Improvement of Automated POST Case Success Rate using Support Vector Machines

Mathew R. Zwack* and Patrick D. Dees†

Jacobs ESSSA Group, Huntsville, AL, 35806, United States

During early conceptual design of complex systems, concept down selection can have a large impact upon program life-cycle cost. Therefore, any concepts selected during early design will inherently commit program costs and affect the overall probability of program success. For this reason it is important to consider as large a design space as possible in order to better inform the down selection process.

For conceptual design of launch vehicles, trajectory analysis and optimization often presents the largest obstacle to evaluating large trade spaces. This is due to the sensitivity of the trajectory discipline to changes in all other aspects of the vehicle design. Small deltas in the performance of other subsystems can result in relatively large fluctuations in the ascent trajectory because the solution space is non-linear and multi-modal.[1,2]

In order to help capture large design spaces for new launch vehicles, the authors have performed previous work seeking to automate the execution of the industry standard tool, Program to Optimize Simulated Trajectories (POST). This work initially focused on implementation of analyst heuristics to enable closure of cases in an automated fashion, with the goal of applying the concepts of design of experiments (DOE) and surrogate modeling to enable near instantaneous throughput of vehicle cases.[3] As noted in [4] work was then completed to improve the DOE process by utilizing a graph theory based approach to connect similar design points.

The conclusion of the previous work illustrated the utility of the graph theory approach for completing a DOE through POST. However, this approach was still dependent upon the use of random repetitions to generate seed points for the graph. As noted in,[4] only 8% of these random repetitions resulted in converged trajectories. This ultimately affects the ability of the random reps method to confidently approach the global optima for a given vehicle case in a reasonable amount of time. With only an 8% pass rate, tens or hundreds of thousands of reps may be needed to be confident that the best repetition is at least close to the global optima. However, typical design study time constraints require that fewer repetitions be attempted, sometimes resulting in seed points that have only a handful of successful completions. If a small number of successful repetitions are used to generate a seed point, the graph method may inherit some inaccuracies as it chains DOE cases from the non-global-optimal seed points. This creates inherent noise in the graph data, which can limit the accuracy of the resulting surrogate models.

For this reason, the goal of this work is to improve the seed point generation method and ultimately the accuracy of the resulting POST surrogate model. The work focuses on increasing the case pass rate for seed point generation. It is expected that by vastly improving the pass rate, more successful repetitions will be completed in the same time frame, resulting in a higher probability that the global optima is found. This ultimately translates to graph data with less noise and surrogate models with higher degrees of accuracy.

Development of the new approach begins with an assessment of the information provided by the random repetitions method. As repetitions are completed, they are identified as converged or failed, with failures marked by a number of different reasons such as minimum or maximum altitude exceeded (altmin/altmax). When looking at the control vector space, regions of different failure types can be identified, leading the team to consider repetitions

*Systems Engineer, Advanced Concepts Office, George C. Marshall Space Flight Center/ED04, Huntsville, AL, AIAA Member.

†Trajectory Engineer, Advanced Concepts Office, George C. Marshall Space Flight Center/ED04, Huntsville, AL, AIAA Member.

generation as a classification problem. Considering the problem in this fashion, a classifier can be fit to the random repetitions data and then be used to predict the success or failure of a new repetition prior to its submission to POST. If the classifier is accurate enough, in can be used as a filter for new repetitions to improve the probability of convergence.

This paper first addresses the feasibility of fitting a classifier to the random repetitions data. Random repetitions data for a simple example problem is produced and Support Vector Machines (SVM) are utilized to classify the data within the vehicle and control vector space. The classifiers are fit to random repetitions datasets of varying size in order to test the prediction accuracy. The SVM are shown to produce very high prediction accuracies for the given example problem, even when utilizing a relatively small amount of data from the random repetitions. This ultimately allows for a relatively short execution of random repetitions prior to execution of cases through the SVM. A comparison is given to show the time savings of this combined approach versus running straight random repetitions.

In addition, the example problem is expanded to include execution of the graph method from reference [4]. This allows for a full comparison between the SVM and random repetitions approaches in terms of surrogate model fitting. The example problem shows the ability of the SVM to produce more optimal seed points than the random reps method due to its increased case pass rate. Illustration of the differences in surrogate model fitting between the techniques ultimately leads to the conclusion that the SVM approach can produce a surrogate model with higher accuracy in a shorter time frame than the straight random repetitions method.

Nomenclature

ACO	Advanced Concepts Office
DOE	Design of Experiments
MSFC	George C. Marshall Space Flight Center
NASA	National Aeronautics and Space Administration
PMF	Propellant Mass Fraction
POST	Program to Optimize Simulated Trajectories
SLS	Space Launch System
SVM	Support Vector Machine

I. Introduction

During early conceptual design of complex systems, concept down selection can have a large impact upon program life-cycle cost. Therefore, any concepts selected during early design will inherently commit program costs and affect the overall probability of program success. For this reason it is important to consider as large a design space as possible in order to better inform the down selection process.

For conceptual design of launch vehicles, trajectory analysis and optimization often presents the largest obstacle to evaluating large trade spaces. This is due to the sensitivity of the trajectory discipline to changes in all other aspects of the vehicle design. Small deltas in the performance of other subsystems can result in relatively large fluctuations in the ascent trajectory because the solution space is non-linear and multi-modal.[5]

In order to help capture large design spaces for new launch vehicles, the authors have performed previous work seeking to automate the execution of the industry standard tool, Program to Optimize Simulated Trajectories (POST). This work initially focused on implementation of analyst heuristics to enable closure of cases in an automated fashion, with the goal of applying the concepts of design of experiments (DOE) and surrogate modeling to enable near instantaneous throughput of vehicle cases.[3] Additional work was then completed to improve the DOE process by utilizing a graph theory based approach to connect similar design points.[4]

The conclusion of the previous work illustrated the utility of the graph theory approach for completing a DOE through POST. However, this approach was still dependent upon the use of random repetitions to generate seed points for the graph. As noted in the previous work, less than 8% of these random repetitions resulted in converged trajectories.[4] This ultimately affects the ability of the random reps method to confidently approach the global optima for a given vehicle case in a reasonable amount of time. With only an 8% pass rate, tens or hundreds of thousands of reps may be needed to be confident that the best

repetition is at least close to the global optima. However, typical design study time constraints require that fewer repetitions be attempted, sometimes resulting in seed points that have only a handful of successful completions. If a small number of successful repetitions are used to generate a seed point, the graph method may inherit some inaccuracies as it chains DOE cases from the non-global-optimal seed points. This creates inherent noise in the graph data, which can limit the accuracy of the resulting surrogate models.

In order to address this issue a new approach was developed and tested, which leverages machine learning to inform the selection of the repetitions. The primary goal of this work was to help improve the pass rate of the repetitions method, which is used to develop seed points for running a graph-theory based DOE for surrogate model fitting. The hypothesis was that an improvement in case pass rate would increase the number of successful repetitions in a given amount of time, thus increasing the probability that the global optima had been found for the given seed point. As noted above, the resolution of these seed points is very important to reducing the noise of the response within the graph runs.

