Carnegie Mellon University

CARNEGIE INSTITUTE OF TECHNOLOGY

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF Doctor of Philosophy

TITLE Data- and Theory-driven Techniques for

Surrogate-based Optimization

PRESENTED BY Alison Cozad

ACCEPTED BY THE DEPARTMENT OF

Chemical Engineering

NIKOLAOS SAHINIDIS	4/28/14
ADVISOR	DATE
LORENZ BIEGLER	4/28/14
DEPARTMENT HEAD	DATE

APPROVED BY THE COLLEGE COUNCIL

VIJAYAKUMAR BHAGAVATULA	4/28/14
DEAN	DATE

Data- and theory-driven techniques for surrogate-based optimization

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Chemical Engineering

Alison L. Cozad

B.S., Chemical Engineering, University of Minnesota

Carnegie Mellon University
Pittsburgh, PA

May, 2014

Acknowledgements

First and foremost, I would like to extend my thanks to my advisor Dr. Nick Sahinidis. Without his guidance and instruction, I would not have developed the combination of optimization, engineering, and statistics that made this thesis possible. His encouragement and feedback have allowed me to grow as an independent researcher. He inspired me to gain the skill set and confidence required to develop as a presenter by exposing me to many diverse presentation opportunities. Without his continued support, this thesis would not be complete.

Both my education and the content of this thesis have been greatly impacted by the members of my committee: Dr. Lorenz Biegler, Dr. Ignacio Grossman, Dr. David Miller, and Dr. Eric Xing. I had the opportunity to receive instruction, either inside or outside of the classroom, from each of these individuals and their teaching fostered the growth of my mathematical fundamentals and engineering creativity. Without timely consultation and encouragement from them, this thesis would not have been possible.

Through fine work and support, the National Energy Technology Laboratory (NETL) served as a touchstone, grounding my research with real-world applications and a rich set of industrial data. In addition to Dr. David Miller, who provided feedback; interesting and challenging problem sets; advice; and expertise, I would like to thank John Eslick, Andrew Lee, Juan Morinelly, Zhihong Yuan, and the other outstanding contributors at NETL who made their time and resources available for this work.

The students in the Sahinidis research group have been a rich source of technical discussions, presentation practice and help, and idea generation. Their help in the form of minor edits, critiques, and brainstorming sessions enabled this thesis to take shape. In particular, I would like to thank Dr. Keith Zorn for his advice, guidance, and support in every step of the processes. At critical points, he helped me to define, plan, and refine the scope of this work to ensure a well-rounded thesis.

Above all, I extend my gratitude to my friends and family who helped me to get to this point in my life and then helped me to get through it. My parents have always encouraged my goals and interests: from plastic-pencil science projects, to over-sized papier-mâché rodents and summer math-workbooks. They taught me the benefits of hard work and instilled a great sense of confidence and competition while helping me learn from my failures. My dad has always piqued my interest by showing and explaining the technical and physical aspects of everyday processes and my mom has always shown me the joy and humor in everyday things. My sister has been a silent teacher my whole life and provides up-to-date feedback on life lessons while being an intelligent, strong woman that I aspire to emulate.

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.

To my parents, for their love and support; and
to Ephie and Heron, who make me smile when I need it most.

Abstract

The goals of optimal decision-making often conflict with high-fidelity prediction targets when designing and optimizing chemical systems. If algebraic first-principles models are unreliable or unavailable, conventional methods require compromising either decision or prediction capabilities. To make high-level decisions, an engineer might sacrifice model fidelity to solve design problems with large length and time scales or complicated optimization topologies. Alternatively, to ensure high prediction capabilities, the scope of the optimization problem may be diminished to accommodate high-fidelity, black-box models due to limited feasible regions and discrete optimal decisions.

The focus of this thesis is to investigate a third option: surrogate-based optimization. We extend the decision compatibilities of of mixed-integer optimization frameworks to include high-fidelity representations of modular process elements. To do this, we substitute high-fidelity simulators and experiments with sets of tailored surrogate models. In Chapters 2 through 4, we investigate the identification of simple, accurate surrogate models generated using a theory- and data-driven approach. The two key aspects of surrogate modeling are model identification and the data set that forms it foundation.

We are interested in developing a technique that learns models that are (a) as accurate as possible and (b) as simple as possible. Requirement (a) is obvious, while requirement (b) is driven by our desire to utilize the model in a larger multi-scale model for optimization, simulation, or analysis. We implement a best subset regression method to select a subset of nonlinear basis functions that when combined, serve as a flexible underlying functional form for the surrogate model. Using a global approach to symbolic regression, we consider a more flexible nonlinear functional surrogate model using only the specification of simple mathematical operators such as addition, subtraction, multiplication, and division.

Data and information are the foundation of an empirical model. The aim of empirical data collection is to achieve the highest model quality from a given data set. To do this, we propose a novel approach to an iterative design of experiments that adaptively searches the surrogate model and black-box problem space to locate areas of model weakness, or high model error. We augment the empirical data set to include theory-based information, response bounds, and physical restrictions by introducing a constrained regression method to enforce these physical limits. This combination of data- and theory-driven techniques leads to high surrogate model quality.

We introduce an implementation of several modeling and data sampling methodologies to perform black-box surrogate modeling. We use this implementation for the design and optimization of a post combustion carbon capture process. We use a surrogate-based approach to systematically and rapidly screen processes to compare best-case scenarios with the aim of identifying the most promising selections of technology combinations.

We demonstrate the accuracy, parsimony, and efficiency of the surrogate modeling methods through numerous example problems and computational studies. The efficacy of these models in surrogate-based optimization is observed on a case study and industrial carbon capture application.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

The balance between optimal decision-making and model fidelity remains a challenge when designing and operating complex, nonlinear chemical systems. A process systems engineering model is a physically feasible system that minimizes or maximizes a cost objective. In practice, engineers are often forced to make one of two primary modeling concessions if they lack a sufficient algebraic, first-principles representation. One choice forgoes the accuracy of simulated, experimental, or chemical plant data in favor of simplifications, such as linearity, to capture process trends when solving problems with large length and time scales and/or complicating optimization topologies [13]. Alternatively, the scope of the optimization problem is sacrificed to accommodate high fidelity models when feasible regions and discrete optimal decisions are limited.

We propose an alternate third choice: surrogate-based optimization of high-fidelity black-box problems using tailored surrogate models. In this work, we expand the scope of mixed-integer optimization to solve problems using high-fidelity simulators by identifying a set of surrogate models that are tailored for algebraic optimization.

These simple and accurate surrogate models are incorporated into a larger, deterministic optimization framework. We focus on two key aspects of surrogate modeling: model identification and the efficient storage of data and information.

Advanced optimization techniques enable high-level system decisions such as network configuration, reactor type selection, and other discrete selections to be optimized alongside continuous conditions such as flow rates, temperatures, and geometries. However, these conventional optimization methods, known as derivative-based or algebraic solvers (e.g., `CONOPT` [29], `IPOPT` [104], and `SNOPT` [38]), require the use of derivative information to discover feasible optimal solutions. More advanced global optimization methods, such as `BARON` [96], require an algebraic functional form to locate and certify a globally optimal solution. Optimization formulations suitable for these techniques are comprised of algebraic objectives and constraints that relate decision variables through first principles models, mass and energy balances, unit models, and design constraints. If relationships or feasible regions cannot be defined through first-principle derivation, simplifications or approximate relationships are substituted. Typically, these simplified models lack fidelity and are an incomplete representation of process trends, particularly for the types of highly nonlinear relationships that are common to chemical engineering problems.

Some local optimization algorithms are capable of solving in the absence of a complete algebraic model if derivatives can be evaluated or approximated. Unfortunately, solver reliability degrades as the complexity of the associated black box models increases. In practice, direct optimization is difficult without specialized algorithms. Due to the high sampling requirements of derivative estimation, the accuracy and efficacy of such solvers are limited by noisy and/or costly function evaluations that arise naturally in real-world and numerical simulations [18].

Despite the optimization challenges they provide, engineers have long used chemical process simulation and computational fluid dynamic methods industrially and academically to design and test systems and processes [30, 80, 90, 100]. The structure of these simulations or black boxes yields high levels of accuracy which lends well to predictive use, but imposes challenges when used for optimization and design [7, 32, 33]. Derivative-free optimization (DFO) is intended to fully or partially overcome the lack of an algebraic model and, in some cases, noisy and/or costly function evaluations. These algorithms are designed for optimization problems when derivatives are unavailable, unreliable, or prohibitively expensive to evaluate [50, 87]. To combat costly function evaluations, DFO solvers attempt to achieve an optimal feasible point using a minimal number of black-box function calls. In practice, however, these methods are unable to find 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 study [87].

Our work links high fidelity black box evaluators to advanced optimization routines through the use of algebraic surrogate models. Often, we utilize black boxes in a modular system by first disaggregating the process into smaller process groups that contain one or more process units. This addresses challenges commonly associated with black-box simulation robustness and results in more accurate surrogate models. Lastly, process decomposition and surrogate modeling allow for access to sophisticated optimization models that arise from a modular structure, including superstructure and discrete optimization. Connectivity and process alternative decisions can be determined using surrogate models in a derivative-based optimization structure.

Prior works in the process systems engineering literature have accomplished surrogate-based optimization by relying upon existing modeling schemes, typically kriging and neural networks, to build surrogate process models that can be optimized with derivative-based optimization techniques. We extend such existing modeling methodologies by introducing novel data-and-theory-driven model-building techniques. The tailored structure of the surrogate models generated by the proposed methods facilitates algebraic optimization. Our surrogates are composed of appropriate functional forms that (a) accurately represent the underlying distribution, (b) are as simple as possible, and (c) make effective use of all available information.

1.2 Surrogate-based optimization

Surrogate-based optimization involves the solution, ideally rigorous, of optimization formulations where one or more of the objectives or constraints are based on a surrogate model. The surrogate model(s) serve as a proxy for corresponding black-box aspects of the problem. This could relate to one or more black-box constraints and/or a black-box objective. Data from the black box is used to build the surrogate models; while, other connectivity variables and algebraic constraints are represented directly in the optimization formulation.

Frequently, a single model of the objective function is approximated before optimization; however, that need not be the case and the constraint set may also be modeled [28, 45, 79]. 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 [18, 43]. 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.

Surrogate-based methods have been effective in solving problems over an entire process system [28, 45, 79] and over a modular or disaggregated process [18, 43]. Palmer and Realff [79] have considered the indirect optimization of steady-state simulators using kriging surrogate models. David and Ierapetritou [28] use full process kriging-based models to identify global solutions and refine them using local response surfaces around the optima. Huang et al. [45] have addressed uncertainty in black-box systems using full process kriging models. Caballero and Grossmann [18] have investigated iterative modular flowsheet optimization using kriging models to represent process units with low-level noise. Recently, Henao and Maravelias [43] have demonstrated success modeling individual chemical process units using artificial neural networks.

1.3 Surrogate modeling

Surrogate models—known in some fields as metamodels or reduced-order models—are algebraic abstractions of a response or black-box output variable as a function of predictor or black-box input variables from data obtained from simulations or experiments. In this work, we investigate response models with explicit functions of predictor variables, however, the proposed methods can be extended to implicit models. Determining the correct model and best data set are central to the surrogate modeling problem. Model identification includes the problem of finding the model's functional form and/or parameter selection for a given a set of data. Equally important, the selection of the most effective data set can play a significant role in surrogate model quality.

Surrogate models can be either interpolative, where the model passes through each data point, or regressive, where a model is chosen to minimize some function of the distance from each the data point. Using regressive methods allows for modeling in the presence of error or noise in the underlying function [77]. A surrogate model is further defined by the region over which the predictor space it is modeled. A surrogate model or set of surrogate models is considered *global* if they represent the data over the entire problem space and *local* if they represent data over a subregion of the problem space.

