Acta Astronautica 176 (2020) 192–203

Contents lists available at ScienceDirect

Acta Astronautica

journal homepage: www.elsevier.com/locate/actaastro

Neural network optimal control in astrodynamics: Application to the missed thrust problem[☆]

Ari Rubinsztejn[a,*], Rohan Sood[a], Frank E. Laipert[b]

[a] Astrodynamics and Space Research Laboratory, Department of Aerospace Engineering and Mechanics, The University of Alabama, Tuscaloosa, AL, 35487, USA
[b] Flight Path Control Group, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109, USA

ARTICLE INFO

Keywords:
Neural networks
Artificial intelligence
Astrodynamics
Missed thrust events
Low-thrust

ABSTRACT

While high-efficiency propulsion techniques are enabling new mission concepts in deep space exploration, their limited thrust capabilities necessitate long thrusting arcs and make spacecraft more susceptible to missed thrust events. To correct for such mishaps, most spacecraft require updated trajectories that are relayed from Earth. While this solution is viable for spacecraft near Earth, in deep space, where one-way communication time is measured in hours, a delay in transmission may prolong the time of flight or result in a complete loss of mission. Such problems can be alleviated by increasing the spacecraft's onboard autonomy in guidance. This paper demonstrates how a computationally lightweight neural network can map the spacecraft's state to a near-optimal control action, autonomously guiding a spacecraft within different astrodynamic regimes and optimality criteria. The neural network is trained using supervised learning and datasets comprised of optimal state-action pairs, as determined through traditional direct and indirect methods. Additionally, the neural network-designed solutions retain optimality and time of flight corresponding to traditional trajectories. Finally, the same neural networks can autonomously correct for most missed thrust events encountered on long-duration low-thrust trajectories. The presented results provide a path for mitigating risks associated with the use of high-efficiency low-thrust propulsion techniques.

1. Introduction

High-efficiency propulsion systems are enabling new deep space science missions. These systems are being incorporated across a wide spectrum of NASA missions, from small and low-cost missions like the LunaH-Map [1] and the Psyche spacecraft [2], to flagship missions like Mars Sample Return (MSR) [3,4]. However, the propulsion systems come with certain challenges; chief among them is their limited thrust. Preliminary trajectory design strategies, originally developed for high-thrust engines, make the assumption that the thrust is applied instantaneously. Removing this assumption has several consequences, and a significant amount of work over the past two decades has gone into developing new trajectory design tools for low-thrust propulsion capabilities. One consequence of removing the instantaneous thrust assumption is that, during the thrusting period, a software glitch or an external factor, such as a micro-meteoroid impact [5], may cause the spacecraft to enter *safe mode* and prematurely cease the thrust stage. An unexpected missed thrust event (MTE) can potentially occur over the thrusting phase and may have severe consequences if the spacecraft has a time-dependent trajectory, e.g., a scheduled planetary flyby. Currently, there exists no general solution to designing low-thrust trajectories that are resilient to such MTEs.

The effects of MTEs are mission dependent. Recent low-thrust heliophysics mission studies often use solar sails as their primary source of propulsion [6,7], where a 10 day MTE (simulating an active trim maneuver failure) may result in a delay of over 200 days [8]. Moreover, planetary science missions that rely on gravity assists or precise rendezvous can be lost entirely if the spacecraft is unable to reach its target after a MTE [8]. The lack of trajectories resilient to MTEs impacts all areas of space science and exploration.

There are several sub-optimal techniques for dealing with MTEs. The simplest approach may temporarily disable the automated fault response systems [9]. While this approach does remove the risk of a missed thrust event, it increases the spacecraft's susceptibility to other risks as the conditions that would trigger a safe mode are often dangerous. A second approach is to artificially limit the spacecraft's thrusters to some fraction of their full potential, called the duty cycle. These artificial limits can then be removed in flight to compensate for

[☆] A previous version was presented at the 2019 AAS/AIAA Astrodynamics Specialist Conference (AAS 19–773).
[*] Corresponding author.
E-mail addresses: arubinsztejn@crimson.ua.edu (A. Rubinsztejn), rsood@eng.ua.edu (R. Sood), Frank.E.Laipert@jpl.nasa.gov (F.E. Laipert).

https://doi.org/10.1016/j.actaastro.2020.05.027
Received 3 December 2019; Received in revised form 11 May 2020; Accepted 12 May 2020
Available online 18 May 2020

MTEs. The duty cycle approach is an attractive option because it is already widely used to simulate the time spent on planned thrust stoppages like communication, and tracking [10]. However, it is by definition sub-optimal. A third approach is to determine where in a trajectory the spacecraft is most susceptible to a MTE, place a forced coasting arc at that time, and then re-optimize the trajectory. This process is repeated until the trajectory is resilient to MTEs shorter than a pre-specified amount [11]. The downside of this approach is, that it does not take into account the likelihood of multiple MTEs on a single trajectory. Building on the idea of a forced coast period is the idea of a rolling coast [12], where a forced coast is placed at some time ahead along the trajectory. If there is no MTE, then as the spacecraft approaches the forced coast, the forced coast is moved to a later state along the trajectory, and the trajectory is re-optimized. This process of moving the forced coast and re-optimizing the trajectory has to be performed continuously throughout the mission, which requires constant communication with mission designers and a large amount of ground-based computational resources.

The most modern approach to dealing with MTEs is to include additional fuel and time margins. In such an instance, a spacecraft may exhaust these propellant reserves in an attempt to compensate for the MTE [13]. The determination of these margins is extremely mission-specific and computationally expensive [14]. Compounding the expense of this approach is after the MTE, the new trajectory still needs to be designed on Earth. Spacecraft-based hardware often lags a decade behind ground-based technology in computing power [15] and, at the time of a MTE, onboard hardware may not be capable of recomputing new optimal trajectories in real-time. This limits spacecraft autonomy and control, one of NASA's research priorities [16,17] for deep space exploration, where one-way communication time is often measured in hours.

One approach to solving the problems inherent to deep space autonomy is to incorporate artificial intelligence in the trajectory design process. Artificial intelligence techniques capable of interacting with the environment, such as evolutionary neurocontrol [18] and reinforcement learning [19,20], have delivered promising results [21]. Additionally, traditional forms of supervised learning have been applied to real-time optimal trajectory design for low-thrust spacecraft in the two-body problem [22] as well as in landing problems [15,23]. One particularly interesting approach solves the optimal control problem using an indirect method and then trains a neural network to correct the co-states as the spacecraft deviates from a nominal trajectory [24,25]. Most mass/time optimal trajectories have a bang-bang control scheme where the controls are discontinuous, while the co-states are smooth. Due to this smoothness, neural networks better approximate the co-states than the controls. Once the neural network generates the new co-states, the controls can be calculated directly. Unfortunately, this approach introduces new problems and difficulties. Foremost, the relationship mapping the co-states to the controls is dependent on both the cost function being optimized, as well as the dynamics of the system [26]. This makes the mapping highly nonlinear, so a small error in the predicted co-states can be magnified. The second problem introduced with this approach is the complexity of generating the training dataset. To generate the co-states, an indirect method is required, which suffers from very small radii of convergence for mass/time-optimal trajectories, difficulties with including path constraints, and difficulties with complex boundary constraints. These limitations with using indirect methods for optimal control have led many astrodynamicists to use direct methods instead [27], which do not produce the co-states required for the above approach.

One commonality of the above artificial intelligence approaches, is that none have explored the application of neural networks to the missed thrust problem. By applying autonomous optimal control to the missed thrust problem, spacecraft can repair their trajectories after a missed thrust event. This is one key element of making spacecraft safe mode tolerant and increasing deep space autonomy. This paper demonstrates that a computationally lightweight neural network, that maps states directly to controls, can be employed to successfully guide a spacecraft using a low-thrust propulsion system through *three* distinct optimal control problems. Through directly mapping the states to controls, this paper allows the use of both direct and indirect optimal control methods for the creation of the neural network. Additionally, when MTEs are introduced, a neural network-based controller is highly effective for autonomously correcting a majority of these events in all *three* problems.

2. Optimal control problems

The three different optimal control problems (OCP) this work investigates are:

1. Two-Body Orbit-to-Orbit Low-Thrust Transfer,
2. Two-Body Interplanetary Low-Thrust Transfer, and
3. Circular Restricted Three-Body Problem Low-Trust Transfers.

These problems span a wide range of boundary conditions and highlight the generalizability of using neural networks for optimal control in the field of astrodynamics.

2.1. Two-body orbit-to-orbit transfer

The two-body orbital transfer OCP is formulated as a transfer between two circular planar orbits centered about the Sun. The spacecraft is equipped with a low-thrust propulsion system, and the equations of motion, in polar form, governing the path of the spacecraft can be expressed as,

$$\ddot{r} = \frac{v_\theta^2}{r} - \frac{GM}{r^2} + \frac{c_1}{m}u\sin(\beta) \tag{1}$$

$$\dot{\theta} = \frac{v_\theta}{r} \tag{2}$$

$$\dot{v}_\theta = \frac{-\dot{r}v_\theta}{r} + \frac{c_1}{m}u\cos(\beta) \tag{3}$$

$$\dot{m} = \frac{-c_1 u}{g_0 c_2}, \tag{4}$$

where r is the distance from the sun, θ is the counter-clockwise angle relative to the inertial x-axis, m is the mass of the spacecraft, v_θ is the tangential velocity, c_1 is the spacecraft's max thrust, c_2 is its specific impulse, GM is the solar gravitational constant, u is the spacecraft's thrust action bounded between 0 and 1, and β is the in-plane control angle as shown in Fig. 1.

