AIAA SciTech Forum
12-16 January 2026, Orlando, FL
AIAA SCITECH 2026 Forum

10.2514/6.2026-2025

Improving the efficiency of multi-fidelity approaches for transfer learning with application to Hall thrusters

Jacqueline M. Thatcher*
Sandia National Laboratories, Livermore, CA, USA

Thomas Marks†
University of Michigan, Ann Arbor, MI, USA

Gianluca Geraci‡, Owen N. Davis§, Michael S. Eldred¶
Sandia National Laboratories, Albuquerque, NM, USA

Alex A. Gorodetsky‖
University of Michigan, Ann Arbor, MI, USA

In this paper, we extend Multi-fidelity Networks (MFNets) for transfer learning in the context of engineering system performance prediction. MFNets build a surrogate representation of multiple information sources and encode the relationship among these sources as a directed acyclic graph. Each information source corresponds to a node on the graph, and relationships between the sources are represented as edges. In this paper we introduce a greedy procedure that adaptively learns the best polynomial degree for each node. Moreover, the algorithm uses the polynomial degree adaptation to drive the allocation of resources needed at each node (i.e., the number of model realizations). Our goal is to deploy this strategy to leverage existing data from systems at well-explored missions and operative conditions (i.e., when data are abundant or can be obtained easily) to predict the performance of a novel system in a different operative scenario where less data is available. As an exemplar application, we target the prediction of in-flight Hall thruster performance using both ground test data and data from other Hall thruster systems. Specifically, we consider a hypothetical interplanetary mission, where we seek to estimate the performance (thrust) of the H9 Hall thruster given uncertain electron transport and neutral gas velocity. A limited dataset of simulated H9 thrusts is augmented with a larger dataset from the Stationary Plasma Thruster (SPT-100) system. To construct the SPT-100 dataset, we perform simulations that are representative of both ground testing and interplanetary flight and include multiple fidelity levels for each of these configurations. This allows us to test various scenarios to determine if the adaptive algorithm can successfully leverage information from lower-cost systems to predict the performance of the H9 Hall thruster.

I. Introduction

Aerospace applications are often amenable to modular system design [1–3]. This approach is attractive because each component or sub-system can be designed independently and then integrated in various configurations depending on mission requirements. Despite the flexibility provided by this engineering design framework, estimating performance of a novel configuration in new operating scenarios remains challenging because access to directly corresponding data can be extremely limited. In [4], we proposed an approach to address this challenge by extracting information from *legacy systems* using transfer learning (TL) techniques. In particular, we leveraged Multi-fidelity Networks (MFNets) [5–7] to extend multi-fidelity surrogate techniques for this task.

*Senior Member of Technical Staff, Computational Data Science, Livermore, CA, USA.
†Assistant Research Scientist, Department of Aerospace Engineering, Ann Arbor, MI, USA.
‡Principal Member of Technical Staff, Optimization and Uncertainty Quantification, Albuquerque, NM, USA, AIAA Senior Member.
§Postdoctoral Appointee, Optimization and Uncertainty Quantification, Albuquerque, NM, USA.
¶Distinguished Member of Technical Staff, Optimization and Uncertainty Quantification, Albuquerque, NM, USA, AIAA Associate Fellow.
‖Associate Professor, Department of Aerospace Engineering, Ann Arbor, MI, USA.

Within MFNets, each system (and potentially more fine-grained aspects such as an individual component), is modeled as a node in a direct acyclic graph (DAG) representing the relationship between these different information sources. In this context, the (directed) edges are used to transfer information and define conditional dependencies among the models or components. In [4], we demonstrated this TL approach on a modular system derived from the "Sandia National Laboratories - Structural dynamics application" benchmark problem originally proposed in [8]. Specifically, we showed that this approach has the potential to aid in the design of new systems by lowering the amount of data required to accurately estimate performance variations over design or uncertain parameter ranges. This study identified several opportunities for algorithmic improvement. In particular, we observed that the effectiveness of the approach when data is sparse is hampered by an inability to separately adapt the polynomial representation of each node. In this contribution, we propose a greedy approach that targets this limitation, and we demonstrate how this improves the TL performance by reducing the data required to obtain an MFNets surrogate with a given accuracy. Moreover, by linking the polynomial representation to the data requirement at each node, the algorithm is also able to control the resource investment by requiring the acquisition of additional data only when and where relevant for the accuracy of the target system.

The remainder of the paper is organized as follows. In Section II, we introduce the system of interest, Hall-effect ion thrusters, that we use to showcase algorithm performance for realistic modeling situations. In particular, we focus on predicting the thrust of an H9 system [9] in interplanetary flight, given limited direct data and a larger availability of data from a Stationary Plasma Thruster (SPT-100) [10] system, with different configurations and flight missions. In Section III, we briefly describe the MFNets approach that we use to build surrogates from data of varying quality, and in Section III.A, we introduce the greedy adaptation algorithm that is the main algorithmic advancement proposed in this work. Numerical results are discussed in Section IV for both a verification test case and three TL scenarios, while conclusions and future work are discussed in Section V.

II. Motivation: Hall thruster modeling and design

In this section, we describe an exemplar application of Hall thruster modeling that we will use as a basis to test our approaches. Hall thrusters are a type of spacecraft electric propulsion device commonly applied for spacecraft station-keeping and recently applied to interplanetary exploration on NASA's Psyche mission [11]. Unlike conventional chemical rockets, electric propulsion systems like Hall thrusters produce a small amount of thrust at high efficiency. This property enables long duration missions which may require a large change in velocity (delta-V) to facilitate multiple complex maneuvers, at a much reduced fuel budget compared to chemical systems.

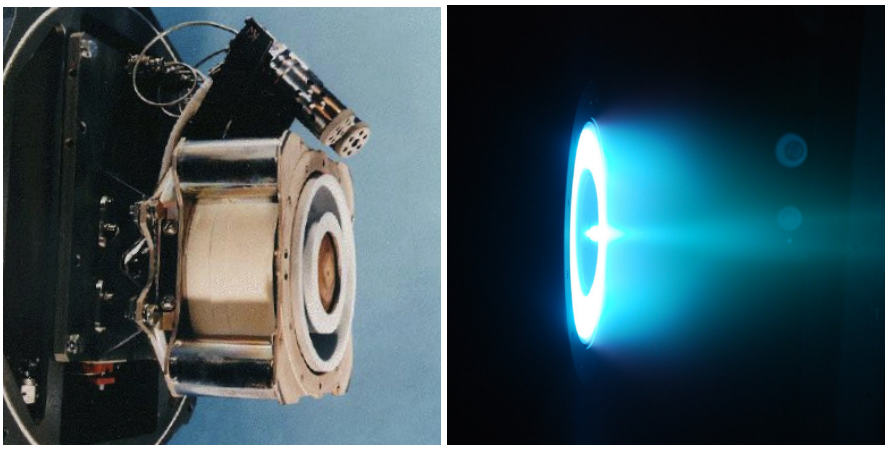
Despite their extensive flight heritage, models of Hall thrusters are not predictive at present, i.e., the thrust and plasma properties cannot be accurately predicted from the geometry and operating conditions alone. In these devices, small-scale plasma instabilities drive enhanced cross-field "anomalous" electron transport [12]. In turn, the magnitude and scaling transport governs the dynamics of plasma acceleration, ultimately determining the thrust and performance of the thruster. Models must instead be calibrated to match experimental data, a time-consuming and often manual process [13]. This calibration must be repeated for each new thruster and even for each new operating condition.

In this paper, we explore whether we can employ TL techniques to map the performance of a Hall thruster from existing designs and operating points to the performance at a new configuration and condition. In principle, this could reduce efforts in recalibration by better leveraging previous learning, transferred from related in-family systems. If successful, this reduction in the amount of data and time required to qualify new Hall thruster designs at new operating points could reduce the time to flight and increase the ability of designers to explore the problem space.

A. Analysis scenarios

In this work, we simulate the SPT-100 [10] and H9 [9] Hall effect thrusters, shown in Fig. 1a and Fig. 1b, respectively. We consider the scenario where we want to estimate the thrust of H9 in an interplanetary mission using limited data from both an interplanetary and ground level H9 system along with a larger dataset for the SPT-100 system in both operational conditions. These operating points (*interplanetary* and *ground*) are distinguished by differences in discharge voltage, mass flow rate, background pressure, and magnetic field strength.

The H9's operating condition for which we want to predict the system's performance is an *interplanetary* operating condition, with discharge voltage of 600 V, background pressure of 1×10^{-8} Torr, a flow rate of 16.3 mg/s, and the magnetic field set to the thruster's "nominal" value. These conditions derive from the 600 V 15 A condition of Ref. [16]. We also consider a *ground* test condition which corresponds to a discharge voltage of 300 V, background pressure of



(a) (b)

Fig. 1 (a) The SPT-100 Hall thruster. Image from Ref. [14]. (b) The H9 Hall thruster operating on xenon. Photograph taken in the the University of Michigan's Large Vacuum Test Facility and reproduced from Ref. [15].

1×10^{-5} Torr, a mass flow rate of 14.8 mg/s, and a magnetic field of 1.125× the nominal value.

The first SPT-100 operating point (*ground*) is representative of SPT-100 ground test measurements in the open literature [10], with a discharge voltage of 300 V, mass flow rate of 5 mg/s, peak magnetic field strength of 150 G, and background pressure of 5×10^{-5} Torr; for this configuration we assume we can generate data more easily. The second operating point represents a hypothetical high specific impulse condition for an lightweight unmanned interplanetary mission. Here, the discharge voltage and the background pressure corresponds to the mission's counterpart for H9, namely 600 V and 1×10^{-8} Torr, respectively. On the other hand, the flow rate is 3 mg/s, while the peak field strength is 200 G.

To make the analysis scenario more realistic, we also consider multiple approximations for the numerical simulations of the SPT-100 thruster. Indeed it is reasonable to assume that even if TL can be used to combine data obtained across systems, configurations, and missions, lower-fidelity simulations should be included as an additional source of information. Since many designs require high-fidelity simulations, considering multiple fidelities may be the only viable way to obtain adequate amounts of data. We define four fidelities, each associated with a different number of grid cells and the maximum charge state allowed for the xenon ion. The fidelities are denoted *high* (H), *medium-1* (M1), *medium-2* (M2), and *low* (L), with the specific combinations of charge states and grid cells defined in Table 1. We note that the simultaneous change of both the number of grid cells and charge states prevents us from arranging the models in a strick one-dimensional hierarchy.

Table 1 Fidelity levels defined for the Hall thruster computations.