1.3.1 Model form and parameter selection

A surrogate model is defined by its functional form and parameter levels. The form of a surrogate model is designated by the set of operations that make up each term or basis function in the model (i.e. linear term, quadratic term, exponential term, etc.) and the complexity of the model, which is often measured by the number of parameters used in the model. The parameters in a surrogate model are continuous decision variables in the surrogate model generation problem. They are chosen to maximize a goodness-of-fit or solution quality metric; commonly, this is done by minimizing the squared error.

In regression problems, the specification of *linear* or *nonlinear regression* is defined by the relationship between the regressor and model parameters. In contrast, the linearity or nonlinearity of a surrogate model relates to the relationship between the regressor and predictor variables; therefore, a nonlinear model could be solved using linear regression if the model is a linear combination of nonlinear terms. For linear regression, the response variable is proportional to all parameters for a given data point; in other words, they are coefficients for each linear or nonlinear term. If the response variable is related nonlinearly to one or more of the parameters, the regression problem is considered nonlinear. For example, nonlinear regression would be required to solve for the parameters β in a surrogate with the following form $\beta_1 x + \beta_2 \exp(x/\beta_3)$.

Common surrogate modeling methods range from simple linear or quadratic regression models to complex kriging [25, 65] and artificial neural networks [42]. Despite their advantageous optimization characteristics, simple models may not represent the highly nonlinear nature of chemical processes. In contrast, more complex models satisfy accuracy requirements but result in rough, complex functions that are difficult to solve using provable derivative-based optimization software.

Ideally, we would strike a balance between model accuracy and optimization tractability, but knowledge of this trade-off is limited by flexible functional forms. To address this uncertainty, we automate the selection of a functional form that best represents the data by identifying combinations of simple terms or basis functions that define a low-complexity, accurate functional form for each response. In Chapter 2, we explore model selection methods to identify functional forms from a large, fixed set of alternative structures. We generate global surrogate models using model and parameter selection methods that require the use of a best subset selection techniques to perform linear regression using nonlinear basis functions. We expand the search space of nonlinear model forms in Chapter 4 to identify model forms more flexibly by regressing the functional form of the model itself using the principles of symbolic regression.

1.3.2 Use of data and information

A regression model is only as strong as the information and data that it was built upon. The goal of regression methods is to select a model that best matches the underlying data distribution over a pre-specified problem-space or range of predictor variables. However, perfect information about a black box is not available, so solution quality is measured over a finite set of training data points. In least squares regression problems, the model form and parameters are chosen to minimize the squared error between the model and the set of training points. Consequently, the training objective is then an estimation of the true objective and the accuracy of this estimation is subject to the placement of the training points. To that end, significant work has been done in the area of design of experiments to select sample points that will generate the best surrogate models.

A *design of experiments* (DOE) deals with the selection of predictor levels to evaluate the black box [16]. The quality of a surrogate model is, in part, reflected in the quality of a DOE. Given a fixed data set size, a well-chosen DOE can result in greater accuracy. Similarly, given a model accuracy goal, a well-chosen DOE can result in fewer black box evaluations required. Perfect information would be ideal to train a model; however, with limited computational resources, function evaluations must also be limited.

Experimental design methods can be classified as either fixed or iterative. Fixed DOEs serve to generate a set of design points, evaluate these points, then move on to a modeling stage. Common fixed DOEs are random points, fractional designs [92], Latin hypercubes [74], and orthogonal arrays [31]. Without prior system knowledge, it is not possible to know how much information is needed or where data points should be sampled *a priori*. Therefore, iterative DOEs may be advantageous where the experimental design and modeling steps are repeated as desired. These approaches use both the current data set and the regression model to locate areas of difficult approximation by locating poorly sampled regions [81], areas of high nonlinearity, or areas of high uncertainty [36, 98]. We expand upon iterative design ideas by selecting new sample points in areas of model divergence or increased model mismatch. By doing this, we sample data only where it is most needed and terminate the sample procedure when no new data is required.

To leverage additional system knowledge, we use freely-available information beyond empirically sampled data. We supplement empirical data with theoretical *a priori* knowledge, including limits on the response variables; known relationships between response; and predictor variables; and relationships among responses. The combination of data-driven modeling and theory-driven *a priori* knowledge results in higher quality of surrogate models.

1.4 Outline of thesis

The main goal of our research is the expansion of complex optimization formulations to incorporate high fidelity experiments or simulations through the use of tailored accurate, low-complexity surrogate models.

In Chapter 2, we introduce the surrogate model learning techniques used throughout this thesis. Accurate, parsimonious surrogate models are generated using an optimization approach to best subset model selection for the identification of functional forms and parameter levels. We introduce an error maximization sampling procedure to perform an iterative design of experiments by locating problematic regions. We describe `ALAMO`, the computational implementation of the proposed methodology, along with examples and extensive computational comparisons between `ALAMO` and a variety of machine learning techniques.

In Chapter 3, we propose a constrained regression methodology that utilizes *a priori* system knowledge and empirical data to generate more accurate and physically realizable models. We introduce a set of constraints that can be applied generally to regression methods to infer parameter relationships using response and predictor relationships. We present several sources of relationships including bounding response variables, safe extrapolation, thermodynamic limitations, and enforcing favorable numerical properties. We also demonstrate the solution quality improvements from these regression restrictions through computational experiments.

In Chapter 4, we extend the functional form of basis function studied in Chapter 2 to the more flexible selection of model nonlinearities. In Chapter 2, the model's functional form is limited to combinations of predetermined basis functions. In Chapter 4, we allow increased modeling freedom and nonlinearity by learning the functional forms and corresponding model parameters simultaneously given only a set of operations such as addition, subtraction, multiplication, and division. Moreover, we propose an optimization formulation for the global optimization of the symbolic regression problem. Finally, we demonstrate the efficacy of this method using an illustrative and two literature examples.

In Chapter 5, we demonstrate the previously described techniques on a the optimization of an industrial post combustion carbon capture process. We detail the superstructure formulation used to optimize this process. The surrogate modeling techniques described in this thesis are used to learn sets of algebraic surrogate models of several reactors over differing technologies. We present results from reactor simulations and surrogate models that are currently used at the National Energy Technology Laboratory for superstructure optimization.

Finally, in Chapter 6, we summarize the contributions of this thesis and offer concluding remarks.

Chapter 2

Learning surrogate models for simulation-based optimization

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 enable the efficient consideration of a large number of possible functional modeling components. The model is then improved systematically through the use of derivative-free optimization solvers to adaptively sample new simulation or experimental points. We describe `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.

2.1 Introduction

Chemical process simulation and computational fluid dynamic methods have been used industrially and academically to design and test systems and processes [12, 13, 30, 80, 90, 100]. 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 [7, 32, 33]. This chapter 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 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 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 [29], IPOPT [104], and SNOPT [38]), requires the use of 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. 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 [96], 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 [21]. 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 [18].

Derivative-free optimization (DFO) offers a class of algorithms designed to solve optimization problems when derivatives are unavailable, unreliable, or prohibitively expensive to evaluate [50, 87]. 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 ten, even in the absence of constraints and integer variables, as shown in a recent computational study [87].

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) of the black-box functions $f(x)$ and/or $g(x)$ [49, 105].

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., reduced-order model–ROM) that has similar response characteristics to the high-fidelity simulation or system [6], 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 [28, 45, 79] or the disaggregated process [18, 43]. Palmer and Realff [79] have considered the indirect optimization of steady-state simulators using kriging surrogate models. In the same vein, David and Ierapetritou [28] 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. [45] have used kriging models on full processes. Caballero and Grossmann [18] have investigated disaggregated (modular) flowsheet optimization using kriging models to represent process units with low-level noise. Recent work by Henao and Maravelias [43] has shown success 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, reduced-order models 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 flowsheet, 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 derivative-free optimization 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 ALAMO (Automated Learning of Algebraic Models for Optimization), 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 (SUMO) [40], fail to generate surrogate models with sufficiently low complexity. Eureqa [89] can be used to search for low-complexity models; however, it operates on a fixed data set 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 chapter, 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 chapter 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 post-combustion carbon capture systems.

2.2 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. Lastly, 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, etc.) 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, etc.). 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 2.1. First, an initial design of experiments is generated over the problem space and the simulation is queried at these points. In the scope of this work, 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 adaptively improved as the procedure progresses, leaving a data set that bears little resemblance to the initial design of experiments. In the included example cases and experimental studies, we use either Latin hypercube sampling [74] or a 2-level factorial design [92] for this step.

Next, we build a simple, algebraic model using this initial training data set. The empirical model error, i.e., the deviation of the model from the data, can be calculated using standard statistical techniques. However, the true error, i.e., 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.

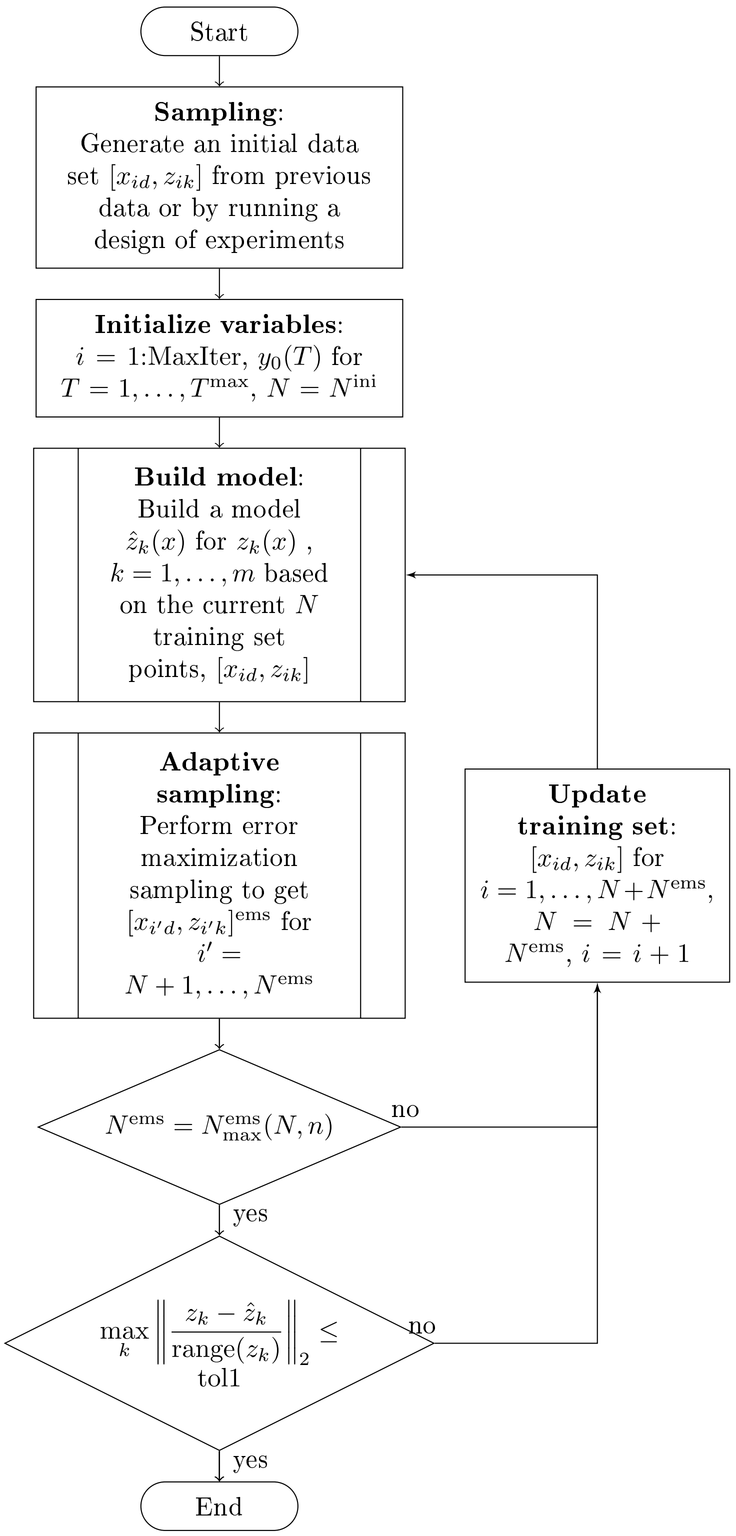

Start
Sampling:
Generate an initial data set $[x_{id}, z_{ik}]$ from previous data or by running a design of experiments
Initialize variables:
i = 1:MaxIter, $y_0(T)$ for $T = 1, \dots, T^{\max}$, $N = N^{\text{ini}}$
Build model:
Build a model $\hat{z}_k(x)$ for $z_k(x)$, $k = 1, \dots, m$ based on 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 $i' = N+1, \dots, N^{\text{ems}}$
Update training set:
$[x_{id}, z_{ik}]$ for $i = 1, \dots, N + N^{\text{ems}}$, $N = N + N^{\text{ems}}$, $i = i + 1$
$N^{\text{ems}} = N^{\text{ems}}_{\max}(N, n)$
no
yes
$\max_k \left\| \frac{z_k - \hat{z}_k}{\text{range}(z_k)} \right\|_2 \leq$ tol1
no
yes
End