Values for the constants associated with Equations (1)–(4) can be found in Table 1, and are chosen to represent a possible architecture for the Earth Return Orbiter in the Mars Sample Return (MSR) mission [3].

For improved simulation performance, non-dimensionalization of the system can be performed by using several characteristic constants. The characteristic mass, m^*, is defined as the spacecraft's initial mass. The characteristic length, l^*, is equal to the Sun-Earth distance. The characteristic time, t^*, is defined to set the non-dimensional gravitational constant equal to 1 and can be calculated using Equation (5).

$$t^* = \sqrt{\frac{l^{*3}}{GM}} \tag{5}$$

The OCP being solved is the design of a trajectory that minimizes the quadratic thrust action, given as a cost function in Equation (6). For preliminary investigation, the trajectory is designed to take the spacecraft from a circular orbit with the semi-major axis of Mars to a circular orbit with the semi-major axis of Earth.

$$J = \int_{t_0}^{t_f} u(t)^2 dt \tag{6}$$

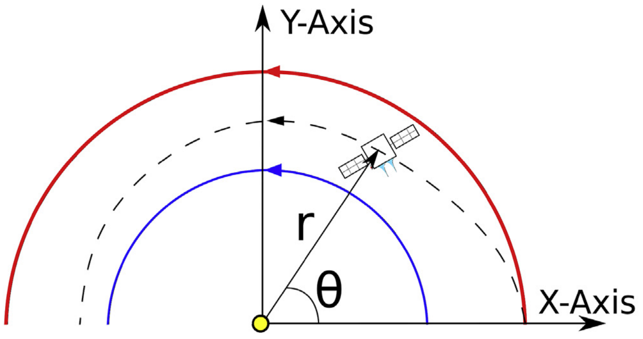

Fig. 1. Problem Setup:Left: Earth and Mars orbit with a representative trajectory right: Spacecraft enlarged to show thrust and control angle.

Table 1
Spacecraft and system parameters.

c_1 (N)	c_2 (s)	$GM\,(km^3/s^2)$	TOF (Year)
1.33	3872	1.3e11	1.1

This OCP is solved using an indirect method, which converts the minimization problem into a two-point boundary value problem (TPBVP) with the following Hamiltonian:

$$\mathscr{H} = \lambda_r \dot{r} + \lambda_\theta \dot{\theta} + \lambda_{\dot{r}} \ddot{r} + \lambda_{v_\theta} \dot{v}_\theta + \lambda_m \dot{m} + u^2, \tag{7}$$

where λ_i is the i^{th} co-state. The optimal controller can be determined by differentiating the Hamiltonian with respect to the control variables as shown in Equations (8) and (9).

$$\frac{\partial \mathscr{H}}{\partial \beta} = 0 \quad , \quad \beta = atan\left(\frac{\lambda_{\dot{r}}}{\lambda_{v_\theta}}\right) \tag{8}$$

$$\frac{\partial \mathscr{H}}{\partial u} = 0 \quad , \quad u = \frac{-\lambda_m c_1}{c_2} + \frac{c_1(\lambda_{\dot{r}}\sin(\beta) + \lambda_{v_\theta}\cos(\beta))}{m} \quad , \quad 0 \le u \le 1 \tag{9}$$

The differential equations defining the co-states $\left(\dot{\lambda}_i = -\frac{\partial \mathscr{H}}{\partial i}\right)$ are given in Equations (10)–(14).

$$\dot{\lambda}_r = -\frac{2\lambda_{\dot{r}} GM}{r^3} + \frac{\lambda_{\dot{r}} v_\theta^2}{r^2} - \frac{\lambda_{v_\theta} \dot{r} v_\theta}{r^2} \tag{10}$$

$$\dot{\lambda}_\theta = 0 \tag{11}$$

$$\dot{\lambda}_{\dot{r}} = -\lambda_r + \frac{\lambda_{v_\theta} v_\theta}{r} \tag{12}$$

$$\dot{\lambda}_{v_\theta} = -\lambda_\theta - \frac{2\lambda_{\dot{r}} v_\theta}{r} + \frac{\lambda_{v_\theta} \dot{r}}{r} \tag{13}$$

$$\dot{\lambda}_m = \frac{\lambda_{\dot{r}} c_1 u \sin(\beta)}{m^2} + \frac{\lambda_{v_\theta} c_1 u \cos(\beta)}{m^2}, \tag{14}$$

which result in a TPBVP with the following boundary conditions at $t = 0$: r_0, θ_0, $\dot{r}_0$, $v_{\theta\,0}$, m_0 and r_{t_f}, $\lambda_{\theta_{t_f}}$, $\dot{r}_{t_f}$, $v_{\theta\,t_f}$, $\lambda_{m_{t_f}}$ at $t = t_f$. The TPBVP is solved using a direct shooting method, and derivatives are calculated using a finite difference method. Chapter 3 of Sood [28], has a good introduction to direct single and multiple shooting methods. A sample optimal orbit-to-orbit trajectory is shown in Fig. 2. The time of flight (TOF) for this trajectory is 1.1-years.

The thrust curve that produces this trajectory is shown in Fig. 3.

2.2. Two-body Mars-To-Earth transfer

While the boundary conditions of two-body orbit-to-orbit transfers cover a wide class of trajectories, another large class of transfers require the arrival at a specific point along the orbit at a precise time. The

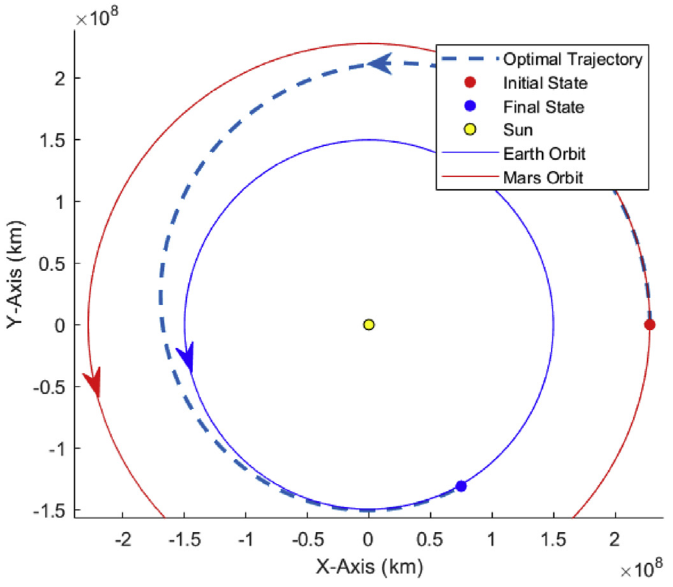

Fig. 2. Sample 1.1-Year orbit-to-orbit quadratic thrust optimal transfer trajectory.

boundary conditions are exemplified by interplanetary rendezvous trajectories, and are investigated in this section. The dynamical foundations for the OCP are similar to the orbit-to-orbit transfer, Equations (1)–(4), but with different boundary conditions and cost function. Instead of targeting an orbit, a moving point is being targeted whose location in polar coordinates can be specified as

$$\vec{X}_{target} = [a_{Earth}, (\dot{\theta}_{Earth} \times t) + \theta_0, 0, \dot{\theta}_{Earth} \times a_{Earth}]. \tag{15}$$

As such, Equation (15) corresponds to the final state of an interplanetary trajectory from Mars-To-Earth with a fixed time of flight. Additionally, the cost function being optimized is

$$J = \int_{t_0}^{t_f} u(t)dt, \tag{16}$$

which corresponds to a mass optimal trajectory. The final difference between this problem setup and the orbit-to-orbit transfer is the application of a direct method to solve the optimal control problem; meaning no co-state equations are needed. In order to place the OCP into the form of a nonlinear programming problem, the trajectory is discretized into 20, evenly spaced in time, nodes. At each node, a Hermite-Simpson collocation method is used to generate the defects in the dynamics, and a path constraint is introduced to ensure the thrust is bounded between 0 and 1. The initial and final states are formed as boundary conditions. Once properly discretized, the nonlinear programming problem is solved using MatLab's fmincon function, using sequential quadratic programming, and an optimality tolerance of 1e-4. This optimality tolerance is chosen as it provided a good trade-off

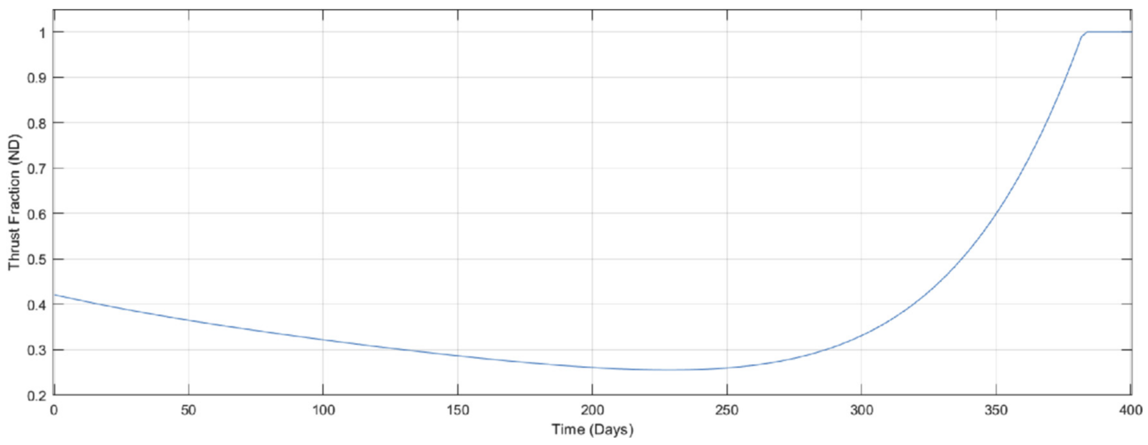

Fig. 3. Thrust curve.

between performance, and speed. Kelly [29] provides an excellent first introduction to direct collocation, and Topputo, and Zhang [30] is a good intermediate level reference for direct transcription methods.

