RAPID TRAJECTORY DESIGN IN COMPLEX ENVIRONMENTS ENABLED VIA SUPERVISED AND REINFORCEMENT LEARNING STRATEGIES

Das-Stuart, A.
Purdue University, USA, das15@purdue.edu

Howell, K.C.
Purdue University, USA, howell@purdue.edu

Folta, D.
NASA Goddard Spaceflight Center, USA, david.c.folta@nasa.gov

The investigation focuses on blending machine learning strategies with traditional trajectory design techniques to uncover solutions and enhance human intuition. A free-form search indiscriminately leverages chaotic and ordered motion to scope the trade-space. Alternatively, models of periodic orbit families trained via Artificial Neural Networks and Support Vector Machines exert more control over the transfer profile and geometry. A spacecraft's performance capabilities deliver *accessible regions*, where supervised learning and free-form strategies illuminate potential waypoints to the destination. Reinforcement learning agents then sequence advantageous waypoints to construct end-to-end solutions. The initial guess is transitioned to a continuous form via traditional numerical strategies.

1. Introduction

Recent advancements in the space exploration field offer opportunities to reach a wide array of destinations, from the Moon, to the asteroid belt, as well as to the outer planets. Such endeavors demand effective mission design strategies that trade-off diverse constraints to ensure mission success. Consequentially, an increase in mission complexity suggests that a rapid trajectory design framework is valuable — one that offers the exploration of broad trade-spaces and is, at least, semi-automated. Such a framework is particularly beneficial in the near term, e.g., to support an efficient cis-lunar transportation architecture that also aids the emergence of new mission concepts beyond the Earth-Moon neighborhood. The knowledge gained from many prior efforts form an integral part of the current investigation.

The construction of a rapid design framework commences with a fundamental understanding of the natural dynamics in the Earth-Moon system and the option to leverage existing structures to enable mission scenarios. Folta et al.[1] offer an interactive catalog of orbit families for applications in multi-body regimes, and additional investigations demonstrate the ability to link various arcs belonging to such orbits in the Circular Restricted Three Body Problem (CR3BP).[2–4] The utility of natural arcs can serve as a basis for transfer design, frequently lowering propellant costs even in efficient low-thrust regimes.[5] Assessments of hyperplane crossings,[6] steering law assumptions and optimal control theory are common tools exploited to facilitate design in such regimes. Collocation and direct transcription strategies have also emerged to address the challenges associated with constrained basins of convergence, and the sensitivity of indirect optimization methods to the initial guess.[7]

The access to a wide range of natural arcs, to be coupled with powered arcs resulting from available thrust capabilities, results in the examination of an infinitely large trade-space to satisfy mission constraints. This approach quickly becomes intractable when addressed solely via manual search methods. Thus, some recent investigations address the problem via combinatorial optimization techniques, generally employing two different strategies. In the first formulation, traditional numerical processes are employed to construct an initial guess database comprised of locally optimal solutions. Then, well known graph-search and machine learning methods are exploited to solve a multi-objective problem by examining combinations within the database to produce a global or nearly-global optimum. Conclusions from Radice and Olmo,[8] Ceriotti and Vasile,[9] Stuart and Howell,[10] as well as Furfaro and Linares[11] demonstrate the potential of heuristic methods such as Ant Colony Optimization (ACO), and Reinforcement Learning (RL) to be effective in various dynamical regimes and in uncovering local optima that may have otherwise remained unknown. Approaches employing genetic algorithms have also proven as beneficial.[12,13] The second type of strategy recasts the problem in terms of pathfinding — where the initial guess itself is constructed via Artificial Intelligence (AI) techniques and is then subjected to a numerical corrections process. Tsirogiannis,[14] as well as Trumbauer and Villac[15] generate impulsive transfer options by constructing a framework of pre-computed natural arcs and using graph search methods to evaluate the links. The nodes then serve as waypoints in a complex dynamical regime. Simplifying assumptions in a two-body model are exploited by Parrish[16] to employ heuristics in solving for consistent low-thrust initial guesses. Das-Stuart et al.[17, 18] demonstrate the ability to construct end-to-end initial guesses in a CR3BP regime for low-thrust and impulsive transfers via exact and reinforcement learning strategies, which are then readily transitioned to continuous optimal solutions via traditional numerical techniques.

The current investigation builds upon the previous work undertaken in Das-Stuart et al.,[17, 18] and Figs. 1 and 2 summarize the modified design approach. The first step in Fig. 1 determines the *reach* of the s/c in a multi-dimensional and infinitely large configuration space via its Accessible Regions (ARs). Multiple avenues exist to render natural conditions for the s/c to evaluate in step 2 (expanded upon in Fig. 2). This step is followed by deploying reinforcement learning agents to appropriately select and sequence advantageous arcs from the ARs to construct end-to-end pathways in an automated manner. The discontinuous arcs are then passed through a traditional numerical corrections process in the final step to produce a continuous solution for a specified engine model. In contrast to the previous investigations[17, 18] where the natural conditions in step 2 in Fig. 1 are established via a pre-generated and discretized catalog (Fig. 2(a)), the modified approach in this paper establishes potential waypoints to the destination on-the-fly within these ARs. Such an approach liberates the pathfinding agents from restriction to waypoints defined within the discretized catalog. A free-form search facilitates the flexibility to leverage both chaotic and ordered motion to broaden the design options, and engage human intuition in the development of the relevant trade-space (Fig. 2(b)). Supervised learning strategies such as Support Vector Machines (SVMs) and Artificial Neural Networks (ANNs) are also exploited to train *flow-models* that aid in influencing the design process (Fig. 2(c)). The framework design effort focuses on addressing the challenges associated with thrust law construction, the sequencing of thrust and coast arcs, and expanding the solution pool to mitigate the limited solution options resulting from narrow basins of convergence when implementing traditional numerical techniques. Such an approach is beneficial in time-critical contingency planning scenarios.

The force models and numerical techniques that support the initial guess generation process are introduced in Sections 2 and 3. A high-level overview of the supervised and reinforcement learning strate-

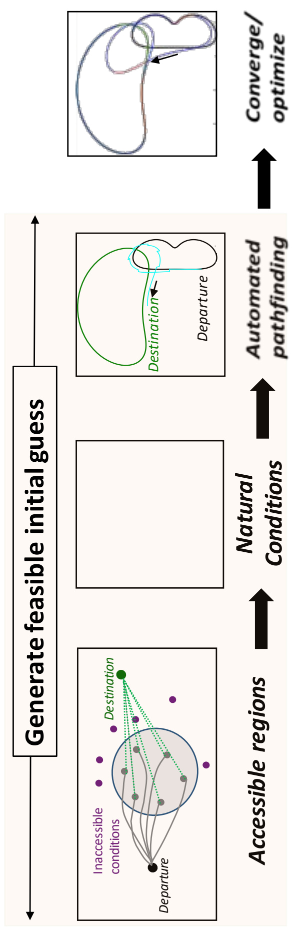


Fig. 1: Overview of design framework

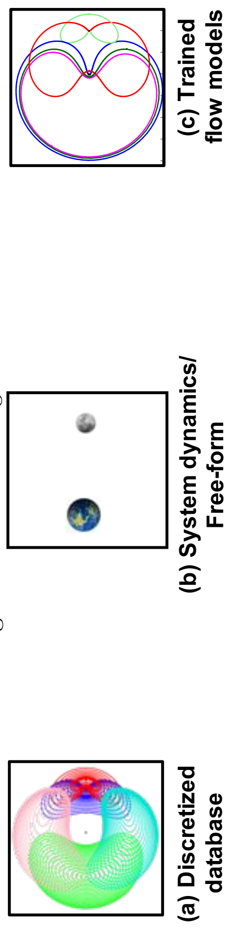


Fig. 2: Alternatives for the natural condition generation step (Fig. 1: Natural Conditions) in the framework

gies appear in Section 4, and a detailed discussion on the associated design choices, along with derivations of the algorithms, are presented in the Appendix. The design framework formalized in Section 5 enables a number of examples in Section 6 that demonstrate the process, including some scenarios that are relevant to current proposals for a facility near the moon.

2. Dynamical and Force Model

Although the dynamical sensitivity is evident, the CR3BP offers an opportunity to approximate the higher-fidelity dynamics and to exploit the natural flows that are otherwise unavailable in simpler dynamical regimes. For the design framework, a robust cis-lunar transportation architecture is currently a priority, so the methodology is applied within the context of the Earth-Moon CR3BP. Here, the Earth (*primary,* P_1) and the Moon (*secondary,* P_2) are assumed to revolve in circular orbits around their common barycenter.[19] The spacecraft mass is assumed to be negligible in comparison to the more massive bodies. The Equations of Motion (EOM) for the spacecraft (P_3) as viewed in a rotating frame, also incorporate a thrust force to model the physical capabilities of an engine/thruster, i.e.,

$$\dot{\boldsymbol{\chi}} = \begin{Bmatrix} \dot{\boldsymbol{r}} \\ \dot{\boldsymbol{v}} \\ \dot{m} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{v} \\ \underbrace{\boldsymbol{f}(\boldsymbol{r}) + \boldsymbol{g}(\boldsymbol{v})}_{\text{natural}} + \underbrace{\frac{\mathbb{T}}{m}\hat{\boldsymbol{u}}}_{\text{low-thrust}} \\ \frac{-\mathbb{T}}{Isp\, g_o} \end{Bmatrix} \qquad [1]$$

The differential equations include contributions from both the natural gravitational and the thrust acceleration sources to capture the motion of the spacecraft (s/c) and its mass-history over time. In these equations, $\boldsymbol{\chi}$ is the full state vector comprised of the vehicle position and velocity vectors ($\boldsymbol{r}$ and $\boldsymbol{v}$, respectively) and the vehicle mass m. The thrust magnitude is represented by $\mathbb{T}$, and the thrust direction by $\hat{\boldsymbol{u}}$ (where a caret identifies unit magnitude). The *Isp* is the engine specific impulse and g_0 is the reference gravitational acceleration. Even without the engine-specific thrusting terms, there is no closed-form solution to the natural EOMs. So, in the thrust-free problem, other quantities such as the period, stability and the Jacobi Constant (*JC*, an energy-like quantity) that are associated with the natural solutions aid in characterizing the motion of the s/c within the confines of the CR3BP model. The *JC* is defined as:

$$JC = 2U^* - v^2 \qquad [2]$$

where U^* is a pseudo-potential term,

$$U^*(x, y, z) = \frac{1-\mu}{r_{13}} + \frac{\mu}{r_{23}} + \frac{1}{2}(x^2 + y^2) \qquad [3]$$

So, U^* is a function of the s/c position relative to the barycenter (x, y, z) and relative to the two primaries r_{13} and r_{23}, as well as the mass ratio of P_2 to the total system mass, μ. Note from Eqn. [2] that the *JC* term is only constant in the autonomous system without propulsive forces.

To deliver the thrust force in Eqn. [1], a relatively new technology, advanced ion-drive propulsion, is gaining popularity due to the ability to improve payload delivery capabilities. This efficiency, however, is delivered at lower thrust levels and, therefore, potentially lengthy flight durations.[20] Transitions to a higher-fidelity model enabled by chemical engines is also investigated.

3. Numerical Corrections and Optimization

Constructing low-thrust optimal trajectories typically involves formulating an optimal control problem, and solving for the time-histories of the thrust magnitude and the thrust force direction to meet the desired boundary conditions using direct or indirect approaches. Direct methods are more robust than indirect ones, but induce a large dimensionality[21, 22] and often require assumptions on the thrust profile (e.g., constant thrust direction over each arc).

3.1 Constant Specific Impulse Engine Parameters

In the Constant Specific Impulse (CSI) regime, the available thrust magnitude is a function of the engine power allocation, ($\mathbb{P}$) and efficiency (*Isp*). The relationship is modeled as:

$$\mathbb{T} = \frac{2\mathbb{P}}{Isp\, g_0} \qquad [4]$$

The constant power and *Isp* parameters dictate a constant thrust magnitude. The basis of the control authority to maneuver the s/c is an on-off engine toggle and thrust-vectoring. A continuous low-thrust trajectory is constructed from a discontinuous initial guess by employing a multiple-shooting scheme. Also, due to the inherent numerical sensitivities associated with indirect methods for the CSI trajectory design, direct methods are selected in this preliminary analysis. An iterative Newton-Raphson scheme is employed to compute a set of design vector variables ($\boldsymbol{X^*}$) that satisfy the specified constraints, $\boldsymbol{F}(\boldsymbol{X}^*) = \boldsymbol{0}$, i.e.,

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{u}_i \\ \tau_i \\ TD_i \end{bmatrix} \quad \boldsymbol{F} = \begin{bmatrix} \boldsymbol{x}_{i desired} - \boldsymbol{x}_{i actual} \\ \tau_i + TD_i - \tau_{i+1} \\ \boldsymbol{\psi}_{desired} - \boldsymbol{\psi}_{actual} \\ \boldsymbol{u}_i{}^T \boldsymbol{u}_i - 1 \end{bmatrix} = \boldsymbol{0} \quad [5]$$

The design vector $\boldsymbol{X}$, is comprised of the states ($\boldsymbol{x}_i$), thrust directions ($\boldsymbol{u}_i$), epoch times (τ_i), and propagation durations (TD_i) at select-nodes (i) that are available from the initial guess. The constraint vector $\boldsymbol{F}$ is sought to meet state-continuity, epoch-continuity, boundary condition ($\boldsymbol{\psi}$), and thrust direction unit-vector constraints at the specified nodes. The converged solution is passed through a Nonlinear Programming (NLP) software such as FMINCON or SNOPT to optimize a cost function that is formulated to offer a favorable solution to the mission objectives and constraints.

3.2 Chemical Engine Parameters

Construction of a converged/optimized trajectory enabled by a chemical engine allows velocity discontinuities as a design parameter. So, a process similar to the direct approach as discussed in Section 3.1 is implemented, with the exclusion of the thrust magnitude and thrust direction terms in the design and constraint vectors.

4. Machine Learning Paradigm

The generation of an initial guess trajectory that transfers a s/c from a departure location to a specific destination is cast as a routing problem. Such a route, however, is constructed from the appropriate sequencing of available waypoints to meet the user's global objectives. Thus, the design process is predicated on the ability to 1) recognize desirable waypoints, and then 2) construct pathways between them. Varied aspects of Machine Learning (ML), a data-driven approach within the field of Artificial Intelligence (AI) to *learn* complex and nonlinear relationships, are adopted to address these separate design constituents. Specifically, traversal towards the destination is achieved by exploiting Reinforcement Learning (RL) to strategically assemble advantageous intermediate conditions uncovered via random exploration and also from those determined to be acceptable via Supervised Learning (SL) strategies.

4.1 Supervised Learning Strategies

Supervised learning algorithms are often employed for pattern recognition and function approximations to aid in data classification and regression tasks. Since the 1950s, numerous algorithms have emerged, e.g., decision trees, naive Bayes, Artificial Neural Networks (ANN) and Support Vector Machines (SVM). The algorithm choice depends greatly on its unique strengths to address the nature, size and complexity of the problem and the ability to balance the desired levels of accuracy and computational efficiency. In this investigation, using both ANNs and SVMs to learn complex relationships in a highly nonlinear and high-dimensional multi-body regime is explored. Both techniques *learn* from sample pairs of input data with the corresponding categorical assignments (labels) or continuous outputs during a training phase. Successful training enables inferences concerning the underlying function relating the variables in the design space and, subsequently, leads to the satisfactory prediction of outputs on unseen datasets.

