Neuroevolution of Central Pattern Generators

Using the NEAT algorithm for the discovery of Continuous-Time Recurrent Neural Networks with CPG-like behaviour

Andrei Faitas

Thesis submitted for the degree of
Master in Robotics and Intelligent Systems
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Autumn 2021

Neuroevolution of Central Pattern Generators

Using the NEAT algorithm for the discovery of Continuous-Time Recurrent Neural Networks with CPG-like behaviour

Andrei Faitas

Abstract

Locomotion for legged robots has been a long standing problem in robotics. The ambition is to see the realisation of control systems that will allow machines to move with the same adaptability to terrain and task as that observed observed in biological life forms.

A promising proposal to achieve this is the design of Central Pattern Generator (CPG) models, inspired by neural networks found in vertebrate and invertebrate animals. These CPG neural networks generate rhythmic patterns that underlie the rhythmic behaviour that guide activities such as walking, running, breathing etc. Experimental attempts to apply CPG-based control systems to robotic locmotion has seen interesting results and a rise in academic interest over the past two decades. The experiments over these decades show an ability for CPGs to express different gait patterns, and flexible adaptation to perturbations in terrain.

To aid in the design of CPG models for robotic control, this thesis argues for an Evolutionary Computing approach; specifically neuroevolution of a Continuous-Time Recurrent Neural Network. As such neural networks have seen success in evolutionary robotics before, and is designed for the very purpose of modelling dynamical systems, this choice seems apt.

For the experiments described herein, the NeuroEvolution of Augmented Topologies (NEAT)[18] evolutionary algorithm is used. The experiments feature eight configurations of the NEAT genome, and two different fitness functions. Each of the genome configurations is tested with each of the two fitness functions, creating 16 different experiment paradigms. The object of the experiments is to show which of the two fitness functions define a better suited fitness space by comparing the variation that occurs when the genome configuration is altered.

The experiments show that the first fitness function, the Root-Mean-Square-Error is unsuited for discovering CTRNNs that are able to modulate the frequency of the output. However it is sufficiently suited for discovering CTRNNs that output rhythmically oscillation without the need for rhythmic input. The experiments for the second fitness function, a negative exponential function, are inconclusive, but the thesis still provides some insight into why it performs the way that it does. With respect to the variables experimented with in the genome configuration, the thesis finds statistically significant evidence that the CTRNN time constant and tanh activation function have a meaningful impact on NEATs ability to discover oscillator CTRNNs, while network size does not.

Acknowledgements

The author would like to express the sincerest gratitude to his supervisors, Kyrre H. Glette and Frank Veenstra, not only for their guidance and insights, but for their enthusiasm which always made it feel like there was a bright side to all the set backs. To his friends, family and fellow students who were always willing to have enlightening discussions and words of encouragement. The author also wishes to acknowledge that no man is an island, and that so many support systems were in place that made this at all possible. Social and academic systems built by predecessors and maintained by contemporaries across the country. The debt of gratitude is deep.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

This chapter will explain the motivation behind this body of work and the goals it hopes to work towards. It will also feature a rough outline of the entire thesis.

1.1.1 Motivation

Multipedal locomotion has long been a problem of great interest in the field of robotics. A robotic platform that can solve such a problem at a level reminiscent of biological lifeforms, may be employed to perform complex tasks in a real-life environment. Machines that can move automatically and maintain effective mobility in the face of unforeseen variations in the environment, introduces the possibility of applying them to tasks we currently require a human agent to perform. Particularly interesting in this regard, are tasks that are dangerous for humans to perform, or tasks in environments hostile to human life. Examples such as search and rescue, high strain manual labour, or extra planetary exploration are typical of the imagined possibilities. Indeed, such tasks involve much more than locomotion, but the ability to walk on legs is a good place to start.

Interestingly, the field of neuroscience has, in the last century, gained enlightening insights into how nature solves this problem. Neural networks that produce coordinated, rhythmic patterns, termed Central Pattern Generators (CPGs) were found to underlie many of the rhythmic behaviours present in vertebrate and invertebrate animals; behaviours such as for example walking. These CPGs show the ability to generate this rhythmic activity without receiving any rhythmic input from sensory systems or control centers in the brain. This allows control centers to engage rhythmic movement with simple signals to the CPG, which then in turn only adapts its rhythmic pattern in response. This way, a control centre is not required to parameterise complex muscle contractions across time.

In his review paper [8], Ijspeert describes the history of applying Central Pattern Generators (CPGs) to the field robotics. Ijspeert observes a rise in interest the past two decades, in trying to create machine analogies to CPGs to allow for stable and flexible gaits in robots. His own work has seen interesting successes [2, 10], with CPGs, showing robots with multiple gaits and the ability to switch between them continuously.

In the above examples, the CPGs were modelled by human designed oscillators. Intuitively, such an approach limits the range of possibilities and the ceiling of success. Compared to its biological counterpart, these oscillators are relatively simple, and arguably must lack the expressive power to reach the levels of adaptability necessary for the ambitious goals imagined. The utilisation of a machine learning approach may further extend the range of possibilities for CPG-based robot control.

Continuing in the vein of biological inspiration, applying Evolutionary Computing (EC) to the task of automatic design of CPGs is an interesting proposal, and showed early success in Ijspeert[9]. Here, researchers applied a style of EC called Genetic Programming to evolve an Artificial Neural Network (ANN) that would model a CPG-based control system. However, the robot it was meant to control was a design based on a lamprey; a type of eel. It could be argued that the task of swimming is simpler than walking, and so the success of the experiments may not be directly applicable to robots designed for legged locomotion. Additionally, early algorithms for evolving ANNs typically were unable to evolve network topology, exclusively evolving connection weights in a fixed size substrate of neurons.

This thesis is motivated by continued effort to develop automatic methods of designing control systems for multipedal locomotion through EC. It seeks to add to the understanding of how EC may be used to discover CPG-based control systems, and to take an incremental step towards creating machines with the locomotive flexibility and expressiveness of biological life.

This thesis aims to explore the automatic creation and tuning of a CPG by way of neuroevolution (NE). NE is a category of Evolutionary Computing (EC) in which the objective of the evolutionary process is to create Artificial Neural Networks (ANN) without manually tuning hyper parameters through trial an error. ANNs have been used to model CPGs in earlier work, for example Tran et. al. [19], often as Recurrent Neural Networks (RNN).

To apply NE to the problem of creating an ANN CPG, the study will make use of the NeuroEvolution of Augmenting Topologies (NEAT) [18] algorithm. With this algorithm, experiments will be performed to evolve Continuous-Time Recurrent Neural Networks (CTRNN)[20]; a kind of RNN whose hidden neuron outputs are calculations of the rate of change of their input. The experiments are designed to investigate what evolutionary circumstances er necessary or helpful when trying to evolve a CTRNN with CPG-like behaviour, particularly with respect to how one should define evolutionary fitness.

1.1.2 Research goals

The primary goal of the thesis is to gain an understanding of the conditions that promote CPG-like behaviour in an ANN. Based on works like [16, 17, 19], a Continuous-Time Recurrent Neural Network (CTRNN) was decided upon for the type of ANN to evolve. Thus the thesis assumes CTRNNs is at least a suitable, if not necessary condition to evolve CPGs.

Further specification of the research goals require some definition of the properties of a CPG:

- Ability to produce rhythmic output without rhythmic input
- Ability to adapt output rhythm to fit changes in envirnoment

The latter point is frequently termed neuromodulation. In biological CPGs, this property is connected a wide range of sensory input, to be able to respond to a multitude of situations. For the scope of this thesis, neuromodulation will be restricted to a change in output frequency, as a response to change in input. The former point on the other hand will be covered in its entirety in the experiments.

In any EC approach, the choice or fitness function, i.e selection pressure, is both difficult and crucial. The domain of the fitness function defines a fitness space in which the evolution takes place. It is very often hard to predict what sort of behaviour a given fitness function will encourage, and so special care is often needed when selecting one or designing one. To try and understand what makes a suitable fitness function for this problem, this thesis performs experiments with two different fitness functions. One is the Root-Mean-Square-Error function commonly used to measure the prediction error of statistical models. The other is an exponential function, naively designed especially for the purpose of these experiments.

In specific terms, the questions this thesis seeks to answer are as follows:

1. Is there a meaningful difference in likelihood to produce high fitness CTRNN solutions from one fitness function to the other?
2. Is there an identifiable feature that fitness functions should possess in order to motivate selection of CPG-like CTRNNs?

Any insight into these two questions may help towards the automated design of robust control systems for robotics pedal locomotion.

1.1.3 Thesis outline

The thesis consists of four chapters; background, methods, experiments and results, and conclusions. Chapter 2 gives an overview of the main background material the thesis builds on, including the basics of EC, principles and examples of NE, the NEAT algorithm, the biological CPG and CTRNNs.

Chapter 3 states important definitions and assumptions used in the experiments, and motivates these. It explains the limitations of the simulation environment and frequency analyses and how it impacts the study. Finally, it details the experiments performed.

Chapter 4 presents the experimental results from the 16 different paradigms. These results are analysed and interpreted both quantitatively and qualitatively to evaluate the importance of the variables changed between experiments.

Chapter 5 discusses whether the experiments did indeed add understanding of the fitness functions used, and what that understanding is. It reviews the experimental results in light of the research questions posed in this introduction, and discusses in what way, and to what extent, these questions can be considered answered. To end, it suggests some concrete next steps in the form future work.

The appendix contains all visualisations of winner CTRNN outputs from all experiments, along side the corresponding graph of the network topology. It also contains the table of critical values used in the Mann-Whitney U test for analysis of experimental results.

Chapter 2

Background

2.1 Evolutionary Computing

In observing natural evolution, one perspective is that the evolutionary process is essentially a search through the very large space of possible genetic combinations to find an expression of genes that is optimized for survival in a given environment. However, this search is not exhaustive, meaning that evaluating the survivability, or fitness, of every possible combination is not necessary. Instead, the individual expressions of a genome are selected, meaning the search is guided by some general rule to determine which individuals (i.e which permutation of genes) is likely to lead to an improvement in the species.

In Darwin's *On the Origin of Species* [7], he introduces the term "natural selection" to describe the guiding rule as he observed it in nature. By this rule, only the selected individuals are subjected to mechanisms such as genetic reproduction and mutation, effectively narrowing the search by focusing on the more apparently viable subjects and deprecating the rest.

With Evolutionary Computing (EC), the goal is to mimic these mechanisms and apply them to structures of data in order to search through a large space of possible permutations of these structures in a similar, non-exhaustive manner. Consider any sort of formal data structure, for example a vector of n elements. The formalization gives a generalized way of looking at any given instance of this structure, comparable to a biological genotype, and any realised instance is considered an individual expressed by the genotype; a phenotype. With the application of defined genetic operators, the selected individuals are permuted to create new individuals that may represent solutions closer to the optimum we are searching for. Finally and crucially, some sort of *fitness function*, a selection rule, must be designed in order to evaluate and rank the individuals in a population from strongest to weakest. Often in EC, creating a fitness function is the most difficult part as it requires the designers to attempt to predict what sort of rule will successfully select for the desired solutions. Such predictions become increasingly hard with higher genome dimensionality, and complexity of behaviour. The field of robotics in particular exhibit both of these traits.

As a more concrete example, consider a genotype in the form of a 6-element vector $v \in \mathbb{N}^6$. It formally encodes all expressions of that genotype to have the form $(n_0, n_1, n_2, n_3, n_4, n_5)$ where each element is a gene. Assuming a given problem defined on the same space has an optimal solution, one specific individual (a phenotypic expression of a genome) must be this solution. Applying an individual v_0 to some fitness function $F(v)$ gives v_0 a fitness score with which to compare it to other individuals in a population of individuals v_k, $k = 1, 2, ..., K$, where K is the population size. The top ranked individuals may then have their genes recombined, typically by employing some crossover operator that combines genes from two parents to create a new permutation. These offspring permutations are then subjected to a mutation operator that has a set probability of changing a gene, thereby creating novel genes in the population. Finally, the cycle of selection, recombination and mutation is repeated until a solution is found. It is however important to note that due to the stochasticity involved, there is no guarantee that the solution is truly optimal.

In summary, an implementation of EC needs to define a genotype that somehow encodes phenotypes; it needs a well designed fitness function that quantifies how well a genome is suited to the problem; and it must employ recombination and mutation operators to increment genomes towards the optimal solution. With these in place the EC algorithm can perform a non-exhaustive search for an optimum in a space of permutations.

2.2 Neuroevolution

A data structure that is applicable to EC is an ANN; a non-exhaustive search through the space of network weights and topologies to optimize it for some defined problem. Knowing that optimizing hyperparameters for ANNs remains an unformalized task (one is typically left tuning manually without knowing what the effects will be), applying EC to achieve this autonomously is a tempting proposition. Known as Neuroevolution (NE), the design and optimization of ANNs through evolution has seen some remarkable contributions to solving problems of robot control, artificial life and general game playing. The success of an NE approach is very much dependent on the fitness function, which poses a substantial challenge as this needs to be designed in a way that ensures both improvement and a desired behavior. Predicting what the effect of a fitness function will be is hard given the complex space that multi-DOF robots operate in, and training time overheads may be large. Indeed, the fitness landscape is usually so complex and filled with local optima, that premature convergence is a problem that must be solved for any evolutionary algorithm that produces ANNs.

One is also faced with the task of selecting an appropriate encoding for the genotype; how do we represent an ANN in a compact way, that may be decoded into some specific expression of ANN? From the background material provided with this work, we can define three styles of encoding:

a) Direct Representations, a one-to-one relationship between elements in the encoding and the actual parameters they represent in the ANN

b) Developmental Represenations, a specification of a process which constructs the ANN

c) Implicit Representations, the expression of a given gene is implied by the context in which it will exist.

The choice of encoding scheme becomes particularly important when applying evolutionary computing to the development of ANNs, due to the potential complexity of a network.

2.2.1 Direct Representations

A direct representation encodes explicitly the topology and weights of the network, requiring some encoded symbol for each separate element in the network. In the simplest cases, the genotype is represented as a string of real numbers or a string of characters, encoding the weights and connections over some fixed topology. Though this representation has shown excellent results in forming smaller, fixed size networks, limitations in regards to genome length and symbol variation indicate that this is not suitable for larger networks where the topology and/or architecture is also to be evolved and optimized.

A successful venture into co-evolving both the weights and architecture of a network is the NeuroEvolution of Augmenting Topologies (NEAT) method, proposed by [18]. In it, the authors include genetic operators that can introduce new genes and disable old ones, thus allowing the topology of the network to evolve. Additionally, the genes include a historical marker to track the gene's first appearance. This allows the evolutionary search to utilize the recentness of genetic innovations to create sub populations (speciation), while also solving the problem of competing conventions.

2.2.2 Developmental Representations

A developmental representation, or indirect encoding, has the genome describe a process that in turn builds the network, establishing the connections between nodes and their weight. The compactness of these representations allow for the evolution of much larger networks that in turn may solve more complex problems.

An example is an extension of the NEAT method; HyperNEAT [15]. Instead of using NEAT to directly construct the ANN, NEAT creates a Compositional Pattern Producing Network (CPPN), which serves as the genome for the ANN. Along with being an indirect encoding, the CPPN exhibits structural repetitions and symmetries reminiscent of the neural structures of the biological brain.

2.2.3 Implicit Representations

Implicit representations is a style of encoding that perhaps most closely resembles the way genomes encode genetic information in biology. The inspiration comes from the discovery of gene regulatory networks (GRN), where we see that interactions between genes is not explicitly encoded. Instead they implicitly follow from the physical and chemical environments in which the genes exist.

An application of this to ANNs is the Analog Genetic Encoding (AGE) [12]. The genome consists of symbols from some alphabet of characters, where certain sequences are prescribed certain interpretations. Each gene in a genome is thus encoded to express a neuron (i.e describing its activation function) and its terminals (i.e its connection to other neurons). The weights however, are not encoded at all, but are calculated by an interaction map I whenever two neurons are connected.

2.3 NeuroEvolution of Augmented Topologies

The evolutionary algorithm used in this thesis is called NeuroEvolution of Augmented Topologies (NEAT) [18]. As the name implies, it is an algorithm that searches for an optimal network topology, as well as optimizing values for weights and other hyperparameters. This is in contrast to earlier NE approaches, where a network topology is defined before subjecting it to an EA, thus excluding the search for a topology from the evolutionary process.

Though it is known that a fully connected ANN can in principle approximate any continuous function, it remains a difficult and time consuming task to guess at exactly which ANN architecture will produce the desired approximation. By leaving the task of finding the right network topology to the EA, a significant amount of time and effort is saved. Furthermore, an exploratory search such as EA might discover architectures that simply do not come easily to a human designer.

2.3.1 Genotype and Genetic Encoding

The genotype in NEAT algorithm is a pair of lists, one for *connection genes*, and one for *node genes*. Each connection gene describes a connection between two nodes, including cases when both nodes is the node (recurrent connection). It provides the ID of the in-node, the out-node, connection weight, disabled/enabled status and an innovation number. This last element is a historical marker used to track genes and avoid the problem of *competing conventions*, where different networks topologies compute the same function (see [18] for details). The node gene contains a list of input, output and hidden nodes in the network that the node may connect to.

2.3.2 Speciation

As a way to protect genetic innovation, NEAT features a speciation scheme. This allows for genomes with very recent topological mutation to compete in a niche of genomes similar to themselves, thus protecting them from being pushed out of the reproductive pool before having a chance to optimise the new structure. To achieve this, the historical marking of genes comes into play again. Under the assumption that the larger difference between two innovation numbers, the less evolutionary history they share, this difference can be used to quantify a genetic distance between two genes. Researchers in [18] propose measuring compatibility distance δ

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \overline{W}$$

where E and D is the number of excess (matching) and disjoint genes respectively, and $\overline{W}$ is the average weight differences of matching genes, including disabled ones. The coefficients c_1, c_2, c_3 are adjustable importance weights to the terms and N is number of genes in the larger genome to normalise in cases where genome sizes are very different.

By defining some compatibility threshold δ_t, the algorithm manages to maintain an ordered list of species that do not overlap.

2.4 Central Pattern Generators

Central Pattern Generators (CPG) are biological networks of neural circuits that produce rhythmic, coordinated output patterns. By generating these rhythmic signal patterns, they cause most all vertebrates to display periodic motor functions such as breathing, walking, running chewing etc. Interestingly, this is done without the need of a rhythmic input pattern, meaning input signals may be substantially simpler than the produced patterns. A high level control center (motor cortex, cerebellum, basal ganglia) may therefore induce these complex patterns without itself having to generate complex signals to a given CPG.

In many vertebrates [8], these neural circuits are found distributed along the spine, where shorter segments contain networks that all are capable of producing rhythmic activity. Examples of this is seen in studies of the lamprey fish and the salamander [1, 13]. The prevailing model conceptualizing the rhythm generation of neurons is the half-center model, proposed by G.T Brown [3] in 1914. A pair of neuron populations are coupled with inhibitory connections that exhibit an excite and fatigue mechanism. This coupling causes activity from the neurons to take on an oscillatory profile that alternates between stimulating extensor and flexor muscles, in turn creating rhythmic motion.

Furthermore, there is clear evidence that rhythms are generated centrally without requiring sensory information; extracting a spinal chord from a body will still generate patterns of activity when stimulated directly with simple electric of chemical stimuli [5]. Instead, sensory information serves to modulate (neuromodulation) the patterns produced, shaping them in order to better suit changes in the environment. This allows vertebrates to adapt their gait to changes in the terrain, increase respiratory rate to support increased activity, or drastically change locomotion style from walking to swimming.

2.5 Continuous-Time Recurrent Neural Networks

CTRNNs are a type recurrent neural network in which each neuron computes the differential of the input with respect to time. Their definition is given by the differential equation. in Eq. 1.

$$\tau_i \frac{dy_i}{dt} = -y_i + \sum_{j=1}^{N} W_{ij}\sigma(y_j + \theta_j) + I(t) \tag{2.1}$$

where:

- y_i represents the current stat of neuron i
- τ_i is the time constant for neuron i
- θ_j the bias of neuron j
- $I(t)$ is the external sensory input. Only non-zero for input neurons.
- W_{ij} is the weight connection from neuron j to neuron i
- $\sigma(x) = \frac{1}{1+e^{-x}}$ is the activation function

Thus, a CTRNN is characterised as a system of Ordinary Differential Equations, where each neuron represents an equation in the system. The time constant τ is factor unique to the CTRNN. It scales the rate of change, effectively deciding how sensitive a neuron is. Higher values of τ means a heavier damping of the neurons rate of change, and so is less responsive. The reverse is true for lower values of τ.

In contrast to the common feed forward structure of most ANNs, a CTRNN is a network in which any given neuron can be connected to any of the other neurons in the network, including itself. A feed forward network only connects neurons in layer n to neurons in layer $n+1$. The recurrent connections are what allow the CTRNN to have internal states through time, modelling a system that takes previous states into account when calculating the next output, which is what allows the calculation of differential with respect to time.

The CTRNN features frequenctly in evolutionary robotics, and was therefore selected for its successes in the field, but also for the ability to model complex behaviour in continuous time. This will a be nice feature with respect to neuromodulation and transitioning between modes of rhythm.

Chapter 3

Methods

3.1 Chapter Introduction

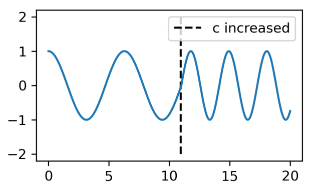
In order, this chapter contains:

- a detailed description of the definition of rhythmic oscillation used and how it is derived
- a description and short discussion of fitness functions
- a description of the simulation environment
- an overview of the experiment paradigms and their motivation

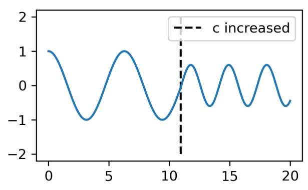
The experiments outlined in this chapter are designed with the intent to shed light on what conditions are required to promote CPG-like behaviour in CTRNNs. Specifically, the goal is to obtain networks that:

1. produce rhythmic output when given constant input
2. modulate the frequency of the output in response to change in the input

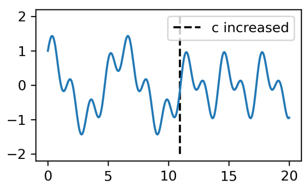
To give an idea on what this thesis is working towards, figure 3.1 shows an imagined ideal of how these CTRNNs should behave. Each figure is made to illustrate outputs when some input control variable c is significantly increased around the halfway point of the time axis. 3.1a shows a pure cosine wave that doubles the frequency at the time of input change with no other side effects. To show that certain side effects



(a) Example output 1: constant amplitude



(b) Example output 2: reduced amplitude



(c) Example output 3: less significant frequency component unaffected.

Figure 3.1: Three examples of high fitness CPG-like behaviour.

are both expected and acceptable, figure 3.1b shows the same cosine, but with a drop in amplitude as the frequency is increased. Lastly, 3.1c shows a cosine with an added, lower amplitude, frequency component. The increase of the control input primarily affects the frequency of the prominent component, and only affects the lesser component as a side effect.

The primary target of investigation is the fitness function, as this is known to be difficult to design and has a large impact on the results of any EC approach. The functions in question are the Root-Mean-Square Error (RMSE), and a second, naively designed function based on the negative exponential of the error. For the sake of brevity, this function will be referred to as NEE in this thesis. A detailed explanation of these two functions are in section 3.3.

The space of the function essentially defines the landscape in which the genomes evolve through in order to reach an optimum. Intuitively, the shape of this landscape has a significant impact on the direction of the evolution and how difficult it is to move through it. A deeper discussion of the shape of the fitness space and the associated challenges can be found in chapter 5.

The simulation environment used in the experiments is relatively simple, and unchanged across all the experiments. Further details, including a comment on side effects the simulation settings may have on the fitness space, can be found in section 3.4

To evolve the CTRNNs, the experiments utilise the NEAT neuroevolution algorithm as implemented by McIntyre et al. [14]. As such, a secondary target of investigation is the impact of a selection of hyperparameters of the algorithm. This is deemed secondary because the findings associated with the NEAT parameters, while interesting, may not be generally applicable to other neuroevolution algorithms. A more detailed description of the parameters and experiment set up is given in section 3.5.

Due to the stochastic nature of EC, each experiment is an evolutionary run of NEAT for 500 generations. This is repeated ten times per experiment to make possible some estimation of whether the results of a particular configuration are consistent. Specifically, the analysis of each experiment is to estimate which variable in the configurations consistently impact the fitness value of the best performing genome. If the impact of configuration variables are greater under one fitness function than the other, then the claim can be made that one defines a better suited fitness space than the other. Thus this thesis only applies quantitative analyses to the best genome from each evolutionary run. For a more detailed overview of each experiment and their configurations, see the subsections of section 3.5.

3.2 Definition of rhythmic behaviour

In order to represent a biological CPG, the evolved CTRNNs must exhibit a rhythmic, oscillatory output behaviour [8]. This means the fitness function must quantitatively express what such behaviour is, in order to rate the fitness of each genome. For the purpose of these experiments, rhythmic output behaviour will be defined as a repeating pattern of extrema, appearing at fixed intervals in time. Any local minimum or maximum will be considered to be extrema of the time-series, a local minimum being defined as a point $f(t)$ where any given point $f(t \pm \epsilon)$ are strictly greater, and vice versa for maxima.

It can be tempting to equate rhythmic output with the notion of periodicity, as a periodic function will repeat itself every period P, which is essentially a rhythm. From calculus we have:

A function f is considered periodic if for some non-zero constant P

$$f(t+P) = f(t)$$

However this definition of periodicity must be considered too rigid for the purpose of these experiments. Within such a definition, a sinusoid with a decaying amplitude would be considered non-periodic as each peak has a lower function value then the preceding peak, hence $f(t+P) \neq f(t)$. Similarly, the presence of some stochastic noise in a time-series analysed in this way will also disqualify the signal. In this study it is desirable to consider the above mentioned time-series output as applicable to the notion of rhythm (the peaks occur at fixed intervals in time), rendering this particular definition of periodicity somewhat ill suited.

In signal processing, it is common to use auto-correlation to find periodic or repeating structures in noisy time-series signals. It allows for discovering patterns that are close to periodic and quantifying the degree of periodicity. Correlation is an analytic measurement of how closely one sequence of data resembles another by applying a slightly modified convolution operation on the two sequences. By measuring the correlation of a sequence with a delayed copy of itself (hence auto-correlation), we can measure how many time shifts before the sequence resembles itself the most, i.e repeats itself. The number of time shifts the signal must be lagged by is the period P of the signal. Finally, it is common practice to normalise the correlation measurement to avoid high variance elements skewing the result to misrepresent the measurement.

The normalized auto-correlation ρ at lag l for a discrete-time signal $y[n]$ is

$$\rho_{yy}(l) = \frac{\sum_{n=-\infty}^{n=\infty} y[n] * y[n-l]}{2\sqrt{\sum_{n=-\infty}^{n=\infty} y^2[n]}}$$

Normalisation scales down the correlation measurement to a value in $[1, -1] \in \mathbb{R}$, where 1 means perfect correlation, and -1 means perfect anti-correlation.

The weakness in this description of periodicity with respect to rhythm, is that it will also consider a constant or mostly constant time-series periodic, and so cannot be used directly. Consider the case where the CTRNN output is perfectly constant, visually speaking a horizontal line. Regardless of what the lag l is, $y[n]$ and $y[n-l]$ will be exactly equal, yielding a perfect correlation score for any given value of l, even though it is clear that a horizontal line is not oscillating. In order to use auto-correlation to capture the desired behaviour, there must simultaneously exist a condition that enforces oscillation.

From calculus and real analysis, there is a concept of oscillation in which one measures oscillation as the difference between a function supremum (also known as least upper bound, LUB) and infimum (also known as greatest lower bound, GLB). The definition of oscillation of a function on an open set should apply, since a time-series is simply a function of time:

Let f be a real-valued function of a real variable. The oscillation ω of f on an interval I is defined as

$$\omega_f(I) = \sup f(t) - \inf f(t)$$

where $t \in I$.

In the case of this body of work, the interval I is the set of time steps over which a CTRNN generates output. Within this interval of time, the supremum and infimum are the upper and lower bounds of the output respectively. This consideration of oscillation does not explicitly capture the notion of rhythm either, as the definition does not require the function to oscillate more than once, yet an event that only occurs once conflicts with the idea of rhythm. Using this definition means CTRNNs with behaviour similar to functions such as $f(t) = \frac{1}{t}$ or $f(t) = at + b$ will be considered to have a non-zero oscillation, without exhibiting the rhythmic behaviour indicative of a CPG.

Of the three possibilities explored above, capturing periodicity by way of auto-correlation seems like the best option due to being flexible enough to discover repeating patterns in potentially noisy time-series. To be considered rhythmic in a CPG-like fashion, however, the following additional conditions will be applied in this study:

Take P to express the logical statement "The signal is periodic."

$$P \Leftrightarrow \exists l \quad | \quad \rho(l) \geq 0.5$$

$$f'(t) = 0 \Rightarrow f'(t+1) \neq 0$$

$$\{t \quad | \quad f'(t) = 0\} \neq \emptyset$$

The first condition sets a minimum requirement for how highly the time-series must auto-correlate (ρ) at a given lag l in order to be considered periodic.

The second condition demands that any point with a zero rate of change cannot be adjacent to another point with a zero rate of change. This rule inhibits constant output CTRNNs.

Finally, the last condition says the set of time points t that correspond to an extrema cannot be empty. This rule inhibits output time-series that have no defined extrema like linear and exponential functions.

Any time-series output that simultaneously meets all three conditions will be considered rhythmic in this thesis.

3.3 Fitness function

The evaluation of a genome's fitness is a process of subjecting the CTRNN expressed by the genome to the simulation, producing an output sequence, and applying this output to a fitness function to quantify its performance. For each genome, the simulation is run several times, each time increasing the input control variable c linearly by sampling from the set of control variables $C = (0, 100] \subset \mathbb{R}$ (see table 3.1). However, the simulations are only run if the output sequence is considered rhythmically oscillating under the conditions given in section 3.2.

The fitness calculation itself can be divided into two parts; a frequency analysis part, and a fitness assignment part. The former is tasked with analysing the output from the genome's CTRNN to decide the frequency of the most prominent frequency component, and then measuring an error between this frequency a desired target frequency (see subsection 3.3.1). This part is the same regardless of which fitness function is used. The latter is tasked with calculating a total fitness measurement as a function of the error from each simulation run (see subsection 3.3.2). In summary, the evaluation loop for a CTRNN individual is as follows:

- Simulate with input c to produce output $x(n)$
- Test for rhythmic oscillation

If $x(n)$ tests positive for rhythmic oscillation:

- Perform frequency analysis to determine dominant frequency
- Measure error between dominant frequency and target frequency
- Sample new value for c and repeat

This loop produces 10 error values (there are 10 values of c) that must be somehow summarised into one quantity that describes the genome's total fitness. As previously noted, this thesis experiments with two fitness evaluations:

$$NEE = \prod^{i} |f_i - f_{t_i}|^{-1} \tag{3.1}$$

$$RMSE = \sqrt{\frac{\sum^{i} |f_i - f_{t_i}|^2}{N}} \tag{3.2}$$

where $i = 1, 2, 3, \ldots, 10$ and indexes each of the 10 input control variables c. For example f_{t_1} is the target frequency inferred by the first value of $c_1 = 10$. Finally $N = 10$ is the total number of simulation runs, and f_i is the measured frequency of the CTRNN's output given input c_i. In both equations, the error term $|f_i - f_{t_i}|$ expresses the difference between the measured frequency and the target frequency, both obtained with the same input control variable. A clearer explanation of how f_i and f_{t_i} are obtained is given below in subsection 3.3.1. The important distinction is that Eq. 1 is a product of errors, while Eq. 2 is a normalised average of errors. The former models a 10-dimensional multi-objective problem with a simultaneous solution in each dimension. The latter in contrast the has only one objective in a 2-dimensional space, which is to minimise the average error. A second important difference is that optima in NEE are maxima, while optima in RMSE are minima. This means high values indicate better fitness in NEE, while the opposite is true for RMSE.

It is notable that Eq. 1 is not defined for $|f_i - f_{t_i}| = 0$. In the practical implementation, Eq. 1 is broken into a two-sided piecewise function to ensure the algorithm makes use of definite values only. For an explanation of its implementation and further details on both functions, see subsection 3.3.2 below. Lastly, it should be made clear that neither of these function explicitly reward rhythmic oscillation. This behaviour is implicitly motivated by not applying the fitness functions at all if the output of a CTRNN is not rhythmic, thus making any reward gain for such CTRNNs impossible.

Simulation No.	1	2	3	4	5	6	7	8	9	10
c	10	20	30	40	50	60	70	80	90	100

Table 3.1: Values of control variable c at each simulation run

3.3.1 Frequency analysis

In order to motivate the NEAT algorithm to discover genomes for CTRNNs that feature a strong relationship between input value and output frequency, the evaluation of each CTRNN requires a frequency analysis of their outputs. It is to be expected that any output time-series will contain multiple frequency components that define the wave form, but only the frequency component with the largest contribution (i.e greatest amplitude) will be given any consideration. Given that the objective is to control frequency, and not specifically the shape of the wave, this study deems it reasonable to focus on the dominant frequency in the output.

Subjecting a CTRNN to the simulation once, produces one discrete time-series $x(n)$ of length $N = \frac{T}{\Delta t}$, where T is the total simulation time in seconds, and Δt is length of each time step in seconds. To inspect the frequency components that make up $x(n)$, the Fast Fourier Transform (FFT) [6] algorithm is employed. The FFT algorithm computes the Discrete Fourier Transform (DFT),

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N}$$

where X_k is the amplitude of discrete frequency component k. However, it is important to note that most implementations of the FFT, including the one used in this study, normalises the frequency range by dividing the frequencies by the sampling frequency $F_s = \frac{1}{\Delta t}$. The effect is that all frequencies given by the DFT are mapped on to a unit free range of values $[0, 1] \in \mathbb{R}$, thus expressing the frequencies relative to F_s where 1 represents the highest frequency F_s can capture. Specifically, the Nyquist-Shannon sampling theorem [11] shows that F_s must be at least twice the rate of the highest frequency of the sampled signal in order for the DFT to avoid so-called aliasing. In other words, 1 represents a frequency $f = \frac{F_s}{2}$. Details about the sampling frequency used in the simulation can be found in section 3.4.

A final note on the effect of normalisation, is that the NumPy implementation of the FFT shifts the range to be centered on 0, making the actual range of normalised frequencies $[-0.5, 0.5]$. Because the frequencies are represented as complex exponentials on the form e^{ix}, any real valued time-series $x(n)$ will have its frequency components described as a complex conjugate pair [11]. The negative half of the normalised frequency range represents the complex conjugates of the positive half. As the DFT is defined for both real and complex signals, such 0-centering is common in signal processing.

Applying the FFT to a CTRNN output time-series $x(n)$ produces a spectrum of frequencies from which the one with the largest amplitude may be extracted. Denoting this frequency as f, it is compared to a target frequency f_t by measuring the error given some input control value c

$$E_c = |f_c - f_{t_c}|$$

To relate f_t to any $c \in C$, a simple transformation of c is necessary as the frequency domain is out of scale with the domain of C. Indeed, the range of C is $(0, 100]$ and the range of normalised frequencies is $[-0.5, 0.5]$.

In the attempt to make the fitness space easier to explore, and to make C easier to relate to the frequency domain, certain transformations have been applied to the frequency domain as well. Firstly, by exploiting the fact that all time-series produced will be real valued, and that it is not the object of the fitness evaluation to restore the time-series from the frequency components, the frequency range has been redefined as $[0, 0.5]$ by simply omitting the negative half. As previously stated, the negative frequencies are the complex conjugate pairs of the positive ones and therefore impart no added information about the frequencies themselves beyond the fact they are derived from a real valued time-series. The conjugate pairs would only be necessary if the time-series was to be reconstructed.

Secondly, the aforementioned, bisected, normalised frequency range has been scaled up by a factor of F_s, undoing the normalisation. The normalisation is only in place to make viewing the spectrum more convenient when dealing with signals with components in the kHz end of the scale. Because a CPG is mainly concerned with controlling robotic limbs, oscillations at even $10Hz$ is pushing the limits of what is realisable. In this regard, the normalisation is not necessary, and possibly detrimental to the search. A brief inspection of performance with normalised frequency range is covered in section 3.4.

Finally, by scaling down any c by $\frac{F_s}{2}$, the input control variable can be directly compared to any non-normalised frequency $f \in [0, \frac{F_s}{2}]$, giving the relationship,

$$f_{t_c} = \frac{c}{\frac{F_s}{2}} = \frac{2c}{F_s}$$

Performing this rescaling of the control variable specifically when comparing to a measured frequency, allows domain of C to remain sufficiently large so as to not lose significance when applied as input to a CTRNN that may evolve an arbitrarily large topology.

3.3.2 Fitness Evaluation

As previously stated, this study features two approaches to fitness evaluation; the RMSE and the negative exponential error. This section discusses their motivation, and in the case of the latter, adds further precision to its definition.

The RMSE evaluation was opted for based on the immediate understanding of the problem; to minimise the difference between CTRNN output frequencies and desired target frequencies. RMSE is a somewhat obvious choice for measuring and minimising a range of errors.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} |x_{1,i} - x_{2,i}|^2}{N}}$$

It is typically used in statistics to measure the error of a predictive model against measured data, which is arguably the very circumstance for these experiments where the output frequencies are the predictions, and the targets are the data the CTRNNs must be fitted to. In more precise terms, it measures the standard deviation across the 10 simulation runs. Thus the proposed objective for the RMSE is to minimise the standard deviation of predicted frequencies with respect to target frequencies.

A known weakness of the RMSE measurement is its sensitivity to outlier data points. However, because the space of predicted frequencies and target frequencies are transformed to identical spaces, as described in section 3.3.1, no outliers can exist. All frequencies, predictions and targets, are within the $[0Hz, 10Hz]$ range.

In the case of the negative exponential function, the error E_c for each applied c is used in an exponential function to calculate a score with respect to the current input c_i.

$$F_{c_i} = E_{c_i}^{-1} = |f_{c_i} - f_{t_i}|^{-1}$$

where c_i is the ith c from the set of input control variables, and f_{t_i} is the target frequency related to c_i by $f_{t_i} = \frac{2c_i}{F_s}$ as seen in section 3.3.1.

To be able to compute F_{c_i}, the function is implemented as a piecewise function to avoid evaluation as $|f_{c_i} - f_{t_i}|$ approaches 0.

$$F_{c_i} = \begin{cases} 10 & E_{c_i} < 0.1 \\ \frac{1}{E_{c_i}} & E_{c_i} \geq 0.1 \end{cases}$$

As the error becomes smaller, the fitness score F_{c_i} approaches infinity. The function is designed to clamp this value to 10 if the predicted frequency f_{c_i} gets within $\frac{1}{10}$th of the target frequency f_{t_i}. If the error is greater than 1, the score is also less than 1, contributing to a reduction of the total fitness score. The total fitness F of the genome is the aggregated product of the scores related to each c_i.

$$NEE = \prod^{i} F_{c_i}$$

Expressing the total fitness as a product of F_{c_i} creates a strong penalty for failing to achieve low errors, as opposed to sums where each F_{c_i} would be an isolated contribution to F. This way, every F_{c_i} with a sufficiently large error will scale down the total fitness, as opposed to simply adding low values. The only way to avoid penalty is for a CTRNN to predict correct output frequencies for all 10 control inputs.

3.4 Simulation environment

Each genome is tested for their fitness in an environment designed to stimulate the CTRNN derived from the genome with a range input control variables c (see table 3.1). The simulation is simple; each individual is subjected to a constant input c for T seconds to produce a time-series output. The simulation is repeated 10 times with a linearly incremented c each time, from $c = 10$ to $c = 100$. The simulation is thus designed to highly motivate a linear relationship between input values and output frequencies.

Every CTRNN is simulated for $T = 25.6s$ seconds with time steps $\Delta t = 0.05$. This produces a time-series with $N = \frac{T}{\Delta t} = 512$ number of time steps. The motivation for this choice is purely practical; the FFT algorithm has minimal computational overhead for sequences of length $N = 2^n \ \forall n \in \mathbb{N}$ [11].

Furthermore, the choice of Δt and T have an important impact on the resolution and, by extension, both the results and fitness space post FFT. The time step Δt is effectively the sampling interval, giving us a sampling frequency $F_s = \frac{1}{\Delta t} = \frac{1}{0.05} = 20Hz$. By the Nyquist limit, the FFT cannot resolve any frequencies higher than $F = \frac{F_s}{2}$. In other words, the configuration of the simulation environment limits the obtainable frequencies to 10 Hz. For one, this means that aliasing may occur for CTRNNs that produce outputs with higher frequencies than $20Hz$, which is presumed to be possible due to the continuous nature of the network. With respect to evolution, the consequence is that rewards may be gained from CTRNNs that output higher frequencies than the control variable actually demands, but that have alias frequencies sufficiently close to the target frequency. Potential aliasing aside, considering the fact the ultimate goal of the CPG is to provide motion control to the limbs of a robot, a maximum output frequency of $10Hz$ seems otherwise reasonable. Possibly more consequential than aliasing is how the sampling frequency defines the frequency domain. As the sampling frequency sets a limit to the frequencies that can be obtained from any sequence, the range in the frequency domain is always $[0, \frac{F_s}{2}]$. If the experiments had been performed with a higher value for F_s, the frequency domain range would obviously be wider. Keeping in mind that the frequency domain is a dimension of the fitness space, the choice of F_s also impacts the shape of this space.

