Recurrent Artificial Neural Networks for Low-Thrust Trajectory Design and Optimization

By
Taylor R. George
M.S., University of Kansas, 2020
B.S., University of Kansas, 2018
A.S., Garden City Community College, 2013

Submitted to the graduate degree program in the Department of Aerospace Engineering and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Chair: Dr. Brian Kaplinger

Dr. Haiyang Chao

Dr. Shawn Keshmiri

Dr. Craig McLaughlin

Dr. Yannan Shen

Date Defended: 5 May 2023

The dissertation committee for Taylor R. George certifies that this is the approved version of the following dissertation:

Recurrent Artificial Neural Networks for Low-Thrust Trajectory Design and Optimization

____
Chair: Dr. Brian Kaplinger

Date Approved: 10 May 2023

Abstract

Implementing low-thrust propulsion on spacecraft can be quite advantageous, resulting in lower fuel consumptions, greater payload fractions, lower launch costs, and more. Due to these advantages, low-thrust propulsion has been used as the primary and secondary means of propulsion in almost every space application, though a majority of spacecraft launched with low-thrust propulsion have been geosynchronous equatorial orbit (GEO) satellites. As the use of spacecraft launched with low-thrust propulsion can be expected to continue increasing significantly, so can the need for quick, accurate, and robust low-thrust trajectory optimization as well as autonomous low-thrust control. However, due to lower thrust magnitudes and longer transfer times, low-thrust trajectory optimization via traditional optimization techniques can be complex as well as computationally-expensive and time-consuming. Artificial neural networks, on the other hand, can provide an alternative to such techniques. However, though the use of artificial neural networks trained by supervised learning models for applications related to low-thrust trajectory design and optimization is extensive, previous research has largely focused on shallow or deep feedforward architectures with little emphasis placed on recurrent ones, even though recurrent architectures are inherently more suited to time histories. Thus, the objective of this research was to investigate the potential of recurrent artificial neural networks, specifically long short-term memory artificial neural networks, for low-thrust trajectory design and optimization with respect to their feedforward equivalents. As a majority of spacecraft launched with low-thrust propulsion have primarily been GEO satellites, the focus of this research was, primarily, on the low-thrust, orbit-raising problem for both time-optimal and fuel-optimal transfers. Overall, this dissertation will present the results and conclusions from this research as well as the scientific contributions.

Acknowledgements

First and foremost, I would like to thank my advisor, Dr. Kaplinger, and my committee members: Dr. Chao, Dr. Keshmiri, Dr. McLaughlin, and Dr. Shen. I would like to thank the rest of the faculty and staff in the Department of Aerospace Engineering as well for their guidance and assistance throughout the years. I would also like to thank the Madison and Lila Self Graduate Fellowship; I am deeply appreciative of their support while pursuing my doctorate degree. Finally, I would like to extend a special thank you to my fellow graduate students as well as my family, all of whom have stuck with me through thick and thin.

Table of Contents

List of Figures

List of Tables

1 Introduction

In this chapter, the objective and motivation of this research, which was on the use of recurrent artificial neural networks for low-thrust trajectory design and optimization, as well as an outline of this dissertation will be provided.

1.1 Objective

The overall objective of this research was to investigate the potential of recurrent artificial neural networks for low-thrust trajectory design and optimization. To complete this objective, long short-term memory (LSTM) artificial neural networks were applied to the low-thrust, orbit-raising problem for both time-optimal and fuel-optimal transfers. A total of four problems were considered: the time-optimal open-loop case, the time-optimal closed-loop case, the fuel-optimal open-loop case, and the fuel-optimal closed-loop case. The following scientific contributions resulted from this investigation and will be presented, in full, in this dissertation.

1. Recurrent artificial neural networks, specifically LSTM artificial neural networks, were applied to the low-thrust, orbit-raising problem for both time-optimal and fuel-optimal transfers.
2. An explicit model for a trade study between feedforward and recurrent artificial neural networks, specifically LSTM artificial neural networks, was developed and performed with a focus on the low-thrust, orbit-raising problem for both time-optimal and fuel-optimal transfers.
3. The applicability of recurrent artificial neural networks, specifically LSTM artificial neural networks, as a closed-loop solution to the low-thrust, orbit-raising problem for both time-optimal and fuel-optimal transfers was investigated.

4. The effect of time bias in data on the performance of recurrent artificial neural networks, specifically LSTM artificial neural networks, and feedforward artificial neural networks was investigated.

1.2 Motivation

Electric, or low-thrust, propulsion has specific impulses typically an order of magnitude or higher than traditional chemical propulsion [1]. Since the specific impulse of an engine is a measure of how efficiently that engine produces thrust, implementing low-thrust propulsion on spacecraft can result in the following advantages [2-4].

1. Decreased Fuel Consumptions
2. Decreased Launch Costs
3. Increased Payload Fractions
4. Increased Returns on Investment (in Terms of Mission Lifetime and/or Payload Objectives)
5. Increased Spacecraft Maneuverability
6. Increased Launch Opportunities

Due to these advantages, low-thrust propulsion has been used as the primary and secondary means of propulsion in almost every space application, from small satellites to heavier low Earth orbit (LEO) and geosynchronous equatorial orbit (GEO) satellites to interplanetary spacecraft. However, as a result of the increasing demand for telecommunication services (both commercial and military), a majority of spacecraft launched with low-thrust propulsion have been GEO satellites. In fact, the percentage of GEO satellites launched with low-thrust propulsion has increased from 10 percent in 1981 to almost 50 percent as of 2018 and, since 1998, more than 30 percent of operational GEO satellites use some form of low-thrust propulsion [3, 4]. Thus, this research was, primarily, focused on the low-thrust, orbit-raising problem. Until recent years, the lack of commercial incentives for developing and launching LEO satellites combined with the rather limited benefits of implementing low-thrust propulsion on LEO satellites (compared to GEO satellites) has stalled the growth of LEO satellites launched with low-thrust propulsion, with the exception of the Iridium satellite constellation launched during 1997 and 1998 [5]. However, the desire for high-speed, low-latency, broadband internet across the world, even in rural and remote areas, has increased the demand for commercial LEO satellites with low-thrust propulsion. In fact, companies such as SpaceX with Starlink [6] and OneWeb [7] are currently developing and launching such satellites. Low-thrust propulsion has even been implemented on small satellites (defined here as satellites less than 50 kilograms) and interplanetary spacecraft, though to a much lesser extent than GEO and LEO satellites. However, as the potential of low-thrust propulsion for small satellites and interplanetary spacecraft is recently being recognized and proven, the use of such satellites launched with low-thrust propulsion can be expected to increase significantly [3, 4].

As the use of spacecraft launched with low-thrust propulsion can be expected to continue increasing significantly, so can the need for quick, accurate, and robust low-thrust trajectory optimization as well as autonomous low-thrust control. Due to lower thrust magnitudes and longer transfer times, low-thrust trajectory optimization via traditional optimization techniques (i.e., direct and indirect methods) can be complex as well as computationally-expensive and time-consuming…this is where artificial neural networks come into play. Artificial neural networks can provide an alternative to such techniques. In fact, artificial neural networks, as well as other machine learning techniques, have been used in a wide range of aerospace-related applications, including low-thrust trajectory design and optimization, with high levels of success. However, though the use of artificial neural networks trained by supervised learning models for applications related to low-thrust trajectory design and optimization is extensive, previous research has largely focused on shallow or deep feedforward architectures with little emphasis placed on recurrent ones, even though recurrent architectures are inherently more suited to time histories due to the presence of internal memory. In fact, to the best of the author's knowledge, recurrent artificial neural networks have rarely been applied and only to attitude control and landing problems (i.e., modeling image-control relationships) or within reinforcement learning models. Thus, as previously mentioned, the overall objective of this research was to investigate the potential of recurrent artificial neural networks, specifically LSTM artificial neural networks, for low-thrust trajectory design and optimization with respect to their feedforward equivalents. As a majority of spacecraft launched with low-thrust propulsion have been GEO satellites, the focus of this research was, primarily, on the low-thrust, orbit-raising problem for both time-optimal and fuel-optimal transfers. As previously mentioned, a total of four problems were considered: the time-optimal open-loop case, the time-optimal closed-loop case, the fuel-optimal open-loop case, and the fuel-optimal closed-loop case.

1.3 Outline

In this dissertation, Chapter 2 provides some general information on low-thrust propulsion and trajectory optimization while Chapter 3 provides some general information on artificial neural networks with an emphasis on LSTM artificial neural networks. Chapter 3 also contains previous research on the use of artificial neural networks for low-thrust trajectory design and optimization. Chapter 4 describes the data used to tune, train, validate, and test the LSTM artificial neural networks as well as the implementation and design of those LSTM artificial neural networks and their feedforward equivalents. Chapter 5 discusses the results for the LSTM artificial neural networks and their feedforward equivalents while Chapter 6 presents the conclusions from these results with respect to the scientific contributions.

2 Low-Thrust Propulsion and Trajectory Optimization

In this chapter, some general information on low-thrust propulsion, specifically the types and history of low-thrust propulsion in space, and low-thrust trajectory optimization as well as direct and indirect methods for low-thrust trajectory optimization will be provided.

2.1 Low-Thrust Propulsion

Low-thrust propulsion has specific impulses typically an order of magnitude or higher than traditional chemical propulsion [1]. Higher specific impulses result in lower fuel consumptions, which correspond to greater payload fractions and lower launch costs as well as other advantages [2-4]. Due to these advantages, low-thrust propulsion has been used as the primary and secondary means of propulsion in almost every space application, from small satellites to heavier low Earth orbit (LEO) and geosynchronous equatorial orbit (GEO) satellites to interplanetary spacecraft. The types of low-thrust propulsion as well as the history of low-thrust propulsion in space will be discussed here.

2.1.1 Types of Low-Thrust Propulsion in Space

In general, there are five types of low-thrust propulsion: ion thrusters, pulsed plasma thrusters, resistojets, Hall thrusters, and arcjet thrusters [3, 4]. Each of these types of low-thrust propulsion will be defined below. Further information can be found in Reference 8 and Reference 9.

1. *Ion Thrusters.* Ion thrusters produce thrust by accelerating ions with an electrostatic field.
2. *Pulsed Plasma Thrusters.* Pulsed plasma thrusters produce thrust by accelerating plasma with an electromagnetic field.

3. *Resistojets.* Resistojets produce thrust by electrothermally heating the propellant using resistance heaters and accelerating through a nozzle.
4. *Hall Thrusters.* Hall thrusters, a type of ion thrusters, produce thrust by accelerating ions with a combination of electrostatic and electromagnetic fields.
5. *Arcjet Thrusters.* Arcjet thrusters produce thrust by electrothermally heating the propellant using an electrical discharge and accelerating through a nozzle.

As shown in Figure 2.1, spacecraft with low-thrust propulsion have primarily been GEO satellites, followed by LEO satellites, small satellites (defined here as satellites less than 50 kilograms), and interplanetary spacecraft [3, 4]. The most used type of low-thrust propulsion for GEO satellites is Hall thrusters (38 percent), followed by resistojets (25 percent), ion thrusters (19 percent), and arcjet thrusters (18 percent) [3, 4]. Resistojets are the most used type of low-thrust propulsion for LEO satellites, though the use of Hall thrusters for LEO satellites has recently been increasing [3, 4]. Small satellites have commonly used variants of pulsed plasma thrusters and ion thrusters as well as resistojets while interplanetary spacecraft have commonly used ion thrusters [3, 4].

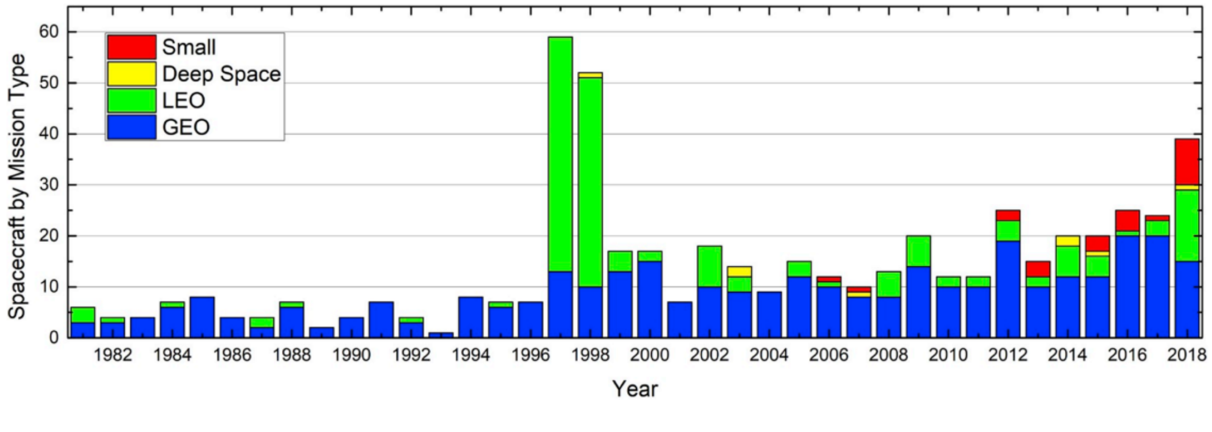


Figure 2.1: Mission Types of Satellites with Low-Thrust Propulsion (as of 2018) [4]

2.1.2 History of Low-Thrust Propulsion in Space

Overall, until the 1980s, only government spacecraft implemented low-thrust propulsion. The first technological demonstration of low-thrust propulsion in space was SERT 1, a suborbital satellite launched in 1964 [3, 4, 10]. Later that same year, the Soviet Union interplanetary spacecraft, Zond 2, became the first spacecraft to operationally use low-thrust propulsion by means of six pulsed plasma thrusters for attitude control [3, 4, 8, 11]. The United States, however, did not use pulsed plasma thrusters on spacecraft until 1968, during which the Lincoln Experimental Satellites were developed and launched with the objective of improving satellite communications [3, 4, 8, 12]. The Lincoln Experimental Satellites used four pulsed plasma thrusters for east-west station-keeping [8, 12]. Resistojets, on the other hand, were first used in 1965 aboard the United States Air Force Vela satellites, which were developed and launched to detect nuclear explosions as well as to study different forms of radiation and charged particles [3, 4, 8, 13]. The first satellite to use Hall thrusters was one of the Soviet Union's Meteor meteorological satellites, launched in 1971 [3, 4, 14].

The first commercial use of low-thrust propulsion was COMSAT's Intelsat 5 satellites, a series of communications satellites launched starting in the early 1980s which used resistojets for north-south station-keeping [3, 4, 8, 15]. AT&T's communications satellite, Telstar 401, was launched in 1993 and became the first satellite to use arcjet thrusters for north-south station-keeping [3, 4, 8, 16]. Telstar 401 was also among the first spacecraft to provide a clear demonstration of the advantages of implementing low-thrust propulsion, specifically the resulting propellent mass and launch cost reductions [8, 16].

Low-thrust propulsion was not implemented as the primary means of propulsion (i.e., for maneuvers more intensive than attitude control and station-keeping) until the late 1990s. The first spacecraft to do so was NASA's Deep Space 1, an interplanetary spacecraft launched in 1998 which used ion thrusters to visit the asteroid Braille and the comet Borrelly [3, 4, 8, 17]. In 2001, ESA's communications satellite Artemis, due to a problem with the final stage of its launch vehicle, became the first satellite to use low-thrust propulsion, specifically ion thrusters, for a significant portion of a geosynchronous transfer orbit (GTO) to GEO transfer [3, 4, 18]. Later, in 2003, ESA's SMART-1 became the first spacecraft to complete a full GTO to GEO transfer using low-thrust propulsion, specifically Hall thrusters, on its way to orbit the moon [3, 4, 19]. Finally, the first all-electric satellite was Boeing's communications satellite ABS-3A, launched in 2015 with ion thrusters [3, 4, 20].

2.2 General Low-Thrust Trajectory Optimization

The continuous-time trajectory optimization problem will be discussed here, following the format in Reference 21 and Reference 22. In the continuous-time trajectory optimization problem, the decision variables are determined such that the objective function (J), or performance index, is minimized. The objective function consists of a boundary objective (φ), or the cost due to the initial and final states, as well as a path integral (L), or the cost due to the trajectory between the initial and final states.

$$J = \varphi\left(t_0, \boldsymbol{x}(t_0), t_f, \boldsymbol{x}(t_f)\right) + \int_{t_0}^{t_f} L\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t)\big)\mathrm{d}t \quad \text{Equation 2.1 [21, 22]}$$

The objective function is subject to the equations of motion ($\dot{\boldsymbol{x}}$), or system dynamics, which are dependent on the decision variables. Typically, the decision variables consist of the time (t), the state variables ($\boldsymbol{x}$), and the control variables ($\boldsymbol{u}$).

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t)\big) \quad \text{Equation 2.2 [21, 22]}$$

The objective function is also subject to the path constraints ($\boldsymbol{h}$) and the boundary constraints ($\boldsymbol{g}$). Path constraints constrain the trajectory between the initial and final states while boundary constraints constrain the initial and/or final states themselves.

$$\boldsymbol{h}\big(t, \boldsymbol{x}(t), \boldsymbol{u}(t)\big) \leq \boldsymbol{0}$$ Equation 2.3 [21, 22]

$$\boldsymbol{g}\left(t_0, t_f, \boldsymbol{x}(t_0), \boldsymbol{x}(t_f)\right) \leq \boldsymbol{0}$$ Equation 2.4 [21, 22]

Finally, there can be bounds on the decision variables as well as bounds on the initial and/or final states.

$$\boldsymbol{x}_{lower} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{upper}$$ Equation 2.5 [21, 22]

$$\boldsymbol{u}_{lower} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{upper}$$ Equation 2.6 [21, 22]

$$t_{lower} \leq t_0 < t_f \leq t_{upper}$$ Equation 2.7 [21, 22]

$$\boldsymbol{x}_{0_{lower}} \leq \boldsymbol{x}(t_0) \leq \boldsymbol{x}_{0_{upper}}$$ Equation 2.8 [21, 22]

$$\boldsymbol{x}_{f_{lower}} \leq \boldsymbol{x}(t_f) \leq \boldsymbol{x}_{f_{upper}}$$ Equation 2.9 [21, 22]

2.2.1 Analytic Solutions

Analytic solutions to the continuous-time trajectory optimization problem do not exist for low-thrust propulsion, aside from special cases of the low-thrust, orbit-raising problem [1]. In the early 1960s, Edelbaum derived the first analytic solution to the low-thrust, orbit-raising problem for a fixed-time, fuel-optimal transfer between non-coplanar, circular orbits [23]. In his derivation, Edelbaum assumed continuous constant acceleration and a constant yaw profile [23]. Wiesel and Alfano extended Edelbaum's derivation, allowing both the acceleration and the yaw angle to change [24]. In the late 1990s, Kechichian reformulated Edelbaum's derivation as a time-optimal transfer between, again, non-coplanar circular orbits [25]. Kechichian [26] as well as Colasurdo and Casalino [27] derived analytic solutions to the low-thrust, orbit-raising problem for a transfer between coplanar, circular orbits in the presences of satellite eclipses by constraining the thrust profile to maintain a zero eccentricity throughout the transfer. Even orbit perturbations (due to the oblateness of Earth) and altitude constraints have been considered [28, 29]. However, due to the assumptions and/or constraints present in Edelbaum's derivation as well as subsequent extensions, analytic solutions to the low-thrust, orbit-raising problem remain suboptimal.

2.3 Direct Low-Thrust Trajectory Optimization

In the continuous-time trajectory optimization problem, direct methods transcribe the system dynamics from a set of differential equations to a set of equality constraints, converting the continuous optimal control problem into a nonlinear programming problem. With direct methods, the time (t) is discretized into N points, or nodes, with a step size of h_k.

$$t_I = t_1 <\ldots< t_k <\ldots< t_N = t_f$$ Equation 2.10 [1, 21]

$$h_k \equiv t_{k+1} - t_k$$ Equation 2.11 [1, 21]

If the time is discretized uniformly, the step size can be defined as follows.

$$h_k \equiv \frac{t_N - t_1}{N-1}$$ Equation 2.12 [1]

The remaining decision variables, typically the state variables ($\boldsymbol{x}$) and the control variables ($\boldsymbol{u}$), are discretized using the discretization scheme used for the time.

$$\begin{gathered} \boldsymbol{x}_k = \boldsymbol{x}(t_k) \\ \therefore \boldsymbol{x}_I = \boldsymbol{x}_1 <\ldots< \boldsymbol{x}_k <\ldots< \boldsymbol{x}_N = \boldsymbol{x}_f \end{gathered}$$ Equation 2.13 [1, 21]

$$\begin{gathered} \boldsymbol{u}_k = \boldsymbol{u}(t_k) \\ \therefore \boldsymbol{u}_I = \boldsymbol{u}_1 <\ldots< \boldsymbol{u}_k <\ldots< \boldsymbol{u}_N = \boldsymbol{u}_f \end{gathered}$$ Equation 2.14 [1, 21]

The changes in the discretized state variables are set equal to the integrated system dynamics ($\dot{\boldsymbol{x}}$), resulting in a set of equality constraints. The integrated system dynamics are approximated using some numerical integration scheme. The order of accuracy of the numerical integration scheme determines the order of accuracy of the direct method solution.

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k = \int_{t_k}^{t_{k+1}} \dot{\boldsymbol{x}} \mathrm{d}t \qquad \text{Equation 2.15 [1, 21]}$$

The objective function can be discretized as well. Any boundary conditions can be applied as equality constraints on the first and/or last nodes while any path conditions can be applied as inequality constraints on the appropriate, if not all, nodes.

Two direct methods commonly used for low-thrust trajectory optimization are direct collocation methods and direct shooting methods. Direct collocation methods are implicit approaches which approximate the state and control variables as polynomial splines. Direct shooting methods are explicit approaches which follow an "aim and shoot" approach. Direct collocation methods are typically more robust than direct shooting methods, especially for applications with path constraints or applications in which the structure of the control law is not known a priori. Direct shooting methods, however, are typically more accurate than direct collocation methods, especially for applications with simple control laws or applications in which the control law (i.e., a good initial guess) is known to some extent. [1, 21, 22]

Direct collocations methods are some of the "best known" and "most implemented" direct methods and, as such, have been applied to the low-thrust, orbit-raising problem as well as lunar and interplanetary transfers [1, 2, 53]. Due to this success, direct collocation methods were chosen as the method of trajectory optimization in the generation of the training, validation, and test data. For more information on the generation of the training, validation, and test data, refer to Chapter 4. Two direct collocation methods (trapezoidal collocation and Hermite-Simpson collocation) will be discussed in more detail here. Higher-order methods, such as orthogonal and/or pseudospectral collocation, do exist [1, 21]. However, such methods will not be discussed here.

2.3.1 Trapezoidal Collocation

With trapezoidal collocation, the system dynamics are converted into a set of collocation constraints using trapezoidal quadrature. Trapezoidal quadrature, or the trapezoidal rule, approximates a definite integral as a trapezoid.

$$\int_{x_0}^{x_1} f(x)\mathrm{d}x \approx (x_1 - x_0) * \frac{1}{2}[f(x_0) + f(x_1)] \qquad \text{Equation 2.16 [30]}$$

As follows, with trapezoidal collocation, the system dynamics ($\boldsymbol{f}$) as well as the control variables ($\boldsymbol{u}$) are approximated as linear splines. Since the system dynamics are approximated as linear splines, the state variables ($\boldsymbol{x}$) are represented by quadratic splines.

$$\dot{\boldsymbol{x}} = \boldsymbol{f} \qquad \text{Equation 2.17 [21]}$$

$$\int_{t_k}^{t_{k+1}} \dot{\boldsymbol{x}}\mathrm{d}t = \int_{t_k}^{t_{k+1}} \boldsymbol{f}\mathrm{d}t \qquad \text{Equation 2.18 [21]}$$

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k = (t_{k+1} - t_k) * \frac{1}{2}(\boldsymbol{f}_k + \boldsymbol{f}_{k+1}) \qquad \text{Equation 2.19 [21]}$$

$$h_k \equiv t_{k+1} - t_k \qquad \text{Equation 2.20 [21]}$$

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k = \frac{1}{2}(h_k)(\boldsymbol{f}_k + \boldsymbol{f}_{k+1}) \qquad \text{Equation 2.21 [21]}$$

If appropriate, the objective function can be approximated using trapezoidal quadrature as well and/or converted to a summation.

2.3.2 Hermite-Simpson Collocation

Hermite-Simpson collocation is an order of accuracy higher than trapezoidal collocation. With Hermite-Simpson collocation, the system dynamics are converted into a set of collocation constraints using Simpson quadrature. Simpson quadrature, or the Simpson rule, approximates a definite integral as a parabolic arc.

$$\int_{x_0}^{x_1} f(x)\mathrm{d}x \approx (x_1 - x_0) * \frac{1}{6}\left[f(x_0) + 4f\left(\frac{x_0+x_1}{2}\right) + f(x_1)\right] \quad \text{Equation 2.22 [31]}$$

As follows, with Hermite-Simpson collocation, the system dynamics ($\boldsymbol{f}$) as well as the control variables ($\boldsymbol{u}$) are approximated as quadratic splines. Since the system dynamics are approximated as quadratic splines, the state variables ($\boldsymbol{x}$) are represented by cubic Hermite splines.

$$\dot{\boldsymbol{x}} = \boldsymbol{f} \qquad \text{Equation 2.23 [1, 21]}$$

$$\int_{t_k}^{t_{k+1}} \dot{\boldsymbol{x}}\mathrm{d}t = \int_{t_k}^{t_{k+1}} \boldsymbol{f}\mathrm{d}t \qquad \text{Equation 2.24 [1, 21]}$$

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k = (t_{k+1} - t_k) * \frac{1}{6}\left(\boldsymbol{f}_k + 4\boldsymbol{f}_{k+\frac{1}{2}} + \boldsymbol{f}_{k+1}\right) \qquad \text{Equation 2.25 [1, 21]}$$

$$h_k \equiv t_{k+1} - t_k \qquad \text{Equation 2.26 [1, 21]}$$

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k = \frac{1}{6}(h_k)\left(\boldsymbol{f}_k + 4\boldsymbol{f}_{k+\frac{1}{2}} + \boldsymbol{f}_{k+1}\right) \qquad \text{Equation 2.27 [1, 21]}$$

For Hermite-Simpson collocation, a second collocation constraint is necessary to enforce the system dynamics at the midpoint of the time interval. The derivation of the second collocation constraint is as follows. Since the system dynamics are approximated as quadratic splines and the state variables are represented by cubic Hermite splines, the state variables and their derivatives can be represented by general polynomials.

$$\begin{gathered}\boldsymbol{x}(t) = a_{k_0} + a_{k_1}t + a_{k_2}t^2 + a_{k_3}t^3 \\ \dot{\boldsymbol{x}}(t) = a_{k_1} + 2a_{k_2}t + 3a_{k_3}t^2\end{gathered} \qquad \text{Equation 2.28 [1]}$$

The time interval can be shifted from $[t_k, t_{k+1}]$ to $[0, h_k]$ where $h_k \equiv t_{k+1} - t_k$, and the polynomial representations of the state variables and their derivatives can be written in matrix form.

$$\begin{bmatrix} \boldsymbol{x}(0) \\ \dot{\boldsymbol{x}}(0) \\ \boldsymbol{x}(h_k) \\ \dot{\boldsymbol{x}}(h_k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_k \\ \dot{\boldsymbol{x}}_k \\ \boldsymbol{x}_{k+1} \\ \dot{\boldsymbol{x}}_{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{f}_k \\ \boldsymbol{x}_{k+1} \\ \boldsymbol{f}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & h_k & {h_k}^2 & {h_k}^3 \\ 0 & 1 & 2h_k & 3{h_k}^2 \end{bmatrix} \begin{bmatrix} a_{k_0} \\ a_{k_1} \\ a_{k_2} \\ a_{k_3} \end{bmatrix} \qquad \text{Equation 2.29 [1]}$$

The coefficients of the polynomial representations of the state variables and their derivatives can be determined by calculating the inverse of the matrix of coefficients.

$$\begin{bmatrix} a_{k_0} \\ a_{k_1} \\ a_{k_2} \\ a_{k_3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{-3}{{h_k}^2} & \frac{-2}{h_k} & \frac{3}{{h_k}^2} & \frac{-1}{h_k} \\ \frac{2}{{h_k}^3} & \frac{1}{{h_k}^2} & \frac{-2}{{h_k}^3} & \frac{1}{{h_k}^2} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_k \\ \boldsymbol{f}_k \\ \boldsymbol{x}_{k+1} \\ \boldsymbol{f}_{k+1} \end{bmatrix}$$ Equation 2.30 [1]

By knowing the coefficients of the polynomial representations of the state variables and their derivatives, the state variables and their derivatives at the midpoint of the time interval, $t = \frac{1}{2}(t_k + t_{k+1}) = \frac{1}{2}h_k$, can be determined.

$$\boldsymbol{x}_{k+\frac{1}{2}} = \frac{1}{2}(\boldsymbol{x}_k + \boldsymbol{x}_{k+1}) + \frac{1}{8}h_k(\boldsymbol{f}_k - \boldsymbol{f}_{k+1})$$
$$\dot{\boldsymbol{x}}_{k+\frac{1}{2}} = -\frac{3}{2h_k}(x_k - x_{k+1}) - \frac{1}{4}(\boldsymbol{f}_k + \boldsymbol{f}_{k+1})$$ Equation 2.31 [1, 21]

The control variables at the midpoint of the time interval, however, can be determined using linear interpolation.

$$\boldsymbol{u}_{k+\frac{1}{2}} = \frac{1}{2}(\boldsymbol{u}_k + \boldsymbol{u}_{k+1})$$ Equation 2.32 [1, 21]

If appropriate, the objective function can be approximated using Hermite-Simpson quadrature as well and/or converted to a summation.

2.4 Indirect Low-Thrust Trajectory Optimization

In the continuous-time trajectory optimization problem, indirect methods use Pontryagin's maximum principle from the calculus of variations to analytically construct the necessary conditions for optimality, converting the continuous optimal control problem into a two-point boundary-value problem. Assume the following objective function, which is a simplified version of the one presented in Equation 2.1.

$$J = \varphi\big(\boldsymbol{x}(t_f), t_f\big) + \int_{t_0}^{t_f} L(\boldsymbol{x}, \boldsymbol{u}, t)\mathrm{d}t$$ Equation 2.33 [1, 2]

The Hamiltonian (H) can be defined as follows. In this equation, $\boldsymbol{\lambda}$ is the adjoint, or costate, multipliers.

$$H = L + \boldsymbol{\lambda}^T \boldsymbol{f}$$ Equation 2.34 [1, 2]

The Euler-Lagrange equations, or the necessary conditions for optimality, can be constructed as follows. In this system of equations, the first equation is equivalent to Equation 2.2. The second equation describes the costate dynamics while the third equation is a direct application of Pontryagin’s maximum principle, which states the control variables must optimize the Hamiltonian. [1, 2]

$$\begin{cases} \dot{\boldsymbol{x}} = \left(\frac{\partial H}{\partial \boldsymbol{\lambda}}\right)^T \\ \dot{\boldsymbol{\lambda}} = -\left(\frac{\partial H}{\partial \boldsymbol{x}}\right)^T \\ \left(\frac{\partial H}{\partial \boldsymbol{u}}\right) = 0 \end{cases}$$ Equation 2.35 [1, 2]

Finally, the boundary constraints ($\boldsymbol{\psi}$) can be defined as follows. In this equation, similar to $\boldsymbol{\lambda}$, $\boldsymbol{v}$ is the final boundary constraints multipliers.

$$\boldsymbol{\lambda}(t_f) = \left[\left(\frac{\partial \boldsymbol{\varphi}}{\partial x}\right) + \boldsymbol{v}^T \left(\frac{\partial \boldsymbol{\psi}}{\partial x}\right)\right]_{t=t_f}$$ Equation 2.36 [1, 2]

2.5 Comparison of Direct and Indirect Methods

In general, direct methods are easier to construct and solve as well as more robust (i.e., able to converge, even with poor initial guesses) than indirect methods. Indirect methods, however, are more accurate than direct methods so long as the initial guesses are “good” enough to achieve convergence. Also, as indirect methods analytically construct the necessary conditions for optimality, the solutions of indirect methods are guaranteed to be local extrema. However, the addition of costate variables doubles the size of the continuous-time trajectory optimization problem as the costate variables need to be solved as well as initialized. This only complicates the initialization of indirect methods further, especially as costate variables lack the physical significance of state variables. [1, 2, 21, 22]

3 Artificial Neural Networks

Information on artificial neural networks, or neural networks, can be found in Reference 32 and Reference 33. Inspired by the human brain, neural networks are data-driven, computational models capable of nonlinear, input-output mapping. Neural networks are able to learn as well as generalize without a priori knowledge and have a multitude of useful properties, such as flexibility, adaptivity, uniformity, and fault tolerance. As a result, neural networks have been used for a variety of applications (both inside and outside the area of astrodynamics), which include pattern association, pattern recognition, and function approximation. More formally, the following definition of a neural network can be found in Chapter I.1 of Haykin [32].

A neural network is a massively parallel distributed processor made up of simple processing units that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1. *Knowledge is acquired by the network from its environment through a learning process.*
2. *Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.*

The first neural network was algorithmically described in 1958 [34]. Rosenblatt's perceptron, which implemented the McCulloch-Pitts model for a nonlinear neuron [35], consisted of a single neuron and was trained by supervised learning to perform binary classification [34]. Multilayer perceptrons, which consist of one or more hidden layers, did not become popular until after the development of the back-propagation algorithm in the late 1980s [36].

3.1 Model of Neuron

A neuron, which is the information-processing unit of a neural network, consists of three elements: a set of synapses, an adder, and an activation function. The set of synapses consists of a set of input signals and the respective weights for those input signals. The adder sums the weighted input signals, and the activation function, following some logic, limits the overall output signal of the neuron to be some finite value. Figure 3.1 shows the model of a neuron. [32, 33]

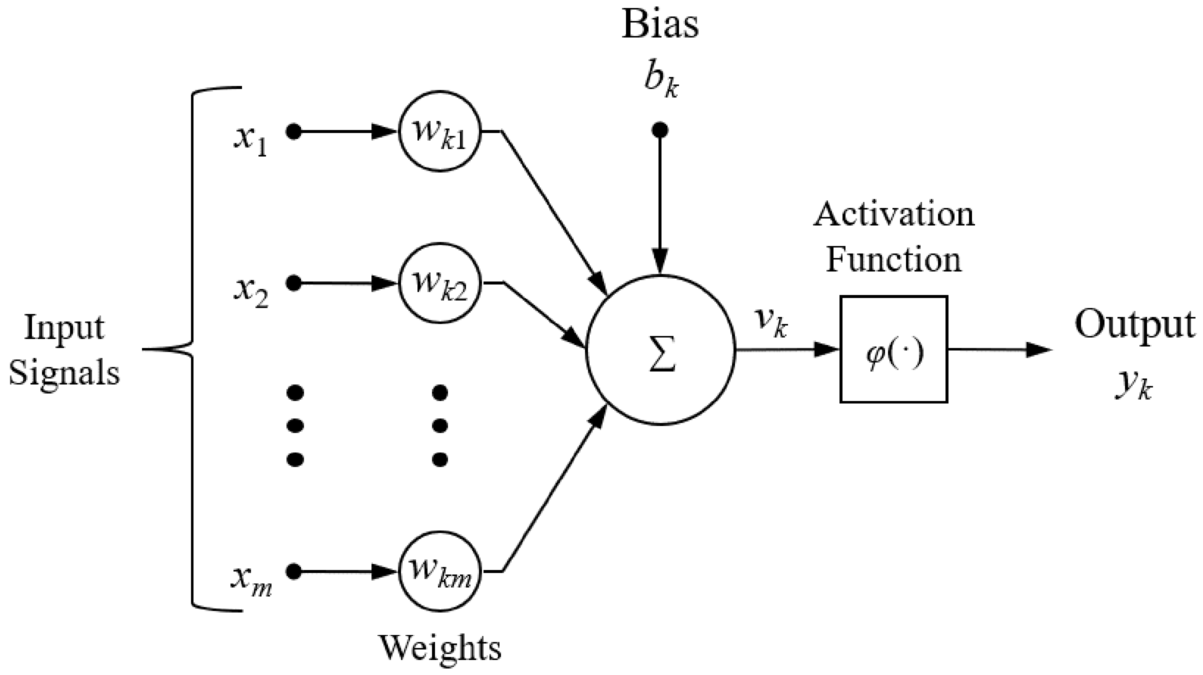


Figure 3.1: Model of Neuron [32]

Mathematically, the model of a neuron, denoted as neuron k, with m-number of input signals can be written as follows.

$$y_k = \varphi\left(\sum_{j=1}^{m} w_{kj} x_j + b_k\right) \qquad \text{Equation 3.1 [32]}$$

In this equation, x_j are the input signals, w_{kj} are the respective weights for those input signals, and y_k is the overall output signal of the neuron. b_k is the bias which, if present, applies an affine transformation to the input for the activation function [32, 33]. The bias can also be modeled as a synapse where the input signal is equal to unity and the respective weight is equal to the desired bias [32, 33]. φ is the activation function. Many types of activation functions exist; some of the more popular ones will be discussed in more detail here.

3.1.1 Types of Activation Functions

The threshold, or Heaviside, function will set the overall output signal of the neuron equal to either one or zero, following the binary logic shown in following equation. Note, in this equation, as well as in subsequent equations, v is simply the input for the activation function.

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}$$ Equation 3.2 [32]

The signum function, which is a variant of the Heaviside function, will set the overall output signal of the neuron equal to positive one, zero, or negative one, following the logic shown in the following equation.

$$\varphi(v) = \begin{cases} +1 & \text{if } v > 0 \\ 0 & \text{if } v = 0 \\ -1 & \text{if } v < 0 \end{cases}$$ Equation 3.3 [32]

The sigmoid function, shown in the following equation, is a continuous version of the Heaviside function. In fact, as the input for the sigmoid function approaches infinity, the sigmoid function will approach the Heaviside function.

$$\varphi(v) = \frac{1}{1+\exp(-v)}$$ Equation 3.4 [32]

The hyperbolic tangent function, shown in the following equation, is a continuous version of the signum function. Similar to the sigmoid function, as the input for the hyperbolic tangent function approaches infinity, the hyperbolic tangent function will approach the signum function.

$$\varphi(v) = \tanh(v)$$ Equation 3.5 [32]

The rectified linear unit function, which is another variant of the Heaviside function, will set the overall output signal of the neuron equal to a positive value or zero, following the binary logic shown in the following equation.

$$\varphi(v) = \begin{cases} v & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases} \qquad \text{Equation 3.6 [37]}$$

3.2 Neural Network Architectures and Learning Processes

Neural networks have, at a minimum, one input layer and one output layer. The input layer contains source neurons whereas the output layer contains computation neurons. In other words, no calculations are performed by the input layer. Hidden layers can be added between the input and the output layer. Hidden layers, which contain computation neurons, help neural networks to model nonlinear behavior and higher-order statistics. A shallow neural network is defined here as having two or less hidden layers while a deep neural network is defined as have three or more hidden layers. [32, 33]

In general, there are two classes of neural network architectures: feedforward neural networks and recurrent neural networks. The difference between a feedforward neural network and a recurrent neural network is that a feedforward neural network does not have a feedback loop whereas a recurrent neural network has, at a minimum, one feedback loop. Feedback loops, which help neural networks to model nonlinear behavior (as well as time dependencies), can have a significant impact on overall performance. Figure 3.2 shows an example of a neural network with a feedforward architecture while Figure 3.3 shows an example of a neural network with a recurrent architecture. [32, 33]

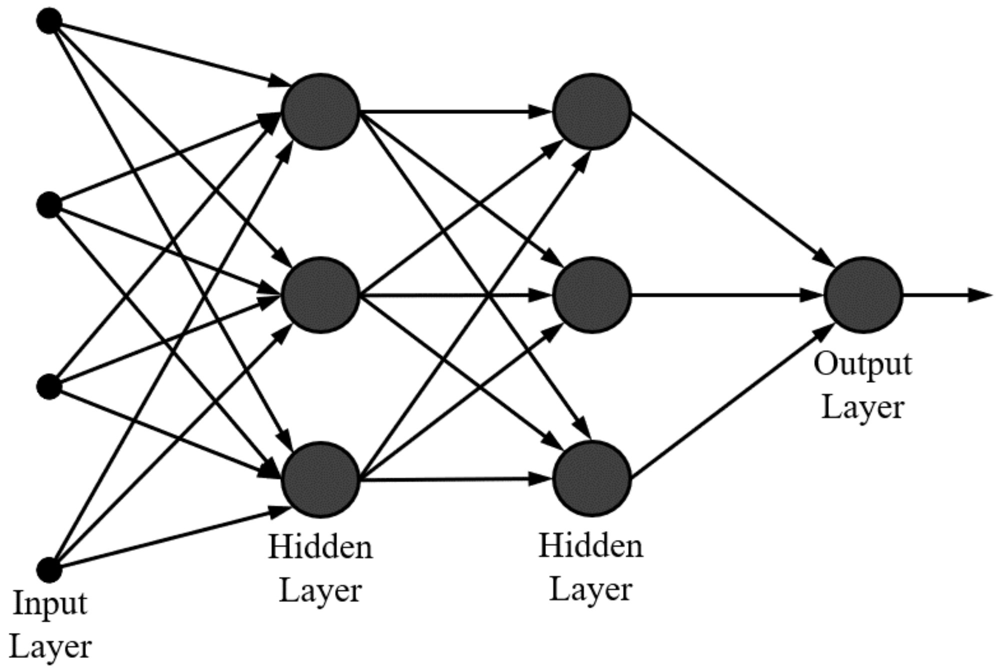


Figure 3.2: Example of Feedforward Neural Network with Two Hidden Layers [32]

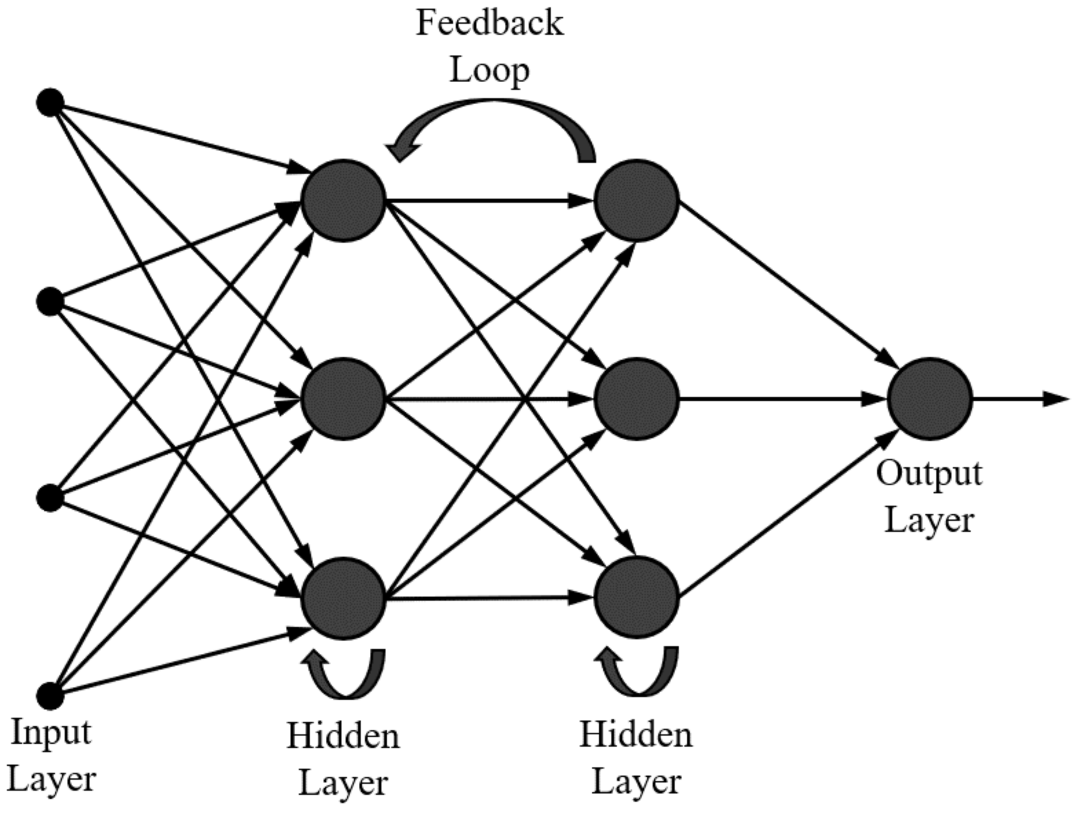


Figure 3.3: Example of Recurrent Neural Network with Two Hidden Layers [32]

There are three classes of learning processes: supervised learning, reinforcement learning, and unsupervised learning. In supervised learning (shown in Figure 3.4), the learning machine learns how to behave with an unknown environment through the help of a “teacher”, or a set of input-output pairs, which provides the current state of the environment and the desired response for that state. Using the error, or the difference between the desired response for the current state and the actual response from the learning machine for that state, the parameters of the learning machine are updated, resulting in the learning machine, ideally, learning how to behave such that the error is minimized. [32, 33]

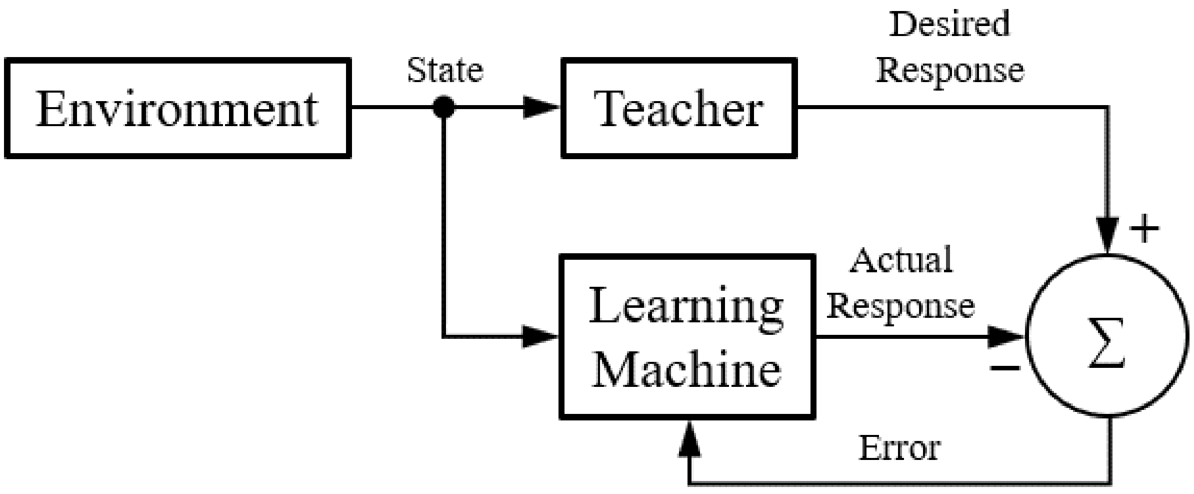


Figure 3.4: Supervised Learning Diagram [32]

In reinforcement learning (shown in Figure 3.5), the learning machine learns how to behave by interacting with the unknown environment. The learning machine, based on the current state of the environment, selects an action following some policy. Then, the learning machine receives feedback from the environment usually in the form of a reward and the future state of the environment. This process continues, resulting in the learning machine, ideally, learning how to behave such that the cumulative reward is maximized. [32, 33]

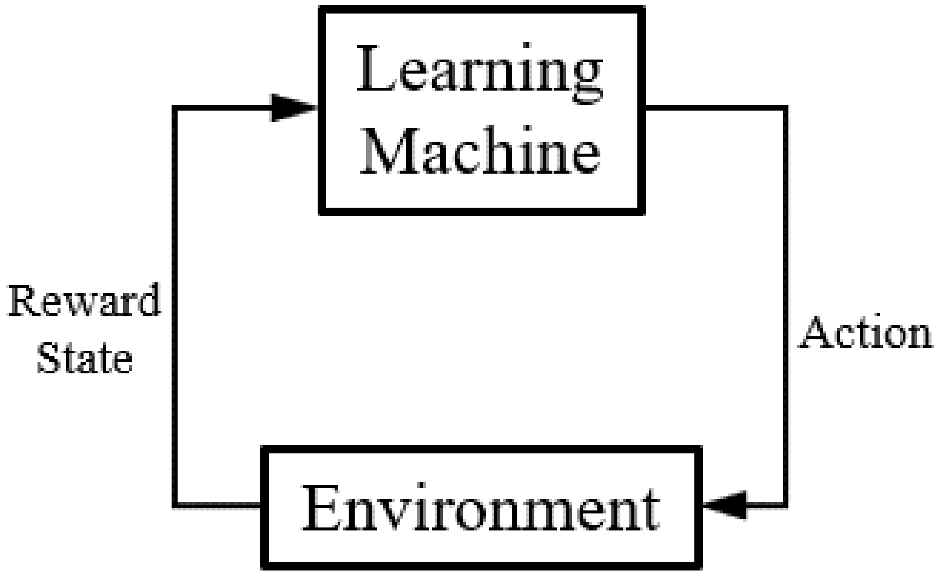


Figure 3.5: Reinforcement Learning Diagram [32]

Unsupervised learning (shown in Figure 3.6) has neither a teacher nor a feedback loop to help guide the learning process. Rather, a task-independent measure is used to optimize the parameters of the learning machine. [32, 33]

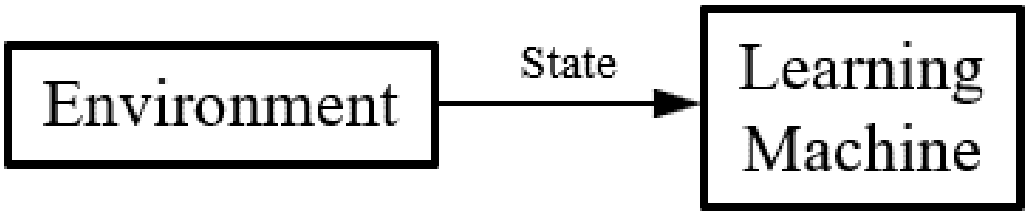


Figure 3.6: Unsupervised Learning Diagram [32]

3.3 LSTM Neural Networks

In 1997, Hochreiter and Schmidhuber [38] developed long short-term memory (LSTM) neural networks, a subclass of recurrent neural networks, to solve the vanishing-gradient problem, which decays the sensitivity of a recurrent neural network to inputs over training. Due to the vanishing-gradient problem, a recurrent neural network will “forget” information, resulting in the loss of long-term dependencies. Hochreiter and Schmidhuber solved the vanishing-gradient problem by enforcing a constant error back-propagation by using “memory” units with multiplicative gates, which opened and closed to access that constant error flow [38].