Name	Number of cells	Max charge state
H	200	Xe^{3+}
M1	100	Xe^{3+}
M2	200	Xe^{2+}
L	100	Xe^{2+}

We present a summary of the available sources of information in Table 2 . Here, we distinguish the *system* (H9 or SPT-100), the *operating point* (interplanetary or ground), the *fidelity* (H, M1, M2, L), and the *dataset size* – a unique combination of system, operating point, and fidelity defines a *model*. We note that the dataset size indicates the equivalent number of simulations that can be generated for each model given a single realization of the *target model*, i.e., H9 flying at the interplanetary operating conditions and modeled with high-fidelity. These dataset sizes do not correspond to the amount of data used to generate the numerical results.

Following [4], we define different analysis scenarios that are inspired by realistic TL analyses. In addition to the TL scenarios in which both H9 and SPT-100 are present (Scenarios 2 and 3), we also present a simpler scenario based only

Table 2 Available models for the Hall thruster TL scenarios. The *interplanetary* mission for H9 corresponds to a discharge voltage of 600 V, background pressure of 1×10^{-8} Torr, a flow rate of 16.3 mg/s and 'nominal' magnetic field. The *ground* condition corresponds to a discharge voltage of 300 V, background pressure of 1×10^{-5} Torr, a mass flow rate of 14.8 mg/s, and a magnetic field of 1.125× the nominal value. For the SPT-100 system, the *interplanetary* mission corresponds to a discharge voltage of 600 V, background pressure of 1×10^{-8} Torr, a flow rate of 3 mg/s, and peak magnetic field strength of 200 G. The *ground* mission corresponds to a discharge voltage of 300 V, background pressure of 5×10^{-5} Torr, a flow rate of 5 mg/s, and peak magnetic field strength of 150 G. The fidelity levels are controlled by changing both the number of computational cells and maximum charge states (see Table 1). The dataset size indicates the amount of data that can be generated for an equivalent cost of 1 data point for the *target* model (H9, at the interplanetary mission with high-fidelity).

System	Mission	Fidelity	Dataset size	Scenario 1	Scenario 2	Scenario 3
H9	Interplanetary	H	1x		X	X
H9	Ground	H	10x		X	
SPT-100	Interplanetary	H	10x	X	X	X
SPT-100	Interplanetary	M1	50x	X		X
SPT-100	Interplanetary	M2	50x	X		X
SPT-100	Interplanetary	L	250x	X		X
SPT-100	Ground	H	100x		X	

on SPT-100 data (Scenario 1). The three scenarios are summarized in both Table 2, where we indicated all the models that belong to a specific scenario.

For our numerical demonstration, we consider three uncertain parameters for the Hall thrusters. Namely, we consider two uncertain coefficients, a_1 and a_2, for the anomalous transport model [17] based on a two-zone Bohm-like model as well as the bulk velocity of neutral gas injected at the anode, u_n. These values follow the parameterization reported in [17]. We additionally adopt the same uniform prior distributions as in [17], with the exception of a_2, which we take to have a lower bound of 2 instead of 10. We report these prior ranges in Table 3. For all other numerical parameters, we selected the median parameter values from [18] for both H9 and SPT-100.

Table 3 Uncertain parameters and the ranges of their prior distributions.

Name	Lower bound	Upper bound	Units
Anomalous transport coeff. a_1	-2.5	-1.0	-
Anomalous transport coeff. a_2	2	100	-
Anode injection neutral velocity u_n	100	500	m/s

Finally, we note that each of our simulations is performed using the 1-D fluid code *HallThruster.jl* [19], with an additional pressure-dependent divergence correction to the thrust computed using the analytical plume model of [20]. Interested readers are referred to [17] and [21] for more detail on these models and how they couple together.

III. Multi-fidelity networks

In this section, we briefly introduce the MFNets [5, 6] framework which models the relationship between surrogates of various information sources through a DAG. In particular, we follow [6] in which the quantity of interest (QoI) for each model is represented by a node and the dependency of a node i on node j is denoted by a directed edge $(j \to i)$.

We consider M nodes/information sources with data represented by tuples $\{(x_k^{(j)}, y_k^{(j)})\}_{j=1}^{n_k}$ for $k = 1 \ldots, M$. Here, n_k is the amount of data, $x_k \in \mathbb{R}^{d_k}$ is the input vector, $y_k \in \mathbb{R}$ is the output of interest, and d_k is the input space dimension. For simplicity we focus here on the case in which $d_k = d$ for all k, i.e., all information sources have the same input dimension. However, the MFNets extension to cases with heterogeneous models is possible and was discussed in [22]. The MFNets framework is general and allows for the use of different choices of the function representing each

node; however, in the present work we restrict the MFNets surrogate to the scale-shift model, i.e.,

$$\hat{f}_k(x) = \sum_{j \in pa(k)} \rho_{jk}(x)\hat{f}_j(x) + \delta_k(x), \tag{1}$$

where, for the shift δ_k and the scaling function ρ_{ij}, we adopt linear-subspace models, i.e.,

$$\rho_{jk} = \Phi_{jk}^T(x)\alpha_{jk}, \quad \text{and} \quad \delta_k(x) = \Psi_k^T(x)\beta_k, \qquad \text{for } k = 1, \ldots, M. \tag{2}$$

We have also indicated with $pa(k)$ all the parents of node k, i.e., all the DAG's nodes whose edges end in the node k. Throughout this work we use a constant (but trainable) scaling parameter ρ_{jk} for the representation of the edge $(j \rightarrow k)$, while we adopt a polynomial chaos expansion (PCE) [23] for the representation of the kth node's shift function with respect to the basis $\Psi_k(x)$. Once the order of the expansion is decided, the coefficients for both the scale and shift can be obtained with various methods. In [4] we adopted a variational inference approach; however, in this work, we focus on the regression case since our objective is to assess and discuss the greedy adaptation.

One challenge in training an MFNet surrogate is setting the degree for the PCE used for each shift function, i.e., δ_k for $k = 1, 2, .., M$. In previous work, e.g., [4], we used the same PCE degree for all the M models. However, this approach ignores potential differences in both cost and complexity of the information sources. For example, it may be more cost-effective to have higher degree PCEs for models that are cheaper to evaluate. An alternative approach is to explore all possible combinations of PCE degrees across the M nodes. However, this quickly becomes infeasible as the number of nodes in the DAG increases. We also note that the optimal PCE degree cannot be discovered for each node in isolation since the degree choice of a node propagates throughout the DAG, from parents to children, and affects the representation of the target node.

In this work we propose a solution for selecting the degree of the shift models that uses a greedy optimization procedure; moreover, we use the greedy procedure to directly guide the investment of computational resources by requiring the satisfaction of a fixed oversampling ratio [24]. Our goal is to find the MFNets surrogate that provides the most accurate approximation of a target model, while leveraging other information sources to minimize cost. The algorithm should identify information sources that do not improve predictions at the target node and, in turn, prevent additional data acquisition from these uninformative sources. Therefore, by design the algorithm does not necessarily find accurate surrogates for all the information sources. We present details on the greedy algorithm in the next section.

A. Adaptive algorithm

In this section, we present details on the MFNets greedy adaptation algorithm. For the adaptation strategy we focus on the degree p_k of the polynomial shift function $\delta_k(x)$ and set the scale functions equal to a constant optimizable value, i.e., $\rho_{jk}(x) = c_{jk}$ for $k = 1, \ldots, M$ and $j \in pa(k)$. Using a constant scale function mitigates an overgrowth of the polynomial degree associated with deeper graphs; in future work, we may explore adapting the scale function as well the shift function. We note that existing work explored similar approaches in the context of multi-fidelity PCE surrogates in the hierarchical context, see, e.g., [25, 26]. Additional greedy approaches were proposed for multi-dimensional hierarchical cases, see, e.g., [27]. However, this is the first extension of a similar strategy to non-hierarchical surrogates based on DAGs.

Each iteration of the MFNets greedy adaptation algorithm is divided into four steps as summarized in Figure 2, and the full algorithm is provided in Algorithm 1. At each iteration, the algorithm selects a specific node, $k = 1, 2, ..., M$, in the graph and either increases or decreases the degree of the corresponding shift function, $\delta_k(x)$. Which node is selected and the degree change depends on a variety of factors, including whether altering the degree leads to improved performance and the associated cost of different model evaluations. In some cases, no viable shift degree changes are identified and an iteration ends without changing the shift functions. At the next iteration, larger degree jumps are allowed in order to increase the candidate pool. Allowing for degree jumps is an essential feature of the algorithm as demonstrated in the example polynomial system we consider in Section IV.A.

Before describing the four steps of the algorithm in detail, we introduce some notation that is used in Algorithm 1 and Figure 2. Additional notation will be provided as each step is described. We use the vector $\boldsymbol{p} = [p_1, p_2, ..., p_M] \in \mathbb{N}^M$ to denote the current polynomial degrees of each shift function and the vector $\Delta\boldsymbol{p} = [\Delta p_1, \Delta p_2, ..., \Delta p_M] \in \mathbb{N}^M$ to represent how much the degree of each shift function can increase when constructing candidate graphs. For example, if $\Delta p_1 = 2$ this implies that we will consider candidates where the degree of the shift function at the first node increases by up to two degrees. The vector $\boldsymbol{n} = [n_1, n_2, ..., n_M] \in \mathbb{N}^M$ is defined such that n_k is the current number of training samples available from the k-th model. Note that as the algorithm proceeds, more data is acquired and the value of

n_k increases. The shorthand $\boldsymbol{z}_{train}$ is used to represent the entire set of training data currently available across the M models. Finally, the integer t represents the target node corresponding to the system we are ultimately trying to approximate. The algorithm is initialized by setting the shift degrees across all nodes to be zero, i.e., $\boldsymbol{p} = \boldsymbol{0}$, and by allowing for only a single degree increase across all the shift functions, i.e., $\Delta\boldsymbol{p} = \boldsymbol{1}$.

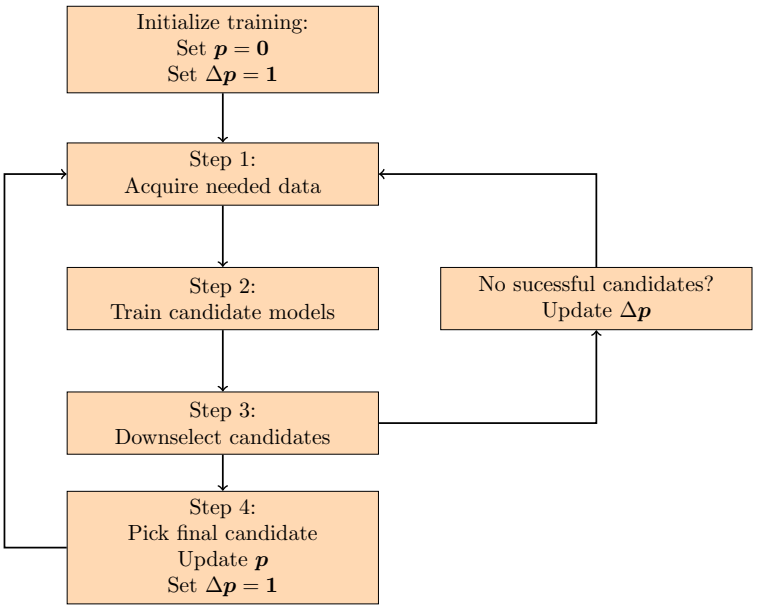


Fig. 2 Overview schematic of MFNets greedy adaptation algorithm. The vector p represents the current shift degrees and the vector Δp represents the maximum increase in degree allowed at each node in the candidate graphs. Each candidate only allows for a degree change at one node in the graph.

