AAS 25-052

REAL-TIME LEARNING BASED PLANNING FOR AUTONOMOUS RENDEZVOUS IN SPACE WITH ACTUATOR LOSS OF EFFECTIVENESS

Satvik G. Kumar [*], Joshua Ibrahim [†], Thomas A. Berrueta [‡], Soon-Jo Chung [§] and Fred Hadaegh [¶]

Spacecraft rely on precise guidance and close communications with ground-control during rendezvous operations. However, when autonomous spacecraft experience faults, real-time adaptation becomes essential to mission success. While methods such as model predictive control in principle allow for online re-planning, the computational demands of solving non-linear optimization problems can exceed what spacecraft are capable of in real-time. We present a framework that enables online adaptation to actuator loss of effectiveness faults. We leverage recent advances in learning-based guidance policies with optimality guarantees. Our framework incorporates data-driven fault detection based on analytical error bounds derived from contraction theory, identifies fault parameters using regularized regression, and performs online adaptation of the neural network guidance policy to enable system recovery. Simulation results for a chaser spacecraft in geostationary orbit demonstrate that our adaptive approach significantly reduces delivery errors compared to non-adaptive methods when faults are present. The proposed method achieves performance close to the nominal case, enabling robust, on-board trajectory re-planning for autonomous rendezvous missions in the presence of actuator faults.

INTRODUCTION

Autonomous rendezvous and docking of spacecraft with both cooperative and uncooperative objects of interest are key capabilities essential to a wide variety of space missions and applications. This includes missions such as rendezvous with asteroids, approaching an uncontrolled satellite, orbital debris removal, encountering interstellar space objects (ISO's),[1] autonomous on-orbit assembly, among many more applications.[2] Spacecraft typically rely on precise guidance and close communications with ground-control during rendezvous operations. However, fully autonomous spacecraft operation is important as it can both reduce costs as well as enable more remote mission concepts.[3]

One key aspect of fully autonomous spacecraft operation is the ability to fully recover from faults. Spacecraft can experience a multitude of faults during operation, including actuator faults, sensor faults, and software faults.[4] One specific type of actuator fault that spacecraft can experience includes actuator loss of effectiveness where actuators are unable to generate its full intended force

[*]Ph.D. Student, Graduate Aerospace Labs (GALCIT), California Institute of Technology, Pasadena, CA.
[†]Ph.D. Student, Control and Dynamical Systems, California Institute of Technology, Pasadena, CA.
[‡]Postdoctoral Scholar, Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA.
[§]Bren Professor of Control and Dynamical Systems, California Institute of Technology, Pasadena, CA.
[¶]Research Professor in Aerospace, California Institute of Technology, Pasadena, CA.

or motion.[5] If a spacecraft has faults or unplanned disturbances, it may deviate from its original planned trajectory. Many traditional techniques for fault-tolerant control focus on adapting the controller to continue to track the original trajectory.[6] However, there are many instances where online re-planning of a new trajectory is necessary.

Autonomous re-planning of trajectories in space is challenging due to the computational constraints of spacecraft. Since computers must be radiation hardened for space operation, spacecraft tend to be equipped with compute-limited processors. This presents issues for online trajectory re-planning because most methods, such as model predictive control (MPC), tend to be computationally expensive. To circumvent this issue, one avenue researchers have explored is the use of analytical approximations amenable to efficient trajectory re-planning such as the linearized Clohessy-Wiltshire equations.[7] While these methods offer computational speed, they sacrifice solution accuracy, potentially hindering critical off-nominal operations such as fault recovery. Another promising family of solutions leverage the use of neural networks and machine learning to avoid solving a nonlinear MPC optimization at each time step. In this framework, offline data is used to train a neural network that can mimic an optimal MPC during online operation. This way, offline computation is leveraged to achieve near-optimal planning performance during online use. This framework was explored for space rendezvous operations by Tsukamoto, et al.,[1] where the authors demonstrated a learning system capable of online planning through the use of spectrally normalized neural networks (SN-DNNs) termed Neural-Rendezvous. However, while this method presents formal performance guarantees of near-optimality during nominal operations, it fails to account for faults.

Here, we present an algorithmic framework for spacecraft rendezvous operations under loss of actuator effectiveness faults, enabling real-time detection, identification, and recovery through online adaptation of learning-based guidance policies with formal guarantees. In particular, we present a method that enables online adaptation of a neural-network based guidance policy with respect to an analytically-derived error bound, allowing our algorithm to detect fault-driven deviations from the original guidance policy. Figure 1 shows the interaction between all components of the spacecraft system and where the proposed framework fits in. Figure 2 displays the problem and a visualization of the real-time adaptation being proposed.

Related Works

Although successful missions with small body surfaces such as NEAR Shoemaker,[8] Hayabusa,[9] and OSIRIS-REx,[10] their safety issues in landing and decent[11] illustrate a growing need for robust and fault-tolerant GNC algorithms for rendezvous. For trajectory tracking and control, traditionally convex optimization methods[12] were used for their computational efficiency due to limited performance capabilities onboard. Although convex optimization methods can perform well in ideal conditions, real-time rendezvous requires robust solutions that must operate under significant nonlinearities—such as irregular body shapes, complex gravitational fields, and solar radiation.[13] Consequently, these non-convex constraints and dynamics often force convex approximations toward suboptimal solutions. To handle non-convexity and to plan complex mission trajectories such as in missions like the Comet Nucleus Sample Return (CNSR), offline planning using Monte Carlo simulations were performed to delineate between a slow versus fast descent strategy.[14] Model Predictive Control (MPC) has emerged as a powerful approach for space missions, offering sophisticated capabilities for handling non-convex optimization challenges with optimal results.[15]

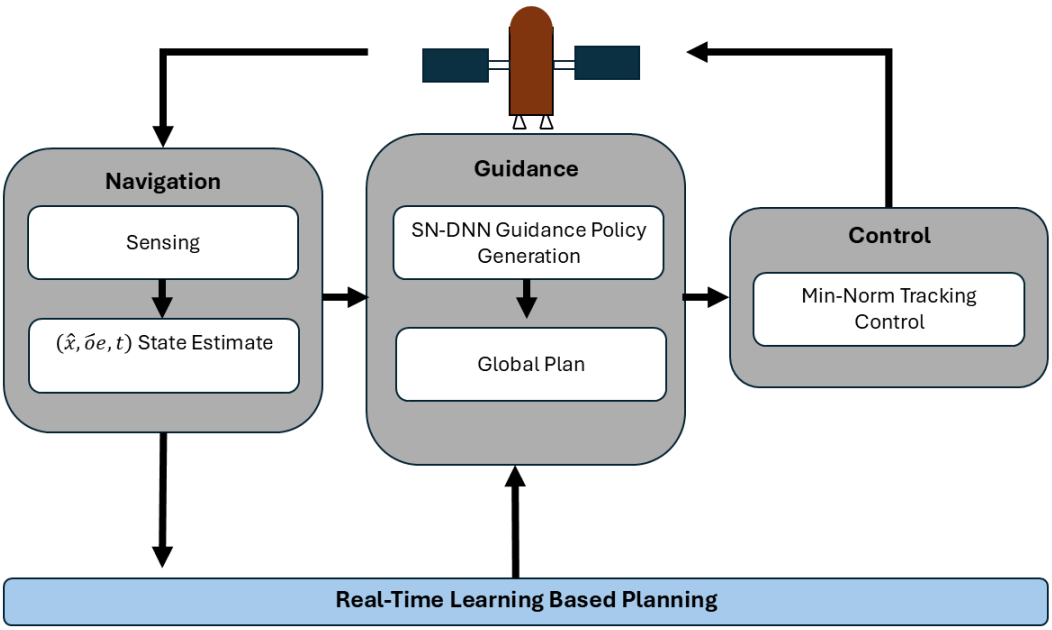

Figure 1 The proposed framework is a real-time learning based planning policy outlined in the blue box that affects the guidance policy of a spacecraft with faults and disturbances. Interactions are shown between the overall guidance policy planned using a SN-DNN, as well as the navigation and conrol subsystems of the spacecraft.