Information on LSTM neural networks can be found in Reference 39 and Reference 40. As previously mentioned, LSTM neural networks use “memory” units, which subsequently consist of memory cells as well as multiplicative (input, output, and forget) gates. The activation function for a gate is typically the sigmoid function. In other words, the state of the gate is limited from 0 (i.e., closed) to 1 (i.e., open). Overall, if the input gate is closed and the forget gate is open, the memory cell will “remember” information from previous inputs rather than information from current inputs, which makes that information available for future use if the output gate is open. However, if the forget gate is closed, the memory cell will reset and “forget” information from previous inputs. Mathematically, the model of a memory unit for an LSTM neural network can be written as follows. [39, 40]

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$$ Equation 3.7 [40]

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right)$$ Equation 3.8 [40]

$$c_t = f_t c_{t-1} + i_t\, tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$ Equation 3.9 [40]

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o)$$ Equation 3.10 [40]

$$h_t = o_t\, tanh(c_t)$$ Equation 3.11 [40]

In these equations, i is the state of the input gate, o is the state of output gate, f is the state of the forget gate, and c is the state of the memory cell. σ is the activation function for the gates. x are the input signals and h is the output signal from the memory cell while W are the weight matrices and b are the biases.

3.3.1 Back-Propagation and Back-Propagation Through Time

In general, neural networks use gradient-based optimization to minimize the loss function. A gradient-based optimization algorithm updates the weights of a neural network such that the value of the loss function moves in some direction, usually the direction of steepest descent (i.e., the negative of the gradient). Gradient-based optimization is implemented in neural networks through the use of the back-propagation algorithm. Information on the back-propagation algorithm can be found in Reference 32 and Reference 33. Overall, the back-propagation algorithm calculates the partial derivatives of the loss function with respect to the weights of the neural network through a repeated application of the chain rule, starting from the output layer and working backwards. During this backwards pass through the neural network, the weights of the neural network are updated. LSTM neural networks, as a subclass of recurrent neural networks, use an extension of the back-propagation algorithm…the back-propagation through time algorithm. The back-propagation through time algorithm propagates the partial derivatives of the loss function with a backwards and forwards pass through the neural network. [32, 33]

3.4 Previous Research

Schuster [41] as well as Izzo et al. [42] and Izzo et al. [43] provide historical overviews on the extent of artificial intelligence in astrodynamics. Overall, neural networks, as well as other machine learning techniques, have been used in a wide range of aerospace-related applications, from spacecraft trajectory design and optimization [44, 45] to spacecraft guidance, navigation, and control [46, 47] to space situational awareness [48-51] to mission operations [52-55]. Neural networks have also been used in a variety of entry, descent, and landing problems. However, such research has largely focused on spacecraft with traditional, impulsive, yet throttleable, propulsion [56-61]. Neural networks have even been used in reinforcement learning models as well as in combination with evolutionary optimization, such as evolutionary neurocontrol and other evolutionary neural control schemes. However, such research will not be discussed here as this research was focused on neural networks trained by supervised learning models for, specifically, applications related to low-thrust trajectory design and optimization.

In 2015, Sreesawet et al. proposed a neural-network based adaptive controller, which performed better than a traditional controller, for the attitude control of a low-thrust spacecraft during an orbit-raising transfer [62]. Ramos et al. used shallow feedforward neural networks to generate pork-chop plots for low-thrust interplanetary transfers from Earth to Mars [63]. Mereta et al. trained and compared various machine learning regressors, including shallow feedforward neural networks, to estimate the final spacecraft mass after a low-thrust transfer with multiple revolutions between two near-Earth asteroids [64]. In terms of the mean absolute error and the root-mean-square error, Mereta et al. found all the machine learning regressors outperformed the Lambert estimate by at least an order of magnitude [64]. However, the neural networks were not best regressor overall [64]. Sánchez-Sánchez and Izzo trained deep feedforward neural networks to learn the control policy for four cases of pinpoint landing: a quadcopter model, a mass-varying spacecraft with bounded thrust, a mass-varying spacecraft using reaction wheels for attitude control, and a mass-varying rocket with thrust vector control [65]. Sánchez-Sánchez and Izzo found the control policies predicted by the neural networks accurately represented the optimal control policies with high success rates and good generalization behaviors [65]. Parrish, in their dissertation, applied deep feedforward neural networks to cislunar and translunar trajectories [66]. As the neural networks were trained to map the difference in the current states from the nominal initial conditions to the difference in the current states to the nominal final conditions, the neural networks were able, in the presence of errors, to accurately correct the trajectories with respect to the nominal trajectory, reducing the overall error in the final states [66]. Izzo et al. used deep feedforward neural networks to satisfactorily (i.e., with small errors) model the guidance profile of a quadratic-optimal and mass-optimal interplanetary transfer from Earth to Mars [42].

In 2019, Izzo et al. used deep feedforward neural networks to predict, with high accuracies, the thrust profile and the fuel consumption (with and without enforced constraints) for a mass-optimal, interplanetary transfer from Earth to Venus [67]. Chen, in their thesis, trained and compared machine learning techniques, including deep feedforward neural networks, to estimate the fuel consumptions and transfer times for rendezvous-related problems [68]. Similar to Chen, Casey trained and compared machine learning regressors, including gradient boosting and feedforward neural networks, to identify transfer feasibility and predict the final spacecraft mass for use in a sequence evaluation and optimization scheme [69]. Song and Gong applied deep feedforward neural networks to the multi-target rendezvous problem by implementing the neural networks, which predicted the transfer times between near-Earth asteroids for solar sails with a mean relative error of less than 1 percent, in a Monte Carlo Tree Search [70]. Li et al. showed deep feedforward neural networks were capable of autonomous, time-optimal, many-revolution (long-duration) orbit-raising [71]. Cheng et al. applied multiscale deep feedforward neural networks in a real-time optimal control approach for a minimum-time interplanetary transfer from Earth to Mars via a solar sail and reported remarkable performance in terms of accuracy, efficiency, and robustness [72]. Li et al. trained deep feedforward neural networks to predict the time, initial costates, and control policy for time-optimal interplanetary transfers [73]. Li et al. found the neural networks were able to accurately predict the time and initial costates with errors less than a tenth of a percent and 2 percent, respectively [73]. When used as the initial solution to an indirection shooting method, the result was a 100 percent success rate and improved convergence efficiency

[73]. Li et al. also found the control policy predicted by the neural networks coincided well with the optimal control policy [73]. Zhu and Luo used multilayer perceptrons (deep feedforward neural networks) to evaluate transfer feasibility and estimate fuel consumption for short interplanetary transfers between virtual bodies [74]. Zhu and Luo showed the multilayer perceptrons, when compared to other machine learning techniques, were the best evaluators of transfer feasibility with an accuracy of more than 98 percent as well as the best approximators of fuel consumption with a relative estimation error of less than half of a percent [74].

In 2020, Sprague et al. presented an approach for an optimal controller with neural networks (capable of adapting online to dynamic objectives) and applied their approach to an inverted pendulum swing-up problem as well as a spacecraft orbit transfer problem [75]. With their approach, Sprague et al. were able to produce trajectories that were near-optimal [75]. Li et al. trained deep feedforward neural networks to estimate the transfer costs for three optimal control problems: the transfer times for time-optimal multitarget transfers, the fuel consumptions for fuel-optimal multitarget transfers, and the required velocity changes for multi-impulse multitarget transfers [76]. Li et al. showed the neural networks estimated the transfer times and fuel consumptions with a mean relative error of less than half of a percent and the required velocity changes with a mean relative error of less than 4 percent [76]. Rubinsztejn et al. applied deep feedforward neural networks to the missed thrust problem for three optimal control problems: two-body orbit-to-orbit transfers, two-body interplanetary transfers from Mars to Earth, and circular restricted three-body transfer from Earth to a Lyapunov orbit [77]. Rubinsztejn et al. showed the neural networks, which had a baseline success rate (without missed thrust events) of 97 percent and an average optimality error of 3 percent, were able to autonomously correct for a majority of missed thrust events [77]. Yin et al. used deep feedforward neural networks to supply the initial solution for an indirect shooting method, improving the computation efficiency and convergence of the method (compared to a random-guess approach) [78]. Yin et al. considered both fuel-optimal and energy-optimal transfers from Earth to three types of nearby orbits: a coplanar circular orbit, a coplanar elliptical orbit, and an inclined circular orbit [78].

In 2021, Izzo and Öztürk trained deep feedforward neural networks to predict the control policy (thrust magnitude and direction) as well as the propellant mass for a mass-optimal interplanetary transfer from Earth to Venus [79]. Izzo and Öztürk showed deep feedforward neural networks are capable of optimal guidance, approximating the thrust magnitude to within an error of 5 percent and the thrust direction to within an error of 1 degree [79]. The propellent mass was approximated with only a 1 percent error [79]. Xie et al, used deep feedforward neural networks to identify feasible transfers between near-Earth asteroids, improving the convergence rate of the direct transcription method used to over 98 percent, and predict the transfer costs, resulting in a mean relative error of less than 1 percent [80].

In 2022, Viavattene and Ceriotti used deep feedforward neural networks to predict the transfer costs (required changes in velocity) and durations of low-thrust rendezvous trajectories between near-Earth asteroids, showing reduced computational times (up to two orders of magnitudes) with correlation coefficients as high as 0.97 and average errors as low as 10 percent for the transfer costs and 8 percent for the transfer durations [81, 82]. Schiassi et al., following the extreme theory of functional connections method, trained physics-informed neural networks to learn the control policies for three optimal control problems: the planar maximum-radius low-thrust orbit transfer, the planar minimum-time low-thrust orbit transfer, and the planar minimum-time orbit transfer for a solar sail [83]. Schiassi et al. showed the neural networks were able to learn the control policies with satisfactory accuracies (in comparison to an optimal control software) and improved convergence [83].

Though the use of neural networks trained by supervised learning models for applications related to low-thrust trajectory design and optimization is extensive, previous research has largely focused on shallow or deep feedforward architectures with little emphasis placed on recurrent ones, even though recurrent architectures are inherently more suited to time histories due to the presence of internal memory. In fact, to the best of the author's knowledge, recurrent neural networks have rarely been applied and only to attitude control and landing problems (i.e., modeling image-control relationships) or within reinforcement learning models. These applications of recurrent neural networks are as follows. Note, this research is the culmination of the work presented in Reference 84 and Reference 85.

Furfaro et al. implemented an LSTM layer within a deep recurrent neural network architecture in their proposal for a fuel-optimal guidance system for pinpoint planetary landing [86]. Furfaro et al. merged convolutional and recurrent (via an LSTM layer) architectures to predict the fuel-optimal control policy for two types of autonomous lunar landings: a one-dimensional landing and a planar two-dimensional landing [87]. For the first scenario, Furfaro et al. were able to predict the thrust levels with an over 99 percent accuracy [87]. For the second scenario, Furfaro et al. were able to predict the thrust levels with an over 98 percent accuracy and the thrust directions with a root-mean-square error of less than 1 degree [87]. Silvestrini and Lavagna implemented shallow LSTM neural networks in a reinforcement learning algorithm for near-optimal, collision-free reconfiguration and maintenance in distributed formation flying spacecraft, showing comparable or superior performance with respect to the baseline algorithm [88]. Scorsoglio et al. applied deep recurrent (via a GRU layer) convolutional neural networks in a reinforcement learning architecture to the lunar landing problem [89, 90]. Biggs and Fournier proposed recurrent-neural-predictive control for the attitude tracking, slew maneuvers, and detumbling of small satellites [91]. In their investigation on the feasibility of deploying small satellites from low Earth orbit to near-Earth asteroids, Jaworski and Kindracki used deep recurrent neural networks with LSTM layers to predict the future states of the International Space Station [92].

4 Methodology

In this chapter, the data used to tune, train, validate, and test the LSTM neural networks as well as the implementation and design of those LSTM neural networks and their feedforward equivalents will be provided. Note, in this dissertation, a theoretical approach to the low-thrust, orbit-raising problem is presented. In other words, the physical or practical limitations of real hardware were not considered.

4.1 Model for Low-Thrust, Orbit-Raising Problem

Figure 4.1 shows a planar, two-body, low-thrust, orbit-raising transfer in polar coordinates. Polar coordinates, along with equinoctial elements, are a common choice for Keplerian or near-Keplerian problems, such as the planar, two-body, low-thrust, orbit-raising transfer [2]. These elements have state variables that change slower than, for example, Cartesian coordinates [1, 2]. As a result, fewer discretization points are needed to model the problem, making the overall optimization process more efficient and robust [1, 2].

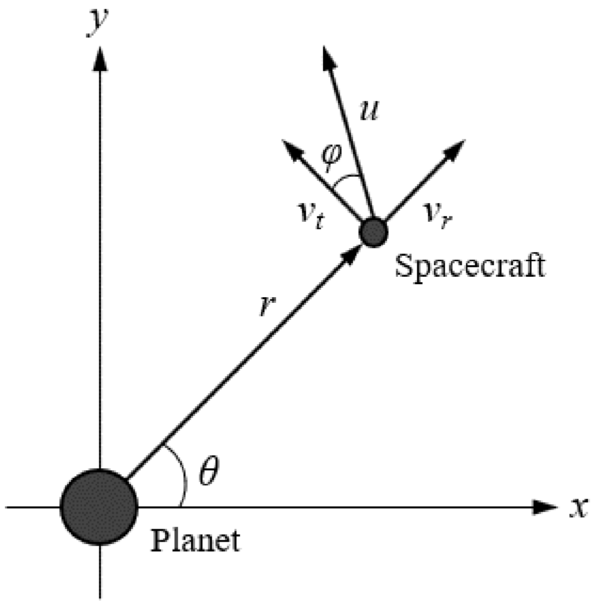


Figure 4.1: Planar, Two-Body, Low-Thrust, Orbit-Raising Transfer [1]

Mathematically, the system dynamics for the planar, two-body, low-thrust, orbit-raising transfer in polar coordinates can be written in canonical units as follows. In these equations, the state variables (r, θ, v_r, and v_t) are the spacecraft radius, phase angle, radial velocity, and tangential velocity, respectively. The control variables (u and φ) are the thrust acceleration and angle, respectively. Note, all state and control variables are scalar values. μ, equal to one, is the gravitational parameter of the central body (Earth).

$$\dot{r} = v_r \qquad \text{Equation 4.1 [1]}$$

$$\dot{\theta} = \frac{v_t}{r} \qquad \text{Equation 4.2 [1]}$$

$$\dot{v}_r = \frac{{v_t}^2}{r} - \frac{\mu}{r^2} + u \sin\varphi \qquad \text{Equation 4.3 [1]}$$

$$\dot{v}_t = -\frac{v_r v_t}{r} + u \cos\varphi \qquad \text{Equation 4.4 [1]}$$

Canonical units, which normalize the problem variables such that the gravitational parameter of the central body is unity, make the overall optimization process more robust by scaling the state variables to be of similar magnitudes [2]. The canonical units for an Earth-orbiting satellite are presented in Table I.

Table I: Canonical Units for Earth-Orbiting Satellite

Variable	Canonical Units	Metric Units
Distance	1 DU	6,378 km
Velocity	1 DU/TU	7.91 km/sec
Time	1 TU	806.80 sec

4.2 Method of Trajectory Optimization

Both direct and indirect methods for trajectory optimization have been applied to a wide range of applications related to low-thrust propulsion. However, as previously mentioned, direct collocations methods are some of the “best known” and “most implemented” methods and, as such, have been applied to the low-thrust, orbit-raising problem as well as lunar and interplanetary transfers [1, 2, 53]. Due to this success, a direct collocation method, specifically the Hermite-Simpson direct collocation method (implemented in MATLAB), was chosen as the method of trajectory optimization in the generation of the training, validation, and test data as well as the method of comparison for neural networks. The following equations are the objective functions for time-optimal and fuel-optimal transfers, respectively.

$$J = t_f - t_0 \qquad \text{Equation 4.5 [1]}$$

$$J = \int_{t_0}^{t_f} u \, \mathrm{d}t \qquad \text{Equation 4.6 [1]}$$

The objective functions shown in Equations 4.5 and 4.6 were minimized using the *fmincon* function in MATLAB, subject to the system dynamic constraints shown in Equations 4.1 through 4.4 as well as the initial and final boundary constraints presented in Table II. Table II also presents the lower and upper bounds on the decision variables. For more information on the datasets, refer to Chapter 4.2.1. For the *fmincon* function in MATLAB, the "interior-point" algorithm and "sqp" algorithm were selected for time-optimal and fuel-optimal transfers, respectively. A constraint violation tolerance of 1e-5 was selected. Other than increasing the maximum number of function evaluations and iterations as well as activating parallel computing, all other options for the *fmincon* function in MATLAB were left as their default values.

Table II: Decision Variables, Boundary Constraints, and Bounds

Decision Variable	Boundary Constraint (Initial)	Boundary Constraint (Final)	Bound (Lower)	Bound (Upper)
r (DU)	1.0470	Varies	0	10
θ (rad)	0	None	0	200
v_r (DU/TU)	0	0	-5	5
v_t (DU/TU)	0.9773	Varies	-5	5
t (TU)	0	None / Varies	0	500
u (DU/TU^2)	None	None	0	0.01
φ (rad)	None	None	-π	π

The optimization software for this research was developed and validated (with respect to the time histories of the control variables) following the results of the practical example for a planar, two-body, low-thrust, orbit-raising transfer (both time-optimal and fuel-optimal) in Topputo and Zhang's *Survey of Direct Transcription for Low-Thrust Space Trajectory Optimization with Applications* [1]. During data generation, the exit messages for the *fmincon* function were checked to verify both the first-order optimality measure and the maximum constraint violation was sufficiently close to zero, satisfying the Karush-Kuhn-Tucker necessary conditions.

4.2.1 Data Generation

The datasets consisted of the optimized trajectories from one low Earth orbit with a radius of 6,678 kilometers to over twenty near-geosynchronous equatorial orbits with radii ranging from 35,000 kilometers to 45,000 kilometers. 15 percent of the training data was used for validation. The test cases for the training datasets consisted of the optimized trajectories from one low Earth orbit with a radius of 6,678 kilometers to one geosynchronous equatorial orbit with a radius of 42,164 kilometers. Table III presents the initial and final orbits for the training datasets and test cases.

Table III: Training Datasets and Test Cases

Training Datasets	
Initial Orbit	**Final Orbit**
6,678 km 1.0470 DU, 0.9773 DU/TU	35,000 km 5.4876 DU, 0.4269 DU/TU
	35,500 km 5.5660 DU, 0.4239 DU/TU
	36,000 km 5.6444 DU, 0.4209 DU/TU
	36,500 km 5.7228 DU, 0.4180 DU/TU
	37,000 km 5.8012 DU, 0.4152 DU/TU
	37,500 km 5.8796 DU, 0.4124 DU/TU
	38,000 km 5.9580 DU, 0.4097 DU/TU
	38,500 km 6.0364 DU, 0.4070 DU/TU
	39,000 km 6.1148 DU, 0.4044 DU/TU
	39,500 km 6.1932 DU, 0.4018 DU/TU
	40,000 km 6.2716 DU, 0.3993 DU/TU
	40,500 km 6.3500 DU, 0.3968 DU/TU
	41,000 km 6.4238 DU, 0.3944 DU/TU
	41,500 km 6.5067 DU, 0.3920 DU/TU
	42,000 km 6.5851 DU, 0.3897 DU/TU
	42,500 km 6.6635 DU, 0.3874 DU/TU
	43,000 km 6.7419 DU, 0.3851 DU/TU
	43,500 km 6.8203 DU, 0.3829 DU/TU
	44,000 km 6.8987 DU, 0.3807 DU/TU
	44,500 km 6.9771 DU, 0.3786 DU/TU
	45,000 km 7.0555 DU, 0.3765 DU/TU
Test Cases	
Initial Orbit	**Final Orbit**
6,678 km 1.0470 DU, 0.9773 DU/TU	42,164 km 6.6108 DU, 0.3889 TU/DU

4.3 Grid Convergence

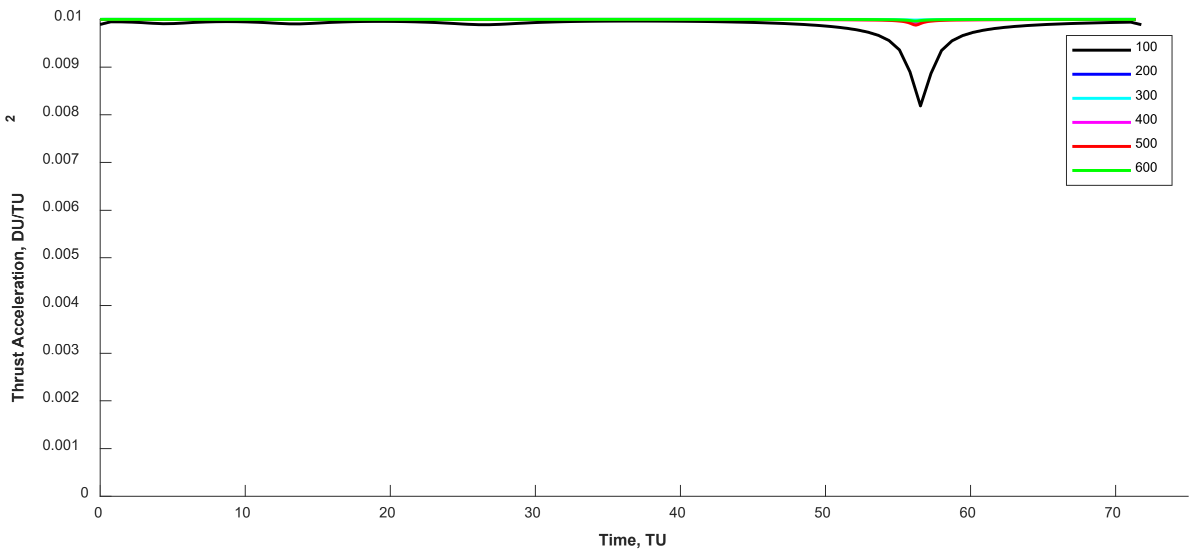
To determine the appropriate number of collocation points for the Hermite-Simpson direct collocation method, grid convergence studies were conducted on the test cases (1.0470 DU to 6.6108 DU). The results from the grid convergence studies will be discussed here.

4.3.1 Time-Optimal Grid Convergence Study

Overall, six simulations were run with the number of points being increased from 100 points to 600 points. For the simulation with 100 points, the initial guess assumed continuous maximum tangential thrust. For a time-optimal transfer, continuous maximum thrust is expected as the spacecraft would continuously thrust throughout the entire transfer, regardless of the spacecraft's location [1]. Also, tangential trust is a common assumption for initial guesses. For all other simulations, the initial guess was the optimized trajectory from the simulation with 100 points expanded to the desired number of points using linear interpolation, which help improve the convergence time of the overall optimization process. Table IV presents the final values for the objective function (i.e., the transfer times) and the convergence times as well as the constraint violations for each simulation. Figure 4.2 shows the optimized thrust accelerations and angles while Figure 4.3 shows the optimized trajectories for each simulation.

Table IV: Final Objective Function Values (Transfer Times) and Convergence Times

Number of Points	Transfer Time (TU)	Convergence Time (hr)	Constraint Violation
100	71.7710	0.03	2.3653e-6
200	71.3835	0.15	4.6349e-14
300	71.3790	0.48	2.1316e-14
400	71.3898	1.14	1.3554e-13
500	71.3931	9.05	8.0635e-11
600	71.3819	2.73	6.8889e-13



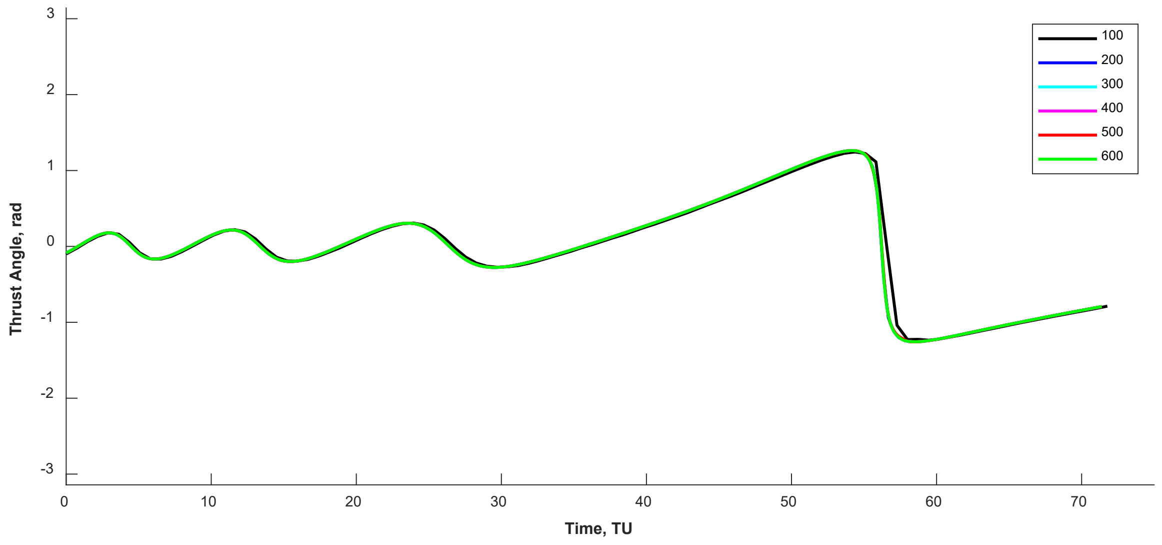


Figure 4.2: Optimized Thrust Accelerations and Angles (Time-Optimal)

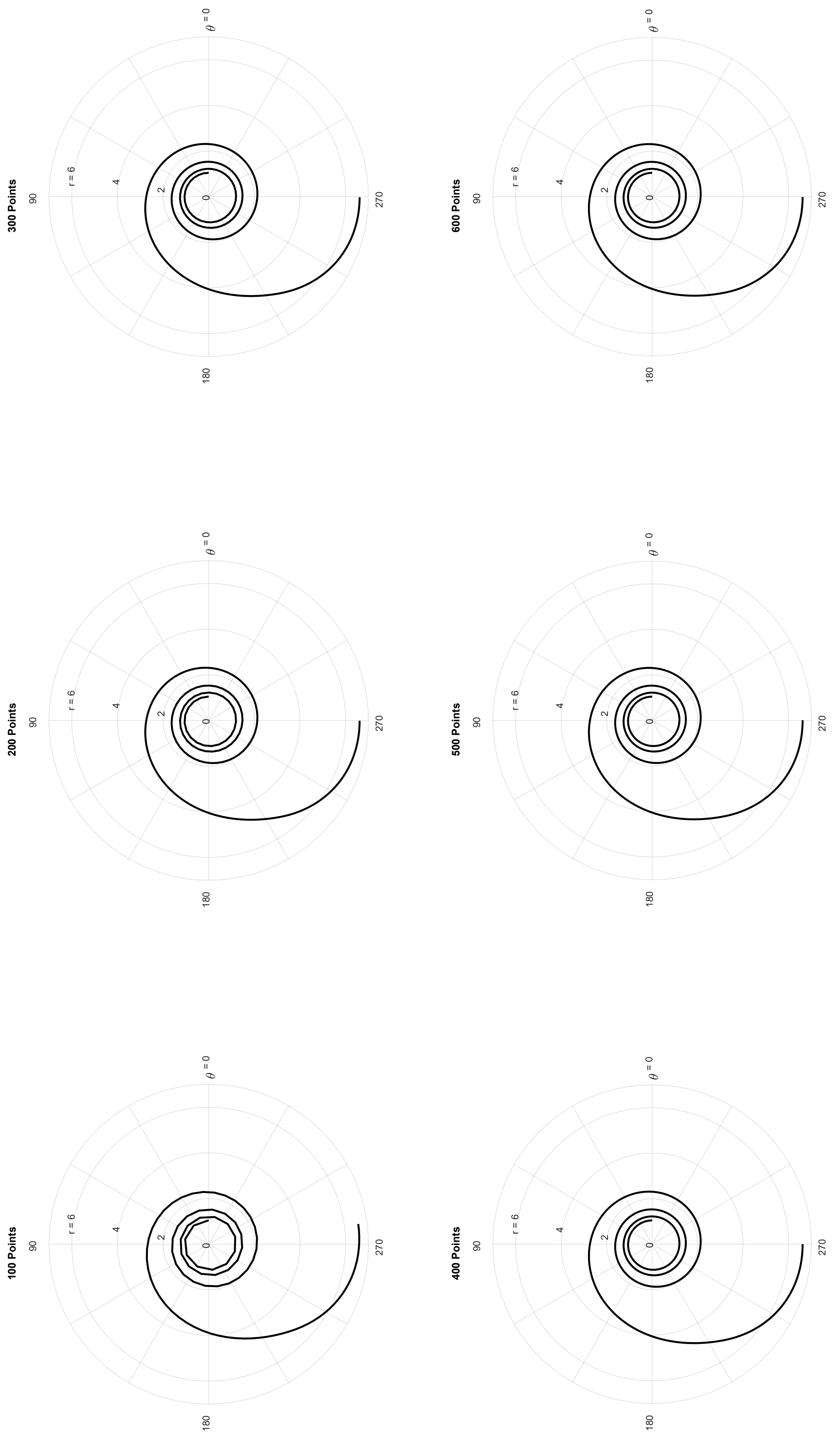


Figure 4.3: Optimized Trajectories (Time-Optimal)

Overall, two unexpected trends were observed from the results of the time-optimal grid convergence study, both of which appeared to be features of the optimization software. First, as seen in Table IV, the transfer time did not consistently decrease as the number of points increased. In fact, almost oscillatory behavior can be seen. This oscillatory behavior appeared to correspond to the relative maximum constraint violations and, thus, was attributed to the chosen step tolerance as well as the chosen constraint violation tolerance. Lower step and constraint violation tolerances were tested. However, as the step size consistently approached the step tolerance much faster than the relative maximum constraint violations approached the constraint violation tolerance, the result was miniscule or no improvement in the transfer time at the cost of longer convergence times or, in some cases, convergence to infeasible points.

Second, as previously mentioned, for a time-optimal transfer, continuous maximum thrust is expected. As seen in the upper plot of Figure 4.2, this trend was observed. However, divot-like structures of various magnitudes were present. Again, this behavior appeared to correspond to the relative maximum constraint violations and, thus, was attributed to the chosen step tolerance and the chosen constraint violation tolerance.

By and large, 300 points was determined to be an appropriate number of collocation points for the Hermite-Simpson direct collocation method on time-optimal transfers. For the test case, 300 points provided suitable local refinement, resulting in more smoother plots than the one generated using 100 and 200 points at more acceptable convergence times than when using 400, 500, and 600 points. In addition, the transfer time for 300 points was the overall minimum transfer time.

4.3.2 Fuel-Optimal Grid Convergence Study

Similar to the time-optimal grid convergence study, six simulations were run with the number of points being increased from 100 points to 600 points. However, in the fuel-optimal grid convergence study, the initial guess for the simulation with 100 points assumed continuous half-maximum tangential thrust to account for the expected discontinuous "bang-bang" control structure as the spacecraft would thrust at locations near perigee and apogee [1]. Again, for all other simulations, the initial guess was the optimized trajectory from the simulation with 100 points expanded to the desired number of points using linear interpolation. Finally, for fuel-optimal transfers, the transfer time was constrained to be twice the optimized transfer time from the time-optimal grid convergence study when using 300 points (71.3790 TU), equal to 142.7580 TU. Table V presents the final values for the objective function (i.e. the thrust accelerations) and the convergence times as well as the constraint violations for each simulation while Figure 4.4 and Figure 4.5 show the optimized thrust accelerations, angles, and trajectories for each simulation.

Table V: Final Objective Function Values (Thrust Acceleration) and Convergence Times

Number of Points	Thrust Acceleration (DU/TU^2)	Convergence Time (hr)	Constraint Violation
100	0.5475	0.05	5.9392e-15
200	0.5451	0.26	7.1100e-11
300	0.5447	0.79	1.4211e-14
400	0.5450	0.16	9.4942e-6
500	0.5447	0.56	2.8810e-6
600	0.5445	0.58	1.0000e-5

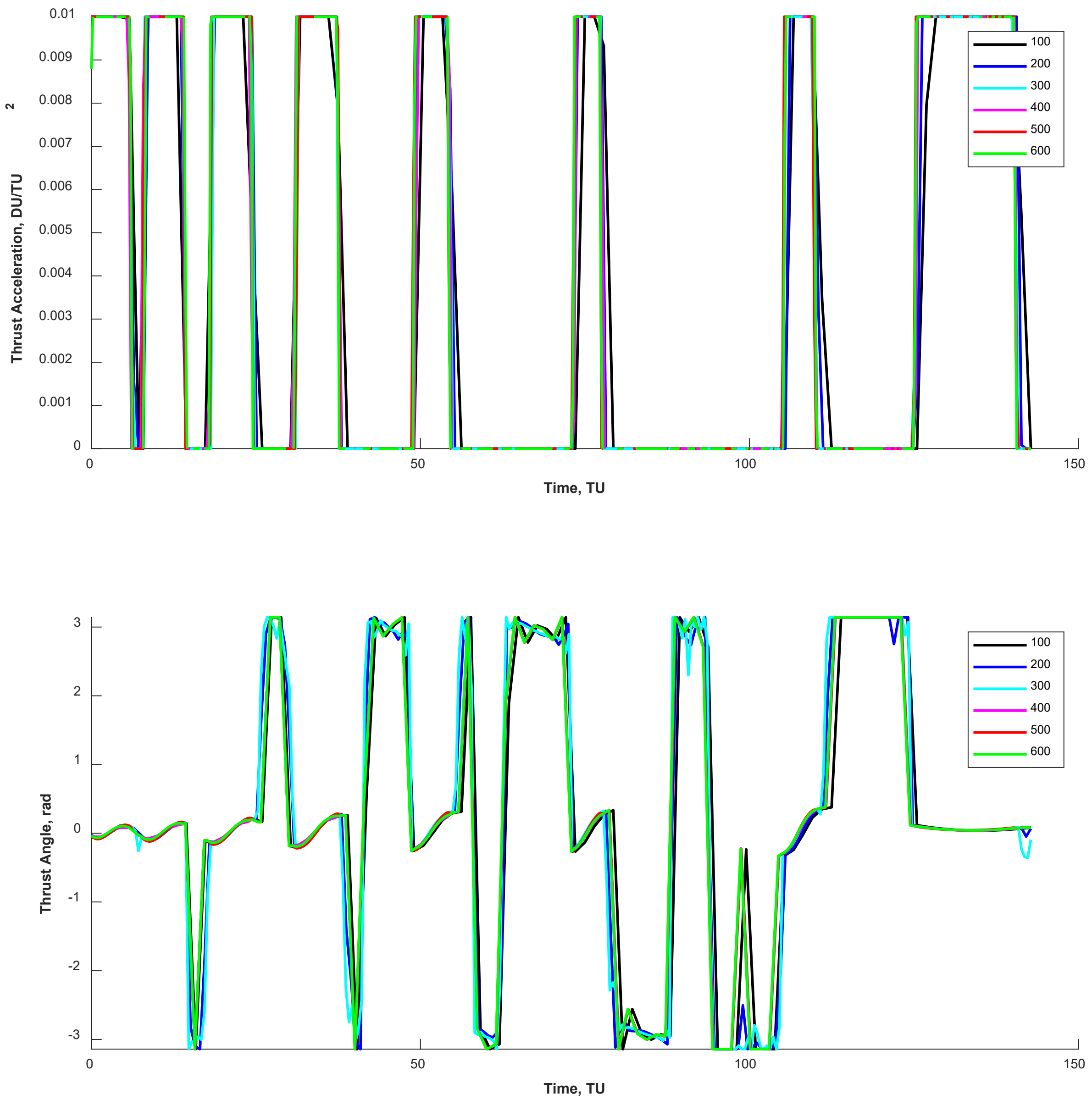


Figure 4.4: Optimized Thrust Accelerations and Angles (Fuel-Optimal)

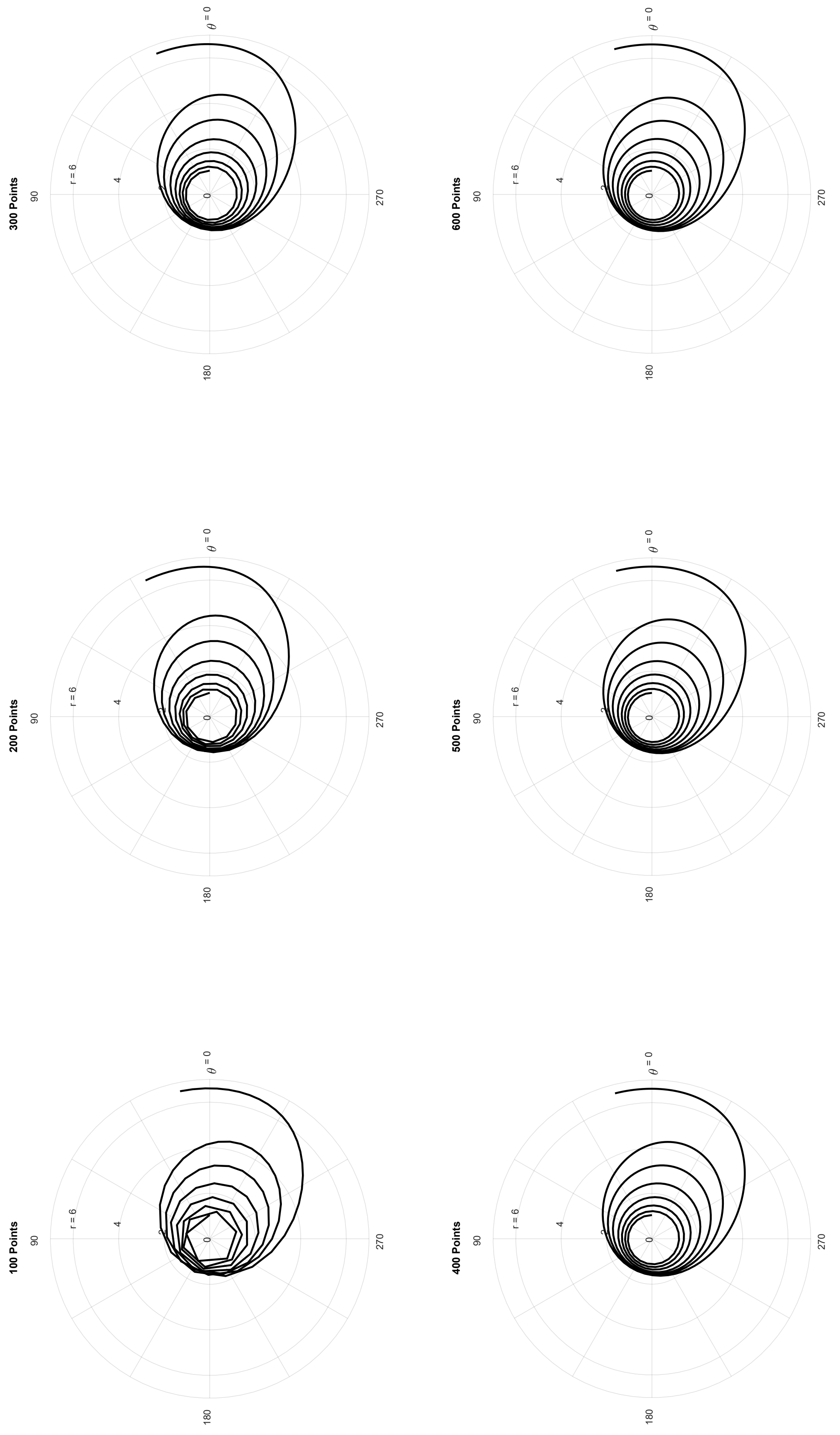


Figure 4.5: Optimized Trajectories (Fuel-Optimal)

First, as seen in Table V, the thrust accelerations did, for the most part, decrease as the number of points increased. Second, as previously mentioned, for a fuel-optimal transfer, a discontinuous "bang-bang" control structure is expected. As seen in the upper plots of Figure 4.4, this trend was observed with definition increasing with the number of points. Again, 300 points was determined to be an appropriate number of collocation points for the Hermite-Simpson direct collocation method on fuel-optimal transfers with constrained transfer times. 300 points provided suitable local refinement at acceptable convergence times. In addition, though the thrust accelerations for 300 points were not the overall minimum thrust accelerations, the thrust accelerations for 300 points differed from the overall minimum by an insignificant amount (i.e., < 1 percent). Note, unnecessary thrust angles were removed from the fuel-optimal datasets before neural network implementation. If the value of the optimized thrust acceleration at any point was less than one percent of the upper bound (0.01 DU/TU^2), the value of the optimized thrust angle was set to zero. Figure 4.6 shows an example of the corrected thrust angles for the test case.