1. Step 1: Candidate definition and data acquisition

In the first step of algorithm, we determine the set of candidates that will be considered and acquire additional data as needed to train the proposed candidates. Each candidate is fully defined by a shift degree vector $\boldsymbol{p}^{(j)} \in \mathbb{R}^M$, and we define the set of admissible candidates as $\mathcal{P} = \{\boldsymbol{p}^{(j)}\}$. By design, $\boldsymbol{p}^{(j)}$ only differs from $\boldsymbol{p}$ at one location, as only one shift model is allowed to change at each iteration. The set of possible candidates is dependent on both the current shift degree vector $\boldsymbol{p}$ and the vector $\Delta\boldsymbol{p}$ defining possible degree increases. Note that the value of $\Delta\boldsymbol{p}$ also changes during the algorithm as described in Section III.A.3. At each iteration, we allow for up to three candidates per node. For example, for the k-th node we consider the following candidates

$$\boldsymbol{p}^{(j_i)} = \boldsymbol{p} + (\Delta p_k - i)\boldsymbol{e}_k \quad \text{for} \quad i \in \{0, 1, 2\} \tag{3}$$

where $\boldsymbol{e}_k$ is the unit vector with the k-th element equal to one and all other elements equal to zero. Although more candidates could be considered by increasing the value of i, here we limit ourselves to three candidates to help accelerate the greedy adaptation algorithm; we plan to investigate the best trade-off between efficiency (low number of candidates) and accuracy (large number of candidates) in future studies. Note that if any of these candidates have entrees that are less than zero, they are not considered. If not already included, we additionally consider a candidate such that $\boldsymbol{p}^{(j)} = \boldsymbol{p}$. This serves as the baseline model and allows us to determine if degree changes improve the target system predictions given the current set of available training data.

Based on this set of candidates, we calculate the required number of additional samples needed from each model. The required sample size depends on the number of parameters specific to a given model, i.e., the number of scale and shift parameters at the corresponding node, the number of input variables, and the collocation ratio. Specifically, increasing the polynomial degree will introduce additional parameters, dependent on the input dimension. This number of parameters is then multiplied by the collocation ratio to determine the number of needed samples. The cost of acquiring the latest batch of data for each of the M models is recorded in a cost vector $\boldsymbol{c}_{update} = [c_{update,1}, c_{update,2}, ..., c_{update,M}] \in \mathbb{R}^M$.

Algorithm 1 MFNets Greedy Adaptation ($\boldsymbol{c}$, r_{col}, d, targetNode, degreeJumpMethod)

▷ *Initialize parameters*

$\boldsymbol{p} = \mathbf{0} \in \mathbb{N}^M$ ▷ Initial shift degrees are all zero

$\Delta\boldsymbol{p} = \mathbf{1} \in \mathbb{N}^M$ ▷ Allow degree to increase by one at each node

$\boldsymbol{n} = \mathbf{0} \in \mathbb{N}^M$

$\boldsymbol{z}_{train} = None$

t = targetNode

▷ *Start greedy adaptation loop*

for $i = 0, 1, 2, \ldots$ **do**

▷ *Step 1: Get candidate pool and acquire needed training data*

  $\mathcal{P} = \{\boldsymbol{p}^{(j)}\}$ = getCandidateDegrees($\boldsymbol{p}, \Delta\boldsymbol{p}$)

  $\boldsymbol{n}_{old} = \boldsymbol{n}$

  $\boldsymbol{n}$ = getSampleSize($\mathcal{P}; r_{col}, d$) ▷ Data needed for each model

  $\boldsymbol{c}_{update}$ = updateCost($\boldsymbol{n}, \boldsymbol{n}_{old}; \boldsymbol{c}$) ▷ Cost of acquiring new data

  $\boldsymbol{z}_{train}$ = acquireData($\boldsymbol{n}, \boldsymbol{z}_{train}$)

  ▷ *Step 2: Train set of candidate models*

  for $j = 1, 2, \ldots, |\mathcal{P}|$ **do**

    $\{\mathcal{L}_k^{(j)}\}_{k=1}^M$ = trainMFNets($\boldsymbol{z}_{train}, \boldsymbol{p}^{(j)}$) ▷ Train and record CV loss at each node

    $\ell_j = where(\boldsymbol{p} \neq \boldsymbol{p}^{(j)})$ ▷ Record updated node

    if $\boldsymbol{p}^{(j)} == \boldsymbol{p}$ **then**

      $L_{k,old} = L_k^{(j)}$ for $k = 1, 2, .., M$ ▷ For candidate that doesn't change, record baseline loss

    end if

  end for

  ▷ *Step 3: Downselect candidates.*

  $\mathcal{J}_t = \{j \mid \mathcal{L}_t^{(j)} < L_{t,old}\}$ ▷ Candidates that decrease loss at target node

  $\mathcal{J}_\ell = \{j \mid \mathcal{L}_{\ell_j}^{(j)} < L_{\ell_j,old}\}$ ▷ Candidates that decrease loss at updated node

  $\mathcal{J}_d = \{j \mid \boldsymbol{p}^{(j)} < \boldsymbol{p}\}$ ▷ Candidates with lower degree

  $\mathcal{J} = \mathcal{J}_t \cap (\mathcal{J}_\ell \cup \mathcal{J}_d)$

  if $|\mathcal{J}| = 0$ **then**

    $\Delta\boldsymbol{p}$ = updateDegreeJump($\Delta\boldsymbol{p}$; degreeJumpMethod)

    continue

  end if

  ▷ *Step 4: Pick final candidate and update shift degrees*

  $\hat{j} = \arg\min_{j \in \mathcal{J}} \boldsymbol{c}_{update,\ell_j} \mathcal{L}_t^{(j)}$ ▷ Find candidate that minimizes cost-scaled loss at target

  $\boldsymbol{p} = \boldsymbol{p}^{(\hat{j})}$ ▷ Record shift degrees of selected candidate

  $\Delta\boldsymbol{p} = \mathbf{1} \in \mathbb{R}^M$ ▷ Reset in case changed

end for

return $\boldsymbol{p}$

For instance, if N additional samples are needed from model k then $\boldsymbol{c}_{update,k} = Nc_k$ where c_k is the cost of running a single simulation of model k.

At the first iteration, the sample size is determined using the collocation ratio approach given above but an additional set of samples is added. The size of this set of samples is a hyperparameter the user can tune, but in practice we set this size to be larger than the number of corner points of the input domain, i.e., larger than 2^d. This allows us to include outputs at the corner points in the first set of training data. Note that after the corner points are obtained, all other inputs are randomly sampled from a uniform distribution over the normalized input domain of $[-1, 1]^d$.

2. Step 2: Training candidates

In the second step of the algorithm, we train the complete set of candidate MFNet graphs using a cross-validation approach. That is, for each candidate shift degree vector $\boldsymbol{p}^{(j)}$, we construct the corresponding MFNets graph $\mathcal{G}^{(j)}$ and perform multiple trainings using different folds of the training data in $\boldsymbol{z}_{train}$. In particular, we use three folds and record the mean MSE validation loss at each node across the folds. Specifically, $\mathcal{L}_k^{(j)}$ is set to equal the validation loss for the jth candidate at node k.

We explored both a local and global approach for training the entire MFNet graph. The results presented here are for local training, as described below. However, global training, where all parameters of the graph are learning simultaneously, may have added benefits that require future exploration. For local training, the training is done sequentially, starting at the leaf nodes and proceeding down the graph. When a specific node is being trained, the parameters at all other nodes are frozen and the loss is the mean-squared error (MSE) at that specific node. Specifically, when training the k-th node, we find the parameters $\boldsymbol{\theta}_k$ that minimize the following loss function:

$$\mathcal{L}_k(\boldsymbol{\theta}_k) = \frac{1}{n_{k,train}} \sum_{i=1}^{n_{k,train}} \left(\hat{f}_k(x_{k,i}; \boldsymbol{\theta}_k) - y_{k,i} \right)^2, \tag{4}$$

where $\hat{f}_k$ is the prediction of MFNets at the k-th node and $\boldsymbol{\theta}_k$ contains the scale/shift parameters for node k.

3. Step 3: Downselect candidates and update degree jumps if needed

In the third step, we find a subset of the candidates that satisfy specific criteria. For these criteria we consider the cross-validation error at both the target node and the updated node. By *updated node*, we refer to the node k such that $p_k \neq p_k^{(j)}$. For some candidates, the target node and updated node are equivalent. The criteria are as follows

1) Decreased target node loss *AND*
2) Decreased updated node loss *OR* updated node has smaller degree.

The first criterion ensures that the accuracy improves for the target system. In some cases increasing complexity at an upstream node does not improve target predictions and may lead to unnecessary data acquisition. The second criterion allows us to select candidates with decreased complexity, even if it is detrimental to predictions at the updated node. However, if the node is updated by increasing the degree, we require predictions improve at the updated node. The requirement that the loss decreases at the updated node is primarily needed if we perform global optimization; specifically, it ensures that unwarranted complexity is not added to a parent node in order to improve loss at the target node.

If no candidates satisfy the above criteria, we update the value of $\Delta\boldsymbol{p}$ to increase the set of possible candidates. At the beginning of the algorithm or if a viable candidate was found in the previous iteration, we have that $\Delta\boldsymbol{p} = \mathbf{1} \in \mathbb{R}^M$. When no viable candidates are found, we propose updating $\Delta\boldsymbol{p}$ using one of two possible approaches, which we denote as "increaseAll" and "increaseUsingCost". Note that this corresponds to the "degreeJumpMethod" referenced in Algorithm 1.

In the "increaseAll" approach, when no viable candidates are found, we increase the value of every element of $\Delta\boldsymbol{p}$ by one, i.e.,

$$\Delta\boldsymbol{p} \leftarrow \Delta\boldsymbol{p} + \mathbf{1}. \tag{5}$$

This implies that the subsequent candidate pool will include candidates with an additional degree at all nodes. This update will continue at each iteration until a viable candidate is found. One issue with this approach is it that, to jump the degree at any node, it requires additional data from every model, which can be costly.

