Learning Surrogate Models for Simulation-Based Optimization

Alison Cozad and Nikolaos V. Sahinidis
National Energy Technology Laboratory, Pittsburgh, PA 15236

Dept. of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213

David C. Miller
National Energy Technology Laboratory, Pittsburgh, PA 15236

DOI 10.1002/aic.14418
Published online March 13, 2014 in Wiley Online Library (wileyonlinelibrary.com)

A central problem in modeling, namely that of learning an algebraic model from data obtained from simulations or experiments is addressed. A methodology that uses a small number of simulations or experiments to learn models that are as accurate and as simple as possible is proposed. 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. Automated learning of algebraic models for optimization (ALAMO), the computational implementation of the proposed methodology, along with examples and extensive computational comparisons between ALAMO and a variety of machine learning techniques, including Latin hypercube sampling, simple least-squares regression, and the lasso is described. © 2014 American Institute of Chemical Engineers *AIChE J*, 60: 2211–2227, 2014

Keywords: design (process simulation), optimization, machine learning

Introduction

Chemical process simulation and computational fluid dynamic methods have been used industrially and academically to design and test systems and processes.[1–4] 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 prediction but can impose challenges when used in an optimization or design setting.[5–7] This article considers the optimization of processes via black-box function evaluators, including simulations and experiments. The general optimization problem we address is

$$
\begin{aligned}
\min \quad & f(x) \\
\text{s.t.} \quad & g(x) \leq 0 \\
& x \in A \subset \mathbb{R}^n
\end{aligned}
$$

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 concerning operating conditions and equipment geometry to discrete decisions about process alternatives and flow sheet 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 available directly 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 gives rise to three primary challenges. First, the standard approach to optimization, utilizing derivative-based or algebraic solvers (e.g., CONOPT,[8] IPOPT,[9] and SNOPT[10]), requires the use of derivative information. However, the objective and/or constraint set must be treated as black boxes as algebraic models and derivatives are not directly available for many simulation packages. Such simulators often incorporate proprietary software, numerical integrators, lookup tables, and other algorithmic constructs whose algebraic forms are unavailable for optimization. More advanced global optimization methods, such as BARON,[11] require an algebraic functional form to locate and certify a globally optimal solution. In either case, algebraic forms for each function f and g are required to first locate and then classify feasible and optimal decision variables. Some standard solvers are capable of optimizing in the absence of an algebraic model if derivatives can be evaluated or approximated. In fact, IPOPT has been used to optimize ASPEN-based simulations directly.[12] However, as simulations become more complex, the reliability of the simulator often degrades. Thus, direct optimization is impossible without specialized algorithms designed to recover from simulator convergence failures. Even perfectly robust simulations exhibit a third challenge: 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.[13]

Correspondence concerning this article should be addressed to N. V. Sahinidis at sahinidis@cmu.edu.

Derivative-free optimization (DFO) offers a class of algorithms designed to solve optimization problems when derivatives are unavailable, unreliable, or prohibitively expensive to evaluate.[14,15] These solvers attempt to locate 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 optimal solutions when the number of degrees of freedom exceeds about 10, even in the absence of constraints and integer variables, as shown in a recent computational study.[15]

To overcome the challenges of simulation-based optimization, significant work has been done to generate surrogate models [known in some fields as metamodels or reduced-order models (ROMs)] of the black-box functions $f(x)$ and/or $g(x)$.[16,17] Most commonly, these methods are applied to purely continuous problems. After generation, the abstracted models can be optimized using traditional algebraic or derivative-based solvers. Previous work incorporates existing techniques from machine learning and statistics; the resulting surrogate-based methods are categorized by modeling method. Reduced-order modeling, the production of a low-dimensional system (i.e., ROM) that has similar response characteristics to the high-fidelity simulation or system,[18] is the most commonly used surrogate modeling technique. The goal of reduced-order modeling is to create an approximation of a black box that necessitates far less CPU time for each function evaluation. In the context of derivative-based optimization, these ROMs are required to have a compact and algebraic form that can be exploited by standard solver packages.

Most often, a single model of the objective function is approximated before optimization; in a few cases, the constraint set is modeled as well. Additionally, some existing techniques disaggregate a black-box simulation into distinct blocks and model each block separately before optimization, ensuring that all relevant connectivity variables are 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[19–21] or the disaggregated process.[13] Palmer and Realff[19] have considered the indirect optimization of steady-state simulators using kriging surrogate models. In the same vein, David and Ierapetritou[21] use full-process kriging models to locate global surrogate model solutions and refine them using local response surfaces around the optima. To address uncertainty concerns in black-box systems, Huang et al.[20] have used kriging models on full processes. Caballero and Grossmann[13] have investigated disaggregated (modular) flow sheet optimization using kriging models to represent process units with low-level noise. Recent work by Henao and Maravelias[22] has shown success in modeling individual chemical process units using artificial neural networks.

Previous work has focused primarily on developing models that are highly accurate. As a result, unless physical simplifications are available, ROMs 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. Because the final purpose of our surrogate models is algebraic optimization, we strive to identify surrogates composed of functional forms that can be easily incorporated into larger mathematical programs without the difficulties imposed by the inherent complexity of standard ROMs.

To address the black-box nature of these simulations as well as the cost and limited robustness of each function evaluation, we have developed a novel surrogate modeling method. To promote simulation robustness, either a single unit or a small set of units is considered. If the existing simulation is complex, such as a complete flow sheet, disaggregation into smaller sections is advantageous. Subsequently, using an adaptive sampling procedure, low-complexity algebraic models are built, tested, exploited, and improved using a combination of derivative-based and DFO solvers, machine learning, and statistical techniques. Surrogate models generated using these techniques can be used in an algebraic optimization framework with flexible objective functions and additional constraints.

We developed automated learning of algebraic models for optimization (ALAMO), a software package designed to automate the proposed methodology. ALAMO interfaces with a user-defined simulator and problem space to iteratively model and interrogate the simulation. Consequently, our flexible implementation is able to identify accurate, low-complexity algebraic models that approximate a variety of high-fidelity systems. Similar existing modeling packages, such as SUorrogate MOdeling lab toolbox,[23] fail to generate surrogate models with sufficiently low complexity. Eureqa[24] can be used to search for low-complexity models; however, it operates on a fixed dataset and can often lead to comparatively complex functional forms. The proposed methodology allows ALAMO to generate compact models that improve and validate the surrogate models by adaptively sampling the simulation.

Previous works in the process systems engineering literature have approached simulation-based optimization by relying on existing modeling schemes, mostly kriging and neural network modeling, to build surrogate process models that can be optimized with derivative-based optimization techniques. In the current article, we depart from the use of existing modeling methodologies. The primary contribution of this work is to introduce a novel model-building methodology that identifies highly accurate surrogate models tailored for optimization tractability.

The remainder of this article is organized as follows. In the next section, we discuss the proposed model-building strategy in detail. To better explain the modeling steps involved, we include an illustrative example. Subsequently, we present computational results to evaluate the accuracy and efficiency of the proposed approach and compare our strategy with common surrogate modeling techniques in the machine learning literature. Finally, the proposed methodology is demonstrated on an industrial case study that quantifies the environmental and economic trade-offs of postcombustion carbon capture systems.

Proposed Methodology

To address challenges commonly associated with black-box simulation robustness, the process may be disaggregated into smaller process groups that contain one or more process units. This step enables access to more sophisticated optimization models, including superstructure optimization and more complex problem topologies, that arise from this modular structure. Last, models generated from less complex systems or smaller process blocks are generally more accurate than those generated for larger process systems. 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 approximating relevant responses for each block. The responses z_k, $k \in \mathcal{K}$, (outlet material and energy stream properties, efficiencies, design requirements and so forth) are modeled as a function $\hat{z}_k(x)$ of input variables x_d, $d \in \mathcal{D}$, which become optimization decision variables (inlet material and energy stream properties, operating conditions, unit geometry and so forth). We assume that the problem domain is bounded for each input variable. The surrogate models for each block can be combined with an algebraic objective, design constraints, and material and energy 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 (DOEs) is generated over the problem space and the simulation is queried at these points. In the scope of this work, the specific DOEs used does not play a strong role in the final solution. This is because the initial sample set is small and adaptively improved as the procedure progresses, leaving a dataset that bears little resemblance to the initial DOEs. In the included example cases and experimental studies, we use either Latin hypercube sampling[25] or a two-level factorial design[26] for this step.

Next, we build a simple, algebraic model using this initial training dataset. The empirical model error, that is, the deviation of the model from the data, can be calculated using standard statistical techniques. However, the true error, that is, the deviation of the model from the true system, is unknown. Moreover, we have no quantification of model accuracy and have not demonstrated a sufficient sampling of the problem space. The current surrogate model is tested subsequently against the simulation using an adaptive sampling technique that we call error maximization sampling (EMS). If the sampling technique discovers model inconsistency larger than a specified tolerance, the 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 outlines the algorithms and strategies used to generate accurate, low-complexity surrogate models and refine them iteratively through adaptive sampling techniques.

Surrogate model generation

For the modeling problem, we have a set of N training points; each training point $i=1,\ldots,N$ has a set of input data x_{id}, $d \in \mathcal{D}$, 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 model the simulation accurately while maintaining adequate simplicity to ensure that the resulting optimization model is tractable in an algebraic optimization framework. For example, surrogate modeling techniques such as kriging[27,28] and artificial neural networks (ANNs)[29] 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 functional simplicity.