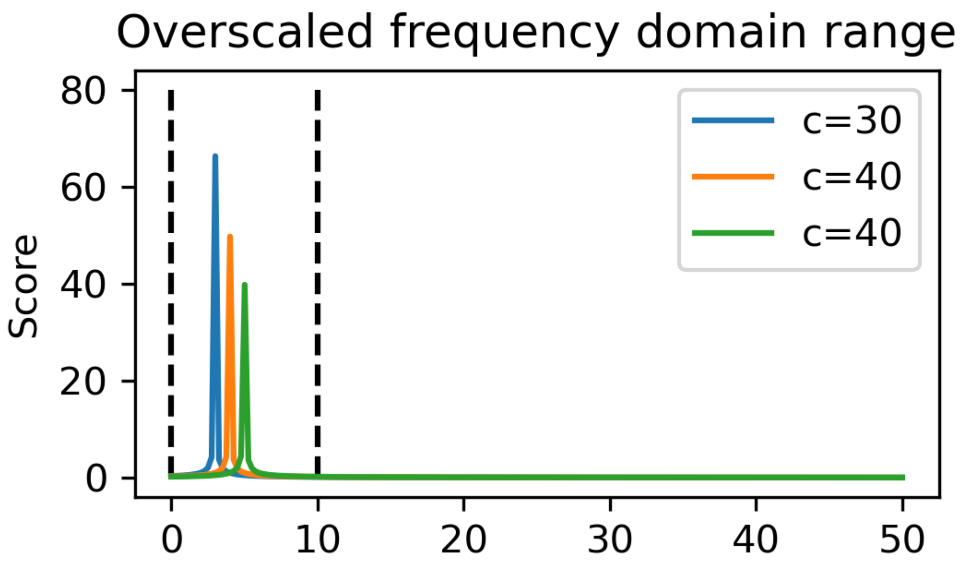


Figure 3.2: Possible fitness scores from NEE over all frequencies for three values of c. Dashed lines indicate the region of interest.

Figure 3.2 shows how this may create unexpected problems, here visualized with the exponential fitness function. Here $F_s = 100Hz$, making the frequency domain range $[0, 50Hz]$. This is a problem because the experiment only targets frequencies in the $[0Hz, 10Hz]$ range, creating a large "no mans land" of fitness scores $0 < F_{c_i} < 1$. When a population of genomes is generated, there are random variations to the genomes. The figure makes an intuitive argument that the majority of the genomes will land somewhere in the "no mans land", with no incentive to explore in any direction. Even if the frequency domain range was normalised by the sampling frequency, as is common, the "promised land" would only be narrowed down in proportion. Somewhat counter to intuition, this indicates that a high resolution time step in the simulation has an adverse effect on the results. It is this observation that motivates the choice of $\Delta t = 0.05$, in turn giving a $F_s = 20Hz$.

For the purpose of testing the winner CTRNNs at the end of evolution, the simulation is reconfigured to apply different input control variables than the ones used during simulation. Sampling the control variable space $C = (0, 100] \subset \mathbb{R}$ with intervals of 25 yields the set of control variables $C_{sampled} = \{25, 50, 75, 100\}$. Given the continuous definition of the CTRNN, the simulation should result in continuous change in the output as the control variable changes through $C_{sampled}$. The motivation for this choice is to see that the CTRNNs do not entrain to the specific values of c used during evolution, but are able to infer correct frequency outputs when given novel control inputs.

3.5 Experiment paradigms and details

The experiments performed have the purpose of attempting to discover the conditions under which CPG-like CTRNNs appear during the evolutionary process of the NEAT algorithm. More specifically, the scope is that the CTRNNs discovered should exhibit the ability to modulate the output frequency in proportion to a control variable c drawn from the set of control variables $C = (0, 100] \subset \mathbb{R}$. To this end, 16 different experimental paradigm have been devised for this thesis, each with the same goal to see what may beneficially influence the algorithm's ability to create CTRNNs with the aforementioned features. The motivation for such a set up is that in comparing the results gained under each paradigm, it should be possible to say something about what conditions make neuroevolved CTRNNs possible, and/or what improves the process of evolving them.

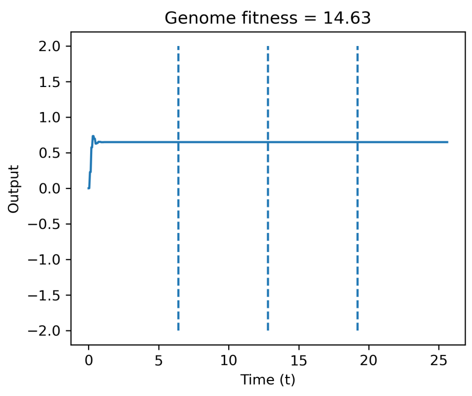
Each experimental paradigm consists of ten runs of the evolutionary algorithm, each running for 500 generations. Each of the ten runs returns one all time best genome; a winner genome. It is the fitness of these winner genomes that will be used as the quantified variable for comparing paradigms.

A paradigm is defined by:

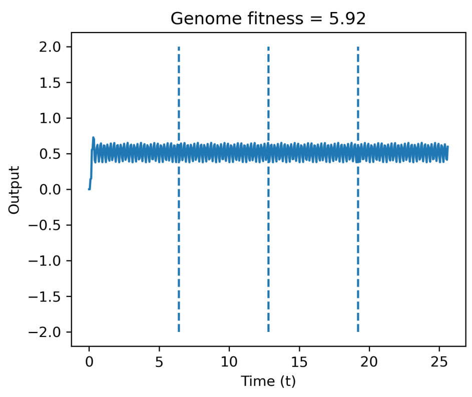
- a choice of fitness function
- a choice of value for the time constant τ
- a choice of default genome configuration which in itself contains several variables

The 16 paradigms are split into RMSE based and NEE based paradigms. For each fitness function, eight configurations of the default genome are applied to the evolutionary runs (table 3.2). Repeating the genome configurations for both fitness functions allows for the comparison of the functions and their respective fitness spaces. If the genome configurations generally shows a lower ability to obtain good fitness scores under one fitness function, but manages significantly better scores in the other, this implies something about the suitability of the fitness functions themselves.

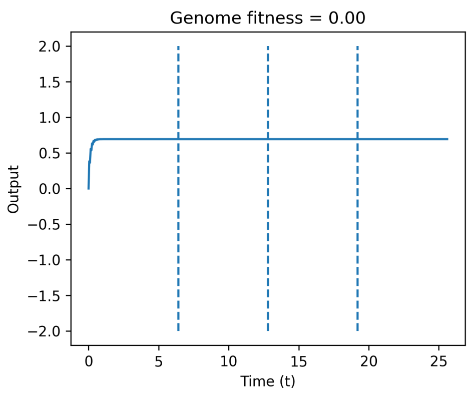
A highly fit individual, is a genome that successfully and proportionately increases the frequency of the output time-series, and whose output is always rhythmic, for all input values $c \in C$. The details for the fitness evaluation can be found in section 3.3 of this chapter.



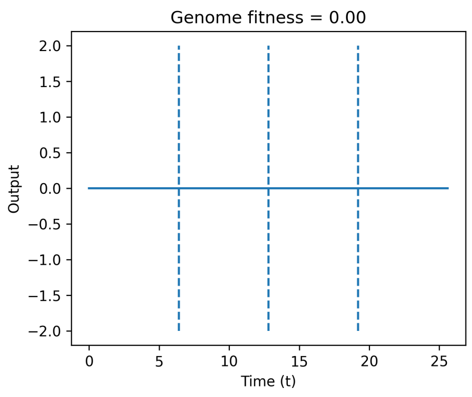
(a) Typical winner CTRNN output with baseline configuration.



(b) Oscillating winner is less common, but possible with baseline configuration.



(c) The first of two categories of output seen with baseline genome configuration and NEE fitness function. Essentially constant with a transient starting phase.



(d) Constant output is one of two categories of output seen with baseline genome configuration and NEE fitness fuction.

Figure 3.3: Samples of four winner CTRNNs; two under the RMSE fitness paradigm and two under NEE. Dashed lines separate control variable regions; $c = 25, 50, 75, 100$.

The default CTRNN genome essentially defines a evolutionary starting point, and the configuration specifies parameters for how evolution may change it. The configuration space for the neat-python implementation is quite large (See table 3.3), and so a selection of variables has been made to focus the scope of the study. The variables investigated are as follows:

- Activation function
- Add connection probability

- Remove connection probability
- Add node probability
- Remove node probability

All these are accessible in the configuration file used by the neat-python implementation. A notable absence from the file is anything related to the time constant. This is simply because there exists no gene in the NEAT genotype for this variable, and means that in this implementation, this variable is not subjected to transformation through evolution, and remains both constant and equal for all neurons in all genomes throughout a run. The time constant is instead changed between each evolutionary run. Table 3.3 shows the different default genome configurations used in each paradigm.

To form a baseline for comparing experimental results, the pre-existing default configuration of a CTRNN genome is used. This configuration was created by the neat-python developers [14] as an example of a configuration that could successfully evolve a CTRNN to solve a robotics benchmark pole balancing problem in 2D space. Figure 3.3 shows samples of RMSE and NEE winners with the baseline configuration. Further analysis and review of these and many other results can be found in chapter 4. The measured performance of this configuration in the evolutionary framework proposed in this thesis will be the foundation from which it measures improvements with regards to creating the desired CTRNN.

3.5.1 Configuration file and Default Configurations

The configuration file allows the specification of certain constraints and initial values of the genomes as well as the evolutionary process itself. The configuration file represented by table 3.3 is the configuration for the default genome used to solve the pole balancing problem and that serves as the baseline configuration. As noted, only a subset of these will be investigated for their effect on generating a CPG-like CTRNN.

BASELINE CONFIGURATION Z		DEFAULT CONFIGURATION D	
Activation function	Sigmoid	Activation function	tanh
Add conn. prob.	0.2	Add conn. prob	0.2
Remove conn. prob	0.2	Remove conn. prob.	0.2
Add node prob.	0.2	Add node prob.	0.2
Remove node prob.	0.2	Remove node prob.	0.2
Time constant τ	0.1	Time constant τ	0.25
DEFAULT CONFIGURATION A		DEFAULT CONFIGURATION E	
Activation function	Sigmoid	Activation function	tanh
Add conn. prob.	0.2	Add conn. prob.	0.2
Remove conn. prob	0.2	Remove conn. prob.	0.2
Add node prob.	0.2	Add node prob.	0.2
Remove node prob.	0.2	Remove node prob.	0.2
Time constant τ	0.25	Time constant τ	0.5
DEFAULT CONFIGURATION B		DEFAULT CONFIGURATION F	
Activation function	Sigmoid	Activation function	tanh
Add conn. prob.	0.2	Add conn. prob.	0.8
Remove conn. prob	0.2	Remove conn. prob.	0.6
Add node prob.	0.2	Add node prob.	0.8
Remove node prob.	0.2	Remove node prob.	0.6
Time constant τ	0.5	Time constant τ	0.1
DEFAULT CONFIGURATION C		DEFAULT CONFIGURATION G	
Activation function	tanh	Activation function	tanh
Add conn. prob.	0.2	Add conn. prob.	0.8
Remove conn. prob	0.2	Remove conn. prob.	0.6
Add node prob.	0.2	Add node prob.	0.8
Remove node prob.	0.2	Remove node prob.	0.6
Time constant τ	0.1	Time constant τ	0.5

Table 3.2: Tables showing the default genome configuration for each experiment. Rows highlighted in gray indicate which variables have been altered with respect to the baseline Z.

In addition to the configurable elements in the genome itself, there are sections in the configuration file not present in the table, that deal with controlling the evolutionary process, speciation, stagnation of species and reproduction. The only of those settings adjusted in these experiments are from the evolutionary control section, where a statement has been set to avoid halting the evolution when some fitness threshold has been met. This ensures that all 500 generations are run, regardless of how high the fitness in previous generations have reached.

Input nodes	1
Hidden nodes	1
Output nodes	1
Initial connection	Partial direct 0.5
Feed forward	False
Compatibility Disjoint coefficient	0.1
Compatibility Weight coefficient	0.6
Add connection probability	0.2
Remove connection probability	0.2
Add node probability	0.2
Remove node probability	0.2
Activation default	Sigmoid
Activation options	Sigmoid
Activation mutate rate	0.0
Aggregation default	Sum
Aggregation options	Sum
Aggregation mutate rate	0.0
Bias initial mean	0.0
Bias initial standard deviation	1.0
Bias replace rate	0.1
Bias mutate rate	0.7
Bias mutate power	0.5
Bias maximum value	30.0
Bias minimum value	-30.0
Response initial mean	1.0
Response initial standard deviation	0.0
Response replace rate	0.0
Response mutate rate	0.0
Response mutate power	0.0
Response maximum value	30.0
Response minimum value	-30
Weight maximum value	30
Weight minimum value	-30
Weight initial mean	0.0
Weight initial standard deviation	1.0
Weight mutate rate	0.8
Weight replace rate	0.1
Weight mutate power	0.5
Enabled default	True
Enabled mutate rate	0.01

Table 3.3: Default genome configuration. An in depth explanation of the configuration file can be found here [14]

The default genome configurations for each experiment, as listed in table 3.2, otherwise have the same values as seen in table 3.3. Further explanation and motivation for the subset of values that are changed can be found in the subsections below.

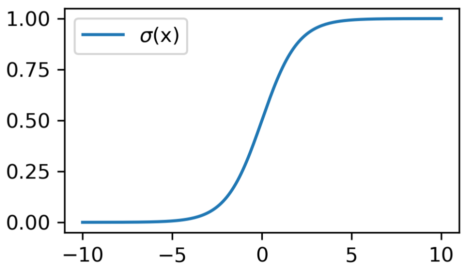
3.5.2 Time Constant

As a reminder, the time constant factors into the differential equation that models the CTRNN neuron.

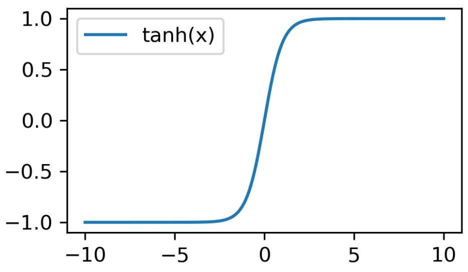
$$\tau_i \frac{dy_i}{dt} = -y_i + \sum_{j=1}^{N} W_{ij}\sigma(y_j + \theta_j) + I(t)$$

From the differential equation, τ should be interpreted as a scaling factor to the differential of y_i. In essence it affects the neuron's sensitivity to change; a high value of τ implies a lowered sensitivity by scaling down the rate of change and vice versa. Special consideration must be taken with regards to the relationship between the value of τ and the size of the time steps with which we advance time during stimulation if the network. The time step Δt is essentially the sampling rate with which the simulation samples values from a given CTRNN. Should τ be too small, thereby scaling up the rate of change, the network may produce oscillations that are too fast to be captured by the sampling rate, adding a risk of aliasing as explained in section 3.3.1. Oversensitivity may also produce CTRNNs with noisy, unstable outputs, that struggle with yielding low frequency output. On the other hand, too large a τ will make the neurons sluggish so that high frequency outputs become unobtainable. This relationship is non-trivial however, and so analytically expressing the optimal value is problematic. However the assumption is that τ should be greater than Δt. For a closer look at the simulation and sampling rate, refer to section 3.4.

Because of the clear effect τ has on the output rate of change, it by extension should have an impact on frequency, with lower values of τ implying a tendency toward high frequency output, and inversely for higher values of τ. This makes it an interesting candidate for experimentation.



(a) The sigmoid function spans from 0 to 1.



(b) The tanh function spans from -1 to 1.

Figure 3.4: Plots of the two activation functions.

With respect to the chosen time step $\Delta t = 0.05$, the experiments will test values for $\tau = 0.1$, $\tau = 0.25$ and $\tau = 0.5$ in the hopes of empirically deciding a suitable value.

3.5.3 Activation Function

Though the field of artificial neural networks feature an abundance of different activation functions that are applicable, only two will be investigated in these experiments; the sigmoid function and the hyperbolic tangent (tanh) function function. The two are defined respectively as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

and

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

Though in terms of shape they appear similar, the tanh function maps argument values to a space between $[-1, 1]$, while the sigmoid maps to the space $[0, 1]$. Intuitively, there is an expectation for the tanh to outperform the sigmoid in this case, as the lack of negative values may bias the neurons toward positive rates of change.

The sigmoid activation function has roots in the early days of machine learning. It models an early understanding of the neuron as being inactive, until its potential crosses a threshold and it fires. This is reflected in the function's co-domain of $[0, 1]$. However, biological CPGs are thought to be clusters of neurons consisting of inhibitory and excitatory networks that in combination produce the characteristic rhythmic pattern [8]. Given this duality of excitation and inhibition, this study posits CTRNN neurons with a tanh activation function may emulate the neuron clusters of biological CPGs more accurately. The reasoning being that the co-domain of tanh is $[-1, 1]$, thus modelling the inhibitory neurons with values < 0.

From table 3.3 it is notable that the genome is configurable to allow for mutation of the activation function, meaning that the evolutionary process itself discovers which activation function provides the higher fitness scores, provided that a list of activation options is supplied and a mutation rate $\mu > 0$ is set. In order to compare the effects these activation function have, however, the experiments are designed to be performed with only one option available at a time.

3.5.4 Network Topology mutation probability

A set of four variables influence how quickly the topology changes throughout an evolutionary run.

1. Add connection probability
2. Remove connection probability
3. Add node probability
4. Remove node probability

The experiments in which these probabilities are altered (F and G) are designed to see if large networks are typical of high performing genomes. To achieve this, the genome configuration for these experiments sets the probability of adding structure high (0.8) and the probability of removing structure slightly below is (0.6). In other words, the mutation operation will add a connection or a node slightly more often than it will remove them. This obviously motivates the genomes to grow large in size.

For clarity, the genotype used by the NEAT algorithm has genes that identify both individual neurons and the connecting weights between them. Since the ability to evolve both weights and neuron topology is a defining feature of the NEAT algorithm, this study makes sure to take advantage of that. As the emergence of Deep Learning has shown us that topology can have a great significance for network performance, the motivation for experimenting with these variables should be apparent. For a better understanding of the genotype, refer to chapter 2

Chapter 4

Experiments and Results

4.1 Chapter Introduction

The following chapter presents the experimental results produced. Initially, it will show the effects of the different genome configurations presented in chapter 3. The chapter contains both a quantitative analysis of the results by way of a Mann-Whitney U test, and a qualitative review of the wave forms produced.

As detailed in chapter 3, section 3.5, there are a total of 16 experimental paradigms in this study. It features 8 different default genome configurations, A, B, C, D, E, F, G and a baseline configuration Z. Each of these are tested with the RMSE and the NEE fitness functions, thus composing the 16 paradigms. The experiment performed under each paradigm is repeated 10 times to obtain a distribution of winner genomes fitness scores from the evolutionary runs, the definition of winner being the best fit genome throughout the 500 generations. The distribution of winners should indicate something about how likely it is to discover highly fit genomes, and how susceptible (if at all) the fitness space is to genome configuration variations.

To quantify the impact the genome configurations have on the results, the Mann-Whitney U test will be employed to compare the distributions obtained from the genome configurations A - G, to the distribution obtained with the baseline genome configuration Z. This comparison will not be performed across fitness functions, as the fitness scores gained through each will not be reasonable to compare; RMSE optima are minima, while NEE optima are maxima. Additionally, since the fitness functions do not explicitly reward for rhythmic oscillation, a qualitative review of the output time-series is also included. Table 4.1 shows how the study categorises winner outputs in terms of qualitative features. In the sections below, only visualisations of the qualitatively best and worst ranked samples from the corresponding distribution will be shown and reviewed. The remaining graphs will be left in the appendix for the reader to review at their own leisure.

Qualitative rank	**Rank explanation**
Rank 1	Output shows rhythmic oscillation and a response to change in input
Rank 2	Output shows response to change in input
Rank 3	Output shows rhythmic oscillation
Rank 4	Output is non-constant, but not oscillating
Rank 5	Output is constant.

Table 4.1: Table of qualitative categories and their rankings for the purpose of this study's qualitative review of results. Oscillation in the context of the study is any curve with at least one clear peak.

The reader is reminded that the main objects of study are the fitness functions. Even though it may appear that much of the focus is directed at the different default genome configurations, these variations are primarily meant to implicitly elucidate the fitness functions' suitability to solve the problem.

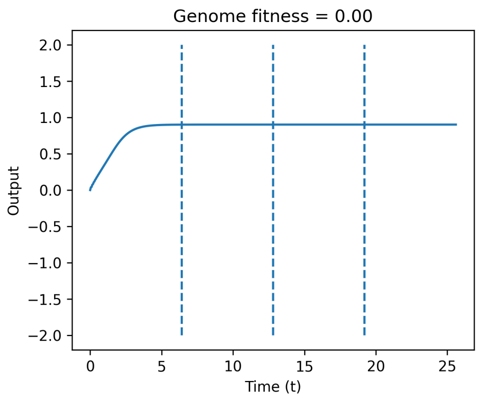
A note on winner genome visualisations

The neat-python [14] implementation used comes equipped with visualisation scripts that draw directed graph representations of the winner CTRNN networks. A brief explanation of the graph components is necessary.

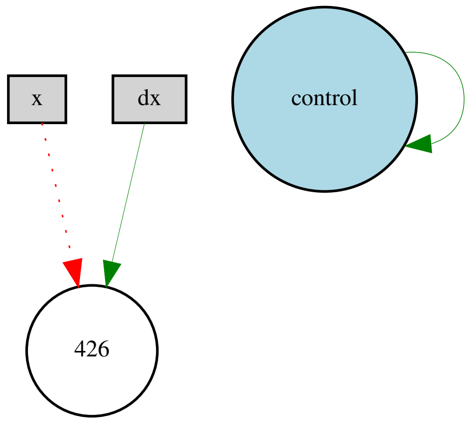
Figure 4.1 shows an example of how a winner genome of interest is presented. Subfigure (a) features dashed lines that indicate at which point during the simulation the input control value c was changed, implicitly defining four regions in which this value is constant. This highlights whether or not the input has the desired effect on the frequency of the output. It is emphasised that this output graph is obtained from the final test simulation of the winner genome after evolution. In these post evolution tests, there are only four samples of c instead of ten as during the evolution. This is motivated in chapter 3.4.

Subfigure (b) features the nodes of the CTRNN network and the connecting weights. Each connection is represented as an arrow with certain variations:

- Green, weight $w > 0$
- Red, weight $w \leq 0$
- Full, enabled
- Dotted, disabled



(a) Dashed lines separate regions of control input value. First quarter has $c = 25$, second $c = 50$, third $c = 75$, and fourth $c = 100$.



(b) Directed graph shows nodes and connections. Gray boxes indicate input nodes, light blue circle indicates output node, and white circles indicate hidden nodes.

Figure 4.1: Rank 4 winner example of output and network graphs.

The enabled/disabled identifiers is in reference to the gene that represents the connection in question. The NEAT algorithm may switch off genes as part of mutation, with a set chance of it being switched back on in a later generation. A disabled connection does not transfer any values, regardless of what the weight of said connection may be.

The gray coloured boxes indicate the input nodes, meaning the symbol x represents what this thesis has consistently called c. An interesting feature is the differential of x, dx has its own separate connection, showing that the CTRNN has the ability to use the input and its differential disjointly.

The light blue circle labeled "control" is the output node, while white nodes are hidden nodes. The number featured on the hidden nodes is a reference to an ID number each gene carries. This is a feature of the NEAT algorithm [18] that allows the recording of genetic innovation by giving each new gene a uniquely identifying integer. This is primarily used to measure genetic distance for the purpose of speciation. Consequently, a high number should be interpreted by the reader as an indication of genetic novelty.

4.2 Results from RMSE based paradigms

Figure 4.2 shows the distribution of all-time best fitness scores under the RMSE based paradigms. In the case of the RMSE experiments, the evolutionary goal is to minimise the RMSE error, meaning that lower RMSE scores imply a better fitness.

From this it is clear that the baseline configuration Z has the highest variance of winner fitness, ranging from a best case of $RMSE = 5.9$ to a global worst $RMSE = 20.0$, the latter being the worst possible error in the experiment.

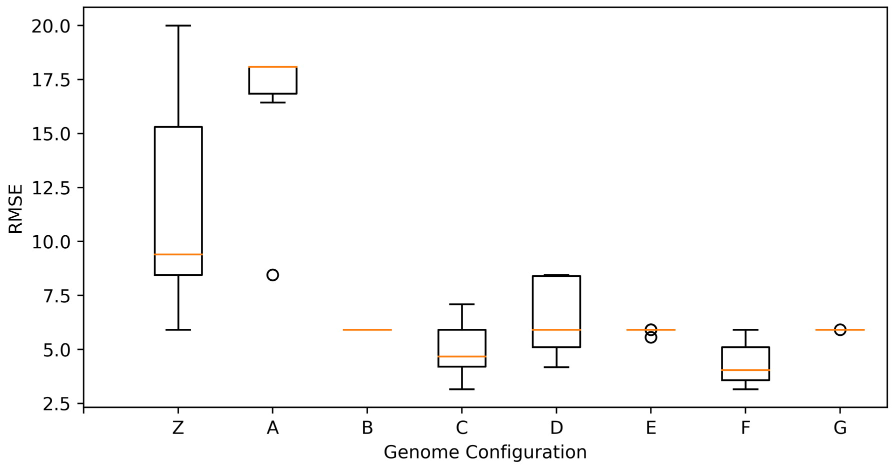


Figure 4.2: Boxplot of the distribution of all-time highest fitness from each genome default configuration. Low values indicate better fitness. Orange lines denote medians and circles denote outlier samples

Experiments with genome configurations B, E, G are on the opposite end of the scale, where all the ten winner genomes have the same fitness, thus making the median represent the entire distribution. The plot reports outliers in E and G, but even these are very close to the median value. Table 4.2 gives a more quantified appraisal of the distributions. In terms minimal error across all the experiments, we find two configurations that outperform the others. Both genome configuration C and F are able to find an optimum of 3.16 in the RMSE fitness landscape. A closer look at what might explain this is provided in the sections below dedicated specifically to each genome configuration.

Config	Median	Min	Max
Z	9.4	5.92	20.0
A	18.09	18.09	8.45 (outlier)
B	5.92	5.92	5.92
C	4.67	3.16	7.09
D	5.92	4.18	8.45
E	5.91	5.56 (outlier)	5.92 (outlier)
F	4.06	3.16	5.92
G	5.91	5.91	5.92 (outlier)

Table 4.2: Companion table to fig. 4.2 showing the precise value for the statics shown in the figure. Configurations that contain the global best fitness $RMSE = 3.16$ are highlighted in gray.

Config	U	p
Z - A	15.5	0.0089
Z - B	10.0	0.0007
Z - C	5.0	0.0007
Z - D	14.5	0.0071
Z - E	2.0	0.0002
Z - F	2.0	0.0003
Z - G	1.0	0.0001

Table 4.3: Table of findings with the Mann-Whitney U test when comparing the baseline Z fitness distribution to the fitness distributions of the other genome configurations. Distributions from configurations C and F are highlighted for cross reference with table 4.2.

Table 4.3 gives insight into whether the differences in the distributions fitness could be considered meaningful. The Mann-Whitney U test measures the probability that two distributions is sampled from the same population. The assumption of this thesis as that if a distribution of fitness scores obtained under a given configuration X can be show to be sampled from a different population than the baseline Z, then the variables in X that differ from Z must explain the difference in fitness distribution.

The Mann-Whitney U test used here is two-sided, so it provides two test statistics $U1$ and $U2$. If the lowest of the two is below the test's critical value, the null hypothesis (the populations are equal) can be rejected. As it is a type of rank sum test, the critical value is defined by the number of sampled in the distributions. A table showing these critical values with respect to sample size can be found in Appendix II. The critical value for a 99% confidence interval is 16. Thus table 4.3 shows that all genome configuration variations have a statistically significant impact (all p-values $p < 0.01$) on the fitness distribution, changing it to such a degree that is almost certain that the fitness scores are not from the same population.

Having shown a high likelihood that the changes the genome configuration have a meaningful effect, the inquiry can move on to look at *why* they are different, and to some extent why some configuration variables have a more positive impact than other.

Config	**No. Rank 1**	**No. Rank 2**	**No. Rank 3**	**No. Rank 4**	**No. Rank 5**
Z	0	0	0	0	0
A	0	0	0	6	4
B	0	0	0	6	4
C	0	0	1	9	0
D	0	0	1	9	0
E	0	0	4	6	0
F	0	1	3	6	0
G	0	0	2	8	0

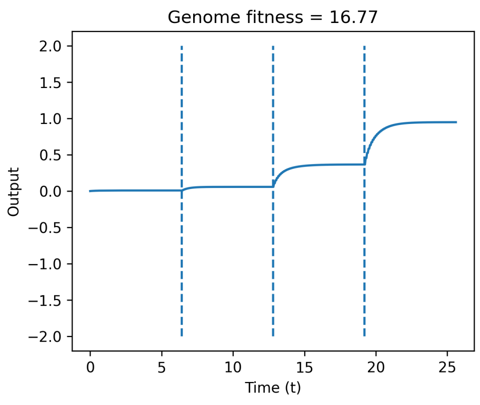
Table 4.4: Table of the number of winner genomes in each rank within each genome configuration distribution. It is notable that configurations Z, A, B have no winners with rank higher than 4, while the rest have no winners of rank 5. Configurations C - G all employ the tanh activation function.

4.2.1 Default genome configuration A

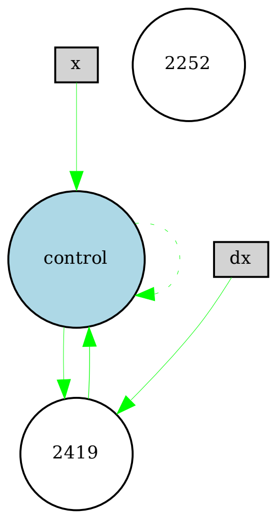
DEFAULT GENOME CONFIGURATION A	
Activation	Sigmoid
Add conn. prob.	0.2
Remove conn. prob.	0.2
Add node prob.	0.2
Remove node prob.	0.2
Time constant τ	0.25

Genome configuration A shows a strong tendency for very high errors as seen in figure 4.2. The only variable changed from the baseline genome configuration is the time constant τ. The observation that this configuration performs the worst, implies both that τ has an import impact on the what fitness scores are possible to obtain. However, genome configuration A is not the only to have a $\tau = 0.25$. Genome configuration D has the same τ, but the activation function is changed to tanh. The improvement is quite drastic (the median changes from 18.09 in A to 5.92 in D), implying the activation function may be even more important.

The configuration does not satisfy in a qualitative perspective either (figures 4.3, 4.4), and is quite aligned with the quantitative evaluation. The best ranked does show an effect on the output amplitude rather than frequency, and the majority of the samples are of the lowest rank (4.5. It can be tempting to explain the better sample's success with the fact that the input node is directly connected to the output node, but this is featured in the network graphs of the worse sample as well.

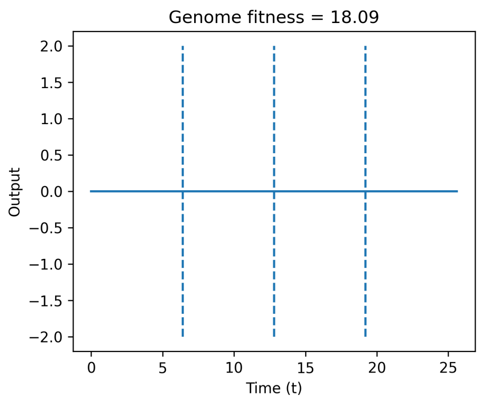


(a) Rank 2 output.

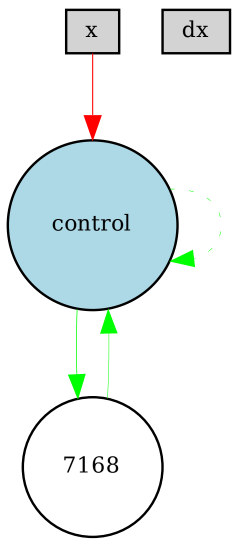


(b) CTRNN topology

Figure 4.3: CTRNN output and network topology of the qualitatively best winner. Evolved in run number 5



(a) Rank 5 output.



(b) CTRNN topology

Figure 4.4: CTRNN Output and network topology of the qualitatively worst winner. Evolved in run number 2

Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
0	1	1	1	7

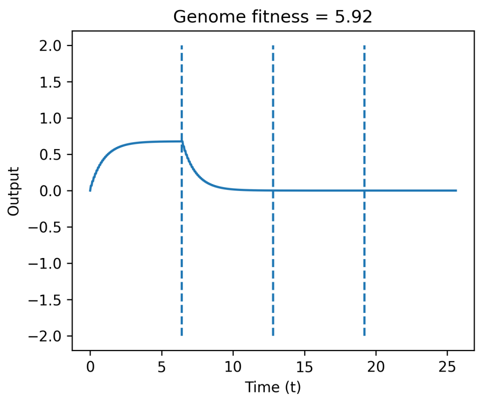
Table 4.5: Table of the number of winner genomes at each qualitative rank in the distribution of configuration A.

4.2.2 Default genome configuration B

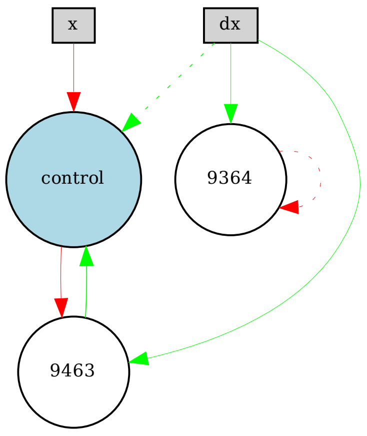
DEFAULT GENOME CONFIGURATION B	
Activation	Sigmoid
Add conn. prob.	0.2
Remove conn. prob.	0.2
Add node prob.	0.2
Remove node prob.	0.2
Time constant τ	0.5

Genome configuration B show a potent tendency towards the RMSE fitness score of 5.92 as none of the ten winners have been scored differently, essential showing 0 variance. This implies the genomes of this type gravitate very heavily to this local optimum. Curiously, where increasing τ from 0.1 in Z to 0.25 in A seemed to reduce the performance significantly, further increasing τ to 0.5 in B has improved the fitness distribution to surpass both Z and A.

In a qualitative view, however, configuration B has a much less impressive advantage over Z and A. Like A, the majority of the samples are categorised as rank 5 (4.6), though it also has the remaining samples exclusively in rank 2. In the best cases, the input's apparent effect seems to also include the amplitude of the output, though only for input values $c < 50$. Figures 4.5 and 4.6 show samples of best and worst cases respectively. Figure 4.6 b) shows an unusually large network for this configuration. Considering both it's qualitative rank and the quantitative measurements, this might suggest that having larger networks has a low impact on the fitness.

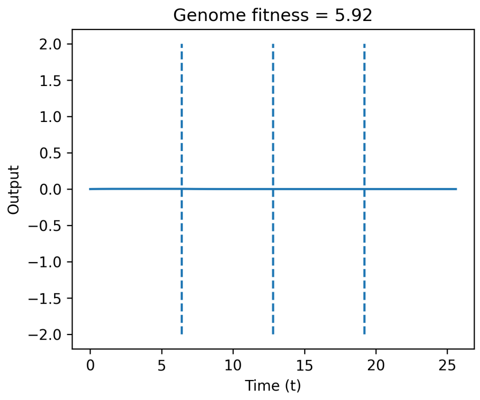


(a) Rank 2 output.

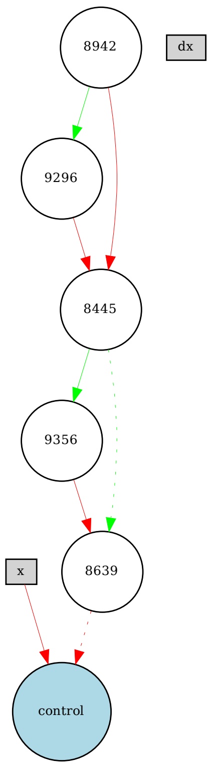


(b) CTRNN topology

Figure 4.5: CTRNN output and network topology of the qualitatively best winner. Evolved in run number 9



(a) Rank 5 output.



(b) CTRNN topology

Figure 4.6: CTRNN output and network topology of the qualitatively worst winner. Evolved in run number 7

Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
0	3	0	0	7

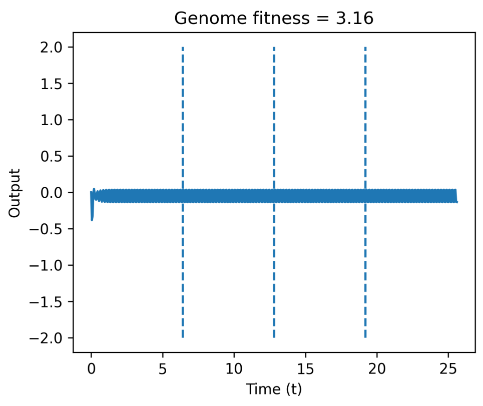
Table 4.6: Table of the number of winner genomes at each qualitative rank in the distribution of configuration B.

4.2.3 Default genome configuration C

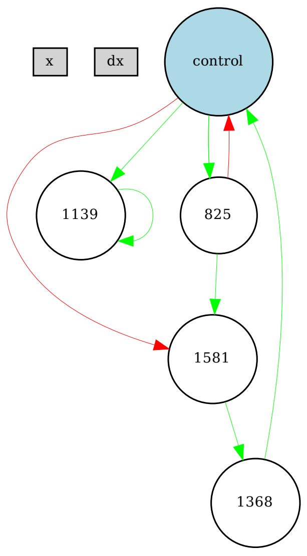
DEFAULT GENOME CONFIGURATION C	
Activation	tanh
Add conn. prob.	0.2
Remove conn. prob.	0.2
Add node prob.	0.2
Remove node prob.	0.2
Time constant τ	0.1

Genome configuration C is the first to sport the tanh activation function. The analysis of the distributions (4.2, 4.2) indicate that this configuration is one of two top performing configurations. Qualitatively it shows no samples of constant output, but also none that are affected by any input value; both an improvement and a worsening (table 4.7).

In observing the network topology graphs 4.7, 4.8, a distinct trait can be seen that is typical for all the networks in this distribution. There is a lack of any connection from the input node x to any other node in the network. Other networks may show the connection as disabled (with a dotted line in the arrow), but they all exhibit an input node that for either reason cannot transmit values to the rest of the network. The reader may find visualisations of this in the appendix.

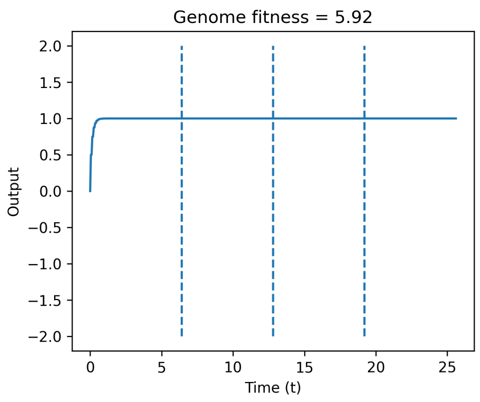


(a) Rank 3 output.

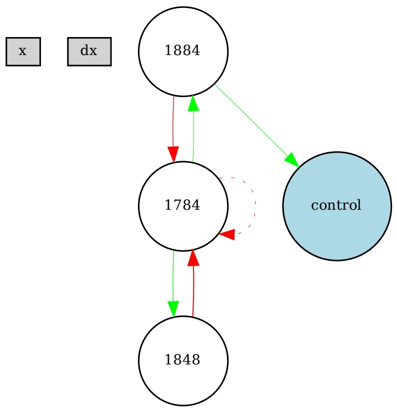


(b) CTRNN topology

Figure 4.7: CTRNN output and network topology of the qualitatively best winner. Evolved in run number 6



(a) Rank 4 output.



(b) CTRNN topology

Figure 4.8: CTRNN output and network topology of the qualitatively worst winner. Evolved in run number 1

Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
0	0	9	1	0

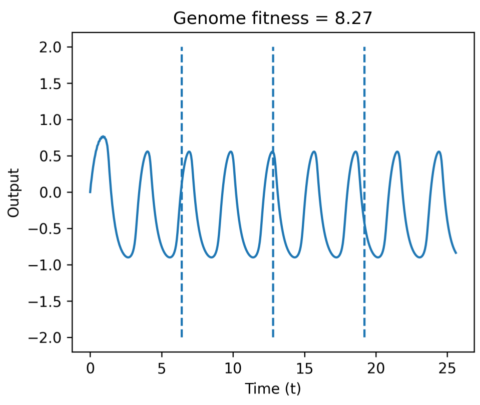
Table 4.7: Table of the number of winner genomes at each qualitative rank in the distribution of configuration C.

4.2.4 Default genome configuration D

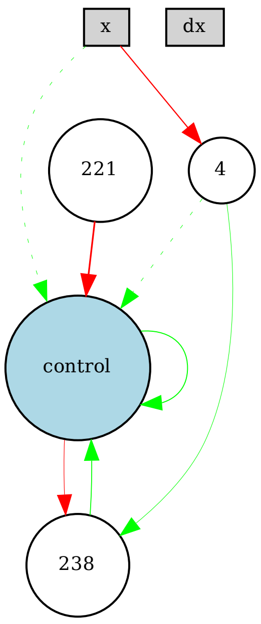
DEFAULT GENOME CONFIGURATION D	
Activation	tanh
Add conn. prob.	0.2
Remove conn. prob.	0.2
Add node prob.	0.2
Remove node prob.	0.2
Time constant τ	0.25

Like in configuration A, the analysis of the distribution shows a loss in performance for configuration D. This configuration also changes the value of τ to 0.25, but the loss in performance relative to the configurations that also use the tanh activation function is is not as pronounced as the disparity between Z and A, or A and B. This reinforces the earlier suggestion that the activation function is more important to the fitness scores achieved than the value of τ.

Figure 4.9 shows another interesting effect the value of τ may have. The frequencies generated are generally lower. This is in line with the a priori observation that τ controls neuron responsiveness by scaling the rate of change. A higher value for τ should decrease the responsiveness, as we may confirm here with the lowered frequency.

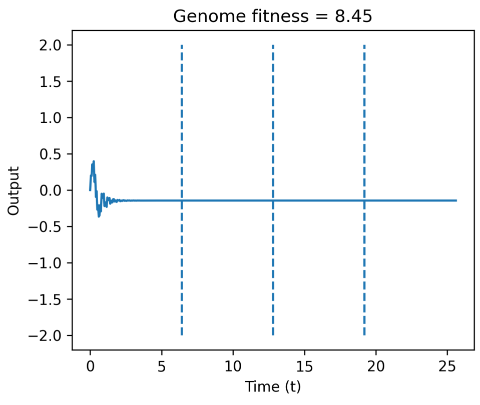


(a) Rank 3 output.

