university of groningen / faculty of science and engineering

Chaos Control: Controlling Heart Arrhythmia Using an Echo State Network Controller

Student: Viktor Veselý, s3970043, v.vesely@student.rug.nl
Supervisor: Prof. Dr. H. Jaeger
Co-Supervisor: Dr. J. P. Borst

A bachelor thesis presented for the degree of
Artificial intelligence

Faculty of Science and Engineering
University of Groningen
The Netherlands
July 2022

Abstract: The regular propagation of the action potential on heart tissues is crucial for the optimal heart muscle contractions. Obstacles to the heart tissues such as scars can cause the propagation to become chaotic - small differences amplify over time and can become fatal for the diseased person. I used an echo state network (ESN) controller to investigate the effectiveness of the method on controlling a simulated heart arrhythmia, which can be modelled as chaotic action potential propagation. ESNs have been previously showed to predict a chaotic time series with unprecedented precision. I altered an existing computational heart model to exhibit such a chaotic dynamics. Whilst the validation error of the controller remained reasonably low, it was unable to control the arrhythmia. The controller performed significantly better on a simpler chaos control task (with 2 dimensions instead of 2560), suggesting that the original control task was too complex and unfeasible with our computational resources.

Contents

1 Introduction

The heart is a vital organ in our bodies that ensures a proper flow of oxygenated blood. The function of our brain, muscles, and organs is highly dependent on the energy and oxygen delivered by the blood. It is thus essential that the heart executes its function flawlessly.

The heart is a complex non-linear dynamical system that pumps the blood around our cardiovascular system. In order to do so, the heart is surrounded by a layer of cardiac muscles that pump the blood around the body (this thesis has been done by a student of an Artificial Intelligence programme, consequently, a lot of anatomical/biological complexities has been simplified. This thesis should be taken as a proof of concept of the machine learning technology in medical use rather than a medical description of the heart). The heart is divided into 4 chambers - 2 upper ones called right/left atria and 2 lower ones called right/left ventricles (Figure 1.1 provides a visual explanation). Each chamber contracts and pumps the blood to the next chamber. The deoxygenated blood comes from the body to the right atrium, going to the right ventricle. The right ventricle pumps the blood to the lungs where it gets oxygenated. The oxygenated blood follows to the left atrium which pushes it to the left ventricle, and finally, the left ventricle distributes it to the rest of the body (Hall et al., 2021).

The cardiac muscles are synchronised by the action potential propagation (Hall et al., 2021). The neural signal begins in a SA node (which is located in the right upper corner of the right atrium) and travels to the left atrium, resulting in the contraction of the upper chambers. The neural signal then travels to the right & left ventricles which cause both of the lower chambers to contract. The process then repeats again - ensuring an optimal blood flow.

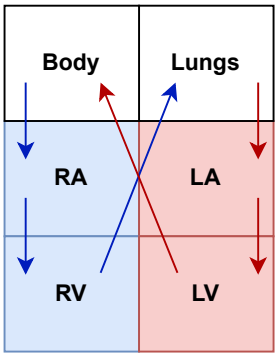


Figure 1.1: Functional diagram of the heart blood flow. RA stands for right atrium, LA for the left atrium, RV for right ventricle, and LV for the left ventricle. The blue sectors signify deoxygenated blood and the red sectors the oxygenated blood

Disruption of this propagation pattern can lead to an *arrhythmia*. The U.S. National Heart Blood and Lungs Institute recognises numerous types of arrhythmias (NHLBI, 2018). However, this thesis focuses mostly on reentry arrhythmia (Goyal et al., 2021) and ventricular tachyarrhythmia due to the simplicity of modelling and their chaotic behaviour. Irregular heartbeats/arrhythmias lead to an under oxygenated brain, organs, and muscles which significantly decrease the quality of the subject's life or in many cases even causes death. It is thus essential to develop technologies that prevent or minimize the risk of such an arrhythmia.

For the reader's & author's convenience, this thesis will shorten the phrase "reentry arrhythmia and ventricular tachyarrhythmia" to RAVF. RAVF causes are fully described in Goyal et al. (2021) & Garner and Miller (2013) but this thesis will mainly focus on RAVF caused by heart-tissue heterogeneities, namely, anatomical obstacles and functional heterogeneities. Once such heterogeneity is present in the heart, it is possible for the neural signal to loop back around the obstacle and initiate a reentry arrhythmia - forming an independent circuit (independent from the SA node). Figure 1.2 shows such a situation. The circuit can block the action potential generated by the SA node and the heart rate is then dictated solely by the reentry circuit. Depending on the circuit length, the heart can start beating dangerously fast without any way of slowing down (Goyal et al., 2021).

Schöll and Schuster (2008) describes the ventricular tachyarrhythmia as a possible consequence of reentry circuits. If the "head" of the action potential wave catches up with the "tail" the wave can break into multiple smaller waves which then travel around the ventricles and atria causing atrial/ventricular fibrillation. Fibrillation is a chaotic contraction and relaxation of heart muscle tissue, leading to a highly-irregular heartbeat and almost certainly to death of the patient.

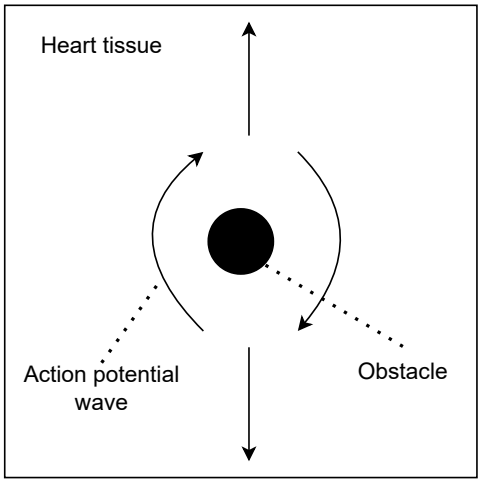


Figure 1.2: A reentry circuit formed due to an anatomical obstacle (functional heterogeneity). The circuit sends signals chaotically around the heart - overriding the target heart-rate

1.1 Existing solutions

Three prevalent solutions are used to treat arrhythmias in medicine: implantable cardioverter defibrillators (ICD), pharmacological treatment, and artificial pacemakers.

1.1.1 Implantable Cardioverter Defibrillators

ICD is a compact defibrillator with a microprocessor that gets attached to a patient's heart by surgery. The main idea is for the microprocessor to scan the heart's neural signals and look for arrhythmia. Once it detects an irregularity it administers an electrical shock with the aim to reset the heart. The shock forces all the neurons to enter their refractory period - a period where the neuron is unable to produce an action potential, thus, ending any reentry circuits and/or any irregularities.

Even though ICD is mostly used to treat serious arrhythmias it also comes with its drawbacks. The detection part of the device is not flawless. Most ICDs are single/dual chambers (attached only to one or two heart chambers), consequently, the detection algorithm always works with a limited amount of information (Francia et al., 2009). When the heart irregularities manifest outside of the detectable range, the microprocessor will not be able to administer the shock in critical scenarios.

Vice versa a regular heartbeat can be misinterpreted by the detection algorithm as an arrhythmia - leading to a painful electrical shock. False positives can make up one-third of all the shocks administered to the patient (Schöll and Schuster, 2008).

1.1.2 Pharmacological Treatment

Antiarrhythmic drugs prolong or shorten the refractory period in order to eliminate any re-entry circuit and/or heartbeat irregularities (Fogoros, 2008). Even though the drugs can help regulate certain types of arrhythmia (Fogoros, 2008) they are generally used as a second option due to the possibility of worsening the arrhythmia (Fogoros, 2008) or in some cases even killing the patient (Ruskin, 1989).

1.1.3 Artificial pacemakers

Similarly to ICDs, pacemakers are implantable devices that track the activation of the heart tissue. Instead of delivering a big shock when the irregular heart rhythm is detected, pacemakers inject a small local shock that usually kickstarts an action potential cascade. Traditionally, pacemakers counter bradycardia (slower than normal heartbeat). However, in recent years their usage has extended to other types of arrhythmias as well (Alasti et al., 2018). The challenge of finding the correct placement and volume of the shock remains due to the heart being a complex non-linear chaotic system (Ferreira et al., 2011).

1.1.4 Machine learning approach

There is a rich history of machine learning techniques being used in the treatment of arrhythmias. Commonly, machine learning helps with:

1. Assistance of ECG analysis (Nagarajan et al., 2021; Isin and Ozdalili, 2017; Perez et al., 2009). ECG is a full heart screening that is performed by a cardiologist. The aforementioned studies show how artificial intelligence can be used to produce more accurate interpretations and more detailed analyses of ECG.

2. Detection/prediction of arrhythmias with ICDs and artificial pacemakers (Alonso-Atienza et al., 2013; Li et al., 2013). However, these techniques do not determine the amount or the time of the administered shocks, therefore, an additional analysis is always needed that calculates the time and the volume of the shock.

3. Control of arrhythmias using fractional-order PID controllers (Momani et al., 2019; Bajpai et al., 2017) or chaos controllers (Ferreira et al., 2011) that act as an artificial pacemakers. Both of the methods require a model of the heart in order to function, however, the former method (FOPID controllers) only performs well on linear systems and, thus, it needs an oversimplified model of the heart.

The reader can find numerous articles connected to topics 1) and 2). Consequently, this thesis will try to expand on the 3rd point - heart control, namely, cardiac chaos control using an echo state network.

1.2 Chaos control

Control tasks are associated with generating an *action* such that if the *action* is applied to a *system*, the *system* behaves according to our instructions. *Dynamical systems* in nature are constantly, vibrating, evolving, and changing according to laws of physics and biology. The task of a control engineer is to design an *controller* which directs the *system* to a specific *trajectory*. It is important to understand chaotic dynamics in order to grasp chaos control.

Chaotic systems can be described best by the sensitivity to the small changes in initial conditions. Two chaotic systems initialized almost identically with only very small differences will, exponentially diverge as time progresses. This is called the "butterfly effect" - where small changes are amplified over time. Chaotic systems are therefore sensitive to perturbations. A small impulse sends a chaotic system to a completely different path. The task of a *controller* is to monitor the system and constantly nudge it to a preferred trajectory.

Chaos control is a popular task required in numerous fields: performing an orbital transfer by steering a spacecraft (Macau and Grebogi, 2006), deciding on an investing strategy in a competitive market (Holyst et al., 1996), or controlling heart dynamics (Ferreira et al., 2011). In the literature, two methods dominate for controlling chaos: 1) the Ott, Grebogi, and Yorke method (Ott et al., 1990) stabilizes the system on a desired orbit by waiting for the system trajectory to approach the wanted region. However, it requires mathematical analysis, some manual hand-tuning, and it relies on the system to approach the desired point/orbit - making it unpredictably slow; 2) The time delay feedback controller (later just TDF) (Ferreira et al., 2011) defines the control equation as a non-linear system where the input is a current state and sequence of the previous states of the system. TDF stabilizes the trajectory by nudging it based on the error. Similarly to a PID controller, TDF is very simple. It requires only a few parameters to design the controller - which is a double-edged sword. It can be very quick to set up for simple chaotic tasks but it might not scale well for more complex problems (such as heart control).

1.3 Echo state networks

Echo state networks (ESNs) belong to a *reservoir computing* paradigm in which the recurrent part of the neural network (simply called a *reservoir*) is generated randomly and only a single output layer is trained (Lukoševičius and Jaeger, 2009). Training RNNs using backpropagation-trough-time is computationally expensive. ESNs avoid this issue by training only the output weights (sometimes also called the *readout weights*) using linear regression. Consequently, the *readout weights* are trained analytically and the loss reaches the global minumum instantly (something that is not possible at all with the gradient methods). The predictions still maintain the benefit of non-linearity, due to the non-linear dynamics of the *reservoir*.

Similarly to support vector machines, ESNs hope that the input becomes more linearly processable in the *reservoir*, which is usually higher-dimensional than the input by order of magnitudes. Additionally, due to recurrent connections in the *reservoir* the current state is dependent on the previous state which is also dependent on the previous-previous state etc. - providing a temporal context (memory) for the output calculation. ESNs belong to a family of the time delay feedback controller, where the ESN receives as input a current state and a previous state which *reverberates reservoir* and the *reservoir* is used to compute an action which nudges the system to the desired trajectory.

It is already known that RNNs can approximate any dynamical system (Funahashi and Nakamura, 1993). They can also be used as a controller for linear problems (Salmen and Plöger, 2005). However, there are not many applications of ESNs being used to control chaotic systems. Jaeger and Haas (2004) showed the superiority of an ESN's ability to predict chaotic time series compared to previously used techniques. The ESN improved the error by four orders of magnitude. The research of Jaeger and Haas (2004) gave strong empirical evidence for the ESNs to be useful in chaotic systems. This thesis wants to contribute another empirical proof by training an ESN controller that treats the heart arrhythmia.

1.4 Research objective

As mentioned before, the research objective is to use an ESN as a controller and treat heart arrhythmias in two simplified computational models of a heart (which are described in the Theoretical Background section 2). More concretely, a healthy heartbeat will be recorded and used as a desired trajectory for the controller. The controller task will be to stabilize the arrhythmic chaotic trajectory on a stable healthy pre-recorded orbit. The controller will be evaluated based on normalised root mean squared error (NRMSE) between the desired trajectory and the actual trajectory taken by the computational heart model.

2 Theoretical Background

This section explains the theoretical background of single heart cell (cardiomyocyte) dynamics, heart dynamics, a simplified 2d chaotic heart model using a modified Van der Pol oscillator, and simple ESN controller architecture.

2.1 Cardiomyocytes

Cardiomyocytes are cardiac cells responsible for neural signal propagation and cardiac muscle contraction/relaxation. This thesis uses a modified version of the ventricular cell model as described in Luo and Rudy (1991) which builds on the famous Hodgkin and Huxley (1952) model. The model describes a formation of an action potential as a movement of ions inside the cell. Figure 2.1 shows the action potential as described in Luo and Rudy (1991). This thesis assumes that the entire heart is covered with ventricular cells, which in practice are different from atrial cells (Goette et al., 2016). However, looking at Figure 6 from the Goette et al. (2016), which shows the difference between the atrial and ventricular cell action potential, one can see that this assumption is not a gross oversimplification.

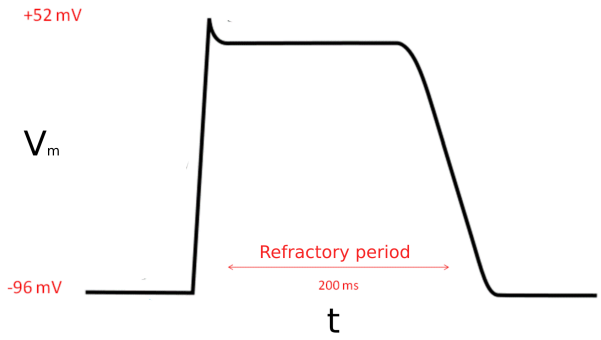


Figure 2.1: An action potential of cardiac cell. The figure represents the change of the membrane voltage (V) over time and under stimulation. Figure adapted from Wikipedia (2021)

A refractory period (RP) is a window of time under which the neuron does not produce another action potential even when stimulated. Multiple factors can influence the length of the RP, namely AV/SA nodes' frequency (Ferrier and Dresel, 1974) or pharmacological injection (Morady et al., 1988). Due to computational reasons, I modified the cell model's RP to be around 40ms (the new parameters can be found in Table 2.1). Shorter RP periods, even though not biologically plausible, allow for a smaller model (described more in-depth in the next section), and thus, a shorter computational time. In theory, it is completely possible to reproduce the results found here with the original parameters' values, however, it requires one to simulate more cardiac cells (in fact as many so the action potential wave, as it loops back, does not catch-up with the cardiac cells which are in the refractory period).

Parameter	Original value	Modified value	Remarks
$\bar{G}_{SI}$	$0.09mS$	$0.018mS$	Lowers the conductance of the silicon channel
$\alpha_X(V)$	$\alpha_X(V)$	$5 \cdot \alpha_X(V)$	Shortens the time constant of the X gate
$\beta_X(V)$	$\beta_X(V)$	$5 \cdot \beta_X(V)$	Shortens the time constant of the X gate
$[K]_o$	5.4	variable	Varies the extracellular potassium concentration

Table 2.1: Parameters that were modified in order to shorten the refractory period of the cardiac cells

The complete set of equations guiding the cell can be found in Appendix section A.1. Here I will describe only the most important dynamics of the cell. The action potential is guided by three differential equations and eight state variables. The *membrane potential* V is defined by

$$\frac{dV}{dt} = -\frac{1}{C}(I_{ions} - I_{stim}). \tag{2.1}$$

I_{ions} is the sum of all ionic currents

$$I_{stim} = \sum_{ion} I_{ion}, \tag{2.2}$$

where $ion \in \{Na, Si, K, K1, Kp, b\}$. I_{stim} is the stimulation current.