Building on the MPC framework, uncertainty-aware replanning methods have been developed that enable both distributional control of trajectory possibilities and landing error correction.[16] Additionally, tube-based MPC methods ensure robustness in asteroid rendezvous by propagating uncertainty bounds that characterize allowable deviations from planned trajectories.[17]

A key limitation of many robust control approaches is the requirement for a priori knowledge of the disturbance set for proper uncertainty quantification. In practice, spacecraft often encounter unanticipated faults during maneuvers. For example, in the Hayabusa mission's first rehearsal, the loss of two reaction wheels led to inaccurate thrust commands and forced a revised landing site.[9] Because spacecraft cannot be repaired once launched, fault-tolerant control (FTC) methods often rely on physical redundancy to compensate for failures.[18] Meanwhile, algorithmic methods have also been developed to address faults. One such class of algorithms include adaptive control schemes such as sliding window methods that have been developed to address known faults by stabilizing attitude control.[19,20] Adaptive controllers have also been developed for spacecraft rendezvous and docking under actuator loss of effectiveness.[21] In the presence of unknown faults, learning-based methods have emerged for actuator fault isolation.[22] While much of the work focuses on the design and adaptation of a fault tolerant controller previous work has also explored trajectory re-planning methods which can be combined with a FTC method under actuator faults.[6,23] While much of the fault-tolerant trajectory re-planning work focuses on setting up and solving optimization problems, this work focuses on generating guidance policies with faults utilizing online adaptation of a neural-network guidance policy.

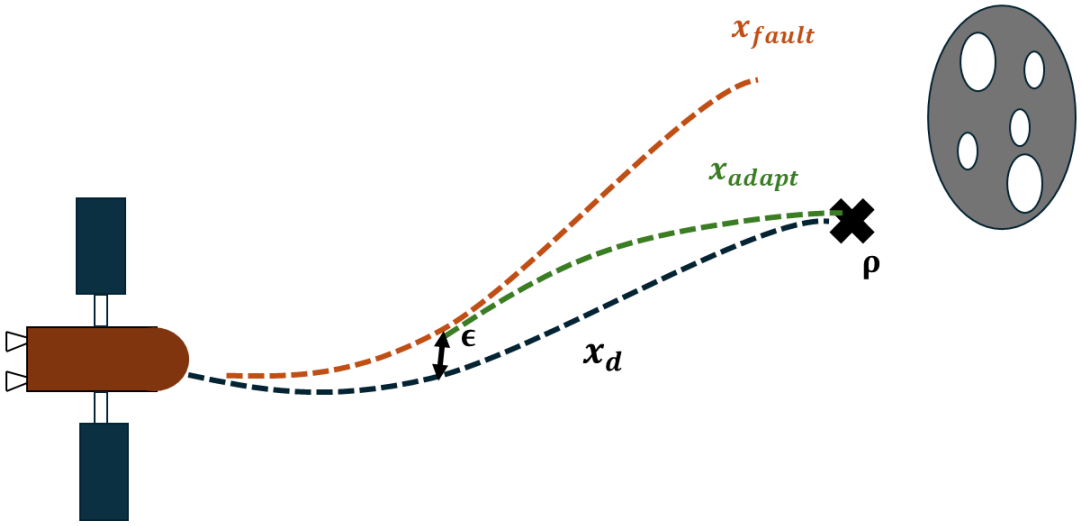

Figure 2 Problem formulation and proposed work of this paper.

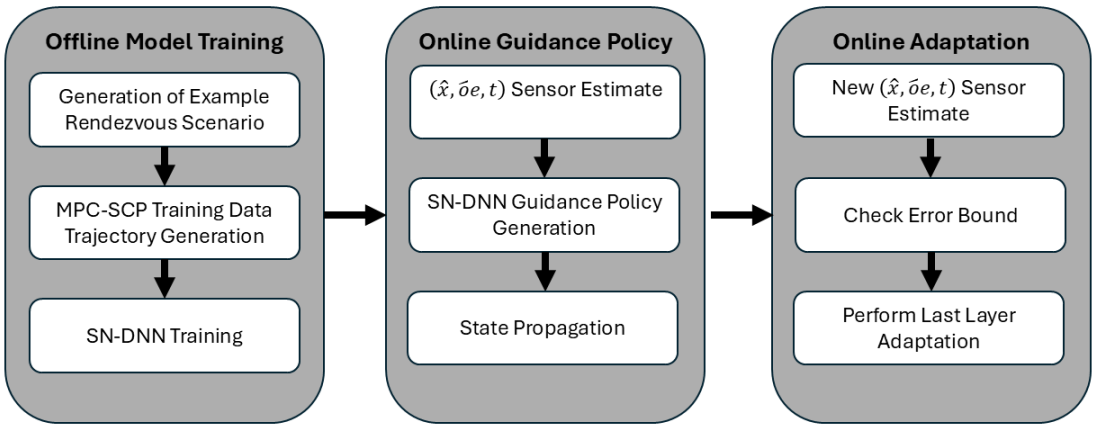

Figure 3 The overall methodology describing the relationship between offline model training, online guidance policy, and online adaptation.

METHODS

To enable real-time fault adaptation during rendezvous operations, we follow a three-pronged approach: data-driven detection of deviations from state-based, analytically-derived error bounds;

regularized-regression-based identification of actuator loss of effectiveness parameters; and online adaptation of the SN-DNN guidance policy for system recovery. At each time step, the system checks for deviations between the current measured position, provided by the navigation framework, and the originally planned trajectory. If the analytically-derived error bound is exceeded, loss of effectiveness parameters are estimated and the final layer of the SN-DNN is adjusted to learn the system's new dynamics. This entire process is visually represented in Fig. 3. Throughout this section, we derive analytical error bounds and formalize our problem setting.

Mathematical Formulation

In this work, we study the translational dynamics of a spacecraft traveling relative to a target (e.g., an asteroid or ISO). The system evolves under a feedback control law $u : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ with dynamics

$$\dot{x}(t) = f(x(t), \text{oe}(t), t) + B(x(t), \text{oe}(t), t)u(\hat{x}(t), \hat{\text{oe}}(t), t) \tag{1}$$

where $t \in \mathbb{R}_{\geq 0}$. The function $\text{oe} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ represents the time-varying orbital elements of the target, which themselves evolve according to a separate dynamical equation. The spacecraft's position and velocity relative to the target, expressed in a local-vertical local-horizontal (LVLH) frame centered at the target, is denoted by $x : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$. The functions $f : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and $B : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}^{n \times m}$ are known smooth mappings that capture the spacecraft's natural motion and how the control input affects it, respectively. In addition, $\hat{\text{oe}} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and $\hat{x} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ are on-board estimates of $\text{oe}(t)$ and $x(t)$, provided by an onboard navigation system*. We specifically consider the mass-normalized passive relative orbit dynamics of the form

$$f(x,t) = \begin{bmatrix} \dot{p} \\ C(\text{oe})\dot{p} + G(p, \text{oe}) \end{bmatrix}, \quad B(x,t) = \begin{bmatrix} O_{3\times 3} \\ I_{3\times 3} \end{bmatrix} \tag{2}$$

and state $x(t) = \begin{bmatrix} p(t) \\ \dot{p}(t) \end{bmatrix} \in \mathbb{R}^6$ as utilized in prior work.[24,25] Terms $C(\text{oe}), G(p, \text{oe}) \in \mathbb{R}^{3\times 3}$ are nonlinear functions defined in the prior work.[24,25]

For the dynamics in Eq. (1), we aim to learn a guidance policy u_ℓ from data generated by an offline planner u_{mpc}, such that $\|x_{\text{mpc}} - x_\ell\|$ is exponentially bounded under learning errors, input disturbances, and loss of effectiveness actuator faults. To achieve this, we first compute an offline optimal trajectory $(x_{\text{mpc}}, u_{\text{mpc}})$ that encodes the desired state and controller that drives the system to that state. Then, we use an SN-DNN to learn a guidance policy that imitates MPC-based expert demonstrations with bounded error $\|u_\ell - u_{\text{mpc}}\| \leq \epsilon$. Lastly, we decompose the controller u_ℓ into offline and online components, adapting the online parameters of the learned policy in real-time.