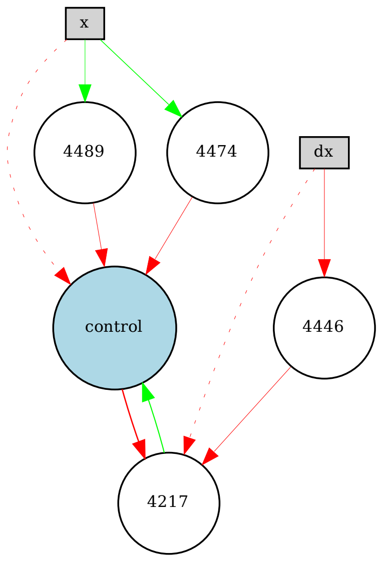


(b) CTRNN topology

Figure 4.9: CTRNN output and network topology of the qualitatively best winner. Evolved in run number 1



(a) Rank 3 output.



(b) CTRNN topology

Figure 4.10: CTRNN output and network topology of the qualitatively worst winner. Evolved in run number 2

Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
0	0	10	0	0

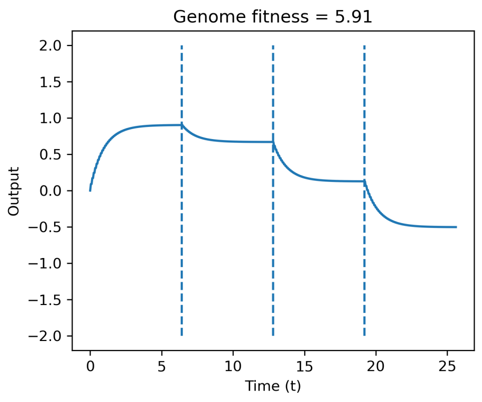
Table 4.8: Table of the number of winner genomes at each qualitative rank in the distribution of configuration D.

4.2.5 Default genome configuration E

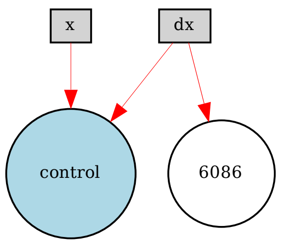
DEFAULT GENOME CONFIGURATION E	
Activation	tanh
Add conn. prob.	0.2
Remove conn. prob.	0.2
Add node prob.	0.2
Remove node prob.	0.2
Time constant τ	0.5

Configuration E is quantitatively outperformed by C and F, but has an interesting and quality that the two rivals do not. Though only one sample in the distribution oscillates rhythmically, this configuration consistently produces CTRNNs that clearly modulate the output when the input value changes. Like with earlier examples of this behaviour, it affects amplitude and not frequency, but it is interesting and surprising that specifically adjusting τ to 0.5 has a higher chance to ensure a relationship between input and output. The *why* behind this behaviour has not been an insight gained in the study, but it is an interesting phenomenon.

The type of network topology seen in figure 4.11 b) features somewhat regularly in this distribution. They are small and usually with a direct connection between input node and output node, but with very few other connections. The network in figure **??**, while also quite small, has no connection from the input node to speak of, but has otherwise more connectivity.

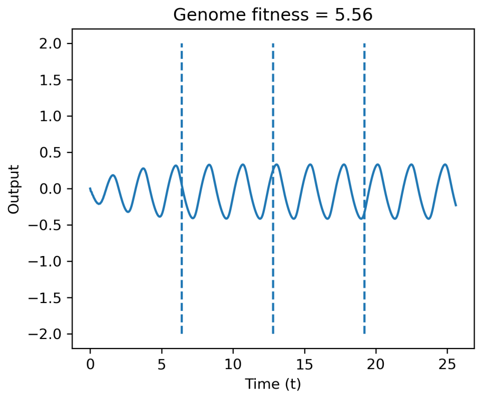


(a) Rank 2 output.

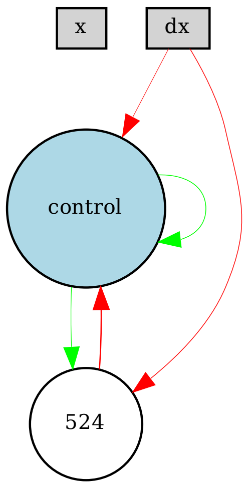


(b) CTRNN topology

Figure 4.11: CTRNN output and network topology of the qualitatively best winner. Evolved in run number 8



(a) Rank 3 output.



(b) CTRNN topology

Figure 4.12: CTRNN output and network topology of the qualitatively worst winner. Evolved in run number 3

Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
0	9	1	0	0

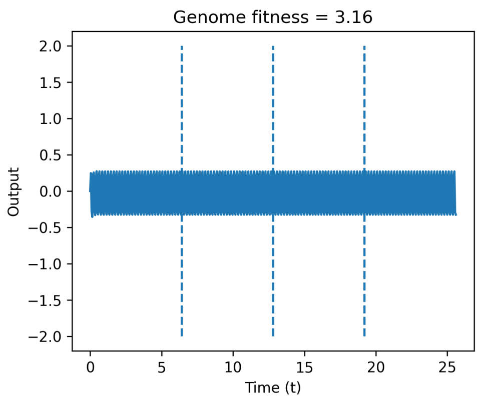
Table 4.9: Table of the number of winner genomes at each qualitative rank in the distribution of configuration E.

4.2.6 Default genome configuration F

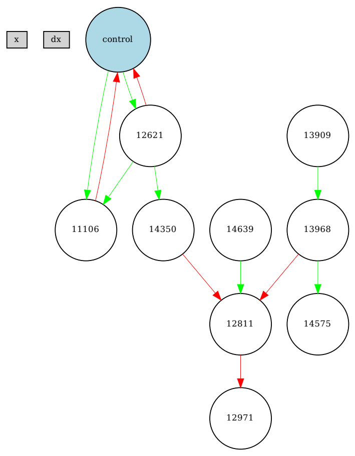
DEFAULT GENOME CONFIGURATION F	
Activation	tanh
Add conn. prob.	0.8
Remove conn. prob.	0.6
Add node prob.	0.8
Remove node prob.	0.6
Time constant τ	0.1

The configuration F has augmented rates of node and connection creation, resulting in a tendency to create larger networks quickly. Looking at the distribution performances in figure 4.2 however, it seems to produce genomes that perform similarly to configuration C. While configuration F is designed to grow large networks, it otherwise shares configuration values with C. This indicates that larger networks does not create an added advantage to solving the problem. A further similarity to C is they both reached the same top score of 3.16. Figure 4.13 b) shows a large network, but a higher interconnectivity appearing between the output node and its two neighbour nodes. Stripping away all the other nodes creates a topology more reminiscent of topologies found in C. This further indicates that the number of connections is more important to explain better fitness than than the number of nodes.

Qualitatively it shares the same characteristics as C and D, where the winners are focused around rank 3 type solutions. All winners are self sustaining oscillator solutions with no connection from the input node.

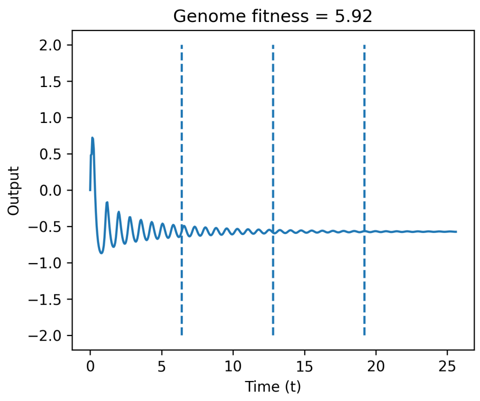


(a) Rank 3 output.

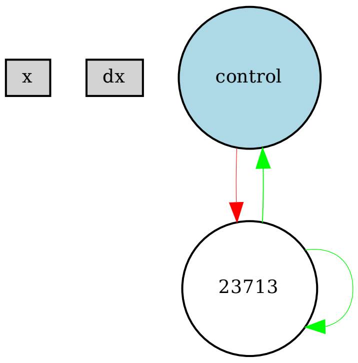


(b) CTRNN topology

Figure 4.13: CTRNN output and network topology of the qualitatively best winner. Evolved in run number 2



(a) Rank 3 output.



(b) CTRNN topology

Figure 4.14: CTRNN output and network topology of the qualitatively worst winner. Evolved in run number 1

Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
0	0	10	0	0

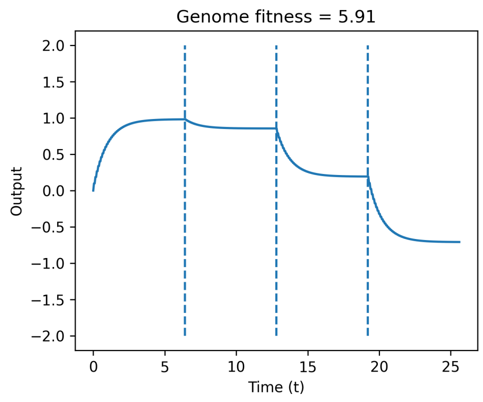
Table 4.10: Table of the number of winner genomes at each qualitative rank in the distribution of configuration F.

4.2.7 Default genome configuration G

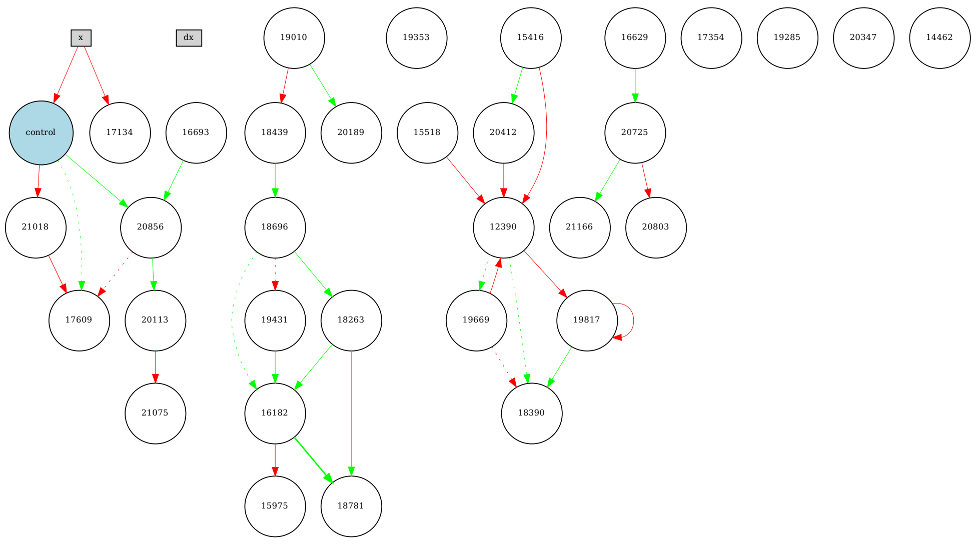
DEFAULT GENOME CONFIGURATION G	
Activation	tanh
Add conn. prob.	0.8
Remove conn. prob.	0.6
Add node prob.	0.8
Remove node prob.	0.6
Time constant τ	0.5

Like F, configuration G is made to have greater tendency towards large networks. Also like F, it has shows clear signs of being more similar, both qualitatively and quantitatively speaking, to configurations that share its value of τ and activation function; namely configuration E. The quantitative and qualitative similarity to E, a configuration that does not feature increased topology growth rate, indicates that the size of the network does not play an important role in explaining the fitness scores obtained. This is much in the same way that the similarities between F and C implies the same for F.

Though topologically vastly different in size, much like the networks in E, the winner networks in F seem to often feature a direct connection betwee input and output nodes.



(a) Rank 3 output.



(b) CTRNN topology

Figure 4.15: CTRNN output and network topology of the qualitatively best winner. Evolved in run number 6

A figure of the qualitatively worst winner has been omitted since they all have the same rank.

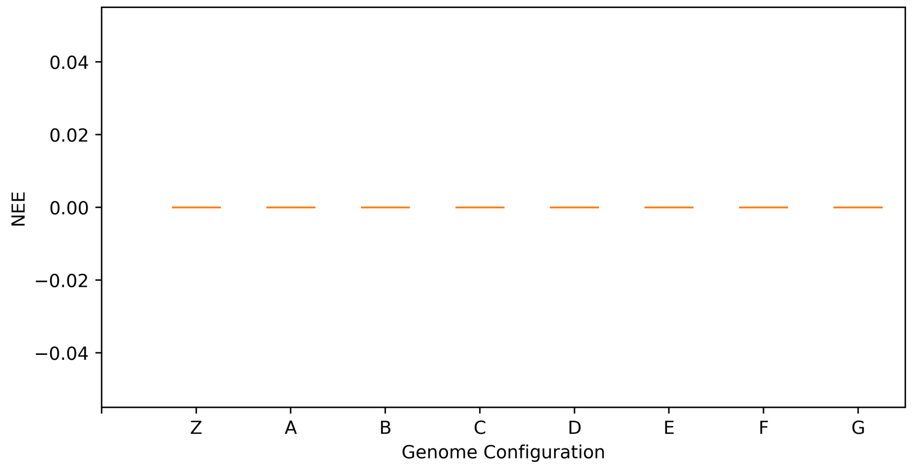


Figure 4.16: Distributions of winner genome fitness scores from each NEE based experimental paradigm

Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
0	10	0	0	0

Table 4.11: Table of the number of winner genomes at each qualitative rank in the distribution of configuration G.

4.3 Results from NEE based paradigms

As in section 4.2, the first step of the analysis is to consider the fitness score distribution of the ten winner genomes resulting from each default genome configuration. Though the boxplot in figure 4.16 is distinctly lacking any interesting variation, it is included to show why any further quantitative analysis is meaningless. All ten runs of all ten configurations failed to discover a single genome that received a fitness score not equal to 0. The orange lines identify the median value of the distribution but since all ten NEE fitness scores are all equal, the median defines the entire distribution.

As analysis of the genome configurations and their effect on the fitness is impossible, this section focuses its attention on the fitness function with suggestions as to what brought about these results. Figures showcasing the CTRNN outputs are left in the appendix.

Chapter 5

Conclusions

5.1 Chapter Introduction

This chapter concludes the study with the following findings:

- The RMSE fitness function is unsuited for the problem
- The NEE fitness function is not suitable in its current form, but may possibly be better suited with a few proposed changes.
- tanh activation function outperforms the sigmoid for creating oscillating responses in CTRNNs
- The selection of time constant τ has a significant impact on obtainable fitness
- Large network size is insignificant, but a high amount of connections between nodes may have a positive effect

The evidence and/or motivation for these conclusions will be discussed in the immediately following section. The end of the chapter will be dedicated to a discussion of how resolve the issues discovered through future work.

5.2 Discussion

First point of order is to discuss the results relating to the two fitness functions, as they were the main focus. The reader is reminded of the research questions posed in the introduction:

1. Is there a meaningful difference in likelihood to produce high fitness CTRNN solutions from one fitness function to the other?
2. Is there an identifiable feature that fitness functions should possess in order to motivate selection of CPG-like CTRNNs?

Neither of the proposed fitness functions show great weaknesses, and ultimately do not seem well suited to produce CPG-like CTRNNs. However, with regards to the first question, one could say the answer is yes.

In looking solely at the distribution of fitness scores, it is clearly a trivial answer when the NEE fitness function never rewarded a single genome with a fitness different from 0. At the same time, the findings from the NEE experiments were inconclusive, so the comparison between them is not truly fair. Ultimately, the conclusion must be that the thesis fails to find a satisfactory answer to question 1.

The second is answered with something of a reversal. Rather than finding a feature that should be present, the thesis finds features the functions should *not* have in order to be successful. This is properly argued in the next two subsections.

Second to considering the fitness functions, the thesis dedicates a portion to discuss the genome configuration variables used to investigate the fitness functions, though the insight into these primarily say something about evolving CTRNNs, rather than CPGs. Though all the NEE based experiments failed, the results from the RMSE based experiments provides interesting insight into what genome configuration promoted aspects of CPG-like behaviour. Firstly, every winner fitness distribution from the genome configurations showed statistically significant indication that they were not distributions from the same population as the baseline configuration (see table 4.3). This means that the variable changes that make up the configurations are very likely to be impactful. From the results presented in chapter 4, it can be concluded that the choice of value for time constant τ and activation function had significant impact on the achieved fitness distributions and qualitative rank, while network growth rate and size did not.

The end of the discussion covers each genome configuration variables and the findings related to these.

5.2.1 RMSE fitness function and fitness space exploration

Investigating the results from the RMSE based experiments offers some clues toward understanding the RMSE's suitability for this problem. Of particualar note is the tendency of the CTRNNs that output rhythmic oscillation to entrain to a single frequency, and not have any sort of relationship between the input control variable c and the output, be it frequency or otherwise. From a high level perspective it seems strange; the fitness function incorporates a range of 10 different target frequencies to hit, yet across all RMSE based experiments the oscillating solutions always maintain one single frequency. Not one shows any tendency towards frequency modulation between for example just two frequencies. This single frequency entrainment is ultimately a pervasive pattern, and so it warrants scrutiny. Across all the eight RMSE based paradigms, there are ten winner CTRNNs, for a total of 80 winners. Of these 80, 28 have been qualitatively categorized (Rank 3) as rhythmically oscillating, all with the aforementioned single frequency entrainment.

Knowing this, it should be enlightening to attempt a visualisation of the RMSE fitness space. The reader is reminded of the RMSE definition in this study:

$$RMSE = \sqrt{\frac{\sum^{i} |f_i - f_{c_i}}{N}} \quad i = 1, 2, 3, \ldots, N$$

where N is the total number of simulations, and therefore equal to the number of cs, f_i is the measured output frequency from simulation i and f_{c_i} is the target frequency derived from the ith c. Figure 5.1 visualises the square error term of the RMSE for all input values $c \in [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]$, which is the sample of control inputs used during simulations of a CTRNN. Each error curve has a corresponding dashed line indicating the target frequency f_{c_i} For example the dashed line at $F = 1Hz$ (remember that $f_{t_c} = \frac{2c}{20Hz}$,

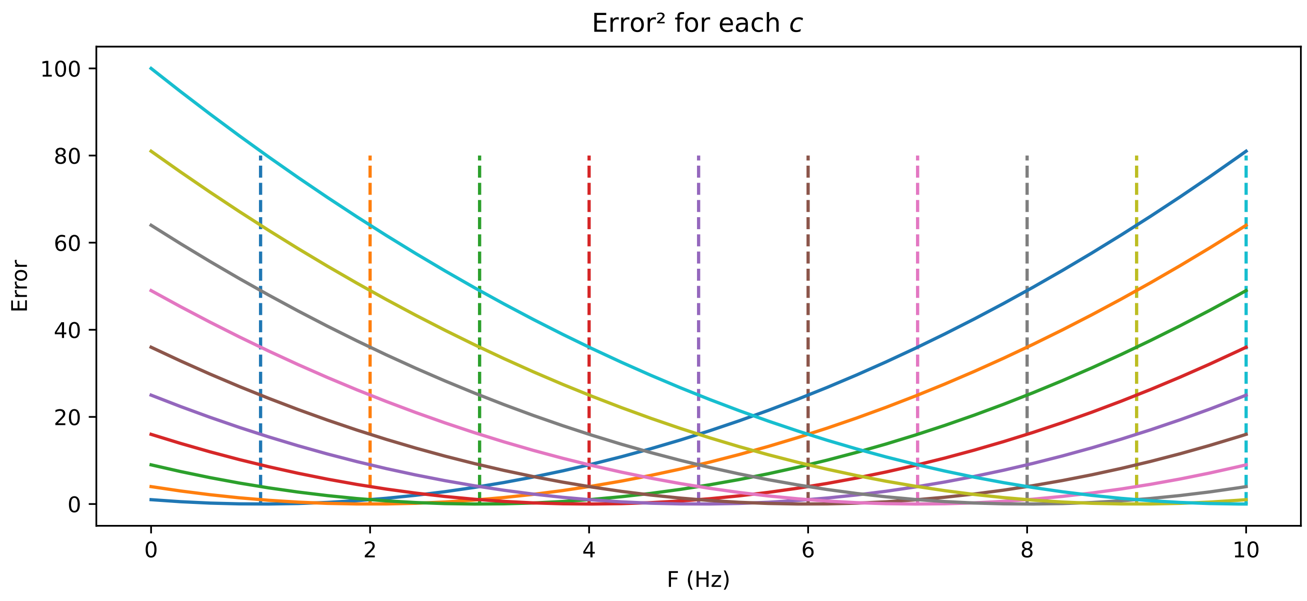


Figure 5.1: Curves showing squared errors for each target frequency derived from the value of input control variable c. Each dashed line shows the target frequency with its color corresponding to an error curve.

3.3.1) indicates the $1Hz$ target frequency. The error curve of the same colour shows the error getting exponentially larger the further away one moves from this target. In other words, the dashed lines show the point where each curve tangentially reaches $Error = 0$, i.e a perfect hit.

However, when we include the mean term, computing the mean between each curve, these target frequency tangents are smoothed over. This transforms the fitness landscape to a space that only has one single optimum at one single point on the frequency axis at $F \approx 5.5Hz$. Figure 5.2 visualises the issue. As one might expect, the square root term does not improve things.

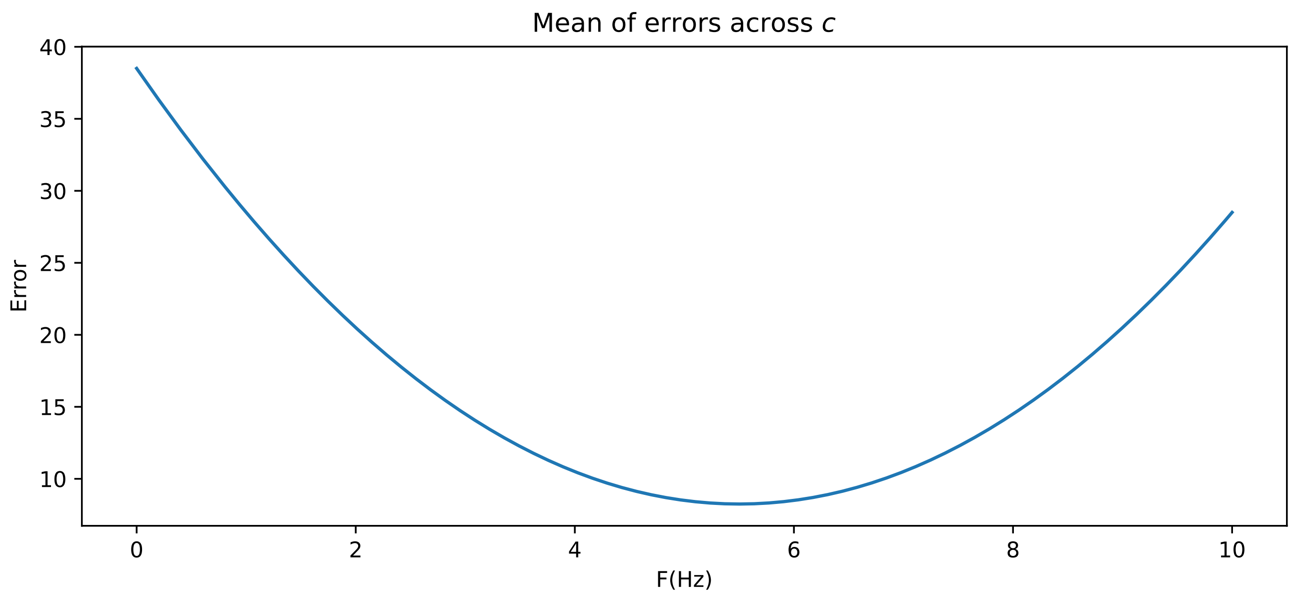


Figure 5.2: All squared error curves averaged.

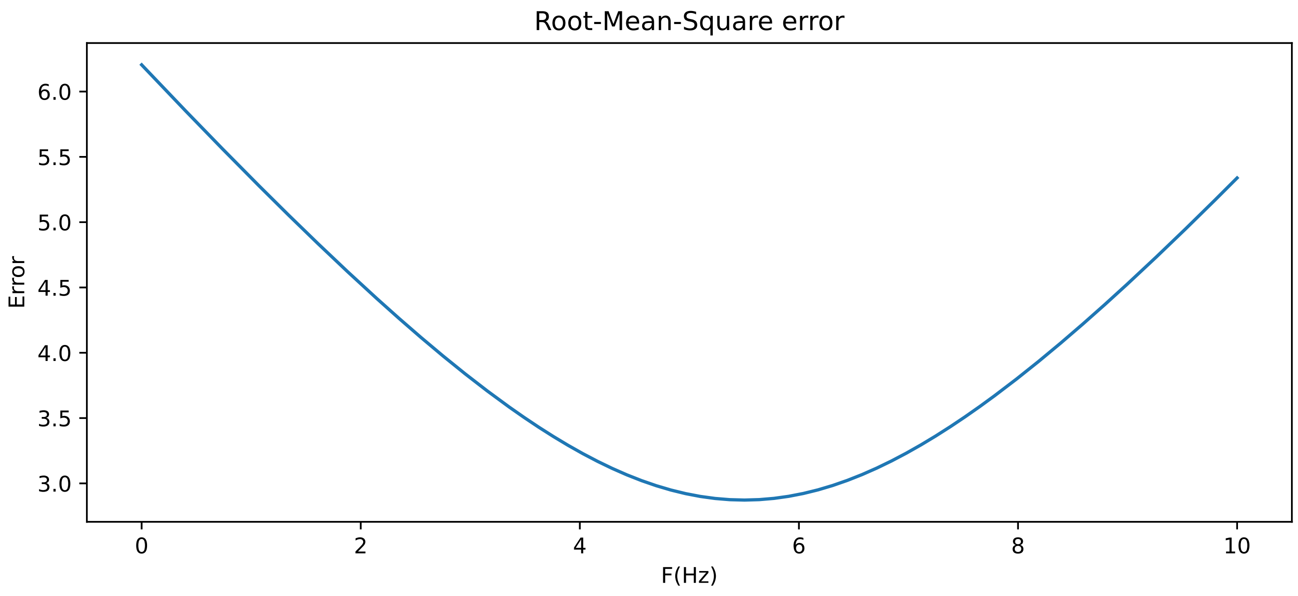


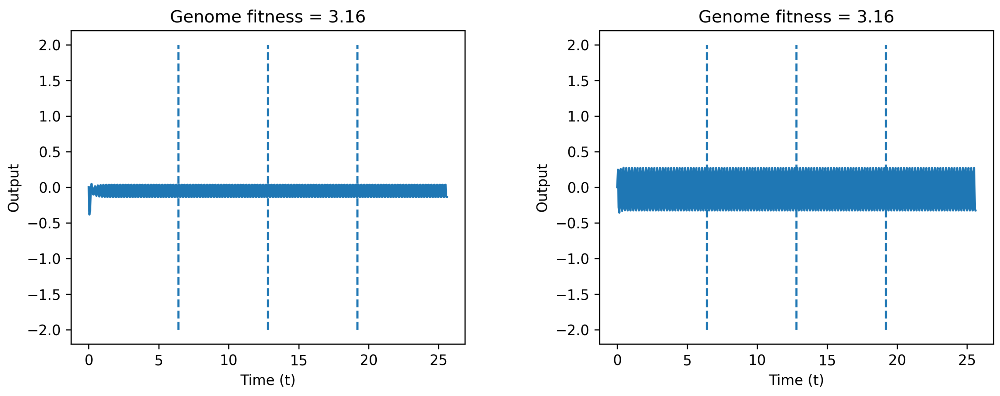
Figure 5.3: RMSE fitness space defining the optimal solution as a constant oscillation at $\sim 5.5Hz$ with RMSE score ~ 2.9.

Adding the square root term actually sharpens the "valley", creating an even stronger motivation towards the $5.5Hz$ minimum (figure 5.3). Making note of the scale of the errors in the RMSE fitness space, the fitness scores of the top two winners across all RMSE based paradigms can be reviewed in light of the shape of RMSE fitness space.

Relative to the total time they are visualised over, the frequencies in figure 5.4 are quite high, making it understandably hard for the reader to interpret. However, the fitness scores should be noted as being quite close to the minimal value of the RMSE space, shown in figure 5.3 to be ~ 2.9. Counting the oscillations reveals that they both oscillate with a frequency of $\sim 5Hz$. These values correspond very well to the fitness space shape visualised in figure 5.3.

The next thing to explain then, is the relatively low number of winner genomes that approaches this optimum of ~ 2.9. Indeed, the most common error score across all eight distributions is a fitness score of ~ 5.9. The number of winners with this exact fitness is driven sharply up by the distributions from configuration B, E and G, the boxplot in figure 4.2 essentially showing these distributions containing exclusively the ~ 5.9 fitness value or values very close to it. Observe that these genome configurations all have a time constant $\tau = 0.5$, and that the higher this value is, the more sluggish the neuron response is. In other words, a higher τ the less able it is to express higher frequencies. Referencing again figure 5.3, it can be seen that an RMSE value of ~ 5.9 roughly corresponds to $1Hz$. The genomes from distributions B, E and G are simply unable to oscillate fast enough to approach higher frequencies, thus only being able to obtain the fitness associated with the lowest frequency target.

The conclusion is that the RMSE fitness function is unsuited and simply cannot produce CPG-like CTRNNs in the set up used for this thesis due to the averaging over all the errors obtained in one simulation. This averaging works against the very essence of what the task is, which is to target specific frequency values, not find a compromise between them all. A consequence of the optimum being one single frequency, is the lack of evolutionary incentive to maintain a connection from the input nodes; hitting the target frequencies does not award good fitness values, so genomes that ignore the input do better. This is in direct conflict with the desired function. Therefore the thesis argues that the RMSE function is not suited.



(a) Top winner genome from configuration C using RMSE fitness function. **(b)** Top winner genome from configuration F using RMSE fitness function.

Figure 5.4: The two best fit winners across all 8 RMSE based paradigms.

5.2.2 NEE fitness function and fitness space exploration

To begin with the reader is first reminded of the fitness function from chapter 3.3

$$NEE = \prod^{i} F_{c_i}$$

which is the product of a CTRNN's scores produced with a range of different control inputs c_i, where each score F_{c_i} is calculated with the piecewise exponential function

$$F_{c_i} = \begin{cases} 10 & E_{c_i} < 0.1 \\ \frac{1}{E_{c_i}} & E_{c_i} \geq 0.1 \end{cases}$$

where E_{c_i} is the error between the ith CTRNN output frequency, and the target frequency derived from the ith sample of input control variables c. Each CTRNN is simulated with the same 10 input control values, hence i iterates over the range $[1, 10]$, and consequently $NEE \to \mathbb{R}^{10}$. This 10-dimensional function space (the domain of NEE is the set of scoring functions F) is difficult to explore, but looking closer at the score functions elucidates deeper challenges.

All experiments using the NEE fitness function yielded results that show no ability to search for any kind of optima regardless of genome configuration, as seen in figure 4.16. The consequence is that it is not possible to use the fitness results to implicitly discern anything in particular about the fitness space under NEE.

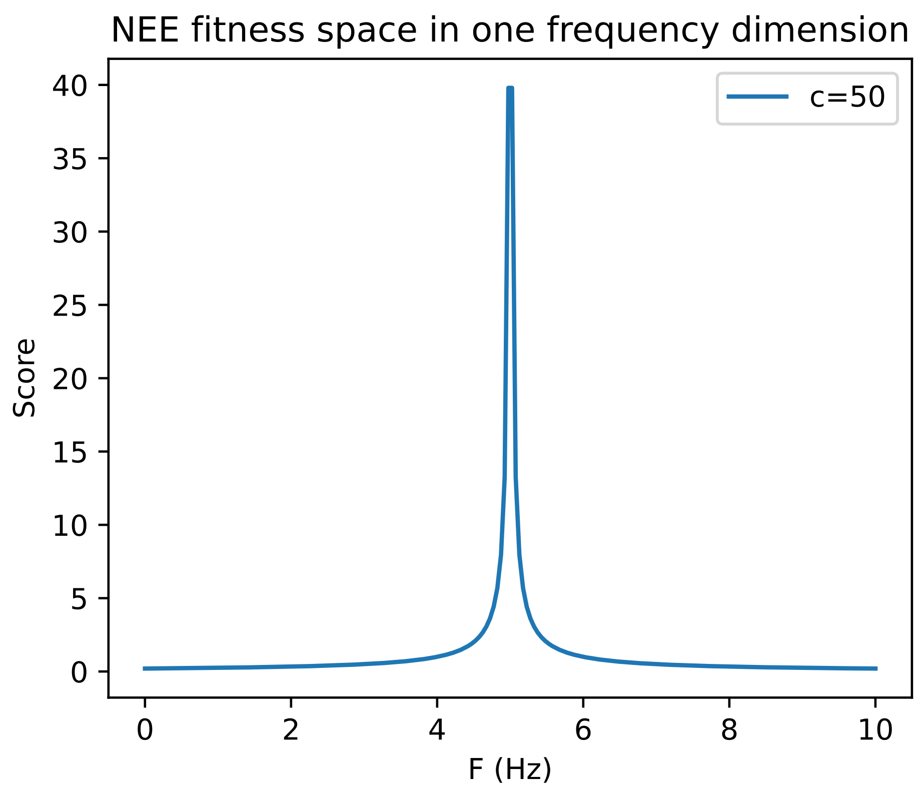


Figure 5.5: Fitness space of NEE when focusing on a single dimension of frequency where the target frequency is $5Hz$.

Figure 5.5 shows the distribution of obtainable points given $c = 50$. As the error $E_c = |f - f_t|$ increases, where $f_t = \frac{2c}{20Hz}$ (see 3.3.1) is the target frequency, the score approaches 0 asymptotically. While it creates a space that yields high rewards for near correct estimates of output frequencies, the space also mostly contains scores $0 < NEE < 1$. Consider a randomly initialised population of genomes. Some fraction of this population may represent genomes for oscillating CTRNNs, and thus by random chance scores points for matching one of the target frequencies. However, if it matches one target, it is very likely to miss all the other targets, scoring nine different scores between 0 and 1. The function definition says to multiply all these scores together, thus scaling the total score down by a factor of 10^{-9}, due to the nine misses.

This shows why all NEE based experiments exclusively obtained a score of 0. The only way to see scores > 0, is if there exists at least one genome in the initial population that by chance already partly solves the problem for enough frequencies that the points from the hits are not drowned out by the misses. This is however highly unlikely to happen. In other words, success with the NEE fitness function is highly dependant on random initialisation, possibly so much so that it reduces the NEAT algorithm to a random search. Possible solutions to this are further discussed in section 5.3.

Still the study cannot conclude whether the NEE function is unsuited or not. Because all of the NEE based experimental paradigms failed, it leaves little grounds to explain fitness function features through genome configuration fitness scores. On one hand, a distribution of 80 winner genomes all scoring 0 points is quite substantial evidence, but since all the different genome configurations had no impact on performance, there is no way to infer information about the NEE fitness space from how genomes respond to it.

5.2.3 Activation Function

The results from the RMSE based experiment showed clear tendencies with regards to the choice of activation function. Genome configurations with the tanh activation function were more likely to produce CTRNNs with rhythmically oscillating output than those with the sigmoid activation. This tendency faded however in the paradigms where the time constant τ was set to 0.5. Disregarding this, comparing the winner genomes from the RMSE based paradigms that had $\tau = 0.1$ and $\tau = 0.25$, the improvement due to the tanh activation is still convincing. Observe the winner fitness distributions from configurations Z and A, and compare them to the distributions from configurations C and D. Z and A had sigmoid activations, while C and D had tanh. While Z and C both had $\tau = 0.1$, Z sports two rhythmically oscillating winner outputs, while C has nine. Similarly, A and D both had $\tau = 0.25$, but A has no rhythmically oscillating winner outputs, while D has eight.

Combining this observation with the fact that the top two winner genomes both came from configurations with tanh activation and $\tau = 0.1$, this suggests that the tanh function is significantly better for solving the problem given that the time constant is low enough allow a suitable rate of change.

5.2.4 Time Constant

In combination with the activation function, the time constant τ was shown to be impactful. An expected result was the effect it would have on frequency; it stands to reason that more responsive neurons are able to express higher frequency outputs.

The major obstacle for low τ configurations seemed to be maintaining, or even rediscovering, a connection to the input neuron. Without this connection, it is obvious that there can be no input driven control of the output. However, as discussed in 5.2.1, the main reason for this disconnect is likely to be the RMSE fitness function.

The initially surprising result was that experiments where $\tau = 0.5$ had a much higher tendency to maintain a connection between the output and the input neurons. From the same discussion on RMSE fitness exploration, it seems plausible that the value of τ indirectly affects the exploration by setting restrictions on rate of change, and by extension the maximum frequency of oscillation.

5.2.5 Network size and growth rate

The genomes that were part of network growth experiments showed little variation, neither quantitatively nor qualitatively, from the genomes in experiments that had baseline growth rates and otherwise had the same configuration of genome variables. Inspecting the network graphs in distributions from genome configurations that heavily featured oscillators (configurations C, D and F), reveals a possible pattern relating a high number of connections to oscillating output.

5.3 Future Work

First and foremost, future work must be focused on continuing the search for a fitness function that is well suited. RMSE should be disregarded, but since findings pertaining to NEE were inconclusive, there are changes that could be interesting. To solve the issue of the fitness space being very sparse, it could be worthwhile to test if replacing the $|f_i - f_{c_i}|^-1$ term with a sinc(x) term $\frac{sin(|f_i - f_{c_i}|)}{|f_i - f_{c_i}|}$ will make the space easier to explore. The new sinc(x)-based fitness function would then be

$$S = \prod^{i} \frac{sin(|f_i - f_{c_i}|)}{|f_i - f_{c_i}|}$$

Figure 5.6 shows an example where the target frequency is $5Hz$. Firstly, this function is defined over the entire domain. The consequence is, among other things, that the optimum is exactly at the target frequency. The NEE is not defined at the target frequency, and so had to be implemented with a tolerance band around the desired target. Secondly, this redefinition deals with the large areas of close-to-zero values that the NEE suffered from. The sinc(x) distributes lesser local maxima across the frequency range that become increasingly better the closer to the target frequency. Thirdly, the sinc(x) term makes the highest possible score for S a maximum of 1, with the possibility of easily rescaling it by adding a scaling factor $\alpha > 0$ to the sinc(x) term $\alpha\frac{sin(|f_i - f_{c_i}|)}{|f_i - f_{c_i}|}$. Figures 5.7 and 5.8 show two other simple transformations that give further control over the shape of the fitness space.

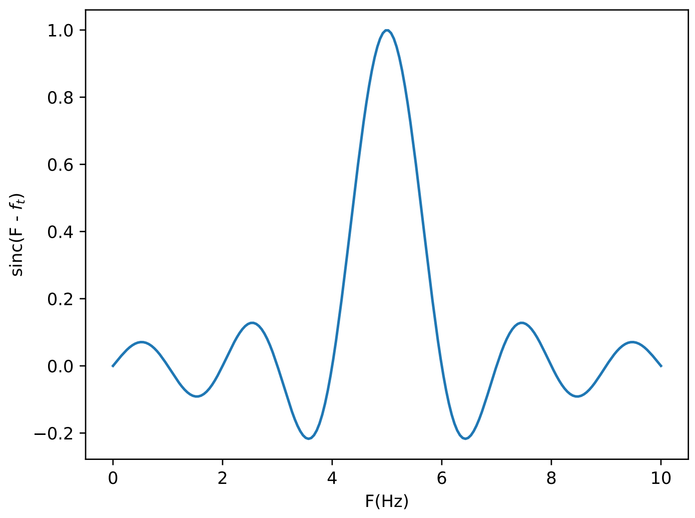


Figure 5.6: $\frac{sin(|f_i - f_{c_i}|)}{|f_i - f_{c_i}|}$ with target frequency $f_{c_i} = 5Hz$. The small local maxima across the frequency range may improve exploration.

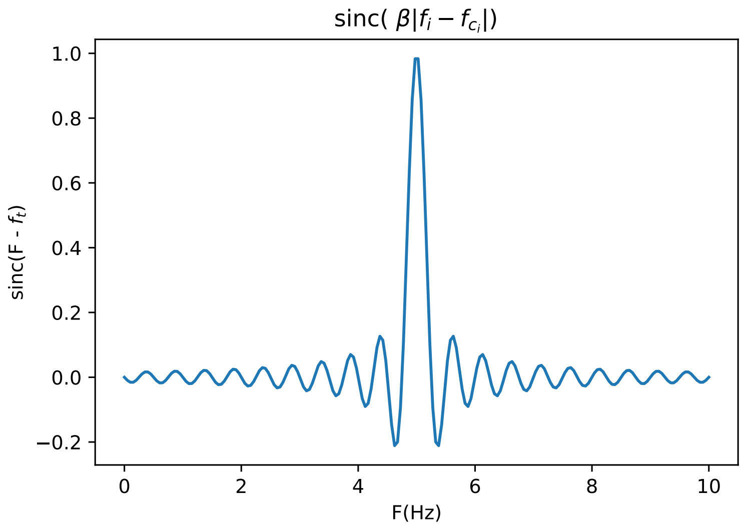


Figure 5.7: Introducing a scaling factor to the difference term gives control over the number of local maxima and width of the bell around the target frequency. Here $\beta = 4$.

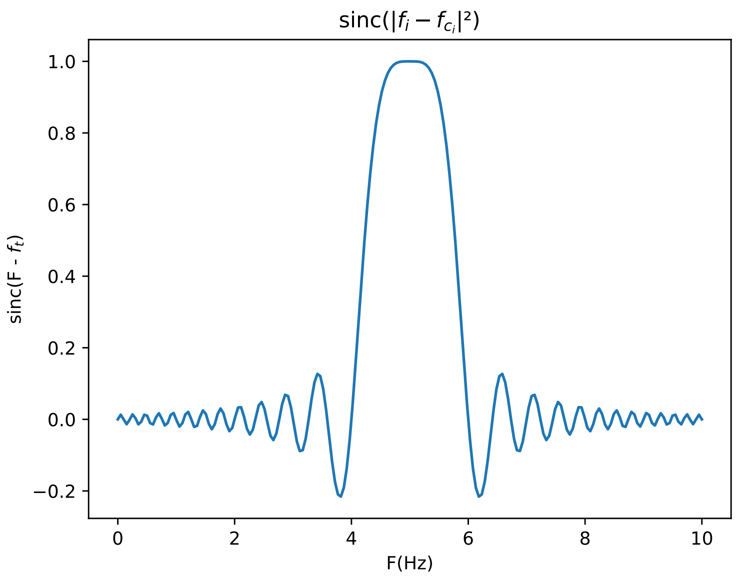


Figure 5.8: Squaring the difference gives yet another profile.