The alternative "increaseUsingCost" approach, involves updating $\Delta\boldsymbol{p}$ in a way that is cognizant of the model costs. That is, we would like to prioritize larger degree jumps in models that are cheaper to evaluate. To do this we first

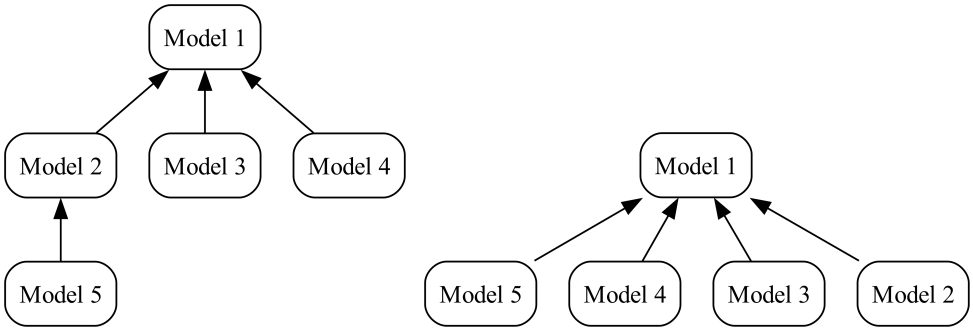


Fig. 3 Graph used for polynomial example where Model 1 represents the target system. (Left) Natural graph and (right) peer graph.

consider the cost of increasing the degree of the shift function for the most expensive model (presumably the target system) by $\Delta p_{max,t} = 1$. Let's denote this cost as $c(\Delta p_{max,t})$. We then calculate the maximum degree by which other models could increase while incurring a cost smaller than $c(\Delta p_{max,t})$. This gives us a degree for each node $\Delta p_{max,i}$. Let $\Delta \boldsymbol{p}_{max}$ containing all of these degrees. Then at each iteration we set

$$\Delta \boldsymbol{p} \leftarrow \Delta \boldsymbol{p} + \mathbf{1}_{\Delta \boldsymbol{p} < \Delta \boldsymbol{p}_{max}}, \tag{6}$$

where $\mathbf{1}_{\Delta \boldsymbol{p} < \Delta \boldsymbol{p}_{max}}$ is 1 where $\Delta \boldsymbol{p} < \Delta \boldsymbol{p}_{max}$ and 0 otherwise. If $\Delta \boldsymbol{p} = \Delta \boldsymbol{p}_{max}$ and still no feasible candidates are found, the algorithm proceeds by setting $\Delta p_{max,t} = 2$ and reevaluating $\Delta \boldsymbol{p}_{max}$.

After $\Delta \boldsymbol{p}$ is updated, the algorithm proceeds to next iteration without changing $\boldsymbol{p}$. This allows for a new candidate pool to be defined based on the updated $\Delta \boldsymbol{p}$ and, in turn, for additional data to be acquired.

4. Step 4: Select final candidates

In the fourth and final step of the algorithm, we select a final candidate from the set of remaining candidates, denoted by $\mathcal{J}$ in Algorithm 1. The final chosen candidate is the one that minimizes a cost-scaled version of the cross-validation loss at the target node. In Algorithm 1, we use $\mathcal{L}_t^{(j)}$ to represent the cross-validation loss at the target node t and ℓ_j to represent the node that is updated for candidate j. The cost-scaled version of this validation loss is given as $c_{update,\ell_j} \mathcal{L}_t^{(j)}$. We then find the candidate j from the set of downselected candidates $\mathcal{J}$ that minimizes this cost-scaled loss and update $\boldsymbol{p}$ accordingly.

IV. Numerical results

In this section we provide some numerical evidence for the performance of our novel greedy adaptation for MFNets. First, we provide a simple polynomial problem that serves as verification case that we can use to discuss the algorithmic features and assess if its behavior is consistent with our intuition and the desired objectives. Afterward, we focus of the three TL scenarios discussed in Section II and defined in Table 2. Throughout, we use a collocation ratio of five, i.e., $r_{col} = 5$, and perform local sequential optimization as describe in Section III.A.2. For each example, we explore how the two degree jump methods, i.e., the two methods for updating $\Delta \boldsymbol{p}$, described in Section III.A.3 impact the results.

A. Example polynomial system

Before considering the Hall thruster system, we first verify the performance of our algorithm on a polynomial system of equations connected by the Natural DAG shown in Figure 3, left panel. The functions are given as follows, where

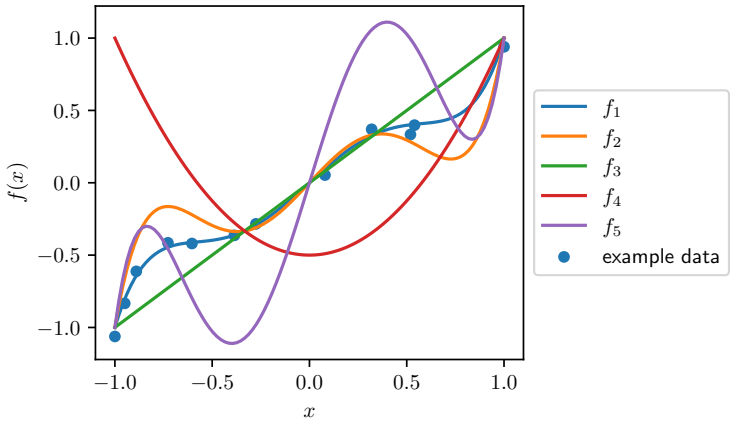


Fig. 4 Visualization of functions for example polynomial system along with example data for the target system. The data is corrupted with noise of $\sigma = 0.05$.

Model k in Figure 3 corresponds to function f_k ($k = 1, \ldots, M$ with $M = 5$),

$$f_1(x) = \frac{1}{2}\left(f_2(x) + f_3(x)\right) \tag{7}$$

$$f_2(x) = \frac{1}{2}\left(f_5(x) + \frac{1}{d}\sum_{i=1}^{d} P_3(x_i)\right) \tag{8}$$

$$f_3(x) = \frac{1}{d}\sum_{i=1}^{d} P_1(x_i) \tag{9}$$

$$f_4(x) = \frac{1}{d}\sum_{i=1}^{d} P_2(x_i) \tag{10}$$

$$f_5(x) = \frac{1}{d}\sum_{i=1}^{d} P_5(x_i) - \frac{1}{d}\sum_{i=1}^{d} P_3(x) + \frac{1}{d}\sum_{i=1}^{d} P_1(x). \tag{11}$$

Here, P_i represents the i-th degree Legendre polynomial and $f_1(x)$ represents the target system. In Figure 4, we provide a visualization of a subset of these functions for a 1-dimensional case along with example data from the target system corrupted by Gaussian noise with $\sigma = 0.05$.

We intentionally include a few features in this example to explore algorithm performance. First we use Legendre polynomials to verify the algorithm is capable of performing degree jumps. That is, since the Legendre polynomials are orthogonal on [-1,1] with respect to the unit weight function, we anticipate that in order to approximate, for example, $f_5(x)$, the algorithm will need to jump from degree 1 to 3 to 5. Secondly, the network is designed so that the target system can leverage data from two parent nodes, specifically nodes with $k = 2$ and $k = 3$. For the node with $k = 2$ there's an additional parent node, $k = 5$, that provides useful information. This allows us to test whether the algorithm can perform well on a graph that's neither purely peer nor hierarchically. Additionally, we can compare the performance of the Natural DAG with a peer counterpart; see Figure 3, right panel. Finally, Node 4 represents a system that does not contain useful information and is included to test the algorithm's ability to recognize this and not acquire additional data from this system.

We train the adaptive MFNets algorithm on an example where the input x is 2-dimensional, i.e., $d = 2$, and the training data is corrupted by Gaussian noise with $\sigma = 0.05$. Additionally, the cost vector for these models is set equal to the following,

$$\boldsymbol{c} = [1.0, 0.1, 0.1, 0.1, 0.01],$$

where c_i represents the cost for a single evaluation of f_i. We set the initial sample size to 20 points plus the additional data required to satisfy the collocation ratio for all models. The algorithm is terminated at a max cost of 125, as this

is the cost for accurately fitting the single fidelity model. We additionally attain 1000 points for testing that are not corrupted by noise.

Figure 5 shows the results of running the adaptive MFNets algorithm for three different data replicates, and Table 4 provides a more in depth analysis of one the data replicates. Figure 5 shows results using both the "increaseAll" and "increaseUsingCost" methods for updating $\Delta\boldsymbol{p}$. For this example, the "increaseUsingCost" approach clearly outperforms the "increaseAll" approach. This is because the "increaseUsingCost" approach avoids increasing the degree of the target system and, thus, the cost of acquiring unnecessary data. For the remainder of this section, we focus on the "increaseUsingCost" results.

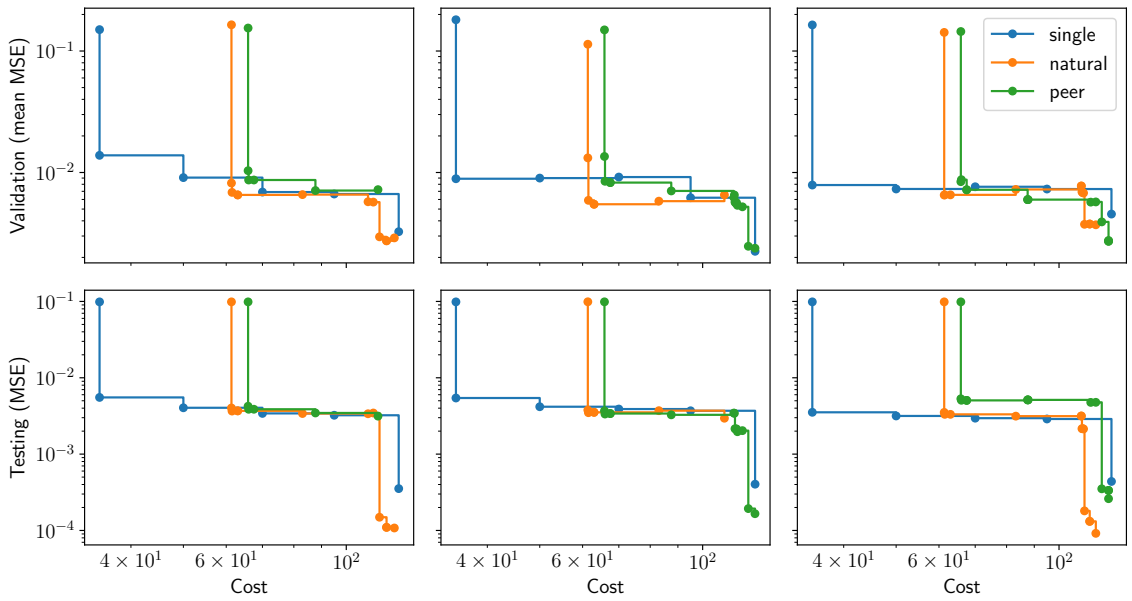
Table 4 Detailed results from running MFNets adaptive refinement on example polynomial systems. Results are shown for each iteration of Algorithm 1 and correspond to the Natural DAG results shown in the second column of Figure 5b.