4.1.1 Artificial Neural Networks

Artificial Neural Networks are a collection of algorithms inspired by the functioning of biological neural networks to solve complex problems. Their inception lies in the *perceptron*, an artificial adaptation of a single biological neuron, first studied by Frank Rosenblatt in the late 1950s.[23] In the 1960s, the discovery of a perceptron's inability to process data that are not linearly separable (e.g., Exclusive-Or, XOR) temporarily halted progress.[24] A decade later, the ability to overcome this challenge by introducing at least one extra layer of artificial neurons evolved into multi-layer feedforward networks and, thus, deep learning. The most basic feed-forward network consists of an input, hidden and output layer; the schematic of a slightly more sophisticated 3-layer network (2 hidden layers) is portrayed in Fig. 3. Similar to human

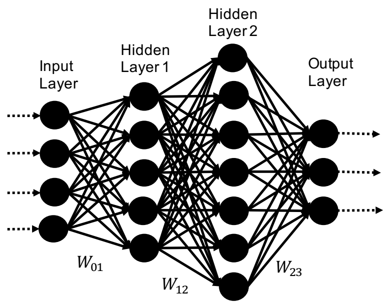


Fig. 3: Sample Artificial Neural Network Architecture *(Modified from Bapu[25])*

brains, the iterative and dynamic strengthening of signals between neurons enable the identification of important relationships to process the data and report results in the output layer. In an '$\mathbb{o}$'-layer network, the output or logit $\mathbb{z}_j$, of the j^{th} neuron in layer l is formulated as the inner product between the signal strengths from neurons i, in previous layer $(l-1)$ (captured via the weights $\mathbb{w}_{i,j}$), and the inputs $\mathbb{x}_{i,j}$:[26]

$$\mathbb{z}_j^{(l)} = \sum_{i=1}^{mm} \mathbb{w}_{i,j} \bullet \mathbb{x}_i + \mathbb{B}^{(l)} \qquad [6]$$

where, $\mathbb{B}$ is a constant bias term that identifies the range of inputs that exercise the greatest influence over the outputs of a layer l, and mm is the total number of neurons in this layer. The output from a hidden layer is modeled as the matrix operation:

$$\mathbb{Z}^{(l)} = \mathbb{W}^T\mathbb{X} + \mathbb{B}^{(l)} \qquad [7]$$

Accurate predictions are achieved via iterative tuning of the weight matrices and bias terms. Various techniques to accomplish this goal render different varieties of ANNs. Note that within the feed-forward network implementation, there are no links between neurons within a given layer, and there is no requirement to enforce equal number of neurons within each hidden layer as well. The choices involved in building an ANN architecture play a key role in its performance and applicability for a given scenario, and consists of the following considerations:[27]

A **Feature and Label Initialization**

In a supervised learning realm, careful selection of the user-specified inputs (features) comprising the $\mathbb{x}$ vector in Eqn. [6] as well as the corresponding outputs/labels greatly influence the *learning* absorbed by the neurons; the efficiency of the networks is impacted as well. The multiplicative and additive operations of the weighted values through each hidden layer also yield fresh features within the hidden layers, introducing additional insights for the network.

B **Data Allocation**

Within a sample-based platform, the quantity of data to which a network is exposed, and the proportion of available points from each data-type to be classified/regressed also impact the training outcomes. The *goodness* of a trained network is assessed via validation and testing phases on unseen data. So, appropriate allocation within a larger dataset for the training, validation and testing phases is an essential consideration - one that can be determined via empirical testing for a particular scenario.

C **Training Components**

Training an ANN involves seeking continual improvements on the weights (signal strengths) to identify critical relationships within the dataset. This goal is achieved by the merits of the techniques flowing the information forward through the hidden layers, and those tracing the errors backwards (backpropagation) to rectify the associated weight parameters accordingly.

C.a. Activation Function

In a biological network, a neuron fires only after the received signal strength exceeds a threshold value. A neuron's logit is, thus, passed through an activation function to help emulate this behavior and also introduce nonlinearity into the outputs to enable the learning of complex relationships. Equation [7] is then modified as follows to compute a layer's output:

$$\mathbb{Y}^{(l)} = f(\mathbb{W}^T\mathbb{X} + \mathbb{B}^{(l)}) = f(\mathbb{Z}^{(l)}) \qquad [8]$$

The frequency with which the spikes (outputs) are released from a neuron is determined by the activation function, $f(\mathbb{Z})$. The nuances associated with a particular problem guide the selection of this function. Although a variety of options are available, the linear, hyperbolic tangent and softmax functions have been incorporated in this investigation. The information associated with their derivation and implementation are detailed in Appendix 9.0.1A.

C.b. Error Minimization

Backpropagation algorithms offer feedback to the activated neurons and, thus, create an opportunity for the weights (signal strengths) to be updated prior to being passed forward through the layers in the next iteration. The updates to the weights are informed by the rate of change of the error E between the true solution $(\mathbb{Y}_t)$ and the ANN prediction $(\mathbb{Y}_p)$ at the output layer $(\mathbb{o})$, with respect to the weights in each layer, $\mathbb{w}_{i,j}^{(l)}$:

$$\Delta\mathbb{w}_{i,j}^{(l)} = -\epsilon\frac{\partial E}{\partial \mathbb{w}_{i,j}^{(l)}} \qquad [9]$$

Here, ϵ is a hyper-parameter called the 'learning rate', that can be tuned during the training phase. The goal then, is to maneuver towards the global minimum of a multi-dimensional error surface engendered by the weight coordinates and is commonly achieved by following the negative gradient on the surface (gradient descent). However, the exacerbation of saddle-point-type conditions due to high dimensionality, and potential ill-conditioning of the Hessian matrix, can lead to fluctuating gradient directions and poor convergence properties. So, the appropriate combination of backpropagation and the associated optimization is problem-specific and does influence the values of the weight updates in Eqn. [9] as well. Conjugate gradient and quasi-Newton methods such as

Levenberg-Marquardt algorithms that aid in navigating challenging/ill-conditioned error surfaces[27,29] have been leveraged in this analysis. The selection of the error function E in Eqn. 9 depends on the activation function employed in the output layer and thus, the problem type. The information associated with the loss functions incorporated in this investigation are detailed in Appendix 9.0.1B.

D **Generalizability**

The *Universal Approximation Theory*, first proved in the 1980s by Cybenko,[32] demonstrated that a single hidden layer could approximate any continuous function. This realization was extended by Funahoshi[33] in 1989 to clarify that a combination of hidden layers and neurons can be exploited by an ANN architecture to approximate any continuous function. However, more neurons/layers do not necessarily result in better approximations. In fact, the challenge in *bias-variance* trade-off necessitates a careful selection of the *size* of the ANN.[34] For example, a small network may result in insufficient parameters to satisfactorily capture the underlying complexities of the problem leading to a high bias and under-fitting of the relationships in the training set. In contrast, a large network can tend to over-fit the function, thus, leading to modeling even the noise in the dataset and exhibiting high variance when subject to new and unseen data. The goal is, therefore, generalizability - the ability to strike a balance between adequate learning of the underlying relationships on known data and adequate predictions of the relationships on unmodeled data. One approach to the iterative training process to establishing a generalizable ANN model is summarized in Fig. 4.

4.2 Support Vector Machines

Similar to ANNs, Support Vector Machines (SVMs) also originate in the perceptron and the ability to discriminate between data within a non-separable set. However, rather than the multi-layered filtering techniques adopted by ANNs, SVMs rely on the construction of maximum margin hyperplanes to deliver conclusions on the relationships within the dataset. Vapnik offered the underlying mathematics supporting this novel decision-making approach in 1963, and further developments over 2-3 decades led to the current form of SVMs.[35]

A **Training Components**

Parallels are apparent between a single-layer ANN and a simple linear SVM classifier. Recall the equation for the hyperplane separating two linearly separable classes of interest in Eqn. [6]. This equation is

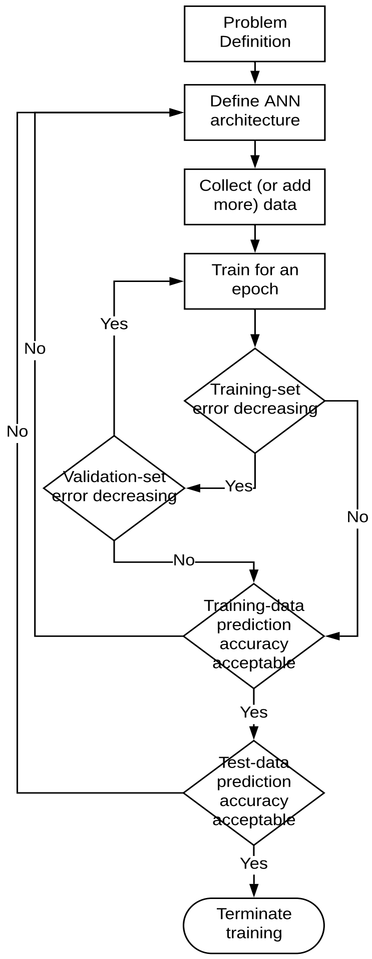

Problem Definition
Define ANN architecture
Collect (or add more) data
Train for an epoch
Training-set error decreasing
Validation-set error decreasing
Training-data prediction accuracy acceptable
Test-data prediction accuracy acceptable
Terminate training
Yes
No

Fig. 4: An implementation of Artificial Neural Network Work Flow. *(Modified from Buduma[27])*

re-written in the form:

$$f(\mathbb{X}) = \sum_{i=1}^{mm} \mathbb{x}_i \bullet \mathbb{w}_{i,j} + \mathbb{B}^{(l)} = \mathbb{X}^T \mathbb{W} + \mathbb{B}^{(l)} = 0. \qquad [10]$$

Two more hyperplanes are introduced to bound the margin of separation, $\mathfrak{d}$, between the two types of data within the set,[36] $f(\mathbb{X}) = \mathbb{X}^T\mathbb{W} + \mathbb{B} \geq \pm 1$. Figure 5 illustrates the mathematical formulation conceptually. The orientation of the hyperplanes is in-

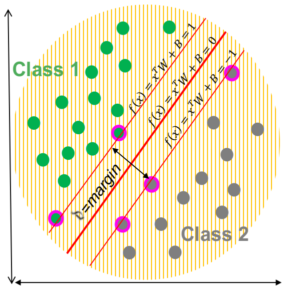


Fig. 5: Schematic illustrating formulation for SVM.

fluenced by the weights $\mathbb{W}$, and the bias $\mathbb{B}$ shifts the hyperplanes along the plane. These two parameters (weights and bias) then comprise the controls to maximize the margin between the separable data types, or the distance between the hyperplanes defining the separation boundaries, $\mathfrak{d} = \frac{2}{||\mathbb{W}||}$. Solving for the maximal margin is equivalent to the optimization problem, where $||\mathbb{W}||$ is minimized (Eqn. [11]). However, data are frequently not easily separable. So, slack variables, ζ, are introduced.[37] Equation [11] then assumes a form where the objective function is appended with penalty measures:

$$\text{minimize} \frac{1}{2}||\mathbb{W}||^2 + C\sum_{i=1}^{N} {\zeta_i}^{\kappa} \qquad [11a]$$

$$\text{subject to: } \mathbb{y}_i[\mathbb{w}_i\mathbb{x}_i + \mathbb{b}] \geq 1 - {\zeta_i}^{\kappa} \qquad [11b]$$

$$\zeta_i^{\kappa} \geq 0 \qquad [11c]$$

where, $\mathbb{y}_i = \pm 1$, indicates the true classes. The variable C is exploited to trade-off the desired size of the discriminating margin and the flexibility allowed with misclassifications, and the selection of κ determines the regularization technique to establish a generalizable trained model. The data-points $\mathbb{x}_i$ that lie on the boundaries, $|\mathbb{y}_i f(\mathbb{x}_i)| = 1$ are denoted the *support vectors*, as they define the separating regions (Fig. 5). Further derivations to compute the weights and biases are outlined in Appendix 9.0.2A.

In many real-world applications, and especially in astrodynamics, the data is often not linearly separable; i.e., the data is non-separable. This challenge is met by mapping the available information to a higher dimensional space by employing the *kernel trick*, where a separating hyperplane for the same data exists. The derivations for computing

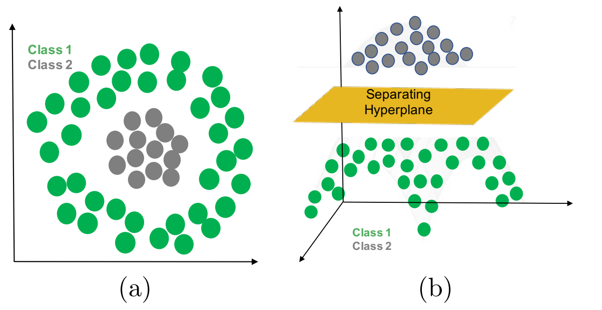


Fig. 6: Schematic demonstrating transformation of (a) non-separable data into (b) separable data in a higher dimension.

the weights, and biases, and the details associated with the classification-enabling transformation of the feature-space to a higher dimension are outlined in Appendix 9.0.2B.

B Generalizability

As discussed with ANNs, the generalizability of a model enables it to render accurate predictions on unseen data. The balance between training and testing accuracy is achieved by regularization measures, and the appropriate tuning of values for C and other hyper-parameters introduced during the mapping to a higher dimensional feature-space. The details associated with the generalization approach implemented in the investigation are discussed in Appendix 9.0.2C.

4.3 Heuristically Accelerated Reinforcement Learning

The generation of an initial guess trajectory that transfers a s/c from a departure location to a specific destination is cast as a routing problem. Das-Stuart et al.[17, 18] explore the advantages of employing heuristic methods over exact strategies to construct solutions in a large and complex design space. The automated agent (software)-based search for a path that achieves a specified goal, when subject to a set of prescribed constraints is summarized from these papers.