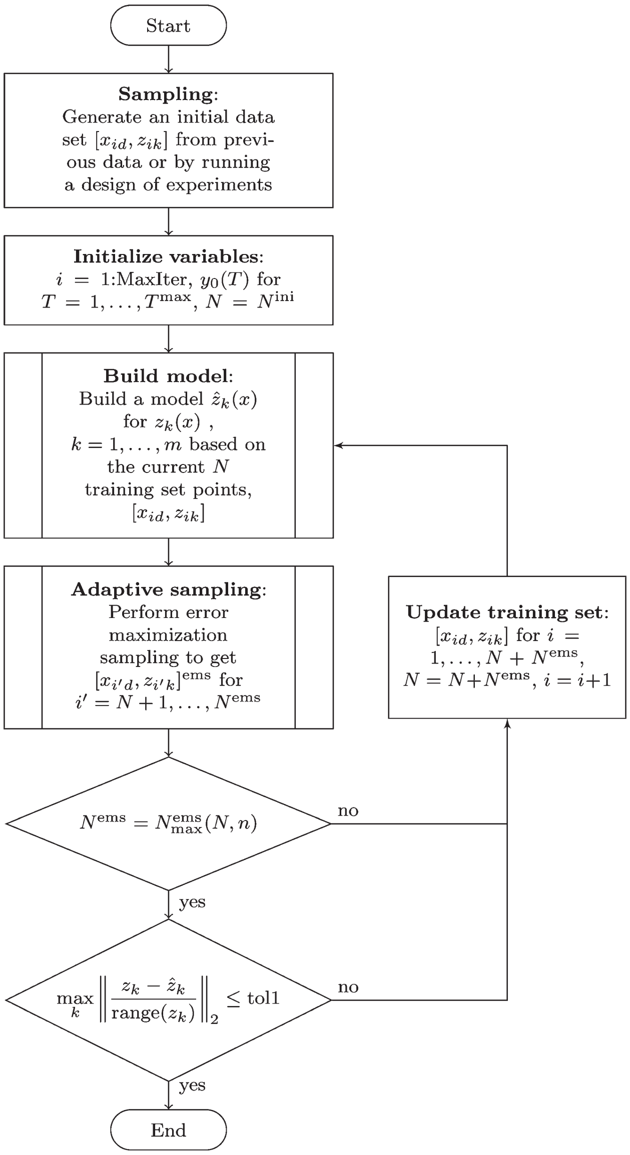
Start
Sampling:
Generate an initial data set $[x_{id}, z_{ik}]$ from previous data or by running a design of experiments
Initialize variables:
i = 1:MaxIter, $y_0(T)$ for $T = 1,\ldots,T^{\max}$, $N = N^{\text{ini}}$
Build model:
Build a model $\hat{z}_k(x)$ for $z_k(x)$, $k = 1,\ldots,m$ based on the 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:
$[x_{id}, z_{ik}]$ for $i = 1,\ldots,N+N^{\text{ems}}$, $N = N+N^{\text{ems}}$, $i = i+1$
$N^{\text{ems}} = N^{\text{ems}}_{\max}(N,n)$
no
yes
$\max_k \left\| \frac{z_k - \hat{z}_k}{\text{range}(z_k)} \right\|_2 \le \text{tol1}$
no
yes
End

Figure 1. Algorithmic flowchart.

We strive to strike a balance between model accuracy and optimization tractability, but knowledge of this key trade-off is limited because the functional form of the true system is unknown. To address this uncertainty, we identify combinations of simple basis functions that define a low-complexity, accurate functional form for each response. The simple basis functions $X_j(x)$, $j \in \mathcal{B}$, are selected from first principles relationships, physical or engineering insights, or statistical fitting functions. Additionally, a sufficiently small subset of the functional forms available to kriging or ANNs could be utilized as potential basis functions used to generate 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 α and γ. Generally, we choose values for α and γ that result in either physically reasonable basis functions or common statistical fitting functions (e.g., $\alpha=\{\pm 0.5, \pm 1, \pm 2, \pm 3, \pm 4\}$ and $\gamma=\{0.1, 1, 10\}$). By choosing diverse and varied terms, we

Table 1. List of Potential Simple Basis Function Forms

Category		$X_j(x)$
I	Polynomial	$(x_d)^\alpha$
II	Multinomial	$\prod_{d \in \mathcal{D}' \subseteq \mathcal{D}} (x_d)^{\alpha_d}$
III	Exponential and logarithmic forms	$\exp\left(\frac{x_d}{\gamma}\right)^\alpha$, $\log\left(\frac{x_d}{\gamma}\right)^\alpha$
IV	Expected bases	From experience, simple inspection, physical phenomena and so forth

expect to provide a sufficient set of potential basis functions, even if these specific functional forms do not match the underlying functional form exactly. For example, if we model $z(x)=2x^{3/2}+1$ over $x \in [0,1]$ we can generate a quadratic surrogate model, $\hat{z}(x)=1.3x^2-0.35x+2$, that has an average error of 1%. By doing this, we are able to model a wide variety of unknown functional forms using a small, but flexible, set of basis functions.

The resulting surrogate model is a linear combination of nonlinear basis functions as follows

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

where the jth basis function is multiplied by a corresponding coefficient β_j.

The ordinary least-squares regression (OLR) problem

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

could be used to solve for the regression coefficients (model parameters), β, by minimizing the sum of the squared model error over the training data points i. In most cases, the complexity would be prohibitively high if we solve (2) to find the least-squares regression coefficients because most or all of the potential bases appear in the surrogate. Model reduction techniques available from 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 overfitting observed in the model by allowing 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.[30] 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[30] can be used to enumerate models for all possible combinations of the basis set, and then to choose the best subset of basis functions using a measure of the model fitness that is sensitive to overfitting. 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 prohibitively expensive for large basis sets. Recent work has seen the addition of graph theory,[31] branch and bound algorithms,[32] and QR decomposition[32] to enhance the scalability and reduce the required 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 overfitting. However, as we show in a subsequent section, the commonly used lasso regularization (L_1-norm penalty function)[33] 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

$$\begin{aligned} (\text{BS}) \qquad & \min_{\mathcal{S},\beta} \quad \Phi(\mathcal{S},\beta) \\ & \text{s.t.} \quad \mathcal{S} \subseteq \mathcal{B} \end{aligned}$$

where $\Phi(\mathcal{S},\beta)$ is a surrogate model goodness-of-fit measure for the subset of basis function $\mathcal{S}$ and regression coefficients β. Using (BS), the following surrogate model is generated using a subset of the basis functions

$$\hat{z}(x)=\sum_{j \in \mathcal{S}} \beta_j X_j(x) \tag{3}$$

Through a series of reformulation and simplification steps, we convert (BS) into a form that can be solved efficiently. First, we define a subset of basis functions using a vector of binary variables y to designate active and inactive bases. For each basis function $j \in \mathcal{B}$, if $j \in \mathcal{S}$, $y_j=1$; otherwise, $j \notin \mathcal{S}$ and $y_j=0$. Using this binary vector, Eq. 3 can be described over the full basis set $\mathcal{B}$

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

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

$$\begin{aligned} (\text{BS}\,1) \qquad & \min_{\beta,y} \quad \Phi(\beta,y) \\ & \text{s.t.} \quad y_j \in \{0,1\} \end{aligned}$$

At this point, it is beneficial to implement three reformulations to (BS1). To remove the complication of integer bilinear terms, we replace $y_j\beta_j$ with $2|\mathcal{B}|$ big-M constraints

$$\beta^{\text{l}} y_j \le \beta_j \le \beta^{\text{u}} y_j$$

that use lower and upper bounds, β^{l} and β^{u}, on β. These constraints force β_j to zero if $y_j=0$, while allowing β_j to take on a non-zero value within its bounds if $y_j=1$. The bounds on β are chosen using logic from the regularized regression technique: the lasso.[33] The lasso method penalizes or constrains a least-squares objective using the L_1 norm of the β vector. By extending this concept, we can use the solution to the OLR problem β^{OLR} to infer loose upper and lower bounds on each value of β. To do this, we set $\beta^{\text{l}} = -\sum_{j \in \mathcal{B}} |\beta_j^{\text{OLR}}|$ and $\beta^{\text{u}} = \sum_{j \in \mathcal{B}} |\beta_j^{\text{OLR}}|$.

The second reformulation stems from the observation that many goodness-of-fit measures can be decoupled into two parts: (a) model sizing and (b) basis and parameter selection, as follows

$$\min_{\beta,T,y} \Phi(\beta,T,y) = \min_{T} \left\{ \min_{\beta,y} \left[\Phi_{\beta,y}(\beta,y)|_T \right] + \Phi_T(T) \right\} \tag{4}$$

Here, $\Phi_T(T)$ and $\Phi_{\beta,y}(\beta,y)|_T$ denote the model sizing and basis/parameter selection contributions to the information criterion, respectively. Hence, we can pose the best subset selection minimization problem as a nested minimization problem, where the inner minimization determines the basis functions and coefficients and the outer minimization defines the complexity of the model. This results in the following problem for some goodness-of-fit measure

$$
\begin{aligned}
&\min_{T\in\{1,\ldots,T^u\}} \quad \left[\Phi_{\beta,y}(\beta,y)|_T\right]+\Phi_T(T)\\
&\text{s.t.}\\
&\qquad \min_{\beta,y} \quad \Phi_{\beta,y}(\beta,y)|_T\\
&\qquad \text{s.t.} \qquad \sum_{j\in\mathcal{B}} y_j=T\\
&\qquad\qquad\qquad \beta^{\mathrm{l}}y_j \le \beta_j \le \beta^{\mathrm{u}}y_j \quad j\in\mathcal{B}\\
&\qquad\qquad\qquad y_j=\{0,1\} \quad j\in\mathcal{B}
\end{aligned}
$$

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 overfitting. The goodness-of-fit measure should reflect the true model error and not simply the empirical error. As mentioned previously, the empirical error of a model is the inaccuracy between the model and the data points used to build the surrogate. As a properly built model increases in complexity, the empirical error of a properly trained model is nonincreasing. 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 the majority portion of the data while reserving a 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, that is, an independent dataset. Generally, this is done several times by reserving different portions of the data for validation. 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 overfitting. Information criteria are able to account for the model complexity directly, unlike cross-validation. These measures are tied to the maximum likelihood method of model parameter estimation[34]; one such case is linear regression given an assumption of normal distribution on the error. Information criteria are comprised of several alternatives for order-selection rules including Akaike information criterion,[35] Bayesian information criterion,[34] generalized information criterion[34] and so forth. Each information criterion gives a measure of the accuracy vs. the complexity of a model.[35,36] Due to the large number of basis functions available to the model, the goodness-of-fit measure used here is the corrected Akaike information criterion[37]

$$
\begin{aligned}
\mathrm{AIC}\,c(\mathcal{S},\beta)=&N\log\left(\frac{1}{N}\sum_{i=1}^{N}\left(z_i-\sum_{j\in\mathcal{S}}\beta_jX_{ij}\right)^2\right)\\
&+2|\mathcal{S}|+\frac{2|\mathcal{S}|(|\mathcal{S}|+1)}{N-|\mathcal{S}|-1}
\end{aligned}
\qquad (5)
$$

which adjusts the Akaike information criterion to account for large basis 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.