The following section will elaborate upon the challenges associated with noise in the graph-based DOE runs and its effect on surrogate model goodness of fit. This section will also give a brief background of Support Vector Machines. In Section III the initially proposed approach is outlined in more detail. Section IV then covers the experiments performed to test the new approach and compare its utility to the current methods. The conclusion of this section will also illustrate the differences in POST surrogate model goodness-of-fit when applying the SVM-based approach.

II. Background

A great amount of previous work has been completed to enable automated POST execution for the purpose of launch vehicle trade space exploration. First, heuristics were developed based upon subject matter expert knowledge of the POST tool.[3] These heuristics were then implemented in an automated fashion in order to mimic the POST execution process of an experienced analyst. This approach ultimately allowed for the execution of a design of experiments (DOE) for broad trade space exploration. However, the initial capability was still limited in terms of case closure rate and surrogate model accuracy.

To improve the case closure rate and surrogate model accuracy, additional work was done to implement a graph theory based approach for executing a DOE.[4] The goal of this work was to further mimic the typical actions of a POST analyst by passing converged u-vectors between similar vehicle cases. To do this, a graph created from the DOE points was used to identify POST cases that could be chained together. The most highly connected points on the graph were executed first in order to "seed" the graph. These seed points were then chained along the edges of the graph until no additional cases could be converged.

The graph based approach ultimately provided improvement in DOE case closure rates and surrogate model accuracy. However, it was still subject to issues associated with resolution of the optimal u-vectors for the seed points. Similar to the initial heuristic-based approach, the graph method relied upon the use of randomly selected repetitions on each vehicle case to find an optimal u-vector. As noted in the previous work, the random repetitions approach gives between a 5 and 8 % pass rate.[3,4] With such a low pass rate, a very large number of repetitions must be tried in order to receive a modest number of converged cases. This number of converged cases ultimately translates into a confidence level that the global optima has been found for the given vehicle case. If this global has not been found and a sub-optimal case is subsequently used to seed the DOE graph, the graph-based cases will be negatively impacted.

In order to illustrate this issue, consider the example shown in Figure 1. The example gives three vehicle cases with different settings of a parameter plotted on the x-axis. For each case, two trials were run using 100 and 1000 random initial u-vector guesses. These trials resulted in between 1 and 15 converged trajectories for each vehicle case. The y-axis represents the optimized value, or "*optval*" from POST for these converged trajectories. Typically *optval* is set up as a maximum payload delivered or a minimum liftoff gross mass depending upon the type of vehicle being analyzed.

For this example, consider the case where a minimum *optval* is desired. As shown in the figure, a large delta exists between the minimum optimum value of the 100 repetition and 1000 repetition trials for cases A and C. For case B, the 1000 repetitions trial resulted in 14 more converged trajectories than the 100 repetitions trial, but the optimum values came out to be nearly identical.

The primary issue is illustrated by the fit lines that pass through the minimum optimum value for each repetitions trial. The dashed line is representative of a surrogate model fit using the minimum values from the 100 repetition trial. Due to the inability of this trial to successfully resolve the minimum for cases A

and C a parabolic curve is fit. This curve suggests that a setting of B for the x-axis parameter should give a minimum optimum value. However, the 1000 repetition trial has better resolved the minimum values for A and C, which clearly shows that setting A should be the minimum. Ultimately, chaining the u-vectors for case A and C from the 100 repetitions trial through the graph DOE would introduce noise into the overall output.

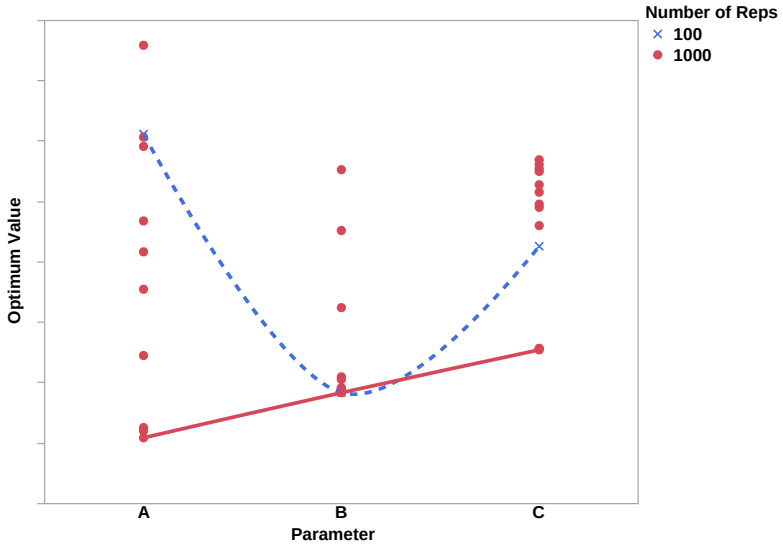


Figure 1: Random repetition noise example

III. Approach

The challenge presented in the previous section ultimately highlights the need for quicker resolution of the seed point u-vectors for the graph DOE method. The random repetitions method can resolve the minima/maxima of the seed points, however, this resolution comes at a high computing cost. The root cause behind the high computing cost is the minute size of the u-vector space that produces converged trajectories for a given vehicle. By randomly sampling the space, a large number of trials are needed just to find this space, let alone find the small area within it that gives an optimal vehicle.

In order to address this shortcoming the composition of the u-vector space with respect to the POST output was investigated. Typically, only the repetitions that provide a converged trajectory are kept during a design study. However, plotting the u-vector space while including failed repetitions provided key observations that lead to the development of the classifier based approach.

An example of the composition of u-vector space with respect to repetition completion reason is given in Figure 2. The figure gives a heat map view of various ending conditions for POST repetitions plotted against two u-vector components. During a typical study only the converged repetitions in green would be kept and the others would be immediately discarded at runtime due to storage constraints. The "unusable" repetitions are those that cause the trajectory to fail before each required phase of its ascent has been satisfied.

This illustration gives an excellent look at the small amount of space where converged trajectories can be found. Only around 15% of the space returns u-vectors that are "usable", meaning POST begins its optimization routine, minimizing the error of the propagated trajectory. Note that this fraction can change depending upon the vehicle parameters, which have an effect on the size, shape, number, and location of the usable region(s). The vehicle used to create Figure 2 for example was relatively easy to fly and returned a usable fraction of 16%. More difficult vehicles from the same dataset ranged between 8 and 10%.