Any endeavor in the space environment is complex. A flexible and sustainable infrastructure benefits from a mission design approach that delivers transfer solutions while accommodating uncertainties in the environment, variable performance measures, and is responsive to shifting goals. Reinforcement Learning (RL) algorithms, originating in the artificial intelligence domain, are an effective tool in balancing sometimes conflicting goals. Implemented via an agent interacting with its environment, such an agent learns to deliver appropriate choices that lead to desirable outcomes. The reinforcement learning approach is commonly formalized as a Markov Decision Process (MDP)[41] that is constructed from a tuple: $< S, A, P, R >$; S then, is a set of states available to an agent; A is a set of actions available to an agent; P is $P(s,a,s')$, the probability that action a in state s leads the agent to arrive at state s', $R : S \times A \to \mathscr{R}$ is the reward received for an action a in state s. The aim is an optimal policy (π^*) that executes an action at a given state to maximize the cumulative rewards received by the agent over multiple episodes. A strategy to accomplish this task involves a value function ($\mathbb{V}^\pi$) that iterates over all the possible actions such that an originating state evolves the system to eventually arrive at the optimal choice ($\mathbb{V}^*$):

$$\mathbb{V}^*(s) \leftarrow \underset{a}{max} \sum_{s'} P(s,a,s')[R(s,a,s') + \gamma \mathbb{V}^*(s')] \quad [12]$$

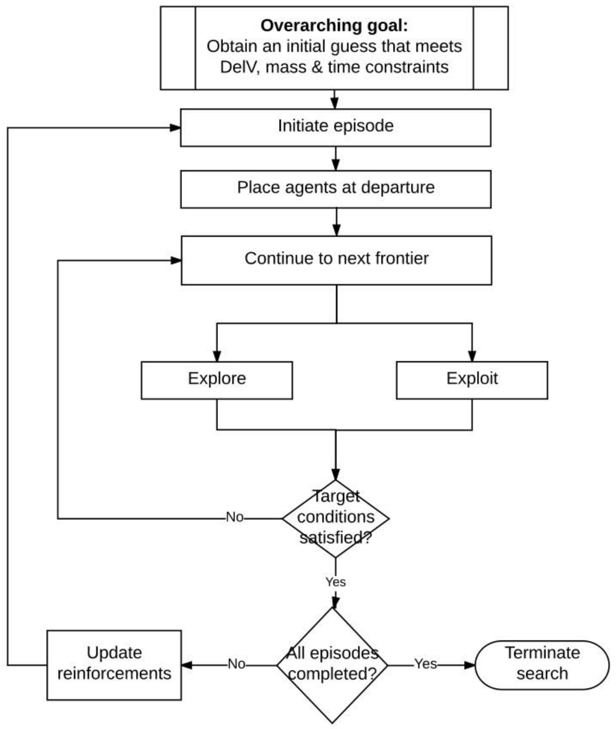
Equation [12] is essentially the *Bellman Equation*, employed widely in dynamic programming, and satisfies the necessary conditions for optimality. The prime symbol ($'$) indicates a state(s) accessible from the *frontier* of neighbors of the current state 's'. In complex regimes where the probabilities defining the system model and the rewards are not known a priori, it is useful to formulate the problem as a model-free, state-value pair approach. A state-value update function is formulated in recursive form as:

$$\begin{aligned}\mathbb{F}(s,a) \leftarrow \mathbb{F}(s,a) + \alpha[R(s,a,s') \\ + \gamma \underset{a'}{max}\, \mathbb{F}(s',a') - \mathbb{F}(s,a)]\end{aligned} \quad (13)$$

and forms the repository of reinforcements from which the agent learns the desirable behavior. The discount factor $0 \leq \gamma < 1$ is a measure of the balance between immediate and future rewards; smaller values of γ favor immediate rewards. The learning rate of the agent is specified by α. It diminishes over subsequent episodes to avoid fixation on only the most recently gained knowledge, and expands trust on prior knowledge that is posed as reinforcements over time. The optimal policy is then constructed as:

$$\pi^*(s) = arg\,\underset{a}{max} \sum_{s'} \mathbb{F}(s,a) \quad [14]$$

The convergence in response to such an optimal policy using the RL approach is guaranteed by re-visiting state-action pairs infinitely many times, and the process may be prolonged in scenarios subject to a large state-space. So, it is beneficial to introduce a heuristic function that accelerates the learning process by biasing the selection of an action a, given a state s, with a reward R. Such an algorithm is termed a Heuristically Accelerated Reinforcement Learning (HARL) algorithm, and the time-complexity depends on the accuracy of the heuristics. The ability of the heuristic function to influence the action-choice is based on the ability to exceed the variations in $\mathbb{F}(s,a)$.[42] Scenarios with large state-spaces and multiple objectives also motivate a coordinated effort in a multi-agent distributed HARL approach, particularly one that exploits parallel processing. In such a distributed network, the agents work cooperatively by updating a centralized reinforcement repository based on the knowledge (cumulatively discounted rewards) gained during a specific episode. Within the context of mission design, the agent continues the search until a stopping condition is satisfied, i.e., a terminal condition and/or a violation of constraints. An overview of the learning process (within the context of mission design) is illustrated in Fig. 7. Clearly,



Overarching goal:
Obtain an initial guess that meets DelV, mass & time constraints
Initiate episode
Place agents at departure
Continue to next frontier
Explore
Exploit
Target conditions satisfied?
No
Yes
All episodes completed?
No
Yes
Update reinforcements
Terminate search

Fig. 7: An implementation of HARL algorithm

the learning process within each episode is comprised of two search scenarios — exploration and exploitation. Exploration enables a training phase where the agent *learns* about likely consequences of actions in the environment; the exploitation phase enables the agent to engage in informed decisions by capitalizing on previously gained knowledge. The policy at a particular state as influenced by the state-action pair and heuristic is:[42]

$$\pi(s) = \begin{cases} \mathbb{E}(\mathbb{F}(s,a) \bowtie \xi \mathbb{H}(s,a)^{\beta}), & \text{if } q \geq p \\ a_{random}, & \text{otherwise} \end{cases} \qquad [15]$$

where ($\mathbb{H} : S \times A \rightarrow \mathscr{R}$) is the heuristic function, $\bowtie$ is a math operator as determined by the RL algorithm and its implementation (e.g., Q-learning, Ant Colony Optimization), ξ and β are weighting parameters that dictate the influence of the heuristic, and a_{random} is the action selected randomly from all those available in state s. The construction of the exploitation function $\mathbb{E}$ is specific to a particular application; some variants include a greedy strategy that incorporates $max(\mathbb{E})$, a minimax approach that minimizes the losses in a maximum loss scenario, and a 'softmax' process that stochastically selects an action.[43] In this equation, p, ($0 \leq p \leq 1$), is the trade-off parameter between exploration and exploitation, and q is a random value from a uniform distribution in [0,1]. Greater values of p encourage exploration and lower values bias the process toward exploitation. Extensive exploration in earlier episodes is beneficial, then a gradual shift to exploitation. A convenient function to control the steady-state value of the trade-off probability, p_{ss} by the k^{th} episode (Ep) is constructed as:[10]

$$p = p_{ss} + (1 - p_{ss})e^{(-\frac{k-1}{ln(Ep)})} \qquad [16]$$

The blending of heuristics with reinforcement learning is a powerful utility in solving many NP-hard and NP-complete pathfinding problems, e.g., the Traveling Salesman Problem (TSP).

5. Development of the Design Framework

The investigation in Das-Stuart et al.[17,18] demonstrates the ability to incorporate automated pathfinding to generate initial guesses, which are then numerically corrected and optimized to enable mission scenarios. The pathfinding process is facilitated by sequential construction of *Accessible Regions* (ARs) that enable a spacecraft to select natural conditions from within these accessible volumes, to maneuver towards the destination. Sequencing an initial guess from natural arcs is also a focus of the current analysis to (i) eliminate the a-priori discretized database generation phase and allow free-form searches based only on the system-dynamics, and to (ii) exploit supervised learning strategies to develop *flow-models*. This recent effort also sequences natural arcs that capture the *general characteristics* of the flow associated with notable periodic orbits families, rather than satisfy any set of exact conditions. The generation of accessible regions and the subsequent interfaces to the natural flow conditions and the pathfinding process serve as the foundation of the design framework.

5.1 Computation of Accessible Regions

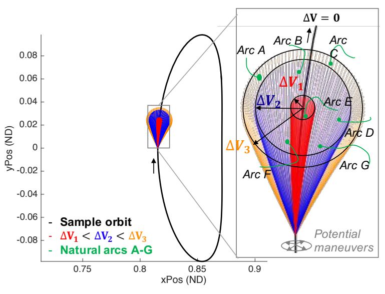
An accessible region establishes the *reach* of a s/c in the design space, and its characteristics are influenced by the spacecraft thrust-to-mass ratio, propellant efficiency, and other performance characteristics. The computation of an Accessible Region (AR) originates with a perturbation of the spacecraft current velocity within a circle/sphere (for the planar/spatial problem, respectively) with a prescribed radius, followed by a propagation of the perturbed and unperturbed states for a fixed time-horizon. The resulting downstream behavior includes a stretching of the perturbed states from the unperturbed natural arc due to the influence of the existing gravitational forces in the system. A measure of the deviation of the perturbed states from the end of the natural arc is exploited to formulate a circular/spherical accessible region in a planar/spatial setting. These simplified geometric shapes are leveraged over the realistic irregular-shaped ARs that result from a numerical propagation, to enable greater control over the quantity of natural arcs available to the s/c. Empirical testing has supported the decision to employ such an approximation. For both the chemical and low-thrust analyses, perturbations in the s/c velocity are introduced via chemical impulses; the magnitude is determined by the engine characteristics and the engine operation/burn time ($\delta\mathbb{t}$), i.e.,

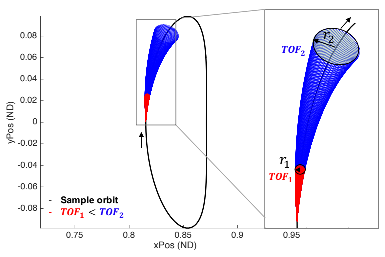
$$\delta V = Isp\ g_0\ ln(\frac{m_0}{m_f}) \qquad [17a]$$

where,

$$m_f = m_0 - (\dot{m} \times \delta\mathbb{t}) \qquad [17b]$$

Inspection of Eqn. [1] and Eqn. [17] illustrate that higher specific impulse values result in smaller δV perturbations for a fixed operation time and aid in delineating the behavior of propellant efficient low-thrust and less efficient chemical regimes. The determination of the perturbation modeled as a δV is not dependent on the horizon-time/propagation-time for the perturbed and unperturbed conditions. However, it influences the footprint of the AR. Figure 8(a) illustrates that larger δV's for a fixed horizon-time yield more extensive ARs and, therefore, a greater number of available natural arcs. Longer propagation times,


(a)


(b)

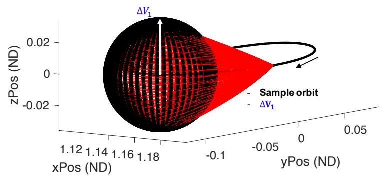

(c)

Fig. 8: Influence of δV_s on the accessible region footprints in the (a) planar case (c) spatial case. (b) Influence of TOF on the footprint of the accessible regions in the planar case.

for a fixed δV, produce a similar effect as plotted in Fig. 8(b) for planar applications and an AR in the spatial regime is illustrated in Fig. 8(c). Note that these ARs for reachable position and velocity conditions are 4-dimensional in the planar realm and 6-dimensional in the spatial problem.

5.2 Sample Conditions from Accessible Regions

Mission design is sometimes envisioned as harnessing the appropriate combination of orbit family geometries, velocities and energy levels to satisfy architecture constraints such as altitude relative to a primary body, line-of-sight for communications, as well as eclipsing conditions, and balancing such constraints with mission objectives. The CR3BP offers a multitude of natural solutions to enable such a combinatorial search — periodic and quasi-periodic orbits in the vicinity of the primaries and the libration points, and manifold behaviors that reflect the flow throughout the region. Only periodic orbit families are included in this analysis, because in many instances, resonant orbits (periodic) qualitatively capture the essence of manifold behavior as well. Das-Stuart et al.[17,18] discuss the inclusion of families within a searchable database based on the geometric semblance between them, during transfer construction. Figure 9 illustrates the advantage of including the Resonant 4 : 3 family in the searchable database when designing transfers between the L_1 and L_3 Lyapunov families; this resonant orbit offers intermediate arcs with suitable flows to transport the s/c between the departure and destination orbit families. In this case, the user controls the motion of the s/c to follow the flow associated with these three families in the searchable volume. The appropriate

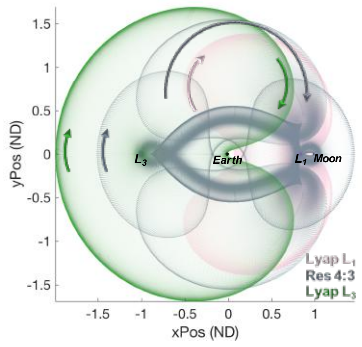


Fig. 9: Illustrating similarities and differences in geometry between the L_3 Lyapunov, Resonant 4:3 and L_1 Lyapunov families. Arrows indicate direction of flow within each family.

selection of intermediate families is a challenge, however, especially when the user is unfamiliar with a particular Three Body (3B) regime. So, it may be desirable to conduct an unconstrained exploration of the configuration-space to offer intuition about the flow in the dynamical regime and also assess the potential trade-space. Hence, there are benefits to both approaches: (i) a free-form search and (ii) a search through a compilation of known families.

5.2.1 Free-Form Search

Accessible regions aid in constraining the searchable design space at any given instant in time. However, with the lack of pre-discretized conditions (as available in Das-Stuart et al.[17, 18]), natural conditions are instantaneously generated within an AR, in real-time. Operating under the assumptions of a complex and highly nonlinear dynamical regime, however, these conditions are a blend of both ordered and chaotic motion. Ordered motion is distinguished from chaotic by context of the predictability of the behavior over all time within the CR3BP. However, it is beneficial to exploit any available and mission-enabling conditions that don't violate mission constraints. Thus, chaotic conditions may offer suitable candidates, if it is predictable over some specified and acceptable time-frame. A receding horizon technique is illustrated in Fig. 10, one that tests for such predictability. Long-term predictability is observed by

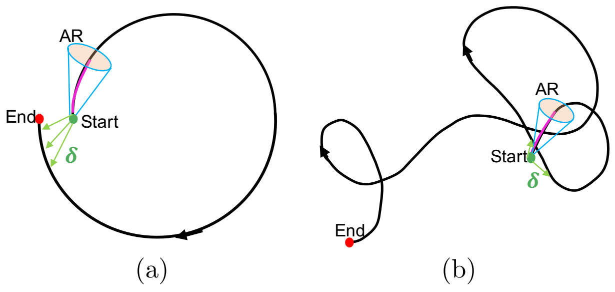


Fig. 10: Two sample scenarios (a) and (b) illustrating look-ahead trajectory segments from *Start* to *End*, the proximity tolerance, δ, and the relatively small fraction along the look-ahead segment to establish an Accessible Region (AR).

propagating a condition from within an AR for some lengthy duration; this look-ahead time-frame is a design parameter and varies based on the dynamical system and/or mission considerations. If any of the states along the propagation return to to the initial position and velocity, to within some specified tolerance value, δ, then the motion is confirmed to remain bounded in the 3B system, and not escape for the specified duration. Such a chaotic arc is set aside for selection by the s/c. Note from Fig. 10, only a small fraction along this propagation time is captured for constructing an AR. The ability to select chaotic states broadens the design options, and eliminates the time investment required to pre-generate a discretized database. This unrestricted approach offers the potential to uncover non-intuitive transfer geometries, otherwise not available from known and ordered natural families.

5.2.2 Flow-Models via Supervised Learning

A free-form search is especially powerful for identifying potential options for a new scenario where the design possibilities are unknown. However, the unrestricted approach diminishes the capacity to bound the motion to a particular geometric profile. Re-introducing an a priori discretized database comprised of select families addresses this challenge, but at the cost of increased computational memory, and risks the emergence of any solution, depending on the fineness of the implemented discretization. So, flow-models for these natural families are developed via ANNs and SVMs to mitigate such additional complications. The design strategy involves employing a regression network and alternatives to fully determine the states for the conditions populated within an AR and then confirming the legitimacy of their belonging to a family of interest via classification schemes.