2.3. Circular restricted three-body injection onto a Lyapunov Orbit

The final OCP is carried out within the framework of the Earth-Moon circular restricted three-body problem (CR3BP). The spacecraft mass is assumed to be negligible when compared to the mass of the Earth and the Moon, which are in turn assumed to move in circular orbits around their mutual barycenter. The following characteristic quantities are defined to non-dimensionalize the system:

$$l^\star = r_1 + r_2 \tag{17}$$

$$m^\star = m_1 + m_2 \tag{18}$$

$$t^\star = \sqrt{\frac{l^{\star 3}}{Gm^\star}} \tag{19}$$

where G is the universal gravitational constant. In the non-dimensional system, the period of the system is 2π, and the distance between the Earth and the Moon is now unity [31].

By converting the inertial reference frame to a rotating reference frame and further constraining the dynamical system to the x-y plane, a pseudo-potential function, Ω, can be expressed as

$$\Omega = \frac{1-\mu}{d} + \frac{\mu}{r} + \frac{1}{2}(x^2 + y^2) \tag{20}$$

where μ is the non-dimensional mass ratio and

$$d = \sqrt{(x+\mu)^2 + y^2} \tag{21}$$

$$r = \sqrt{(x-1+\mu)^2 + y^2}\,. \tag{22}$$

Spatial derivatives of the pseudo potential are expressed as

$$\frac{\partial \Omega}{\partial x} = \Omega_x \quad , \quad \frac{\partial^2 \Omega}{\partial x^2} = \Omega_{xx}. \tag{23}$$

As a result, the equations of motion can be re-written in a compact form as

$$\ddot{x} = 2\dot{y} + \Omega_x + \frac{t^{\star 2}}{l^\star}\frac{c_1}{m_3}u\sin(\beta) \tag{24}$$

$$\ddot{y} = -2\dot{x} + \Omega_y + \frac{t^{\star 2}}{l^\star}\frac{c_1}{m_3}u\cos(\beta) \tag{25}$$

where $\frac{t^{\star 2}}{l^\star}$ non-dimensionalizes the thrust, c_1 is the maximum thrust of the spacecraft, m_3 is the mass of the spacecraft, u is the thrust action bounded between [0,1], and β is the direction of thrust, defined in

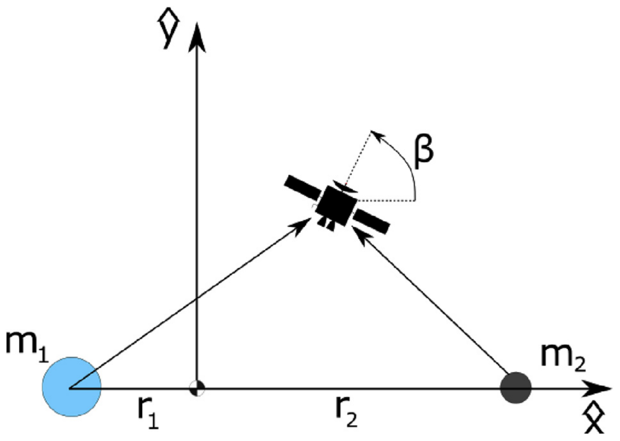

Fig. 4. Generic setup of the CR3BP in the rotating reference frame.

Table 2
System and spacecraft parameters.

μ	$l^\star$ (km)	$t^\star$ (s)	c_1 (N)	m_3 (kg)
0.01215	384400	375190	10	44,000

Fig. 4. For this investigation, values for the characteristic quantities and spacecraft parameters can be found in Table 2. The parameters are chosen to represent an Earth-Moon system, and a payload capable of being launched by a Falcon 9 Heavy [32] powered by 2 × 3 Xenon thrusters [33].

This model does not include rotational inertia, which means the spacecraft can instantaneously pitch in any direction in the plane of motion, and makes β a control input. Additionally, mass change is neglected in this problem, as the nominal time of flight for this optimal control problem is 2 non-dimensional time units, which corresponds to about 8.7 days. If the spacecraft were to apply max thrust continuously for the entirety of the trajectory, it would use up only 1% of the spacecraft's mass, and, as will be shown later in Fig. 11, the spacecraft only uses 1/20th of its max thrust. The cost function is

$$J = \int_0^{t_f} c_1^2 u^2 dt. \tag{26}$$

The quadratic nature of this cost function makes it easier for the optimal trajectory to converge. Alternatively, the function could be changed to a mass-optimal cost function using a homotopy similar to the one used in Sanchez et al. [15].

Using Pontryagin's minimization principal [26], the following Hamiltonian can be obtained

$$\mathscr{H} = \lambda_x \dot{x} + \lambda_y \dot{y} + \lambda_{\dot{x}} \left(2\dot{y} + \Omega_x + \frac{t^{\star 2}}{l^{\star}} \frac{c_1 u}{m_3} \sin(\beta)\right) + \lambda_{\dot{y}} \left(-2\dot{x} + \Omega_y + \frac{t^{\star 2}}{l^{\star}} \frac{c_1 u}{m_3} \cos(\beta)\right) + c_1^2 u^2. \tag{27}$$

Similar to the two-body orbit-to-orbit transfer, the optimal controller can be determined by differentiating the Hamiltonian with respect to the control variables as shown below.

$$\frac{\partial \mathscr{H}}{\partial \beta} = 0 \quad , \quad \beta = atan\left(\frac{\lambda_{\dot{x}}}{\lambda_{\dot{y}}}\right) \tag{28}$$

$$\frac{\partial \mathscr{H}}{\partial u} = 0 \quad , \quad u = -\frac{t^{\star 2}}{l^{\star}} \frac{c_1}{m_3} (\lambda_{\dot{x}} \sin(\beta) + \lambda_{\dot{y}} \cos(\beta)). \tag{29}$$

The differential equation defining the co-states $\left(\dot{\lambda}_i = -\frac{\partial \mathscr{H}}{\partial i}\right)$ gives the following relationships:

$$\dot{\lambda}_x = -\lambda_{\dot{x}} \Omega_{xx} - \lambda_{\dot{y}} \Omega_{xy} \tag{30}$$

$$\dot{\lambda}_y = -\lambda_{\dot{x}} \Omega_{xy} - \lambda_{\dot{y}} \Omega_{yy} \tag{31}$$

$$\dot{\lambda}_{\dot{x}} = -\lambda_x + 2\lambda_{\dot{y}} \tag{32}$$

$$\dot{\lambda}_{\dot{y}} = -\lambda_y - 2\lambda_{\dot{x}}, \tag{33}$$

which result in a two point boundary value problem (2PBVP) with the following boundary conditions at $t = 0$ $x_0, y_0, \dot{x}_0, \dot{y}_0$ and $x_{t_f}, y_{t_f}, \dot{x}_{t_f}, \dot{y}_{t_f}$ at $t = t_f$ with

$$\mathscr{H}(\mathbf{X}(t_f)) = 0. \tag{34}$$

A direct single shooting method is then used to solve the 2PBVP and derivatives are calculated through a first order finite difference method.

In this paper the targeted state, listed in Table 3, corresponds to a state on a Lyapunov orbit around the L_1 Lagrange point in the Earth-Moon system. This particular state is chosen because periodic orbits around Lagrange points are often a desired destination for spacecraft like the Wilkinson Microwave Anisotropy Probe [34] and the James Webb Space Telescope which has a planned orbit around the Sun-Earth L_2 Lagrange point [35].

3. Methods

3.1. Modeling missed thrust events

When spacecraft are outside expected operating conditions, they may autonomously retreat into a protective operational mode called *safe mode*. While in safe mode, the spacecraft shuts down all non-essential components, like propulsion and payload, to conserve its power, establish communications with Earth, become thermally safe, and ensure its survival. These safe mode events are unpredictable, and can be triggered by any number of scenarios, from software glitches to micrometeorite impacts [5]. If a safe mode event occurs while the spacecraft is thrusting, the engine prematurely shuts off, thus starting a missed thrust event (MTE). With high-thrust engines, where thrust arc durations are measured in seconds to minutes, the likelihood of a safe mode event overlapping with a thrust arc are minuscule and can safely be ignored in the initial design phase [8] or automated safe mode responses can be turned off during the brief thrusting period [9]. In contrast, spacecraft using low-thrust propulsion may have thrust arcs that are months to years long. In such cases, the overlap between a thrust arc and a safe mode event is not only likely, but nearly certain for long duration transfers [10].

Table 3
Target state.

x (ND)	y (ND)	$\dot{x}$ (ND)	$\dot{y}$ (ND)
0.82847	0	0	0.07558

Table 4
Weibull Parameters for Safe Mode Events from Imken et al. [10].

	Scale (Λ)	Shape (k)
Time between MTEs (days)	230.5	0.87
Duration of MTE (days)	2.41	1.17

The occurrence and duration of safe mode events are inherently random, but, by drawing on a database of deep space missions, Imken et al. (2018) determined the probability distributions of both the start of a safe mode event as well as its duration [10]. Both conform to two different Weibull distributions whose parameters can be found in Table 4. These two parameters, Λ and k, adjust the scale and shape of the Weibull distribution respectively. The shape parameter determines the rate of change of the instantaneous likelihood. A value of less than *one* indicates that the instantaneous failure rate decreases with an increase in the duration of the trajectory. Conversely, a shape value greater than *one* indicates that the instantaneous failure rate increases over a particular trajectory. If the shape is held constant, increasing the scale parameter increases the distribution's mean and standard deviation. Hence, as the time since the last MTE increases, the likelihood of another MTE decreases.

