# Surrogate Neural Networks for Efficient Simulation-based Trajectory Planning Optimization

by

**Evelyn Ruff**

B.S. Mechanical Engineering, Northeastern University (2018)
M.S. Mechatronics, Northeastern University (2019)

Submitted to the System Design and Management Program
in partial fulfillment of the requirements for the degree of

**Master of Science in Engineering and Management**

at the

Massachusetts Institute of Technology

June 2023



Authored by: Evelyn Ruff
System Design & Management Program
May 12, 2023
Certified by: Jonathan P. How
Maclaurin Professor of Aeronautics and Astronautics, MIT
Thesis Supervisor
Certified by: Piero Miotto
Laboratory Fellow, Draper
Thesis Supervisor
Accepted by: Joan Rubin
Executive Director, System Design & Management Program

# Surrogate Neural Networks for Efficient Simulation-based Trajectory Planning Optimization

by

Evelyn Ruff

Submitted to the System Design and Management Program
on May 12, 2023, in partial fulfillment of the
requirements for the degree of
Master of Science in Engineering and Management

## Abstract

This paper presents a novel methodology that uses surrogate models in the form of neural networks to reduce the computation time of simulation-based optimization of a reference trajectory. Simulation-based optimization is necessary when there is no analytical form of the system accessible, only input-output data that can be used to create a surrogate model of the simulation. Like many high-fidelity simulations, this trajectory planning simulation is very nonlinear and computationally expensive, making it challenging to optimize iteratively. Through gradient descent optimization, our approach finds the optimal reference trajectory for landing a hypersonic vehicle. In contrast to the large datasets used to create the surrogate models in the prior literature, our methodology is specifically designed to minimize the number of simulation executions required by the gradient descent optimizer. We demonstrated this methodology be more efficient than the standard practice of hand-tuning the inputs through trial-and-error or randomly sampling the input parameter space. Due to the intelligently selected input values to the simulation, our approach yields better simulation outcomes that are achieved more rapidly and to a higher degree of accuracy. Optimizing the hypersonic vehicle's reference trajectory is very challenging due to the simulation's extreme nonlinearity, but even so, this novel approach found a 74% better performing reference trajectory compared to nominal, and the numerical results clearly show a substantial reduction in computation time for designing future trajectories.

Thesis Supervisor: Jonathan P. How
Maclaurin Professor, Massachusetts Institute of Technology

Thesis Supervisor: Piero Miotto
Laboratory Staff, Draper Laboratory

# Acknowledgments

I would like to thank my advisors, Professor Jonathan P. How of MIT and Piero Miotto of Draper, for supporting me as I learned, tried, failed, and worked towards developing the novel algorithm presented in this thesis. Jon, your ability to immediately identify the areas I needed to keep pushing in helped me learn so much. Thank you for constantly asking the right questions and guiding me through this process. Piero, your willingness to support me of the past two years has been incredible. Thank you for steering me through the would of guidance, navigation, and control with great compassion and technical aptitude.

Furthermore, I would to thank Rebecca Russell for helping me grasp many of the finer details of machine learning and always be willing to draw on the whiteboard. I am so grateful for our hours together debugging code and debating new ways to tackle complicated problems. And thank you to Matthew Stoeckle for working through the trajectory algorithm and simulation integration with me. No one else could have showed me exactly how to combine those scripts which were invaluable for my application.

I would also like to thank my family and friends for supporting me throughout the last two years as I spent far too many hours at MIT. To my parents and siblings, thank you for pushing me to work hard, but not too hard, and continue a lifetime of learning. To my friends, thank you for bringing me endless joy and being there for me through it all.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

High-fidelity simulations are used to analyze the dynamics of complex systems in many engineering and scientific disciplines. For most of these applications, there is some desired system outcome, such as the curvature of an airfoil or the path planned for a robot but, lacking a closed-form model, the simulation must be used in the design process. Here arises the need for simulation-based optimization, as opposed to traditional optimization with analytical models, generally with the goal of optimizing the simulation inputs values to achieve the desired outputs.

There are many methods of simulation-based optimization [1, 2] applied across disciplines. A key issue in these approaches is that the execution of such simulations often requires a large amount of computational power and/or processing time. When the simulation itself is too computationally expensive, surrogate models are needed to minimize the number of queries to the simulation. This minimizes the total computation time needed to reach an optimal set of inputs. The surrogate model is designed to imitate the input-output behavior of the simulation. Input-output data is required from the simulation to create this surrogate model, so efficiency in creating that model is a key concern [3]. The methodology presented herein uses neural networks as surrogate models and provides a novel sampling strategy to reduce the number of simulation runs [4]. Minimizing number of simulation runs is a crucial con-

cern when each run of the simulation takes several minutes, and optimization might require hundreds, if not thousands [5], of simulation runs.

The use case simulation for this methodology is an Approach and Landing (A/L) simulation for hypersonic vehicles (HV) combined with the reference trajectory calculation [6]. The HV simulation includes high-fidelity models for flex, slosh, engine thrust, aerodynamics, and full flight software in addition to the trajectory planning algorithm which propagates the vehicle down many possible trajectories. This combined trajectory planning algorithm and A/L simulation takes 3-4 minutes to execute. The relationship between inputs and outputs is highly nonlinear due to the series of high-fidelity models listed. The system inputs are the 13 trajectory design parameters and the three outputs are the most important performance metrics of the vehicle during approach and landing.

Hence, finding the set of trajectory design parameters that provides the best landing trajectory for the HV is typically done by manually tuning the various input parameters, running the full simulation, and evaluating the performance outputs. Then, iteratively changing the input parameters values intuitively through trial-and-error. Requiring a human-in-loop means many inefficient best guesses at the trajectory design parameters and no deterministic way to evaluate the results, meaning the final trajectory isn't optimal but merely "good enough". Instead, a Monte Carlo approach could be implemented by randomly selecting hundred or thousands of input values until a sufficiently good reference trajectory is found. However, this is incredibly computationally expensive due the high number of simulation queries that must be run until a solution is randomly found. Finding a sufficiently good A/L reference trajectory using either of these approaches is a very time-consuming iterative process that requires extensive domain knowledge to either understand the complicated input-output mapping or to accurately evaluate the performance outcomes.

The algorithm outlined in this paper offers a novel simulation-based optimization approach that eliminates the need for hand-tuning input parameters and significantly reduces the overall computation time. The main contributions of this methodology are:

- Novel algorithm using neural network gradients to intelligently select simulation input values that improves desired simulation outcome by 74%.

- Computationally efficient methodology for optimizing general black-box simulations with desired outputs shown to be six times faster than Monte Carlo approach;

- New automated optimal trajectory planning tool for hypersonic vehicles that takes hours as opposed to days.

## 1.2 Related Work

This paper aims to improve upon two categories of research: simulation-based optimization via surrogate models and optimal trajectory planning.

### 1.2.1 Simulation-based Optimization via Surrogate Models

There are many simulation-based optimization algorithms, and their suitability relies heavily on the specific application. Whether the system is continuous or discrete, cheap or expensive to evaluate, and deterministic or stochastic all must be considered [2]. This makes various simulation-based optimization methods very challenging to compare. In literature, these methods are often divided into four categories: statistical selection, surrogate models (metamodels), stochastic gradient estimation, and global search [1]. Surrogate models reduce the computational burden of optimizing the simulation outputs by creating a simpler, less-expensive model of the simulation to use instead of the simulation itself.

For this purpose of this paper, only simulation-based optimization via surrogate models will be analyzed because the chosen application is continuous and very expensive to run, making surrogate modeling the ideal type of algorithm to use [1]. There have been many kinds of surrogate models applied across various fields. Most notable include algebraic models [3], kriging [7], polynomial response surfaces [8], and

radial basis functions [9]. However, with the proliferation of machine learning in recent decades, surrogate models have more recently been created using support-vector machines [10] and neural networks [11].

Once the surrogate model has been fitted to the data, it is integrated with an optimizer to find the optimal set of inputs that achieve the desired output to the simulation. Neural networks are exceptional for surrogate modeling as they are very good at fitting data. The applications where neural networks have been applied as surrogate models are very diverse, such as fluids systems [12–14] and airfoil design [15, 16]. Instead of taking advantage of the gradient information available from the neural network, previous methods generally have relied on gradient-free optimization, such as genetic algorithms [13–15], particle swarm optimization [16], and the grey wolf optimizer [12].

Ref. [17] uses the gradients of a neural network in combination with the IPOPT software library to optimize a topology. However, the simulation was simple enough that only a single-layer feedforward neural network was necessary which meant the derivatives had to be calculated analytically and fed into the optimizer. Ref. [5] uses a similar method of gradient-based optimization using multi-layer neural networks combined with the SNOPT software library to optimize airfoils. Both methods rely on creating a highly accurate surrogate model that only needs to be trained once to be used by the optimizer. Ref. [17] does this through a complicated training procedure using the Sobolov error and [5] does this through collecting tens of thousands of data points.

In the method proposed in this paper, the algorithm collects more data from the actual simulation iteratively in areas of interest suggested by the surrogate model to reduce total number of queries. Reducing the total number of queries is done by requiring less initial training data for the surrogate neural network and then increasing it's fidelity in only promising regions of the input parameter space in order to save on computation cost.

### 1.2.2 Optimal Trajectory Planning

Reference trajectory optimization is a very challenging process that requires formulating a nonlinear optimal control problem with multiple constraints and solving, directly or indirectly [18]. In general, reference trajectories are calculated offline and stored in the HV's computer although some algorithms work to reduce computation time in order to be calculated during flight [19]. For comparison, only offline algorithms will be considered in this review.

Ref. [20] proposes a methodology for HV trajectory optimization by combining particle swarm optimization and non-intrusive polynomial chaos. This method works to minimize the flight time of the HV over the course of the trajectory while being robust to physical uncertainties. While a time-based objective function works for the re-entry phase of hypersonic flight, it does not allow for optimizing specific properties at phase completion. Furthermore, this is a direct numerical method which requires access to the dynamical functions, which is not the case in the application of the algorithm proposed in this paper. The same is true for the generalized polynomial chaos algorithm proposed by [21] to optimize landing trajectories of airplanes and the mapped Chebyshev pseudospectral method presented by [22].

Few optimal trajectory planning algorithms take advantage of recent advances in machine learning. For example, [23] generates 6-degree-of-freedom optimal trajectories for hypersonic vehicle reentry through traditional dynamical equations and only leverages deep neural networks to create feedback control during flight. Similarly, [8] uses a reinforcement learning framework to generate commands for control of an aircraft but not for the initial generation of an optimal trajectory.

These optimal trajectory planning methods require a closed-form solution accurately describing vehicle dynamics and the generation of numerical inequality and equality constraints. Currently, there are no methods of optimal trajectory planning that use neural networks as surrogate models for simulation-based optimization.

## 1.3 Contributions

As stated above, this methodology expands upon several areas of existing work. While there are other methods that leverage the gradients of surrogate neural networks, none of them use iteration to intelligently improve the surrogate model. Starting with a smaller training data set and using the surrogate model to query the model in only promising regions of the input parameter space means computation time is not wasted querying the model in areas that are known to produce sub-optimal results. Through the iterative querying process, this methodology is designed to find better simulation results faster.

When tested on two different hypersonic vehicle trajectory planning simulations, this methodology improved the desired simulation outcomes by 74% and 100%. The efficiency of this algorithm is shown through comparison to the Monte Carlo approach or random sampling of the input parameter space. When compared to randomly selected queries to the simulation, the intelligently selected queries improve simulation outcomes six times faster.

Furthermore this algorithm provides Draper with an automated tool to planning optimal reference trajectories for hypersonic vehicles. Previously, a guidance, navigation, and control (GNC) engineer would have to enter the initial best guess at the 13 trajectory design parameters, wait for the trajectory planning algorithm to execute, transfer the calculated trajectory into the A/L digital twin simulation, and wait for that simulation to execute. A single iteration of this process takes about 5 minutes. The GNC engineer must now evaluate the simulation outputs and attempt to discern which of the 13 trajectory design parameters to tune to improve the next iteration's outcome. Every time an parameter in the simulation changes, a new reference trajectory must be found that produces a good landing of the hypersonic vehicle. Depending on the number of parameters that are changed and how much that variation is, finding the new reference trajectory can take weeks. With the implementation of this new automated tool, calculating optimal trajectories for hypersonic vehicles takes hours.

# Chapter 2

# Preliminaries

## 2.1 A/L Trajectory Planning Simulation

Throughout this thesis, the application of the algorithm is referred to as a 'trajectory planning simulation'. In more detail, the trajectory planning simulation is actually a a trajectory planning algorithm integrated with an approach and landing trajectory simulation. The resultant reference trajectory calculated by the planning algorithm is tested by the high-fidelity simulation to measure the success of landing the hypersonic vehicle. These two scripts were integrated together in MATLAB to eliminate the need to manually transfer the trajectory from one script to the next and create a 'black-box' simulation. Combined they take 3-4 minutes to run depending on the initial design parameters input to the trajectory planning algorithm. A poor initial guess for the initial design parameters means the planning algorithm will take longer to converge.

