Learning surrogate models for simulation-based optimization

Alison Cozad\textsuperscript{1,2}, Nikolaos V. Sahinidis\textsuperscript{*1,2}, and David C. Miller\textsuperscript{1}

\textsuperscript{1}National Energy Technology Laboratory, Pittsburgh, PA 15236
\textsuperscript{2}Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213

October 5, 2013

Abstract

We address a central problem in modeling, namely that of learning an algebraic model from data obtained from simulations or experiments. We propose a methodology that uses a small number of simulations or experiments to learn models that are as accurate and as simple as possible. The approach begins by building a low-complexity surrogate model. The model is built using a best subset technique that leverages an integer programming formulation to allow for the efficient consideration of a large number of possible functional components in the model. The model is then improved systematically through the use of derivative-free optimization solvers to adaptively sample new simulation or experimental points. We describe \textsc{ALAMO}, the computational implementation of the proposed methodology, along with examples and extensive computational comparisons between \textsc{ALAMO} and a variety of machine learning techniques, including Latin hypercube sampling, simple least squares regression, and the lasso.

1 Introduction

Chemical process simulation and computational fluid dynamic methods have long been used industrially and academically to design and test systems and processes [12, 32, 35, 38]. These numerical models offer high levels of accuracy and precision in their predictive capabilities at the cost of requiring specialized simulation software. The structure of these simulations lends well to predictive use but can impose challenges when used in an optimization or design

\textsuperscript{*}Author for correspondence. E-mail: sahinidis@cmu.edu
setting [4, 13, 14]. This paper considers the optimization of processes via simulation, experiments, and, more generally, black-box processes. The general optimization problem we address is

$$\min \ f(x)$$

s.t. \ $g(x) \leq 0$

$x \in A \subset \mathbb{R}^n$

where we desire to minimize a cost function, $f(x)$, with respect to the degrees of freedom $x$. These degrees of freedom can range from continuous decisions of operating conditions and geometry to discrete decisions about process alternatives and flowsheet configuration. Furthermore, they are required to satisfy a set of constraints $g(x) \leq 0$ as well as box constraints $A$, which include lower and upper bounds. We assume that one or more of the functions $f$ and $g$ are not directly available in algebraic form, but, for any given value of $x$, the corresponding $f(x)$ and $g(x)$ can be computed via an input-output black box.

The above optimization problem comes with three primary challenges. First, the standard approach to optimization using derivative-based or algebraic solvers (e.g., CONOPT [11], IPOPT [39], and SNOPT [17]) optimize using derivative information. However, the objective and/or constraint set must be treated as black boxes since algebraic models and derivatives are not directly available for many simulation packages. This is often because simulators may incorporate proprietary software, numerical integrators, lookup tables, etc. More advanced global optimization methods, such as BARON [37], require an algebraic functional form to find and prove a globally optimal solution. In either case, algebraic forms for each function $f$ and $g$ are required to first locate then classify feasible and optimal decision variables. Some standard optimization solvers can solve in the absence of an algebraic form so long as derivatives can be evaluated or approximated. In fact, IPOPT has been used to directly optimize ASPEN based simulations [7]. However, as these simulations become more complex, they become difficult to reliably converge. Thus, such direct optimization is difficult without specialized algorithms. Even given a perfectly robust simulation, a third challenge is still a factor: costly and/or noisy function evaluations. Due to the high sampling requirements of derivative estimation, costly function evaluations hinder the use of such solvers. Noisy function evaluations that arise naturally in numerical simulations and experiments limit the accuracy and efficacy of derivative estimations [6].

Derivative-free optimization (DFO) offers a class of solvers intended to fully or partially overcome these three challenges. These algorithms are designed for optimization problems when derivatives are unavailable, unreliable, or prohibitively expensive to evaluate [25, 34]. These solvers attempt to achieve an optimal feasible point using a minimal number of black-box function calls. Although DFO methods can be used to address black-box models with costly and noisy function evaluations, these methods are often unable to find true optimal solutions when the number of degrees of freedom exceeds about ten, even in the absence of constraints and integer variables, as shown in a recent computational
To overcome the challenges of simulation-based optimization, there has been significant work done to generate one or more surrogate models (known in some fields as metamodels or reduced-order models) of the black-box functions $f(x)$ and/or $g(x)$ [24, 40]. These methods are most commonly applied to purely continuous problems. Once generated, these abstracted models can be optimized using more traditional algebraic or derivative-based solvers. These surrogate-based methods can be distinguished by the modeling method chosen. Previous work has focused on existing modeling methods primarily from machine learning and statistics. The main source of modeling techniques has been reduced-order modeling, which is defined as the production of some low-dimensional system (i.e., reduced-order model–ROM) that has similar response characteristics to the modeled simulation or system [3]. The goal of reduced-order modeling is to create an approximation of a black box that necessitates far less CPU time to evaluate. In the context of using models for optimization, ROMs must also have an algebraic form that can be exploited by derivative-based optimization software.

Most often, a single model of the objective function is approximated then optimized; in fewer cases, the constraint set is also modeled. However, work has been done to first disaggregate a black-box simulation into distinct blocks and model each block separately, ensuring that all relevant connectivity variables are also modeled. By disaggregating the process, smaller, more robust, simulation units are explored. These disaggregated process units can be combined with disjunctive constraint sets and blocks linked via connectivity variables to formulate complex mixed-integer optimization models. Significant work has been done using kriging models to either model the full system [10, 21, 31] or the disaggregated process [6]. Palmer and Realf [31] have looked at the indirect optimization of steady-state simulators using kriging surrogate models. Along a similar path, David and Ierapetritou [10] use full process kriging models to determine global solutions and refine the solution using local response surfaces around the optima. To address uncertainty concerns in the black-box systems, Huang et al. [21] have used kriging models on full processes. Caballero and Grossmann [6] have investigated disaggregated (modular) flowsheet optimization using kriging models to represent process units with low-level noise. Recent work by Henao and Maravelias [20] has shown success modeling individual chemical process units using artificial neural networks.

Previous work has focused solely on developing models that are highly accurate. Unless physical simplifications are available, common forms of reduced-order modeling often have a bumpy and complex functional form which is disadvantageous in algebraic optimization where smaller, compact algebraic forms are desirable. Our work aims to develop accurate surrogates that are tailored to reduce the difficulty and improve the tractability of the final optimization model. By considering the final purpose of the surrogate models, we strive to identify functional forms that can be easily incorporated into larger optimization models without the difficulty of inherently complex ROMs.

To address the black-box nature of these simulations as well as costly func-
tion evaluations that have limited robustness, we have developed a surrogate model generation method. To ensure the robustness of the simulation, either a single unit or a small set of units is considered. If the existing simulation is complex, such as a complete flowsheet, it is advantageous to disaggregate it into smaller blocks. Subsequently, using an adaptive sampling procedure, low-complexity algebraic models are built, tested, exploited, and improved using a combination of derivative and derivative-free optimization solvers, machine learning, and statistical techniques. Surrogate models generated using this methodology can be used in an algebraic optimization framework with flexible objective functions and additional constraints.

We have developed the package ALAMO that implements the proposed methodology. ALAMO (Automated Learning of Algebraic Models for Optimization) interfaces with a user-defined simulator and problem space to iteratively model and interrogate the simulation to identify an accurate, low complexity set of algebraic models to approximate the system. Similar existing modeling packages, such as SUorrogate MOdelling lab toolbox (SUMO) [18], fail to generate surrogate models with sufficiently low complexity. Eureqa [28] can be used to search for low complexity models; however, it operates on a fixed data set and can often lead to rather complex functional forms. The proposed methodology allows ALAMO to generate compact models that improve and validate the surrogate models by adaptively sampling the simulation.