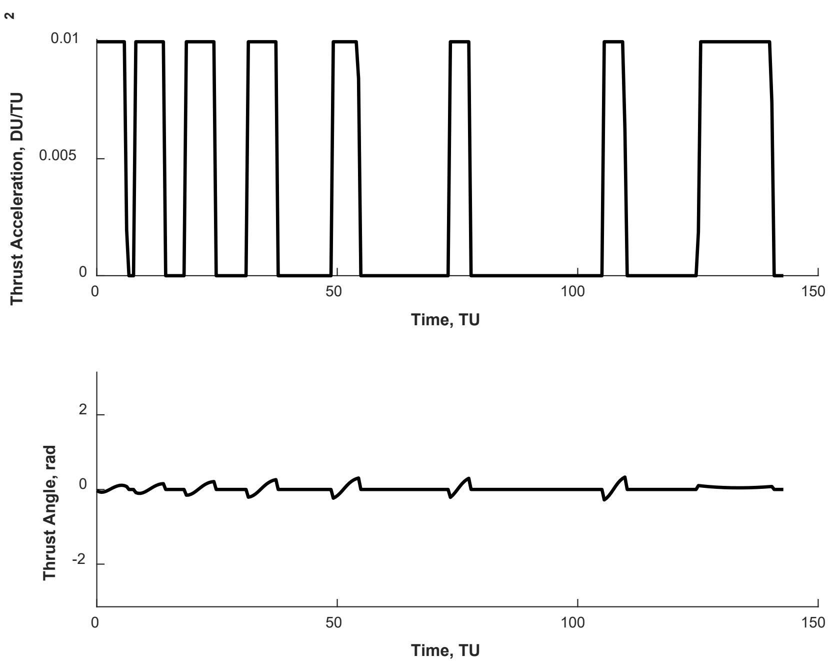


Figure 4.6: Thrust Accelerations and Corrected Angles (Fuel-Optimal)

4.4 Time Bias

To investigate the effect of the time bias in the datasets on the neural networks, datasets with consistent timesteps between collocation points, shown in Table VI, were generated as well. For the time-optimal dataset, the number of points to be used for each trajectory was determined from the optimized transfer time for that trajectory such that the timestep between points was approximately 0.25 TU. For the fuel-optimal datasets with constrained transfer times, the timestep between points was approximately 0.50 TU.

Table VI: Training Datasets and Test Cases with Consistent Timesteps

Training Datasets		Time-Optimal	Fuel-Optimal (Constrained Transfer Time)
Initial Orbit	**Final Orbit**	**Points (0.25 TU)**	**Points (0.50 TU)**
6,678 km	35,000 km	259	259
	35,500 km	261	261
	36,000 km	263	263
	36,500 km	265	265
	37,000 km	267	267
	37,500 km	269	269
	38,000 km	270	270
	38,500 km	272	272
	39,000 km	274	274
	39,500 km	276	276
	40,000 km	278	278
	40,500 km	280	280
	41,000 km	281	281
	41,500 km	283	283
	42,000 km	290	290
	42,500 km	287	287
	43,000 km	288	288
	43,500 km	290	290
	44,000 km	292	292
	44,500 km	294	294
	45,000 km	295	295
Test Cases		**Time-Optimal**	**Fuel-Optimal (Constrained Transfer Time)**
Initial Orbit	**Final Orbit**	**Points (0.25 TU)**	**Points (0.50 TU)**
6,678 km	42,164 km	286	286

4.5 Neural Network Implementation

The LSTM neural networks were implemented in Python 3.8 [93] using Keras [94] and TensorFlow [95]. Figure 4.7 shows a diagram of the general architecture for the LSTM neural networks. The LSTM neural networks were tuned and trained over a maximum of 60 epochs with a batch size of one using the *MeanSquaredError* regression loss and the *Adam* optimizer. The *Adam* optimizer is a first-order, gradient-based, optimization algorithm able to handle noisy and/or sparse gradients as well as non-stationary objectives [96]. For the "LSTM" layers, the activation functions and recurrent activation functions were kept as their default values (the hyperbolic tangent function and the sigmoid function, respectively).

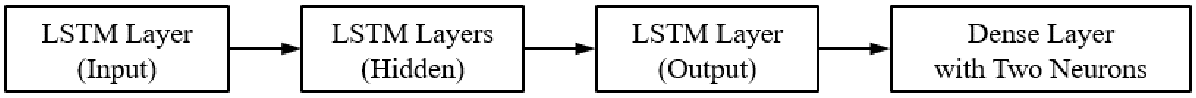


Figure 4.7: Diagram of LSTM Neural Network Architecture

The following parameters and hyperparameters of the LSTM neural networks were determined using the *Hyperband* Tuner in Keras, which is a variant of the *RandomSearch* Tuner that uses early stopping to speed-up the overall tuning process. The *Hyperband* Tuner monitored the validation loss with a patience of 15 epochs. Best weights were restored.

1. Number of Memory Units in Input "LSTM" Layer (from 32 to 512)
2. Number of Hidden "LSTM" Layers (from 1 to 3)
3. Number of Memory Units in Hidden "LSTM" Layers (from 32 to 512)
4. Number of Memory Units in Output "LSTM" Layer (from 32 to 512)
5. Activation Function for "Dense" Layer with Two Neurons ("relu", "sigmoid", or "tanh")
6. Learning Rate for Adam Optimizer (0.01, 0.001, or 0.0001)

The feedforward equivalents of the LSTM neural networks were implemented in a similar fashion with a similar general architecture using "Dense" layers instead of "LSTM" layers. Like the LSTM neural networks, the feedforward neural networks were tuned and trained over a maximum of 60 epochs with a batch size of one using the *MeanSquaredError* regression loss and the *Adam* optimizer. The following parameters and hyperparameters of the feedforward neural networks were determined using, as with the LSTM neural networks, the *Hyperband* Tuner in Keras. Again, the *Hyperband* Tuner monitored the validation loss with a patience of 15 epochs and best weights were restored.

1. Number of Memory Units in Input "Dense" Layer (from 32 to 512)
2. Activation Function for Input "Dense" Layer ("relu", "sigmoid", or "tanh")
3. Number of Hidden "Dense" Layers (from 1 to 3)
4. Number of Memory Units in Hidden "Dense" Layers (from 32 to 512)
5. Activation Functions for Hidden "Dense" Layers ("relu", "sigmoid", or "tanh")
6. Number of Memory Units in Output "Dense" Layer (from 32 to 512)
7. Activation Function for Output "Dense" Layer ("relu", "sigmoid", or "tanh")
8. Activation Function for "Dense" Layer with Two Neurons ("relu", "sigmoid", or "tanh")
9. Learning Rate for Adam Optimizer (0.01, 0.001, or 0.0001)

All data was preprocessed with feature-wise normalization (using the *MinMaxScalar* class in the scikit-learn library) and the sliding window method (using the *shift* function in the pandas library). Feature-wise normalization normalizes the data to have a mean of zero and a standard deviation of one while the sliding window method converts a time series into a set of input-output pairs [37, 97].

5 Results

In this chapter, the results for the LSTM neural networks, including a comparison to their feedforward equivalents, will be presented. A total of four problems were considered: the time-optimal open-loop case, the time-optimal closed-loop case, the fuel-optimal open-loop case, and the fuel-optimal closed-loop case. Using the *Hyperband* Tuner in Keras, the LSTM neural networks were tuned and trained on the time-optimal and fuel-optimal datasets. Two cases (open-loop and closed-loop) were considered for each of the time-optimal and fuel-optimal datasets. When considering the open-loop case, the neural networks were tuned and trained on the state variables at the current timesteps to predict the control (thrust acceleration and angle) at those timesteps. When considering the closed-loop case, the neural networks were tuned and trained on the state variables at the previous timesteps to predict the control at the next, or current, timesteps. During the inference process for the closed-loop case, the predicted control for the state variables at the previous timesteps were fed into a fourth-order Runge-Kutta method (initialized with maximum tangential thrust) to calculate the state variables at the next timesteps. Those calculated state variables were then fed back into the neural networks, repeating the process.

5.1 Time-Optimal

The results for the LSTM neural networks, including a comparison to their feedforward equivalents, when tuned and trained on the time-optimal datasets will be discussed here. Two datasets (inconsistent and consistent timesteps) as well as two sets of inputs (states and states with the desired changes for those states) were considered for both the open-loop case and the closed-loop case. In accordance with the tuning process, the neural networks were trained over a maximum of 60 epochs with a batch size of one using the *MeanSquaredError* regression loss and the *Adam* optimizer. Early stopping was used to speed-up the overall training process by monitoring the validation loss with patience of 15 epochs. Best weights were restored. Finally, each neural network was re-trained 20 times. The results presented here are representative of the averages for those 20 trials.

5.1.1 Time-Optimal Open-Loop

Table VII through Table X present the tuned parameters and hyperparameters of the LSTM neural networks for the time-optimal open-loop case. The LSTM neural networks presented in Table VII and Table VIII were tuned and trained on the time-optimal dataset with inconsistent timesteps using the current states and the current states with the desired changes for those states as inputs, respectively. The LSTM neural networks presented in Table IX and Table X were tuned and trained on the time-optimal dataset with consistent timesteps using, again, the current states and current states with the desired changes for those states as inputs, respectively. Overall, tuning on the time-optimal dataset with consistent timesteps resulted in LSTM neural networks with fewer layers compared to tuning on the time-optimal dataset with inconsistent timesteps.

Table VII: Time-Optimal Open-Loop LSTM Tuning Results (Current State – Control Pairs with Inconsistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
LSTM Layer (Input)	192	Default
LSTM Layer (Hidden 1)	352	
LSTM Layer (Hidden 2)	288	
LSTM Layer (Output)	96	
Dense Layer	2	“tanh”
Learning Rate = 0.001		

Table VIII: Time-Optimal Open-Loop LSTM Tuning Results (Current State and Changes – Control Pairs with Inconsistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
LSTM Layer (Input)	512	Default
LSTM Layer (Hidden 1)	352	
LSTM Layer (Hidden 2)	320	
LSTM Layer (Hidden 3)	192	
LSTM Layer (Output)	256	
Dense Layer	2	“tanh”
Learning Rate = 0.001		

Table IX: Time-Optimal Open-Loop LSTM Tuning Results
(Current State – Control Pairs with Consistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
LSTM Layer (Input)	320	Default
LSTM Layer (Hidden 1)	320	
LSTM Layer (Output)	448	
Dense Layer	2	"sigmoid"
Learning Rate = 0.0001		

Table X: Time-Optimal Open-Loop LSTM Tuning Results
(Current State and Changes – Control Pairs with Consistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
LSTM Layer (Input)	64	Default
LSTM Layer (Hidden 1)	96	
LSTM Layer (Output)	160	
Dense Layer	2	"sigmoid"
Learning Rate = 0.001		

Figure 5.1 and Figure 5.2 compare the current control predictions (thrust accelerations and angles, respectively) from the tuned LSTM neural networks to the optimized values from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. Table XI presents an overall summary of the tuning and training results. When considering the thrust acceleration predictions, the LSTM neural networks performed well with the more severe deviations from the optimized values occurring near the divot-like structures discussed previously. As the magnitude of those divot-like structures differed significantly between trajectories in the training datasets, this behavior was attributed to generalization error. Better performance in terms of average RMSE was observed from the LSTM neural networks tuned and trained on the time-optimal dataset with consistent timesteps. When considering the thrust angle predictions, the LSTM neural networks performed excellent. Better performance in terms of average RMSE was observed from the LSTM neural networks tuned and trained with the current states and the desired changes for those states as inputs.

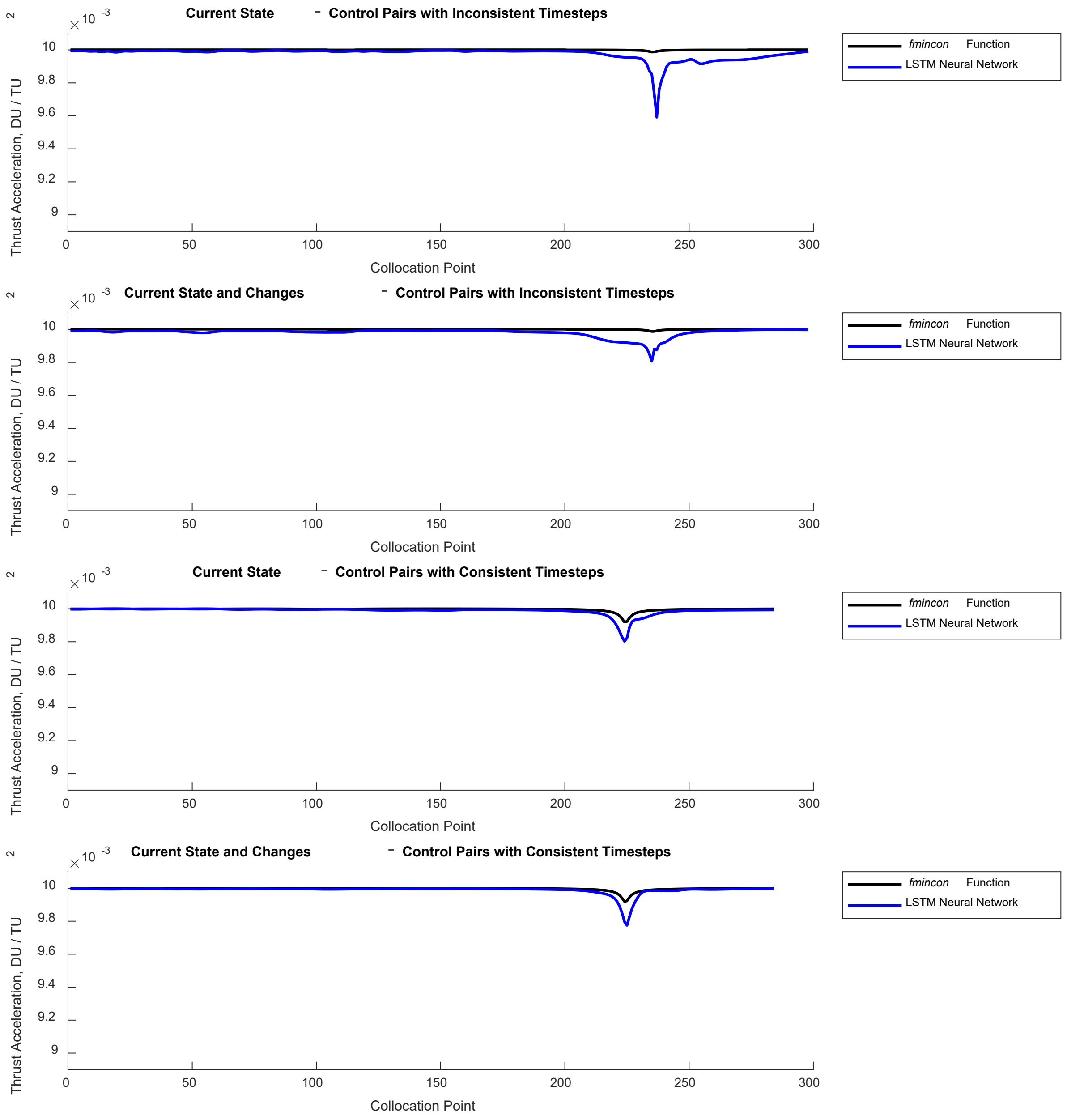


Figure 5.1: Time-Optimal Open-Loop LSTM Test Case Results (Thrust Accelerations)

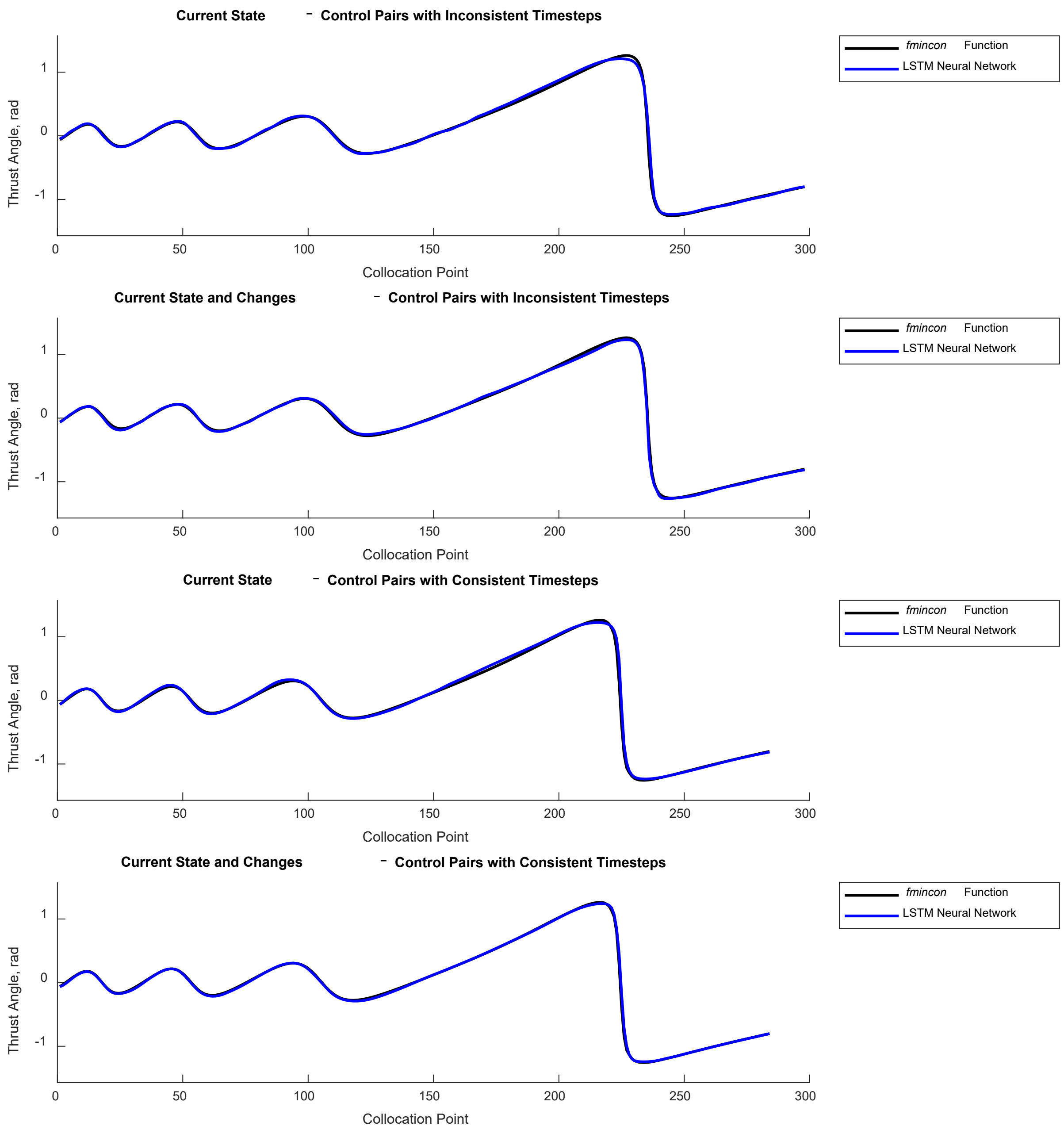


Figure 5.2: Time-Optimal Open-Loop LSTM Test Case Results (Thrust Angles)

Table XI: Summary of Time-Optimal Open-Loop LSTM Tuning and Training Results

LSTM Neural Network	Tuning Time	Test Case RMSE	Training Time
Current State – Control Pairs with Inconsistent Timesteps	14.04 hr	4.4322e-5 DU/TU^2 0.0329 rad	0.29 hr
Current State and Changes – Control Pairs with Inconsistent Timesteps	15.54 hr	3.2387e-5 DU/TU^2 0.0228 rad	0.59 hr
Current State – Control Pairs with Consistent Timesteps	15.38 hr	1.8116e-5 DU/TU^2 0.0353 rad	0.33 hr
Current State and Changes – Control Pairs with Consistent Timesteps	12.13 hr	1.8251e-5 DU/TU^2 0.0240 rad	0.07 hr

5.1.2 Time-Optimal Closed-Loop

Table XII through Table XV present the tuned parameters and hyperparameters of the LSTM neural networks for the time-optimal closed-loop case. The LSTM neural networks presented in Table XII and Table XIII were tuned and trained on the time-optimal dataset with inconsistent timesteps using the previous states and the previous states with the desired changes for those states as inputs, respectively. The LSTM neural networks presented in Table XIV and Table XV were tuned and trained on the time-optimal dataset with consistent timesteps using, again, the previous states and previous states with the desired changes for those states as inputs, respectively. As in the time-optimal open-loop case, tuning on the time-optimal dataset with consistent timesteps resulted in LSTM neural networks with fewer layers compared to tuning on the time-optimal dataset with inconsistent timesteps. However, compared to the time-optimal open-loop case, the relative size of the tuned architectures for the time-optimal closed-loop case increased.

Table XII: Time-Optimal Closed-Loop LSTM Tuning Results (Previous State – Current Control Pairs with Inconsistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
LSTM Layer (Input)	384	Default
LSTM Layer (Hidden 1)	384	
LSTM Layer (Hidden 2)	480	
LSTM Layer (Hidden 3)	192	
LSTM Layer (Output)	512	
Dense Layer	2	"sigmoid"
Learning Rate = 0.0001		

Table XIII: Time-Optimal Closed-Loop LSTM Tuning Results (Previous State and Changes – Current Control Pairs with Inconsistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
LSTM Layer (Input)	32	Default
LSTM Layer (Hidden 1)	352	
LSTM Layer (Hidden 2)	416	
LSTM Layer (Hidden 3)	192	
LSTM Layer (Output)	288	
Dense Layer	2	"tanh"
Learning Rate = 0.001		

Table XIV: Time-Optimal Closed-Loop LSTM Tuning Results (Previous State – Current Control Pairs with Consistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
LSTM Layer (Input)	192	Default
LSTM Layer (Hidden 1)	224	
LSTM Layer (Hidden 2)	416	
LSTM Layer (Output)	96	
Dense Layer	2	"sigmoid"
Learning Rate = 0.001		

Table XV: Time-Optimal Closed-Loop LSTM Tuning Results (Previous State and Changes – Current Control Pairs with Consistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
LSTM Layer (Input)	416	Default
LSTM Layer (Hidden 1)	416	
LSTM Layer (Hidden 2)	64	
LSTM Layer (Output)	512	
Dense Layer	2	"relu"
Learning Rate = 0.0001		

Figure 5.3 and Figure 5.4 compare the current control predictions (thrust accelerations and angles, respectively) from the tuned LSTM neural networks to the optimized values from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. Table XVI presents a summary of the tuning and training results. Overall, the LSTM neural networks for the time-optimal closed-loop case performed worse in terms of average RMSE than the LSTM neural networks for the time-optimal open-loop case. As in the time-optimal open-loop case, when considering the thrust acceleration predictions, the more severe deviations from the optimized values typically occurred near the divot-like structures. Again, this behavior was attributed to generalization error. Also, like the time-optimal open-loop case, better performance

in terms of average RMSE was observed from the LSTM neural networks tuned and trained on the time-optimal dataset with consistent timesteps as well as with the previous states and the desired changes for those states as inputs. Like the time-optimal open-loop case, when considering the thrust angle predictions, better performance in terms of average RMSE was observed from the LSTM neural networks tuned and trained on the time-optimal dataset with consistent timesteps as well as with the previous states and the desired changes for those states as inputs.

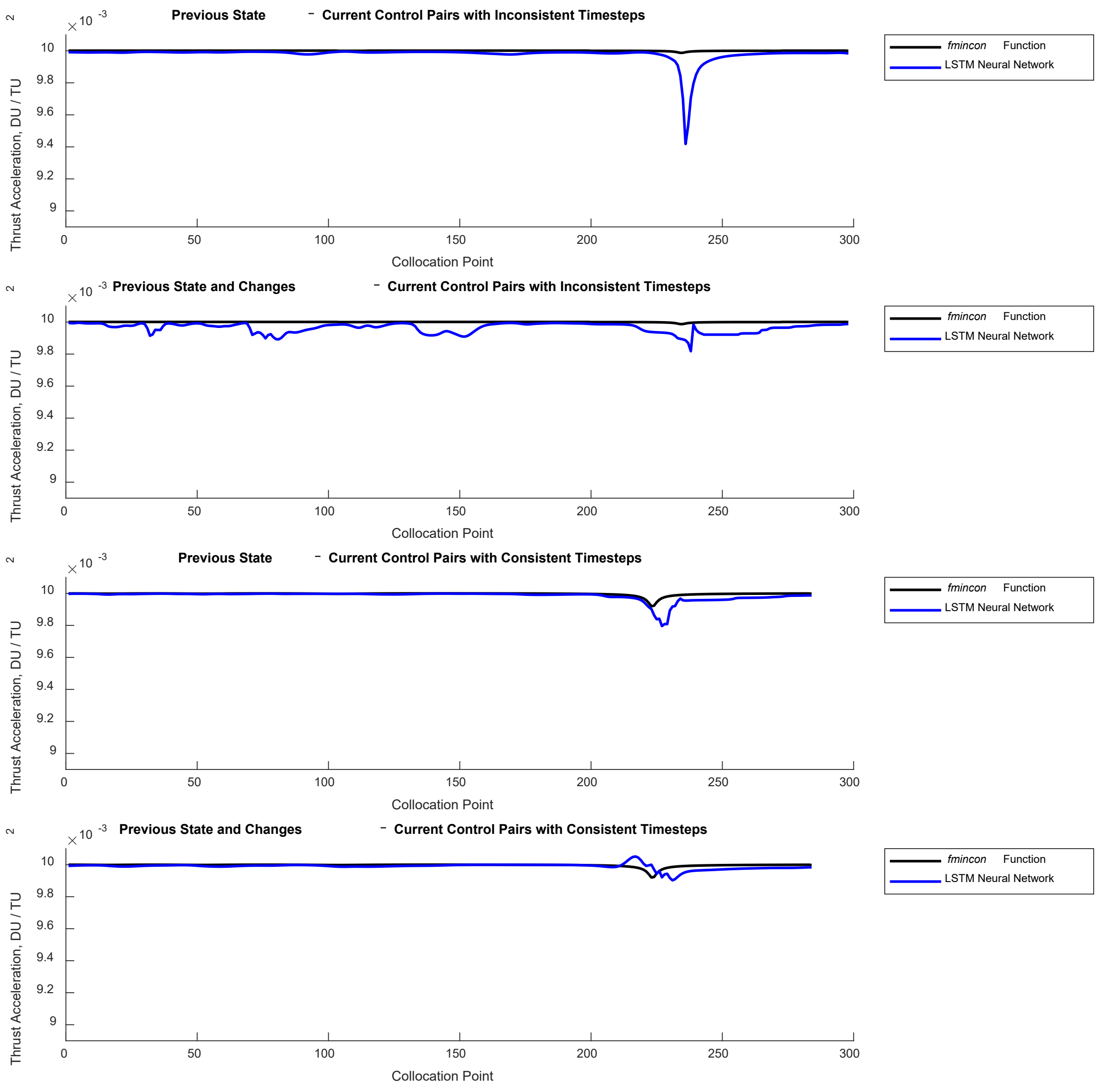


Figure 5.3: Time-Optimal Closed-Loop LSTM Test Case Results (Thrust Accelerations)

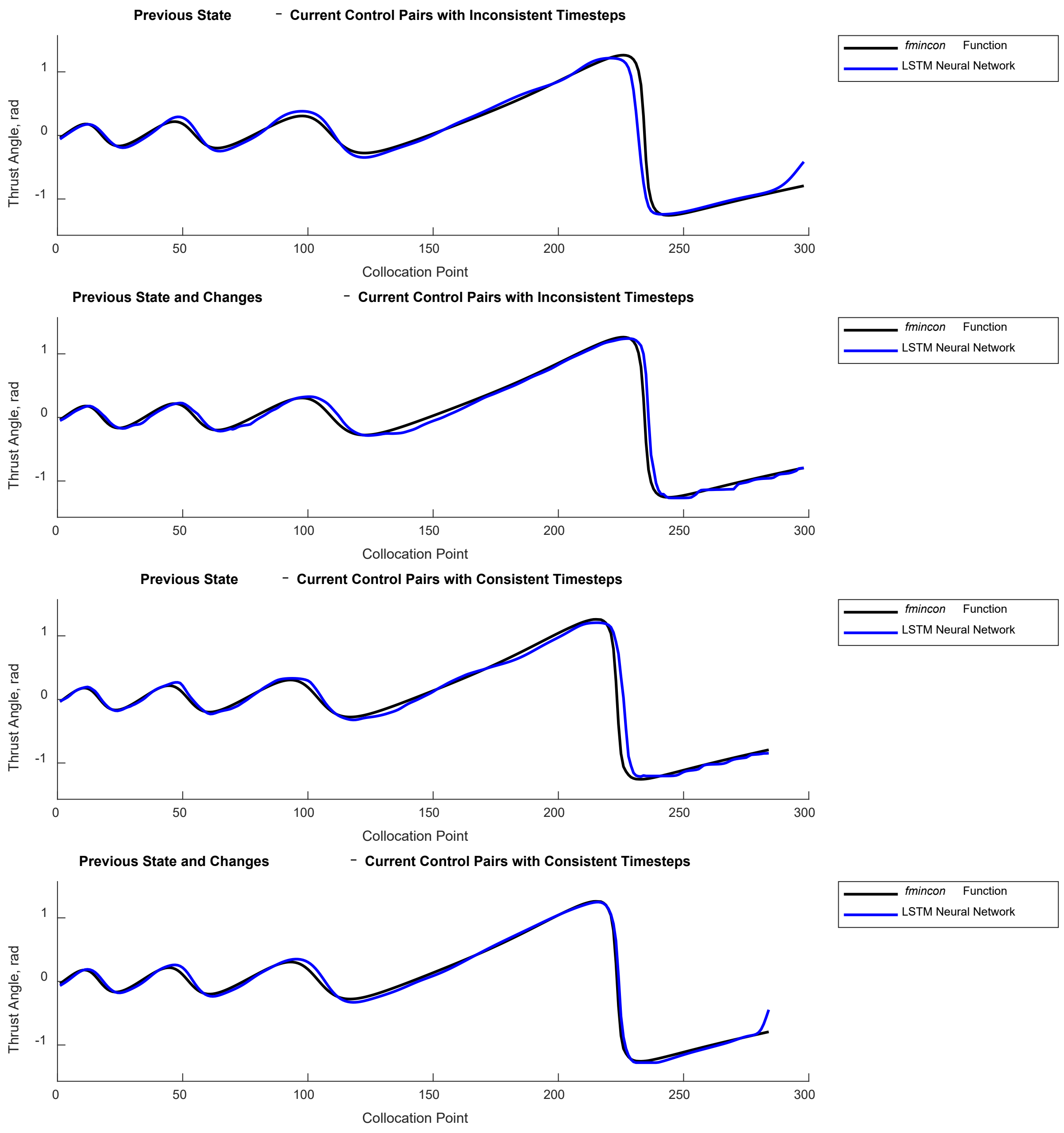


Figure 5.4: Time-Optimal Closed-Loop LSTM Test Case Results (Thrust Angles)

Table XVI: Summary of Time-Optimal Closed-Loop LSTM Tuning and Training Results

LSTM Neural Network	Tuning Time	Test Case RMSE	Training Time
Previous State – Current Control Pairs with Inconsistent Timesteps	14.54 hr	6.5119e-5 DU/TU^2 0.1602 rad	0.71 hr
Previous State and Changes – Current Control Pairs with Inconsistent Timesteps	16.08 hr	4.7014e-5 DU/TU^2 0.1093 rad	0.47 hr
Previous State – Current Control Pairs with Consistent Timesteps	13.57 hr	2.7477e-5 DU/TU^2 0.1352 rad	0.34 hr
Previous State and Changes – Current Control Pairs with Consistent Timesteps	15.41 hr	1.9001e-5 DU/TU^2 0.0663 rad	0.43 hr

5.1.3 Initial Comparison to Feedforward Architectures

Table XVII through Table XX present the tuned parameters and hyperparameters of the feedforward neural networks for the time-optimal open-loop case. The feedforward neural networks presented in Table XVII and Table XVIII were tuned and trained on the time-optimal dataset with inconsistent timesteps using the current states and the current states with the desired changes for those states as inputs, respectively. The feedforward neural networks presented in Table XIX and Table XX were tuned and trained on the time-optimal dataset with consistent timesteps using, again, the current states and current states with the desired changes for those states as inputs, respectively. Like the LSTM neural networks, tuning on the time-optimal dataset with consistent timesteps resulted in feedforward neural networks with slightly fewer layers and neurons compared to tuning on the time-optimal dataset with inconsistent timesteps.

Table XVII: Time-Optimal Open-Loop Feedforward Tuning Results (Current State – Control Pairs with Inconsistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
Dense Layer (Input)	288	"relu"
Dense Layer (Hidden 1)	320	"relu"
Dense Layer (Hidden 2)	416	"relu"
Dense Layer (Output)	384	"tanh"
Dense Layer	2	"tanh"
Learning Rate = 0.0001		

Table XVIII: Time-Optimal Open-Loop Feedforward Tuning Results (Current State and Changes – Control Pairs with Inconsistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
Dense Layer (Input)	128	"tanh"
Dense Layer (Hidden 1)	512	"tanh"
Dense Layer (Hidden 2)	384	"relu"
Dense Layer (Output)	352	"sigmoid"
Dense Layer	2	"sigmoid"
Learning Rate = 0.0001		

Table XIX: Time-Optimal Open-Loop Feedforward Tuning Results (Current State – Control Pairs with Consistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
Dense Layer (Input)	320	"relu"
Dense Layer (Hidden 1)	352	"tanh"
Dense Layer (Output)	384	"relu"
Dense Layer	2	"tanh"
Learning Rate = 0.0001		

Table XX: Time-Optimal Open-Loop Feedforward Tuning Results (Current State and Changes – Control Pairs with Consistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
Dense Layer (Input)	224	"relu"
Dense Layer (Hidden 1)	128	"sigmoid"
Dense Layer (Hidden 2)	320	"relu"
Dense Layer (Output)	96	"tanh"
Dense Layer	2	"sigmoid"
Learning Rate = 0.0001		

Figure 5.5 and Figure 5.6 compare the current control predictions (thrust accelerations and angles, respectively) from the tuned feedforward neural networks to the optimized values from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. Table XXI presents a summary of the tuning and training results. When considering the thrust acceleration predictions, the feedforward neural networks, like the LSTM neural networks, performed well with the more severe deviations from the optimized values occurring near the divot-like structures. Again, this behavior was attributed to generalization error. Also, like the LSTM neural networks, better performance in terms of average RMSE was observed from the feedforward neural networks tuned and trained on the time-optimal dataset with consistent

timesteps. When considering the thrust angle predictions, the feedforward neural networks, like the LSTM neural networks, performed excellent. Also, like the LSTM neural networks, better performance in terms of average RMSE was observed from the feedforward neural networks tuned and trained with the current states and the desired changes for those states as inputs.

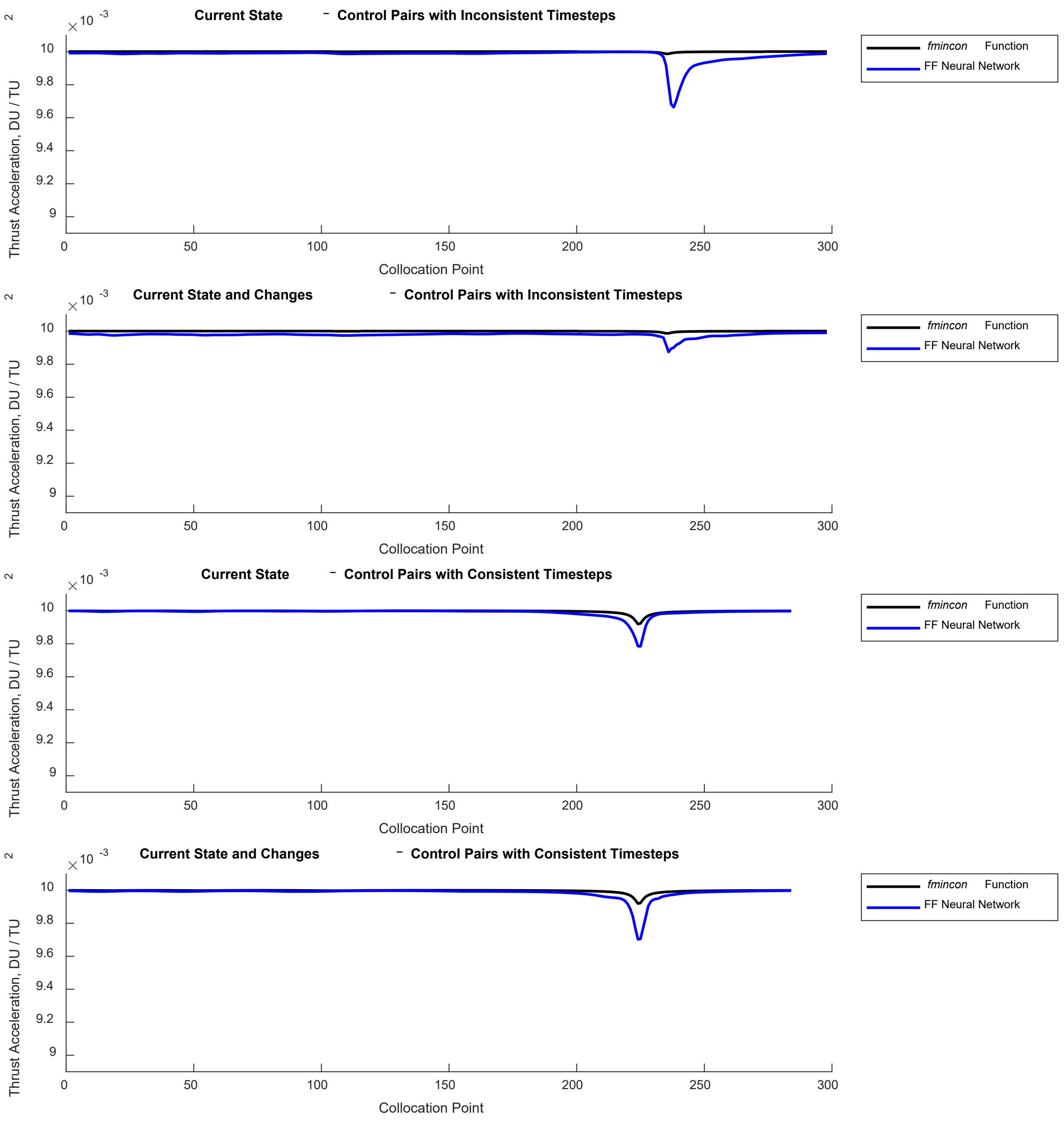


Figure 5.5: Time-Optimal Open-Loop Feedforward Test Case Results (Thrust Accelerations)

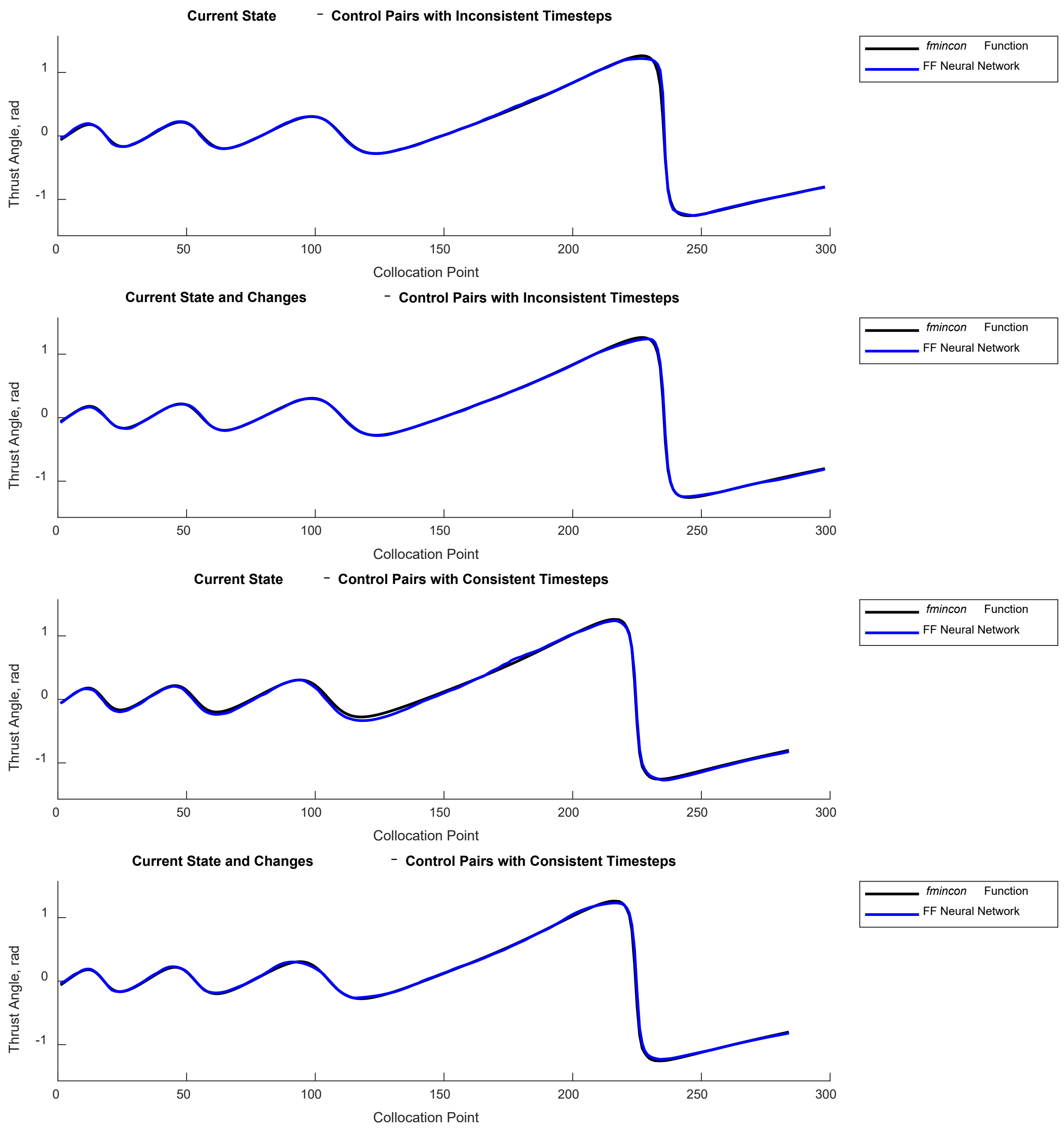


Figure 5.6: Time-Optimal Open-Loop Feedforward Test Case Results (Thrust Angles)

Table XXI: Summary of Time-Optimal Open-Loop Feedforward Tuning and Training Results

Feedforward Neural Network	Tuning Time	Test Case RMSE	Training Time
Current State – Control Pairs with Inconsistent Timesteps	1.68 hr	4.3869e-5 DU/TU^2 0.0295 rad	0.06 hr
Current State and Changes – Control Pairs with Inconsistent Timesteps	1.68 hr	2.4688e-5 DU/TU^2 0.0158 rad	0.06 hr
Current State – Control Pairs with Consistent Timesteps	1.54 hr	1.8545e-5 DU/TU^2 0.0313 rad	0.05 hr
Current State and Changes – Control Pairs with Consistent Timesteps	1.45 hr	2.9324-5 DU/TU^2 0.0300 rad	0.04 hr

Table XXII through Table XXV present the tuned parameters and hyperparameters of the feedforward neural networks for the time-optimal closed-loop case. The feedforward neural networks presented in Table XXII and Table XXIII were tuned and trained on the time-optimal dataset with inconsistent timesteps using the previous states and the previous states with the desired changes for those states as inputs, respectively. The feedforward neural networks presented in Table XXIV and Table XXV were tuned and trained on the time-optimal dataset with consistent timesteps using, again, the previous states and previous states with the desired changes for those states as inputs, respectively. No significant trends were observed from the tuned architectures.

Table XXII: Time-Optimal Closed-Loop Feedforward Tuning Results (Previous State – Current Control Pairs with Inconsistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
Dense Layer (Input)	96	“relu”
Dense Layer (Hidden 1)	448	“sigmoid”
Dense Layer (Output)	96	“relu”
Dense Layer	2	“sigmoid”
Learning Rate = 0.001		

Table XXIII: Time-Optimal Closed-Loop Feedforward Tuning Results (Previous State and Changes – Current Control Pairs with Inconsistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
Dense Layer (Input)	352	“relu”
Dense Layer (Hidden 1)	128	“tanh”
Dense Layer (Hidden 2)	480	“sigmoid”
Dense Layer (Output)	320	“tanh”
Dense Layer	2	“sigmoid”
Learning Rate = 0.0001		

Table XXIV: Time-Optimal Closed-Loop Feedforward Tuning Results (Previous State – Current Control Pairs with Consistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
Dense Layer (Input)	480	“tanh”
Dense Layer (Hidden 1)	320	“relu”
Dense Layer (Hidden 2)	320	“relu”
Dense Layer (Output)	192	“sigmoid”
Dense Layer	2	“sigmoid”
Learning Rate = 0.0001		

Table XXV: Time-Optimal Closed-Loop Feedforward Tuning Results (Previous State and Changes – Current Control Pairs with Consistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
Dense Layer (Input)	352	“relu”
Dense Layer (Hidden 1)	32	“sigmoid”
Dense Layer (Output)	96	“tanh”
Dense Layer	2	“sigmoid”
Learning Rate = 0.001		

Figure 5.7 and Figure 5.8 compare the current control predictions (thrust accelerations and angles, respectively) from the tuned feedforward neural networks to the optimized values from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. Table XXVI presents a summary of the tuning and training results. Overall, like the LSTM neural networks, the feedforward neural networks for the time-optimal closed-loop case performed worse in terms of average RMSE than the feedforward neural networks for the time-optimal open-loop case. As in the time-optimal open-loop case, when considering the thrust acceleration predictions, the more severe deviations from the optimized values typically occurred near the divot-like structures. Again, this behavior was attributed to generalization error. Also, like the LSTM neural networks, better performance in terms of average RMSE was observed from the feedforward neural networks tuned and trained on the time-optimal dataset with consistent timesteps as well as with the previous states and the desired changes for those states as inputs. As in the time-optimal open-loop case, when considering the thrust angle predictions, better

performance in terms of average RMSE was observed from the feedforward neural networks tuned and trained with the previous states and the desired changes for those states as inputs.