Model Predictive Control Problem

For our learning-based approach we require access to offline expert demonstrations with which to learn an imitation-learning-based guidance policy. To this end, we synthesize offline expert trajectories by solving a fixed-horizon optimal control problem in MPC fashion. We formulate our

*For information regarding notation, refer to Table (2).

MPC problem as follows,

$$
\begin{aligned}
u^*(x(t), \text{oe}(t), t) = \underset{\{x(\tau), u(\tau)\}}{\arg\min} \quad & \int_t^{t_f} \ell(x(\tau), u(\tau), \tau) d\tau + F\left(x\left(t_f\right)\right) \\
\text{s.t.} \quad & \dot{x}(\tau) = f(x(\tau), \text{oe}(\tau), \tau) + B(x(\tau), \text{oe}(\tau), \tau) u(\tau), \\
& x(\tau) \in \mathcal{X}, \quad u(\tau) \in \mathcal{U}, \quad \forall \tau \in [t, t_f],
\end{aligned}
\tag{3}
$$

where $t \in (0, t_f]$ is the current time at which Eq. 3 is being solved, $\ell(x, u, \tau)$ is an L^2 stage cost and $F(x)$ is the terminal cost. We use L^2 control effort as our stage cost and terminal relative position error as our terminal cost. Lastly, $\mathcal{X}$ and $\mathcal{U}$ enforce allowable state and control constraints such as velocity limits and thruster magnitude limits. We denote

$$
u_{\text{mpc}}(\hat{x}(t), \hat{\text{oe}}(t), t) = u^*(\hat{x}(t), \hat{\text{oe}}(t), t) \tag{4}
$$

as the optimal offline controller applied to Eq. (1) at each instant in time t resulting in the following closed-loop system dynamics,

$$
\dot{x}_{\text{mpc}}(t) = f(x_{\text{mpc}}(t), \text{oe}(t), t) + B(x_{\text{mpc}}, \text{oe}(t), t) u_{\text{mpc}}(\hat{x}(t), \hat{\text{oe}}(t), t). \tag{5}
$$

Thus, we construct an offline dataset of expert demonstrations by repeatedly solving Eq. (3) from different initial/terminal conditions and appending the resulting trajectories of $(x_{\text{mpc}}, u_{\text{mpc}})$ as determined by Eqs. (4) and (5). The MPC problem is solved utilizing the Sequential Convex Programming (SCP) approach outlined in prior work.[26] In an ideal scenario, we would compute an optimal control input according to Eq. (4) for every instance in time t. However, for a spacecraft subject to computational constraints and bandwidth limitations, this is computationally expensive. For this reason, we propose an imitation learning approach that uses offline data to train an SN-DNN-based guidance policy.

Imitation Learning

In this section, we describe the imitation learning methodology and derive novel bounds on the performance of our learned guidance policy when subject to additive disturbances. First, let $u_\ell(\hat{x}(t), \hat{\text{oe}}(t), t, \theta_{nn}, \theta_a)$ denote the learning-based guidance policy where θ_{nn} represents neural network parameters and θ_a represents adaptation parameters. Additionally, let $\varphi_\ell^t(x, \text{oe}, \tau, \theta_{nn}, \theta_a)$ and $\varphi_{\text{mpc}}^t(x, \text{oe}, \tau)$ denote the flow (i.e., solution trajectory) of the system under u_ℓ and u_{mpc}, respectively.

We parameterize our learned guidance policy, u_ℓ, in terms of two components: a neural network mapping $\Phi(x, \text{oe}, t, \theta_{nn})$ trained with offline data and a linear online adaptation component $\theta_a(t)$, which we optimize in real time. Effectively, $\theta_a(t)$ can be thought of as the final linear layer of the SN-DNN representing the guidance policy. Together, these components comprise our learned control policy in the following way

$$
u_\ell = \theta_a(t) \Phi(x, \text{oe}, t, \theta_{nn}) \tag{6}
$$

where the SN-DNN $\Phi(x, \text{oe}, t, \theta_{nn})$ maps its inputs onto $\mathbb{R}^m$, and $\theta_a(\cdot) \in \mathbb{R}^{m \times m}$ is a matrix of adaptation parameters that reweigh the contribution of the neural network output vectors in real time. We note that $m = 3$ for the dynamics in Eq. (1).

We utilize a spectrally normalized deep neural network (SN-DNN). Spectral normalization bounds the Lipschitz constant of the learned neural network, allowing error bounds to be computed even when making use of neural networks.

To learn an offline guidance model, we fix an initial set of adaptation parameters $\theta_a(\cdot) = \theta_a^{\text{fix}}$ and define a loss function with respect to which we train our neural network, in line with previous work.[1] We proceed by minimizing

$$\mathcal{L}_{nn}(\theta_{nn}) = \mathbb{E}\Big[\|u_\ell(\bar{x},\bar{\text{oe}},\bar{t},\theta_{nn},\theta_a^{\text{fix}}) - u_{\text{mpc}}(\bar{x},\bar{\text{oe}},\bar{t})\|_{C_u} + \|\varphi_\ell^{\bar{t}+\Delta\bar{t}}(\bar{x},\bar{\text{oe}},\bar{t},\theta_{nn},\theta_a^{\text{fix}}) - \varphi_{\text{mpc}}^{\bar{t}+\Delta\bar{t}}(\bar{x},\bar{\text{oe}},\bar{t})\|_{C_x}\Big] \tag{7}$$

with respect to θ_{nn} in data-driven manner, replacing the expectation with an empirical average taken over samples drawn from our offline dataset of expert demonstrations with $C_x, C_u > 0$ as constants to balance between control-input and state-output imitation performance.

Now, we proceed by deriving analytical error bounds on system performance, $\|x_\ell - x_{\text{mpc}}\|$, when subject to errors in the learned guidance policy as well as additive disturbances. Consider the dynamics in Eq. (5) with controller $u_{\text{mpc}}(\hat{x}(t),\hat{\text{oe}}(t),t) = u^*(\hat{x}(t),\hat{\text{oe}}(t),t)$ and state x_{mpc}. Suppose that the error between the learned controller and the planned controller is bounded as

$$\|u_\ell\left(\bar{x},\bar{\text{oe}},\bar{t},\theta_{nn},\theta_a^{\text{fix}}\right) - u_{\text{mpc}}(\bar{x},\bar{\text{oe}},\bar{t})\| \leq \varepsilon \tag{8}$$

where ε provides a bound on the offline learning error. In addition to the learning error above, we consider additive disturbances to the learned policy $u_\ell + d$, such that $\|d(\bar{x},\bar{\text{oe}},t)\| \leq \bar{d},\ \forall \bar{x},\bar{\text{oe}},t$.