Prior works in the process systems engineering literature have approached simulation-based optimization by relying on existing modeling methodologies, mostly kriging and neural network modeling, to build surrogate process models that can be optimized with derivative-based optimization techniques. In the current paper, we depart from the use of existing modeling methodologies. The primary contribution of this work is to introduce a novel model-building methodology. In our approach, surrogate models are built in a systematic fashion that tailors these models for optimization tractability, while also aiming to ensure high model accuracy. The remainder of this paper is structured as follows. In the next section, we discuss the proposed methodology in detail. To better explain the steps involved in our proposed method, we present the modeling of an illustrative problem. Then, computational results are presented demonstrating the accuracy and efficiency of the proposed approach against common surrogate model learning methodologies in the machine learning literature. Finally, the proposed methodology is demonstrated on a case study to quantify the environmental and economic trade-offs of power plant carbon capture systems.

2 Proposed methodology

To address commonly found challenges in black-box simulation robustness, the process may be disaggregated into smaller process blocks that contain one or more process units. This step also enables access to larger optimization models that arise from this modular nature including superstructure optimization
and more complex problem topologies. Lastly, models generated from less complex systems or smaller process blocks are generally more accurate than larger process counterparts. The connectivity and process alternatives decisions can be determined using the surrogate models in a derivative-based optimization structure.

After process disaggregation, we identify a set of surrogate models of relevant responses for each block. The set of responses $z_k, k \in K$, (outlet material and energy stream properties, efficiencies, design requirements, etc.) are modeled as a function $\hat{z}_k(x)$ of input variables $x_d, d \in D$, which become optimization decision variables (inlet material and energy stream properties, operating conditions, unit geometry, etc.). We assume that the problem domain is bounded for each input variable. The surrogate models for each block can then be combined with an algebraic objective, design constraints, and heat and mass balances to formulate an algebraic optimization problem.

Surrogate models are constructed using an iterative method as depicted in Figure 1. First, an initial design of experiments is generated over the problem space and the simulation is queried at these points. For the purposes here, the specific design of experiments used does not play a strong role in the final solution. This is because the initial sample set is small and is adaptively improved as the procedure progresses, leaving a data set that bears little resemblance to the initial design of experiments. In the example cases and experimental studies shown, we use either Latin hypercube sampling or a 2-level factorial design for this step.

Next, we build a simple, algebraic model using this initial training set of data. The empirical model error, i.e., the deviation of the model from the data, can be calculated at this point. However, the true error, i.e., the deviation of the model from the true system, is unknown. Furthermore, we do not know if we have an accurate model or have sampled the problem space sufficiently. The current surrogate models are subsequently tested against the simulation using an adaptive sampling technique that we refer to as error maximization sampling (EMS). If the model is shown to be inconsistent above a specified tolerance, those newly sampled data points are added to the training set. The surrogate models are iteratively rebuilt and improved until the adaptive sampling routine fails to find model inconsistencies.

This section provides further details of the specific techniques used to generate the accurate, low-complexity surrogate models which can then be refined by adaptively sampling the simulation using error maximization sampling.

2.1 Surrogate model generation

For the modeling problem, we have a set of $N$ training points; each training point has a set of input data $x_{id}$ for each point $i = 1, \ldots, N$, and a set of responses $z_{ik}, k = 1, \ldots, m$. We assume that the underlying functional form of the response surfaces is unknown. We would like to generate a model for each response with sufficient complexity to accurately model the simulation while maintaining adequate simplicity such that the resulting optimization model is tractable in an
Sampling:
Generate an initial data set \([x_{id}, z_{ik}]\) from previous data or by running a design of experiments.

Initialize variables:
\(i = 1, \text{MaxIter}, y_0(T)\) for \(T = 1, \ldots, T_{\text{max}}, N = N_0\)

Build model:
Build a model \(\hat{z}_k(x)\) for \(z_k(x), k = 1, \ldots, m\)
based on the current \(N\) training set points, \([x_{id}, z_{ik}]\)

Adaptive sampling:
Perform error maximization sampling to get \([x_{i'd}, z_{i'k}]_{\text{ems}}\) for \(i' = N + 1, \ldots, N_{\text{ems}}\)

Update training set:
\([x_{i'd}, z_{i'k}]\) for \(i = 1, \ldots, N + N_{\text{ems}}, N = N + N_{\text{ems}}, i = i + 1\)

\[N_{\text{ems}} = N_{\text{max}}(N, n)\]

\[\max_k \frac{z_k - \hat{z}_k}{\text{range}(z_k)} \leq \text{tol} \]

Figure 1: Algorithmic flowchart
algebraic optimization framework. For example, surrogate modeling techniques such as kriging [9, 26] and ANNs [19] satisfy accuracy requirements but result in rough, complex functions that are difficult to solve using provable derivative-based optimization software. On the other end of the spectrum, linear regression models may not represent the highly nonlinear nature of chemical processes despite their advantageous optimization simplicity.

We would like to strike a balance between model accuracy and optimization tractability. This is difficult because the functional form of the model is unknown. To address this hurdle, we identify combinations of simple basis functions that define a low-complexity, accurate functional form for each response. The simple basis functions $X_j(x)$, $j \in B$, can be chosen from physical principles or statistical fitting functions. Therefore, the functional forms available to kriging or ANNs could also be potential basis functions while avoiding a full set of either form in the interest of obtaining a lower-complexity surrogate. In most cases, we allow for constant terms and the basis functional forms shown in Table 1 with user specified values for $\alpha$ and $\gamma$. The resulting surrogate model is a linear combination of nonlinear basis set as follows:

$$\hat{z} = \sum_{j \in B} \beta_j X_j(x)$$  \hspace{1cm} (1)

where the $j$th basis function is multiplied by a corresponding coefficient $\beta_j$.

The ordinary least squares regression problem,

$$\min_{\beta} \sum_{i=1}^{N} \left( z_i - \sum_{j \in B} \beta_j X_{ij} \right)^2$$  \hspace{1cm} (2)

could be used to solve for the regression coefficients (model parameters), $\beta$, by minimizing the sum of the squared model error over the training data points $i$. In most cases, if we solve (2) to find the least squares regression coefficients, we would run into the same high complexity issues we wish to avoid because most or all of the potential bases appear in the surrogate. Model reduction techniques that are widely used in statistics and machine learning can be used to reduce the number of terms in a model or, similarly, to attain a sparse regression coefficient vector. These methods have the added benefit of reducing the over fitting seen in the model by using only a subset of the available basis functions. Model reduction methods can be as simple as backward elimination, forward selection, or stepwise regression to find a statistically significant model [5]. However, these methods can easily miss synergistic effects from multiple basis functions that may exhibit poor fitting properties on an individual basis. To explore all possible combined effects, a best subset method [5] can be used to enumerate models for all possible combinations of the basis set then to choose the best subset of basis functions using a measure of the model fitness that is sensitive to over fitting. This method is guaranteed to pick the best model according to the chosen fitness measure; however, due to the factorial complexity of the modeling algorithm, this method is often ruled out for large basis sets. Recent work has
Table 1: Potential simple basis function forms

<table>
<thead>
<tr>
<th>Category</th>
<th>$X_j(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. Polynomial</td>
<td>$(x_d)\alpha$</td>
</tr>
<tr>
<td>II. Multinomial</td>
<td>$\prod_{d \in D' \subseteq D} (x_d)^{\alpha_d}$</td>
</tr>
<tr>
<td>III. Exponential and logarithmic forms</td>
<td>$\exp\left(\frac{x_d}{\gamma}\right)^\alpha$, $\log\left(\frac{x_d}{\gamma}\right)^\alpha$</td>
</tr>
<tr>
<td>IV. Expected bases</td>
<td>From experience, simple inspection, physical phenomena, etc.</td>
</tr>
</tbody>
</table>

seen the addition of graph theory [16], branch and bound algorithms [15], and QR decomposition [15] to enhance the scalability and computation time of the best subset problem. Regularization techniques use a squared objective that is penalized by a function of the magnitude of the regression coefficients to perform model reduction and reduce over fitting. However, as we show in Section 4, the commonly used lasso regularization ($L_1$-norm penalty function) results in far less accurate solutions, likely due to the highly coupled and structured set of basis functions.