Figure 2.1: Algorithmic flowchart

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.

2.2.1 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 [25, 65] and ANNs [42] 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.

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

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, etc.

Table 2.1: List of potential simple basis function forms

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 2.1 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 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 $z(x) = 2\,x^{3/2} + 1$ over $x \in [0, 1]$ we can generate a quadratic surrogate model, $\hat{z}(x) = 1.3\,x^2 - 0.35\,x + 2$, 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:

$$\hat{z} = \sum_{j \in \mathcal{B}} \beta_j X_j(x) \tag{2.1}$$

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

The ordinary least squares regression problem,

$$\min_{\beta} \sum_{i=1}^{N} \left(z_i - \sum_{j \in \mathcal{B}} \beta_j X_{ij} \right)^2 \tag{2.2}$$

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 (2.2) 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 [17]. 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 [17] 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 [35], branch and bound algorithms [34], and QR decomposition [34] 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) [99] 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

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

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:

$$\hat{z}(x) = \sum_{j \in \mathcal{S}} \beta_j X_j(x). \tag{2.3}$$

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, Equation (2.3) 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:

$$\begin{aligned} (BS1) \quad \min_{\beta,y} \quad & \Phi(\beta,y) \\ \text{s.t.} \quad & y_j \in \{0,1\}. \end{aligned}$$

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

$$\beta^l y_j \leq \beta_j \leq \beta^u y_j$$

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 nonzero value within its bounds if $y_j = 1$. The bounds on β are chosen using logic from the regularized regression technique: the lasso [99]. 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 ordinary least squares regression problem β^{OLR} to infer loose upper and lower bounds on each value of β. To do this, we set $\beta^l = -\sum\limits_{j\in\mathcal{B}} \left|\beta_j^{\mathrm{OLR}}\right|$ and $\beta^u = \sum\limits_{j\in\mathcal{B}} \left|\beta_j^{\mathrm{OLR}}\right|$.

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:

$$\min_{\beta,T,y} \Phi(\beta,T,y) = \min_{T} \left\{ \min_{\beta,y} \left[\Phi_{\beta,y}(\beta,y)|_T\right] + \Phi_T(T) \right\} \tag{2.4}$$

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:

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

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 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 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, i.e., an independent data set. 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 [95]; 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 [5], Bayesian information criterion [95], generalized information criterion [95], etc.. Each information criterion gives a measure of the accuracy versus the complexity of a model [5, 75]. 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 [46],

$$AICc(\mathcal{S}, \beta) = N \log \left(\frac{1}{N} \sum_{i=1}^{N} \left(z_i - \sum_{j \in \mathcal{S}} \beta_j X_{ij} \right)^2 \right) + 2|\mathcal{S}| + \frac{2|\mathcal{S}| \left(|\mathcal{S}| + 1\right)}{N - |\mathcal{S}| - 1} \tag{2.5}$$

which adjusts the Akaike information criterion to account for large basis sets. Equation (2.5) can be given in the form of (2.4) 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, 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:

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

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

$$\frac{d}{d\beta_j} \sum_{i=1}^{N} \left(z_i - \sum_{j \in \mathcal{S}} \beta_j X_{ij} \right)^2 \propto \sum_{i=1}^{N} X_{ij} \left(z_i - \sum_{j \in \mathcal{S}} \beta_j X_{ij} \right) = 0, \quad j \in \mathcal{S} \tag{2.6}$$

Equation (2.6) is used as a set of big-M constraints to define the basis coefficients:

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

where we calculate U_j to be the maximum of $\sum_{i=1}^{N} X_{ij} \left(z^i - \sum_{j \in \mathcal{B}} \beta_j X_{ij} \right)$ 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:

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

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 2.1), 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 [103], we have solved instances with as many as 14 inputs successfully [76].

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

2.2.2 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 design of experiments (DOE) can be classified into two categories: single pass and iterative.

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 $k \leftarrow 1$ 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 $AICc(t)$
        if $t > 1$ then
            if $AICc(t) > AICc(t-1)$ then        ▷ Information criterion worsened
                Set the final model complexity $T^{\mathrm{f}} = t - 1$
                break
            else if $\frac{AICc(t) - AICc(t-1)}{AICc(t-1)} \leq$ 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 $\beta = \beta(T^f)$ and $y = y(T^f)$
```

Single pass DOE methods, like fractional designs [92], Latin hypercubes [74], and orthogonal arrays [31], 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 data set and the current model to locate and sample complicating areas either by performing exploration-based or exploitation-based methods [26, 27]. Exploration-based methods aim to sample the problem space [81] 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 [65]) or there is another available error metric, these estimates can be used to locate areas of high uncertainty [36, 98].

Another relevant field of active learning that can be used in a single pass or iterative implementation is optimal design of experiments [82]. 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 [95]. Optimal design of experiments 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 design of experiments method is likely to be diluted by the presence of numerous unused basis functions.

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:

$$\max_{x^l \leq x \leq x^u} \left(\frac{z(x) - \hat{z}(x)}{z(x)} \right)^2 \tag{2.7}$$

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 [23]. As shown recently [87], these methods are most effective in low-dimensional cases. Thus, the technique of decomposing a large simulation into lower-dimensional blocks provides favorable conditions. Since the quality of DFO solutions may suffer in higher-dimensional spaces, we recommend using less than twenty 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 pre-specified 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.

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 Equation (2.7), $e_{ik} \leftarrow \left(\frac{z_{ik} - \hat{z}_k(x_i)}{z_{ik}}\right)^2$ and $E_i \leftarrow \sum_{k=1}^{m} e_{ik}$.
Initialize the number of error maximization points added, $N^{\text{ems}} \leftarrow 0$
while $N^{\text{ems}} \leq N^{\text{ems}}_{\max}$ 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 $x^{\text{req}} \leftarrow$ 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 $N^{\text{ems}} \geq N^{\text{ems}}_{\min}$ then
            if $\max_{ik}(e_{ik}) \geq$ tol4 then
                return $x, z, N^{\text{ems}}$
            end if
        end if
    end for
end while
```

2.3 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 $\frac{kg}{m^3}$, 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 2.1, where $\alpha = 0, \pm\frac{1}{3}, \pm\frac{1}{2}, \pm 1, \pm 2, \pm 3$ and $\gamma = 10,000\,W$. This leads to a potential functional form shown below with 13 basis functions:

$$\begin{aligned}\hat{\rho}(Q) &= \beta_0 + \beta_1 Q + \tfrac{\beta_2}{Q} + \beta_3 Q^2 + \tfrac{\beta_4}{Q^2} + \beta_5 Q^3 + \tfrac{\beta_6}{Q^3} + \beta_7\sqrt{Q} + \tfrac{\beta_8}{\sqrt{Q}} \\ &\quad + \beta_9\sqrt[3]{Q} + \tfrac{\beta_{10}}{\sqrt[3]{Q}} + \beta_{11} e^{\frac{Q}{10000}} + \beta_{12}\ln\frac{Q}{10000}\end{aligned}$$

A process schematic is included in Figure 2.2, 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 data set.

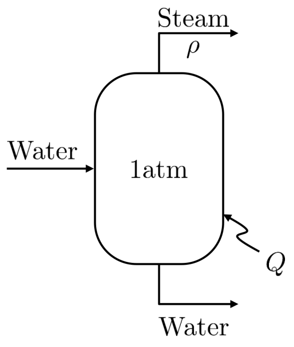


Figure 2.2: Diagram of flash drum

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 $Q \in [13000\,W, 40000\,W]$. 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 $N_{\min}^{\text{ems}} = 1$ and $N_{\max}^{\text{ems}}(N)$ to be the greater of 20% of the current data set 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 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.

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	$0.1086 \ln(0.0001\,Q) - (4.754 \cdot 10^{-10})\,Q^2 + \frac{1.348 \cdot 10^{11}}{Q^3}$
4	$2.339 \sqrt[3]{0.0001\,Q} - 1.385 \ln(0.0001\,Q) - \frac{9518.0}{Q} + \frac{2.075 \cdot 10^{11}}{Q^3}$
5	$80.7 \sqrt[3]{0.0001\,Q} - \frac{39.0}{\sqrt{0.0001\,Q}} - 41.54 \ln(0.0001\,Q) + \frac{3.911 \cdot 10^{11}}{Q^3} - (1.571 \cdot 10^{-13})\,Q^3$
6	$\frac{27.06}{\sqrt{0.0001\,Q}} - 0.000908\,Q + 84.94\sqrt{0.0001\,Q} - 91.6 \sqrt[3]{0.0001\,Q} - \frac{1.159 \cdot 10^{5}}{Q} + \frac{4.768 \cdot 10^{11}}{Q^3}$

Table 2.2: Surrogate models built at increasing complexity for Iteration 7 of the flash drum

The $AICc$ is calculated for the one-term model. Afterwards, the best two-term model is identified. In this case, 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 2.3 for $AICc$ versus model size for the previous iteration). Since $AICc$ 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 2.2 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 are

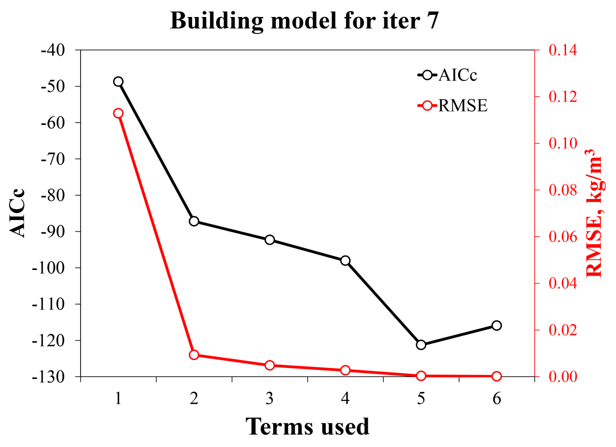


Figure 2.3: Error and goodness-of-fit during the model building for Iteration 7

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.