Iteration	$\Delta\boldsymbol{p}$	$\boldsymbol{n}$	$\boldsymbol{p}$	Test MSE	Total Cost
0			[0,0,0,0,0]		
1	[1,1,1,1,1]	[50,40,35,35,35]	[0,0,0,0,1]	3.77E-03	61.35
2	[1,1,1,1,1]	[50,40,35,35,50]	[0,0,1,0,1]	3.49E-03	61.5
3	[1,1,1,1,1]	[50,40,50,35,50]	[0,0,1,0,0]	3.53E-03	63
4	[1,1,1,1,1]	[50,40,50,35,50]	[0,0,1,0,0]	3.53E-03	63
5	[1,2,2,2,2]	[50,55,70,50,50]	[0,0,1,0,0]	3.53E-03	68
6	[1,3,3,3,3]	[50,75,95,70,70]	[0,0,1,0,0]	3.48E-03	74.7
7	[1,4,4,4,4]	[50,100,125,95,95]	[0,0,1,0,0]	3.50E-03	82.95
8	[1,4,4,4,5]	[50,100,125,95,125]	[0,0,1,0,5]	2.48E-03	83.25
9	[1,1,1,1,1]	[50,100,125,95,160]	[0,0,1,0,5]	2.45E-03	83.6
10	[1,2,2,2,2]	[50,100,125,95,200]	[0,0,1,0,5]	2.47E-03	84
11	[1,3,3,3,3]	[50,100,125,95,245]	[0,3,1,0,5]	3.15E-04	84.45

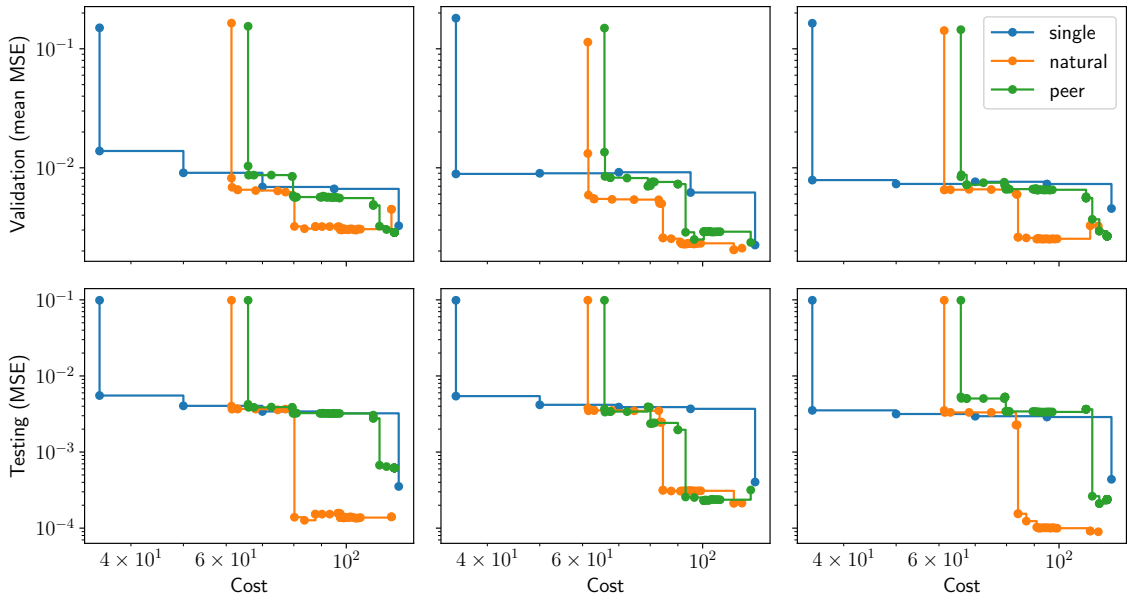
Looking at Table 4, we see for the "increaseUsingCost" approach the algorithm proceeds as expected. This table shows detailed results for each iteration of the greedy adaptation algorithm, including the values of $\boldsymbol{p}$, $\Delta\boldsymbol{p}$, and total training sample size $\boldsymbol{n}$. For reference, we include the testing MSE and total cost, which are also plotted in Figure 5b, second column. Recall that in the vectors $\boldsymbol{p}$, $\Delta\boldsymbol{p}$ and $\boldsymbol{n}$, the i-th element corresponds to Node i, i.e., Model i given by f_i in Equations 7–11. The algorithm first identifies that changing from a constant to linear function at Node 5 provides the most benefit (see $\boldsymbol{p}$ at Iteration 1). This is reasonable given that Model 5 is the cheapest to evaluate. However, later the algorithm finds that a linear shift function at Node 3 is more beneficial (see $\boldsymbol{p}$ at Iteration 2) and changes the shift function at Node 5 back to a constant (see $\boldsymbol{p}$ at Iteration 3). The algorithm then fails to find viable candidates and must gradually update $\Delta\boldsymbol{p}$ from Iteration 5 to Iteration 8, until a viable candidate is found. By Iteration 11, the algorithm settles on the correct set of shift degrees. That is, it finds $\boldsymbol{p} = [0, 3, 1, 0, 5]$, which matches the degrees of the example polynomial system given by Equations 7–11. Note that, although not shown in Table 4, we allow the algorithm to continue past Iteration 11; see results for Natural DAG in Figure 5b, second column. During these subsequent iterations, the algorithm adds additional, unnecessary, degrees but not decrease the final accuracy.

Looking at the results more wholistically in Figure 5b, we see that the MFNets greedy adaptation algorithm improves predictions compared with both the single fidelity case and the algorithm applied to the peer DAG. That is a comparable testing error is achieved at a lower cost. Although this example is constructed in such a way that improvement compared with the single fidelity is expected, the improvement compared to the peer DAG is perhaps more surprising. Investigating this more, we find that in the peer case the algorithm learns high-degree polynomials, i.e., ≥ 5 degrees, at both Node 2 and Node 5 which incurs a higher cost. This cost can be avoided by leveraging the relationship between Node 2 and Node 5 as done in the Natural DAG (see Figure 7).

Finally, we provide a visualization of the training/testing parity plots for a single data replicate in Figure 6. These parity plots compare the prediction of the Natural DAG and single fidelity approach for the first replicate, i.e., first column of Figure 5b, when we set the maximum cost equal to 100. We clearly see that the adaptive algorithm leads to improved performance on the testing data with fewer samples from the target system.



(a) Set Δp using "increaseAll"

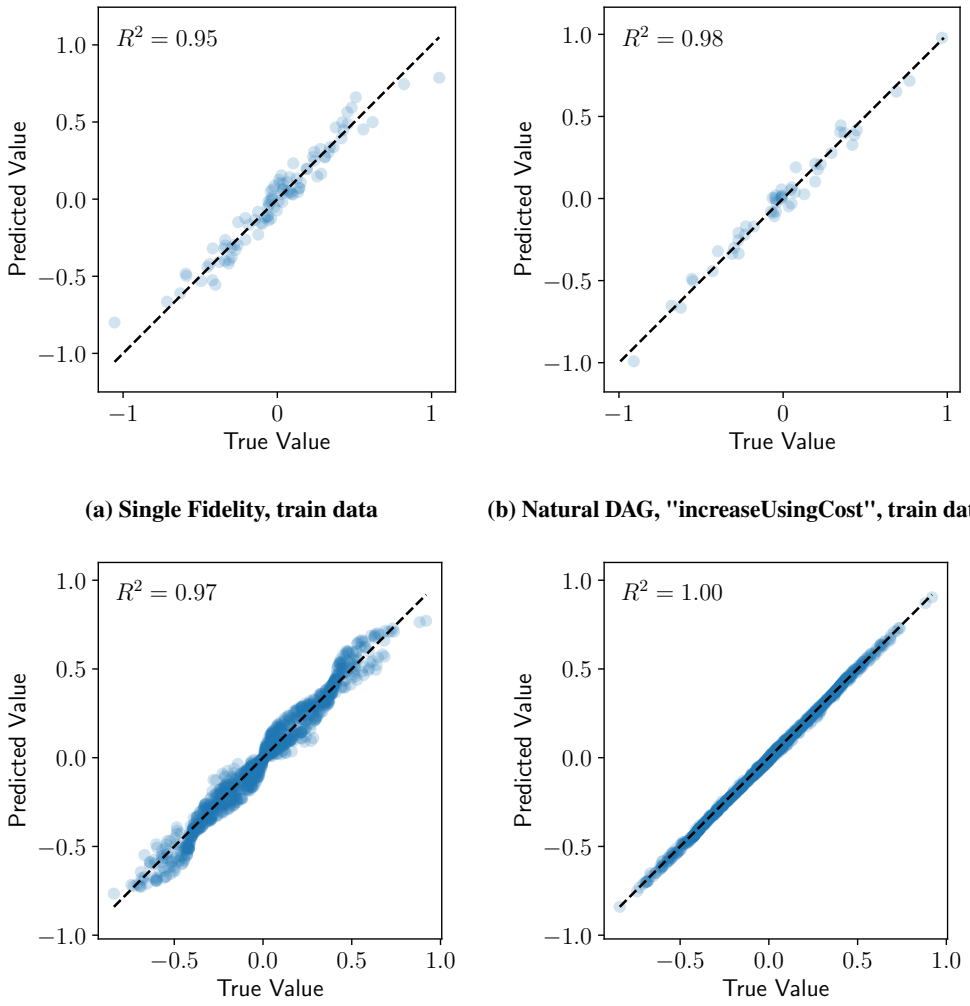


(b) Set Δp using "increaseUsingCost"

Fig. 5 Adaptive MFNets algorithm with Natural DAG leads to improved target predictions for the polynomial example system. Comparing performance of the adaptive algorithm using a Natural and peer DAG with the single fidelity approach for two different methods of increasing Δp. Each column represents training results using different realizations of the data. The top row shows the average MSE for the cross-validation step during training at the target node. The bottom row shows the MSE on a set of testing data at the target node. The testing data is the same for each data replicate.

B. Hall thrusters transfer learning scenarios

Before discussing the numerical results that we obtained for the Hall thrusters, in Figure 7 we show the Natural DAG corresponding to each TL scenario. We note that the Natural DAG is the one that expresses our *a priori* knowledge



(a) Single Fidelity, train data

(b) Natural DAG, "increaseUsingCost", train data

(c) Single Fidelity, test data

(d) Natural DAG, "increaseUsingCost", test data

Fig. 6 Parity plots for both training and testing data for example polynomial system when we consider a maximum cost of 100. Results shown correspond to the first column of Figure 5b. Note the training data, but not the testing data, is corrupted with noise.

regarding the relationships among models.