The general best subset problem can be represented as

\[(BS) \min_{S, \beta} \Phi(S, \beta) \quad \text{s.t.} \quad S \subseteq B\]

where $\Phi(S, \beta)$ is a surrogate model goodness-of-fit measure for the subset of basis function $S$ and regression coefficients $\beta$. Using (BS), the following surrogate model can be found using a subset of the basis functions:

$$
\hat{z}(x) = \sum_{j \in S} \beta_j X_j(x).
$$

(3)

Through a series of reformulation and simplification steps, we reformulate (BS) into a form that we can efficiently solve. We first define the subset of basis functions using a vector of binary variables $y$ to determine if a basis function is active in the model or not. For each basis function $j \in B$, if $j \in S$, then $y_j = 1$; otherwise, $j \notin S$ and $y_j = 0$. Using this binary vector, Equation (3) can be described over the full basis set $B$:

$$
\hat{z}(x) = \sum_{j \in B} y_j \beta_j X_j(x).
$$

The vector $y$ can also be used to reformulate (BS) into a mixed-integer nonlinear problem:

\[(BS1) \min_{\beta, y} \Phi(\beta, y) \quad \text{s.t.} \quad y_j \in \{0, 1\}.\]
At this point, it is beneficial to make three main reformulations to (BS1). To remove the complication of integer bilinear terms, we replace $y_j \beta_j$ with $2 * |\mathcal{B}|$ big-M constraints
\[
\beta^l y_j \leq \beta_j \leq \beta^u y_j
\]
which uses lower and upper bounds, $\beta^l$ and $\beta^u$, on $\beta$. These constraints force $\beta_j$ to zero if $y_j = 0$, while allowing $\beta_j$ to take on a nonzero value within its bounds if $y_j = 1$.

The second reformulation stems from observing that many goodness-of-fit measures can be decoupled into two parts: (a) model sizing and (b) basis and parameter selection, as follows:
\[
\min_{\beta, T, y} \Phi(\beta, T, y) = \min_T \left\{ \min_{\beta, y} [\Phi_{\beta, y}(\beta, y)|T] + \Phi_T(T) \right\}
\]

Here, $\Phi_T(T)$ and $\Phi_{\beta, y}(\beta, y)|T$ denote the model sizing and basis/parameter selection parts of the information criterion, respectively. Hence, we can pose the best subset selection minimization problem as a nested minimization problem, where the inner minimization problem determines the basis functions and coefficients and the outer minimization problem defines the complexity of the model. This results in the following problem for some goodness-of-fit measure:
\[
\min_{T \in \{1, \ldots, T_u\}} \left\{ \min_{\beta, y} [\Phi_{\beta, y}(\beta, y)|T] + \Phi_T(T) \right\}
\]
s.t.
\[
\sum_{j \in \mathcal{B}} y_j = T
\]
\[
\beta^l y_j \leq \beta_j \leq \beta^u y_j \quad j \in \mathcal{B}
\]
\[
y_j \in \{0, 1\} \quad j \in \mathcal{B}
\]

The selection of a model fitness measure is fundamental to the success of these methods. The measure must reflect the accuracy of the model while remaining sensitive to over fitting the model. This brings up an important distinction between empirical error and true error. As mentioned previously, the empirical error of a model is the inaccuracy seen between the model and the data points used to build the data. As a properly built model increases in complexity, the empirical error of a properly trained model is non-increasing. The true error of a model represents the deviation of the model from the true function. Ideally, this would be the best fitness measure of a model. However, unless the algebraic form of the true function is known, the true error can only be estimated. Two common methods to estimate model fitness are cross-validation and information criteria.

Cross-validation techniques train the model on a majority portion of the data while reserving the minority of the data for validation. This is done so that cross-validation is able to test the model on data that was not used to build the model, i.e., an independent data set. This is generally done several times reserving different portions of the data for validations. By doing this, an estimate of the true model error is achieved.
Like cross-validation, information criteria are sensitive to both the empirical error and over-fitting. Information criteria are able to directly account for the model complexity, which is not the case for cross-validation. These measures are tied to the maximum likelihood method of model parameter estimation [36]; one such case is linear regression if the error is assumed to take on a normal distribution. Information criteria are comprised of several alternatives for order-selection rules including, the widely known, Akaike information criterion [2], Bayesian information criterion [36], generalized information criterion [36], etc.

Each information criterion gives a measure of the accuracy versus the complexity of a model [2, 29]. Due to the large number of basis functions available to the model, the goodness-of-fit measure we use is the corrected Akaike information criterion [22],

$$AIC_c(S, \beta) = N \log \left( \frac{1}{N} \sum_{i=1}^{N} \left( z_i - \sum_{j \in S} \beta_j X_{ij} \right)^2 \right) + 2|S| + \frac{2|S| (|S| + 1)}{N - |S| - 1} \quad (5)$$

which simplifies to the Akaike information criterion for large data sets. Equation (5) can be given in the form of (4) and can be posed as a nested minimization problem. We further reformulate the inner objective function to obtain an inner objective equivalent to the least-squares objective.

We make two simplifications to ensure tractability and efficiency of the final algorithm. First, we leverage the finite solution space of the outer minimization problem by parameterizing it with respect to $T$. The inner minimization problem is solved for increasing values of $T$ until a minimum is reached. To enforce this, we include the following constraint:

$$\sum_{j \in B} y_j = T \quad j \in B$$

Second, in order to pose the inner problem as a mixed-integer linear problem (MILP), we remove the nonlinear objective and replace it with the $L_1$-norm error as follows:

$$\min SE = \sum_{i=1}^{N} \left| z_i - \sum_{j \in B} \beta_j X_{ij} \right|$$

and then replace each instance of $|w|$ in $SE$ by $w'$ and add constraints $w' \geq w$ and $w' \geq -w$ in the formulation. However, to retain the least squares nature of the resulting coefficients, we use the stationarity condition with respect to the parameters $\beta$:

$$\frac{d}{d\beta_j} \sum_{i=1}^{N} \left( z_i - \sum_{j \in S} \beta_j X_{ij} \right)^2 \propto \sum_{i=1}^{N} X_{ij} \left( z_i - \sum_{j \in S} \beta_j X_{ij} \right) = 0, \quad j \in S \quad (6)$$
Equation (6) is used as big-M constraints to define the basis coefficients:

\[-U_j(1 - y_j) \leq \sum_{i=1}^{N} X_{ij} \left( z^i - \sum_{j \in B} \beta_j X_{ij} \right) \leq U_j(1 - y_j).\]

In doing this, we choose which basis functions to use in the model based on linear error and the value of the regression parameters based on a squared error.

These reformulations and simplifications result in the following MILP:

\[
\begin{align*}
\text{(M)} & \quad \min \sum_{i=1}^{N} w_i \\
\text{s.t.} \quad & \quad w_i \geq z_i - \sum_{j \in B} \beta_j X_{ij}, \quad i = 1, \ldots, N \\
& \quad w_i \geq \sum_{j \in B} \beta_j X_{ij} - z_i, \quad i = 1, \ldots, N \\
& \quad \sum_{j \in B} y_j = T \\
& \quad -U_j(1 - y_j) \leq \sum_{i=1}^{N} X_{ij} \left( z^i - \sum_{j \in B} \beta_j X_{ij} \right) \leq U_j(1 - y_j), \quad j \in B \\
& \quad \beta^l y_j \leq \beta_j \leq \beta^u y_j, \quad j \in B \\
& \quad y_{kj} \in \{0, 1\}, \quad j \in B \\
& \quad \beta^l_j \leq \beta_j \leq \beta^u_j, \quad j \in B
\end{align*}
\]

Model (M) is used to solve for the best $T$-term subset of the original set of basis functions. By solving (M) with a small $T$ and increasing that value until the information criterion worsens, the proposed method is able to efficiently solve the best subset problem to find accurate low-complexity models. A more detailed accounting of the algorithm used to find the most accurate and simple model given a set of data points can be seen in Algorithm 1.

