AN ABSTRACT OF THE THESIS OF

Andrew Gabler for the degree of Master of Science in Robotics presented on June 17, 2015.

Title: Learning-based Control of Experimental Hybrid Fuel Cell Power Plant

Abstract approved: _____

Kagan Tumer

Direct fired Solid Oxide Fuel Cell (SOFC) Turbine hybrid plants have the potential to dramatically increase power plant efficiency, decrease emissions, and provide fast response to transient loads. The US Department of Energy's (DOE) Hybrid Performance Project is an experimental hybrid SOFC plant, built at the National Energy Technology Laboratory (NETL). One of the most significant challenges in the development and commercialization of this plant is control. Traditional control techniques are inadequate for this plant due to poor system models, high nonlinearities, and extreme coupling between state variables. Learning-based control techniques do not require explicit system models, and are well suited for controlling nonlinear and highly coupled systems. In this work, we use neuro-evolutionary control algorithms to develop a set-point controller for fuel cell flow rate in this plant, and demonstrate a controller that can accurately track a desired turbine speed profile within 50 RPM, even in the presence of 10% sensor noise. In order to

ensure the neuro-evolutionary algorithm is computationally tractable, we develop a computationally efficient neural network simulator of the plant, using data collected from actual plant operation. We also present an iterative method to improve plant the controller and simulation performance based on plant run data allowing for expansion of the operation range of the plant in simulation, and control the plant for high efficiency operation.

Learning-based Control of Experimental Hybrid Fuel Cell Power Plant

by

Andrew Gabler

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 17, 2015
Commencement June 2016

Master of Science thesis of Andrew Gabler presented on June 17, 2015.

APPROVED:

Major Professor, representing Robotics

Head of the School of Mechanical, Industrial, and Manufacturing Engineering

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Andrew Gabler, Author

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Kagan Tumer, for his guidance and support of this project. Also, Dr. Mitchell Colby for his help with all of the code involved in this work, as well as the other members of the AADI Laboratory, who spent many hours in aiding me. I would also like to acknowledge the funding agency, The US Department of Energy's National Energy Technology Laboratory for making this work possible under grant DE-FE0012302. I would also like to thank David Tucker and his team at NETL for providing us with access to the power plant data and giving us their time showing us the power plant. Lastly, I would like to thank Melissa Gabler for her continual support throughout my studies, and without whom I would have completed this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

Chapter 1 Introduction

Difficult real world problems, from air traffic control to self-driving cars, from computer networking to power generation and distribution all require increasingly complex systems as the demand for use increases. More and more often, these complex systems require complex control decisions to regulate and optimize the performance of the system. Implementation of complex systems has lead to distributed systems, hybrid technologies, and large amounts of data, yet the interactions in the complex system are difficult to model and are often poorly understood. Nonlinearites, system noise, stochastic dynamics, coupled control actions, and heterogeneous technologies make control of these complex systems difficult with traditional control methods.

Learning based control methods for these complex systems have been successful in many of these challenging domains, such as air traffic control [1], planetary exploration [18], computer networking [36], power grid managment [28], UAV coordination [5], micro air vehicle control [32], fin-less rocket guidance [12], satellite vibration elimination [24], nonlinear control for robots [13, 11, 15], power plant boiler control [22] and many other real world applications [6, 8, 9, 26, 31]. By taking advantage of statistical models, machine learning applied to control has the ability to outperform many of the current state of the art control. Because models are limited and subtle, complex interactions in the real world are difficult to find or

quantify, traditional control methods are difficult to implement and limited in their use for these systems. Learning based control can offer insights into the dynamics of the system while providing a solution to these control problems.

One example of one of a complex system which is difficult to model and control is the US Department of Energy's Hybrid Performance Project (HyPer), a hybrid power plant which incorporates both a gas turbine power cycle and a solid oxide fuel cell. The increasing demand for efficient, clean, and cost-effective energy generation sources has created a desire for energy sources which are thermally efficient and flexible to rapidly changing load demands. New advanced and experimental power plants are being designed to meet these demands, and many new power plant designs incorporate many different types of technologies into their designs. The HyPer plant's hybrid configuration creates challenges in modeling and control [30].

HyPer is a hybrid fuel cell turbine plant in a direct fired configuration, a design which offers many benefits. First, the turbine can recover waste heat from the fuel cell which operates at high temperatures, resulting in high system efficiency. Second, the fuel cell can act as a "thermal battery," allowing for large amounts of heat to quickly be transferred to the turbine. The design of HyPer allows the fuel cell to operate as a baseline energy source, and the turbine can then quickly respond to transient loads [42].

The single largest challenge in the development of HyPer is determining how to adequately control the plant. Due to the hybrid nature of the plant, there are highly nonlinear system dynamics and a high degree of coupling between state variables. As a result, accurate physics based models of HyPer have been difficult

or impossible to develop. Furthermore, empirical system models such as empirical transfer functions are difficult to obtain, and typically only describe limited operating regimes [37]. Without an accurate model of the HyPer facility, traditional model-based control techniques are inadequate for control of HyPer.

Learning-based controls, such as neuroevolution, offer advantages over traditional controllers since they are well suited to nonlinear control, reject noise, and require no explicit system models [13, 14]. However, learning-based controllers require a large number of calls to a simulator in order to evaluate different control policies, meaning that a fast system simulator is required for these algorithms to be computationally tractable. Current models of HyPer (both analytical and empirical) are inaccurate and computationally expensive, and are thus inadequate for training a learning-based controller.

In this paper, we develop a fast and computationally efficient simulator of the HyPer facility, by training a neural-network with data collected from physical HyPer experiments. This neural network can run orders of magnitude faster than real time, allowing for different controllers to be quickly and efficiently evaluated. We then use a neuro-evolutionary algorithm to develop a controller for HyPer, and the learned controller is capable of controlling the HyPer plant across several important variables. The specific contributions of this paper are outlined below.

1. Statistic based, learned model for the HyPer Facility which allows for efficient simulation of the power plant and accurately matches the plant dynamics.

2. Robust, evolved neural network controller capable of reaching arbitrary set

points for historic plant states.

3. Robust neural network control capable following an arbitrary turbine speed trajectory

4. Expanded simulation of the plant and control results using all plant control valves for high efficiency operation at a given state.

5. General method to expand the operation range of the power plant and improve the power plant simulation.

The rest of this paper is organized as follows. Chapter 2 describes background material and related work. Chapter 3 describes our plant simulator and neuro-evolutionary algorithm. Chapter 4 presents our experimental results. Chapter 5 describes the results of the expanded simulation and control. Finally, Chapter 6 discusses the results and concludes the paper.

Chapter 2 Background

In the following sections we describe the HyPer facility, current control methods, neuro-evolutionary algorithms, and fitness approximation.

2.1 The HyPer Facility

The Hybrid Performance Project facility, HyPer, is located at the Department of Energy's National Energy Technology Laboratory campus in Morgantown, West Virginia. The purpose of this experimental plant is to study the complex interactions of the direct fired SOFC hybrid configuration, and to develop controllers for this plant.

The HyPer project is a small scale SOFC hybrid hardware simulation, capable of emulating 300 to 900 kilowatt hybrid plants [42]. HyPer uses real power plant hardware in conjunction with a hardware in the loop simulation of a fuel cell to study the interactions in these hybrid fuel cell systems. The fuel cell simulation makes this plant unique in that it allows for a wide range of fuel cells to be simulated without additional cost, and controllers may be tested without risk of damaging a costly fuel cell. The plant can be reset in software rather than rebuilt in hardware, allowing for significant progress in the understanding of such hybrid configurations [34].

The standard Brayton cycle plant with regeneration is a fundamental building block of the HyPer plant. The Brayton cycle, shown in Figure 2.1 is the typical gas turbine model for power generation. Gas cycle turbines are desirable because they have fast start up times, can be built in a wide range of power outputs, and use readily available fuels such as natural gas [41]. In a typical Brayton cycle power

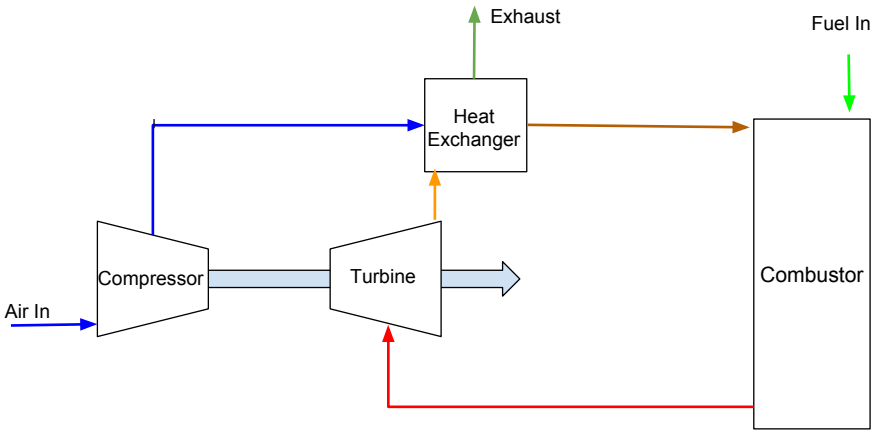

Figure 2.1: The Brayton cycle with regeneration. Air travels along the arrows shown. First air enters the compressor, then is preheated before the combustor. Hot air is then run through the turbine, and exhaust gasses are used to heat the heat exchanger. Typical controls (not shown in diagram) include fuel in, compressor bleed air valves and hot air bleed valves before the turbine. These valves control flow rate and temperature at critical points in the cycle such as the entrance to the turbine.

plant, atmospheric air is drawn through a compressor, the pressurized air is then mixed with fuel in a combustion chamber and the fuel is ignited adding thermal energy to the air. The exhaust gasses from the combustion chamber are then expanded through a gas turbine, generating mechanical work in a rotating shaft

used to drive the compressor and an electric generator. A standard gas turbine then vents the exhaust gasses from the turbine out into the atmosphere. In a turbine equipped with regeneration, the hot exhaust from the turbine is used to preheat the air entering the combustion chamber with a heat exchanger, reducing the amount of fuel required to heat the air.

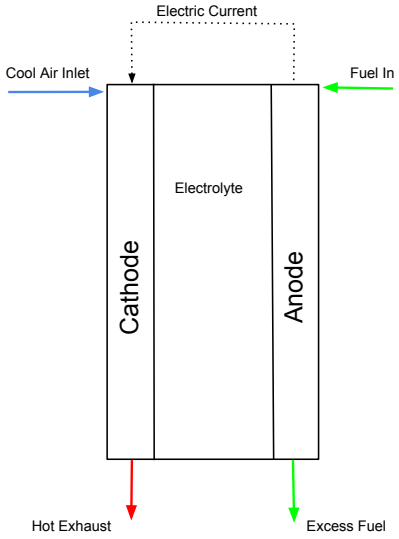

Figure 2.2: Fuel cell configuration. Fuel enters the anode of the fuel cell, air enters the cathode. Hydrogen ions contained in the fuel are transported across the ceramic electrolyte and creating an electric potential difference across the fuel cell.

Solid oxide fuel cells utilize a ceramic electrolyte to channel oxygen ions to react with hydrogen producing an electric current. Fuel cells operate at a very high efficiency, typically up to 60% and with low carbon emissions [42]. These fuel cells operate at a high temperature, reforming natural gas or other hydrocarbon

fuels to produce the hydrogen needed for the reaction, and ionizing the hydrogen and oxygen to be transported across the electrolyte. Temperatures can reach upwards of 600° C, much of which is wasted in the exhaust gas. Stand alone fuel cell generation facilities take significant time to heat and start up, limiting their use in applications requiring fast plant startup [41]. Pressurized air enters the cathode of the fuel cell, and fuel enters the anode. Hot exhaust leaves the fuel cell at a high temperature, with any unconverted fuel and other gasses.