Another suggestion is to try a multi-objective optimisation approach, where each target frequency is considered an objective. A review paper [4] presents several multi-objective approaches, all of which are especially directed at evolutionary computing.

The effect of a higher degree of network interconnectivity is not methodically explored in this thesis, but superficial observations indicate it may be another significantly impactful variable such as the time constant and activation function. The experimental set up in this these could be easily altered to include such an investigation. This could be done by another genome configuration where the configuration variable "add connection probability" is set to a higher value than "add node probability". This will cause evolution to create connections more frequently. Presumably the consequence will be that the high number of connections must be distributed over the lesser number of nodes.

These future endeavours should lead to better insight into what fitness functions are suited for CPG-like CTRNNs, enabling possibilities for improvements in robotic locomotion.

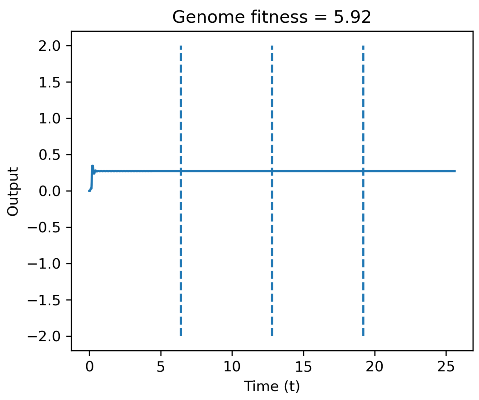
Appendix A

Appendix

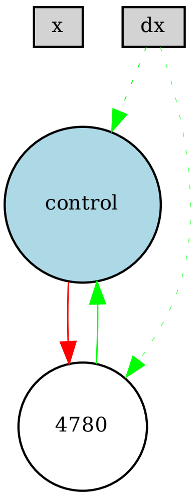
A.1 Appendix I - Winner Genome Visualisations

Appendix I contains visualisations of CTRNN outputs and their network topologies. It is divided into subsections which each contain genome visualisations from one paradigm, starting with RMSE based paradigms and ending with NEE.

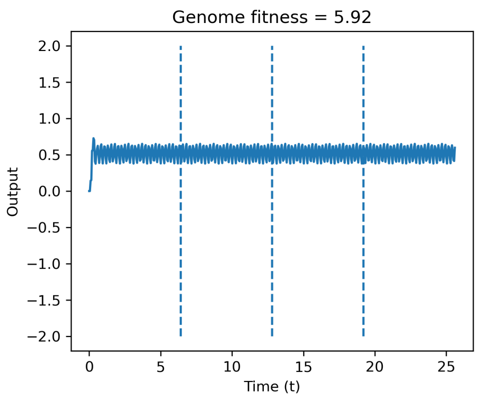
A.1.1 Default Genome Configuration Z - RMSE



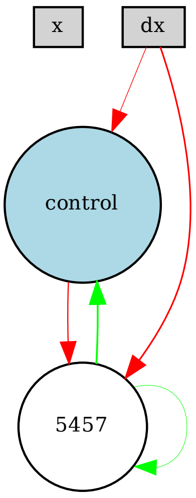
(a) CTRNN output over 25.6s from winner 1. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



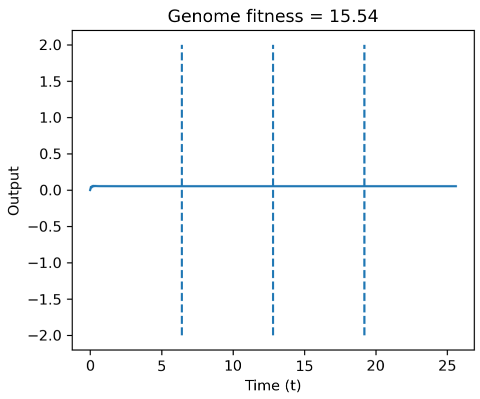
(b) Network topology of winner 1.



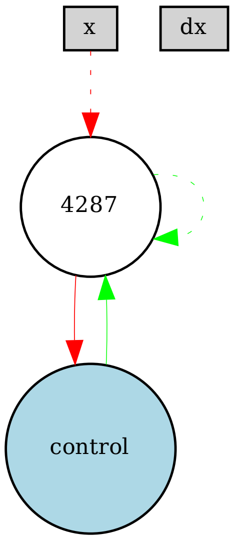
(a) CTRNN output over 25.6s from winner 2. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



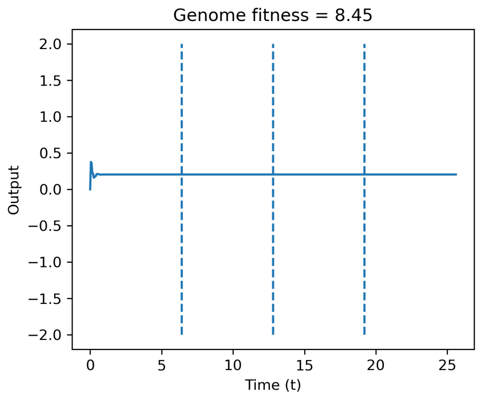
(b) Network topology of winner 2.



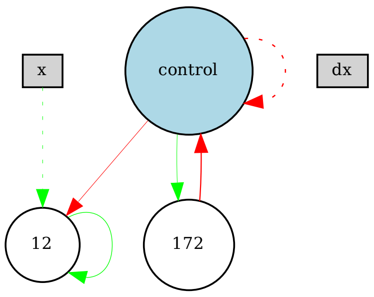
(a) CTRNN output over 25.6s from winner 3. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



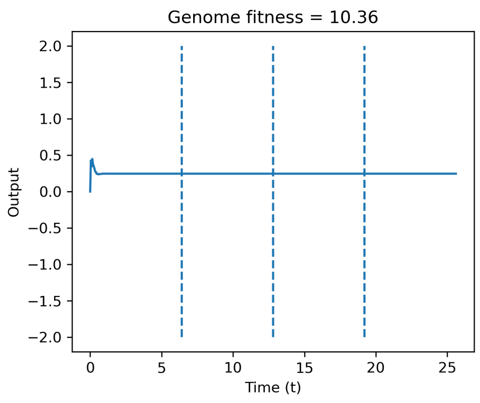
(b) Network topology of winner 3.



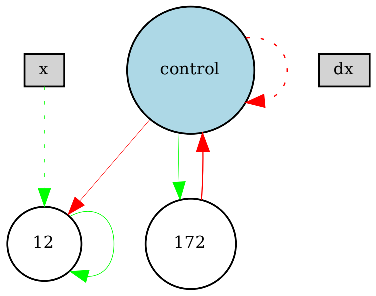
(a) CTRNN output over 25.6s from winner 4. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



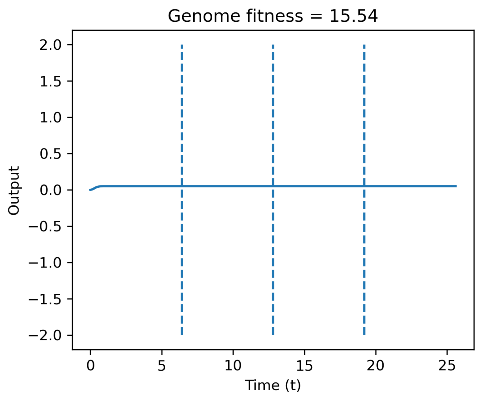
(b) Network topology of winner 4.



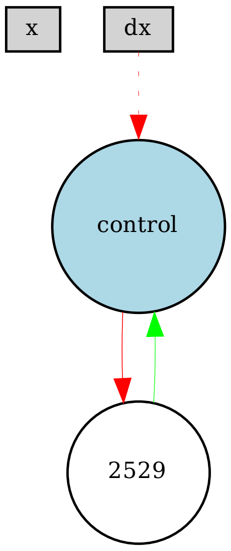
(a) CTRNN output over 25.6s from winner 5. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



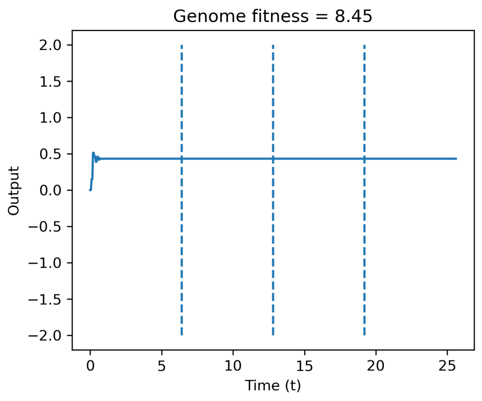
(b) Network topology of winner 5.



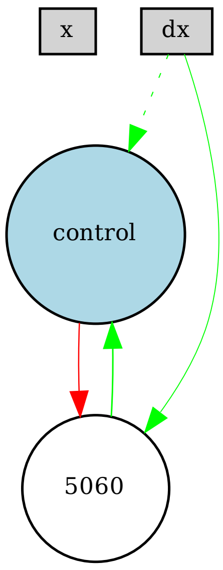
(a) CTRNN output over 25.6s from winner 6. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



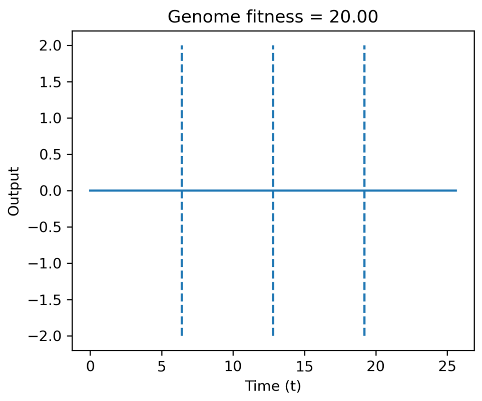
(b) Network topology of winner 6.



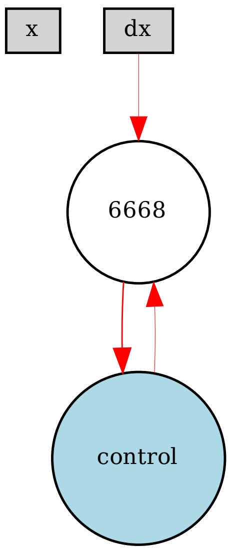
(a) CTRNN output over 25.6s from winner 7. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



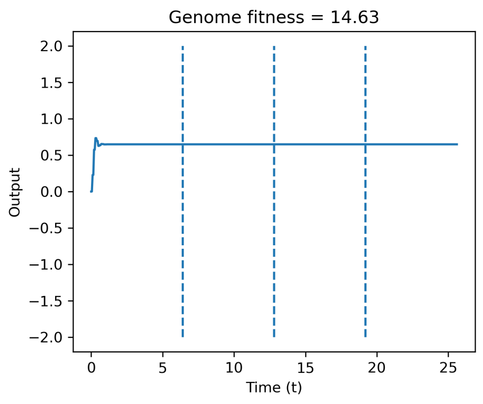
(b) Network topology of winner 7.



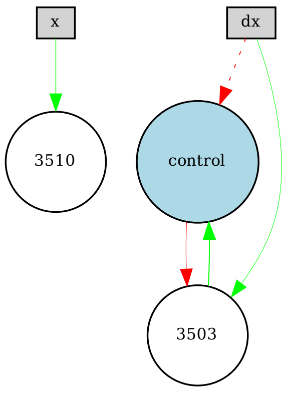
(a) CTRNN output over 25.6s from winner 8. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



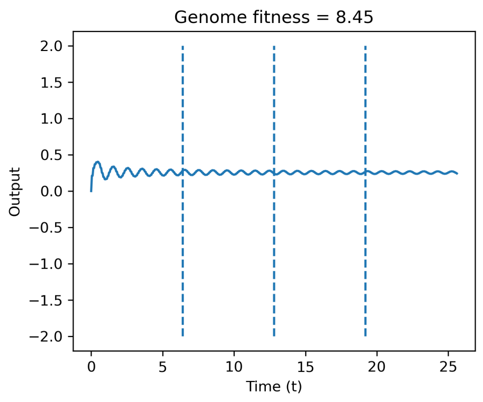
(b) Network topology of winner 8.



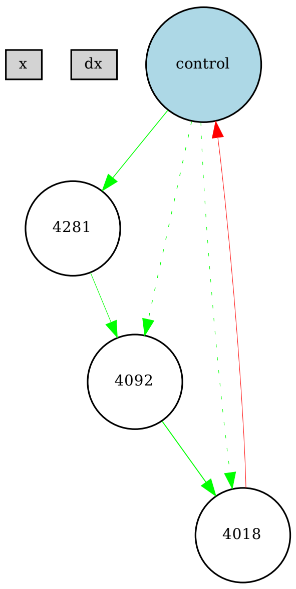
(a) CTRNN output over 25.6s from winner 9. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 9.

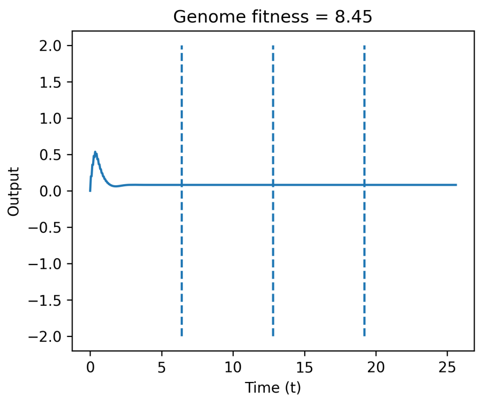


(a) CTRNN output over 25.6s from winner 10. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.

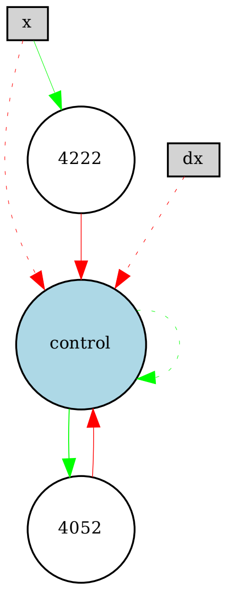


(b) Network topology of winner 10.

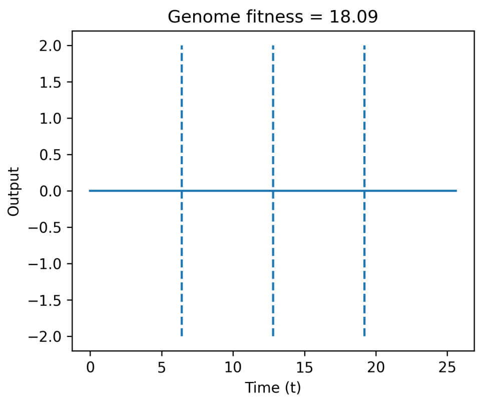
A.1.2 Default Genome Configuration A - RMSE



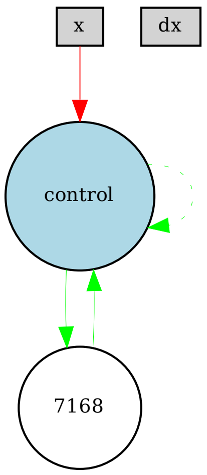
(a) CTRNN output over 25.6s from winner 1. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



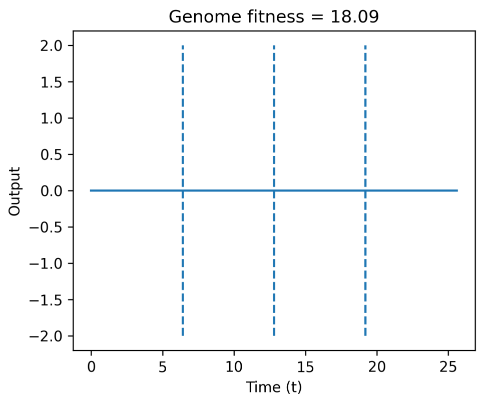
(b) Network topology of winner 1.



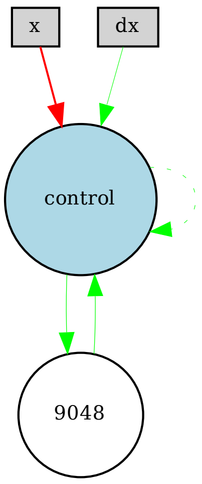
(a) CTRNN output over 25.6s from winner 2. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



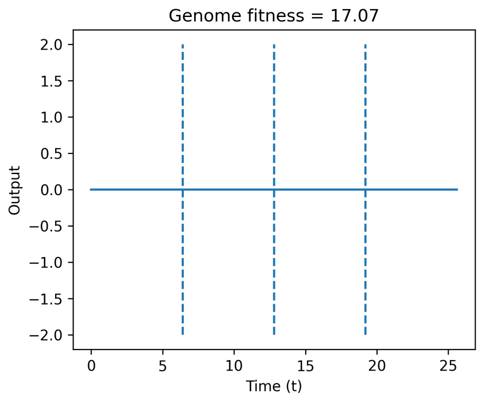
(b) Network topology of winner 2.



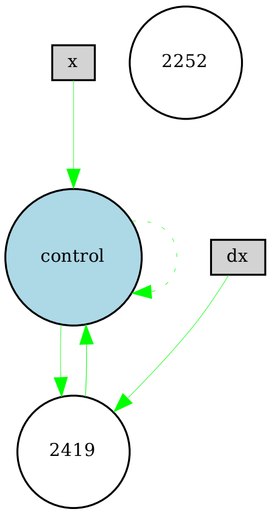
(a) CTRNN output over 25.6s from winner 3. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



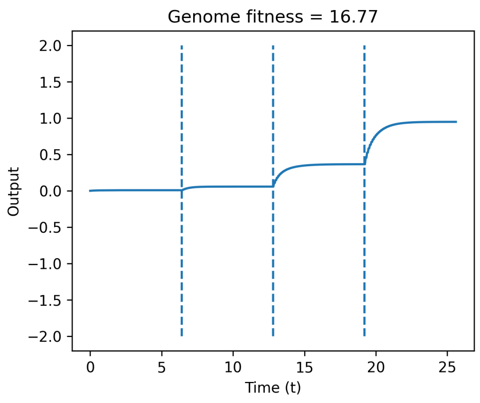
(b) Network topology of winner 3.



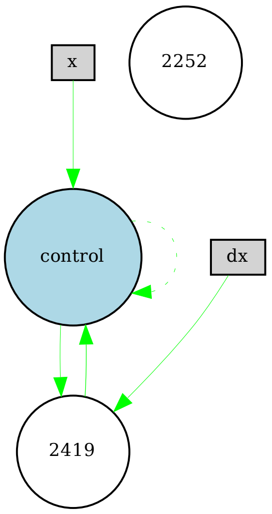
(a) CTRNN output over 25.6s from winner 4. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



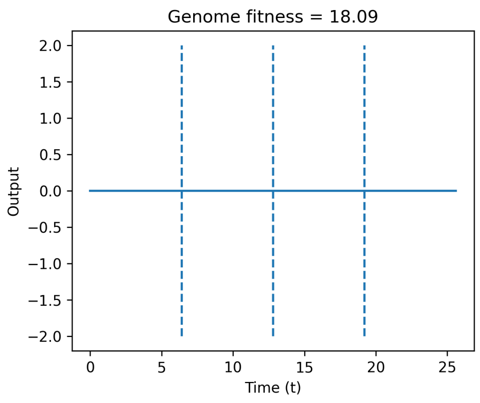
(b) Network topology of winner 4.



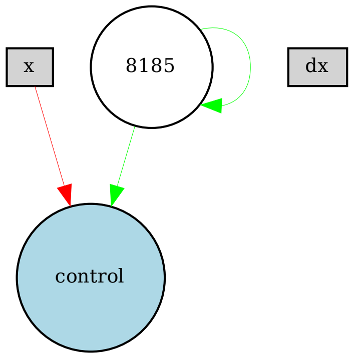
(a) CTRNN output over 25.6s from winner 5. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



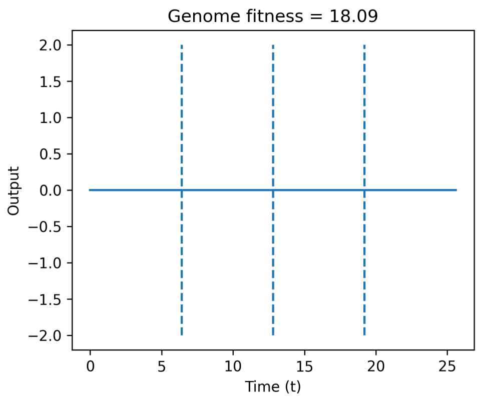
(b) Network topology of winner 5.



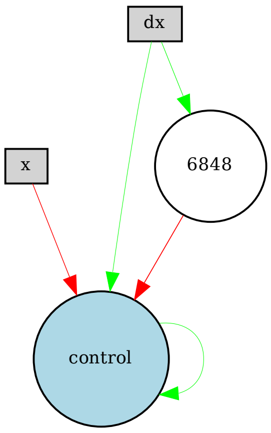
(a) CTRNN output over 25.6s from winner 6. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



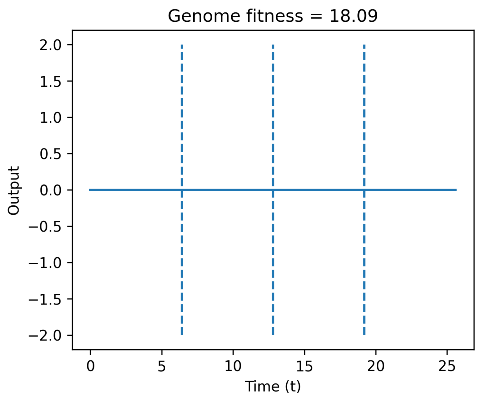
(b) Network topology of winner 6.



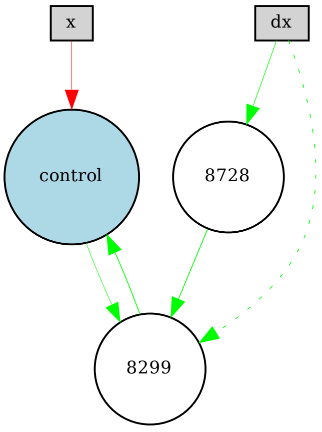
(a) CTRNN output over 25.6s from winner 7. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



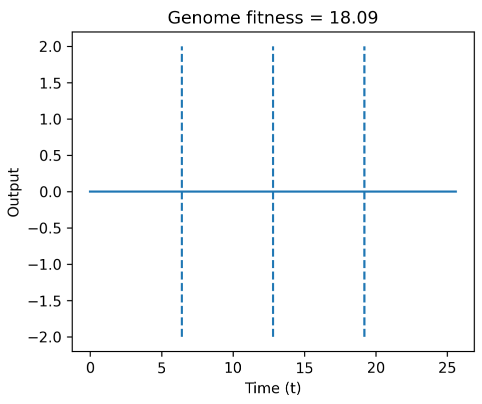
(b) Network topology of winner 7.



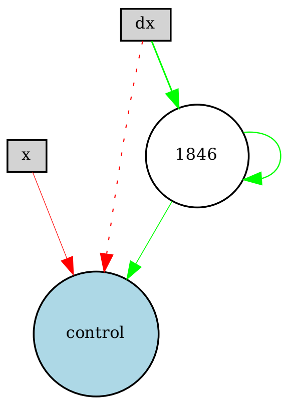
(a) CTRNN output over 25.6s from winner 8. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



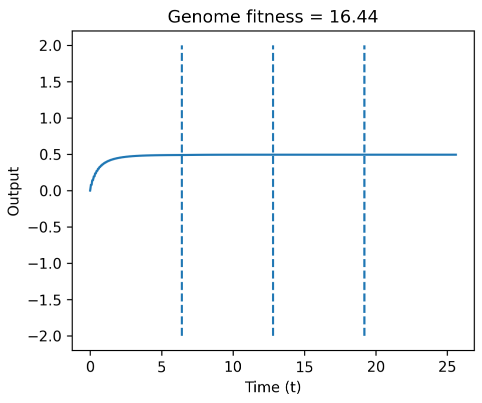
(b) Network topology of winner 8.



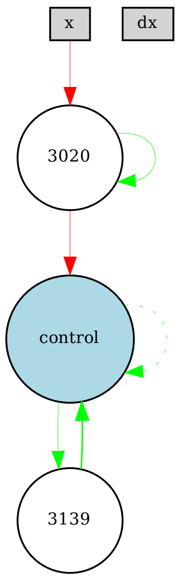
(a) CTRNN output over 25.6s from winner 9. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 9.

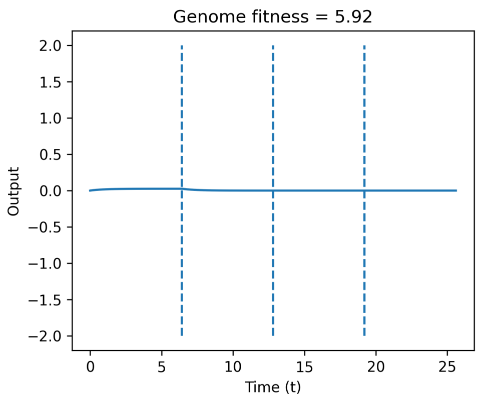


(a) CTRNN output over 25.6s from winner 10. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.

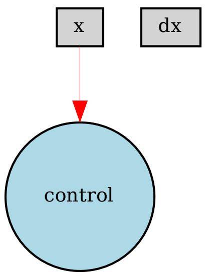


(b) Network topology of winner 10.

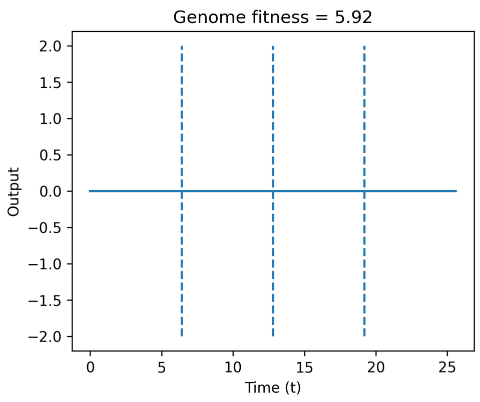
A.1.3 Default Genome Configuration B - RMSE



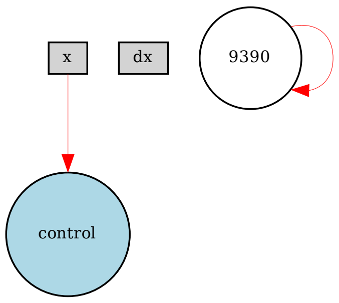
(a) CTRNN output over 25.6s from winner 1. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



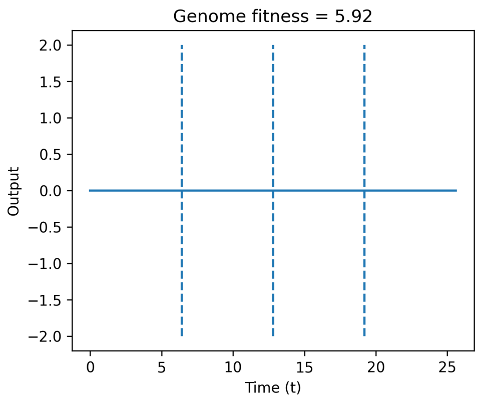
(b) Network topology of winner 1.



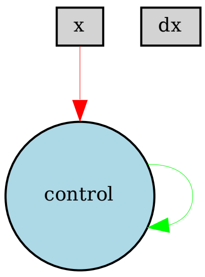
(a) CTRNN output over 25.6s from winner 2. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



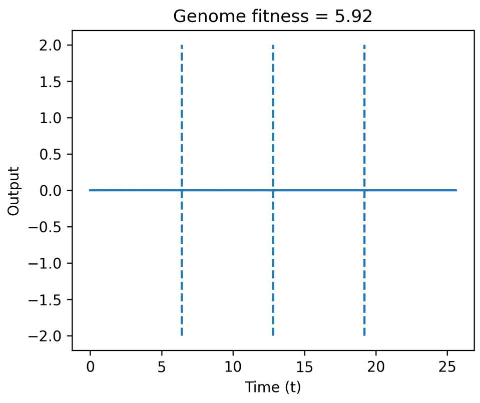
(b) Network topology of winner 2.



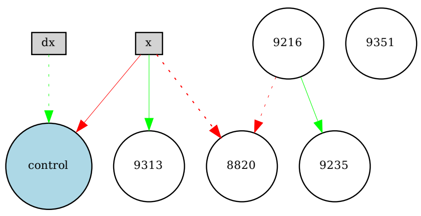
(a) CTRNN output over 25.6s from winner 3. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



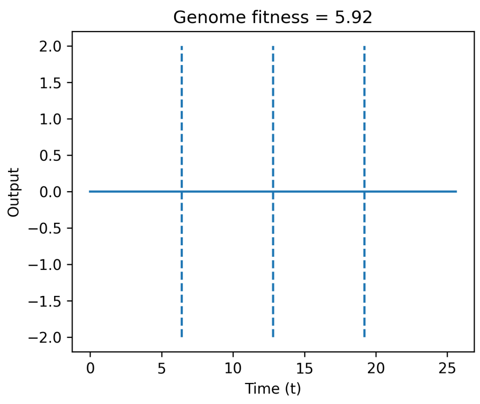
(b) Network topology of winner 3.



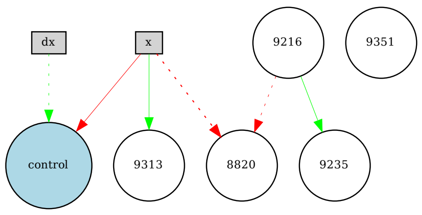
(a) CTRNN output over 25.6s from winner 4. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



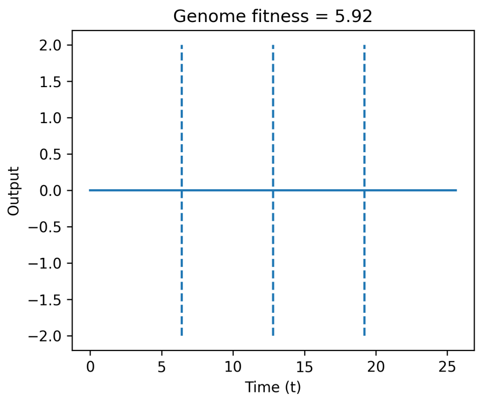
(b) Network topology of winner 4.



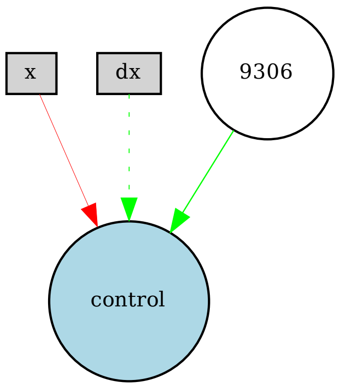
(a) CTRNN output over 25.6s from winner 5. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



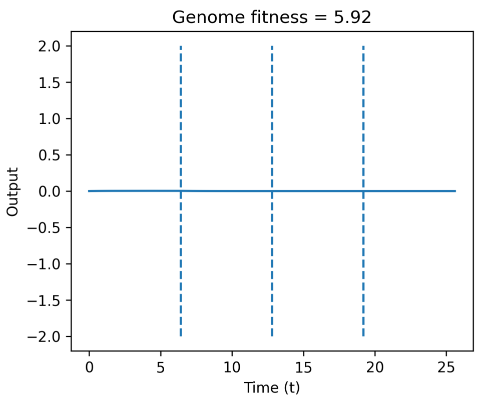
(b) Network topology of winner 5.



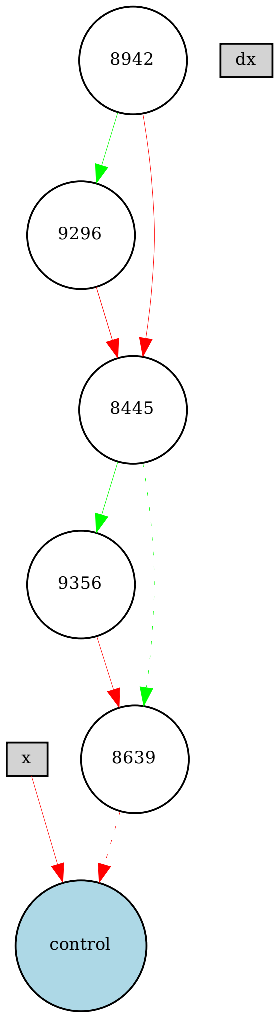
(a) CTRNN output over 25.6s from winner 6. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



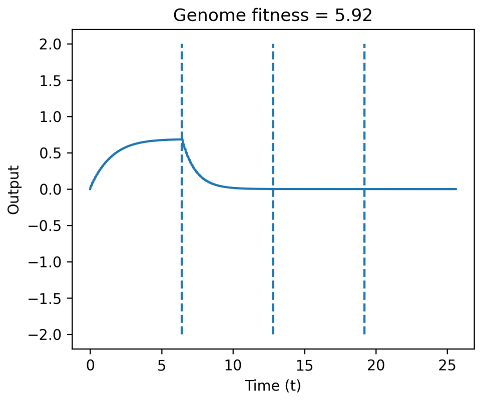
(b) Network topology of winner 6.



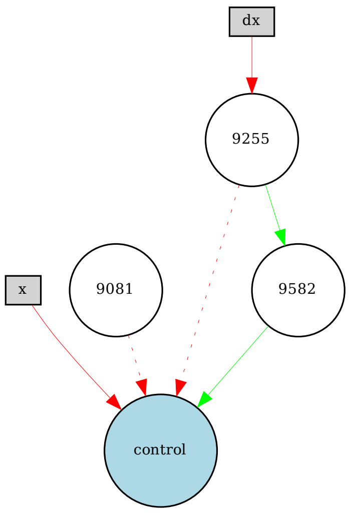
(a) CTRNN output over 25.6s from winner 7. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



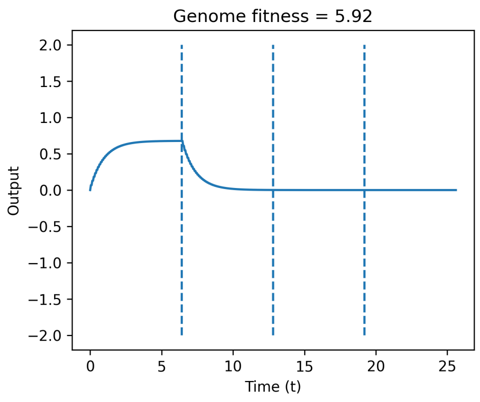
(b) Network topology of winner 7.



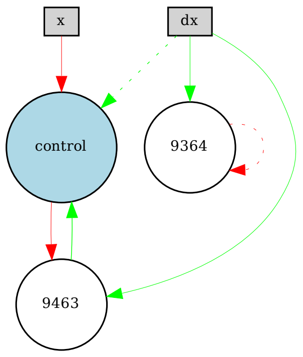
(a) CTRNN output over 25.6s from winner 8. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



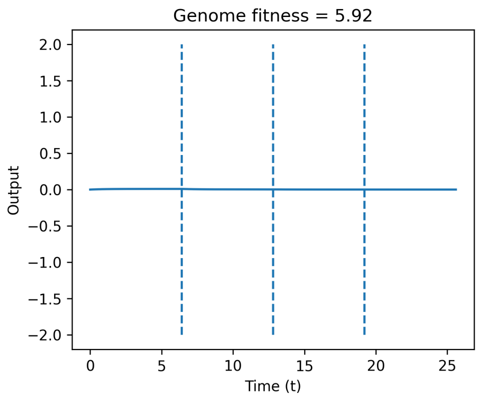
(b) Network topology of winner 8.



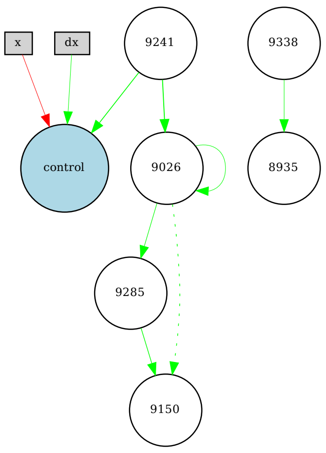
(a) CTRNN output over 25.6s from winner 9. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 9.

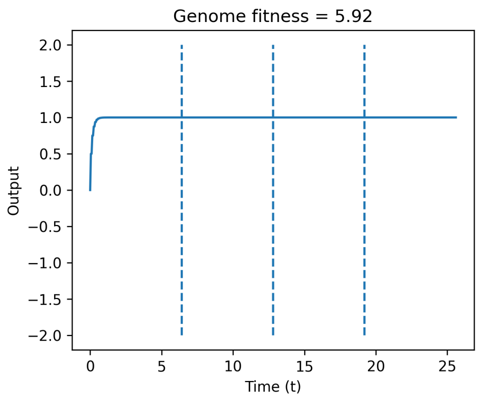


(a) CTRNN output over 25.6s from winner 10. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.

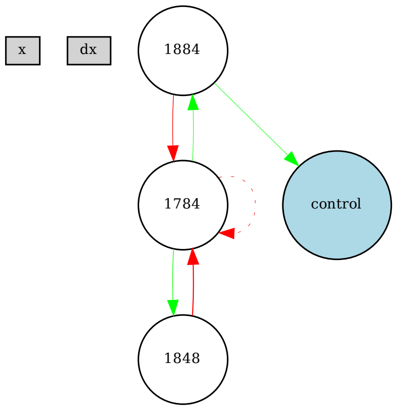


(b) Network topology of winner 10.

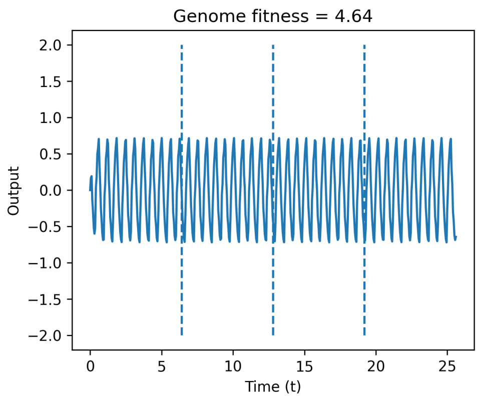
A.1.4 Default Genome Configuration C - RMSE



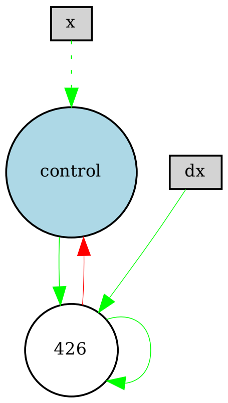
(a) CTRNN output over 25.6s from winner 1. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



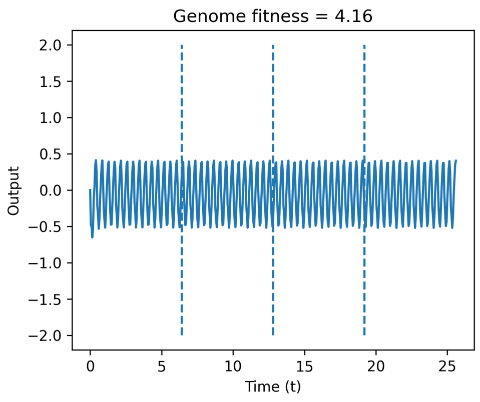
(b) Network topology of winner 1.



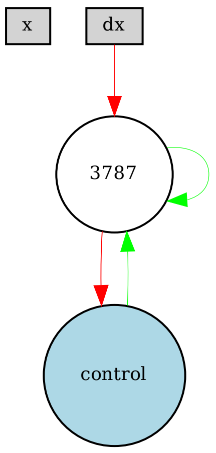
(a) CTRNN output over 25.6s from winner 2. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



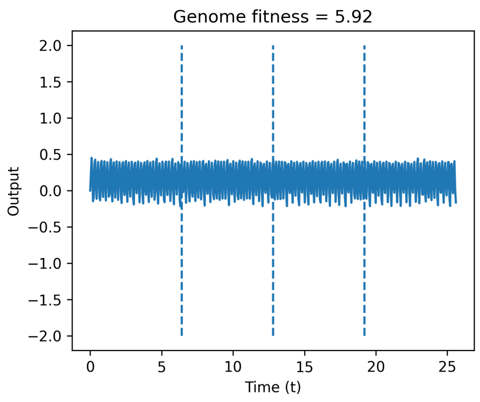
(b) Network topology of winner 2.



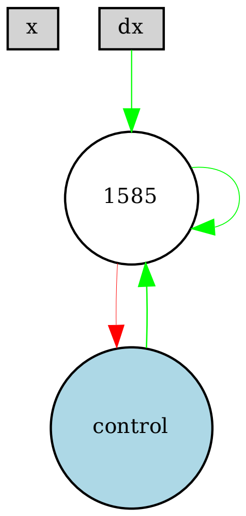
(a) CTRNN output over 25.6s from winner 3. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



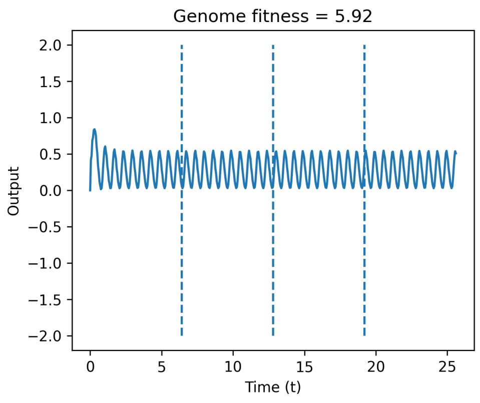
(b) Network topology of winner 3.