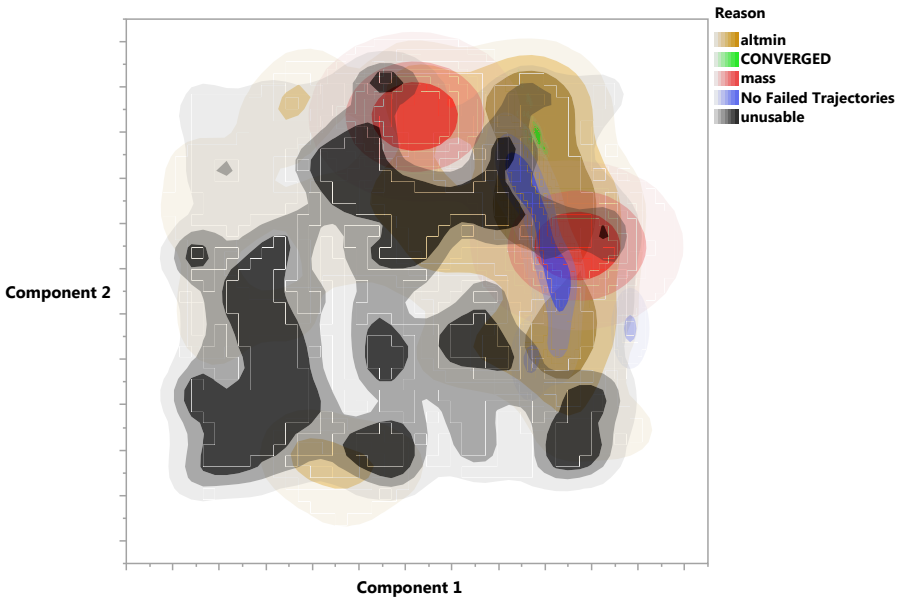


Figure 2: End reason for POST repetitions

Considering the fact that over 85% of the u-vector space will produce unusable POST repetitions, it is unsurprising that the case pass rate in previous work was found to be around 5% to 8%. Following this observation, the focus of this work shifted to developing an approach for identifying the usable u-vector space for a given vehicle prior to repetition submission. If this region were known a priori, then the repetitions could be filtered before being executed, which could drastically increase the pass rate of the cases that actually get sent to POST. At this point it was hypothesized that a classifier could be fit to an existing set of data and then be used as a filter to estimate where the usable region may lie for a new vehicle case.

It is important to note that adaptive sampling techniques were also considered, which could narrow down the random u-vector guesses to focus on the usable region. However, during simple testing it was found that these techniques typically require a large amount of time to successfully resolve this region. In addition, the adaptiveness of the technique would need to be applied to each vehicle individually, which presented additional challenges for data storage and information handling during runtime. In comparison, fitting a classifier would allow for a single technique to generalize across the vehicle space, allowing for model reuse during studies with similar bounds.

III.A. SVM Implementation

There are many different techniques available for performing machine learning for the purpose of classification including artificial neural networks, decision trees, kth-nearest neighbors, and support vector machines among others. Although a full comparison of various machine learning techniques would be of interest for this problem, support vector machines were ultimately chosen for application. The reader is referred to the literature for a more detailed discussion and comparison of these techniques.[6,7]

The selection of SVM was made primarily due to the ability of the model to deal with highly multidimensional data where a nonlinear relationship exists between the input and output features.[7] In addition, SVM models are very flexible and have been successfully applied to many different problems such as text recognition, face detection, time series prediction, and others.[8,9] It is noted in the literature that the generalization of the SVM model either matches or significantly outperforms alternative methods for most example problems.[8]

Implementation of Support Vector Machines were completed in the Python package scikit-learn, which provides utilities for fitting a variety of models for classification and regression. Scikit-learn utilizes the *libsvm* library to perform fitting of various support vector classifiers. A brief description of the formulation within *libsvm* is given below and the reader is referred to the literature for more detail.[10]

The support vector classification formulation within *libsvm* solves the following optimization problem given training vectors $x_i, i = 1, ..., l$ and an indicator vector y:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2} w^T w \; + \; C \sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \; i = 1, ..., l \end{aligned}$$

In addition, the dual problem is solved:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \alpha^T Q \alpha \; - \; e^T \alpha \\ \text{subject to} \quad & y^T \alpha = 0, \\ & 0 \leq \alpha_i \leq C, \; i = 1, ..., l \end{aligned}$$

where the matrix Q contains the kernel function $K(x_i, x_j)$:

$$Q_{ij} \equiv y_i y_j K(x_i, x_j)$$

and the radial basis function kernel in *libsvm* is defined as:

$$K(x_i, x_j) = e^{-\gamma ||x_i - x_j||^2}$$

It is important to note that the above optimization problems are designed for binary classification and require extension for use in multi-class problems. Within *libsvm*, the one-against-one approach is utilized when training is done on multi-class data.[10] In this approach $k(k-1)/2$ classifiers are created, with each one training on data from two classes. A voting strategy is then used between all of the classifiers to identify the most likely class for a new data point. Although other techniques exist for multi-class SVM classification, the one-against-one approach was shown in the literature to be competitive across multiple test datasets.[11]

Since the primary goal of this work was to test and demonstrate the usage of SVM on a POST problem, the default settings for support vector classification were used within scikit-learn. The default settings included a radial basis function kernel and balanced class weights. The primary parameters that were adjusted for model fitting were the penalty parameter of the error term, C, and the kernel coefficient, γ.

III.B. Proposed Approach

The proposed classifier-based approach requires multiple steps for successful implementation. First, a training set is needed to enable the fitting of an SVM. This set is generated via execution of the random repetitions method on a small subset of the vehicle cases within the POST DOE. After executing some number of repetitions for each of the vehicle cases, the classifier is fit to the dataset. Following satisfactory SVM fitting, the model can then be used to execute additional POST repetitions in order to more fully resolve the seed point optima. Once resolved, the seed points are utilized to carry out the remainder of the DOE and surrogate modeling process as discussed in the previous work.[4]

Due to the classifier's reliance upon random repetitions data for training, a comparison of the runtime of the two methods was of great interest. In order to justify spending time upfront fitting the classifier it was also important to compare the resolution of the optimal cases between the full random reps and classifier methods. Multiple tests were set up and conducted to address these comparisons. Section IV discusses the setup and results of these experiments.

IV. Experimentation

In order to test the SVM-based repetitions method, three primary experiments were carried out. The first experiment aimed to identify the appropriate settings for the initial repetitions used to fit the SVM. This test identified the settings that balanced repetition run-time with SVM goodness-of-fit. The second test then utilized the best fits identified in the first experiment to compare the case pass rate from the SVM to that of the random repetitions method. Finally, a test was carried out to compare the goodness-of-fit of surrogate models using cases from the SVM and random repetitions approaches.

All three of the experiments were based upon a single example problem using a two-stage Mars ascent vehicle. A two-stage MAV was chosen due to the simplicity of the POST input deck, which allows for very rapid case execution. This ultimately helped reduce the computational and time requirements for executing the experiments. In Section IV.A the input variables and ranges are given for the example problem DOE. Relevant fixed parameters are also noted in this section. Experiment 1 is covered in Section IV.B, while the results of experiment 2 are detailed in Section IV.C. The final surrogate modeling test is discussed in Section IV.D.

IV.A. Design of Experiments

A single design of experiments was used for all tests of the SVM-based approach. This DOE was developed to capture relevant trades for a two-stage to orbit Mars ascent vehicle similar to previous studies.[12,13] As shown in Table 1, the primary inputs include propulsion parameters, liftoff location, payload, destination, and vehicle propellant mass fractions. For the purposes of this work the propellant mass fractions were calculated as the stage propellant mass divided by the stage propellant mass plus the stage burnout mass, or: $m_p/(m_p + m_{bo})$. Note the destination was defined using an excess delta-v parameter. This represents the delta-v required to reach the destination orbit from the intermediate orbit and was set as the variable *dvmarr* in the POST deck.