They HyPer project places a fuel cell in between the regeneration heat exchangers and combustion chamber of the typical Brayton cycle shown above. Figure 2.3 on Page 11 shows the configuration of HyPer. Primary heat generation to run the turbine comes from the fuel cell exhaust, the combustion chamber burns any unspent fuel, assists in start up, and regulates turbine inlet temperature. Exhaust from the turbine runs through a set of parallel heat exchangers to preheat air into the fuel cell. 120 sensors are located across the plant designed to feed realtime information to the controller and log the state of the system during experiments.

Along with the addition of the fuel cell, three bypass air valves are used as actuators. These three valves in tandem with the fuel input are the controls to HyPer. By utilizing bypass air, temperature and pressure can be regulated at important locations in the system such as the entrance to the fuel cell or turbine. These bypass valves are described below:

- Cold Air Bypass: This valve brings cold air to the cathode of the fuel cell.
- Hot Air Bypass: This valve brings high pressure hot air directly from the

heat exchanges and sends it to the combustion chamber.

- Bleed Air Valve: Vents high pressure air from the compressor out to the atmosphere.

- Fuel Valve: Besides the three bypass valves, fuel rate into the anode side of the fuel cell. Some of the fuel is bled into the secondary combustor.

The HyPer plant uses a hardware simulation of the fuel cell. The hardware in the loop simulation of the fuel cell uses a high fidelity model of a fuel cell in software to control a combustion chamber with the flow and heat transfer characteristics of the real fuel cell. Thus, novel control techniques can be studied with low risk of damaging a costly fuel cell, while accurate interactions between plant components can still be observed. This hardware simulation of the fuel cell has been validated in the studies of the plant. By simulating the fuel cell in hardware, the interactions between fuel cell and turbine can be studied in detail [34].

System wide analytic models and software simulations have yet to capture authentic behavior of these interactions and limit the study of control systems for this plant. In the next section we discuss the current modeling and methods of control for this plant.

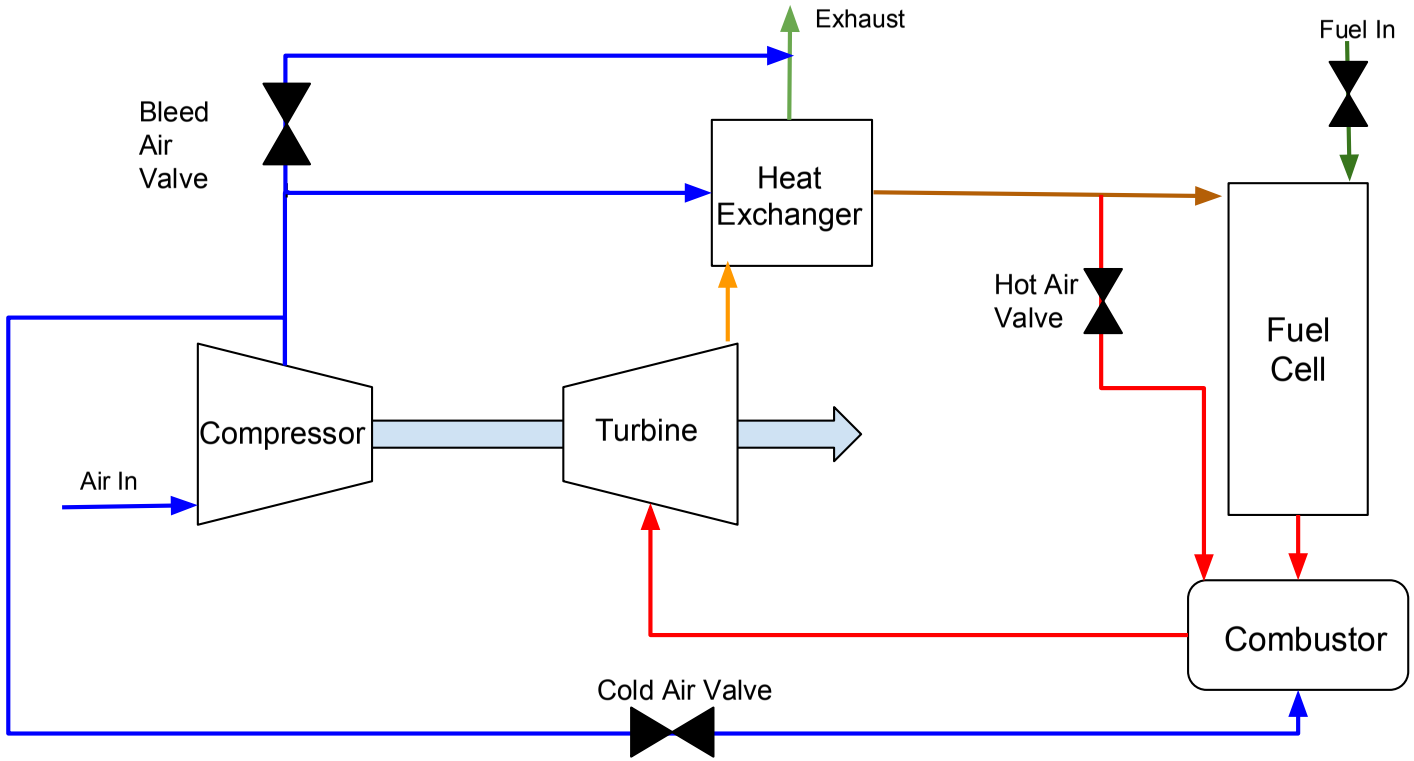

Figure 2.3: A simplified diagram of the HyPer facility. Air travels in the direction of the arrows shown first entering the compressor, then passing though a heat exchanger to preheat the air before the fuel cell. The fuel cell exhaust gasses and unused fuel are output to a small combustion chamber, and hot air is fed though the turbine. The exhaust heats the heat exchanger before exiting the system. Four control valves control the flow of HyPer and can regulate temperature, fuel, and flow rate into system components.

2.2 Current Control Methods

Modern control theory was designed to deal with large, multi-input, multi-output systems, and is the basis for modern power plant controls. Historically, developing models for traditional coal fired power plants has been successful in creating advanced and robust controllers. Thermodynamics, heat transfer, and fluid dynamics within the plant can be modeled, and assumptions used to create these models have typically held in real-world power plants. Model Predictive Control (MPC) was developed to use these models for power plant systems to compute optimal controls [27]. Further, many model-based control methods in power plants, including MPC, Linear Quadratic Regulator, and H_∞, allow for control robust to noise and nonlinearities [25]. As a result of the availability of relatively accurate plant models, model based control methods are prevalent in many power plants. These model based control methods rely on two major assumptions.

1. Accurate models exist for the system.

2. Models can be linearized about a feasible operation point.

In HyPer, accurate analytical models of the plant have been difficult to obtain for several reasons. Operation characteristics for interaction of the control valves, turbo machinery, and fuel cell is not well understood [3]. Computational models of the system do not match the physical realities of the plant [37]. Stochastic dynamics due to turbulence cause flow imbalance and can lead to stall surge in the turbine further increasing the difficulty for modeling and control in the plant [39]. Thus, most of the expirimental controllers in the plant have been derived

using empirical data. Several of these empirical methods of modeling and control of HyPer have been used to achieve some demonstration of the potential of the hybrid fuel cell technology but lack the ability to control the full range of the plant's operation window [3, 30, 37, 38, 40].

Empirical data from plant runs was proposed by Tsai et al. in [37] to build empirical transfer functions as models for controllers. Previous work on valve characterization in [3, 40] used plant data to gain insights into the controllability of the plant, but did not use the data to derive models for plant controllers. Tsai uses systematic valve characterization to build empirical transfer functions. The empricial transfer functions serve to model the frequency response of the plant by fitting the data and mapping the controls to frequency response.

In [38], Tsai et al, use the empirical transfer functions described above to implement a robust H_∞ controller for the plant. Because sensor noise and model inaccuracies are an issue in the plant, an optimal robust controller was implemented in the plant. The controller is successful in setpoint control for several variables, although nonlinearities in the plant cause difficulties in reaching many states in the plant.

Restrepo in [30] uses Design of Experiment (DoE) techniques to expand these empirical transfer functions and more accurately model the plants response to control inputs. Factorial characterization data demonstrates the complex nonlinear interactions of the bypass control valves. Using the updated transfer functions, a Model Perdictive Controller is implemented for the HyPer plant.

Most notably, an adaptive scheme presented by Tsai et al. in [39] offers some solutions to the coupled, nonlinear control system in the HyPer plant. Using a Model Reference Adaptive Control (MRAC) the plant was shown to be controllable in simulation. This control scheme uses empirical transfer functions built from plant characterization tests as a non-linear model of the plant. A control loop uses these plant models to determine the optimal control input.

There are several disadvantages of this method for the HyPer facility. While empirical transfer functions of the plant contain plant response data that captures coupling and nonlinearities in the system, these functions are extremely costly to acquire. These transfer functions are developed through costly and time consuming plant characterization runs, where the frequency of actuation is varied and the plant states captured. The work presented in [39] uses transfer functions outdated by hardware changes. This method of control also causes unwanted oscillations due to the nature of the empirical transfer functions, rapid fluctuations in valve control and in plant state variables is undesired. The use of MRAC with dynamic models is difficult to compute on the fly for the fast responses needed out of the system. Methods for numerical simulation of the plant were found by trial and error and by hand tuning the parameters. This method requires filtering plant state data and sensor input into the controller, adding delay to control decisions.

The current state of the art controllers used in HyPer have many limitations. First, these methods use empirical transfer functions as a model; these transfer functions are difficult to develop and only describe a limited operating regime, so the controllable region of the HyPer operating regime is fairly limited. Second,

the empirical transfer functions run slower than real time, resulting in difficulties for MRAC to make real-time control decisions. Finally, due to the limited models, MRAC is extremely sensitive to noise, resulting in suboptimal control decisions being made when sensor calibrations are incorrect. Also, nonlinearities in the empirical models are significant, and when linearized, are not a sufficient approximation of the plant to base a controller upon. Although HyPer has potential for providing efficient and reliable energy, the control of HyPer must be further developed and refined prior to commercial implementation.

2.3 Neural Networks

Neural networks are a distributed computational model used for complex input/output mappings, and are extremely robust to noise, making them good candidates to model systems or encode controllers [4]. Further, neural networks can model any continuous n-dimensional function to arbitrary accuracy [2].

As shown in Figure 2.4, inputs to the network are connected via weights to hidden units. The hidden units are connected to the outputs. This is a fully connected network, each unit is connected each unit in the following layer. Units are made up of activation functions, in this work sigmoidal activation functions are used and defined as follows:

$$H(x) = \frac{1}{1 - e^{x}} \tag{2.1}$$

Where x is the input to the activation function.

$$x = \sum_i w_i x_i \tag{2.2}$$

By passing the inputs through the series of activation functions, highly non-linear relationships can be encoded in the network weights. A wide range of input/output mappings are quickly computed with a single pass through network.