The probability density function for a Weibull distribution is shown in Equation (35), where ξ is one possible input, and $f(\xi)$ gives the probability of that input occurring.

$$f(\xi) = \begin{cases} \frac{k}{\Lambda}\left(\frac{\xi}{\Lambda}\right)^{k-1} e^{-(\xi/\Lambda)^k} & \xi \geq 0 \\ 0 & \xi < 0 \end{cases} \tag{35}$$

The process for simulating a trajectory with MTEs is detailed in Algorithm 1. Here, an optimal trajectory is computed and simulated until a MTE occurs. The start of the MTE is demarcated by mte_{start}, while its duration is demarcated as $mte_{duration}$. After the MTE, and its corresponding ballistic component, the optimal control problem is again solved using the spacecraft's current state as the updated initial conditions, IC. Additionally, at this step, the time and duration of the next MTE is calculated. Both are evaluated by drawing a random number from the appropriate Weibull distribution. If the start of the next MTE is after the final time of the trajectory, t_f, the system is only integrated until the final time, satisfying that the end conditions are met.

Algorithm 1 Simulating MTEs

```
fullTrajectory = []
$t = 0$
while $t < t_f$ do
  if $mte_{start} < t$ then
    $mte_{start}, mte_{duration}$ = generateNewMTE ()
  end if
  if $t + mte_{start} \geq t_f$ then
    Solve OCP with IC, no MTE = trajectoryLeg
    Add trajectoryLeg to fullTrajectory
    $t = t_f$
  end if
  Solve OCP with IC & t = TrajectoryLeg
  Add TrajectoryLeg to fullTrajectory
  IC = TrajectoryLeg $(t + mte_{start})$
  $t = t + mte_{start}$
  Ballistically propagate IC from $t$ to $t + mte_{duration}$ = TrajectoryLeg
  IC = TrajectoryLeg $(t + mte_{duration})$
  $t = t + mte_{duration}$
  Add TrajectoryLeg to fullTrajectory
end while
return fullTrajectory
```

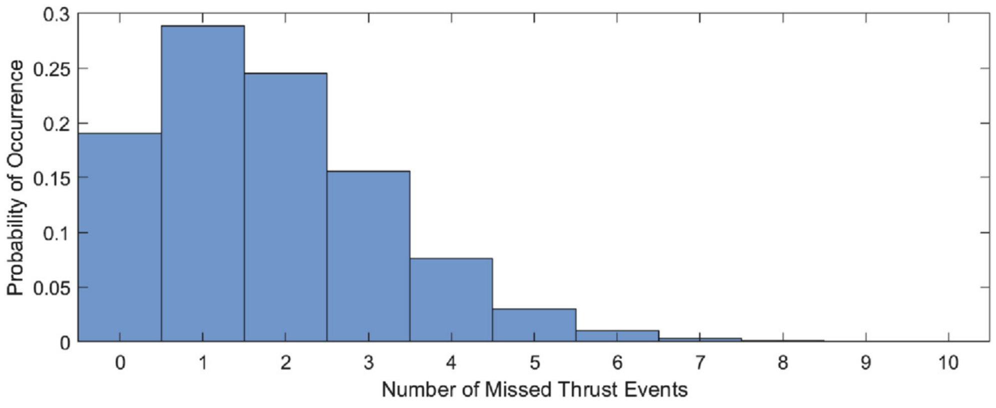

Fig. 5. Simulation displaying the likelihood of multiple MTEs over a 1.1-Year thrusting arc.

Algorithm 1 allows multiple MTEs to be simulated for each trajectory. By running 1,000,000 Monte Carlo simulations, it is apparent that for a low-thrust spacecraft on a 1.1-year-long trajectory, where the spacecraft thrusts continuously, *one* MTE is the nominal case and having a trajectory with *no* MTEs would be anomalous, as shown in Fig. 5. Additionally, for trajectories with the same 1.1-year time of flight and continuous thrust, it is evident that the spacecraft is more likely to have *two* MTEs than *no* MTEs.

Cumulatively, while trajectories with either *zero* or *one* MTE make up a large amount of trajectories encountered, the likelihood of *two* or more MTEs make up the majority of scenarios. For trajectories longer than a year, any technique for mitigating MTEs must be able to deal with multiple MTEs per trajectory. Using a similar Monte Carlo approach, it is determined that trajectories with thrust arcs longer than 0.68 years and 1.3 years have *one* and *two* expected MTEs, respectively.

Note, that under the continuous thrust assumption, the expected number of MTEs encountered in a trajectory is purely a function of the time of flight and is independent of the trajectory itself. A 1.1-year Mars-To-Earth transfer trajectory will have the same expected number of MTEs as a 1.1-year Earth-To Moon transfer trajectory or the first 1.1-year segment of an Earth-To-Jupiter transfer trajectory. Additionally, all calculations in this section assume that the spacecraft is thrusting over the entire trajectory. This assumption holds for trajectories that minimize quadratic thrust, but often falls apart for mass optimal trajectories where the controller more closely resembles a bang-bang controller. Moving from ideal trajectories to actual mission designs the constant thrust assumption still holds in certain cases. In one reference design for the Earth Return Orbiter, the outbound Earth-To-Mars rendezvous trajectory thrusts for approximately 85% of the trajectory [4]. Conversely, on the Mars-To-Earth return trajectory, the spacecraft is on a flyby trajectory, resulting in a thrust fraction of approximately 70% [4]. Another low thrust fraction case is the Bepi-Colombo mission, which thrusts for less than half its total trajectory due to 11 gravity assists [11]. The Dawn mission, on its trajectory from Vesta-To-Ceres, thrusted for 93% of its trajectory until a MTE occurred 19 days before arrival [9]. Additionally, Dawn's Ceres approach phase had a thrust fraction of around 88% [9]. In summary, the constant thrust assumption is well founded for low-thrust trajectories that rendezvous with the target. This assumption degrades when the spacecraft has a flyby trajectory or uses multiple gravity assists, which makes the constant thrust assumption a conservative estimate. Finally, the constant thrust assumption is solely used in the generation of Fig. 5. For the remainder of this paper, the duration and timing of thrust arcs is determined by the particular optimal control solver used in that scenario to better match specific mission conditions.

3.2. Neural network design

Neural networks have been investigated by a number of researchers with varying applications [36,37]. A neuron is generally a function that has a single input, a configurable bias that offsets the input, and a single output. These functions are usually simple and nonlinear, like the hyperbolic tangent function (Tanh). Tanh functions were some of the earliest functions used in neural networks because, as illustrated in Fig. 6, they are smooth and, despite having an infinite domain, have a bounded range.

Rectified Linear Unit (ReLU) functions, shown in Fig. 7, are another popular choice for neurons. They are governed by the following piecewise equation and often provide better performance for neural networks during training [38].

$$Relu(\zeta) = \begin{cases} 0 & \zeta \le 0 \\ \zeta & 0 \le \zeta \end{cases} \tag{36}$$

As the name neural network implies, several neurons are linked together such that the output of one set of neurons acts as the input to the next set of neurons. Each set of neurons is called a layer. A scalar gain, K_{ij}, is placed between each neuron output and neuron input. A simple schematic for a two-layer neural network is shown in Fig. 8.

For simplicity, all neurons in this example are of the same function. However, a general form with different functions may be selected. In Fig. 8, there are 5 Tanh () neurons, connected by 6 different gains, K_{ij}. As the system grows in layers and neurons, the complexity of the network grows exponentially. The complex interaction of the nonlinear functions allows the network to describe complex systems. In the language of systems engineering, the network has emergence. The final step in completing the neural network is to compute the individual biases and gains, K_{ij}. This process begins by providing an input to the neural network and comparing the network's output with the desired output. The difference is used to update the biases and gains in the network through a process known as *backpropagation* [38]. By providing enough input/output pairs, the neural network is expected to *learn* the proper biases and gains.

All the neural networks used in this investigation have 5 layers (4

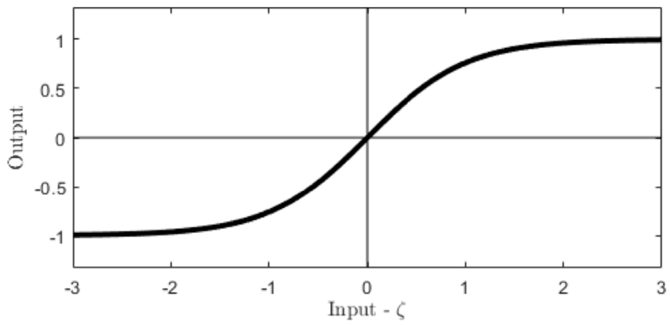

Fig. 6. Tanh function with 0 bias.

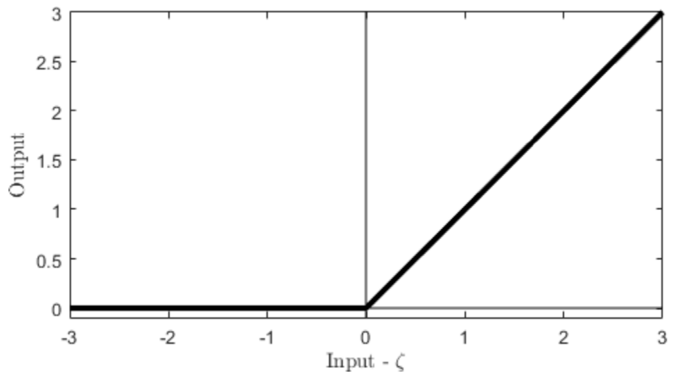

Fig. 7. ReLU function with 0 bias.