The mean squared error and $AICc$ for each model size is shown in Figure 2.3. 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 2.1, 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

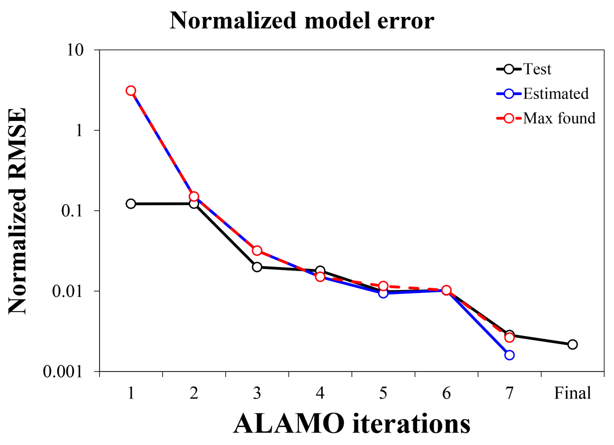


Figure 2.4: Estimated, maximum found, and test model errors found building $\hat{\rho}(Q)$

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 case, the basis functions are fixed, the sparse β vector is updated using a simple least squares estimation with the latest data set, and the algorithm terminates.

Figure 2.4 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 2.4. 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 2.5 and 2.6, and Table 2.3. Figures 2.5 and 2.6 show several snapshots during execution of the algorithm. The models for Iterations 1, 3, 5, and 7 are shown alongside the true

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	$0.4522 \ln(0.0001\,Q) - (1.418 \cdot 10^{-5})\ Q + \frac{1.409 \cdot 10^{12}}{Q^3}$
6	1	3	$0.4094 \ln(0.0001\,Q) - (1.262 \cdot 10^{-5})\ Q + \frac{1.39 \cdot 10^{12}}{Q^3}$
7	11	5	$58.9 \sqrt[3]{0.0001Q} - \frac{60.94}{\sqrt{0.0001\,Q}} - 45.15 \ln(0.0001\,Q) + \frac{3.524 \cdot 10^{12}}{Q^3} - (6.259 \cdot 10^{-15})\ Q^3$
Final	NA	5	$62.37 \sqrt[3]{0.0001Q} - \frac{64.53}{\sqrt{0.0001\,Q}} - 47.81 \ln(0.0001\,Q) + \frac{3.659 \cdot 10^{12}}{Q^3} - (6.576 \cdot 10^{-15})\ Q^3$

Table 2.3: Surrogate model information from modeling the flash drum.

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 2.3). Figures 2.5 and 2.6 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 2.3. After the final iteration, we have a five-term surrogate model of the following form:

$$\begin{aligned}\hat{\rho}(Q) \quad &= \quad 62.37\,\sqrt[3]{0.0001Q} - \tfrac{64.53}{\sqrt{0.0001\,Q}} - 47.81\,\ln(0.0001\,Q) \\ &\qquad + \tfrac{3.659\cdot 10^{12}}{Q^3} - (6.576\cdot 10^{-15})\,Q^3.\end{aligned}$$

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

2.4 Implementation and computational results

ALAMO combines a tailored model generation solver with an error maximization sampling 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 SNOBFIT (Stable Noisy Optimization by Branch and FIT) [47] for adaptive sampling. MINQ [78] 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 (2.2) using all of the available basis functions.

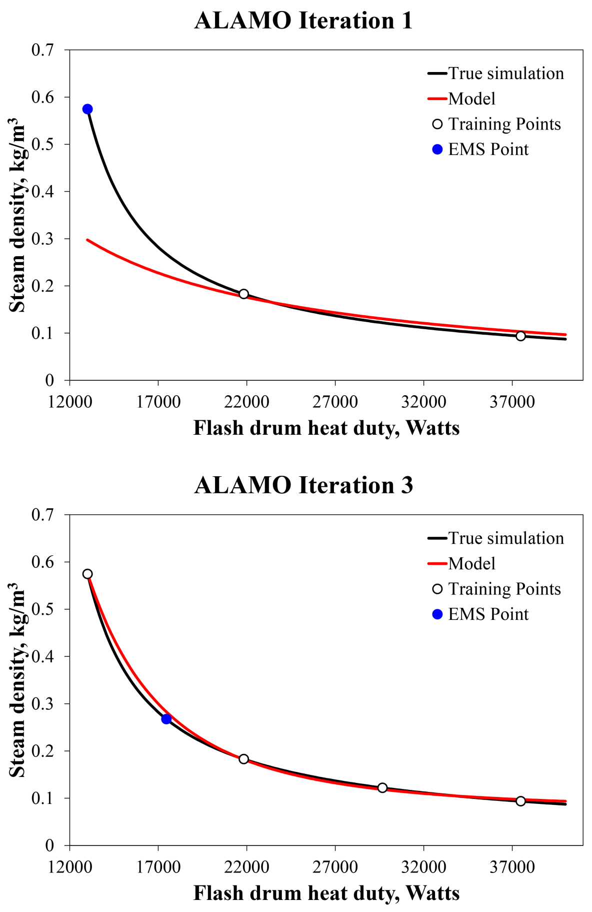


Figure 2.5: Comparison of the true simulated data and the current surrogate models for Iterations 1 and 3 along with current and EMS data sets.

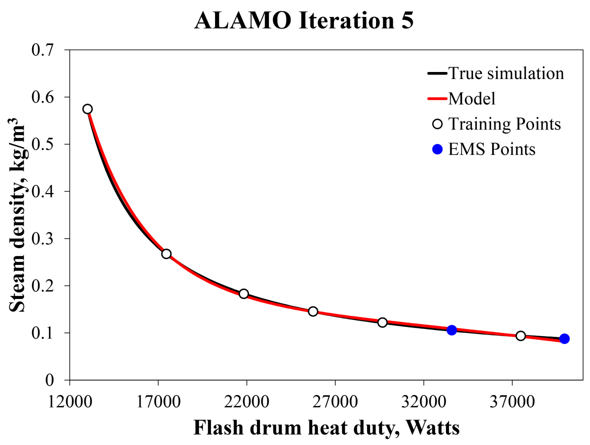
2. Exhaustive search best subset method (EBS)

   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



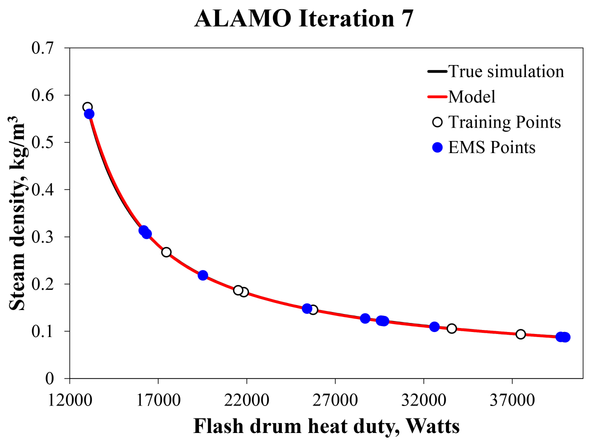


Figure 2.6: Comparison of the true simulated data and the current surrogate models for Iterations 5 and 7 along with current and EMS data sets.

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 to for surrogate modeling with two and three input variables. Basis functions from Table 2.1 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

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	$z(x) = \dfrac{\beta}{\gamma + x_i^{\alpha}}$

Table 2.4: Functional forms for test set B.

from two to ten 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 2.4. 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 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, a test function is modeled using the full algorithm a second model is generated subsequently using a single Latin hypercube (SLH) with the same number of data points. In this way, we are able to calculate the amount of information per data point extracted by the EMS sampling method compared to a Latin hypercube design of experiments. The resulting normalized test error is calculated as follows:

$$e_k^{\text{norm}} = \frac{1}{|\mathcal{T}|} \frac{\sqrt{\displaystyle\sum_{i \in \mathcal{T}} (z_{ik} - \hat{z}_k(x_i))^2}}{\displaystyle\max_{i \in \mathcal{T}} z_{ik} - \min_{i \in \mathcal{T}} z_{ik}} \tag{2.8}$$

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 2.7 and 2.8, and Tables 2.5 and 2.6.

Figures 2.7 and 2.8 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 Equation 2.8. 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 2.5 and 2.6 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 since 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 2.9 and 2.10 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. A smaller deviation signifies fewer required terms in the final surrogate model. The results

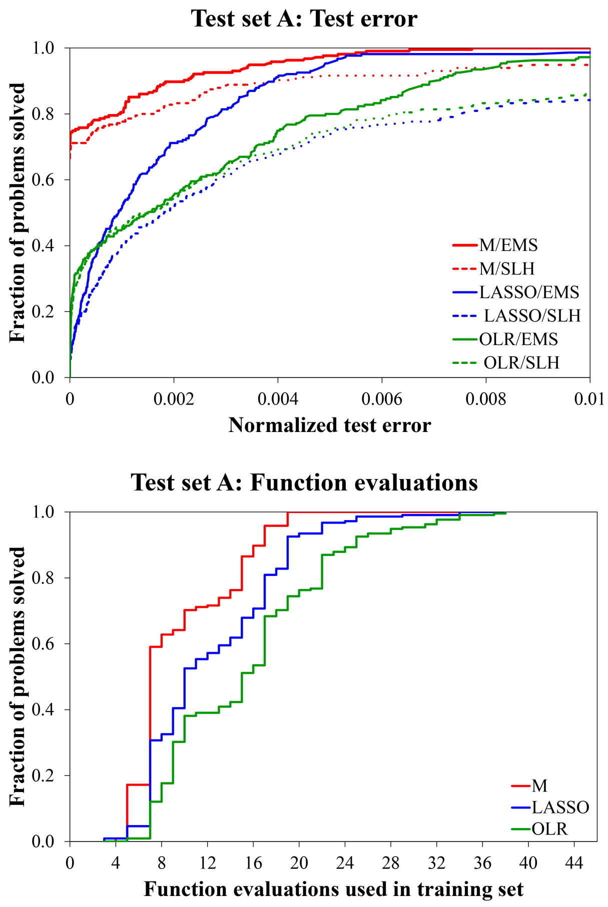
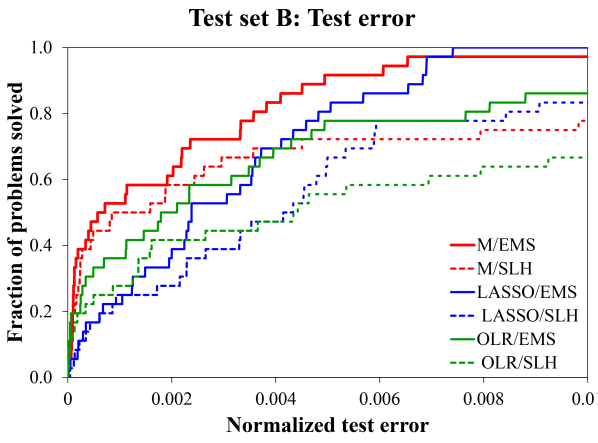


Figure 2.7: *Top:* Fraction of functions in test set A modeled within a normalized test error. *Bottom:* Fraction of functions in test set A modeled within a specified number of function evaluations. Since the EMS and SLH sampling schemes use the same number of data points, only the differences between models is important here. *Note:* EBS is not plotted since the CPU time was too great for several of the more complex problems.