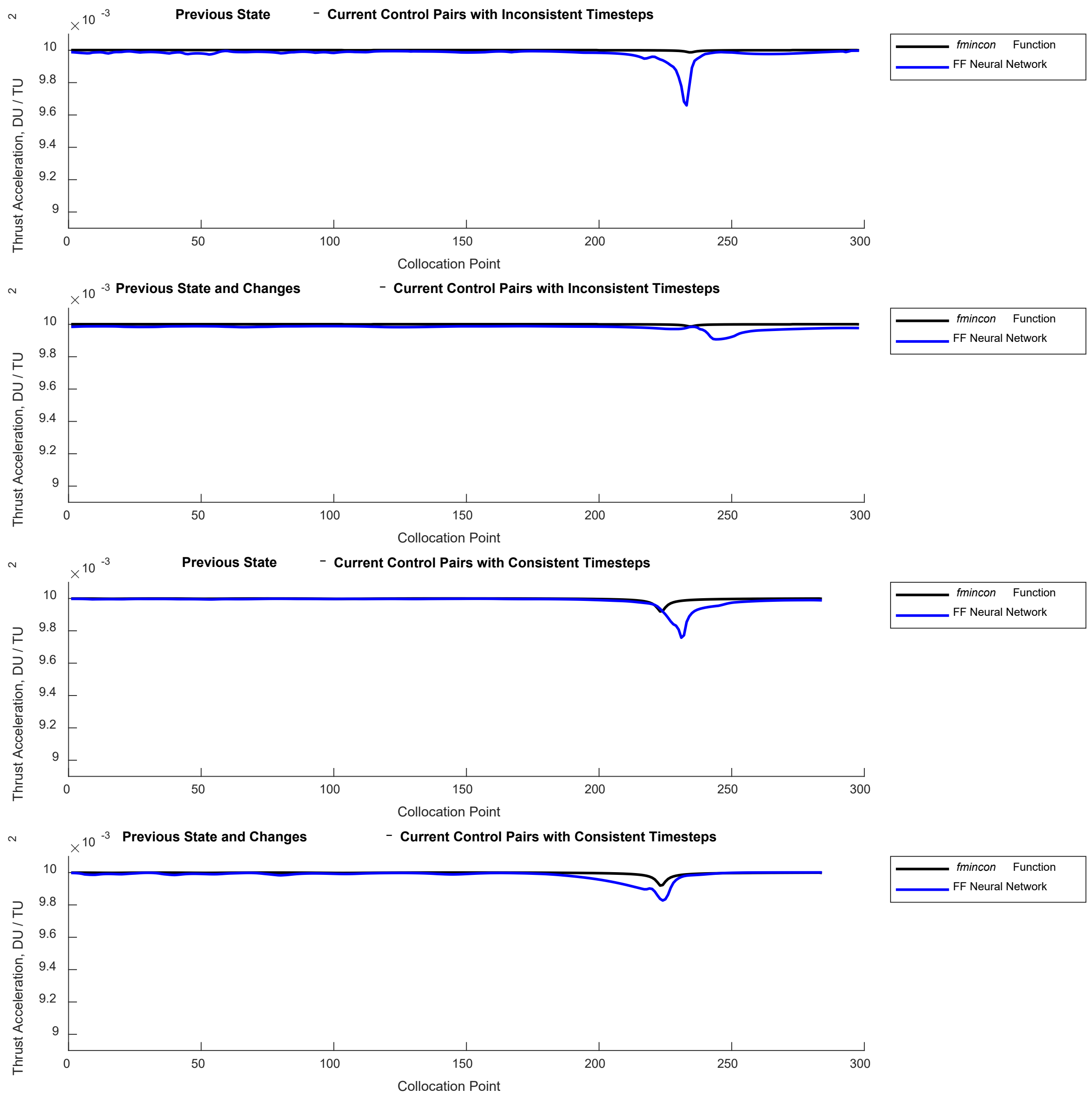


Figure 5.7: Time-Optimal Closed-Loop Feedforward Test Case Results (Thrust Accelerations)

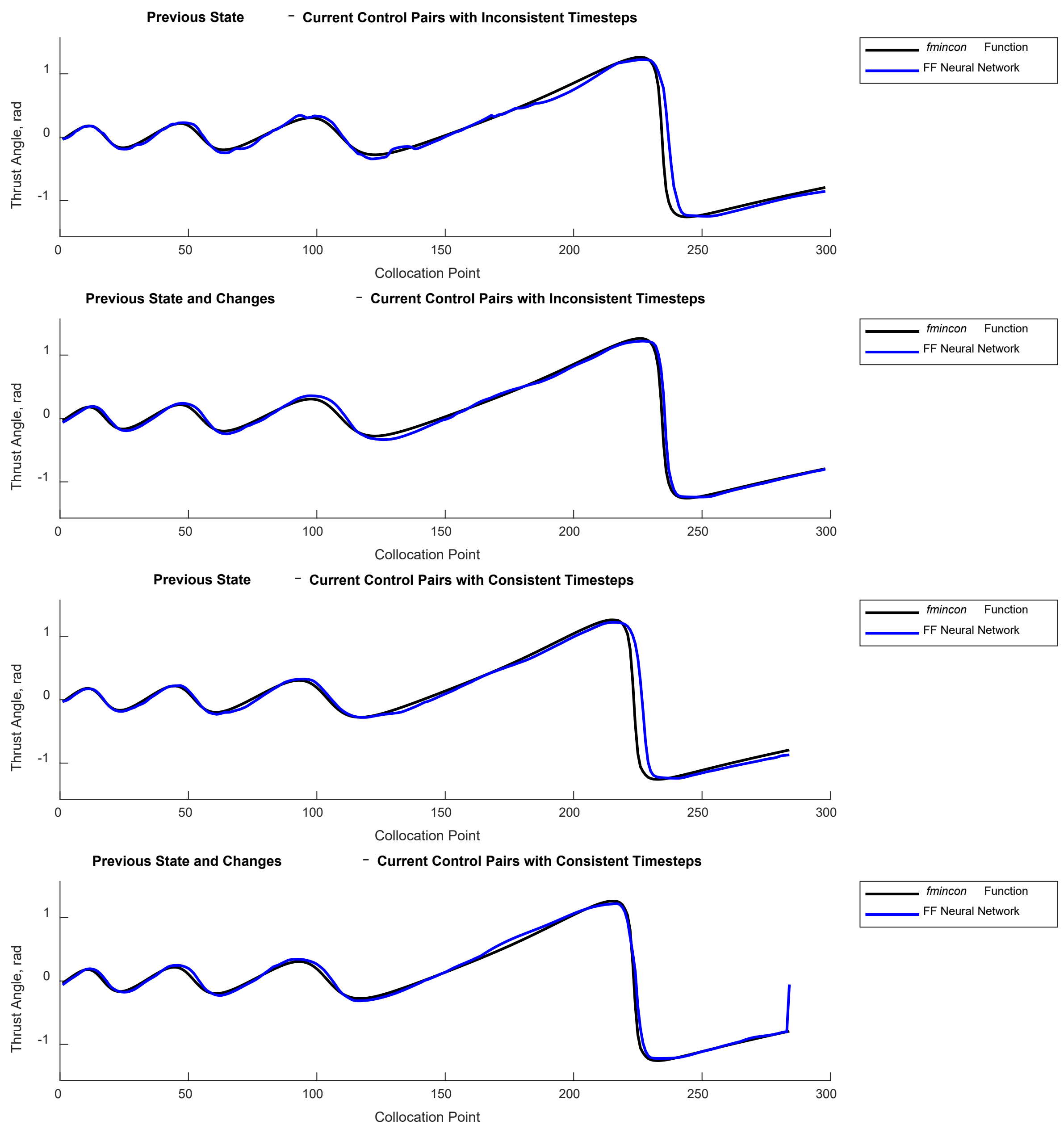


Figure 5.8: Time-Optimal Closed-Loop Feedforward Test Case Results (Thrust Angles)

Table XXVI: Summary of Time-Optimal Closed-Loop Feedforward Tuning and Training Results

Feedforward Neural Network	Tuning Time	Test Case RMSE	Training Time
Previous State – Current Control Pairs with Inconsistent Timesteps	1.63 hr	4.0379e-5 DU/TU^2 0.1336 rad	0.04 hr
Previous State and Changes – Current Control Pairs with Inconsistent Timesteps	1.84 hr	2.4870e-5 DU/TU^2 0.0762 rad	0.05 hr
Previous State – Current Control Pairs with Consistent Timesteps	1.62 hr	3.0371e-5 DU/TU^2 0.1783 rad	0.05 hr
Previous State and Changes – Current Control Pairs with Consistent Timesteps	1.58 hr	2.3940e-5 DU/TU^2 0.0743 rad	0.04 hr

When considering the thrust acceleration predictions for the time-optimal open-loop case, the LSTM neural networks with a minimum RMSE of 1.8116e-5 DU/TU^2 (using consistent timesteps with the current states as inputs) performed on par (i.e., with less than 10 percent difference) with the feedforward neural networks with a minimum RMSE of 1.8545e-5 DU/TU^2 (again, using consistent timesteps with the current states as inputs). However, when considering the thrust angle predictions for the time-optimal open-loop case, the LSTM neural networks with a minimum RMSE of 0.0228 radians (using inconsistent timesteps with the current states and the desired changes for those states as inputs) performed worse than the feedforward neural networks with a minimum RMSE of 0.0158 radians (again, using inconsistent timesteps with the current states and the desired changes for those states as inputs). Also, for the time-optimal open-loop case, the LSTM neural networks had significantly longer average training times than the feedforward neural networks.

When considering the control predictions for the time-optimal closed-loop case, the LSTM neural networks with minimum RMSEs of 1.9001e-5 DU/TU^2 and 0.0663 radians (using consistent timesteps with the previous states and the desired changes for those states as inputs) performed better than the feedforward neural networks with minimum RMSEs of 2.3940e-5 DU/TU^2 and 0.0743 radians (again, using consistent timesteps with the previous states and the desired changes for those states as inputs). However, as in the time-optimal open-loop case, the LSTM neural networks had significantly longer average training times than the feedforward neural networks.

Overall, though the LSTM neural networks typically performed on par with or better than their feedforward equivalents, particularly in the time-optimal closed-loop case, the significantly shorter average training times of the feedforward neural networks was a considerable advantage. Attempting to narrow the gap between the average training times for the LSTM neural networks and their feedforward equivalents, a batch size investigation was conducted to determine the effect of increasing batch size on average RMSE and training time for both the LSTM neural networks and the feedforward neural networks.

5.1.4 Batch Size Investigation

Before the results from the batch size investigation for the LSTM neural networks are discussed, it is necessary to mention a significant trend observed from their feedforward equivalents. Overall, increasing the batch size for the feedforward neural networks significantly and negatively impacted the ability of the feedforward neural networks to learn. Figure 5.9 shows an example representative of the relationships between batch size and the average predicted control. The upper plot of Figure 5.9 shows the thrust acceleration predictions by the feedforward neural network with the best time-optimal open-loop performance for thrust acceleration (using consistent timesteps with the current states as inputs). The lower plot of Figure 5.9 shows the thrust angle predictions by the feedforward neural network with the best time-optimal open-loop performance for thrust angle (using inconsistent timesteps with the current states and the desired changes for those states as inputs). As seen in Figure 5.9, even when the batch sizes for the feedforward neural networks were only increased to five, the ability of the feedforward neural networks to learn the control features was significantly diminished. This trend was present, and typically more severe, for all feedforward neural networks tuned and trained on all time-optimal datasets (both open-loop and closed-loop), regardless of the set of inputs.

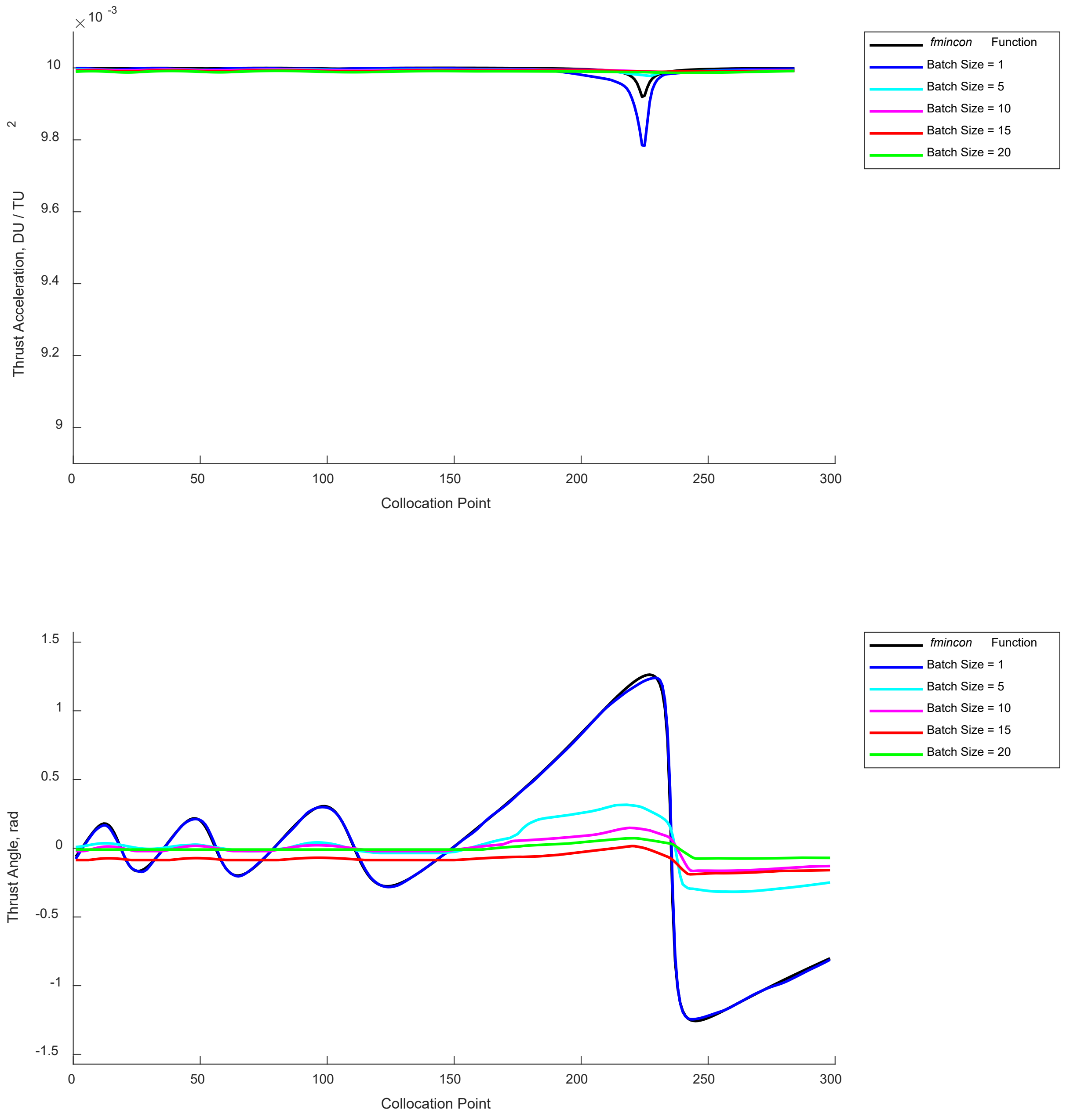


Figure 5.9: Example Representative of Time-Optimal Feedforward Batch Size Investigation

The results from the batch size investigation for the LSTM neural networks for the time-optimal open-loop case are shown in Figure 5.10 through Figure 5.13. Figure 5.10 shows the relationships between batch size and the average predicted control RMSEs as well as the average training times. Figure 5.11, Figure 5.12, and Figure 5.13 compare those relationships (thrust acceleration, thrust angle, and training time, respectively) to the results from their feedforward equivalents with a batch size of one.

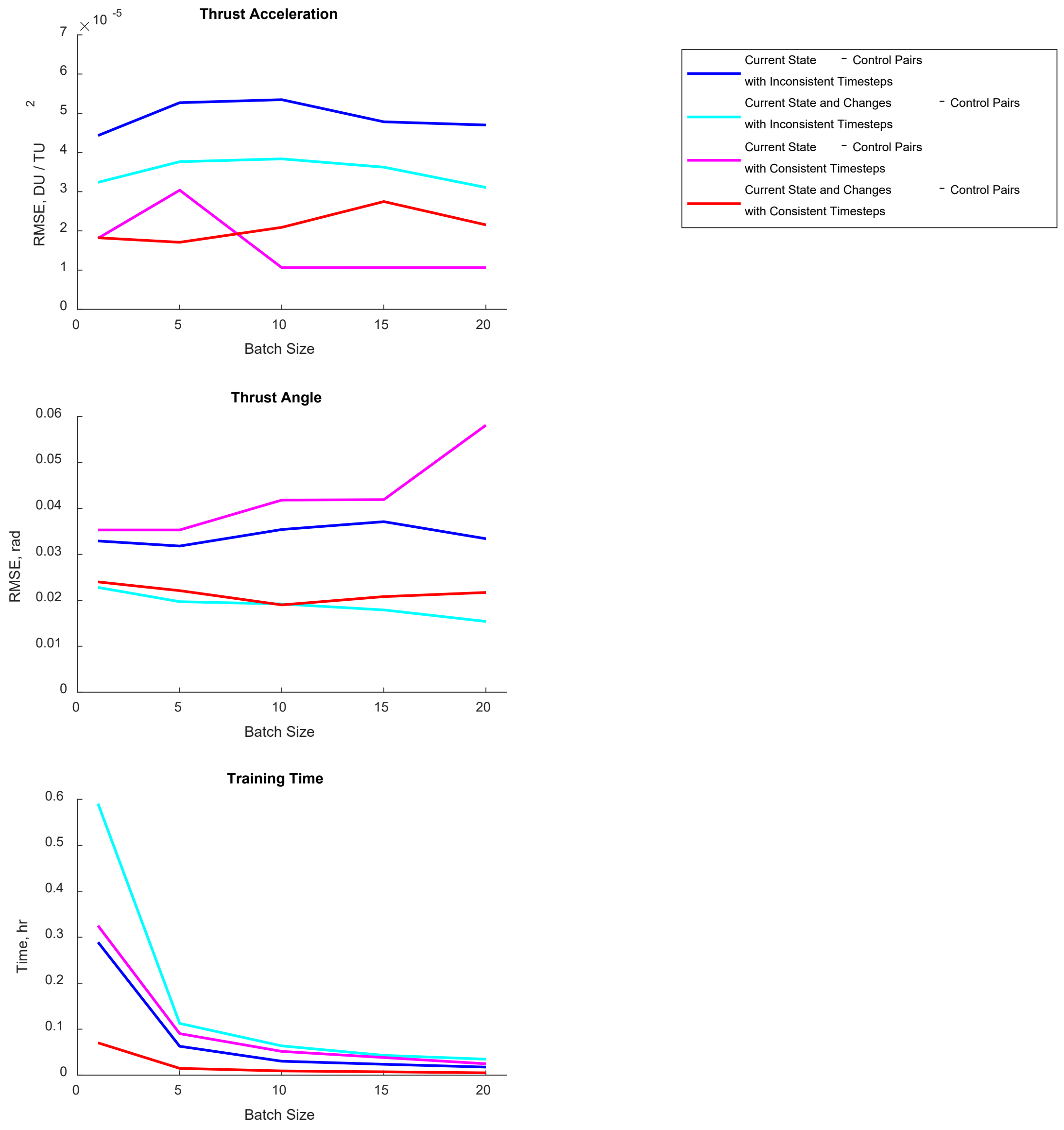


Figure 5.10: Time-Optimal Open-Loop LSTM Batch Size Investigation

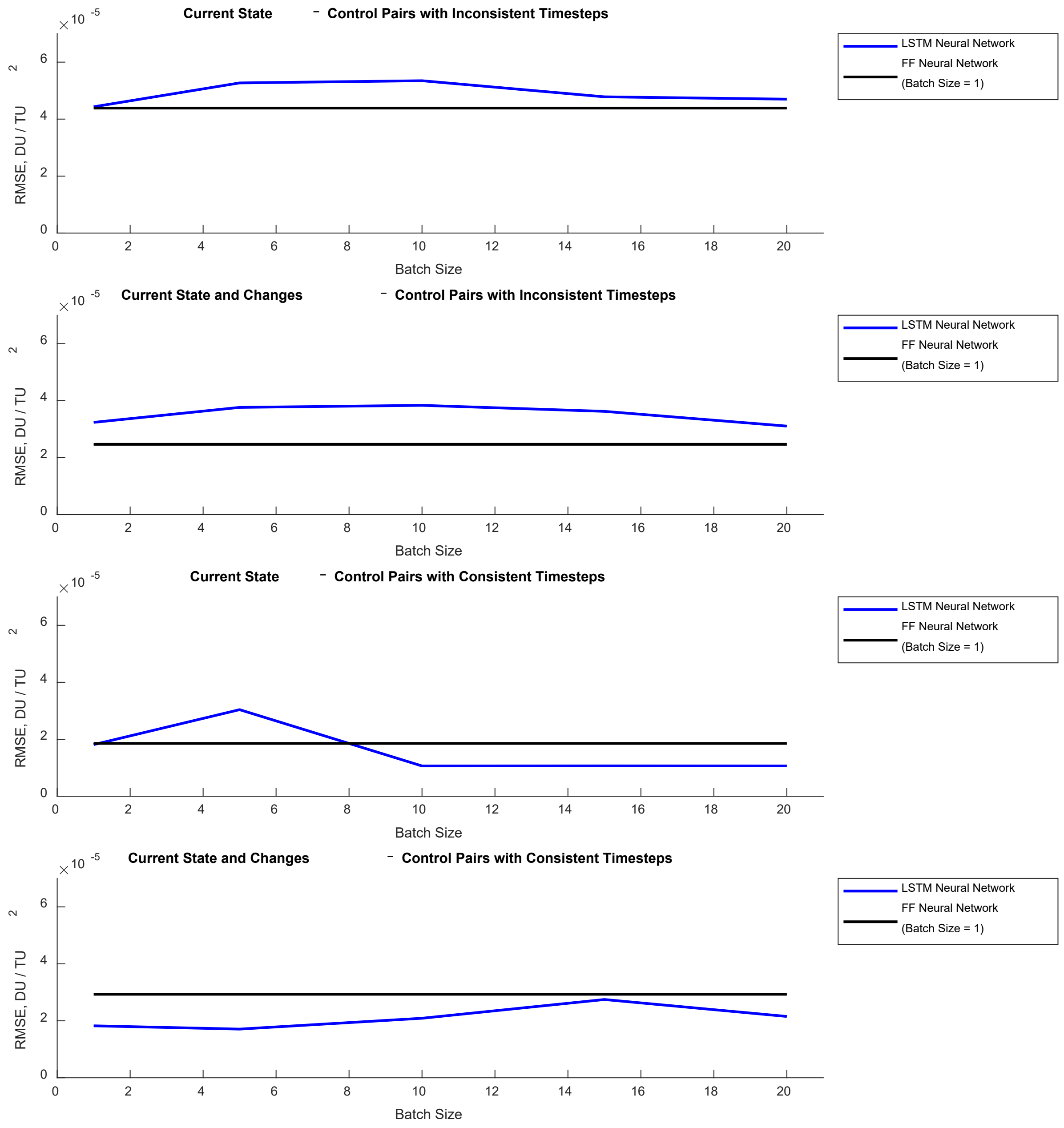


Figure 5.11: Time-Optimal Open-Loop LSTM Batch Size Investigation (Thrust Accelerations)

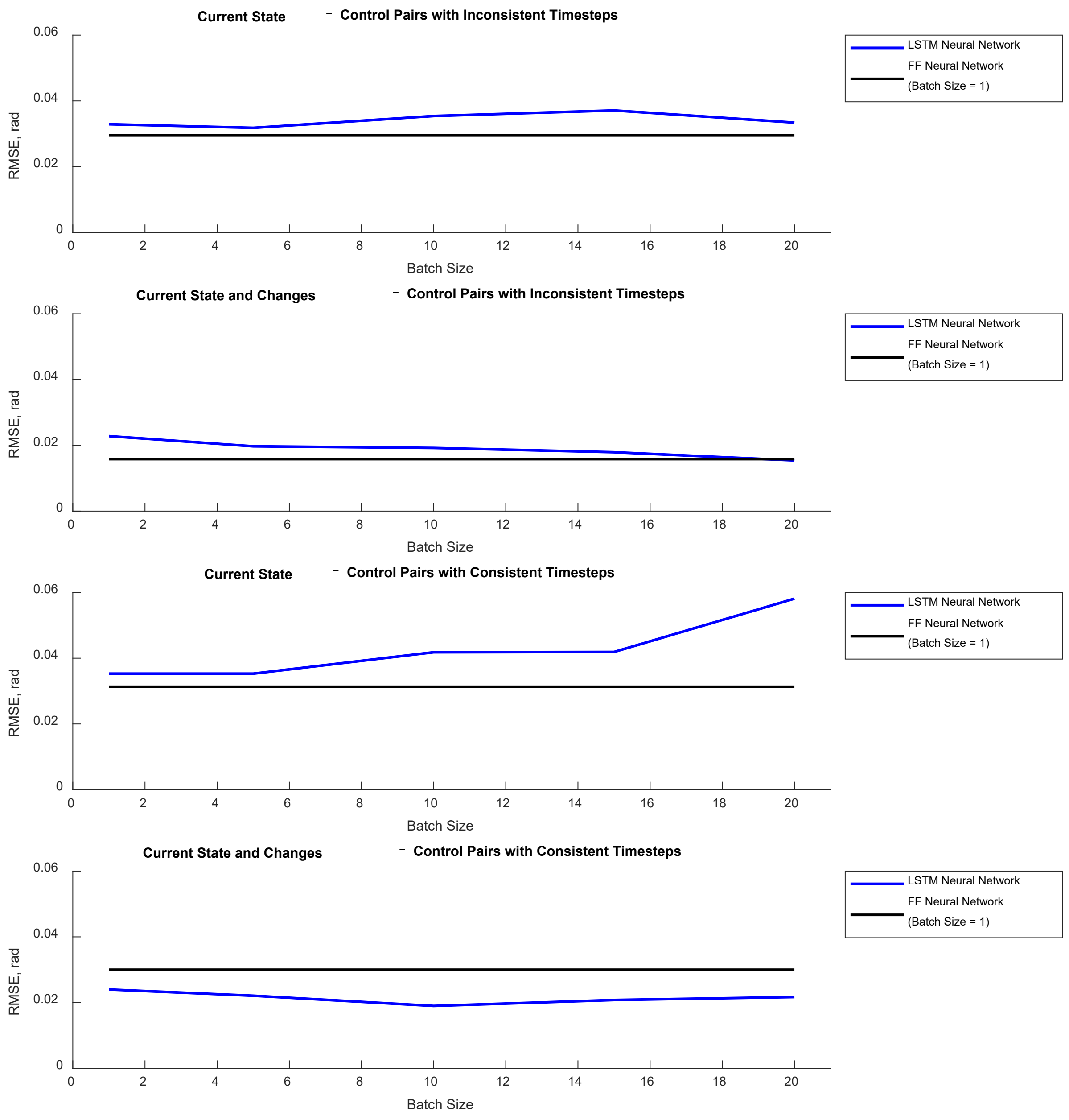


Figure 5.12: Time-Optimal Open-Loop LSTM Batch Size Investigation (Thrust Angles)

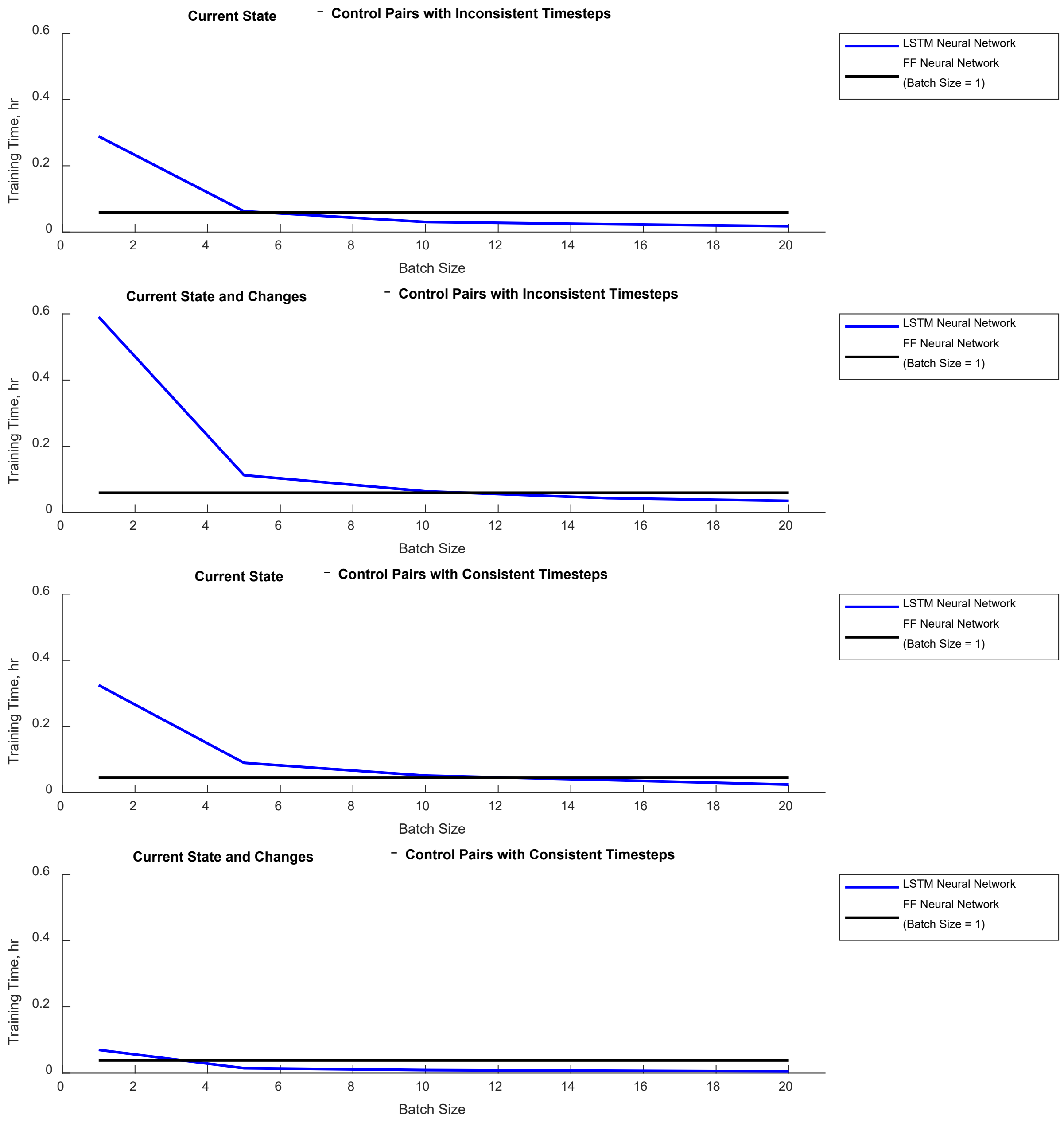


Figure 5.13: Time-Optimal Open-Loop LSTM Batch Size Investigation (Training Times)

As seen in these figures, increasing the batch size for the LSTM neural networks significantly improved the average training time with minimal impact on the average predicted control RMSE. In fact, with some models, slightly improved performance was observed. Diminishing returns for the average training time were observed near a batch size of 20 and, after a batch size of 15, the average training times of the LSTM neural networks were consistently shorter than the feedforward neural networks. Specifically, with a batch size of 20, the LSTM neural networks (when using consistent timesteps) had a lower RMSE for thrust acceleration prediction than the feedforward neural networks. When considering thrust angle prediction with, again, a batch size of 20, the LSTM neural networks (when using the current states and the desired changes for those states as inputs) had a lower RMSE than the feedforward neural networks.

The results from the batch size investigation of the LSTM neural networks for the time-optimal closed-loop case are shown in Figure 5.14 through Figure 5.17. Figure 5.14 shows the relationships between batch size and the average predicted control RMSEs as well as the average training times. Figure 5.15, Figure 5.16, and Figure 5.17 compare those relationships (thrust acceleration, thrust angle, and training time, respectively) to the results from their feedforward equivalents with a batch size of one. As in the time-optimal open-loop case, increasing the batch size for the LSTM neural networks significantly improved the average training time with typically minimal impact on the average predicted control RMSE. In fact, for thrust acceleration prediction, slightly improved performance was observed. Again, as in the time-optimal open-loop case, diminishing returns for the average training time were observed near a batch size of 20 and, with a batch size of 20, the average training times of the LSTM neural networks were typically shorter than the feedforward neural networks. With a batch size of 20, the LSTM neural networks typically had lower RMSEs for thrust acceleration prediction than the feedforward neural networks.

However, when considering thrust angle prediction with, again, a batch size of 20, the LSTM neural networks typically had higher RMSEs than the feedforward neural networks.

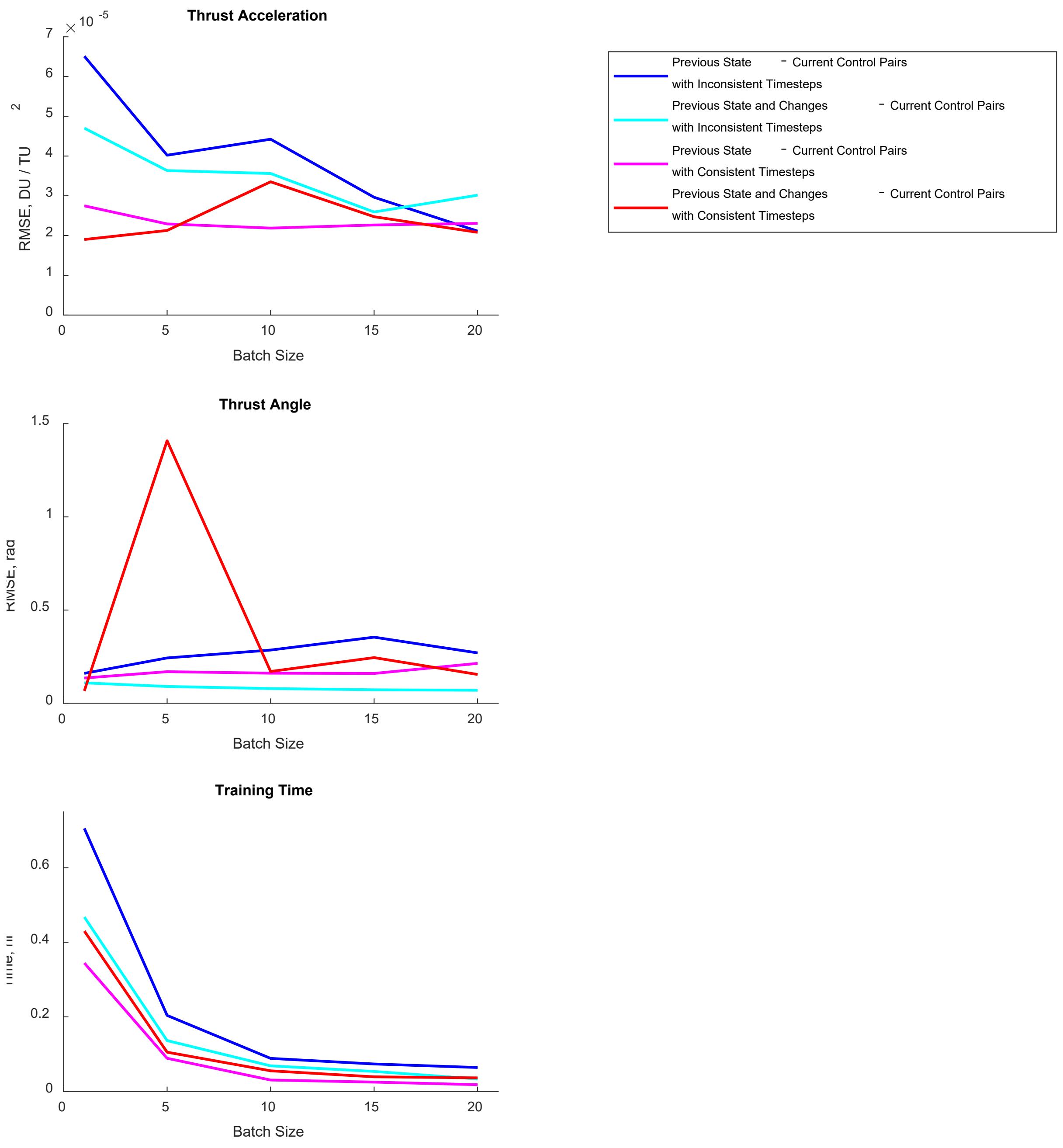


Figure 5.14: Time-Optimal Closed-Loop LSTM Batch Size Investigation

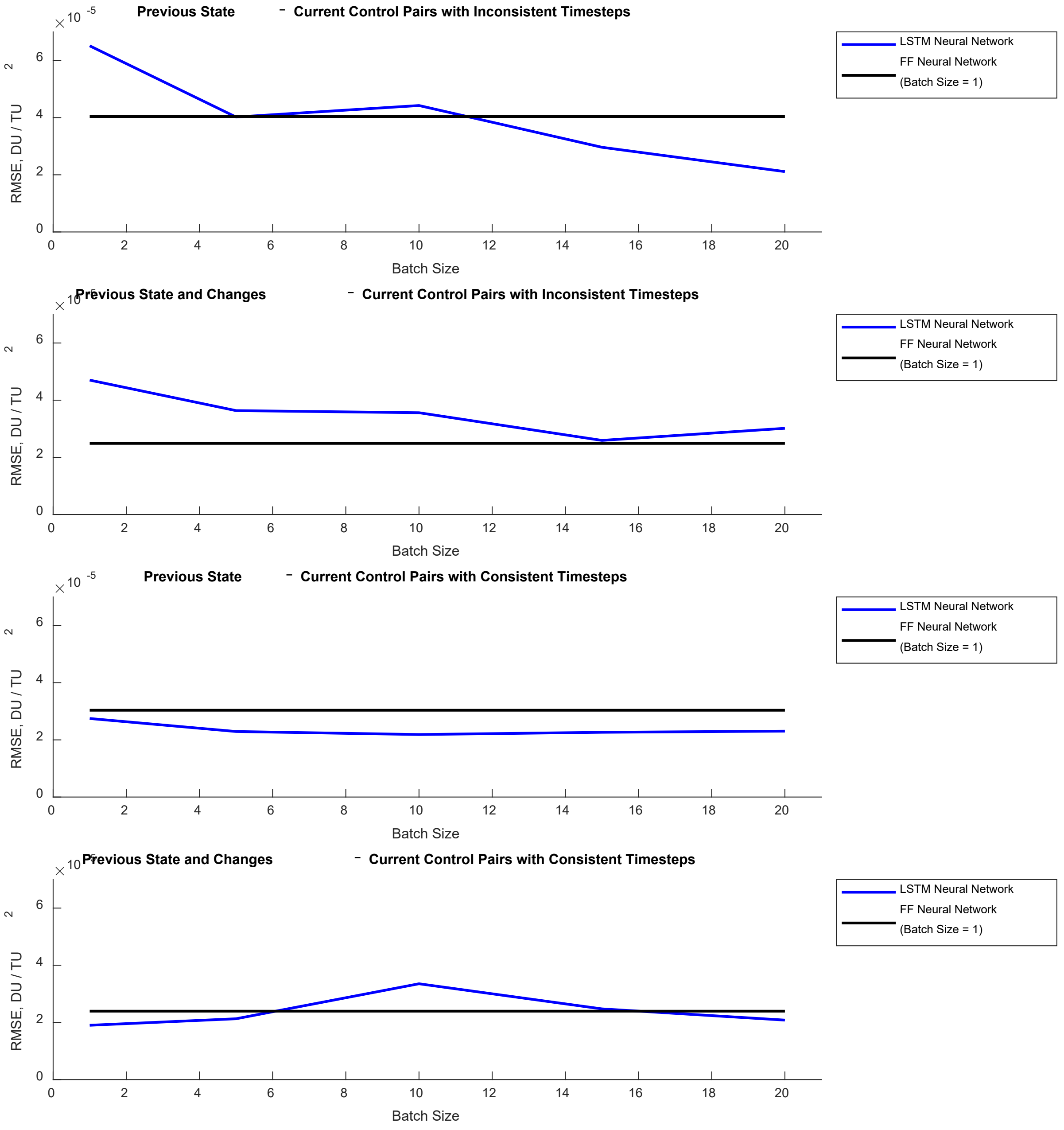


Figure 5.15: Time-Optimal Closed-Loop LSTM Batch Size Investigation (Thrust Accelerations)

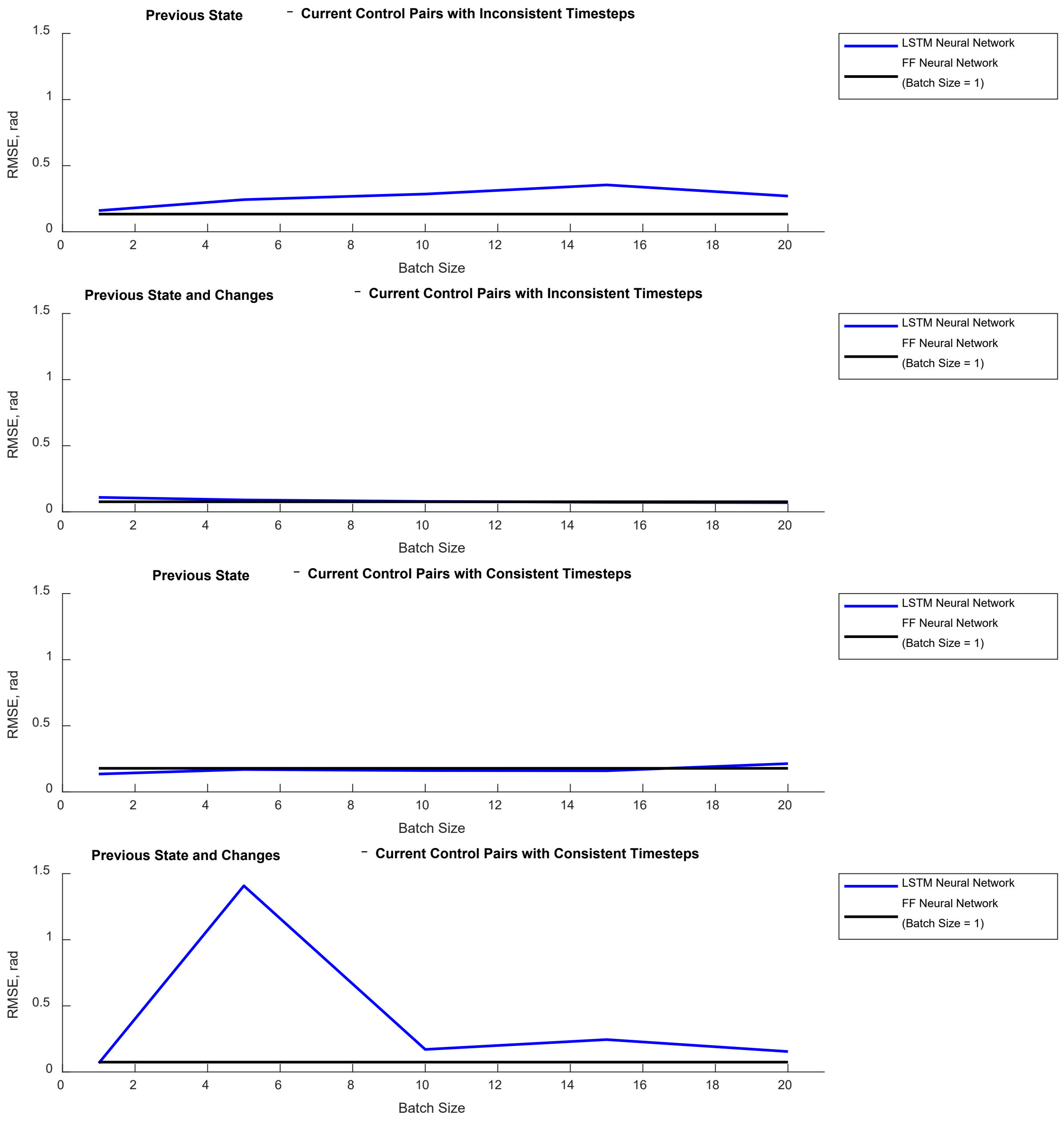


Figure 5.16: Time-Optimal Closed-Loop LSTM Batch Size Investigation (Thrust Angles)

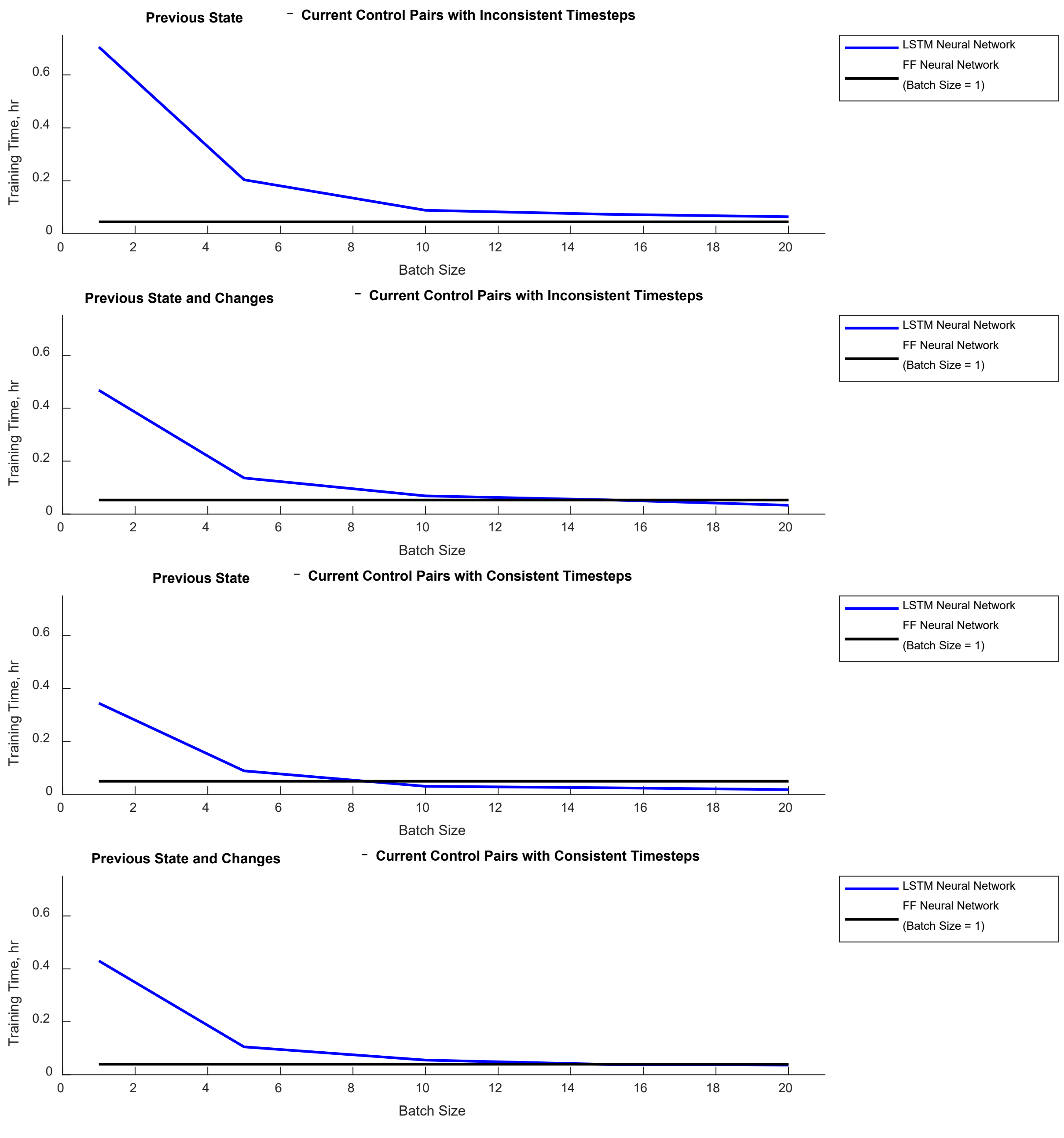


Figure 5.17: Time-Optimal Closed-Loop LSTM Batch Size Investigation (Training Times)

5.1.5 Final Comparison to Feedforward Architectures

When considering the thrust acceleration predictions for the time-optimal open-loop case, the LSTM neural networks with a minimum RMSE of 1.0632e-5 DU/TU^2 (using a batch size of 20 and consistent timesteps with the current states as inputs) performed better than the feedforward neural networks with a minimum RMSE of 1.8545e-5 DU/TU^2 (using a batch size of one and, again, consistent timesteps with the current states as inputs). When considering the thrust angle predictions for the time-optimal open-loop case, the LSTM neural networks with a minimum RMSE of 0.0154 radians (using a batch size of 20 and inconsistent timesteps with the current states and the desired changes for those states as inputs) performed on par (i.e., with less than 10 percent difference) with the feedforward neural networks with a minimum RMSE of 0.0158 radians (using a batch size of one and, again, inconsistent timesteps with the current states and the desired changes for those states as inputs). Also, for the time-optimal open-loop case, the LSTM neural networks (89 to 125 seconds, respectively) had shorter average training times (by about 41 to 46 percent) than the feedforward neural networks (166 to 213 seconds, respectively).