Next, we define a virtual system $q(\mu,t)$ parameterized by $\mu \in [0,1]$ with particular solutions $q(\mu=1,t) = x_{\text{mpc}}$ and $q(\mu=0,t) = x_\ell$.[27] The virtual system dynamics are given by

$$\dot{q} = \xi(q,x_{\text{mpc}},x_\ell,u_\ell,t) + d_q(\mu,x_{\text{mpc}},x_\ell,u_\ell,t) \tag{9}$$

such that

$$\xi|_{q=x_{\text{mpc}}} = f(x_{\text{mpc}},\text{oe},t) + B(x_{\text{mpc}},\text{oe},t)u_{\text{mpc}}(\hat{x},\hat{\text{oe}},t)$$
$$\xi|_{q=x_\ell} = f(x_\ell,\text{oe},t) + B(x_\ell,\text{oe},t)(u_\ell(\hat{x},\hat{\text{oe}},t,\theta_{\text{nn}},\theta_a^{\text{fix}}) + d(\bar{x},\bar{\text{oe}},t))$$

and $d_q = \mu\big(B(x,\text{oe},t)(u_\ell\left(\bar{x},\bar{\text{oe}},\bar{t},\theta_{nn},\theta_a^{\text{fix}}\right) + d(\bar{x},\bar{\text{oe}},t) - u_{\text{mpc}}(\bar{x},\bar{\text{oe}},\bar{t}))\big)$. In this sense, μ is a parameter that allows one to continuously interpolate between the system dynamics according to u_{mpc} and u_ℓ. With these preliminaries out of the way, we may now state our main theorem.

Theorem 1. *If $\exists\beta \in [0,\infty)$ such that $\|B(x,\text{oe},t)\| \leq \beta,\ \forall x,\text{oe},t$ and $u_{\text{mpc}} = u^*$ such that there exists a contraction metric $M(q,x_{\text{mpc}},x_\ell,u_\ell,t) = \Theta^\top\Theta \succ 0$ and constants $\alpha,\underline{m},\overline{m} \in \mathbb{R}_{>0}$ satisfying the contraction conditions:*

$$\dot{M} + M\frac{\partial\xi}{\partial q} + \frac{\partial\xi}{\partial q}^\top M \leq -2\alpha M \tag{10}$$

$$\underline{m}I \preceq M \preceq \overline{m}I \tag{11}$$

$\forall q,x_{\text{mpc}},x_\ell,u_\ell,t$, then the error $e(t) = x_{\text{mpc}}(t) - x_\ell(t)$ is exponentially bounded as:

$$\|e(t)\| \leq \frac{V(0)}{\sqrt{\underline{m}}}e^{-\alpha t} + \frac{\beta(\varepsilon+\bar{d})}{\alpha}\sqrt{\frac{\overline{m}}{\underline{m}}}(1-e^{-\alpha t}) \tag{12}$$

where $V(t) = \int_{x_\ell}^{x_{\text{mpc}}}\|\Theta\delta q\|$.

Proof. Let $V(q, \delta q, t) = \int_{x_\ell}^{x_{\text{mpc}}} \|\Theta \delta q\|$, $\partial_\mu q = \partial q / \partial \mu$ and $\partial_\mu d_q = \partial d_q / \partial \mu$. Differentiating and using the contraction inequality in Eq. (10), for $M = \Theta^\top \Theta$,

$$\begin{aligned} \frac{d}{dt} \|\Theta(q,t)\partial_\mu q\| &= (2\|\Theta(q,t)\partial_\mu q\|)^{-1} \frac{d}{dt} \partial_\mu q^\top M(q,t) \partial_\mu q \\ &\leq -\alpha \|\Theta(q,t)\partial_\mu q\| + \|\Theta(q,t)\partial_\mu d_q\| \end{aligned} \tag{13}$$

From $\|\partial_\mu d_q\| \leq \beta(\varepsilon + \bar{d})$, $\|\Theta(q,t)\| \leq \sqrt{\overline{m}}$ and integrating with respect to μ gives

$$\frac{d}{dt} \int_0^1 \|\Theta \partial_\mu q\| \, d\mu \leq \int_0^1 -\alpha \|\Theta \partial_\mu q\| \, d\mu + \beta(\varepsilon + \bar{d})\sqrt{\overline{m}} \int_0^1 d\mu \tag{14}$$

which implies that $\dot{V} \leq -\alpha V + \beta(\varepsilon + \bar{d})\sqrt{\overline{m}}$. By the Comparison Lemma,[28]

$$V(t) \leq e^{-\alpha t} V(0) + \frac{\beta(\varepsilon + \bar{d})\sqrt{\overline{m}}}{\alpha}(1 - e^{-\alpha t}). \tag{15}$$

Since we know that $x_{\text{mpc}} - x_\ell = \int_{x_\ell}^{x_{\text{mpc}}} \delta q$, we additionally know that $\|x_{\text{mpc}} - x_\ell\| = \| \int_{x_\ell}^{x_{\text{mpc}}} \delta q\| \leq \int_{x_\ell}^{x_{\text{mpc}}} \|\delta q\| \leq \int_{x_\ell}^{x_{\text{mpc}}} \|\Theta^{-1}\| \|\Theta \delta q\|$, which means that $V(t) = \int_{x_\ell}^{x_{\text{mpc}}} \|\Theta \delta q\|$ is lower bounded by $\sqrt{\overline{m}} \|x_{\text{mpc}} - x_\ell\| \leq V(t)$ at all points in time. Lastly, dividing by $\sqrt{\overline{m}}$ gives the desired result. □

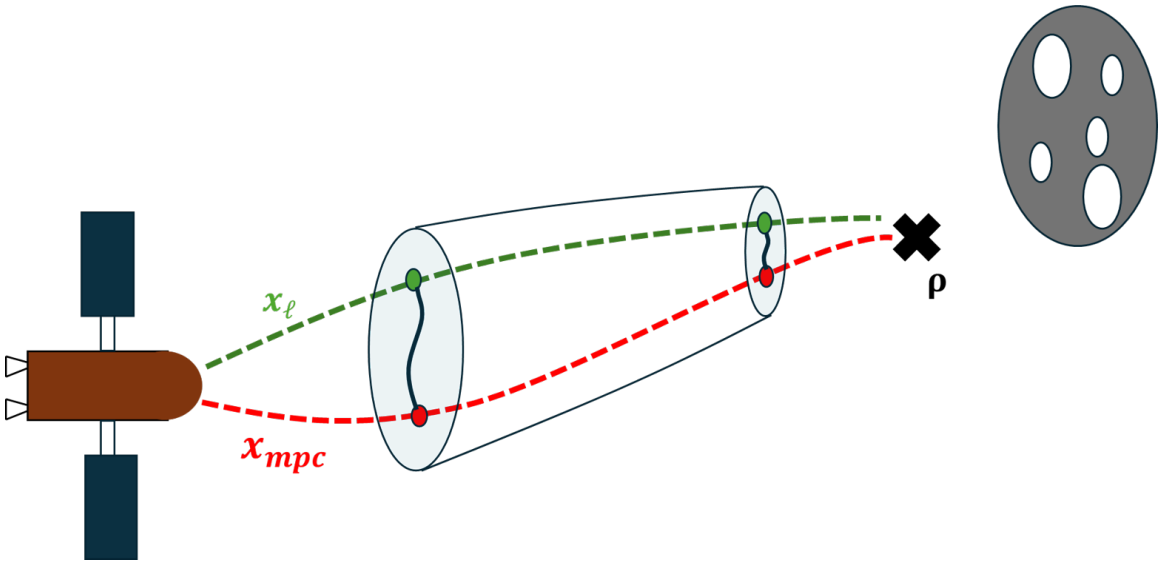

Figure 4 Visualization of the exponentially bounded error

The bound in Eq. (12) shows that the difference in trajectories $\|x_{\text{mpc}} - x_\ell\|$ varies linearly with respect to additive faults $\bar{d}$ and learning error ε. Moreover, since we know that the system dynamics under controller u_{mpc} (Eq. (1)) are exponentially stable—and that u_ℓ is within ε of u_{mpc}—we have proven that our learning-based guidance policy provides an exponentially stabilizing solution to the underlying optimal control problem (Eq. (3)) without the need for additional computation in the presence of additive faults. Figure 4 shows a visualization of the exponentially bounded error.