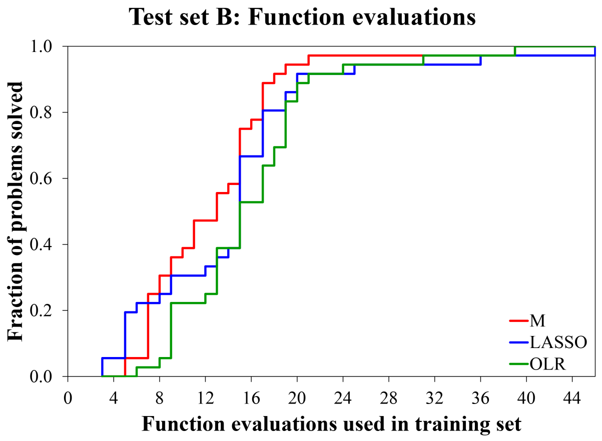


Figure 2.8: *Top:* Fraction of functions in test set B modeled within a normalized test error. *Bottom:* Fraction of functions in test set B modeled within a specified number of function evaluations.

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 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 function 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 $\hat{z}(x) = \sqrt{x} + 0.1\,x + 1$ and $\hat{z}(x) = 1.27\log(x) + 0.21\,x + 1.8$ have differing sets of basis functions. However, these two

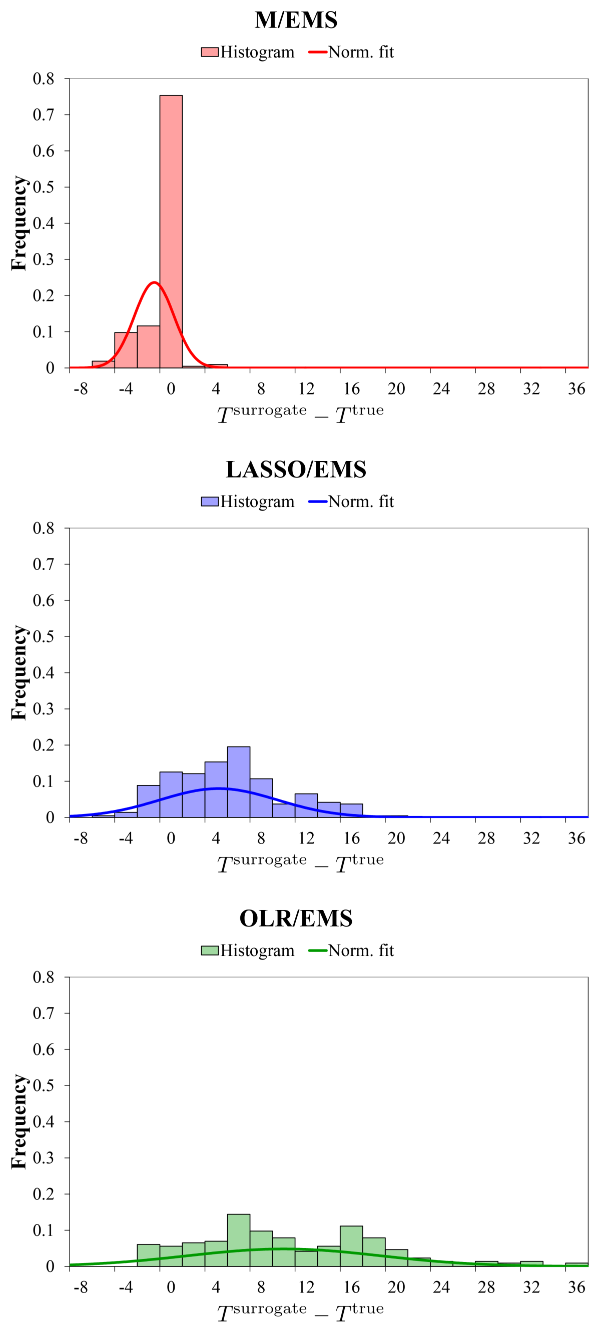


Figure 2.9: Model complexity comparison for each modeling method using EMS from test set A

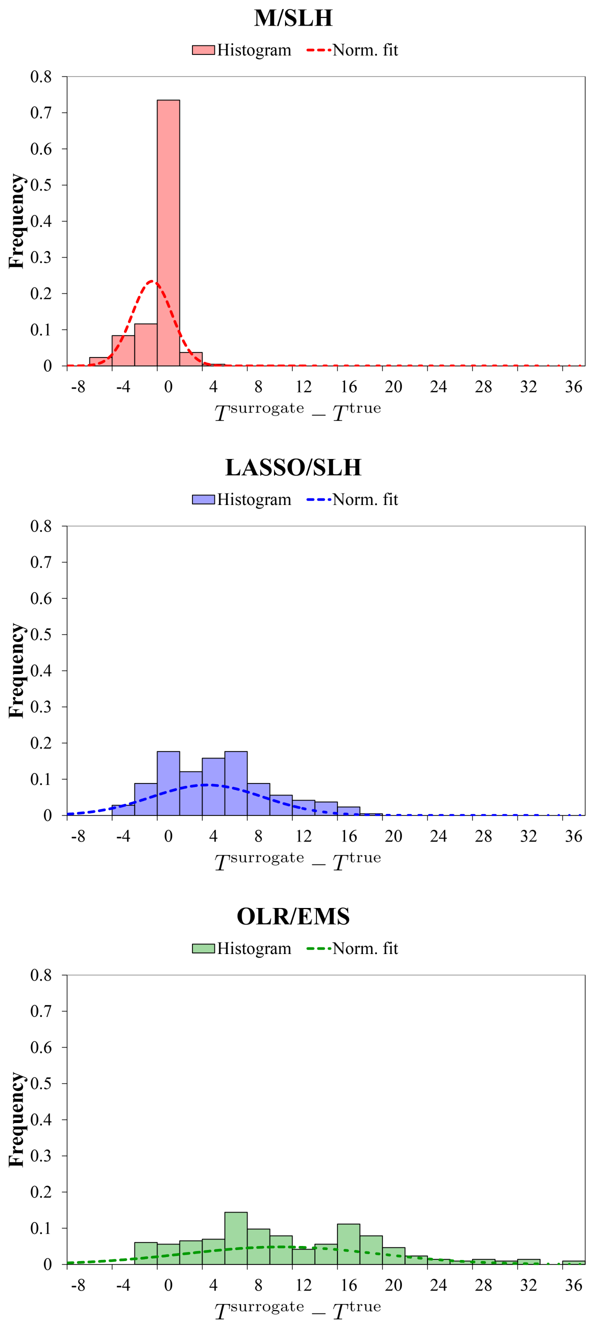


Figure 2.10: Model complexity comparison for each modeling method using a single Latin hypercube from test set A

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*	NA*	[6, 14]	[7, 12]	[10, 17]	[10, 17]
2	10	[4, 8]	[4, 8]	NA*	NA*	[5, 14]	[7, 14]	[10, 14]	[10, 14]

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

Table 2.5: The average minimum and maximum number of surrogate model terms for test set A.

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.

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]

Table 2.6: 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 2.7: 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

2.5 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 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 [68]. Figure 2.11 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.

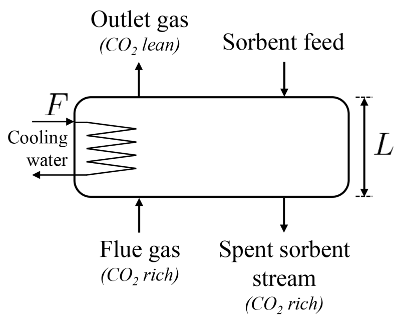


Figure 2.11: Diagram of the solid sorbent carbon capture adsorber

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 a significant impact 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:

$$r = \frac{\text{CO}_2 \text{ in outlet flow}}{\text{CO}_2 \text{ in flue gas}}. \tag{2.9}$$

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 $F \in \left[16.7\,\frac{\text{kmol}}{\text{s}}, 1666.7\,\frac{\text{kmol}}{\text{s}}\right]$ using the model builder M and the EMS adaptive sampling routine. Since 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 2.12. 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.

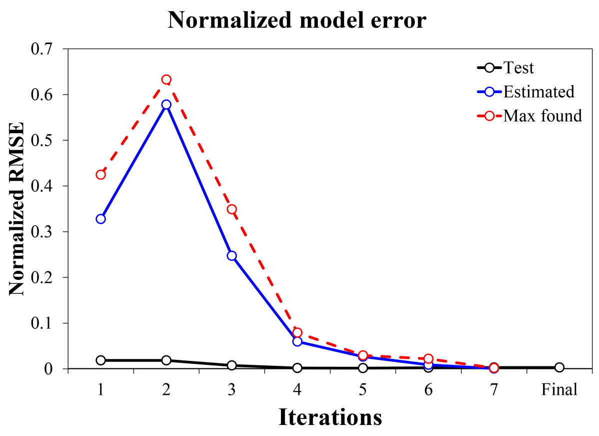


Figure 2.12: Normalized error progression through the entire algorithm

Figures 2.13 and 2.14 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 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 2.8.

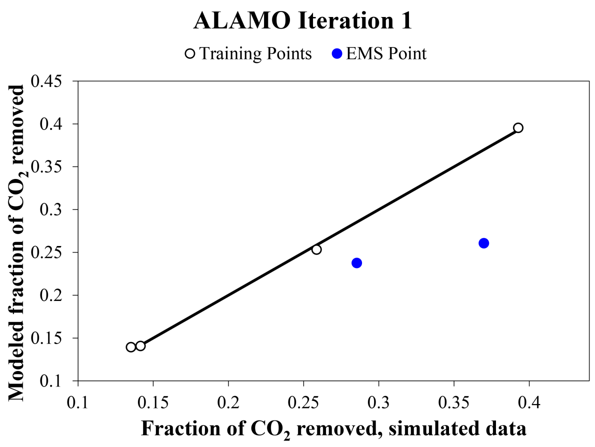
Figure 2.15 shows the solution path as the complexity of the model is allowed to increase from one to thirteen basis functions. The complete basis set has 67 terms available to build each model. The decrease in error seen in Figure 2.15 from twelve to thirteen terms does not justify the use of a thirteen term model. The final model chosen is the following twelve-term model:

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

Once the model has been generated it can be used to analyze the trade-offs among bed depth and cooling water flowrate and percentage of CO_2 removed from the entering gas. Figure 2.16 presents the results of the pareto analysis showing the trade-offs

Table 2.8: Surrogate models built at each iteration for the BFB adsorber

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



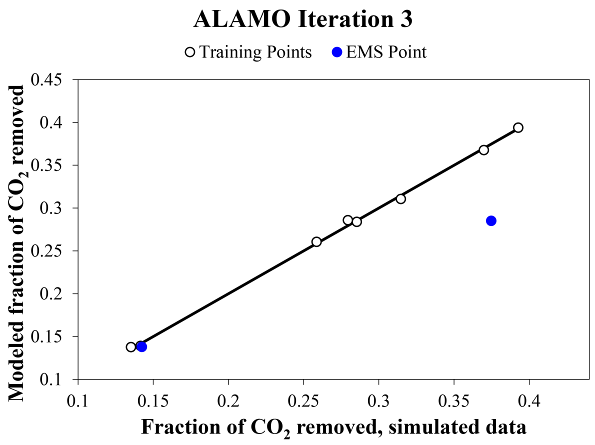


Figure 2.13: Modeled current and EMS data for Iterations 1 and 3 of the proposed methodology compared with true simulated data

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 the National Energy Technology Laboratory power systems financial model [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.