Relevant fixed parameters for the POST cases are listed in Table 2. For all cases the intermediate orbit was fixed as a 54 by 135 nautical mile orbit. The stage engine settings were also fixed, with the first stage always utilizing three engines and the second stage using only one. The thrust and I_{sp} values for the engines on both stages were set to the same values based upon the ranges shown in Table 1. During ascent the first stage was allowed to throttle if necessary and the minimum throttle level was assumed to be 20% thrust for each engine.

The POST input deck used common assumptions for the Martian environment. The gravitational model for Mars comes from the 1997 Astronomical Almanac.[14] The atmospheric data was taken from the Space and Planetary Environment Criteria Guidelines for Use in Space Vehicle Development.[15] The reference area for MAV aerodynamics and the speed of sound input were taken from reference [12].

The dependent variables in the POST input deck defined the target orbit using geocentric radius, flight path angle, and inertial velocity. Independent variables for optimization consisted of seven pitch rates, launch azimuth, payload mass, and throttle level for a throttling event during ascent. The first pitch event occurs at five seconds into the ascent, followed by a gravity turn at forty-five seconds. Between the end of the gravity turn and insertion into the initial orbit, six additional pitch events occur. The gravity turn is ended by the first stage reaching zero propellant. At this point, the first stage is jettisoned and the second stage begins thrusting at its optimized throttle level. The final Main Engine Cut-Off (MECO) for the two-stage vehicle occurs when the MAV achieves its initial orbit of 54 by 135 nautical miles. Simultaneously, the MAV is constrained to have a remaining delta-v equal to the excess amount required to reach the final target orbit. The optimized variable in the deck or "*optval*" was the gross liftoff mass of the vehicle, which was desired to be at a minimum for the given inputs.

From the inputs and ranges in Table 1, a primary 2,000 case DOE was generated, which is typical size for a POST design study. The seed points used in Section IV.B and IV.C were then identified as the top 100 highest degree nodes within the graph connecting the primary 2,000 cases. Note that the highest degree nodes are simply those that have the highest number of connections to their neighboring cases. The neighboring cases are defined by the connections within the graph or "minimally spanning tree". A more detailed description of the graph definition is given in the previous work.[4]

In addition to the primary DOE, a secondary DOE of 500 cases was generated. This DOE was executed separate from the experiments to serve as a test set for the SVM goodness-of-fit. It was executed using the random repetitions method, with 1000 random u-vector guesses per vehicle case and thus served as a

baseline measurement of the case pass rate and time required to execute the current approach. A summary of information from the test set can be seen in Table 3, which includes total number of repetitions, total runtime, the average number of converged repetitions per vehicle case, and the overall repetition pass rate. As shown in the table, the pass rate is significantly lower than previous work, which was between 5 and 8 %. The lower case pass rate for the MAV cases is due to the use of propellant mass fractions within the deck as opposed to vehicle masses. The PMF values are significantly more sensitive, leading to more difficulty in producing acceptable initial u-vector guesses.

All of the POST runs for the secondary DOE and experiments were run using the multiPOST tool as discussed in the previous work.[3] The runs were spread across multiple workstation computers allowing for the use of 80 simultaneous analysis threads. For each experiment the same number of threads were utilized in order to allow for a direct comparison between the techniques.

Table 1: Two-stage Mars Ascent Vehicle DOE Inputs and Ranges

Variable	**Range**
Liftoff Latitude	0 - 60 deg.
Excess Delta-v	4,000 - 6,000 ft/s
Thrust per engine	15,000 - 45,000 lbf
Engine I_{sp}	300 - 460 sec
Payload	4,000 - 11,000 lbs
Stage 1 PMF	0.75 - 0.95
Stage 2 PMF	0.65 - 0.9

Table 2: Fixed Parameters

Parameter	**Value**
Initial insertion orbit	54 nm by 135 nm
Stage 1 # of engines	3
Stage 2 # of engines	1
Minimum engine throttle	20%

Table 3: Test Set Information

Total Repetitions	**Total Wall Time**	**Average Converged**	**Pass Rate**
500,000	6:04:09	5	0.5%

IV.B. Experiment 1: SVM Fitting

The purpose of the first experiment was to test the fitting of Support Vector Machines to POST data. As discussed in Section III, the runtime and ability of the approach to successfully resolve the optima are tied to the execution settings of the initial random reps as well as the SVM model parameters. Therefore, the first experiment was designed to answer the following question: what are the execution settings that yield a balance between highest SVM accuracy and minimum required run time?

The accuracy of the model can be thought of in terms of its ability to successfully classify the region in which cases will converge. This ability maps directly to multiple aspects of the training set of data, which is generated via a reduced run of random repetitions. First, the training set must contain enough converged cases to enable the classification of the convergence region. In addition, the overall size and composition of the dataset will have an impact on not only the model accuracy, but the required runtime of the repetitions and the fit time of the SVM.

On the POST execution side of the proposed approach, the user has control over multiple parameters that will affect the runtime as well as the number of converged cases. The most obvious parameter is the number of random repetitions that will constitute the SVM training set. As this number increases the number of converged cases within the set will increase, however, the runtime required to produce the set as well as the time required to fit the model will increase drastically. An additional consideration for POST execution settings are the number of iterations POST is allowed to execute. Higher settings for POST iterations may allow more cases to reach convergence, however, there is a time penalty for utilizing too many iterations. From these considerations, the initial number of random repetitions and the number of POST iterations were selected as parameters to be tested in experiment 1.

Secondary to the POST execution parameters are the SVM fit parameters, which will affect fit accuracy and time. As discussed in Section III.A the primary parameters for user control of SVM fitting in scikit-learn were C and γ. To simplify the execution of experiment 1 a simple grid search was implemented across these parameters in order to find the best fit. This approach was similar to that suggested in the *libsvm* library literature.[10]

To perform experiment 1, combinations of the POST execution parameters were used to run sets of random repetitions for a set of 100 unique vehicle cases. These cases were then used as separate training sets for fitting an SVM. For each training set a grid search was performed on the model parameters in order to find the best fit.

Table 4 gives a summary of the POST runs for the various combinations of initial repetitions and iterations. Note the number of repetitions are per case, therefore with 100 vehicle cases and a setting of 100 repetitions, 10,000 total repetitions were executed. The converged column notes the overall number of cases that converged out of the entire set of repetitions. For example, only 5 out of a total of 10,000 repetitions successfully converged for trial number 1.

In Table 4 both total wall time and POST time are given. POST time represents the total amount of time spent running POST. Since the POST executions are done in parallel the actual wall time of these sets was much smaller. Typically, the total wall time of each set was between 20 and 30% of the quoted POST time. If more analysis threads were used, the total wall time would continue to decrease, while the POST time would be nearly identical for each set. This is due to the fact that the total number of repetitions sent to POST will be the same.

As illustrated in the table, the runtime was not impacted by the POST iterations as much as it was by the number of repetitions. Each step in number of iterations shows a slight increase in runtime, however, increasing the number of repetitions nearly doubles the runtime. Within each set of fixed number of repetitions it is also clear that the number of converged cases depends upon the number of POST iterations. Doubling the number of POST iterations doubled the number of converged cases in a majority of the test sets.