The methodology presented throughout this section provides guarantees on the performance of our learning-based offline guidance policy in the presence of disturbances and deviations between

the learned policy and the underlying MPC. However, due to its offline nature it is not capable of adapting to other kinds of faults experienced by the spacecraft over the course of its deployment. To this end, in the following section we augment this procedure with real-time online adaptation, which will allow our algorithmic framework to detect, identify, and recover from faults in real time.

Real-Time Adaptation

The bound in Theorem 1 establishes a baseline for the performance of our system under nominal conditions—that is, under conditions where our system experiences bounded disturbances and deviations from an optimal reference. Having access to such an analytically-derived baseline is powerful because it provides a means of data-driven fault detection. If the bound in Eq. (12) is violated, then we know that the system is experiencing a fault beyond learning errors or noisy actuation, requiring additional adaptation in order for the spacecraft to recover. In this section, we present a real-time adaptation methodology grounded in the bounds presented in the previous section that allows the system to recover from loss of actuator effectiveness faults. Consider the actuator effectiveness parameter $E(t)$ defined in Eq. (16) where $e_1(t)$, $e_2(t)$, and $e_3(t)$ represent loss of effectiveness in the x, y, and z directions, respectively.

$$E(t) = \begin{bmatrix} e_1(t) & 0 & 0 \\ 0 & e_2(t) & 0 \\ 0 & 0 & e_3(t) \end{bmatrix} \tag{16}$$

If the commanded control input is u_{cmd} then the actual control input u_{act} is found utilizing Eq. (17)

$$u_{act} = E(t)u_{cmd} \tag{17}$$

If the bound in Eq. (12) is violated, our method proceeds by first identifying differences between a trajectory utilizing the system's nominal dynamics and its measured state-evolution. Using the newly identified model, we compute an online adaptation to the learned guidance policy by modulating $\theta_a(t)$ in real-time, until system performance returns to nominal conditions.

To this end, we compute two components, $\theta_a^{\text{dyn}}(t) \in \mathbb{R}^{m\times m}$ identifies differences between the actual and nominal system dynamics, and $\theta_a^{\text{fault}}(t) \in \mathbb{R}^{m\times m}$ identifies parameters needed to adapt the guidance policy and recover from the fault. The adaptation component ends up becoming the fault adaptation parameter, $\theta_a(t) = \theta_a^{\text{fault}}(t)$. In order to achieve such adaptation in real-time, we compute $\theta_a^{\text{fault}}(t)$ and $\theta_a^{\text{dyn}}(t)$ every Δt seconds. We discretize this interval into N timesteps t_k such that $\Delta t = N\delta t$ and $t_k = k\delta t + t_0$, for all $k \in \{0, \cdots, N\}$, where t_0 is the time at the moment in which the adaptation procedure begins. Then, we are able to frame surrogate data-driven optimizations over $\theta_a^{\text{fault}}(t)$ and $\theta_a^{\text{dyn}}(t)$ by constructing a dataset $\mathcal{D}_{\Delta t}$ of $(\hat{x}(t_k), \hat{\text{oe}}(t_k), u_\ell(t_k))$ for all $k \in \{0, \cdots, N\}$.

We proceed by identifying $\theta_a^{\text{dyn}}(t_0)$, the adaptation parameters that best identify the system dynamics over the previous Δt seconds. In order to achieve this, we perform a ridge regularized least-squares optimization over $\mathcal{D}_{\Delta t}$ by framing the optimization problem in terms of the spacecraft's measured state $\hat{x}_k = \hat{x}(t_k)$, and the expected spacecraft state given current model of dynamics $x_k = x(t_k)$. We calculate the expected spacecraft state through a linearized, discretized

version of the nonlinear continuous-time equations of motion based on a zero-order hold approach as outlined by Morgan et al.,[26]

$$x_{k+1} = A_k^d x_k + B_k^d u_k + C_k^d, \tag{18}$$

where $u_k = \theta_a^{\text{dyn}} \Phi(x(t_k), \text{oe}(t_k), t_k, \theta_{nn})$ and $\theta_a^{\text{dyn}}(t_k) = \theta_a^{\text{dyn}}$ for all k since we hold it constant as we optimize over the interval Δt. Then, using this dynamics model we can express our ridge regression problem as follows,

$$\underset{\theta_a^{\text{dyn}}}{\arg\min} \quad \frac{1}{N} \sum_{k=0}^{N-1} \left|\left| \hat{x}_{k+1} - \left(A_k^d x_k + B_k^d (\theta_a^{\text{dyn}} \Phi(x_k, \text{oe}_k, t_k, \theta_{nn})) + C_k^d\right) \right|\right| + \lambda \left|\left| \theta_a^{\text{dyn}} \right|\right|. \tag{19}$$

Eq. (19) is solved efficiently online using the closed-form solution of a linear ridge-regression problem.

Then, equipped with an estimate for θ_a^{dyn} we formulate a similar optimization over θ_a^{fault}. Rather than optimize residuals between spacecraft dynamics estimates, we formulate this secondary objective with respect to a desired trajectory $x_k^{\text{des}} = x_{\text{des}}(t_k)$. In doing so, we aim to use our identified parameters to help the guidance policy compensate for loss of effectiveness faults in our system. In this case, the optimization problem is as follows

$$\underset{\theta_a^{\text{fault}}}{\arg\min} \quad \frac{1}{N} \sum_{k=0}^{N-1} \left|\left| x_{k+1}^{\text{des}} - \left(A_k^d x_k + B_k^d (\theta_a^{\text{fault}} \theta_a^{\text{dyn}} \Phi(x_k, \text{oe}_k, t_k, \theta_{nn})) + C_k^d\right) \right|\right| + \lambda \left|\left| \theta_a^{\text{fault}} \right|\right|. \tag{20}$$

Lastly, we combine our adaptation methodologies into Algorithm 1. Using this approach, we are able to perform real-time learning-based fault adaptation in spacecraft simulations, as the following sections outline.

Algorithm 1: Real-Time Learning Based Adaptive Planning

Input : Trained SN-DNN Model from Eq. (6)
Output: Control Input u_ℓ and $a(t)$ from Eq. (6)
$t_0, x_0 \leftarrow$ initial time, relative position
$t_f \leftarrow$ final time
$\Delta t \leftarrow$ planning time interval
$\Delta t_a \leftarrow$ adaptation time interval
$t_N \leftarrow$ current time - t_0
Generate $x_d(t)$ utilizing SN-DNN and RK45 Forward Integration
while $t_N < t_f$ **do**
  $a(t_N) = a(t_{N-1})$
  if $||x_N - x_d(t_N)|| > \epsilon$ *and* $t_k > \Delta t_a$ **then**
    $a(t_N)$ = RealTimeAdaptation()
  end
  $u_\ell(t_N) = \Phi(\hat{x}_N, \hat{\text{oe}}_N, t_N) a(t_k)$
end

SIMULATION

Scenario

The proposed method is demonstrated on a rendezvous problem with an object such as a satellite (target) in geostationary orbit drawing inspiration from the scenario in Chan et al.[2] The task is to generate a guidance policy for a spacecraft looking to rendezvous with a target. The spacecraft is initialized at a position $100m$ away from the target and it must rendezvous at a location $5m$ away from the target. The total time allocated for the maneuver ($t_f - t_0$) is set to be 2 hours ($7200s$). The guidance policy is generated every $10s$. The maximum mass-normalized control input is assumed to be 0.02 m/s^2.

Training Data Generation

500 candidate trajectories are generated with an 80-20 training/validation split. To generate each trajectory, the spacecraft initial position relative to the target is sampled randomly from the surface of a ball with radius 100 meters and the spacecraft desired position (ρ) relative to the target is also sampled from the surface of a ball with radius 5 meters. Initial orbital parameters of the target are chosen randomly from a distribution. Semimajor axis values (a) are chosen randomly from $R_{geo} \pm 25m$ where R_{geo} is the radius of a geostationary orbit, eccentricity (e) values are randomly chosen between 0 and 5.0048×10^{-4}, inclination values (i) are chosen randomly between -0.1° and 0.1°. Right ascension of the ascending node (RAAN - Ω), Argument of the periapsis (ω), and true anomaly (ν) values are chosen randomly between 0 and 2π.

