them together as above
```

Notice the `u[1:-1, 1:-1]`. For non-python enthusiasts, essentially this only applies the function over the 2D interior points, due to the presence of the boundary conditions that are still the unchanged Dirichlet.
This concludes all the important mathematically relevant changes made to the code. As one would guess, the animation function of the dynamic model undergoes drastic changes as well, but complicated matplotlib explanations are not the focus of this project.

6.4 Final Output of Model:



As always, here is the animation video: Click Here:.

7 Dead Cell Induced Spiral Formation

The propagation of electrical waves in the heart is essential for maintaining a regular rhythm. As seen back in chapter 1, a healthy heart has electric pulses originating from the top left in the sinoatrial node. However, disruptions caused by damaged cardiac tissue, such as dead cells, can lead to formations of their own spirals upon contact with the normal electric heart pulses. By introducing a region of non-conductive "dead cells," the periodic waves are disrupted, resulting in the formation of spirals.
These spirals can then interfere with the normal periodic heartbeat waves, sometimes even forming their own, brand new spirals.

7.0.1 Incorporating the 2D Heart Mask Function

To simulate electrical signals within a realistic heart shape, I'll also be adding a 2D visual heart mask to restrict the FHN model to the heart's interior. The mask is generated using a mathematical function applied to all computations and future models in this project, as shown below:

```python
def heart_mask(points):
    x = np.linspace(-1.2, 1.2, points)
    y = np.linspace(1.2, -1.2, points)
    heart = ((X**2 + Y**2 - 1)**3 - X**2 * Y**3) <= 0
    return heart
```

7.0.2 A Higher Range of Freedom with External Stimulations:

Another quick and short addition, the external Gaussian perturbations will now be incorporated directly into the FitzHugh-Nagumo PDE through the `stim[time:,y:,x:]` variable allowing greater range of control over the timing and location of perturbations within the heart.

```python
du = ((u - (1/3)*u**3 - v + stim[t, :, :]) / epsilon) + L(u)
```

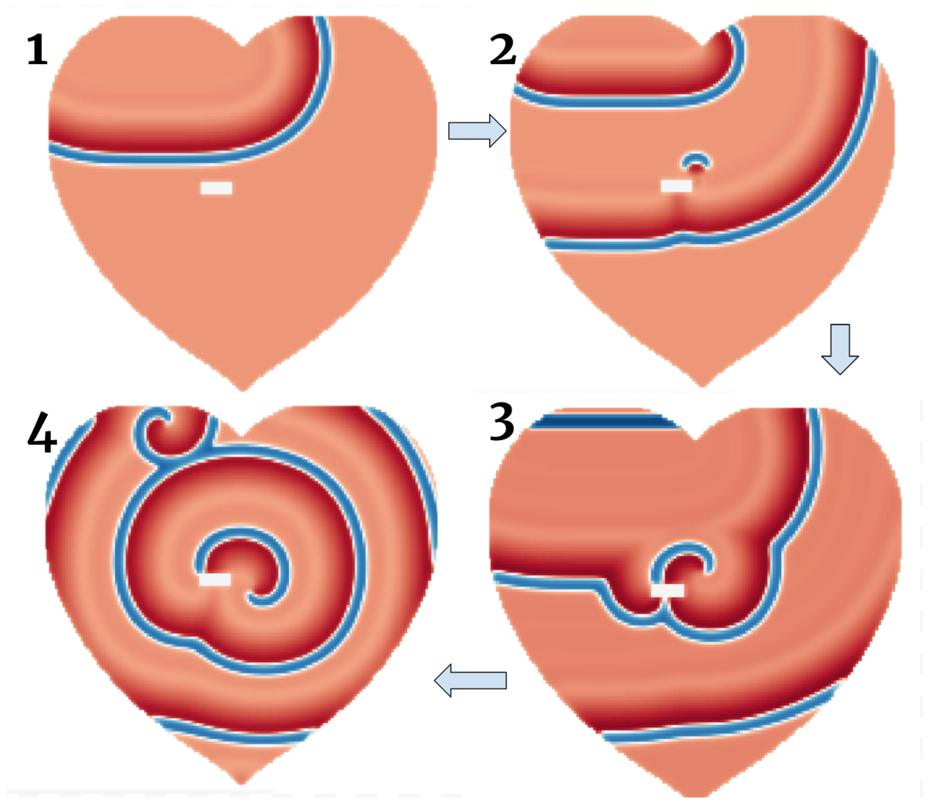
7.1 Incorporating Dead Cells

To simulate a region of dead cells, the values of u and v are set to zero. This represents tissue that cannot get excited, thus won't participate in the propagation of electrical signals.

The presence of these unexcitable regions disrupt the normal wave propagation, forcing the electrical signals to curve around them. This disruption can lead to the formation of spiral waves, as continually attempting to propagate around the obstacle will create rotating patterns in the surrounding excitable tissue.
Do note that the formation of spirals is not always guaranteed, and statistical randomness influencing their creation will be covered later.