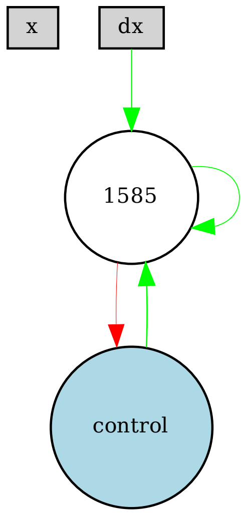
(a) CTRNN output over 25.6s from winner 4. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



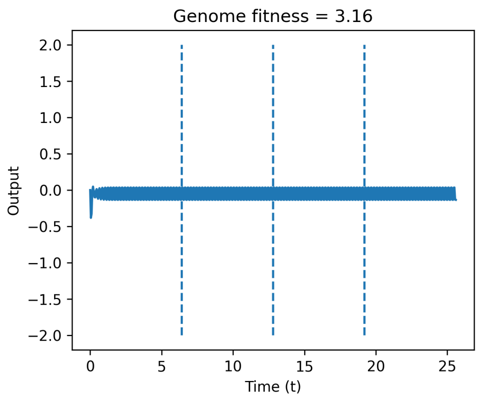
(b) Network topology of winner 4.



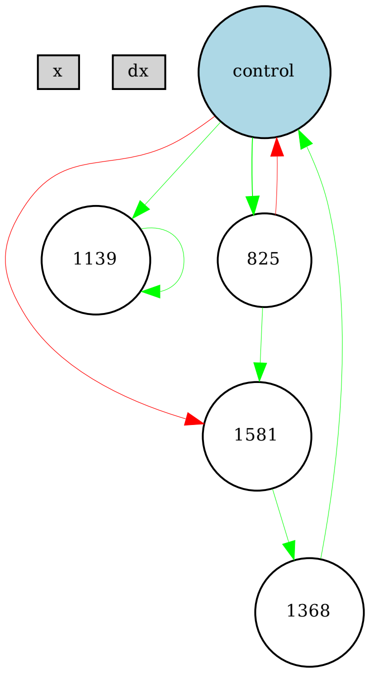
(a) CTRNN output over 25.6s from winner 5. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



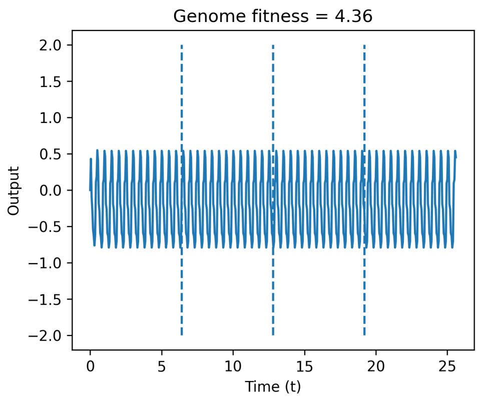
(b) Network topology of winner 5.



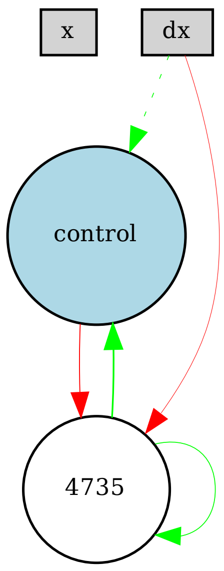
(a) CTRNN output over 25.6s from winner 6. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



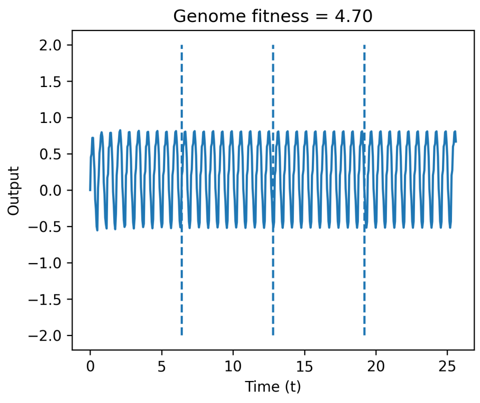
(b) Network topology of winner 6.



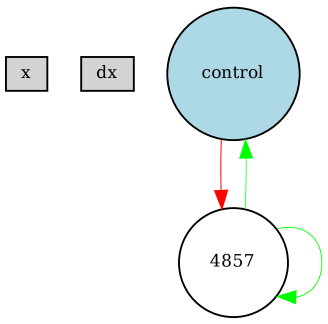
(a) CTRNN output over 25.6s from winner 7. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



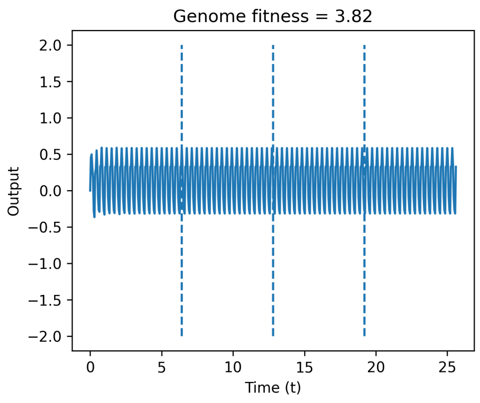
(b) Network topology of winner 7.



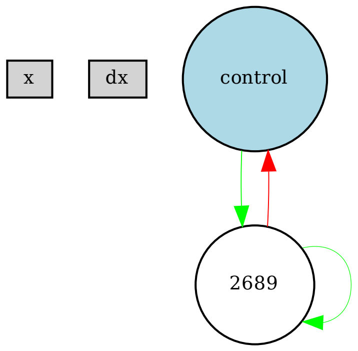
(a) CTRNN output over 25.6s from winner 8. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



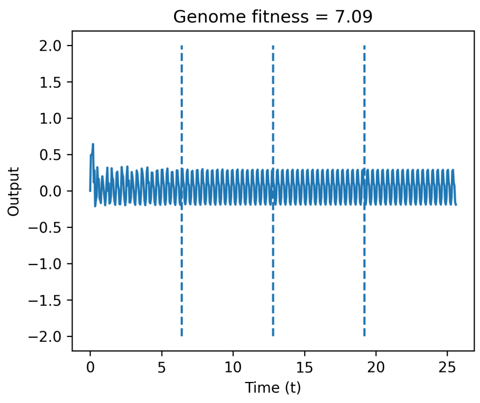
(b) Network topology of winner 8.



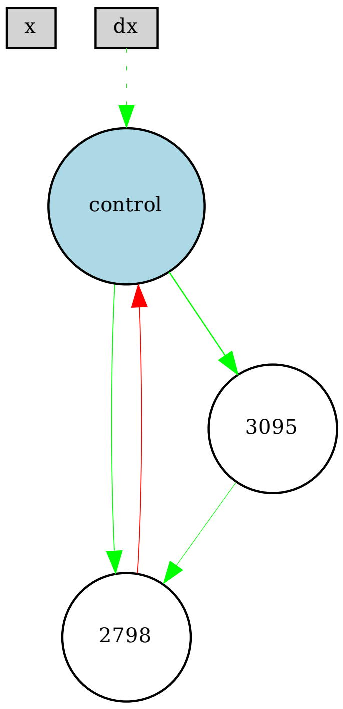
(a) CTRNN output over 25.6s from winner 9. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 9.

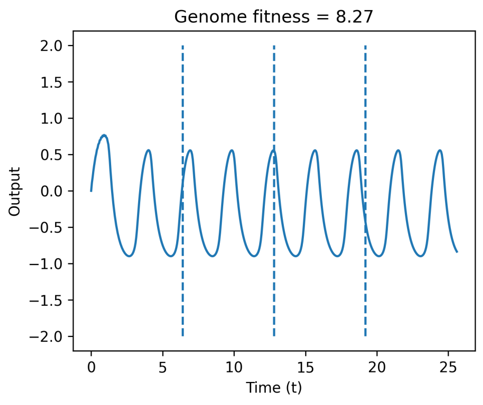


(a) CTRNN output over 25.6s from winner 10. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.

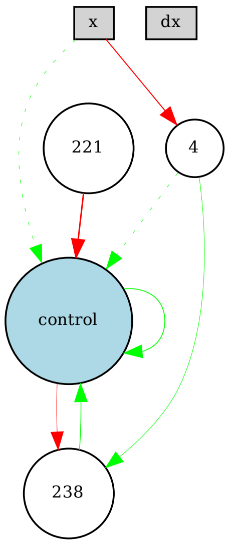


(b) Network topology of winner 10.

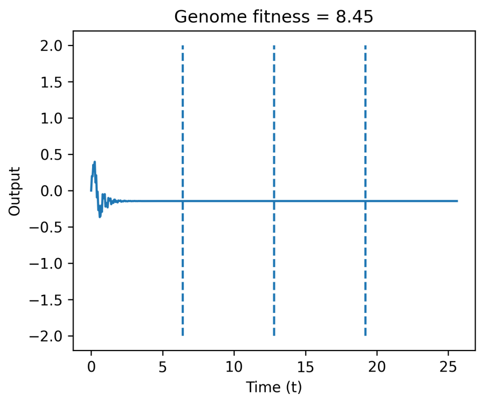
A.1.5 Default Genome Configuration D - RMSE



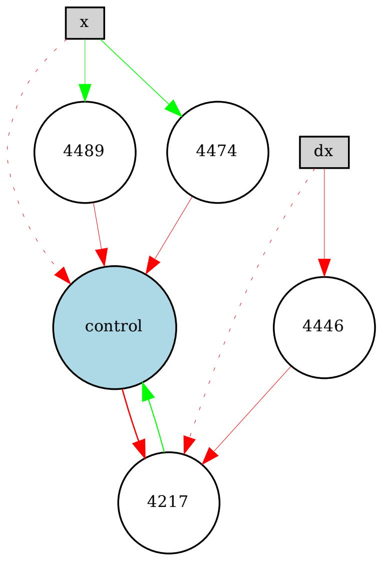
(a) CTRNN output over 25.6s from winner 1. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



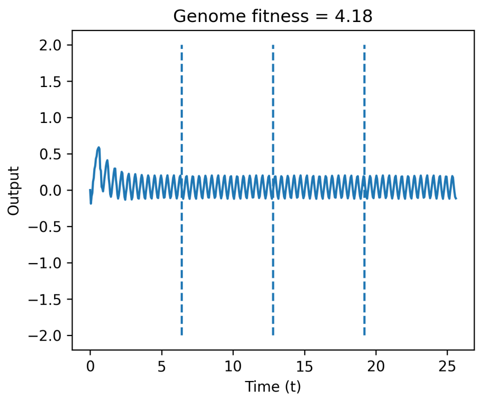
(b) Network topology of winner 1.



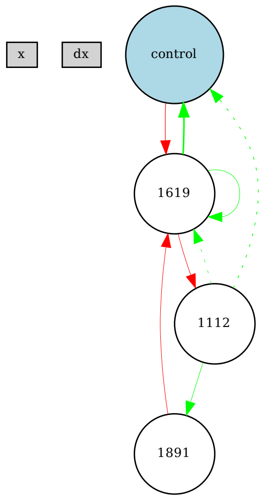
(a) CTRNN output over 25.6s from winner 2. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



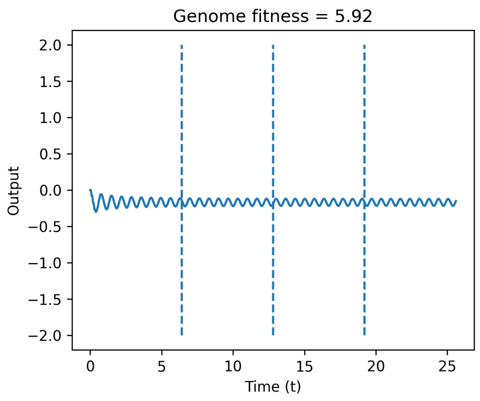
(b) Network topology of winner 2.



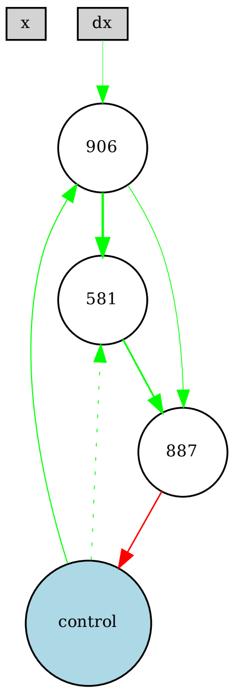
(a) CTRNN output over 25.6s from winner 3. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



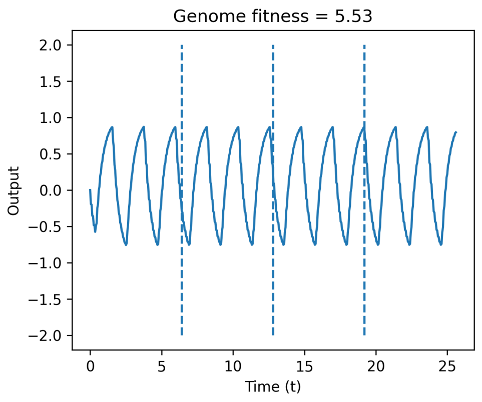
(b) Network topology of winner 3.



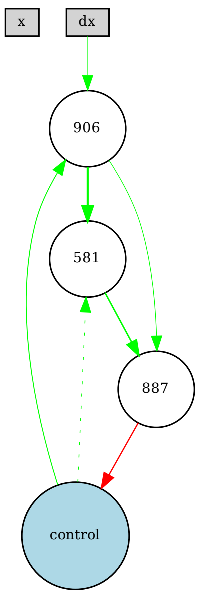
(a) CTRNN output over 25.6s from winner 4. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



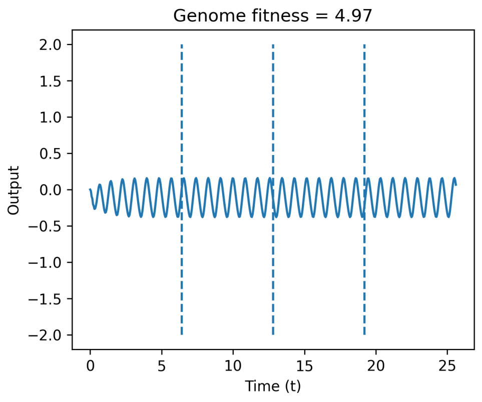
(b) Network topology of winner 4.



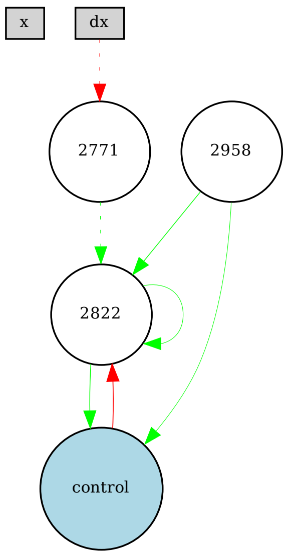
(a) CTRNN output over 25.6s from winner 5. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



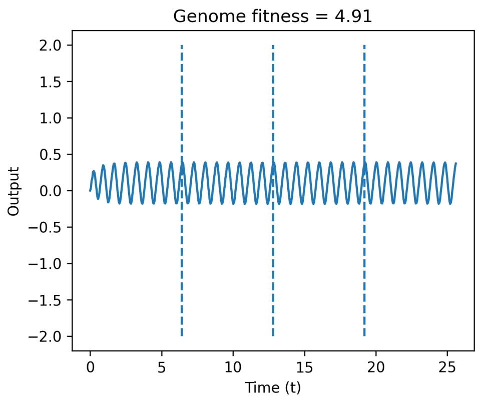
(b) Network topology of winner 5.



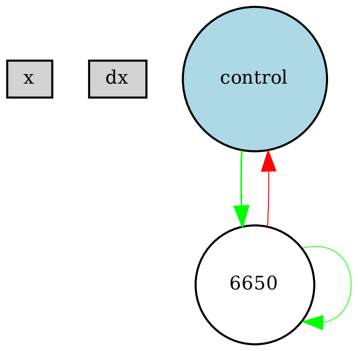
(a) CTRNN output over 25.6s from winner 6. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



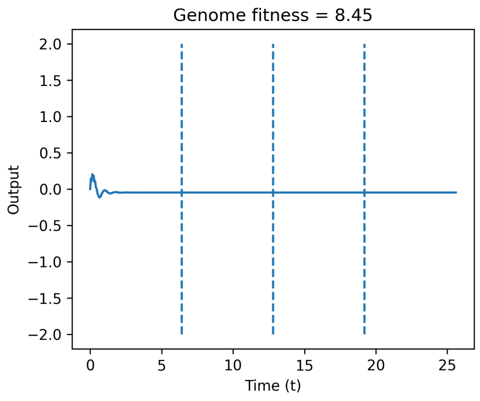
(b) Network topology of winner 6.



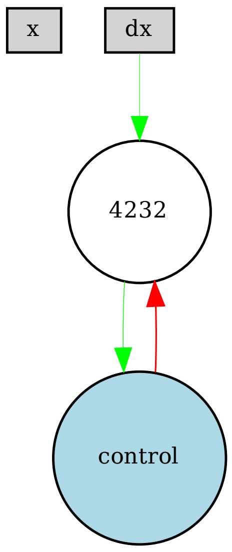
(a) CTRNN output over 25.6s from winner 7. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



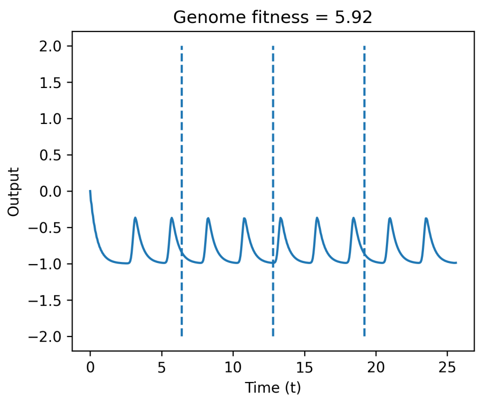
(b) Network topology of winner 7.



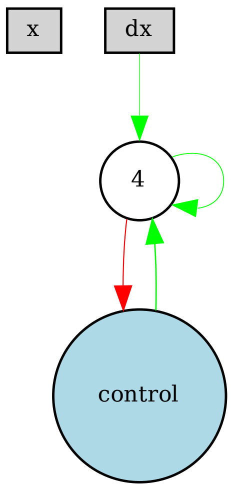
(a) CTRNN output over 25.6s from winner 8. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



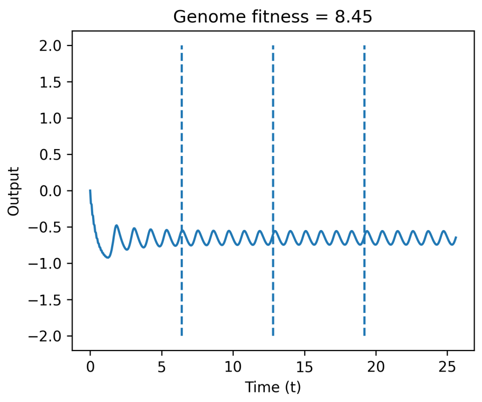
(b) Network topology of winner 8.



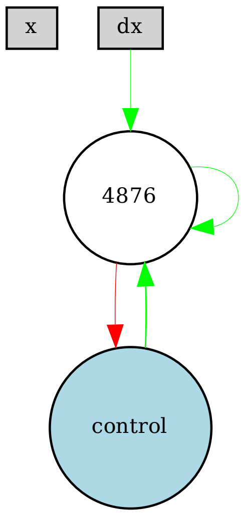
(a) CTRNN output over 25.6s from winner 9. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 9.

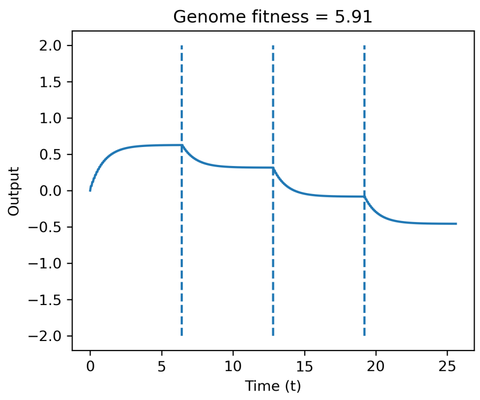


(a) CTRNN output over 25.6s from winner 10. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.

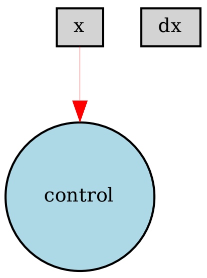


(b) Network topology of winner 10.

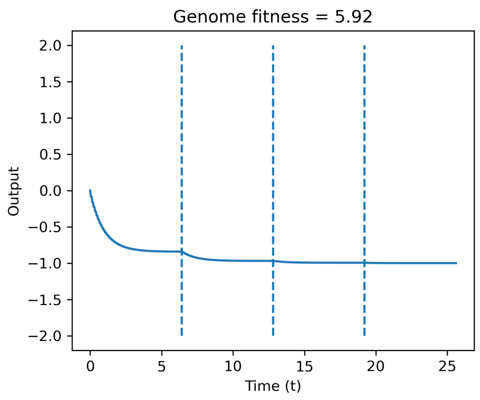
A.1.6 Default Genome Configuration E - RMSE



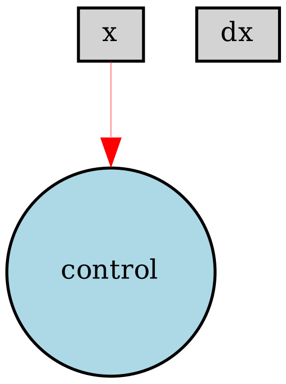
(a) CTRNN output over 25.6s from winner 1. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



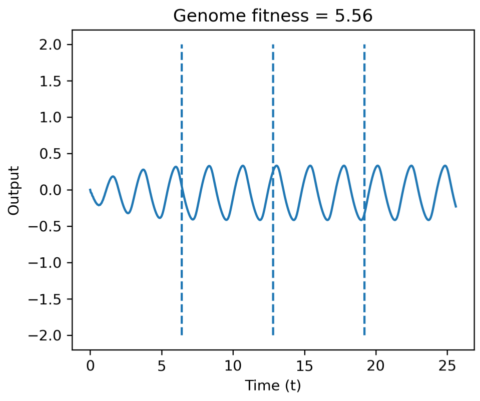
(b) Network topology of winner 1.



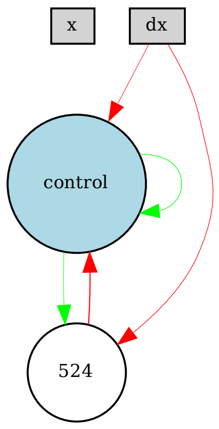
(a) CTRNN output over 25.6s from winner 2. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



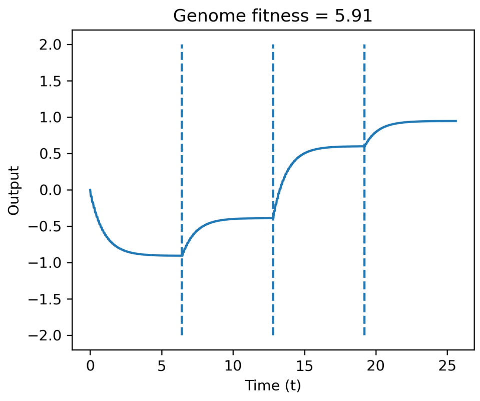
(b) Network topology of winner 2.



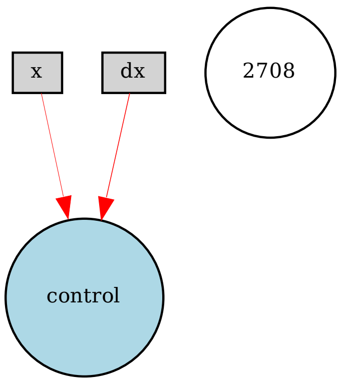
(a) CTRNN output over 25.6s from winner 3. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



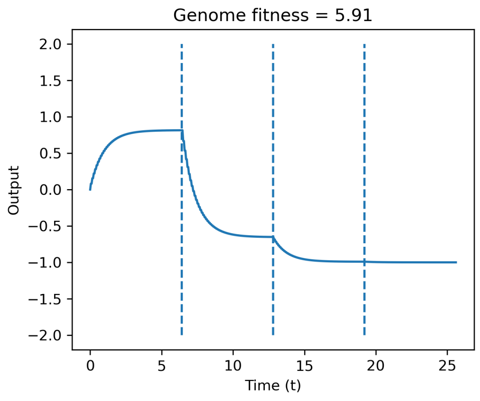
(b) Network topology of winner 3.



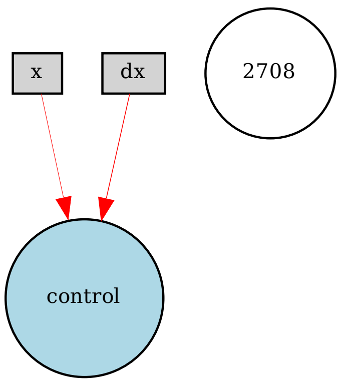
(a) CTRNN output over 25.6s from winner 4. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



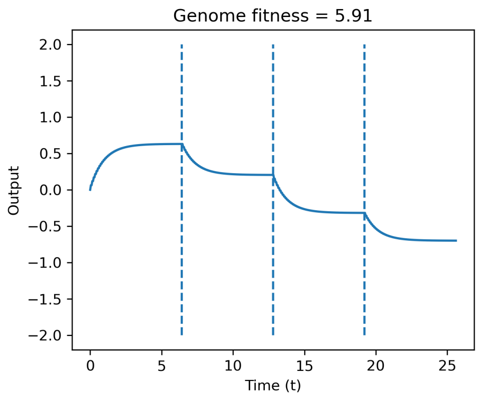
(b) Network topology of winner 4.



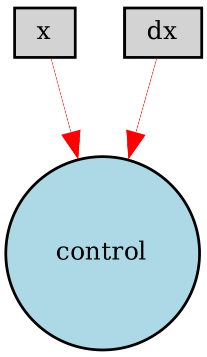
(a) CTRNN output over 25.6s from winner 5. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



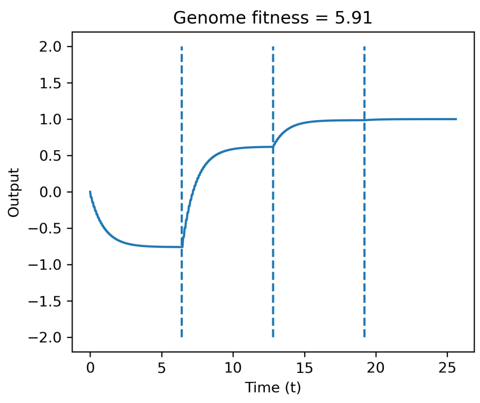
(b) Network topology of winner 5.



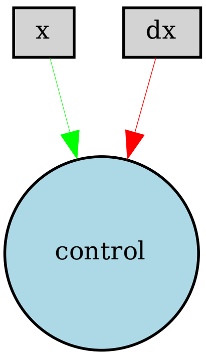
(a) CTRNN output over 25.6s from winner 6. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



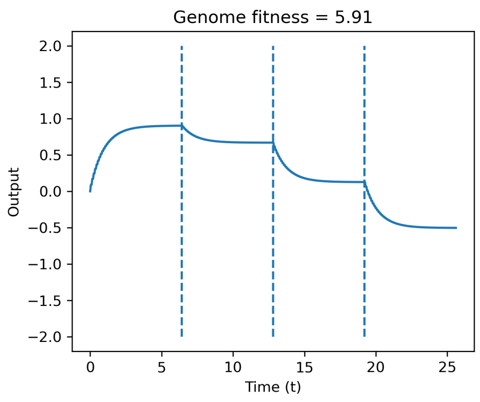
(b) Network topology of winner 6.



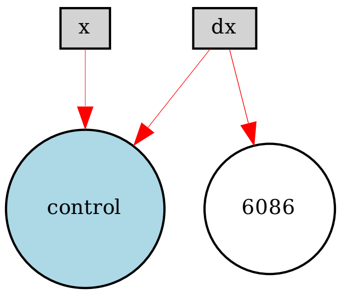
(a) CTRNN output over 25.6s from winner 7. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



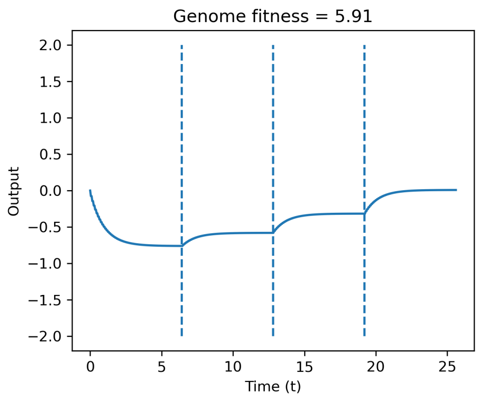
(b) Network topology of winner 7.



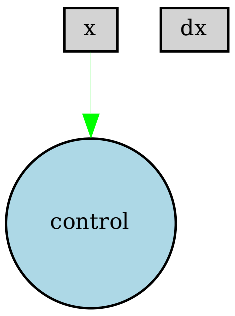
(a) CTRNN output over 25.6s from winner 8. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



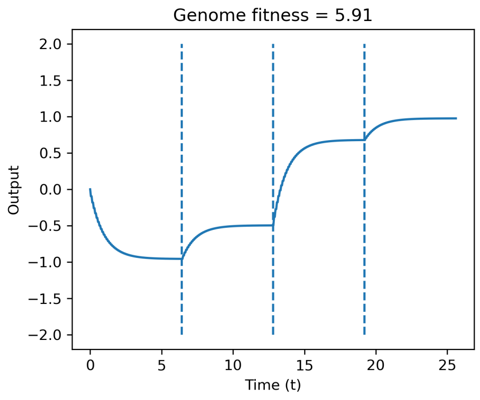
(b) Network topology of winner 8.



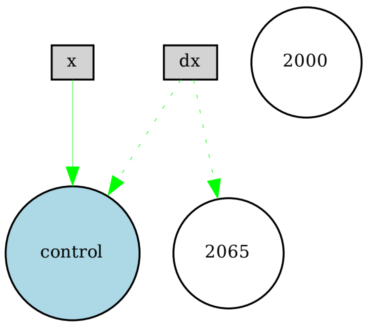
(a) CTRNN output over 25.6s from winner 9. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 9.

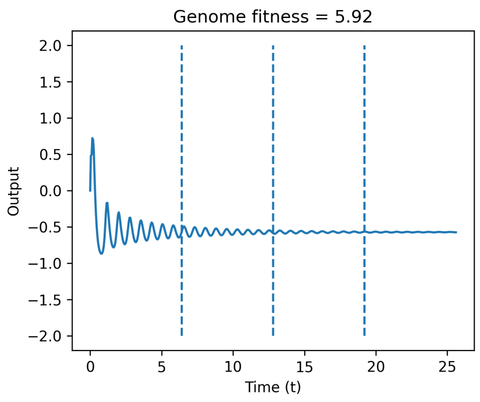


(a) CTRNN output over 25.6s from winner 10. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.

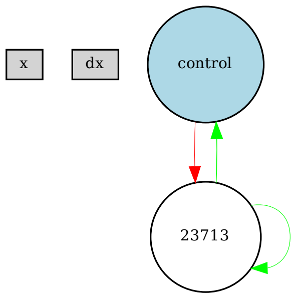


(b) Network topology of winner 10.

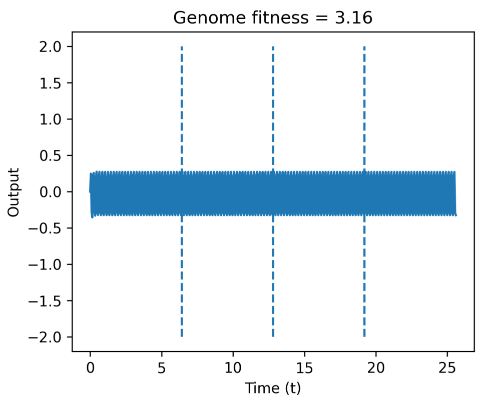
A.1.7 Default Genome Configuration F - RMSE



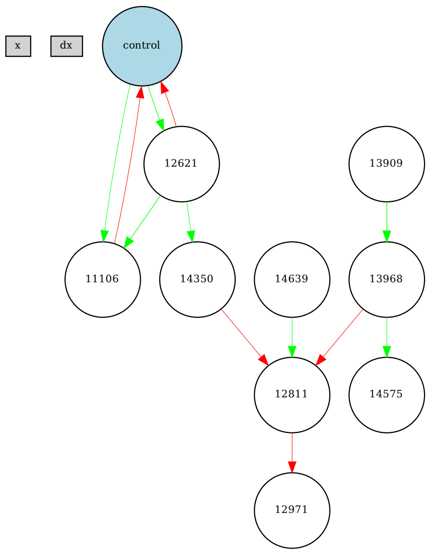
(a) CTRNN output over 25.6s from winner 1. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



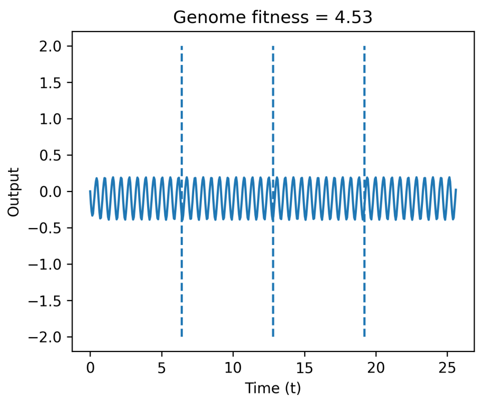
(b) Network topology of winner 1.



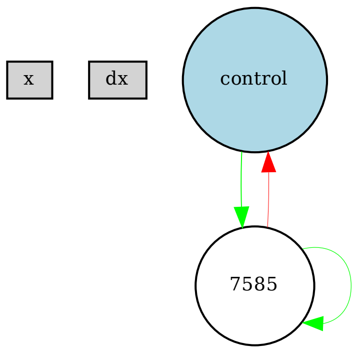
(a) CTRNN output over 25.6s from winner 2. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



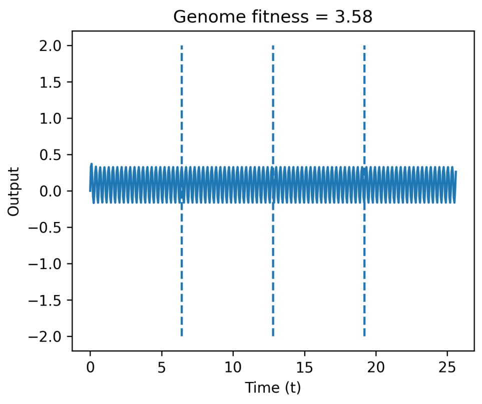
(b) Network topology of winner 2.



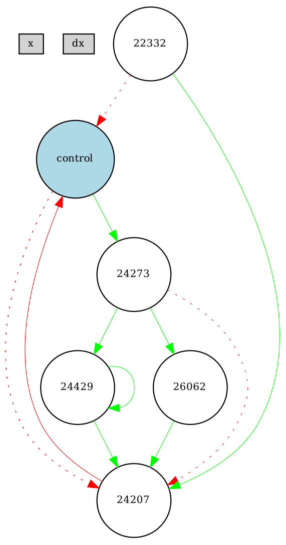
(a) CTRNN output over 25.6s from winner 3. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



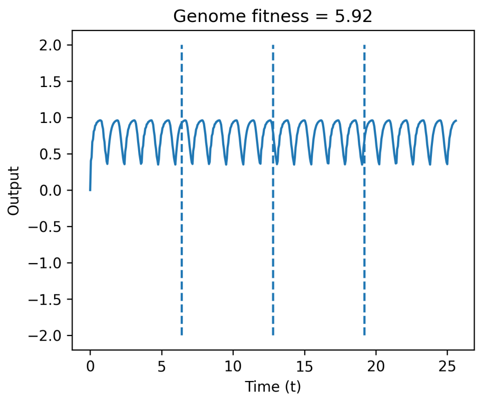
(b) Network topology of winner 3.



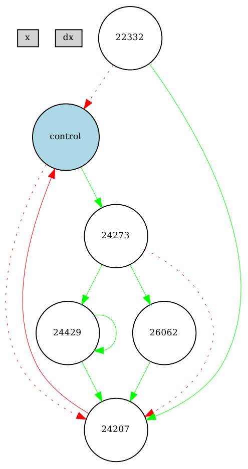
(a) CTRNN output over 25.6s from winner 4. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



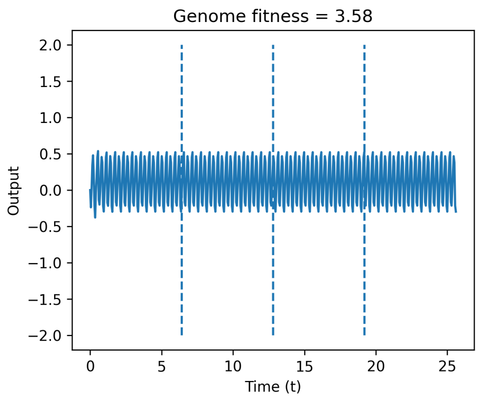
(b) Network topology of winner 4.



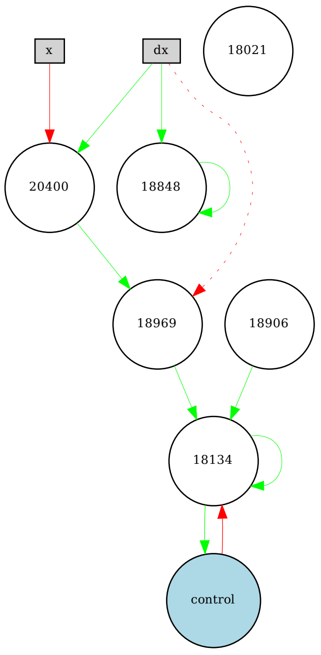
(a) CTRNN output over 25.6s from winner 5. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



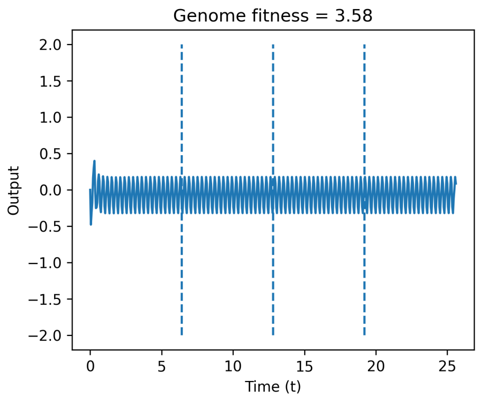
(b) Network topology of winner 5.



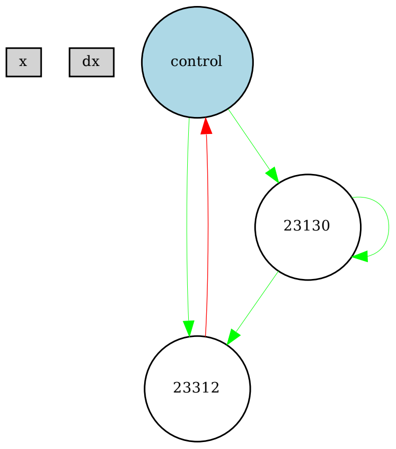
(a) CTRNN output over 25.6s from winner 6. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



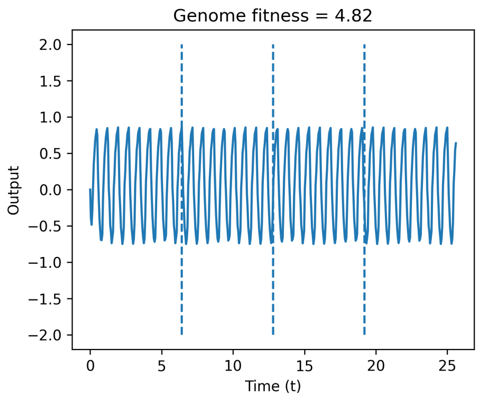
(b) Network topology of winner 6.



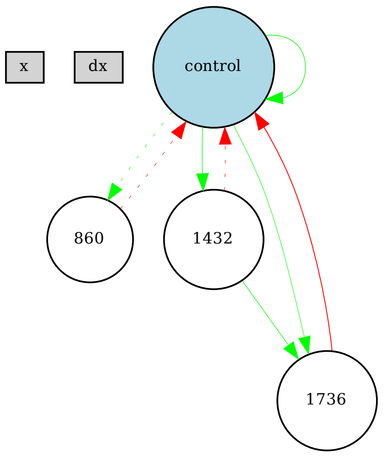
(a) CTRNN output over 25.6s from winner 7. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



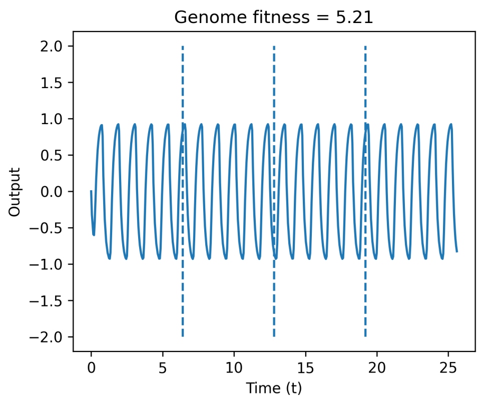
(b) Network topology of winner 7.



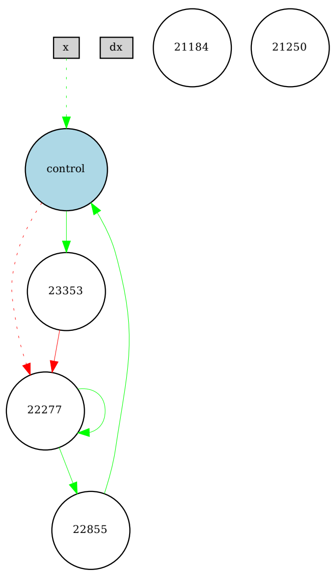
(a) CTRNN output over 25.6s from winner 8. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



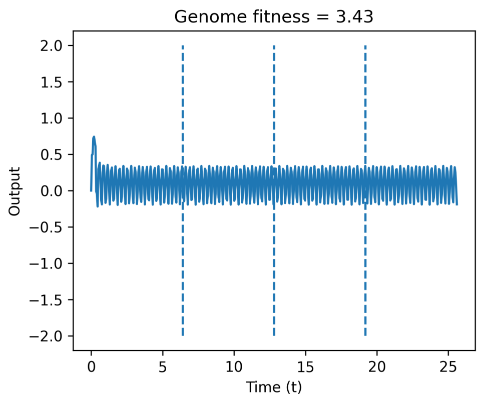
(b) Network topology of winner 8.