Control inputs u_{mpc} along each candidate trajectory are generated using the MPC-SCP method described in a prior section. Trajectory flow $\varphi_{\text{mpc}}^{t+\Delta t}$ along each candidate trajectory are generated by numerical integration using an RK45 integrator. The MPC-SCP problem is solved with a time step of 10 seconds, the same time step that the SN-DNN guidance policy is generated. The total time that is allowed for the rendezvous process is 7200 seconds.

SN-DNN Training

A subset of the 288,000 total data points generated are utilized to train the SN-DNN. This is to improve the generalization of the network while reducing the computational time and resources required for training. 30,000 data points are chosen uniformly and randomly ensuring that 75 training data points are chosen from each of the 400 trajectories allocated for training and approximately the same number of data points (41-42) are chosen at each time step within the time interval. Of the 30,000 data points, 24,000 are utilized for training and 6,000 are utilized for validation.

The input data are transformed as shown in Equation (21) similar to the transformation performed by Tsukamoto et al.[1]

$$\text{SN-DNN Input} = (p_i - \rho_i, \dot{p}_i, \rho_i, t_f - t_i, \omega_{z,i}, G(p_i, \text{oe}_i)) \tag{21}$$

Furthermore, the input and output of the SN-DNN are divided by the absolute value of the largest value in each input parameter of the training data. This ensures that all training inputs and outputs

to the SN-DNN are between zero and one and minimizes the risk of inputs with larger magnitudes dominating the network.

The neural network has an input layer with 14 neurons, three hidden layers with 32, 64, and 64 neurons and an output layer that outputs a control input utilizing 3 neurons. The activation function is selected to be rectified linear unit (ReLU). Spectral normalization is applied to each layer to control the Lipschitz constant of the network. The parameter c_x is set to 10^8 and the parameter c_u is set to 10^9 as these parameters were found to give the best results. The Adaptive Moment Estimation (Adam) optimizer is utilized. The SN-DNN is trained for 1000 epochs.

Error Detection and Adaptation

For error detection and adaptation, the initial θ_a^{fault} and θ_a^{dyn} are set to the identity matrices for initial model training. While Theorem 1 gave a theoretical formulation for a bound between $\|x_\ell - x_{\text{mpc}}\|$, since the original system is contracting the bound would apply for any given desired trajectory x_d as long as the controller is still feasible. Since the x_{mpc} trajectory is not available to us in space, errors are detected based on the deviation from a desired x_d trajectory. The x_d trajectory is calculated from the initial spacecraft position x_0. The trajectory is generated by generating a simulated guidance policy from the SN-DNN for future time-steps and performing forward integration using RK-45. This would simulate a desired trajectory for the spacecraft given that there are no faults or disturbances (perfect sensing and actuation). Thus although the derived bound is given in terms of a perfect MPC solution in simulation a bound is determined utilizing the deviation from an initially generated trajectory. In this work, a bound is selected through experimentation to demonstrate the online adaptation technique. However, the derivation of the theoretical bound provides justification for being able to utilize a bound to detect errors. Future work will investigate the use of the theoretically introduced bound.

For the online θ_a^{fault} and θ_a^{dyn} optimization problems, the regularization parameter λ is set to a value of 10^{-25} demonstrating that for the particular data considered in this simulation regularization may not have been necessary.

RESULTS

This section describes the results for the geostationary orbit scenario outlined in the Simulation section. Table 1 provides the final position error averaged over 100 trials for each experiment that is run. The final delivery error is calculated by taking the two-norm of the x, y, and z position delivery errors.

Table 1 Geostationary Orbit Table of Errors

Simulation Type	Overall Error (m)
No Fault	1.33
Loss of Actuator Effectiveness with Real-Time Adaptive Policy	3.05
Loss of Actuator Effectiveness with No Policy	24.79
Real-Time Introduction of Loss of Actuator Effectiveness	4.29

The experiment labeled *No Fault* in the table is the baseline case which utilizes imitation learning with no loss of actuator effectiveness. Figure 5 displays the results for this case. The average

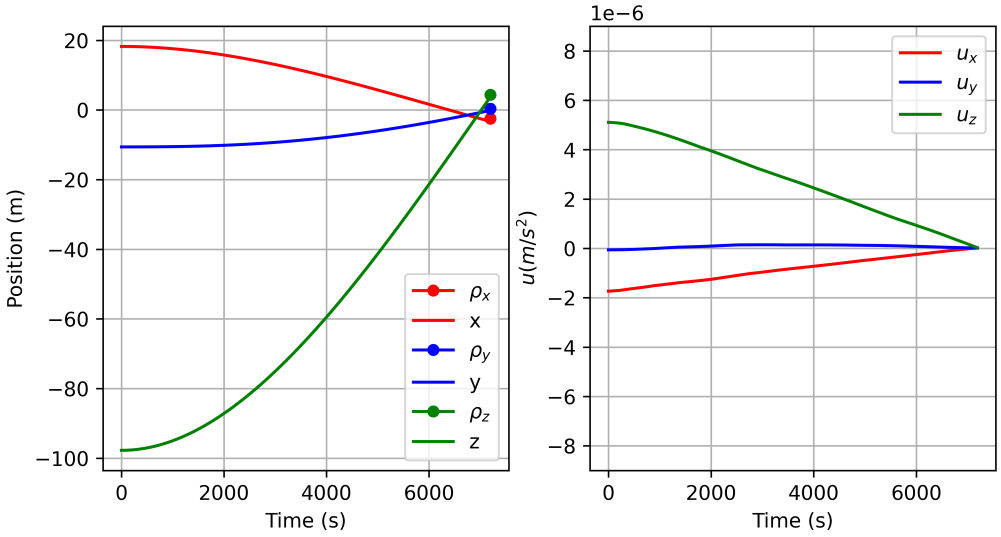

Figure 5 Position and Control Input from the SN-DNN for the nominal cause where there is no fault

delivery error of 1.33m is due to errors from the learning process. The forward and backward integration process along with a min-norm controller as implemented in prior work[1] can be utilized to bring the actual delivery error to zero despite the planning delivery error being 1.33 meters.

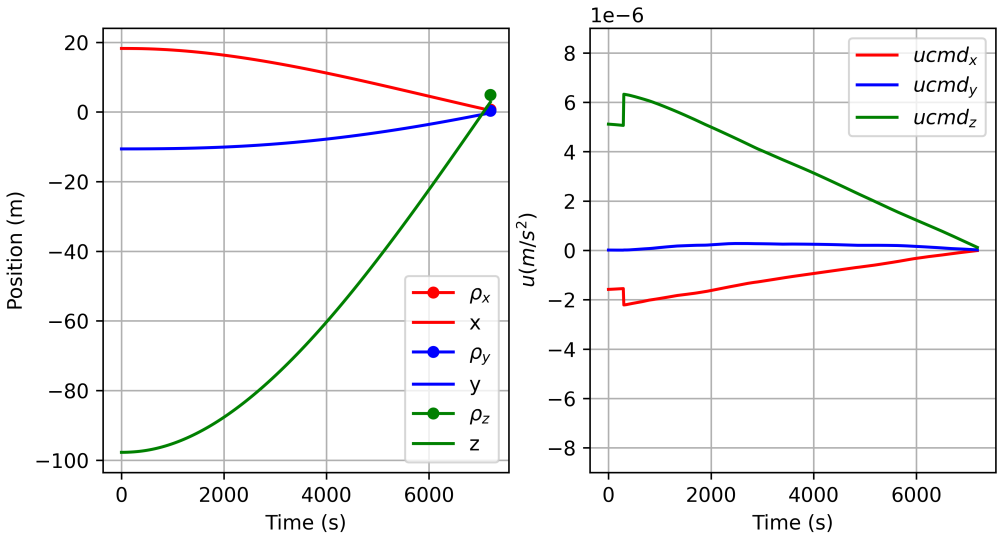

Figure 6 Position and control input for the second simulation where there is a fault during the entire time period and adaptation is performed

The experiment labeled *Loss of Actuator Effectiveness with Real-Time Adaptive Policy* demonstrates the use of our adaptation policy for a case with loss of actuator effectiveness for the entire 2-hour period. The loss of actuator effectiveness, $E(t)$ is set to a constant value through the time interval where $e_1(t) = 0.7$, $e_2(t) = 0.6$ and $e_3(t) = 0.8$. The adaptive guidance control

policy is run for the entire 2-hour period where the optimization problem is solved every $k = 30$ steps. Figure 6 displays the results from this case. In Figure 6, examining the spike on the graph of commanded control input (Adapted SN-DNN output) shows where the adaptation takes place.