The amount of current going through each ionic channel depends on the state of the *gates* located inside of the channel. *Gates* can be either closed or open. If all gates are open, the ions rush in/out of the cell - creating a current. Let y be a probability of an ionic channel being open, α_y probability of the gate to close when being open, and β_y the probability of the gate to open when being closed. The rate of change of y can be thought of as a difference in the current value of y and the so called "steady state" of y written as y_∞.

$$\frac{dy}{dt} = \frac{y_\infty - y}{\tau_y} \tag{2.3}$$

For every voltage V, the probability y converges towards y_∞. The speed of convergence is determined by the time constant of the gate τ_y. Both τ_y and y_∞ can be expressed in terms of α_y and β_y

$$\begin{aligned} y_\infty(V) &= \frac{\alpha_y(V)}{\alpha_y(V) + \beta_y(V)}, \text{ and} \\ \tau_y(V) &= \frac{1}{\alpha_y(V) + \beta_y(V)}. \end{aligned} \tag{2.4}$$

There are six distinct gates, $y \in \{m, h, j, d, f, x\}$. A detailed definition of each of the gates can be found in Appendix A.1.

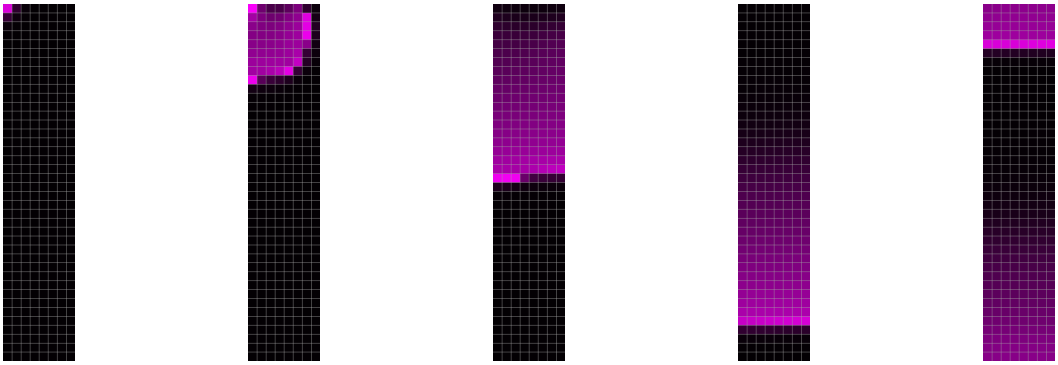
Lastly, under short stimulation of the cell, the intracellular/extracellular ionic concentrations (denoted by $[Ion]_{\{i,o\}}$) do not change significantly, except the intracellular concentration of calcium $[Ca]_i$ - making it the eighth state variable

$$\frac{d[Ca]_i}{dt} = -10^{-4} \cdot I_{Si} + 0.07 \cdot (10^{-4} - [Ca]_i). \tag{2.5}$$

2.2 Heart model

The model used for this thesis is an adapted version of the model described in Blanc et al. (2001) which is based on a reaction-diffusion equation. Reaction-diffusion equations are usually used to model a spread of chemical/fluid/heat across some spatial dimension. Blanc et al. (2001) modified it to represent the propagation (a.k.a spread) of the action potential on the heart tissue. The model is guided by the following partial differential equation:

$$\frac{1}{S_v}\nabla V \cdot (\rho \nabla V) = C_m \cdot \frac{\partial V}{\partial t} + I_{ion} - I_{stim}, \tag{2.6}$$



(a) Frame 1 **(b) Frame 2** **(c) Frame 3** **(d) Frame 4** **(e) Frame 5**

Figure 2.2: Showcase of the action potential wave propagating around the healthy heart tissue. Black cells correspond to the cardiomyocytes that have a resting membrane voltage and purple cells signify an occurring action potential

where S_v is the surface to volume ratio, V the membrane potential, ρ the resistivity matrix, C_m the membrane capacitance, I_{ion} the sum of the ionic currents calculated from the cell model described in the previous subsection, I_{stim} the external current used for heart-beat initiation and the pacemaker current injection, and ∇V is a 2-dimensional gradient of the membrane potential.

I used the forward Euler method to solve the Equation 2.6 with $dt = 0.015ms$ and $dx = dy = 200 \cdot 10^{-4} = 0.02cm$. Discretizing Equation 2.6 yields:

$$\begin{aligned} &\frac{1}{Sv\Delta y^2}\left(\frac{V^n_{i-1,j} - V^n_{i,j}}{\rho^{i-1,i}} + \frac{V^n_{i+1,j} - V^n_{i,j}}{\rho^{i+1,i}}\right) + \frac{1}{Sv\Delta x^2}\left(\frac{V^n_{i,j-1} - V^n_{i,j}}{\rho^{j-1,j}} + \frac{V^n_{i,j+1} - V^n_{i,j}}{\rho^{j+1,j}}\right) \\ &= C_m \cdot \frac{V^{n+1}_{i,j} - V^n_{i,j}}{\Delta t} + I^n_{i,j}, \end{aligned} \tag{2.7}$$

where $I^n_{i,j}$ is the difference of I_{ion} and I_{stim}, $\rho^{i-1,i}$ is the resistance between neurons at position (i, j) and $(i-1, j)$, and analogously $\rho^{j-1,j}$ is the resistance between neurons at position (i, j) and $(i, j-1)$. Equation 2.7 was used to solve the final system.

It is important to say that a unidirectional boundary condition was used for the x_2 axis. Consequently, the system can be described topologically **not** as a torus but as an open cylinder. The unidirectional conduction boundaries are biologically plausible (Schöll and Schuster (2008), subsection *Reentry*). To illustrate, the action potential can travel from $V_{g_y,j}$ to $V_{1,j}$ (where g_y is the number of simulated cells in x_2 dimension) but **not** vice versa. Mathematically speaking, the partial derivative term in Equation 2.7 transforms for the $i = 1$ coordinate likewise:

$$\begin{aligned} V^{n+1}_{1,j} = \cdots + \frac{1}{Sv\Delta y^2}\left(\frac{V^n_{g_y,j} - V^n_{1,j}}{\rho^{g_y,j}} + \frac{V^n_{2,j} - V^n_{i,j}}{\rho^{2,j}}\right)\cdots, \\ \forall j \text{ where } i = 1. \end{aligned} \tag{2.8}$$

As mentioned beforehand, a smaller proof-of-concept version of the model was used. I simulated a small 2D rectangle of the cardiac cells with dimensions:

$$\begin{aligned} w &= dx \cdot g_x \\ &= 0.02 \cdot 8 = 0.16cm = 16mm, \text{ and} \\ h &= dy \cdot g_y \\ &= 0.02 \cdot 40 = 0.8cm = 80mm \end{aligned} \tag{2.9}$$

Where w stands for width, h for height, $d_{\{x,y\}}$ for spatial discretization, and $g_{\{x,y\}}$ for number of cell simulated in the respective dimensions.

For visually-oriented readers notice Figure 2.2. The figure shows how the action potential wave propagates around the healthy heart model. Sub-figure 2.2a shows the initiation of the heartbeat in a cell with coordinates $(1, 1)$ (the most top left cell). Sub-figures 2.2a-2.2d display the propagation of the action potential from atria (upper-part of the model) to ventricles (lower-part of the model). Due to the minification of the heart, it takes the action potential wave only $\approx 117ms$ to travel from the top to the bottom. Lastly, the sub-figure 2.2e illustrates how the uni-directional boundary condition works. The action potential wave loops from the bottom back to the top.

In order to introduce some heterogeneity to the model, a special resistivity mask ρ was implemented (notice sub-figure 2.4e for visualization of the resistivity mask). The resistivity mask determines the speed and strength of the propagation of the action potential. Higher resistivity results in a slower action potential propagation and vice versa. The resistivity mask represents scarred heart tissue, which can be one of the causes of arrhythmia.

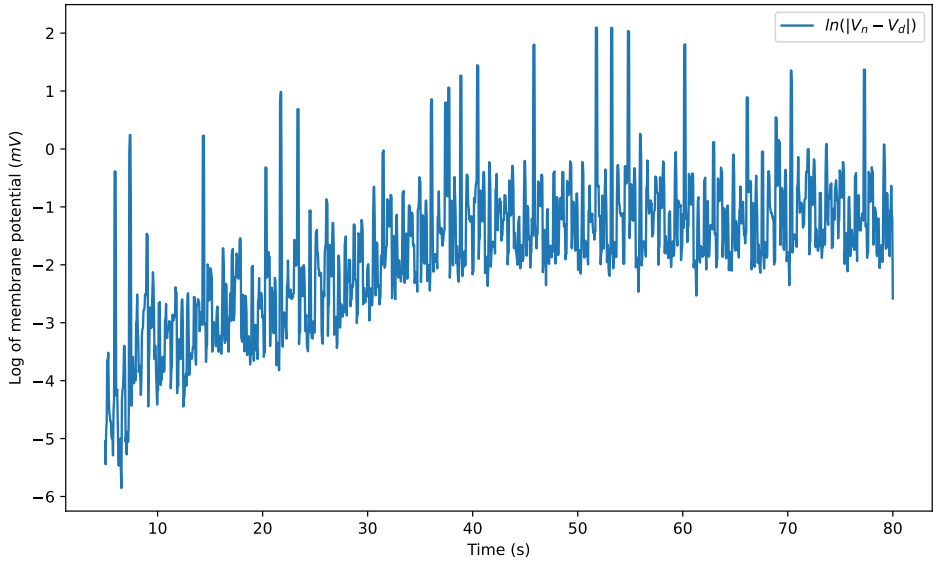


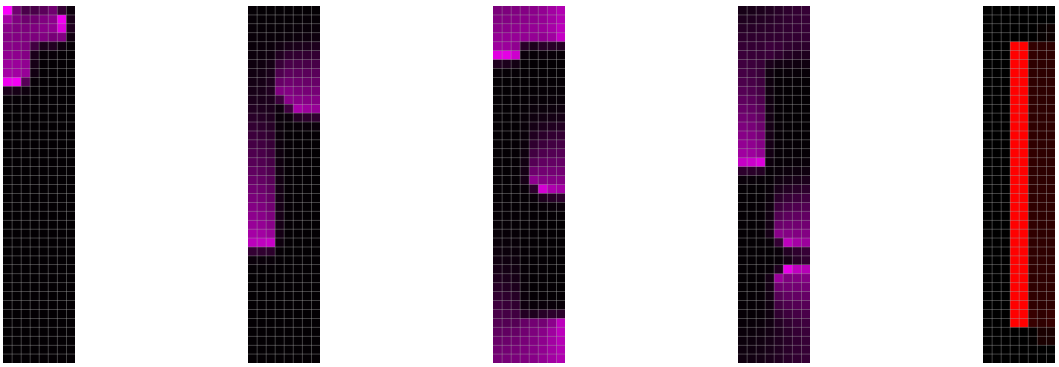
Figure 2.3: Log difference between two heart state trajectories, where the starting conditions differed only by $10^{-4}mV$

The introduction of the resistivity mask made the heart chaotic. This was tested by simulating two heart models with two different starting conditions V_n and V_d. Notice that

$$V_d = V_n + \epsilon, \tag{2.10}$$

where ϵ is a vector with magnitude $||\epsilon|| = 10^{-4}mV$. Figure 2.3 shows the log difference between the two heart's state trajectories after simulating both of them for 80 seconds. The two trajectories separate. This gives an intuitive suggestion that the scarred heart model has chaotic dynamics.

The precise mask together with the simulation code is available on my Github repository (Veselý, 2021).



(a) Frame 1 **(b) Frame 2** **(c) Frame 3** **(d) Frame 4** **(e) Resistivity mask**

Figure 2.4: Showcase of the action potential wave propagating around the scarred heart tissue. Black cells correspond to the cardiomyocytes that have a resting membrane voltage and purple cells signify an occurring action potential. Last sub-figure shows the resistivity mask ρ (more red colours signifies stronger resistance)

2.3 Van der Pol oscillator

Since the previously mentioned reaction-diffusion heart model has 2560 dimensions, a 2-dimensional Van der Pol oscillator (VPO) was also used to model the heart. The simplified VPO model provides an additional check whether the echo state network controller works or not.

VPO was used to model a limit cycle in electrical circuits but their domain extends beyond electrical engineering. VPO has been used to model two tectonic plates in a *geological fault*, left & right vocal cords, and the action potential of a neuron. VPO is modelled using a second-order differential equation with non-linear damping:

$$\frac{d^2x}{dt^2} - \mu(1 - x^2)\frac{dx}{dt} + x = F(t) \tag{2.11}$$

The unforced VPO is a special case when $F(t) = 0$. I will use the VPO to model the SA node of the heart. The SA node is a special bottleneck neuron which has the ability to spark an action potential independently - acting as a pacemaker (Klabunde, 2021). The pacemaker ability of the SA node could be interpreted as an external force $F(t)$ acting on the neuron. Consequently, a forced VPO was used with $F(t) = \beta \cdot cos(\omega \cdot t)$ as it is widely used in the literature (Marios, 2006; Cooper et al., 2017).

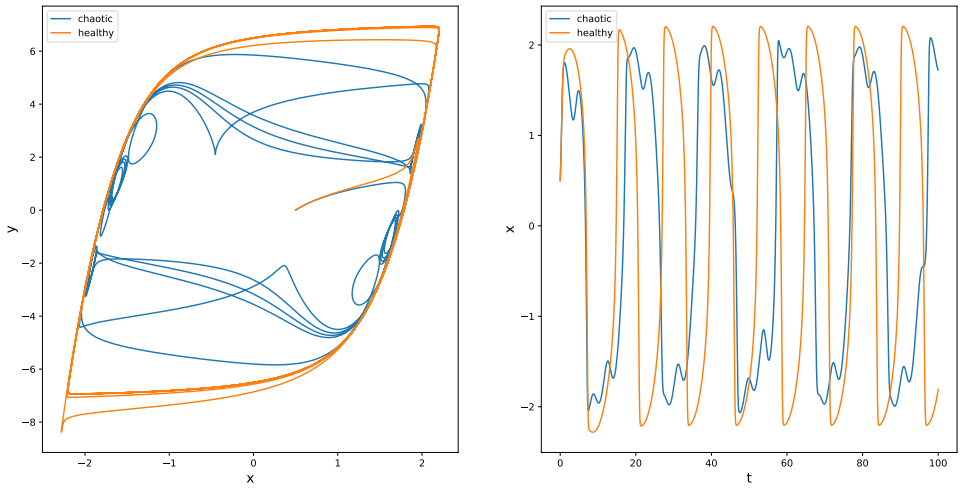


Figure 2.5: Comparison between a "healthy" ($\mu = 5, \beta = 5, \omega = 1$) VPO & "arrhythmic" chaotic ($\mu = 5, \beta = 5, \omega = 3.37015$) VPO

Equation 2.11 can be equivalently rewritten into two-dimensional system of differential equations using a Liénard system transformation

$$\begin{aligned} \dot{x} &= y - \mu\left(\frac{x^3}{3} - x\right), \\ \dot{y} &= -x + F(t). \end{aligned} \tag{2.12}$$

It needs to be said that representing the heart, an unimaginably complex organ, using Equation 2.12 is a gross oversimplification. This thesis will try to control a chaotic VPO as mere evidence (and/or as a baby step) that the ESN controller is working.

Most choices of the parameters μ, β, ω lead to a dynamical system with a stable orbit atractor. Figure 2.5 shows the state trajectory of the VPO as system as defined in Equation 2.12 evolved using a forward Euler algorithm with $dt = 0.02, \mu = 5, \beta = 5, \omega = 1, x_0 = 0.5, y_0 = 0.0$. As stated in Marios (2006) setting $\mu = 5, \beta = 5, \omega = 3.37015$ leads to creation of an chaotic atractor. The dynamics of chaotic VPO are also seen in Figure 2.5.

Introducing $\omega = 3.37015$ made the oscillator chaotic. This was tested by simulating two oscillators with two different starting conditions s_n and s_d. Notice that

$$V_d = V_n + \epsilon, \tag{2.13}$$

where ϵ is a vector with magnitude $||\epsilon|| = 10^{-5}$. Figure 2.6 shows the log difference between the two evolved states and providing an intuition that the defining $\omega = 3.37015$ indeed brings chaos to the system.