When considering the thrust acceleration predictions for the time-optimal closed-loop case, the LSTM neural networks with a minimum RMSE of 2.0808e-5 DU/TU^2 (using a batch size of 20 and consistent timesteps with the previous states and the desired changes for those states as inputs) performed better than the feedforward neural networks with a minimum RMSE of 2.3940e-5 DU/TU^2 (using a batch size of one and, again, consistent timesteps with the previous states and the desired changes for those states as inputs). When considering the thrust angle predictions for the time-optimal closed-loop case, the LSTM neural networks with a minimum RMSE of 0.0696 radians (using a batch size of 20 and inconsistent timesteps with the previous states and the desired changes for those states as inputs) performed on par with the feedforward neural networks with a minimum RMSE of 0.0743 radians (using a batch size of one and consistent timesteps with the previous states and the desired changes for those states as inputs). Also, for the time-optimal closed-loop case, the LSTM neural networks (131 to 120 seconds, respectively) had shorter average training times (by about 8 to 16 percent) than the feedforward neural networks (143 seconds).

Overall, in consideration of the results for the time-optimal case (both open-loop and closed-loop), the following conclusions were reached.

1. For the time-optimal open-loop case, the LSTM neural networks performed on par with or better than their feedforward equivalents in terms of average RMSE.
2. For the time-optimal closed-loop case, the LSTM neural networks performed on par with or better than their feedforward equivalents in terms of average RMSE.
3. For both the time-optimal open-loop case and the time-optimal closed-loop case, the LSTM neural networks with an appropriate batch size performed (i.e., trained) faster than their feedforward equivalents.
4. For the time-optimal open-loop case, using the current states as inputs improved thrust acceleration prediction while using the current states and the desired changes for those states as inputs improved thrust angle prediction. These trends were observed for both the LSTM neural networks and their feedforward equivalents.
5. For the time-optimal closed-loop case, using the previous states and the desired changes for those states as inputs improved both thrust acceleration prediction and thrust angle prediction for both the LSTM neural networks and their feedforward equivalents.

6. For both the time-optimal open-loop case and the time-optimal closed-loop case, consistent timesteps improved thrust acceleration prediction while inconsistent timesteps typically improved thrust angle prediction.

5.2 Constrained Fuel-Optimal

The results for the LSTM neural networks, including a comparison to their feedforward equivalents, when tuned and trained on the fuel-optimal datasets with constrained transfer times will be discussed here. Two datasets (inconsistent and consistent timesteps) as well as two sets of inputs (states and states with the desired changes for those states) were considered for both the open-loop case and the closed-loop case. In accordance with the tuning process, the neural networks were trained over a maximum of 60 epochs with a batch size of one using the *MeanSquaredError* regression loss and the *Adam* optimizer. Early stopping was used to speed-up the overall training process by monitoring the validation loss with patience of 15 epochs. Best weights were restored. Finally, each neural network was re-trained 20 times. The results presented here are representative of the averages for those 20 trials.

5.2.1 Constrained Fuel-Optimal Open-Loop

Table XXVII through Table XXX present the tuned parameters and hyperparameters of the LSTM neural networks for the constrained fuel-optimal open-loop case. The LSTM neural networks presented in Table XXVII and Table XXVIII were tuned and trained on the constrained fuel-optimal dataset with inconsistent timesteps using the current states and the current states with the desired changes for those states as inputs, respectively. The LSTM neural networks presented in Table XXIX and Table XXX were tuned and trained on the constrained fuel-optimal dataset with consistent timesteps using, again, the current states and the current states with the desired changes for those states as inputs, respectively. Overall, tuning on the constrained fuel-optimal dataset with consistent timesteps resulted in LSTM neural networks with slightly fewer layers and neurons compared to tuning on the constrained fuel-optimal dataset with inconsistent timesteps.

Table XXVII: Fuel-Optimal (Constrained) Open-Loop LSTM Tuning Results (Current States – Control Pairs with Inconsistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
LSTM Layer (Input)	160	Default
LSTM Layer (Hidden 1)	64	
LSTM Layer (Hidden 2)	480	
LSTM Layer (Output)	512	
Dense Layer	2	"relu"
Learning Rate = 0.001		

Table XXVIII: Fuel-Optimal (Constrained) Open-Loop LSTM Tuning Results (Current States and Changes – Control Pairs with Inconsistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
LSTM Layer (Input)	416	Default
LSTM Layer (Hidden 1)	128	
LSTM Layer (Hidden 2)	352	
LSTM Layer (Output)	448	
Dense Layer	2	"tanh"
Learning Rate = 0.001		

Table XXIX: Fuel-Optimal (Constrained) Open-Loop LSTM Tuning Results (Current States – Control Pairs with Consistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
LSTM Layer (Input)	384	Default
LSTM Layer (Hidden 1)	64	
LSTM Layer (Output)	416	
Dense Layer	2	"sigmoid"
Learning Rate = 0.001		

Table XXX: Fuel-Optimal (Constrained) Open-Loop LSTM Tuning Results (Current States and Changes – Control Pairs with Consistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
LSTM Layer (Input)	64	Default
LSTM Layer (Hidden 1)	128	
LSTM Layer (Hidden 2)	480	
LSTM Layer (Output)	352	
Dense Layer	2	"sigmoid"
Learning Rate = 0.001		

Figure 5.18 and Figure 5.19 compare the current control predictions (thrust accelerations and angles, respectively) from the tuned LSTM neural networks to the optimized values from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. Table XXXI presents an overall summary of the tuning and training results. When considering the control predictions, better performance in terms of average RMSE was observed from the LSTM neural networks tuned and trained on the constrained fuel-optimal dataset with consistent timesteps.

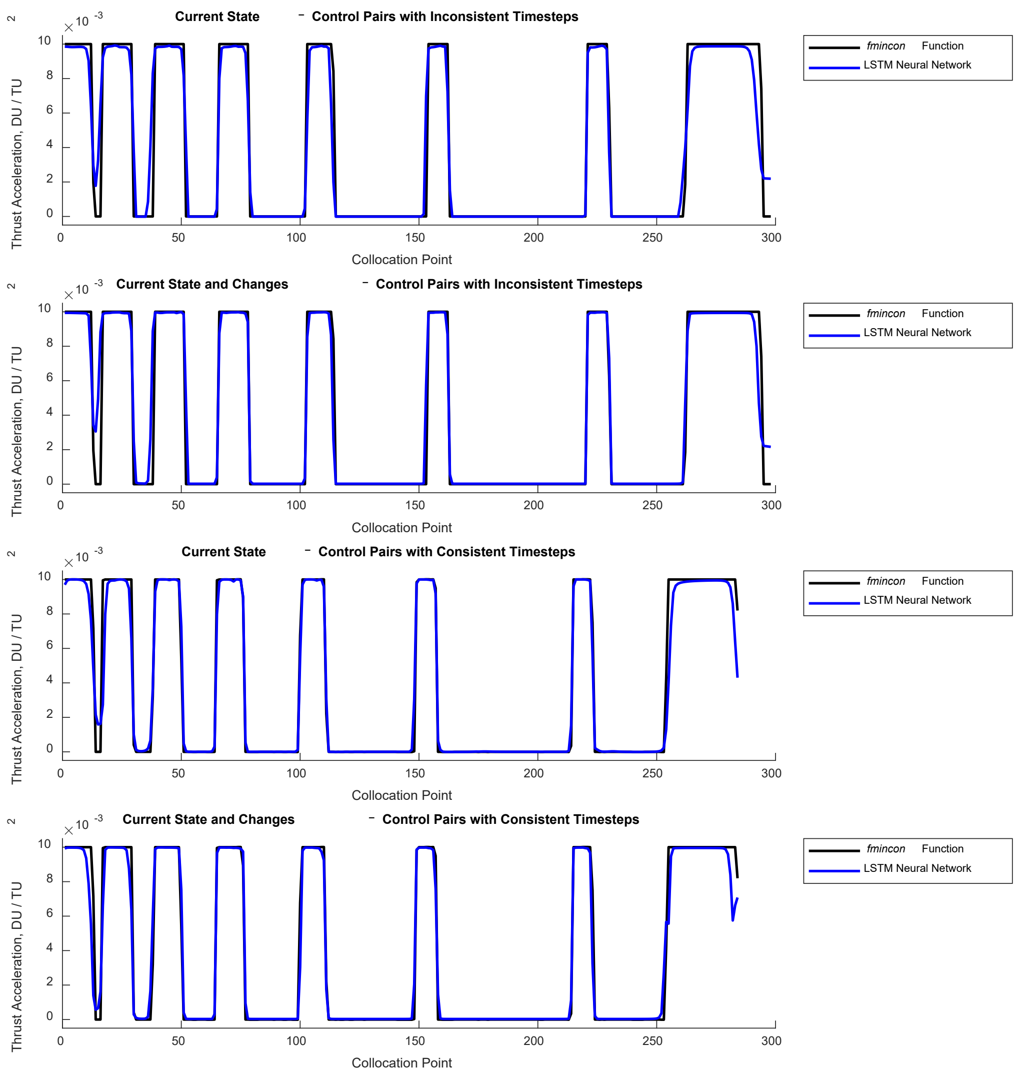


Figure 5.18: Constrained Fuel-Optimal Open-Loop LSTM Test Case Results (Thrust Accelerations)

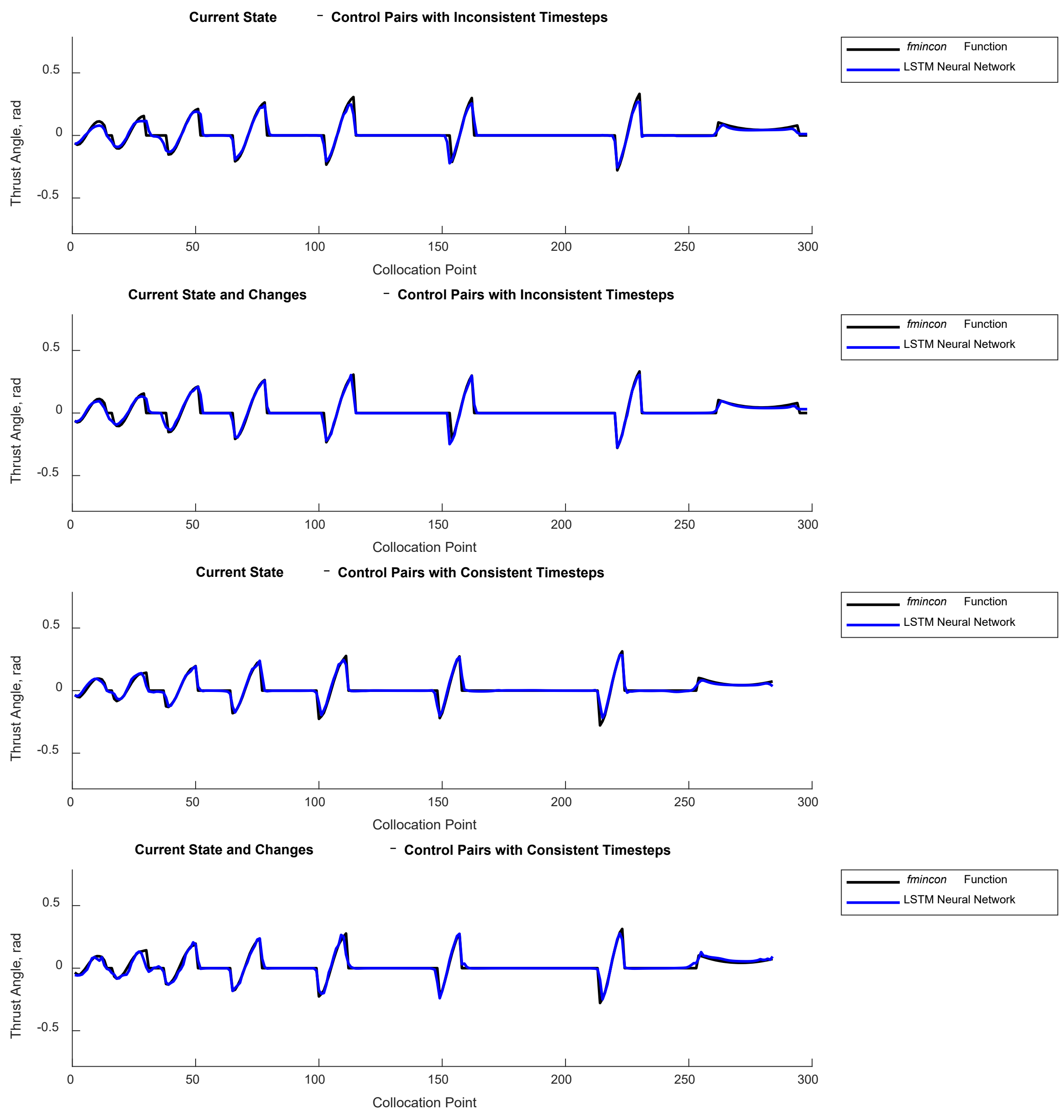


Figure 5.19: Constrained Fuel-Optimal Open-Loop LSTM Test Case Results (Thrust Angles)

Table XXXI: Summary of Constrained Fuel-Optimal Open-Loop LSTM Tuning and Training Results

LSTM Neural Network	Tuning Time	Test Case RMSE	Training Time
Current State – Control Pairs with Inconsistent Timesteps	16.17 hr	1.2591e-3 DU/TU^2 0.0264 rad	0.48 hr
Current State and Changes – Control Pairs with Inconsistent Timesteps	15.55 hr	1.1380e-3 DU/TU^2 0.0252 rad	0.42 hr
Current State – Control Pairs with Consistent Timesteps	13.17 hr	8.3816e-4 DU/TU^2 0.0224 rad	0.27 hr
Current State and Changes – Control Pairs with Consistent Timesteps	12.80 hr	8.5982e-4 DU/TU^2 0.0239 rad	0.36 hr

5.2.2 Constrained Fuel-Optimal Closed-Loop

Note, the results for the constrained fuel-optimal closed-loop case were, overall, unfavorable and were only included for the sake of completeness. Table XXXII through Table XXXV present the tuned parameters and hyperparameters of the LSTM neural networks for the constrained fuel-optimal closed-loop case. The LSTM neural networks presented in Table XXXII and Table XXXIII were tuned and trained on the constrained fuel-optimal dataset with inconsistent timesteps using the previous states and the previous states with the desired changes for those states as inputs, respectively. The LSTM neural networks presented in Table XXXIV and Table XXXV were tuned and trained on the constrained fuel-optimal dataset with consistent timesteps using, again, the previous states and the previous states with the desired changes for those states as inputs, respectively. No significant trends were observed from the tuned architectures other than, compared to the constrained fuel-optimal open-loop case, the relative size of the tuned architectures for the constrained fuel-optimal closed-loop case decreased.

Table XXXII: Fuel-Optimal (Constrained) Closed-Loop LSTM Tuning Results (Previous States – Current Control Pairs with Inconsistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
LSTM Layer (Input)	448	Default
LSTM Layer (Hidden 1)	96	
LSTM Layer (Output)	352	
Dense Layer	2	"relu"
Learning Rate = 0.001		

Table XXXIII: Fuel-Optimal (Constrained) Closed-Loop LSTM Tuning Results (Previous States and Changes – Current Control Pairs with Inconsistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
LSTM Layer (Input)	352	Default
LSTM Layer (Hidden 1)	480	
LSTM Layer (Output)	256	
Dense Layer	2	"relu"
Learning Rate = 0.001		

Table XXXIV: Fuel-Optimal (Constrained) Closed-Loop LSTM Tuning Results (Previous States – Current Control Pairs with Consistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
LSTM Layer (Input)	384	Default
LSTM Layer (Hidden 1)	192	
LSTM Layer (Output)	448	
Dense Layer	2	"sigmoid"
Learning Rate = 0.001		

Table XXXV: Fuel-Optimal (Constrained) Closed-Loop LSTM Tuning Results (Previous States and Changes – Current Control Pairs with Consistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
LSTM Layer (Input)	384	Default
LSTM Layer (Hidden 1)	448	
LSTM Layer (Hidden 2)	96	
LSTM Layer (Output)	160	
Dense Layer	2	"sigmoid"
Learning Rate = 0.001		

Figure 5.20 and Figure 5.21 compare the current control predictions (thrust accelerations and angles, respectively) from the tuned LSTM neural networks to the optimized values from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. Table XXXVI presents an overall summary of the tuning and training results. Overall, closing the loop for the constrained fuel-optimal case introduced instability into the training and inference process, resulting in the LSTM neural networks for the constrained fuel-optimal closed-loop case performing much worse in terms of average RMSE than the LSTM neural networks for the constrained fuel-optimal open-loop case. Due to this instability, not only did the LSTM neural

networks struggle to infer the limits, particularly the upper limit, of the thrust acceleration, but the LSTM neural networks also struggled to predict the correct locations for propulsive maneuvers in the trajectory, particularly for propulsive maneuvers located near the end of the trajectory. However, better performance in terms of average RMSE was observed from the LSTM neural networks tuned and trained on the constrained fuel-optimal dataset with consistent timesteps. When tuned and trained on the constrained fuel-optimal dataset with consistent timesteps, the LSTM neural networks were able to infer, consistently, the upper limit of the thrust acceleration as well as better predict, but not consistently, the correct locations for propulsive maneuvers. Note, attempting to address this instability, the effects of modifying the following parameters and hyperparameters were investigated.

1. *Inputs.* The effects of including the time, the time remaining, and the control at the previous timesteps as well as including information from multiple previous timesteps and information from current timesteps as inputs were tested and compared.
2. *Initialization.* Different weight and bias initializers were tested and compared.
3. *Training.* Different optimizers as well as the effects of modifying the parameters within those optimizers, particularly any constants related to learning and numerical stability, were tested and compared. A weight study was also conducted, showing significant differences from trial to trial for the weights and biases of the input and output gates as well as the memory cells for the LSTM neural networks. The effect of removing all biases was tested.
4. *Underfitting / Overfitting.* The effects of decreasing / increasing the number of epochs, the batch size, and the validation split as well as the patience for early stopping were tested and compared.

Overall, modifying the preceding parameters and hyperparameters did not have any significantly positive effects on the observed instability. As such, this behavior was eventually attributed to a combination of generalization and integration error, magnified by the discontinuous nature of the constrained fuel-optimal problem.

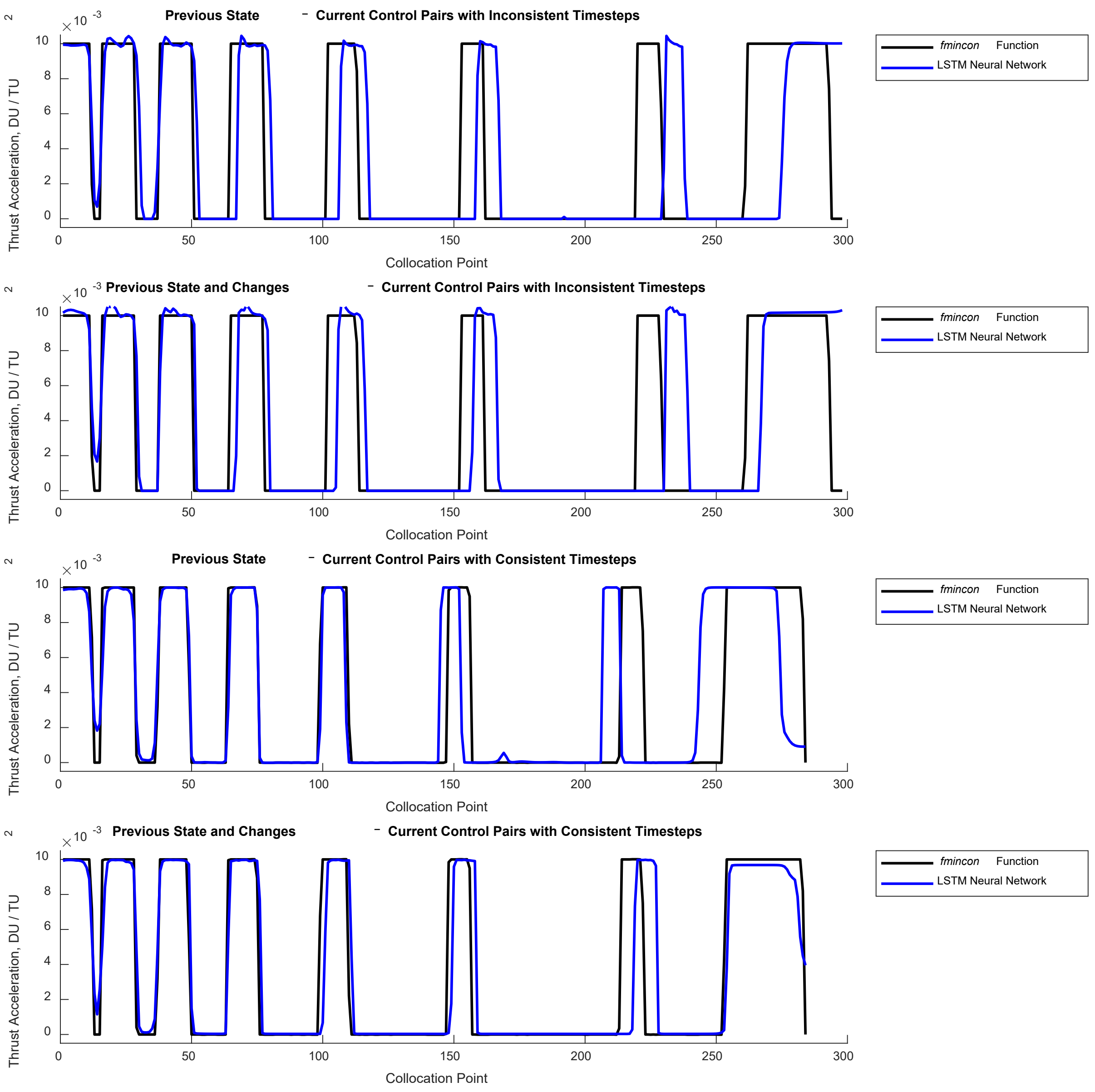


Figure 5.20: Constrained Fuel-Optimal Closed-Loop LSTM Test Case Results (Thrust Accelerations)

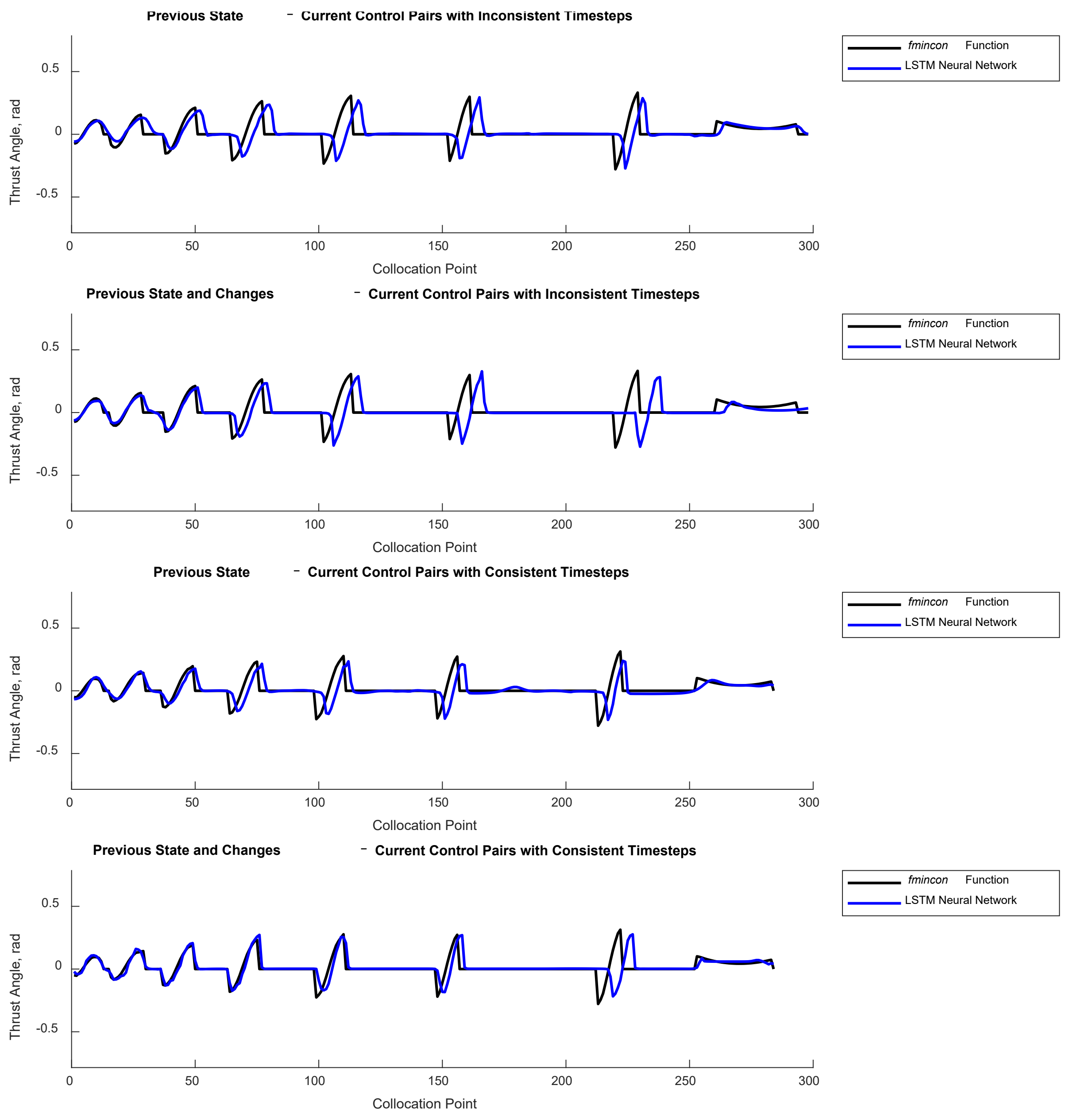


Figure 5.21: Constrained Fuel-Optimal Closed-Loop LSTM Test Case Results (Thrust Angles)

Table XXXVI: Summary of Constrained Fuel-Optimal Closed-Loop LSTM Tuning and Training Results

LSTM Neural Network	Tuning Time	Test Case RMSE	Training Time
Previous State – Current Control Pairs with Inconsistent Timesteps	12.98 hr	4.6160e-3 DU/TU^2 0.1018 rad	0.28 hr
Previous State and Changes – Current Control Pairs with Inconsistent Timesteps	16.87 hr	4.1382e-3 DU/TU^2 0.1044 rad	0.47 hr
Previous State – Current Control Pairs with Consistent Timesteps	14.05 hr	3.7150e-3 DU/TU^2 0.0819 rad	0.30 hr
Previous State and Changes – Current Control Pairs with Consistent Timesteps	14.92 hr	2.7703e-3 DU/TU^2 0.0733 rad	0.38 hr

5.2.3 Initial Comparison to Feedforward Architectures

Table XXXVII through Table XL present the tuned parameters and hyperparameters of the feedforward neural networks for the constrained fuel-optimal open-loop case. The feedforward neural networks presented in Table XXXVII and Table XXXVIII were tuned and trained on the constrained fuel-optimal dataset with inconsistent timesteps using the current states and the current states with the desired changes for those states as inputs, respectively. The feedforward neural networks presented in Table XXXIX and Table XL were tuned and trained on the constrained fuel-optimal dataset with consistent timesteps using, again, the current states and current states with the desired changes for those states as inputs, respectively. Unlike the LSTM neural networks, tuning on the constrained fuel-optimal dataset with consistent timesteps resulted in LSTM neural networks with slightly more layers and neurons compared to tuning on the constrained fuel-optimal dataset with inconsistent timesteps.

Table XXXVII: Fuel-Optimal (Constrained) Open-Loop Feedforward Tuning Results (Current States – Control Pairs with Inconsistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
Dense Layer (Input)	64	“tanh”
Dense Layer (Hidden 1)	128	“relu”
Dense Layer (Hidden 2)	288	“tanh”
Dense Layer (Output)	352	“tanh”
Dense Layer	2	“sigmoid”
Learning Rate = 0.0001		

Table XXXVIII: Fuel-Optimal (Constrained) Open-Loop Feedforward Tuning Results (Current States and Changes – Control Pairs with Inconsistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
Dense Layer (Input)	128	"relu"
Dense Layer (Hidden 1)	384	"relu"
Dense Layer (Hidden 2)	224	"relu"
Dense Layer (Output)	224	"tanh"
Dense Layer	2	"tanh"
Learning Rate = 0.0001		

Table XXXIX: Fuel-Optimal (Constrained) Open-Loop Feedforward Tuning Results (Current States – Control Pairs with Consistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
Dense Layer (Input)	352	"tanh"
Dense Layer (Hidden 1)	192	"relu"
Dense Layer (Hidden 2)	448	"tanh"
Dense Layer (Hidden 3)	416	"relu"
Dense Layer (Output)	416	"sigmoid"
Dense Layer	2	"sigmoid"
Learning Rate = 0.0001		

Table XL: Fuel-Optimal (Constrained) Open-Loop Feedforward Tuning Results (Current States and Changes – Control Pairs with Consistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
Dense Layer (Input)	448	"tanh"
Dense Layer (Hidden 1)	416	"relu"
Dense Layer (Hidden 2)	160	"sigmoid"
Dense Layer (Output)	192	"relu"
Dense Layer	2	"sigmoid"
Learning Rate = 0.0001		

Figure 5.22 and Figure 5.23 compare the current control predictions (thrust accelerations and angles, respectively) from the tuned feedforward neural networks to the optimized values from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. Table XLI presents an overall summary of the tuning and training results. When considering the control predictions, better performance in terms of average RMSE, like the LSTM neural networks, was observed from the feedforward neural networks tuned and trained on the

constrained fuel-optimal dataset with consistent timesteps as well as with the current states and the desired changes for those states as inputs.

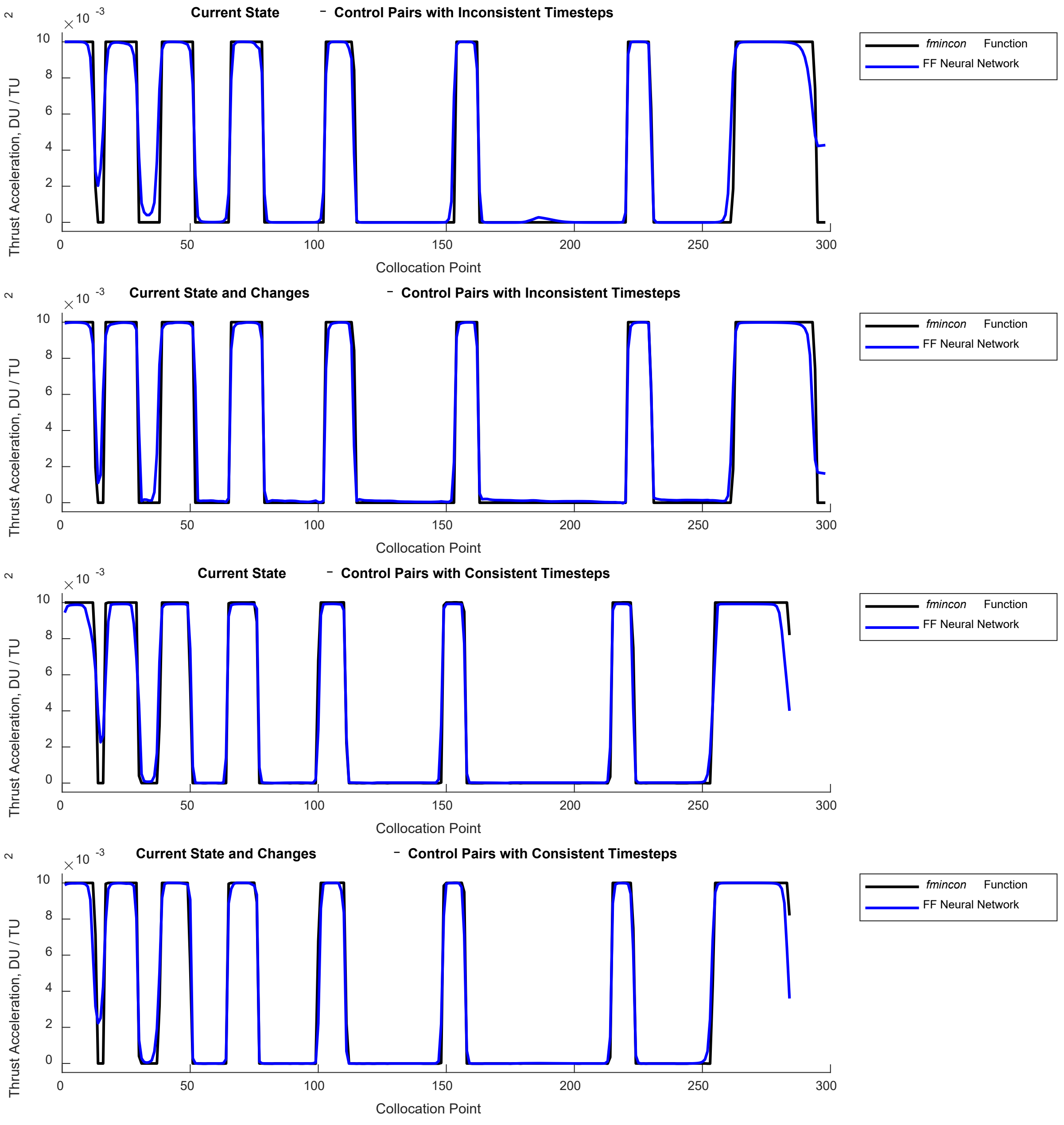


Figure 5.22: Constrained Fuel-Optimal Open-Loop Feedforward Test Case Results (Thrust Accelerations)

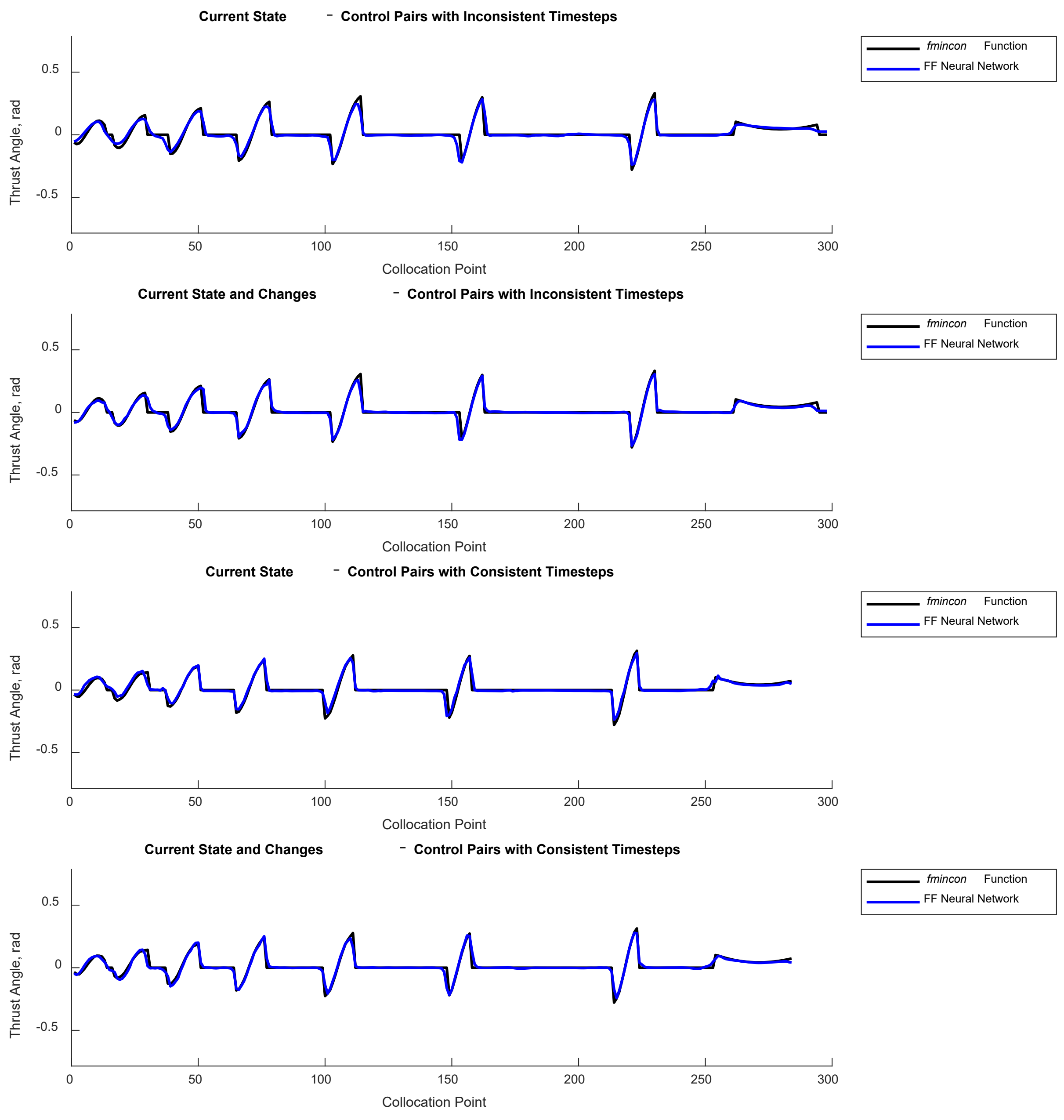


Figure 5.23: Constrained Fuel-Optimal Open-Loop Feedforward Test Case Results (Thrust Angles)

Table XLI: Summary of Constrained Fuel-Optimal Open-Loop Feedforward Tuning and Training Results

Feedforward Neural Network	Tuning Time	Test Case RMSE	Training Time
Current State – Control Pairs with Inconsistent Timesteps	1.97 hr	1.1997e-3 DU/TU^2 0.0275 rad	0.05 hr
Current State and Changes – Control Pairs with Inconsistent Timesteps	2.07 hr	1.0302e-3 DU/TU^2 0.0237 rad	0.06 hr
Current State – Control Pairs with Consistent Timesteps	1.59 hr	8.5885e-4 DU/TU^2 0.0236 rad	0.09 hr
Current State and Changes – Control Pairs with Consistent Timesteps	1.87 hr	8.1203e-4 DU/TU^2 0.0219 rad	0.06 hr

Table XLII through Table XLV present the tuned parameters and hyperparameters of the feedforward neural networks for the constrained fuel-optimal closed-loop case. The feedforward neural networks presented in Table XLII and Table XLIII were tuned and trained on the constrained fuel-optimal dataset with inconsistent timesteps using the previous states and the previous states with the desired changes for those states as inputs, respectively. The feedforward neural networks presented in Table XLIV and Table XLII were tuned and trained on the constrained fuel-optimal dataset with consistent timesteps using, again, the previous states and previous states with the desired changes for those states as inputs, respectively. As in the constrained fuel-optimal open-loop case, tuning on the constrained fuel-optimal dataset with consistent timesteps resulted in feedforward neural networks with slightly more layers and neurons compared to tuning on the constrained fuel-optimal dataset with inconsistent timesteps.

Table XLII: Fuel-Optimal (Constrained) Closed-Loop Feedforward Tuning Results (Previous States – Current Control Pairs with Inconsistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
Dense Layer (Input)	96	“tanh”
Dense Layer (Hidden 1)	480	“relu”
Dense Layer (Hidden 2)	192	“relu”
Dense Layer (Output)	32	“tanh”
Dense Layer	2	“relu”
Learning Rate = 0.0001		

Table XLIII: Fuel-Optimal (Constrained) Closed-Loop Feedforward Tuning Results (Previous States and Changes – Current Control Pairs with Inconsistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
Dense Layer (Input)	480	"relu"
Dense Layer (Hidden 1)	320	"tanh"
Dense Layer (Hidden 2)	224	"sigmoid"
Dense Layer (Output)	128	"relu"
Dense Layer	2	"tanh"
Learning Rate = 0.0001		

Table XLIV: Fuel-Optimal (Constrained) Closed-Loop Feedforward Tuning Results (Previous States – Current Control Pairs with Consistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
Dense Layer (Input)	384	"relu"
Dense Layer (Hidden 1)	64	"relu"
Dense Layer (Hidden 2)	224	"relu"
Dense Layer (Hidden 3)	256	"relu"
Dense Layer (Output)	352	"relu"
Dense Layer	2	"tanh"
Learning Rate = 0.0001		

Table XLV: Fuel-Optimal (Constrained) Closed-Loop Feedforward Tuning Results (Previous States and Changes – Current Control Pairs with Consistent Timesteps)

Layer	Number of Neurons	Activation Function(s)
Dense Layer (Input)	384	"tanh"
Dense Layer (Hidden 1)	416	"relu"
Dense Layer (Hidden 2)	480	"relu"
Dense Layer (Output)	320	"tanh"
Dense Layer	2	"sigmoid"
Learning Rate = 0.0001		

Figure 5.24 and Figure 5.25 compare the current control predictions (thrust accelerations and angles, respectively) from the tuned feedforward neural networks to the optimized values from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. Table XLVI presents an overall summary of the tuning and training results. Overall, like the LSTM neural networks, the results from the feedforward neural networks for the constrained fuel-optimal closed-loop case were unfavorable and were only included for the sake of completeness. Again, like the LSTM neural networks, closing the loop for the constrained fuel-

optimal case introduced a similar instability into the training and inference process. However, like the LSTM neural networks, better performance in terms of average RMSE was observed from the feedforward neural networks tuned and trained on the constrained fuel-optimal dataset with consistent timesteps. Again, the observed instability was eventually attributed to a combination of generalization and integration error.