```
#rectangle 5 tall & 8 wide
u[64:69, 64:76] = 0.0
v[64:69, 64:76] = 0.0
```

7.2 The Full Model:



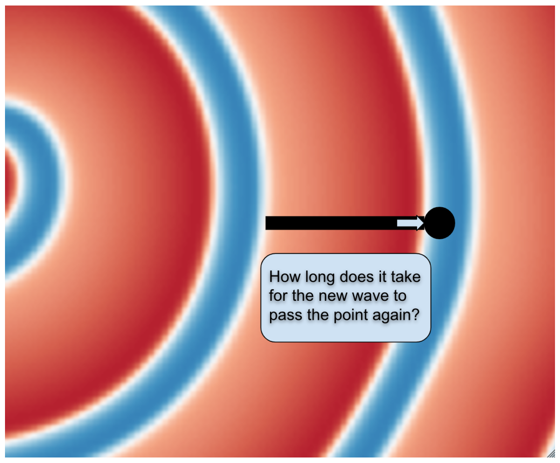
- **Panel 1:** A normal wave propagates uniformly across the heart, encountering no obstacles or disruptions.
- **Panel 2:** The wavefront reaches the region of dead cells (white rectangle), sparking a disturbance in the wave propagation.
- **Panel 3:** The disturbance forces the wave to curve around the unexcitable region, forming the initial spiral structure.
- **Panel 4:** The spiral wave becomes fully established, dominating the tissue and forming new spirals upon contact with the normal heartbeat waves.

To see the full animation visually: Click Here:.

8 Analyzing the Period of the 2D Spiral Waves

The period in the context of the spiral wave refers to the time it takes for the wave at a specific point to complete one full oscillation and return to the same phase or value (e.g., hitting a local maximum in u again).

For a spiral wave, this corresponds to the time it takes for the same part of the spiral wave to pass over that point in the grid again. Since the period is the duration between successive peaks, this will be the subject of the model below.


How long does it take for the new wave to pass the point again?

8.0.1 Extracting the Time Series

The designated point I will choose here is the midpoint, where `mid_idx` is $\frac{\text{points}}{2}$.

8.1 Calculating the Average Period

To determine the average period of oscillations:

1. **Identifying Peaks:** Local maxima (peaks) throughout the many iterations of the code are identified using the condition:

$$u_t[i-1] < u_t[i] > u_t[i+1].$$

The corresponding times of these peaks are stored in an array

```python
max_times = [
    times[i]
    for i in range(1, len(u_t) - 1)
    if u_t[i - 1] < u_t[i] > u_t[i + 1]]
```

2. **Calculating Periods:** The period between consecutive peaks is calculated as:

$$\text{periods}[i] = \text{max\_times}[i+1] - \text{max\_times}[i],$$

for all $i = 1, 2, \ldots, n-1$, where n is the number of identified peaks.

```python
periods = [max_times[i + 1] - max_times[i]
for i in range(len(max_times) - 1)]
```

3. **Average Period:** The average period is computed as:

$$\text{Average Period} = \frac{\sum_{i=1}^{n-1} \text{periods}[i]}{n-1}.$$

This value represents the typical time interval between successive oscillations at the midpoint.