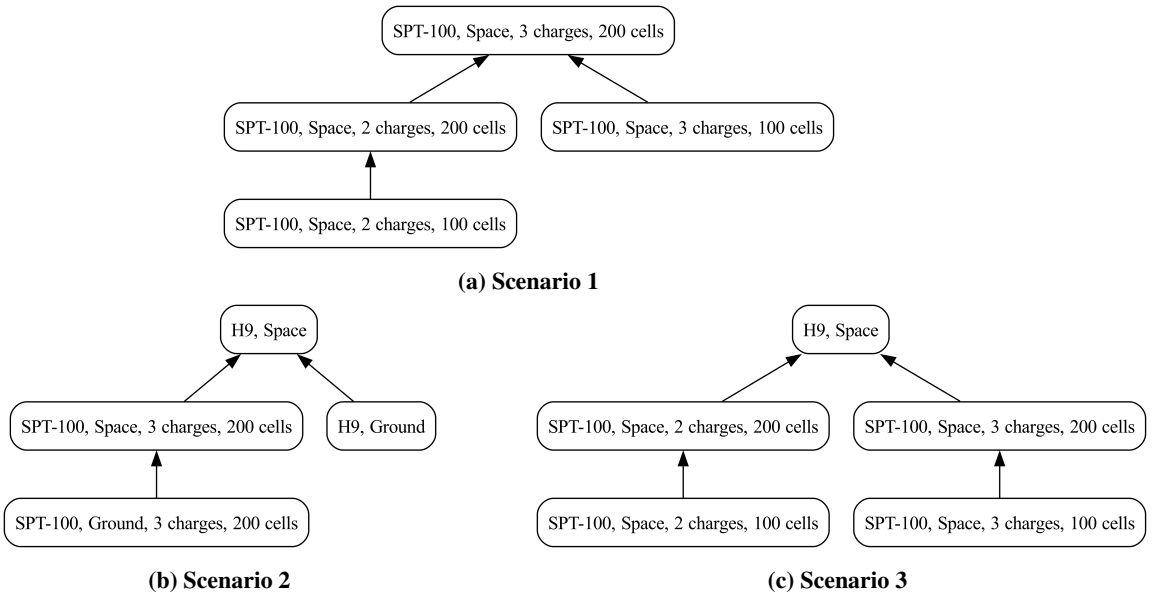
1. Hall thruster: Scenario 1

For Scenario 1 (see Figure 7a) in the Hall thruster system, we ran the MFNets greedy adaptation algorithm using the two methods for updating $\Delta \boldsymbol{p}$ as described in Section III.A.3. Recall that the models all have $d = 3$ inputs and the following costs were used for running the algorithm:

$$\text{cost(SPT-100, Space, 3 charges, 200 cells)} = 1.00 \tag{12}$$
$$\text{cost(SPT-100, Space, 2 charges, 200 cells)} = 0.20 \tag{13}$$
$$\text{cost(SPT-100, Space, 3 charges, 100 cells)} = 0.20 \tag{14}$$
$$\text{cost(SPT-100, Space, 2 charges, 100 cells)} = 0.04. \tag{15}$$

Note that as the fidelity decreases, we assume that the cost of a model decreases by a factor 5.

In Figure 8, we show the results for both approaches for updating $\Delta \boldsymbol{p}$, looking at three different data replicates. We



(a) Scenario 1

(b) Scenario 2

(c) Scenario 3

Fig. 7 Depiction of the Natural DAGs used for the three scenarios considered for the Hall thruster systems.

find that for a specific cost range (from $\sim 100 - 500$), the algorithm leads to improved performance when we compare the results between the Natural DAG and the single fidelity model. Additionally, in this example the two approaches for updating $\Delta \boldsymbol{p}$ lead to similar results. This is because the "increaseAll" approach already does a reasonable job of not sampling from more expensive systems unnecessarily as not many jumps in the shift degree are required.

In Figure 9, we provide a visualization of how much the results are improved at a given cost. Specifically, suppose we are limited to a maximum cost of 300 and consider results shown in the third column of Figure 8. In Figure 9 we provide a parity plot showing the prediction of the testing data at this max cost for both the Natural DAG and single fidelity approach. From this visualization it is clear that the adaptive algorithm with the Natural DAG leads to improved performance at this cost.

2. Hall thruster: Scenario 2

For Scenario 2 (see Figure 7b) in the Hall thruster system, we again ran the MFNets greedy adaptation algorithm using the two methods for updating $\Delta \boldsymbol{p}$ as described in Section III.A.3. Again the input dimension is $d = 3$ and we use the following model costs. Note these are different than the previous section since we set the target system to always have a cost of one.

$$\text{cost(H9, Space)} = 1.00 \tag{16}$$

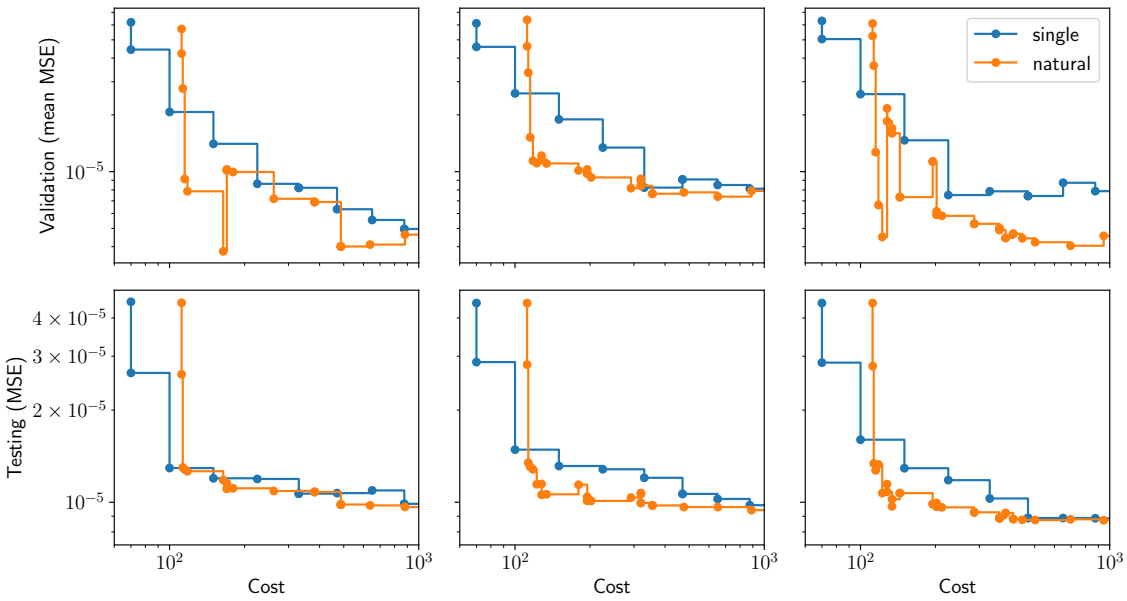
$$\text{cost(H9, Ground)} = 0.10 \tag{17}$$

$$\text{cost(SPT-100, Space, 3 charges, 200 cells)} = 0.10 \tag{18}$$

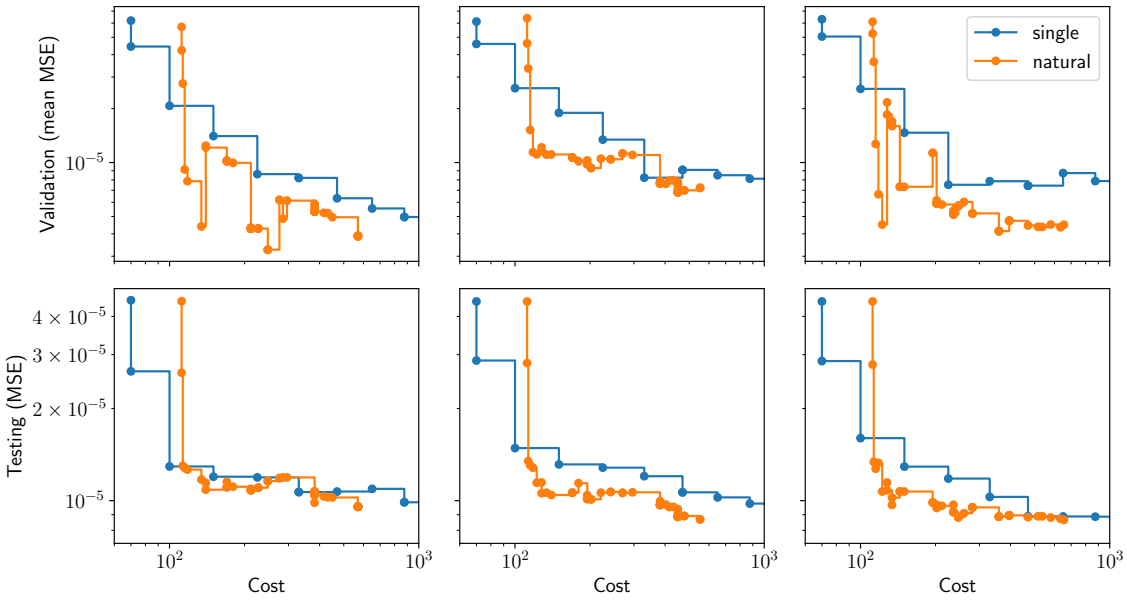
$$\text{cost(SPT-100, Ground, 3 charges, 200 cells)} = 0.01. \tag{19}$$

Here, we scale by a cost of 10 for the different systems/configurations.

In Figure 10, we show the results for both approaches, looking at three different data replicates. For this scenario, the two approaches for updating $\Delta \boldsymbol{p}$ lead to slightly different results. When we use the "increaseAll" approach, we find that the adaptive learning algorithm performs comparable to the single fidelity approach; see Figure 10a. However, the "increaseUsingCost" approach forces to algorithm to explore higher degree jumps in the cheaper models, leading to an improvement in the testing error at low costs. However, ultimately this improvement is minor and does not significantly reduce the error. Additionally, the algorithm gets stuck increasing the degrees of nodes corresponding to alternative systems, even though ultimately increasing the degree of the target system may be necessary to achieve an error comparable to the single fidelity approach.