### 2.2 Adaptive sampling

Adaptive sampling, active learning, or supervised learning is a means of learning a system or process by querying the system at desired input levels. By intelligently selecting sample points, a model can be more accurate with less information. Preferably, perfect information could be used to train a model. However, with limited computational resources, function evaluations must also be limited. Without knowing a system, it is not possible to know how much information is needed or where data points should be chosen a priori. Yet, there are techniques that use previous system knowledge or model structure to better sample a system so as to balance the need for information with the computational cost of that information.
**Algorithm 1 Build model**

Given a training set of dependent and independent data points \([x_{id}, z_{ik}]\) for \(i = 1, \ldots, N, \ d = 1, \ldots, n, \ k = 1, \ldots, m\); initial values for the vector of binary variables, \(y_0(T)\); a list of basis functions to be evaluated, \(X_j(x) \ \forall j \in \mathcal{B}\); and relevant tolerance values

Generate basis value set, \(X_{ij} \leftarrow X_j(x_i)\), for \(i = 1, \ldots, N\) and \(j \in \mathcal{B}\)

Initialize a high maximum error allowed at each point, \(e^{\text{max}}\)

for \(k \leftarrow 1\) to \(m\) do

Generate a set of bounds for \(\beta_j\) and \(U_i\)

Calculate the maximum terms allowed, \(\text{maxTerms} \leftarrow \max (1, \min(N, |\mathcal{B}|))\)

for \(t \leftarrow 1\) to \(\text{maxTerms}\) do

Solve (M) for \(T = t\) to determine the optimal \(t\) basis functions and \(\beta\)

Store \(\beta(t)\) and \(y(t)\)

Compute \(AICc(t)\)

if \(t > 1\) then

if \(AICc(t) > AICc(t - 1)\) then \(\triangleright\) Information criterion worsened

Set the final model complexity \(T^f = t - 1\)

break

else if \(\frac{AICc(t) - AICc(t - 1)}{AICc(t - 1) \leq \text{tol2}}\) then

Set the final model complexity \(T^f = t\)

break

else if \(\frac{1}{N} \sum_{i=1}^{N} \left| z_i - \sum_{j \in \mathcal{B}} \beta_j X_{ij} \right| \leq \text{tol3}\) then

Set the final model complexity \(T^f = t\)

break

end if

end if

end if

Update error bounds \(e^{\text{max}} \leftarrow \frac{1}{N} \sum_{i=1}^{N} \left| z_i - \sum_{j \in \mathcal{B}} \beta_j X_{ij} \right|\)

end for

return final values for \(\beta = \beta(T^f)\) and \(y = y(T^f)\)
One field of active learning that is relevant here is optimal design of experiments [33]. The approach is motivated by the knowledge of how that data is used to model the system. It is possible to estimate the variance of the final model parameters $\beta$ by calculating a Fisher information matrix, which is a function of only the input variable values and final model functional form. The Fisher information matrix gives a summary of the amount of data due to the model parameters [36]. Optimal design of experiments methods attempt to maximize a function of this matrix, or minimize its inverse. For the proposed approach, we only have a very loose knowledge about what the final model’s functional form is before we actually obtain it. It is, therefore, likely that the strength of this method could be diluted by the presence of numerous unused basis functions.

Instead, we interrogate the simulation in locations of model inconsistency. In doing this, we are given two important pieces of information: (a) the location of a data point that helps the next iteration’s model accuracy and (b) a conservative estimate of the true error of the model. We use this information to both improve and validate the current model. Algorithm 2 presents the specific details of this procedure.

We pose this sampling technique as a black-box optimization problem to find areas in the problem space that maximize the squared relative model error:

$$\max_{x^l \leq x \leq x^u} \left( \frac{z(x) - \hat{z}(x)}{z(x)} \right)^2$$

(7)

While the algebraic form of the current surrogate model is known, the true black-box value is not; therefore, the entire objective function must be treated as a black box. This necessitates the use of derivative-free optimization algorithms, a class of algorithms that do not require the availability of analytical expressions for the objective function and constraints of the problem to be optimized [8]. As shown recently in [34], these methods are most effective in low-dimensional cases, like we have here thanks to the decomposition of the original large simulation into pieces. As the derivative-free solver progresses, the error can be calculated at newly sampled candidate points. If areas of sufficient model mismatch are located, the new points are added to the training set and the model is rebuilt. At the end of this step, the true model error can be estimated by what is, effectively, holdout cross validation using the newly sampled points. If the new true error estimated is above tolerance, the model is retrained using an updated training set. If the error estimate is below tolerance, the proposed approach has converged to a final surrogate model.

### 3 Illustrative example

To better demonstrate the proposed methodology in its most basic form, we walk through the modeling of steam density, $\rho$, as a function of heat duty, $Q$, in a flash drum modeled in Aspen Plus. The thermodynamics of this process are defined by the 1995 IAPWS steam table formulation. We identify a surrogate
Algorithm 2 Error maximization sampling

Given a set of dependent and independent data points \([x_{id}, z_{ik}]\) for \(i = 1, \ldots, N, d = 1, \ldots, n, k = 1, \ldots, m\) and a set of models \(\hat{z}_k(x)\), \(k = 1, \ldots, m\). Additionally, prespecified values for range of \(x\), \([x^{\text{min}}, x^{\text{max}}]\); the minimum number of sample points, \(N^{\text{ems}}_{\text{min}}\); the maximum number of sample points as a function of \(N\), \(N^{\text{ems}}_{\text{max}}(N)\); and a tolerance on the maximum error, tol4, are given.

Calculate the squared error and combined error at each point using the objective in Equation (7),
\[ e_{ik} \leftarrow \left( \frac{z_{ik} - \hat{z}_k(x_i)}{z_{ik}} \right)^2 \text{ and } E_i \leftarrow \sum_{k=1}^{m} e_{ik}. \]

Initialize the number of error maximization points added, \(N^{\text{ems}} \leftarrow 0\)

\[ \text{while } N^{\text{ems}} \leq N^{\text{ems}}_{\text{max}} \text{ do} \]

Using a black-box solver (we call it \texttt{df0([objective function], [initial objective values], [initial x values], [x^{\text{min}}, x^{\text{max}}, [requested points]])}) that meets the requirements listed, request new sample points \(x^{\text{req}} \leftarrow \texttt{df0}(E(x), E_i, x_i, x^{\text{min}}, x^{\text{max}}, N^{\text{req}})\)

\[ \text{for } i \leftarrow 1 \text{ to } N^{\text{req}} \text{ do} \]

Sample simulation at \(x^{\text{req}}_i\) to get \(z^{\text{req}}_i\)

Append error values, \(e_{N+i,k} \leftarrow e(x^{\text{req}}_i, z^{\text{req}}_i)\) and \(E_{N+i} \leftarrow E(x^{\text{req}}_i, z^{\text{req}}_i)\)

\[ \text{Update } N^{\text{ems}} \leftarrow N^{\text{ems}} + 1 \text{ and } N \leftarrow N + 1 \]

\[ \text{if } N^{\text{ems}} \geq N^{\text{ems}}_{\text{min}} \text{ then} \]

\[ \text{if } \max_{ik}(e_{ik}) \geq \text{tol4} \text{ then} \]

\[ \text{return } x, z, N^{\text{ems}} \]

\end if

\end if

\end for

\end while
model, $\hat{\rho}(Q)$ in $\frac{kg}{m^3}$, as a function of heat duty from 13,000 W to 40,000 W. The functional form includes basis functions of the form shown in Table 1, where $\alpha = 0, \pm \frac{1}{3}, \pm 1, \pm 2, \pm 3$ and $\gamma = 10,000$ W. This leads to a potential functional form shown below with 13 basis functions:

$$
\hat{\rho}(Q) = \beta_0 + \beta_1 Q + \frac{\beta_2}{Q} + \beta_3 Q^2 + \frac{\beta_4}{Q^2} + \beta_5 Q^3 + \frac{\beta_6}{Q^3} + \beta_7 \sqrt{Q} + \frac{\beta_8}{\sqrt{Q}} + \beta_9 \sqrt[3]{Q} + \frac{\beta_{10}}{\sqrt[3]{Q}} + \beta_11 e^{10000 \frac{Q}{10000}} + \beta_12 \ln \frac{Q}{10000}
$$

The process is shown in Figure 2, where water enters a 1 atm flash drum at 1 atm and 298 K. The water stream is flashed into a steam and liquid stream. To facilitate the flash, heat is added to the flash drum. The aim of the proposed methodology is to find an accurate surrogate model using only the basis functions needed to properly fit the simulated system. This is done using a minimal but flexible data set size.