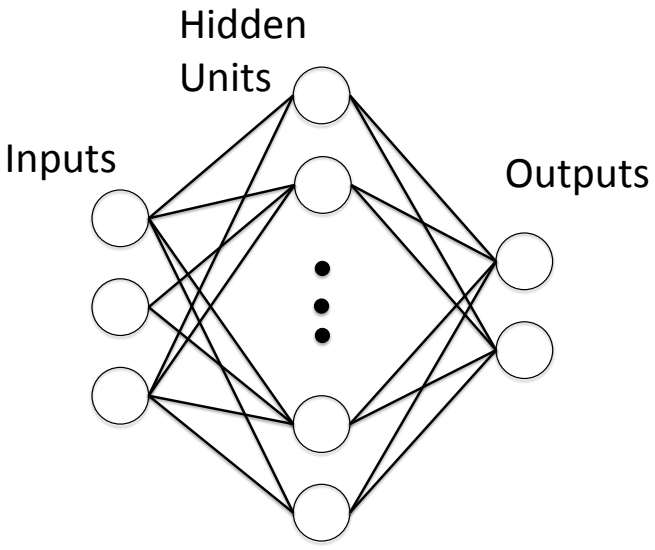

Figure 2.4: Fully connected neural network representation.

2.3.1 Backpropagation

Error backpropagation is an algorithm used to train a neural network based on data. The weights of the network can be trained by computing the error between the target data point and the output of the network. Once the error is computed, the weights are updated by performing gradient descent in the weight space minimizing mean squared error. The update rule for a particular weight of a network is given by:

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}} = -\alpha \delta_j x_i + \eta \Delta w_{ij}^{t-1} \tag{2.3}$$

$$\delta_j = \begin{cases} (output_j - target_j) H(in_j)(1 - H(in_j)) & output\ unit \\ (\sum_l \delta_l w_{jl}) H(in_j)(1 - H(in_j)) & hidden\ unit \end{cases}$$

Where α is a learning rate and determines how far along the gradient we step, δ is the gradient of the error backpropagated though the network and is unique depending on the layer of the network being updated, and x_i is the input to the weight being updated. The second term is called momentum, momentum uses the previous weight update Δw_{ij}^{t-1} scaled by a momentum factor η to prevent becoming stuck in local optima.

After all data points are presented to the network, the data points are shuffled and are presented again for another epoch. We split the data between a learning set and a validation set. The mean squared error of the network for the training and validation set is saved for each epoch. By comparing the error for the training and validation set, we can determine the optimal stopping point for training. When

the validation error stops decreasing, the ability for the network to generalize to new inputs. By stopping learning at this point, the network will avoid over fitting the data.

In Colby et al. a backpropagation network was used to decrease the computational complexity of simulation to determine fitness in a evolutionary algorithm to optimize the configuration of a wave energy converter [6]. Through solving energy and momentum equations and simulating many wave energy converter geometries, they were able to decrease the computational time required to evolve new geometries. By evolving geometry for the converter in question, the authors were able to drastically improve the performance of the converter compared to the previous state of the art geometry.

2.4 Neuro-evolutionary Control

Training neural networks with backpropagation is highly effective to learn weights when the targets for learning are available, or when gradient information is present. Neuro-evolution is used when no information about the targets is available [12]. In difficult control problems, information about the optimal target control outputs is often unavailable [1, 13, 15].

Neural network controllers can be trained with evolutionary algorithms, which are stochastic optimization algorithms inspired by biological evolution [6]. Neuro-evolutionary controllers are robust to noise, are computationally efficient (allowing for real-time control), are inherently nonlinear, and are adaptive to rapidly chang-

ing control inputs. Further, developing neuro-evolutionary controllers does not require an explicit system model, making them ideal candidates for control of systems in which accurate system models are unavailable (such as HyPer) [35] . Neuro-evolutionary control has been demonstrated on many difficult control problems such as micro air vehicle control in wind currents [32], finless rocket altitude and attitude control [12], wave energy converter ballast control [6], and even hybrid power plants [31] .

In neuroevolution, a population of n neural networks is maintained. In each generation, n new networks are created, which are slightly altered (mutated) versions of the networks currently in the population. Each of the $2n$ networks are evaluated in the target system, and assigned fitness values based on their effectiveness. Finally, n networks are selected to survive to the next generation, with higher fitness values corresponding to a higher probability of selection. This process is repeated until a desired performance threshold or maximum number of generations is reached [6]. Many evolutionary algorithms use crossover in the mutation operation, combining solutions to create new population members. In neuro-evolution, this is unnecessary as replacing sections of random weights with another section of random weights is not often productive, and neuro-evolution is successful without this step [6, 18, 32].

2.5 Fitness Approximation

In order to assign fitness in the evolutionary algorithm, the candidate controller must be evaluated. Fitness approximation techniques are used when fitness evaluation is difficult or expensive to compute [7, 21, 29]. In the case of the HyPer power plant, models of the system exist, but are computationally costly and do not accurately model all system coupling and nonlinearities. In order to preform an neuro-evolutionary algorithm for control, a computationally efficient approximation of the plant is required. Work in fitness modeling has demonstrated the ability to use approximations of the fitness function in lieu of costly simulations or hardware experiments [16].

Many methods exist to use fitness models to speed up fitness evaluation. Examples of fitness modeling include online updates of the fitness model [23], the use of simplified surrogate problems [10, 17, 20], or neural network approximation trained with real world data [19]. Neural networks trained with backpropagation are particularity powerful as fitness models since they have the representational power to model highly nonlinear functions to an arbitrary degree [2, 4]. Neural network fitness models have been shown to decrease computational cost of fitness evaluation, while maintaining useful fidelity even while modeling highly nonlinear functions [6, 33].

The use of fitness approximation to evaluate controllers, is a well established tool for evolutionary algorithms. By applying it to the HyPer plant, we evolve controllers which (i) match plant dynamics and (ii) can be computed efficiently.

Chapter 3 Method

In this section, we describe the method to evolve neural network controllers for the power plant. The outline of the method is shown in Figure 3.1. First, we found measured data from actual HyPer experiments to train a neural network simulator (① in Figure 3.1). We use the data to train a neural network using backpropagation, a gradient descent technique to find optimal network weights (② in Figure 3.1). If the resulting simulator has insufficient accuracy, we find regimes where more measurements needed and conduct more HyPer experiments to collect measurements or use data from previous experiments.

Once the neural network simulator of HyPer is complete, we develop a neural network-based time domain simulation of HyPer (③ in Figure 3.1). This time domain simulator is used to evaluate and assign fitness to controllers developed by the neuro-evolutionary algorithm (④ in Figure 3.1).

3.1 Function Approimation Plant Simulation

The neuro-evolutionary algorithm used to develop HyPer controllers requires that many different controllers be evaluated. Current models of HyPer are either computationally expensive or inaccurate, and are thus inadequate for the neuro-evolutionary algorithm. To address this problem, we develop a neural network

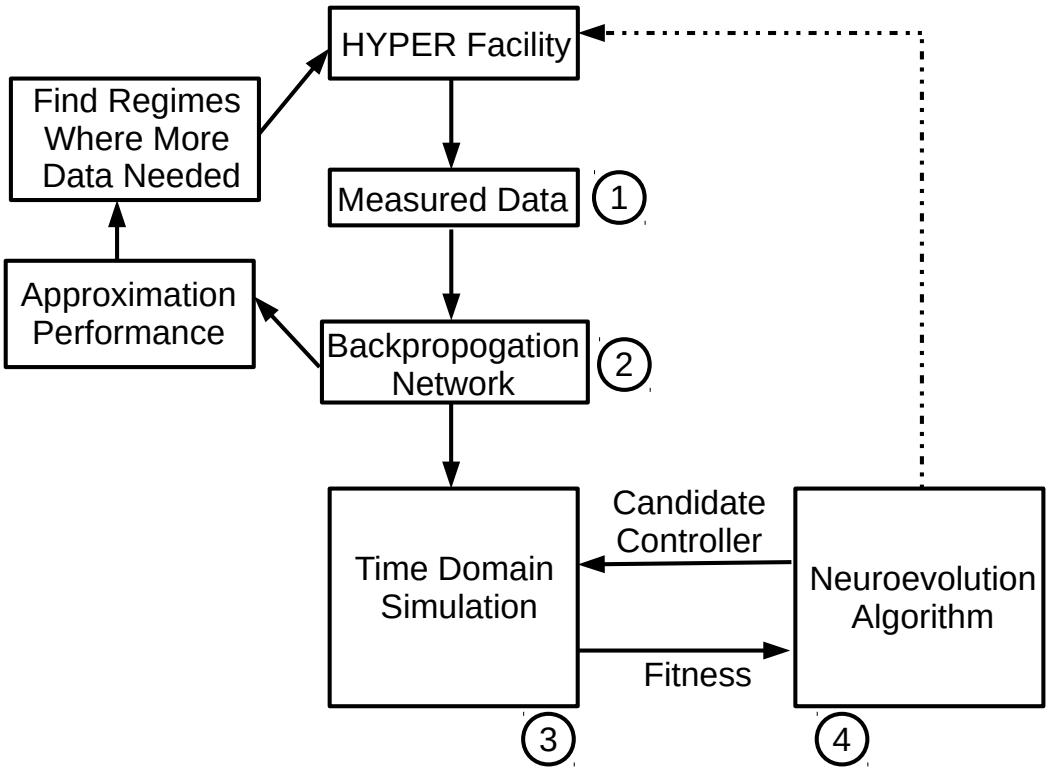

Figure 3.1: Visualization of the development of the neuro-evolutionary controller for HyPer.

function approximator trained on data collected from physical HyPer runs to create a fast and accurate simulator for HyPer.

Valve characterization runs in HyPer are used as a data source to learn a controller for the cathode air flow rate. By using data from real plant runs, we can learn a simulation of the plant that both matches underlying interactions in the plant, but also represents a reasonable operating range of the plant. We selected data from cold air valve characterization runs corresponding to ① in Figure 3.1. Valve characterization consists of varying the cold air valve between 10% and 80%

and allowing the plant to reach steady state operation. Sensor outputs across the plant were recorded at each timestep. The resultant data set contains 19 plant state variables and two control variables.

The data set contained 30,626 data points sampled at 12.5 Hz, or about 10 minutes of run data. To decrease the training time for the neural network simulator, data was downsampled by averaging every 25 timesteps. Thus, 1225 points are presented to the network to learn. Each simulation step will then amount to 2 seconds in the real plant. Assuming control decisions for hot and cold air bypass valves are made every 2 seconds does not limit the fidelity of the simulation, nor does it limit the control of the plant, as these valves are simply used to tune the steady state response of the plant.

A single hidden layer neural network is created with 21 inputs for the current state and control inputs, and 19 outputs return the plant state at the next time step. Initial network weights are initiated from a Gaussian distribution with a standard deviation of 0.75 with full connections to the hidden unit layer and output layer. The network is trained using backpropagation, where the collected data from HyPer runs is used to tune the network weights.

Backpropagation is the process of training a neural network using gradient descent. The gradient of the network approximation error as a function of the network weights is determined, and this gradient is used to incrementally change the weights in order to minimize approximation error. A training data set and a validation data set are created. As neural networks can approximate functions to arbitrary accuracy, they tend to "overfit" and learn noise in the training data.

Network training is stopped once approximation error in the validation data converges. As gradient descent has the property of converging to local optima, the network training process is run for 750 independent trials, to ensure the trained network is not located at an undesirable local optimum.

3.2 Time Domain Simulation

Once trained, the network is used to develop time domain simulation of the plant. The network maps the current plant state and control decisions to the next plant state, so a time domain simulator simply needs to record the current state of the plant, get control decisions from the controller, and call the neural network at each time step. Algorithm 1 describes the simulation in detail.

The learned simulation of the plant allows for fast evaluation of controllers. Since physics based models of the power plant are difficult to simulate in real-time and plant runs are costly and time consuming, a learning based controller using these methods for simulation becomes computationally intractable. In Table 3.1 we compare the run-time of the learned simulation to the required run-time for models computing at or near real-time.

Run-time	Neural Network	HyPer Models
Single Sim	< 0.001 seconds	10 min
Single Generation	< 1 seconds	160 hours
One EA (5000 Gen)	42 minutes	47.5 years

Table 3.1: Run-time comparisons for HyPer Plant, HyPer Model, and NN function approximation.