Table 4: Summary of POST results for experiment 1

Repetitions	POST Iterations	Converged	Wall Time	POST Time
100	5	5	5:51	1:21:28
100	10	16	5:54	1:29:17
100	25	27	5:58	1:34:39
100	50	54	6:30	1:52:28
250	5	13	15:36	3:20:58
250	10	26	15:37	3:42:59
250	25	71	15:42	4:04:21
250	50	118	16:04	4:47:30
500	5	31	31:10	6:34:04
500	10	49	31:14	7:20:28
500	25	144	31:15	8:05:40
500	50	260	31:35	9:09:07
1000	5	36	57:50	13:03:58
1000	10	76	57:52	14:23:15
1000	25	303	58:00	16:04:33
1000	50	522	58:13	18:51:10

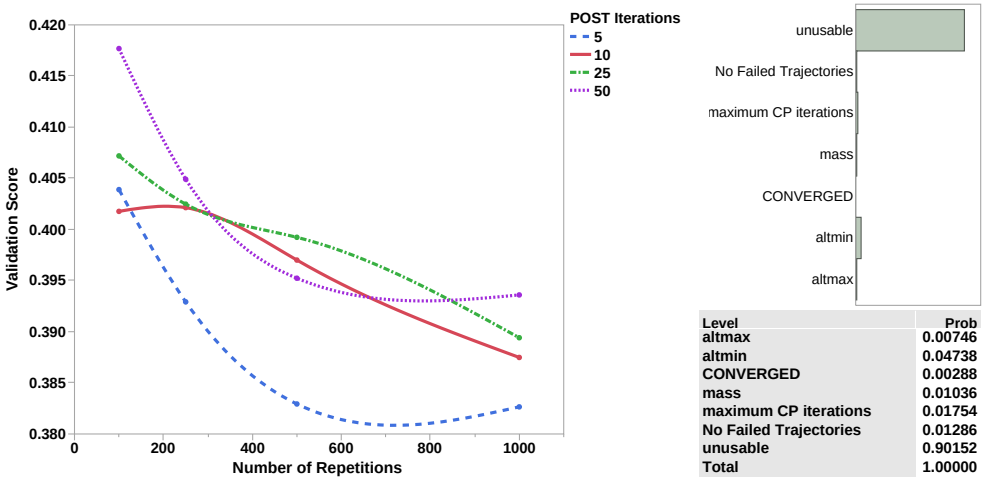
Following completion of the training data, a simple grid search was implemented to fit an SVM for each set. The initial fit trial was done using all the repetitions from each set, which had major implications in terms of required fit time and accuracy. Figure 3 depicts the issues encountered during initial fitting. This figure illustrates the validation score, which is the percentage of correct predictions made against the test set in Table 3. Additionally, an example distribution is given of "reason", which is the end condition of the POST cases.

As shown in Figure 3a, the validation score decreased with increasing number of repetitions. This is counter-intuitive as it is expected that more repetitions, and thus more converged cases would improve the SVM fit. However, this trend is caused due to the distribution of the reason, shown in Figure 3b. This distribution illustrates the lopsided nature of the POST end conditions with 90% of the cases coming back as unusable. In terms of SVM fitting, the large portion of unusable cases was overshadowing the other categories, thus causing the goodness-of-fit to fall as the total number of repetitions increased.

In order to mitigate these fitting issues, it was necessary to reduce the number of unusable cases within each dataset. A simple test was done to examine the effects of reducing the unusable subset on the fit accuracy and time required. For this test, the unusable subset was reduced by eliminating a random percentage of the repetitions ranging from 0 to 99. Figure 4 illustrates the effects of the reductions on fit time and validation score. As clearly shown in the figure, reducing the number of unusable cases linearly reduces the amount of time required to fit the model. In addition, the validation score tends to increase linearly except at the extremes, where a sharp drop off occurs.

Considering these trends it was decided that the unusable subsets be reduced by 85% for the second round of fit testing. This percentage was chosen because it is near the very maximum of the validation score seen in Figure 4, but is not too close as to risk a potential drop off in score. An 85% reduction in the datasets typically brought the number of unusable cases down to about 2-3 times the number of the next most common reason. For example, in Figure 3b there were around 2,400 altmin occurrences. After reduction of the unusable subset, around 6,000 unusable occurrences remained.

In addition to reducing the unusable subsets, the SVM fit parameters C and γ were fixed based upon the initial fit test. For all subsets the grid search yielded the same parameters, which maximized the C value and kept the γ value near the low end of the search range. Additional testing is needed, but it is expected that the fit parameters are problem dependent, meaning they could be held at constant values for a given POST input deck and DOE.



(a) Number of repetitions vs. validation score (b) Reason distribution

Figure 3: SVM fit issues when using all repetitions for each training set

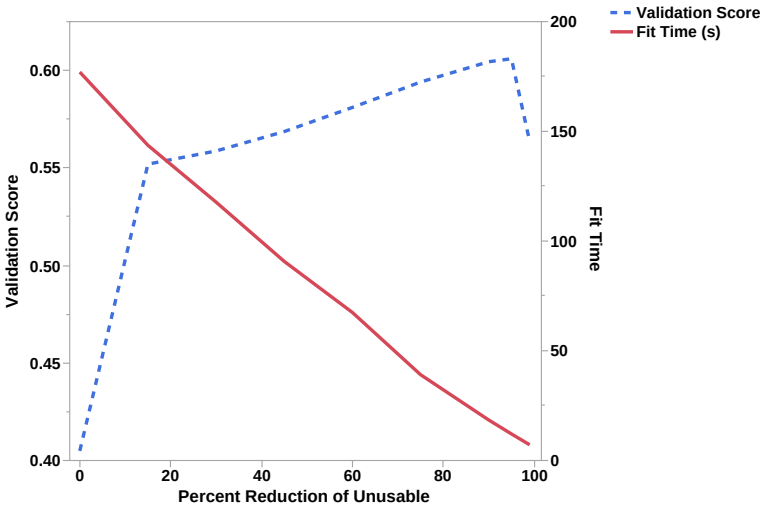
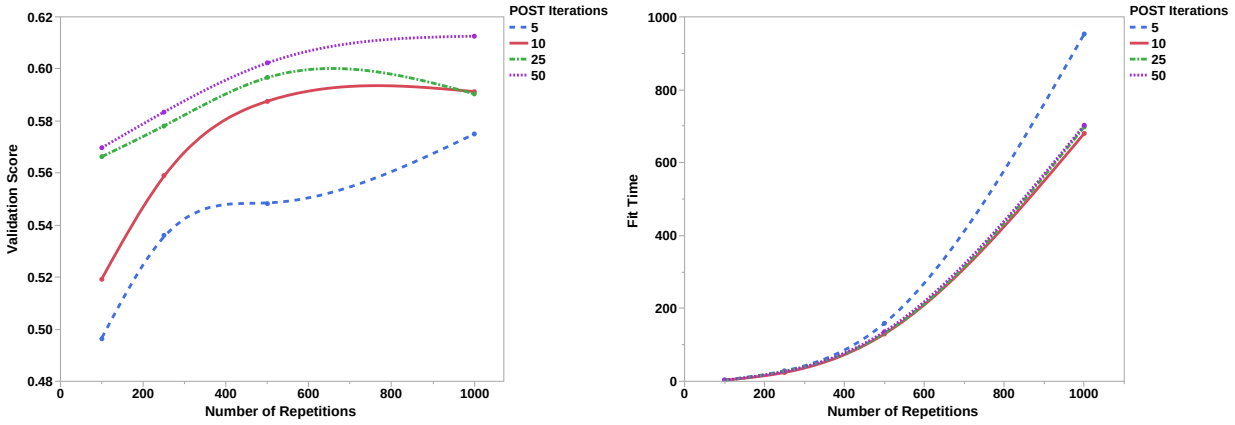