Implementation of Regression Flow Models

The position and velocity attributes establish the unique identity of a state within a 3B system - assigning its 'belonging' to a particular periodic orbit family, and its location within the family as well. Thus, a first step in incorporating specific family conditions within the searchable volume involves resolving the identity of populated data points within each AR. This task is accomplished by training the neural networks to compute the velocity components associated with the positions in an AR. Therefore, the training input attributes must also be instantaneously extractable from within an AR. The position components of the points within an AR, and the general direction of flow as suggested by the in-plane and spatial velocity angles (ϕ and ψ) corresponding to the center AR state are such quantities. These attributes are exploited to train the regression models as illustrated in Fig. 11. Discretized conditions for each orbit

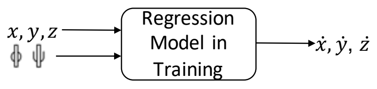


Fig. 11: Input and output components for a regression model in training

family are generated via the scheme discussed in Das-Stuart et al.[17,18] A subset of this discretized dataset ($3\% - 5\%$ of the total data, amounting to $\approx 60,000$ points) is further partitioned into training, validation and test data to train a periodic orbit family model. The Levenberg-Marquardt technique along with evaluation of the Mean Square Errors (MSE) and early stopping conditions as introduced in Section 4.1.1 are employed for the training process. Three-layer networks (2 hidden layer, 1 output layer) are observed as satisfactory in modeling the underlying relationships in the data for the various orbit families in this investigation. A modification to these neural network architecture components may be warranted for other scenarios.

The results associated with training both a planar and spatial orbit family are reported in Table 1. Each trained model is queried using the entire dataset

Family	Training MSE	Validation MSE	Testing MSE
(a)	$5.88e-08$	$6.30e-08$	$6.47e-08$
(b)	$1.84e-08$	$2.07e-08$	$2.62e-08$

Table 1: Quantification of training, validation and testing errors for sample periodic orbit family regression flow models. (a) L_2 Lyapunov, (b) L_2 Southern Halo.

for the corresponding periodic orbit family to assess their prediction capabilities for the remaining 99% of the unseen dataset. Figure 12 illustrates the outcomes for the planar L_2 Lyapunov family, and the spatial L_2 southern halo flow models. The largest

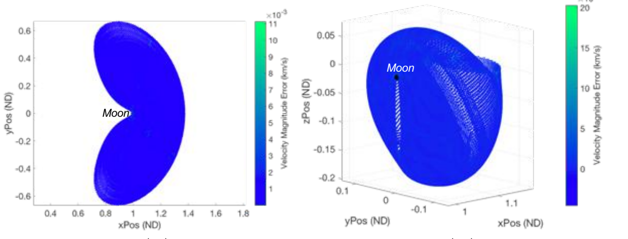


Fig. 12: Examples demonstrating ability of regression ANN flow models to predict the velocities for planar and spatial families.

discrepancies are observed to be near the primary. Further improvement on the training accuracy via further tuning of the ANN architectures is not pursued here, as these models are sufficient and successful in generating initial guesses for this preliminary investigation. During real-time pathfinding, the angular components (ϕ and ψ) for the AR center velocity only serve to aid in approximating the true velocity components associated with other positions in the AR. This implementation operates under the assumption that the flow-directions emanating from within a given AR are similar. Confidence in these approximations improves when verified by classification schemes. An inspection of the families in Fig. 13 demonstrates that a position can be non-unique for different velocity conditions, within the same family. The results in Fig. 14, however, illustrate that

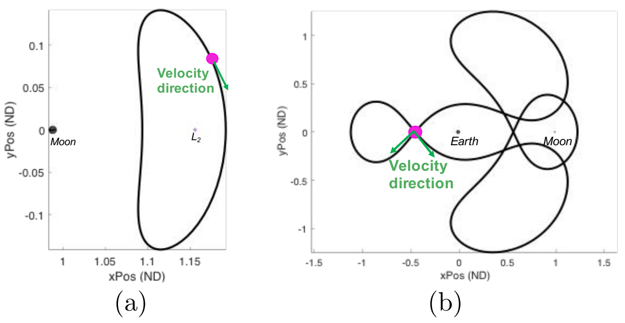


Fig. 13: (a) Each position along an L_2 Lyapunov orbit possesses a unique velocity direction. (b) Each position along a Resonant 4:3 orbit does not possess a unique velocity direction.

the training inputs are not always sufficient to offer accurate predictions for all conditions within the Resonant 4:3 family. However, a comparison of Figs. 13

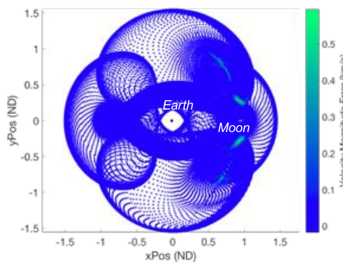


Fig. 14: Challenge for regression models to resolve velocities for overlapping position conditions, when no insight into the associated energy conditions is provided.

and 14 illustrate that large prediction errors do not, in fact, overlap at positions for a single orbit, but rather, where different family members overlap in position space. At these junctures, a lack of information about the differing energy levels (and, thus, velocity magnitudes) inhibits accurate predictions. However, the velocity magnitudes are not available as inputs to the training process; because the exact velocity for a particular position within an AR is not known, and, in fact, is the output. Thus, an alternative strategy in contrast to Fig. 11 is employed as per Fig. 15. Due to the absence of a regression model, a set of velocity magnitudes to characterize the AR are computed. These range from the spacecraft instantaneous veloc-

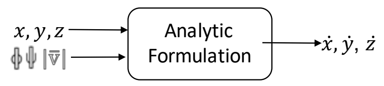


Fig. 15: Analytic approach to computing velocity component approximations

ity at the center of the specified AR to a maximum value that establishes the radius of the velocity AR. Incorporation of the velocity magnitude $||\overline{V}||$, along with the angular facets (ϕ and ψ), resolves the cartesian velocity components analytically. Thus, a sufficiently large pool of solutions is populated to introduce to the classification models for filtration and identification. Both the regression models and the analytical computations successfully support pathfinding by delivering *family-like* velocity conditions. The latter approach demands an increased online computational effort (i.e., during the pathfinding phase), whereas the former requires an offline time investment to train and pre-generate the regression models. The analytical approach is especially useful when the regression models are challenged to deliver predictions for the more complex periodic orbit families with multiple member overlaps. Nevertheless, the approximations and assumptions for computing the velocities aid the pathfinding agents in sequencing natural arcs that capture the *general* characteristics of the true conditions within each AR to generate a feasible initial guess - that eventually facilitates higher-fidelity analysis.

A regression network always produces an output based on the best-fit model determined during the training phase, even when the bounds on the input variables are violated. The analytical approach also always delivers a solution for the given inputs. So, recognizing invalid queries and disregarding the corresponding outputs is also incorporated - one approach might include allowable sets for the inputs over all the periodic orbit families for validation. However, this technique is not a failsafe process since families might assume complex structures in configuration space, possess self-intersections, or lack monotonic growth in space. Thus, classification models are also trained and adopted to judge the validity of a regressed output belonging to a particular family. For the sake of clarity, a trained regression model is labelled M_R and a trained classification model as M_C.

Implementation of Classification Flow Models

Classification of states into orbit families, or desirable and undesirable categories are typically executed via various techniques. One approach is training models that distinguish conditions between a select variety of families. However, if an input is introduced that does not belong to the available families in the trained classification model M_C, it is re-assigned to a family with the closest relationship. An alternative, more sophisticated approach trains separate models for each family - and a simple assessment determines if a given condition belongs. Thus, a binary classifier is trained. The two classes are - *Class1*: conditions from the desired family, and *Class2*: any other conditions that do not belong. This second class is termed *chaos* in association with a particular M_C. It is generated by introducing random perturbations (within a prescribed radius) in position and velocity from each of the family conditions (Fig. 16(a)). This perturbation radius is a design parameter. Figure 16(b) illustrates the binary classes that are employed to train a model for an L_2 Lyapunov family. The ANN and SVM binary classifiers are trained

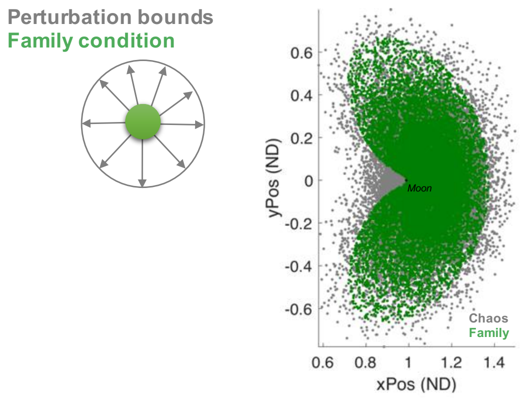


Fig. 16: (a) Generating chaos for family conditions. (b) Family and chaos conditions for the L_2 Lyapunov family.

by supplying labeled examples from the two-classes. They are employed to accept the position and velocity information for a certain condition, and then report the associated classification output/posterior probability. The condition then belongs to the class with the greater value for these probabilities. Figure 17 captures this process. These input conditions

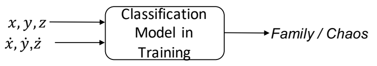
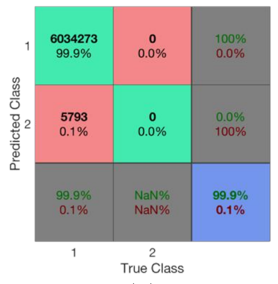


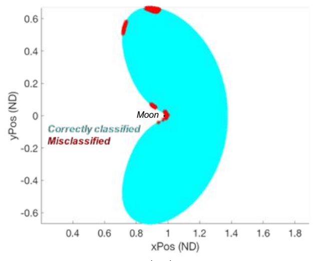
Fig. 17: Input and output components for a classification model in training

to an M_C can either be outputs from an M_R, or state conditions selected directly from an AR. Once each of the conditions from an AR passes the regression and classification tests, they become available for selection by the pathfinding module.

Similar to the discussion on the accuracy of the regression networks, the classification models are also tested for reliability. Figure 18 illustrates a *confusion* matrix that aids in quantifying the accuracy of predictions from a trained model. In this example, the trained model is one that distinguishes between the true L_2 Lyapunov conditions (class 1) and the chaotic conditions (class 2) as introduced in Fig. 16(b). The performance of the trained model is tested by assessing its ability to accurately classify the entire population of a discretized periodic orbit family. The confusion matrix in Fig. 18(a) illustrates that there are no chaotic conditions in the input dataset and that $\approx 0.1\%$ of the total data have been misclassified as chaos. These misclassifications are identified in



(a)



(b)

Fig. 18: Quantifying performance of trained L_2 Lyapunov ANN classification model on the full population of the periodic orbit family. Output layer probability cut-off = 0.5.

red in Fig. 18(b), and their locations are traced back to areas of relatively sparse L_2 Lyapunov conditions in Fig. 16, and also the highly nonlinear region near the primary (Moon). Some approaches to increasing this accuracy include incorporating additional data in the areas corresponding to misclassifications, and assessing the impact of varying the choices made for the ANN architecture as well. Hence, training the models is an iterative process.

The SVM classification models are trained via Gaussian kernels that transform the data to a higher-dimensional space where they are separable. The values for the corresponding kernel hyper-parameter γ (in Eqn. [39] in Appendix 9.0.2B) and box constraint C, that aid in training SVM models, are computed via cross-validation and Bayesian optimization[40] which minimizes the classification errors. Figure 19 illustrates a sample of the profile associated with optimizing the hyper-parameter values for the L_2 Lyapunov family SVM model using a subset of the larger data-set. The values of γ and C that correspond to the minimum for the error surface are extracted to train the entire training data-set to develop the final L_2 Lyapunov family SVM model. A grid-search across ranges of γ and C is also a valid option to determine suitable values for these hyper-parameters. Figure. 19 illustrates an example of the error surface that results during the Bayesian optimization process adopted to minimize the classification losses by varying the value of the hyper-parameters in unison (see Appendix 9.0.2C for details).

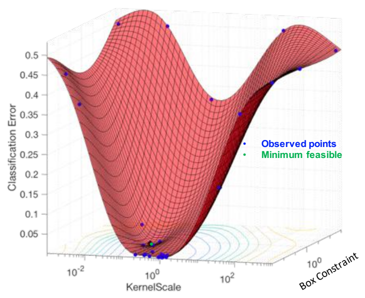
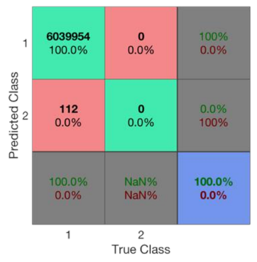
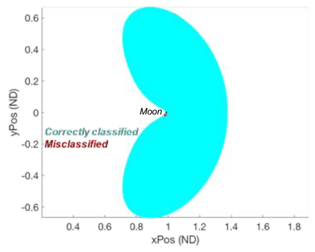


Fig. 19: Optimizing hyper-parameters in preparation for SVM training for the L_2 Lyapunov family.

The tuned hyper-parameters aid in constructing an optimized classifier, that subsequently leads to fewer classification errors on unseen data. Figure 20(a) reports the performance of the trained SVM model when it is directed to classify all the discretized conditions for the L_2 Lyapunov family. A comparison of the results to Fig. 18(a) demonstrates that the SVM classifier outperforms the ANN model in classifying the L_2 Lyapunov family conditions. A smaller number of data points have been misclassified as chaos in Fig. 20(a), but the misclassifications that do occur, exist closer to the primary (Fig. 20(b)), where the dynamics are highly nonlinear. Recall that these



(a)



(b)

Fig. 20: Quantifying performance of trained L_2 Lyapunov SVM classification model on the full population of the periodic orbit family. Posterior probability cut-off = 0.5.

binary classifiers are developed to identify any 'non-belonging' condition as *chaos*. Thus, an additional test considers the use of these models to also classify conditions belonging to other periodic orbit families as chaos. Referring to the previous discussion on the degree of overlap in position/velocity-space and also to the direction of flow along various periodic orbit families (e.g., Fig. 9), the L_2 Lyapunov SVM model accurately classifies all the L_4 short-period conditions as chaos (Fig. 21(a)), due to the low similarities between these families. In contrast, certain regions of the DRO family that overlap in position and possess a semblance to the L_2 Lyapunov family flow characteristics are more challenging for the L_2 Lyapunov SVM model to label as *chaos*. These hard-to-distinguish areas are colored in cyan within the highlighted box in Fig. 21(b). Since the output probabilities of the

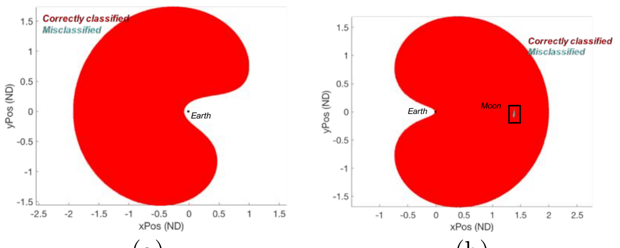


Fig. 21: Quantifying performance of trained L_2 Lyapunov SVM classification model by assessing its capability to identify other periodic orbit families as *chaos*. (a) L_4 short period family, (b) DRO family. The boxed region in (b) highlights the conditions that the L_2 Lyapunov model finds challenging to distinguish as a DRO/L_2 Lyapunov member due to the high geometric similarities between these families in this particular region.

output layer/posterior probabilities always sum to 1, the higher the probability output associated with the *family* class, the higher the confidence that the condition is not chaotic. Note that a lower cut-off does assign conditions as *belonging to a family* that actually do not truly belong to the test family, but qualifies as family-like (e.g., Lyapunov L_2-like) motion for selection during the pathfinding phase. Such a classification cut-off lever is, thus, exploited to capture the desired general characteristics of the flow associated with a particular family, for supporting the path-finding phase. It can be observed from Fig. 22, that the ANN M_C is not as successful at correctly classifying many Lyapunov L_2 conditions, compared to the SVM model. As mentioned earlier, the accuracy of the ANN models could be improved with additional training data. However, such an effort has not been pursued here, and the ANN classifiers are not incorporated during the pathfinding phase. The SVM models, even with very similar training conditions, offer superior and sufficiently accurate classification models to demonstrate one of the primary aims of this investigation, the ability to constrain the geometric profile of the initial guess to involve only specified families. Passing a classification test marks a regressed input-output pair (position and velocity states) as acceptable for selection from an AR by the pathfinding phase.