Algorithm 1 Time Domain Simulation

Input: Control commands, initial state, setpoints, controller F, plant model M
Output: Plant state trajectory, setpoint error

Initialize $output = \{\}$
Initialize $error = \{\}$
Initialize $state = state_{initial}$
Initialize $currentError = (state - setpoint)$
For Each (Time Step)
    output.Add(state)
    For Each error.Add(currentError)
        control = $F(state, setpoint)$
        state = $M(state, control)$

return output

As seen in Table 3.1, a single simulation of the plant can be run in fractions of a second for minutes of run-time. Even with an order of magnitude speed up in simulation of the physics models of the plant, evolutionary methods for control of the plant would still require far too much computation to be useful. By developing a fast simulation of the plant, we can quickly evaluate controllers and assign them fitness values in an evolutionary algorithm.

To validate the simulator's performance, the time domain simulator is tested against an actual HyPer run. The simulated and actual measured plant variables are shown in Figures 4.1, 4.2, 4.3, and 4.4. The neural network approximation of the plant rejects the noise found in the data, and qualitatively reflects an accurate representation of the plant state.

3.3 Evolutionary Algorithm

We use a neuro-evolutionary algorithm to train a controller for HyPer, using the time domain simulator developed in Section 3.1 to assign fitness to controllers. A population of 100 neural networks is randomly initialized, with weights drawn from a normal distribution with zero mean and a standard deviation of 3.5. The controller inputs include the current state of the system and the desired state of the system. The controller outputs control decisions for the cold air bypass and fuel valves.

Networks are mutated by selecting 33 randomly chosen weights and adding a random value drawn from a normal distribution with zero mean and a standard deviation of 1.0. Networks are assigned fitness using the time domain simulator from Section 3.1, where controllers which keep the plant closer to a desired trajectory have a higher fitness (error calculated using mean squared error). Networks are selected for survival using a binary tournament selection technique, based on their assigned fitness values. Ties are broken arbitrarily.

Two types of controllers are evolved for the plant. First we evolve a controller capable of reaching an arbitrary setpoint. Secondly,we evolve a controller capable of following a trajectory. Both are tested with and without the presence of noise.

Algorithm 2 Neuro-evolutionary Algorithm

Initialize n networks
For Each Generation
    For Each(Network in Population)
        create a copy of Network;
        mutate Network copy;
    For Each (Network in Population)
        Evaluate network in time domain simulation;
        Assign fitness to network;
    Select population members for survival;
return best network found;

Chapter 4 Results

The following chapter provides the results of the learned plant model simulation as well as control results for the initial model. Section 4.1 shows the results of the model validation. Section 4.2 shows the evolved setpoint controller for fuel cell flow rate. Section 4.3 shows the evolved trajectory controller for turbine speed.

4.1 Model Validation Results

To validate the neural networks simulation of the plant, the simulation is given the initial plant state from the original data and then allowed to respond to the control inputs from the original data. Figures 4.1. 4.2, 4.3, & 4.4 show the response of the system to the original control inputs found in the data. We show the comparison of the original data set to the learned simulation for several plant variables: turbine speed, fuel cell flow rate, total fuel input, and turbine inlet temperature.

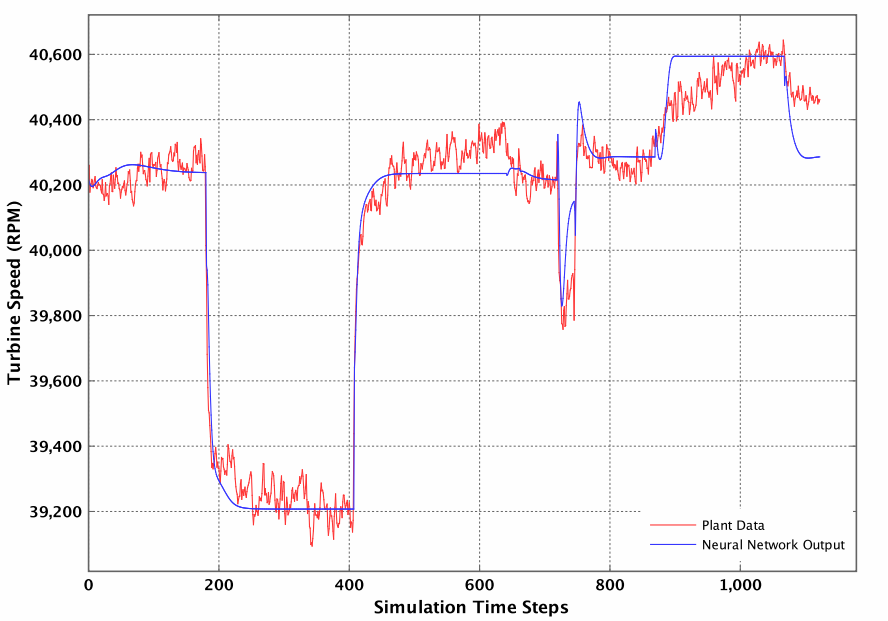

Figure 4.1: Simulation validation for turbine speed. The neural network simulator accurately models the HyPer facility, with the key difference between simulated and hardware runs being the lack of noise in the simulator.

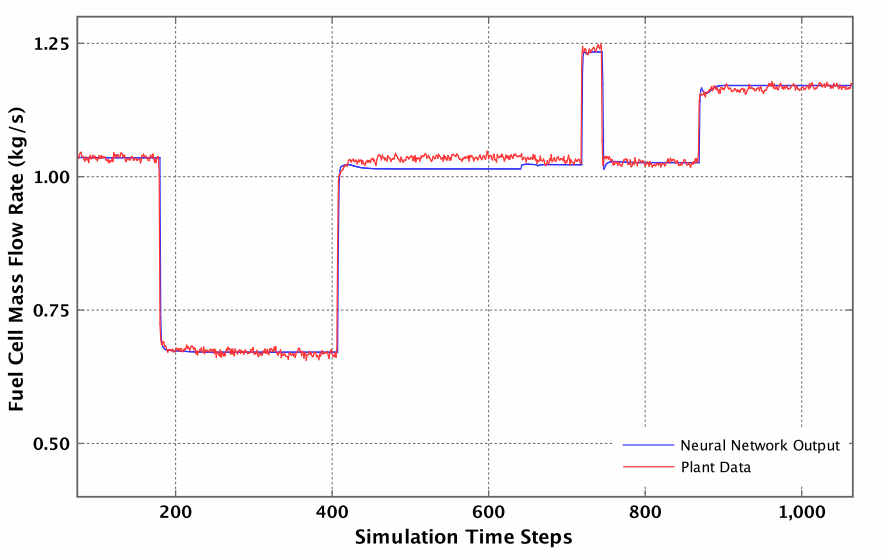

Figure 4.2: Simulation validation for fuel cell flow rate.

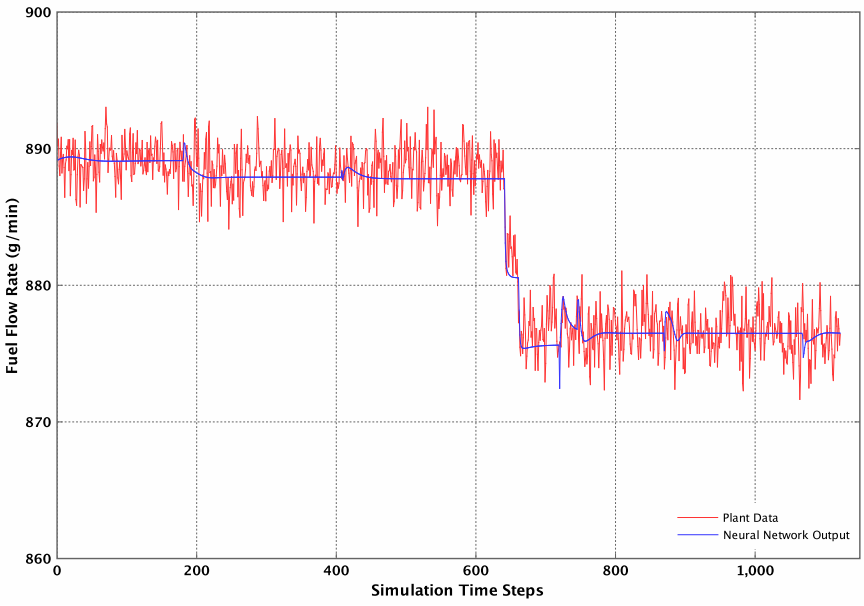

Figure 4.3: Simulation validation for total fuel entering the plant system.

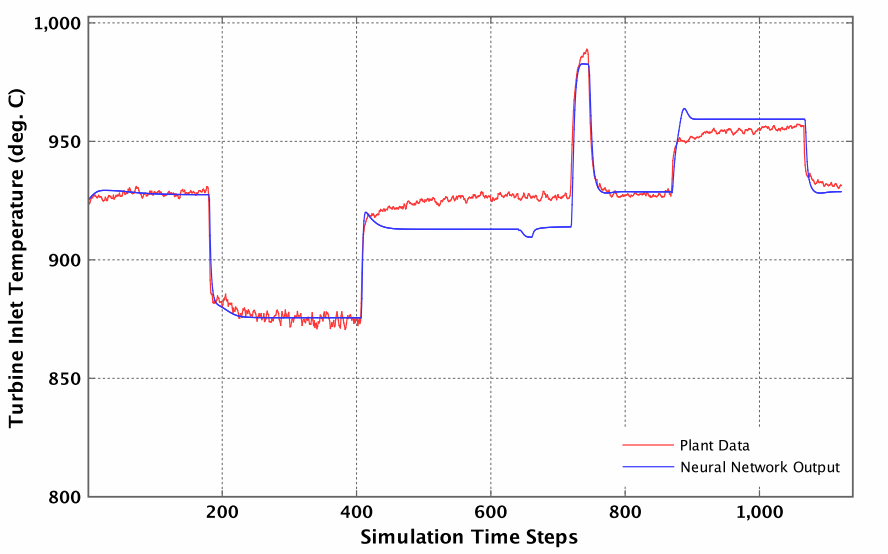

Figure 4.4: Simulation validation for turbine inlet temperature.

Mean squared error across all data points is 0.115%. As seen in the validation plots, the maximum error of the network approximation corresponds to about 5% at any point in the simulation for turbine speed, one of the most difficult plant state variables to model in the data. The large discrepancy in turbine speed shows it is largely dependent on the time state of the system, intuitively this discrepancy can be accounted for by turbine inertia, and that turbine speed is non-Markov without turbine acceleration in the state. This is deemed acceptable since the approximation of the state is within 5%, and the state information is directly gathered from the sensors.

Fuel cell flow rate in Figure 4.2, shows the network capturing the behavior of the flow rate very closely, with the network capturing underlying dynamics of the plant. We conclude that the simulation of the plant captures plant dynamics and

can be used to assign fitness to controllers.

4.2 Setpoint Control

We ran three sets of experiments with the controller to demonstrate its effectiveness. Control of the fuel cell flow rate of with the cold air valve is shown for several set points. The evolved controller is then tested with sensor and actuator noise.