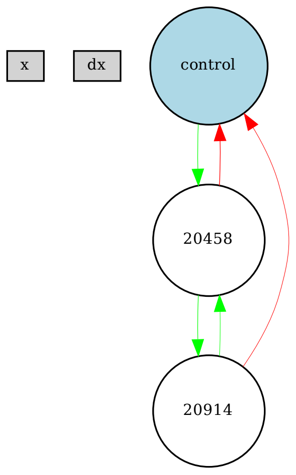
(a) CTRNN output over 25.6s from winner 9. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 9.

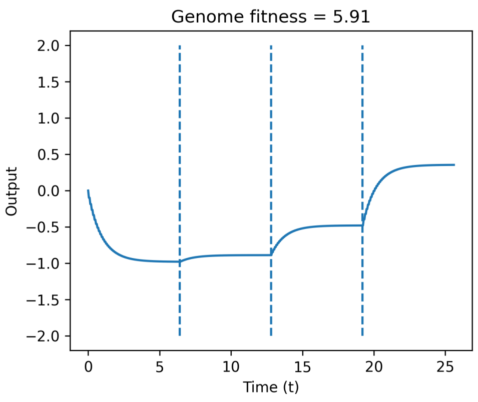


(a) CTRNN output over 25.6s from winner 10. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.

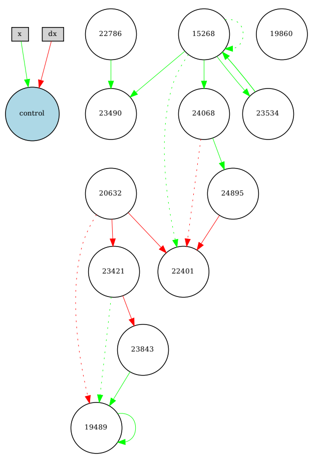


(b) Network topology of winner 10.

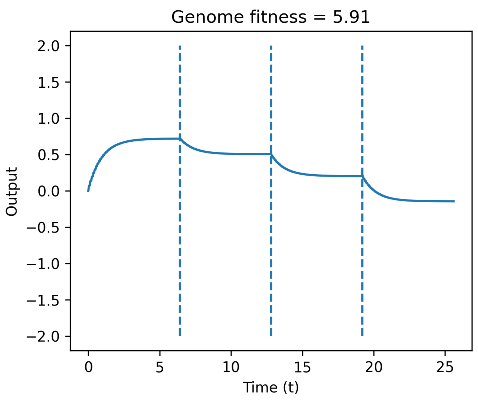
A.1.8 Default Genome Configuration G - RMSE



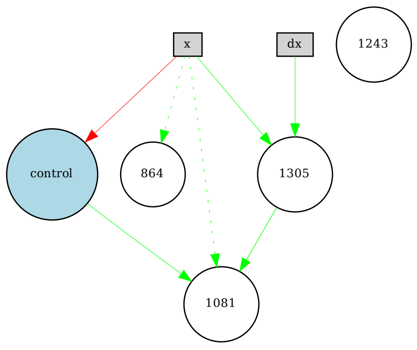
(a) CTRNN output over 25.6s from winner 1. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



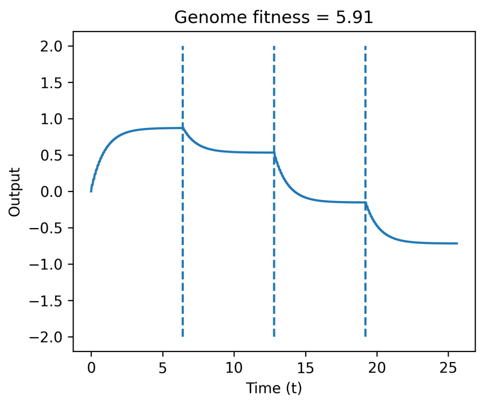
(b) Network topology of winner 1.



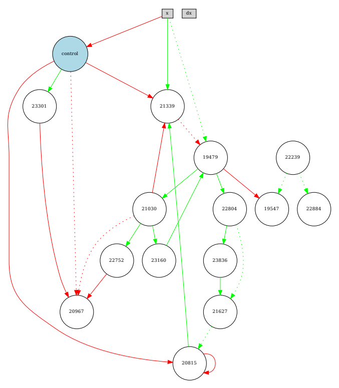
(a) CTRNN output over 25.6s from winner 2. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



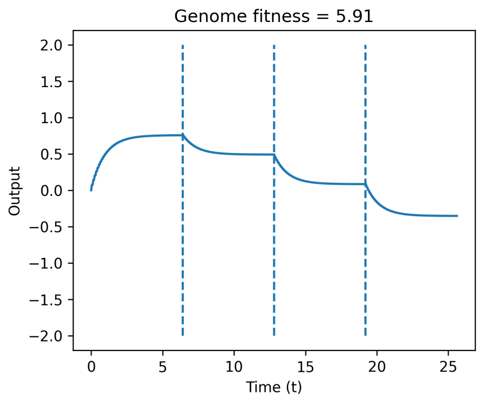
(b) Network topology of winner 2.



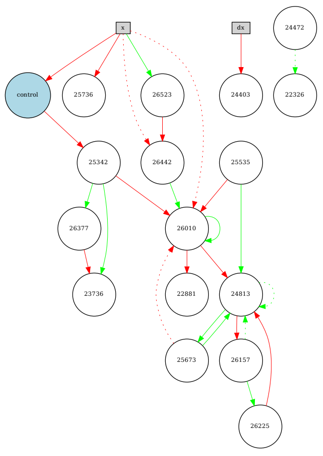
(a) CTRNN output over 25.6s from winner 3. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



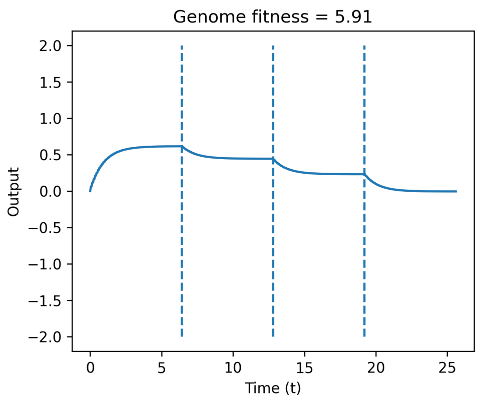
(b) Network topology of winner 3.



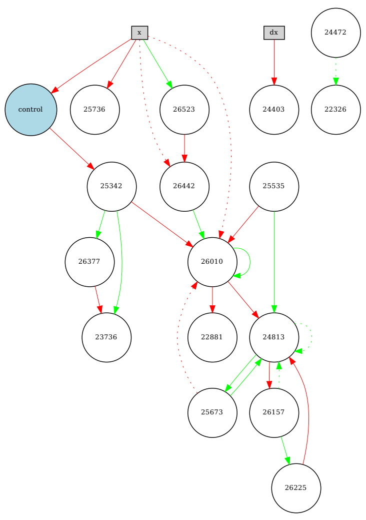
(a) CTRNN output over 25.6s from winner 4. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



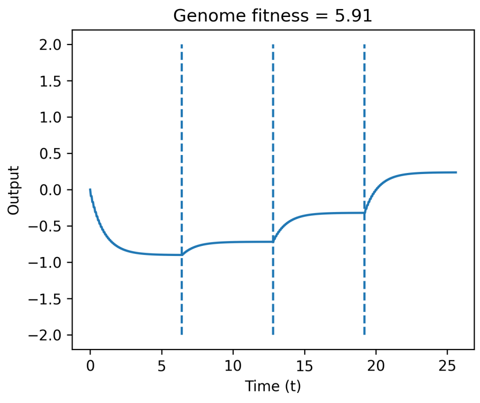
(b) Network topology of winner 4.



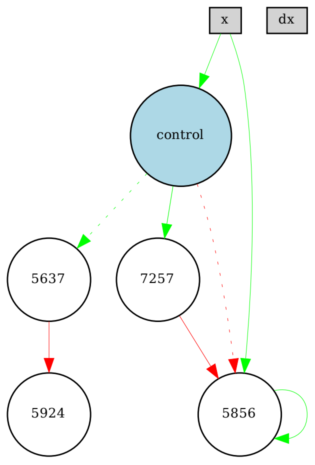
(a) CTRNN output over 25.6s from winner 5. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



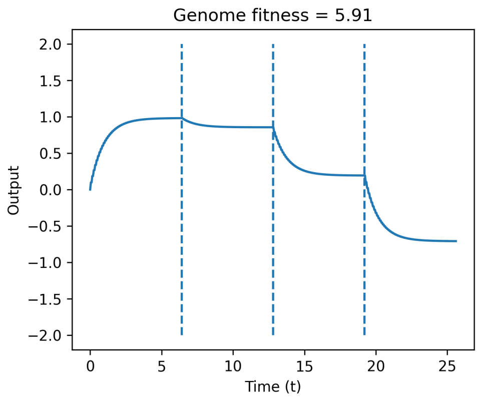
(b) Network topology of winner 5.



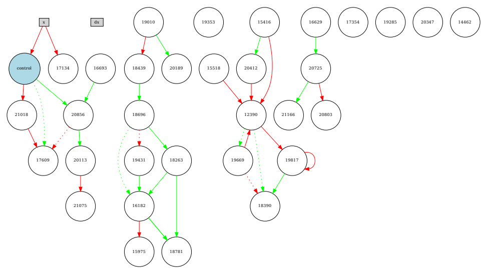
(a) CTRNN output over 25.6s from winner 6. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



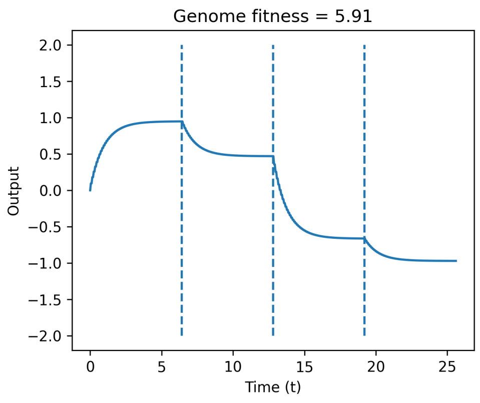
(b) Network topology of winner 6.



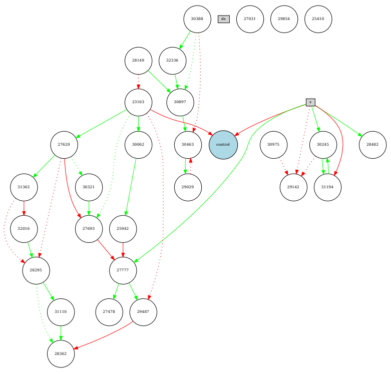
(a) CTRNN output over 25.6s from winner 7. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



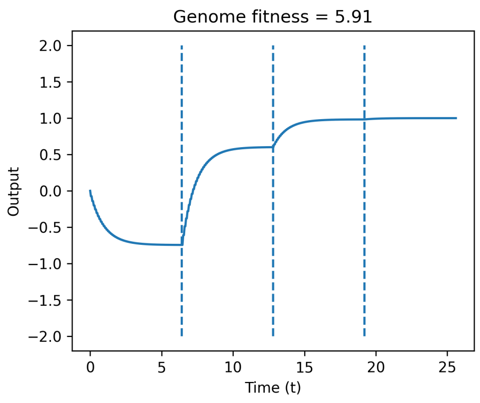
(b) Network topology of winner 7.



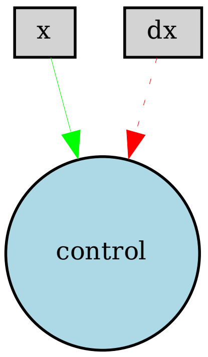
(a) CTRNN output over 25.6s from winner 8. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



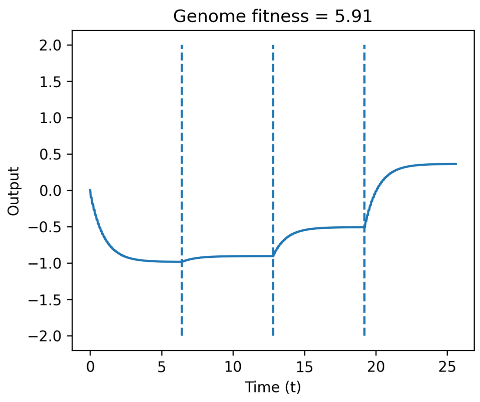
(b) Network topology of winner 8.



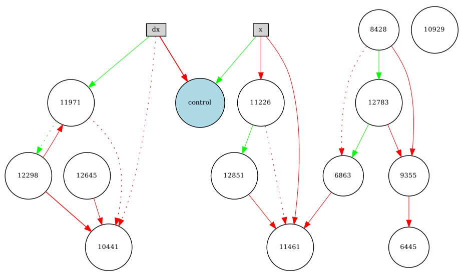
(a) CTRNN output over 25.6s from winner 9. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 9.

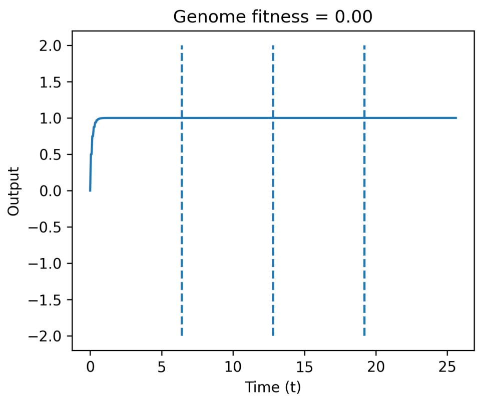


(a) CTRNN output over 25.6s from winner 10. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.

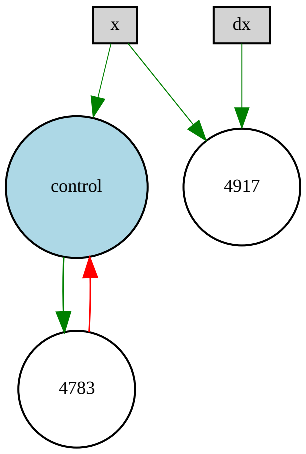


(b) Network topology of winner 10.

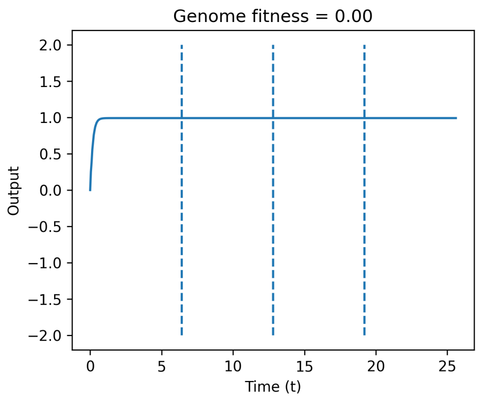
A.1.9 Default Genome Configuration Z - NEE



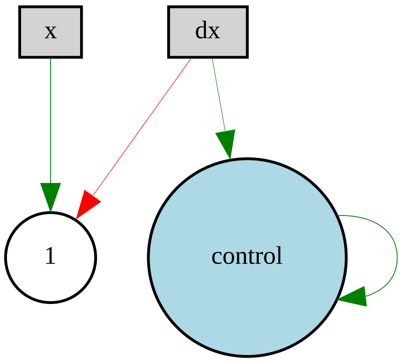
(a) CTRNN output over 25.6s from winner 1. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



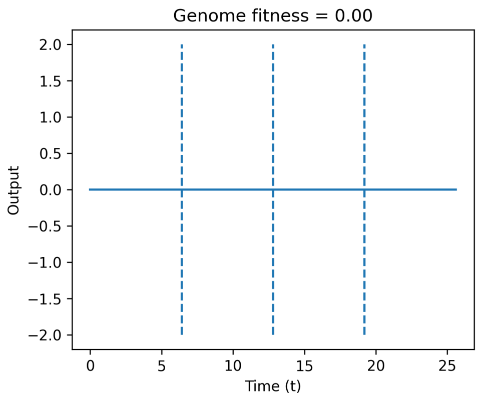
(b) Network topology of winner 1.



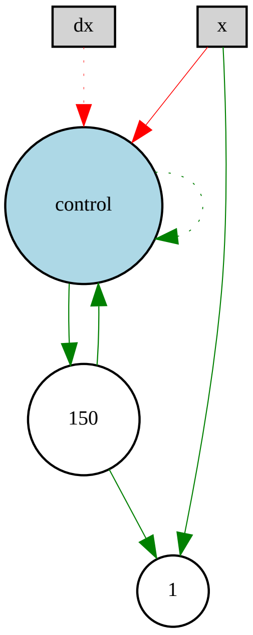
(a) CTRNN output over 25.6s from winner 2. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



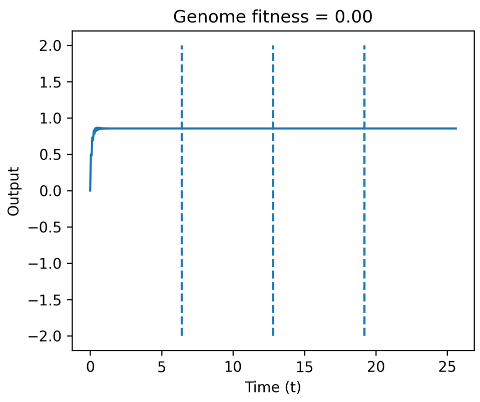
(b) Network topology of winner 2.



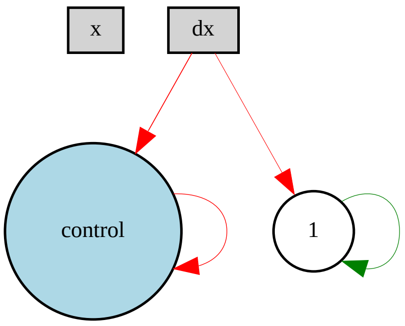
(a) CTRNN output over 25.6s from winner 3. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



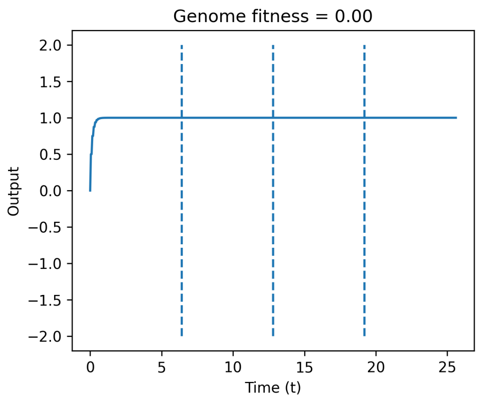
(b) Network topology of winner 3.



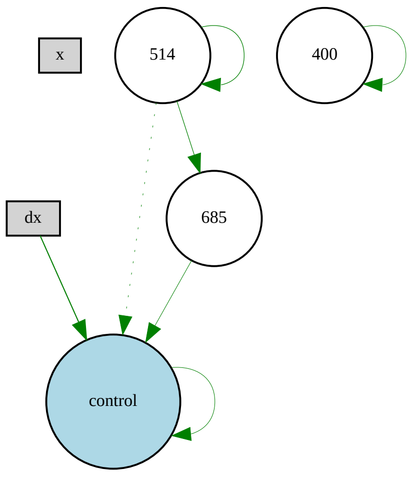
(a) CTRNN output over 25.6s from winner 4. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



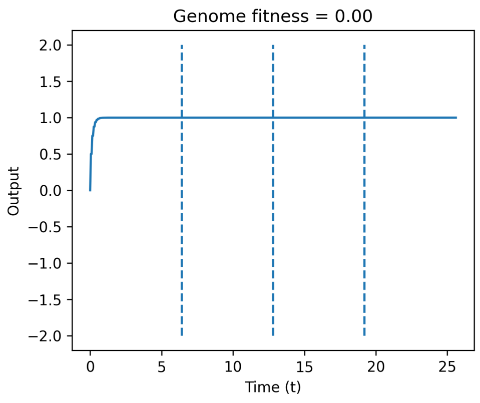
(b) Network topology of winner 4.



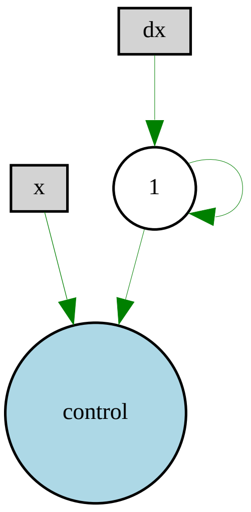
(a) CTRNN output over 25.6s from winner 5. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



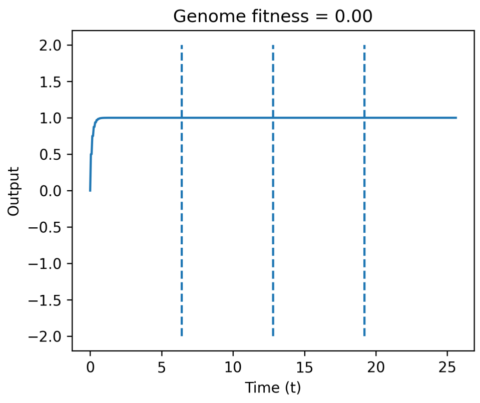
(b) Network topology of winner 5.



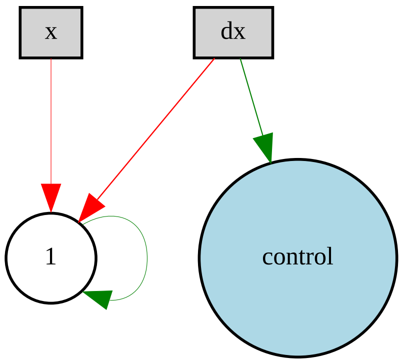
(a) CTRNN output over 25.6s from winner 6. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



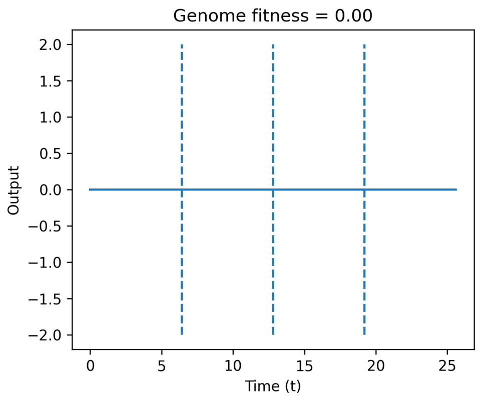
(b) Network topology of winner 6.



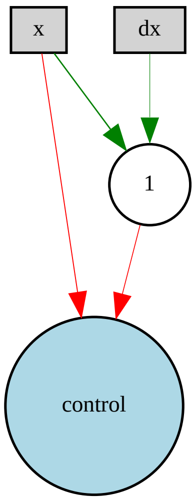
(a) CTRNN output over 25.6s from winner 7. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



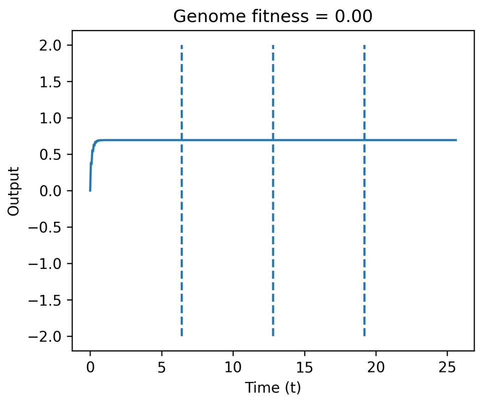
(b) Network topology of winner 7.



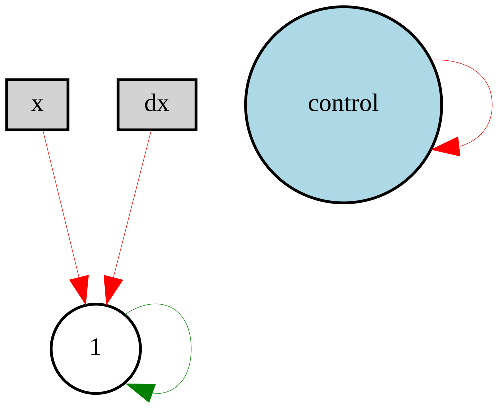
(a) CTRNN output over 25.6s from winner 8. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



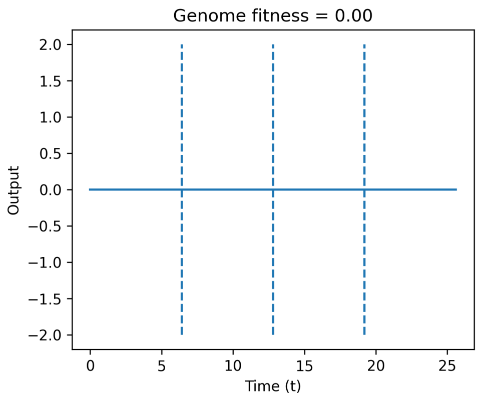
(b) Network topology of winner 8.



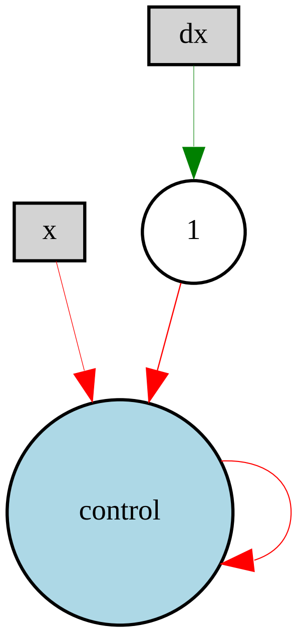
(a) CTRNN output over 25.6s from winner 9. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 9.

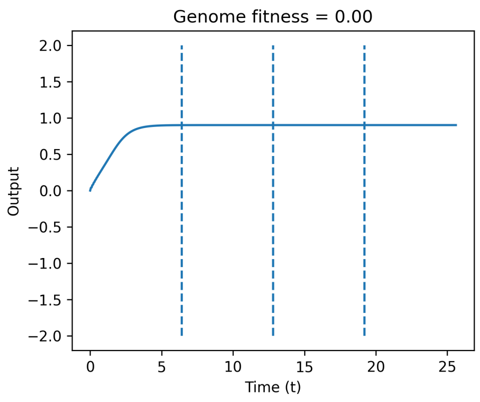


(a) CTRNN output over 25.6s from winner 10. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.

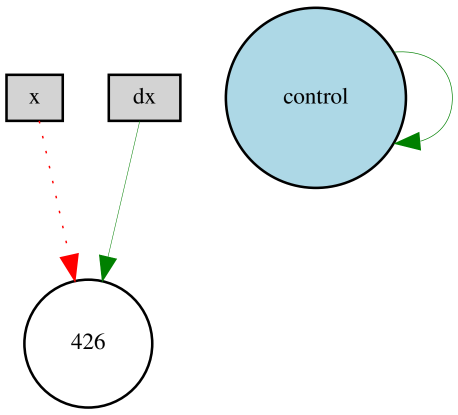


(b) Network topology of winner 10.

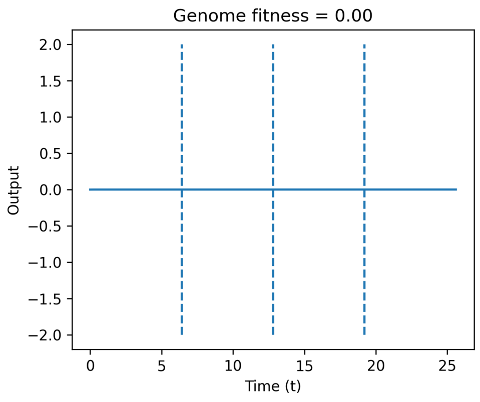
A.1.10 Default genome configuration A - NEE



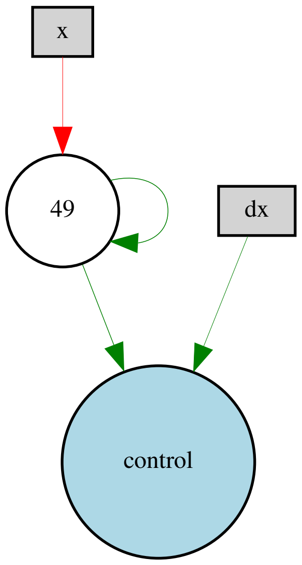
(a) CTRNN output over 25.6s from winner 1. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



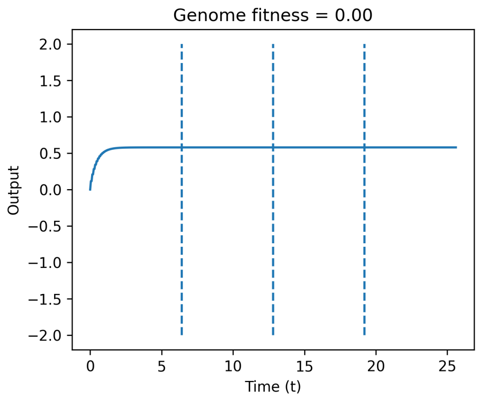
(b) Network topology of winner 1.



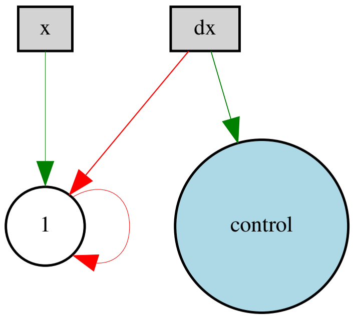
(a) CTRNN output over 25.6s from winner 2. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



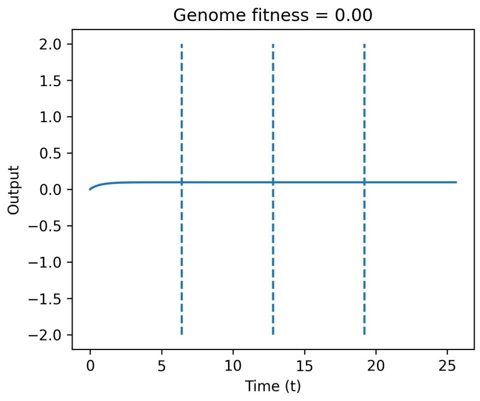
(b) Network topology of winner 2.



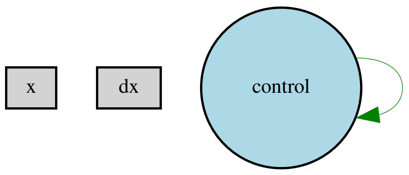
(a) CTRNN output over 25.6s from winner 3. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



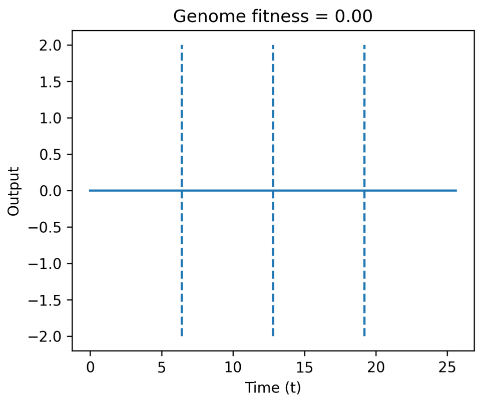
(b) Network topology of winner 3.



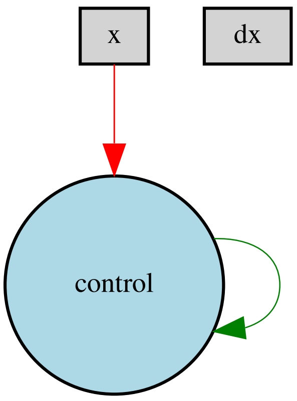
(a) CTRNN output over 25.6s from winner 4. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



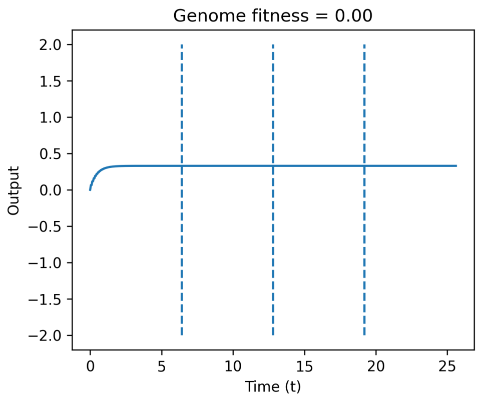
(b) Network topology of winner 4.



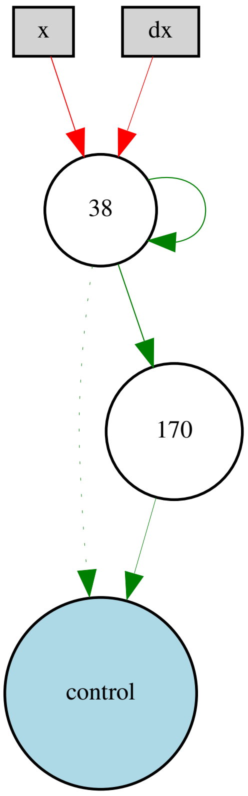
(a) CTRNN output over 25.6s from winner 5. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



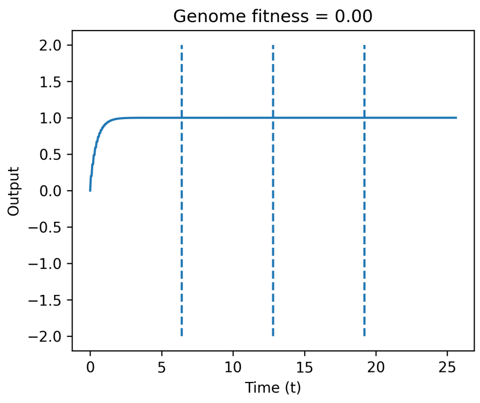
(b) Network topology of winner 5.



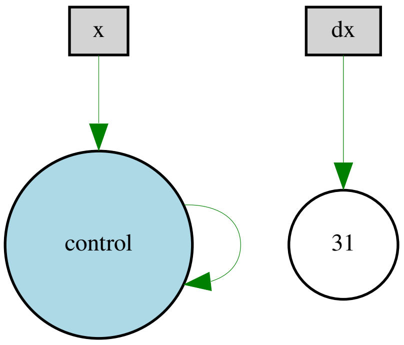
(a) CTRNN output over 25.6s from winner 6. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



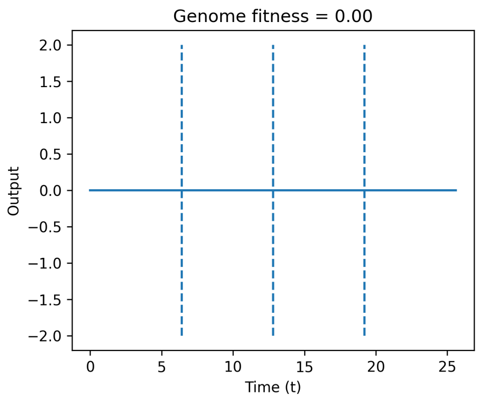
(b) Network topology of winner 6.



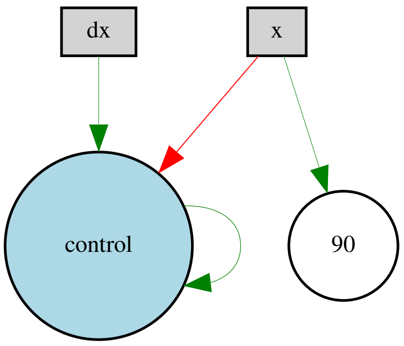
(a) CTRNN output over 25.6s from winner 7. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



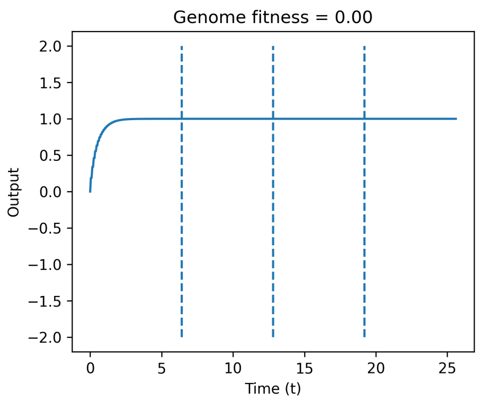
(b) Network topology of winner 7.



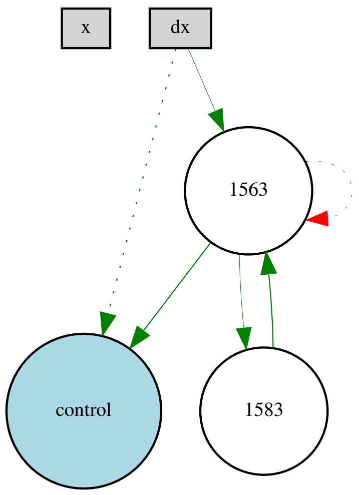
(a) CTRNN output over 25.6s from winner 8. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



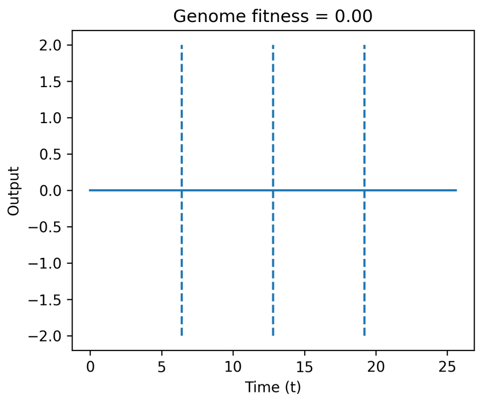
(b) Network topology of winner 8.



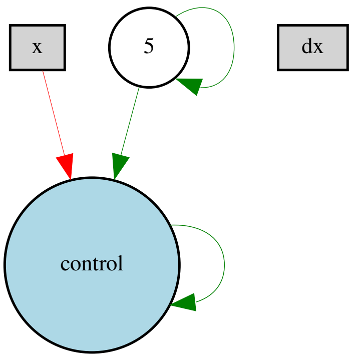
(a) CTRNN output over 25.6s from winner 9. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 9.

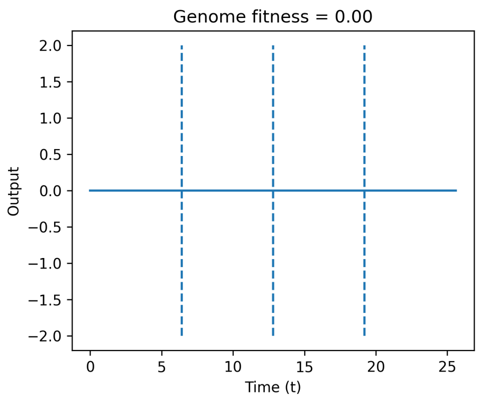


(a) CTRNN output over 25.6s from winner 10. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.

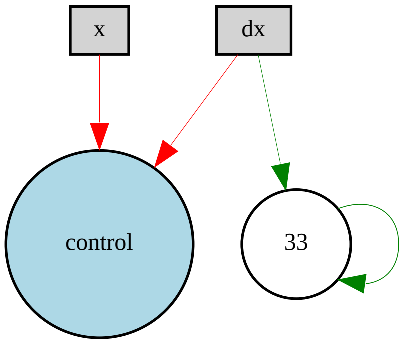


(b) Network topology of winner 10.

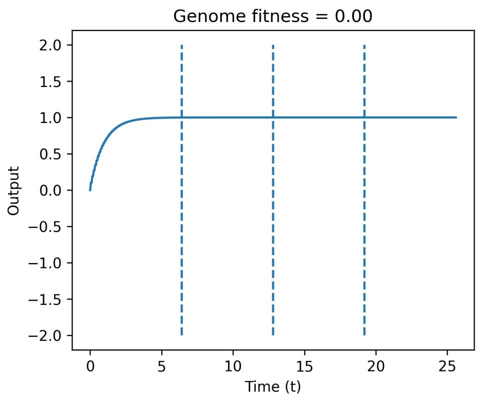
A.1.11 Default Genome Configuration B - NEE



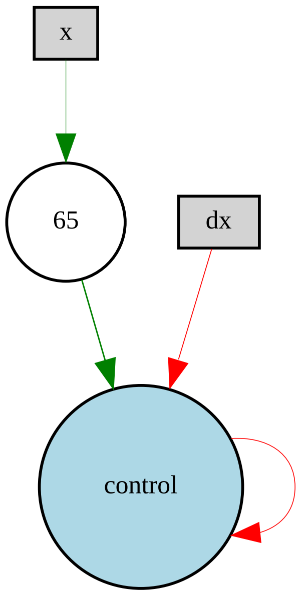
(a) CTRNN output over 25.6s from winner 1. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



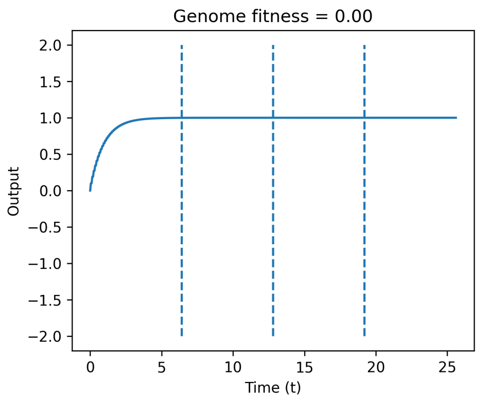
(b) Network topology of winner 1.



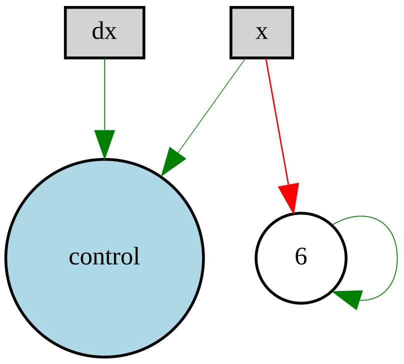
(a) CTRNN output over 25.6s from winner 2. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



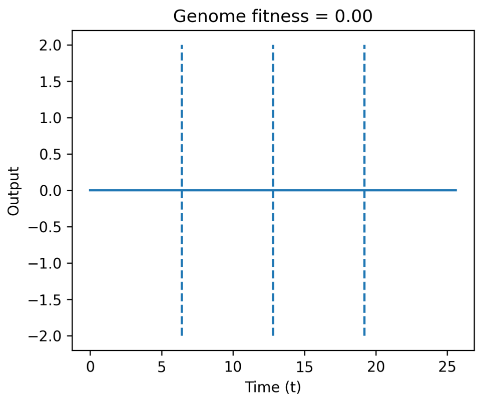
(b) Network topology of winner 2.