Figure 2: Diagram of flash drum

The algorithm begins by taking an initial data set of Latin hypercube data points over the prespecified range of the input variables. For illustration, we consider two data points over $Q \in [13000 \text{ W}, 40000 \text{ W}]$. During each iteration, a model is built using a subset of the basis functions shown above that models the information in the current data set with enough bases to have high accuracy but not so many as to over fit the data or have unneeded bases. For this example, in Algorithm 2, we choose $N_{\text{min}} = 1$ and $N_{\text{max}}(N)$ to be the greater of 20% of the current data set size and $|D| + 10 = 11$.

To better illustrate the model building step in this method, we look more closely at the modeling process in the last iteration. At this stage, there are 9 data points in the training set. To begin, the best model using a single term is established. The best surrogate model is determined by minimizing the squared error of the model. The $AICc$ is calculated for the one-term model. Following this, the best two-term model is chosen from $\binom{13}{2} = 156$ basis combinations. This is done using the MILP model (M). From here, the $AICc$ of the two-term model is compared to the one-term model (see Figure 3 for $AICc$ versus model size for the last iteration). Since $AICc$ decreases with the addition of the second term, there is
<table>
<thead>
<tr>
<th>Terms allowed</th>
<th>Surrogate model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \frac{1.524 \cdot 10^{11}}{Q^3} )</td>
</tr>
<tr>
<td>2</td>
<td>( \frac{1.212 \cdot 10^{11}}{Q^3} + 0.07519 )</td>
</tr>
<tr>
<td>3</td>
<td>( 0.1086 \ln(Q) - (4.754 \cdot 10^{-10}) Q^2 ) ( + \frac{1.348 \cdot 10^{11}}{Q^3} )</td>
</tr>
<tr>
<td>4</td>
<td>( 2.339 \sqrt[3]{0.0001 Q} - 1.385 \ln(Q) ) ( - \frac{9518.0}{Q} + \frac{2.075 \cdot 10^{11}}{Q^3} )</td>
</tr>
<tr>
<td>5</td>
<td>( 80.7 \sqrt[3]{0.0001 Q} - \frac{39.0}{\sqrt[3]{0.0001 Q}} - 41.54 \ln(Q) + \frac{3.911 \cdot 10^{11}}{Q^3} ) ( - (1.571 \cdot 10^{-13}) Q^3 )</td>
</tr>
<tr>
<td>6</td>
<td>( 27.06 \sqrt[3]{0.0001 Q} - 0.000908 Q + 84.94 \sqrt[3]{0.0001 Q} ) ( - 91.6 \sqrt[3]{0.0001 Q} - \frac{1.159 \cdot 10^5}{Q} + \frac{4.768 \cdot 10^{11}}{Q^3} )</td>
</tr>
</tbody>
</table>

*Table 2: Surrogate models built at increasing complexity for iteration 7 of the flash drum*

enough information to say that there are at least two-terms in the model and we test to see if a three term model would be a better fit. This continues until the AICc worsens, as it does going from a five- to six-term model. Table 2 shows the models found for each model size.

As is often the case, allowing an additional term does not just append a single term to the previous model. Instead, many terms are completely changed as the model complexity is allowed to increase. This gives further justification to leveraging the basis functions’ synergistic effects, something that is not possible using more common statistical techniques such as backward elimination or forward selection.

The mean squared error and AICc for each model size is shown in Figure 3. An important thing to note is that, even though the model error decreases from five to six terms, the information criterion is showing that this increase in accuracy is not worth the added complexity of that sixth term. The models for each iteration are built in a similar way.

If we rewind for a moment to look at the algorithm from the beginning, we can examine the progress throughout the rest of the algorithm. As mentioned previously, a number of new simulation points are chosen using the adaptive sampling technique referred to as error maximization. New points are selected to maximize the current iterate’s error. These points are then simulated and compared to the model values to determine the model error. Each iteration is terminated in this example when (a) the error exceeds a tolerance of 1%,
normalized by the range of $\rho$ or (b) the maximum number of points sampled, $N_{\text{max}}(N)$, has been reached. In the case of (a), a new model is rebuilt using the previously and newly simulated data points. If the maximum number of points sampled for the iteration has been reached, the current error is estimated by the normalized root mean square error (RMSE). If this error exceeds the specified tolerance of 0.5%, the training set is updated and a new iteration begins. Since these error maximization sampling (EMS) points are likely to be a conservative estimate of the average model error over the entire domain, if the estimated error tolerance is not violated, then the model is considered sufficiently accurate. In this situation, the basis functions are fixed and the sparse $\beta$ vector is updated using a simple least squares estimation using the latest data set and the algorithm terminates.

Figure 4 shows the estimated and maximum errors for each iteration of the algorithm. After the completion of the algorithm, we also compared each model to an independent data set of 200 evenly sampled points. This calculation gave the test error shown in Figure 4. This figure demonstrates that the estimated error is, generally, a conservative estimate of the test error. Additionally, as the algorithm progresses, the EMS sampling is able to intelligently select new sample points to give better information to the model building step. As more information about the system is obtained from simulations, the model building step is able to show more complex models that best represent the data as seen in Figures 5 and 6, and Table 3.

Figures 5 and 6 show several snapshots in the algorithm. The models for Iterations 1, 3, 5, and 7 are shown alongside the true simulation curve. The training data set, shown in white, and the newly sampled EMS points, shown in blue, are depicted as well. During Iterations 1–6, either one or two EMS points were added per iteration (Table 3). Figures 5 and 6 illustrate the effectiveness of selecting points with high model mismatch. During the seventh and final iteration, eleven EMS points were added. None of these points violated the 1% maximum error tolerance and lead to an estimated normalized error of 0.16%. The models and model complexity for each iteration, are also shown in Table 3.
Figure 4: Progression through the algorithm as shown by the estimated, maximum found, and test model errors

After the final iteration, we have a five-term surrogate model of the following form:

\[
\hat{\rho}(Q) = 62.37 \sqrt{0.0001Q} - \frac{64.53}{\sqrt{0.0001Q}} - 47.81 \ln(0.0001Q) + 3.659 \cdot 10^{12} Q^3 - (6.576 \cdot 10^{-15}) Q^3
\]

The terms in the model are chosen using a training set of 9 points and the coefficients are reevaluated using the full data set of 20 points.