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

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

Second, 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

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

and then replace each instance of $|w|$ in SE by w' and add constraints $w'\ge w$ and $w'\ge -w$ in the formulation. To retain the least-squares representation of the resulting coefficients, we use the stationarity condition with respect to the parameters β

$$
\frac{d}{d\beta_j}\sum_{i=1}^{N}\left(z_i-\sum_{j\in\mathcal{S}}\beta_jX_{ij}\right)^2 \propto \sum_{i=1}^{N}X_{ij}\left(z_i-\sum_{j\in\mathcal{S}}\beta_jX_{ij}\right)=0, \quad j\in\mathcal{S} \qquad (6)
$$

Equation 6 is used as a set of big-M constraints to define the basis coefficients

$$
-U_j(1-y_j)\le\sum_{i=1}^{N}X_{ij}\left(z^i-\sum_{j\in\mathcal{B}}\beta_jX_{ij}\right)\le U_j(1-y_j)
$$

where we calculate U_j to be the maximum of $\sum_{i=1}^{N}X_{ij}\left(z^i-\sum_{j\in\mathcal{B}}\beta_jX_{ij}\right)$ using the upper and lower bounds of β. Thus, we select active basis functions based on linear error and the value of the regression parameters based on a squared error.

The reformulations and simplifications described above result in the following MILP

$$
\begin{aligned}
\text{(M)}\quad &\min \sum_{i=1}^{N}w_i\\
&\text{s.t. } w_i\ge z_i-\sum_{j\in\mathcal{B}}\beta_jX_{ij}, \quad i=1,\ldots,N\\
&w_i\ge\sum_{j\in\mathcal{B}}\beta_jX_{ij}-z_i, \quad i=1,\ldots,N\\
&\sum_{j\in\mathcal{B}}y_j=T\\
&-U_j(1-y_j)\le\sum_{i=1}^{N}X_{ij}\left(z^i-\sum_{j\in\mathcal{B}}\beta_jX_{ij}\right)\le U_j(1-y_j), \quad j\in\mathcal{B}\\
&\beta^{\mathrm{l}}y_j\le\beta_j\le\beta^{\mathrm{u}}y_j, \quad j\in\mathcal{B}\\
&y_{kj}\in\{0,1\}, \quad j\in\mathcal{B}\\
&\beta_j^{\mathrm{l}}\le\beta_j\le\beta_j^{\mathrm{u}}, \quad j\in\mathcal{B}
\end{aligned}
$$

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 solve the best subset problem efficiently and to generate accurate low-complexity models. Though this technique is far more efficient and scalable than the enumerative best subset method, the complexity of these problems increases combinatorially with the number of inputs. If we allow for, say, five levels of α for polynomials and multinomials, up to pairwise multinomial terms with equal exponents, and $\gamma=1$ (See Table 1), a two-, four-, and ten-dimensional problem will have 20, 59, and 246 potential basis functions, respectively. Despite the fact that MILP problems are NP-hard,[38] we have solved instances with as many as 14 inputs successfully.[39]

A detailed outline of the algorithm used to generate the most accurate and simple model given a fixed dataset is included in Algorithm 1.

Adaptive sampling

Adaptive sampling, active learning, or supervised learning is a means of acquiring information about the response surface or process by querying the system at desired input levels. Through the careful selection of sample points, more accurate models can be generated with less sample information. Preferably, perfect information would be used to train a model. In reality, however, computational resources and available function evaluations are limited. Without prior knowledge of a black-box system, it is impossible to know how much information is required or where data sample points should be located *a priori*. Yet, techniques exist to utilize insights gained through previous system knowledge or model structure to sample a system selectively so as to balance the need for information with the computational cost of that information.

Before we examine our approach, it is worth noting that, generally, a DOEs can be classified into two categories: single pass and iterative. Single pass DOE methods, like fractional designs,[26] Latin hypercubes,[25] and orthogonal arrays,[40] first generate a set of design points, evaluate these points, and then move on to a modeling stage. These methods are computationally efficient and are simple to implement, but the resulting surrogate models often lack fidelity. In such cases, single pass models can be refined using an iterative approach. Iterative approaches use both the current dataset and the current model to locate and sample complicating areas either by performing exploration-based or exploitation-based methods.[41,42] Exploration-based methods aim to sample the problem space[43] evenly. Exploitation-based techniques sample in difficult-to-model areas such as points of high nonlinearity or discontinuity. If the modeling method provides error estimates (such as kriging[27]) or there is another available error metric, these estimates can be used to locate areas of high uncertainty.[44,45]

Another relevant field of active learning that can be used in a single pass or iterative implementation is optimal DOEs.[46] This approach is motivated by knowledge of data modeling techniques. It is possible to estimate the variance of the final model parameters β 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 model parameters.[34]Optimal DOEs methods attempt to maximize a function of this matrix, or minimize its inverse. A key concept in our approach is the flexibility afforded through the selection of active and inactive basis functions. Because the final model's functional form is flexible, we have no *a priori* knowledge of basis function activity. Consequently, the strength of the optimal DOEs method is likely to be diluted by the presence of numerous unused basis functions.

Algorithm 1: Build model

```
Given a training set of dependent and independent data points $[x_{id}, z_{ik}]$ for $i=1,\dots,N, d=1,\dots,n, k=1,\dots,m$; 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 $i=1,\dots,N$ and $j \in \mathcal{B}$
Initialize a high maximum error allowed at each point, $e^{\max}$
for $k \leftarrow 1 \text{to}\, m$ do
  Generate a set of bounds for $\beta_j$ and $U_i$
  Calculate the maximum terms allowed, maxTerms $\leftarrow \max(1, \min(N, |\mathcal{B}|))$
  for $t \leftarrow 1$to maxTerms do
    Solve (M) for $T=t$ to determine the optimal $t$ basis functions and $\beta$
    Store $\beta(t)$ and $y(t)$
    Compute AIC$c(t)$
    if $t > 1$ then
      if $AIC\,c(t) > AIC\,c(t-1)$ then Information criterion worsened
        Set the final model complexity $T^{\mathrm{f}}=t-1$
        break
      else if $\frac{AIC\,c(t)-AIC\,c(t-1)}{AIC\,c(t-1)} \leq$ tol2 then
        Set the final model complexity $T^{\mathrm{f}}=t$
        break
      else if $\frac{1}{N}\|z_k-\hat{z}_k\|^{\frac{1}{2}} \leq$ tol3 then
        Set the final model complexity $T^{\mathrm{f}}=t$
        break
      end if
    end if
    Update error bounds $e^{\max} \leftarrow \frac{1}{N}\sum_{i=1}^{N}\left|z_i-\sum_{j\in\mathcal{B}}\beta_j X_{ij}\right|$
    end for
  end for
  return final values for $\beta=\beta(T^{\mathrm{f}})$ and $y=y(T^{\mathrm{f}})$
```

Instead, we interrogate the simulation in locations of model inconsistency using EMS. Doing this provides us with 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

$$\max_{x^l \le x \le x^u} \left(\frac{z(x)-\hat{z}(x)}{z(x)} \right)^2 \tag{7}$$

Although 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 DFO 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.[47] As shown recently,[15] these methods are most effective in low-dimensional cases. Thus, the technique of decomposing a large simulation into lower-dimensional blocks provides favorable conditions. As the quality of DFO solutions may suffer in higher-dimensional spaces, we recommend using less than 20 model variables to ensure strong sampling. 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 a prespecified tolerance, the model is retrained using an updated training set. If the error estimate is below tolerance, then the proposed approach has converged to a final surrogate model.

Illustrative Example

To better illustrate the proposed methodology, we provide a simple example modeling steam density, ρ, 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 model, $\hat{\rho}(Q)$ in kg/m^3, as a function of heat duty from 13,000–40,000 W. The functional form includes basis functions of the form shown in Table 1, where $\alpha=0, \pm\frac{1}{3}, \pm\frac{1}{2}, \pm 1, \pm 2, \pm 3$ and $\gamma=10,000\text{W}$. This leads to a potential functional form shown below with 13 basis functions

$$\hat{\rho}(Q)=\beta_0+\beta_1 Q+\frac{\beta_2}{Q}+\beta_3 Q^2+\frac{\beta_4}{Q^2}+\beta_5 Q^3+\frac{\beta_6}{Q^3}+\beta_7\sqrt{Q}$$

$$+\frac{\beta_8}{\sqrt{Q}}+\beta_9\sqrt[3]{Q}+\frac{\beta_{10}}{\sqrt[3]{Q}}+\beta_{11}\mathrm{e}^{\frac{\mathrm{Q}}{10,000}}+\beta_{12}\ln\frac{\mathrm{Q}}{10,000}$$

A process schematic is included in Figure 2, where water enters a 1-atm flash drum at 1 atm and 298 K. The source water stream is flashed into vapor and liquid products. To facilitate the flash, heat is added to the drum. Our goal is to find an accurate surrogate model describing the relationship between steam density and added heat using only the basis functions required to fit the simulated system. This is done using a minimal but flexible dataset.

The algorithm begins by using a Latin hypercube to select an initial set of data points over the range of the input variables. For illustration, we consider two data points over $Q\in[13,000\text{W}, 40,000\text{W}]$. During each iteration, a model is built using a subset of the basis functions shown above that

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, $[x^{\min}, x^{\max}]$; the minimum number of sample points, $N^{\text{EMS}}_{\min}$; the maximum number of sample points as a function of N, $N^{\text{EMS}}_{\max}(N)$; and a tolerance on the maximum error, tol4, are given.

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

```
Initialize the number of error maximization points added, $N^{\text{EMS}} \leftarrow 0$
while $N^{\text{EMS}} \le N^{\text{EMS}}_{\max}$ do
  Using a black-box solver (we call it dfo([objective function], [initial objective values], [initial $x$ values], $x^{\min}, x^{\max}$, [requested points])) that meets the requirements listed, request new sample points $x^{\text{req}} \leftarrow \text{dfo}(E(x), E_i, x_i, x^{\min}, x^{\max}, N^{\text{req}})$

  for $i \leftarrow 1$ to $N^{\text{req}}$ do
    Sample simulation at $x_i^{\text{req}}$ to get $z_i^{\text{req}}$
    Append error values, $e_{N+i,k} \leftarrow e(x_i^{\text{req}}, z_{ik}^{\text{req}})$ and $E_{N+i} \leftarrow E(x_i^{\text{req}}, z_i^{\text{req}})$
    Update $N^{\text{EMS}} \leftarrow N^{\text{EMS}}+1$ and $N \leftarrow N+1$
    if $N^{\text{EMS}} \ge N^{\text{EMS}}_{\min}$ then
      if $\max_{ik}(e_{ik}) \ge \text{tol4}$ then
        return $x, z, N^{\text{EMS}}$
      end if
    end if
  end for
end while
```

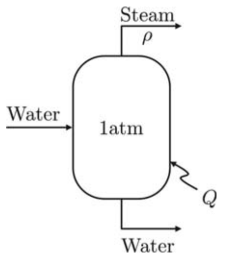

Figure 2. Diagram of flash drum.

models the information in the current dataset 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 $N_{\min}^{\text{EMS}}=1$ and $N_{\max}^{\text{EMS}}(N)$ to be the greater of 20% of the current dataset size and $|\mathcal{D}|+10=11$.