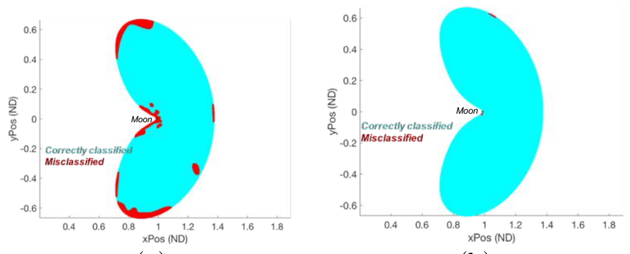


Fig. 22: Comparing performance of trained L_2 Lyapunov ANN and SVM classification models under similar training conditions, and highlighting the impact of selecting varied posterior-probability cut-offs (0.9 in this example). (a) ANN model (b) SVM model

5.2.3 Pathfinding via HARL Algorithm

Pathfinding is implemented to *sequence* the appropriate conditions for maneuvering the s/c from origin to destination. The available conditions are the natural arcs available within each constructed accessible region; they constitute the *frontier* of solutions that are selectable by the pathfinding agents. The overarching goal is posed as an optimization problem to prioritize the payload mass or the transfer duration in the persistent *mass-time* tradeoff challenge:

$$min\ \mathbb{J} = W_t t_f + W_p m_p \qquad [18a]$$

$$min\ \mathbb{J} = W_t t_f + W_p \Delta V \qquad [18b]$$

where, the propellant mass m_p or ΔV are always weighted against the transfer duration, t_f, such that the weighting on TOF, i.e., W_t is always 1, and W_p is a design variable. Note that δV in Eqn. [17] is the velocity perturbation to generate the ARs, and ΔV in Eqn. 18b represents the cumulative sum of the maneuvers executed within each AR to reach the destination. The transfer duration is the aggregate sum of the horizon times associated with each natural arc assembled within the transfer sequence, and the propellant consumption/ΔV_s are computed as a result of the velocity discontinuities between each of the arcs based on the ideal rocket equation. The schematic of an AR in Fig. 8(a) illustrates that exercising the maximum δV at a given node locates a s/c on the circumference of its associated AR, whereas a coast arc places it in the center of this region. So, the δV at a given node to reach another node within an AR lies in the interval $[0, \delta V]$. The associated mass updates are computed using Eqn. [17].

A HARL algorithm does not guarantee optimality due to the inclusion of heuristics, but multiple search episodes can enable convergence to a nearly-optimal solution to address the mass-time priority objective. The parallelizable capabilities within HARL offers a flexible architecture where the inter-node costs are computed on-demand as the agent progresses through the database towards the destination. As a result, either equation in Eqn. [18] may be adopted based on the priorities of the user. Given the process in Figure 7, the agent initiates the search by randomly exploring the state-space and gradually employs more exploitation over subsequent episodes as devised by Eqn. [16]. The exploitation of a particular node from an AR is guided by a heuristic constructed as:

$$\mathbb{H} = \frac{m_f^{W_m}}{d^{W_d}} \qquad [19a]$$

$$\mathbb{H} = \frac{1}{\Delta V^{W_v} \times d^{W_d}} \qquad [19b]$$

$$\text{where}, d = ||\boldsymbol{x} - \boldsymbol{x}_T|| \qquad [19c]$$

and $\boldsymbol{x}$ is the full state vector associated with a particular node, and $\boldsymbol{x}_T$ is the full state vector corresponding to the target condition(s). The heuristic, $\mathbb{H}$, is crafted to accommodate two aspects during the decision making process for selecting a node from a frontier: (a) the discontinuities in velocity or mass loss between selected arc segments (ΔV, m_f); and (b) a measure of 'goodness' d, for the selected node in terms of its proximity to the target condition(s). The weights are also design variables, where W_v or W_m grant the user control over the desired s/c performance (ΔV and propellant consumption), and W_d influences the manner in which the transfer trajectory tends towards the destination. These qualities also render the heuristic applicable as a measure of reward for a select number of agents ranked by their overall performance (Eqn. 18) for successful completion of the transfer. Specifically, the reward is computed as the accrued sum of the heuristic values ($\mathscr{R} = \sum \mathbb{H}$) along a successful transfer path, and is assigned to each contributing node. The automated pathfinding strategy aims to establish general desirable pathways toward the destination rather than identify distinct and discrete beneficial nodes in configuration space. So, neighboring nodes along the successful pathways are also reinforced using the same rewards (Fig. 23). As a consequence, additional local variability is introduced even during the exploitation phase to encourage local improvements during the search for a pathway that addresses the global objectives. The sheer volume of the state-space variable combinations prompts the tracking of a node's relevance to the transfer by recording its cumulative rewards over the evaluated episodes within a global repository. This implementation is a variation on the traditional approach in reinforcement learning algorithms where a table of the link quality between the nodes is maintained. As a result, concerns regarding the link quality on an arc between particular nodes that is potentially identical regardless of the originating node is allayed by enforcing ARs that constrain the accessibility between nodes. This adjustment also sets the discount factor γ equal to zero such that only the immediate rewards are solely valued. This decision is also supported by the fact that the node-to-node cost is variable and is a function of the s/c mass history. Note that the reward function may require scaling in Eqn. [13] to ensure that the bracketed term remains positive. Furthermore, the quality assigned to a node is simply a guide in the overall search strategy, as stochasticity is introduced in the selection of a favorable node from within an AR with the probability $\mathscr{P}$:

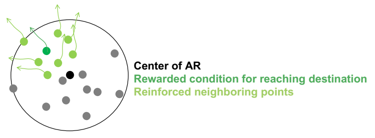

Center of AR
Rewarded condition for reaching destination
Reinforced neighboring points

Fig. 23: Reinforcing neighbors of the rewarded condition within an AR to aid the pathfinding process and also introduce further variability to the stochastic search.

$$\mathscr{P} = \frac{\mathbb{F}_i \ \mathbb{H}_i}{\sum \mathbb{F}_i \ \mathbb{H}_i} \qquad [20]$$

This probability $\mathscr{P}$ emphasizes two important considerations that bias the selection of a node i: (a) consistency in the relevance of the node to the transfer via its accrued rewards over prior episodes ($\mathbb{F}_i$), and (b) its current attractiveness as represented by the heuristic $\mathbb{H}_i$. The previous measures are enforced within a distributed, cooperative environment where the incremental knowledge acquired by each agent is combined to generate the final transfer trajectory that satisfies the goals and the constraints imposed by the user.

5.2.4 Initial Guess to Higher-Fidelity Solution

In this investigation, a final solution is defined as an end-to-end transfer trajectory that adheres to the constraints imposed within a CR3BP regime augmented by the selected propulsion forces. Note that such a solution is still merely an initial guess for a simulation in a higher-fidelity ephemeris model. The information associated with the arcs assembled by the automated pathfinding process is sufficient to initiate the numerical corrections process described in Section 2, namely the position, velocity, mass, thrust direction, thrust magnitude and time estimates. The user selects a desired computational tool for executing the convergence/optimization process. Note that varying thrust magnitudes are interspersed within the lower-fidelity solution, thus, it can serve as an initial guess for a range of engine capabilities in a higher-fidelity simulation. However, the range of thrust magnitudes is bounded in the corrections updates.

6. Results

The execution of the steps in the design framework yield initial guesses for diverse transfer scenarios and seeds the transition of these solutions to a higher-fidelity engine model. The chemical-engine results are constructed for a s/c with the following characteristics: $Acceleration = 0.036\ m/s^2$, $Mass = 500\ kg$, $Isp = 224\ seconds$, $Thrust = 18\ N$. All low-thrust results assume the following low-thrust s/c specifications: $Acceleration = 2.2e-4\ m/s^2$, $Mass = 1000\ kg$, $Isp = 4000\ seconds$. The results are organized to exhibit the following capabilities of the framework:

1. Planar transfers
   DRO to DRO
   (a) Establish proof-of-concept of free-form search
   (b) Transfer between two stable orbits
   (c) Customize transfers for different s/c specifications in the planar realm

   L_1 Lyapunov to DRO
   (a) Demonstrate advantages of manifold behavior for unstable departure orbit
   (b) Exploit free-form search to explore vast trade-space; exploit trained models for restricted searches

2. Spatial transfers
   Southern L_2 NRHO to Southern L_2 Flat Halo
   (a) Demonstrate advantages for reverse-time seeding in a higher-dimensional spatial realm, especially for departure from a stable orbit
   (b) Exploit free-form search to explore vast trade-space; exploit trained model for restricted searches

   Southern L_2 NRHO to DRO
   (a) Demonstrate capability to transfer between the spatial and planar realms
   (b) Demonstrate transfers between stable orbits in the spatial realm
   (c) Manage size of accessible regions (via maneuver size) to influence transfer geometry
   (d) Customize transfers for different s/c specifications in the spatial realm

All initial guesses that emerge via pathfinding are converged and optimized to continuous solutions using traditional numerical corrections and optimization techniques (Section 3). In Fig. 24, the free-form pathfinding strategy yields a straightforward transfer between two DROs in a stable regime for both a chemical and low-thrust s/c. The TOF, in particular, correlates reasonably well with the ΔV and mass-optimal solutions for the chemical and low-thrust solutions, respectively, in Fig. 24. The initial guess for the delivered mass fraction is also comparable, although the optimization process improves considerably upon the initial guess for the chemical s/c. In this particular scenario, the transfer geometry and performance traits are similar between the chemical and low-thrust s/c. The search for connections in an unstable regime, however, elicit a greater diversity in transfer options for any s/c type, especially due to the existence of manifold structures that are a natural transport mechanism in a multi-body regime. Figure 25 illustrates the direction and span of natural flows that can be expected to depart the originating Lyapunov orbit. Accordingly, the free-form search for connections from the Lyapunov orbit to the DRO illustrated in Fig. 26 are not restricted to the vicinity of these orbits; many geometries that flow towards the Earth and return to the originating/destination orbit vicinities are influenced by the natural flows that emanate from the L_1 Lyapunov orbit. Geometries that extend through the exterior of the system prior to reaching the destination are also uncovered via the free-form search. Each of these solutions possess varied implications for satisfying mission constraints and also performance outcomes. The designer gains the ability to improve or expand intuition about the solution space for the given transfer scenario. Also notable, trades between the different options are available to determine a viable option for

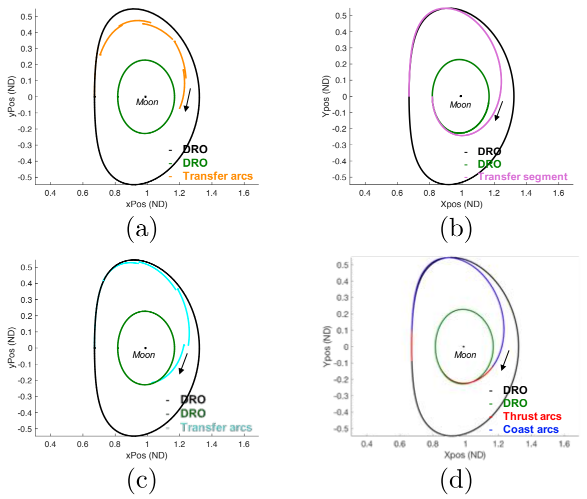


Fig. 24: Examples demonstrating the capability of free-form search to support chemical and low-thrust engine options. Departure DRO: $JC = 2.85$, Destination DRO: $JC = 2.935$.
(a) Chemical Initial Guess: Estimated TOF $= 14$ $days$, Estimated $\Delta V = 113.54 m/s$.
(b) Chemical ΔV Optimal: TOF $= 18.44$ $days$, $\Delta V = 73.28 m/s$.
(c) Low-Thrust Initial Guess: Estimated TOF $= 15$ $days$, Estimated $\frac{m_f}{m_0} = 99.61\%$.
(d) Low-Thrust Mass Optimal: TOF $= 18$ $days$, Thrust Duration $= 5.56$ $days$, $\frac{m_f}{m_0} = 99.73\%$, $\Delta V Equivalent = 105.82 m/s$

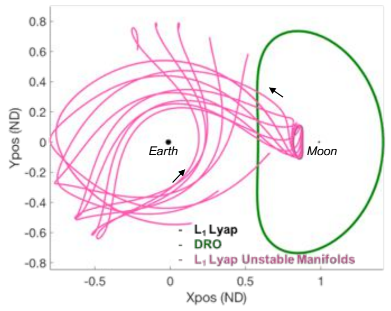


Fig. 25: Unstable manifolds of L_1 Lyapunov Departure Orbit

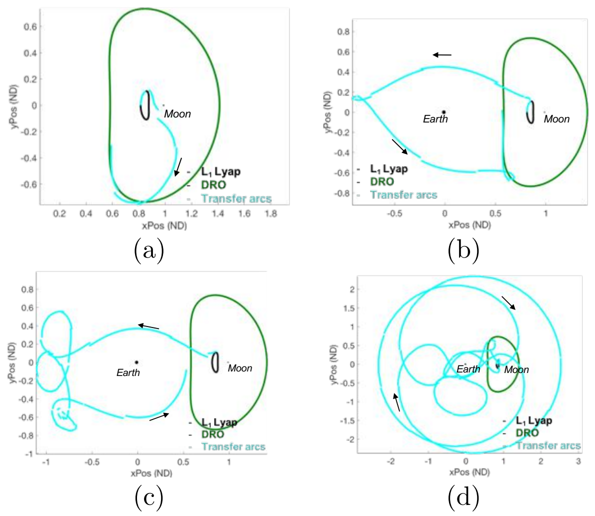


Fig. 26: Free-form search to uncover varied transfer geometries between L_1 Lyapunov (JC: 3.14) and DRO (JC: 3.7847) for initial approximations.
(a) Estimated TOF $= 24$ $days$, Estimated $\frac{m_f}{m_0} = 99.52\%$.
(b) Estimated TOF $= 45$ $days$, Estimated $\frac{m_f}{m_0} = 98.89\%$.
(c) Estimated TOF $= 108$ $days$, Estimated $\frac{m_f}{m_0} = 97.74\%$.
(d) Estimated TOF $= 147$ $days$, Estimated $\frac{m_f}{m_0} = 96.43\%$.

the mission. Although the advantage of the free-form search is its potential to establish variety across solutions, it can be challenging to restrict the search to specific areas of the configuration space. Exploiting the regressed and classified flow-models are one option to address such a limitation. For example, the desirable connection profile may be consistent with motion closely following the flows in the families of the departure and destination orbits (much like the geometry identified in Fig. 26(a)), then the L_1 Lyapunov and DRO family flow-models might be added to deliver sufficient options. Adding further insight to the transfer trade-space, examples of yet different links between the orbits are illustrated in Fig. 27; these transfer profiles are now restricted to motion belonging to either the L_1 Lyapunov or DRO families - the transfer arcs exhibit L_1 Lyapunov-like or DRO-like behavior. Optimized low-thrust solutions for the initial guesses in Fig. 26(b) and Fig. 27(c) are captured in Fig. 28. The free-form and model-based pathfinding is extendable to the spatial realm as well. The search dimensions are, however, expanded when including the out-of-plane position and velocity components during the search process, thus, leading to potentially increased computational resource requirements to uncover transfer options. As such, rather than a blind search, one option is to propagate the manifolds of the departure and/or destination orbits

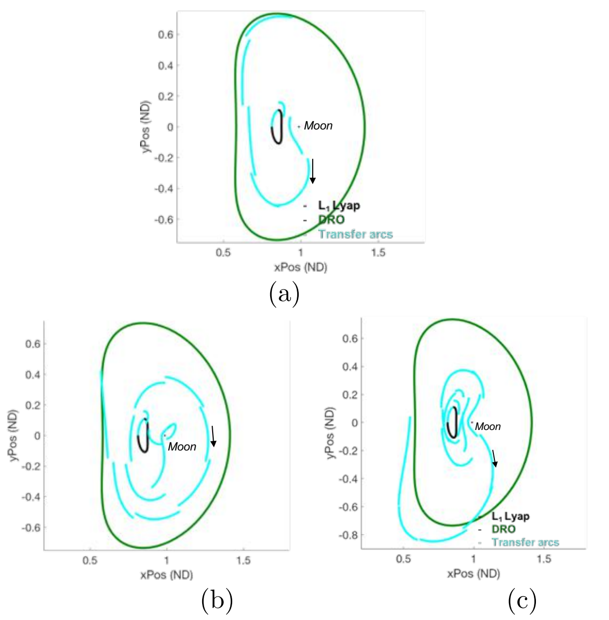


Fig. 27: Restricting search space using trained flow models to transfer between L_1 Lyapunov (JC: 3.14) and DRO (JC: 3.7847)
(a) Estimated TOF = 21 *days*, Estimated $\frac{m_f}{m_0} = 99.34\%$
(b) Estimated TOF = 45 *days*, Estimated $\frac{m_f}{m_0} = 98.31\%$
(c) Estimated TOF = 60 *days*, Estimated $\frac{m_f}{m_0} = 97.64\%$.