4 Implementation and computational results

We combined the tailored model generation solver with the error maximization sampling routine to set up the basis of ALAMO. The front end code is written in Matlab and uses the optimization software GAMS/CPLEX for the solution of all model building optimization subproblems and SNOBFIT (Stable Noisy Optimization by Branch and FIT) [23] for the black-box solver in the adaptive sampling. MINQ [30] is used to solve SNOBFIT’s quadratic programming subproblems. In order to facilitate comparisons with existing techniques, several more standard model generation alternatives are integrated into ALAMO including

1. Ordinary least-squares regression (OLR)
   OLR solves (2) using all of the available basis functions.

2. Exhaustive search best subset method (EBS)
   EBS exhaustively searches all of the possible best \( T \) subsets of a problem. Like the method proposed, \( T \) is parameterized. The best \( T \) subset is chosen to minimize the squared model error.
Figure 5: Surrogate model for Iterations 1 and 3 of the proposed methodology along with current and EMS data sets compared with true simulated data.
Figure 6: Surrogate model for Iterations 5 and 7 of the proposed methodology along with current and EMS data sets compared with true simulated data.
<table>
<thead>
<tr>
<th>Iter</th>
<th>EMS points</th>
<th>Terms</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 1 1</td>
<td></td>
<td>$\frac{3867.0}{Q}$</td>
</tr>
<tr>
<td>2</td>
<td>1 1 1</td>
<td></td>
<td>$\frac{1.299 \cdot 10^{12}}{Q^3}$</td>
</tr>
<tr>
<td>3</td>
<td>1 2 2</td>
<td></td>
<td>$\frac{1.098 \cdot 10^{12}}{Q^3} + 0.07649$</td>
</tr>
<tr>
<td>4</td>
<td>1 2 2</td>
<td></td>
<td>$\frac{1.094 \cdot 10^{12}}{Q^3} + 0.07398$</td>
</tr>
<tr>
<td>5</td>
<td>2 3 0.4522 $\ln(0.0001 Q) - (1.418 \cdot 10^{-5}) Q + \frac{1.409 \cdot 10^{12}}{Q^3}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1 3 0.4094 $\ln(0.0001 Q) - (1.262 \cdot 10^{-5}) Q + \frac{1.39 \cdot 10^{12}}{Q^3}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>11 5 58.9 $\sqrt[3]{0.0001 Q} - \frac{60.94}{\sqrt[3]{0.0001 Q}} - 45.15 \ln(0.0001 Q) + \frac{3.524 \cdot 10^{12}}{Q^3} - (6.259 \cdot 10^{-15}) Q^3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Final</td>
<td>NA 5 62.37 $\sqrt[3]{0.0001 Q} - \frac{64.53}{\sqrt[3]{0.0001 Q}} - 47.81 \ln(0.0001 Q) + \frac{3.659 \cdot 10^{12}}{Q^3} - (6.576 \cdot 10^{-15}) Q^3$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 3*: Progression of modeling the flash drum through the proposed methodology
<table>
<thead>
<tr>
<th>Function type</th>
<th>Functional form</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>( z(x) = \beta x^\alpha \exp(x_j) )</td>
</tr>
<tr>
<td>II</td>
<td>( z(x) = \beta x_i^\alpha \log(x_j) )</td>
</tr>
<tr>
<td>III</td>
<td>( z(x) = \beta x_i^\alpha x_j^\nu )</td>
</tr>
<tr>
<td>IV</td>
<td>( z(x) = \frac{\beta}{\gamma + x_i^\alpha} )</td>
</tr>
</tbody>
</table>

*Table 4: Functional forms for test set B.*

3. The lasso regularization (LASSO)

LASSO uses the MATLABR2011b implementation of the lasso algorithm and chooses the regularization parameter based on 10-fold cross-validation. Latin hypercube sampling has also been added to ALAMO for an alternative sampling method. In this section, we look at the model accuracy, quality of point sampling, and final model length by comparing these alternatives to our implementation of the best subset method which solves (M) for increasing \( T \) and uses the EMS proposed here for the model builder and sampling routines, respectively.

Two test sets were considered. Test set A considers the problem of learning functional forms that are present in the algorithm’s basis set. The functions are made up of basis functions that are available to the model with two and three input variables. Basis functions from Table 1 were used with \( \alpha = \{\pm 3, \pm 2, \pm 1, \pm 0.5\} \) for polynomials, \( \alpha = \{\pm 2, \pm 1, \pm 0.5\} \) for pairwise polynomials, and exponential and logarithmic functions with \( \alpha = 1, \gamma = 1 \). A total of 27 two-dimensional functions were generated with varying complexity from two to ten randomly chosen terms, where three functions were generated at each complexity. In addition, 18 three-dimensional functions generated similarly from two to seven randomly chosen terms.

The 12 functions in test set B were generated using functional forms unknown to the modeling method. These input functional forms can be seen in Table 4. Function parameters for test sets A and B are chosen from uniform distributions where \( \beta \in [-1, 1], \alpha, \nu \in [-3, 3], \gamma \in [-5, 5], \) and \( i, j \in \{1, 2\} \).

Each test function was modeled as a black-box simulation using M, OLR, EBS, and LASSO with five different random seeds. To compare the EMS adaptive sampling with a single Latin hypercube, once each test function was modeled using the full algorithm a second model was generated using a single Latin hypercube (SLH) with the same number of data points. In doing this, we are able to ascertain whether or not the EMS sampling method is able to extract more information per data point than a Latin hypercube design of experiments.
The resulting normalized test error is calculated as follows:

$$e_{k}^{\text{norm}} = \frac{1}{|T|} \sqrt{\sum_{i \in T} \left( z_{ik} - \hat{z}_k(x_i) \right)^2}$$

where $e_{k}^{\text{norm}}$ is the normalized error for dependent variable $z_k$ calculated from an independent set of data points $i \in T$. The normalized error, final model terms, and required function evaluations for these tests is shown in Figures 7 and 8, and Tables 5 and 6.

Figures 7 and 8 show the performance profile of each modeling and sampling combination for test sets A and B, respectively. The profile is determined by finding the percentage of the total test set that was solved within a normalized test error, given as the $x$-axis. For example, the proposed method using M/EMS was able to solve 80% of test set A to within 0.1% error as calculated using Equation 8. Across both test sets, the proposed method, shown in solid red, was able to outperform all other modeling and sampling combinations. This can be seen first through increased model accuracy. Secondly, this accuracy was achieved using, in general, fewer function evaluations. This shows that the EMS sampling method is able to get more information per function evaluation than the SLH over each of the modeling methods. In most cases, the M/EMS approach was able to attain this high accuracy with less complex models than either LASSO or OLR.

Tables 5 and 6 show the range of final model complexities, $T_{\text{surrogate}}$, found by each set of modeling and sampling routines. In most cases, this value is given as a range since each test was repeated with five different initial random seeds. In nearly every case, the method we have outlined here, M/EMS, was able to reach a model that was no bigger (and often much smaller) than any of the alternative method combinations. Our proposed method was able to find models that are the most accurate and are at least as simple as the tested alternatives.

For test set A we are able to analyze the number of terms more thoroughly, since the true number of terms, $T_{\text{true}}$, is known and can be compared against $T_{\text{surrogate}}$. The results of this comparison are shown in detail in Figure 9 and summarized by the mean terms deviation, $T_{\text{surrogate}} - T_{\text{true}}$, as well as the standard deviation of this value for each run of test set A. The smaller this deviation is the fewer terms were needed to fit the test function. The results show that, not only, is the proposed modeling method (M) more consistent, but it also exhibits fewer terms than the other alternatives. However, the sampling method chosen seems to have little effect on the resulting model size for this test set.

The results of these experiments show the success of the proposed method in terms of our three main modeling goals: accuracy, model simplicity, and modeling efficiency, with respect to function evaluations. In the next section, we proceed to demonstrate how the derived models can be used to perform optimization studies.
Figure 7: Top: Fraction of functions in test set A modeled within a normalized test error. Bottom: Fraction of functions in test set A modeled within a specified number of function evaluations.