### 2.1.1 Reference Trajectory Planning Algorithm

Designing the approach and landing trajectory for a hypersonic vehicle is very challenging due to the vehicle's lack of thrusters and large complex physical uncertainties. The hypersonic vehicles must use a speed brake system to ensure they maintain enough energy to reach the runway without overshooting it. The process is so compu-

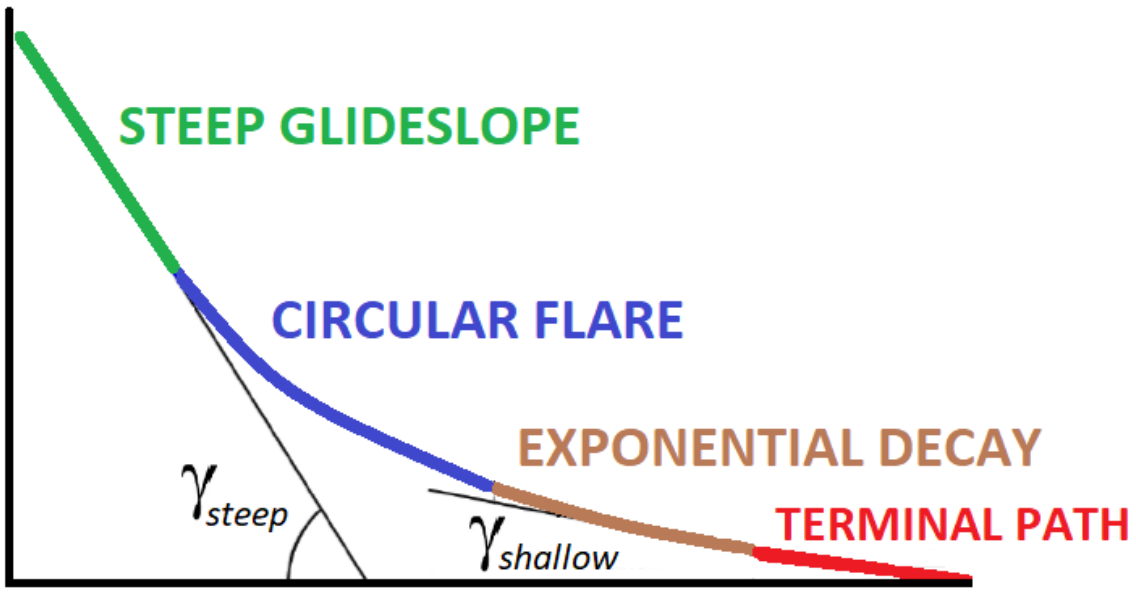


Figure 2-1: Defined geometric segments of reference trajectory for approach and landing of hypersonic vehicle calculated by the Autolanding I-Load Program.

tationally expensive that the nominal trajectory is calculated offline and then stored in an onboard computer for the guidance and control system to follow using speed brakes and unboosted maneuvers.

Our simulation uses the Autolanding I-load Program (ALIP) [6] to calculate the A/L reference trajectory for the HV. The A/L of the vehicle typically starts at an altitude of 10,000 ft and ends at touchdown on the runway [6]. This program has legacy from Orbital Science's X-34 and NASA's Shuttle Program [24]. ALIP relies on initializing the parameters defining the geometric segments of the trajectory with an accurate prediction and then optimizes those parameters to achieve the desired dynamic pressure on the vehicle at touchdown. Various physical constraints are applied to reduce the trajectory design problem to a two-point boundary value problem [6].

There are 13 input parameters to Autolanding I-Load Program that define the initial trajectory that all must be optimized to result in a successful landing of the HV. Slight changes in the input values will affect not only the resultant reference trajectory but also computation time to varying degrees. These parameters are defined in Table 2.1. Fig. 2-1 shows an example reference trajectory broken into its geometric segments.

Table 2.1: Input Parameters to Trajectory Planning Algorithm (ALIP)

| Parameter Name | Init. Value | Description |
|---|---|---|
| qbar_steep | 25 psf | Dynamic pressure at start of steep glide slope |
| land_vel | 50 knots | Desired velocity at touchdown |
| XAIMPT | 250 ft | Shallow glide slope x-intercept with runway |
| FLARE_MAX | 1.5 g | Value of max g's during radius circular flare |
| GAMMA_REF_2 | 1 deg | Flight path angle on shallow glide slope |
| HDECAY | 15 ft | Start of exponential decay altitude |
| GAM_FF | 1.0 deg | Flight path angle at touchdown |
| H_TFP_I | 1.5 ft | Altitude to start terminal flight path maneuver |
| H_TFP_F | 6 ft | Altitude to end terminal flight path maneuver |
| db1, db2, db3, db4 | 0.5 | Speed brake position |

There are four geometric segments: steep glideslope, circular flare, exponential flare decay, and shallow glideslope (terminal path). Altering any of the 13 trajectory design parameters into ALIP will result in a different geometric segments defining the reference trajectory. Once ideal dynamic pressure is achieved, the geometry defining the reference trajectory can be tested through a high-fidelity simulation for verification.

### 2.1.2 A/L Digital Twin Simulation

Once a reference trajectory has been defined using ALIP, that trajectory is tested through a digital twin simulation. The digital twin simulation is a high-fidelity 6-degree-of-freedom model includes a rotating, oblate earth, sensor models, actuator models, and actual Draper guidance, navigation, and control flight software in loop. The simulation also includes models for flex, slosh, engine thrust, atmosphere density, wind, aerodynamics, and 18x18 spherical harmonic gravity.

This simulation outputs certain metrics which quantify the success of the HV touchdown. Specifically, there is an ideal rate at which the vehicle loses altitude (sink rate), distance from start of the runway where landing happens (downrange position), and horizontal velocity of the HV at touchdown as shown in Table 2.2. The digital

Table 2.2: Target Outputs from A/L Simulation

| Parameter Name | Target Value | Description |
|---|---|---|
| Sink Rate | 2 ft/s | Rate at which vehicle loses altitude |
| Horizontal Velocity | 54 knots | Vehicle's horizontal velocity at touchdown |
| Downrange Position | 400 ft | Landing distance from runway threshold |

twin simulation is executed using physical parameters from a proprietary hypersonic vehicle.

All inputs parameters to ALIP and all output parameters from the landing simulation have been standardized and selectively scaled to protect any proprietary information.

## 2.2 Neural Networks as Surrogate Models

Neural networks (NNs) are very good at modeling complex relationships between inputs and outputs. In fact, multilayer feedforward networks can approximate any measurable function to any degree of accuracy, provided enough training data [25]. Neural networks are capable of "mapping" any deterministic function and are therefore universal approximators. Multilayer NNs can find intricate structures in high-dimensional data and learn hierarchical feature representations with multiple levels of abstraction [26].

Furthermore, neural networks can simultaneously estimate the derivatives of an approximated function, even if the function does not have classically differentiable functions [25]. This makes them a perfect choice for surrogate models. Most commonly, the derivatives of the neural network, referred to as gradients, are used to backpropagate error through the neural network to train them on an initial dataset. These gradients can also be used for sensitivity analysis [17] or for gradient-based optimization as in this proposed method.

## 2.3 Objective Function with Constraints

For all simulations, there must be $m$-dimensional inputs ($x \in \mathbb{R}^{\mathrm{m}}$) into the simulation $f(x)$ and $n$-dimensional outputs ($y \in \mathbb{R}^{\mathrm{n}}$). Furthermore, there are $m$-dimensional minimum and maximum bounds ($x_{\min}, x_{\max} \in \mathbb{R}^{\mathrm{m}}$) for each input as well as a $n$-dimensional desired set of outputs ($y_{\text{target}} \in \mathbb{R}^{\mathrm{n}}$).

Using an initial training dataset, a neural network is trained to be a surrogate model $\hat{f}(x)$ for the real simulation $f(x)$. The surrogate model $\hat{f}(x)$ is needed because it is impossible to obtain an analytical representation of the actual simulation $f(x)$. This surrogate model is able to predict $n$-dimensional outputs based on real $m$-dimensional inputs

$$\hat{y} = \hat{f}(x) \tag{2.1}$$

where $\hat{y}$ is the predicted outputs from the surrogate model. This surrogate model is then used to minimize the loss between the desired outputs and the predicted outputs from the surrogate model. This loss is evaluated by the objective function

$$g\left(\hat{y}, y_{\text{target}}\right). \tag{2.2}$$

The objective function could easily be altered to minimize or maximize certain outputs instead of targeting certain output values. Note that the parameters being optimized are not the predicted output from the surrogate model, but the inputs to the surrogate model, making the objective a function of $x$:

$$g\left(\hat{f}(x), y_{\text{target}}\right) \tag{2.3}$$

Further note that this is a constrained optimization problem because the inputs must be bounded within the initial training data range or else the surrogate model will not be able to accurately predict the input's respective output. The inputs are bounded in the physically feasible range by the inequality conditions:

$$c_1(x) : x \geq x_{\min} \tag{2.4}$$

$$c_2(x) : x \leq x_{\max} \tag{2.5}$$

These constraints could be implemented in several different ways such as Lagrange multipliers or linear programming. They could also be added to the objective function as a soft constraint. Similarly, equality conditions can be added to the objective function as well. Theoretically, infinite constraints could be added to the objective function and weighted appropriately by how important they are to the application. However, it is important to note the objective function should be continuous and differentiable for gradient-based optimization.

Because the objective function is a function of the inputs, its gradients can be used to optimize the inputs to minimize the objective loss. The gradient of the objective function with respect to the inputs, ignoring any additional constraints, is

$$\frac{\partial}{\partial x}\left[g(\hat{f}(x))\right] = g'(\hat{f}(x))\hat{f}'(x). \tag{2.6}$$

This derivation shows the need for the surrogate model and its gradients for optimizing the inputs to minimize the objective function. While other methods of gradient-free optimization are possible, gradient-based optimization tend to converge faster on smooth functions.

## 2.4 Black-box Optimization with Surrogate Models

Simulation-based optimization is part of a larger area of research called 'black-box optimization'. For many real-world problems, black-boxes are computationally expensive, multi-constrained, and high-dimensional [27]. This had led to an increased interest in surrogate modeling to assist in the optimization process. However, optimization via surrogate modeling has competing goals. The short-term goal of such algorithms is to exploit the current knowledge of the black-box system to find the best

possible solution. The long-term goal of such algorithms is the increase the surrogate models understanding of the system through exploration to find better solutions later on. Finding a balance between exploration and exploitation is a common issue in machine learning and especially reinforcement learning. There are many techniques to balance these goals that generally require much fine-tuning and supervision [28].

In adversarial machine learning, such techniques are crucial for 'black-box attacks'. Here, algorithms work to find vulnerabilities in neural networks used for applications such as image classification and object detection. Less queries to the black-box generally mean the attack is more efficient due to more queries leading to higher cost or adversarial detection. Ref. [29] uses natural gradient descent to train their neural networks and update the input parameters for the attack to reduce total number of queries. Similar to the application of this methodology, the goal of adversarial attacks is to create an input to the black-box system to achieve a desired output. And in doing so, reduce total number of queries to actual system.

# Chapter 3

# Approach

This chapter details the general methodology for using surrogate neural networks to optimize a black-box simulation, as shown in Algorithm 1. The following sections detail how to apply the methodology to a specific trajectory planning simulation.

**Algorithm 1**

1: Query the black-box simulation $f(x)$ to obtain an initial dataset $\mathcal{D} = \{x \in \mathbb{R}^{\mathrm{m}}\}$ from a quasi-random sample. [3.1]
2: Train a neural network surrogate model $\hat{f}(x)$ on $\mathcal{D}$ to approximate black-box simulation $f(x)$. [3.2]
3: Initialize set of quasi-randomly distributed input parameters $x_{\text{init}}$ across initial dataset.
4: Using stochastic gradient descent on $x_{\text{init}}$ to find the set $x_{\text{local}}$ that locally minimizes the objective function $g(\hat{f}(x), y_{\text{target}})$. [3.3]
5: Select the input $x_{\text{best}}$ that produces minimum objective loss according to objective function $g(\hat{f}(x), y_{\text{target}})$. [3.5]
6: Intelligently query the simulation $f(x)$ with the selected $x_{\text{best}}$. [3.6]
7: **if** stopping criteria $= True$ **then**
8: Take $x_{\text{best}}$ and $y_{\text{best}} = f(x_{\text{best}})$.
9: **else**
10: Add $x_{\text{best}}$ and $y_{\text{best}}$ to the training dataset $\mathcal{D}$.
11: Iterate from Step 2. [3.7]

## 3.1 Initial Data Collection

Each input to the simulation is bounded by physical constraints inside the trajectory planning simulation. These bounds can be adjusted based on historical knowledge of the simulation to reduce the size of the input parameter space. In this application, the 13-dimensional input parameters are bounded around a known set of nominal inputs that produce an acceptable landing of the hypersonic vehicle. The bounds can be increased or decreased depending on the vehicle requirements and the success of searching different areas of the input parameter space.

To create sufficient coverage of the high-dimensional input parameter space, the initial input values were sampled using a Sobol sequence [30]. Sobol sequences are low-discrepancy, quasi-random sequences that provide a more uniform distribution than compared to other quasi-random sampling methods such as Latin hypercube [31]. Low-discrepancy sequences use a measure of uniformity to minimize the difference between the percentage of samples falling in a certain region of the input parameter space and the percentage of volume occupied by this space. Sobol sequences have been proven to be effective for initial sampling for surrogate models [32].

Enough data samples from the simulation must be collected to sufficiently train the neural network. Too few samples will lead to the surrogate model being a poor representation of the actual simulation and too many samples lead to an unnecessarily high computation cost and time. This methodology aims to reduce the total number of queries to the simulation while balancing exploration and exploitation.

Although the input parameter space is large, the number of samples can be reduced because the surrogate model only needs to understand the gradients of the predicted outputs based on the inputs. Then with each successive iteration and simulation query, the surrogate model's fidelity will improve in the most promising regions of the input parameter space. An initial sample size of 400 points was chosen to collect input-output mapping for training the surrogate neural network. Increasing the size of the training data past 400 points leads to diminishing improvements to the surrogate's model accuracy as shown in Fig. 3-2.