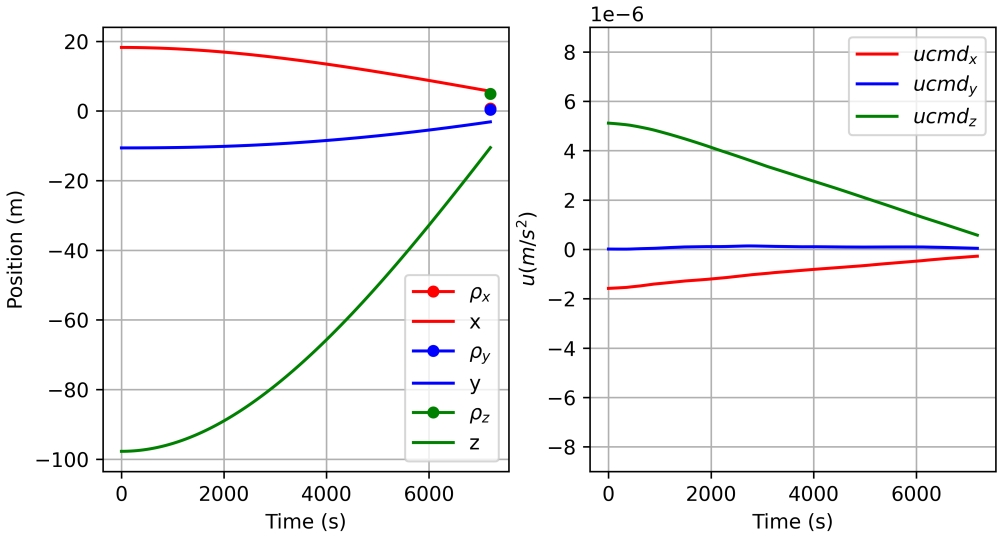

Figure 7 Position and control input for the third simulation where there is a fault during the entire time period and no adaptation is performed

The experiment labeled *Loss of Actuator Effectiveness with No Policy* shows the final delivery error with a loss of actuator effectiveness term without our adaptation policy. It is important to note that the network even without adaptation will inherently compensate for loss of actuator effectiveness by outputting higher control inputs to compensate for the larger $p_t - \rho_t$ at any given time after the fault. However, without adaptation, this results in a large delivery error. Figure 7 displays results for this case.

The experiment labeled *Real-Time Introduction of Loss of Actuator Effectiveness* is the case where a fault is introduced into the system at a time of 1000 seconds. The fault is detected and the adaptation process begins when the error goes over 0.25 meters. Figure 8 shows results for this case. Looking at the *Rate of Change of Error* plot makes the adaptation extremely clear. When the fault is introduced, the error rate of changes starts to increase. After adaptation is performed, the error rate of change (between the original planned trajectory and the spacecraft position) begins to decrease showing that the fault has been compensated for. The error graph continues to increase however even after adaptation as the error is calculated between the original planned trajectory and the location of the spacecraft. However, after adaptation the spacecraft may not necessarily follow the original planned trajectory.

CONCLUSION

This work proposed an algorithmic framework for spacecraft rendezvous operations under loss of actuator effectiveness faults. A three-pronged approach was proposed including an analytically-derived error bound to allow our algorithm to detect fault-driven deviations from the guidance policy, identification of fault parameters utilizing regularized regression, and online

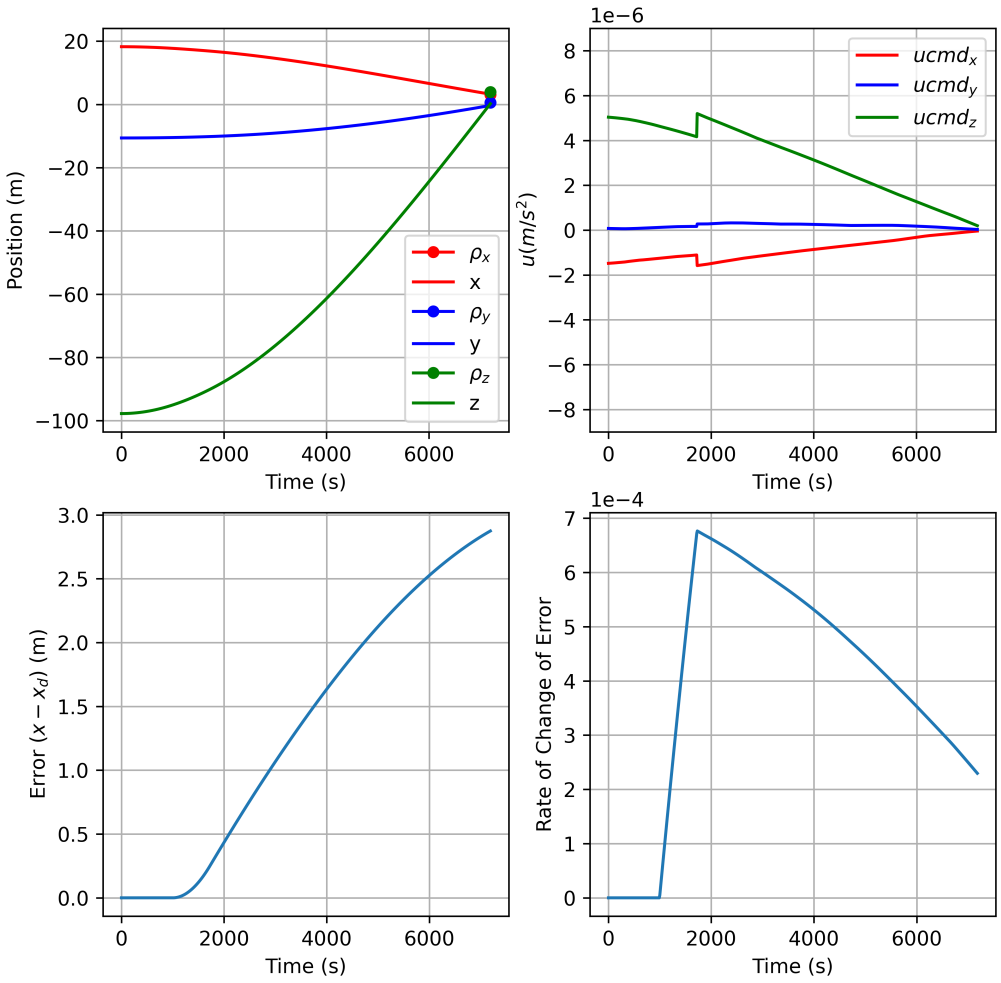

Figure 8 Position and control input for the fourth simulation where a fault is introduced into the system and bounds are utilized to determine when to begin adaptation

adaptation of the neural network guidance policy to enable system recovery. We demonstrated the potential of this method in simulations. Further work will focus on incorporating a variety of disturbances and faults into the framework. Additionally, further work will focus on alternative approaches to perform adaptation to neural-network parameters.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 2139433. This work is supported in part by the Technology Innovation Institute (TII).