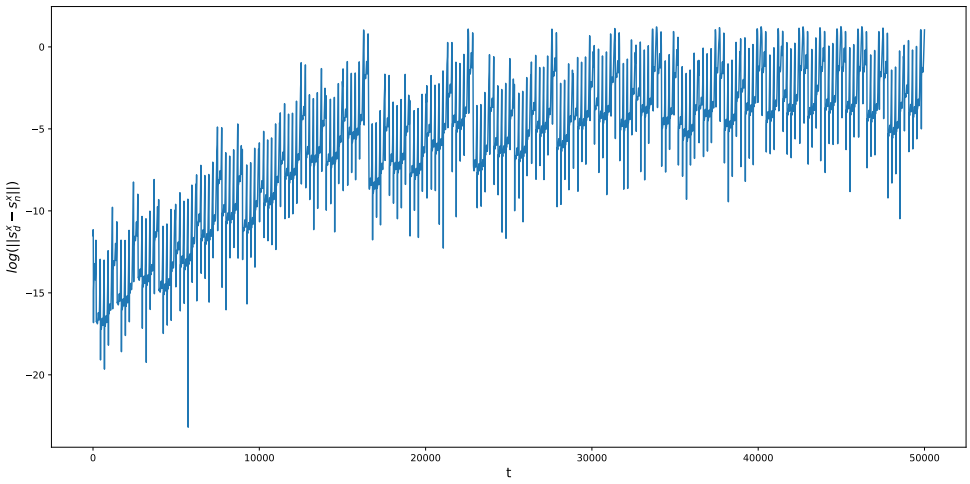
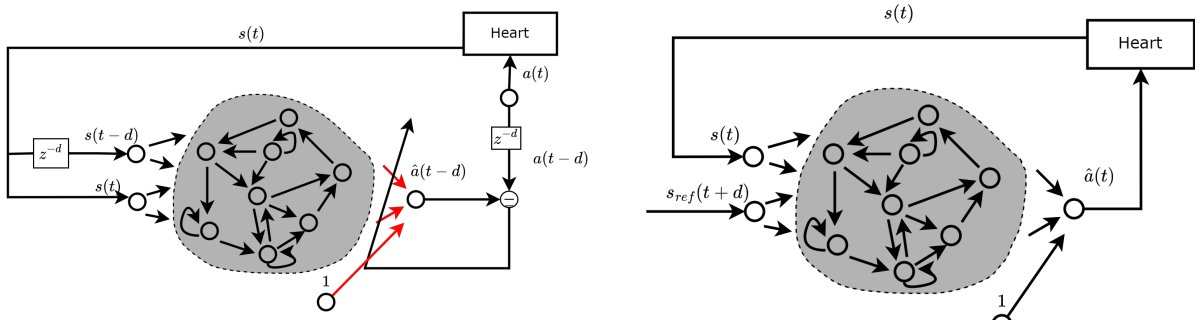


Figure 2.6: Log difference between two heart state trajectories, where the starting conditions differed only by 10^{-5}



(a) Training phase

(b) Exploiting phase

Figure 2.7: Sketch of the training and exploiting procedures. (a): the network associates how actions $a(t)$ will influence the transition from $s(t-d)$ to $s(t)$. The trainable weights are shown in red. (b): The network is given the current heart state $s(t)$ and desired state $s_{ref}(t+d)$ and is expected to output an appropriate action $\hat{a}(t)$ that will get the heart to the desired state in d steps. Figure adapted from Jaeger and Haas (2004)

2.4 Echo state network as controller

2.4.1 Dynamics

The most simple architecture of controller ESN can be found in Figure 2.7. During the training phase, the controller receives two states as input: the past $s(t-d)$ state and the present state $s(t)$, where $d \in \mathbb{N}$ is an integer and the optimal d needs to optimized for every task. Usually, the controller receives the full model state (all state variables), however, in my case, I tried to replicate the realistic settings of a pacemaker which does not "see" the entire heart. Consequently, the state variable s is a proper subset of a full-state (the precise definition of s will differ and is be explained in the Methodology section 3).

The concatenation of the input (noted as $[s(t-d); s(t)]$) gets projected to a reservoir using matrix W_{in}. The reservoir $x(t)$ is a non-linear dynamical system where the internal weights are initialised semi-randomly. The following equation represents the discrete update step of the reservoir:

$$x(t+1) = x(t) \cdot (1-\alpha) + \alpha \cdot tanh(W_{in} \cdot [s(t-d); s(t)] + W \cdot x(t)). \tag{2.14}$$

If not specified otherwise the input weights W_{in} are created by sampling a normal distribution with parameters (μ_{in}, σ_{in})

The weight matrix W is a random sparse matrix with the spectral radius (usually denoted as ρ but I will use r_s) lower or equal to 1. The spectral radius is the magnitude of the biggest eigenvector of matrix W. It

has been found that setting $r_s \leq 1$ is usually enough to grant the reservoir the *echo state property* (there are other algebraic conditions that need to be satisfied and avid mathematicians can read about them in Buehner and Young (2006)). The echo state property (ESP) ensures that the initial state of the reservoir $x(0)$ which is set randomly (therefore, bearing no information about the task) will be washed out.

More rigorously, two reservoir's trajectories x_1 and x_2 with the same internal weight matrices W but with different initial states driven using the same input sequence $[s(t-d); s(t)]$ will converge arbitrarily close to each other after some time. Mathematically speaking, there exists t_w such that $|x_1(t_w) - x_2(t_w)| \leq \epsilon$, where $\epsilon > 0$. The time t_w is called a *washout period* (since the initial differences are washed out). Figure 2.8 shows the echo state property in a computer simulation.

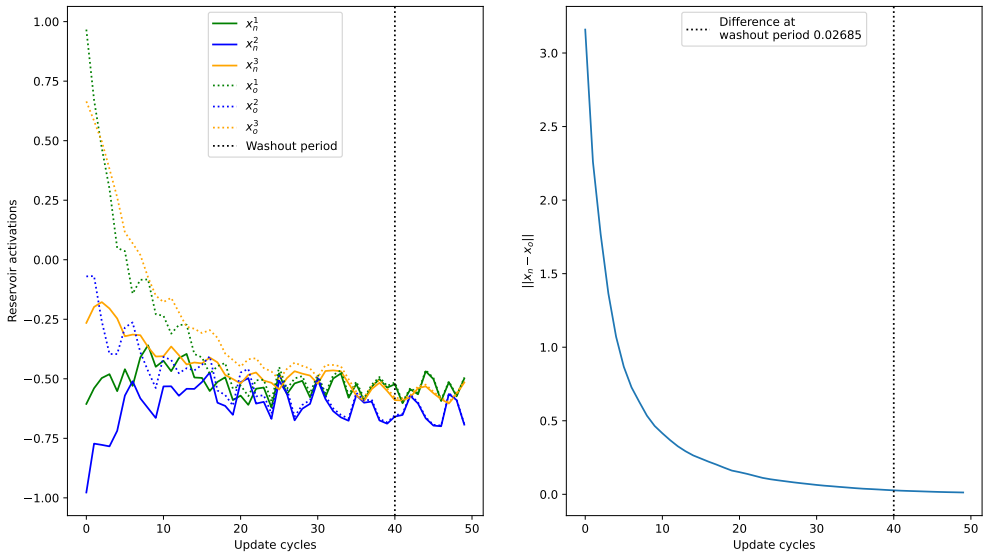


Figure 2.8: Showcase of the *echo state property*. Two almost identical reservoirs x_o and x_n, were driven with the same input sequence. The only difference was the initial conditions of the reservoirs - which were set randomly. The left plot shows how the individual neurons converged to each other. The right plot shows the Euclidean distance between the reservoirs. Washout period 40 is marked on both plots

The leaky mask α is a vector with the same dimension as x. It is created randomly by sampling a uniform distribution with bounds (min, max), where $0 \leq min \leq max \leq 1$. A leaky mask transforms the neurons into leaky integrator neurons. Notice in Equation 2.14 that setting the a close to 0 for x_n minimizes the influence of the input and other neurons on x_n. Thus, neuron x_n reacts slowly to the immediate changes and retains information for a longer period of time. Vice verse setting a close to 0 for x_n makes the neuron more reactive to new changes. By making the leaky mask diverse the reservoir can react to both short-term & long-term effects seen in the input sequence.

The output is calculated as a linear combinations of the reservoir state $x(t)$, input $[s(t-d); s(t)]$, and bias term 1.

$$\hat{a}(t) = W_{out} \cdot [x(t); s(t-d); s(t); 1]. \tag{2.15}$$

The controller learns how to output an action sequence such that if it is applied to the heart model, the heart will transition from $s(t-d)$ to $s(t)$. $a(t)$ is a vector of electrical currents (mA). The currents are applied to the heart model via the term I_{stim} from Equation 2.6. In order to make the task more realistic, the controller could not control the entire I_{stim} matrix. The controller was given 4 evenly separated "injection points". Figure 2.9 shows the points visually. All other parts of the I_{stim} matrix were set to 0.

2.4.2 Training

The training procedure is shown in sub-figure 2.7a. It is done by driving the echo state network using a pre-recorded dataset of states s and actions a. The creation process of the dataset is described in the methodology section. During training the readout neurons (the concatenation of the input, reservoir state, and the bias term) are saved into a collection matrix $C \in \mathbb{R}^{(2\cdot|s|+|x|+1)\times T}$, where $T = N - washout$ is the

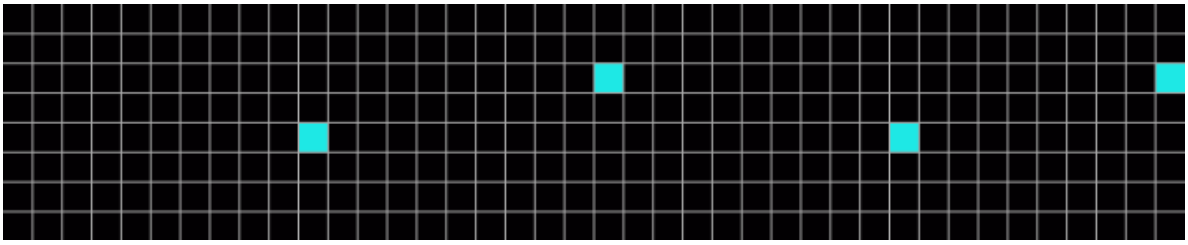


Figure 2.9: Showcase of the controller injection points. The cyan cells indicate where the controller was able to stimulate the heart tissue. The plot is rotated $90°$ degrees clockwise

number of training samples minus the washout period. All the training states before the washout period are disregarded.

Let $A \in \mathbb{R}^{|a| \times T}$ be a matrix containing the teacher variable. Notice that in Equation 2.15 the $\hat{a}$ has a linear relation to the collection matrix C. Therefore the output weights W_{out} can be calculated using linear regression which minimizes the difference between a and $\hat{a}$ as defined by the mean-squared-error function.

$$W_{out} = \underset{M \in \mathbb{R}^{|a| \times (2 \cdot |s| + |x| + 1)}}{\arg\min} \frac{1}{|a|} \left(\sum_{i=1}^{|a|} (a_i - \hat{a}_i)^2 + \beta \cdot |M_i|^2 \right), \tag{2.16}$$

where $\beta \cdot |M_i|^2$ is a Tikhonov regularization term. Equation 2.16 is also known as a *ridge regression* and can be solved analytically with

$$W_{out} = AC^T(CC^T + \beta I)^{-1}, \tag{2.17}$$

where I is an identity matrix and $\beta \in \mathbb{R}$ to be found by hyperparameter optimization.

After a successful training the controller can be exploited by replacing the past state $s(t-d)$ with current state of the heart $s(t)$ and second input by the reference state $s_{ref}(t+d)$. Notice that $s(t)$ cannot come from a pre-recorded dataset since the controller action influence the state of the heart $s(t)$ during the exploitation. The reference sequence is an arbitrary sequence of states which the model should follow. Since I want to treat the arrhythmic heart I used a recording of the non-scarred healthy heart as the reference sequence (look at Figure 2.2 as a reminder of the healthy heart). Ideally, the controller produces a sequence of actions $\hat{a}$ such that if they are applied to the heart model the heart follows the reference sequence/trajectory.

Notice that the ESN controller is a hybrid between an open & closed loop controller. During the training procedure, the controller acts as an open loop system since the actual actions produced by the ESN $\hat{a}$ are not fed to the model of the heart. This yields two benefits. Firstly, the simulation of the heart model is computationally very expensive - simulating the heart for 30 minutes requires $\approx$ 2 days on a high-performant computing cluster (Peregrine). Thus, an dataset of pre-recorded states $(s(0), s(1), ..., s(N))$ & actions $(a(0), a(1), ..., a(N))$ can be used to speed up significantly the training procedure. Secondly, since the optimal output weights W_{out} are initially unknown, the predicted actions $(\hat{a}(0), \hat{a}(1), ..., \hat{a}(N))$ bear almost no meaning during the training procedure. Therefore, plugging them into the heart would produce non-interesting dynamics of the heart and ultimately lead to poor learning. The importance of the applied actions is explained in sub-section 3.1. After the training and the update of the output weights W_{out}, the controller becomes a closed-loop system, therefore, a "live" heart model is required for the exploitation phase.

3 Methodology

3.1 Dataset creation

As mentioned before training the controller is a supervised task and since there are no available datasets I needed to create one. The goal is to produce a sequence of action $(a(0), a(1), a(2), ..., a(N))$ such that if they are applied to the heart they produce a sequence of states $(s(0), s(1), s(2), ..., s(N))$. Ideally, the sequences of states s should be representative of all possible dynamics that can occur in the heart model. Consequently, it is crucial to select a proper sequence of actions a that produce various modes of functioning of the heart model.

Additionally, it is also necessary to select a the right sampling frequency f_s that matches the natural frequencies of the heart model. Choosing f_s too high would lead to inefficient use of the *reservoir* in the echo

state network controller. Driving the *reservoir* with highly temporally correlated data hinders the memory capacity of the controller since any effects of past data get minimized by re-applying Equation 2.14 with very similar input. On contrary, setting f_s too low would skip over important updates in the heart model. Therefore the input would reassemble a random variable instead of a temporally correlated signal. Thus, 3 datasets were created with sampling frequencies being $f_s = \{100Hz, 50Hz, 20Hz\}$. Avid readers will notice that the controller frequency is significantly lower than the update frequency of the heart model (which is $\approx 67000Hz$). This creates an arbitrary decision of what action to apply between the update step of the controller. I settled on applying the last action produced that was produced by the controller. To illustrate, let a_1 be an action produced by the controller at time t_1. My algorithm consistently performs action a_1 at timesteps $(t_1, t_1 + dt, t_1 + 2 \cdot dt, t_1 + 3 \cdot dt, ..., t_1 + n \cdot dt)$ where n can be approximated to the ratio of the controller period T_c and the model period T_m; $n \approx \frac{T_c}{T_m} \approx \{666, 1333, 3333\}$ for the respective sampling frequencies.

Recall that each of the modelled heart neurons consist of 8 state variables (described in sub-section 2.1), however, only the membrane voltage V was fed to the controller as the input. The heart consists of $8 \times 40 = 320$ neuron and a state is 320 dimensional vector $s \in \mathbb{R}^{320}$. The membrane voltages were clamped into range $(-90mV, 50mV)$ and the input was normalised to range $(-1, 1)$. The actions were normalised to range $(0, 1)$.

3.1.1 Action selection

The algorithm for the action selection goes as follows. A 10.2-second block of white noise was created. A high-pass filter was applied to the white noise. The cutoff frequency was selected from a uniform distribution with bounds $(2Hz, 5Hz)$. Then, first, 0.2 seconds were dropped from the block in order to remove the high-pass filter artefacts. In order to allow only for negative stimulation current, the actions were clamped in range $(0, \infty)$ (if confused why the range is not $(-\infty, 0)$ notice the negative term $-I_{stim}$ in Equation 2.6). A visual example of generated actions can be found in the upper plot in Figure 3.1.