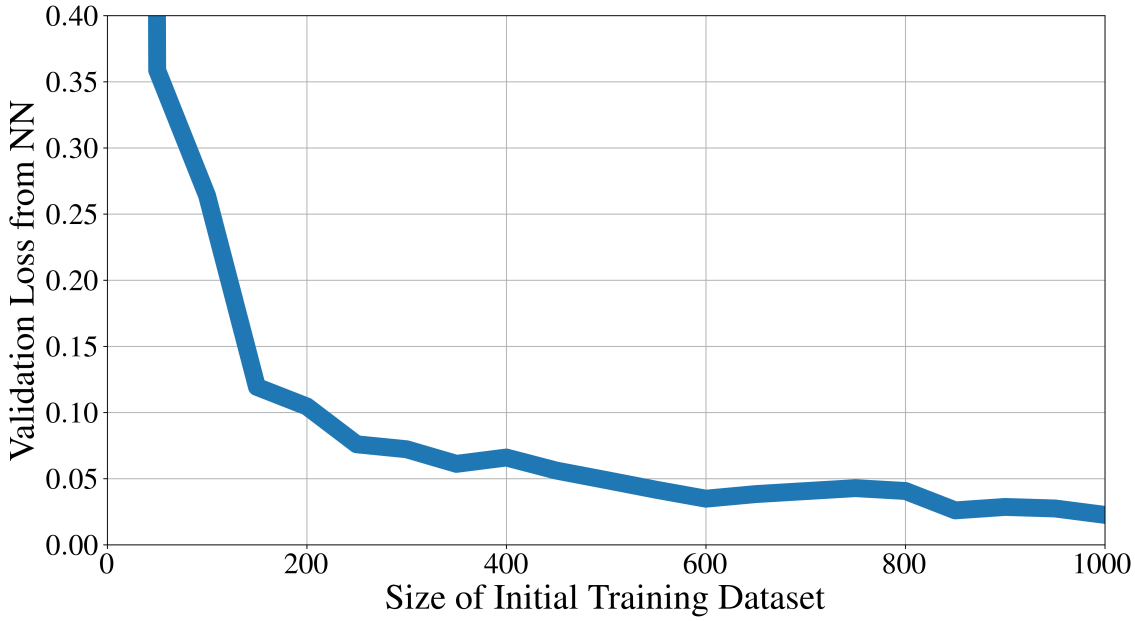


Figure 3-2: Normalized surrogate validation losses from surrogate neural network depending on size of training data set. Diminishing returns on minimizing validation loss after 400 data points.

## 3.2 Surrogate Model Training

Once the initial sampling is done, the data must be standardized for training a neural network to be the best possible surrogate model to accurately represent the simulation. The neural network is trained to minimize the mean absolute error (MAE) between the outputs predicted by the network and the outputs from the training data using the Adam optimizer in PyTorch. The weights of the network are tuned to reduce the MAE using a variant of stochastic gradient descent with backpropagated gradients. The model can be further improved by tuning the hyperparameters defining the neural network's numbers of layers, number of nodes per layer, batch size, and learning rate. The hyperparameters were optimized using the Optuna software library [33] integrated with PyTorch.

Ideally, the neural network would be able to perfectly predict the three outputs of the simulation based on the same set of inputs. Due to the highly nonlinear nature of the simulation and the limited size of the training data, this is not possible. A comparison of the predicted versus actual outputs is shown in Fig. 3-3 for all three

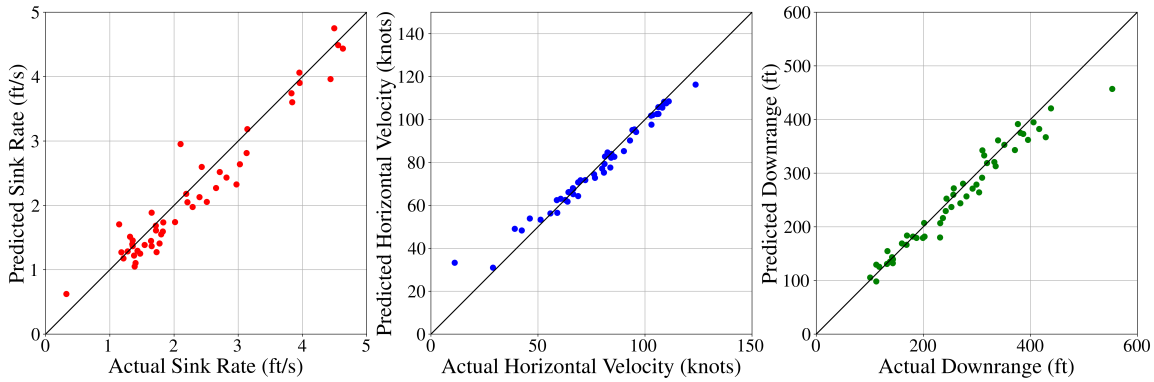


Figure 3-3: Predicted outputs from surrogate neural network versus actual outputs from simulation. Ideally, these outputs would be perfectly correlated but the neural network is able to mostly accurately predict all three outputs with a few outliers. Sink rate is the most challenging output to predict.

outputs of the simulation. If the surrogate neural network was a perfect model, all the points would fall exactly on the diagonal black line. In Fig. 3-3, it is clear sink rate is the hardest of the three outputs to predict accurately due to the larger distribution away from the diagonal line. This is further exemplified in Fig. 3-4 by sink rate loss converging to a slightly higher validation MAE (surrogate loss) than horizontal velocity and downrange position loss.

Fig. 3-5 shows the validation MAE for each output as it converges over epochs. The scale of this validation loss varies based on the outputs feasible range in their physical units. For example, the possible range for sink rate only varies between 0 and 5 ft/s, so the surrogate model predicting to an accuracy under 0.2 ft/s is quite good. Similarly, when the feasible range of physical values for downrange position is 0 to 500 feet, predicting within 10 feet is good.

## 3.3 Custom Objective Function

The objective of this constrained optimization problem is to minimize the distance between desired and predicted outputs. For this application, the objective function to be minimized evaluates the mean absolute error between the desired outputs and the predicted outputs from the surrogate model. MAE calculates the average of

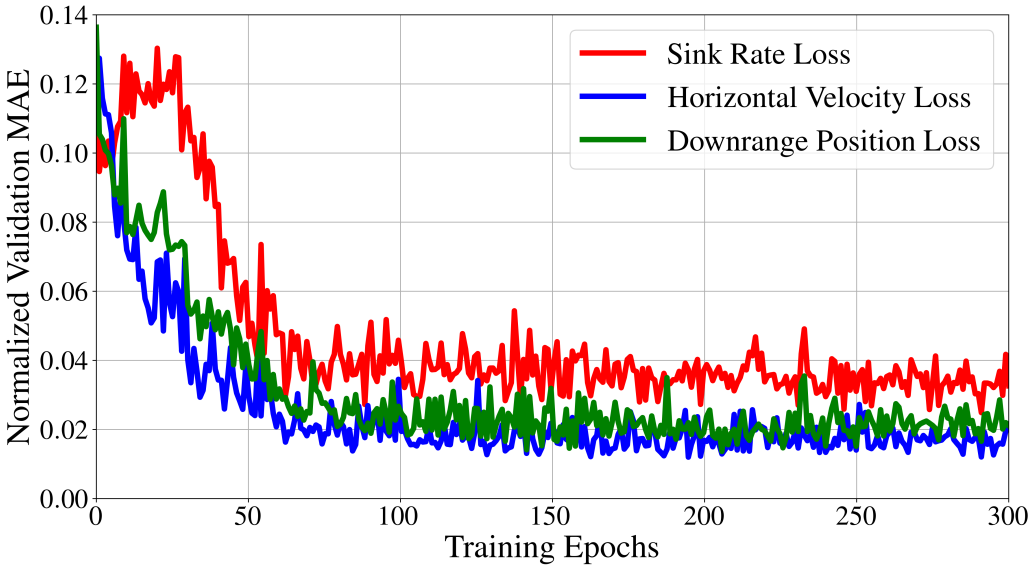


Figure 3-4: The validation surrogate loss from all three outputs converge to low values. Sink Rate is the hardest of the outputs to predict due it slightly higher final normalized loss.

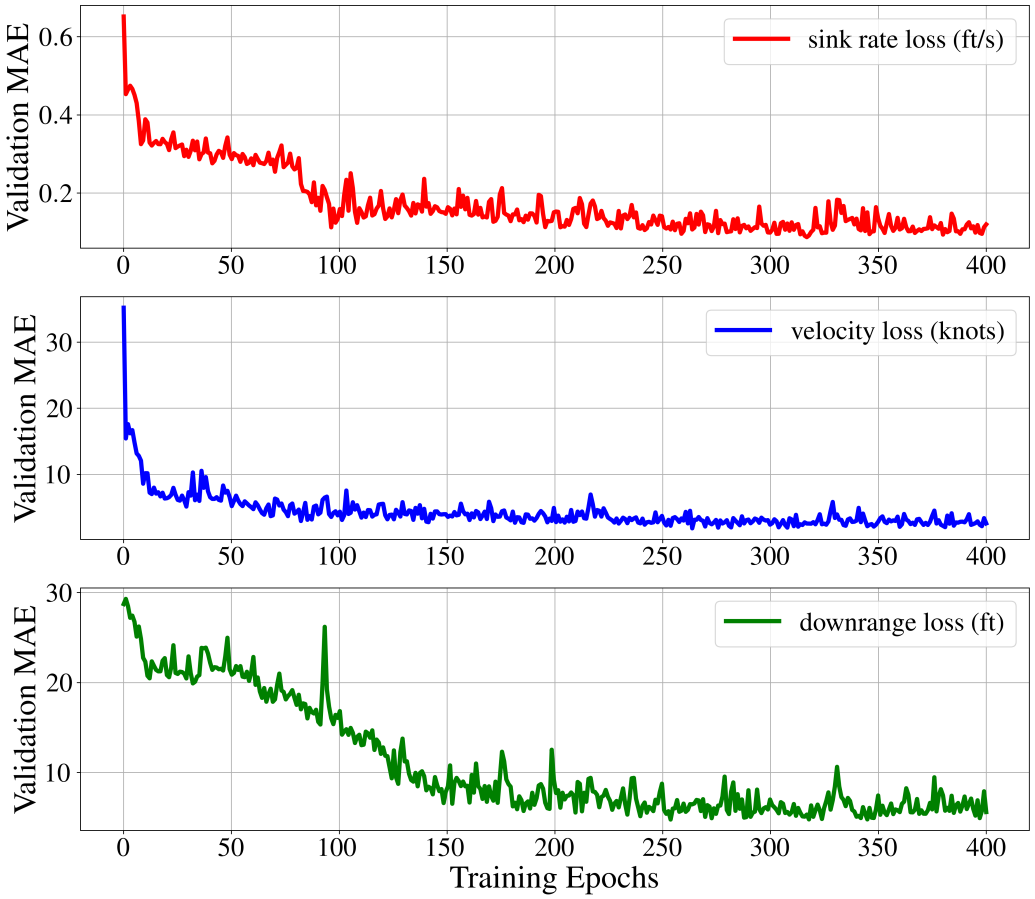


Figure 3-5: Validation surrogate losses for each simulation output over training epoch. All three outputs converge to low loss relative to their scale.

the absolute distance between all the predicted and desired outputs. MAE is more robust to outliers and ensure stable gradients with the different terms in the objective function. This makes the objective function

$$g(\hat{y}) = |y_{\text{target}} - \hat{y}| \,. \tag{3.1}$$

However, the parameters being optimized are not the three predicted outputs from the surrogate model $\hat{y}$, but the 13 inputs to the surrogate model $x$, making the objective function

$$g(x) = \left| y_{\text{target}} - \hat{f}(x) \right| . \tag{3.2}$$

Furthermore, each of the desired outputs can be weighted based on domain knowledge of the simulation. In this application, sink rate is more important to a successful landing than either horizontal velocity or downrange position. Therefore, the MAE or objective loss of predicted versus target sink rate is scaled by a factor of two. And the desired downrange position is too high to be realistically achievable by any combination of input values, so its MAE is reduced by a factor of 10 to ensure it does not overpower the other two terms.

To constrain the inputs to the physically feasible parameter space as well as the training data limits, the inequality constraints are added directly to the objective function as a soft penalty. When either the upper or lower bounds are violated, these penalties are triggered as

$$\begin{aligned} g(x) = \left| y_{\text{target}} - \hat{f}(x) \right| + \alpha \max\left(0, x_{\min} - x\right) \\ + \alpha \max\left(0, x - x_{\max}\right) \end{aligned} \tag{3.3}$$

where $\alpha = 1$, but can be tuned depending on the scale of the other components of the objective function. These penalties contribute to the objective loss and act to steer the inputs being optimized to stay in the constrained input parameter space through gradient descent. The gradients of the objective function with respect to the inputs

are used to optimize the inputs to minimize the objective function. The gradient of the objective function with respect to the inputs when the input bounds are violated is

$$\frac{\partial g}{\partial x} = \hat{f}'(x) \left( \frac{\hat{f}(x) - y_{\text{target}}}{\left| y_{\text{target}} - \hat{f}(x) \right|} \right) \pm \alpha \tag{3.4}$$

As defined in Table 2.2, the outputs are sink rate, downrange position from the start of the runway, and horizontal velocity of the HV at touchdown. The target values for each of the outputs is determined by domain knowledge and vehicle requirements. It is important to note that the number of desired outputs as well as their values could be changed, and this novel algorithm would still work. For example, if the requirement for sink rate was lowered 1 ft/s, the objective function can be adjusted to produce a new optimal reference trajectory. The constraints could be adjusted as well with no loss of functionality to the algorithm.

## 3.4 Sensitivity Analysis and Objective Loss Landscape Visualization

It is impossible to easily visualize the gradients of the objective function with respect to all 13 inputs. While it would be possible to do a dimensional reduction to two dimensions, because these inputs have real meanings the two most important inputs can simply be identified. A sensitivity analysis was conducted to identify the two most important inputs, meaning the two inputs that most affect the output of the simulation. This was done by excluding one input at a time from the NN training and evaluating which input increased the mean surrogate loss the most, therefore most effecting the neural network's ability to accurately predict the outputs. In Fig. 3-6, the two most important inputs for predicting the outputs are shown to be the landing velocity of the HV and the circular flare radius. When either of those inputs are excluded from the neural network training data, the neural network struggles to accurately predict the simulation outputs therefore leading to higher validation loss.