NOTATION

Table 2 Notation

$O_{m\times n}$	$=$	$m \times n$ zero matrix
$I_{m\times n}$	$=$	$m \times n$ identity matrix
$x(t),\ \hat{x}(t),\ x_{\mathrm{mpc}}(t)$	$=$	true, estimated, and MPC generated state of the spacecraft relative to ISO in the LVLH frame at time t
$u,\ u_\ell,\ u_{\mathrm{mpc}}$	$=$	true, learned, and MPC generated control policy
$\mathfrak{oe}(t),\ \hat{\mathfrak{oe}}(t)$	$=$	true and estimated orbital elements at time t
$p(t),\ \hat{p}(t)$	$=$	true and estimated position of the spacecraft relative to ISO in the LVLH frame at time t
ρ	$=$	terminal position of the spacecraft relative to ISO
$\varphi_\ell^t,\ \varphi_{\mathrm{mpc}}^t$	$=$	learned and MPC solution trajectories at time t
$\theta_{nn},\ \theta_a$	$=$	neural network and adaptation parameters
$a(t)$	$=$	adaptation mapping weights at time t

REFERENCES

[1] H. Tsukamoto, S.-J. Chung, Y. K. Nakka, B. Donitz, D. Mages, and M. Ingham, "Neural-Rendezvous: Provably Robust Guidance and Control to Encounter Interstellar Objects," *Journal of Guidance, Control, and Dynamics*, Vol. 47, No. 12, 2024, pp. 2525–2543.

[2] M. Chan and R. Sargent, "Rendezvous Approach Guidance for Uncooperative Tumbling Satellites," *2020 IEEE Aerospace Conference*, IEEE, 2020, pp. 1–17.

[3] W. F. Truszkowski, M. G. Hinchey, J. L. Rash, and C. A. Rouff, "Autonomous and autonomic systems: A paradigm for future space exploration missions," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 36, No. 3, 2006, pp. 279–291.

[4] S. H. Pourtakdoust, M. Fakhari Mehrjardi, M. H. Hajkarim, and F. Nasihati Gourabi, "Advanced fault detection and diagnosis in spacecraft attitude control systems: Current state and challenges," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 237, No. 12, 2023, pp. 2679–2699.

[5] Q. Jia, R. Ma, C. Zhang, and R. Varatharajoo, "Spacecraft attitude fault-tolerant stabilization against loss of actuator effectiveness: A novel iterative learning sliding mode approach," *Advances in Space Research*, Vol. 72, No. 2, 2023, pp. 529–540.

[6] A. Chamseddine, C. Join, and D. Theilliol, "Trajectory planning/re-planning for satellite systems in rendezvous mission in the presence of actuator faults based on attainable efforts analysis," *International journal of systems science*, Vol. 46, No. 4, 2015, pp. 690–701.

[7] W. Clohessy and R. Wiltshire, "Terminal guidance system for satellite rendezvous," *Journal of the aerospace sciences*, Vol. 27, No. 9, 1960, pp. 653–658.

[8] D. W. Dunham, R. W. Farquhar, J. V. McAdams, M. Holdridge, R. Nelson, K. Whittenburg, P. Antreasian, S. Chesley, C. Helfrich, W. M. Owen, *et al.*, "Implementation of the first asteroid landing," *Icarus*, Vol. 159, No. 2, 2002, pp. 433–438.

[9] J. Kawaguchi, A. Fujiwara, and T. Uesugi, "Hayabusa—Its technology and science accomplishment summary and Hayabusa-2," *Acta Astronautica*, Vol. 62, No. 10-11, 2008, pp. 639–647.

[10] J. Gal-Edd and A. Cheuvront, "The OSIRIS-REx asteroid sample return mission operations design," *2015 IEEE Aerospace Conference*, IEEE, 2015, pp. 1–9.

[11] B. G. Williams, "Technical challenges and results for navigation of NEAR Shoemaker," *Johns Hopkins APL technical digest*, Vol. 23, No. 1, 2002, pp. 34–45.

[12] B. Acikmese and S. R. Ploen, "Convex programming approach to powered descent guidance for mars landing," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 5, 2007, pp. 1353–1366.

[13] D. Scheeres, B. G. Williams, and J. K. Miller, "Evaluation of the dynamic environment of an asteroid: Applications to 433 Eros," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 3, 2000, pp. 466–475.

[14] J. De Lafontaine, "Autonomous spacecraft navigation and control for comet landing," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 3, 1992, pp. 567–576.
[15] U. Eren, A. Prach, B. B. Koçer, S. V. Raković, E. Kayacan, and B. Açıkmeşe, "Model predictive control in aerospace systems: Current state and opportunities," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 7, 2017, pp. 1541–1566.
[16] M. AlandiHallaj and N. Assadian, "Asteroid precision landing via probabilistic multiple-horizon multiple-model predictive control," *Acta Astronautica*, Vol. 161, 2019, pp. 531–541.
[17] K. Albee, C. Oestreich, C. Specht, A. Terán Espinoza, J. Todd, I. Hokaj, R. Lampariello, and R. Linares, "A robust observation, planning, and control pipeline for autonomous rendezvous with tumbling targets," *Frontiers in Robotics and AI*, Vol. 8, 2021, p. 641338.
[18] S. Yin, B. Xiao, S. X. Ding, and D. Zhou, "A review on recent development of spacecraft attitude fault tolerant control system," *IEEE Transactions on Industrial Electronics*, Vol. 63, No. 5, 2016, pp. 3311–3320.
[19] B. Xiao, Q. Hu, and Y. Zhang, "Adaptive sliding mode fault tolerant attitude tracking control for flexible spacecraft under actuator saturation," *IEEE Transactions on Control Systems Technology*, Vol. 20, No. 6, 2011, pp. 1605–1612.
[20] Q. Shen, D. Wang, S. Zhu, and E. K. Poh, "Integral-type sliding mode fault-tolerant control for attitude stabilization of spacecraft," *IEEE Transactions on Control Systems Technology*, Vol. 23, No. 3, 2014, pp. 1131–1138.
[21] B. Jiang, Q. Hu, and M. I. Friswell, "Fixed-time rendezvous control of spacecraft with a tumbling target under loss of actuator effectiveness," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 52, No. 4, 2016, pp. 1576–1586.
[22] Y. Huang, S. Li, and J. Sun, "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.
[23] X. Miao, L. Cheng, Z. Zhang, J. Li, and S. Gong, "Convex optimization for post-fault ascent trajectory replanning using auxiliary phases," *Aerospace Science and Technology*, Vol. 138, 2023, p. 108336.
[24] D. Morgan, S.-J. Chung, L. Blackmore, B. Acikmese, D. Bayard, and F. Y. Hadaegh, "Swarm-keeping strategies for spacecraft under J2 and atmospheric drag perturbations," *Journal of Guidance, Control, and Dynamics*, Vol. 35, No. 5, 2012, pp. 1492–1506.
[25] S.-J. Chung, U. Ahsun, and J.-J. E. Slotine, "Application of synchronization to formation flying spacecraft: Lagrangian approach," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 2, 2009, pp. 512–526.
[26] D. Morgan, S.-J. Chung, and F. Y. Hadaegh, "Model predictive control of swarms of spacecraft using sequential convex programming," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 6, 2014, pp. 1725–1740.
[27] H. Tsukamoto, S.-J. Chung, and J.-J. E. Slotine, "Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview," *Annual Reviews in Control*, Vol. 52, 2021, pp. 135–169.
[28] H. Khalil, "Nonlinear systems," 2002.