Let $a(i) = [31mA, 30mA, 30mA]$ be an action produced at time i. Normally, the action vector a is 4-dimensional vector but this example will use a 3-dimensional action vector for better clarity. Assume that there are no ongoing action potentials happening right before time i. Actions $a(i)$ will start 3 individual action potential cascaded (at their respective injection points). The cascades tend to cancel out as they meet each other. This behaviour was commonly seen and is unwanted. Consequently, an amplification factor γ was introduced. The idea behind γ is to select the strongest action and amplify it whilst minimizing all other actions and maintaining the same magnitude of the action vector. To illustrate, let $\gamma = 3.0$ and z_γ the amplification function, then

$$z_{3.0}(a(i)) \approx [48, 15, 15]. \tag{3.1}$$

Function $z_\gamma(a)$ can be also thought of as a rotation of the vector a on a hyper-sphere towards the axis that represents the strongest action. The precise definition of z_γ is very convoluted (due to performance reasons), thus, only certain properties of the function will be listed here. The implementation of z_γ can be found on my Github Veselý (2021) in `noise.py`. It holds that as γ approaches infinity the output of the function z will be a vector fully rotated towards the axis with the highest action.

$$\lim_{\gamma \to \infty} z_\gamma(a) = [\text{all 0 except the highest action which is } |a|]. \tag{3.2}$$

On the other hand setting $\gamma = 1$ yields an identity transformation.

$$z_1(a) = a. \tag{3.3}$$

Amplification parameter γ was sampled from a uniform distribution with bounds $(1, 3)$. Figure 3.1 shows the effect of the amplification.

The action block was re-initialized every 10 seconds of the simulation time. The final datasets were created by simulating the heart model for 40 hours which yielded 2 400 000 training samples containing a tuple of the current state and the action applied at that state $(s(t), a(t))$. One-tenth of the dataset was used for validation and the remaining was used for training.

For the exploitation phase, a reference state trajectory was acquired by "recording" the non-scarred heart (with homogenous resistivity mask ρ) for 30 seconds with the respective frequencies $f_s = \{100Hz, 50Hz, 20Hz\}$. Figure 2.2 shows the healthy heart dynamics and Figure 3.2 displays an part of the reference signal.

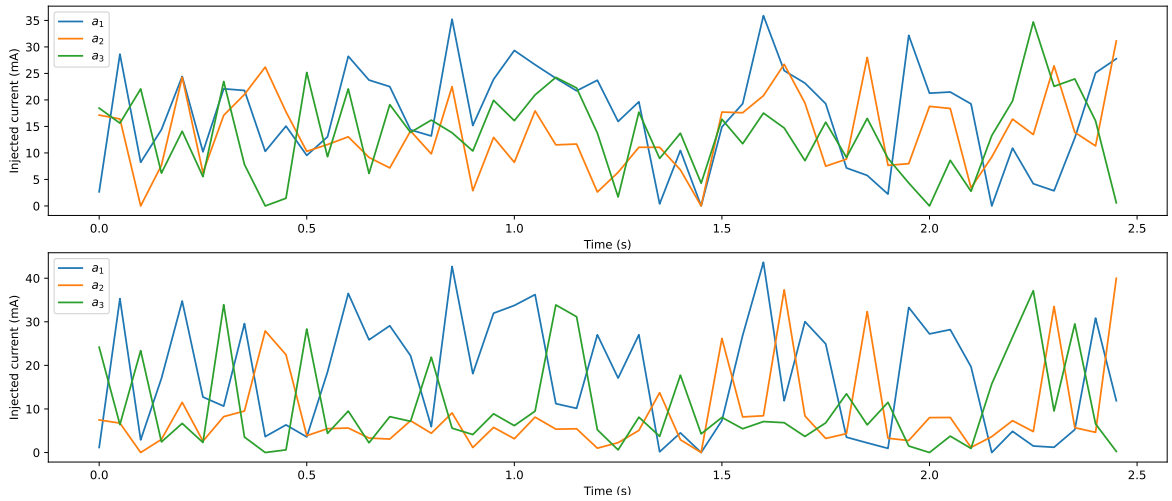


Figure 3.1: Example of selected actions and demonstration of the action amplification algorithm. The upper and lower plots show the same actions, however, the upper plot has amplification factor $\gamma = 1$ (no amplification) and the lower plot has amplification factor $\gamma = 3$

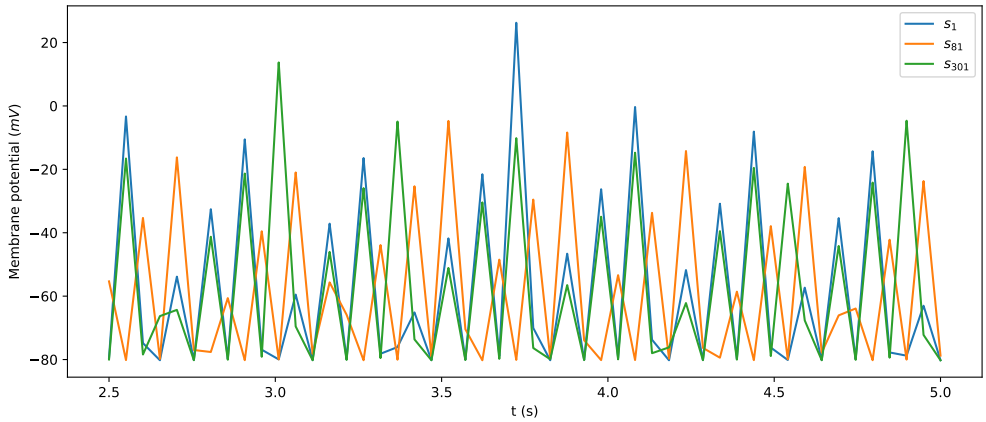


Figure 3.2: Example of the reference signal s_{ref} using $f_s = 100Hz$, the recording is done based on the healthy heart version

3.2 Controller designs

The simple version of the echo state network controller as described in sub-section 2.4 is not sufficient to control the 2560 dimensional heart model (the Van der Pol system is an exception and is explained in sub-section 3.3). The point of the reservoir is to reverberate, whilst driven with the input, in such a way that it provides additional information about the heart model. Randomly created reservoir weights rely only on a random chance of getting a useful reverberation, thus, such reservoirs need to be significantly bigger than reservoirs with a thoughtfully crafted architecture. Three designs will be presented that try to achieve better performance than the vanilla-flavoured reservoir: the full-state design which mimics the heart model, the local design which introduces multiple reservoirs, and the PCA design which pre-processes the input using a principal component analysis.

3.2.1 Full-state design

The full-state design (later just FSD) tries to compensate for the fact that the controller does not see the entire state. The idea of FSD is to mimic closely the real-life scenario. Surgical pacemakers have access only to a partial state of the heart (in some cases only to one or two neurons). Therefore, a state in the FSD design can be defined as a subset of the membrane potential matrix ($s \subsetneq V$ in Equation 2.6), more precisely a state $s \in \mathbb{R}^8$ at time t can be defined as

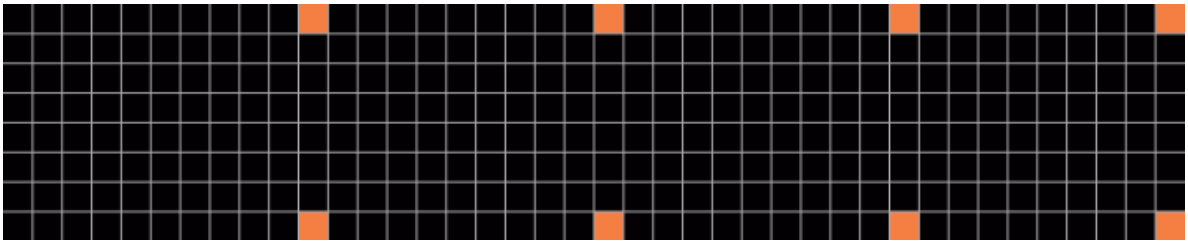


Figure 3.3: Placement of the "detection points" on the heart tissue (indicated by orange). FSD design was only able to perceive the membrane potential (V) via these points. The plot is rotated 90° degrees clockwise

$$s(t) = \begin{bmatrix} V_{1,1}(t) \\ V_{1,8}(t) \\ V_{10,1}(t) \\ V_{10,8}(t) \\ V_{20,1}(t) \\ V_{20,8}(t) \\ V_{30,1}(t) \\ V_{30,8}(t). \end{bmatrix} \tag{3.4}$$

Figure 3.3 shows the placement of the probes/detection points on the heart tissue. The placement was selected such that they were covering as much area as possible whilst being as far apart from each other as possible. There are no probes on the bottom row of the heart (a.k.a $V_{40,j}$) because there is an unidirectional boundary condition connecting neurons $(40, j)$ and $(1, j)$ (as described in sub-section 2.2) - making them topologically close.

Reservoir weights

FSD introduces a new reservoir which topologically mimics the heart model. The reservoir has 2560 neurons where each neuron tries to mirror one of the state variables of the heart model. Let $W_{heart} \in \mathbb{R}^{2560\times 2560}$ be the weight matrix for the heart model, x_{heart} be the neuron activation of the heart reservoir, $V_{i,j}$ be an membrane potential at i, j, and $s^n_{i,j}$ one of the other 7 state variables. Notice in Equations 2.6 and 2.7 that only the membrane potential V has a direct influence on the neighbouring membrane potentials. Other state variables (such as α_h, β_h, or $[Ca]_i$) are isolated and interact only locally. W_{heart} has to capture these interactions. I defined, arbitrarily, that the ideal x_{heart} state should hold the following structure:

$$x_{heart}(t) = \begin{bmatrix} V_{1,1}(t) \\ s^1_{1,1}(t) \\ s^2_{1,1}(t) \\ s^3_{1,1}(t) \\ s^4_{1,1}(t) \\ s^5_{1,1}(t) \\ s^6_{1,1}(t) \\ s^7_{1,1}(t) \\ V_{1,2}(t) \\ s^1_{1,2}(t) \\ s^2_{1,2}(t) \\ s^3_{1,2}(t) \\ \vdots \\ s^6_{40,8}(t) \\ s^7_{40,8}(t) \end{bmatrix}. \tag{3.5}$$

W_{heart} needs to be designed in a special way in order to enforce the structure. W_{heart} is a sparse matrix defined with these rules:

1. Every neuron needs to be connected to itself with a positive weight since every neuron represents a state variable which is dependent on its previous values.
2. If neurons x_i and x_j represent two membrane potentials V which are neighbours in the heart model - both need to be connected to each other with a positive weight.

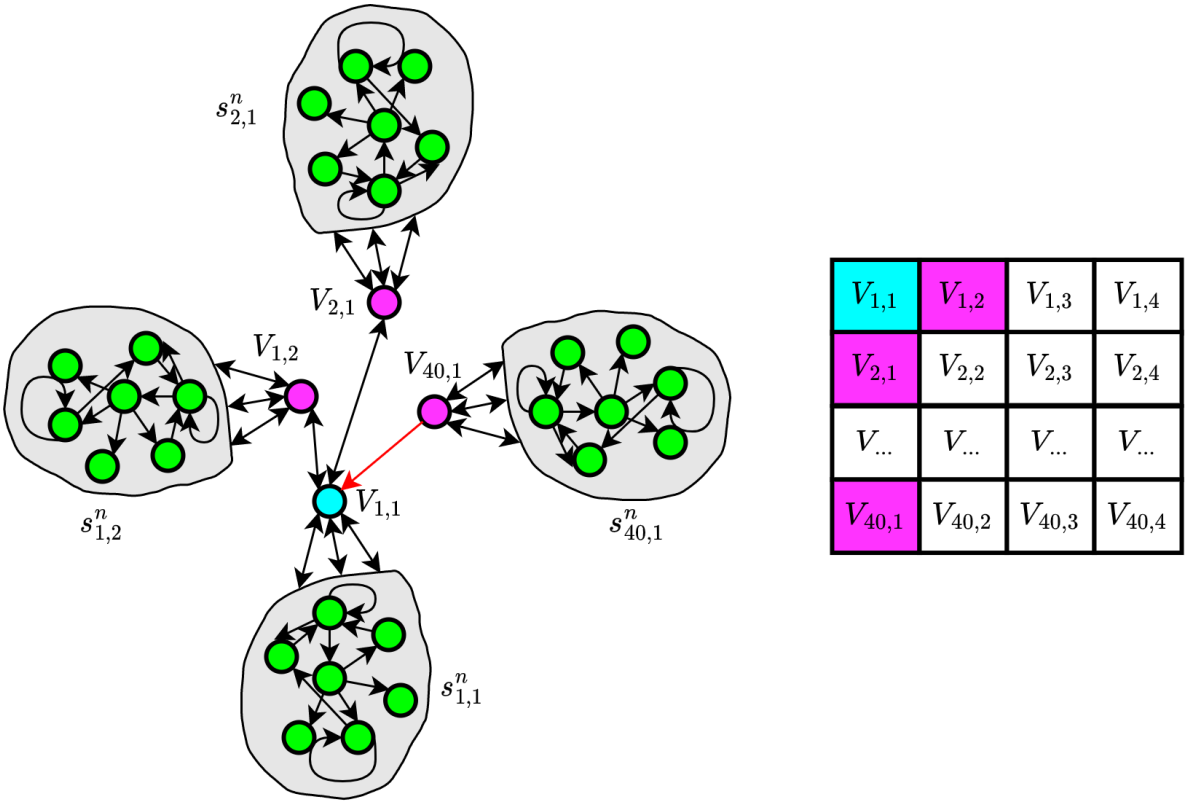


Figure 3.4: Sketch of the heart reservoir x_{heart} connectivity. For explanation see text

3. If neuron x_i represents any last row membrane potential $V_{40,j}$ and neurons x_j represent the first-row membrane potential with the same column $V_{1,j}$ - there needs to be a positive connection going from x_i to x_j but not vice versa. This rule mimics the unidirectional boundary condition in the heart model.

4. All neurons representing the 8 state variables in one position (i, j) need to be fully connected with random weights to each other.

5. Other entries of W_{heart} are set to 0.

To illustrate the connections, Figure 3.4 displays the connectivity for the heart cell located at $(1, 1)$. The gray regions are the fully connected neurons which represents the 7 state variables (all of the state variables except V). Every neuron from the gray area is also connected to its membrane potential neuron residing at the same location. Neuron $V_{1,1}$ is **bi**directionally connected with $V_{2,1}$ and $V_{1,2}$. There is a **uni**directional connection from $V_{40,1}$ to $V_{1,1}$.

All the weights were sampled from a normal distribution which was defined differently for the positive weights (μ^+, sd^+) and the general weights (μ, sd).

A parameter `adjecencyTolerance` the number of connections between neurons corresponding to the membrane potential using a Manhattan distance. If `adjecencyTolerance` was set to 1 only topological neighbours with Manhattan distance 1 were connected. To illustrate, setting `adjecencyTolerance` to 2 would result in $V_{1,1}$ to be connected with $(V_{1,2}, V_{1,3}, V_{2,1}, V_{2,2}, V_{3,1}, V_{40,1}, V_{40,2}, V_{39,1})$ (consult the right part of Figure 3.4 for visual aid). If $w_{i,j}$ would be a weight connecting two different membrane potential neurons then the weight would be re-scaled based on the Manhattan distance likewise

$$w_{i,j} := \frac{w_{i,j}}{l+1}, \tag{3.6}$$

where l is the Manhattan distance between the two heart neurons. Consequently, further neighbours have less influence on each other.

The other reservoir x fulfilled a more general purpose and was initialised independently from the heart reservoir. Let W be a sparse weight matrix for the reservoir x. The parameter `connectivity` determined the portion of non-zero elements and the non zero-elements were sampled from a uniform distribution with bounds $(-\sigma, \sigma)$. The heart reservoir was densely connected to the general reservoir but there were no connections going the other way since they would destroy the dynamics of the heart reservoir.

Both weight matrices W_{heart} and W were re-scaled such that their spectral radii were below 1; $r_s < 1$. The precise definition of the parameters can be found in Section 4.

Leaky mask

Let α be the leaky mask for the general purpose reservoir x and α_{heart} for the heart reservoir x_{heart}, then α was sampled from a uniform distribution with bounds $(\alpha^{min}, \alpha^{max})$. The mask α needs to be diverse in order to be able to react to long-term changes as well as short-term changes.

On the other hand, α_{heart} was split into two parts. Elements in α_{heart} which corresponded to the membrane potential neurons V were set homogeneously to a constant α^V_{heart} since the heart model should function on the same timescale. A small exception to this rule is the heterogeneous resistivity mask ρ. The idea is that the resistivity mask heterogeneity is already captured by the variations of different weight strengths in W_{heart}. It is obvious that the semi-random initialization of W_{heart} will never match the resistivity mask ρ. This was done on purpose in order to mimic a real-life scenario where the pacemaker does not possess information about the ρ. After all, the heart reservoir needs to be only an approximation of the heart model.