To further 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 nine 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 AIC*c* is calculated for the one-term model. Afterward, the best two-term model is identified. In this case, the best two-term model is chosen from $\begin{pmatrix}13\\2\end{pmatrix}=156$ basis combinations. This is done using the MILP model (M). From here, the AIC*c* of the two-term model is compared to the one-term model (see Figure 3 for AIC*c* vs. model size for the previous iteration). As AIC*c* decreases with the addition of a second term, there is sufficient evidence to increase the model size to two terms and we test a three-term model for a better fit. This continues until the AICc worsens, as it does going from a five-term to a six-term model. Table 2 shows the models found for each model size.

As is often the case, increasing the number of terms does not result in appending a single basis function to the previous model. Instead, the activity of many bases is changed as model complexity is allowed to increase. This demonstrates the benefits afforded by leveraging the synergistic effects of basis function combinations, something that is not possible using standard statistical techniques such as backward elimination or forward selection.

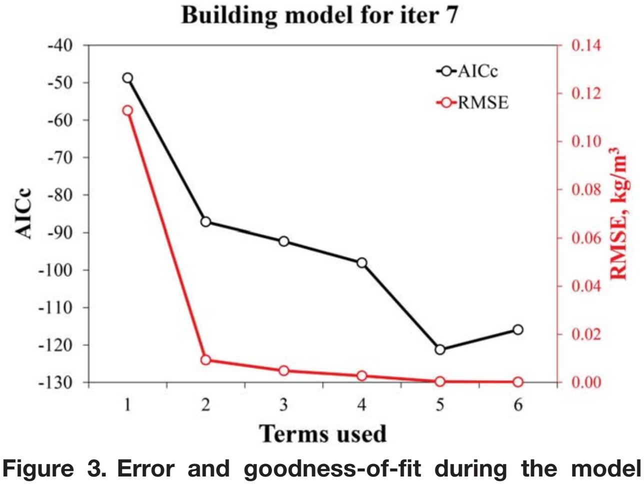

Figure 3. Error and goodness-of-fit during the model building for Iteration 7.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Table 2. Surrogate Models Built at Increasing Complexity for Iteration 7 of the Flash Drum

Terms Allowed	Surrogate Model
1	$\frac{1.524\cdot 10^{11}}{Q^3}$
2	$\frac{1.212\cdot 10^{11}}{Q^3}+0.07519$
3	$0.1086\ \ln(0.0001Q)-\left(4.754\cdot 10^{-10}\right)Q^2+\frac{1.348\cdot 10^{11}}{Q^3}$
4	$2.339\sqrt[3]{0.0001Q}-1.385\ln(0.0001\mathrm{Q})-\frac{9518.0}{\mathrm{Q}}+\frac{2.075\cdot 10^{11}}{\mathrm{Q}^3}$
5	$80.7\sqrt[3]{0.0001Q}-\frac{39.0}{\sqrt{0.0001\mathrm{Q}}}-41.54\ln(0.0001\mathrm{Q})$ $+\frac{3.911\cdot 10^{11}}{Q^3}-\left(1.571\cdot 10^{-13}\right)Q^3$
6	$\frac{27.06}{\sqrt{0.0001Q}}-0.000908Q+84.94\sqrt{0.0001Q}-91.6\sqrt[3]{0.0001Q}$ $-\frac{1.159\cdot 10^5}{Q}+\frac{4.768\cdot 10^{11}}{Q^3}$

The mean squared error and AIC*c* for each model size is shown in Figure 3. It is important to note that, even though the model error decreases from five to six terms, the information criterion shows that this increase in accuracy is not worth the added complexity of a sixth term. Surrogate models at each iteration are built similarly.

Recalling Figure 1, we can examine the progress throughout the rest of the algorithm. As mentioned previously, a number of new simulation points are selected using an adaptive sampling technique called error maximization. New points are selected to maximize the current candidate's modeling error. Next, these points are simulated and compared with the current model to determine the actual model error. Each iteration is terminated in this example when (a) the error exceeds a tolerance of 1%, normalized by the range of ρ or (b) the maximum number of points sampled, $N_{\max}^{\text{EMS}}(N)$, has been reached. In the case of (a), a new model is rebuilt using previous 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. If this error exceeds the specified tolerance of 0.5%, the training set is updated and a new iteration begins. As these 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 case, the basis functions are fixed, the sparse β vector is updated using a simple least-squares estimation with the latest dataset, 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 dataset 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 provide valuable information during 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 during execution of the algorithm. The models for Iterations 1, 3, 5, and 7 are shown alongside the true simulation curve. The training dataset, shown in white, and the newly sampled EMS points,

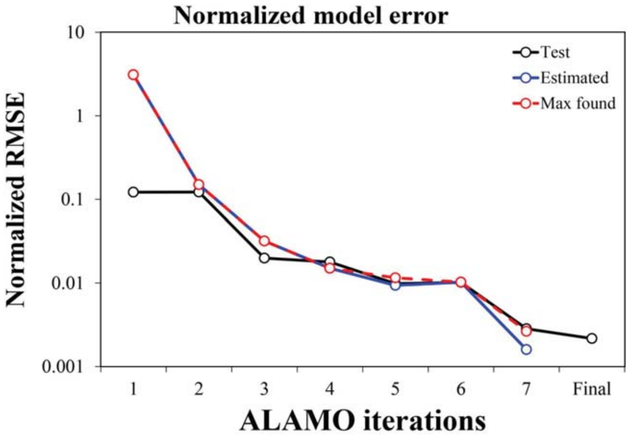

Figure 4. Estimated, maximum found, and test model errors found building $\hat{\rho}(Q)$.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

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

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

The terms in the model are chosen using a training set of nine points and the coefficients are re-evaluated using the full dataset of 20 points.

Implementation and Computational Results

ALAMO combines a tailored model generation solver with an EMS routine. 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 the black-box solver Stable Noisy Optimization by Branch and FIT (SNOBFIT)[48] for adaptive sampling. MINQ[49] is used to solve SNOBFIT's quadratic programming subproblems. To facilitate

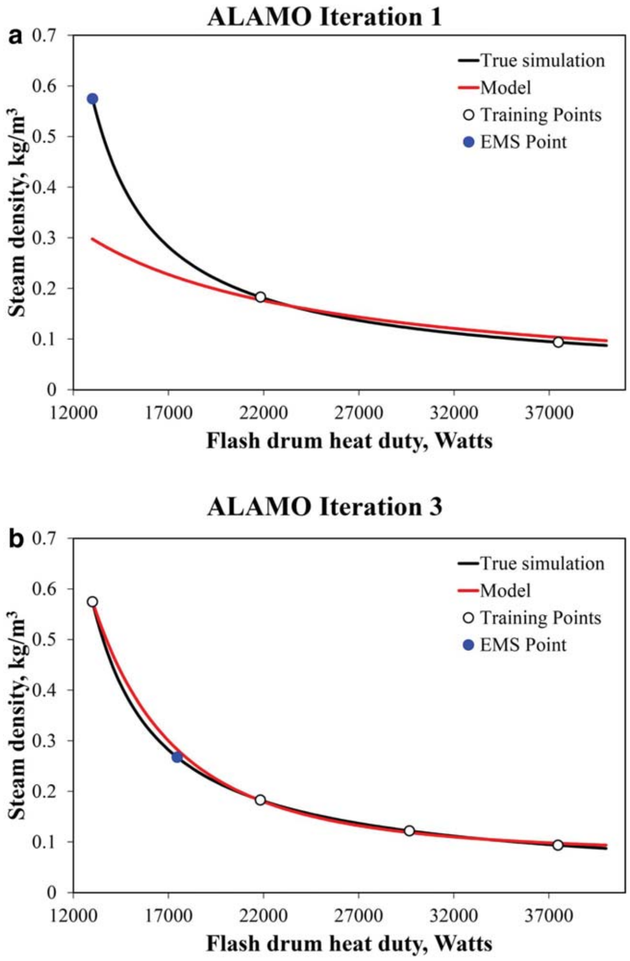

Figure 5. Comparison of the true simulated data and the current surrogate models for Iterations 1 and 3 along with current and EMS datasets.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

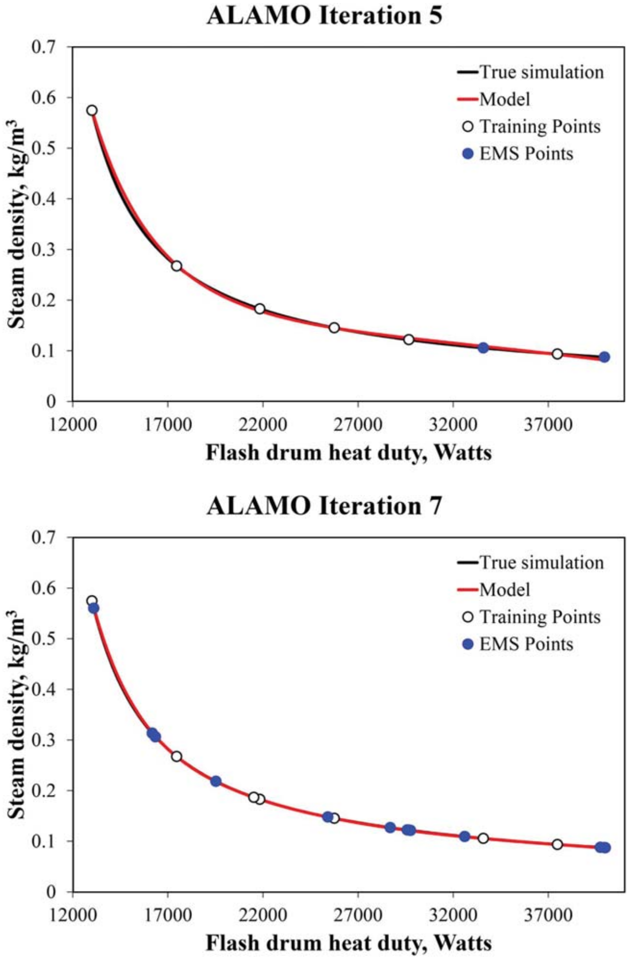

Figure 6. Comparison of the true simulated data and the current surrogate models for Iterations 5 and 7 along with current and EMS datasets.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Table 3. Surrogate Model Information from Modeling the Flash Drum