While 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

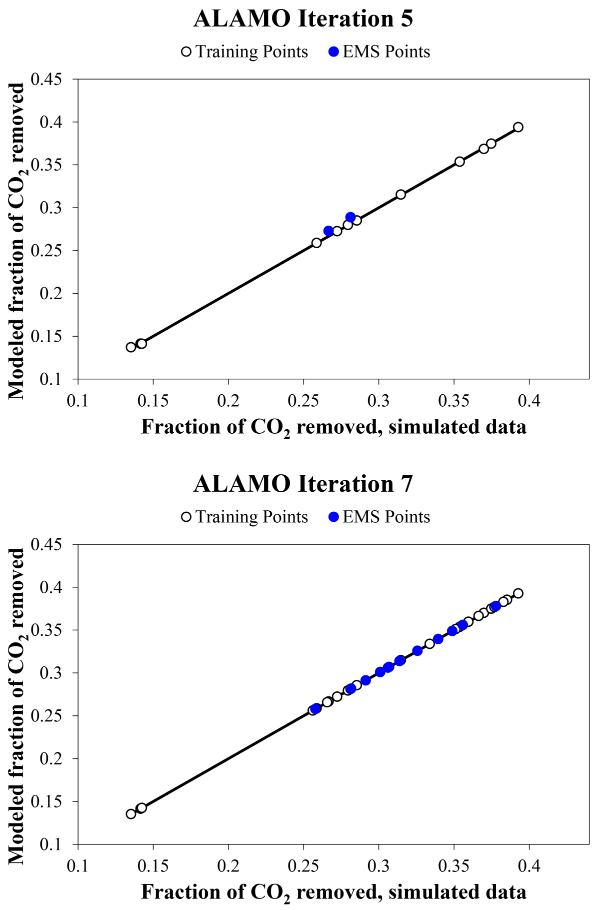


Figure 2.14: Modeled current and EMS data for Iterations 5 and 7 of the proposed methodology compared with true simulated data

the first derivatives of the rigorous simulation. After an optimum is located, this can be verified in several ways. Fully linear models [22] 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 methods to define flowsheets and search complicated topologies to fix integer decision variables and discover favorable starting points for continuous variable derivative-free optimization [76]. To verify the points on the pareto curve in Figure 2.16, 394 evenly sampled test points are evaluated and are plotted on the same axis. Each of these points represents a

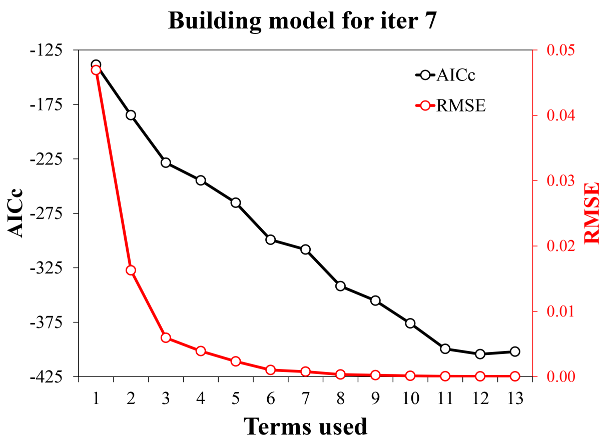


Figure 2.15: Error and goodness-of-fit during the model building for Iteration 7

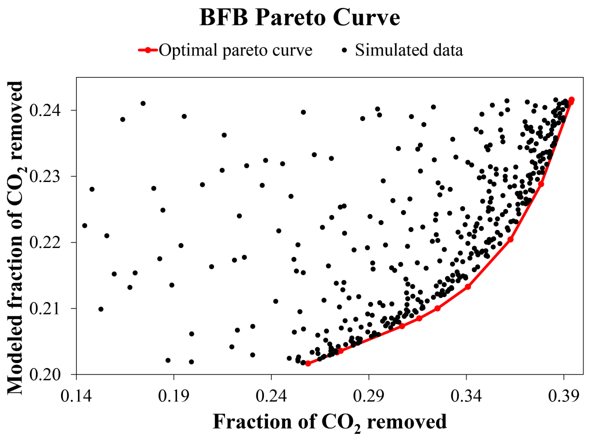


Figure 2.16: Pareto analysis of the BFB adsorber

feasible point, though most represent suboptimal process conditions since 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.

2.6 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 chapter. The proposed methodology is equally applicable for fitting experimental data.

Chapter 3

Constraining regression problems in the predictor and response variable domains

We address an a central theme of empirical model building: the incorporation of information in the model building problem. By enabling modelers to leverage all available information, regression models can be constructed using both empirical data but as well as theory-driven knowledge of response variable bounds, thermodynamic limitations, and boundary conditions, among others.

We expand the inclusion of regression constraints beyond intraparameter relationships to relationships between combinations of predictors and response variables. Since the functional form of these constraints is more intuitive, they can be used to reveal hidden relationships between regression parameters that are otherwise unknown to the modeler. We present a semi-infinite programming approach to the incorporation of these novel constraints then offer several application areas and computational results.

3.1 Introduction

Often, modelers are faced with the decision of (a) utilizing first principles models, intuition, etc. or (b) building a surrogate model using empirical data. In the case of an incomplete first principles analysis, modelers are forced to forgo theoretical derivations and construct regression models based on experimental data. We aim to augment empirical regression with first principles information, intuition, and other *a priori* system characterization techniques to build accurate, physically-realizable models. By doing this, we leverage the synergistic effects of empirical data, first-principles derivation, and intuition. Observed data points are often sampled at a premium, incurring costs associated with computational time, raw materials, and/or other resources. Frequently, additional insights provided by system knowledge, intuition, or the application of first principles analysis are available without additional computational, financial, or other types of costly resources requirements. Knowledge of a less exact nature, including limits on the response variables; known relationships between response and predictor variables; and relationships among responses, can be applied in conjunction with experimental data. For example, ensuring the nonnegativity of a modeled geometric length, enforcing a sum-to-one constraint on modeled chemical compositions, and ensuring that derivative bounds obey thermodynamic principles are all practical applications of beneficial nonempirical insights.

We aim to build regression models (U):

$$\text{(U)} \quad \min_{\beta \in \mathcal{A}} \qquad g(\beta; x_1, x_2, \ldots, x_N, z_1, z_2, \ldots, z_N)$$

that determine m regression parameters or coefficients β that minimize a given loss function g (e.g., squared error, regularized error, information criterion, etc.) over a set of original regression constraints $\mathcal{A}$ based on data points (x_i, z_i), $i = 1 \ldots N$. For conciseness, we will refer to the $g(\beta; x_1, x_2, \ldots, x_N, z_1, z_2, \ldots, z_N)$ data points as $g(\beta)$.

To formally define this insightful nonempirical information, we would like to enforce the following constraint on a regression problem:

$$\Omega(\mathcal{X}) := \{\beta \in \mathbb{R}^m : f\left[x, \hat{z}(x;\beta)\right] \leq 0,\ x \in \mathcal{X}\} \tag{3.1}$$

where function f is a constraint in the space of the predictor(s) x and modeled response(s) $\hat{z}$, and $\mathcal{X}$ is a nonempty subset of $\mathbb{R}^n$. Equation 3.1 can be used to reduce the feasible region Ω for any general regression analysis formulation: linear least squares, nonlinear least squares, regularized regression, best subset methods, and other characterization techniques.

By using system knowledge beyond sampled data, we will refine the feasible domain as the intersection of $\mathcal{A}$ and Ω and solve problem (C):

$$\text{(C)} \quad \min_{\beta \in \mathcal{A} \cap \Omega(\mathcal{X})} \quad g(\beta)$$

where Ω is defined over the domain $x \in \mathcal{X}$ while the original regression problem (U) exists in the space $\beta \in \mathcal{A}$.

Rao [83] and Bard [10] were the first to use *a priori* parameter relationships in the form of simple equality constraints. Recently, the use of such relationships has expanded to include inequality constraints in the space of the regression parameters, a case that arises more naturally in practice [55]. Inequality relationships between regression parameters have been applied to both linear and nonlinear least squares problems in the fields of statistics [51, 69], economics [86, 97], and engineering [37]. Most notably, Korkhin has investigated the properties of simple parameter restrictions [56, 59, 60], nonlinear parameter restrictions [57, 58], and, more recently, the formulation of inequality constraints with deterministic and stochastic right-hand-sides [61].

Previous work employs *a priori* knowledge to reveal relationships between subsets of regression parameters that serve to restrict their range. To our knowledge, there has been no investigation into the enforcement of *a priori* information relating the predictors and regressors. We aim to use these relationships between predictors and regressors to restrict the feasible region of the original problem space.

Since previous applications of constrained regression have been restricted to the parameter space β of the regression problem, these techniques are inherently specific to the functional form of the response. For example, consider a quadratic response model, $\hat{z}(x) = \beta_0 + \beta_1 x + \beta_2 x^2$, and the *a priori* insight $\beta_1 \geq \beta_2$. If an exponential function, $\hat{z}(x) = \beta_0 + \beta_1 \exp(x)$, produces a more favorable fit, there is no standard way to translate constraints from the quadratic to the exponential model. Enforcing a lower bound on the response, $\hat{z}(x) \geq 0 \ \forall x$, instead of the β-space produces a constraint that is independent of the model's functional form. Additionally, system insight in the x-domain may be more intuitive and readily available than knowledge of a single contrived regression model's functional form.

A complication arises from the realization that Equation 3.1 is valid for the full problem space and, therefore, needs to be enforced for every point $x \in \mathcal{X}$. As a result, the feasible region is semi-infinite. Semi-infinite programming (SIP) problems are optimization models that have finitely many variables and infinitely many constraints [85]. These problem formulations are common in the fields of approximation theory, optimal control, and eigenvalue computations, among others. In each case, one or more semi-infinite constraints result in one constraint for each value of an optimization parameter (in this case x) that varies within a given domain [44, 85].

The first significant work on SIP, due John [48], provides the necessary and sufficient conditions for the solution to a semi-infinite program. Initially, SIP research focused on linear and convex nonlinear semi-infinite programming [39, 44, 85]. More recently, advances in global optimization, including `BARON` [96], have made the solution of general nonconvex SIP problems more practical. In problem (C), the objective is often convex, as is the case for linear least squares regression. The feasible region, however, is generally nonlinear nonconvex as we will show in Section 3.4. The key to solving a SIP problem, independent of the solution method, is the optimization of $\max_{x \in \mathcal{X}} f\left[x, \hat{z}(x;\beta)\right]$ to locate the maximum violation. This subproblem is significant because $\beta \in \Omega(\mathcal{X})$ if and only if $\max_{x \in \mathcal{X}} f\left[x, \hat{z}(x;\beta)\right] \leq 0$ [84].

To assess the benefits afforded by augmenting standard regression problems (U) with *a priori* information as in (C), we utilize the test platform `ALAMO` [24]. `ALAMO` is a software package designed to generate models that are as accurate and as simple as possible. This combination of accuracy and simplicity is well-suited for regression.

The remainder of the paper is organized as follows. In Section 3.2, we outline the modeling and sampling methods of the `ALAMO` test platform. We propose a methodology to solve a semi-infinite problem in its most general form in Section 3.3. In Section 3.4, we list applications of problem domain constrained regression using our solution strategy: restricting individual and multiple responses, constraining response model derivatives, and expanding or contracting the enforcement domain. Next, we offer computational studies to demonstrate the effectiveness of the proposed methods in Section 3.5. Finally, we offer conclusions and comments in Section 3.6.

3.2 ALAMO

`ALAMO` is a learning software that identifies simple, accurate surrogate models using a minimal set of sample points from black-box emulators such as experiments, simulations, and legacy code. `ALAMO` initially builds a low-complexity surrogate model using a best subset technique that leverages a mixed-integer programming formulation to consider a large number of potential functional forms. The model is subsequently tested, exploited, and improved through the use of derivative-free optimization solvers that adaptively sample new simulation or experimental points. For more information about ALAMO, see [24].