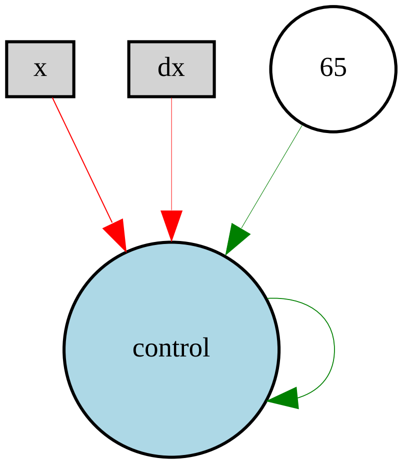
(a) CTRNN output over 25.6s from winner 3. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



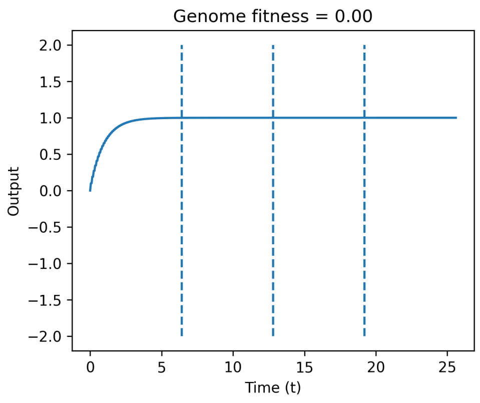
(b) Network topology of winner 3.



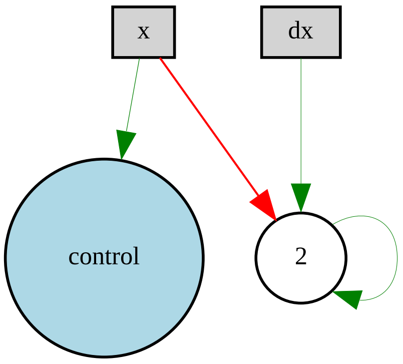
(a) CTRNN output over 25.6s from winner 4. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



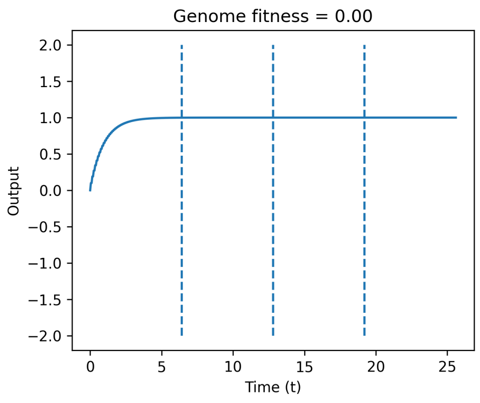
(b) Network topology of winner 4.



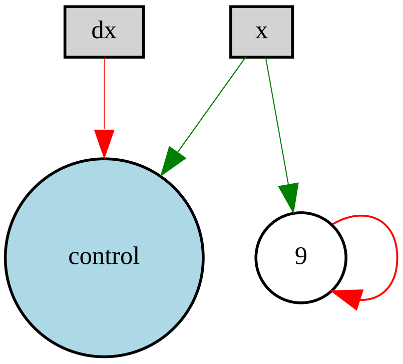
(a) CTRNN output over 25.6s from winner 5. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



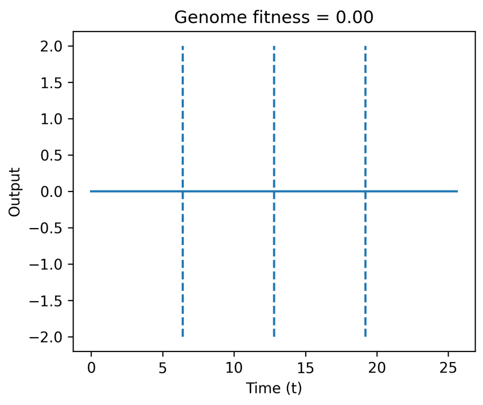
(b) Network topology of winner 5.



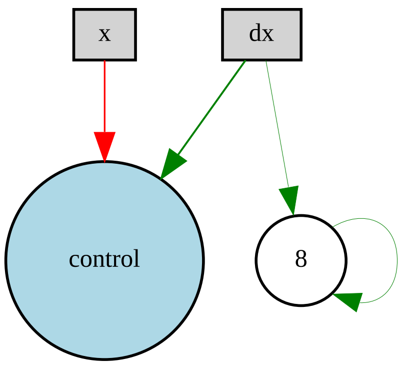
(a) CTRNN output over 25.6s from winner 6. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



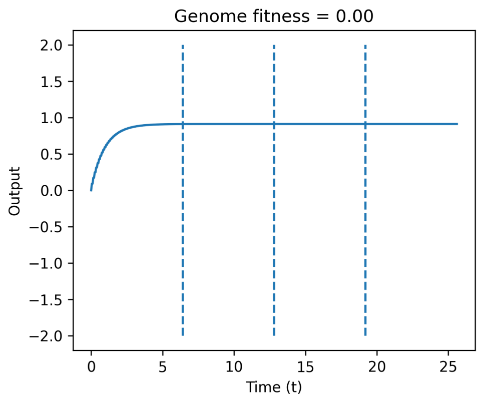
(b) Network topology of winner 6.



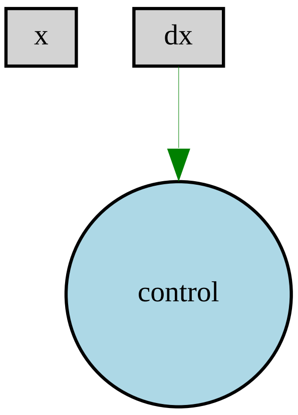
(a) CTRNN output over 25.6s from winner 7. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



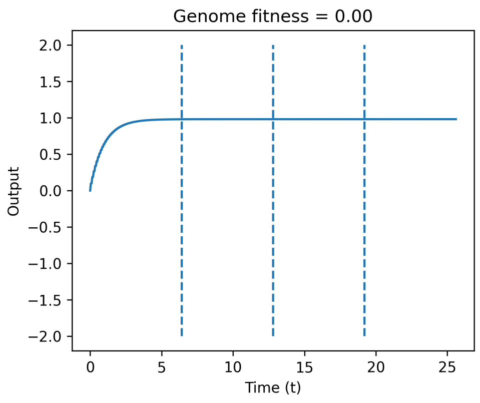
(b) Network topology of winner 7.



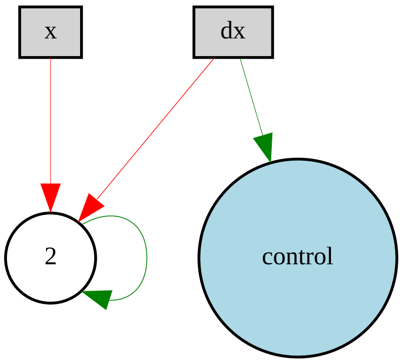
(a) CTRNN output over 25.6s from winner 8. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



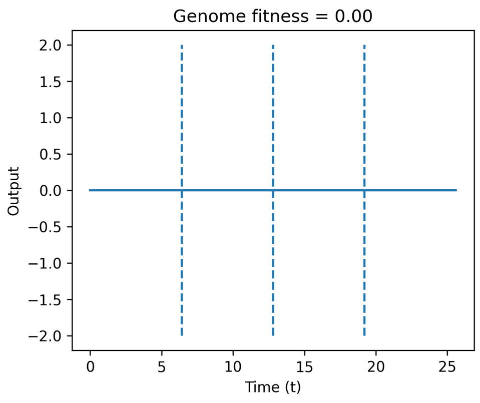
(b) Network topology of winner 8.



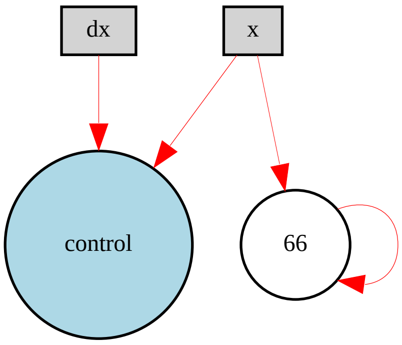
(a) CTRNN output over 25.6s from winner 9. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 9.

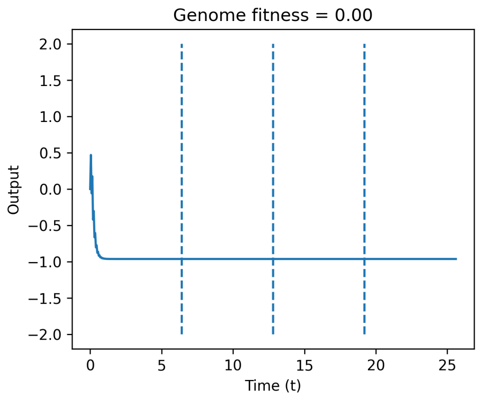


(a) CTRNN output over 25.6s from winner 10. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.

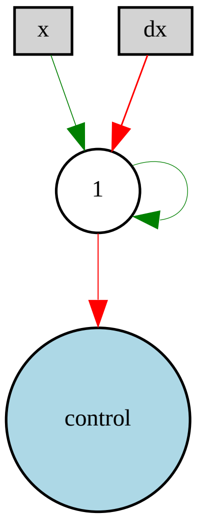


(b) Network topology of winner 10.

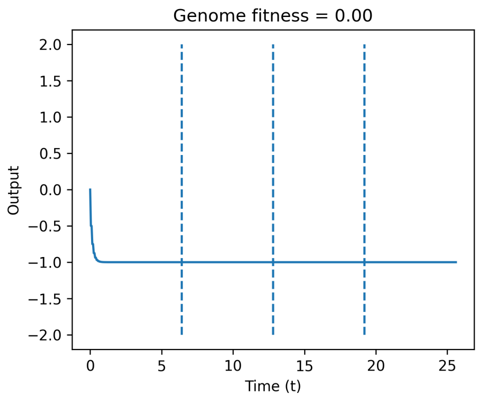
A.1.12 Default Genome Configuration C – NEE



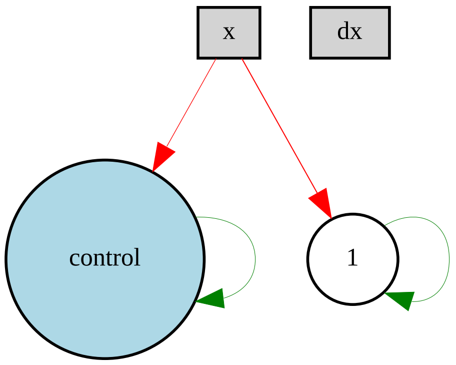
(a) CTRNN output over 25.6s from winner 1. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



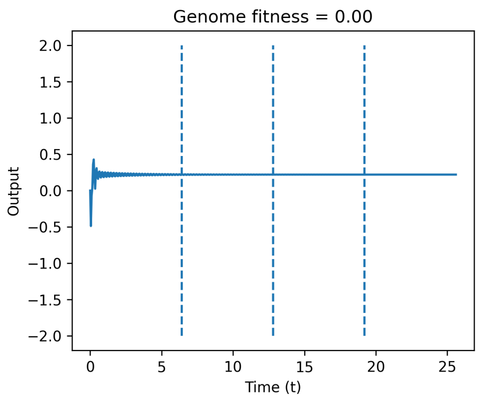
(b) Network topology of winner 1.



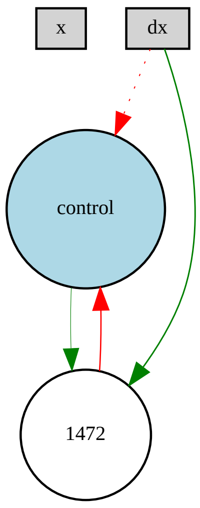
(a) CTRNN output over 25.6s from winner 2. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



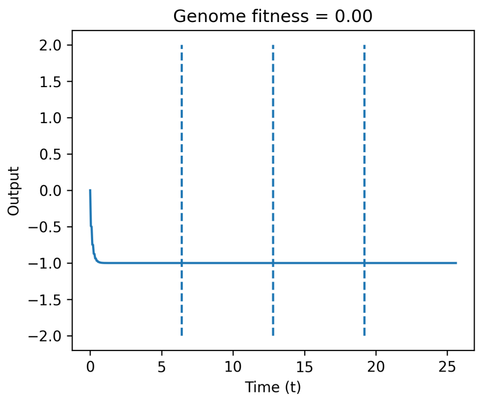
(b) Network topology of winner 2.



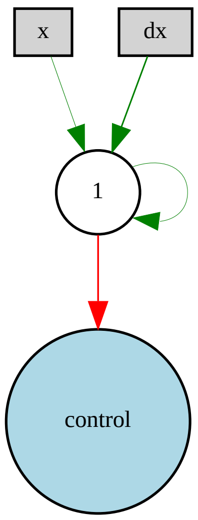
(a) CTRNN output over 25.6s from winner 3. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



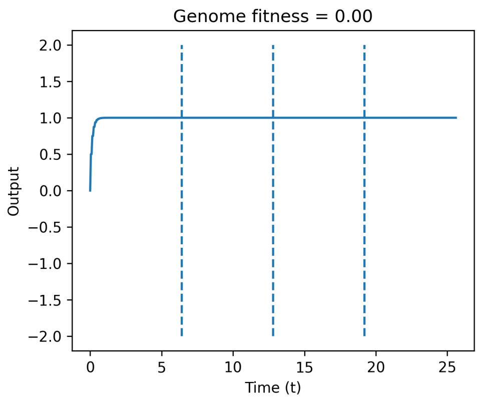
(b) Network topology of winner 3.



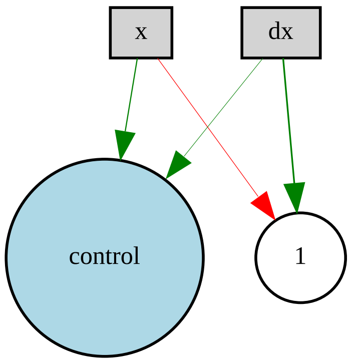
(a) CTRNN output over 25.6s from winner 4. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



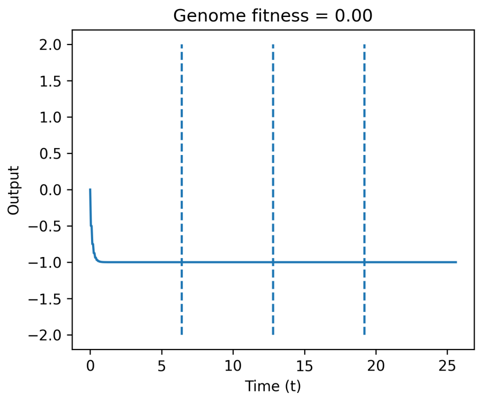
(b) Network topology of winner 4.



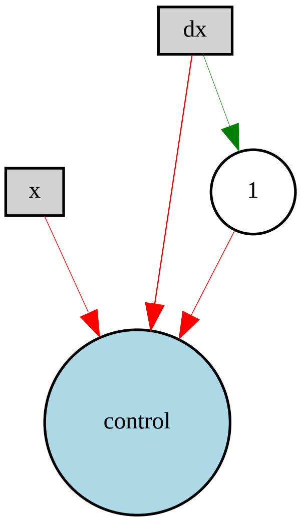
(a) CTRNN output over 25.6s from winner 5. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



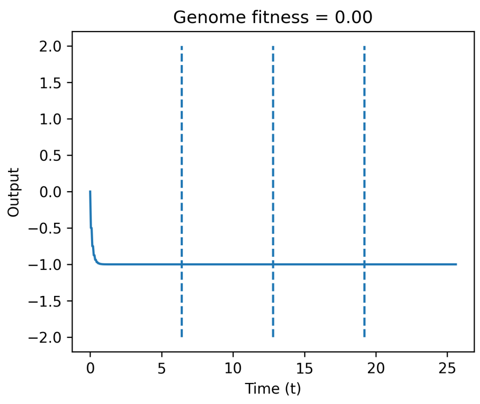
(b) Network topology of winner 5.



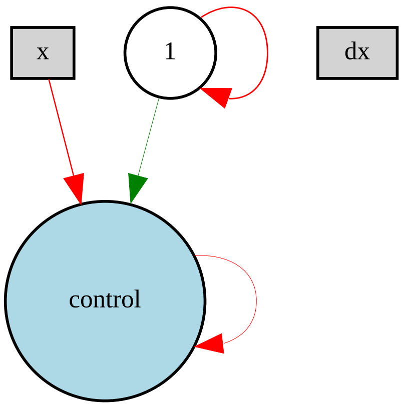
(a) CTRNN output over 25.6s from winner 6. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



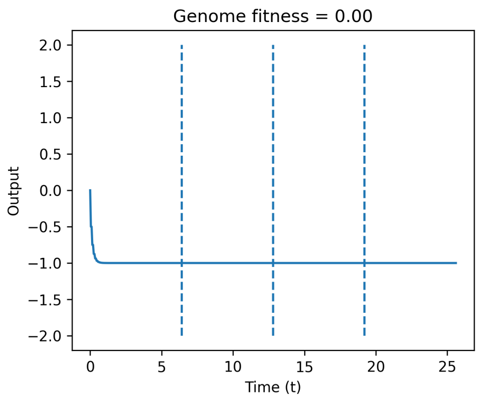
(b) Network topology of winner 6.



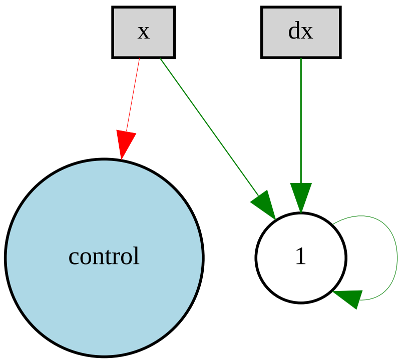
(a) CTRNN output over 25.6s from winner 7. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



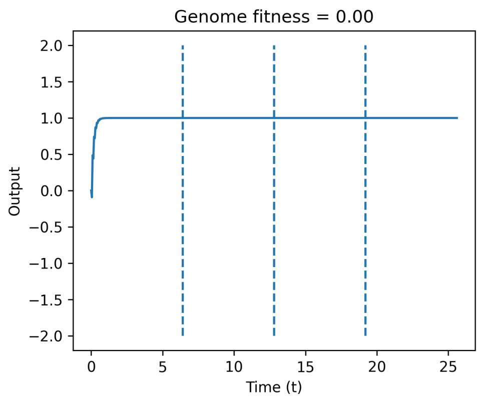
(b) Network topology of winner 7.



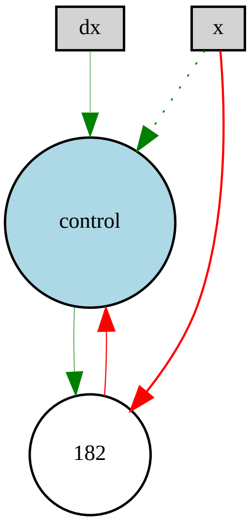
(a) CTRNN output over 25.6s from winner 8. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



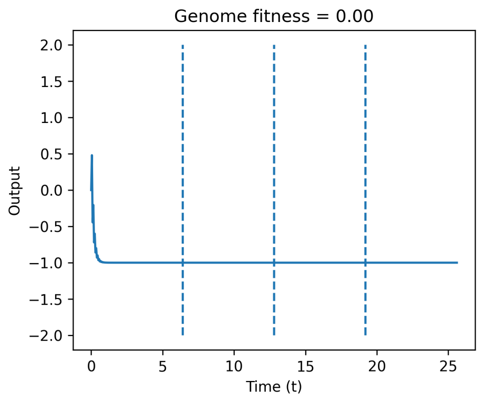
(b) Network topology of winner 8.



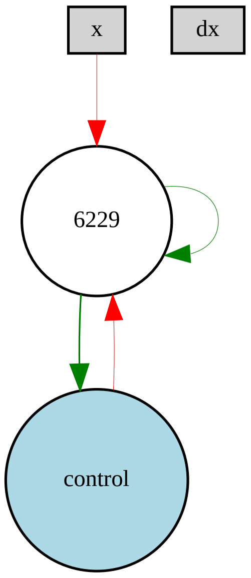
(a) CTRNN output over 25.6s from winner 9. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 9.

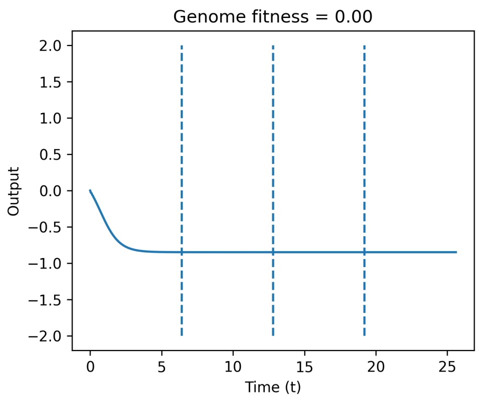


(a) CTRNN output over 25.6s from winner 10. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.

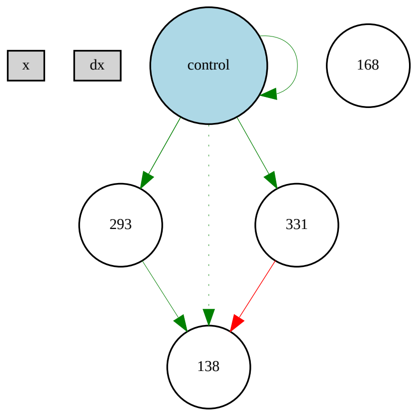


(b) Network topology of winner 10.

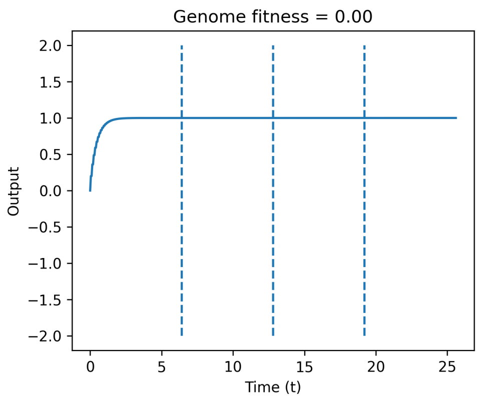
A.1.13 Default Genome Configuration D – NEE



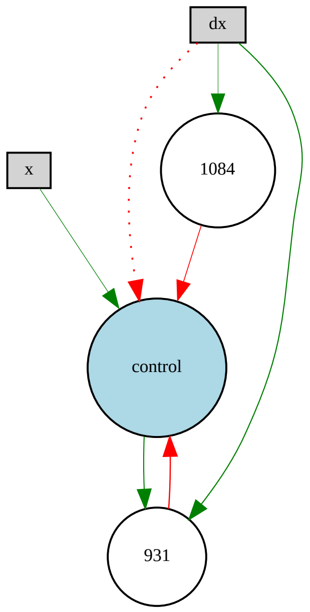
(a) CTRNN output over 25.6s from winner 1. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



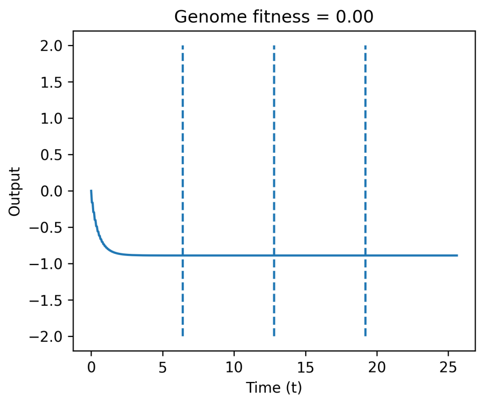
(b) Network topology of winner 1.



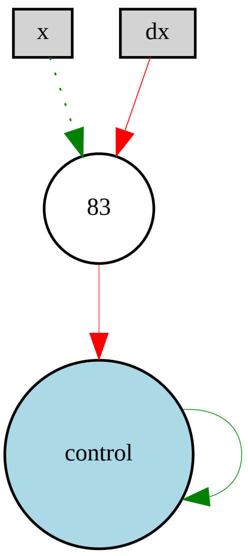
(a) CTRNN output over 25.6s from winner 2. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



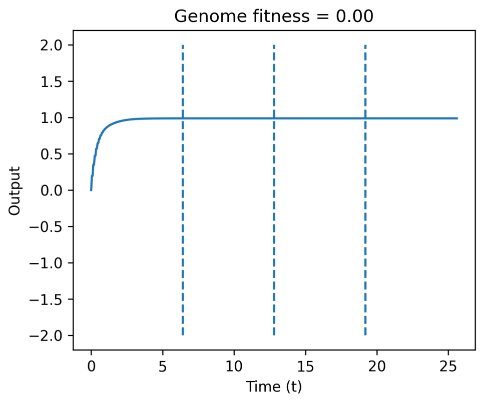
(b) Network topology of winner 2.



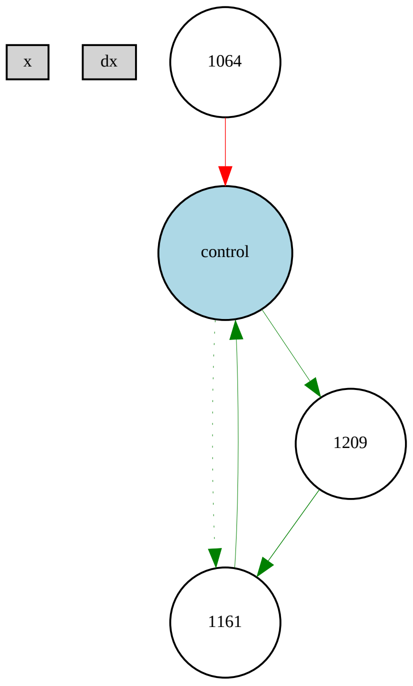
(a) CTRNN output over 25.6s from winner 3. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



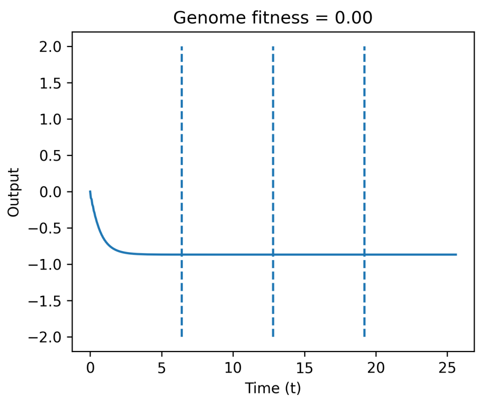
(b) Network topology of winner 3.



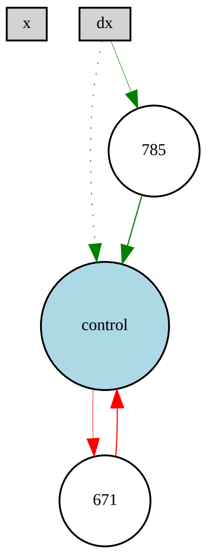
(a) CTRNN output over 25.6s from winner 4. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



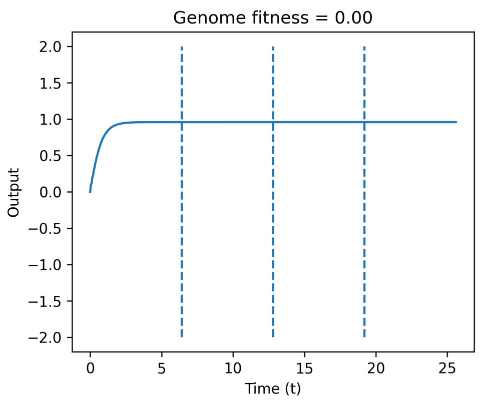
(b) Network topology of winner 4.



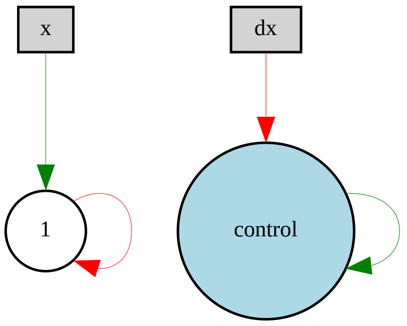
(a) CTRNN output over 25.6s from winner 5. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



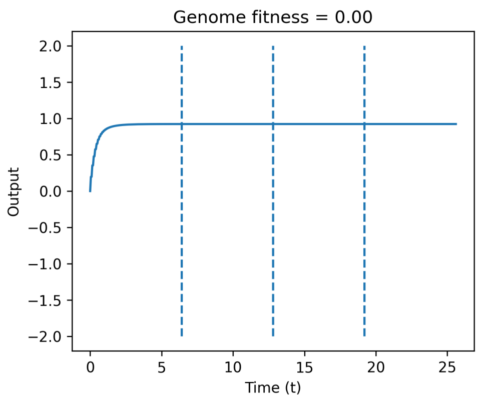
(b) Network topology of winner 5.



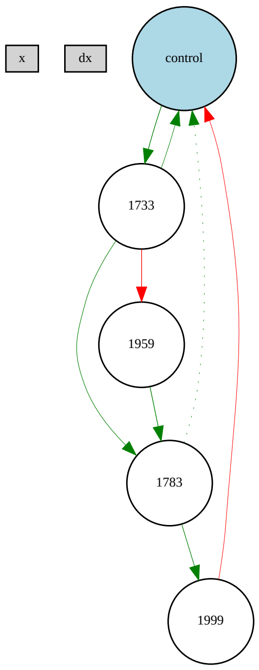
(a) CTRNN output over 25.6s from winner 6. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



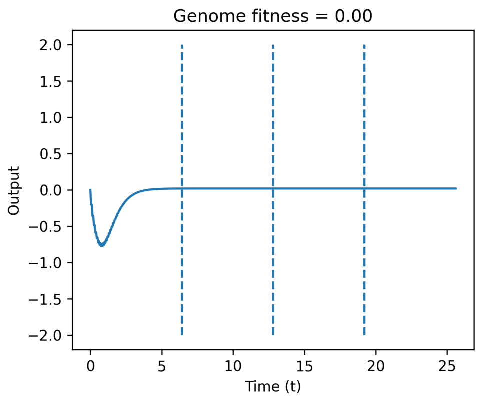
(b) Network topology of winner 6.



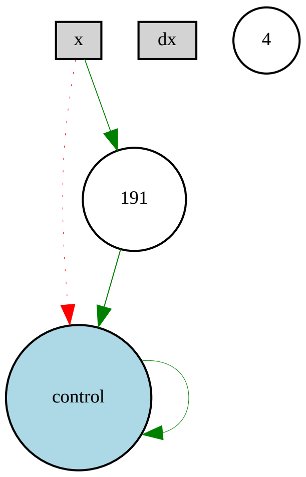
(a) CTRNN output over 25.6s from winner 7. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



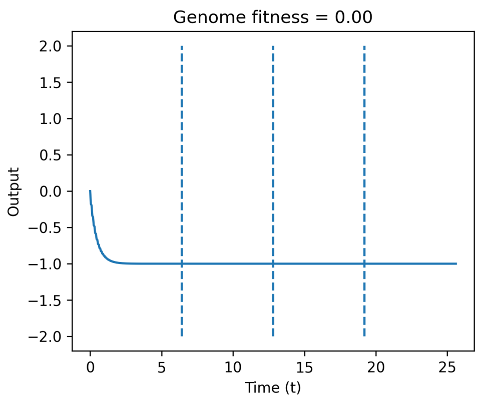
(b) Network topology of winner 7.



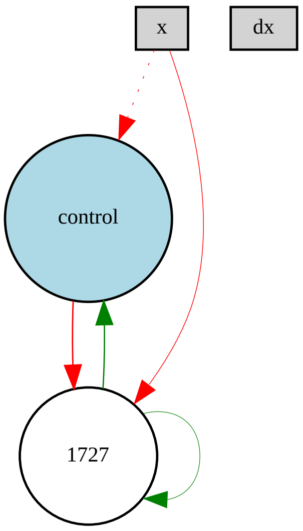
(a) CTRNN output over 25.6s from winner 8. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



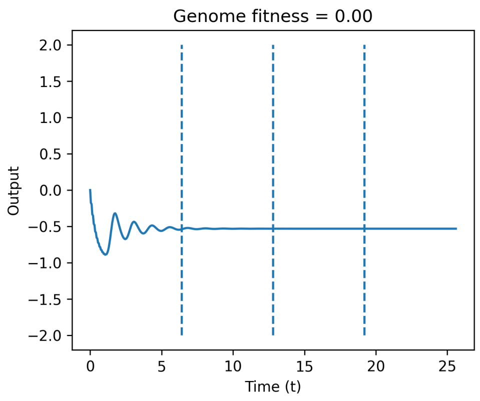
(b) Network topology of winner 8.



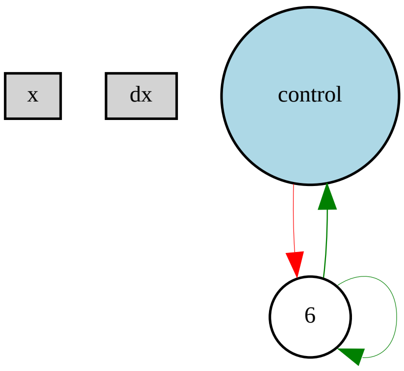
(a) CTRNN output over 25.6s from winner 9. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 9.

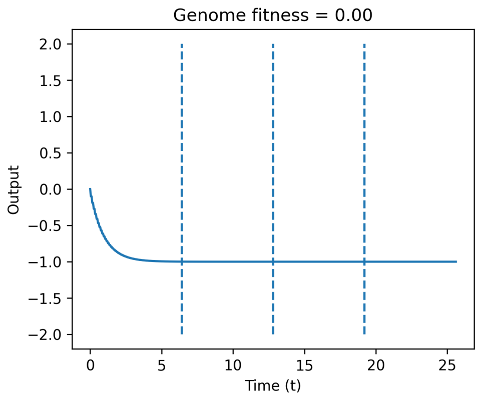


(a) CTRNN output over 25.6s from winner 10. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.

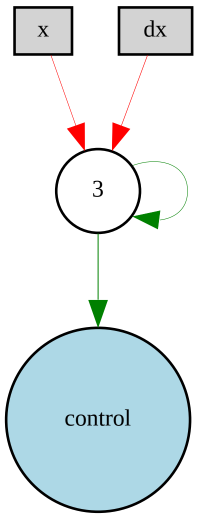


(b) Network topology of winner 10.

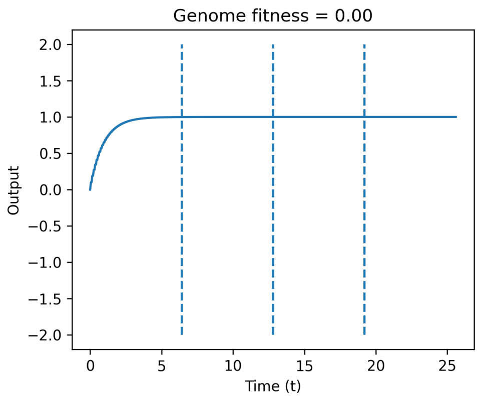
A.1.14 Default Genome Configuration E - NEE



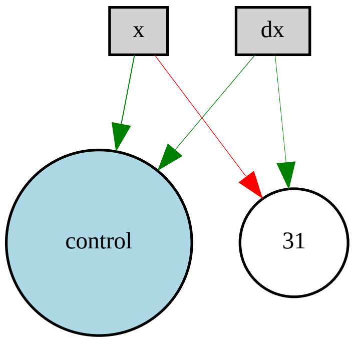
(a) CTRNN output over 25.6s from winner 1. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



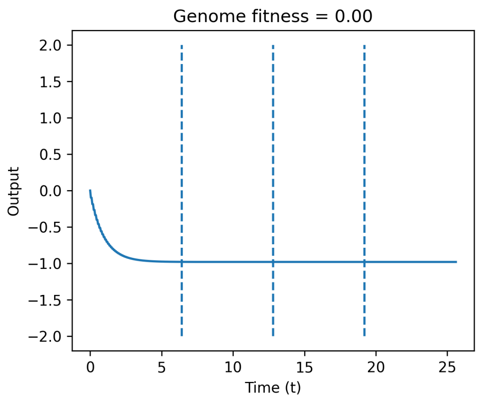
(b) Network topology of winner 1.



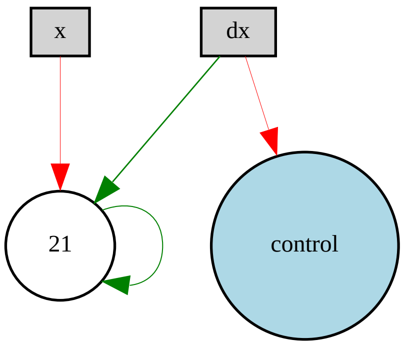
(a) CTRNN output over 25.6s from winner 2. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



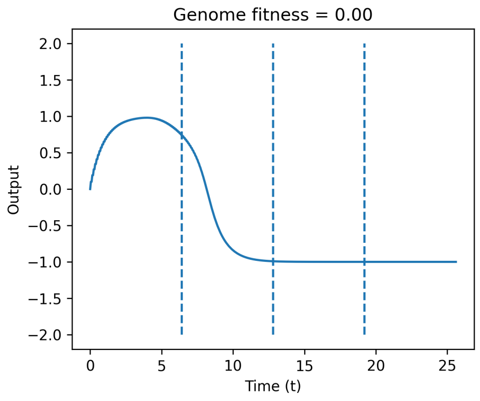
(b) Network topology of winner 2.



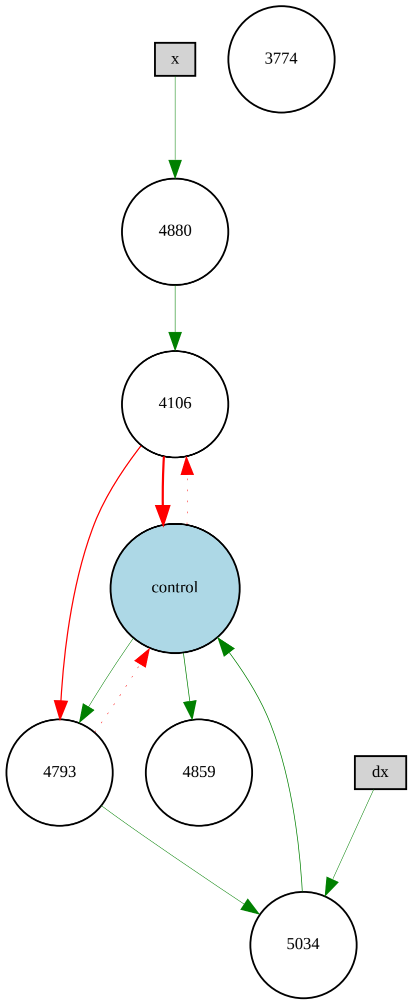
(a) CTRNN output over 25.6s from winner 3. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



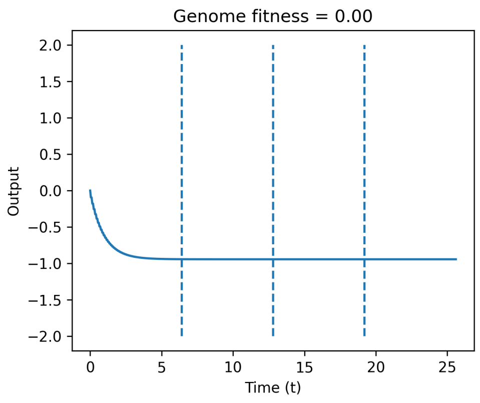
(b) Network topology of winner 3.



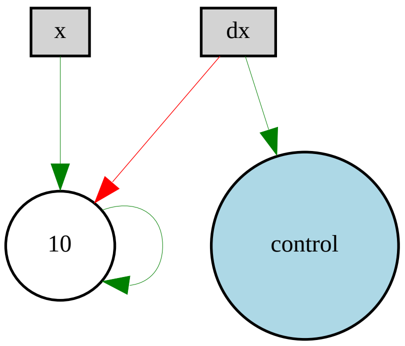
(a) CTRNN output over 25.6s from winner 4. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



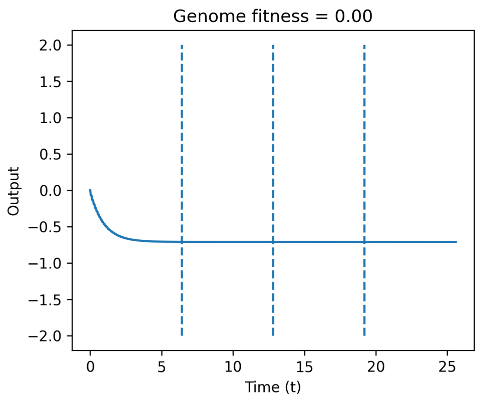
(b) Network topology of winner 4.



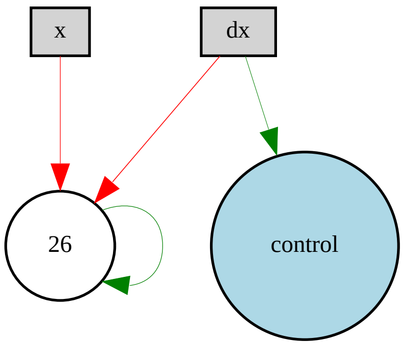
(a) CTRNN output over 25.6s from winner 5. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



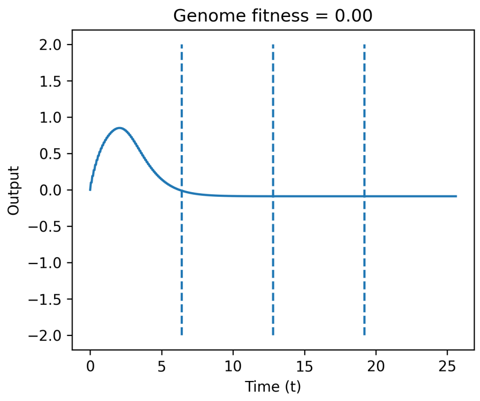
(b) Network topology of winner 5.