Iter.	EMS Points	Terms	Model
1	1	1	$\frac{3867.0}{Q}$
2	1	1	$\frac{1.299\cdot 10^{12}}{Q^3}$
3	1	2	$\frac{1.098\cdot 10^{12}}{Q^3}+0.07649$
4	1	2	$\frac{1.094\cdot 10^{12}}{Q^3}+0.07398$
5	2	3	$0.4522\ln(0.0001Q)-\left(1.418\cdot 10^{-5}\right)Q+\frac{1.409\cdot 10^{12}}{Q^3}$
6	1	3	$0.4094\ln(0.0001Q)-\left(1.262\cdot 10^{-5}\right)Q+\frac{1.39\cdot 10^{12}}{Q^3}$
7	11	5	$58.9\sqrt[3]{0.0001Q}-\frac{60.94}{\sqrt{0.0001\mathrm{Q}}}-45.15\ln(0.0001\mathrm{Q})+\frac{3.524\cdot 10^{12}}{\mathrm{Q}^3}-\left(6.259\cdot 10^{-15}\right)\mathrm{Q}^3$
Final	NA	5	$62.37\sqrt[3]{0.0001Q}-\frac{64.53}{\sqrt{0.0001\mathrm{Q}}}-47.81\ln(0.0001\mathrm{Q})+\frac{3.659\cdot 10^{12}}{\mathrm{Q}^3}-\left(6.576\cdot 10^{-15}\right)\mathrm{Q}^3$

comparisons with existing techniques, several more standard model generation alternatives are integrated into ALAMO including

1. 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.
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 been added to ALAMO as an alternative sampling method using Matlab's lhsdesign() function. In this section, we look at the model accuracy, quality of point sampling, and final model complexity 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 are considered. Test set A considers the problem of learning the functional forms of equations containing only terms that are present in the algorithm's basis function set. These functions are composed of basis functions that are available for surrogate modeling with two and three input variables. Basis functions from Table 1 are 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 are generated with varying complexity from 2 to 10 randomly chosen terms, where three functions are generated at each complexity. In addition, 18 three-dimensional functions are generated from two to seven randomly selected terms.

The 12 equations in test set B are generated using functional forms that are unavailable to the surrogate modeling method. These functional forms are included 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 $i,j\in\{1,2\}$.

Table 4. Functional forms for Test Set B

Function Type	Functional Form
I	$z(x)=\beta x_i^\alpha\exp(x_j)$
II	$z(x)=\beta x_i^\alpha\log(x_j)$
III	$z(x)=\beta x_1^\alpha x_2^\nu$
IV	$z(x)=\frac{\beta}{\gamma+x_i^\alpha}$

Each test function is 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 (SLH), a test function is modeled using the full algorithm a second model is generated subsequently using a SLH with the same number of data points. In this way, we are able to calculate the amount of information per

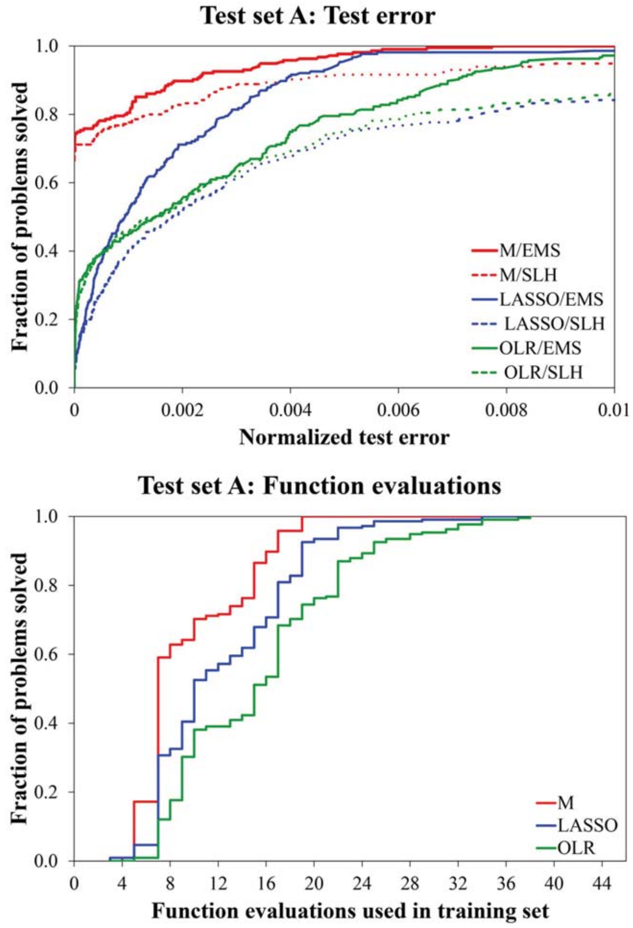

Figure 7. Top: Fraction of functions in test set A modeled within a normalized test error.

Bottom: Fraction of functions in test set A modeled within a specified number of function evaluations. As the EMS and SLH sampling schemes use the same number of data points, only the differences between models is important here. Note: EBS is not plotted since the CPU time was too great for several of the more complex problems. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

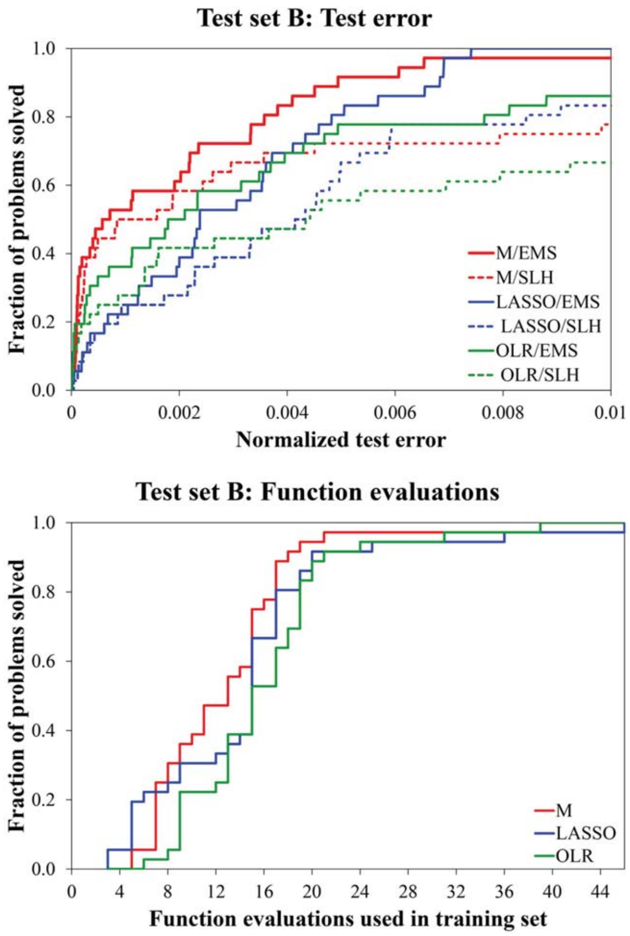

Figure 8. Top: Fraction of functions in test set B modeled within a normalized test error.

Bottom: Fraction of functions in test set B modeled within a specified number of function evaluations. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

data point extracted by the EMS sampling method compared to a Latin hypercube DOEs. The resulting normalized test error is calculated as follows

$$e_k^{\text{norm}} = \frac{1}{|\mathcal{T}|} \frac{\sqrt{\sum_{i \in \mathcal{T}} \left(z_{ik} - \hat{z}_k(x_i)\right)^2}}{\max\limits_{i \in \mathcal{T}} z_{ik} - \min\limits_{i \in \mathcal{T}} z_{ik}} \tag{8}$$

where e_k^{norm} is the normalized error for dependent variable z_k calculated from an independent set of 1000 data points $i \in \mathcal{T}$. The normalized error, number of basis function terms in the final model, and number of required function evaluations for these tests are shown in Figures 7 and 8 and Tables 5 and 6.

Figures 7 and 8 display the performance profile of each modeling and sampling combination for test sets A and B, respectively. Each profile is constructed by calculating the percentage of the total test set solved within a normalized test error, given as the x axis. For example, the proposed method using M/EMS is able to solve 80% of test set A to within 0.1% error as calculated using Eq. 8. Across both test sets, the proposed method, shown in solid red, outperforms all other modeling and sampling combinations. In fact, the proposed implementation generally provides increased model accuracy and requires fewer function evaluations. This shows that the EMS sampling method discovers more useful information per function evaluation than the SLH over each of the modeling methods. In most cases, the M/EMS approach is able to attain highly accurate surrogate models using fewer terms than either LASSO or OLR.

Tables 5 and 6 show the range of final model complexities, $T^{\text{surrogate}}$, found by each combination 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 outlined here, M/EMS, is able to generate a model that is no more complex—and often much simpler—than any of the alternative competing combinations. In addition to finding the most accurate models, our proposed method often requires fewer terms than tested alternatives.

For test set A, we perform a more detailed analysis of the number of terms present in the final surrogate model as the true number of terms, T^{true}, is known and can be compared against $T^{\text{surrogate}}$. The results of this comparison are shown in detail in Figures 9 and 10 and Table 7 as 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. A smaller deviation signifies fewer required terms in the final surrogate model. The results show that, not only, the proposed modeling method (M) is 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 ALAMO process does not guarantee a unique solution. In fact, there are several cases where different sets of initial points lead to different active basis functions sets. Solution quality, however, does not suffer in these cases. It is often the case that, over certain ranges of x, different basis functions behave very similarly. For example, the following two surrogate models $\hat{z}(x) = \sqrt{x} + 0.1x + 1$ and $\hat{z}(x) = 1.27\log(x) + 0.21x + 1.8$ have differing sets of basis

Table 5. The Average Minimum and Maximum Number of Surrogate Model Terms for Test Set A

No. of Inputs	No. of True Terms	M/EMS	M/SLH	EBS/EMS	EBS/SLH	LASSO/EMS	LASSO/SLH	OLR/EMS	OLR/SLH
2	2	2	[2, 2]	2	2	[6, 8]	[6, 11]	[12, 15]	[12, 15]
2	3	3	3	3	3	[5, 12]	[5, 10]	[12, 14]	[12, 14]
2	4	[3, 4]	[3, 4]	[3, 4]	[3, 4]	[8, 11]	[8, 10]	[11, 12]	[11, 12]
2	5	[2, 4]	[2, 4]	[2, 5]	[2, 5]	[3, 12]	[4, 11]	[10, 16]	[10, 16]
2	6	[5, 6]	[6, 6]	[5, 6]	[6, 6]	[7, 10]	[6, 7]	[11, 13]	[11, 13]
2	7	[4, 6]	[4, 6]	[4, 7]	[4, 7]	[7, 11]	[6, 12]	[8, 13]	[8, 13]
2	8	[4, 5]	[5, 6]	[4, 5]	[5, 6]	[6, 8]	[6, 9]	[10, 15]	[10, 15]
2	9	[4, 6]	[4, 6]	NA[a]	NA[a]	[6, 14]	[7, 12]	[10, 17]	[10, 17]
2	10	[4, 8]	[4, 8]	NA[a]	NA[a]	[5, 14]	[7, 14]	[10, 14]	[10, 14]

[a]Note: Due to large CPU times EBS tests were not run with greater than 9 true terms.