Elements in α_{heart} corresponding to the other 7 state variables were sampled from uniform distribution with bounds $(\alpha^{min}_{heart}, \alpha^{max}_{heart})$. All of the state variables work on haste time scales, thus, the bounds are much less diverse and closer to 1 than in the general leaky mask.

Input weight matrix

The input weights matrix W_{in} connects only the past state ($s(t-d)$ during the training period and $s(t)$ during the exploitation period) to the heart reservoir. W_{in} respects the topology of the input state and the topology of the heart reservoir, meaning that s_z representing membrane potential $V_{i,j}$ would get correctly relayed to a reservoir neuron x_o which represents the same membrane potential $V_{i,j}$. All the input weights connecting the state to the heart reservoir were strictly positive (so that the physical meaning of the input is not lost) and were sampled from a normal distribution $(\mu^{in}_{heart}, sd^{in}_{heart})$.

Both the future state $s(t)$ and the past state $s(t-d)$ were connected fully to the general reservoir. Those input weights were sampled from a normal distribution (μ^{in}, sd^{in}).

Figure 3.5 shows a summary of the connectivity of the FSD controller during the training period.

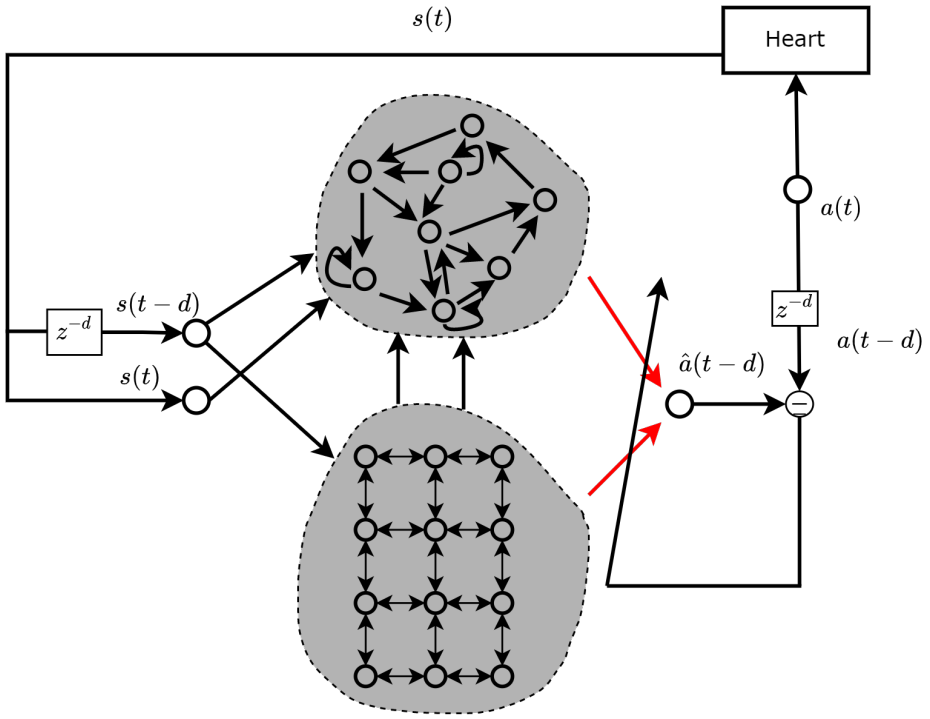


Figure 3.5: Overview the full state design controller during the training period

3.2.2 Local design

The local design follows the state of the approach to processing a spatial-chaotic system by exploiting the fact that these systems have similar (sometimes completely the same) dynamics across the spatial dimensions. The idea is to have numerous medium size reservoirs each being responsible only for a small spatial region.

The architectures found in Vlachas et al. (2020) and Pathak et al. (2018) were modified. Their echo state networks were responsible for predicting the next evolution of a spatiotemporal chaotic model. Their ESNs receive as input a last state of the system and were asked to output the next state. They dedicated a new reservoir to each "discretized pixel" and the reservoir was responsible to output a new pixel at the same spatial location. Notice that they could only do that because the input had the same structure as the output - this is not the case with the ESN described in this paper.

The general idea of those designs still holds. I dedicated one reservoir for each predicted action a_n and the reservoir was responsible for processing the states which were spatially close to the "injection point" for the given action. Due to having 4 "injection points", 4 reservoirs were created. Sceptical readers can be curious whether having 4 reservoirs instead of 1 could help. The local design has two major benefits. Firstly, the 4 reservoirs are always faster than 1 reservoir with the same number of neurons due to the quadratic scaling of the internal weight matrix W and, thus, the time complexity of the operations that include the weight matrix W. Secondly, the reservoirs are independent of each other and can be therefore trained in parallel.

The local design is more lenient of what is possible in the real world, as a consequence of that, the state $s \in \mathbb{R}^{320}$ included the entire membrane potential matrix $s(t) = V(t)$, whilst the controller was still oblivious to the other 7 state variables.

Reservoir(s) weights
I will be referring to each of the reservoirs as x and their weight matrices as W since all of the reservoirs are created according to the same rules. W is a fully connected matrix where the elements come from a normal distribution (μ_W, sd_W). A sparse matrix is usually the most popular choice since it creates localized groups of neurons where each of the groups is only slightly dependent on the others. These localized groups should ideally process different parts of the input - providing the reservoir with diverse information. Since the local design already subdivides the input the importance of the sparse matrix is minimized.

The internal weights are scaled such that their spectral radius is below 1; $r_s < 1$.

Leaky mask(s)
The dynamic of the heart is similar but not the same across the heart tissue due to the heterogeneity of the resistivity mask ρ. Leaky masks were therefore created randomly by sampling an uniform distribution with bounds $(\alpha_{min}, \alpha_{max})$.

Input weights
Figure 3.6 shows the input wiring. A parameter `adjecencyInput` controls how many neurons are processed by one reservoir. It defines the maximal Manhattan distance from the injection point of the given reservoir to the membrane potential $V_{i,j}$ which is still processed by the reservoir. Figure 3.6 shows the connections using `adjecencyInput` $= 1$. For demonstration purposes, setting `adjecencyInput` to 0 makes the reservoir process only the $V_{i,j}$ corresponding to the same location as the "injection point". Setting `adjecencyInput` to 50 will cause an overlap of the input to the different reservoirs (the reservoirs still remain independent from each other).

A small penalty was introduced if the input $V_{i,j}$ was too far away from the injection point. Let w_k^{in} be the weight responsible for connecting the input $V_{i,j}$ to a reservoir then the penalty was applied following

$$w_k^{in} := \frac{w_k^{in}}{(l+1) \cdot 0.4}, \tag{3.7}$$

where l is the Manhattan distance between the $V_{i,j}$ and the injection point corresponding to the given reservoir.

Each of the reservoir was responsible for both the future state $s(t)$ and the past state $s(t-d)$. As figure 3.6 shows, both inputs were from the same topological location.

Lastly, as always the weights were sampled from a normal distribution.

3.2.3 Principal component analysis design

As mentioned before driving the reservoir with strongly correlated values can lead to less diverse signals in the reservoir and therefore lower the quality of the prediction. The state s, which is still equal to the membrane voltage matrix $(s(t) = V(t))$, has numerous correlations within itself. The principal component analysis design (later just PCAD) circumvents that by reducing the input dimension using PCA. However, the full membrane voltage matrix was still available for the action $\hat{a}$ computation. This was done by extending the collection matrix C (described in sub-section 2.4.2) with $V(t)$ without feeding it into the reservoir. Figure 3.8 shows the training period of the PCAD controller.

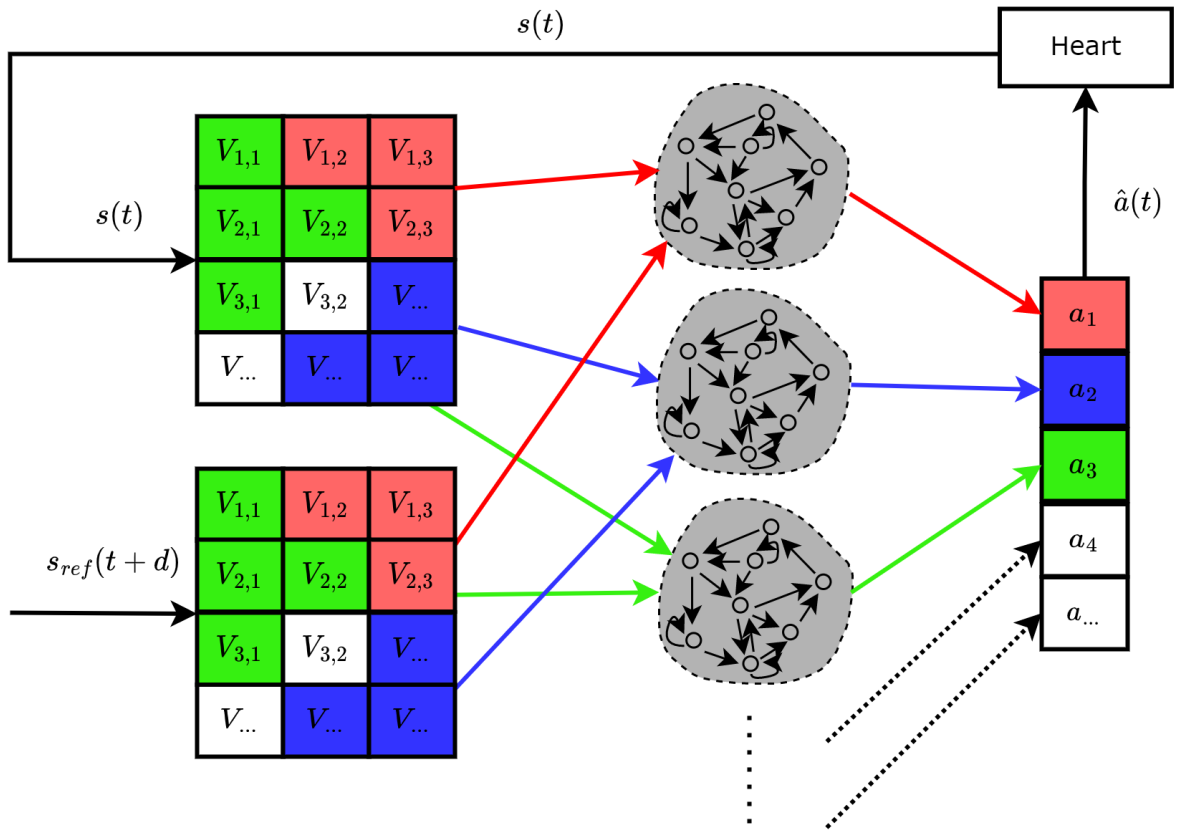


Figure 3.6: Local architecture design showcase. The architecture is demonstrated during the exploit period. The figure is only an approximation of the architecture, read to text for more information

The PCA transformation matrix was calculated on the training part of the data ($^8/_9$ of the entire dataset). It is usually not recommended to use PCA as a part of pre-processing since the axes of the maximal variance can change for different data (for example for the validation dataset). To avoid such an issue, I compared the difference between three PCA transformation matrices M^1, M^2, M^3. Let M^1 be the PCA transformation matrix that was calculated based on the training data, M^2 based on the entire dataset, and M^3 using 5000 states that were recorded during the testing scenario (exploitation period). Figure 3.7 shows the difference between the matrices. The difference between two matrices was calculated as an average of the euclidean distance between their column vectors

$$d(M^n, M^k) = \frac{1}{dim} \sum_{j=1}^{dim} ||M^n_{*,j} - M^k_{*,j}||, \tag{3.8}$$

where dim is the reduced number of dimensions. As seen from Figure 3.7, the biggest difference is ≈ 0.0035 meaning that the maximal axes of variance are only very slightly different, therefore, PCA can be safely used in this case as a pre-processing operation for the input.

Reservoir weights
The reservoir designs mentioned in the previous subsections cannot be used here since the PCA destroys the spatial property and the semantics of the input states s. A simple sparse reservoir was used with sparsity defined by the `connectivity` parameter and the weights were sampled from a normal distribution. The internal weights were scaled such that their spectral radius is below 1; $r_s < 1$.

Leaky mask
A diverse leaky mask was used that was sampled from a uniform distribution with bounds $(\alpha_{min}, \alpha_{max})$.

Input weights
Fully connected input weights W_{in} were created by sampling a normal distribution. A penalty was introduced, which controlled for the fact that not all of the PCA axes are equally important. Let $var(PCA_n)$ be the variance of the data spanning in the n axis. If one wants to reduce the dimensionality of the input to 30 using PCA the 30 most important axes need to be selected. If $var(PCA_j) > var(PCA_i)$ then the i-axis is

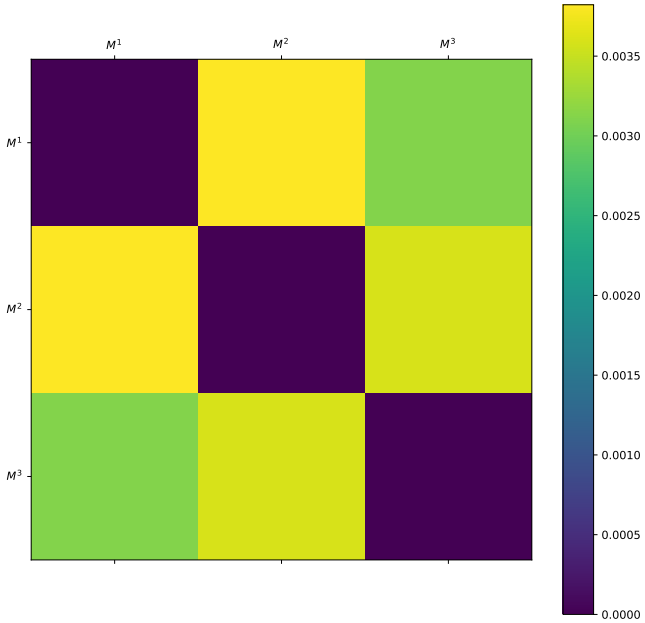


Figure 3.7: Difference between the PCA transformation matrices. M_1 was generated using training data, M_2 using both training and validation dataset, and M_3 using states that were seen in during testing

less "important" than the j-axis. Let m be the axis with the maximum variance and w_o be the input weight vector corresponding to the o-axis, then the input weight vector was penalized accordingly

$$w_o := w_o \cdot \frac{2 \cdot var(PCA_o)}{var(PCA_m)}. \tag{3.9}$$

3.2.4 Common modifiers

Two other techniques of enriching the reservoir signals were used which are not bound to any single design.

Squared reservoir
As described in Pathak et al. (2018) each network can be extended by another reservoir x_s which is a squared version of the original one; $x_s(t) = x^2(t)$, where the squaring procedure is an elementwise operation. The x_s reservoir does not get updated by Equation 2.14 but it always follows the original reservoir x. Squaring the reservoir introduces another non-linear transformation (second after the $tanh$) making it easier for the linear output layer to predict the action a. Secondly, it introduces $||x||$ new trainable weights. The squared reservoir is implemented by extending the collection matrix C (described in sub-section 2.4.2).