In this section, we detail relevant ALAMO model-building methods as applied to parametric regression. The functional form of a regression model is assumed to be unknown to ALAMO. Instead, ALAMO poses a simple set of basis functions, say x, x^2, $1/x$, $\log(x)$, and a constant term. Once a set of potential basis functions is defined, ALAMO attempts to construct the lowest complexity function that accurately models sample data. To do this, the following MIQP (Mixed-Integer Quadric Program) is solved for increasing model complexity, T.

$$
\begin{array}{lll}
(A) \quad \min & g(\beta) = \displaystyle\sum_{i=1}^{N} \left(z_i - \sum_{i=1}^{N} \beta_j \, X_j(x_i) \right)^2 & \\
\text{s.t.} & \displaystyle\sum_{j \in \mathcal{B}} y_j = T & \\
& \beta_j^{\text{lo}} y_j \leq \beta_j \leq \beta_j^{\text{up}} y_j & j \in \mathcal{B} \\
& \beta_j \in [\beta_j^{\text{lo}}, \beta_j^{\text{up}}] & j \in \mathcal{B} \\
& y_j \in \{0, 1\} & j \in \mathcal{B}.
\end{array}
$$

In problem (P), the simple basis functions, $X_j(x)$, $j \in \mathcal{B}$, are active when the corresponding binary variable $y_j = 1$ and inactive when $y_j = 0$. Here, set $\mathcal{A}$ represents the feasible region of problem (A) as a function of regression coefficients β and activity indicators y. The size of the model, specified by the parameter T in first constraint, is increased until a goodness-of-fit measure, such as the corrected Akaike Information Criterion [46], worsens with an increase in model size.

Using the list of basis functions given above, the optimization problem is as follows.

$$
\begin{aligned}
\text{(M)} \quad \min \quad & g(\beta) = \sum_{i=1}^{N} \left(z_i - \left[\beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 \frac{1}{x} + \beta_4 \log(x) \right] \right)^2 \\
\text{s.t.} \quad & y_0 + y_1 + y_2 + y_3 + y_4 = T \\
& \beta^{\text{lo}} y_0 \le \beta_0 \le \beta^{\text{up}} y_0 \\
& \beta^{\text{lo}} y_1 \le \beta_1 \le \beta^{\text{up}} y_1 \\
& \beta^{\text{lo}} y_2 \le \beta_2 \le \beta^{\text{up}} y_2 \\
& \beta^{\text{lo}} y_3 \le \beta_3 \le \beta^{\text{up}} y_3 \\
& \beta^{\text{lo}} y_4 \le \beta_4 \le \beta^{\text{up}} y_4 \\
& y_0, y_1, y_2, y_3, y_4 \in \{0, 1\} \\
& \beta_j^{\text{lo}} \le \beta_0, \beta_1, \beta_2, \beta_3, \beta_4 \le \beta_j^{\text{up}}
\end{aligned}
$$

While a typical basis set is often far larger, this simple example illustrates the form of the objective $g(\beta)$ and original constraint set $\beta \in \mathcal{A}$ before intersection with new *a priori* constraints $\Omega(\mathcal{X})$.

Once a model has been identified, it is improved systematically through the use of an adaptive sampling technique that adds new simulation or experimental points to the training set. New sample points are selected to maximize model inconsistency in the original design space, $x \in \mathcal{X}$, as defined by box constraints on x, using derivative-free optimization methods [87].

3.3 Proposed methodology

In this section, we outline a numerical solution method for semi-infinite programming problem (C) and demonstrate key algorithmic insights using an illustrative example. Problem (C) is solved using the standard, two-phase SIP method. For details on this approach see [84]. In Phase I, we solve a relaxation of (C), where the semi-infinite constraint is only enforced over a finite subset $x \in \mathcal{X}^l \subset \mathcal{X}$:

$$\text{(PI)} \quad \min_{\beta \in \mathcal{A} \cap \Omega(\mathcal{X}^l)} \qquad g(\beta)$$

By solving (PI), we find an approximation of the regression parameters β^l over the relaxed feasible region defined below.

$$\Omega(\mathcal{X}^l) := \left\{\beta \in \mathbb{R}^m : f\left[x, \hat{z}(x;\beta)\right] \leq 0,\ x \in \mathcal{X}^l\right\} \tag{3.2}$$

Phase II involves solving the maximum violation problem as given in (PII):

$$\text{(PII)} \quad \max_{x \in \mathcal{X}} \qquad f\left(x, \hat{z}(x;\beta^l)\right)$$

Typically, after (PI) is solved, its solution values β^l are used to solve (PII). If the solution to (PII) satisfies $f(x^l) \leq 0$, the method terminates; otherwise, an updated feasible set $\mathcal{X}^l = \mathcal{X}^l \cup x^l$ is used to repeat Phase I. In general, (PI) will preserve the convexity of the original regression problem (U), though several specific exceptions to this observation are listed in Section 3.4. For most linear regression problems, the feasible region $\Omega(\mathcal{X}^l)$ is linear. (PII), however, is generaly nonconvex and necessitates the use of a global optimization solver.

Žaković and Rustem [110] solve (PII) to find (a) the maximum violation and (b) any feasible violation. In both cases, they employ a global search strategy using a multistart local optimization approach. For (b), the complete solution of (PII) is required before feasibility to the original problem (C) can be proven. In their findings, (a) requires fewer iterations, while (b) results in a significant reduction of computational effort. The authors indicate that (b) is the superior solution method. We propose a method that is similar to (b), but we employ an optimization solver that can rigorously guarantee global optimality. Additionally, we seek to locate several isolated feasible solutions to (PII) during Phase II with the aim of reducing the total number of constrained regression iterations.

We begin the algorithm with either an empty feasible set $\mathcal{X}^0 = \emptyset$ or some nonempty set of feasible points, for example those selected by a design of experiments (e.g., random sampling, Latin-hypercube sampling [74], factorial design [92], etc.). Next, we solve (PI) to find an initial approximation for β^0. Using the built-in functionality of the global optimizer BARON, we solve (PII$_{\text{feas}}$) with β^0 to locate up to n_{viol} isolated feasible points

$$
\begin{aligned}
(\mathrm{PII}_{\text{feas}}) \quad & \max_{x \in \mathcal{X}} && f\left(x, \hat{z}(x; \beta^l)\right) \\
& \text{s.t.} && f\left(x, \hat{z}(x; \beta^l)\right) - \epsilon_{\text{viol}} \geq 0
\end{aligned}
$$

Often, feasible solutions in continuous optimization problems are located close enough to each other as to be nearly identical. We ensure that the feasible points are not redundant by selecting isolated feasible solutions such that $\|x_i^0 - x_{i'}^0\|_\infty \geq \epsilon_{\text{isol}}$ for every pair of points i and i' [88]. By solving (PII$_{\text{feas}}$) using an objective function that reflects the magnitude of violation, we enable BARON to locate a set of feasible points with comparatively large, ranked violations. We also exclude points $f\left(x, \hat{z}(x; \beta^l)\right) = 0$ that are feasible to the original problem (C) by using a small number ϵ_{viol} to ensure a strict violation.

After updating $\mathcal{X}^{l+1} = \mathcal{X}^l \cup x^l$, (PI) is solved again with an updated feasible region $\Omega(\mathcal{X}^{l+1})$. We proceed until it can be shown that the current iteration's parameters, β^l, are both optimal and feasible for (C). This is true if and only if β^l is the solution to (PI) and $\max\limits_{x \in \mathcal{X}} \; f\left(x, \hat{z}(x; \beta^l)\right) \leq 0$. An outline of the proposed semi-infinite constrained regression algorithm is included in Algorithm 3.

Algorithm 3 Solve Semi-Infinite Constrained Regression Problem

```
Given a training set of dependent and independent data points $[x_{id}, z_{ik}]$ for $i = 1, \dots, N,\ d = 1, \dots, n,\ k = 1, \dots, m$; requested feasible points $n_{\text{viol}}$; and relevant tolerance values

Initialize $l = 0$ and $\mathcal{X}^0$ as $\emptyset$ or by using a design of experiments
while $(f(x^l) > \epsilon)$ or $(l < 1)$ do
    $\beta^l \leftarrow$ solve (PI)
    $x^l, f(x^l) \leftarrow$ find $n_{\text{viol}}$ isolated feasible points for (PII$_{\text{feas}}$)
    if $x^l == \emptyset$ then
        $\beta^l$ is both feasible and optimal to (C)
        break
    else
        Update feasible set, $\mathcal{X}^{l+1} \leftarrow \mathcal{X}^l \cup x^l$
    end if
    $l \leftarrow l + 1$
end while
```

3.3.1 Illustrative example

To detail the steps involved in the proposed technique, we construct a model for $z = x^5$ using a fixed functional form $\hat{z}(x) = \beta_1 x + \beta_2 x^3$ over $0 \leq x \leq 1$. The true function is nonnegative over this region; therefore, we pose the following constrained regression problem:

$$\begin{aligned} \min_{\beta_1, \beta_2} \quad & \sum_{i=1}^{4} \left(z_i - \left[\beta_1 x + \beta_2 x^3 \right] \right)^2 \\ \text{s.t.} \quad & \beta_1 x + \beta_2 x^3 \geq 0 \qquad\qquad x \in [0, 1] \end{aligned}$$

where model parameters β_1^l and β_2^l are selected to minimize model error over four data points while ensuring nonnegativity in the response model. An ordinary least squares regression objective for a regression model with a fixed functional form is used to illustrate this example more concisely. The discretized Phase I problem:

$$\begin{aligned} \min_{\beta_1,\beta_2} \quad & \sum_{i=1}^{4} \left(z_i - \left[\beta_1\, x + \beta_2\, x^3\right]\right)^2 \\ \text{s.t.} \quad & \beta_1\, x + \beta_2\, x^3 \geq 0 \qquad\qquad\qquad\qquad x \in \mathcal{X}^l \end{aligned}$$

is formulated by enforcing the semi-infinite constraint for discrete values of x in the feasible set $\mathcal{X}^l$ for constrained regression iteration l. We use following Phase II problem:

$$\begin{aligned} \min \quad & \beta_1^l\, x + \beta_2^l\, x^3 \\ \text{s.t.} \quad & \beta_1^l\, x + \beta_2^l\, x^3 \leq 0 - \epsilon \\ & 0 \leq x \leq 1 \end{aligned}$$

to maximize the nonnegativity violation, where feasible solutions to this problem x^l are added to the feasible set before the Phase I problem is resolved. If the maximum violation problem is infeasible for iteration l, the model parameters β_1^l and β_2^l are feasible and optimal to the semi-infinite problem.

Table 3.1 shows the regression models from each Phase I solution. Since the initial feasible set $\mathcal{X}^l$ is empty, the Phase I model for $l = 1$ matches the ordinary least squares or unconstrained regression model. The training and test errors are included in Table 3.1. Here, the training error is calculated using the four training points and the test error is calculated using 1000 evenly-distributed sample points. For all results, unless stated otherwise, all errors are calculated using the root mean squared error between a given data set and a model. During each iteration, the newly imposed constraints worsen the objective or training error by restricting the feasible space. Despite this degradation of the objective function, the added constraints result in improved test error during each iteration. This is because the squared error objective is an approximation of the true objective or true error.