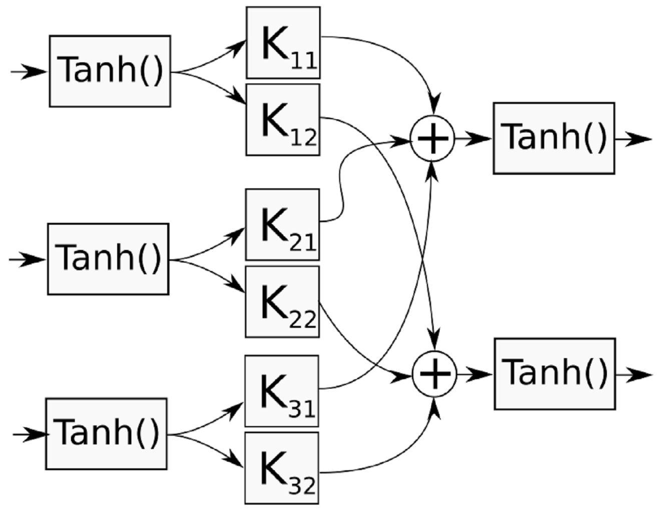

Fig. 8. A simple neural network with 5 tanh () neurons, connected by 6 scalar gains.K_{ij}

hidden, 1 output) with 32 neurons per hidden layer, for a total of 130 neurons, and 3394 trainable parameters. Larger models have already been implemented on a VerTex-5QV FPGA co-processor onboard the MARS 2020 mission [24]. The function types for the hidden layers in this work are ReLUs, and Tanh functions are chosen for the output layer, because, like the control variables, Tanh has a bounded range. The neural networks are implemented using Keras with a TensorFlow back end, and employ an Adam method optimizer [39]. The networks are trained with a batch size of 40,000, and a patience of 50. A patience of 50 indicates that the training process is terminated after the neural network is unable to improve its performance over 50 consecutive training iterations. These hyperparameters and network structure are chosen based on empirical testing and used on all optimal control problems in this paper. While the hyperparameters and network structure are the same on all 3 problems, each network is re-trained from a scratch using new initialization biases and gains for each problem.

3.3. *Why deep neural networks work*

Any optimal trajectory can be broken into a set of optimal states, $\boldsymbol{X}$, and the corresponding optimal actions, $\boldsymbol{U}^*$, taken at the state [26]. Conversely, if at every state, the corresponding optimal action is taken, then an optimal trajectory will be formed. Thus, there exists a function $f()$ that maps each state to its corresponding optimal action as given by Equation (37).

$$f(\boldsymbol{X}) = \boldsymbol{U}^* \tag{37}$$

Table 5
Upper and lower bounds for initial conditions in the two-body orbit-to-orbit transfer optimal control problem.

	r_0 (km)	$\dot{r}_0$ (km/s)	$v_{\theta\,0}$ (rad/s)	m_0 (kg)
Upper	2.3933e8	1	25.3353	3366
Lower	2.2566e8	− 1	24.0082	3366

Determining the function $f()$ is only achievable in a few trivial examples, and often can't be expressed in elementary functions. Fortunately, neural networks are general purpose function approximators. Previous work has shown that a neural network with an infinite number of neurons can, under some relatively benign conditions, reproduce any function [40]. In this work, a finite number of neurons are used to approximate $f()$ over a range of conditions that a spacecraft may encounter. Because the spacecraft's state after a missed thrust event is likely to be in the area where the neural network is a close approximation of $f()$, it can begin to create a near-optimal trajectory for the spacecraft towards the desired target.

3.4. *Dataset 1: Optimal trajectories*

Before a neural network can be trained using supervised learning, a dataset is needed. By varying the initial conditions, within the bounds specified in Tables 5–7, and solving the corresponding optimal control problem (OCP), a dataset of 150,000 trajectories is created for each OCP. The bounds for the initial condition found in Table 7, are chosen to represent a spacecraft perturbed off the stable manifold of the Lyapunov orbit being targeted [28]. Each trajectory is discretized into 100 optimal state-action pairs, where a pair consists of the spacecraft's instantaneous state and the corresponding control action. As a result, a dataset of 15,000,000 optimal state-action pairs is generated.

The dataset is then post-processed by subtracting off that states mean and dividing that states standard deviation. Finally, the dataset is shuffled and divided into two sets. The first set contains 85% of the state-action pairs and is used to train the neural network. Whereas, the second contains the remaining 15% of the state-action pairs and is used to validate the neural network. These post-processing steps are introduced to the work-flow to improve neural network performance and decrease over-fitting.

3.5. *Dataset 2: Optimal trajectories with MTEs*

One question that naturally arises from the preliminary work is, *can performance be boosted by including MTEs in the training dataset?* The neural network, as described in this paper, can only “learn” from the datasets it has been trained on, and the optimal trajectories in dataset 1 only contains trajectories without MTEs. In order to test the potential performance improvement, a new dataset of 150,000 optimal trajectories is created in the two-body Orbit-To-Orbit transfer, and is called dataset 2. Unlike the original dataset, dataset 2 has a single MTE, randomly distributed in time, during each trajectory. After the MTE is completed, the optimal control problem is re-solved, with the new post MTE state. For example, a trajectory in dataset 2, has a 3 day MTE occur 125 days into the trajectory. After the MTE occurs, the trajectory is re-optimized and the new state-action pairs are added to dataset 2. Dataset 2 is post-processed in the same way as the first dataset.

Table 6
Upper and lower bounds for initial conditions in the two-body Mars-to-Earth transfer optimal control problem.

	r_0 (km)	$\dot{r}_0$ (km/s)	$v_{\theta\,0}$ (rad/s)	m_0 (kg)	TOF (Days)
Upper	2.2839e8	0.01	24.3702	3366	405
Lower	2.2748e8	− 0.01	23.8876	3366	398

Table 7
Upper and lower bounds for initial conditions in the circular restricted three body transfer optimal control problem.

	x_0 (km)	y_0 (km)	$\dot{x}_0$ (km/s)	$\dot{y}_0$ (km/s)
Upper	1.862e5	−7.3151e4	0.8406	0.4324
Lower	1.8605e5	−7.3459e4	0.8379	0.4291

By combining trajectories from datasets 1 and 2, *three* new datasets, each consisting of 150,000 trajectories are created. Each dataset has a different fraction of trajectories with a MTE. The fraction of trajectories with MTEs in these combined dataset varies between 1% and 50%. For example, in the new 20% dataset, 120,000 trajectories are taken from the dataset with no MTEs and combined with 35,000 trajectories taken from the dataset with MTEs. Each new dataset is randomized and divided into a training and validation set.

3.6. Neural network driven trajectories and performance metrics

Neural network driven trajectories (NNDT) are generated by integrating the system dynamics and allowing the neural network to control the spacecraft through the integration process. The integration is carried out using a Livermore solver [41], with a maximum error of 10^{-12} and the states are stored at time steps which ensured 1000 samples per trajectory.

Three metrics are adopted to determine how well the neural network is learning the optimal control solution. The first metric used to evaluate the performance is the *success rate*. The trajectory is classified as a success if the minimum miss distance (MMD) is within a certain tolerance. The MMD is defined as the state closest to the target using the 2-norm. For the two-body orbit-to-orbit transfer, a MMD of less than 1% the range of the difference between Mars' orbit and Earth's orbit is considered to be a success. For the Mars-To-Earth optimal control problem (OCP), the spacecraft's MMD is required to be within 1 Earth sphere of influence (SOI) and for the V-inf to be less than 4.5 km/s, representing the maximum allowable V-inf for re-entry in the MSR mission profile [4]. Within the CR3BP, for insertion onto the Lyapunov orbit, a MMD less than 2% of the range of the states is chosen to ensure an arrival velocity difference of less than 1 m/s.

The next two metrics used to evaluate the NNDT are *optimality error* and *lateness*. Optimality error, is computed as the percent difference between the neural network's final cost function, (for successful trajectories) and the final cost function from an optimal trajectory, J^*, as expressed in Equation (38).

$$J\%_{diff} = \frac{2|J^* - J_{nn}|}{J^* + J_{nn}} \times 100\% \tag{38}$$

Lateness is computed by calculating the time difference between the neural network's closest approach to the target verses the nominal 1.1-year trajectory. In this metric, a negative result indicates that the NNDT arrived at the target in less than 1.1-years.

4. Results

4.1. Baseline performance of the neural network

In each of the 3 optimal control problems (orbit-to-orbit, Mars-To-Earth, CR3BP), 10,000 initial conditions are uniformly sampled from the respective initialization areas, Tables 5–7, and propagated forward in time using a neural network as a controller. All initial conditions within the bounds would result in a failed trajectory if no control was applied. However, with a neural network based controller, detailed results in Table 8 show a baseline success rate of at least 97%. Additionally, the average difference between the optimal final cost and the neural network driven final cost is less than 3% in all cases. Promising results are found in the two-body orbit-to-orbit optimal control problem (OCP). However, the results in the two-body Mars-To-Earth OCP are more limited. The behavior can be explained by the more complex boundary conditions and cost function in the Mars-To-Earth transfer.

Table 8
Baseline performance of the neural network.