```python
avg_period = sum(periods) / len(periods)
```

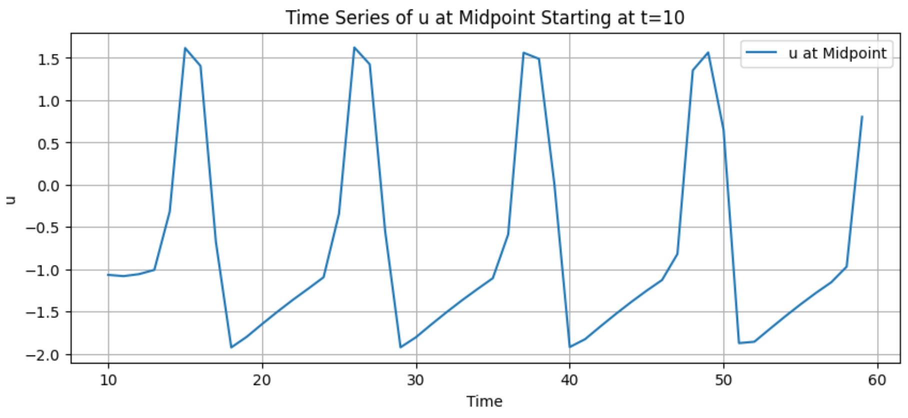
Additionally, due to the period undergoing serious fluctuations the first few milliseconds of the animation, the average will start after t = 10.

8.2 Result:

After running the code, the printed out Average Period is:

```
Average period (after t=10): 11.33 units
```

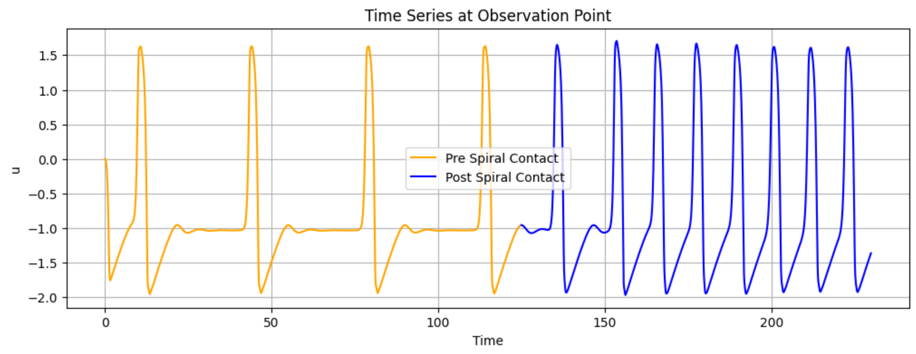
Modeled:



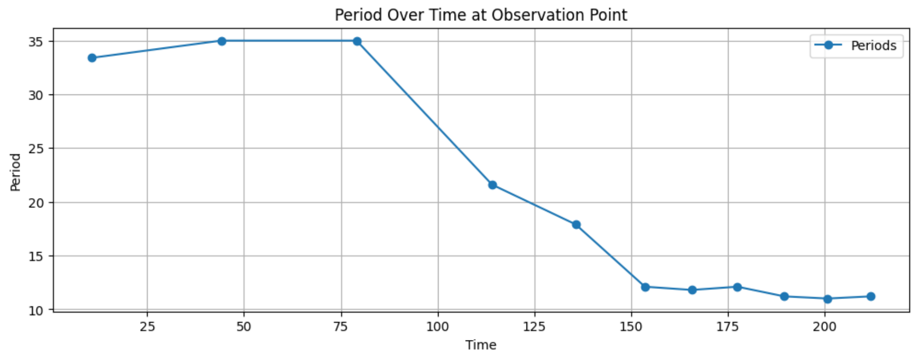
8.3 Measuring the Period of the Dead Cell model:

Recall from the previous chapter how the simulation begins with regular periodic normal heartbeat waves propagating uniformly across the domain. However, upon impact with the dead cell region, the formed spiral soon dominates the dynamics of the heart, pushing back against the periodic normal heartbeat waves. This alters the period observed at a given point as the spiral's faster rotation shifts the observed oscillations to a new smaller periodicity.

The designated point I will choose here to measure the period is $(16, 46)$, which is located near the top left border of the heart.



- Initially regular heartbeat waves are replaced by a more complex pattern as the spiral gains control of the system.
- On average, the period (distance between peaks) is significantly decreased, meaning the designated point gets 'hit' by more waves, as the spiral oscillates faster.


Period Over Time at Observation Point
Periods
Period
Time