<table>
<thead>
<tr>
<th>No. of inputs</th>
<th>No. of true terms</th>
<th>M/EMS</th>
<th>M/SLH</th>
<th>EBS/EMS</th>
<th>EBS/SLH</th>
<th>LASSO/EMS</th>
<th>LASSO/SLH</th>
<th>OLR/EMS</th>
<th>OLR/SLH</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>[2, 2]</td>
<td>[6, 8]</td>
<td>[6, 11]</td>
<td>[12, 15]</td>
<td>[12, 15]</td>
<td>[12, 15]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>[3, 4]</td>
<td>[3, 4]</td>
<td>[5, 12]</td>
<td>[5, 10]</td>
<td>[12, 14]</td>
<td>[12, 14]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>[3, 4]</td>
<td>[5, 12]</td>
<td>[5, 10]</td>
<td>[11, 12]</td>
<td>[11, 12]</td>
<td>[10, 16]</td>
<td>[10, 16]</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>[5, 6]</td>
<td>[5, 6]</td>
<td>[7, 10]</td>
<td>[6, 7]</td>
<td>[11, 13]</td>
<td>[11, 13]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>[4, 6]</td>
<td>[4, 6]</td>
<td>[7, 11]</td>
<td>[6, 12]</td>
<td>[8, 13]</td>
<td>[8, 13]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>NA*</td>
<td>NA*</td>
<td>[5, 6]</td>
<td>[6, 9]</td>
<td>[10, 15]</td>
<td>[10, 15]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>[5, 6]</td>
<td>[6, 9]</td>
<td>[5, 14]</td>
<td>[10, 14]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Note: Due to large CPU times EBS tests were not run with greater than 9 true terms

Table 5: The average minimum and maximum number of surrogate model terms for test set A.
Figure 8: Top: Fraction of functions in test set B modeled within a normalized test error. Since the EMS and SLH sampling schemes use the same number of data points, only the differences between models is important here. Bottom: Fraction of functions in test set B modeled within a specified number of function evaluations. Note: EBS is not plotted since the CPU time was too great for several of the more complex problems.
Table 6: The average minimum and maximum number of surrogate model terms for test set B. Note: Due to large CPU times EBS tests were not run on test set B.

<table>
<thead>
<tr>
<th>True function type</th>
<th>Function ID</th>
<th>M/EMS</th>
<th>M/SLH</th>
<th>LASSO/EMS</th>
<th>LASSO/SLH</th>
<th>OLR/EMS</th>
<th>OLR/SLH</th>
</tr>
</thead>
<tbody>
<tr>
<td>I a</td>
<td>[4, 10]</td>
<td>5</td>
<td>[3, 5]</td>
<td>[4, 9]</td>
<td>[6, 17]</td>
<td>[6, 17]</td>
<td></td>
</tr>
<tr>
<td>I c</td>
<td>[3, 10]</td>
<td>4</td>
<td>[6, 9]</td>
<td>[8, 9]</td>
<td>[4, 10]</td>
<td>[13, 17]</td>
<td>[13, 17]</td>
</tr>
<tr>
<td>II a</td>
<td>[4, 6]</td>
<td>1</td>
<td>[4, 10]</td>
<td>[8, 15]</td>
<td>[7, 9]</td>
<td>[15, 19]</td>
<td>[15, 19]</td>
</tr>
<tr>
<td>II b</td>
<td>[1, 7]</td>
<td>3</td>
<td>[1, 9]</td>
<td>[13, 16]</td>
<td>[11, 17]</td>
<td>[13, 30]</td>
<td>[13, 30]</td>
</tr>
<tr>
<td>II c</td>
<td>[5, 12]</td>
<td>4</td>
<td>[5, 12]</td>
<td>[9, 13]</td>
<td>[9, 16]</td>
<td>[9, 19]</td>
<td>[9, 19]</td>
</tr>
<tr>
<td>III a</td>
<td>[3, 4]</td>
<td>5</td>
<td>[1, 4]</td>
<td>[2, 5]</td>
<td>[2, 5]</td>
<td>[9, 20]</td>
<td>[9, 20]</td>
</tr>
<tr>
<td>III b</td>
<td>[3, 4]</td>
<td>4</td>
<td>[1, 4]</td>
<td>5</td>
<td>5</td>
<td>[9, 20]</td>
<td>[9, 20]</td>
</tr>
<tr>
<td>III c</td>
<td>[3, 4]</td>
<td>3</td>
<td>[3, 4]</td>
<td>[5, 8]</td>
<td>[5, 9]</td>
<td>[18, 24]</td>
<td>[18, 24]</td>
</tr>
<tr>
<td>IV a</td>
<td>[7, 8]</td>
<td>7</td>
<td>[4, 10]</td>
<td>[8, 17]</td>
<td>[11, 18]</td>
<td>[13, 19]</td>
<td>[13, 19]</td>
</tr>
<tr>
<td>IV b</td>
<td>[8, 9]</td>
<td>6</td>
<td>[9, 10]</td>
<td>[8, 12]</td>
<td>[10, 14]</td>
<td>[9, 17]</td>
<td>[9, 17]</td>
</tr>
<tr>
<td>IV c</td>
<td>[6, 9]</td>
<td>6</td>
<td>[9, 10]</td>
<td>[5, 13]</td>
<td>[4, 12]</td>
<td>[13, 15]</td>
<td>[13, 15]</td>
</tr>
</tbody>
</table>

Table 7: Mean and standard deviation values for $T_{\text{surrogate}} - T_{\text{true}}$ from test set A for each modeling and sampling method tested

<table>
<thead>
<tr>
<th>Method</th>
<th>M/EMS</th>
<th>M/SLH</th>
<th>LASSO/EMS</th>
<th>LASSO/SLH</th>
<th>OLR/EMS</th>
<th>OLR/SLH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>-0.860</td>
<td>-0.814</td>
<td>4.619</td>
<td>3.879</td>
<td>10.177</td>
<td>10.177</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>1.688</td>
<td>1.703</td>
<td>5.005</td>
<td>4.745</td>
<td>8.261</td>
<td>8.261</td>
</tr>
</tbody>
</table>
Figure 9: Histograms and normal distribution fits for $T_{\text{surrogate}} - T_{\text{true}}$ from test set A for each modeling and sampling method tested
5 Case Study: Carbon Capture Adsorber

The synthesis of optimal carbon capture processes is of immense importance for identifying promising technologies and materials to reduce greenhouse gas emissions from fossil fuel power plants. Because of the complex interaction of material behavior and process configuration, an optimization-based process synthesis provides a rigorous means to assess the tradeoffs among capital costs, parasitic power consumption, and other operating costs for a given technology and material. However, to accurately predict the performance of such systems, rigorous equipment models are necessary. The ALAMO methodology provides a way to use the required rigorous models (via algebraic surrogates) within a superstructure-based optimization framework for process synthesis. Here, we demonstrate the ALAMO methodology to create an algebraic surrogate model from a solid sorbent adsorber model, which is one technology under development for post-combustion carbon capture.

The adsorber under consideration is a bubbling fluidized bed (BFB), which is modeled as a system of PDEs in Aspen Custom Modeler and is described by Lee and Miller [27]. Figure 10 shows the major features of the model. CO2 rich gas enters the bottom of the reactor and contacts the solid. CO2 lean solid enters the top of the bed and leaves as a CO2 rich stream from the bottom of the bed in the underflow configuration. Cooling water flows through internal cooling tubes to remove excess heat of adsorption from the reactor.

The goal is to develop an algebraic surrogate model, which accurately captures the behavior of the BFB adsorber under a range of operating and design conditions so that the surrogate model can ultimately be used for process synthesis. To demonstrate the performance of ALAMO on such a rigorous model, we consider two degrees of freedom that strongly affect the performance of the adsorber: reactor bed depth, \( L \), and cooling water flow rate, \( F \). The accuracy of the model is measured based on the percentage of CO\(_2\) removed, \( r \), from the flue gas stream:

\[
r = \frac{\text{CO}_2 \text{ in outlet flow}}{\text{CO}_2 \text{ in flue gas}}.
\]  

(9)

Figure 10: Diagram of the solid sorbent carbon capture adsorber
The CO$_2$ removed increases with diminishing return as both the bed depth and cooling water flow rate are increased.

We use ALAMO to develop a surrogate model for $r$ as a function of $L \in [2 \text{ m}, 10 \text{ m}]$ and $F \in [1000 \frac{\text{kmol}}{\text{h}}, 100000 \frac{\text{kmol}}{\text{h}}]$. We use the model builder M and the EMS adaptive sampling routine. Since we expect many of the solutions of the pareto analysis to be at the bounds, the initial sample set is at the corner points of the design space to ensure that we do not extrapolate beyond the sampled region.