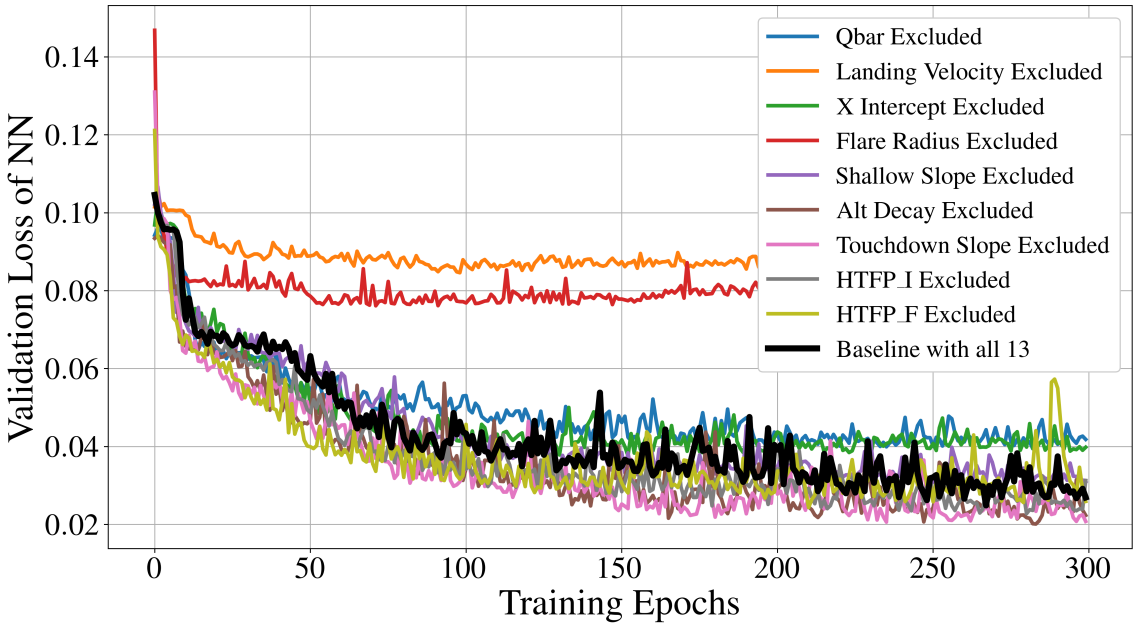


Figure 3-6: Results from sensitivity analysis. Landing velocity and flare radius inputs have the most significant impact on the accuracy of the surrogate model.

With the two most important inputs identified, how varying the inputs affect the objective function can be visualized. While not necessary for optimization, these visualizations are to better understand and validate this methodology. Linearly spaced samples across the input range for landing velocity and flare radius were selected while all other inputs were frozen to their nominal values. Each combination of landing velocity and flare radius was fed through the trained surrogate neural network, and the corresponding output was evaluated by the objective function to calculate loss. This objective loss landscape in Fig. 3-7 shows how the loss from the objective function varies across the two most important inputs, as predicted by the surrogate model.

It is evident that the surrogate neural network has identified regions in the input parameter space as more promising for minimizing the objective function than others. Specifically, a trough in the middle where both variables increase proportionally and a section of low flare radius and high landing velocity. To validate the surrogate model, the true loss landscape can be calculated by taking the same combinations of landing velocity and flare radius and querying the simulation. The resultant true loss

landscape is shown in Fig. 3-8. The predicted objective loss landscape very closely resembles the true objective loss landscape, showing the same promising trough in the middle and to the bottom right. The minimum objective loss shown in predicted loss landscape is 0.18, compared to the true loss landscape minimum of 0.27. This means the surrogate model is more optimistic about how optimal of a trajectory it can achieve.

## 3.5 Input Optimization

With the initial surrogate model trained, the 13 inputs can now be optimized to minimize the objective function using gradient-based descent. Similar to how the gradients of the neural network are used to optimize the weights during the initial training to minimize surrogate loss, the gradients are used to optimize the inputs to the simulation to minimize the objective function loss (3.3).

One of the issues with gradient-based optimization is the convergence to local minima or maxima. To mitigate this concern, many input vectors are optimized. These input vectors are quasi-randomly distributed across the input parameter space using a Sobol sequence. Even if some of the inputs become stuck at local minima or plateaus, at least one of the vectors will find the global minimum.

Through trial-and-error, the best optimizer was found to be stochastic gradient descent with added momentum. Because the many inputs are optimized in batches, this process is extremely rapid compared to the time it takes to query the simulation, seconds as opposed to minutes. This means it is computationally cheap to initialize hundreds of inputs and optimize over many steps to ensure convergence. In the methodology proposed in this paper, 100 input vectors are optimized over 500 steps.

This process is visualized in Fig. 3-9 using the same predicted objective loss landscape from the previous section. Again, the inputs are limited to the two most important parameters, landing velocity and flare radius, so the optimization corresponds to the objective loss landscape background. Only 10 of the 100 input vectors are shown, their initial and final points shown in red and green, respectively. The

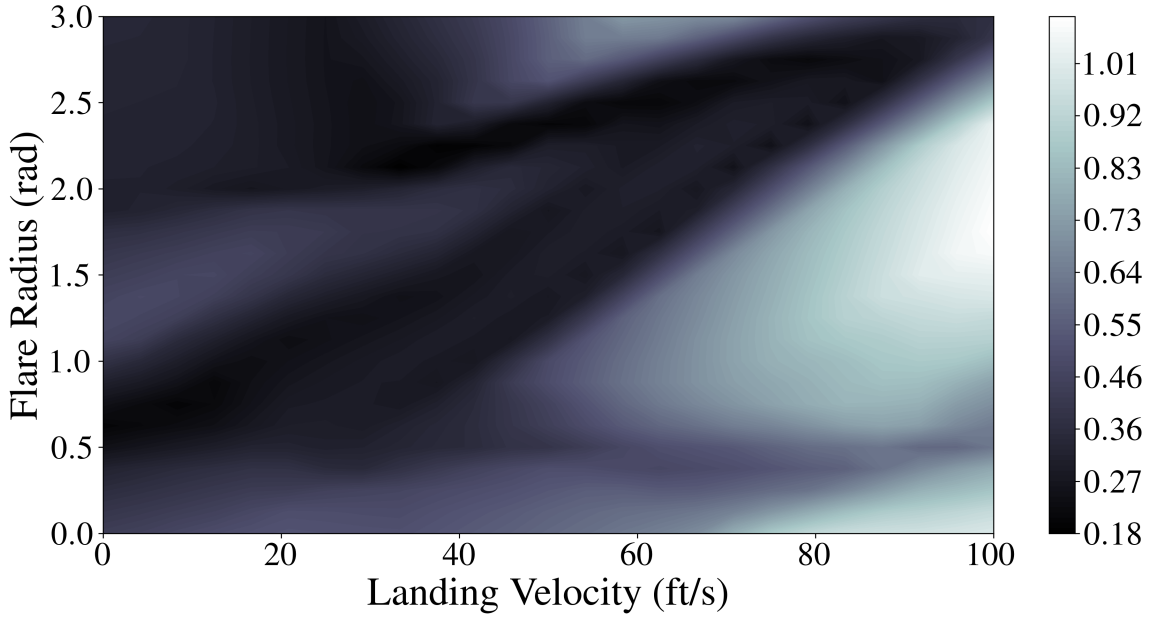


Figure 3-7: Objective loss landscape as predicted by surrogate neural network. Resembles the true loss landscape shown below very closely as the surrogate neural network identified the same regions of the input parameter space to minimize objective loss. The predicted objective loss landscape is more optimistic about minimizing loss than the actual simulation.

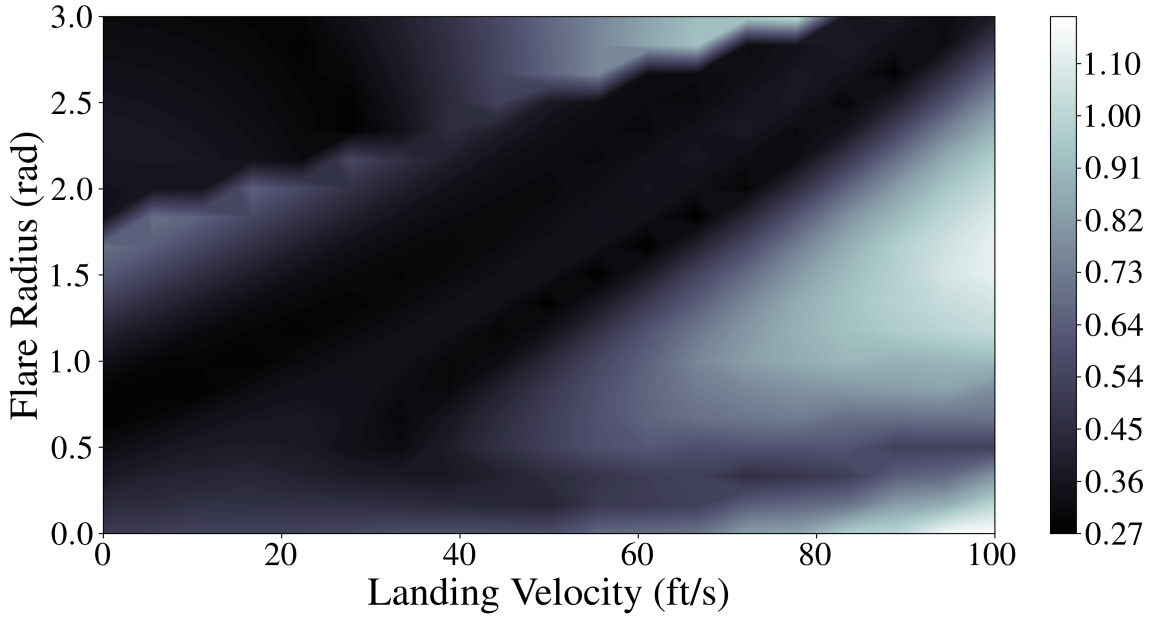


Figure 3-8: Objective loss landscape from real outputs of the trajectory planning simulation. The combinations of landing velocity and flare radius that created the best outcomes from the simulation are reflected in the gradients of the predicted objective loss landscape above.

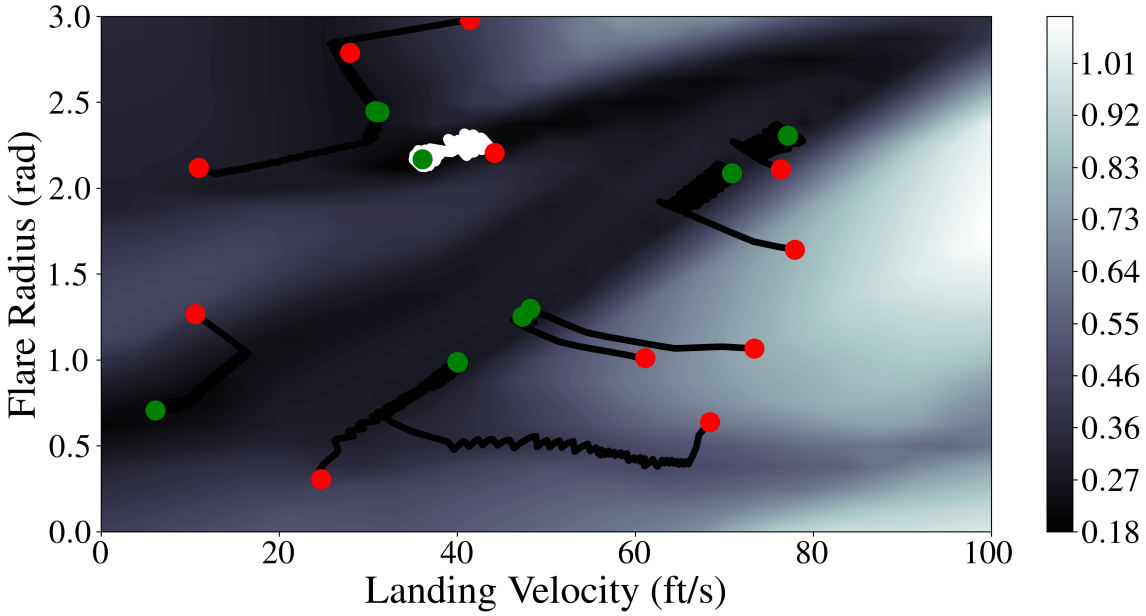


Figure 3-9: Optimization of inputs across objective loss landscape for a single algorithm iteration. The input vectors are optimized from places with higher loss to areas that, according to the surrogate model, will reduce the loss from objective function.

input vector that finds the global minimum has its path in white instead of black.

The input vectors clearly travel down the gradient from places with higher estimated loss to places with lower estimated loss over optimization. It is also shown that the input vectors are successfully constrained within the bounds of the inputs. This visualization represents the optimization step of a single iteration of this algorithm, not the overall simulation-based optimization of the trajectory planning algorithm.

## 3.6 Intelligent Querying

The input vector that found the global minimum of the objective loss landscape is now sent to the actual simulation. While there are many input vectors that show promising results for minimizing the objective function according to the surrogate model, only the best is sent to the trajectory planning simulation. The input selected to query the simulation is intelligently selected instead of being randomly selected or intuitively chosen by an engineer. This reduces the overall computational cost of optimizing the simulation and finding an exceptional reference trajectory for the HV.