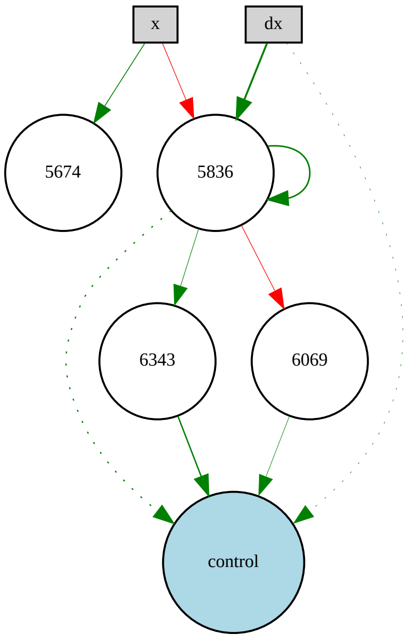
(a) CTRNN output over 25.6s from winner 6. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



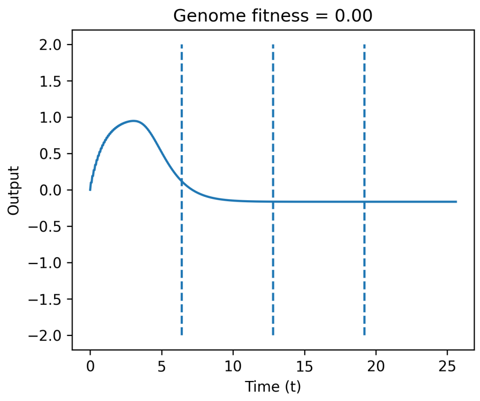
(b) Network topology of winner 6.



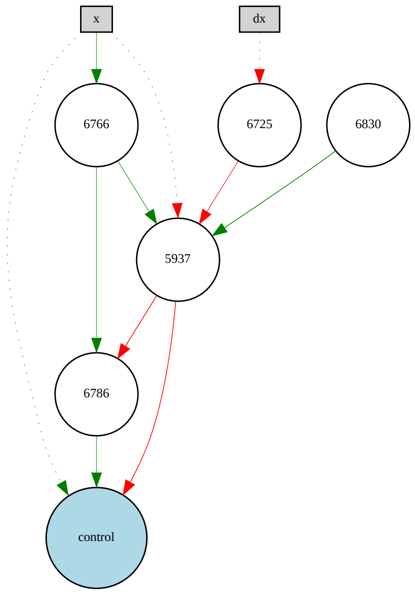
(a) CTRNN output over 25.6s from winner 7. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



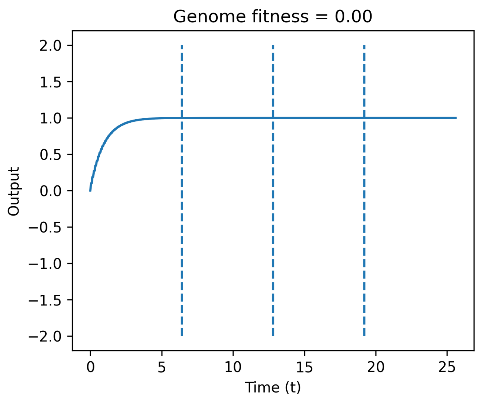
(b) Network topology of winner 7.



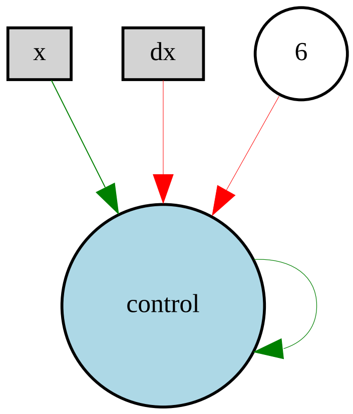
(a) CTRNN output over 25.6s from winner 8. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



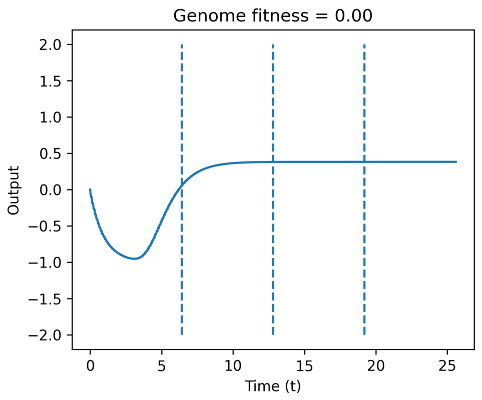
(b) Network topology of winner 8.



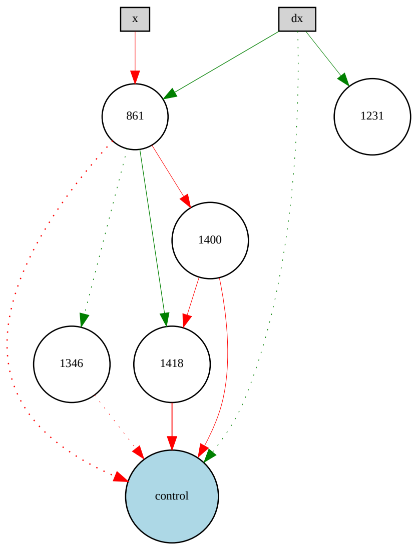
(a) CTRNN output over 25.6s from winner 9. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 9.

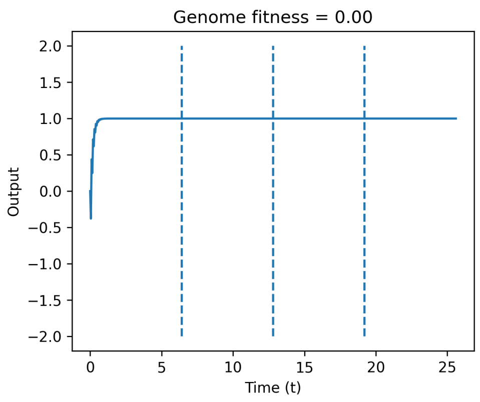


(a) CTRNN output over 25.6s from winner 10. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.

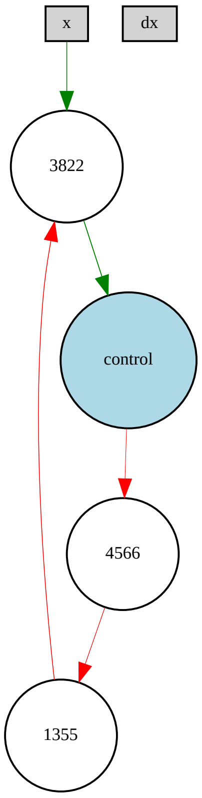


(b) Network topology of winner 10.

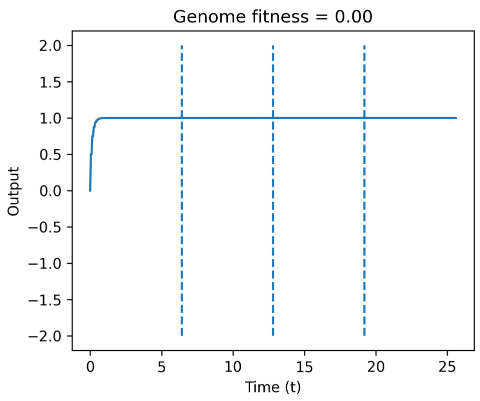
A.1.15 Default Genome Configuration F - NEE



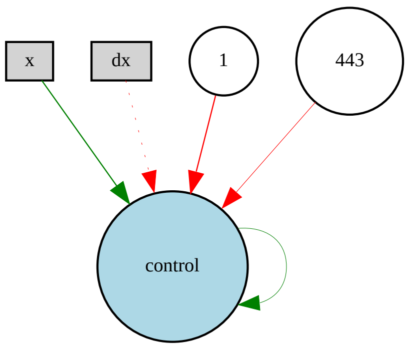
(a) CTRNN output over 25.6s from winner 1. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



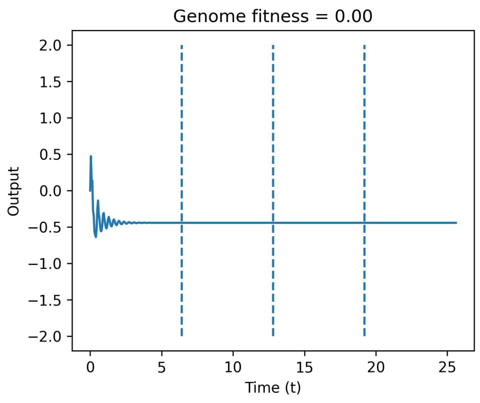
(b) Network topology of winner 1.



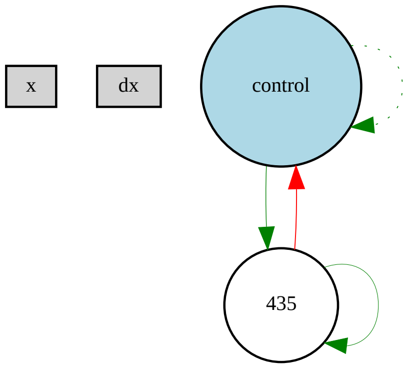
(a) CTRNN output over 25.6s from winner 2. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



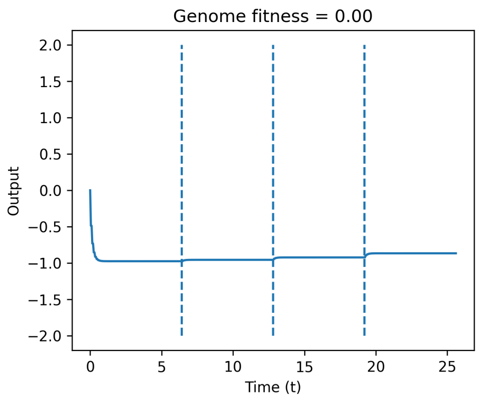
(b) Network topology of winner 2.



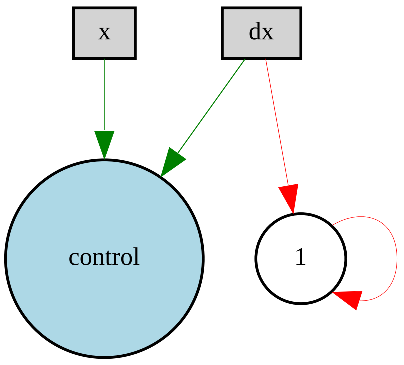
(a) CTRNN output over 25.6s from winner 3. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



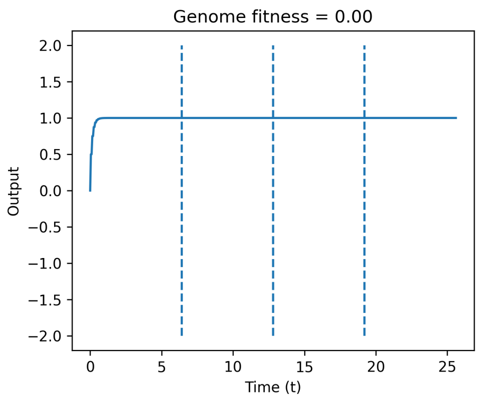
(b) Network topology of winner 3.



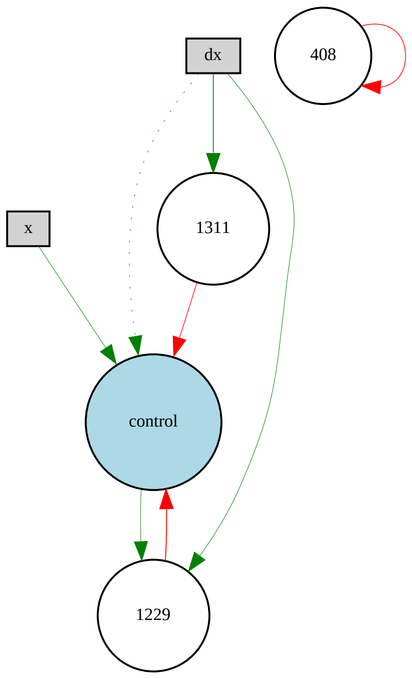
(a) CTRNN output over 25.6s from winner 4. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



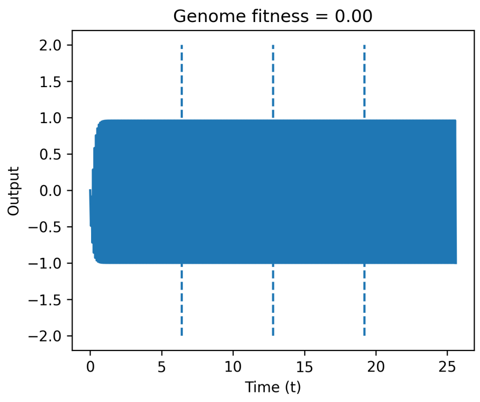
(b) Network topology of winner 4.



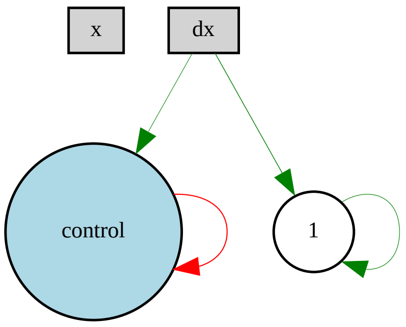
(a) CTRNN output over 25.6s from winner 5. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



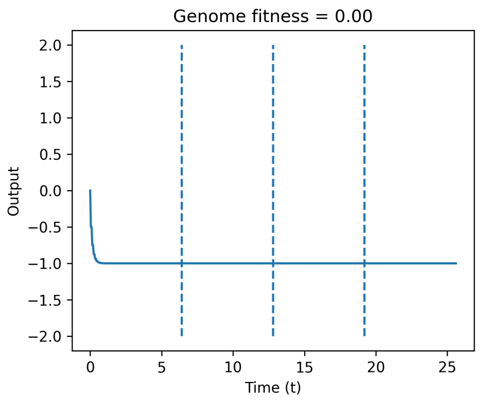
(b) Network topology of winner 5.



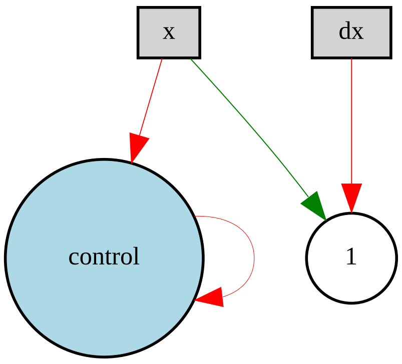
(a) CTRNN output over 25.6s from winner 6. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



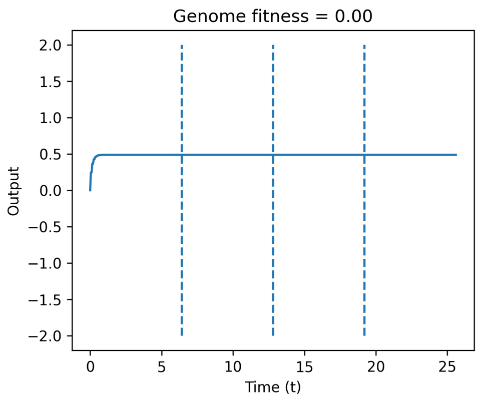
(b) Network topology of winner 6.



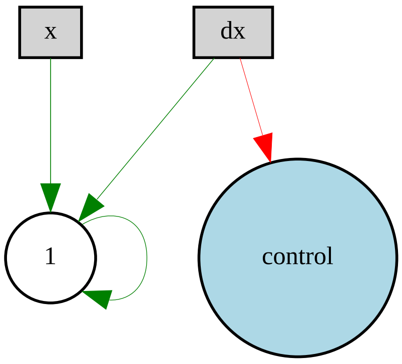
(a) CTRNN output over 25.6s from winner 7. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



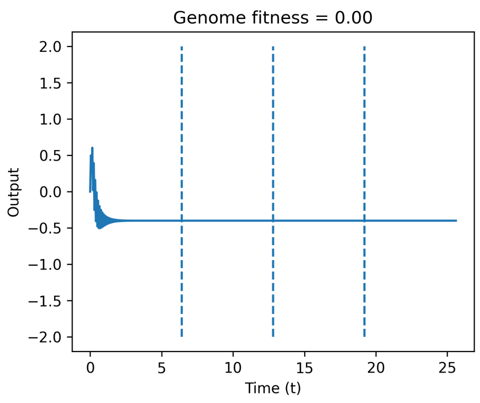
(b) Network topology of winner 7.



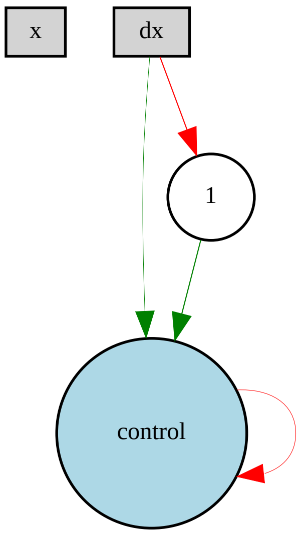
(a) CTRNN output over 25.6s from winner 8. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



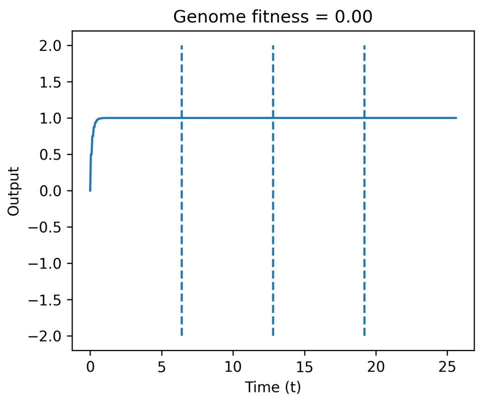
(b) Network topology of winner 8.



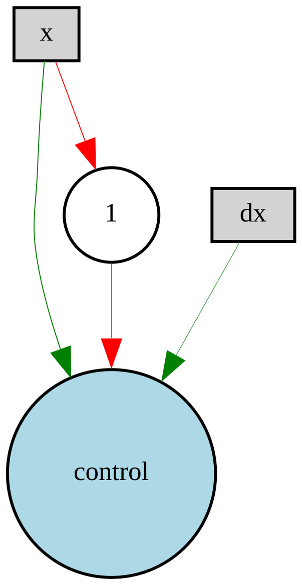
(a) CTRNN output over 25.6s from winner 9. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 9.

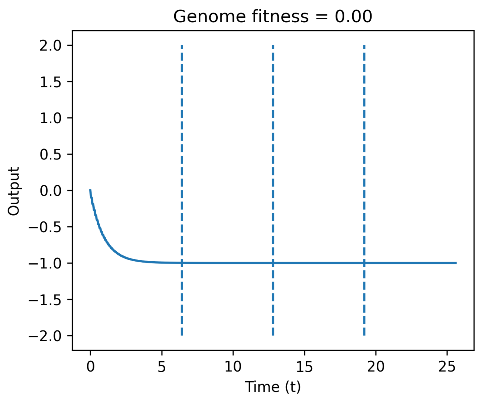


(a) CTRNN output over 25.6s from winner 10. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.

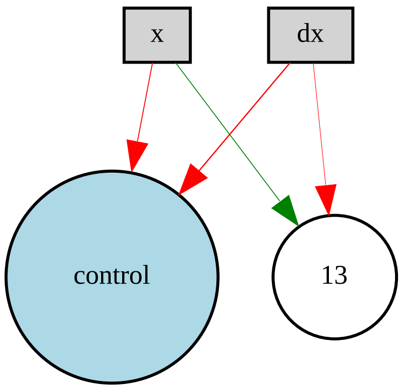


(b) Network topology of winner 10.

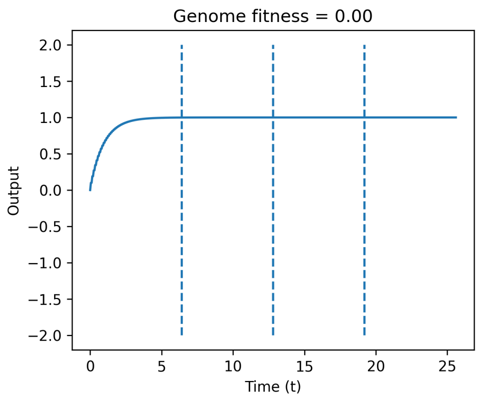
A.1.16 Default Genome Configuration G - NEE



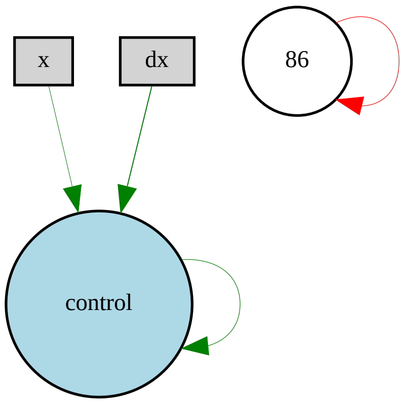
(a) CTRNN output over 25.6s from winner 1. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



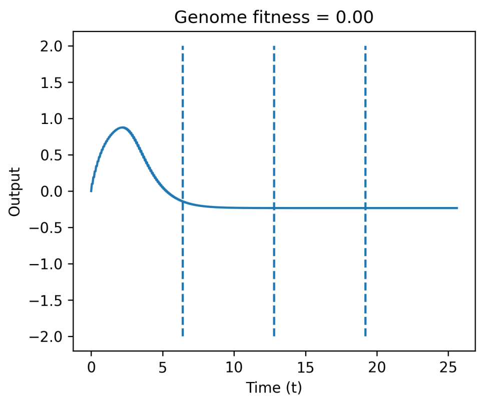
(b) Network topology of winner 1.



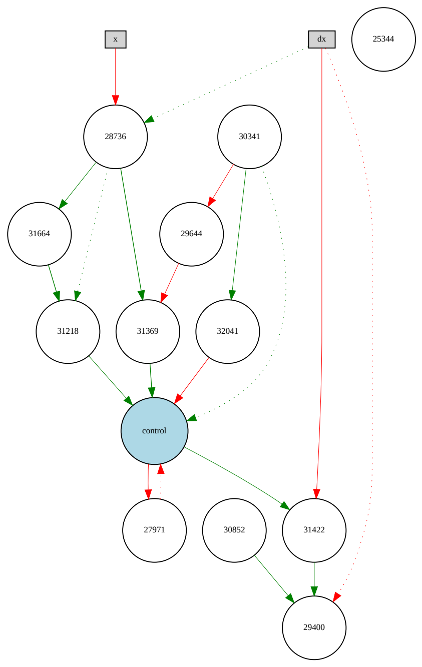
(a) CTRNN output over 25.6s from winner 2. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



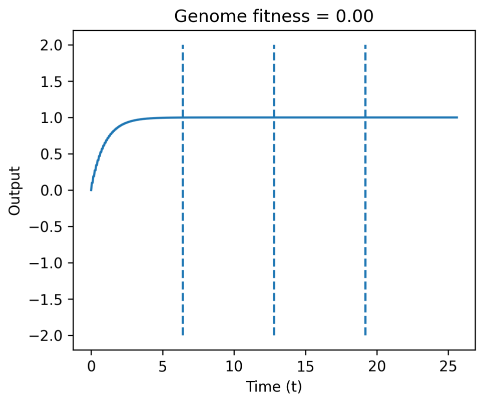
(b) Network topology of winner 2.



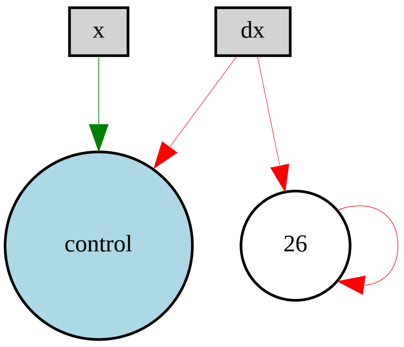
(a) CTRNN output over 25.6s from winner 3. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



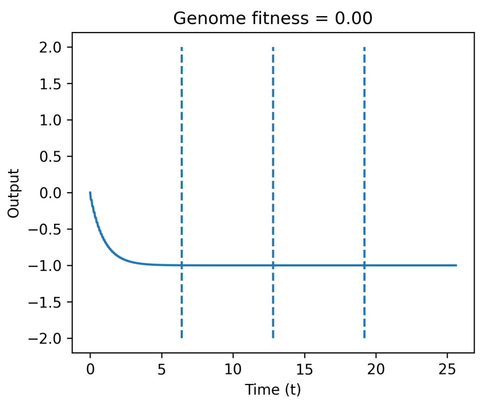
(b) Network topology of winner 3.



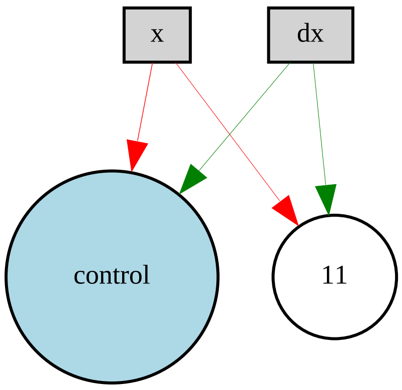
(a) CTRNN output over 25.6s from winner 4. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



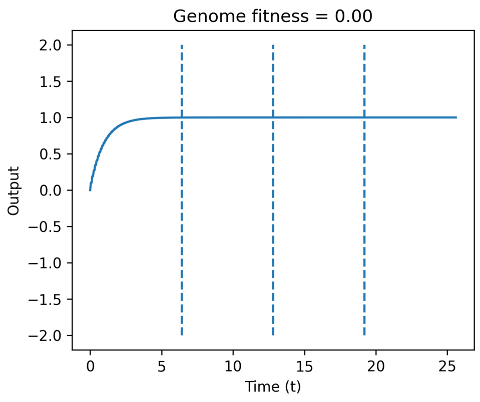
(b) Network topology of winner 4.



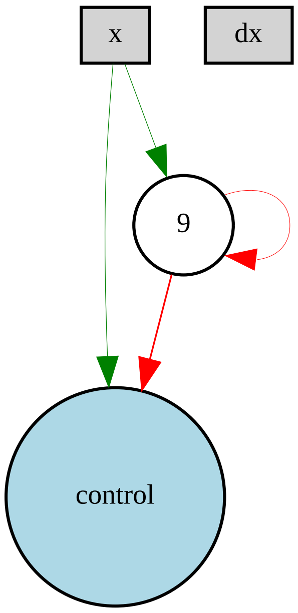
(a) CTRNN output over 25.6s from winner 5. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



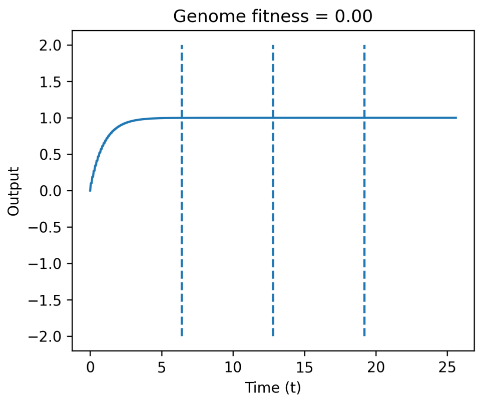
(b) Network topology of winner 5.



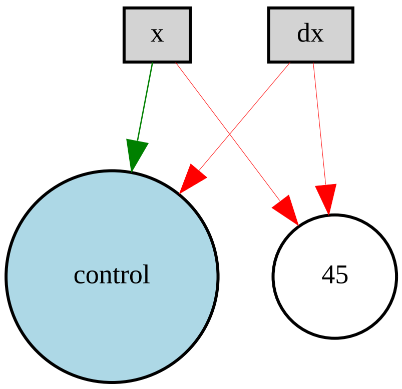
(a) CTRNN output over 25.6s from winner 6. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



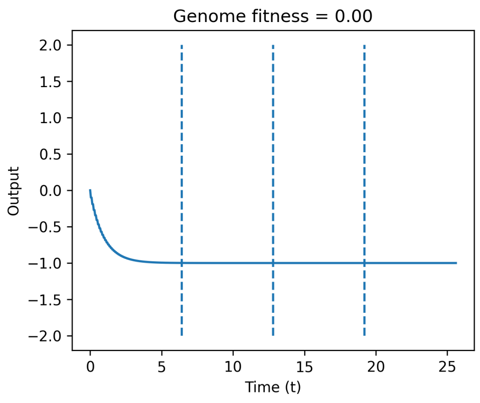
(b) Network topology of winner 6.



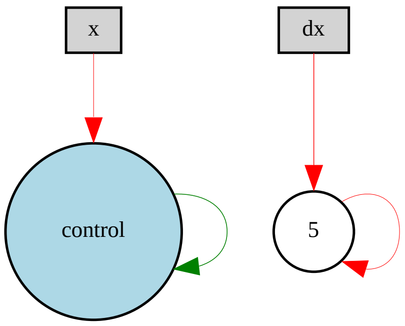
(a) CTRNN output over 25.6s from winner 7. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



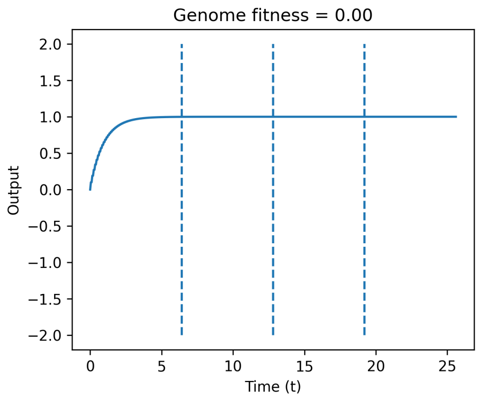
(b) Network topology of winner 7.



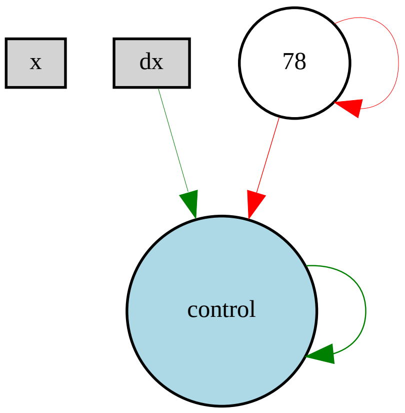
(a) CTRNN output over 25.6s from winner 8. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



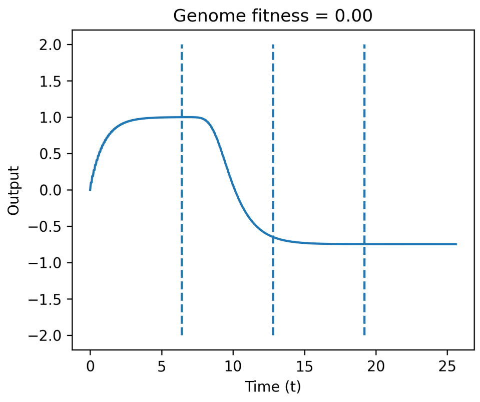
(b) Network topology of winner 8.



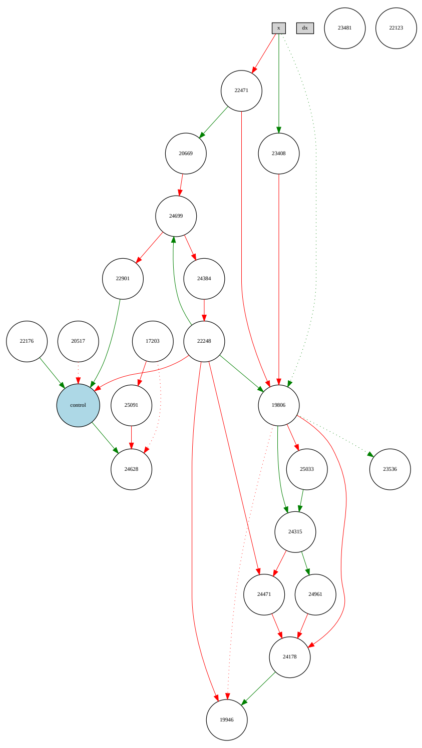
(a) CTRNN output over 25.6s from winner 9. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 9.



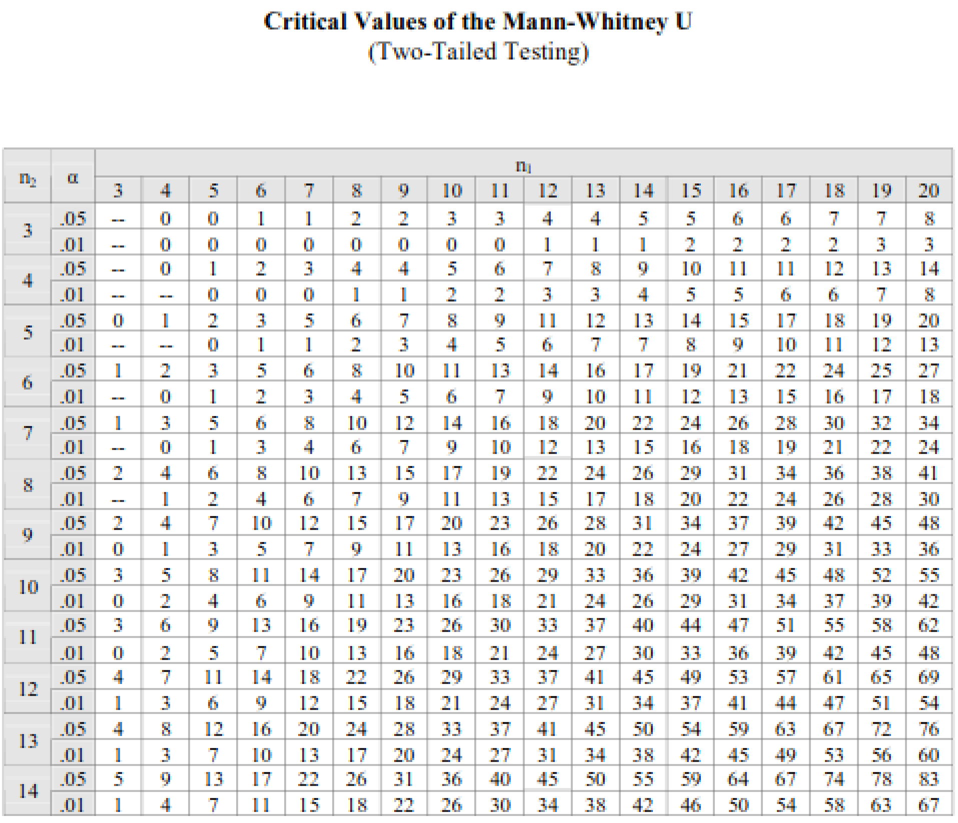
(a) CTRNN output over 25.6s from winner 10. Dashed lines separate regions for the input control c. 1st quarter $c = 25$, 2nd quarter $c = 50$, 3rd quarter $c = 75$, 4th quarter $c = 100$.



(b) Network topology of winner 10.

A.2 Appendix II - Critical values for Mann-Whitney U Test

Appendix II contains the table of critical values used in the Mann-Whitney U test to reject or confirm the null hypothesis H_0. In all experimental paradigms to which this test was applied, the samples were of equal length, $n_1 = n_2 = 10$. To reject H_0, the critical value must be lower than the least of the two test statistics $U1$ and $U2$ obtained from the test. The two possibilities for α, 0.5 and 0.1 represent 95% and 99% confidence intervals respectively.



Critical Values of the Mann-Whitney U
(Two-Tailed Testing)

n_2	α	n_1																	
		3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
3	.05	--	0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8
	.01	--	0	0	0	0	0	0	0	0	1	1	1	2	2	2	2	3	3
4	.05	--	0	1	2	3	4	4	5	6	7	8	9	10	11	11	12	13	14
	.01	--	--	0	0	0	1	1	2	2	3	3	4	5	5	6	6	7	8
5	.05	0	1	2	3	5	6	7	8	9	11	12	13	14	15	17	18	19	20
	.01	--	--	0	1	1	2	3	4	5	6	7	7	8	9	10	11	12	13
6	.05	1	2	3	5	6	8	10	11	13	14	16	17	19	21	22	24	25	27
	.01	--	0	1	2	3	4	5	6	7	9	10	11	12	13	15	16	17	18
7	.05	1	3	5	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34
	.01	--	0	1	3	4	6	7	9	10	12	13	15	16	18	19	21	22	24
8	.05	2	4	6	8	10	13	15	17	19	22	24	26	29	31	34	36	38	41
	.01	--	1	2	4	6	7	9	11	13	15	17	18	20	22	24	26	28	30
9	.05	2	4	7	10	12	15	17	20	23	26	28	31	34	37	39	42	45	48
	.01	0	1	3	5	7	9	11	13	16	18	20	22	24	27	29	31	33	36
10	.05	3	5	8	11	14	17	20	23	26	29	33	36	39	42	45	48	52	55
	.01	0	2	4	6	9	11	13	16	18	21	24	26	29	31	34	37	39	42
11	.05	3	6	9	13	16	19	23	26	30	33	37	40	44	47	51	55	58	62
	.01	0	2	5	7	10	13	16	18	21	24	27	30	33	36	39	42	45	48
12	.05	4	7	11	14	18	22	26	29	33	37	41	45	49	53	57	61	65	69
	.01	1	3	6	9	12	15	18	21	24	27	31	34	37	41	44	47	51	54
13	.05	4	8	12	16	20	24	28	33	37	41	45	50	54	59	63	67	72	76
	.01	1	3	7	10	13	17	20	24	27	31	34	38	42	45	49	53	56	60
14	.05	5	9	13	17	22	26	31	36	40	45	50	55	59	64	67	74	78	83
	.01	1	4	7	11	15	18	22	26	30	34	38	42	46	50	54	58	63	67

Figure A.161: Table of critical values for the Mann-Whitney U test.

Bibliography

[1] Devolvé et al. “Fictive rhythmic motor patterns induced by NMDA in an in vitro brain stem-spinal cord preparation from an adult urodele.” In: *Journal of Neurophysiology* 18 (1999), pp. 1074–1077.

[2] Aude Billard and Auke J. Ijspeert. “Biologically inspired neural controllers for motor control in a quadruped robot.” English. In: *Proceedings of the International Joint Conference on Neural Networks*. Vol. 6. Cited By :58. 2000, pp. 637–641. URL: www.scopus.com.

[3] T.G Brown. “On the nature of the fundamental activity of the nervous centres; together with an analysis of the conditioning of rhythmic activity in progression, and a theory of the evolution of function in the nervous system.” In: *Journal of Physiology* (1914), pp. 18–46.

[4] C.A. Coello Coello. “Evolutionary multi-objective optimization: a historical view of the field.” In: *IEEE Computational Intelligence Magazine* 1.1 (2006), pp. 28–36. DOI: 10.1109/MCI.2006.1597059.

[5] Cohen and Wallen. “The neural correlate of locomotion in fish. fictive swimming “induced in a in vitro preparation of the lamprey spinal cord”.” In: *Experimental Brain Research* 80 (1980), pp. 11–18.

[6] James W. Cooley, Peter A. W. Lewis, and Peter D. Welch. “The Fast Fourier Transform and its Applications.” In: *IEEE Transactions on Education* Vol. 12, Issue 1, p. 27-34 (1969). DOI: https://doi.org/10.1109/TE.1969.4320436.

[7] Charles Darwin. “On the origin of species.” In: 1st ed. London: Routledge, 2003. DOI: https://doi.org/10.4324/9780203509104.

[8] Auke Jan Ijspeert. “Central pattern generators for locomotion control in animals and robots: A review.” In: *Neural Networks* Vol. 21, Issue 4, p. 642-653 (2008). DOI: https://doi.org/10.1016/j.neunet.2008.03.014.

[9] Auke Jan Ijspeert and Jérôme Kodjabachian. “Evolution and Development of a Central Pattern Generator for the Swimming of a Lamprey.” In: *Artificial Life* 5.3 (July 1999), pp. 247–269.

[10] D. Lachat, A Crespi, and Auke J. Ijspeert. “Boxybot: A swimming crawling fish robot controlled by a central pattern generator.” In: *Proceedings of the first IEEE/RAS-EMBS international conference on biomedical robotics*. 2006.

[11] Dimitris G. Manolakis and Vinay K. Ingle. “Applied Digital Signal Processing.” In: 1st ed. New York, New York, USA: Cambridge University Press, 2012. ISBN: 9780521110020.

[12] Claudio Mattiussi and Dario Floreano. “Analog Genetic Encoding for the Evolution of Circuits and Networks.” In: *IEEE Transactions on Evolutionary Computation* 11.5 (2007), pp. 596–607. DOI: 10.1109/TEVC.2006.886801.

[13] McClellan and Jang. “Mechanosensory inputs to the central pattern generators for locomotion in the lamprey spinal cord: Resetting, entrainment, and computer modeling.” In: *ournal of Neurophysiology* vol. 70, Issue 6 (1993), pp. 161–166.

[14] Alan McIntyre et al. *neat-python*. https : / / github . com / CodeReclaimers/neat-python.

[15] S. Risi and K.O Stanley. “An Enhanced Hypercube-Based Encoding for Evolving the Placement, Density, and Connectivity of Neurons.” In: *Artificial Life* Vol. 18, Issue 4 (2012).

[16] José Santos and Ándel Campo. “Biped locomotion control with evolved adaptive center-crossing continuous time recurrent neural networks.” In: *Neurocomputing* Vol. 86, p. 86-96 (2012).

[17] Jiang Shan, Cheng Junshi, and Chen Jiapin. “Design of central pattern generator for humanoid robot walking based on multi-objective GA.” In: *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat.*

No.00CH37113). Vol. 3. 2000, 1930–1935 vol.3. DOI: 10.1109/IROS.2000.895253.

[18] Kenneth O. Stanley and Risto Miikkulainen. “Evolving Neural Networks through Augmenting Topologies.” In: *Evolutionary Computation* Vol. 10, Issue 2, p. 99-127 (2002). DOI: 10.1162/106365602320169811. URL: https://ieeexplore.ieee.org/abstract/document/6790655.

[19] Duc Trong Tran et al. “Central pattern generator based reflexive control of quadruped walking robots using a recurrent neural network.” In: *Robotics and Autonomous Systems* Volume 62, Issue 10 (2014).

[20] Ronald J. Williams and David Zipser. “A Learning Algorithm for Continually Running Fully Recurrent Neural Networks.” In: *Neural Computation* 1.2 (June 1989), pp. 270–280.