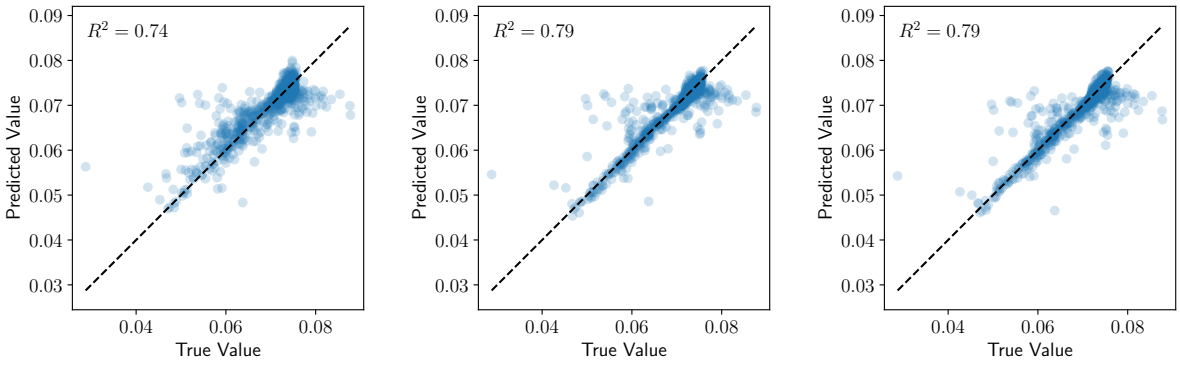


(a) Set Δp using "increaseAll"



(b) Set Δp using "increaseUsingCost"

Fig. 8 Adaptive algorithm results for Scenario 1 of the Hall thruster system for two different methods of increasing Δp. Each column represents training results using different realizations of the data. The top row shows the average MSE for the cross-validation step during training at the target node. The bottom row shows the MSE on a set of testing data at the target node. The testing data is the same for each data replicate.



(a) Single Fidelity **(b) Natural DAG, "increaseAll"** **(c) Natural DAG, "increaseUsingCost"**

Fig. 9 Parity plots showing testing data for Scenario 1 of the Hall thruster system when we consider a maximum cost of 300. Results shown correspond to the third column of Figure 8.

3. Hall thruster: Scenario 3

For Scenario 3 (see Figure 7c) in the Hall thruster system, we again have the input dimension $d = 3$ and use the following model costs.

$$\text{cost(H9, Space)} = 1.00 \tag{20}$$

$$\text{cost(SPT-100, Space, 3 charges, 200 cells)} = 0.10 \tag{21}$$

$$\text{cost(SPT-100, Space, 2 charges, 200 cells)} = 0.02 \tag{22}$$

$$\text{cost(SPT-100, Space, 3 charges, 100 cells)} = 0.02 \tag{23}$$

$$\text{cost(SPT-100, Space, 2 charges, 100 cells)} = 0.004. \tag{24}$$

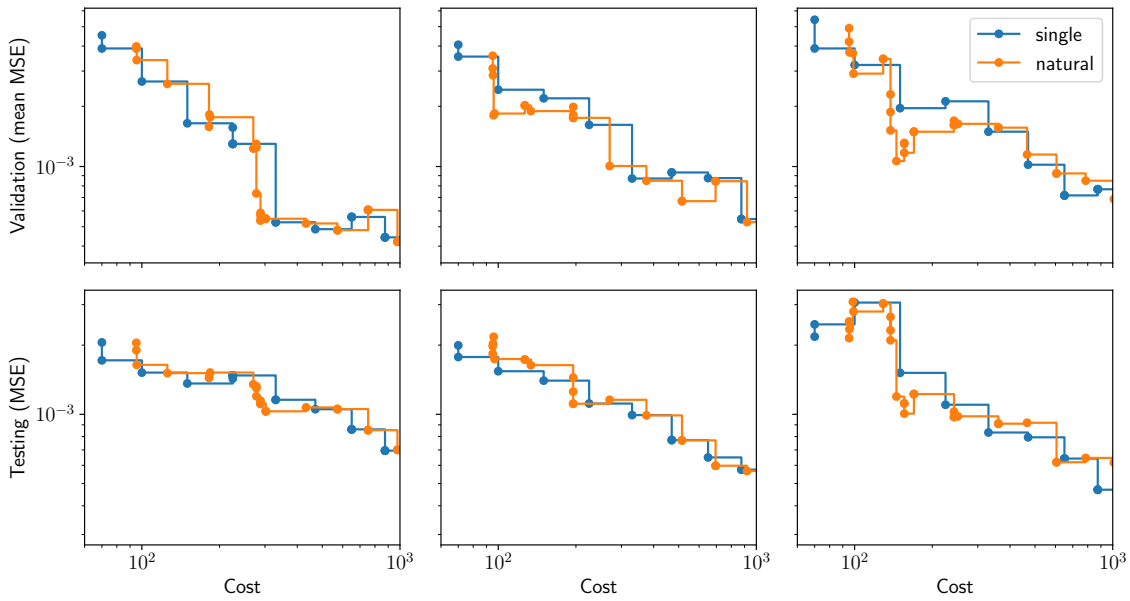
Note that the scaling between different systems/configurations/fidelities is the same as in the previous two examples.

Figure 11 shows the results of three data replications for this example. Here, we only show the results for the "increaseAll" approach. However, we note that using the "increaseUsingCost" approach did not lead to any improvement in the results. The algorithm immediately decides to start sampling the target system rather than the alternative systems. Therefore, we do not see a significant difference between the adaptive algorithm and the single fidelity approach.

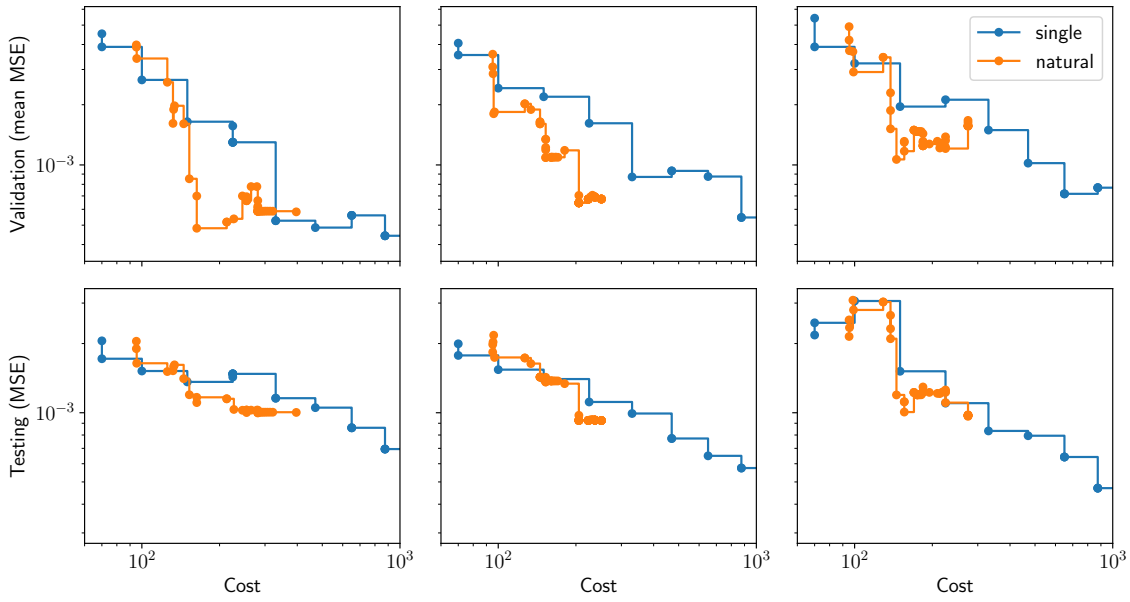
V. Conclusions

In this contribution, we presented a novel greedy adaptation strategy for the efficient construction of a surrogate network using the MFNets framework, which constructs surrogates by fusing realizations from models arranged in a direct acyclic graph. We have expanded this approach to support transfer learning tasks that have the potential to allow the modeling and design of systems even if large datasets cannot be acquired. For instance, in this manuscript we focus on the estimation of the performance (thrust) for an Hall-effect ion thruster (H9) in interplanetary flight for which limited flight data are available. To overcome the scarcity of data for this problem, we leverage additional data from a related system, namely a Stationary Plasma Thruster (SPT-100), when flying a similar mission or analyzed during ground tests.

We first provided numerical evidence for the greedy algorithm by discussing a verification polynomial test case. The example polynomial system demonstrated that the algorithm behaved as expected and that including accurate dependency relationships among the models improved algorithm performance. For the Hall thruster modeling case, we proposed three transfer learning scenarios. For the first scenario, we only included SPT-100 data and we found that the algorithm led to improved performance with respect to the single-fidelity case when adopting the Natural DAG. For the remaining two transfer learning scenarios, we considered the full task of transferring information from SPT-100 datasets to H9. As expected, these latter scenarios were more challenging with more of a knowledge gap to bridge. The algorithm led to improved predictions compared with a single fidelity approach; however, the algorithm struggled with leveraging information from different systems/configurations. We hypothesize that this is because these alternative information sources did not contain useful information for fitting a polynomial surrogate, rather than due to



(a) Set Δp using "increaseAll"



(b) Set Δp using "increaseUsingCost"

Fig. 10 Adaptive algorithm results for Scenario 2 of the Hall thruster system for two different methods of increasing Δp. Each column represents training results using different realizations of the data. The top row shows the average MSE for the cross-validation step during training at the target node. The bottom row shows the MSE on a set of testing data at the target node. The testing data is the same for each data replicate.

the algorithm not finding the polynomial degrees that would result in a successful transfer. Moreover, our approach at this time is based on a polynomial expansion and some numerical experiments have suggested that the dependency of thrust on the input parameters may be non-smooth. Future investigations can further investigate this aspect and consider alternative representations for the nodes that can be more tolerant of noise and/or discontinuity.