Table 6. The Average Minimum and Maximum Number of Surrogate Model Terms for Test Set B

True Function Type	Function ID	M/EMS	M/SLH	LASSO/EMS	LASSO/SLH	OLR/EMS	OLR/SLH
I	a	5	5	[3, 5]	[4, 9]	[6, 17]	[6, 17]
I	b	[4, 10]	[4, 10]	[10, 14]	[5, 8]	[8, 17]	[8, 17]
I	c	[3, 10]	[6, 9]	[8, 9]	[4, 10]	[13, 17]	[13, 17]
II	a	[4, 6]	[4, 10]	[8, 15]	[7, 9]	[15, 19]	[15, 19]
II	b	[1, 7]	[1, 9]	[13, 16]	[11, 17]	[13, 30]	[13, 30]
II	c	[5, 12]	[5, 12]	[9, 13]	[9, 16]	[9, 19]	[9, 19]
III	a	[3, 4]	[1, 4]	[2, 5]	[2, 5]	[9, 20]	[9, 20]
III	b	4	[1, 4]	5	5	[9, 20]	[9, 20]
III	c	[3, 4]	[3, 4]	[5, 8]	[5, 9]	[18, 24]	[18, 24]
IV	a	[7, 8]	[4, 10]	[8, 17]	[11, 18]	[13, 19]	[13, 19]
IV	b	[8, 9]	[9, 10]	[8, 12]	[10, 14]	[9, 17]	[9, 17]
IV	c	[6, 9]	[9, 10]	[5, 13]	[4, 12]	[13, 15]	[13, 15]

Note: Due to large CPU times EBS tests were not run on test set B.

functions. However, these two models only deviate by 0.076% over $x \in [2, 10]$. Depending on the initial sample set, either of these models might be chosen to equal effect by the proposed method. The similarity of model performance with different sets of basis function also allows us to model large, complicated systems with a limited, but flexible set of basis functions.

The results of these experiments show the success of the proposed method in terms of our three main goals: model accuracy, model simplicity, and modeling efficiency, with respect to function evaluations. In the next section, we demonstrate the effectiveness of these derived models in the context of an algebraic optimization study.

Case Study: Carbon Capture Adsorber

The synthesis of an optimal carbon capture process 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 trade-offs 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 use the ALAMO methodology to create an algebraic surrogate model from a solid sorbent adsorber simulation, which is one technology under development for postcombustion carbon capture.

The adsorber under consideration is a bubbling fluidized bed (BFB), which is modeled as a system of partial differential equation (PDEs) in Aspen Custom Modeler and is described by Lee and Miller.[50] Figure 11 shows the major features of the model. CO_2 rich gas enters the bottom of the reactor and contacts the solid. CO_2 lean solids enter the top of the bed and leave as a CO_2 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 have

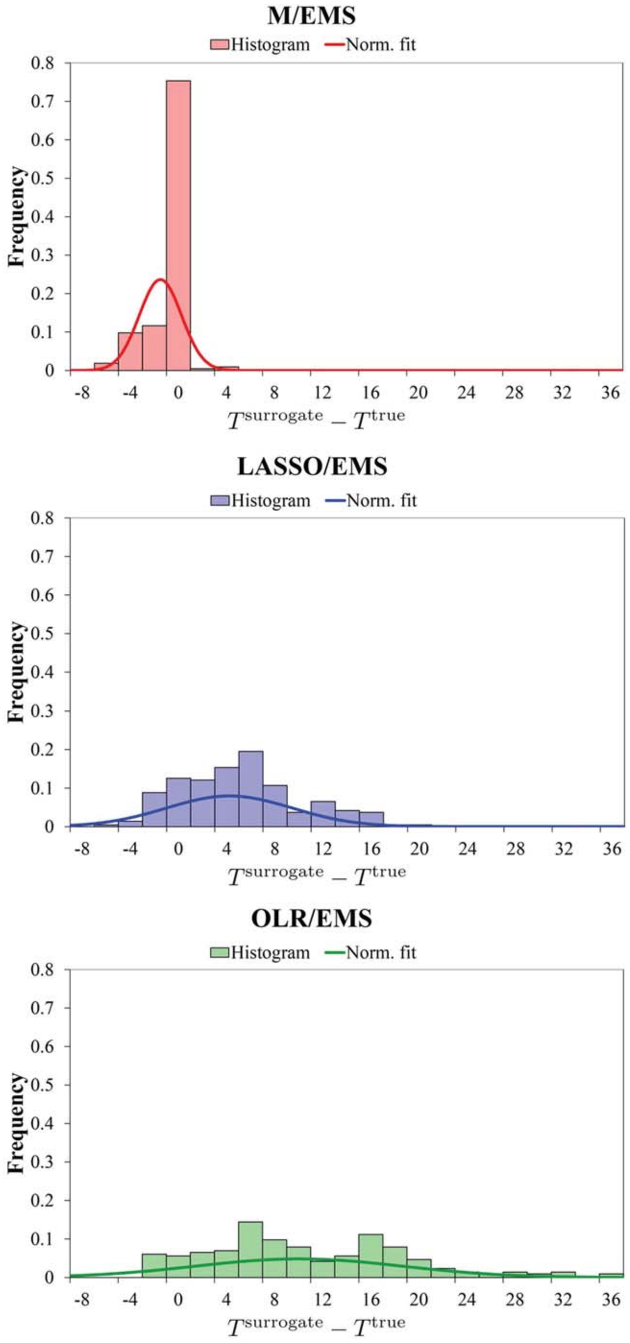

Figure 9. Model complexity comparison for each modeling method using EMS from test set A.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

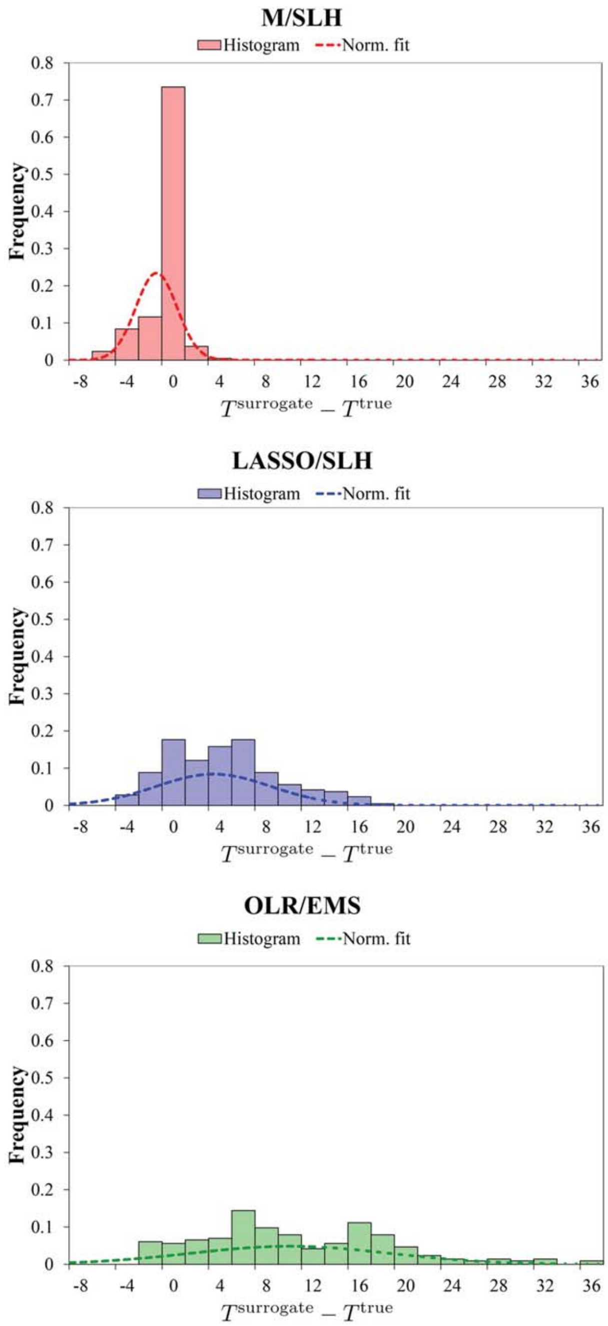

Figure 10. Model complexity comparison for each modeling method using a SLH from test set A.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

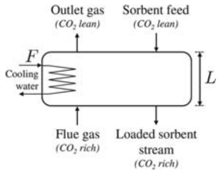

Figure 11. Diagram of the solid sorbent carbon capture adsorber.

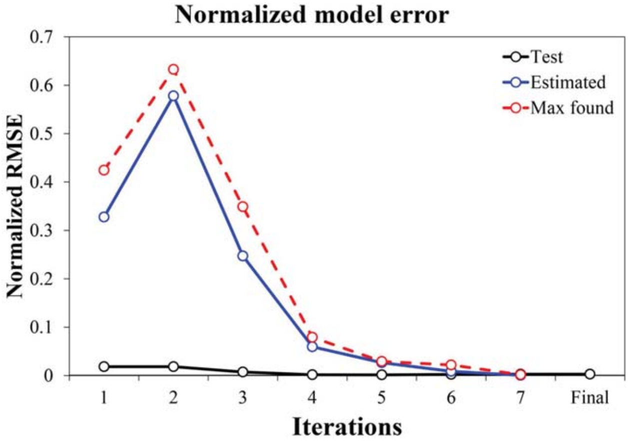

Figure 12. Normalized error progression through the entire algorithm.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

a significant impact on 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

$$r = \frac{CO_2 \text{ in outlet flow}}{CO_2 \text{ in flue gas}} \tag{9}$$

The CO_2 removed increases with diminishing return as both the bed depth and cooling water flow rate are increased.

We employ ALAMO to develop a surrogate model for r as a function of $L \in [2\text{m}, 10\text{m}]$ and $F \in [16.7\,\text{kmol/s},\ 1666.7\,\text{kmol/s}]$ using the model builder M and the EMS adaptive sampling routine. As we expect many of the solutions comprising the pareto analysis to be at variable bounds, we select an initial sample set 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 12. A separate test set of 394 data points is 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 evidence 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 here as well.