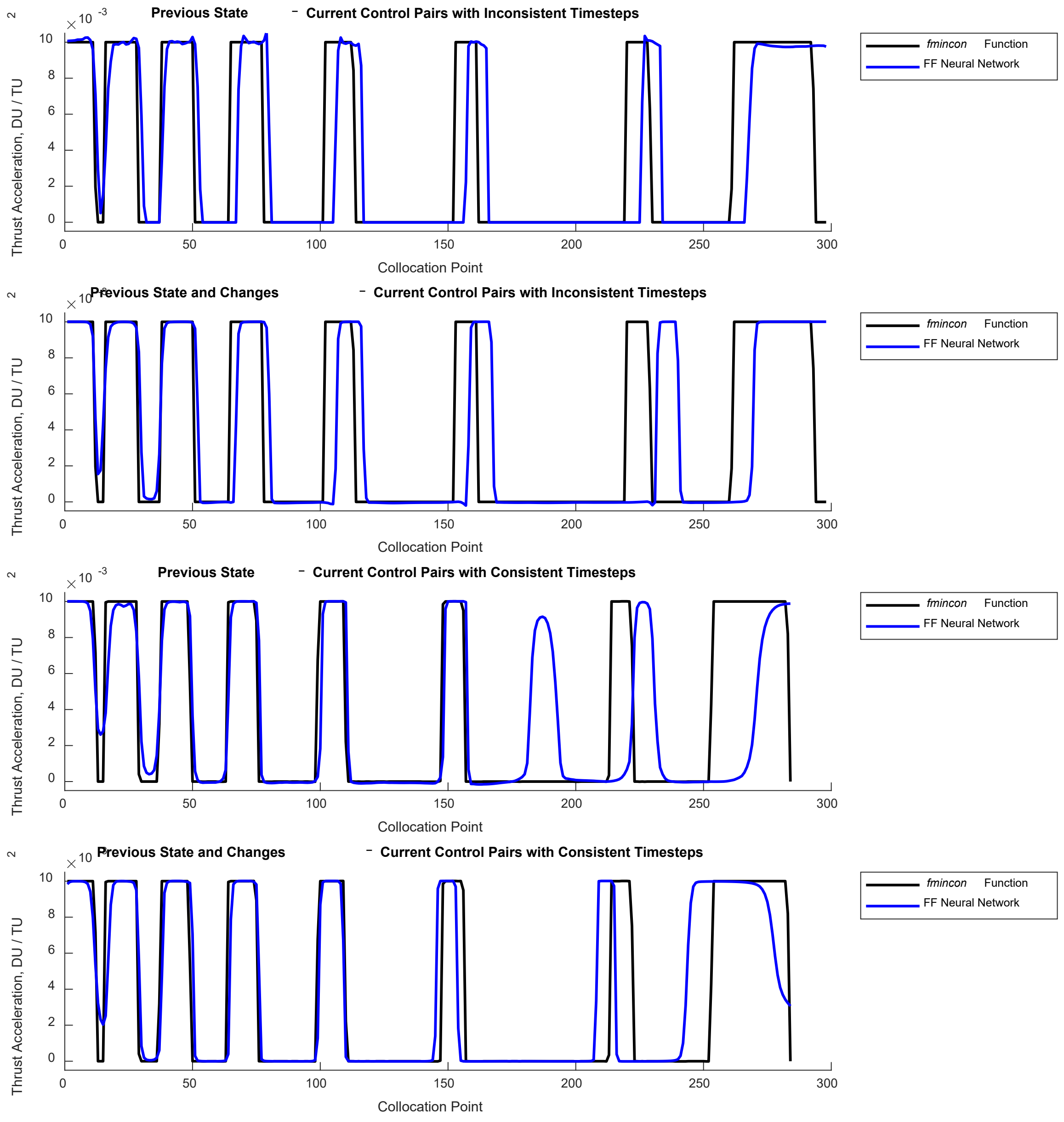


Figure 5.24: Constrained Fuel-Optimal Closed-Loop Feedforward Test Case Results (Thrust Accelerations)

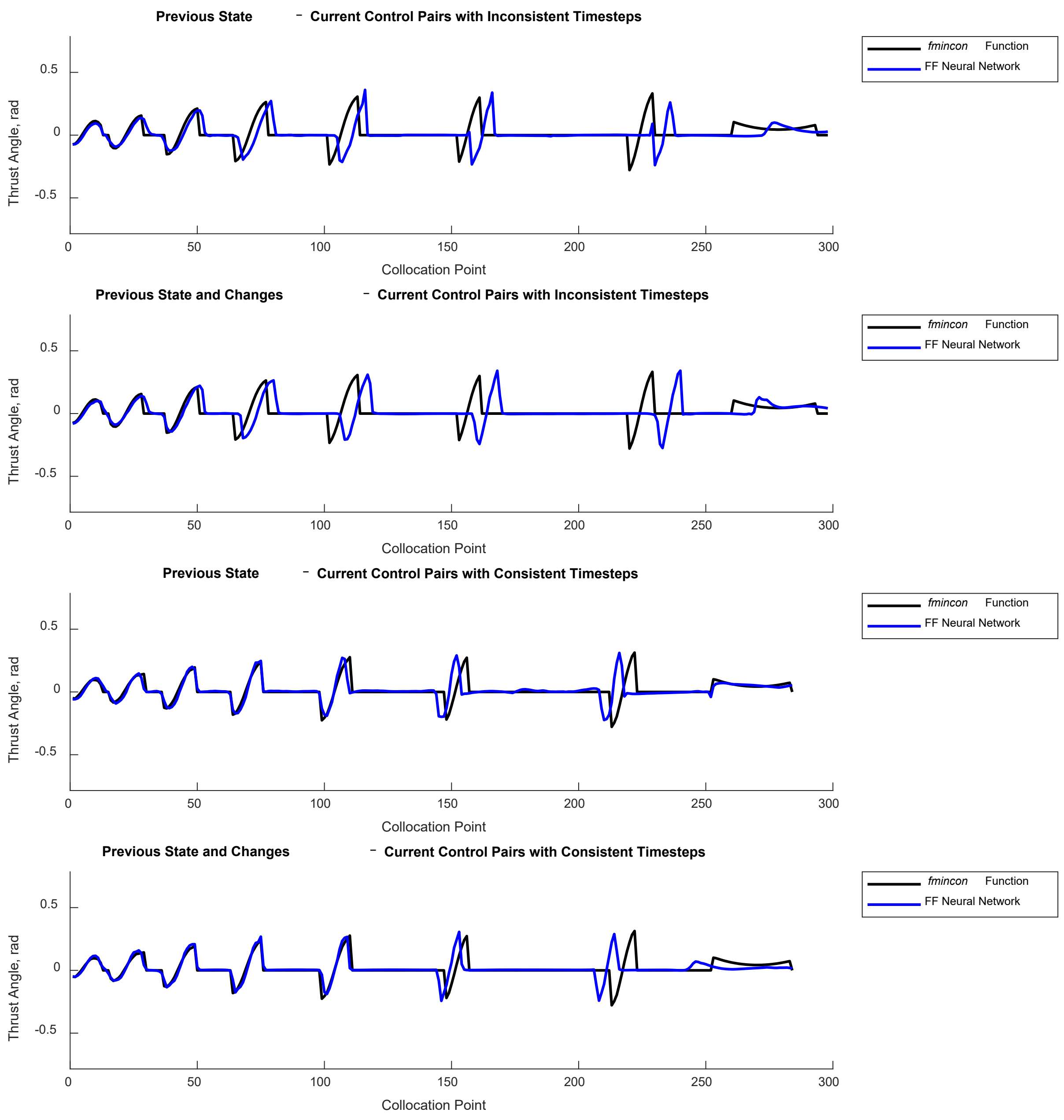


Figure 5.25: Constrained Fuel-Optimal Closed-Loop Feedforward Test Case Results (Thrust Angles)

Table XLVI: Summary of Constrained Fuel-Optimal Closed-Loop Feedforward Tuning and Training Results

Feedforward Neural Network	Tuning Time	Test Case RMSE	Training Time
Previous State – Current Control Pairs with Inconsistent Timesteps	2.54 hr	3.9761e-3 DU/TU^2 0.0982 rad	0.05 hr
Previous State and Changes – Current Control Pairs with Inconsistent Timesteps	1.58 hr	4.3707e-3 DU/TU^2 0.1121 rad	0.06 hr
Previous State – Current Control Pairs with Consistent Timesteps	2.04 hr	3.9794e-3 DU/TU^2 0.0846 rad	0.05 hr
Previous State and Changes – Current Control Pairs with Consistent Timesteps	1.70 hr	3.2679e-3 DU/TU^2 0.0777 rad	0.07 hr

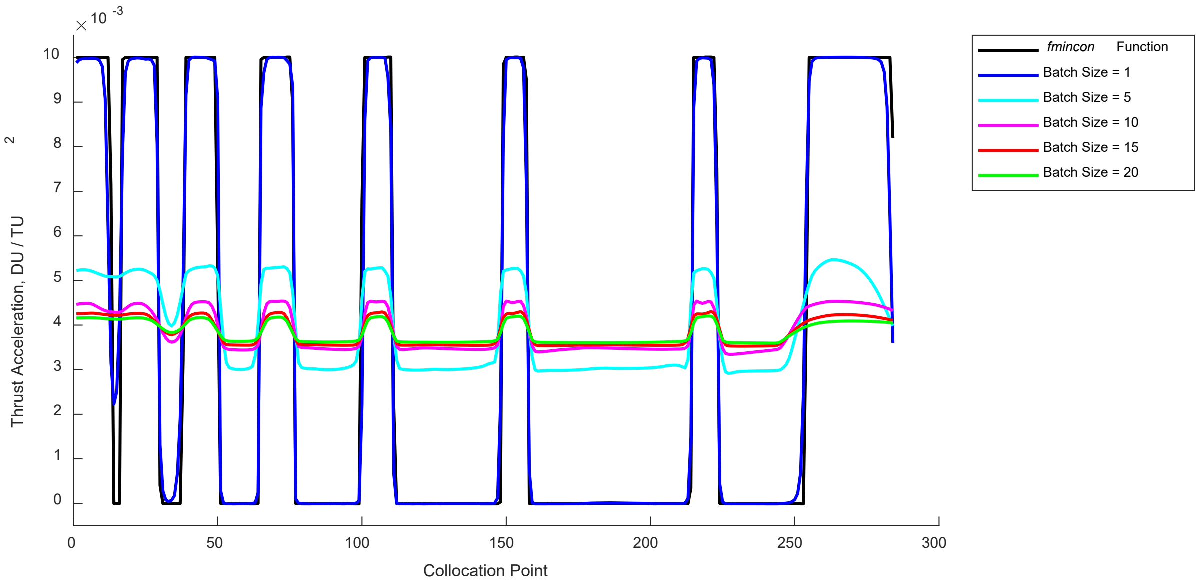
When considering the control predictions for the constrained fuel-optimal open-loop case, the LSTM neural networks with minimum RMSEs of 8.3816e-4 DU/TU^2 and 0.0224 radians (using consistent timesteps with the current states as inputs) performed on par (i.e., with less than 10 percent difference) with the feedforward neural networks with minimum RMSEs of 8.1203e-4 DU/TU^2 and 0.0219 radians (again, using consistent timesteps, but with the current states and the desired changes for those states as inputs). However, for the constrained fuel-optimal open-loop case, the LSTM neural networks had significantly longer average training times than the feedforward neural networks.

When considering the control predictions for the constrained fuel-optimal closed-loop case, the LSTM neural networks with minimum RMSEs of 2.7703e-3 DU/TU^2 and 0.0733 radians (using consistent timesteps with the previous states and the desired changes for those states as inputs) performed better than the feedforward neural networks with minimum RMSEs of 3.2679e-3 DU/TU^2 and 0.0777 radians (again, using consistent timesteps with the previous states and the desired changes for those states as inputs). However, as in the constrained fuel-optimal open-loop case, the LSTM neural networks had significantly longer average training times than the feedforward neural networks.

Overall, though the LSTM neural networks typically performed on par with or better than their feedforward equivalents, the significantly shorter average training times of the feedforward neural networks was a considerable advantage. Attempting to narrow the gap between the average training times for the LSTM neural networks and their feedforward equivalents, a batch size investigation was conducted to determine the effect of increasing batch size on average RMSE and training time for both the LSTM neural networks and the feedforward neural networks.

5.2.4 Batch Size Investigation

Overall, as in the time-optimal case, increasing the batch size for the feedforward neural networks significantly and negatively impacted the ability of the feedforward neural networks to learn. Figure 5.26 shows an example representative of the relationships between batch size and the average predicted control. The upper and lower plots of Figure 5.26 show the thrust acceleration and thrust angle predictions, respectively, by the feedforward neural network with the best constrained fuel-optimal open-loop performance (using consistent timesteps with the current states and the desired changes for those states as inputs). As seen in Figure 5.26, even when the batch sizes for the feedforward neural networks were only increased to five, the ability of the feedforward neural networks to learn the control features was significantly diminished. This trend was present for all feedforward neural networks tuned and trained on all constrained fuel-optimal datasets (both open-loop and closed-loop), regardless of the set of inputs.



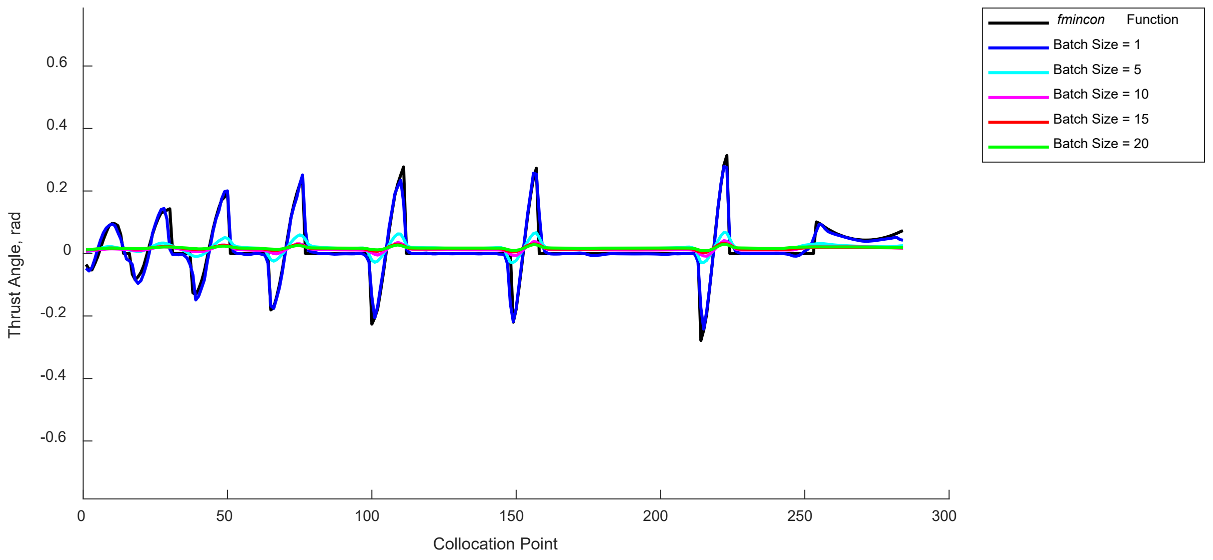


Figure 5.26: Example Representative of Constrained Fuel-Optimal Feedforward Batch Size Investigation

The results from the batch size investigation for the LSTM neural networks for the constrained fuel-optimal open-loop case are shown in Figure 5.27 through Figure 5.30. Figure 5.27 shows the relationships between batch size and the average predicted control RMSEs as well as the average training times. Figure 5.28, Figure 5.29, and Figure 5.30 compare those relationships (thrust acceleration, thrust angle, and training time, respectively) to the results from their feedforward equivalents with a batch size of one.

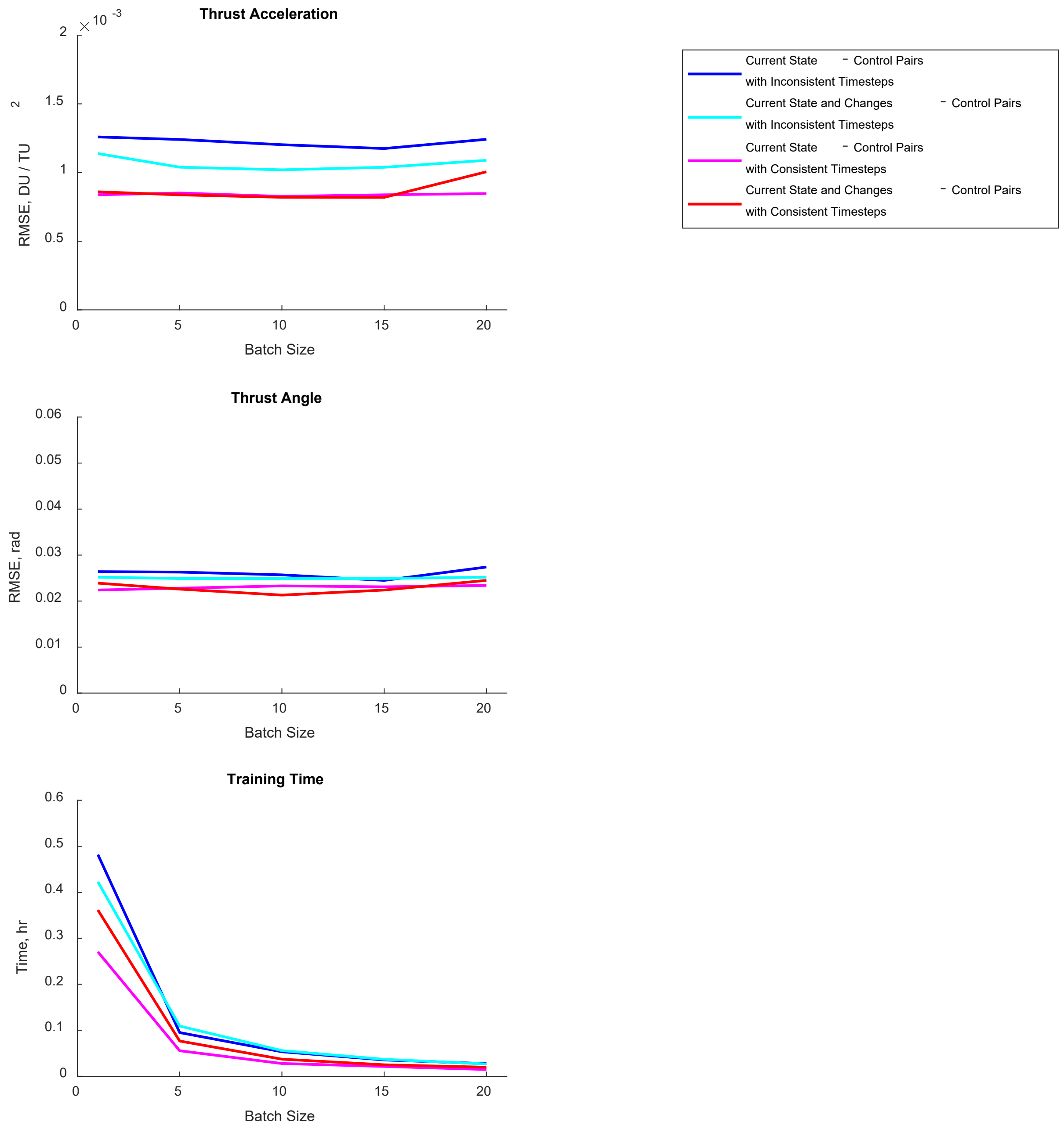


Figure 5.27: Constrained Fuel-Optimal Open-Loop LSTM Batch Size Investigation

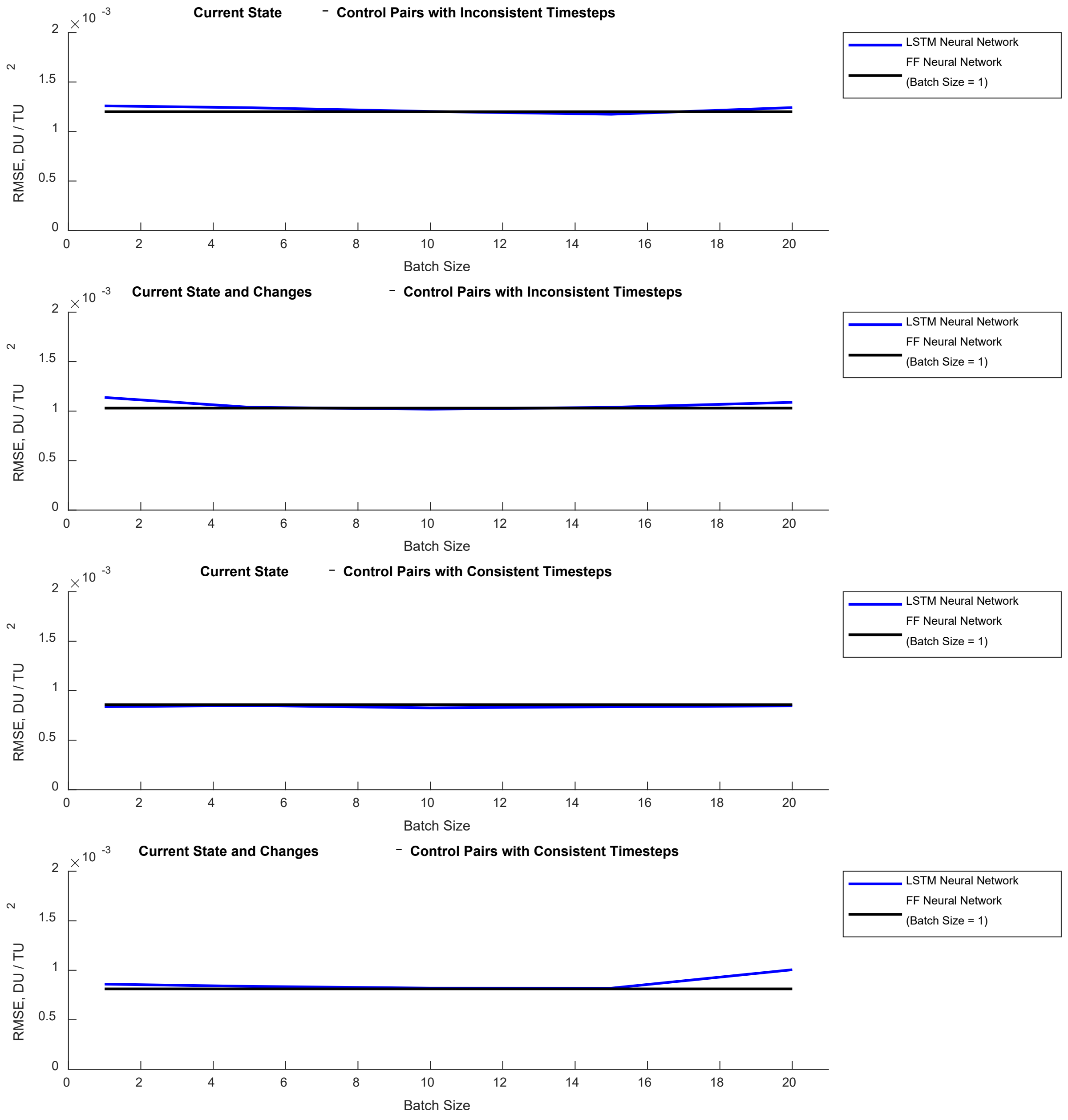


Figure 5.28: Constrained Fuel-Optimal Open-Loop LSTM Batch Size Investigation (Thrust Accelerations)

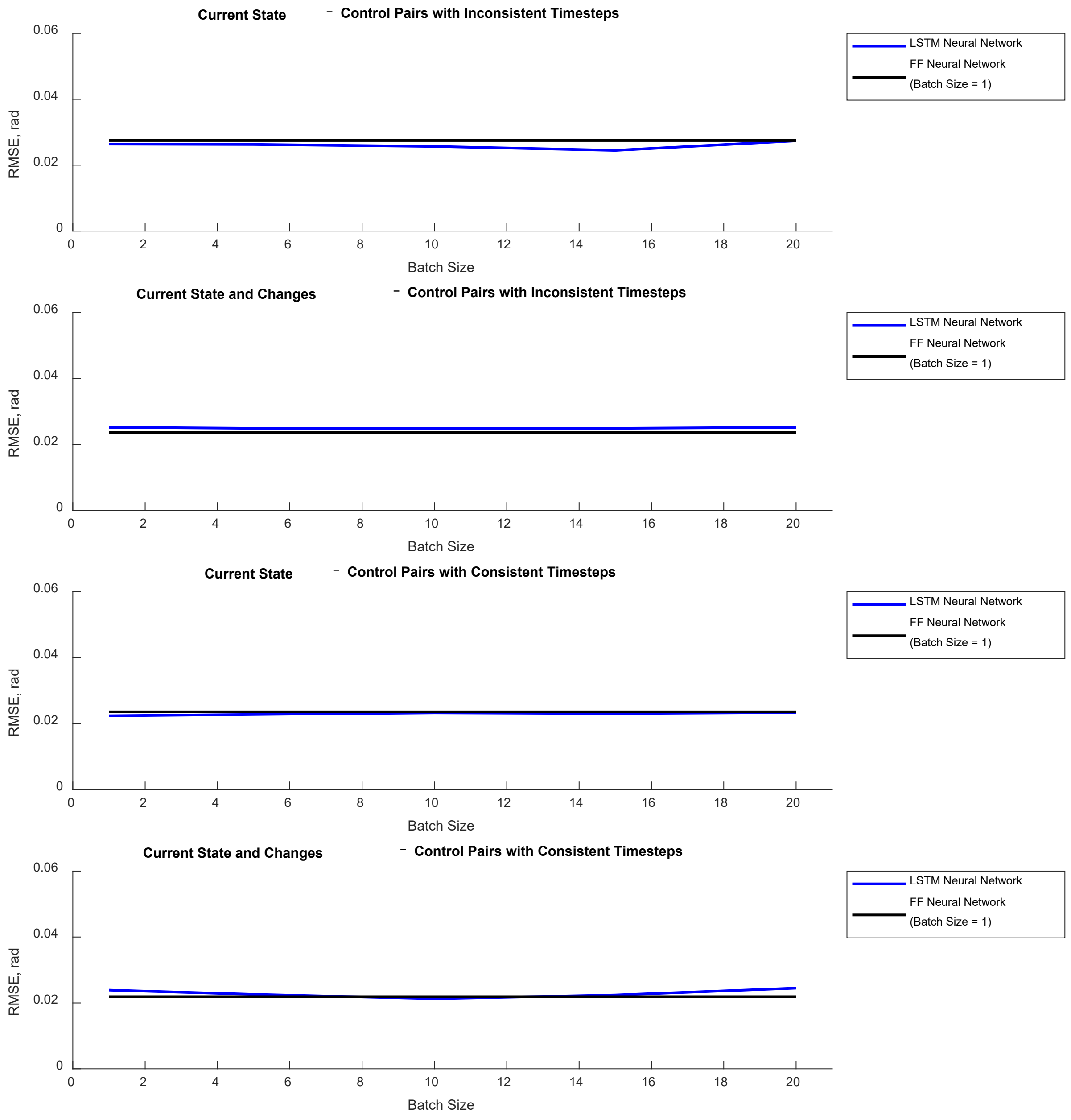


Figure 5.29: Constrained Fuel-Optimal Open-Loop LSTM Batch Size Investigation (Thrust Angles)

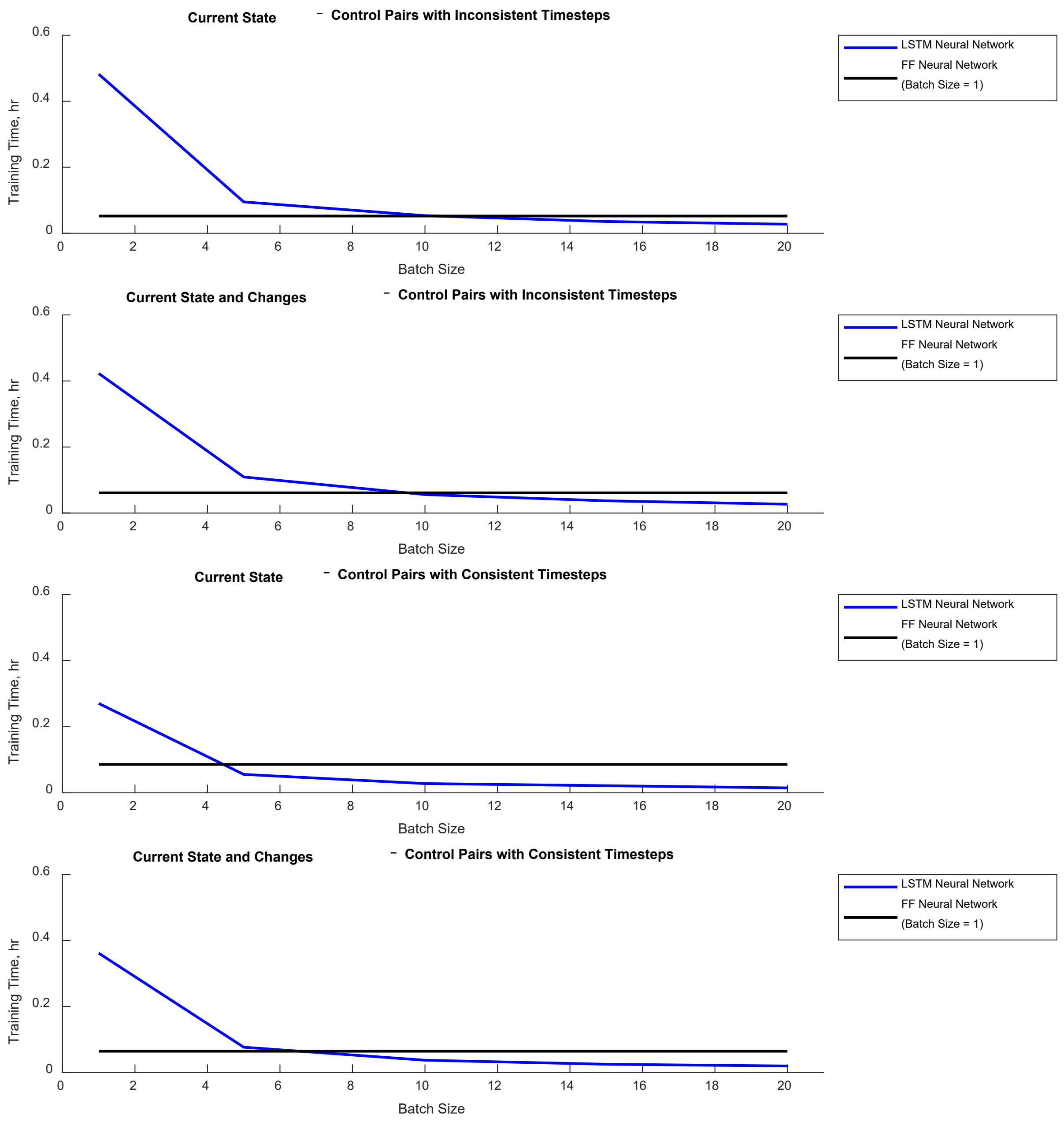


Figure 5.30: Constrained Fuel-Optimal Open-Loop LSTM Batch Size Investigation (Training Times)

As seen in these figures, increasing the batch size for the LSTM neural networks significantly improved the average training time with minimal impact on the average predicted control RMSE. Diminishing returns for the average training time were observed near a batch size of 20 and, after a batch size of 10, the average training times of the LSTM neural networks were consistently shorter than the feedforward neural networks. Regardless of batch size, the LSTM neural networks typically performed on par (i.e., with less than 10 percent difference) with the feedforward neural networks.

The results from the batch size investigation of the LSTM neural networks for the constrained fuel-optimal closed-loop case are shown in Figure 5.31 through Figure 5.34. Figure 5.31 shows the relationships between batch size and the average predicted control RMSEs as well as the average training times. Figure 5.32, Figure 5.33, and Figure 5.34 compare those relationships (thrust acceleration, thrust angle, and training time, respectively) to the results from their feedforward equivalents with a batch size of one. As in the constrained fuel-optimal open-loop case, increasing the batch size for the LSTM neural networks significantly improved the average training time with typically minimal impact on the average predicted control RMSE. Again, as in the constrained fuel-optimal open-loop case, diminishing returns for the average training time were observed near a batch size of 20 and, after a batch size of 10, the average training times of the LSTM neural networks were consistently shorter than the feedforward neural networks. Regardless of batch size, the LSTM neural networks typically performed on par with the feedforward neural networks.

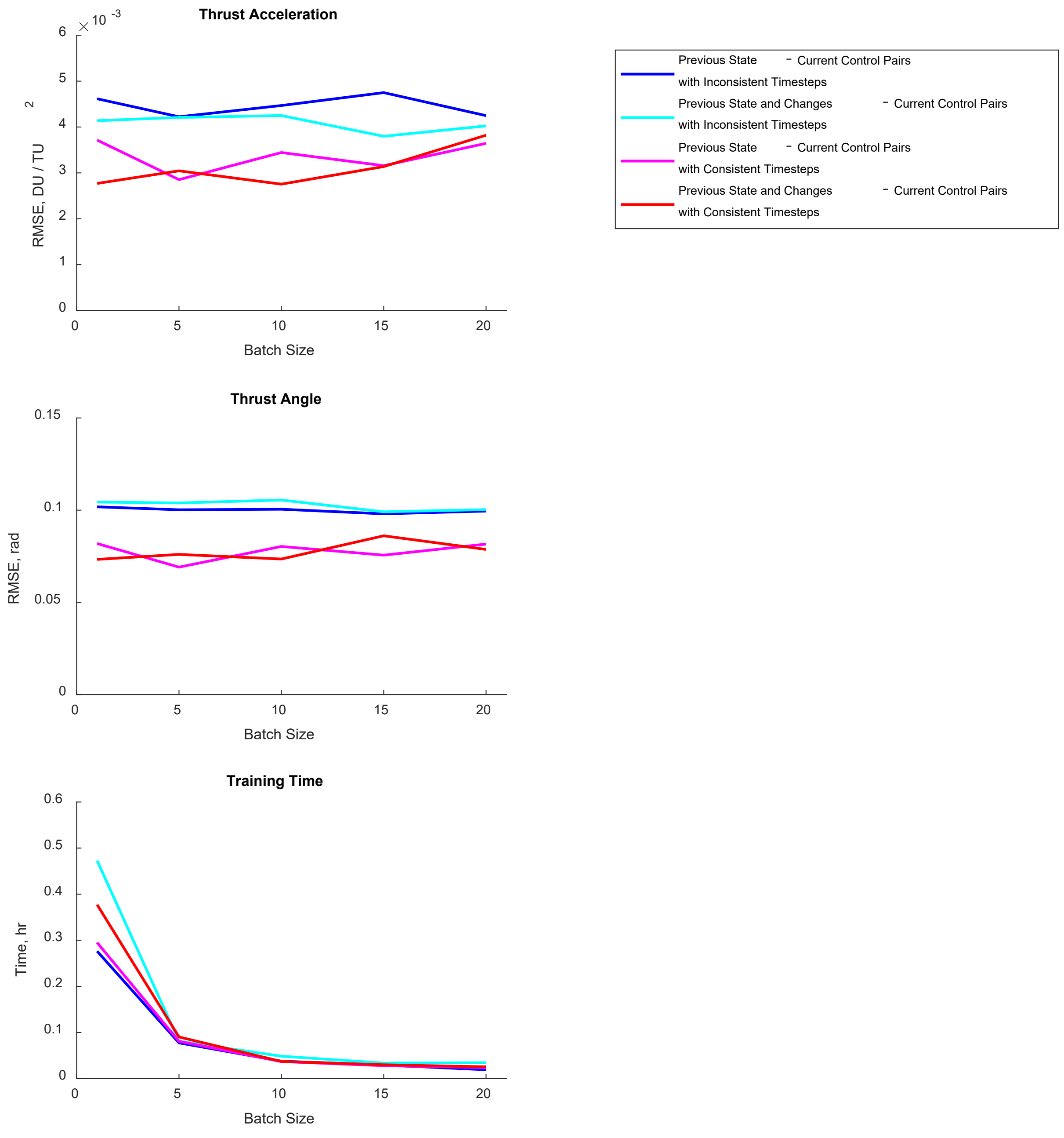


Figure 5.31: Constrained Fuel-Optimal Closed-Loop LSTM Batch Size Investigation

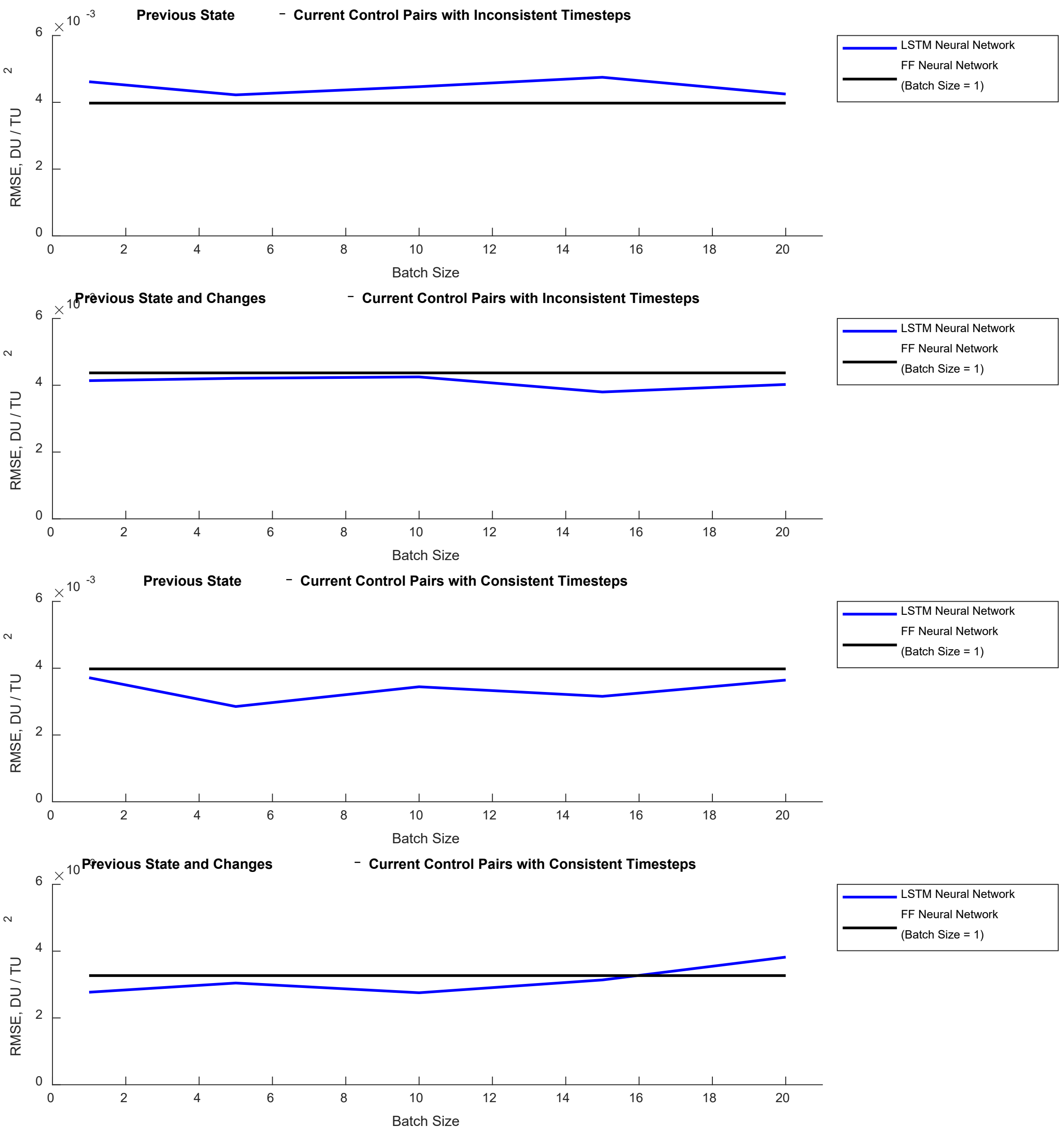


Figure 5.32: Constrained Fuel-Optimal Closed-Loop LSTM Batch Size Investigation (Thrust Accelerations)

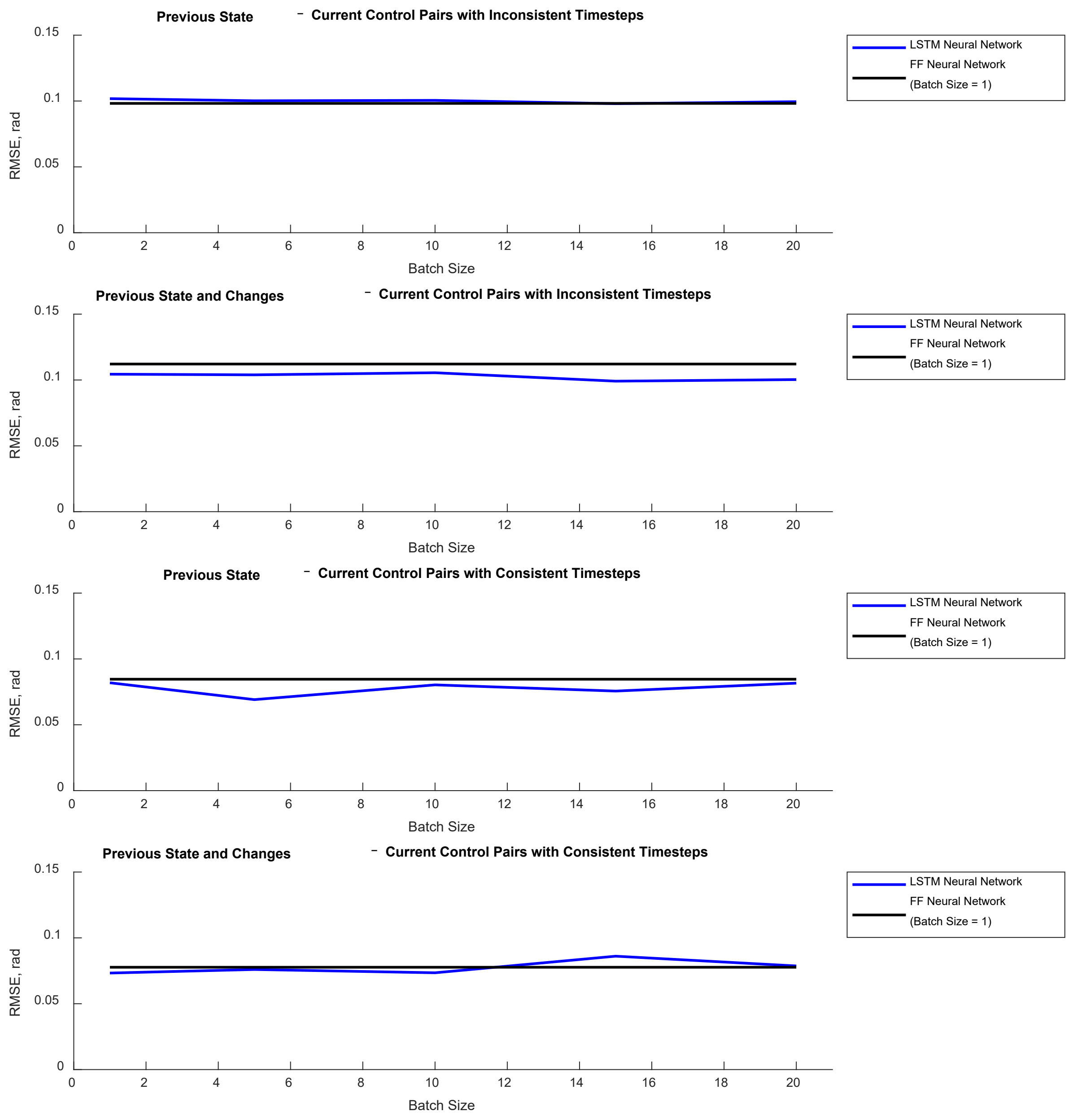


Figure 5.33: Constrained Fuel-Optimal Closed-Loop LSTM Batch Size Investigation (Thrust Angles)

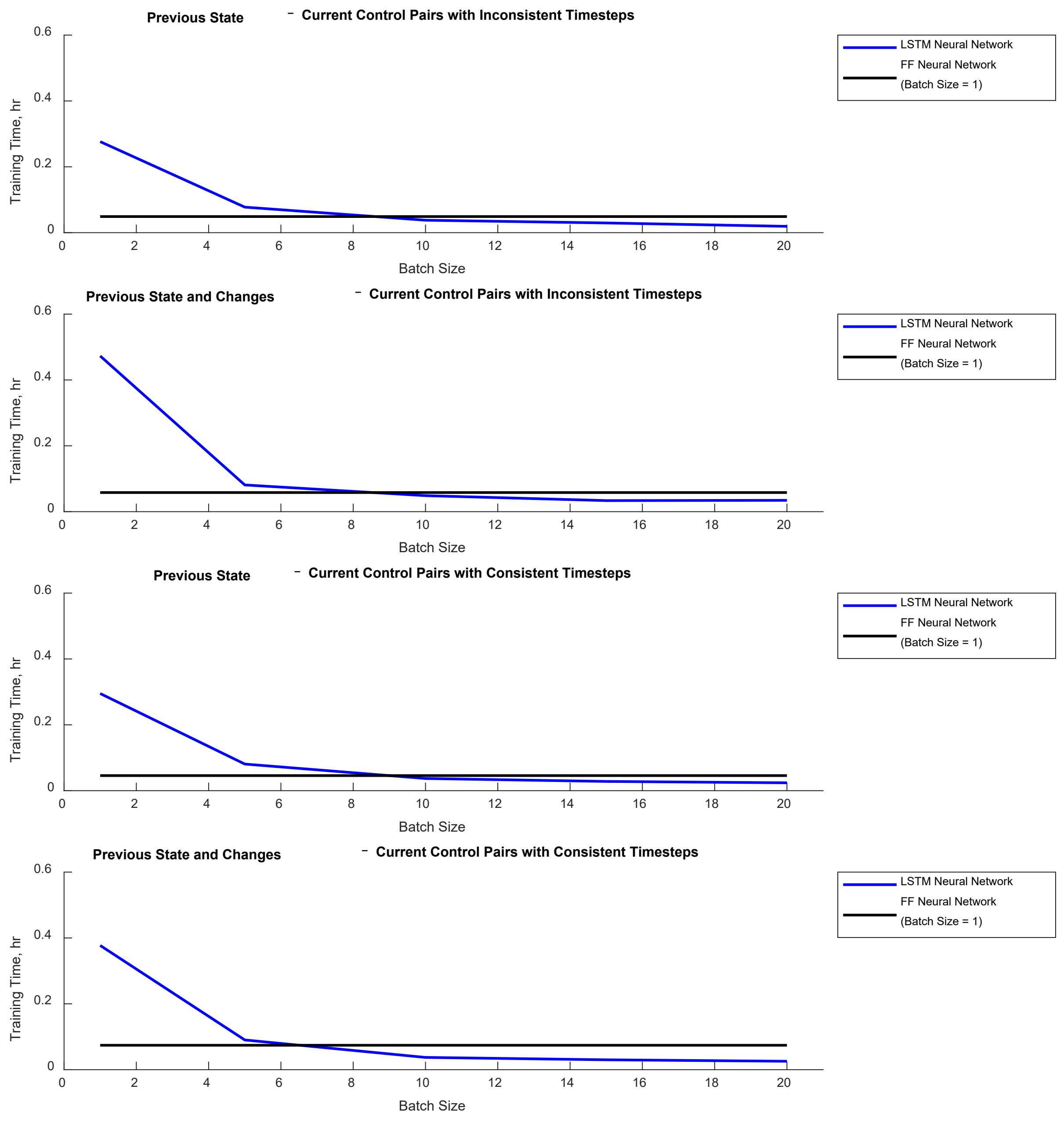


Figure 5.34: Constrained Fuel-Optimal Closed-Loop LSTM Batch Size Investigation (Training Times)

5.2.5 Final Comparison to Feedforward Architectures

When considering the thrust acceleration predictions for the constrained fuel-optimal open-loop case, the LSTM neural networks with a minimum RMSE of 8.4675e-4 DU/TU^2 (using a batch size of 20 and consistent timesteps with the current states as inputs) performed on par (i.e., with less than 10 percent difference) with the feedforward neural networks with a minimum RMSE of 8.1203e-4 DU/TU^2 (using a batch size of one and, again, consistent timesteps, but with the current states and the desired changes for those states as inputs). When considering the thrust angle predictions for the constrained fuel-optimal open-loop case, the LSTM neural networks with a minimum RMSE of 0.0234 radians (again, using a batch size of 20 and consistent timesteps with the current states as inputs) performed on par with the feedforward neural networks with a minimum RMSE of 0.0219 radians (again, using a batch size of one and consistent timesteps with the current states and the desired changes for those states as inputs). However, for the time-optimal open-loop case, the LSTM neural networks (53 seconds) had shorter average training times (by about 77 percent) than the feedforward neural networks (232 seconds).