	Success %	Optimality Error (SD)	Lateness Days (SD)
Orbit-to-Orbit	99	1.01 (1)	−1.4 (1.7)
Mars-to-Earth	97	2.9 (2.4)	−9.1 (1.4)
CR3BP	98	1.4 (1.1)	.85 (0.039)

An observation of the lateness parameter in Table 8 indicates that the neural networks, on average, arrived earlier than the nominal TOF. While their average early arrival is interesting, it should be noted that even the average 9 day early arrival in the Mars-To-Earth trajectory is only 2% of the total 1.1 year TOF. By plotting the thrust magnitude and angle curves from the neural network driven trajectory against the optimal trajectory in the CR3BP, deeper insights to the overall performance of the neural network can be gained. Figs. 9 and 10 show both sets of controls to be quite similar. The neural network curves are not as smooth as the optimal solutions, and occasionally deviate from the optimal solutions.

This tendency of the neural network driven trajectory (NNDT) to closely mimic the optimal trajectory is found in the two-body orbital transfer OCP as well. Qualitatively the neural network produces a control that is similar to the optimal trajectory, but not as smooth. In Fig. 11, the velocity difference between the neural network trajectory and the optimal trajectory corresponds to the regions where the thrust curves differ. This indicates that the differences in control inputs may be due to the differences in the spacecraft's state. While the difference is small, on the order of 10^{-4}, it may be enough to cause the divergence in control strategy.

The optimal thrust curve for the Mars-To-Earth trajectory is qualitatively different from the other two OCP because of the mass optimal cost function. Shown in Fig. 12, the bang-bang structure is readily apparent. Here, the thrust curve for the neural network driven trajectories (NNDT) lags behind the optimal thrust curve. Additionally, the optimal thrust curve ramps down towards the end, while the NNDT thrust curve continues thrusting until arrival. This difference may, in part, be due to differences in the spacecraft's state caused by the lagging control in the initial portion of the trajectory.

4.2. Sensitivities of neural networks to MTE's

To evaluate the resiliency of the neural network to MTEs, 10,000 new NNDT are computed for each OCP. A single nominal initial condition is chosen in each scenario which ensured any fluctuation in results are solely due to the stochasticity of MTEs. For both two-body orbit-to-orbit and Mars-To-Earth OCPs, the initial conditions are set as $r_0 = a_{mars}$, $\theta_0 = 0$ (radians), $\dot{r}_0 = 0$, $v_{\theta\,0} = v_{\theta\,Mars}$, with a mass of 3366 kg. Each trajectory had a single MTE that is uniformly temporally distributed. This step allowed for evaluating the sensitivity of MTEs over different portions of each trajectory.

By plotting the success rates of the neural network with a single MTE, shown in Figs. 13–15, the sensitivity of the neural network to a MTE along the trajectory is apparent. Two features are evident from Fig. 13, which investigates sensitivity in the two-body orbit-to-orbit transfer. First, the trajectories are more sensitive to a MTE in the beginning of the transfer. As the MTE moves further from the initial state, the trajectories slowly become less sensitive to the effects of a potential MTE. Second, there is an increase in the sensitivity $3/4^{th}$ of the way into the trajectory (270 days), as evident by a slight decrease in the success state rate.

Comparing Fig. 13 to the nominal thrust curve for the system, Fig. 3,

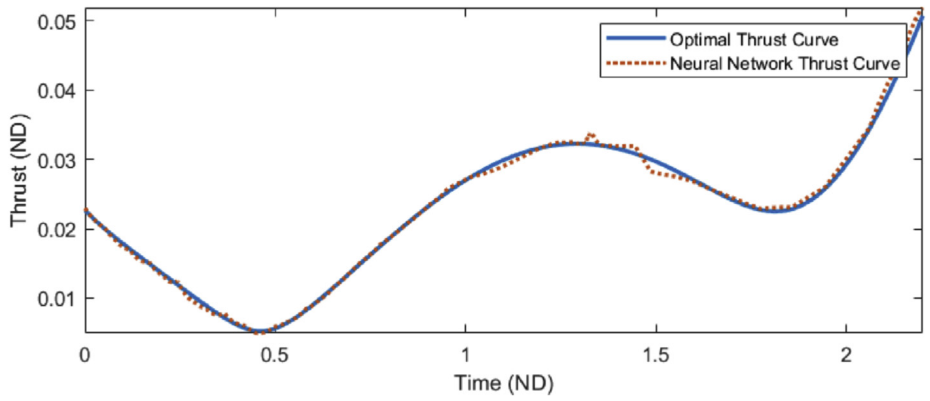

Fig. 9. Thrust curve comparison - CR3BP.

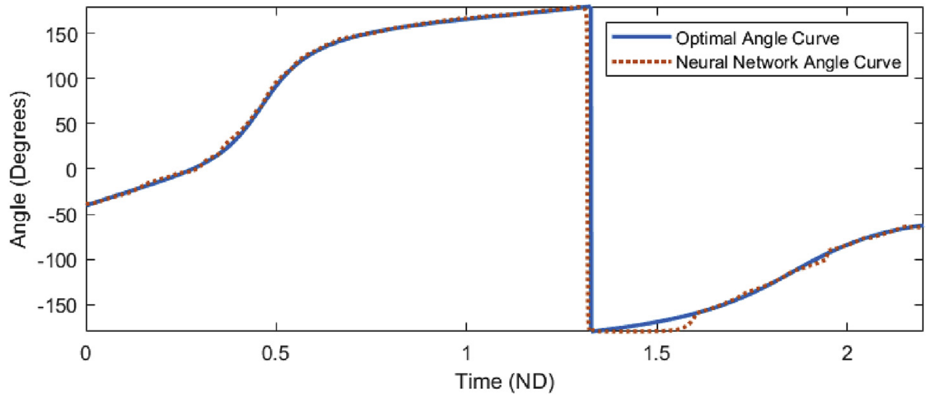

Fig. 10. Angle curve comparison - CR3BP.

the trajectories that can tolerate the longest MTEs, around 200 days into the trajectory, correspond well to the region with the lowest thrust fraction. As the nominal thrust fraction begins to increase after this point, trajectories are more susceptible to long duration MTEs. This fits well with the suggestion that as a spacecraft has more available thrust, it is better able to compensate for MTEs.

One feature that is not readily apparent from Fig. 13 is the distribution of the duration of the MTEs. Because of the Weibull distribution, about 60% of MTEs would last 3 days or less, and about 90% of MTEs are 8 days or less [10]. As a consequence, 60% of MTEs are inside the bottom sixth of the figure, demarcated by a green horizontal line, below which, all the trajectories are successful. Additionally, 90% of the trajectories will be successful if they arrive at the 180 day mark without a MTE. After approximately 320 days into the trajectory, the neural network becomes much more resilient to MTEs. This behavior can be attributed to the fact that the spacecraft is close enough to the desired target that any deviation from a lack of maneuvering may not be significant enough to keep the trajectory outside the arrival bounds. If the arrival bounds are tightened, this region should correspondingly shrink in size.

Repeating this procedure for the Mars-To-Earth orbit transfer, it can be seen in Fig. 14 that different portions of the trajectory are now sensitive. Because this OCP uses a mass optimal cost function, the strategy is close to a bang-bang controller, see Fig. 12. The failure regions are grouped in two large stalactites (red) that have a near overlap with the thruster being turned on. Other large areas of success correspond to when the thruster is off. Once again, the green line demarcates a MTE duration of 3-days.

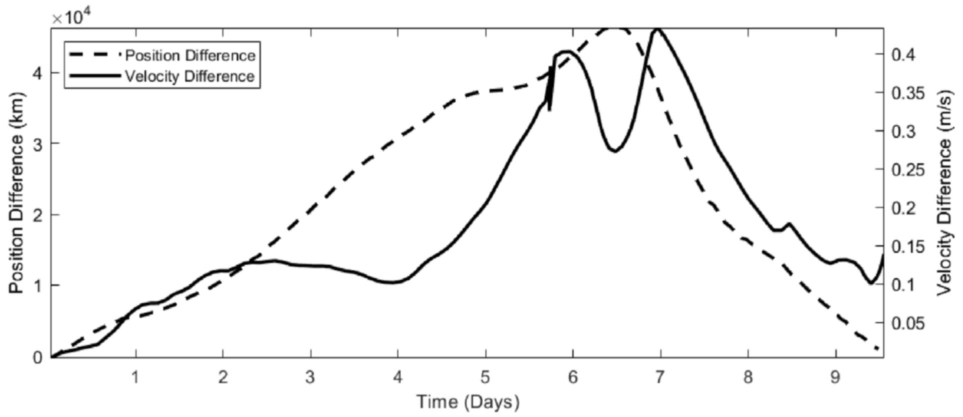

Fig. 11. Absolute position (dashed) and velocity (solid) error - CR3BP.

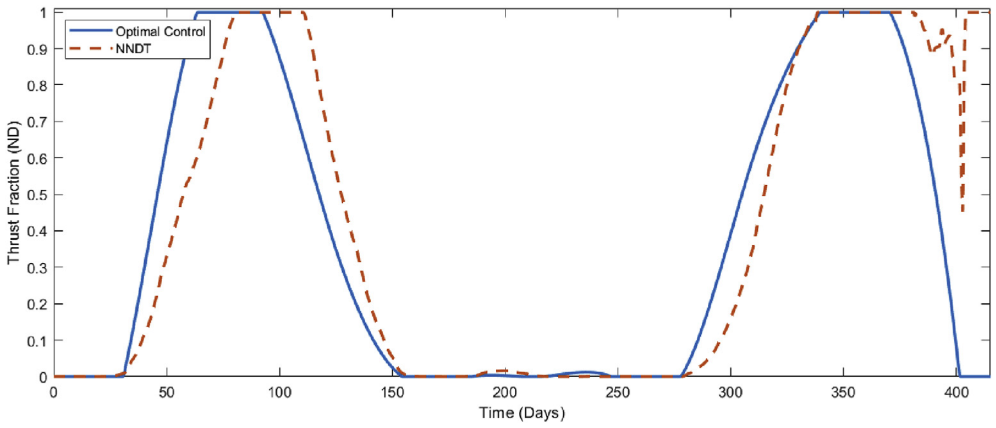

Fig. 12. Thrust curve comparison - Mars-to-Earth transfer.