Figures 13 and 14 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 areas of high model mismatch shown in

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

	M/EMS	M/SLH	LASSO/EMS	LASSO/SLH	OLR/EMS	OLR/SLH
Mean	−0.860	−0.814	4.619	3.879	10.177	10.177
Standard deviation	1.688	1.703	5.005	4.745	8.261	8.261

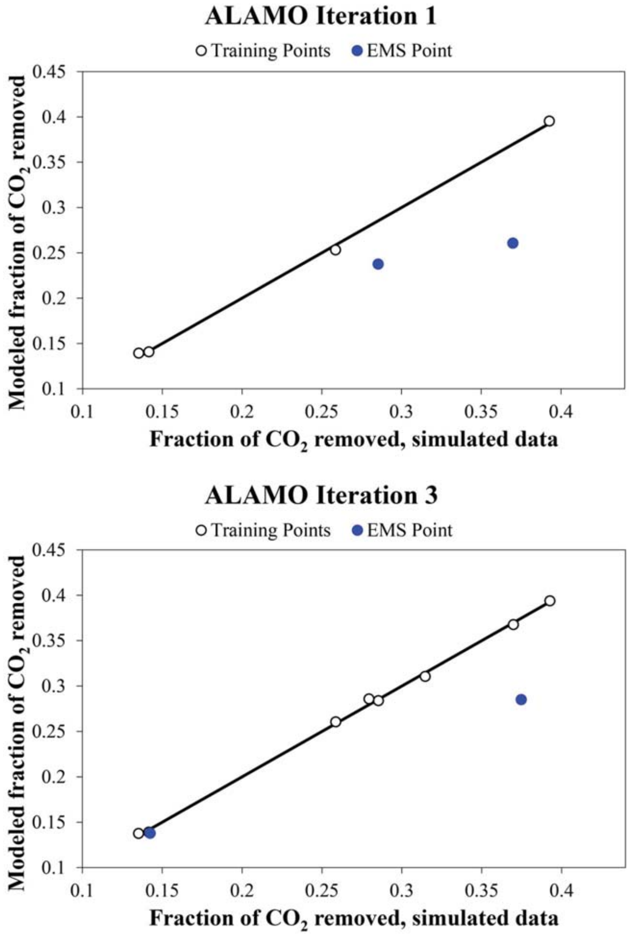

Figure 13. Modeled current and EMS data for Iterations 1 and 3 of the proposed methodology compared with true simulated data.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

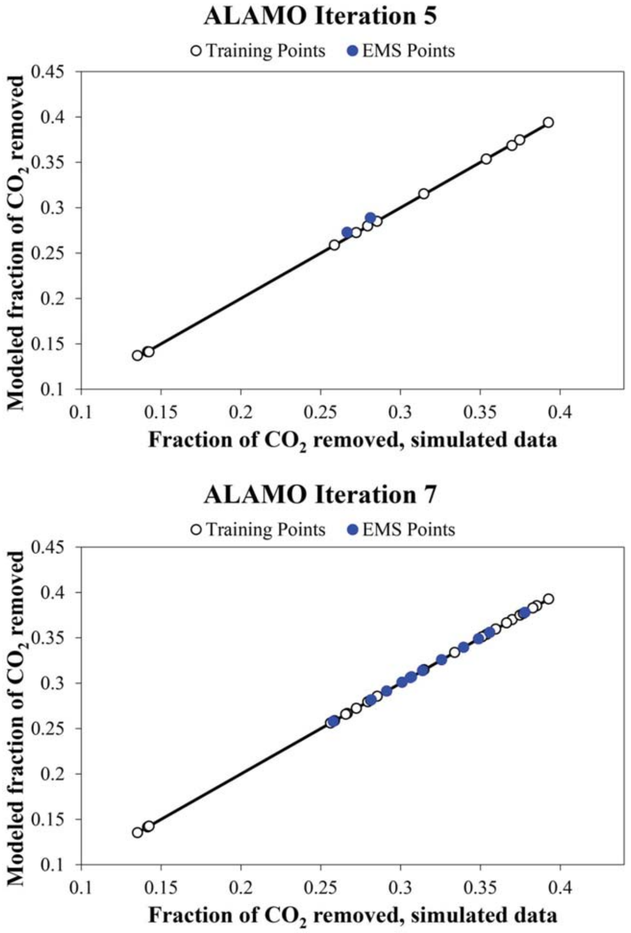

Figure 14. Modeled current and EMS data for Iterations 5 and 7 of the proposed methodology compared with true simulated data.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

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% and maximum error tolerance of 1%. The resulting models, model size, and number of EMS points generated for each iteration are shown in Table 8.

Figure 15 shows the solution path as the complexity of the model is allowed to increase from 1 to 13 basis functions. The complete basis set has 67 terms available to build each model. The decrease in error seen in Figure 15 from 12 to 13 terms does not justify the use of a 13-term model. The final model chosen is the following 12-term model

$$\hat{r}(L,F)=0.201\log(0.2L)+\left(6.55\cdot 10^{-10}\right)F^2L-\frac{\left(6.96\cdot 10^{-6}\right)F}{\sqrt{L}}$$
$$-\frac{0.609L}{\sqrt{F}}-\left(4.27\cdot 10^{-4}\right)\sqrt{F}L+\left(4.36\cdot 10^{-4}\right)\sqrt{F}$$
$$+\left(4.09\cdot 10^{-4}\right)L^2+\frac{0.387}{\sqrt[3]{L}}+0.147\sqrt[3]{L}+\frac{0.145\mathrm{L}^2}{\mathrm{F}}$$
$$+\frac{3.00\cdot 10^{3}}{F^2L^2}-\frac{3.23\cdot 10^{6}}{F^4L^4}$$

Once the model has been generated it can be used to analyze the trade-offs among bed depth and cooling water flow rate and percentage of CO_2 removed from the entering gas. Figure 16 presents the results of the pareto analysis showing the trade-offs 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 cost of electricity (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 the National Energy Technology Laboratory power systems financial model.[51] 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*.

Although we have accurately modeled the actual CO_2 removal, we have not modeled the first derivatives. Initially, we must assume that the surrogate accurately models the first derivatives of the rigorous simulation. After an optimum is located, this can be verified in several ways. Fully linear models[52] have bounded model error and derivative error in the neighborhood of a solution. Finite difference techniques can be used to approximate the first derivatives near the solution. Using simulator evaluations, we can apply either technique to prove that our solution is a local optimum. Alternatively, we have seen promise in using the proposed

Table 8. Surrogate Models Built at Each Iteration for the BFB Adsorber

Iter.	EMS Points	Terms	Model
1	2	2	$\left(4.88\cdot10^{-5}\right)F\sqrt{L}+0.138$
2	2	4	$\left(2.14\cdot10^{-4}\right)F\sqrt{L}-\left(2.58\cdot10^{-8}\right)F^2L+\frac{1.00\cdot10^4}{F^4}-\frac{\left(7.45\cdot10^{-8}\right)F^2}{L}$
3	2	3	$0.0556\sqrt[3]{F}-\left(2.64\cdot10^{-4}\right)F\sqrt{L}+\left(6.78\cdot10^{-5}\right)FL$
4	2	6	$0.0458\sqrt[3]{F}-\left(3.51\cdot10^{-10}\right)F\exp(L)-\frac{\left(3.13\cdot10^{-4}\right)F}{\sqrt{L}}-\frac{0.0687}{L}+\frac{0.0707}{\sqrt[3]{L}+\frac{\left(1.28\cdot10^{-13}\right)F^4}{L^4}}$
5	2	5	$0.0709\sqrt[3]{F}-\left(9.60\cdot10^{-3}\right)\sqrt{F}-\frac{\left(1.06\cdot10^{-3}\right)F}{L^2}-\frac{\left(2.01\cdot10^{-7}\right)F^2}{L}+\frac{\left(7.67\cdot10^{-7}\right)F^2}{L^2}$
6	9	8	$\frac{0.365}{L^3}-\frac{0.314}{L^2}-\frac{0.492L}{\sqrt{F}}+0.257\sqrt[3]{L}+\frac{0.120L^2}{F}+\frac{2.21\cdot10^3}{F^2L^2}-\frac{2.38\cdot10^6}{F^4L^4}-\left(2.75\cdot10^{-6}\right)FL$
7	12	12	$0.200\log(0.2L)+\left(6.52\cdot10^{-10}\right)F^2L-\frac{\left(6.12\cdot10^{-6}\right)F}{\sqrt{L}}-\frac{0.609L}{\sqrt{F}}-\left(4.25\cdot10^{-4}\right)\sqrt{F}L$ $+\left(3.98\cdot10^{-4}\right)\sqrt{F}+\left(4.04\cdot10^{-4}\right)L^2+\frac{0.385}{\sqrt[3]{L}+0.148\sqrt[3]{L}+\frac{0.145L^2}{F}+\frac{3.00\cdot10^3}{F^2L^2}-\frac{3.23\cdot10^6}{F^4L^4}}$
Final	NA	12	$0.201\log(0.2L)+\left(6.55\cdot10^{-10}\right)F^2L-\frac{\left(6.96\cdot10^{-6}\right)F}{\sqrt{L}}-\frac{0.609L}{\sqrt{F}}-\left(4.27\cdot10^{-4}\right)\sqrt{F}L$ $+\left(4.36\cdot10^{-4}\right)\sqrt{F}+\left(4.09\cdot10^{-4}\right)L^2+\frac{0.387}{\sqrt[3]{L}+0.147\sqrt[3]{L}+\frac{0.145L^2}{F}+\frac{3.00\cdot10^3}{F^2L^2}-\frac{3.23\cdot10^6}{F^4L^4}}$

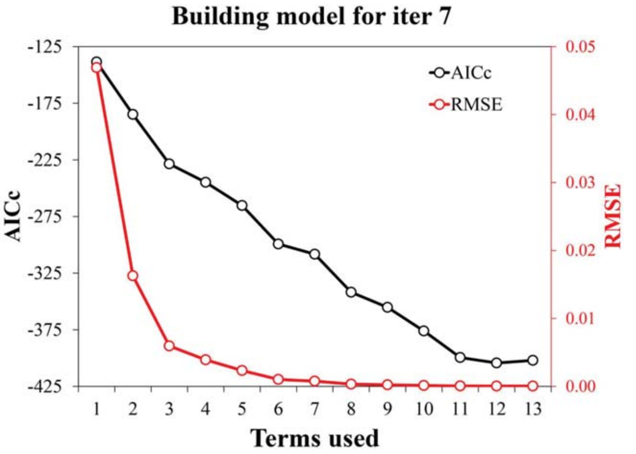

Figure 15. Error and goodness-of-fit during the model building for Iteration 7.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

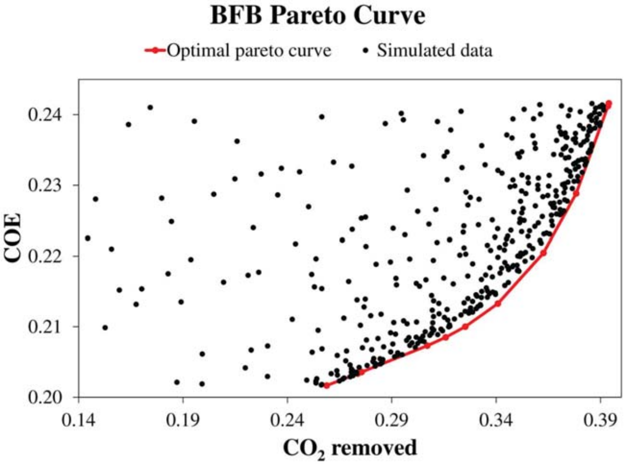

Figure 16. Pareto analysis of the BFB adsorber.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

methods to define flow sheets and search complicated topologies to fix integer decision variables and discover favorable starting points for continuous variable DFO.[39] To verify the points on the pareto curve in Figure 16, 394 evenly sampled test points are evaluated and are plotted on the same axis. Each of these points represents a feasible point, though most represent suboptimal process conditions as either a reduction of costs or an increase in removed CO_2 is possible. The algebraic model derived by ALAMO can now be used, not only, for optimization, but also for uncertainty quantification via thousands of surrogate simulations that consume a fraction of the CPU time required by a single evaluation of the original simulation.