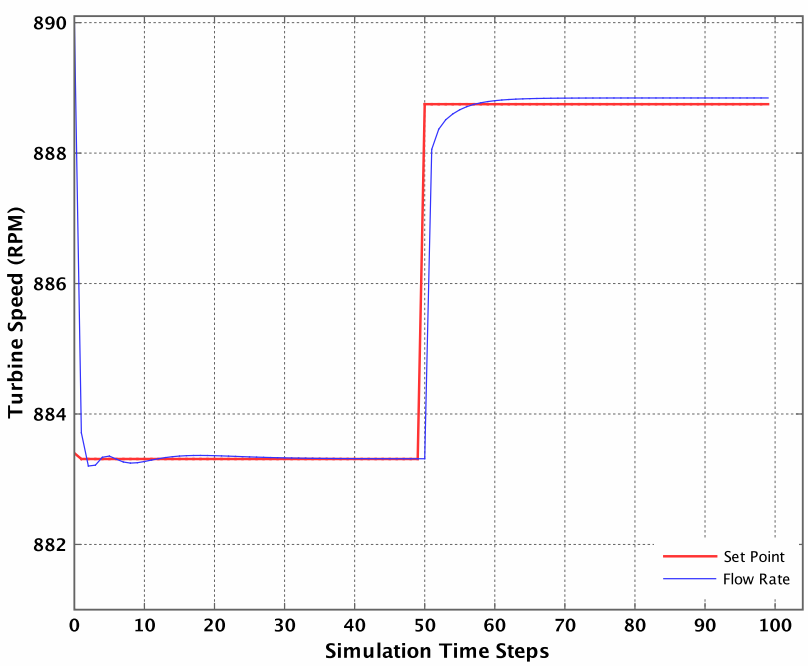

Figure 4.5: Control of fuel cell flow rate with the cold air valve response to step input in control. Initial step down shows small controller overshoot, the step up shows lightly damped behavior.

Setpoint Control: Figure 4.5 shows the fuel cell flow rate response for cold air valve actuation with neuro-evolutionary control. Initial state of the plant indicates

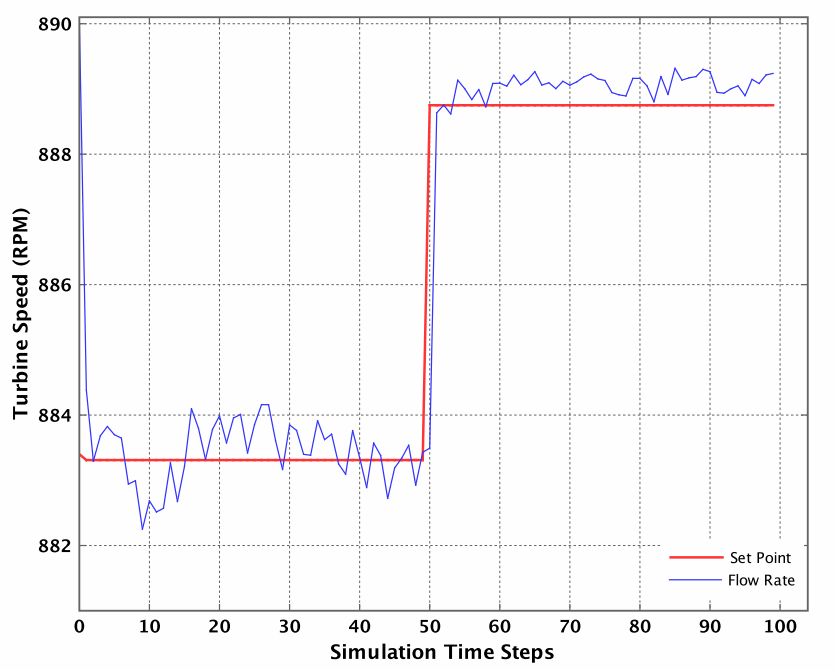

Figure 4.6: Evolved neural network controller response to 10% sensor noise. Notice that 10% sensor noise is larger than the commanded control step input.

the normalized fuel cell flow rate is at 0.863. Two setpoints are given to the controller after it has reached a steady state. As we can see in the figure, the controller immediately corrects the flow to the desired initial setpoint. Slight overshoot is observed with the first step down, indicative of the drastic change in plant state occurring with the neural network control. When the setpoint is switched at timestep 50, the controller follows a smooth trajectory to the desired setpoint. Because the network is trained on many setpoints, it adaptively adjusts its response due the the severity of the control change. Once the controller reaches steady state, the error is minimal.

Control with Sensor Noise: The evolved controller does very well for these state changes on a variable that is difficult to directly control when the input to the system reflects the actual state of the plant. As can be seen in Figures 4.1. 4.2, 4.3, & 4.4 the actual sensor readings of the plant are very noisy, with upwards of 6% noise on the states. In the actual power plant, the sensor noise is filtered and averaged over several timesteps to obtain inputs for the controllers [39]. Neural network controllers, on the other hand, have demonstrated to handle noise well in the inputs. An ideal controller for this power plant will reject at least the same level of noise found in the actual plant to control the plant to the desired setpoint.

The evolved controller now is given a noisy input from the plant. The controller was not trained on with noisy states, but on true state data output by the simulation. We now add 10% Gaussian noise to the input of the controller at each timestep. Figure 4.6 shows the system response to noisy controller input. The actual plant state is plotted for a noisy input.

10% noise on the input of the network is larger than the expected change in the setpoint. In any controller with output directly proportional to the input, one would expect noise to over shadow the control input. In our neural network controller noisy inputs are averaged out since the output depends on all plant state variables. The control decision is robust to the noise in the plant. Plant output is still noisy, but at the worst 10% sensor noise maps to less than 2% noise in the true output of the plant. This demonstration shows the noise rejection capabilities of the neural network controller to be far greater than needed in the plant. Implemented in the plant, this controller would receive filtered data with

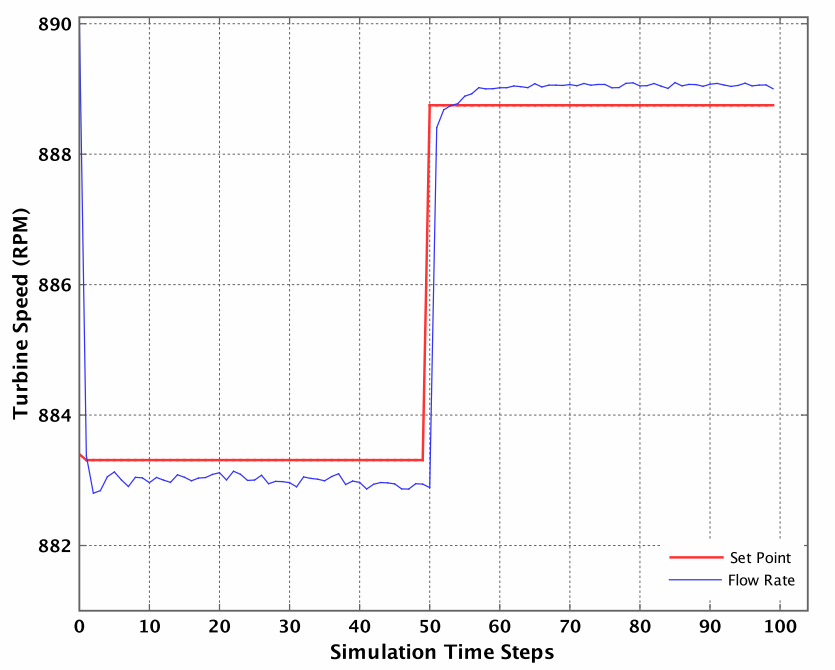

Figure 4.7: Control with 5% actuation noise. The level of noise in the flow rate is les than 0.2%, although a small steady state error is present.

noise far below 5%.

Control with Actuation Noise In addition to sensor noise, we would like to test the controller's sensitivity to its own output. We know from the first experiment that the control inputs can very change the plant state in few timesteps, and do so very drastically. If there are any deviations in control input to the system compared to the control command, the controller must be able to recover without damaging the system or causing drastic changes in plant state. In this experiment 5% actuation noise is added to the output of the controller, while sensors report the true plant state.

Figure 4.7, shows the results of adding actuation noise to the system. Large actuator noise in the plant is unlikely. Each control valve is equipped with its own feedback system. This experiment demonstrates that if the learned controller makes unexpected decisions, the impact on the plant operations will be minimal. 5% actuation noise corresponds to less than 0.2% noise in the plant state.

4.3 Trajectory Tracking

The evolved controller was tested for a turbine speed trajectory for an arbitrary load cycle. We performed two experiments on the controller using the learned simulation of the plant. First we track a turbine speed trajectory, in order to demonstrate the ability of our learned controller to adequately control the plant, and provide a baseline performance level to compare against. Second, we add noise to the plant sensors and actuators, in order to demonstrate the capabilities of the learned controller in a more realistic environment.

We present an arbitrary trajectory to the controller to track, where the initial plant state was chosen arbitrarily within the simulator operating range. The controller tracks the turbine to within a maximum of 0.001% of the desired trajectory. As can be seen in Figure 4.8, initial turbine speed is at 41,500 rpm. In 25 seconds the turbine speed has reached the desired setpoint, and continues to accurately track the desired response from that point forward. This result demonstrates the ability of the learned controller to track a desired trajectory. In addition to turbine speed of the plant, the cold air valve actuation command is plotted. This shows

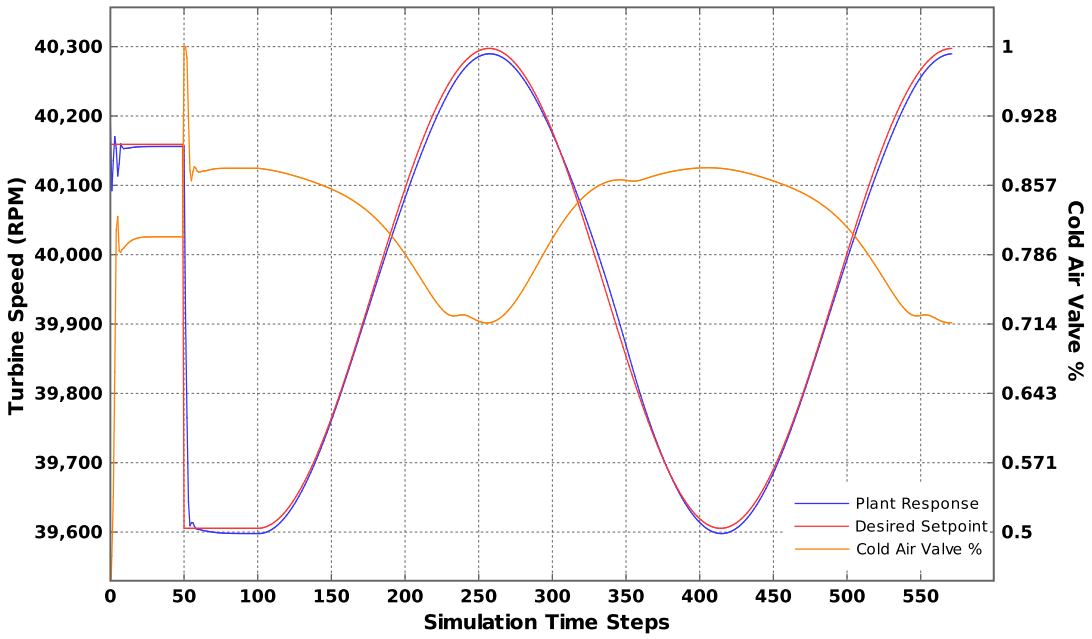

Figure 4.8: Neruroevolutionary control of turbine speed in the power plant. The control action to produce the tracking is shown as the out of phase signal on the plot.

that even for drastic changes in state control remains within actuation limits and does not put unrealistic requirements on valve actuation. The process of developing a neuro-evolutionary controller would not have been possible without the fast time domain simulation developed in Section 3.1, because the neuro-evolutionary algorithm requires every controller created to be evaluated in simulation. By developing a fast and computationally efficient simulator, we ensured that a neuro-evolutionary algorithm was computationally tractable.