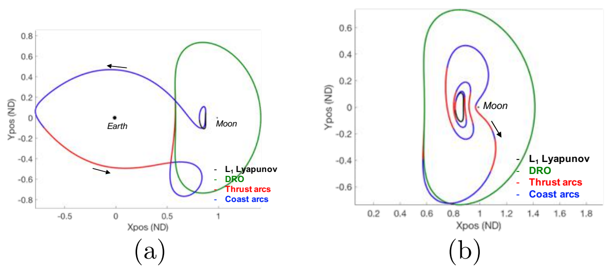


Fig. 28: Optimized low-thrust solutions constructed from initial guesses in Figs. 26 and 27.
(a) TOF = 52.32 *days*, Thrust Duration = 11.67 *days*, $\frac{m_f}{m_0} = 99.43\%$, $\Delta V Equivalent = 222.45 m/s$
(b) TOF = 58.29 *days*, Thrust Duration = 20.48 *days*, $\frac{m_f}{m_0} = 99.01\%$, $\Delta V Equivalent = 391.16 m/s$

to aid with seeding potential pathways. As an example, the departure orbit in Fig. 29 (southern L_2 NRHO with a periapsis altitude of $\approx 1763\ km$) is stable and possesses no unstable manifolds. So, the stable manifolds of the destination halo orbit (periapsis altitude of $\approx 49,215\ km$) are propagated in reverse-time to gain intuition into the transfer problem. It is evident from Fig. 29 that both interior

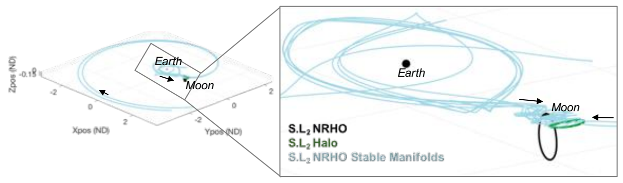


Fig. 29: Stable manifolds approaching southern L_2 halo orbit

and exterior flows toward the planar halo orbit exist. However, the manifolds for this orbit all remain relatively planar as well; no obvious connecting pathways from the NRHO to the destination orbit, especially in terms of the exterior flows exist. So, an alternative strategy employs the accessible region generation process in reverse-time. The states within ARs generated in reverse time from the arrival orbit should lead to this destination orbit in forward time. The states along the destination orbit are perturbed by the maximum ΔV possible for the selected engine burn-time and for the given s/c capabilities (as discussed in Section 5.1), and are propagated in reverse time for a duration to create an accessible region. The state conditions within this first accessible region are then each similarly perturbed and propagated in reverse time for the specified time horizon; this process is repeated multiple times as deemed appropriate for the specific problem under consideration. Figure 30 illustrates the resulting *seed* states in configuration space. Although these waypoints follow a similar geometrical path to the stable manifolds as illustrated in Fig. 29, this latter approach produces spatial waypoints between departure and destination orbits as well. The seed states, pre-configured with reinforcements (Fig. 30) are introduced to the reinforcement learning paradigm to construct transfers from the NRHO to the planar halo. Examples of two transfer geometries from such a search are plotted in Fig. 31. The results in this figure illustrate that the stochastic nature of the reinforcement learning paradigm ensures variety in the transfer geometries, even with the inclusion of *suggested* waypoints. One option to investigate interior transfers between the orbits is restricting the search to within the trained flow-model for the southern L_2 halo fam-

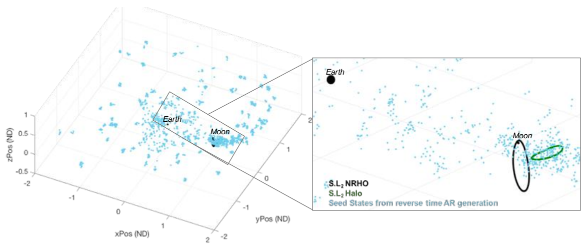
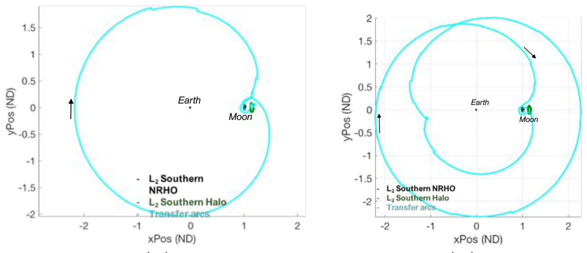


Fig. 30: Seed states resulting from accessible region generation performed in reverse time from destination orbit (L_2 southern halo)

.



(a) (b)

Fig. 31: Free-form spatial transfer example geometries from southern L_2 NRHO ($rPAlt = 1763.31km$) to southern L_2 halo ($rPAlt = 49,215.45km$). Note, rP $\rightarrow$ Periapsis.
(a) Estimated TOF = 75 *days*, Estimated $\frac{m_f}{m_0} = 95.39\%$
(b) Estimated TOF = 129 *days*, Estimated $\frac{m_f}{m_0} = 92.92\%$

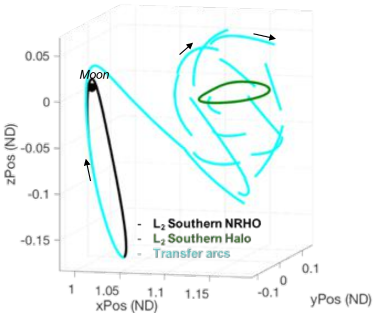
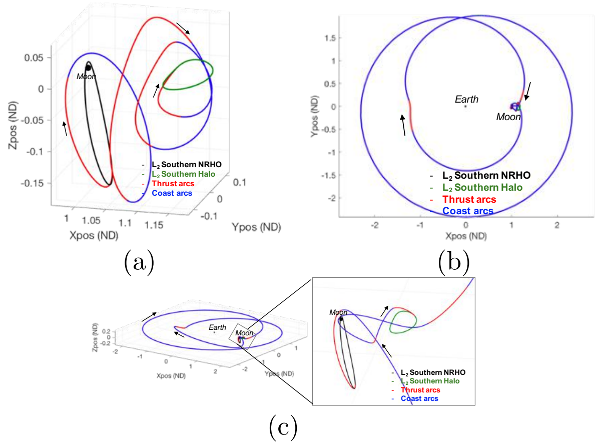


Fig. 32: Trained flow-model enabled spatial transfer example from southern L_2 NRHO ($rPAlt = 1763.31km$) to southern L_2 halo ($rPAlt = 49,215.45km$). Note, rP $\rightarrow$ Periapsis.
Estimated TOF = 42 *days*, Estimated $\frac{m_f}{m_0} = 96.42\%$

ily. Such a transfer profile is constructed in Fig. 32. The free-form and model-based initial guesses possess the state, time, thrust magnitude and direction histories that are input to the optimization scheme to produce the low-thrust continuous transfers that appear in Fig. 33. The low-thrust continuous solutions maintain a similar geometry and comparable performance measures to those declared for their corresponding initial guesses. The interior transfer in the scenarios computed in Fig. 33(a) enables faster transport between the departure and destination, but with one fewer lunar flyby. Consequently, the increased thrust duration elicits higher propellant consumption than the exterior transfer.

The reverse-time seeding technique is especially useful in illuminating pathways under circumstances where no natural flows exist into/out of stable orbits in the spatial realm, as with the southern L_2 NRHO and DRO (Fig. 34). In this example, an alternative strategy to flow-models is restricting the nature of the transfers to produce smaller accessible regions and induce a more gradual exit from the NRHO. Das-Stuart et al.[17,18] share the process and examples such that the time horizon associated with constructing the ARs is reduced to diminish their size



(a) (b)

(c)

Fig. 33: Mass optimal low-thrust transfers from L_2 NRHO to southern L_2 halo - constructed from initial guess in Figs. 31 and 32.
(a) TOF = 48.48 *days*, Thrust Duration = 26.86 *days*, $\frac{m_f}{m_0} = 98.70\%$, $\Delta V Equivalent = 513.85m/s$
(b) TOF = 131.32 *days*, Thrust Duration = 23.88 *days*, $\frac{m_f}{m_0} = 98.84\%$, $\Delta V Equivalent = 456.56m/s$
(c) Side-view of (b)

and, thus, restrict the variety of natural arcs available. Activating the low-thrust engine over a time-horizon of 3 days in Fig. 31 results in an equivalent $\Delta V \approx 57\ m/s$ per transfer arc from the NRHO. This ΔV kick, along with the additional boost received from the Moon during the NRHO departure produces large ARs and exposes the s/c to arcs that lead to external transfers. Reducing the time horizon to 2 days reduces the 'kick' to an equivalent $\Delta V \approx 38\ m/s$ and shrinks the size of the ARs sufficiently to enable interior transfers as well. Similarly, a 1 hour engine-burn time and 1 day propagation duration for the chemical s/c enables interior transfers as well. The appropriate burn duration and the corresponding ΔV magnitude for a particular scenario is currently determined empirically. So, the example in Fig. 34 reveals that trained flow models are not 'necessary' to construct interior transfers, even though exploiting the models may be beneficial for extracting additional geometries as demonstrated. Note that the estimated TOF for

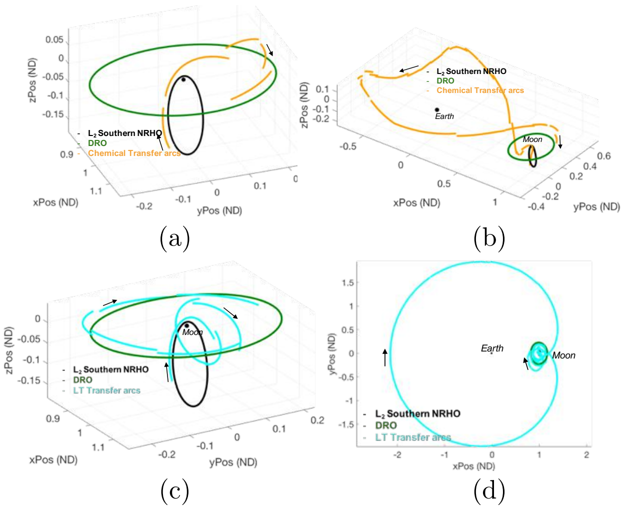


Fig. 34: Manipulating engine-burn time and thus, maneuver and AR size to construct initial guesses linking the spatial and planar realm between two stable orbits (free-form searches only). Departure: NRHO ($rPAlt = 1763.31km$) to DRO (JC: 2.935, Period: $\approx 13days$). Note, rP $\rightarrow$ Periapsis.
(a) Chemical Interior: Estimated TOF = 9 *days*, Estimated $\Delta V = 711.61 m/s$.
(b) Chemical Exterior: Estimated TOF = 48 *days*, Estimated $\Delta V = 2.64 km/s$.
(c) Low-Thrust Interior: Estimated TOF = 20 *days*, Estimated $\frac{m_f}{m_0} = 97.93\%$.
(c) Low-Thrust Exterior: Estimated TOF = 126 *days*, Estimated $\frac{m_f}{m_0} = 90.16\%$

the exterior transfer initial guess in 34(b) is high at $2.64\ km/s$. This initial guess is constructed by implementing maneuvers at 1 day time-intervals; in reality, a chemical transfer would consist of fewer burns and longer ballistic arcs. However, once the initial guess is acquired, numerical corrections and optimization eliminate undesirable maneuvers and render a more realistic chemical transfer solution (Fig. 35). The

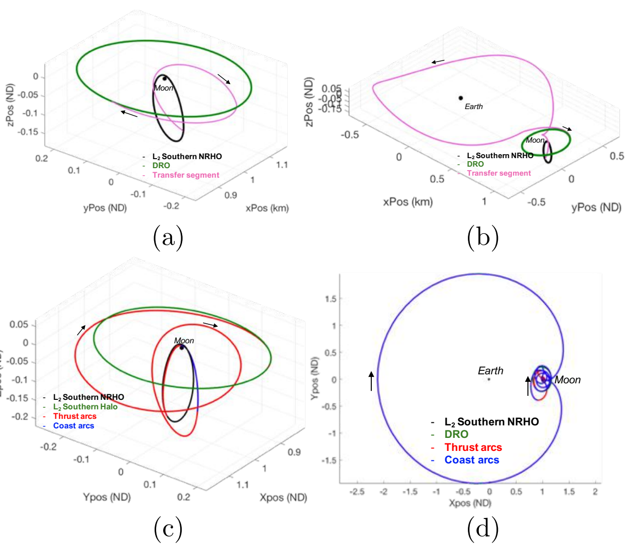


Fig. 35: ΔV Optimal chemical and mass-optimal low-thrust solutions from free-form search initial guesses in Fig. 34.
(a) Chemical Interior ΔV Optimal: TOF = 10.78 *days*, $\Delta V = 481.04\ m/s$.
(b) Chemical Exterior ΔV Optimal: TOF = 50 *days*, $\Delta V = 626.48\ m/s$.
(c) Low-Thrust Interior Mass Optimal: TOF = 28.22 *days*, Thrust Duration = 26.64 *days*, $\frac{m_f}{m_0} = 98.71\%$, $\Delta V Equivalent = 509.67 m/s$
(d) Low-Thrust Exterior Mass Optimal: TOF = 124.16 *days*, Thrust Duration = 21.17 *days*, $\frac{m_f}{m_0} = 98.97\%$, $\Delta V Equivalent = 404.54 m/s$

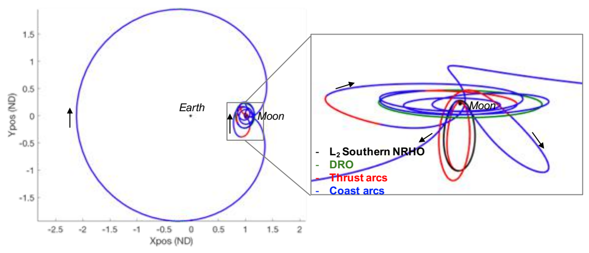


Fig. 36: Zoomed-in view of solution in Fig. 35(d).

initial guesses prove sufficient in enabling the chemical and low-thrust spacecraft to transition between varied energy levels, inclinations and locations in configuration space.