State difference
Equation 2.6 can be used to derive the optimal control equation - meaning the equation for the action matrix I_{stim} (this section will refer to it with a capital 'A'). Firstly, assuming $A = I_{stim}$ and

$$\leftrightarrow^n = \frac{V^n_{i-1,j} - V^n_{i,j}}{\rho^{i-1,i}} + \frac{V^n_{i+1,j} - V^n_{i,j}}{\rho^{i+1,i}}, \tag{3.10}$$

$$\updownarrow^n = \frac{V^n_{i,j-1} - V^n_{i,j}}{\rho^{j-1,j}} + \frac{V^n_{i,j+1} - V^n_{i,j}}{\rho^{j+1,j}}. \tag{3.11}$$

Equation 2.6 can be rewritten into

$$\left(\frac{1}{S_v \Delta x^2} (\leftrightarrow^n + \updownarrow^n) - Ion^n_{i,j} + A^n_{i,j} \right) \frac{\Delta t}{C} + V^n_{i,j} = V^{n+1}_{i,j}. \tag{3.12}$$

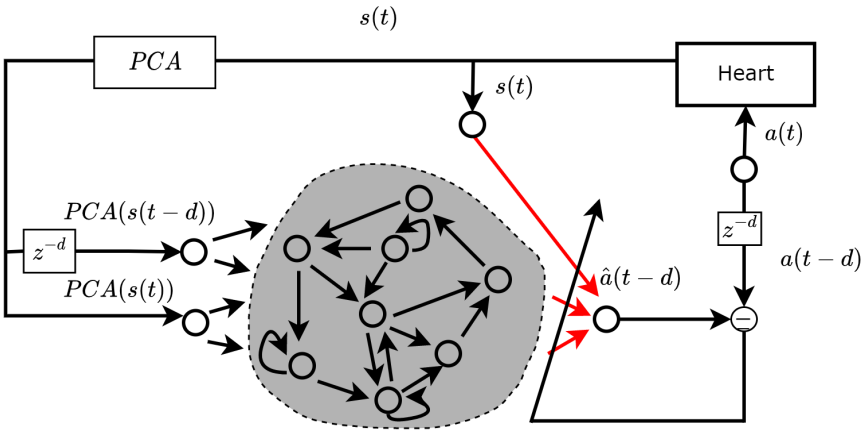


Figure 3.8: The architecture of the PCA controller, for an explanation, see the text

Expressing Equation 2.6 for the next time step (a.k.a. for V^{n+1} and V^{n+2}), then substituting it back to Equation 3.12, repeating the process until $n+d$ timestep is reached, the optimal action A^n can be expressed as

$$A^n_{i,j} = \frac{1}{N}\left[C \cdot \frac{V^{n+N}_{i,j} - V^n_{i,j}}{\Delta t} + \sum_{k=0}^{N-1} Ion^{n+k}_{i,j} - \frac{1}{Sv\Delta x^2}\sum_{k=0}^{N-1}(\leftrightarrow^{n+k} + \updownarrow^{n+k})\right]. \tag{3.13}$$

The full derivation can be found in Appendix A.2. Equation 3.13 is only solvable iteratively (in combination with a forward Euler method). The ESN controller does not have access to the entire matrix A (I_{stim}), thus, it should only be an approximation of Equation 3.13. Notice that Equation 3.13 uses the difference between the current state and a reference state; $V^{n+N}_{i,j} - V^n_{i,j}$ which is an error term frequently used in control theory. Therefore, a boolean `subState` parameter was introduced. If set to true the input to the reservoir was extended by the error term $V^{n+N}_{i,j} - V^n_{i,j}$. The error term was also directly used for the calculation of the output $\hat{a}$ (as described in Equation 2.15). The error term should in theory help the controller predict the action a by pre-computing an important value for the controller (the error term).

3.3 Van der Pol design

Controlling the chaotic Van der Pol oscillator requires significantly less computing abilities than controlling the full 2560 dimensional heart model. Consequently, a simple reservoir design, as described in sub-section 2.4, was used. Where the reservoir weight matrix is a sparse matrix, inputs are fully connected to the reservoir and the leaky mask is diverse. There is, however, one difference that deviates from the previously described designs.

The current state $s(t)$ and the reference state $s_{ref}(t)$, are not only time-shifted versions of the other. The current state $s(t)$ got extended with the information about the external driving force $F(t) = \beta \cdot cos(\omega \cdot t)$ because that is also needed to fully describe the system.

$$s(t) = \begin{bmatrix} x(t) \\ y(t) \\ \beta \cdot cos(\omega \cdot t) \end{bmatrix}, \text{ and } s_{ref}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}. \tag{3.14}$$

4 Results

The normalized root mean squared error (NRMSE) was used to evaluate the controllers:

$$NRMSE(x, \hat{x}) = \frac{1}{F}\sum_{f=1}^{F}\sqrt{\frac{1}{var(x_f)\cdot T}\sum_{n=1}^{T}(x_f(n) - \hat{x}_f(n))^2}. \tag{4.1}$$

The NRMSE normalizes the root mean squared error by dividing it by the variance of the teacher variable. Two errors were used: 1) The validation error which is the difference between the predicted action and the actual action on unseen data; $NRMSE(a, \hat{a})$. 2) The testing error which the difference between the reference

state trajectory s_{ref} (recorded from the healthy heart model) and the actual state trajectory s taken by the heart model whilst being controlled by the ESN; $NRMSE(s_{ref}, s)$.

Table 4.1 shows the result for each of the designs. The Van der Pol controller performed the best with validation NRMSE being 0.019 and testing NRMSE 0.090. The local design was the best for the full 2560 dimensional heart model with the validation error being 0.1378 - as shown in Figure 4.1. None of the controllers was able to actually control the full heart model, as seen by the poor testing errors. Figure 4.2 shows the inability of the controller to follow the state trajectory. On the other hand, the Van der Pol design was successfully able to to predict the teacher variable a (Figure 4.3) and control the chaotic oscillator (Figure 4.4).

	Validation error			**Testing error**
	No modifiers	**Squared reservoir**	**State difference**	**No modifiers**
Full-state	0.5781	0.5343	0.5469	1.2576
Local	0.1378	X	0.1323	1.4283
PCA	0.2034	0.1910	0.2047	1.3632
Van der Pol	0.0190	X	X	0.090

Table 4.1: Validation & Testing errors (NRMSE) for the different architectures designs, where "X" indicates missing data. Dataset with sampling frequency $f_s = 20hz$ was used to train the controller

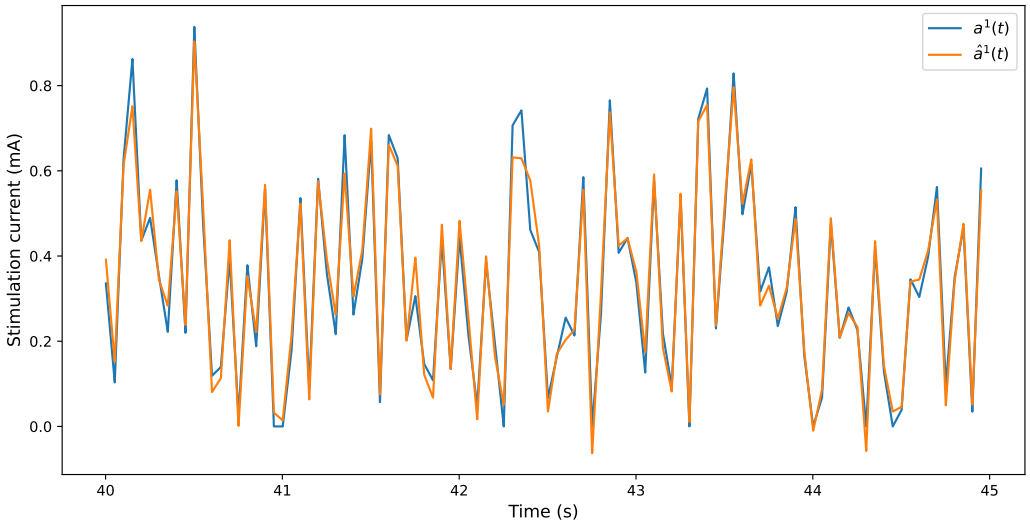


Figure 4.1: Validation error of 0.1378. The plot shows the difference between the predicted action $\hat{a}$ by the local design controller and the teacher variable a

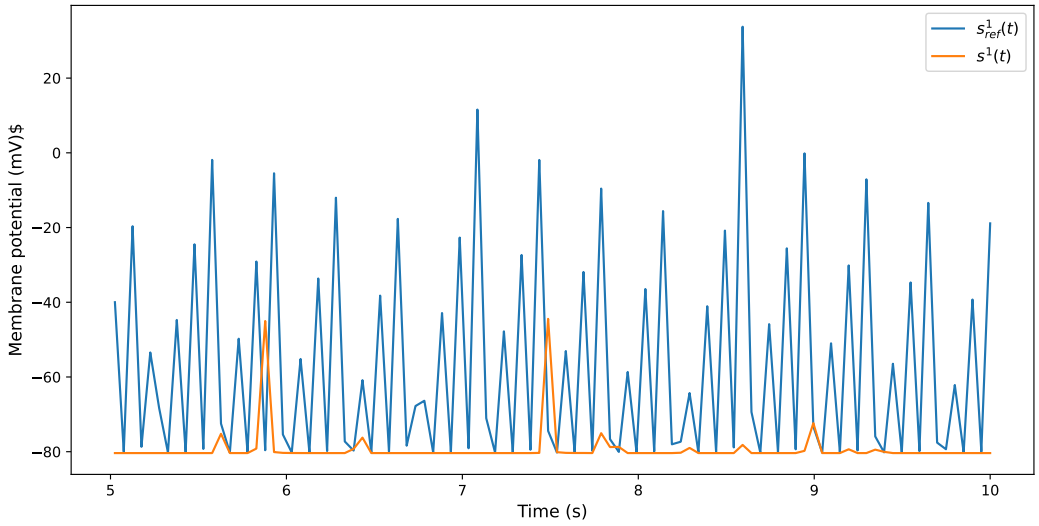


Figure 4.2: Testing error of 1.3632. The plot shows the difference between the reference state trajectory s_{ref} and the actual trajectory s taken by the heart whilst being controlled with the PCA design controller

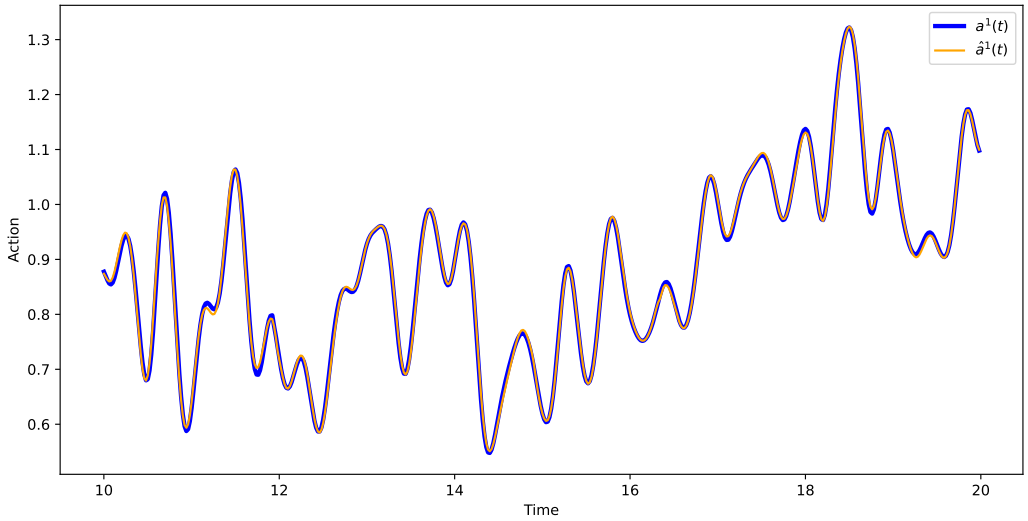


Figure 4.3: Validation error of 0.0190 in the Van der Pol task. The plot shows the similarity of the predicted action $\hat{a}$ by the Van der Pol design controller and the teacher variable a.

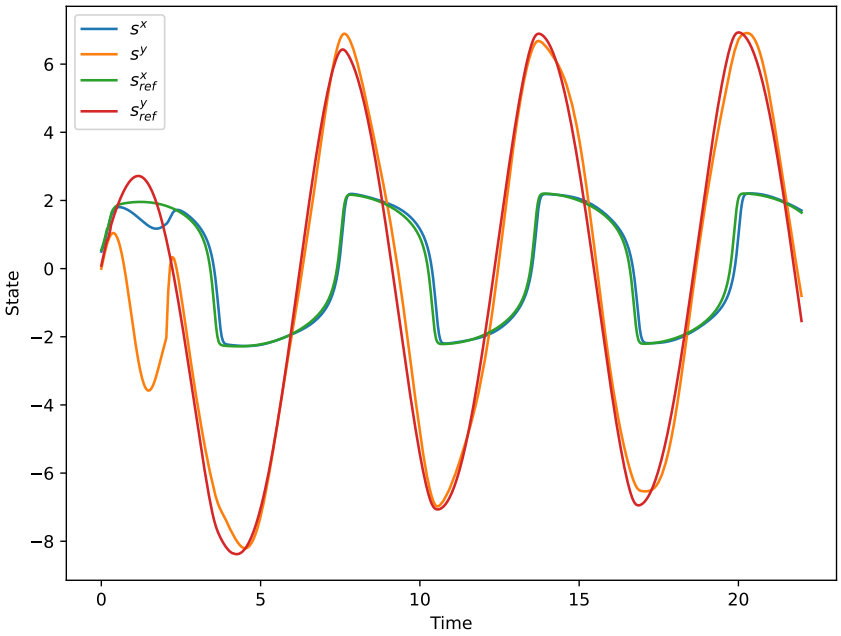


Figure 4.4: Testing error of 0.09 in the Van der Pol task. The plot shows the difference between the reference state trajectory s_{ref} and the actual trajectory s taken by the Van der Poll chaotic oscillator whilst being controlled with the ESN controller

4.1 Hyper-parameters

The hyper-parameters were optimized manually together with a grid search algorithm. The best performing models were trained on the dataset with sampling frequency $f_s = 20Hz$, thus, only those hyper-parameters will be mentioned in this thesis. There were no remarkable differences between the best hyper-parameters found for each of the datasets ($f_s = \{20Hz, 50Hz, 100Hz\}$), except for the delay parameter d (which is to be expected). The precises values for the hyper-parameters for the full-state, local, PCA, and Van der Pol designs can be found in Appendix section A.3.

5 Discussion

As said before, none of the reaction-diffusion controller designs was able to control the heart arrhythmia and restore a regular heartbeat. This section will describe the potential problems and causes behind the unsuccessful task. The Van der Pol task was successful and there will not be any analysis of that controller.

5.1 Potential problems and improvements

5.1.1 Dataset variety

One of the major issues that were found at the end of this bachelor thesis was the fact that 90% of the dataset was redundant. I generated my datasets by simulating 10 hearts simultaneously for 30 simulation minutes (yielding 5 hours of simulated heartbeat). Recall that the actions were generated by firstly producing white noise. I used the `numpy.random.random` function to sample the uniform distribution. It generates pseudo-random numbers based on a given seed. If the seed is not given the `numpy` package generates a random seed based on your hardware fingerprints and the current time to ensure that the random numbers are different each time you run the program.

For some unexplained reasons if I ran the simulation on multiple cores using Python's `multiprocessing` library, each of the cores shared the same seed used for generating the random actions. Consequently, the same actions were generated which lead to the same heart states, and thus, 90% of the dataset is a copy of the remaining 10%. After I noticed the issue, I generated a new validation dataset so that the validation errors provided here are based on actual unseen data (obviously the hyper-parameters were re-optimized too

for the unseen data). The new validation errors were suspiciously close to the old validation errors (which were training errors), leading me to believe that either the network is very resilient to overfitting or that the 80% of the dataset did not have any significant impact on the training quality.

There are two reasons why I did not notice such a major issue. Firstly, the issue happens exclusively on the high-performance computing cluster that I used to generate the dataset. I tested the dataset generation algorithm numerous times, however, only on the data generated by my local computing machine where the error is not present and the seed gets initialized differently for each of the cores. Secondly, even though each core outputs a byte-perfect copy of the other cores (meaning there are not even small numerical differences), the PCA transformation matrices computed individually on each of the core's datasets differ slightly (as seen by Figure 3.7). This gave me the impression that the datasets had to be different. After more testing, I realized that the library `sklearn.decomposition` I used to compute the PCA transformation matrix was not deterministic, it produces a different matrix each time it ran. Investigating why the `sklearn.decomposition.PCA` algorithm is not deterministic is beyond my pay grade. I implemented the algorithm myself using purely `numpy` and it yielded the same transformation matrix each time it re-ran.

Lastly, it was too late to re-generate the full dataset correctly due to the time constraint of this thesis. The model I have chosen to control is too bulky to work with and the dataset generation alone takes 2 full days on the Peregrine cluster.

5.1.2 Injector points placement

The injector points location (as seen in Figure 2.9) was done more or less arbitrarily and was not experimented with at all. I speculate that using more injector points and positioning them in tailored locations would give the controller a higher chance to treat the arrhythmia.

Firstly, 2 out of 4 injector points are placed directly in the high resistivity area. Looking at Sub-figure 5.1 one can see the resistivity mask ρ indicated by red cells (cells with brighter red colour are more resistant) and the placement of the injector points indicated by cyan colour. Going from the top of the Sub-figure 5.1 the second and the fourth injector point is surrounded by high resistivity cells. The issue arises when I inspected how those injection points' areas react to the actions $a(t)$ used for the dataset generation. Figure 5.1 shows the failure of initiation of the action potential cascade despite the individual action potential being triggered successfully. Sub-figures 5.1b-5.1d display the heart model between $t = 690ms$ & $t = 740ms$. Sub-figure 5.1e presents the action that was applied to the heart during that time period. As indicated in Sub-figure 5.1e in the shaded region, this analysis will focus on the action that corresponds to the second most top injector point. Sub-figure 5.1b shows that the action spiked an individual action potential. The action potential then propagated to 2 neighbouring cells (Sub-figure 5.1c), after which it swiftly died out (Sub-figure 5.1d).