The results of the addition of 10% sensor and actuator noise noise(approximately 150 RPM) for the same trajectory are shown in Figure 4.9. The addition of noise

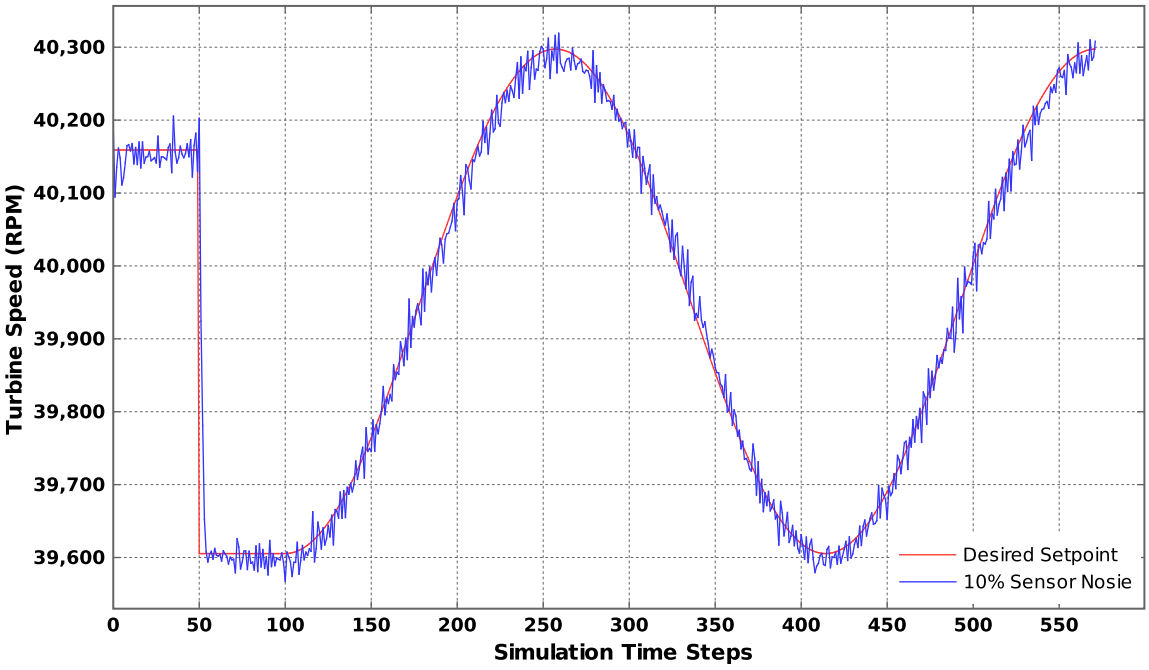

Figure 4.9: Neruroevolutionary control with 5% noise. The controller is able to track the trajectory to within approximatly 50 rpm at any given point in the simulation.

does not significantly alter the performance of the learned controller, demonstrating the ability of the neural network controller to adequately reject noise and disturbances while still maintaining control of the plant.

Chapter 5 Expanded Simulation and Control Results

As we can see above, control is limited to the range of data found in the original data set. This chapter describes the expansion of the original simulation to a wider operation regime based on factorial characterization of the control valves. Section 5.1 describes the neural network architecture for the updated simulation. Section 5.2 shows the performance of the updated function approximation simulation, and Section 5.3 shows the results of controlling the HyPer simulation for high efficiency operation.

Factorial characterization data is from plant experiments systematically changing all actuators simultaneously. By changing all of the controls, the resulting data demonstrates a wider range of plant states, as well as contains the dynamics of the interacting actuators. Expanding the simulation to include these actuators and map the plant states, allows us to create a controller able to control the whole plant.

5.1 Neural Network Architecture

In order to expand the simulation of the power plant to include more states and actuators, the neural network simulation must be updated. We found it to be difficult to replicate the results of for the cold air characterization data with a

Table 5.1: Table of factorial data vs cold air data learning parameters and MSE

Data Set	Hidden Units	Best MSE
Cold Air(best network)	35	0.115
Factorial	10	8.35
Factorial	15	5.52
Factorial	25	3.43
Factorial	35	2.69

factorial valve characterization set. Recall from Chapter 3, that the optimal neural network weights for the learned simulation required many random restarts to find a suitable network accurate enough to be used as a fitness approximation. When the back propagation algorithm used was to learn a simulation based on the factorial data, the resulting networks were not comparable in accuracy to the original cold air data set. Results showing the comparison of the data sets is shown in Table 5.1.

The factorial data set is far more complex than the original cold air data set. 4 control valves are simultaneously adjusted, demonstrating the nonlinear nature of the plant to the coupled control inputs. The neural networks learned from this data suffered from cross-talk.

Cross-talk occurs when a network has insufficient hidden units to learn the underlying function to the data. For example, the cold air valve will cause a drop in flow rate entering the fuel cell while simultaneously increasing the flow rate at the turbine. The effects of the opposite signs on the weights average out in the network, and the output is not consistent with the data.

To solve the cross-talk problem, increasing the number of hidden units is useful,

but the more complex network takes more time to train, and also required many random restarts to find the globally optimal weights.

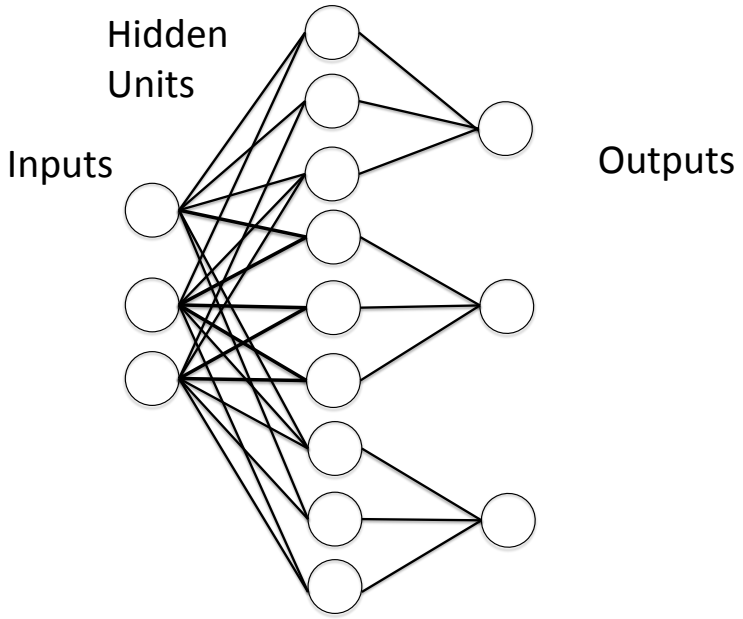

Figure 5.1: Example of network architecture for the expanded simulation. The input layer is fully connected to the hidden layer, however outputs are isolated from one another using a sparse network. This prevents cross-talk and allows us to learn a plant simulation from more complex data.

Instead, we change the network architecture to isolate the outputs of the network. Rather than a fully connected network as shown in Figure 2.4, a sparse network is created. Figure 5.1 shows the architecture scheme applied to the factorial data sets.

Each output is given by a sub-network isolated from each other output. For each sub-network for an output, an independent network is created with the current full state and actuation decision as input, a single hidden layer, and a single

output: the variable at the next time step. Each of the sub-networks can be trained independently and have individualized hidden units. A sub-network is created for each variable in the output vector for the next time step. Thus, the combined network can be used for a time domain simulation in place of the original neural network simulation described in Chapter 3.

5.2 Simulation Performance

All the sub-networks are trained as outlined in Chapter 3. Each sub-network is trained with 3 hidden unit neurons for 1000 epochs and 100 random restarts. As can be seen comparing Figure 4.1 with Figure 5.2, the new neural network architecture outperforms the single network.

When presented with the factorial data, the learned simulation outperforms all of the single networks found in the earlier search shown in Table 5.1. With fewer neurons and fewer random restarts.

As can be seen in Figure 5.4, the isolated network architecture can learn complex functions for in the data. The simulation also expands the states to learn controllers from.

5.3 Control Results

One difficult control problem for the plant is operating at peak fuel efficiency for a given power plant state. With an accurate simulation that uses all plant actuators,

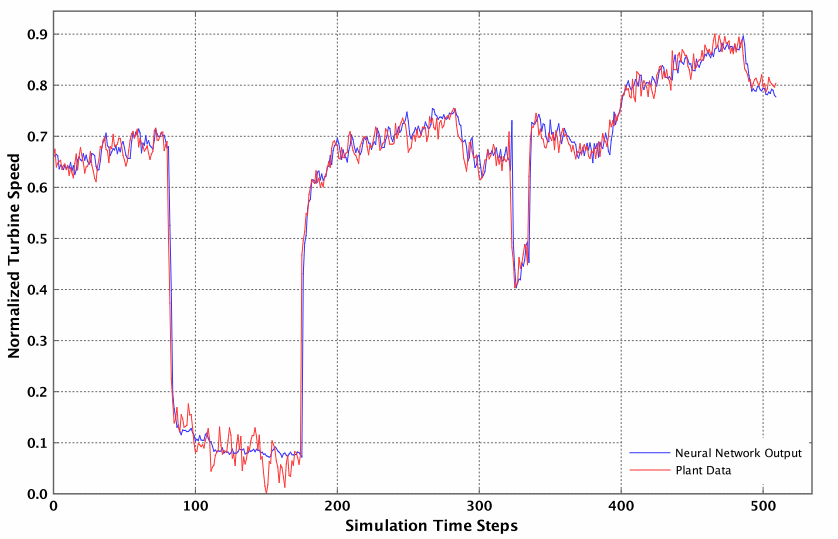

Figure 5.2: Verification of the new network architecture for the turbine speed, compared to Figure 4.1 we can see that the expanded simulation network architecture outperforms the single hidden layer network. At any given point, the network is error is less than that of the previous simulation.

we can now control the plant for maximum fuel efficiency.

Since we know plant state variables such as turbine speed, temperatures, air mass flow rates, and fuel mass flow rate, we can approximate the power output of the plant and the fuel efficiency. Power out of the turbine is proportional to the air mass flow and the temperature difference across the turbine at steady state.

$$W_{turbine} \propto \dot{m_{air}}(T_{in} - T_{out}) \tag{5.1}$$

Power output of the fuel cell is also proportional to the air mass flow and temperature across the fuel cell at steady state.

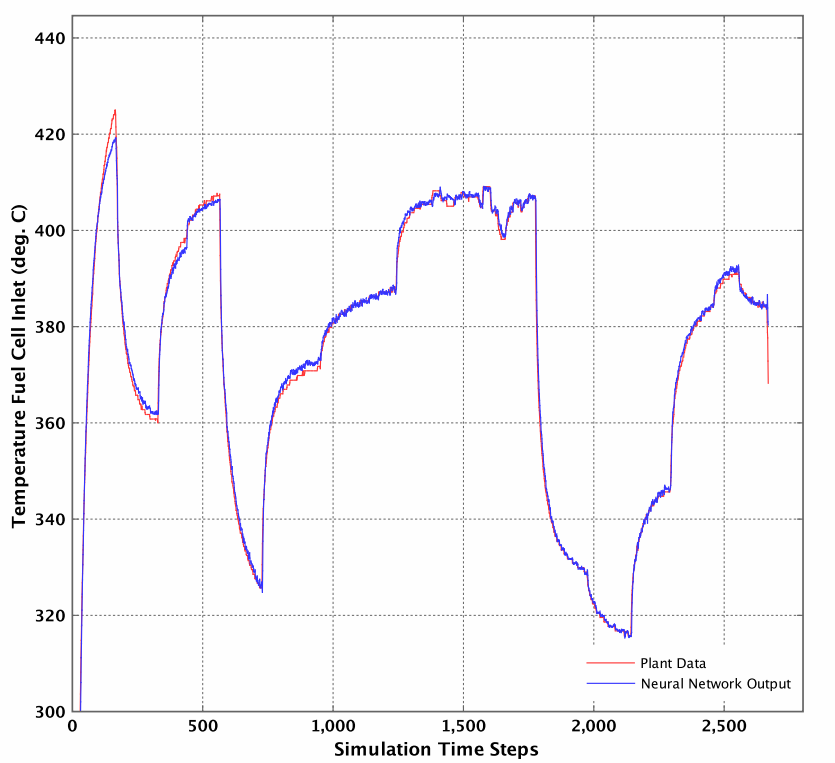

Figure 5.3: Verification of the new network architecture for turbine inlet temperature.

$$W_{fuelcell} \propto \dot{m_{air}}(T_{out} - T_{in}) \tag{5.2}$$

By maximizing the sum of power output per unit of fuel at each timestep in the simulation, we can learn a controller to operate the plant at a high efficiency. We perform an evolutionary algorithm using the approximate power output and the fuel mass flow rate to regulate the plant. The equation used for the fitness of the plant is shown below in equation 5.3.