Figure 4: Effects of reducing the unusable subset on validation score and fit time

The results of the second round of SVM fitting can be seen in Figure 5. This figure illustrates a much more intuitive trend, where the validation score tends to increase with increasing number of repetitions and POST iterations. Note that the 10 and 25 iterations cases actually decrease slightly in terms of validation score with increasing number of repetitions, while the 50 iterations case seems to flatten out. This would suggest a point of diminishing returns exists as the number of repetitions are increased. This is especially true when considering fit time, which increases exponentially with number of repetitions in Figure 5b. For example, at 1000 repetitions and 50 iterations the fit time was 700 seconds. Reducing to 250 repetitions and 50 iterations brings the fit time down to just 27 seconds, while reducing the validation score by less than 3%.

Due to the drastic increase in fit time two models were selected to move forward into experiment 2. First, the 1000 repetition, 50 iteration model was selected as it performed best in terms of validation score. This model was expected to provide the best POST case pass rate. A second model was selected in order to further explore the differences in validation score and fit time. The second model was the 250 repetition, 50 iteration case. As noted above this model has a slightly lower validation score, however, its fit time was 25 times shorter than the best model. In the following section the two selected classifiers will be referenced by the number of repetitions per case that were used to generate the training set. Thus the best fit in terms of validation score will be the "1000 Rep" classifier and the second model will be the "250 Rep" classifier.



(a) SVM score versus number of repetitions

(b) SVM fit time (s) versus number of repetitions

Figure 5: SVM fit information after 85% reduction in unusable subsets

IV.C. Experiment 2: Run-time Comparison

The second step within the proposed SVM method is to execute additional repetitions through POST using the classifier as a filter. As discussed above, two classifiers were selected in experiment 1 for additional testing. Experiment 2 was therefore developed to evaluate the overall runtime requirements of the SVM method versus the status quo of random repetitions. The primary goal of the experiment was to determine whether the additional classifier creation steps were worth the upfront cost in terms of time, or if simply running random repetitions from the beginning would give satisfactory results.

Considering the primary goal, the experiment was set up to mimic a typical seed point generation run in preparation for a graph based DOE. As mentioned earlier, the seed points represent the highest degree nodes within a graph based DOE. Typically the top 100 highest degree nodes are used. Based upon the authors' experience, for each of the seed points it is desired that at least 50 successful repetitions are completed. This helps ensure that the optima found for each point is at least very near the global. For these reasons experiment 2 utilized a 100 case set, requiring 50 successful repetitions per case.

Within multiPOST the two classifiers were used to filter initial u-vector guesses. During runtime random u-vectors are generated and passed to the classifier. If the classifier identifies a potential repetition as converged the repetition is put into the queue to be run through POST. This process continued until 50 successful repetitions were received for a given case. At that point the case was removed from the queue, allowing the analysis to focus on cases that had not reached the requirement yet. The SVM method was allowed to pass a maximum of 1000 repetitions to POST per vehicle case.

Execution of the random repetitions method was essentially the same as the classifier, but without the use of the SVM in the loop. While executing random repetitions, all the initial u-vector guesses were passed to POST. In similar fashion, cases were no longer operated on when 50 successful repetitions were complete. As is customary with the random repetitions method, it was allowed to submit as many repetitions as needed for each vehicle case. The execution was then ended manually when multiPOST was no longer returning converged cases.

Table 5 gives a summary of the results for experiment 2. The "Cases with Data" column represents the number of cases out of the original 100 that received at least 1 successful repetition. The "Cases Completed" column then represents the number of cases that actually reached 50 or more successful repetitions. The "Repetitions" column gives the the total number of repetitions that were actually run through POST. Note the wall and POST times in the table are only for the execution of the repetitions within this experiment. The total time from start to finish for the SVM-based method will be discussed later in this section.

As illustrated by Table 5 each of the trials were very close in terms of number of cases completed, however, their submitted repetitions and times are significantly different. First, the random repetitions trial illustrates the brute force nature of the method, which required over 2 million POST runs. Thus, the POST time for this trial was nearly ten times higher than the others. However, due to parallel processing the total wall time for these runs came in between the two classifiers.

Figure 6 gives a closer look at the pass rates of each trial, which further explains the trends in Table 5.

Note that the time elapsed has been condensed in the figure and only includes the POST time required to close the completed cases. Also note that the large reduction in pass rate for both classifiers maps to the requirement of 50 successful repetitions per case. During the early part of the execution, the easiest cases are completed, resulting in peak pass rates. However, as these cases reach the repetitions requirement they are removed from the execution and the classifier begins to work on the more difficult cases. The difficult cases require more submissions to POST and thus have a negative impact on the pass rate.

As shown in Figure 6, the 1000 rep classifier peaks at just over a 16% pass rate, while the 250 rep SVM is around 8%. This difference explains why the 1000 rep classifier was able to complete more cases with 20,000 fewer repetitions. The figure also extenuates the improvement in pass rate of the SVM trials over the random repetitions method. The random method peaks around 0.5%, which means the 250 rep and 1000 rep classifiers show a 16 and 32x improvement in pass rate, respectively.

When comparing the classifiers the 1000 rep SVM was clearly more efficient in terms of pass rate, completing 9 more cases while submitting 20,000 fewer repetitions. However, the wall time illustrates the added complexity of this SVM model. The major difference in this case was the amount of time required to evaluate the SVM and find a repetition that was predicted to converge. Although the POST time for the 1000 rep SVM was 14 hours shorter, the increase in evaluation time of the classifier caused the wall time to quadruple as compared to the 250 rep SVM.

Table 5: Summary of POST results for experiment 2

Set	Cases with Data	Cases Completed	Repetitions	Wall Time	POST Time
1000 Rep SVM	87	70	51,777	49:10:00	30:25:53
250 Rep SVM	88	61	70,777	11:39:08	44:31:32
Random	92	69	2,044,765	23:16:34	327:20:18