7. Concluding Remarks

Trajectory design is a careful balance that juggles diverse constraints, priorities and requirements to enable successful missions. Additionally, operating in highly nonlinear and chaotic regimes offers an infinitely large combinatorial optimization problem - one that is intractable to explore thoroughly via a manual approach. So, an automated search strategy is sought where design efforts are refocussed on defining the constituents in support of the broader mission goals, and machine learning strategies are incorporated to examine the combinatorics and offer attractive solutions. The design of end-to-end trajectories is facilitated by constructing a framework composed of four essential steps - (1) simulating the *reach* of the s/c to assess the regions it can access in the dynamical regime, (2) identifying the natural conditions 'on-the-fly' that are available within these accessible regions (AR), (3) implementing automated pathfinding via reinforcement learning to sequence the natural arcs and formulate a discontinuous yet complete route to the destination, and (4) transitioning the solution to a higher-fidelity engine/dynamical model via a numerical corrections process.

The investigation leverages the strengths of various branches of machine learning to establish a successful design framework. Distributed and cooperative reinforcement learning agents undertake computationally efficient pathfinding strategies assisted by (i) a free-form scheme, and (ii) a trained regression and classification model-based natural condition generation scheme. A free-form search exploits solely the system dynamics to instantaneously generate natural conditions within an AR for the pathfinding agents. Searches adopting this scheme in the higher-dimensional spatial realm benefit from seed-waypoints generated via reverse-time AR computations from the destination. In contrast, the pattern-recognition and predictive capabilities of Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs) are exploited to expose links exhibiting useful approximately periodic behavior, to transport the s/c to the destination.

Both free-form and model-based approaches possess unique strengths. The former empowers the pathfinding agents to explore in an unconstrained manner and, thus broadens design options via - traversal of both chaotic and ordered motion, revealing otherwise inconceivable transfer structures. A broad trade-space may be exposed, and any insight into the dynamical regime and the particular transfer scenario is introduced. The trained models restrict the search-space to desired geometries, leverage insight and eliminate challenges associated with a discretized database. The inherent variety in solutions established by these different techniques enable flexibility in the design framework and offset converges issues associated with some traditional trajectory design approaches.

The results in this analysis demonstrate benefits of machine learning strategies to yield initial guesses that harbor the required state and time histories to initiate numerical corrections. Even the ground work for advantageous thrust and coast locations is informed by the freedom to construct transfer segments with nil to a maximum allowed position and velocity discontinuity within the ARs. Applicable to both the planar and spatial realms, the mass-time performance reports from the sequencing of natural arcs have in general, proven to be reliable estimates of the performance metrics following the optimization process. The ability to generalize this design capability across various engine platforms (low-thrust and chemical) is realized by incorporating a spacecraft's particular specifications when establishing its *reach* via ARs, during each step of the pathfinding process.

The essence of this investigation is exploring the potential of automated pathfinding and machine learning schemes to aid with trajectory design. The findings reveal that the cadence set in motion does not replace human input. For example, the model-based approach is only as good as the human subject matter expertise which influences the design choices during training. Furthermore, revelations by the machine learning explorations bear the potential to encourage more sophisticated modifications/fine-tuning efforts by humans to deliver better mission-enabling solutions — the design efforts continue to remain an iterative process, albeit, with enhanced and quicker insights into the mission scenario.

8. Acknowledgements

This investigation has been conducted in conjunction with the NASA Goddard Spaceflight Center, through NASA Grant NNX17AH89G. The authors also wish to acknowledge the facilities at the Purdue Eliasen Visualization Laboratory, and thank the Purdue Multi-Body Dynamics research group, as well as Jeffrey Stuart and Roby Wilson at NASA JPL, for valuable insights and suggestions.

References

[1] D. C. Folta, N. Bosanac, D. Guzzetti, K. C. Howell, "An Earth-Moon System Trajectory Design Reference Catalog," *Acta Astronautica*, Vol. 100, pp. 341-353, May-June 2015.

[2] D. Guzzetti, N. Bosanac, A. Haapala, K. C. Howell and D. Folta, "Rapid Trajectory Design in the Earth-Moon Ephemeris System via an Interactive Catalog of Periodic and Quasi-Periodic Orbits," *Acta Astronautica*, Vol. 126, pp. 439-455, September - October 2016.

[3] M. Vaquero and K. C. Howell, “Leveraging resonant orbit manifolds to design transfers between libration point orbits in multi-body regimes,” *Journal of Guidance, Control and Dynamics*, Vol. 37, No. 4, 2014, pp. 1143-1157.

[4] A. Haapala and K.C. Howell, “Representations of Higher-Dimensional Poincaré Maps with Application to Spacecraft Trajectory Design,” *Acta Astronautica*, Vol. 96, March-April 2014, pp. 23-41.

[5] F. Topputo, M. Vasile, and F. Bernelli-Zazzera, “Low Energy Interplanetary Transfers Exploiting Invariant Manifolds of the Restricted Three Body Problem,” *The Journal of Astronautical Sciences*, Vol. 53, No. 4, October-December 2005, pp. 353-372.

[6] G. Mingotti, F. Topputo and F. Bernelli-Zazzera, “Efficient Invariant-Manifold, Low-Thrust Planar Trajectories to the Moon,” *Communications in Nonlinear Science and Numerical Simulations*, Vol. 17, No. 2, February 2012, pp. 817-831.

[7] R. Pritchett, K.C. Howell, and D. Grebow, “Low-Thrust Transfer Design Based on Collocation Techniques: Applications in the Restricted Three-Body Problem,” *AAS/AIAA Astrodynamics Specialist Conference*, Columbia River Gorge, Stevenson, Washington, August 21 - 24, 2017.

[8] G. Radice and G. Olmo, “Ant Colony Algorithms for Two-Impulse Interplanetary Trajectory Optimization,” *Journal of Guidance, Control and Dynamics*, Vol. 29, No. 6, 2006, pp. 1440-1444.

[9] M. Ceriotti and M. Vasile, “MGA Trajectory Planning With an ACO-Inspired Algorithm,”*Acta Astronautica*, Vol. 67, No. 9-10, 2010, pp. 1202-1217.

[10] J. Stuart, K. C. Howell, and R. Wilson, “Design of End-to-End Trojan Asteroid Rendezvous Tours Incorporating Potential Scientific Value,” *Journal of Spacecraft and Rockets*, Vol. 53, No. 2, 2016, pp. 278-288.

[11] R. Furfaro and R. Linares, “Waypoint-Based Generalized ZEM/ZEV Feedback Guidance for Planetary Landing via a Reinforcement Learning Approach,” *3rd IAA Conference on Dynamics and Control of Space Systems*, Moscow, Russia, May 30 - June 1, 2017.

[12] R. Beeson, V. Shah, J. Aurich, and D. Ellison, “Automated Solution of Low Energy Trajectories,” *AAS/AIAA Astrodynamics Specialist Conference*, Columbia River Gorge, Stevenson, Washington, August 21 - 24, 2017.

[13] S. L. McCarty, M. L. McGuire, “Parallel Monotonic Basin Hopping for Low Thrust Trajectory Optimization,” *AIAA SciTech Forum*, Kissimmee, Florida, January 8-12, 2018.

[14] G. A. Tsirogiannis, “A Graph Based Method for Mission Design,”*Celestial Mechanics and Dynamical Astronomy*, Vol. 114, No. 4, 2012, pp. 353-363.

[15] E. Trumbauer and B. Villac, “Heuristic Search-Based Framework for Onboard Trajectory Redesign,” *Journal of Guidance, Control and Dynamics*, Vo. 37, No. 1, 2014, pp. 164-175.

[16] N. Parrish, “A* Pathfinding for Continuous-Thrust Trajectory Optimization,” *AAS 37th Annual Guidance & Control Conference*, Breckenridge, January 2014.

[17] A. Das-Stuart, K. C. Howell, D. Folta, “Rapid Trajectory Design in Complex Environments Enabled by Reinforcement Learning and Exact Methods, ”*Acta Astronautica* (Under Revision), 2018

[18] A. Das-Stuart, K. C. Howell, D. Folta, “A Rapid Trajectory Design Strategy for Complex Environments Leveraging Attainable Regions and Low-Thrust Capabilities, ”*68th International Astronautical Congress*, Adelaide, Australia, Sep 25 - 29, 2017.

[19] V. Szebehely, “Theory of Orbits,” *Academic Press*, New Haven, Connecticut, 1st edition, 1967.

[20] C. Saltzer, R. Craig, and C. Fetheroff, “Comparison of Chemical and Electric Propulsion Systems for Interplanetary Travel,” *Proceedings of the IRE*, Vol. 48, No. 4, pp.465-476, April 1960.

[21] F. Topputo and C. Zhang,“Survey of Direct Transcription for Low-Thrust Space Trajectory Optimization with Applications,” *Abstract and Applied Analysis*, Vol. 2014, No. 2, pp. 1-15, 2014.

[22] J. Stuart, K. C. Howell, and R. Wilson, “Automated Design of Propellant-Optimal, End-to-End, Low-Thrust Trajectories for Trojan Asteroid Tours,” *Journal of Spacecraft and Rockets*, Vol. 51, No. 5, September 2014, pp. 1631-1647.

[23] F. Rosenblatt, “The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain,” *Psychological Review*, Vol. 65 (6), pp. 386-408, November 1958.

[24] M. Minsky and S. Papert, “Perceptrons: An Introduction to Computational Geometry,” *MIT Press*, Cambridge, Massachusetts, USA, 1969.

[25] J. Bapu, “The Artificial Neural Networks handbook: Part 1,” *https://www.datasciencecentral.com/profiles/blogs/the-artificial-neural-networks-handbook-part-1*, Last Accessed Date: [28 August 2018].

[26] I. Goodfellow, Y. Bengio and A. Courville, “Deep Learning,” *MIT Press*, Cambridge, Massachusetts, USA, 2016.

[27] N. Buduma, with contributions from N. Locascio, “Fundamentals of Deep Learning,” *O’Reilly Media*, Sebastopol, California, USA, 2017.

[28] J. Platt, “Probabilities for SV Machines. In: Advances in Large Margin Classifiers,” Editors: A. J. Smola, P. L. Bartlett, B. Schölkopf, D. Schuurmans, *MIT Press*, Cambridge, Massachusetts, USA, 2000.

[29] S. Haykin, “Neural Networks and Learning Machines,” *Prentice Hall*, New Jersey, USA, 2008.

[30] M. Akay, “Handbook of Neural Engineering,” *IEEE Press*, Piscataway, New Jersey, USA, 2007.

[31] P. Dahal, “Classification and Loss Evaluation - Softmax and Cross Entropy Loss,” *https://deepnotes.io/softmax-crossentropy*, Last Accessed Date: [23 August 2018].

[32] G. Cybenko, “Approximation by Superpositions of a Sigmoidal Function,” *Mathematics of Control, Signals, and Systems*, Vol. 2, pp. 303-314, 1989.

[33] K. Funahashi, “On the Approximate Realization of Continuous Mappings by Neural Networks,” *Neural Networks*, Vol. 2, pp. 183-192, 1989.

[34] G. Stuart, E. Bienenstock and R. Doursat, “Neural Networks and the Bias/Variance Dilemma,” *Neural Computation*, Vol. 4(1), pp. 1-58,1992.

[35] C. Cortes and V. Vapnik, “Support-Vector Networks,” *Machine Learning*, Vol. 20(3), pp. 273-297, 1995.

[36] N. Cristianini and J. Shawe-Taylor, “Support Vector Machines and Other Kernel-Based Learning Methods,” *Cambridge University Press*, Cambridge, UK, 2000.

[37] J. Shawe-Taylor and N. Cristianini, “Kernel Methods for Pattern Analysis,” *Cambridge University Press*, Cambridge, UK, 2004.

[38] J. Zhu, S. Rosset, T. Hastie, R. Tibshirani, “1-Norm Support Vector Machines,” *NIPS Proceedings of the 16th International Conference on Neural Information Processing Systems*, Whistler, British Columbia, Canada, December 09-11, 2003.

[39] J. Platt, “Fast Training of Support Vector Machines Using Sequential Minimal Optimization. In: Advances in Kernel Methods - Support Vector Learning,” Editors: B. Schokopf, C. Burges, A. Smola, *MIT Press*, Cambridge, Massachusetts, USA, 1998.

[40] J. Snoek, H. Larochelle, R. P. Adams, “Practical Bayesian optimization of machine learning algorithms,” *Advances in Neural Information Processing Systems*, 25: 2960-2968, 2012.

[41] D. L. Poole and A. K. Mackworth, “Artificial Intelligence, Foundations of Computational Agents,” *Cambridge University Press*, New York, 2010.

[42] R. A. C Bianchi, C. h. C. Ribeiro, and A. H. R. Costa, “On the Relation between Ant Colony Optimization and Heuristically Accelerated Reinforcement Learning,” *1st International Workshop on Hybrid Control of Autonomous Systems - Integrating Learning, Deliberation and Reactive Control*, Pasadena, California, July 2009.

[43] B. Fernandez-Gauna, I. Etxeberria-Agiriano, and M. Graña, “Learning Multirobot Hose Transportation and Deployment by Distributed Round-Robin Q-Learning,” *PLoS One*, Vol. 10, No. 7, 2015.

9. Appendix

9.0.1 Derivations - Artificial Neural Networks (ANNs)

This section details the ANN design choices incorporated in this investigation, and the associated derivations.

A *Activation Functions Incorporated in this Investigation*

Linear Function (Regression model output layer)
The linear activation function is often incorporated as the output layer in regression networks, where the outputs can be unbounded:

$$f(\mathbb{z}) = A\mathbb{z} \qquad [21]$$

Hyperbolic Tangent Function (Hidden layers)
The ’tanh’ function scales the sigmoid function to be centered around a zero-mean. Its mathematical formulation is:

$$f(\mathbb{z}) = \frac{1-e^{-2\mathbb{z}}}{1+e^{-2\mathbb{z}}}. \qquad [22]$$

This activation function diminishes the output for insignificant logit values.