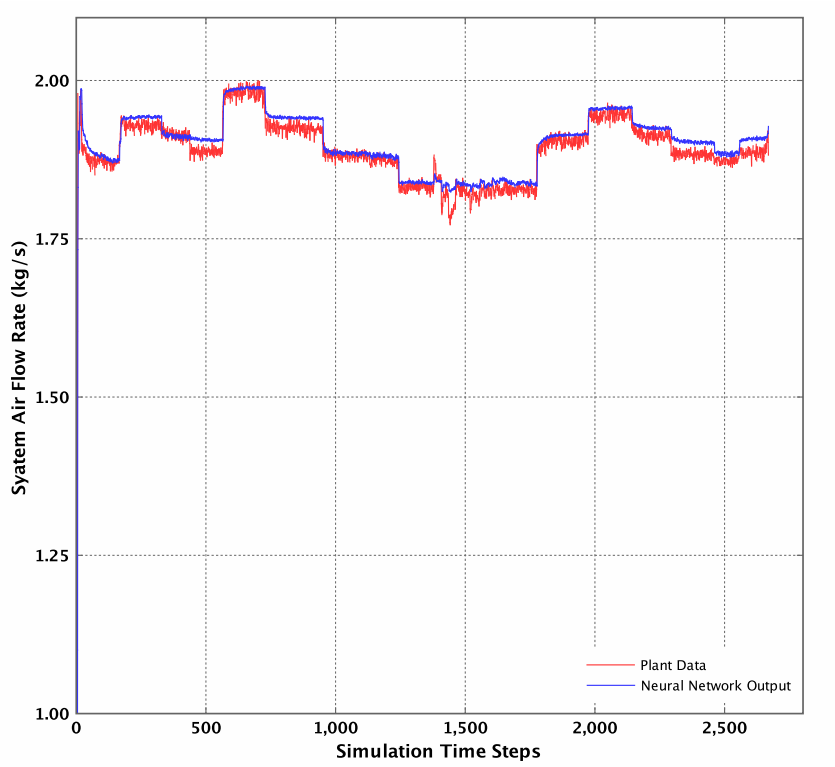

Figure 5.4: Verification of the new network architecture for mass flow rate.

$$Fitness = \sum_{timesteps} \frac{ApproximateTotalPowerOutput}{FuelIn} \tag{5.3}$$

A neuro-evolutionary algorithm is applied as outlined in Chapter 3, using the above fitness as an evaluation function. The resulting controller is an open loop neural network controller which seeks to maximize plant fuel efficiency. Input to the network is current plant state, output is control decisions to improve plant efficiency.

Looking at the Figure 5.5 it appears as though little is going on in the plant. The approximate plant efficiency remains relatively level, except in the first several time steps and after timestep 50. Figure 5.5 shows the control action being taken by the plant. The bleed air valve is changing dramatically, while the other valves remain in their standard open or closed positions. The bleed air valve is changing rapidly, but the total power output remains relatively constant. Plant state variable must be also be changing in order to keep relative power output constant.

Figure 5.6 shows some plant state variables response to the controller. Here plant state variables are imposed on the same graph to show the relative changes in the plant state. All plant state variables, including approximate power output, are shown in the normalized form output from the simulator to show the dramatic changes in plant state compared to the power output.

The controller uses the bleed air valve to control many plant state variable to work together in order to output at maximum efficiency. Figure 5.6 demonstrates the complexity of the power plant. Distinct operation regimes and nonlinear relationships are evident between all variables shown. As we would expect, we see the power output by the fuel cell increase while the temperature at the turbine inlet decreases. Here we see the controller taking advantage of the high efficiency found in the fuel cell. Also, we see use of the bypass control valve to cyclically decrease the total flow rate, which is proportional to both power output and fuel consumption. Oscillations occur where operation regimes transition such as when the bleed air valve is fully open. These oscillations could also be caused by the effects of the heat exchanges capturing waste exhaust heat. The controller bleeds

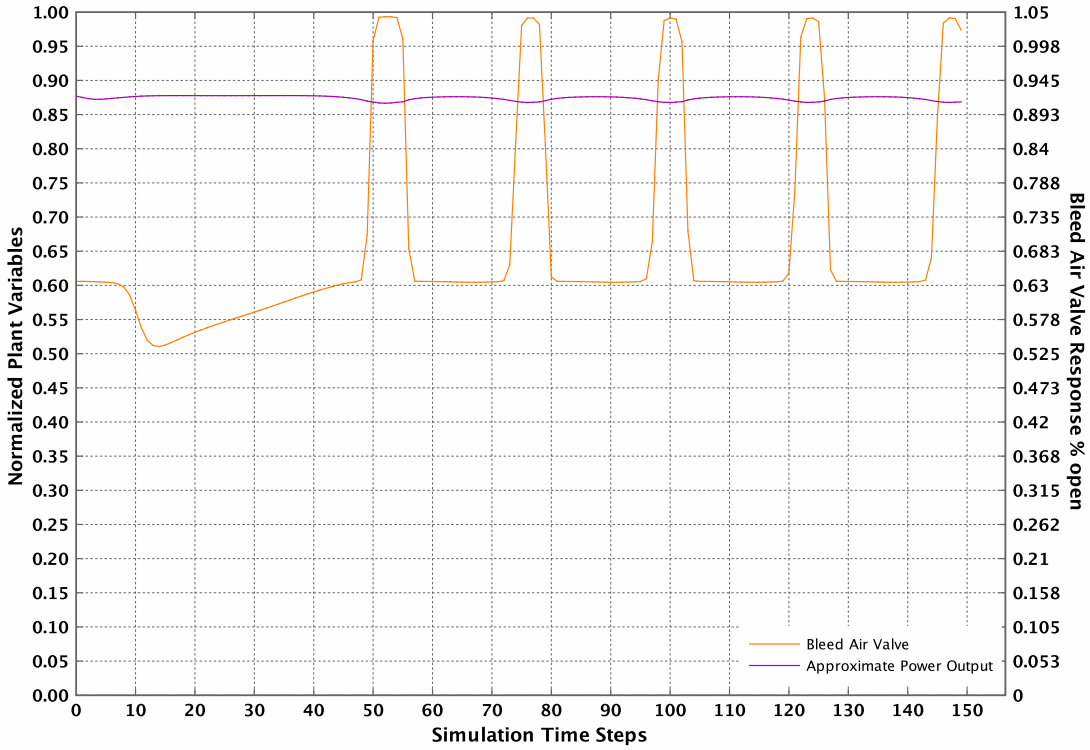

Figure 5.5: Approximate plant efficiency for 150 time step simulation showing the control action taken by the bleed air valve. Plant power output is relatively constant despite the large changes in control actuation. The periodic response of the plant control valve suggests that other plant state variables are changing significantly.

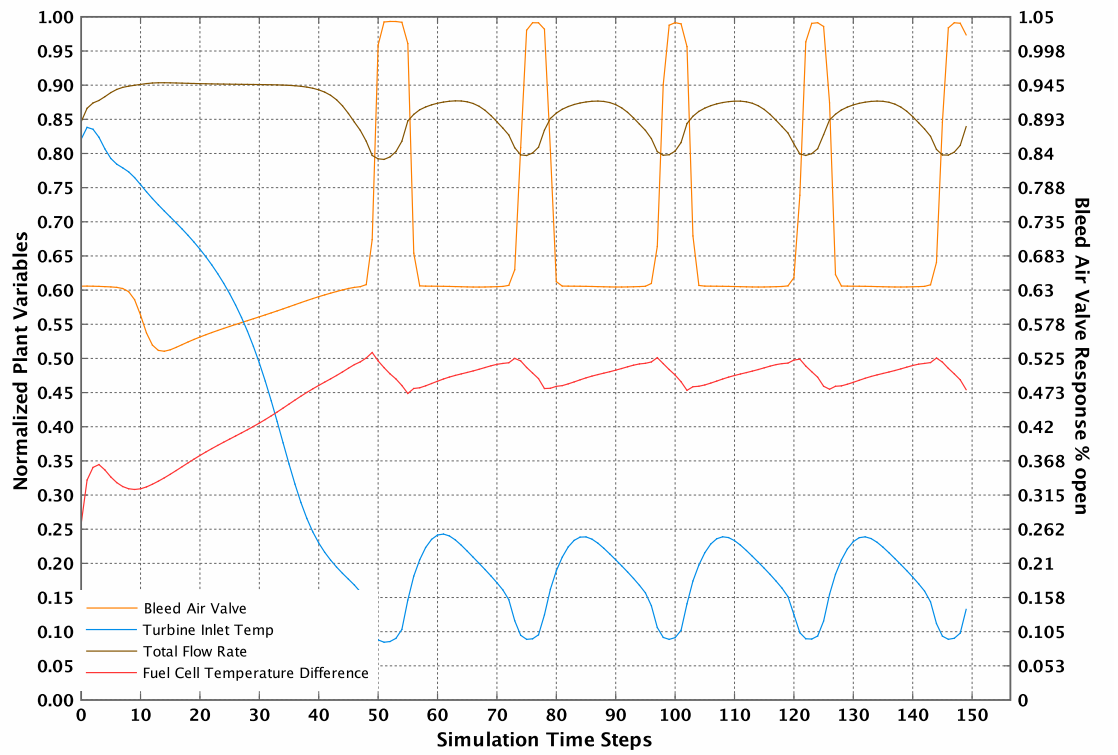

Figure 5.6: Approximate plant efficiency for 150 time step simulation including plant state. The bleed air valve causes many changes in turbine speed, flow rate, and temperature across the fuel cell. These effects add to produce power at a lower fuel cost compared to nominal operation at this power output. The controller decreases the reliance on the turbine speed for power generation and increases fuel cell utilization to operate at a higher efficiency.

off air when the fuel cell begins to run hot, such as at time step 50 in Figure 5.6.

Only the bleed air valve is being used by this controller, which leads us to conclude that a controller which primarily relies on the bleed air valve is easy to find in the neural network weight space, and represents a local optima. Future work should address the difficulty of using interacting valves to control the plant, algorithms such as cooperative co-evolution or multi-objective fitness assignment methods could address these difficulties.

It is worth noting that the approximate power output and efficiency scale is arbitrary and means little in the reality of the plant as presented here. The use of these metrics as an evaluation function serve to demonstrate that we can control the plant at a given state with a decreased fuel cost. The problem of controlling the power plant at maximum efficiency is inherently full of trade offs. More research is needed to quantify these trade offs and transition between desired power output levels.