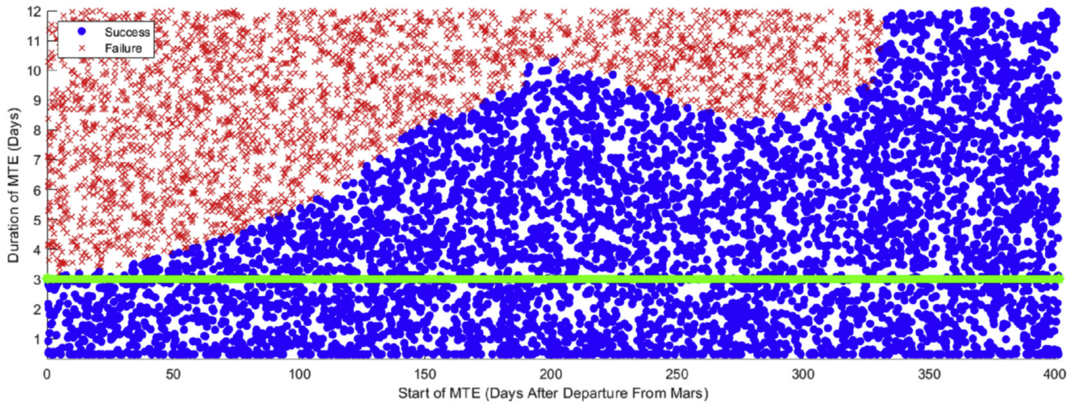

Fig. 13. Sensitivity to single MTEs for the two-body orbit-to-orbit transfer.

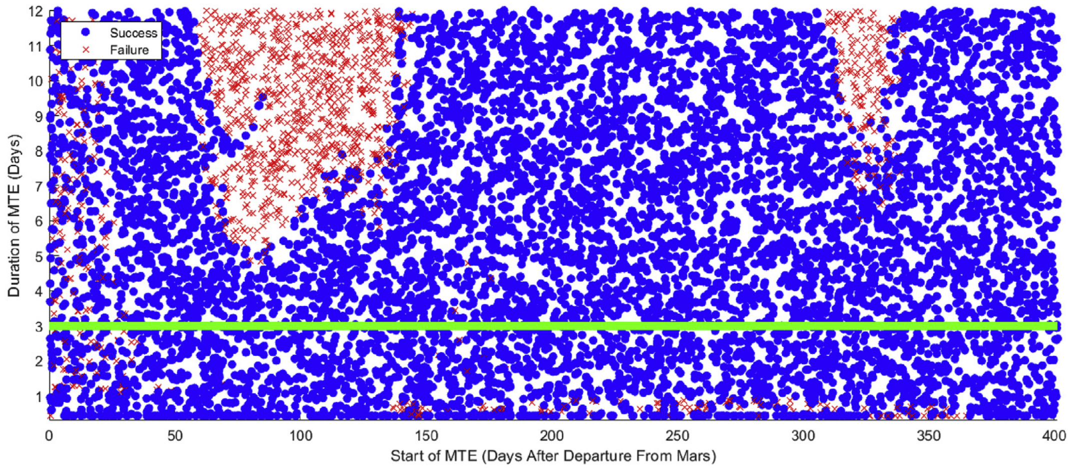

Fig. 14. Sensitivity to single MTEs for the two-body Mars-to-Earth transfer.

Repeating this procedure within the Earth-Moon CR3BP, to insert a spacecraft onto a Lyapunov orbit, allows for the exploration of sensitivity on a much shorter time scale. In Fig. 15, sensitivity of a ~13 day trajectory is displayed. MTEs with a duration longer than the remaining time in the trajectory are discarded from the figure and the beginning of a coast period is demarcated by a black line. Once again, the terminal portion of the trajectory is much more resilient to MTEs than any MTE in the earlier portions of the trajectory. It is apparent from Fig. 15 that the spacecraft is much more susceptible to MTEs from 3 to 6 days after the start of the trajectory than in any of the other regions. Here, within the 3–6 days time, any single MTE with a duration of more than a day would cause a failed trajectory.

Similarly to the Orbit-to-Orbit case, the length of the MTE that can be tolerated in the CR3BP is inversely proportional to the thrust fraction at that point. The nominal thrust curve for the CR3BP case, Fig. 9, has a initial dip at the onset of the trajectory, which is closely mirrored in Fig. 15, by a jump in the length of MTEs that the spacecraft can tolerate.

4.3. Neural network performance with multiple MTEs

Next, for each OCP, a second dataset consisting of 10,000 trajectories is constructed. These datasets have multiple MTEs per trajectory,

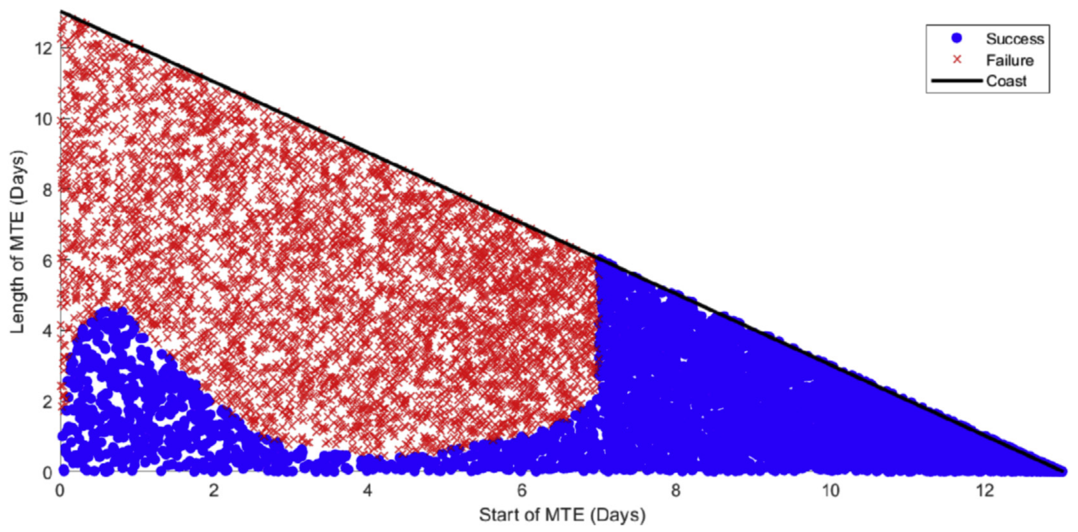

Fig. 15. Sensitivity to single MTEs for the CR3BP insertion onto a lyapunov orbit.

Table 9
Performance of the neural network with multiple MTEs.

	Success %	Optimality Error % (SD)	Lateness Days (SD)
Orbit-to-Orbit	79	7.6 (7.3)	5.2 (6.1)
Mars-To-Earth	77	9.8 (8.9)	−14 (15)
CR3BP	76	5.2 (4.4)	.78 (.12)

Table 10
Boosted networks on trajectories with No MTE.

Percentage of training dataset with MTE	0%	5%	20%	50%
Success Rate	99	99	98	95
Optimality Error	1.0	1.7	1.2	2.9

Table 11
Boosted network performance on trajectories with multiple MTEs.

Percentage MTE in dataset	0%	5%	20%	50%
Success Rate	79	77	77	77
Optimality Error (Average)	7.6	2.8	2.5	2.3
Optimality Error (SD)	7.3	2.6	1.2	1.6
Lateness (SD)	6.1	3.4	1.4	1.4

thus simulating likely scenarios during a long duration trajectory [10]. Results can be seen in Table 9, that indicate promising solutions. Here, although the success rates decreased, the percent difference in optimality and average lateness both increased. These changes are expected because, the more MTEs that occur over a trajectory, the less average control authority the spacecraft has, which results in both a lower success rate and an overall less optimal trajectory.

4.4. Boosted neural network performance with MTEs: Orbit-to-orbit

For the two-body orbit-to-orbit OCP, incorporating trajectories with MTEs into the training dataset has various effects on the performance of the neural network. As the percentage of trajectories corrected after MTEs within the training dataset is increased from 0% to 50%, listed in Table 10, the success rate for trajectories with no MTEs falls from 99% to 95%, while the difference in optimality increases.

The results become more interesting once the boosted neural networks are used on trajectories with multiple MTEs. As shown in Table 11, while the success rate still decreases by 2%, the average optimality error dramatically improves. Additionally, the standard deviation for the optimality error and the lateness also tighten.

If the baseline success rates are acceptable, including re-optimized trajectories in the training dataset may prove a valuable tool in improving *optimality error* and *lateness*.

5. Conclusion and future work

This work investigates how relatively small deep neural networks can be used as near-optimal controller over different astrodynamic regimes, from simple two-body orbit transfers to the CR3BP with a low-thrust spacecraft. Additionally, it is shown that these neural networks can autonomously correct for a majority of missed thrust events, a concern that hampers the widespread adoption of electric propulsion methods. The performance of these neural networks is encouraging, opens up new possibilities with respect to spacecraft autonomy, and highlights the resiliency of neural network-based control to unexpected events. Because the neural networks are recovering from a majority of MTEs and designing new near-optimal trajectories, the results are promising and suggest that the network is learning the local solution to the Hamilton-Jacobi-Bellman equations underlying each optimal control problem. Although, this method can be used by ground teams to rapidly recalculate spacecraft trajectories after MTEs, the method has shown true potential when spacecraft are able to autonomously depart safe mode. Similar to how recovery shepherding helps enable fault tolerant computing by repairing the application after a fault occurs [42], neural network optimal control delivers MTE tolerance to spacecraft by repairing the trajectory after a MTE occurs. To further build the methodology for autonomous spacecraft operation, future work will explore increasing the fidelity of the training model, exploring performance under uncertain state information, and developing a baseline metric that can be used to compare different autonomous responses to MTEs.