In the examples we considered, we explored two different approaches for creating a set of candidate MFNet

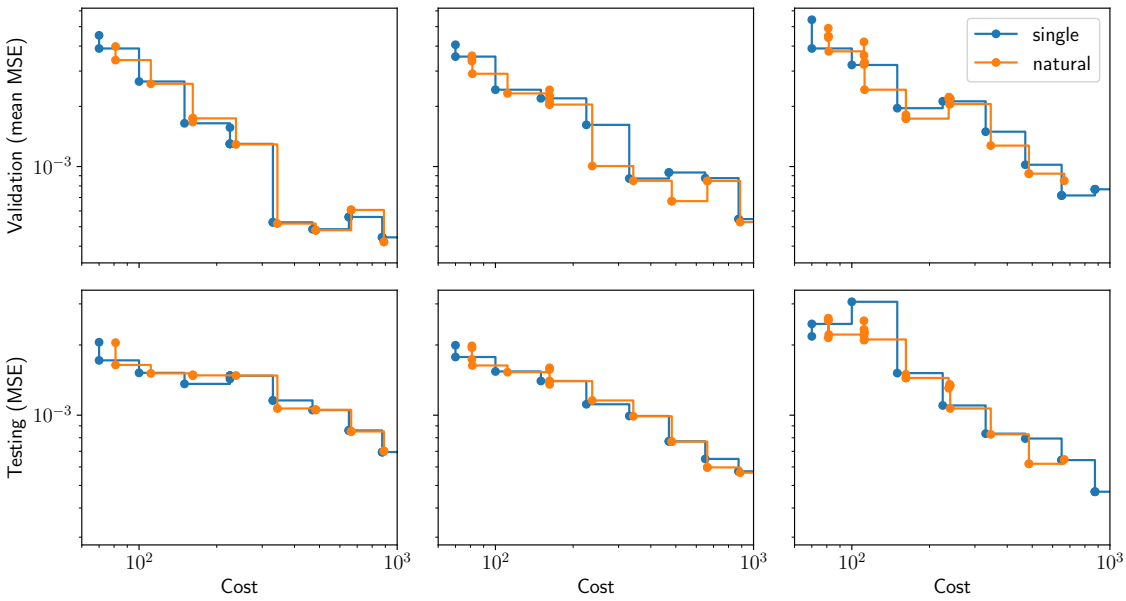


Fig. 11 Adaptive algorithm results for Scenario 3 of the Hall thruster system. Each column represents training results using different realizations of the data. The top row shows the average MSE for the cross-validation step during training at the target node. The bottom row shows the MSE on a set of testing data at the target node. The testing data is the same for each data replicate.

surrogates. One approach allowed for polynomial degree jumps without considering model costs. The other approach attempted to take into account model cost information and allow jumps in less expensive models prior to the more expensive models. We found the first approach sampled the target system too liberally, e.g., see polynomial example in Section IV.A, whereas the second approach avoided sampling the expensive target system but continued to increase the degrees of cheaper models despite very limited improvements, e.g., in Scenario 2 for the Hall thruster system. Ultimately, a middle ground between these two approaches is likely warranted.

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) Program through the FASTMath Institute. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC (NTESS), a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration (DOE/NNSA) under contract DE-NA0003525. This written work is authored by an employee of NTESS. The employee, not NTESS, owns the right, title and interest in and to the written work and is responsible for its contents. Any subjective views or opinions that might be expressed in the written work do not necessarily represent the views of the U.S. Government. The publisher acknowledges that the U.S. Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this written work or allow others to do so, for U.S. Government purposes. The DOE will provide public access to results of federally sponsored research in accordance with the DOE Public Access Plan. Thomas Marks was supported by NASA through the Joint Advanced Propulsion Institute (JANUS), a NASA Space Technology Research Institute, under grant number 80NSSC21K1118. This research was additionally supported in part through computational resources provided by Advanced Research Computing at the University of Michigan.

References

[1] Doty, J., and Camberos, J., "Statistical, Modular Systems Integration Using Combined Energy & Exergy Concepts," *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2010. https://doi.org/10.2514/6.2010-

278, URL https://arc.aiaa.org/doi/abs/10.2514/6.2010-278, AIAA-2010-278.

[2] Rinauto, B., Gupta, S., Chowdhury, S., and Maldonado, V., "Design of a Modular Offline Reconfigurable Unmanned Aerial Vehicle," *AIAA Information Systems-AIAA Infotech @ Aerospace*, 2017. https://doi.org/10.2514/6.2017-1159, URL https://arc.aiaa.org/doi/abs/10.2514/6.2017-1159, AIAA-2017-1159.

[3] Kishimoto, N., and Natori, M., "Control and Mechanical Characteristics of Hierarchical Modular Structures," *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 2005. https://doi.org/10.2514/6.2005-1957, URL https://arc.aiaa.org/doi/abs/10.2514/6.2005-1957, AIAA-2005-1957.

[4] Wentz, J., Geraci, G., Davis, O., Eldred, M., Jakeman, J., Mustaev, A., and Gorodetsky, A., "Multi-Fidelity Bayesian Networks for Transfer Learning in Modular Systems," *AIAA SCITECH 2025 Forum*, 2025, p. 1691. https://doi.org/10.2514/6.2025-1961, URL https://arc.aiaa.org/doi/abs/10.2514/6.2025-1961.

[5] Gorodetsky, A., Jakeman, J., Geraci, G., and Eldred, M., "MFNets: multi-fidelity data-driven networks for Bayesian learning and prediction," *International Journal for Uncertainty Quantification*, Vol. 10, 2020, pp. 595–622.

[6] Gorodetsky, A., Jakeman, J., and Geraci, G., "MFNets: data efficient all-at-once learning of multifidelity surrogates as directed networks of information sources," *Computational Mechanics*, Vol. 68, 2021, pp. 741–758.

[7] Zeng, X., Geraci, G., Gorodetsky, A., Jakeman, J., and Ghanem, R., "Boosting Efficiency and Reducing Graph Reliance: Basis Adaptation Integration in Bayesian Multi-Fidelity NetworksA Comprehensive Survey on Transfer Learning," *Computer Methods in Applied Mechanics and Engineering*, 2025. Accepted for publication.

[8] Red-Horse, J., and Paez, T., "Sandia National Laboratories Validation Workshop: Structural dynamics application," *Computer Methods in Applied Mechanics and Engineering*, Vol. 197, No. 29, 2008, pp. 2578–2584. https://doi.org/https://doi.org/10.1016/j.cma.2007.09.031, URL https://www.sciencedirect.com/science/article/pii/S0045782507005002, validation Challenge Workshop.

[9] Hofer, R. R., Cusson, S. E., Lobbia, R. R., and Gallimore, A. D., "The H9 Magnetically Shielded Hall thruster," *Proc. of the 35th International Electric Propulsion Conference, Atlanta, GA, USA*, 2017. IEPC-2017-232.

[10] Sankovic, J., Hamley, J., and Haag, T., "Performance Evaluation of the Russian SPT-100 Thruster at NASA LeRC," *Proc. of the 23rd International Electric Propulsion Conference, Seattle, Washington, USA*, 1993. IEPC-1993-094.

[11] Snyder, J. S., Sereno, V., Kerl, T., and Li, J., "Electric Propulsion for the Psyche Mission: One Year to Launch," *AIAA Propulsion and Energy 2021 Forum*, 2021. https://doi.org/10.2514/6.2021-3426, URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-3426.

[12] Boeuf, J.-P., "Tutorial: Physics and Modeling of Hall Thrusters," *Journal of Applied Physics*, Vol. 121, No. 1, 2017, p. 011101. https://doi.org/10.1063/1.4972269.

[13] Marks, T. A., and Jorns, B. A., "Challenges with the Self-Consistent Implementation of Closure Models for Anomalous Electron Transport in Fluid Simulations of Hall Thrusters," *Plasma Sources Science and Technology*, Vol. 32, No. 4, 2023, p. 045016. https://doi.org/10.1088/1361-6595/accd18.

[14] SPT-100, "SPT-100," http://www.astronautix.com/s/spt-100.html, 1997-2019. [Online; accessed 11-December-2024].

[15] Su, L. L., and Jorns, B. A., "Performance comparison of a 9-kW magnetically shielded Hall thruster operating on xenon and krypton," *Journal of Applied Physics*, Vol. 130, No. 16, 2021, p. 163306. https://doi.org/10.1063/5.0066849, URL https://doi.org/10.1063/5.0066849.

[16] Su, L. L., Marks, T. A., and Jorns, B. A., "Trends in mass utilization of a magnetically shielded Hall thruster operating on xenon and krypton," *Plasma Sources Science and Technology*, Vol. 33, No. 6, 2024, p. 065008. https://doi.org/10.1088/1361-6595/ad52be.

[17] Eckels, J. D., Marks, T. A., Allen, M. G., Jorns, B. A., and Gorodetsky, A. A., "Hall thruster model improvement by multidisciplinary uncertainty quantification," *Journal of Electric Propulsion*, Vol. 3, No. 19, 2024. https://doi.org/10.1007/s44205-024-00079-w.

[18] Marks, T. A., Eckels, J. D., Mora, G. E., and Gorodetsky, A. A., "Uncertainty quantification of a multi-component Hall thruster model at varying facility pressures," *Journal of Applied Physics*, Vol. 138, No. 15, 2025, p. 153305. https://doi.org/10.1063/5.0283796, URL https://doi.org/10.1063/5.0283796.

[19] Marks, T., Schedler, P., and Jorns, B., "HallThruster.Jl: A Julia Package for 1D Hall Thruster Discharge Simulation," *Journal of Open Source Software*, Vol. 8, No. 86, 2023, p. 4672. https://doi.org/10.21105/joss.04672.

[20] Allen, M. G., Eckels, J. D., Byrne, M. P., Gorodetsky, A. A., and Jorns, B. A., "Application of Optimal Experimental Design to Characterize Pressure Related Facility Effects in a Hall Thruster," *Proc. of the 37th International Electric Propulsion Conference*, 2022.

[21] Marks, T. A., Eckels, J. D., Mora, G. E., and Gorodetsky, A. A., "Uncertainty quantification of a multi-component Hall thruster model at varying facility pressures," *J. Appl. Phys.*, Vol. 138, No. 15, 2025. https://doi.org/10.1063/5.0283796.

[22] Zeng, X., Geraci, G., Eldred, M. S., Jakeman, J. D., Gorodetsky, A. A., and Ghanem, R., "Multifidelity uncertainty quantification with models based on dissimilar parameters," *Computer Methods in Applied Mechanics and Engineering*, Vol. 415, 2023, p. 116205. https://doi.org/https://doi.org/10.1016/j.cma.2023.116205, URL https://www.sciencedirect.com/science/article/pii/S0045782523003298.

[23] Maitre, O. L., and Knio, O., *Spectral Methods for Uncertainty Quantification. With Applications to Computational Fluid Dynamics*, Springer Netherlands, 2010.

[24] Hadigol, M., and Doostan, A., "Least squares polynomial chaos expansion: A review of sampling strategies," *Computer Methods in Applied Mechanics and Engineering*, Vol. 332, 2018, pp. 382–407. https://doi.org/https://doi.org/10.1016/j.cma.2017.12.019, URL https://www.sciencedirect.com/science/article/pii/S0045782517307697.

[25] Geraci, G., Eldred, M. S., Gorodetsky, A., and Jakeman, J., "Recent advancements in Multilevel-Multifidelity techniques for forward UQ in the DARPA Sequoia project," *AIAA Scitech 2019 Forum*, 2019, p. 0722.

[26] "Dakota 6.23.0 documentation." Tech. Rep. SAND2025-14179O, Sandia National Laboratories, Albuquerque, NM, November 2025. URL http://snl-dakota.github.io.

[27] Jakeman, J. D., Eldred, M. S., Geraci, G., and Gorodetsky, A., "Adaptive multi-index collocation for uncertainty quantification and sensitivity analysis," *International Journal for Numerical Methods in Engineering*, Vol. 121, No. 6, 2020, pp. 1314–1343. https://doi.org/https://doi.org/10.1002/nme.6268, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.6268.