Chapter 6 Discussion

Advanced power generation systems such as HyPer are growing increasingly complex in order to meet the demands for clean, efficient, and available energy needs. As the complexity of these plants increase, they become more more difficult to control due to complex interactions between plant components not yet well understood. Plant noise, variable coupling, system nonlinearities, and stochastic dynamics lead to inaccurate physics based models. Further, empirical models based on plant runs are difficult and expensive to obtain, run slower than real time, and often only describe a limited operating regime. The inability to develop accurate system models results in traditional model based control techniques to be inadequate for newly designed advanced power generation systems such as the HyPer hybrid plant.

In this work, we developed a computationally efficient simulator of the HyPer facility by training a neural network function approximator using actual plant run data. Rather than developing a simulator through physics based models, we used plant operational data to train a data based simulation to capture the complex mapping between the plant state and control decisions to the following plant state. The resulting time domain simulator can quickly and accurately model the response of the HyPer facility. As opposed to other empirical methods, the neural network simulation of the plant is both fast to compute and accurately captures plant

dynamics.

We then used an evolutionary algorithm to train a neural network controller for HyPer. Fitness assignment in the neuro-evolutionary algorithm was conducted using our time domain simulator, which allowed for thousands of controllers to be quickly evaluated. Without the fast time domain simulator, a neuro-evolutionary algorithm would have been computationally intractable. The function approximation using neural networks to approximate fitness in of controllers in the plant is crucial to make this work possible. The evolutionary algorithm with our learned simulation runs in hours as opposed to years if current state of the art models of the HyPer faccility were used to evaluate controllers.

The evolved neural network controller were able to reach arbitrary setpoints to a high degree of accuracy, and the trained controller was shown to accurately track desired system trajectories within 0.001% tracking error, even in the presence of 10% noise. Further, we demonstrate that the control policy learned during evolution does not place undue stress on actuators, and outputs feasible control policies at each time step.

In general, this method is applicable to learn controllers for any system with computationally difficult models or no models without data. Evolved neural network controllers are particularly well suited to these highly nonlinear problems with a high degree of noise and multiple inputs and outputs. Once a controller is found, we can use the simulation in conjunction with the power plant, and existing experimental data to determine where more plant measurements are needed. By expanding the simulation, we can evolve a controller capable of reaching more

states and utilizing more control actuators.

The controllers developed in this work demonstrate the potential of the hybrid technology in the configuration above, as well as the capabilities of the learning based control methods at the heart of this work, but in many ways shows the limitations of the process as well. Three areas of future research into the power plant have been identified. First, further work on the power plant will need to include using multiple plant controls to track power demands while optimizing for the multiple objectives found in the plant, such as fuel cost and power output, or turbine and fuel cell wear. Secondly, both the model and the evolved controllers were prone to local optima and required many random restarts and large population sizes to find better solutions. Future work will need to answer these limitations before implementation in the power plant. Third, work is still needed on how to combine the benefits of setpoint control, trajectory tracking and open loop power output control to create a unified system for operation of the plant.

Bibliography

[1] A. K. Agogino and K. Tumer. A multiagent approach to managing air traffic flow. *Autonomous Agents and MultiAgent Systems*, 24:1–25, 2012.

[2] Jean-Gabriel Attali and Gilles Pagès. Approximations of functions by a multilayer perceptron: a new approach. *Neural networks*, 10(6):1069–1081, 1997.

[3] Larry Banta, Jason Absten, Alex Tsai, Randall Gemmen, and David Tucker. Control sensitivity study for a hybrid fuel cell/gas turbine system. In *ASME 2008 6th International Conference on Fuel Cell Science, Engineering and Technology*, pages 653–659. American Society of Mechanical Engineers, 2008.

[4] Tianping Chen and Robert Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, 1995.

[5] M. Colby, M. Knudson, and K. Tumer. Multiagent flight control in dynamic environments with cooperative coevolutionary algorithms. In *2014 AAAI Spring Symposium on Modeling in Human Machine Systems: Challenges for Formal Verification*. Palo Alto, CA, 2014.

[6] M. Colby, E. Nasroullahi, and K. Tumer. Optimizing ballast design of wave energy converters using evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1739–1746, Dublin, Ireland, July 2011.

[7] Wu Deng, Rong Chen, Jian Gao, Yingjie Song, and Junjie Xu. A novel parallel hybrid intelligence optimization algorithm for a function approximation problem. *Computers & Mathematics with Applications*, 63(1):325–336, 2012.

[8] Dimitris C Dracopoulos. Evolutionary control of a satellite. In *Proceedings of the Second Annual Conference*, pages 77–81, 1997.

[9] Dario Floreano and Francesco Mondada. Automatic creation of an autonomous agent: Genetic evolution of a neural network driven robot. In *Proceedings of the third international conference on Simulation of adaptive*

behavior: From Animals to Animats 3, number LIS-CONF-1994-003, pages 421–430. MIT Press, 1994.

[10] Chi Keong Goh, Dudy Lim, Learning Ma, Yew-Soon Ong, and PS Dutta. A surrogate-assisted memetic co-evolutionary algorithm for expensive constrained optimization problems. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 744–749. IEEE, 2011.

[11] Faustino J Gomez and Risto Miikkulainen. Solving non-markovian control tasks with neuroevolution. In *IJCAI*, volume 99, pages 1356–1361, 1999.

[12] Faustino J. Gomez and Risto Miikkulainen. Active guidance for a finless rocket using neuroevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 2084–2095, San Francisco, 2003.

[13] Faustino John Gomez and Risto Miikkulainen. *Robust non-linear control through neuroevolution*. Computer Science Department, University of Texas at Austin, 2003.

[14] J. Shepherd III and K. Tumer. Robust neuro-control for a micro quadrotor. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1131–1138, Portland, OR, July 2010.

[15] A. Iscen, A. Agogino, V. SunSpiral, and K. Tumer. Controlling tensegrity robots through evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Amsterdam, The Netherlands, July 2013.

[16] Yaochu Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing*, 9(1):3–12, 2005.

[17] Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.

[18] M. Knudson and K. Tumer. Adaptive navigation for autonomous robots. *Robotics and Autonomous Systems*, 59:410–420, 2011.

[19] Vladimir M Krasnopolsky, Dmitry V Chalikov, and Hendrik L Tolman. A neural network technique to improve computational efficiency of numerical oceanic models. *Ocean Modelling*, 4(3):363–383, 2002.

[20] Minh Nghia Le, Yew Soon Ong, Stefan Menzel, Yaochu Jin, and Bernhard Sendhoff. Evolution by adapting surrogates. *Evolutionary computation*, 21(2):313–340, 2013.

[21] Minh Nghia Le, Yew Soon Ong, Stefan Menzel, Chun-Wei Seah, and Bernhard Sendhoff. Multi co-objective evolutionary optimization: Cross surrogate augmentation for computationally expensive problems. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8. IEEE, 2012.

[22] Jianyong Li, ET Ososanya, and RA Smoak. The neural network control application in a power plant boiler. In *Southeastcon'96. Bringing Together Education, Science and Technology., Proceedings of the IEEE*, pages 521–524. IEEE, 1996.

[23] Haiping Ma, Minrui Fei, Dan Simon, and Hongwei Mo. Update-based evolution control: A new fitness approximation method for evolutionary algorithms. *Engineering Optimization*, (46):1–14, 2014.

[24] M Moshrefi-Torbati, AJ Keane, SJ Elliott, MJ Brennan, and E Rogers. Passive vibration control of a satellite boom structure by geometric optimization using genetic algorithm. *Journal of Sound and Vibration*, 267(4):879–892, 2003.

[25] Fabian Mueller, Faryar Jabbari, Jacob Brouwer, S Tobias Junker, and Hossein Ghezel-Ayagh. Linear quadratic regulator for a bottoming solid oxide fuel cell gas turbine hybrid system. *Journal of Dynamic Systems, Measurement, and Control*, 131(5):051002, 2009.

[26] Mehdi Abbaszadeh Naseri and Alireza Yazdizadeh. Neural network-based imc-pid controller design for main steam temperature of a power plant. In *Advances in Neural Networks–ISNN 2009*, pages 1059–1068. Springer, 2009.

[27] S Joe Qin and Thomas A Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7):733–764, 2003.

[28] Sarvapali D Ramchurn, Perukrishnen Vytelingum, Alex Rogers, and Nick Jennings. Agent-based control for decentralised demand side management in the smart grid. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 5–12. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[29] Esmat Rashedi, Hossien Nezamabadi-Pour, and Saeid Saryazdi. Filter modeling using gravitational search algorithm. *Engineering Applications of Artificial Intelligence*, 24(1):117–122, 2011.

[30] Bernardo Restrepo. *Experimental characterization of fuel cell gas turbine power system and determination of optimal trajectories of operation using a model predictive controller*. WEST VIRGINIA UNIVERSITY, 2011.

[31] Kyoungsoo Ro and Saifur Rahman. Two-loop controller for maximizing performance of a grid-connected photovoltaic-fuel cell hybrid power plant. *Energy Conversion, IEEE transactions on*, 13(3):276–281, 1998.

[32] M. Salichon and K. Tumer. A neuro-evolutionary approach to micro aerial vehicle control. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Portland, OR, July 2010.

[33] C Sivapathasekaran, Soumen Mukherjee, Arja Ray, Ashish Gupta, and Ramkrishna Sen. Artificial neural network modeling and genetic algorithm based medium optimization for the improved production of marine biosurfactant. *Bioresource technology*, 101(8):2884–2887, 2010.

[34] Thomas P Smith, Comas L Haynes, William J Wepfer, David Tucker, and Eric A Liese. Hardware-based simulation of a fuel cell turbine hybrid response to imposed fuel cell load transients. In *ASME 2006 International Mechanical Engineering Congress and Exposition*, pages 319–328. American Society of Mechanical Engineers, 2006.

[35] James C Spall and John A Cristion. Model-free control of nonlinear stochastic systems with discrete-time measurements. *Automatic Control, IEEE Transactions on*, 43(9):1198–1210, 1998.

[36] Xiaoyan Sun, Dunwei Gong, Yaochu Jin, and Shanshan Chen. A new surrogate-assisted interactive genetic algorithm with weighted semisupervised learning. *Cybernetics, IEEE Transactions on*, 43(2):685–698, 2013.

[37] Alex Tsai, Larry Banta, Larry Lawson, and David Tucker. Determination of an empirical transfer function of a solid oxide fuel cell gas turbine hybrid system via frequency response analysis. *Journal of fuel cell science and technology*, 6(3):034505, 2009.

[38] Alex Tsai, Larry Banta, David Tucker, and Randall Gemmen. Multivariable robust control of a simulated hybrid solid oxide fuel cell gas turbine plant. *Journal of Fuel Cell Science and Technology*, 7(4):041008, 2010.

[39] Alex Tsai, David Tucker, and Tooran Emami. Adaptive control of a nonlinear fuel cell-gas turbine balance of plant simulation facility. *Journal of Fuel Cell Science and Technology*, 11(6):061002, 2014.

[40] David Tucker, Larry Lawson, and Randall Gemmen. Characterization of air flow management and control in a fuel cell turbine hybrid power system using hardware simulation. In *ASME 2005 Power Conference*, pages 959–967. American Society of Mechanical Engineers, 2005.

[41] David Tucker, Michael Shelton, and A Manivannan. The role of solid oxide fuel cells in advanced hybrid power systems of the future. *The Electrochemical Society Interface*, 18(3):45, 2009.

[42] Wolfgang Winkler, Pedro Nehter, Mark C Williams, David Tucker, and Randy Gemmen. General fuel cell hybrid synergies and hybrid system testing status. *Journal of Power Sources*, 159(1):656–666, 2006.