## 3.7 Iteration

Each query to the simulation take 3-4 minutes to execute which dominates the majority of the time to run a full iteration of this algorithm. Training each neural network and optimizing the inputs to minimize objective loss takes less than 1 minute. This algorithm will iteratively train a new neural network, optimize the objective function, and intelligently query the simulation until the stopping criteria is met. The stopping criteria for this application checks for three conditions every iteration:

1. If the goal criteria for the outputs been met
2. If the algorithm converged on a solution outside of the goal criteria
3. If the maximum number of iterations been met

Each iteration of the algorithm allows the surrogate model to improve its representation of the actual system and increase its fidelity in areas that appear promising for minimizing objective loss. Without initializing and training a new neural network each time, the algorithm would always search around the same global minimum. Instead, the random weight initialization for each iteration's surrogate model means each neural network will be slightly different. Furthermore, having only one surrogate model does not capture any 'model uncertainty'. Creating a new model every time is more similar to ensemble learning and is shown to be more robust to outliers [34]. This encourages more exploration during optimization instead of continually exploiting the same promising regions.

The first iteration's objective loss landscape can be compared to the last iteration's objective loss landscape in Fig. 3-10 and Fig. 3-11, respectively. Over 50 iterations, the objective loss landscape learns different areas of the input parameter space are more promising for optimization and the lowest predicted objective loss decreases from 0.21 to 0.15. These two objective loss landscapes can be compared by mapping the difference of predicted loss from the first iteration to the 50th, as shown in Fig. 3-12. This comparative objective loss landscape shows the areas of the input parameter space where the model changed its predicted objective loss. The areas that appeared

promising in the first iteration decreased in predicted objective loss while the areas that didn't increased in objective loss.

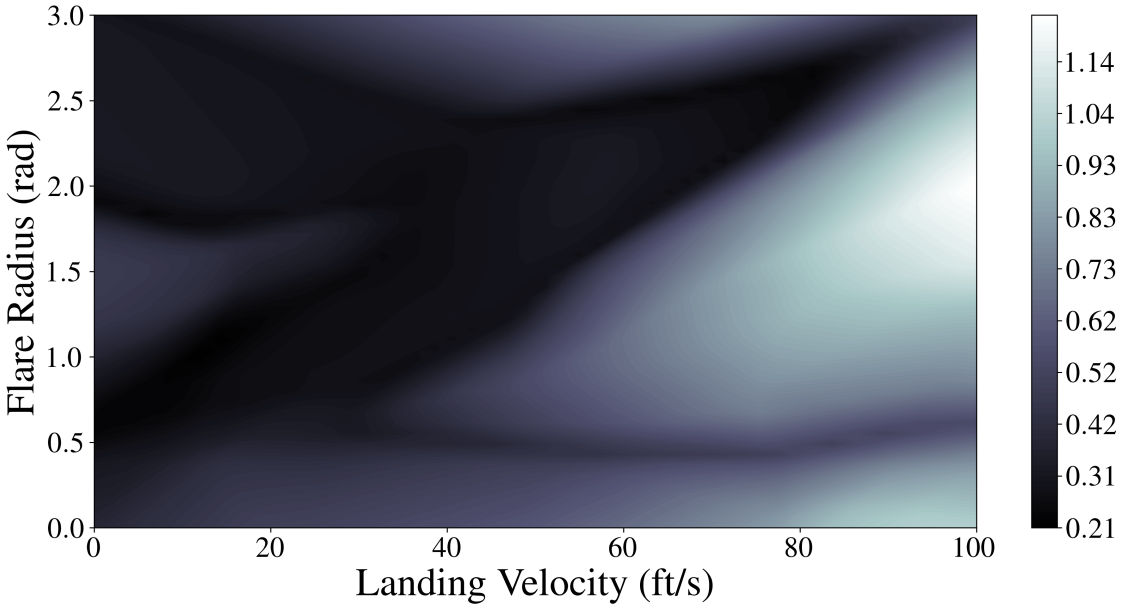


Figure 3-10: Initial objective loss landscape predicted by surrogate neural network from only training data before any intelligent queries added. Predicted lowest objective loss is 0.21.

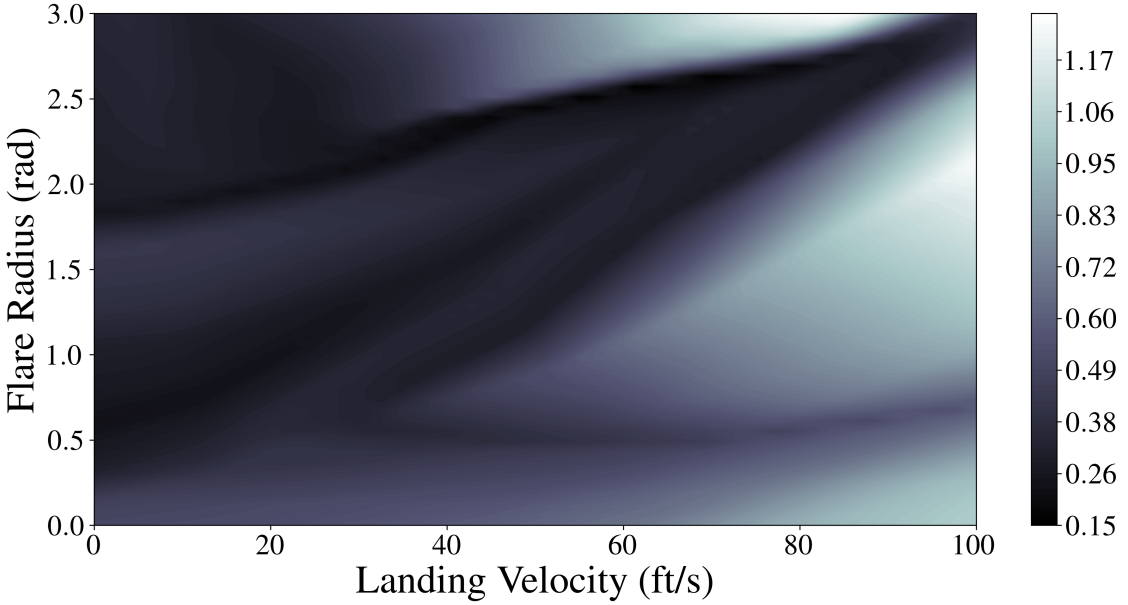


Figure 3-11: Final objective loss landscape after 50 new intelligent queries to the simulation added to the training data. The loss landscape is more refined in promising areas of the input parameter space and the surrogate neural network is now predicting the lowest possible objective loss is 0.15.

It is important to note that these objective loss landscapes demonstrate the ability of the algorithm to generate optimal reference trajectories by only tuning the two most important inputs, leaving the other 11 at their nominal values. Because of this, the lowest possible objective loss predicted by this surrogate neural network is higher than objective losses possible if all 13 inputs are tuned over successive iterations.

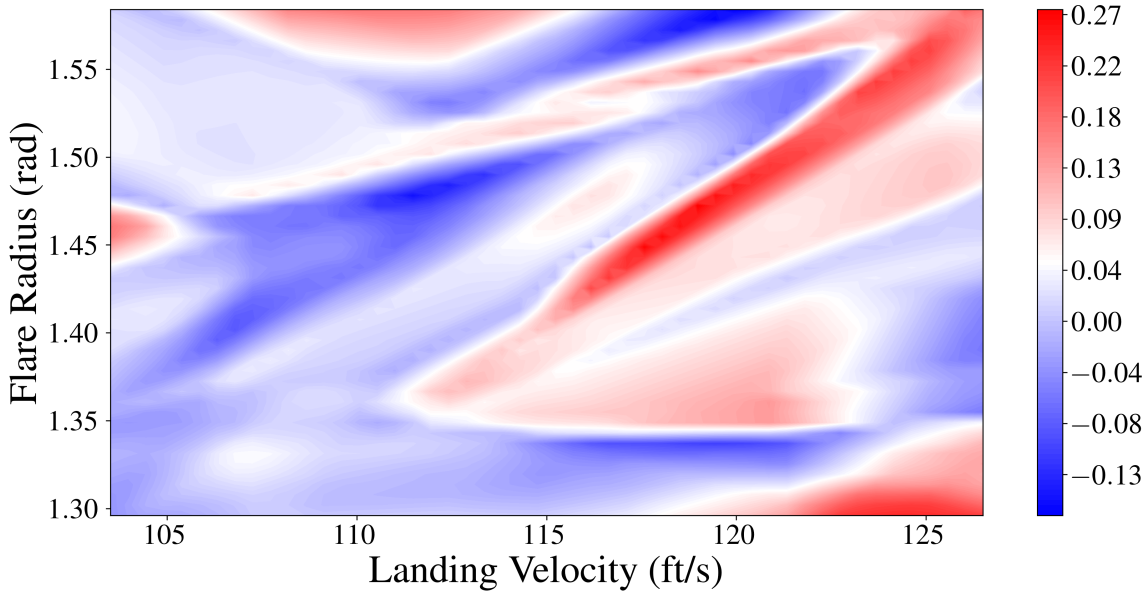


Figure 3-12: Comparison between first and last iteration's objective loss landscape. Areas of the input parameter space that appeared promising initially decreased in predicted objective loss while the areas that didn't increased in objective loss.

# Chapter 4

# Numerical Results

## 4.1 Optimal Reference Trajectory

This section demonstrates the success of this algorithm when applied to optimizing the reference trajectory for a proprietary hypersonic vehicle. This algorithm found a new reference trajectory for approach and landing that outperformed the existing nominal solution. The new reference trajectory produced a 74% decrease in loss from the objective function when compared to the nominal solution as shown in Table 4.1.

Table 4.1: First Reference Trajectory Comparison

| | Performance Results of Trajectories | | | |
|---|---|---|---|---|
| | Sink Rate (ft/s) | Horizontal Velocity (knots) | Downrange Position (ft) | Loss |
| *Target* | *2.00* | *54.0* | *400.0* | *0.00* |
| Nominal | 1.48 | 69.9 | 193.2 | 0.24 |
| New | 2.00 | 53.8 | 378.4 | 0.11 |

The algorithm is very successful at finding values for the 13 simulation inputs that result in values for sink rate and horizontal velocity almost exactly at their target values. Sink rate is the most important of the outputs to achieve and was weighted more heavily in the objective function so achieving a sink rate of 2.00 ft/s demonstrates the effectiveness of this algorithm in finding optimal reference trajectories. However,

downrange position is harder to achieve due to internal simulation constraints. The algorithm is essentially trying to maximize downrange position without compromising the results for sink rate and horizontal velocity.

This new optimal reference trajectory was obtained by training the surrogate neural network on an initial dataset of 400 points quasi-randomly selected across the input parameter space. The set of input parameter values that produced this optimal reference trajectory were found after only 10 queries to the actual simulation. The optimal set of input parameters to the simulation are compared to their nominal (prior best solution) in Table 4.2. Several of the inputs needed to be adjusted by over 100% in order to achieve the optimal landing of the hypersonic vehicle. These changes demonstrate the ability of this tool to generate significantly better reference trajectories as compared to hand-tuning.

Each iteration of the algorithm attempts to reduce the objective loss as much as possible. An increase in objective loss in iterations is due to the surrogate model predicting a certain set of inputs could achieve a lower loss than is actually feasible from the simulation. This could demonstrate exploration of the input parameter space. The objective loss of the true outputs from the simulation is shown over those 10 iterations until the end condition of reaching desired values is satisfied in Fig. 4-1. The target objective loss is not 0.0 due to simulation's inability to achieve a perfect downrange position due to physical constraints of the vehicle.

Furthermore, the three outputs of the simulation are shown over those 10 iterations converging to their target outputs in Fig. 4-2. Over 10 iterations, the surrogate neural network is learning a better understanding of the promising regions of the input parameter space. This can be shown through the reduction of difference between the predicted and actual outputs. For example, the initial percent difference between predicted and actual sink rate is 8.9% from the first iteration. By the 10th iteration, that difference has decreased to only 0.3%. It is also interesting to note that the surrogate model believes it can achieve the target horizontal velocity immediately while it takes four iterations to believe it can achieve the target sink rate.

Additionally, the inequality constraints in the objective function can be validated

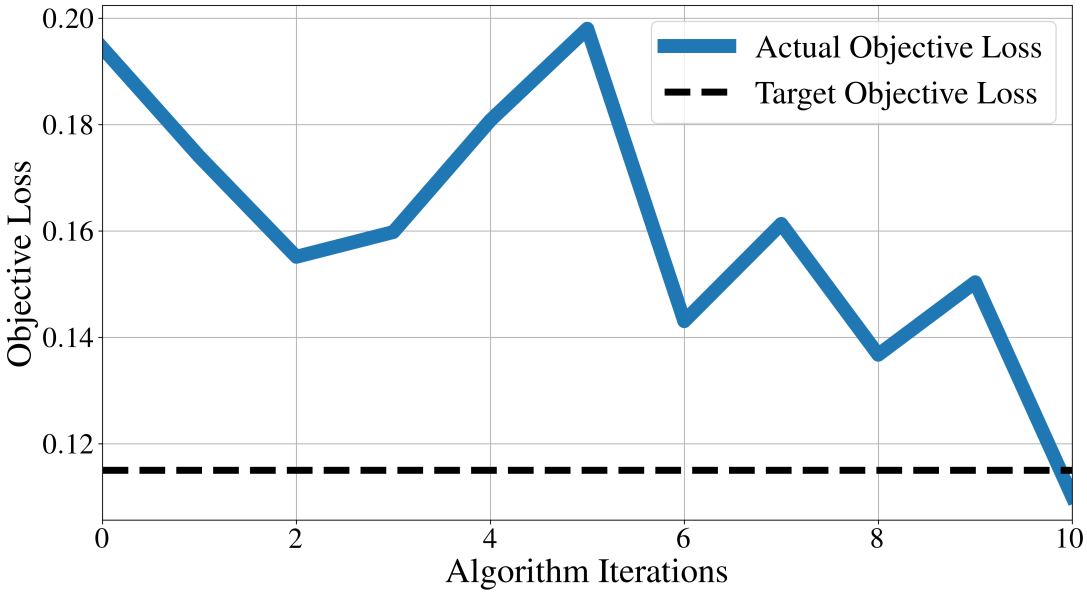


Figure 4-1: Objective loss over algorithm iterations until successful simulation outputs are achieved. Loss varies from iteration to iteration as algorithm searches for optimal input parameter values. Over 10 iterations objective loss decreases by over 50%.