The estimated and maximum normalized error calculated at each iteration is shown in Figure 11. A separate test set of 394 data points were collected to test the model at each iteration. The estimated normalized test error using this data is also shown. As with the illustrative example, we see an increase in the estimated and maximum error found during EMS from iteration 1 to 2. This is an effect of sampling improvement and not of model inaccuracy. The test error remains constant between these two models. The effectiveness of using EMS points to generate a conservative estimate of model error for use as a stopping criterion can be seen as well.

![Normalized model error](image_url)

*Figure 11: Normalized error progression through the entire algorithm*

Figures 12 and 13 show the model improvement and EMS sampling for Iterations 1, 3, 5, and 7. For example, during Iteration 1 the model (on the y-axis) was built based on the four simulated data points (on the x-axis). The adaptive sampling routine returned two points of high model error shown in blue. After the final EMS iteration, 7, the newly sampled points no longer violate the model beyond the estimated normalized error tolerance of 0.5. The resulting models, model size, and number of EMS points generated for each iteration are shown in Table 8.

Figure 14 shows the complexity solution path as the complexity of the model is allowed to increase from one to thirteen basis functions. The potential basis set had 67 possible terms available to build each model. The decrease in error seen in Figure 14 from twelve to thirteen terms does not justify the use of a
Figure 12: Modeled current and EMS data for Iterations 1 and 3 of the proposed methodology compared with true simulated data.
Figure 13: Modeled current and EMS data for Iterations 5 and 7 of the proposed methodology compared with true simulated data
<table>
<thead>
<tr>
<th>Iter</th>
<th>EMS points</th>
<th>Terms</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>$(8.13 \cdot 10^{-7}) \ F \sqrt{L} + 0.138$</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>$(3.57 \cdot 10^{-6}) \ F \sqrt{L} - (7.17 \cdot 10^{-12}) \ F^2 L + \frac{1.3 \cdot 10^{11}}{F L} - \frac{(2.07 \cdot 10^{-11}) \ F^2}{L}$</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>$0.0142 \sqrt{F} - (4.4 \cdot 10^{-6}) \ F \sqrt{L} + (1.13 \cdot 10^{-6}) \ F L$</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>6</td>
<td>$0.0117 \sqrt{F} - (5.85 \cdot 10^{-12}) \ F e^L - \frac{(5.22 \cdot 10^{-6}) \ F}{\sqrt{L}} - \frac{0.0687}{L} + \frac{0.0707}{\sqrt{F}} + \frac{(9.85 \cdot 10^{-21}) \ F^4}{L^4}$</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>5</td>
<td>$0.0181 \sqrt{F} - 0.00124 \sqrt{F} - \frac{(1.77 \cdot 10^{-5}) \ F}{L^2} - \frac{(5.57 \cdot 10^{-11}) \ F^2}{L^2} + \frac{(2.13 \cdot 10^{-10}) \ F^4}{L^4}$</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>8</td>
<td>$\frac{0.365}{L^3} - \frac{0.314}{L^2} - \frac{3.81 \sqrt{L}}{F} + 0.257 \sqrt{L} + \frac{7.2 L^2}{F} + \frac{7.94 \cdot 10^6}{F^2 L} - \frac{3.08 \cdot 10^{13}}{F^4 L^4} - (4.58 \cdot 10^{-8}) \ F L$</td>
</tr>
<tr>
<td>7</td>
<td>12</td>
<td>12</td>
<td>$0.2 \log(0.2L) + (1.81 \cdot 10^{-13}) \ F^2 L - \frac{(1.02 \cdot 10^{-7}) \ F}{\sqrt{L}} - \frac{4.72 \sqrt{L}}{F L} - (5.49 \cdot 10^{-5}) \ F \sqrt{L} + (5.14 \cdot 10^{-5}) \ \frac{F L}{\sqrt{L}} + (4.04 \cdot 10^{-4}) L^2 + \frac{0.385}{\sqrt{L}} + 0.148 \sqrt{L} + \frac{8.71 L^2}{F} + \frac{1.08 \cdot 10^7}{F^2 L^2} - \frac{4.18 \cdot 10^{13}}{F^4 L^4}$</td>
</tr>
<tr>
<td>Final</td>
<td>NA</td>
<td>12</td>
<td>$0.201 \log(0.2L) + (1.82 \cdot 10^{-13}) \ F^2 L - \frac{(1.16 \cdot 10^{-7}) \ F}{\sqrt{L}} - \frac{4.72 \sqrt{F}}{L} - (5.51 \cdot 10^{-5}) \ \frac{F L}{\sqrt{L}} + (5.63 \cdot 10^{-5}) \ \frac{F L}{\sqrt{F}} + (4.09 \cdot 10^{-4}) L^2 + \frac{0.387}{\sqrt{L}} + 0.147 \sqrt{L} + \frac{8.71 L^2}{F} + \frac{1.08 \cdot 10^7}{F^2 L^2} - \frac{4.18 \cdot 10^{13}}{F^4 L^4}$</td>
</tr>
</tbody>
</table>
The final model chosen is the following twelve-term model:

\[ \hat{r}(L, F) = 0.201 \log(0.2 \cdot L) + (1.82 \cdot 10^{-13}) F^2 \cdot L - \frac{(1.16 \cdot 10^{-3}) F}{\sqrt{L}} - (5.51 \cdot 10^{-5}) F \cdot L - (5.63 \cdot 10^{-5}) \frac{L}{F} - (4.09 \cdot 10^{-4}) \frac{L}{F^2} - (4.72 \cdot 10^{-13}) \frac{L}{F^4} \]

\[ + 0.387 \cdot 10^{-3} \cdot \sqrt{L} + 0.147 \cdot 10^{-13} \cdot \sqrt{L} + 1.08 \cdot 10^{-7} \cdot L^2 - 4.18 \cdot 10^{-13} \cdot L^4 \]

Figure 14: Progression through the model building and sizing step for Iteration 7

Once this model has been generated it can be used to analyze the tradeoffs among bed depth and cooling water flowrate and percentage of CO2 removed from the entering gas. Figure 15 presents the results of the pareto analysis showing the tradeoffs between cost and environmental impacts. This curve was generated by solving an algebraic optimization model using the surrogate model of \( r \) and the algebraic model of \( COE \) with a weighted objected function of cost and environmental impact. For more details on the form of the increased cost of electricity to the consumer \( COE \), see [1]. The red line shows the pareto curve generated by solving a nonconvex nonlinear weighted objective function problem using BARON. For every given point on that line, there is no bed length or cooling water flow that could increase \( r \). To verify this curve, the 394 test points are plotted on the same axis, each of these points are represents a feasible process while most represent suboptimal processes since either a reduction of costs or an increase in CO2 removed is possible. The algebraic model derived by ALAMO can now be used not only for optimization but also for uncertainty quantification via thousands of simulations that consume a fraction of the CPU time to simulate the original model.

6 Conclusions

Simulation-based optimization provides a rich avenue for defining design parameters according to some cost function. As the number of decision variables increases, current derivative-free methods become less effective. We have presented a novel algorithm for use in simulation-based optimization. Surrogate
models of components of more computationally complex simulations are tailored for accuracy, ease of optimization, and simulation cost reduction. These models are built using a reformulation of the generalized best subset method to avoid the costly combinatorics of full enumeration, while maintaining the high accuracy of this method. At the same time, the statistical theory used to avoid overfitting a model is also used to ensure simple, compact models which aid in the eventual use of the surrogates in optimization. A new active sampling method, EMS, has been shown to improve the quality of sampled points. This was seen over several different modeling methods.

We can conclude that, if simulation is needed to optimize a system or process, surrogate models can be accurately and efficiently abstracted for the purpose of algebraic optimization with flexible objectives and constraints using the method outlined in this paper. The proposed methodology is equally applicable for fitting experimental data as well.

As part of the National Energy Technology Laboratory’s Regional University Alliance (NETL-RUA), a collaborative initiative of the NETL, this technical effort was performed under the RES contract DE-FE0004000, as part of the Carbon Capture Simulation Initiative.

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect
those of the United States Government or any agency thereof.
References


36