Conclusions

Simulation-based optimization provides a rich avenue for evaluating design parameters according to cost functions. 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 component blocks comprising a more computationally complex simulation 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, statistical techniques are used to avoid overfitting and to ensure the generation of simple, compact models that promote their eventual use as surrogates in algebraic optimization. A new active sampling method, EMS, has been shown to improve the quality of sampled points. The efficacy of these methods is demonstrated over several competing modeling methods.

We can conclude that, if a simulation is required to characterize and 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 article. The proposed methodology is equally applicable for fitting experimental data.

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.

Literature Cited

1. Seader JD, Seider WD, Pauls AC, Hughes RR. *FLOWTRAN Simulation: An Introduction*. Cambridge, MA: CACHE, 1977.
2. Pantelides CC. SpeedUp—recent advances in process simulation. *Comput Chem Eng*. 1988;12:745–755.
3. Evans LB, Boston JF, Britt HI, Gallier PW, Gupta PK, Joseph B, Mahalec V, Ng E, Seider WD, Yagi H. ASPEN: an advanced system for process engineering. *Comput Chem Eng*. 1979;3:319–327.
4. Turton R, Bailie RC, Whiting WB, Shaeiwitz JA. *Modeling and Simulation in Chemical Engineering*. Upper Saddle River, NJ: Pearson Education, 2008.
5. Fu MC. Optimization for simulation: theory vs. practice. *INFORMS J Comput*. 2002;14:192–215.
6. April J, Glover F, Kelly JP, Laguna M. Practical introduction to simulation optimization. *Proceedings of the 2003 Winter Simulation Conference, 2003*, IEEE, New Orleans, LA, Vol. 1. 2003:71–78.
7. Fu MC, Glover FW, April J. Simulation optimization: a review, new developments, and applications. *Proceedings of the Winter Simulation Conference, 2005*. IEEE, Austin, TX, 2005:13.
8. Drud A, ARKI Consulting and Development A. *CONOPT 3 Solver Manual*. 2003. Available at http://www.gams.com/dd/docs/solvers/conopt.pdf. Last accessed January 15, 2014.
9. Wächter A, Biegler LT. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Math Program*. 2006;106:25–57.
10. Gill PE, Murray W, Saunders MA. *User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming*. 2008. Available at http://www.sbsi-sol-optimize.com/manuals/SNOPT%20Manual.pdf. Last accessed January 15, 2014.
11. Tawarmalani M, Sahinidis NV. A polyhedral branch-and-cut approach to global optimization. *Math Program*. 2005;103:225–249.
12. Chen W, Shao Z, Qian J. Interfacing IPOPT with Aspen open solvers and CAPE-OPEN. *10th International Symposium on Process Systems Engineering: Part A*. Vol. 27. Amsterdam, The Netherlands: Elsevier, 2009:201–206.
13. Caballero J, Grossmann IE. An algorithm for the use of surrogate models in modular flowsheet optimization. *AIChE J*. 2008;54:2633–2650.
14. Jones DR, Schonlau M, Welch WJ. Efficient global optimization of expensive black-box functions. *J Glob Optim*. 1998;13:455–492.
15. Rios LM, Sahinidis NV. Derivative-free optimization: a review of algorithms and comparison of software implementations. *J Glob Optim*. 2013;56:1247–1293.
16. Jones DR. A taxonomy of global optimization methods based on response surfaces. *J Glob Optim*. 2001;21:345–383.
17. Wang GG, Shan S. Review of metamodeling techniques in support of engineering design optimization. *J Mech Des*. 2007;129:370–380.
18. Antoulas AC, Sorensen DC, Gugercin S. A survey of model reduction methods for large-scale systems. *Contemp Math*. 2001;280:193–218.
19. Palmer K, Realff M. Metamodeling approach to optimization of steady-state flowsheet simulations: model generation. *Chem Eng Res Des*. 2002;80:760–772.
20. Huang D, Allen T, Notz W, Zeng N. Global optimization of stochastic black-box systems via sequential kriging meta-models. *J Glob Optim*. 2006;34:441–466.
21. Davis E, Ierapetritou M. A kriging method for the solution of nonlinear programs with black-box functions. *AIChE J*. 2007;53:2001–2012.
22. Henao CA, Maravelias CT. Surrogate-based superstructure optimization framework. *AIChE J*. 2011;57:1216–1232.
23. Gorissen D, Crombecq K, Couckuyt I, Dhaene T, Demeester P. A surrogate modeling and adaptive sampling toolbox for computer based design. *J Mach Learn Res*. 2010;11:2051–2055.
24. Schmidt M, Lipson H. Distilling free-form natural laws from experimental data. *Science*. 2009;324:81–85.
25. McKay MD, Beckman RJ, Conover WJ. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*. 1979;21:239–245.
26. Simpson TW, Peplinski J, Koch PN, Allen JK. Metamodels for computer-based engineering design: survey and recommendations. *Eng Comput*. 2001;17:129–150.
27. Krige DG. A statistical approach to some mine valuations and allied problems at the witwatersrand. Masters Thesis. South Africa: University of Witwatersrand, 1951.
28. Cressie N. Spatial prediction and ordinary kriging. *Math Geol*. 1988; 20:405–421.
29. Hassoun M. *Fundamentals of Artificial Neural Networks*. Cambridge, MA: MIT Press, 1995.
30. Burnham KP, Anderson DR. *Model selection and multimodel inference: A practical information-theoretic approach*. New York: Springer, 2002.
31. Gatu C, Yanev PI, Kontoghiorghes EJ. A graph approach to generate all possible regression submodels. *Comput Stat*. 2007;52:799–815.
32. Gatu C, Kontoghiorghes EJ. Parallel algorithms for computing all possible subset regression models using the QR decomposition. *Parallel Comput*. 2003;29:505–521.
33. Tibshirani R. Regression shrinkage and selection via the Lasso. *J R Stat Soc Ser B Methodol*. 1996;58:267–288.
34. Stoica P, Selén Y. Model-order selection: a review of information criterion rules. *IEEE Signal Process Mag*. 2004;21:36–47.
35. Akaike H. A new look at the statistical model identification. *IEEE Trans Automat Control*. 1974;19:716–723.
36. McQuarrie AD, Tsai CL. *Regression and Time Series Model Selection*. Singapore: World Scientific Publishing Company, 1998.
37. Hurvich CM, Tsai CL. A corrected Akaike information criterion for vector autoregressive model selection. *J Time Ser Anal*. 1993;14: 271–279.
38. Von zur Gathen J, Sieveking M. A bound on solutions of linear integer equalities and inequalities. *Proc Am Math Soc*. 1978;72:155–158.
39. Miller DC, Sahinidis NV, Cozad A, Lee A, Kim H, Morinelly J, Eslick J, Yuan Z. Computational tools for accelerating carbon capture process development. *The 38th International Technical Conference on Clean Coal & Fuel Systems*. Coal Technology Associated, Clearwater, FL, Vol. 1. 2013.
40. Fang K-T, Lin DKJ. Uniform experimental designs and their applications in industry. Chapter 4 in Statistics in Industry, edited by Khattree R, Rao CR, Elsevier, Amsterdam, The Netherlands, 2003; 22:131–170.
41. Crombecq K, Laermans E, Dhaene T. Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling. *Eur J Oper Res*. 2011;214:683–696.
42. Crombecq K, Tommasi L, Gorissen D, Dhaene T. A novel sequential design strategy for global surrogate modeling. *Winter Simulation Conference*. IEEE. Austin, TX, Vol. 1. 2009:731–742.
43. Provost F, Jensen D, Oats T. Efficient progressive sampling. *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, San Diego, CA, Vol. 1. 1999:23–32.
44. Geest J, Dhaene T, Fach N, Zutter DD. Adaptive CAD-model building algorithm for general planar microwave structures. *IEEE Trans Microwave Theory Tech*. (Special Issue on Multilayer Microwave Circuits). 1999;9:1801–1809.
45. Thompson S. *Sampling*. Hoboken, NJ: Wiley, 2002.
46. Pukelsheim F. *Optimal Design of Experiments*. New York: Society for Industrial and Applied Mathematics, 2006.
47. Conn AR, Scheinberg K, Vicente LN. *Introduction to Derivative-Free Optimization*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2009.
48. Huyer W, Neumaier A. SNOBFIT—stable noisy optimization by branch and fit. *ACM Trans Math Softw*. 2008;9:1–25.

49. Neumaier A. MINQ–general definite and bound constrained indefinite quadratic programming. 1998. Available at http://www.mat.univie.ac.at/neum/software/minq/. Last accessed January 15, 2014.
50. Lee A, Miller DC. A one-dimensional, three region model for a bubbling fluidised bed adsorber. *Ind Eng Chem Res.* 2013;52:469–484.
51. *NETL Power Systems Financial Model Version 5.0*. 2008. Available at http://www.netl.doe.gov/business/solicitations/ssc2008/references/PSFM.
52. Conn AR, Scheinberg K, Vicente LN. Geometry of interpolation sets in derivative-free optimization. *Math Program.* 2008;111:141–172.

Manuscript received Oct. 5, 2013, and revision received Jan. 16, 2014.