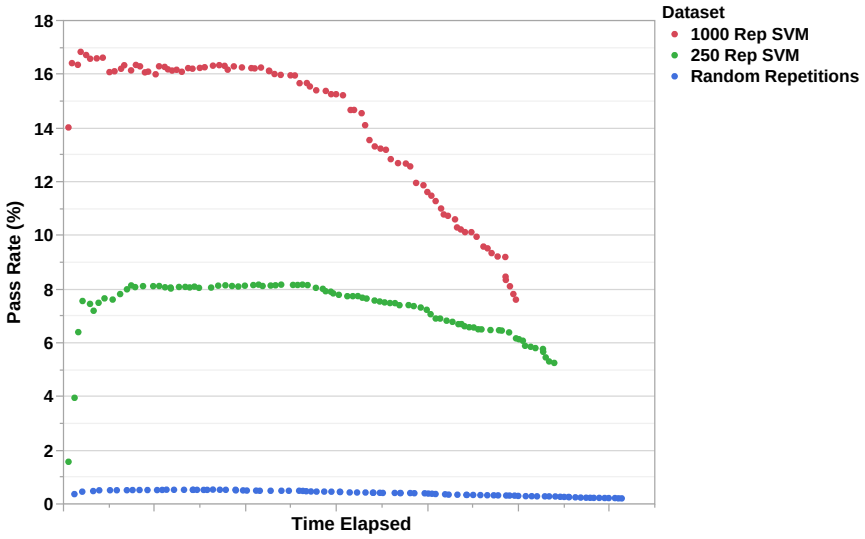


Figure 6: Pass rate versus POST time elapsed for SVM and random trials

At first glance, Table 5 suggests that the random repetitions method can complete more cases than the 250 rep SVM and the same number of cases as the 1000 rep SVM in half the time. However, the ultimate goal of the seed point generation method is to increase the probability that the global optima has been found for each case. Therefore, when looking deeper at the trial results, the SVM methods become more attractive.

An additional comparison of the trial data is given in Table 6. This table gives a comparison between the three trials with respect to the *optval* received for each case. For the example problem a minimum *optval* was desired, therefore the minimum values were compared across the trial cases. In the left-most column of the table, the name of each trial is given along with the total number of cases that received data in parentheses. Moving across the rows, the upper triangular gives the total number of cases that performed better in terms of *optval* than the trial listed in each column. For example, the 1000 rep SVM had 55

cases that performed better than the 250 rep classifier and 74 cases that performed better than the random method. The lower triangular shows the reciprocal numbers but noted in percentages. For example, the random method completed 92 cases, but was bested by the 250 rep classifier on 76 of them. Therefore, only 16 of the random method's 92, or 17.4% of the cases were better than the 250 rep classifier.

The comparison in Table 6 illustrates the downsides of applying the random repetitions method. Although this method was able to complete nearly the same amount of cases as the 1000 rep SVM in half the wall time, only around 20% of them performed better. Comparing to the 250 rep classifier is even worse for the random method, which only bested the SVM in 17.4% of the cases while requiring twice as much wall time. In other words, both classifiers were significantly better at resolving the optimal values for the seed points.

Another interesting point to note is the total number of successful repetitions completed by each set. The total successful reps were 4,209 for the random method, 3,921 for 1000 rep SVM, and 3,716 for the 250 rep SVM. This ultimately backs up the conclusion that the SVM method is better at resolving the optimal values for the seed points. Not only did it produce better results for 80% of the cases, but it did so while completing almost 10% fewer repetitions than the random method.

Table 6: Comparison of minimum *optval* across three trials

	1000 Rep SVM	**250 Rep SVM**	**Random**
1000 Rep SVM (87)	0	55	74
250 Rep SVM (88)	37.5%	0	76
Random (92)	19.5%	17.4%	0

After considering Table 6 the 1000 rep classifier looks to be the best choice in terms of resolution of the seed point optima because it beats the 250 rep SVM in 37.5% of the cases. However, 10 of the 55 cases that beat the 250 rep SVM were within 10 pounds or less. Considering these cases as nearly identical, the 250 rep SVM is either better or within 10 pounds of the 1000 rep classifier for just under 50% of the cases. In addition, the total runtime of each trial supports the selection of the 250 rep classifier as the best for the given problem.

Table 7 shows a comparison of the total time required for executing the SVM trials. The training data column refers to the wall time for generating the training set for the SVM, while the repetitions column gives the time required to complete the seed points as in Table 5. The model fitting column refers to the total time taken to run through the grid search of SVM parameters to find the best fit. If the best parameters were known a priori these times would be the same as noted in Section IV.B; around 700 seconds for the 1000 rep SVM and 27 seconds for the 250 rep SVM. Typically, the parameters will not be known, which is why the full grid search time is quoted in the table. Finally, the total time required for all steps is quoted in the last column.

As illustrated by Table 7, the 250 rep SVM produced its results in around one fifth of the time. A majority of this extra time was spent executing repetitions through the 1000 rep classifier. Although its pass rate was double that of the 250 rep classifier, Table 7 depicts what was discussed earlier. The more complex classifier ultimately required longer to evaluate. Even if the training data execution was further parallelized, no time could be saved during repetitions execution.

The results of experiment 2 clearly show the advantage of applying the SVM-based method over simple random repetitions. Both classifiers were able to better resolve the seed point optima and the 250 rep SVM was able to do so in half the amount of time. The data from experiment 2 also illustrated an important point regarding the SVM method. Although more accurate SVM fits can be achieved by using more training data, time is ultimately lost during the repetitions phase, where the more complex model slows down the analysis. Due to this observation, future studies should seek a balance between fit accuracy and complexity. From the models tested in experiment 2, the 250 rep classifier is the most desirable because it struck the best balance.

Table 7: Comparison of time required to complete each SVM trial

Classifier	**Training Data**	**Model Fitting**	**Repetitions**	**Total Wall Time**
1000 Rep SVM	58:13	24:07	49:10:00	50:20:20
250 Rep SVM	16:04	1:05	11:39:08	11:56:17

IV.D. Surrogate Modeling

The primary purpose of implementing the SVM-based approach was to improve the seed point generation for executing a graph-based DOE through POST. As discussed in Section II, successful resolution of the seed point optima will have a profound impact upon the data gathered from the graph DOE. Ultimately, this data will impact the goodness-of-fit of the POST surrogate model that is created at the end of the process.

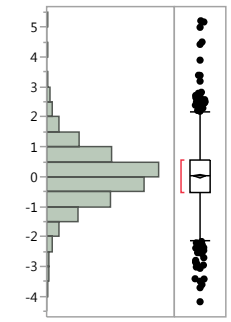
Experiment 2 illustrated the advantage of using the SVM method, which produced better optimal values than the random method for 80% of the seed points. In addition, the best classifier was able to do so in just under 12 hours. The final test for this work therefore aimed at evaluating the goodness-of-fit of a surrogate model created using the SVM and random repetitions methods.

The data used to compare the surrogate models came from actual trades carried out by ACO for SLS. They were selected for use in this work because they represent the actual application of the SVM method within an ACO study. The DOE's for these trades were very similar to those laid out in the previous work and focused on upper stage parameters.[4, 16, 17]

Both sets in this case were run over similar time periods, each using the same parallel computing resources with 80 available threads. The random repetitions data was executed over the course of two and a half days, while the SVM data ran for just under three days. Over this time the SVM based seeds made over 7,700 successful connections on the graph, while the random seeds produced just 1,700.

Following the execution of the graph method, surrogate model fits were created from both sets using the process outlined in the previous work.[16] The percent error distributions of the resulting surrogates can be seen in Figure 7 below. As depicted in the figure, the surrogate from the random repetitions method had a maximum error of 5.2% with a standard deviation around one. In comparison, the maximum error for the SVM-based run is less than half and the standard deviation is about one fifth of the random repetitions run.