Softmax Function (Classfication model output layer)
A softmax function, usually employed in the output layer, parses the knowledge gained through the network into a probability distribution over $\mathbb{C}$ outcomes (classes). The sum of the outputs of the neurons in the layer add to 1; such a rule is useful for classification problems, as higher probabilities can be associated with strong predictions about an input belonging to a certain class.[28] This approach must be exercised with caution, as the presence of a datapoint that does not belong to any of the expected classes can lead to false conclusions. This function is modeled as:

$$f(\mathbb{z}_\iota) = \frac{e^{z_i}}{\sum_{\mathbb{c}=1}^{\mathbb{C}} e^{z_\mathbb{c}}} \qquad [23]$$

where, $\iota = 1, 2, 3, ...\mathbb{C}$

B *Error Minimization*

Softmax Output Layer and Cross-Entropy Loss
The predicted probability distribution for a neuron at the softmax output layer from Eqn. [23] is set equal to the predicted output class values, $\mathbb{Y}_{p_\iota}$:

$$\mathbb{Y}_{p_\iota}^{(\mathbb{o})} = f(\mathbb{z}_\iota^{(\mathbb{o})}) \qquad [24]$$

Cross-entropy loss (error) is often implemented for classification problems where the information from a softmax layer is representative of a probability distribution,[30] and is defined at the output layer $(\mathbb{o})$, as:

$$E = \sum_{\iota=1}^{\mathbb{C}} \mathbb{Y}_{t_\iota}^{(\mathbb{o})} log(\mathbb{Y}_{p_\iota}^{(\mathbb{o})}), \qquad [25]$$

where $\mathbb{Y}_{t_\iota}$ are the true probabilities of belonging to a particular class. Note that logistic regression for binary classification is a particular instance of the multi-class classification problem. The process of weight updates in each layer begins with establishing a relationship between the error and the logit at the output layer:[31]

$$\begin{aligned}
\frac{\partial E}{\partial \mathbb{z}_\iota^{(\mathbb{o})}} &= \sum_{\mathbb{c}=1}^{\mathbb{C}} \frac{\partial E}{\partial \mathbb{Y}_{p_\mathbb{c}}^{(\mathbb{o})}} \frac{\partial \mathbb{Y}_{p_\mathbb{c}}^{(\mathbb{o})}}{\partial \mathbb{z}_\iota^{(\mathbb{o})}} \\
\frac{\partial E}{\partial \mathbb{Y}_{p_\iota}^{(\mathbb{o})}} &= -\frac{\mathbb{Y}_{t_\iota}^{(\mathbb{o})}}{\mathbb{Y}_{p_\iota}^{(\mathbb{o})}} \qquad [26] \\
\frac{\partial \mathbb{Y}_{p_\iota}^{(\mathbb{o})}}{\partial \mathbb{z_c}^{(\mathbb{o})}} &= \begin{cases} \mathbb{Y}_{p_\iota}^{(\mathbb{o})}(1-\mathbb{Y}_{p_\iota}^{(\mathbb{o})}) & \iota = \mathbb{c} \\ -\mathbb{Y}_{p_\iota}^{(\mathbb{o})}\mathbb{Y}_{p_c} & \iota \neq \mathbb{c} \end{cases}
\end{aligned}$$

So, Eqn. [26] is re-written as:

$$\begin{aligned}
\frac{\partial E}{\partial \mathbb{z}_\iota^{(\mathbb{o})}} &= \frac{\partial E}{\partial \mathbb{Y}_{p_\iota}^{(\mathbb{o})}} \frac{\partial \mathbb{Y}_{p_\iota}^{(\mathbb{o})}}{\partial \mathbb{z}_\iota^{(\mathbb{o})}} - \sum_{\mathbb{c}\neq\iota} \frac{\partial E}{\partial \mathbb{Y}_{p_c}} \frac{\partial \mathbb{Y}_{p_c}}{\partial \mathbb{z}_\iota^{(\mathbb{o})}} \\
&= -\mathbb{Y}_{t_\iota}^{(\mathbb{o})}(1-\mathbb{Y}_{p_\iota}^{(\mathbb{o})}) + \sum_{\mathbb{c}\neq\iota} \mathbb{Y}_{t_c}^{(\mathbb{o})}\mathbb{Y}_{p_\iota}^{(\mathbb{o})} \qquad [27] \\
&= \mathbb{Y}_{p_\iota}^{(\mathbb{o})}[\mathbb{Y}_{t_\iota}^{(\mathbb{o})} + \sum_{\mathbb{c}\neq\iota} \mathbb{Y}_{t_c}^{(\mathbb{o})}] - \mathbb{Y}_{t_\iota}^{(\mathbb{o})}
\end{aligned}$$

As the term $[\mathbb{Y}_{t_\iota}^{(\mathbb{o})} + \sum_{\mathbb{c}\neq\iota} \mathbb{Y}_{t_c}^{(\mathbb{o})}]$ in Eqn. 27 equals 1,

$$\frac{\partial E}{\partial \mathbb{z}_\iota^{(\mathbb{o})}} = \mathbb{Y}_{p_\iota}^{(\mathbb{o})} - \mathbb{Y}_{t_\iota}^{(\mathbb{o})} \qquad [28]$$

Linear Output Layer and Mean Square Error
The output of a linear layer is characterized as:

$$\mathbb{Y}_{p_\varphi}^{(\mathbb{o})} = f(\mathbb{z}_\varphi^{(\mathbb{o})}) = \mathbb{z}_\varphi^{(\mathbb{o})} \qquad [29]$$

The Mean Square Error (MSE) is appropriate for the continuous and unbounded variable outputs associated with a linear output layer for regression.

$$E = \sum_{\varphi=1}^{\Upsilon} \frac{(\mathbb{Y}_{t_\varphi}^{(\mathbb{o})} - \mathbb{Y}_{p_\varphi}^{(\mathbb{o})})^2}{\Upsilon} \qquad [30]$$

The rate of change of the error with respect to the within-layer logit is then:

$$\begin{aligned}
\frac{\partial E}{\partial \mathbb{z}_\varphi^{(\mathbb{o})}} &= \frac{\partial E}{\partial \mathbb{Y}_{p_\varphi}^{(\mathbb{o})}} \frac{\partial \mathbb{Y}_{p_\varphi}^{(\mathbb{o})}}{\partial \mathbb{z}_\varphi^{(\mathbb{o})}} \\
\frac{\partial E}{\partial \mathbb{Y}_{p_\varphi}^{(\mathbb{o})}} &= -\frac{2}{\Upsilon}(\mathbb{Y}_{t_\varphi}^{(\mathbb{o})} - \mathbb{Y}_{p_\varphi}^{(\mathbb{o})}) \qquad [31] \\
\frac{\partial \mathbb{Y}_{p_\varphi}^{(\mathbb{o})}}{\partial \mathbb{z}_\varphi^{(\mathbb{o})}} &= 1
\end{aligned}$$

So, Eqn. [31] is re-written as:

$$\frac{\partial E}{\partial \mathbb{z}_\varphi^{(\mathbb{o})}} = \phi(\mathbb{Y}_{p_\varphi}^{(\mathbb{o})} - \mathbb{Y}_{t_\varphi}^{(\mathbb{o})}) \qquad [32]$$

where, $\phi = \frac{2}{\Upsilon}$.

Backpropagation of the Error to Hidden Layers
Equations [28] and [32] describe the error with respect to the logit at the last or ouput layer. This error must be propagated back through to the input layer to trace the influence of the weights in the hidden layers that lead to the final error, E.[29] For a hidden layer l within a network with j neurons, a preceding layer $(l-1)$ with i neurons and a superseding layer $(l+1)$ with k neurons, the relationship between the final error and weights are captured via partial derivatives,

$$\frac{\partial E}{\partial \mathbb{w}_{i,j}^{(l)}} = \frac{\partial E}{\partial \mathbb{z}_k^{(l+1)}} \frac{\partial \mathbb{z}_k^{(l+1)}}{\partial \mathbb{y}_p{}^{(l)}} \frac{\partial \mathbb{y}_p{}^{(l)}}{\partial \mathbb{z}_j^{(l)}} \frac{\partial \mathbb{z}_j^{(l)}}{\partial \mathbb{w}_{i,j}^{(l)}} \qquad [33]$$

where, each of the above quantities have been previously defined. Note that the predicted output from a layer $\mathbb{y}_p^{(l)}$, is equivalent to the input entering the next layer $\mathbb{x}^{(l+1)}$.

C Generalizability of ANNs

An appropriate network size that balances the speed of training, accuracy and generalizability of the model is most successful. So, adopting techniques such as *regularization*[26] and early stopping techniques are effective measures to offer insights into over-fitting.[29] The latter option (employed in this investigation) validates the model periodically with 'unseen' data, and terminates the training when the validation accuracy begins to stagnate even as the training accuracy continues to decrease.

9.0.2 Derivations - Support Vector Machines (SVMs)

This section details the SVM design choices incorporated in this investigation, and the associated derivations.

A *Computing Weights and Biases for Separating Hyperplanes.*

Solving for $\mathbb{w}_i$ and $\mathbb{B}$ is initiated by calculating the primal form of the Lagrangian, where the constraints in Eqn. [11b] and [11c] are appended with non-negative Lagrange multipliers $\nu_i \geq 0$ and $\mathbb{v}_i \geq 0$ and adjoined with the objective function:[36]

$$\underset{W,B,\zeta}{min}\ \mathscr{L}_p(\nu, \mathbb{v}, \mathbb{x}, \mathbb{w}, \zeta) = \frac{1}{2}\mathbb{W}^T\mathbb{W} + C\sum_i^N \zeta_i - \sum_i^N \nu_i[\mathbb{y}_i f(\mathbb{x}_i) - 1 + \zeta_i] - \sum_i^N \mathbb{v}_i\zeta_i \quad [34]$$

Determining stationary points of $\mathscr{L}_p$ with regards to the primal variables $\mathbb{W}$, $\mathbb{b}$ and ζ leads to the following conditions:

$$\mathbb{W} = \sum_i \nu_i \mathbb{y}_i \mathbb{x}_i \quad [35a]$$

$$0 = \sum_i \nu_i \mathbb{y}_i \quad [35b]$$

$$C - \nu_i - \mathbb{v}_i = 0 \quad [35c]$$

These relationships are substituted into Eqn. [34] to construct the dual formulation (Eqn. [36]), useful in scenarios requiring analysis in a higher dimensional space.

$$\underset{\nu}{max}\ \mathscr{L}_d(\nu, \mathbb{v}, \mathbb{x}, \mathbb{y}) = \sum_i^{\mathbb{N}} \nu_i - \frac{1}{2}\sum_i^{\mathbb{N}}\sum_j^{\mathbb{N}} \nu_i\nu_j\mathbb{y}_i\mathbb{y}_j\langle \mathbb{x}_i, \mathbb{x}_j\rangle \quad [36]$$

The combination of the conditions in Eqns. [11c] and [35c] constrain the bounds on the Lagrange multipliers (ν_i), that is,

$$\nu_i \leq C. \quad [37]$$

So, C is also recognized as a *box constraint* since it bounds the values of ν_i. Note that the Karush-Kuhn-Tucker (KKT) conditions supply the necessary conditions for optimality and dictate the values for ν_i:[37]

$$\begin{cases} \mathbb{y}_i f(\mathbb{x}_i) \geq 1 & \nu_i = 0 \\ \mathbb{y}_i f(\mathbb{x}_i) \leq 1 & \nu_i = C \\ \mathbb{y}_i f(\mathbb{x}_i) = 1 & 0 < \nu_i < C \end{cases} \quad [38]$$

Since only the non-zero ν_i contribute towards the maximization process of the Lagrangian, the corresponding $\mathbb{x}_i$ assume the role of *support vectors* for the problem. Similarly, the non-zero ζ_i also assume the role of support vectors for scenarios associated with non-separable data. The value of $\mathbb{B}$ is extracted from the third KKT condition in Eqn. [38], i.e., $\mathbb{y}_i f(\mathbb{x}_i) = 1$ as $\zeta_i = 0$ when $0 < \nu_i < C$.

B *Transformation of Feature Space to a Higher Dimension*

This process involves mapping the inner products of input pairs (introduced in Eqn. [36]), to a higher dimension by a *feature* or *kernel* function, $\mathfrak{K}(\mathbb{x}_i, \mathbb{x}_j)$. To be identified as a kernel function, the *Mercer kernel* property must be satisfied, that is, the kernel must be positive semi-definite.[29] Thus, the dual form of the Lagrangian $\mathscr{L}_d$, is convex, eliminating potential encounters with problematic local-minima encountered with ANNs during optimization. The infinite dimensional mapping afforded by gaussian Mercer kernels is adopted in this investigation:

$$\mathfrak{K}(\mathbb{x}_i, \mathbb{x}_{\mathbb{j}}) = e^{-\mathbb{\sigma}||\mathbb{x}_i - \mathbb{x}_j||^2}. \quad [39]$$

The selection of the kernel function is problem specific, and is informed by a priori knowledge of the underlying relationships in the data. The dual form of the Lagrangian is modified to include the kernel functions as:

$$\underset{\nu}{max}\ \mathscr{L}_d(\nu, \mathbb{X}, \mathbb{W}) = \sum_i^{\mathbb{N}} \nu_i - \frac{1}{2}\sum_i^{\mathbb{N}}\sum_j^{\mathbb{N}} \nu_i\nu_j\mathbb{y}_i\mathbb{y}_j\mathfrak{K}(\mathbb{x}_i, \mathbb{x}_j) \quad [40]$$

Note that $\mathfrak{K}(\mathbb{x}_i, \mathbb{x}_j)$ is equivalent to the inner product of the higher dimensional transformation of the original data by a nonlinear function, $\langle \Phi(\mathbb{x}_i), \Phi(\mathbb{x}_j)\rangle$. Since the dual formulation is a direct function of the input training data points ($\mathbb{x}_i \in \mathscr{R}^{\mathbb{m}}$), the dual formulation in Eqn. [40] guarantees that the computational burden is not drastically increased even when the kernel function transforms the problem into a higher dimensional feature space. The computationally efficient Sequential Minimal Optimization (SMO) algorithm is adopted to solve the convex optimization problem - optimizing one pair of Lagrange multipliers at a time, selected via heuristics.[39] The appropriately tuned weights and bias via the optimized Lagrange multipliers deliver either a positive or negative output score ($f(\mathbb{x}_i) = \mathbb{S}_i$), that aids in binary classification of the data. A desirable measure of confidence in the classification is obtained by transforming this score to an output's posterior probability, based on the prior belief as determined by the selected kernel function.[28] A parametric form of the sigmoid function is employed to extract the appropriate mapping of the scores to approximate the posterior probability:[28]

$$\mathfrak{P}(\mathbb{S}_i) = \frac{1}{1 + exp(\mathscr{A}\mathbb{S}_i + \mathscr{B})} \quad [41]$$

The term $\mathfrak{P}(\mathbb{S}_i)$ in Eqn. [41] is equivalent to the predicted probabilities of the data-points belonging to a class of interest as discussed in Section 4.1.1. So, $\mathscr{A}$ and $\mathscr{B}$ are computed using any optimization algorithm that minimizes the log-losses measured by cross-entropy error.

C Generalizability of SVMs

The geometric-complexity of the margin delivered by the box constraint C, and the kernel hyper-parameter(s), determine the total number of support vectors and, thus, also the memory requirements associated with the trained model. These hyper-parameters are determined either via a grid-search or optimization techniques that minimize the classification errors. This investigation incorporates the readily available *BayesOpt* library for computing the hyper-parameters.[40] The accuracy of the trained model may be tested using either hold-out or cross-validation sets.[28] The hold-out technique sets aside a small fraction of the data for testing purposes. In the cross-validation approach, the data is equally partitioned into $\mathbb{k}$ *folds* such that each *fold* is exploited as a validation set for $\mathbb{k}$ rounds of training.

25