Table 4.2: First HV Input Parameters Comparison

| Parameter Name | Nominal Value | Optimal Value | Percent Difference |
|---|---|---|---|
| qbar_steep | 75 psf | 50 psf | 40% |
| land_vel | 50 knots | 8.7 knots | 140% |
| XAIMPT | 250 ft | 470 ft | 61% |
| FLARE_MAX | 1.5 g | 0.30 g | 130% |
| GAMMA_REF_2 | 1.0 deg | 0.8 deg | 21% |
| HDECAY | 15 ft | 2.9 ft | 135% |
| GAM_FF | 1.0 deg | 1.9 deg | 64% |
| H_TFP_I | 12.5 ft | 16.2 ft | 26% |
| H_TFP_F | 6.0 ft | 8.4 ft | 33% |
| db1 | 0.50 | 0.77 | 43% |
| db2 | 0.50 | 0.46 | 8% |
| db3 | 0.50 | 0.86 | 53% |
| db3 | 0.50 | 0.49 | 2% |

over the algorithm iterations. This can be done by tracking the predicted best inputs over each iteration and comparing them to the input parameter upper and lower bounds as shown in Fig. 4-3. Over 50 iterations, none of the 13 inputs are optimized outside of the bounds. This ensures the neural network optimization stays within the

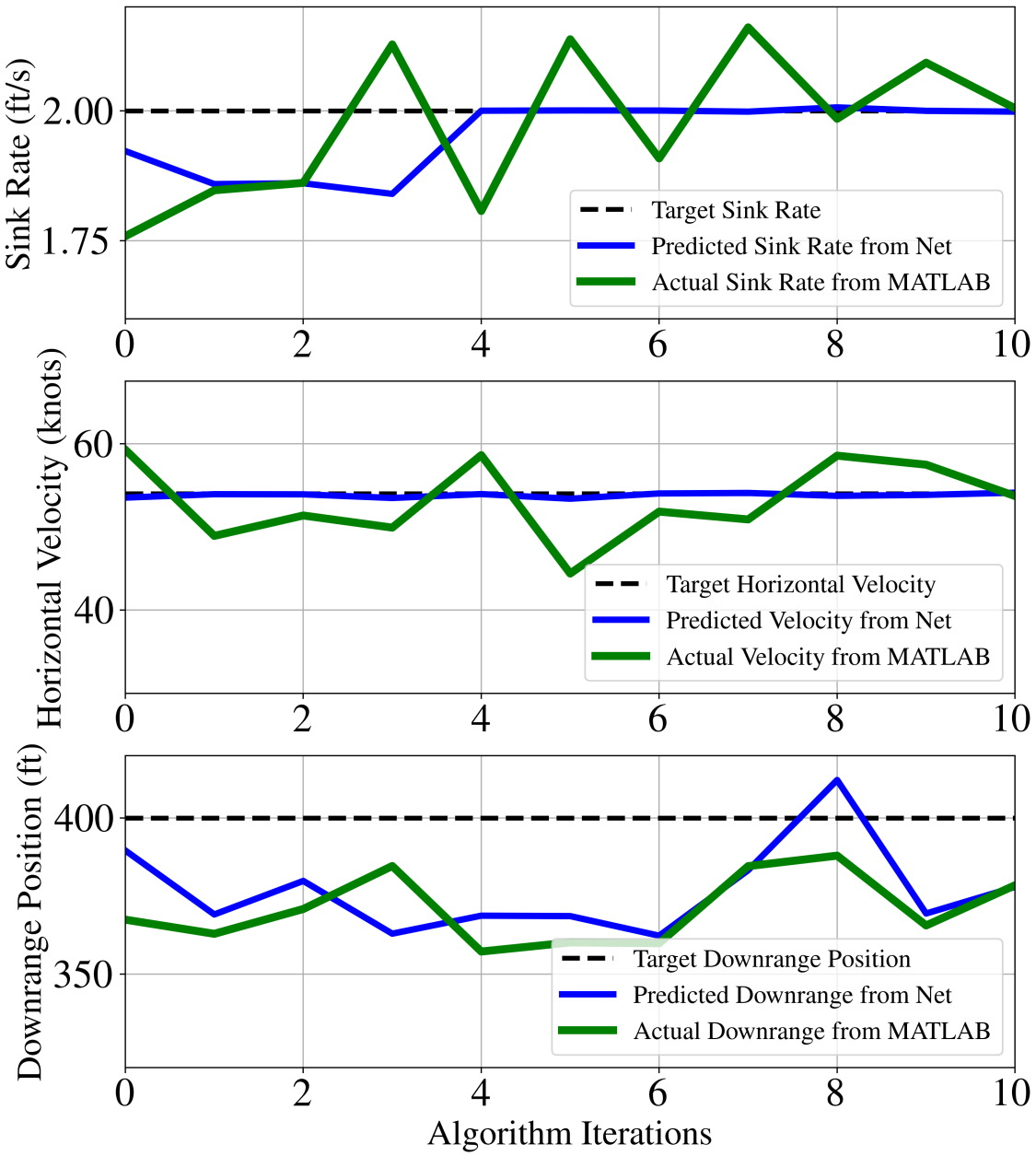


Figure 4-2: True outputs from simulation compared to predicted outputs from surrogate neural network over 10 algorithm iterations. Ideally, over each iteration both predicted and actual outputs would converge to the target output. The algorithm believes from the very first iteration it can achieve ideal horizontal velocity while it takes four iterations to believe it can achieve ideal sink rate. By the 10th iteration, the actual simulation outputs for sink rate and horizontal velocity have converged to their target values.

training data and the queries to the simulation are constrained to the physical range.

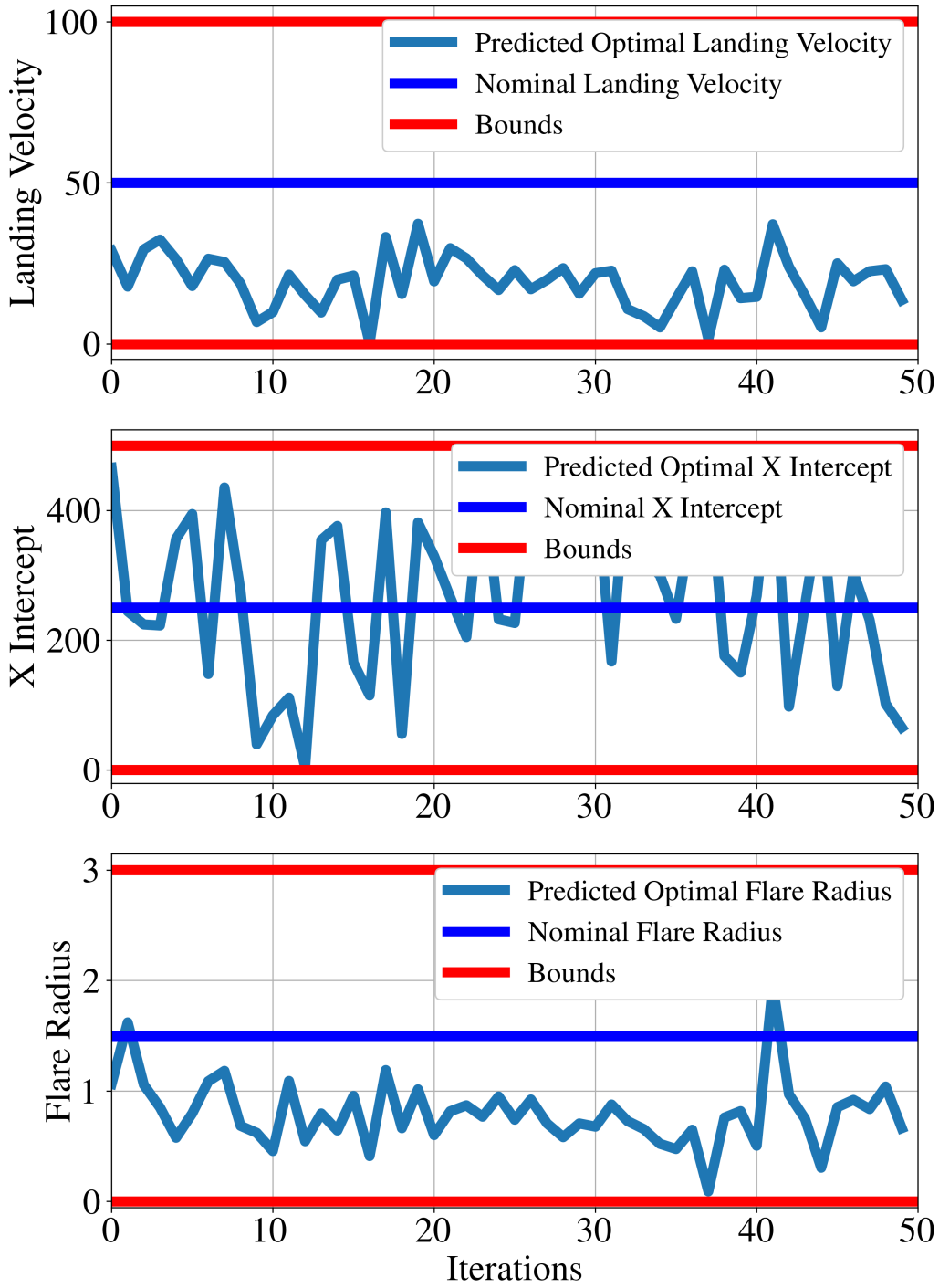


Figure 4-3: Three inputs over algorithm iterations to validate they stay within their constraints. The inputs that minimize the objective loss every iteration stay within the constraints imposed on them in the objective function through loss penalties.

## 4.2 Second Optimal Reference Trajectory

The values for simulation outputs targeted in Section 4.1 were determined by domain knowledge and vehicle requirements. Specifically, sink rate and horizontal velocity values are typically limited by the performance of the landing gear, meaning they ensure the HV does not hit the runway to hard or too fast. The desired sink rate balances a soft touchdown while allowing for vehicles with higher lift to touchdown on the runway. Similarly, the desired horizontal velocity balances a slow touchdown without going so slow that the vehicle loses lift.

If the vehicle properties or landing requirements change, the desired values for sink rate, horizontal velocity, and downrange position may change as well. For example, if the landing gear requires a softer touchdown the target values for sink rate and horizontal velocity must be reduced. To test this algorithm's ability to adapt to different simulation outcomes, the values for sink rate, horizontal velocity, and downrange position are all reduced to 1.5 ft/s, 46 knots, and 300 ft, respectively. This change can easily be implemented due to the flexibility of the objective function.

Once the target values are changed, the objective loss landscape according to the surrogate neural network will change also. The surrogate neural network now understands other areas of the input parameter space are better for minimizing objective loss and optimizing simulation outputs. The new objective loss landscape is shown in Fig. 4-4.

This methodology finds a new set of input parameters to optimize the simulation outcomes in 28 queries. As shown in Table 4.3, this methodology achieves the exact target values for sink rate and horizontal velocity. The target value for downrange position of 300 ft was not feasible for this simulation to achieve.

## 4.3 Monte Carlo Comparison

Although this methodology has proved its ability to optimize the trajectory planning simulation, the true motivation of this algorithm is in its timesaving. It is able

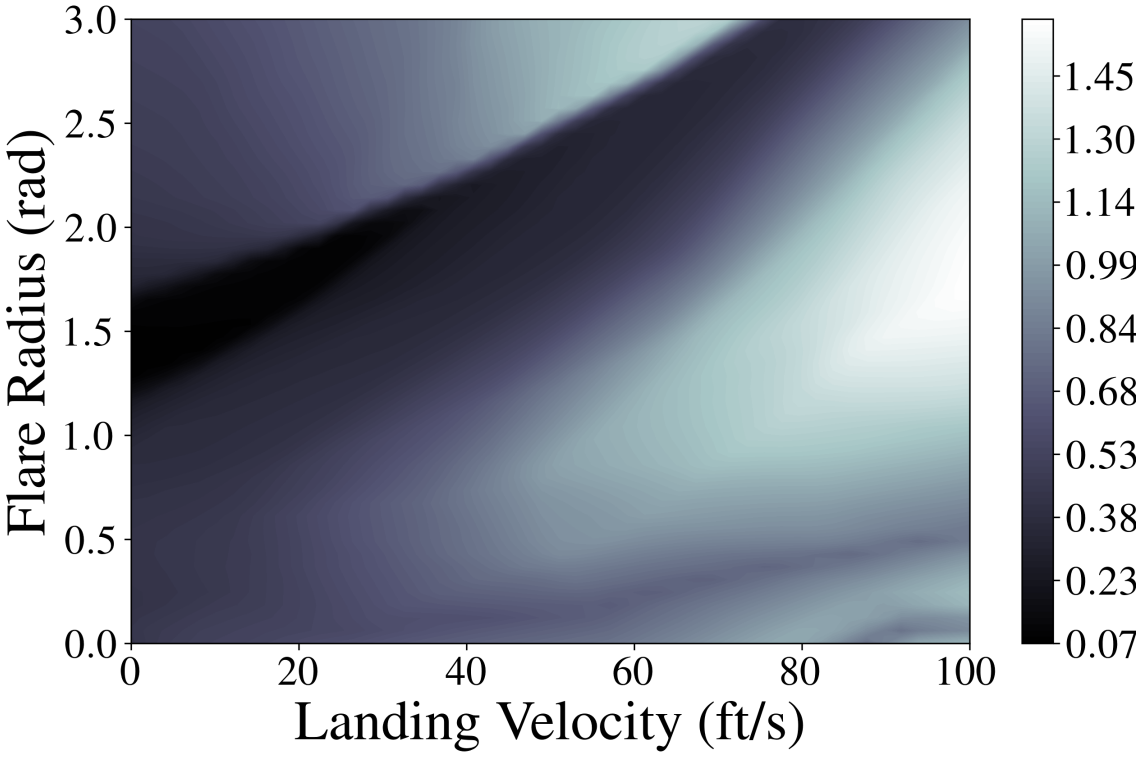


Figure 4-4: Objective loss landscape as predicted by surrogate neural network with lowered target output values. New areas of the input parameter space have been identified to minimize loss from the objective function due to lowered values for sink rate, horizontal velocity, and downrange position..