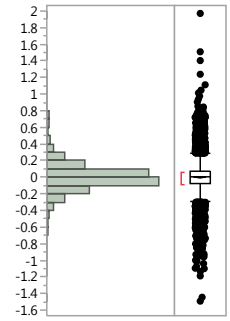
As noted in previous work, the typical rule of thumb for acceptance of these distributions is a mean as close to zero as possible and a standard deviation of less than one. In the authors' experience with the random repetitions method, the standard deviation is typically always close to one. When the SVM-based fit in Figure 7b was first analyzed it was immediately labeled the "best POST surrogate" seen to date by the authors. After recent experience in implementing the graph method with SVM-based seed points, it is clear that the resulting data is much less noisy, allowing for significantly better surrogate models.



Quantiles		
100.0%	maximum	5.2067
99.5%		3.8115
97.5%		2.1497
90.0%		1.1465
75.0%	quartile	0.5465
50.0%	median	0.0257
25.0%	quartile	-0.529
10.0%		-1.133
2.5%		-1.989
0.5%		-3.353
0.0%	minimum	-4.176

Summary Statistics	
Mean	0.02168
Std Dev	1.00621
Std Err Mean	0.02868
Upper 95% Mean	0.07795
Lower 95% Mean	-0.0346

(a) Random repetitions method



Quantiles		
100.0%	maximum	1.9693
99.5%		0.7203
97.5%		0.4031
90.0%		0.194
75.0%	quartile	0.0698
50.0%	median	-0.002
25.0%	quartile	-0.077
10.0%		-0.18
2.5%		-0.353
0.5%		-0.645
0.0%	minimum	-1.49

Summary Statistics	
Mean	0.00056
Std Dev	0.18427
Std Err Mean	0.00185
Upper 95% Mean	0.00419
Lower 95% Mean	-0.0031

(b) SVM classifier method

Figure 7: Comparison of surrogate model percent error using both seed point generation methods

V. Conclusion

The primary goal of this work was to extend the capabilities of the graph-based DOE method for trade space exploration using POST. Although the graph method itself provided a boost in runtime, its reliance on the random repetitions method for generating seed points left it susceptible to noise. The noise in seed point data ultimately caused a reduction in the goodness-of-fit of the final POST surrogate model.

After observing the characteristics of the u-vector space it was hypothesized that a classifier could be used to help identify the usable region for a new vehicle case. If successful, it was expected that this classifier could drastically improve the case pass rate during seed point generation. Simultaneously it was expected to provide an improvement in the ability to resolve the optima for each case.

A basic strategy was then developed to utilize support vector machines to classify the u-vector space. Multiple tests were run using various POST and SVM parameters in order to test the goodness-of-fit of the classifiers. In Section IV.B it was shown that the classifiers could indeed be fit to the POST repetitions data. Although the scores of these models left some to be desired, they still drastically improved the case pass rate.

Section IV.C performed tests to identify the case pass rates when using two different classifiers. These classifiers were selected in order to find a balance between time required to fit the model and goodness-of-fit of the model. Within this section it was shown that the classifiers increased the case pass rate 16 and 32 times that of the random repetitions method. Overall, this improvement in pass rate allowed the classifiers to better resolve the optimal value for over 80% of the seed points.

Finally, a test set was given as an example in Section IV.D of the improvement in final surrogate model accuracy when using the SVM method. As illustrated in that section, the surrogate model from the SVM based run had half the maximum error of the random repetitions trial with a significantly smaller standard deviation of the error distribution. The reduction in error was primarily due to the ability of the SVM to better resolve the seed point optima, thus reducing the noise injected into the graph-based DOE.

The examples in Sections IV.C and IV.D illustrated the utility of the SVM approach for improving the graph-based DOE method. This improvement is tied directly to the drastic increase in case pass rate through POST. Ultimately, the use SVM will provide a much more reliable method for resolving the seed points for a new DOE. In doing so, the resulting surrogate models will be much more accurate and will provide the ACO team with a far more powerful tool for trade space exploration.

References

[1]Nelson, D., *Qualitative and Quantitative Assessment of Optimal Trajectories by Implicit Simulation (OTIS) and Program to Optimize Simulated Trajectories (POST)*, Master's thesis, Georgia Institute of Technology, April 2001.

[2]Steffens, M. J., Edwards, S. J., and Mavris, D. N., "Capturing the Global Feasible Design Space for Launch Vehicle Ascent Trajectories," *AIAA SciTech*, Kissimmee, FL, January 2015.

[3]Dees, P. D., Zwack, M. R., Steffens, M., Edwards, S., Diaz, M. J., and Holt, J. B., "An Expert System-Driven Method for Parametric Trajectory Optimization During Conceptual Design," AIAA SPACE 2015 Conference & Exposition, Pasadena, CA, 2015.

[4]Dees, P. D., Zwack, M. R., Steffens, M., and Edwards, S., "Augmenting Conceptual Design Trajectory Tradespace Exploration with Graph Theory," AIAA SPACE 2016 Conference & Exposition, Long Beach, CA, September 2016.

[5]Brauer, G., Cornick, D., Habeger, A., Peterson, F., and Stevenson, R., "Program to Optimize Simulated Trajectories (POST) Volume 1: Formulation Manual," *NASA-CR-132689*, April 1975.

[6]Kiang, M., "A comparative assessment of classification methods," *Decision Support Systems*, Vol. 35, 2003, pp. 441 – 454.

[7]Kotsiantis, S., "Supervised Machine Learning: A Review of Classification Techniques," *Informatica*, Vol. 31, 2007, pp. 249 – 268.

[8]Burges, C., "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, Vol. 2, 1998, pp. 121 – 167.

[9]Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., and Scholkopf, B., "Support vector machines," *IEEE Intelligent Systems and their applications*, Vol. 13, No. 4, 1998, pp. 18–28.

[10]Chang, C.-C. and Lin, C.-J., "LIBSVM: a library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, Vol. 2, No. 3, 2011, pp. 27.

[11]Hsu, C.-W. and Lin, C.-J., "A comparison of methods for multiclass support vector machines," *IEEE transactions on Neural Networks*, Vol. 13, No. 2, 2002, pp. 415–425.

[12]Polsgrove, T., Thomas, D., Sutherlin, S., Stephens, W., and Rucker, M., "Mars Ascent Vehicle Design for Human Exploration," *AIAA SPACE Conference and Exposition*, Pasadena, CA, August 2015.

[13]Polsgrove, T., Thomas, D., Stephens, W., Collins, T., Rucker, M., Gernhardt, M., Zwack, M., and Dees, P. D., "Human Mars Ascent Vehicle Configuration and Performance Sensitivities," *IEEE Aerospace Conference*, Big Sky, MT, March 2017.

[14]U.S. Naval Observatory, *The Astonomical Almanac*, U.S. Government Printing Office, 1997.

[15]Smith, R. and West, G., *Space and Planetary Environment Criteria Guidelines for Use in Space Vehicle Development*, Vol. 1, National Aeronautics and Space Administration, 1982.

[16]Zwack, M. R. and Dees, P. D., "Application of Design of Experiments and Surrogate Modeling within the NASA Advanced Concepts Office, Earth-to-Orbit Design Process," 2016.

[17]Dees, P. D. and Zwack, M. R., "Automation of POST Cases via External Optimizer and "Artificial p2" Calculation," 2017.