Declaration of competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] C. Hardgrove, J. Bell, J. Thangavelautham, A. Klesh, R. Starr, T. Colaprete, M. Robinson, D. Drake, E. John-Son, J. Christian, A. Genova, D. Dunham,

B. Williams, D. Nelson, A. Babuscia, P. Scowen, K.M. Cheung, T. Mckinney, A. Taits, V. Hernandez, P. Wren, A. Thoesen, A. Godber, M. Beasley, The lunar polar hydrogen mapper (Lunah-Map) mission: mapping hydrogen distributions in permanently shadowed regions of the moon's south Pole, Lunar Exploration Analysis Group (2015) 1–2 No. January.

[2] D.Y. Oh, S. Collins, D. Goebel, B. Hart, G. Lantoine, S. Snyder, G. Whiffen, L. Elkins-tanton, P. Lord, Z. Pirkl, L. Rotlisburger, Development of the Psyche mission for NASA's discovery program, 35th International Electric Propulsion Conference, 2017, pp. 1–19.

[3] F. Laipert, A. Nicholas, R. Woolley, R. Lock, AAS 19-345 hybrid chemical-electric trajectories for a Mars sample return orbiter, *Space Flight Mechanics Meeting*, No. 818, 2019, pp. 1–8.

[4] A.K. Nicholas, A. Didion, F. Laipert, Z. Olikara, R.C. Woolley, R. Lock, J. Huesing, Mission analysis for A potential Mars sample return campaign in the 2020's, Space Flight Mechanics Meeting, 2019, pp. 1–20.

[5] B. Cooke, Meteoroid-Induced Anomalies on Spacecraft, (2015).

[6] R. Sood, K.C. Howell, Solar sail transfers and trajectory design to sun-earth L4, L5: solar observations and potential Earth trojan exploration, Adv. Astronaut. Sci. 158 (2019) 1605–1624, https://doi.org/10.1007/s40295-018-00141-4.

[7] D. Guzzetti, R. Sood, L. Chappaz, H. Baoyin, Stationkeeping analysis for solar sailing the L4 region of binary asteroid systems, J. Guid. Contr. Dynam. (2019) 1–13, https://doi.org/10.2514/1.g003994.

[8] F.E. Laipert, T. Imken, A Monte Carlo approach to measuring trajectory performance subject to missed thrust, Space Flight Mechanics Meeting (2018), https://doi.org/10.2514/6.2018-0966.

[9] M.D. Rayman, Dawn at Ceres: the first exploration of the first dwarf planet discovered, Acta Astronaut. (2019), https://doi.org/10.1016/j.actaastro.2019.12.017.

[10] T. Imken, T. Randolph, M. DInicola, A. Nicholas, Modeling spacecraft safe mode events, IEEE Aerospace Conference Proceedings, Vol. 2018-March, 2018, pp. 1–13, , https://doi.org/10.1109/AERO.2018.8396383.

[11] P. Muñoz, Missed-thrust analysis of BepiColombo's interplanetary transfer to mercury orbit, Adv. Astronaut. Sci. (2019).

[12] D.Y. Oh, D. Landau, T. Randolph, P. Timmerman, J. Chase, J. Sims, T. Kowalkowski, Analysis of system margins on deep space missions utilizing solar electric propulsion, 44th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, 2008, https://doi.org/10.2514/6.2008-5286.

[13] F.E. Laipert, J.M. Longuski, Automated missed-thrust propellant margin analysis for low-thrust trajectories, J. Spacecraft Rockets 52 (No. 4) (2015), https://doi.org/10.2514/1.A33264.

[14] J. M. S. Pérez and G. I. Varga, "Missed Thrust Analysis for A Potential Mars Sample Return Orbiter," pp. 1–18.

[15] C. Sánchez-Sánchez, D. Izzo, Real-time optimal control via deep neural networks: study on landing problems, J. Guid. Contr. Dynam. 41 (No. 5) (2018), https://doi.org/10.1023/B:CJOP.0000010527.13037.22.

[16] NASA, NASA Technology Roadmaps TA 5: Communications, Navigation, and Orbital Debris Tracking and Characterization Systems, *NASA Technology Roadmaps*, 2015.

[17] NASA, Strategic Technology Investment Plan National Aeronautics and Space Administration, (2017).

[18] B. Dachwald, Evolutionary neurocontrol: a smart method for global optimization of low-thrust trajectories, Collection of Technical Papers - AIAA/AAS Astrodynamics Specialist Conference, vol. 3, 2004, pp. 1705–1720, , https://doi.org/10.2514/6.2004-5405 No. May.

[19] A. Harris, T. Teil, H. Schaub, Spacecraft Decision-Making Autonomy Using Deep Reinforcement Learning, (2019), pp. 1–19.

[20] J. Broida, R. Linares, Spacecraft rendezvous guidance in cluttered environments via reinforcement learning, Space Flight Mechanics Meeting, 2019, pp. 1–12.

[21] G. Hartman, Z. Shiller, A. Azaria, Deep Reinforcement Learning for Time Optimal Velocity Control Using Prior Knowledge, (2018).

[22] D. Izzo, C. Sprague, D. Tailor, Machine learning and evolutionary techniques in interplanetary trajectory design, Modeling and Optimization in Space Engineering (2019) 191–210.

[23] R. Furfaro, I. Bloise, M. Orlandelli, P. Di Lizia, F. Topputo, R. Linares, Deep learning for autonomous lunar landing, Adv. Astronaut. Sci. 167 (2018) 3285–3306.

[24] N.L. Parrish, D.J. Scheeres, Optimal low-thrust trajectory correction with neural networks, Adv. Astronaut. Sci. 167 (2018) 1483–1502.

[25] N.L. Parrish, Low Thrust Trajectory Optimization in Cislunar and Translunar Space, PhD thesis University of Colorado, 2018, https://doi.org/10.1017/CBO9781107415324.004.

[26] D. Kirk, Optimal Control Theory: an Introduction, (1970), https://doi.org/10.1109/TAC.1972.1100008 No. 3, Dover.

[27] J.T. Betts, A survey of numerical methods for trajectory optimization, Mathematics and Engineering Analysis 67 (No. 4) (2012) 396–398, https://doi.org/10.2514/2.4231.

[28] R. Sood, Solar Sail Applications for Mission Design in Sun-Planet Systems from the Perspective of the Circular Restricted Three-Body Problem, (2012).

[29] M. Kelly, An introduction to trajectory optimization: how to do your own direct collocation, SIAM Rev. 59 (No. 4) (2017) 849–904, https://doi.org/10.1137/16M1062569.

[30] F. Topputo, C. Zhang, Survey of direct transcription for low-thrust space trajectory optimization with applications, Abstr. Appl. Anal. 2014 (2014) 1–20, https://doi.org/10.1155/2014/851720.

[31] J.B. Pezent, R. Sood, A. Heaton, Near Earth asteroid (NEA) scout solar sail contingency trajectory design and analysis, 2018 Space Flight Mechanics Meeting, 2018, pp. 1–20, , https://doi.org/10.2514/6.2018-0199.

[32] A. Mann, Heavy-lift rocket poised to boost space science, Science 359 (Jan) (2018), https://doi.org/10.1126/science.359.6374.376 376 LP – 377.

[33] B.A. Jorns, A.D. Gallimore, S.J. Hall, P.Y. Peterson, J.E. Gilland, Update on the Nested Hall Thruster Subsystem for the NextSTEP XR-100 Program Benjamin A. Jorns and Alec D. Gallimore, (2018), https://doi.org/10.2514/6.2018-4418.

[34] D.N. Spergel, R. Bean, O. Dor, Wilkinson Microwave Anisotropy Probe (WMAP) Three Year Observations: Implications for Cosmology, (2007), pp. 1–91 arXiv:astro-ph/0603449vol. 2.

[35] K.R. Wayne, H. Yu, Launch Window Trade Analysis for the James Webb Space Telescope, (2014), pp. 1–20 No. 1.

[36] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Network. 61 (2015) 85–117, https://doi.org/10.1016/j.neunet.2014.09.003.

[37] Q. Zhang, L.T. Yang, Z. Chen, P. Li, A survey on deep learning for big data, Inf. Fusion 42 (2018) 146–157, https://doi.org/10.1016/j.inffus.2017.10.006.

[38] D. Li, A tutorial survey of architectures, algorithms, and applications for deep learning, APSIPA Transactions on Signal and Information Processing 3 (2014) 1–29, https://doi.org/10.1017/ATSIP.2013.99.

[39] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, (2015) (Software available from: tensorflow.org).

[40] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Network. 4 (No. 2) (1991) 251–257, https://doi.org/10.1016/0893-6080(91)90009-T.

[41] K. Radhakrishnan, Description and Use of LSODE , the Livermore Solver for Ordinary Differential Equations, Tech. Rep. 113855 Lawrence Livermore National Laboratory, 1993.

[42] F. Long, S. Sidiroglou-Douskos, M. Rinard, Automatic runtime error repair and containment via recovery shepherding, SIGPLAN Not 49 (June 2014), https://doi.org/10.1145/2666356.2594337 227238.