Table 4.3: Second Reference Trajectory Comparison

| | Performance Results of Trajectories | | | |
|---|---|---|---|---|
| | Sink Rate (ft/s) | Horizontal Velocity (knots) | Downrange Position (ft) | Loss |
| *Target* | *1.50* | *46.0* | *300.0* | *0.00* |
| New | 1.50 | 46.0 | 406.4 | 0.08 |

to intelligently and efficiently search to the input parameter space to find an optimal solution faster than tuning by hand or through a Monte Carlo random search approach.

This can further be proved by tracking the lowest objective loss found so far by the quasi-random sampling and this intelligent algorithm. Over 50 iterations, this approach consistently finds better solutions faster and with less variation than quasi-

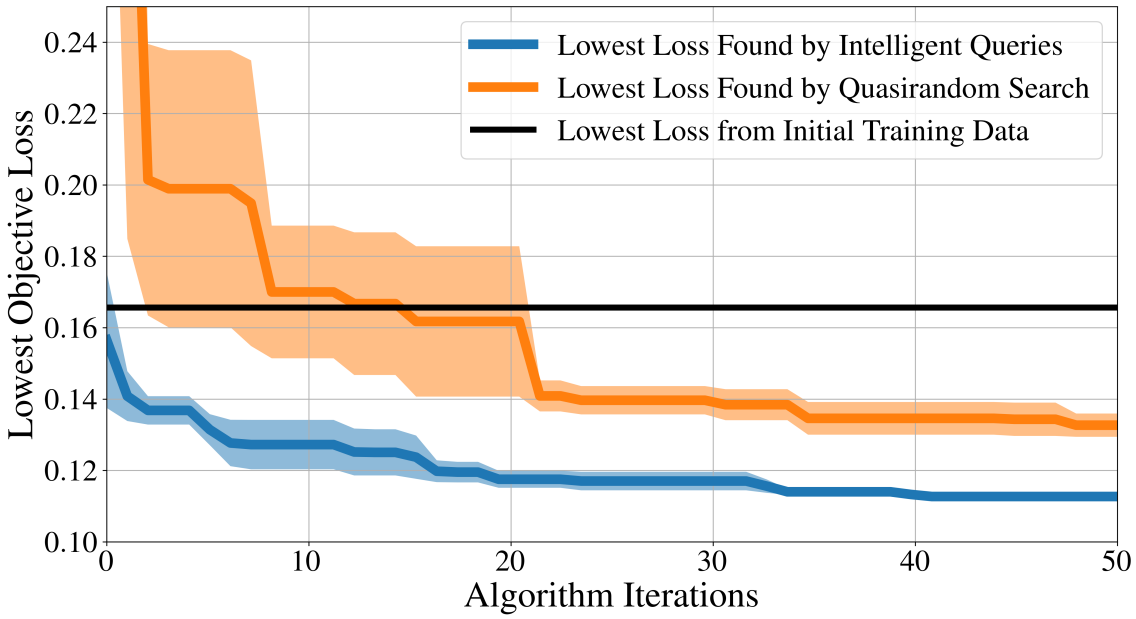


Figure 4-5: Lowest objective loss from intelligent queries compared to quasi-random queries over 50 algorithm iterations with $0.5\sigma$ shaded, averaged over five trials. The intelligent queries result in lower losses quicker than quasi-random queries.

random sampling. Fig. 4-5 shows the loss from the best solution found so far by this algorithm as compared to the quasi-random search, averaged over five trials.

Evidently, this algorithm finds inputs that minimize the objective function much faster than a quasi-random search can. The quasi-random search takes 30 simulation queries to reduce the objective loss from 0.16 to 0.13, while the intelligent search reduces the same loss in the first five queries, making this method six times faster. Furthermore, it consistently reaches a lower objective loss, meaning it finds a better reference trajectory. Similarly, Fig. 4-6 shows how the algorithm will converge to the desired output values faster than the quasi-random search.

Quasi-random search over 50 algorithms is unable to find a set of input values that result in a sink rate of the desired 2.0 ft/s. Compared to the intelligent queries, which finds a sink rate of 2.0 ft/s in 10 queries, this is incredibly inefficient.

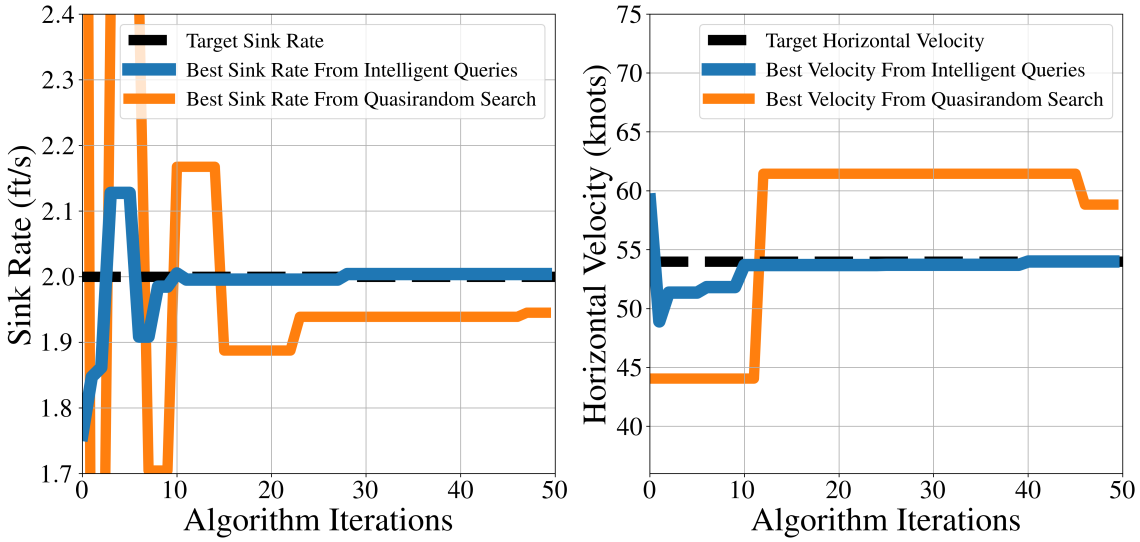


Figure 4-6: Best outputs found by intelligent queries compared to Monte Carlo quasi-random search. The intelligent queries converge much faster to the ideal outputs than the quasi-random search.

## 4.4 Second Hypersonic Vehicle Simulation

All the results so far show the optimization of an A/L reference trajectory based on a simulation of a proprietary hypersonic vehicle. If the underlying physical parameters of the vehicle change, a new reference trajectory will need to be calculated to meet the new set of requirements. For example, if additional wind tunnel testing results in an update to the vehicle aerodynamic model. This methodology can quickly be applied to calculate a new optimal reference trajectory, negating the need for an engineer to spend copious time hand-tuning the input variables.

To test this, the existing HV's aerodynamic properties were perturbed randomly within a range of $3\sigma$. Specifically, the drag was increased by 11%, the lift was reduced by 1% and the pitch moment coefficient was increased by 13%. In practice, the set of inputs found in Section 4.1 are now the nominal results to test and compare further trajectories against. However, the set of inputs that resulted in a high-performing landing for the first hypersonic vehicle, no longer do so for this perturbed vehicle. That set of inputs result in a very low sink rate and therefore a poor landing of the HV. When applied, this methodology finds a better performing reference trajectory

that decreases loss by 100% compared to the inputs from Section 4.1, as shown in Table 4.4.

Table 4.4: Second Hypersonic Vehicle Comparison

| | Performance Results of Trajectories | | | |
|---|---|---|---|---|
| | Sink Rate (ft/s) | Horizontal Velocity (knots) | Downrange Position (ft) | Loss |
| *Target* | *2.00* | *54.0* | *400.0* | *0.00* |
| Previous | 1.21 | 52.5 | 365.8 | 0.51 |
| New | 1.76 | 53.4 | 354.0 | 0.17 |

To find the input values that would result in a high-performing landing of the perturbed hypersonic vehicle, this algorithm had to tune all 13 input values, which is very challenging to do intuitively. Furthermore, many of the inputs had to be changed from their nominal value, some by up to 177% as shown in Table 4.5.

Table 4.5: Second HV Input Parameters Comparison

| Parameter Name | Nominal Value | Optimal Value | Percent Difference |
|---|---|---|---|
| qbar_steep | 50 psf | 34 psf | 38% |
| land_vel | 8.7 knots | 35.9 knots | 122% |
| XAIMPT | 470 ft | 81 ft | 141% |
| FLARE_MAX | 0.30 g | 2.2 g | 152% |
| GAMMA_REF_2 | 0.8 deg | 1.2 deg | 40% |
| HDECAY | 2.9 ft | 2.2 ft | 27% |
| GAM_FF | 1.9 deg | 1.5 deg | 23% |
| H_TFP_I | 16.2 ft | 11.9 ft | 31% |
| H_TFP_F | 8.4 ft | 1.4 ft | 143% |
| db1 | 0.77 | 0.80 | 4% |
| db2 | 0.46 | 0.53 | 14% |
| db3 | 0.86 | 0.99 | 14% |
| db3 | 0.49 | 0.03 | 177% |

This methodology successfully decreased the objective loss however the sink rate is not as close to the target value of 2.00 ft/s as in Section 4.1. This is because the aerodynamic changes are not reflected in the vehicle's control system, meaning tracking the reference trajectory is much more challenging for the HV. However, the

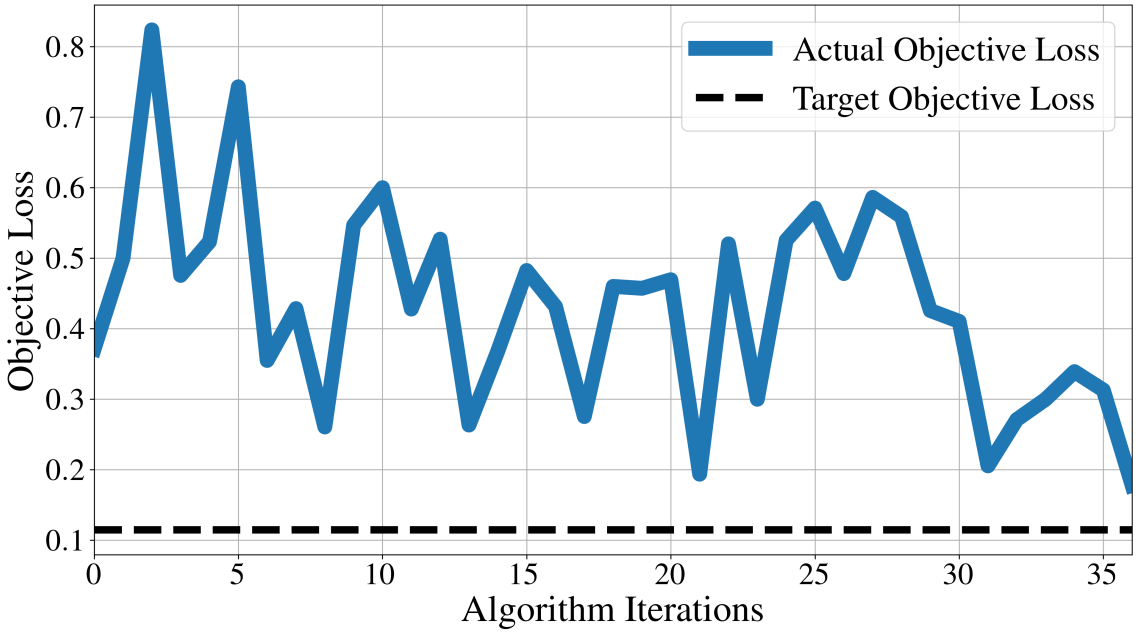


Figure 4-7: Objective loss over algorithm iterations until optimal trajectory is achieved for the aerodynamically altered hypersonic vehicle. Over 37 iterations, the objective loss is decreased by over 70%.

objective loss is significantly reduced from 0.38 to 0.17 as shown in Fig. 4-7, meaning this methodology is finding the best possible trajectory this vehicle can achieve.

As demonstrated in Section 4.3 for the first hypersonic vehicle, this algorithm once again proves to be significantly better at minimizing objective loss. The average loss from the intelligently queries is reduced to a lower value in far few iterations than through a quasi-random selection of input parameter values, as shown in Fig. 4-8. Averaged over three trials, the intelligent queries reduce objective loss from 0.30 to 0.15 over 50 algorithm iterations. In comparison, the quasi-random search is unable to select input values that create a simulation output with loss lower than 0.28 on average.