When considering the thrust acceleration predictions for the constrained fuel-optimal closed-loop case, the LSTM neural networks with a minimum RMSE of 3.6441e-3 DU/TU^2 (using a batch size of 20 and consistent timesteps with the previous states as inputs) performed worse than the feedforward neural networks with a minimum RMSE of 3.2679e-3 DU/TU^2 (using a batch size of one and, again, consistent timesteps, but with the previous states and the desired changes for those states as inputs). When considering the thrust angle predictions for the constrained fuel-optimal closed-loop case, the LSTM neural networks with a minimum RMSE of 0.0787 radians (using a batch size of 20 and consistent timesteps with the previous states and the desired changes for those states as inputs as inputs) performed on par with the feedforward neural networks with a minimum RMSE of 0.0777 radians (again, using a batch size of one and consistent timesteps with the previous states and the desired changes for those states as inputs). However, for the constrained fuel-optimal closed-loop case, the LSTM neural networks (86 to 92 seconds) had shorter average training times (by about 66 to 68 percent) than the feedforward neural networks (267 seconds).

Overall, in consideration of the results for the constrained fuel-optimal case (both open-loop and closed-loop), the following conclusions were reached.

1. For the constrained fuel-optimal open-loop case, the LSTM neural networks performed on par with their feedforward equivalents in terms of average RMSE.
2. For the constrained fuel-optimal closed-loop case, the LSTM neural networks performed on par or worse than their feedforward equivalents in terms of average RMSE.
3. For both the constrained fuel-optimal open-loop case and the constrained fuel-optimal closed-loop case, the LSTM neural networks with an appropriate batch size performed (i.e., trained) faster than their feedforward equivalents.
4. For the constrained fuel-optimal open-loop case, using the current states as inputs improved the control predictions for the LSTM neural networks while using the current states and the desired changes for those states as inputs improved the control predictions for their feedforward equivalents.
5. For the constrained fuel-optimal closed-loop case, using the previous states and the desired changes for those states as inputs improved the control predictions for both the LSTM neural networks and their feedforward equivalents.

6. For both the constrained fuel-optimal open-loop case and the constrained fuel-optimal closed-loop case, consistent timesteps improved the control predictions for both the LSTM neural networks and their feedforward equivalents.

5.3 Additional Results

Further comparisons between the LSTM neural networks and their feedforward equivalents will be presented and discussed here.

5.3.1 Time-Optimal

Figure 5.35 and Figure 5.36 compare the integrated trajectories from the neural networks tuned and trained on the time-optimal datasets with inconsistent and consistent timesteps, respectively, for the open-loop case while Figure 5.37 and Table XLVII compare the integrated trajectories from the "best" LSTM neural network and the "best" feedforward neural network (i.e., thrust acceleration RMSE and thrust angle RMSE were considered in tandem, rather than separately). For the time-optimal open-loop case, when using inconsistent timesteps, the LSTM neural networks performed on par with feedforward neural networks. Both the LSTM neural networks and the feedforward neural networks had final radial velocities close to zero as well as final radii and final tangential velocities within 1 percent of the final values for the integrated trajectory from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. When using consistent timesteps, the LSTM neural networks, again, performed on par with the feedforward neural networks. Both the LSTM neural networks and the feedforward neural networks had final radial velocities close to zero as well as final radii and final tangential velocities within 2 percent of the final values for the integrated trajectory from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. Overall, for the time-optimal open-loop case, the "best" LSTM neural network (using a batch size of 20 and inconsistent timesteps with the current states and the desired changes for those states as inputs) performed on par with the "best" feedforward neural network (using a batch size of 1 and, again, inconsistent timesteps with the current states and the desired changes for those states as inputs). Both the "best" LSTM neural network and the "best" feedforward neural network had a final radial velocity close to zero as well as a final radius and a final tangential velocity within 1 percent of the final values for the integrated trajectory from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB.

Figure 5.38 and Figure 5.39 compare the integrated trajectories from the neural networks tuned and trained on the time-optimal datasets with inconsistent and consistent timesteps, respectively, for the closed-loop case while Figure 5.40 and Table XLVIII compare the integrated trajectories from the "best" LSTM neural network and the "best" feedforward neural network. For the time-optimal closed-loop case, when using inconsistent timesteps, the LSTM neural networks generally performed worse than the feedforward neural networks. Both the LSTM neural networks and the feedforward neural networks had final radial velocities close to zero. However, the LSTM neural networks had final radii and final tangential velocities within 5 percent of the final values for the integrated trajectory from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB while the final values for the feedforward neural networks were within 2 percent. When using consistent timesteps, the LSTM neural networks, again, generally performed worse than the feedforward neural networks. Both the LSTM neural networks and the feedforward neural networks had final radial velocities close to zero. However, the LSTM neural networks had final radii and final tangential velocities within 8 percent of the final values for the integrated trajectory from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB while the final values for the feedforward neural networks were within 3 percent. However, overall, for the time-optimal closed-loop case, the “best” LSTM neural network (using a batch size of 20 and inconsistent timesteps with the previous states and the desired changes for those states as inputs) performed on par with the “best” feedforward neural network (using a batch size of 1 and consistent timesteps with, again, the previous states and the desired changes for those states as inputs). Both the LSTM neural network and the feedforward neural network had a final radial velocity close to zero as well as a final radius and a final tangential velocity within 3 percent of the final values for the integrated trajectory from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB.

Table XLIX and Table L present the predicted fuel usage (in terms of the predicted thrust acceleration integrated using Simpson quadrature) for the time-optimal open-loop case and the time-optimal closed-loop case, respectively. Overall, for both the time-optimal open-loop case and the time-optimal closed-loop case, the predicted fuel usage for both the LSTM neural networks and the feedforward neural networks was within 1 percent of the optimized fuel usage from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. In fact, the percent difference of the predicted fuel usage for the LSTM neural networks was typically within hundredths of a percent of the predicted fuel usage for their feedforward equivalents. Also, consistent timesteps improved the predicted fuel usage for both the LSTM neural networks and the feedforward neural networks, which was expected as, discussed in Chapter 5.1, consistent timesteps improved the thrust acceleration prediction for both the time-optimal open-loop case and the time-optimal closed-loop case.

Table LI presents the transfer and convergence times as well as the constraint violations when the results from the “best” LSTM neural network (using a batch size of 20 and inconsistent timesteps with the current states and the desired changes for those states as inputs) and the “best” feedforward neural network (using a batch size of 1 and, again, inconsistent timesteps with the current states and the desired changes for those states as inputs) were used as initial guesses for the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. Overall, when using the results from the LSTM neural network, the *fmincon* function converged to a minimum transfer time within less than a half of a percent of the minimum transfer time from the time-optimal grid convergence study 63 percent faster. When using the results from the feedforward neural network, the *fmincon* function converged to a minimum transfer time within much less than a tenth of a percent of the minimum transfer time from the time-optimal grid convergence study 13 percent faster. These results demonstrate the potential both LSTM and feedforward neural networks have as initializers for traditional optimization techniques.

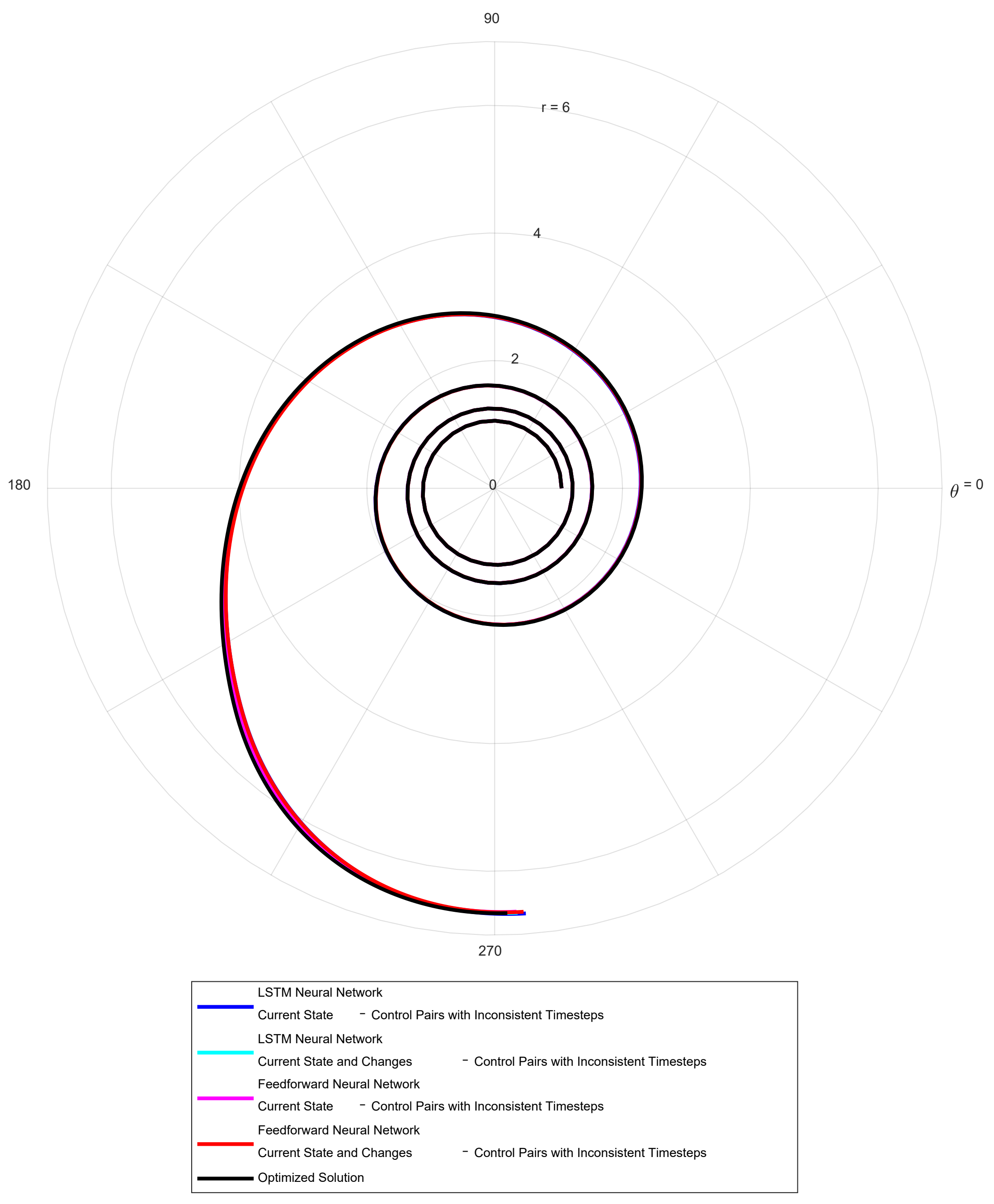


Figure 5.35: Comparison of Time-Optimal Open-Loop Trajectories using Inconsistent Timesteps

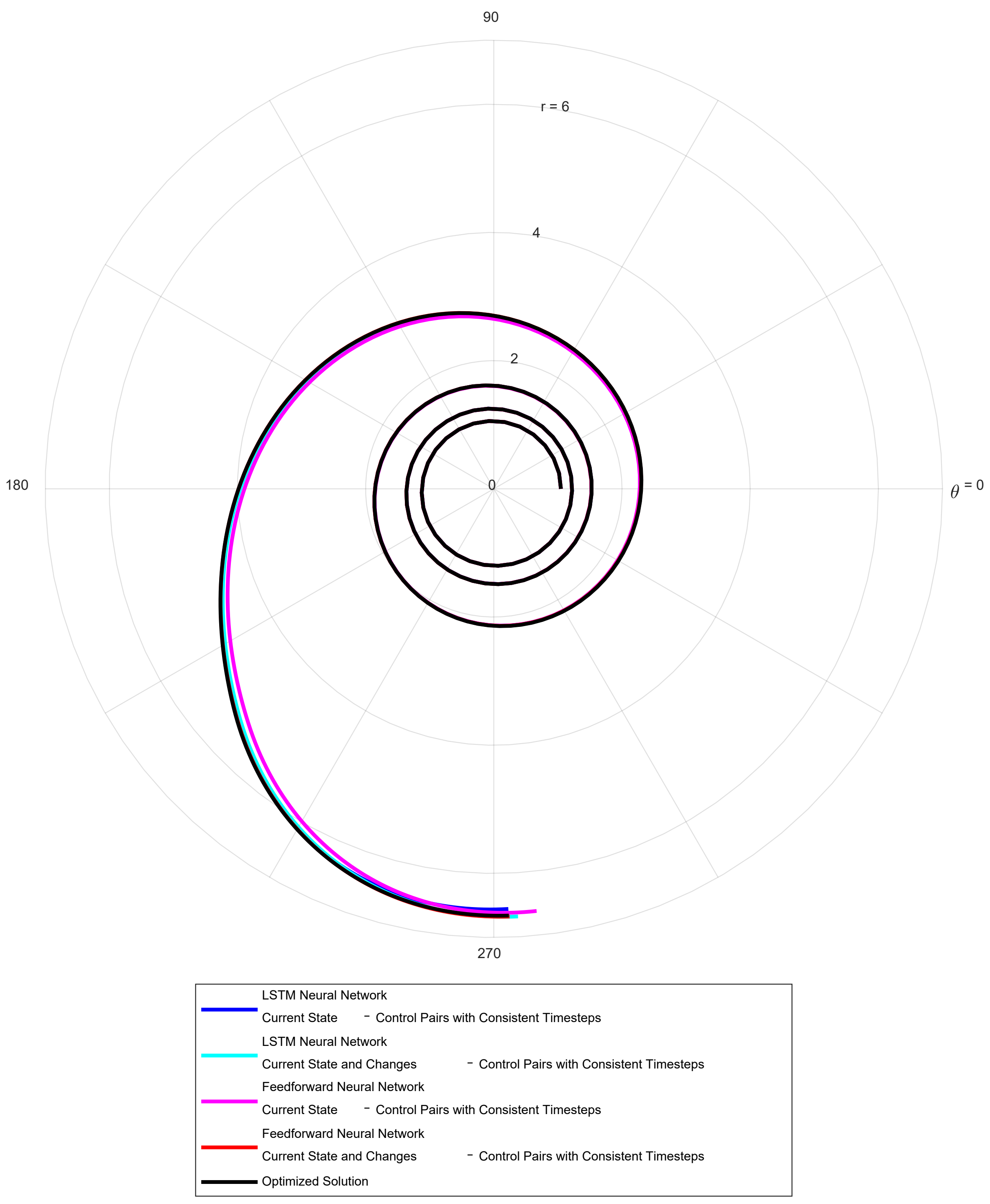


Figure 5.36: Comparison of Time-Optimal Open-Loop Trajectories using Consistent Timesteps

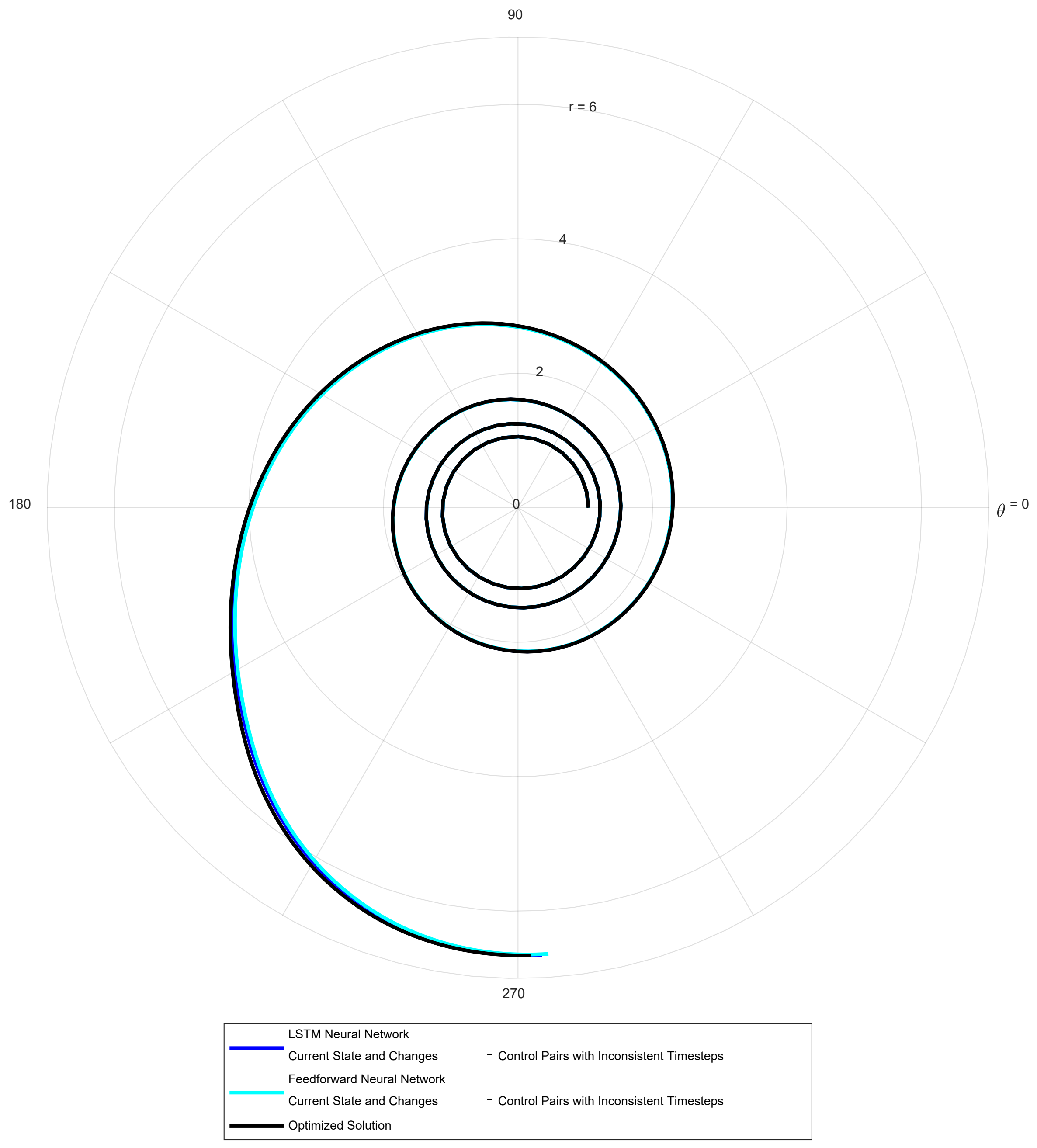


Figure 5.37: Comparison of "Best" Time-Optimal Open-Loop Trajectories

Table XLVII: Comparison of "Best" Time-Optimal Open-Loop Trajectories

Neural Network	Test Case RMSE	Training Time	Final Conditions $[r, \theta, v_r, v_t]$
LSTM - Current State & Changes with Inconsistent Timesteps	3.1079e-5 DU/TU^2 0.0154 rad	0.03 hr	[6.6595 DU, 23.62 rad, 0.0030 DU/TU, 0.3862 DU/TU] [0.01%, N/A, N/A, 0.15%]
FF - Current State and Changes with Inconsistent Timesteps	2.4688e-5 DU/TU^2 0.0158 rad	0.06 hr	[6.6495 DU, 23.63 rad, 0.0016 DU/TU, 0.3855 DU/TU] [0.15%, N/A, N/A, 0.02%]

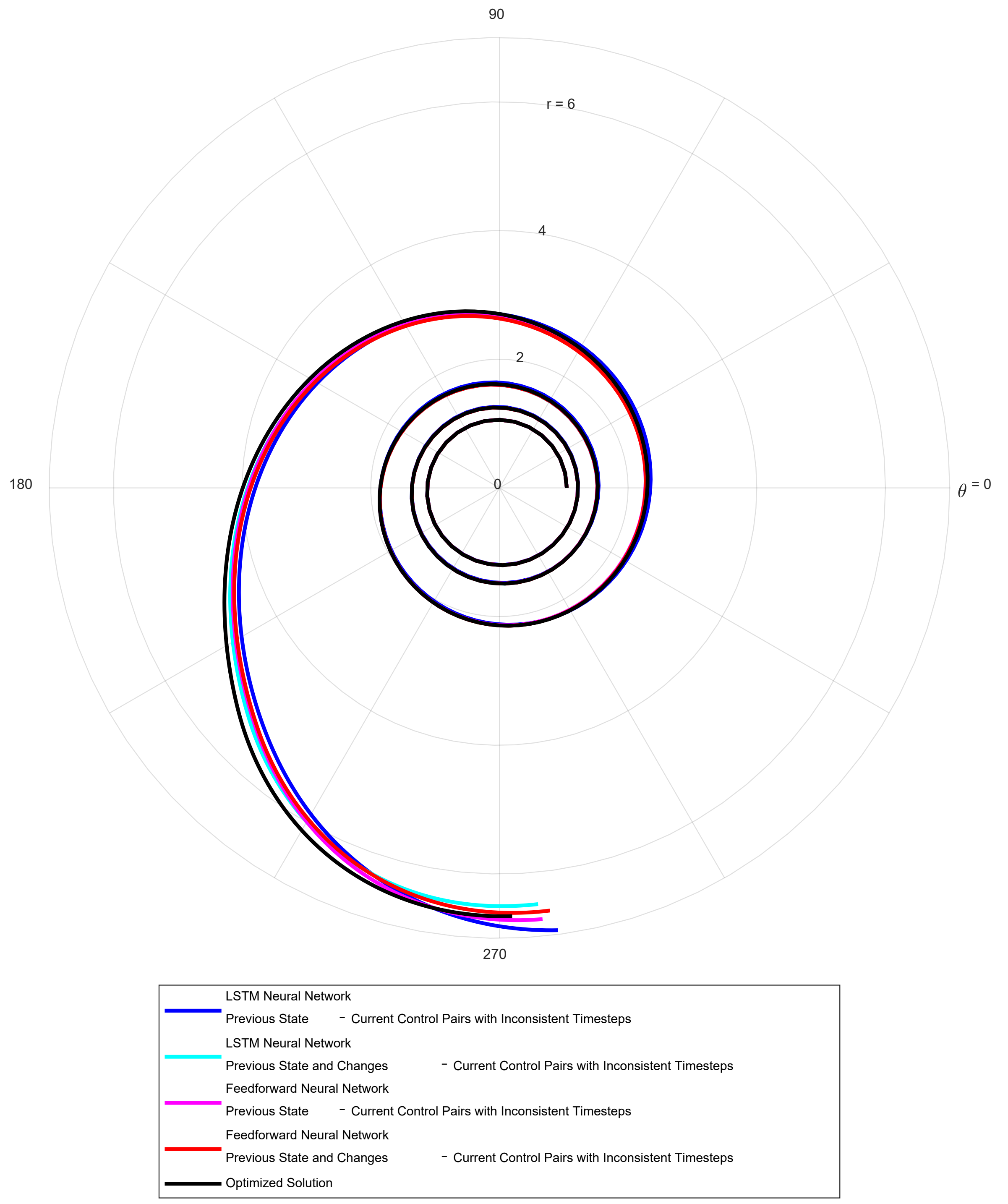


Figure 5.38: Comparison of Time-Optimal Closed-Loop Trajectories using Inconsistent Timesteps

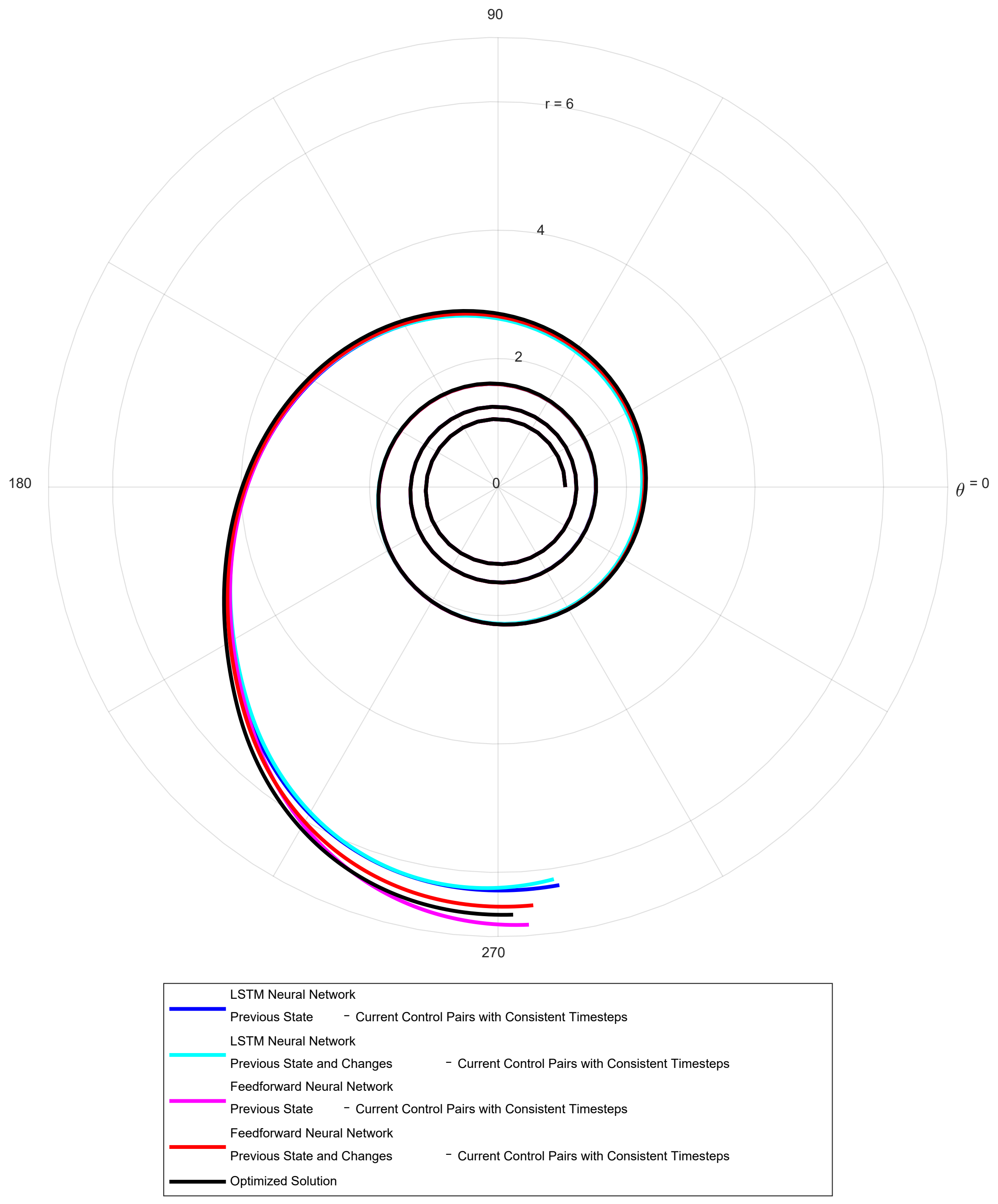


Figure 5.39: Comparison of Time-Optimal Closed-Loop Trajectories using Consistent Timesteps

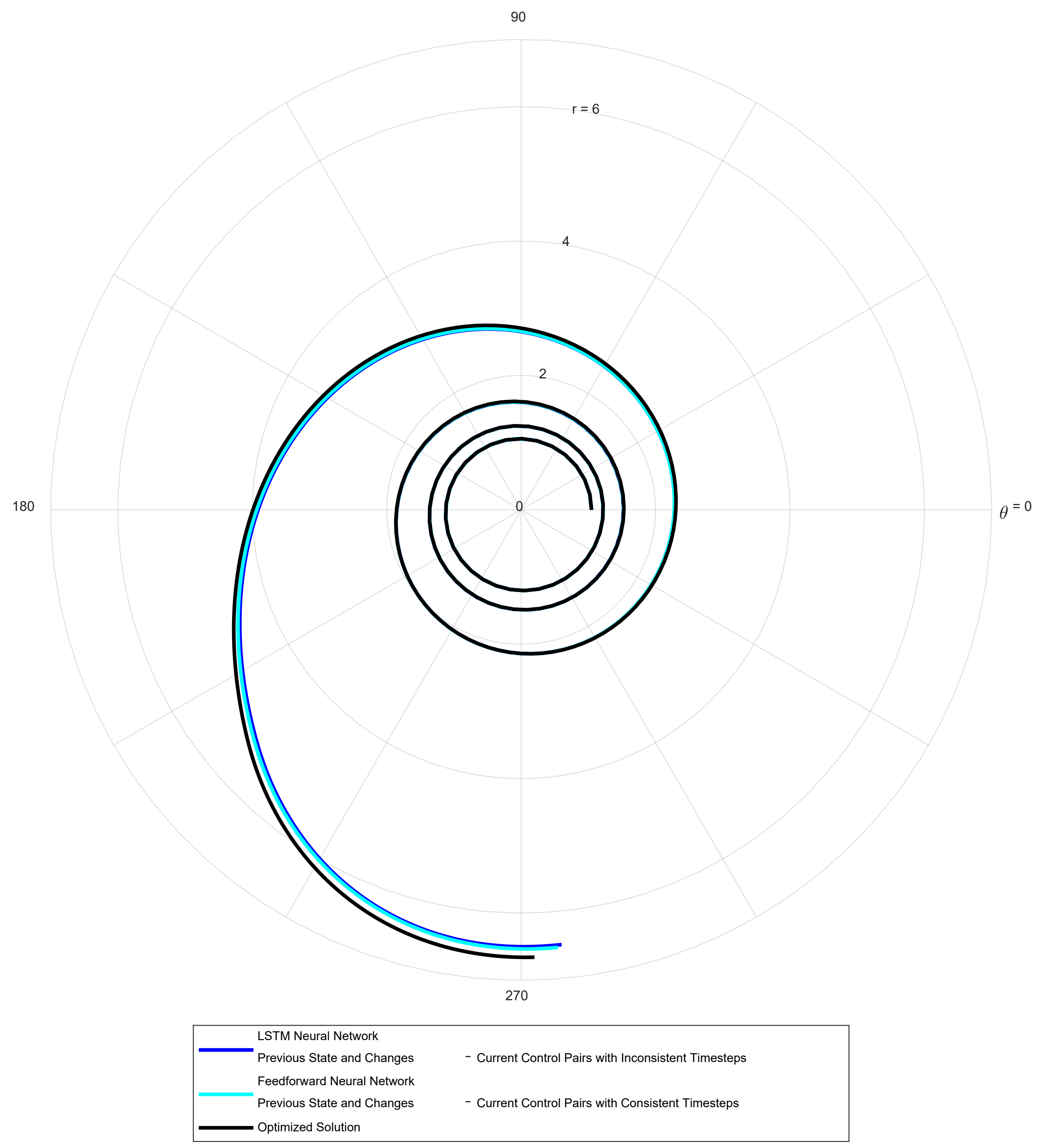


Figure 5.40: Comparison of "Best" Time-Optimal Closed-Loop Trajectories

Table XLVIII: Comparison of "Best" Time-Optimal Closed-Loop Trajectories

Neural Network	Test Case RMSE	Training Time	Final Conditions $[r, \theta, v_r, v_t]$
LSTM - Previous State & Changes with Inconsistent Timesteps	3.0157e-5 DU/TU^2 0.0696 rad	0.03 hr	[6.4964 DU, 23.65 rad, 0.0064 DU/TU, 0.3934 DU/TU] [2.44%, N/A, N/A, 2.02%]
FF - Previous State and Changes with Consistent Timesteps	2.3940e-5 DU/TU^2 0.0743 rad	0.04 hr	[6.5338 DU, 23.65 rad, 0.0029 DU/TU, 0.3933 DU/TU] [1.88%, N/A, N/A, 2.00%]

Table XLIX: Comparison of Time-Optimal Open-Loop Fuel Usage

Neural Network		Optimized Fuel Usage (DU/TU^2)	Predicted Fuel Usage (DU/TU^2)	Percent Difference
LSTM	Current State with Inconsistent Timesteps	0.7138	0.7122	0.22%
Feedforward			0.7123	0.21%
LSTM	Current State and Changes with Inconsistent Timesteps		0.7124	0.20%
Feedforward			0.7124	0.20%
LSTM	Current State with Consistent Timesteps	0.7136	0.7139	0.06%
Feedforward			0.7132	0.06%
LSTM	Current State and Changes with Consistent Timesteps		0.7130	0.08%
Feedforward			0.7127	0.13%

Table L: Comparison of Time-Optimal Closed-Loop Fuel Usage

Neural Network		Optimized Fuel Usage (DU/TU^2)	Predicted Fuel Usage (DU/TU^2)	Percent Difference
LSTM	Previous State with Inconsistent Timesteps	0.7138	0.7131	0.10%
Feedforward			0.7124	0.20%
LSTM	Previous State and Changes with Inconsistent Timesteps		0.7125	0.18%
Feedforward			0.7126	0.17%
LSTM	Previous State with Consistent Timesteps	0.7136	0.7131	0.07%
Feedforward			0.7130	0.08%
LSTM	Previous State and Changes with Consistent Timesteps		0.7126	0.14%
Feedforward			0.7130	0.08%

Table LI: Comparison as Initial Guesses (Time-Optimal)

Initial Guess	Transfer Time (TU)	Convergence Time (hr)	Constraint Violation
LSTM Neural Network	71.6726	0.18	5.3041e-9
Feedforward Neural Network	71.3795	0.41	2.1260e-12

5.3.2 Constrained Fuel-Optimal

Figure 5.41 and Figure 5.42 compare the integrated trajectories from the neural networks tuned and trained on the constrained fuel-optimal datasets with inconsistent and consistent timesteps, respectively, for the open-loop case while Figure 5.43 and Table LII compare the integrated trajectories from the "best" LSTM neural network and the "best" feedforward neural network (i.e., thrust acceleration RMSE and thrust angle RMSE were considered in tandem, rather than separately). For the constrained fuel-optimal open-loop case, when using inconsistent timesteps, the LSTM neural networks generally performed worse than the feedforward neural networks. Both the LSTM neural networks and the feedforward neural networks had final radial velocities close to zero. However, the LSTM neural networks had final radii and final tangential velocities within 14 percent of the final values for the integrated trajectory from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB while the final values for the feedforward neural networks were within 7 percent. When using consistent timesteps, the LSTM neural networks performed on par with the feedforward neural networks. Both the LSTM neural networks and the feedforward neural networks had final radial velocities close to zero as well as final radii and final tangential velocities within 8 percent of the final values for the integrated trajectory from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. Overall, for the constrained fuel-optimal open-loop case, the "best" LSTM neural network (using a batch size of 20 and consistent timesteps with the current states as inputs) performed on par with the "best" feedforward neural network (using a batch size of 1 and, again, consistent timesteps, but with the current states and the desired changes for those states as inputs). Both the "best" LSTM neural network and the "best" feedforward neural network had a final radial velocity close to zero as well as a final radius and a final tangential velocity within 8 percent of the final values for the integrated trajectory from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB.

Figure 5.44 and Figure 5.45 compare the integrated trajectories from the neural networks tuned and trained on the constrained fuel-optimal datasets with inconsistent and consistent timesteps, respectively, for the closed-loop case while Figure 5.46 and Table LIII compare the integrated trajectories from the "best" LSTM neural network and the "best" feedforward neural network. For the constrained fuel-optimal closed-loop case, when using inconsistent timesteps, both the LSTM neural networks and the feedforward neural networks performed poorly with, typically, non-zero final radial velocities and final radii and final tangential velocities significantly different from the final values for the integrated trajectory from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. This was expected as, discussed in Chapter 5.2, closing the loop for the constrained fuel-optimal case introduced instability and, subsequently, error into the training and inference process. However, slightly better performance in terms of appearance and average RMSE was observed from both the LSTM neural networks and the feedforward neural networks tuned and trained on the constrained fuel-optimal dataset with consistent timesteps. However, even when using consistent timesteps, both the LSTM neural networks and the feedforward neural networks performed poorly with, again, typically non-zero final radial velocities and final radii and final tangential velocities significantly different from the final values for the integrated trajectory from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. Overall, for the constrained fuel-optimal closed-loop case, the “best” LSTM neural network (using a batch size of 20 and consistent timesteps with the previous states and the desired changes for those states as inputs) performed worse than the “best” feedforward neural network (using a batch size of 1 and, again, consistent timesteps with the previous states and the desired changes for those states as inputs).

Table LIV and Table LV present the predicted fuel usage (in terms of the predicted thrust acceleration integrated using Simpson quadrature) for the constrained fuel-optimal open-loop case and the constrained fuel-optimal closed-loop case, respectively. Overall, for the constrained fuel-optimal open-loop case, the predicted fuel usage for both the LSTM neural networks and their feedforward equivalents was within 2 percent of the optimized fuel usage from the *fmincon* function. As was somewhat expected, the predicted fuel usage for both the LSTM neural networks and their feedforward equivalents in the constrained fuel-optimal closed-loop case was worse (i.e., greater than 2 percent of the optimized fuel usage from the *fmincon* function) than in the constrained fuel-optimal open-loop case.

Table LVI presents the thrust accelerations and convergence times as well as the constraint violations when the results from the "best" LSTM neural network (using a batch size of 20 and consistent timesteps with the current states as inputs) and the "best" feedforward neural network (using a batch size of 1 and, again, consistent timesteps, but with the current states and the desired changes for those states as inputs) were used as initial guesses for the *fmincon* function. Overall, when using the results from the LSTM neural network, the *fmincon* function converged to a minimum thrust acceleration within less than a tenth of a percent of the minimum thrust acceleration from the constrained fuel-optimal grid convergence study 9 percent faster. When using the results from the feedforward neural network, the *fmincon* function converged to a minimum thrust acceleration within much less than a tenth of a percent of the minimum thrust acceleration from the constrained fuel-optimal grid convergence study 70 percent faster. Again, these results demonstrate the potential both LSTM and feedforward neural networks have as initializers for traditional optimization techniques.

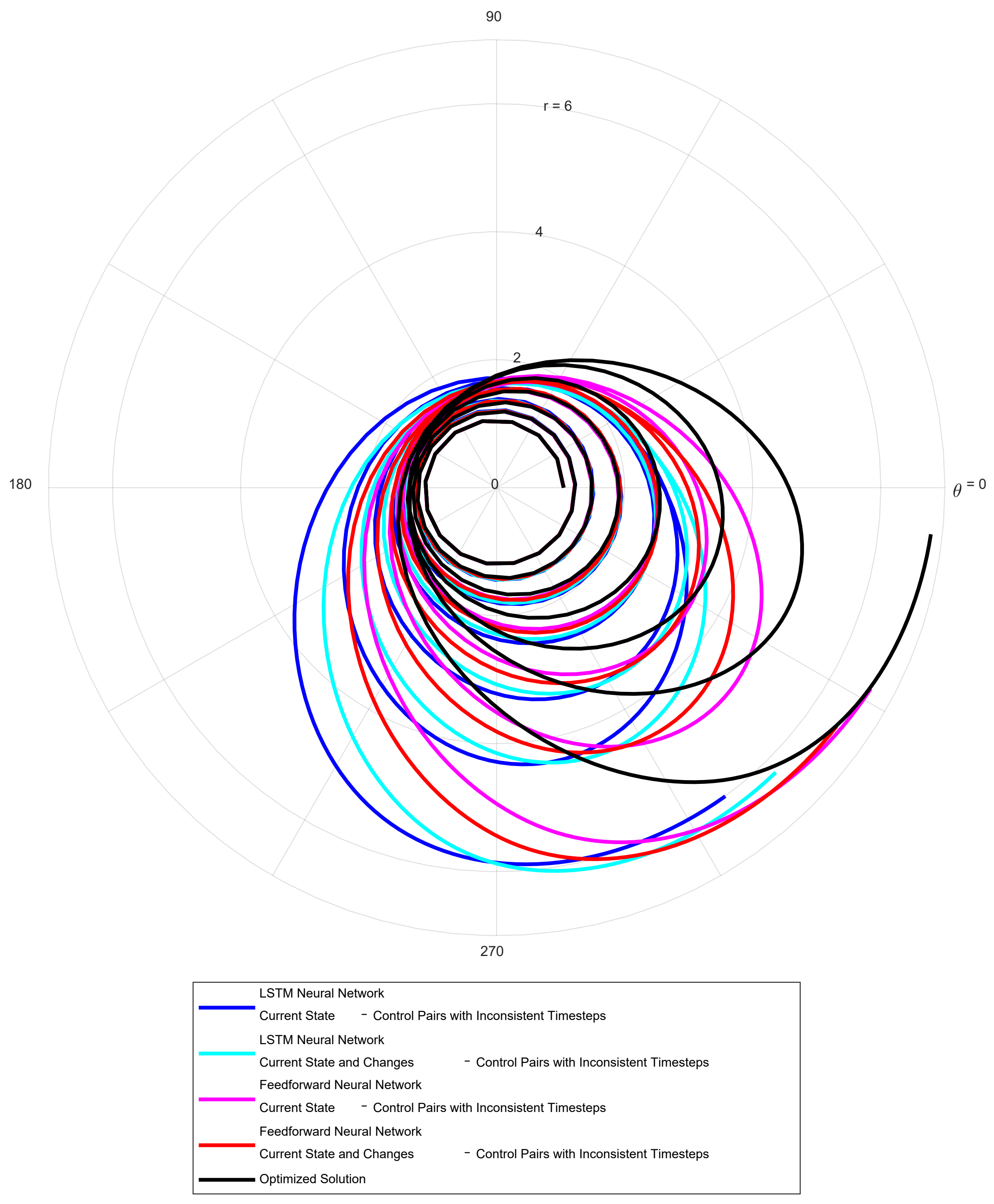


Figure 5.41: Comparison of Constrained Fuel-Optimal Open-Loop Trajectories using Inconsistent Timesteps

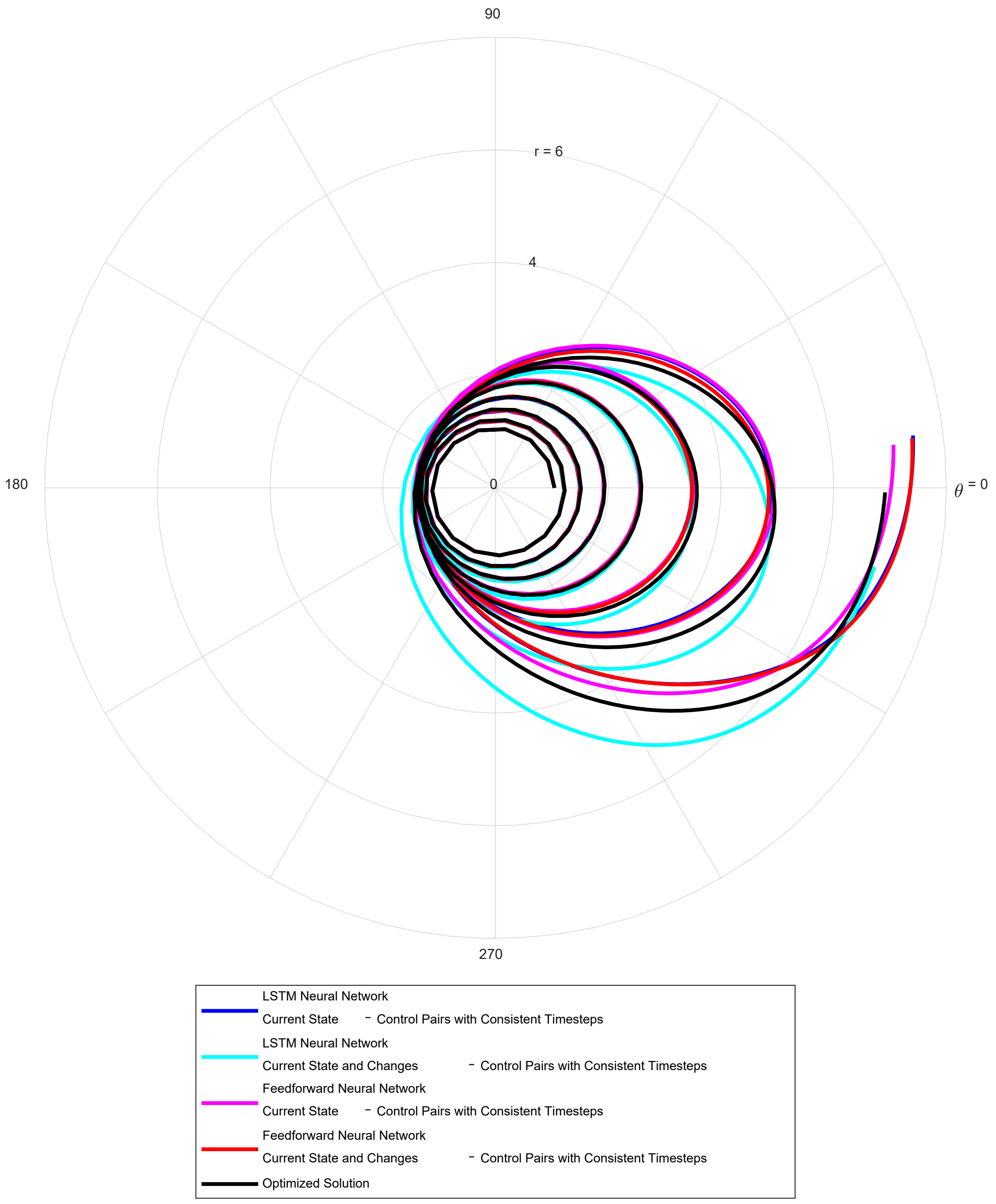


Figure 5.42: Comparison of Constrained Fuel-Optimal Open-Loop Trajectories using Consistent Timesteps