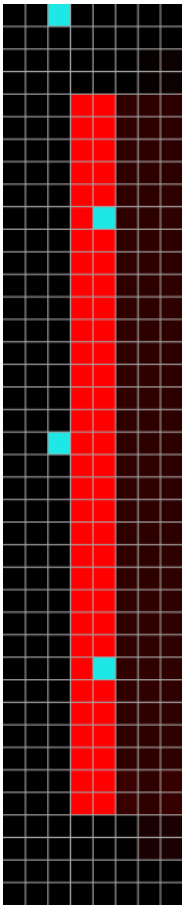
On the other hand, the green-coloured action that came right after the aforementioned orange-coloured action (Sub-figure 5.1e) triggered an action potential cascade (Sub-figures 5.1c & 5.1d) despite the green action being weaker than the orange action. It is not impossible to initiate an action potential cascade in the high-resistivity areas but the action applied to those areas needs to have large amplitude (around $\approx 40mA$) and it needs to be applied for a longer period of time ($t_{period} \approx 150ms$). These types of action were under-represented in the training and validation dataset. Consequently, granting the controller only 2 effective injector points - which is simply too little to control the 2560 dimensional system.

Later research will be more successful if they place more injectors, especially on the scarred tissue (as indicated by the slightly red-ish region in Sub-figure 5.1a on the right since that is the region of the heart model which is responsible for the arrhythmia and needs fixing.

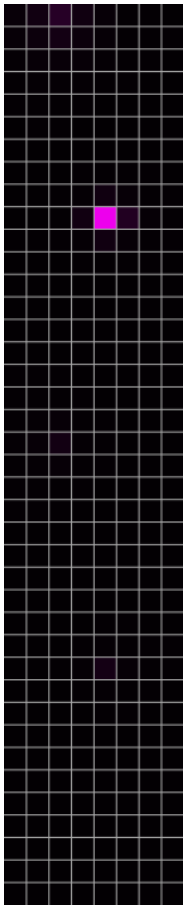
5.1.3 Under-utilization of parallel processing

The local design, with the best scoring validation error, can be trained in parallel. Despite that, due to the time constraints of this thesis, I trained the reservoirs in series - greatly increasing the training time of the controller. To make the situation even worse I used only one internal weight matrix W and reservoir x which was set up in a way so it perfectly mimics the design described in Sub-section 3.2.2. However, it did not have the performance & the memory boost as the true parallel design. Therefore, I needed to use significantly less reservoir neurons than Pathak et al. (2018); Vlachas et al. (2020).

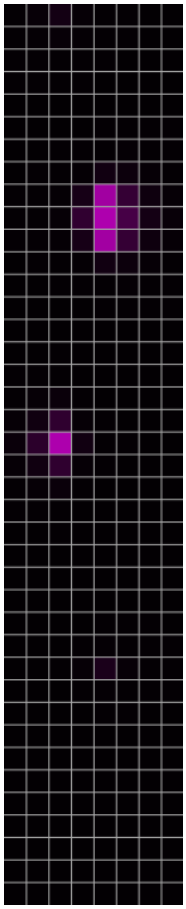
I am fully aware that the argument "add more neurons" can be seen as a lazy analysis but the local design was indeed under-utilized. The validation error decreased linearly as more neurons were added to the local design whilst, as an illustration, adding more neurons to the PCA design had an exponentially decreasing effect on the validation error. Figure 5.2 shows the effect on the validation error whilst adding more neurons to the two aforementioned designs.



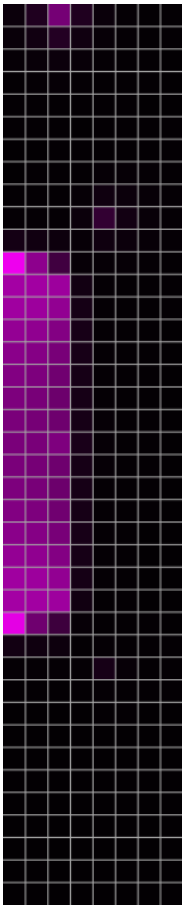
(a) Injection points & ρ



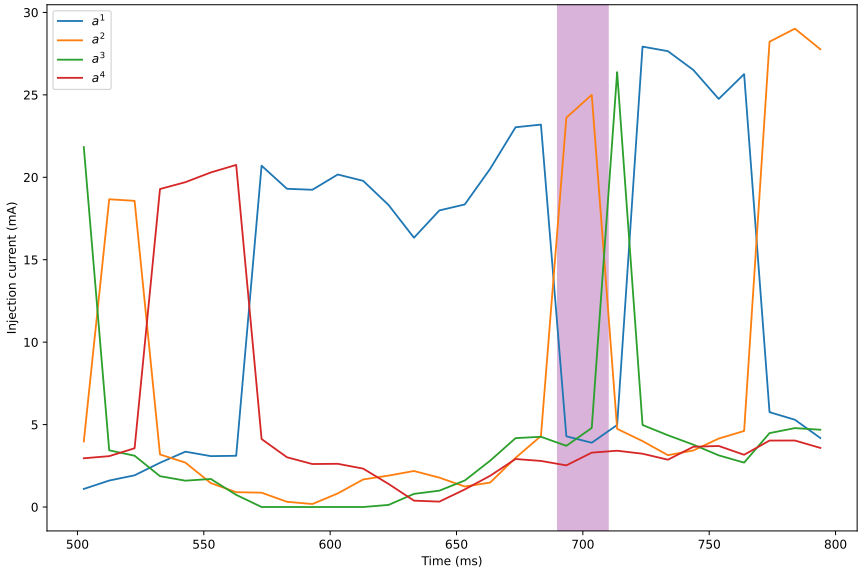
(b) $t = 690ms$



(c) $t = 715ms$



(d) $t = 740ms$



(e) Actions applied to the heart

Figure 5.1: Showcase of a failed attempt to initiate an action potential cascade due to the strong resistivity mask surrounding the injection point. For full explanation see the text

I could not continue adding more neurons to the local design due to the quadratic scaling of the internal weight matrix W and the memory/performance limitation of my local machine. Future researchers/bachelor students should implement the parallel training of the reservoirs which would allow them to train much bigger networks even on their laptops.

5.1.4 Spectral radius of 0

Notice that the best found spectral radius r_s for the PCA design is 0, meaning that all internal weights are set to 0 and the network becomes a simple feed-forward network. A spectral radius of 0 removes any memory capabilities of the controller. If one would choose a very low sampling frequency the sequence of states $(s(0), s(1), s(2), ..., s(N)$ would appear more as a stochastic variable coming from an unknown distribution rather continuous states dependent on each other. Therefore, any information about the previous state would be almost redundant if the sampling frequency would be too low for the prediction of the next action a. Simply put the reservoir memory capabilities would introduce noise and removing them (by setting the spectral radius 0) would improve the prediction.

However, I also performed numerous hyper-optimization runs with the other datasets with higher sampling frequencies ($f_s = \{50Hz, 100Hz\}$) and none of them was able to outperform the $20Hz$ dataset. It is also important to mention that a non-zero spectral radius was found as the best parameter choice for the other datasets - again, restoring the notion that memory is important for controlling the heart.

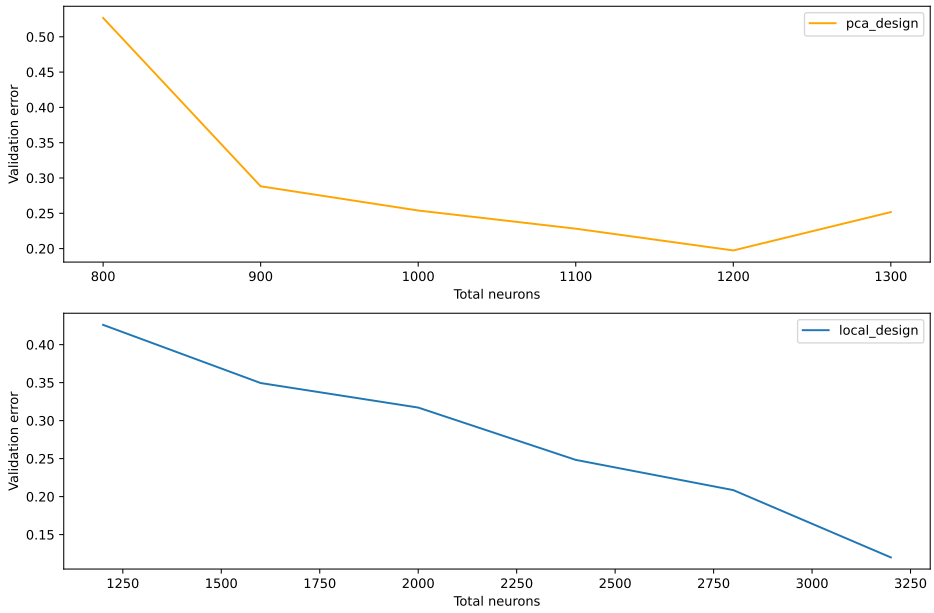


Figure 5.2: Comparison of how adding more reservoir neurons affects the validation error. Both figures show the total number of neurons in the controller reservoir(s) vs the validation error (NRMSE)

5.2 Conclusion

For any future bachelor/master students reading my thesis, chaos control using an echo state network is an incredibly fun topic for a BSc thesis. It provides endless possibilities of what system to control and strikes a perfect balance between theory & application. Numerous times I found myself spending 12 hours a day, coding and exploring what more can be done to improve the model/controller (only to regret that one day has just 24 hours). The only advice I want to give to you is to choose a simpler more lightweight model to control. Spending half an hour just to test one iteration of the controller is too much for one's sanity.

I would like to thank my supervisor professor Herbert Jaeger for mentoring me throughout this incredible learning experience, not imposing his master mind too much, letting me choose my topic, making my errors, and learning from them by myself. I would also like to thank my co-supervisor professor Jelmer Borst for being patient with my late thesis submission and for reading and grading the thesis. Thank you!

References

Mohammad Alasti, Colin Machado, Karthikeyan Rangasamy, Logan Bittinger, Stewart Healy, Emily Kotschet, David Adam, and Jeff Alison. Pacemaker-mediated arrhythmias. *Journal of Arrhythmia*, 34(5):485–492, 2018.

Felipe Alonso-Atienza, Eduardo Morgado, Lorena Fernandez-Martinez, Arcadi García-Alberola, and José Luis Rojo-Alvarez. Detection of life-threatening arrhythmias using feature selection and support vector machines. *IEEE Transactions on Biomedical Engineering*, 61(3):832–840, 2013.

Shweta Bajpai, MS Alam, and MA Ali. Intelligent heart rate controller using fractional order PID controller tuned by genetic algorithm for pacemaker. *International Journal of Engineering Research & Technology*, 6(5):715–720, 2017.

Olivier Blanc, Nathalie Virag, J-M Vesin, and Lukas Kappenberger. A computer model of human atria with reasonable computation load and realistic anatomical properties. *IEEE Transactions on Biomedical Engineering*, 48(11):1229–1237, 2001.

Michael Buehner and Peter Young. A tighter bound for the echo state property. *IEEE Transactions on Neural Networks*, 17(3):820–824, 2006.

Matthew Cooper, Peter Heidlauf, and Timothy Sands. Controlling chaos—forced van der pol equation. *Mathematics*, 5(4):70, 2017.

Bianca Borem Ferreira, Aline Souza de Paula, and Marcelo Amorim Savi. Chaos control applied to heart rhythm dynamics. *Chaos, Solitons & Fractals*, 44(8):587–599, 2011.

Gregory R Ferrier and Peter E Dresel. Relationship of the functional refractory period to conduction in the atrioventricular node. *Circulation Research*, 35(2):204–214, 1974.

Richard N Fogoros. *Antiarrhythmic drugs: A practical guide*. John Wiley & Sons, 2008.

Pietro Francia, Cristina Balla, Arianna Uccellini, and Riccardo Cappato. Arrhythmia detection in single-and dual-chamber implantable cardioverter defibrillators: The more leads, the better? *Journal of Cardiovascular Electrophysiology*, 20(9):1077–1082, 2009.

Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, 6(6):801–806, 1993.

John B Garner and John M Miller. Wide complex tachycardia–ventricular tachycardia or not ventricular tachycardia, that remains the question. *Arrhythmia & Electrophysiology Review*, 2(1):23, 2013.

Andreas Goette, Jonathan M Kalman, Luis Aguinaga, Joseph Akar, Jose Angel Cabrera, Shih Ann Chen, Sumeet S Chugh, Domenico Corradi, Andre D'Avila, Dobromir Dobrev, et al. EHRA/HRS/APHRS/SOLAECE expert consensus on atrial cardiomyopathies: definition, characterisation, and clinical implication. *Journal of arrhythmia*, 32(4):247–278, 2016.

Amandeep Goyal, Benjamin Senst, Poonam Bhyan, and Roman Zeltser. *Reentry Arrhythmia*. StatPearls Publishing, Treasure Island (FL), 2021. URL http://europepmc.org/books/NBK537089.

John Edward Hall, Arthur C. Guyton, and Michael E. Hall. Cardiac muscle; the heart as a pump and function of the heart valves. In *Guyton and Hall Textbook of Medical Physiology*, chapter 1, page 113–126. Elsevier, 2021.

Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

Janusz A Holyst, Tilo Hagel, Günter Haag, and Wolfgang Weidlich. How to control a chaotic economy? *Journal of Evolutionary Economics*, 6(1):31–42, 1996.

Ali Isin and Selen Ozdalili. Cardiac arrhythmia detection using deep learning. *Procedia Computer Science*, 120:268–275, 2017.

Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 2004.

Richard E. Klabunde. Sinoatrial node action potentials, Jan 2021. URL https://www.cvphysiology.com/Arrhythmias/A004.

Qiao Li, Cadathur Rajagopalan, and Gari D Clifford. Ventricular fibrillation and tachycardia classification using a machine learning approach. *IEEE Transactions on Biomedical Engineering*, 61(6):1607–1613, 2013.

Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

Ching-hsing Luo and Yoram Rudy. A model of the ventricular cardiac action potential. depolarization, repolarization, and their interaction. *Circulation Research*, 68(6):1501–1526, 1991.

Elbert EN Macau and Celso Grebogi. Control of chaos and its relevancy to spacecraft steering. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 364(1846):2463–2481, 2006.

T Marios. Theoretical and numerical study of the Van der Pol equation. *PhD thesis, Aristotle University of Thessaloniki School of Sciences ...*, 2006. URL https://arxiv.org/ftp/arxiv/papers/0803/0803.1658.pdf.

Shaher Momani, Iqbal M Batiha, and Reyad El-Khazali. Design of pi λ d δ-heart rate controllers for cardiac pacemaker. In *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 1–5. IEEE, 2019.

Fred Morady, Steven D Nelson, William H Kou, Richard Pratley, Stephen Schmaltz, Michael De Buitleir, and Jeffrey B Halter. Electrophysiologic effects of epinephrine in humans. *Journal of the American College of Cardiology*, 11(6):1235–1244, 1988.

Venkat D Nagarajan, Su-Lin Lee, Jan-Lukas Robertus, Christoph A Nienaber, Natalia A Trayanova, and Sabine Ernst. Artificial intelligence in the diagnosis and management of arrhythmias. *European Heart Journal*, 42(38):3904–3916, 2021.

NHLBI. Arrhythmia, 2018. URL https://www.nhlbi.nih.gov/health-topics/arrhythmia.

Edward Ott, Celso Grebogi, and James A Yorke. Controlling chaos. *Physical Review Letters*, 64(11):1196, 1990.

Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical Review Letters*, 120(2):024102, 2018.

Marco V Perez, Frederick E Dewey, Swee Y Tan, Jonathan Myers, and Victor F Froelicher. Added value of a resting ecg neural network that predicts cardiovascular mortality. *Annals of Noninvasive Electrocardiology*, 14(1):26–34, 2009.

Jeremy N Ruskin. The cardiac arrhythmia suppression trial (cast). *New England Journal of Medicine*, 321 (6):386–388, 1989.