Similarly, the output values from the simulation converge to their target values much faster through intelligent queries than through quasi-random search, as shown in Fig. 4-9. The quasi-random search over 50 iterations finds the best possible sink rate to be 1.35 ft/s while the intelligent queries are able to achieve a sink rate of 1.76 ft/s, much closer to the target sink rate of 2 ft/s. Over those same 50 iterations,

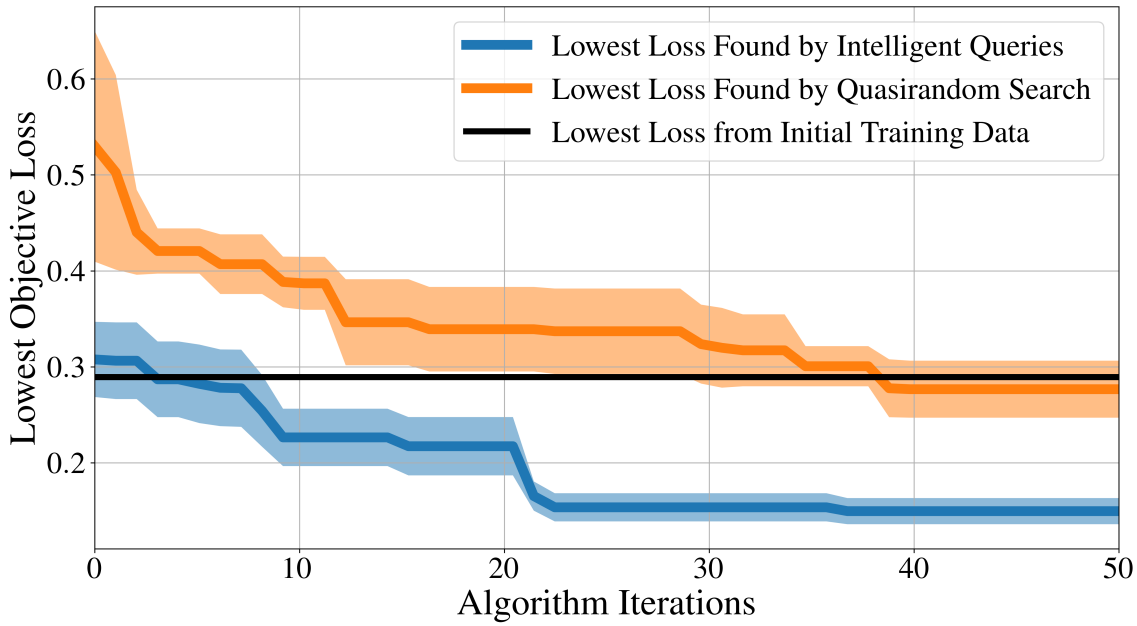


Figure 4-8: Lowest loss from intelligent queries compared to quasi-random queries over 50 algorithm iterations with $0.5\sigma$ shaded for the second hypersonic vehicle, averaged over three trials. The intelligent queries result in lower losses quicker than quasi-random queries.

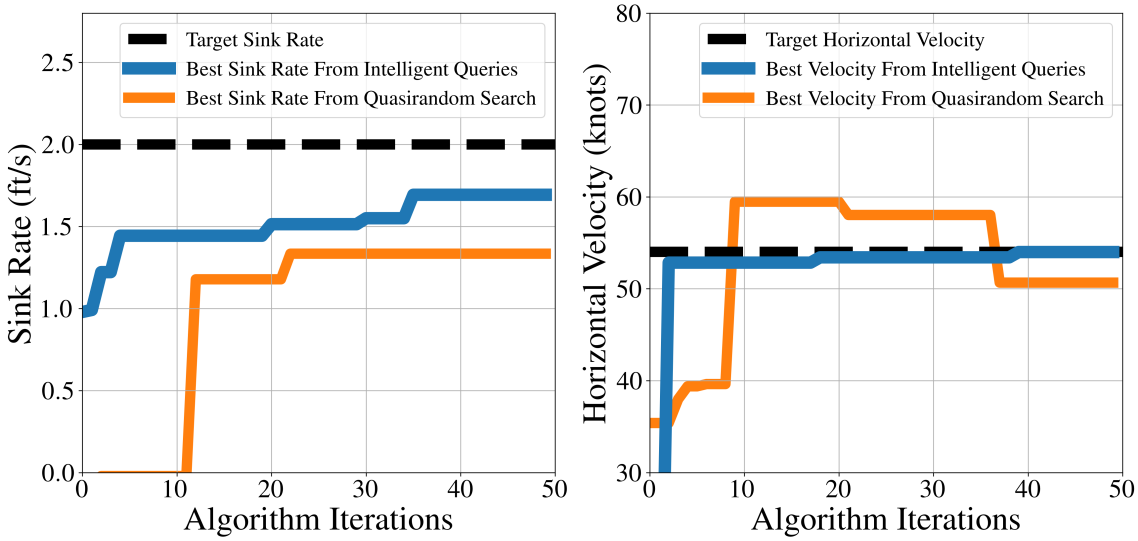


Figure 4-9: Best outputs found by intelligent queries compared to Monte Carlo quasi-random search. The intelligent queries converge much faster to the ideal outputs than the quasi-random search.

the quasi-random search is unable to achieve a horizontal velocity close to the target value of 54 ft/s. The intelligent queries in comparison converge to the target value in less than 20 iterations.

# Chapter 5

# Conclusions and Future Work

This paper presents a novel simulation-based optimization algorithm that uses surrogate neural networks and their ability to produce gradients to drastically reduce computation time. The novelty of this algorithm is that it uses neural network surrogate models to intelligently select queries to the simulation. By doing so, the total number of simulation runs can be reduced while simultaneously finding the optimal reference trajectory. Furthermore, no current optimal trajectory planning algorithms make use of neural networks as surrogate models, their efficiency, and their ability to produce gradients.

When applied to a highly nonlinear approach and landing reference trajectory planning simulation for hypersonic vehicles, this algorithm rapidly optimized the 13 input parameters to produce the best possible result for landing the hypersonic vehicle. It was able to find a reference trajectory that resulted in the target sink rate exactly, the most important metric for determining the performance of landing on the runway. When the value for target sink rate was lowered, this algorithm was similarly able to achieve a high-performing landing of the same hypersonic vehicle. Furthermore, when this algorithm was applied to a different hypersonic vehicle it found a solution that improved the landing of the vehicle by 100% as compared to the previous solution.

Compared to quasi-random search, this algorithm works six times faster to find the optimal output of the simulation by intelligently querying areas that minimize

loss from the objective function as predicted by the surrogate model. Compared to hand-tuning by an engineer, this algorithm reduces time to find an optimal reference trajectory from days to hours. Not only is this algorithm faster, it also finds areas of the input parameter space that lead to simulation outcomes with significantly less objective loss as compared to quasi-random search.

This generalized methodology has been shown to work for different hypersonic vehicles simulations. However, future work includes testing this methodology in an entirely different field. This methodology should be able to accommodate any black-box simulation with any number of inputs and outputs and constraints. Ideally, the simulation should be computationally expensive as to take advantage of the efficiency of this methodology in reducing total simulation queries and therefore total computation time.

Furthermore, total number of queries could be reduced by eliminating the initial quasi-random training dataset so that all queries are intelligently selected by the surrogate model. Initially, the surrogate neural network would have such a poor understanding of the actual simulation due to lack of training data that the initial queries selected to the send to the simulation would essentially be random. Or every other query could be selected at random until a certain level of fidelity is reached with the surrogate model. Eventually, the surrogate neural network would be able to replicate the simulation well enough to start only intelligently selecting queries in promising areas of the input parameter space.

It would also be interesting to explicitly incorporate the epistemic uncertainty of the neural network to better balance and understand exploration versus exploitation. Currently, all uncertainty is innate to the surrogate model's imperfect representation of the actual simulation and the random weight initialization of each iteration's neural network. Instead of training a single neural network every algorithm iteration, an ensemble of models could be trained, all initialized with random weights. This ensemble could be used to quantify what areas of the input parameter space have more uncertainty, meaning the amount of entropy between neural networks.

This novel algorithm was shown to produce exceptional reference trajectories for

two different optimal sets of landing requirements and on two different hypersonic vehicles. It negates the need for any hand-tuning of input parameters and produces better trajectories than were previously thought possible. The application of this methodology will enable much more rapid calculation of optimal approach and landing reference trajectories for hypersonic vehicles.

# Bibliography


[1] A Ammeri. A comprehensive litterature review of mono-objective simulation optimization methods. *Advances in Production Engineering & Management*, 6(4), 2011.

[2] Satyajith Amaran, Nikolaos V Sahinidis, Bikram Sharda, and Scott J Bury. Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, 240:351–380, 2016.

[3] Alison Cozad, Nikolaos V Sahinidis, and David C Miller. Learning surrogate models for simulation-based optimization. *AIChE Journal*, 60(6):2211–2227, 2014.

[4] Evelyn Ruff, Rebecca Russell, Matthew Stoeckle, Piero Miotto, and Jonathan P How. Surrogate neural networks for efficient simulation-based trajectory planning optimization. *arXiv preprint arXiv:2303.17468*, 2023.

[5] Xiaosong Du, Ping He, and Joaquim RRA Martins. Rapid airfoil design optimization via neural networks-based parameterization and surrogate modeling. *Aerospace Science and Technology*, 113:106701, 2021.

[6] Gregg Barton and Steven Tragesser. Autolanding trajectory design for the x-34. In *24th Atmospheric Flight Mechanics Conference*, page 4161, 1999.

[7] Timothy W Simpson, Timothy M Mauery, John J Korte, and Farrokh Mistree. Kriging models for global approximation in simulation-based multidisciplinary design optimization. *AIAA journal*, 39(12):2233–2241, 2001.

[8] Pu Li, Haiyan Li, Yunbao Huang, Kefeng Wang, and Nan Xia. Quasi-sparse response surface constructing accurately and robustly for efficient simulation based optimization. *Advances in Engineering Software*, 114:325–336, 2017.

[9] Stefan Jakobsson, Michael Patriksson, Johan Rudholm, and Adam Wojciechowski. A method for simulation based optimization using radial basis functions. *Optimization and Engineering*, 11:501–532, 2010.

[10] Xiaotao Wan, Joseph F Pekny, and Gintaras V Reklaitis. Simulation-based optimization with surrogate models—application to supply chain management. *Computers & chemical engineering*, 29(6):1317–1328, 2005.

[11] Robert D Hurrion. An example of simulation optimisation using a neural network metamodel: finding the optimum number of kanbans in a manufacturing system. *Journal of the Operational Research Society*, 48(11):1105–1112, 1997.

[12] Partha Majumder and TI Eldho. Artificial neural network and grey wolf optimizer based surrogate simulation-optimization model for groundwater remediation. *Water Resources Management*, 34:763–783, 2020.

[13] J Sreekanth and Bithin Datta. Comparative evaluation of genetic programming and neural network as potential surrogate models for coastal aquifer management. *Water resources management*, 25:3201–3218, 2011.

[14] Baha Y Mirghani, Emily M Zechman, Ranji S Ranjithan, and G Mahinthakumar. Enhanced simulation-optimization approach using surrogate modeling for solving inverse problems. *Environmental Forensics*, 13(4):348–363, 2012.

[15] Gang Sun and Shuyue Wang. A review of the artificial neural network surrogate modeling in aerodynamic design. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 233(16):5863–5872, 2019.

[16] Jun Tao and Gang Sun. Application of deep learning based multi-fidelity surrogate model to robust aerodynamic design optimization. *Aerospace Science and Technology*, 92:722–737, 2019.

[17] Daniel A White, William J Arrighi, Jun Kudo, and Seth E Watts. Multiscale topology optimization using neural network surrogate models. *Computer Methods in Applied Mechanics and Engineering*, 346:1118–1135, 2019.

[18] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

[19] Xinfu Liu and Zuojun Shen. Rapid smooth entry trajectory planning for high lift/drag hypersonic glide vehicles. *Journal of Optimization Theory and Applications*, 168:917–943, 2016.

[20] Chunyun Dong, Zhi Guo, and Xiaolong Chen. Robust trajectory planning for hypersonic glide vehicle with parametric uncertainties. *Mathematical Problems in Engineering*, 2021:1–19, 2021.

[21] Kazuhide Okamoto and Takeshi Tsuchiya. Optimal aircraft control in stochastic severe weather conditions. *Journal of Guidance, Control, and Dynamics*, 39(1):77–85, 2016.

[22] Jianying Wang, Haizhao Liang, Zheng Qi, and Dong Ye. Mapped chebyshev pseudospectral methods for optimal trajectory planning of differentially flat hypersonic vehicle systems. *Aerospace Science and Technology*, 89:420–430, 2019.

[23] Runqi Chai, Antonios Tsourdos, Al Savvaris, Senchun Chai, Yuanqing Xia, and C. L. Philip Chen. Six-dof spacecraft optimal trajectory planning and real-time attitude control: A deep neural network-based approach. *IEEE Transactions on Neural Networks and Learning Systems*, 31(11):5005–5013, 2020.

[24] Andrew Clay Grubler. *New methodologies for onboard generation of terminal area energy management trajectories for autonomous reusable launch vehicles*. PhD thesis, Massachusetts Institute of Technology, 2001.

[25] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[27] Samineh Bagheri, Wolfgang Konen, and Thomas Bäck. Online selection of surrogate models for constrained black-box optimization. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, 2016.

[28] Melanie Coggan. Exploration and exploitation in reinforcement learning. *Research supervised by Prof. Doina Precup, CRA-W DMP Project at McGill University*, 2004.

[29] Pu Zhao, Pin-Yu Chen, Siyue Wang, and Xue Lin. Towards query-efficient black-box adversary with zeroth-order natural gradient descent. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6909–6916, 2020.

[30] Sebastian Burhenne, Dirk Jacob, Gregor P Henze, et al. Sampling based on sobol'sequences for monte carlo techniques applied to building simulations. In *Proc. Int. Conf. Build. Simulat*, pages 1816–1823, 2011.

[31] Diane Donovan, Kevin Burrage, Pamela Burrage, TA McCourt, B Thompson, and EŞ Yazici. Estimates of the coverage of parameter space by latin hypercube and orthogonal array-based sampling. *Applied Mathematical Modelling*, 57:553–564, 2018.

[32] Ingrida Steponavičė, Mojdeh Shirazi-Manesh, Rob J. Hyndman, Kate Smith-Miles, and Laura Villanova. *On Sampling Methods for Costly Multi-Objective Black-Box Optimization*, pages 273–296. Springer International Publishing, Cham, 2016.

[33] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[34] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.