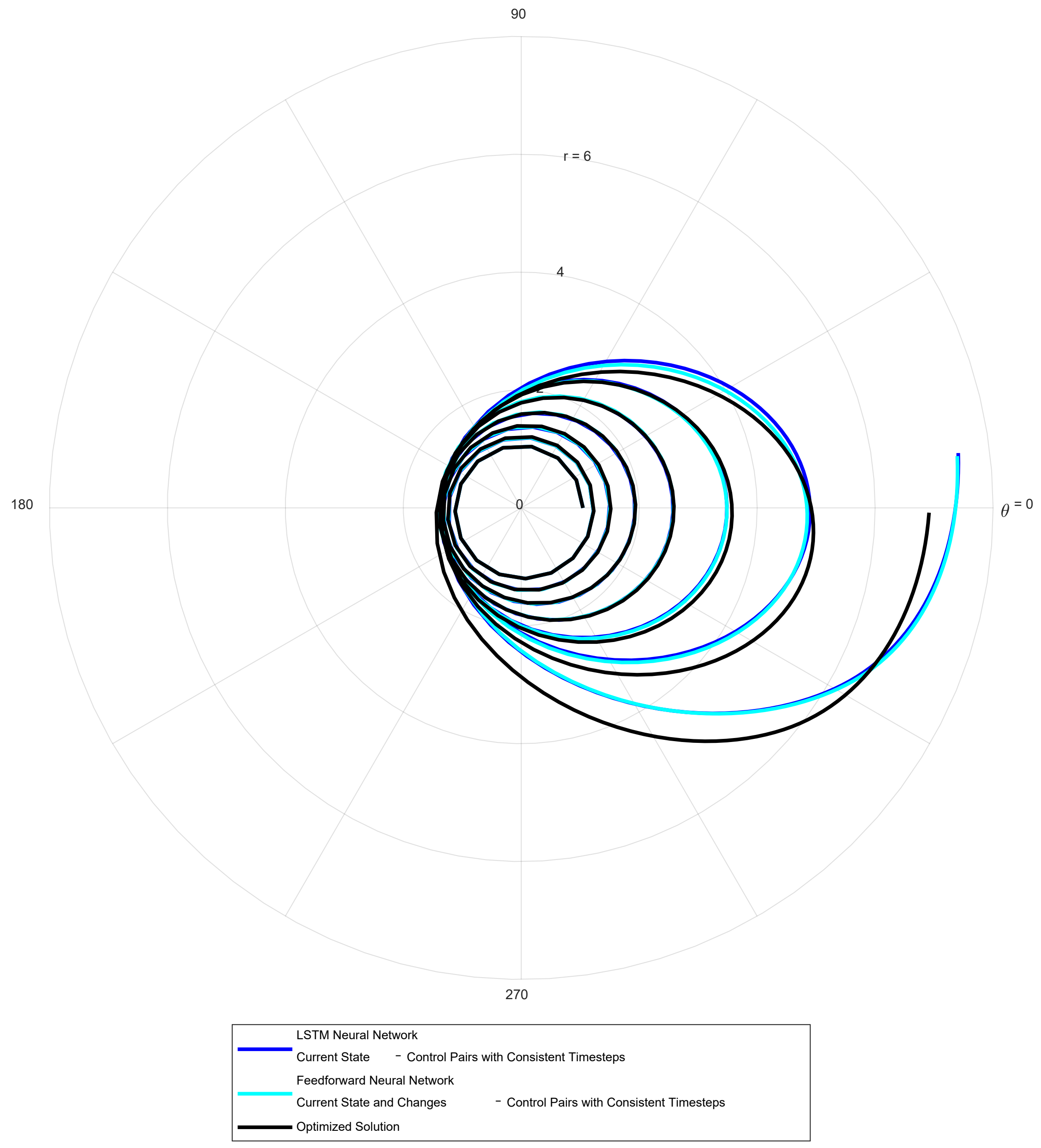
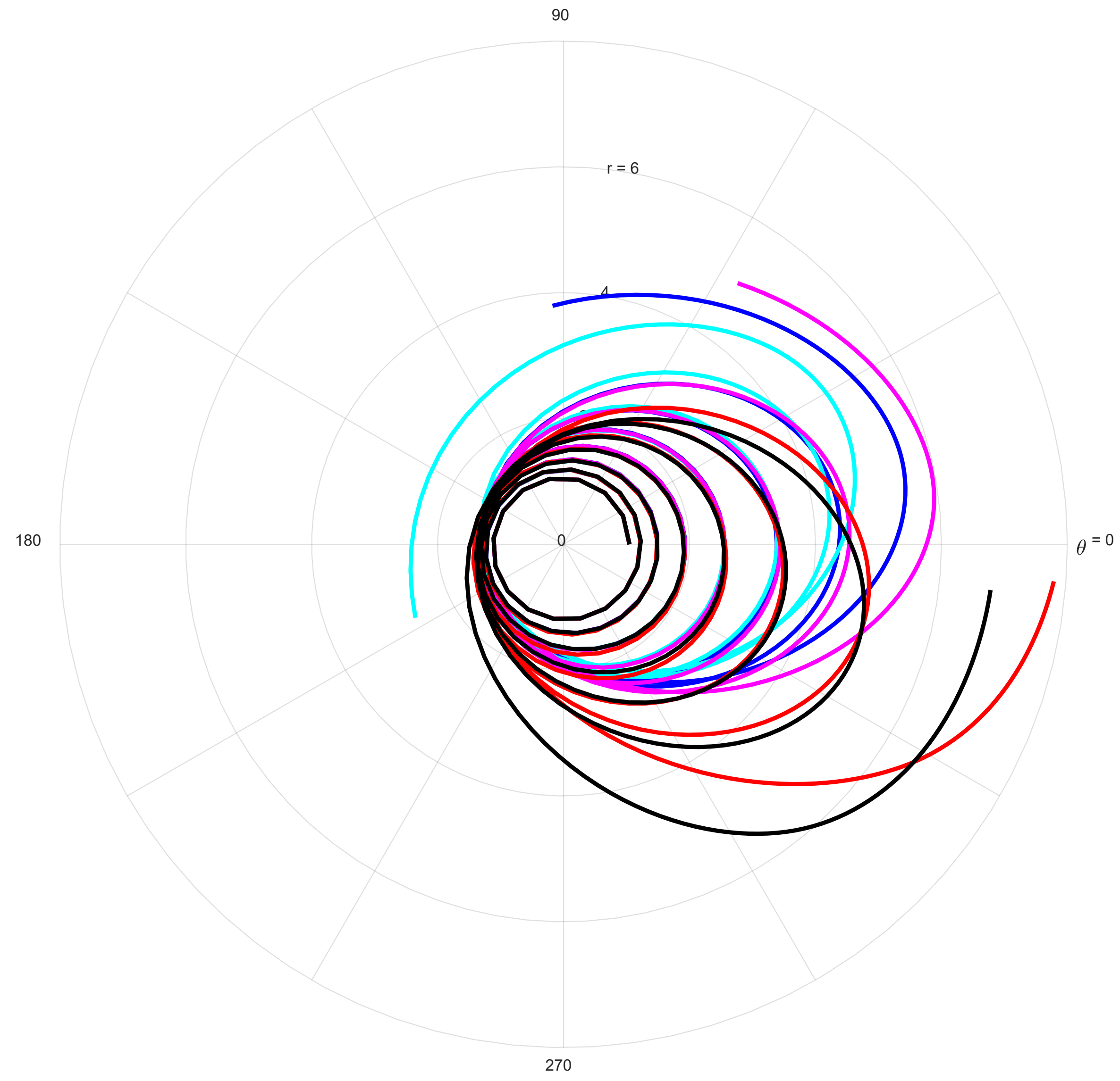


Figure 5.43: Comparison of "Best" Constrained Fuel-Optimal Open-Loop Trajectories

Table LII: Comparison of "Best" Constrained Fuel-Optimal Open-Loop Trajectories

Neural Network	Test Case RMSE	Training Time	Final Conditions $[r, \theta, v_r, v_t]$
LSTM - Current State with Consistent Timesteps	8.4675e-4 DU/TU^2 0.0234 rad	0.01 hr	[7.4672 DU, 44.11 rad, 0.0342 DU/TU, 0.3417 DU/TU] [7.99%, N/A, N/A, 4.70%]
FF - Current State and Changes with Consistent Timesteps	8.1203e-4 DU/TU^2 0.0219 rad	0.06 hr	[7.4447 DU, 44.10 rad, 0.0262 DU/TU, 0.3364 DU/TU] [7.66%, N/A, N/A, 6.20%]



▬	LSTM Neural Network Previous State – Current Control Pairs with Inconsistent Timesteps
▬	LSTM Neural Network Previous State and Changes – Current Control Pairs with Inconsistent Timesteps
▬	Feedforward Neural Network Previous State – Current Control Pairs with Inconsistent Timesteps
▬	Feedforward Neural Network Previous State and Changes – Current Control Pairs with Inconsistent Timesteps
▬	Optimized Solution

Figure 5.44: Comparison of Constrained Fuel-Optimal Closed-Loop Trajectories using Inconsistent Timesteps

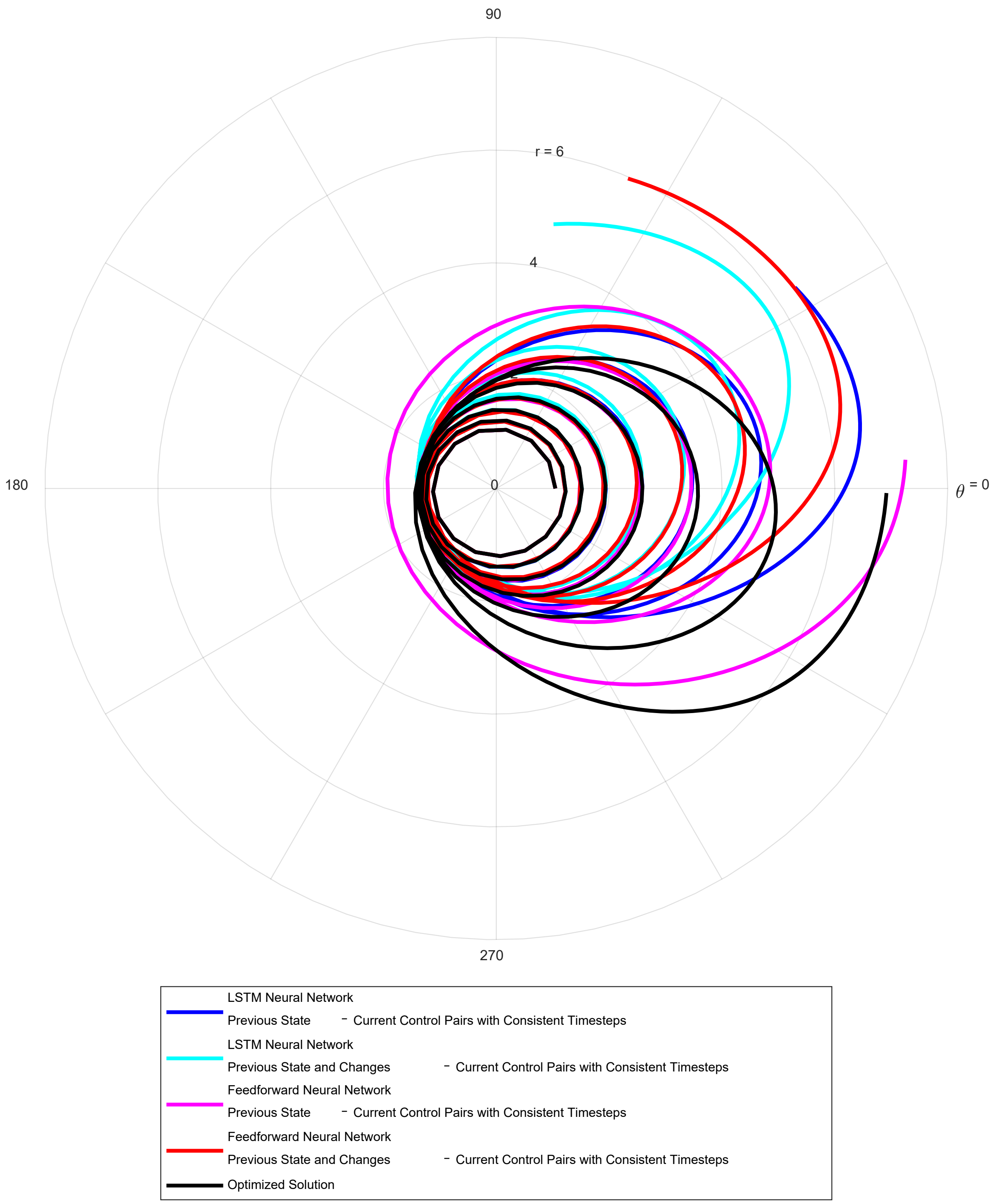


Figure 5.45: Comparison of Constrained Fuel-Optimal Closed-Loop Trajectories using Consistent Timesteps

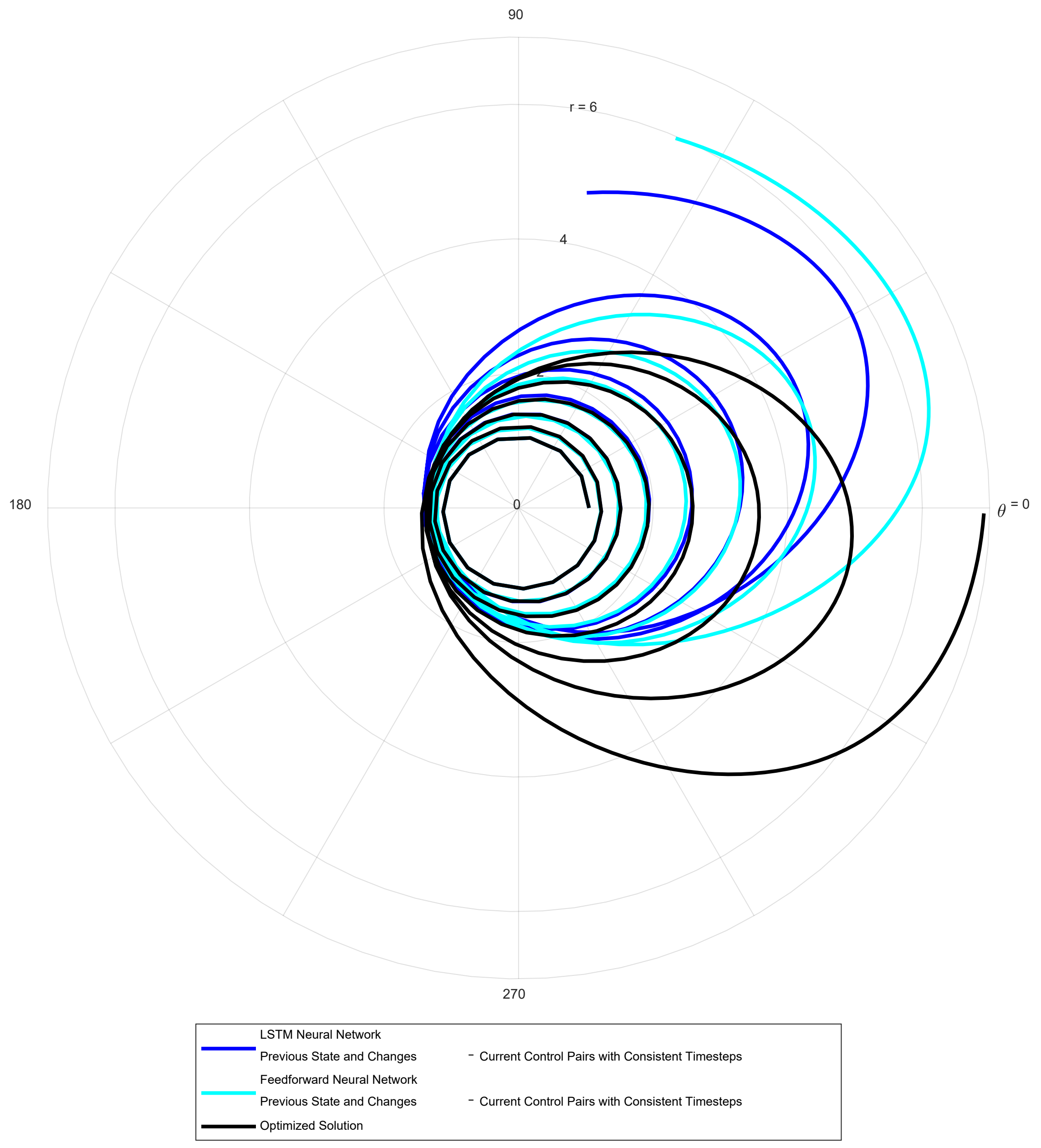


Figure 5.46: Comparison of "Best" Constrained Fuel-Optimal Closed-Loop Trajectories

Table LIII: Comparison of "Best" Constrained Fuel-Optimal Closed-Loop Trajectories

Neural Network	Test Case RMSE	Training Time	Final Conditions $[r\ ,\ \theta\ ,\ v_r\ ,\ v_t\ ]$
LSTM - Previous State & Changes with Consistent Timesteps	3.8212e-3 DU/TU^2 0.0787 rad	0.03 hr	[4.7928 DU, 45.34 rad, 0.1264 DU/TU, 0.4585 DU/TU] [30.69%, N/A, N/A, 27.87%]
FF - Previous State and Changes with Consistent Timesteps	3.2679e-3 DU/TU^2 0.0777 rad	0.07 hr	[5.9699 DU, 45.15 rad, 0.0448 DU/TU, 0.4250 DU/TU] [13.67%, N/A, N/A, 18.52%]

Table LIV: Comparison of Constrained Fuel-Optimal Open-Loop Fuel Usage

Neural Network		Optimized Fuel Usage (DU/TU^2)	Predicted Fuel Usage (DU/TU^2)	Percent Difference
LSTM	Current State with Inconsistent Timesteps	0.5429	0.5427	0.04%
Feedforward			0.5512	1.53%
LSTM	Current State and Changes with Inconsistent Timesteps		0.5468	0.72%
Feedforward			0.5492	1.16%
LSTM	Current State with Consistent Timesteps	0.5433	0.5366	1.23%
Feedforward			0.5372	1.12%
LSTM	Current State and Changes with Consistent Timesteps		0.5337	1.77%
Feedforward			0.5378	1.01%

Table LV: Comparison of Constrained Fuel-Optimal Closed-Loop Fuel Usage

Neural Network		Optimized Fuel Usage (DU/TU^2)	Predicted Fuel Usage (DU/TU^2)	Percent Difference
LSTM	Previous State with Inconsistent Timesteps	0.5429	0.4948	8.86%
Feedforward			0.5011	7.70%
LSTM	Previous State and Changes with Inconsistent Timesteps		0.5233	3.61%
Feedforward			0.5308	2.23%
LSTM	Previous State with Consistent Timesteps	0.5433	0.4917	9.50%
Feedforward			0.5032	7.38%
LSTM	Previous State and Changes with Consistent Timesteps		0.4685	13.77%
Feedforward			0.5308	2.30%

Table LVI: Comparison as Initial Guesses (Constrained Fuel-Optimal)

Initial Guess	Thrust Acceleration (DU/TU^2)	Convergence Time (hr)	Constraint Violation
LSTM Neural Network	0.5451	0.72	3.1829e-7
Feedforward Neural Network	0.5446	0.24	3.2193e-6

5.4 Final Discussion

For the time-optimal problem (both open-loop and closed-loop), though the LSTM neural networks had the overall minimum RMSE for both thrust acceleration and thrust angle (when the control RMSEs were considered separately), the “best” LSTM neural network performed relatively on par with the “best” feedforward neural network (when the control RMSEs were considered in tandem, rather than separately) with respect to the control RMSEs, the training times, the final boundary conditions for the integrated trajectories, and the predicted fuel usage. Note, the LSTM neural networks, with an appropriate batch size, had training times consistently faster than their feedforward equivalents. With regards to the effect of the time bias in the datasets on the neural networks, consistent timesteps typically improved the thrust acceleration predictions for both the LSTM neural networks and their feedforward equivalents whereas the thrust angle predictions were typically improved by using the desired changes in the states alongside the states as inputs.

For the constrained-fuel optimal open-loop case, though the LSTM neural networks did not have the overall minimum RMSE for either thrust acceleration or thrust angle (when the control RMSEs were considered separately), the "best" LSTM neural network performed relatively on par with the "best" feedforward neural network (when the control RMSEs were considered in tandem, rather than separately) with respect to the control RMSEs, the training times, the final boundary conditions for the integrated trajectories, and the predicted fuel usage. Note, the LSTM neural networks, with an appropriate batch size, had training times consistently faster than their feedforward equivalents. With regards to the effect of the time bias in the datasets on the neural networks, consistent timesteps improved the control predictions for both the LSTM neural networks and their feedforward equivalents. Using the desired changes in the states alongside the states as inputs also appeared to improve the control predictions as well. For the constrained fuel-optimal closed-loop case, both the LSTM neural networks and their feedforward equivalents performed poorly with slightly better performance observed from both the LSTM neural networks and their feedforward equivalents when tuned and trained on consistent timesteps.

6 Conclusions

As the use of spacecraft launched with low-thrust propulsion can be expected to continue increasing significantly, so can the need for quick, accurate, and robust low-thrust trajectory optimization. However, low-thrust trajectory optimization via traditional optimization techniques can be complex as well as computationally-expensive and time-consuming. Artificial neural networks, on the other hand, can provide an alternative to such techniques. However, though the use of artificial neural networks trained by supervised learning models for applications related to low-thrust trajectory design and optimization is extensive, previous research has largely focused on shallow or deep feedforward architectures with little emphasis placed on recurrent ones, even though recurrent architectures are inherently more suited to time histories. Thus, the overall objective of this research was to investigate the potential of recurrent artificial neural networks, specifically LSTM artificial neural networks, for low-thrust trajectory design and optimization with respect to their feedforward equivalents. As a majority of spacecraft launched with low-thrust propulsion have been GEO satellites, the focus of this research was, primarily, on the low-thrust, orbit-raising problem for both time-optimal and fuel-optimal transfers. A total of four problems were considered: the time-optimal open-loop case, the time-optimal closed-loop case, the fuel-optimal open-loop case, and the fuel-optimal closed-loop case. Overall, this dissertation presented and discussed the results of the LSTM neural networks and their feedforward equivalents. In consideration of these results (specific to the methodology outlined in this dissertation), the following conclusions (organized according to scientific contribution) were reached.

Contribution* $^{\#}$*1: *Recurrent artificial neural networks, specifically LSTM artificial neural networks, were applied to the low-thrust, orbit-raising problem for both time-optimal and fuel-optimal transfers.*

1. For time-optimal transfers (open-loop solution), LSTM neural networks (using a batch size of 20 with the current states and the desired changes for those states as inputs) performed excellent, capable of an average predicted thrust acceleration RMSE of 3.1079e-5 DU/TU^2 and an average predicted thrust angle RMSE of 0.0154 radians with an average training time of roughly 1.8 minutes. Also, the integrated trajectory from the LSTM neural networks had final radial velocities close to zero as well as final radii and final tangential velocities within 1 percent of the final values for the integrated trajectory from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. Finally, the predicted fuel usage for the LSTM neural networks was within 1 percent of the optimized fuel usage from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB.
2. For fuel-optimal transfers (open-loop solution), LSTM neural networks (using a batch size of 20 with the current states as inputs) performed well, capable of an average predicted thrust acceleration RMSE of 8.4675e-4 DU/TU^2 and an average predicted thrust angle RMSE of 0.0234 radians with an average training time of roughly 0.6 minutes. Also, the integrated trajectory from the LSTM neural networks had final radial velocities close to zero as well as final radii and final tangential velocities within 8 percent of the final values for the integrated trajectory from the Hermite-Simpson direct collocation method implemented using the *fmincon* function in MATLAB. Finally, the predicted fuel usage for the LSTM neural networks was within 2 percent of the optimized fuel usage from the Hermite-

Simpson direct collocation method implemented using the *fmincon* function in MATLAB.

Contribution #2: *An explicit model for a trade study between feedforward and recurrent artificial neural networks, specifically LSTM artificial neural networks, was developed and conducted with a focus on the low-thrust, orbit-raising problem for both time-optimal and fuel-optimal transfers.*

1. For time-optimal transfers (open-loop solution), LSTM neural networks performed relatively on par with feedforward neural networks with respect to the following metrics: control RMSEs, the final boundary conditions for the integrated trajectories, and the predicted fuel usage. However, LSTM neural networks, with an appropriate batch size, trained faster than their feedforward equivalents.
2. For fuel-optimal transfers (open-loop solution), LSTM neural networks performed relatively on par with feedforward neural networks with respect to the following metrics: control RMSEs, the final boundary conditions for the integrated trajectories, and the predicted fuel usage. However, LSTM neural networks, with an appropriate batch size, trained faster than their feedforward equivalents.

Contribution #3: *The applicability of recurrent artificial neural networks, specifically LSTM artificial neural networks, as a closed-loop solution to the low-thrust, orbit-raising problem for both time-optimal and fuel-optimal transfers was investigated.*

1. For time-optimal transfers (closed-loop solution), both LSTM neural networks and their feedforward equivalents performed excellent with the LSTM neural networks performing relatively on par with the feedforward neural networks with respect to the following metrics: control RMSEs, the final boundary conditions for the integrated

trajectories, and the predicted fuel usage. Again, LSTM neural networks, with an appropriate batch size, trained faster than their feedforward equivalents.

2. For fuel-optimal transfers (closed-loop solution), both LSTM neural networks and their feedforward equivalents performed poorly. The results for the constrained fuel-optimal closed-loop case were unfavorable, but were included in this dissertation for the sake of completeness.

Contribution #4: *The effect of time bias in data on the performance of recurrent artificial neural networks, specifically LSTM artificial neural networks, and feedforward artificial neural networks was investigated.*

1. For time-optimal transfers (both open-loop and closed-loop solutions), consistent timesteps typically improved the thrust acceleration predictions for both LSTM neural networks and their feedforward equivalents whereas the thrust angle predictions were typically improved by using the desired changes in the states alongside the states as inputs.
2. For fuel-optimal transfers (both open-loop and closed-loop solutions), consistent timesteps improved the control predictions for both LSTM neural networks and their feedforward equivalents. Using the desired changes in the states alongside the states as inputs also appeared to improve the control predictions as well.

6.1 Possibilities for Future Research

The ability of recurrent artificial neural networks to perform relatively on par with, but faster than, feedforward artificial neural networks for the planar, two-body, low-thrust, orbit-raising problem for both time-optimal and fuel-optimal transfers supports their potential for low-thrust trajectory design and optimization, particularly when considering more complex problems in which the increasing nonlinearity might allow for the full exploitation of the benefits associated with recurrent artificial neural networks. Thus, two possibilities for future research are to expand the investigation on the potential of recurrent artificial neural networks for low-thrust trajectory design and optimization to a three-dimensional, low-thrust, orbit-raising transfer as well as other low-thrust problems, such as rendezvous and interplanetary trajectories (both planar and three-dimensional).

A third, and final, possibility for research is to improve the applicability of recurrent artificial neural networks as a closed-loop solution to the low-thrust, orbit-raising problem for fuel-optimal transfers. Smaller timesteps between points in the datasets could improve performance as well as a different optimization or integration scheme. Also, the methodology outlined in this dissertation may not be the best way to approach the constrained fuel-optimal closed-loop problem with respect to artificial neural networks. Crucial information, such as mass dependencies, could be getting lost in the translation of the problem.

References

[1] Topputo, F., and Zhang, C., "Survey of Direct Transcription for Low-Thrust Space Trajectory Optimization with Applications," *Abstract and Applied Analysis*, Vol. 2014, No. 2, 2014.

[2] Conway, B., *Spacecraft Trajectory Optimization*, 1st ed., Cambridge University Press, New York, 2010.

[3] Lev, D., Myers, R., Lemmer, K., Kolbeck, J., Keidar, M., Koizumi, H., Liang, H., Yu, D., Schönherr, T., Gonzalez del Amo, J., Choe, W., Albertoni, R., Hoskins, A., Yan, S., Hart, W., Hofer, R., Funaki, I., Lovstov, A., Polzin, K., Olshanskii, A., and Duchemin, O., "The Technological and Commercial Expansion of Electric Propulsion in the Past 24 Years," *International Electric Propulsion Conference (IEPC)*, 2017.

[4] Lev, D., Myers, R., Lemmer, K., Kolbeck, J., Koizumi, H., and Polzin, K., "The Technological and Commercial Expansion of Electric Propulsion," *Acta Astronautica*, Vol. 159, 2019, pp. 213-227.

[5] *Iridium 4*, National Aeronautics and Space Administration (NASA) Space Science Data Coordinate Archive, 2022. Available from https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1997-020E.

[6] *Starlink*, Starlink, 2022. Available from https://www.starlink.com/.

[7] *OneWeb*, OneWeb, 2022. Available from https://oneweb.net/.

[8] Jahn, R., and Choueiri, E., "Electric Propulsion", *Encyclopedia of Physical Science and Technology*, 3rd ed., Academic Press, California, 2001, pp. 125 – 141.

[9] Sutton, G., and Biblarz, O., "Electric Propulsion", *Rocket Propulsion Elements*, 9th ed., John Wiley & Sons, Inc., New Jersey, 2017, pp. 620 – 670.

[10] *SERT 2A*, National Aeronautics and Space Administration (NASA) Space Science Data Coordinate Archive, 2022. Available from https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1970-009A.

[11] *Zond 2*, National Aeronautics and Space Administration (NASA) Space Science Data Coordinate Archive, 2022. Available from https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1964-078C.

[12] *LES 6*, National Aeronautics and Space Administration (NASA) Space Science Data Coordinate Archive, 2022. Available from https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1968-081D.

[13] *Vela 3A*, National Aeronautics and Space Administration (NASA) Space Science Data Coordinate Archive, 2022. Available from https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1965-058A.

[14] *Meteor 1-10*, National Aeronautics and Space Administration (NASA) Space Science Data Coordinate Archive, 2022. Available from https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1971-120A.

[15] *Intelsat 5 F-2*, National Aeronautics and Space Administration (NASA) Space Science Data Coordinate Archive, 2022. Available from https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1980-098A.

[16] *Telstar 401*, National Aeronautics and Space Administration (NASA) Space Science Data Coordinate Archive, 2022. Available from https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1993-077A.

[17] *Deep Space 1*, National Aeronautics and Space Administration (NASA) Space Science Data Coordinate Archive, 2022. Available from https://nssdc.gsfc.nasa.gov/nmc/

spacecraft/display.action?id=1998-061A.

[18] *Artemis*, National Aeronautics and Space Administration (NASA) Space Science Data Coordinate Archive, 2022. Available from https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=2001-029A.

[19] *SMART 1*, National Aeronautics and Space Administration (NASA) Space Science Data Coordinate Archive, 2022. Available from https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=2003-043C.

[20] *ABS 3A*, National Aeronautics and Space Administration (NASA) Space Science Data Coordinate Archive, 2022. Available from https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=2015-010A.

[21] Kelly, M., "An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation," *Society for Industrial and Applied Mathematics (SIAM) Review*, Vol. 59, No. 4, 2017, pp. 849 – 904.

[22] Kelly, M., "Transcription Methods for Trajectory Optimization: A Beginners Tutorial," *arXiv*, 2017.

[23] Edelbaum, T., "Propulsion Requirements for Controllable Satellites," *Journal of the American Rocket Society*, Vol. 31, No. 8, 1961, pp. 1079 – 1089.

[24] Wiesel, W., and Alfano, S., "Optimal Many-Revolution Orbit Transfer," *Journal of Guidance, Control, and Dynamics*, Vol. 8, No. 1, 1985, pp. 155 – 157.

[25] Kechichian, J., "Reformulation of Edelbaum's Low-Thrust Transfer Problem Using Optimal Control Theory," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 5, 1997, pp. 988 – 994.

[26] Kechichian, J., "Low-Thrust Eccentricity-Constrained Orbit Raising," *Journal of*

Spacecraft and Rockets, Vol. 35, No. 3, 1998, pp. 327 – 335.

[27] Colasurdo, G., and Casalino, L., "Optimal Low-Thrust Maneuvers in Presence of Earth Shadow," *American Institute of Aeronautics and Astronautics (AIAA) / American Astronomical Society (AAS) Astrodynamics Specialist Conference and Exhibit*, 2004.

[28] Kechichian, J., "Minimum-Time Constant Acceleration Orbit Transfer with First-Order Oblateness Effect," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 4, 2000, pp. 595 – 603.

[29] Kechichian, J., "Optimal Altitude-Constrained Low-Thrust Transfer Between Inclined Circular Orbits," *The Journal of the Astronautical Sciences*, Vol. 54, No. 3 & 4, 2006, pp. 485 – 503.

[30] *Trapezoidal Rule*, Wolfram Alpha LLC, 2022. Available from https://www.wolframalpha.com/input/?i=trapezoidal+rule.

[31] *Simpson Rule*, Wolfram Alpha LLC, 2022. Available from https://www.wolframalpha.com/input/? i=simpson+rule.

[32] Haykin, S., *Neural Networks and Learning Machines*, 3rd ed., Pearson Education, Inc., New Jersey, 2009.

[33] Russell, S., and Norvig, P., *Artificial Intelligences: A Modern Approach*, 3rd ed., Pearson Education, Inc., New Jersey, 2010.

[34] Rosenblatt, F., "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, Vol. 65, No. 6, 1958, pp. 386 – 408.

[35] McCulloch, W., and Pitts, W., "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, Vol. 5, 1943, pp. 115 – 133.

[36] Rumelhart, D., and McClelland, J., *Parallel Distributed Processing: Exploration in the*

Microstructure of Cognition, Vol. 1, The MIT Press, Massachusetts, 1986.

[37] Brownlee, J., *Machine Learning Mastery*, 2022. Available from https://machinelearningmastery.com/.

[38] Hochreiter, S., and Schmidhuber, J., "Long Short-Term Memory," *Neural Computation*, Vol. 9, No. 8, 1997, pp. 1735 – 1780.

[39] Graves, A., "Supervised Sequence Labelling with Recurrent Neural Networks," *Studies in Computational Intelligences*, Vol. 385, 2012.

[40] Graves, A., "Generating Sequences with Recurrent Neural Networks," *arXiv*, 2014.

[41] Schuster, A., "Artificial Intelligence for Space Applications", *Intelligent Computing Everywhere*, 1st ed., Springer, United Kingdom, 2007, pp. 235 – 254.

[42] Izzo, D., Sprague, C., and Tailor, D., "Machine Learning and Evolutionary Techniques in Interplanetary Trajectory Design," *arXiv*, 2018.

[43] Izzo, D., Märtens, M., and Pan, B., "A Survey on Artificial Intelligence Trends in Spacecraft Guidance Dynamics and Control," *arXiv*, 2018.

[44] Ampatzis, C., and Izzo, D., "Machine Learning Techniques for Approximation of Objective Functions in Trajectory Optimisation," *International Joint Conference on Artificial Intelligence (IJCAI), Proceedings*, Vol. 673, 2009, pp. 1 – 6.

[45] Zhu, Y., Luo, Y., and Yao, W., "Fast Accessibility Evaluation of the Main-Belt Asteroids Manned Exploration Mission Based on a Learning Method," *2018 Institute of Electrical and Electronics Engineers (IEEE) Congress on Evolutionary Computation (CEC)*, 2018.

[46] Youmans, E., and Lutze, F., "Neural Network Control of Space Vehicle Intercept and Rendezvous Maneuvers," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 1, 1998, pp. 116 – 121.

[47] Yang, B., and Li, S., "Fast Solver for J2-Perturbed Lambert Problem Using Deep Neural Networks," *Journal of Guidance, Control, and Dynamics (Articles in Advance)*, 2022.

[48] Peng, H., and Bai, X., "Improving Orbit Prediction Accuracy through Supervised Machine Learning," *arXiv*, 2018.

[49] Peng, H., and Bai, X., "Using Artificial Neural Networks in Machine Learning Approach to Improve Orbit Prediction Accuracy," *American Institute of Aeronautics and Astronautics (AIAA) SciTech Forum*, 2018.

[50] Peng, H., and Bai, X., "Artificial Neural Network - Based Machine Learning Approach to Improve Orbit Prediction Accuracy," *Journal of Spacecraft and Rockets*, Vol. 55, No. 5, 2018, pp. 1248 – 1260.

[51] Guého, D., Schwab, D., Singla, P., and Melton, R., "A Comparison of Parametric and Non-Parametric Machine Learning Approaches for the Uncertain Lambert Problem," *American Institute of Aeronautics and Astronautics (AIAA) SciTech Forum*, 2020.

[52] Lary, D., "Artificial Intelligence in Aerospace," *Aerospace Technologies Advancements*, 1st ed., IntechOpen, United Kingdom, 2010, pp. 1 – 24.

[53] Chadalavada, P., Farabi, T., and Dutta, A., "Sequential Low-Thrust Orbit-Raising of All-Electric Satellites," *Aerospace*, Vol. 7, No. 74, 2020, pp. 1 – 27.

[54] George, T., "The Use of Long Short-Term Memory Artificial Neural Networks for the Global Prediction of Atmospheric Density," M.S. Thesis, Dept. of Aerospace Engineering, Univ. of Kansas, Lawrence, Kansas, 2020.

[55] George, T., and McLaughlin, C., "The Use of Long Short-Term Memory Artificial Neural Networks for the Global Prediction of Atmospheric Density," *American Astronomical Society (AAS) Astrodynamics Specialist Conference*, 2020.

[56] Gelly, G., and Vernis, P., "Neural Networks as a Guidance Solution for Soft-Landing and Aerocapture," *American Institute of Aeronautics and Astronautics (AIAA) Guidance, Navigation, and Control Conference*, 2009.

[57] Sánchez-Sánchez, C., Izzo, D., and Hennes, D., "Optimal Real-Time Control Using Deep Neural Networks," *6th International Conference on Astrodynamics Tools and Techniques (ICATT)*, 2016.

[58] Sánchez-Sánchez, C., Izzo, D., and Hennes, D., "Learning the Optimal State-Feedback Using Deep Neural Networks," *Institute of Electrical and Electronics Engineers (IEEE) Symposium Series on Computational Intelligence (SSCI)*, 2016.

[59] Cheng, L., Wang, Z., Song, Y., and Jiang, F., "Real-Time Optimal Control for Irregular Asteroid Landing Using Deep Neural Networks," *arXiv*, 2019.

[60] Huang, Y., Li, S., and Sun, J., "Mars Entry Fault-Tolerant Control Via Neural Network and Structure Adaptive Model Inversion," *Advances in Space Research*, Vol. 63, No. 1, 2019, pp. 557 – 571.

[61] Cheng, L., Wang, Z., Jiang, F., and Li, J., "Fast Generation of Optimal Asteroid Landing Trajectories Using Deep Neural Networks," *Institute of Electrical and Electronics Engineers (IEEE) Transactions on Aerospace and Electronic Systems*, Vol. 56, No. 4, 2020, pp. 2642 – 2655.

[62] Sreesawet, S., Pappu, V., Dutta, A., and Steck, J., "Neural Network Based Adaptive Controller for Attitude Control of All-Electric Satellites," *American Astronomical Society Astrodynamics Specialist Conference*, 2015.

[63] Ramos, E., Mishra, P., Edwards, S., and Mavris, D., "Response Surface Regressions for Low-Thrust Interplanetary Mission Design," *American Institute of Aeronautics and*

Astronautics (AIAA) SPACE Forum, 2016.

[64] Mereta, A., Izzo, D., and Wittig, A., "Machine Learning of Optimal Low-Thrust Transfers between Near-Earth Objects," *Hybrid Artificial Intelligent Systems (HAIS) – 12th International Conference, Proceedings*, Vol. 10334, 2017, pp. 543 – 553.

[65] Sánchez-Sánchez, C., and Izzo, D., "Real-Time Optimal Control Via Deep Neural Networks: Study on Landing Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 5, 2018, pp. 1122 – 1135.

[66] Parrish, N., "Low Thrust Trajectory Optimization in Cislunar and Translunar Space," Ph.D. Dissertation, Dept. of Aerospace Engineering Sciences, Univ. of Colorado, Boulder, Colorado, 2018.

[67] Izzo, D., Öztürk, E., and Märtens, M., "Interplanetary Transfers via Deep Representations of the Optimal Policy and/or of the Value Function," *arXiv*, 2019.

[68] Chen, G., "Machine Learning Regression for Estimating Characteristics of Low-Thrust Transfers," M.S. Thesis, Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, Georgia, 2019.

[69] Casey, J., "A Methodology for Sequential Low Thrust Trajectory Optimization Using Prediction Models Derived from Machine Learning Techniques," M.S. Thesis, Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, Georgia, 2019.

[70] Song, Y., and Gong, S., "Solar-Sail Trajectory Design for Multiple Near-Earth Asteroid Exploration Based on Neural Networks," *Aerospace Science and Technology*, Vol. 91, 2019, pp. 28 – 40.

[71] Li, H., Topputo, F., and Baoyin, H., "Autonomous Time-Optimal Many-Revolution Orbit

Raising for Electric Propulsion GEO Satellites via Neural Networks," *arXiv*, 2019.

[72] Cheng, L., Wang, Z., Jiang, F., and Zhou, C., "Real-Time Optimal Control for Spacecraft Orbit Transfer via Multiscale Deep Neural Networks," *Institute of Electrical and Electronics Engineers (IEEE) Transactions on Aerospace and Electronic Systems*, Vol. 55, No. 5, 2019, pp. 2436 – 2450.

[73] Li, H., Baoyin, H., and Topputo, F., "Neural Networks in Time-Optimal Low-Thrust Interplanetary Transfers," *Institute of Electrical and Electronics Engineers (IEEE) Access*, Vol. 7, No. 1, 2019, pp. 156413 – 156419.

[74] Zhu, Y., and Luo, Y., "Fast Evaluation of Low-Thrust Transfers via Multilayer Perceptions," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 12, 2019, pp. 2627 – 2637.

[75] Sprague, C., Izzo, D., and Ögren, P., "Learning a Family of Optimal State-Feedback Controllers with Homotopy and Trajectory Optimisation," *arXiv*, 2020.

[76] Li, H., Chen, S., Izzo, D., and Baoyin, H., "Deep Networks as Approximators of Optimal Low-Thrust and Multi-Impulse Cost in Multitarget Missions," *Acta Astronautica*, Vol. 166, 2020, pp. 469 – 481.

[77] Rubinsztejn, A., Sood, R., and Laipert, F., "Neural Network Optimal Control in Astrodynamics: Application to the Missed Thrust Problem," *Acta Astronautica*, Vol. 176, 2020, pp. 192 – 203.

[78] Yin, S., Li, J., and Cheng, L., "Low-Thrust Spacecraft Trajectory Optimization via a DNN-Based Method," *Advances in Space Research*, Vol. 66, No. 7, 2020, pp. 1635 – 1646.

[79] Izzo, D., and Öztürk, E., "Real-Time Guidance for Low-Thrust Transfers Using Deep Neural Networks," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 2, 2021, pp.

315 – 327.

[80] Xie, R., and Dempster, A., “An On-Line Deep Learning Framework for Low-Thrust Trajectory Optimisation,” *Aerospace Science and Technology*, Vol. 118, No. 3, 2021, pp. 1 – 15.

[81] Viavattene, G., and Ceriotti, M., “Artificial Neural Network for Preliminary Multiple NEA Rendezvous Mission Using Low Thrust,” *International Astronautical Congress (IAC)*, 2019.

[82] Viavattene, G., and Ceriotti, M., “Artificial Neural Networks for Multiple NEA Rendezvous Missions with Continuous Thrust,” *Journal of Spacecraft and Rockets*, Vol. 59, No. 2, 2022, pp. 574 – 586.

[83] Schiassi, E., D’Ambrosio, A., Drozd, K., Curti, F., and Furfaro, R., “Physics-Informed Neural Networks for Optimal Planar Orbit Transfers,” *Journal of Spacecraft and Rockets (Articles in Advance)*, 2022.

[84] George, T., and Kaplinger, B., “The Use of Long Short-Term Memory Artificial Neural Networks for the Low-Thrust Orbit-Raising Problem,” *American Institute of Aeronautics and Astronautics (AIAA) SciTech*, 2022.

[85] George, T., and Kaplinger, B., “Autonomous Low-Thrust Orbit-Raising Using Long Short-Term Memory Neural Networks,” *American Astronomical Society (AAS) Astrodynamics Specialist Conference*, 2022.

[86] Furfaro, R., Bloise, I., Orlandelli, M., Di Lizia, P., Topputo, F., and Linares, R., “A Recurrent Deep Architecture for Quasi-Optimal Feedback Guidance in Planetary Landing,” *International Academy of Astronautics (IAA) / American Astronomical Society (AAS) SciTech Forum on Space Flight Mechanics and Space Structures and Materials*,

2018.

[87] Furfaro, R., Bloise, I., Orlandelli, M., Di Lizia, P., Topputo, F., and Linares, R., "Deep Learning for Autonomous Lunar Landing," *American Astronomical Society (AAS) / American Institute of Aeronautics and Astronautics (AIAA) Astrodynamics Specialist Conference*, 2018.

[88] Silvestrini, S., and Lavagna, M., "Relative Trajectories Identification in Distributed Spacecraft Formation Collision-Free Maneuvers Using Neural-Reconstructed Dynamics," *American Institute of Aeronautics and Astronautics (AIAA) SciTech Forum*, 2020.

[89] Scorsoglio, A., Furfaro, R., Linares, R., and Gaudet, B., "Image-Based Deep Reinforcement Learning for Autonomous Lunar Landing," *American Institute of Aeronautics and Astronautics (AIAA) SciTech Forum*, 2020.

[90] Scorsoglio, A., D'Ambrosio, A., Ghilardi, L., Gaudet, B., Curti, F., and Furfaro, R., "Image-Based Deep Reinforcement Meta-Learning for Autonomous Lunar Landing," *Journal of Spacecraft and Rockets*, Vol. 59, No.1, 2022, pp. 153 – 165.

[91] Biggs, J., and Fournier, H., "Neural-Network-Based Optimal Attitude Control Using Four Impulsive Thrusters," *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 2, 2020, pp. 299 – 309.

[92] Jaworski, S., and Kindracki, J., "Feasibility of Synchronized CubeSat Missions from Low Earth Orbit to Near-Earth Asteroids," *Journal of Spacecraft and Rockets*, Vol. 57, No.6, 2020, pp. 1232 – 1245.

[93] *Python*, Software Package, Ver. 3.8.8, Python Software Foundation, 2021, Available from https://www.python.org/.

[94] Chollet, F., *Keras*, Software Package, Ver. 2.7.0, 2021. Available from https://keras.io/.

[95] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia,Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, Software Package, Ver. 2.8.0, 2022. Available from https://www.tensorflow.org/.

[96] Kingma, D., and Ba, J., "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations (ICLR)*, 2015.

[97] Chollet, F., *Deep Learning with Python*, 1st ed., Manning Publications Company, New York, 2018.