Matthias Salmen and Paul G Plöger. Echo state networks used for motor control. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 1953–1958. IEEE, 2005.

Eckehard Schöll and Heinz Georg Schuster. Control of cardiac electrical nonlinear dynamics. In *Handbook of Chaos Control*, page 683–701. Wiley Online Library, 2008.

Viktor Veselý. Chaos, Sep 2021. URL https://github.com/viktorvesely/CHAOS.

Pantelis R Vlachas, Jaideep Pathak, Brian R Hunt, Themistoklis P Sapsis, Michelle Girvan, Edward Ott, and Petros Koumoutsakos. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks*, 126:191–217, 2020.

Wikipedia. Ventricular action potential, Oct 2021. URL https://en.wikipedia.org/wiki/Ventricular_action_potential.

A Appendix

A.1 Cell equations

The equations are taken and modified from Luo and Rudy (1991). For the explanation of the modifications and equations see Section 2.1.

A.1.1 Units

Name	Notation	Units	Notes
Membrane voltage	V_m or V	mV	Spatiotemporally chaotic
Cell current	I	$\frac{\mu A}{cm^2}$	Flow of ions
Gating parameters	α and β	$\frac{1}{ms}$	Determine the speed of the gates
Concentration of ions	$[Ion]_{\{i,o\}}$	Unitless	Intracellular (i) Extracellular (o)
Voltage source	E	mV	Resting potential
Time	t	ms	No remarks
Membrane capacitance	C	$\frac{\mu F}{cm^2}$	No remarks

Table A.1: Description of units

A.1.2 State variables

Membrane potential

$$\frac{dV}{dt} = -\frac{1}{C}(I_{ions} - I_{stim}), \tag{A.1}$$

with

$$V(t=0) = -81.1014, \tag{A.2}$$

where I_{ions} is the sum of all ions defined later in this section and I_{stim} is the stimulation current.

Gates

Let y be any gate in the cell, then

$$\begin{aligned} \frac{dy}{dt} &= \frac{y_\infty - y}{\tau_y}, \\ y_\infty(V) &= \frac{\alpha_y(V)}{\alpha_y(V) + \beta_y(V)}, \\ \tau_y(V) &= \frac{1}{\alpha_y(V) + \beta_y(V)}, \end{aligned} \tag{A.3}$$

with

$$y(t=0) = y_\infty(V(t=0)). \tag{A.4}$$

Cell calcium uptake

$$\frac{d[Ca]_i}{dt} = -10^{-4} \cdot I_{si} + 0.07 \cdot (10^{-4} - [Ca]_i), \tag{A.5}$$

with

$$[Ca]_i(t=0) = 2 \cdot 10^{-4}. \tag{A.6}$$

A.1.3 Inward currents

Fast time-dependent sodium current

For all V

$$
\begin{aligned}
&I_{Na}(t) = \hat{G}_{Na} \cdot m^3(t) \cdot h(t) \cdot j(t) \cdot (V(t) - E_{Na}),\\
&\hat{G}_{Na} = 23 \cdot 1.3,\\
&E_N a = 54.4,\\
&\alpha_m(V) = \frac{0.32 \cdot (V + 47.13)}{1 - e^{-0.1 \cdot (V+47.13)}},\\
&\beta_m(V) = 0.08 \cdot e^{-\frac{V}{11}}.
\end{aligned}
$$

For V larger or equal to -40 mV

$$
\begin{aligned}
&\alpha_h(V) = \alpha_j(V) = 0,\\
&\beta_h(V) = \frac{1}{0.13\left(1 + e^{\frac{V+10.66}{-11.1}}\right)},\\
&\beta_j(V) = \frac{0.3 \cdot e^{-2.535 \cdot 10^{-7} V}}{1 + e^{-0.1 \cdot (V+32)}}.
\end{aligned}
\tag{A.7}
$$

For V less than -40 mV

$$
\begin{aligned}
&\alpha_h(V) = 0.135 \cdot e^{\frac{80+V}{-6.8}},\\
&\alpha_j(V) = \left(-1.2714 \cdot 10^5 \cdot e^{0.2444 \cdot V} - 3.474 \cdot 10^{-5} \cdot e^{-0.04391 \cdot V}\right) \cdot \frac{V + 37.78}{1 + e^{0.311 \cdot (V+79.23)}},\\
&\beta_h(V) = 3.56 \cdot e^{0.079 \cdot V} + 3.1 \cdot 10^5 \cdot e^{0.35 \cdot V},\\
&\beta_j(V) = \frac{0.1212 \cdot e^{-0.01052 V}}{1 + e^{-0.1378 \cdot (V+40.14)}}.
\end{aligned}
$$

Slow time-dependent silicon current

$$
\begin{aligned}
&I_{si}(t) = 0.2 \cdot \bar{G}_{si} \cdot d(t) \cdot f(t) \cdot (V(t) - E_{si}),\\
&\bar{G}_{si} = 0.09,\\
&E_{si} = 7.7 - 13.0287 ln([Ca]_i),\\
&\alpha_d(V) = \frac{0.095 \cdot e^{-0.01 \cdot (V-5)}}{1 + e^{-0.072 \cdot (V-5)}},\\
&\beta_d(V) = \frac{0.07 \cdot e^{-0.017 \cdot (V+44)}}{1 + e^{0.05 \cdot (V+44)}},\\
&\alpha_f(V) = \frac{0.012 \cdot e^{-0.008 \cdot (V+28)}}{1 + e^{0.15 \cdot (V+28)}},\\
&\beta_f(V) = \frac{0.0065 \cdot e^{-0.02 \cdot (V+30)}}{1 + e^{-0.2 \cdot (V+30)}}.
\end{aligned}
\tag{A.8}
$$

A.1.4 Outward currents

Time-dependent potassium current

$$
\begin{aligned}
&I_K(t) = \bar{G}_K \cdot X(t) \cdot X_i \cdot (V(t) - E_K),\\
&\bar{G}_K = 0.282 \cdot \sqrt{\frac{[K]_o}{5.4}},\\
&[K]_o = 5.4,\\
&E_K = -77,\\
&X_i(V) = \begin{cases} 2.837 \cdot \dfrac{e^{0.04\cdot(V+77)} - 1}{(V+77)\cdot e^{0.04(V+35)}} & \text{for } V > -100mV\\ 1 & \text{otherwise} \end{cases},\\
&\alpha_x(V) = \frac{0.0005 \cdot 5 \cdot e^{0.083\cdot(V+50)}}{1 + e^{0.057\cdot(V+50)}},\\
&\beta_x(V) = \frac{0.0013 \cdot 5 \cdot e^{-0.06\cdot(V+20)}}{1 + e^{-0.04\cdot(V+20)}}.
\end{aligned}
\tag{A.9}
$$

Time-independent potassium current

$$
\begin{aligned}
&I_{K1}(V) = \bar{G}_{K1} \cdot K1_\infty \cdot (V(t) - E_{K1}),\\
&\bar{G}_{K1} = 0.6047 \cdot \sqrt{\frac{[K]_o}{5.4}},\\
&[K]_o = 5.4,\\
&E_{K1} = -84,\\
&\alpha_{K1}(V) = \frac{1.02}{1 + e^{0.2385\cdot(V - E_{K1} - 59.215)}},\\
&\beta_{K1} = \frac{0.49124 \cdot e^{0.08032\cdot(V - E_{K1} + 5.476)} + e^{0.06175\cdot(V - E_{K1} - 594.31)}}{1 + e^{-0.5143\cdot(V - E_{K1} + 4.753)}}.
\end{aligned}
\tag{A.10}
$$

Time-independent plateau potassium current

$$
\begin{aligned}
&I_{Kp}(V) = \bar{G}_{Kp} \cdot Kp \cdot (V(t) - E_{Kp}),\\
&\bar{G}_{Kp} = 0.0183,\\
&E_{Kp} = E_{K1} = -84,\\
&Kp(V) = \frac{1}{1 + e^{\frac{7.488 - V}{5.98})}}.
\end{aligned}
\tag{A.11}
$$

Time-independent background current

$$
\begin{aligned}
&I_{Kp}(V) = \bar{G}_b \cdot (V(t) - E_b),\\
&\bar{G}_b = 0.03921,\\
&E_b = -59.87.
\end{aligned}
\tag{A.12}
$$

A.2 Derivation of the control equation

This sections derives the optimal control equation from the heart model equation 2.6. Firstly, discretizing Equation 2.6 yields

$$
\frac{1}{Sv\Delta x^2}\left(\frac{V^n_{i-1,j} - V^n_{i,j}}{\rho^{i-1,i}} + \frac{V^n_{i+1,j} - V^n_{i,j}}{\rho^{i+1,i}}\right) + \frac{1}{Sv\Delta y^2}\left(\frac{V^n_{i,j-1} - V^n_{i,j}}{\rho^{j-1,j}} + \frac{V^n_{i,j+1} - V^n_{i,j}}{\rho^{j+1,j}}\right) = C\cdot\frac{V^{n+1}_{i,j} - V^n_{i,j}}{\Delta t} + Ion^n_{i,j} - A^n_{i,j}. \tag{A.13}
$$

Assuming $\Delta x = \Delta y$ (which is true in the model) and

$$
\leftrightarrow^n = \frac{V^n_{i-1,j} - V^n_{i,j}}{\rho^{i-1,i}} + \frac{V^n_{i+1,j} - V^n_{i,j}}{\rho^{i+1,i}}, \tag{A.14}
$$

$$
\updownarrow^n = \frac{V^n_{i,j-1} - V^n_{i,j}}{\rho^{j-1,j}} + \frac{V^n_{i,j+1} - V^n_{i,j}}{\rho^{j+1,j}}. \tag{A.15}
$$

The equation can be then rewritten into a more readable form

$$\frac{1}{Sv\Delta x^2}(\leftrightarrow^n + \updownarrow^n) = C \cdot \frac{V_{i,j}^{n+1} - V_{i,j}^n}{\Delta t} + Ion_{i,j}^n - A_{i,j}^n. \tag{A.16}$$

The echo state network controller works on a different time-scale than the heart model (different update frequency), therefore, let N be the number of update steps of the heart model needed for the controller to update. N can be calculated as the ratio of the controller period T_c and the model period T_m; $N = \frac{T_c}{T_m}$. Given V^n, what is the optimal action I_{stim} such that the heart model will transition to $Vn + N$.

Firstly, re-range for the future state V^{n+1}

$$\left(\frac{1}{S_v\Delta x^2}(\leftrightarrow^n + \updownarrow^n) - Ion_{i,j}^n + A_{i,j}^n\right)\frac{\Delta t}{C} + V_{i,j}^n = V_{i,j}^{n+1}. \tag{A.17}$$

Then express the Equation A.16 for $n + 1$

$$\frac{1}{Sv\Delta x^2}(\leftrightarrow^{n+1} + \updownarrow^{n+1}) = C \cdot \frac{V_{i,j}^{n+2} - V_{i,j}^{n+1}}{\Delta t} + Ion_{i,j}^{n+1} - A_{i,j}^{n+1}. \tag{A.18}$$

Substitute $V_{i,j}^{n+1}$ from Equation A.17 and re-arrange

$$\frac{1}{Sv\Delta x^2}[(\leftrightarrow^n + \updownarrow^n) + (\leftrightarrow^{n+1} + \updownarrow^{n+1})] = C \cdot \frac{V_{i,j}^{n+2} - V_{i,j}^n}{\Delta t} + Ion_{i,j}^{n+1} + Ion_{i,j}^n - A_{i,j}^{n+1} - A_{i,j}^{n+1}. \tag{A.19}$$

It is important to realize that by the definition of my controller $A^n = A^{n+k}$ until the controller makes a new decision. Thus, the equation can simplified

$$\frac{1}{Sv\Delta x^2}[(\leftrightarrow^n + \updownarrow^n) + (\leftrightarrow^{n+1} + \updownarrow^{n+1})] = C \cdot \frac{V_{i,j}^{n+2} - V_{i,j}^n}{\Delta t} + Ion_{i,j}^{n+1} + Ion_{i,j}^n - 2 \cdot A_{i,j}^n. \tag{A.20}$$

The formula can be generalized for any number of steps (by substitution)

$$\frac{1}{Sv\Delta x^2}\sum_{k=0}^{N-1}(\leftrightarrow^{n+k} + \updownarrow^{n+k}) = C \cdot \frac{V_{i,j}^{n+N} - V_{i,j}^n}{\Delta t} + \sum_{k=0}^{N-1} Ion_{i,j}^{n+k} - N \cdot A_{i,j}^n. \tag{A.21}$$

Rearranging for A^n

$$A_{i,j}^n = \frac{1}{N}\left[C \cdot \frac{V_{i,j}^{n+N} - V_{i,j}^n}{\Delta t} + \sum_{k=0}^{N-1} Ion_{i,j}^{n+k} - \frac{1}{Sv\Delta x^2}\sum_{k=0}^{N-1}(\leftrightarrow^{n+k} + \updownarrow^{n+k})\right]. \tag{A.22}$$

Since the controller has less degrees of freedom

$$A_{i,j}^n = \begin{cases} 0 & \text{when there is no injection point} \\ A_{i,j}^n & \text{where there is an injection point} \end{cases}.$$

Due to the last limitation, the optimal solution may not exists.

A.3 ESN controller parameters

In this section the best parameters are presented for each of the controller's design using the dataset with sampling frequency equal to $20Hz$ (except the Van der Poll design which had only one dataset). Table A.2 shows the best hyper-parameters for the full-state design, Table A.3 for the local design, Table A.4 for the PCA design, and Table A.5.

Parameter	Value	Remark
β	0.000001	Regularization
d	1	Delay
washout	50	Washout period
r_s	0.5	Spectral radius
(w^{min}, w^{max})	$(-0.8, 0.8)$	General reservoir weights (uniform distribution)
$\|\|x\|\|$	700	General reservoir size
`connectivity`	0.2	Of the general reservoir
(μ^+, sd^+)	$(1, 0.5)$	Membrane potential weights (normal distribution)
(μ, sd)	$(0, 0.8)$	Other state variable weights (normal distribution)
$(\alpha^{min}, \alpha^{max})$	$(0.01, 0.8)$	Leaky mask for the general reservoir
α^V_{heart}	0.87	Leaky mask for the membrane potential
$(\alpha^{min}_{heart}, \alpha^{max}_{heart})$	$(0.7, 0.9)$	Leaky mask for the other state variable
(μ^{in}, sd^{in})	$(0, 0.2)$	Input weights for the general reservoir
$(\mu^{in}_{heart}, sd^{in}_{heart})$	$(0.2, 0.05)$	Input weights for the heart reservoir
`adjecencyTolerance`	15	See methods section

Table A.2: The best parameters found for the full-state design

Parameter	Value	Remark
β	0.000001	Regularization
d	1	Delay
washout	50	Washout period
r_s	0.3	Spectral radius
(w^{min}, w^{max})	$(-0.8, 0.8)$	Reservoir weights (uniform distribution)
$\|\|x\|\|$	800	Size of the one reservoir out of four
`connectivity`	0.85	Of each reservoir
(μ^{in}, sd^{in})	(0, 0.5)	Input weights (normal distribution)
`adjecencyInput`	18	See methods section
$(\alpha^{min}, \alpha^{max})$	$(0.1, 0.9)$	Leaky mask for the general reservoir

Table A.3: The best parameters found for the local design

Parameter	Value	Remark
β	0.000001	Regularization
d	1	Delay
washout	50	Washout period
r_s	0.0	Spectral radius
(w^{min}, w^{max})	$(-0.8, 0.8)$	Reservoir weights (uniform distribution)
$\|\|x\|\|$	1200	Size of the reservoir
`connectivity`	0.4	Density of the weights
(μ^{in}, sd^{in})	(0, 0.08)	Input weights (normal distribution)
$(\alpha^{min}, \alpha^{max})$	$(0.1, 0.9)$	Leaky mask for the general reservoir

Table A.4: The best parameters found for the PCA design

Parameter	Value	Remark
β	0.000001	Regularization
d	3	Delay
washout	100	Washout period
r_s	0.95	Spectral radius
(w^{min}, w^{max})	$(-1, 1)$	Reservoir weights (uniform distribution)
$\|\|x\|\|$	800	Size of the reservoir
`connectivity`	0.45	Density of the weights
$(w_{in}^{min}, w_{in}^{max})$	(-0.09, 0.09)	Input weights (normal distribution)
$(\alpha^{min}, \alpha^{max})$	$(0.4, 0.7)$	Leaky mask for the general reservoir

Table A.5: The best parameters found for the Van der Pol design