Table 3.1: Surrogate models and errors for the illustrative example

Surrogate models		Training error, 10^{-3}	Test error, 10^{-3}
Constrained regression:			
Phase I ($l = 1$):	$\hat{z}(x) = -0.308\,x + 1.30\,x^3$	19.1	0.172
Phase I ($l = 2$):	$\hat{z}(x) = -0.588\,x + 1.02\,x^3$	34.4	0.0303
Final:	$\hat{z}(x) = 0.000\,x + 0.954\,x^3$	40.2	0.00353
Ordinary least squares regression:			
Final:	$\hat{z}(x) = -0.308\,x + 1.30\,x^3$	19.1	0.172

Here, we introduce an error comparison metric representing the error factor calculated for method m using the following equation:

$$EF_m = \frac{\text{RMSE}_m}{\text{RMSE}_{\text{best}}} \tag{3.3}$$

where RMSE_m is the root mean squared error for the method of interest and $\text{RMSE}_{\text{best}}$ is the error of the best solution for a given modeling problem. Therefore, the best modeling method has an EF$= 1$ while larger values quantify the diminishing success exhibited by other methods. The training error factors for the unconstrained and the constrained problem of the above example are 1 and 2.11, respectively. This shows that the unconstrained problem is more accurate on the training data set. In contrast, the test error factors for the unconstrained and constrained problems are 48.9 and 1, which shows that the constrained model is far more effective at predicting the values of future samples.

For each Phase II iteration, we add two points to the feasible set. These points are listed in Table 3.2. By adding multiple feasible points per round, we avoid unnecessary Phase I solutions during trial model generation. This is significant because the solution of Phase I is often resource intensive.

Table 3.2: Feasible set for illustrative example

Constrained regression iteration, l	Point added to feasible set, x^l	Modeled value, $\hat{z}(x^l)$
1	0.240	-0.0560
	0.281	-0.0578*
2	0.120	-0.0053
	0.138	-0.0054*
3	Bounds guranteed	

* Optimal solution

To illustrate the form of the Phase I and Phase II problems, an instance of each problem corresponding to iteration 2 is included below.

Phase I ($l = 2$):

$$\begin{aligned} \min_{\beta_1,\beta_2} \quad & \sum_{i=1}^{4} \left(z_i - \left[\beta_1\, x + \beta_2\, x^3\right]\right)^2 \\ \text{s.t.} \quad & 0.240\,\beta_1 + 0.0138\,\beta_2 \geq 0 \\ & 0.281\,\beta_1 + 0.0223\,\beta_2 \geq 0 \end{aligned}$$

Phase II ($l = 2$):

$$\begin{aligned} \min \quad & -0.588\,x + 1.02\,x^3 \\ \text{s.t.} \quad & -0.588\,x + 1.02\,x^3 \leq 0 - \epsilon \\ & 0 \leq x \leq 1 \end{aligned}$$

Increasing the size of the feasible set allows β_1^l and β_2^l to move closer to the feasible space defined by $\beta_1\, x + \beta_2\, x^3 \geq 0$. This feasible region as well as the constrained parameter solutions are depicted in Figure 3.1 to illustrate discretized solution improvement with respect to feasibility of the semi-infinite problem. The true function, ordinary least squares regression model, and constrained regression model are plotted with the training data and feasible set in Figure 3.2.

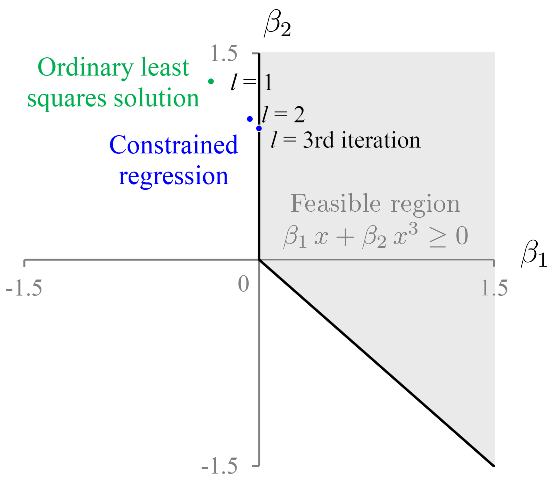


Figure 3.1: Constrained regression feasible region in β-space

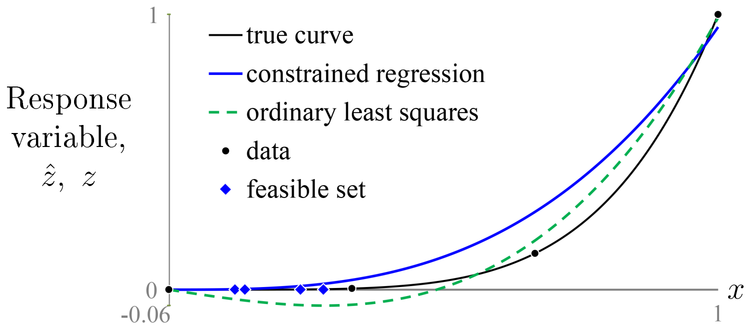


Figure 3.2: Model results and true function for $z = x^5$

By constraining the regression model in the space of the predictor and response variables using freely available *a priori* information, we are able to infer constraining relationships in the β-space that generate physically realizable models that better predict the response surface.

3.4 Classes of constrained regression in the x- and z-domain

In this section, we describe several classes of problems with structures that benefit naturally from constraining the original problem space. Initially, we consider constraining individual responses—a direct application of the methods discussed in the previous section. Next, we extend the proposed methods to the enforcement of relationships among several response variables. Finally, we discuss two specialized applications: restricting response derivatives and an expansion and contraction of the enforcement domain.

3.4.1 Restricting individual responses

Constraints placed independently on individual response variables result in the semi-infinite feasible region:

$$\Omega(\mathcal{X}) := \{\beta \in \mathbb{R}^m : a\,\hat{z}(x;\beta) + h(x) \leq 0,\ x \in \mathcal{X}\} \tag{3.4}$$

where h is a function of the predictors x and the coefficient $a \in \mathbb{R}$ effectively scales the response model. We classify constraints of this form by the order of $h(x)$. Zero-order constraints can be used to enforce upper and lower limits on $\hat{z}$; while higher-order constraints can be used to enforce complex restrictions on the feasible region of each response model. As in Equation 3.4, the discretized Phase I constraints for this problem are linear in β for linear regression. In this section, we restrict our investigation to constraints that are linear in $\hat{z}$.

Zero-order restrictions

We begin by discussing an implementation of the proposed methodology on a restriction of the response model via upper and lower bounds. The most common use of this type of bounding is the enforcement of nonnegativity. Examples of nonnegative response variables include flow rates, absolute pressures, and geometric dimensions, among others. In addition, many other response variables have natural lower and/or upper bounds; compositions and probability distributions range from 0 to 1, logistic functions have asymptotic limits, etc..

To enforce these limits, we impose $\hat{z}(x;\beta) - z^{\text{up}} \leq 0$ for the upper bound, z^{up}, and $z^{\text{lo}} - \hat{z}(x;\beta) \leq 0$ for the lower bound, z^{lo}. Computational results for combinations of upper and lower bounds are included in Section 3.5. The application of intuitive bounds on response variables results in the following Phase I and Phase II problem formulations for both lower and upper bounds.

$$
\begin{aligned}
\left(\text{PI}^{\text{bnd}}\right) \quad \min_{\beta \in \mathcal{A}} \quad & g(\beta) & \\
\text{s.t.} \quad & \hat{z}(x;\beta) \leq z^{\text{up}} & x \in \mathcal{X}^l \\
& \hat{z}(x;\beta) \geq z^{\text{lo}} & x \in \mathcal{X}^l
\end{aligned}
$$

$$
\begin{aligned}
\left(\text{PII}^{\text{lobnd}}_{\text{feas}}\right) \quad \min_{x} \quad & \hat{z}(x;\beta^l) \\
\text{s.t.} \quad & \hat{z}(x;\beta^l) \leq z^{\text{lo}} - \epsilon_{\text{viol}} \\
& x \in \left[x^{\text{lo}}, x^{\text{up}}\right]
\end{aligned}
$$

$$
\begin{aligned}
\left(\text{PII}^{\text{upbnd}}_{\text{feas}}\right) \quad \max_{x} \quad & \hat{z}(x;\beta^l) \\
\text{s.t.} \quad & \hat{z}(x;\beta^l) \geq z^{\text{up}} + \epsilon_{\text{viol}} \\
& x \in \left[x^{\text{lo}}, x^{\text{up}}\right]
\end{aligned}
$$

where $\mathcal{X} = \left[x^{\text{lo}}, x^{\text{up}}\right]$ defines the problem space of the original predictors for any given upper and lower bounds on x.

Modeling chemical reactions using statistical or polynomial fitting functions often results in composition profiles that are not physically realizable. In the next example, we examine a batch reactor and two first-order reactions in series:

$$A \xrightarrow{k_a} B \xrightarrow{k_b} C$$

where we model the concentration of B, $[B]$, as a function of batch time $t \in [0.6, 10]$ with kinetic constants $(k_a, k_b) = (0.473, 1.44)$. The reactor has an initial concentration of $[A]_0 = 1$, $[B]_0 = 0$, and $[C]_0 = 0$. Since we know *a priori* that the concentration of B must be nonnegative and less than the initial concentration of A, we impose these bounds to improve the regression model.

For model generation, we use a training set of ten data points from a Latin hypercube design of experiments and a test set of 1000 randomly sampled points. We allow for potential terms including exponentials, logarithms, and several powers: ± 0.5, ± 1, ± 2, ± 3, ± 4. In Section 3.5, we include full computational results for several randomly generated reaction problems. This example is drawn from this test set.

Here, models are generated both with and without restriction. The models and test errors found for both cases are compared in Table 3.3. Both methods result in five-term models. However, the unconstrained method had a test error 7.25 times higher than that of the constrained regression. In fact, every constrained regression model from one to five terms exhibits a test error less than the final unconstrained model.

Up to five points are added to the feasible set during a constrained regression iteration. Table 3.4 includes these points and corresponding modeled values for each point in the feasible set. For all problems discussed, we have started with an empty feasible set which, for this problem, is only expanded during the solution of the two-term model. The new feasible set is then enforced on the following terms' Phase I problem.

The addition of upper and lower limits on the response output significantly enhances model quality without the requirement of additional sampling. The final models for both cases plotted against a curve representing the true concentration of B in Figure 3.3.

Table 3.3: Successive models for $[\hat{B}](t)$

Terms	Test error	Model, $[\hat{B}](t)$
Unconstrained models:		
1	0.0418	$0.261/t$
2	0.248	$1.38/t^2 - 1.38/t^3$
3	0.124	$-0.255/\sqrt{t} + 0.818/t - 0.470/t^3$
4	0.793	$-0.298/t + 4.02/t^2 - 7.91/t^4 + 4.67/t^4$
5: final	0.0856	$0.339 \log t + 2.31/\sqrt{t} - 0.911/t^2 + 0.318/t^4 - 1.494$
Constrained models:		
1	0.0418	$0.261/t$
*2	–	$1.38/t^2 - 1.38/t^3$
*2	–	$-0.102 \log t + 0.219$
2	0.0245	$-0.0325 \log t + 0.236/\sqrt{t}$
3	0.0101	$-0.0609\, t + 0.363\, t^2 + 0.263$
4	0.0171	$-1.13 \log t - 0.805/t + 1.04\sqrt{t} - 0.0604\, t$
5: final	0.0118	$-0.175 \log t + 0.885/\sqrt{t} - 0.711/t + 0.00395\, t^2 - 0.000197\, t^3$

*Unsuccessful trial model

Table 3.4: Feasible set for constraining the regression model for $[B]$

Point added to feasible set, t^l		Modeled value, $[\hat{B}](t^l)$
Iteration l=1:	0.6076	-2.3759
	0.6022	-2.4750
	0.6014	-2.4889
	0.6006	-2.5039
	0.6000	-2.5155
Iteration l=2:	9.9468	-0.0144
	9.9596	-0.0145
	9.9712	-0.0146
	9.9889	-0.0148
	10.0000	-0.0149

Points added during the solution of the two-term model

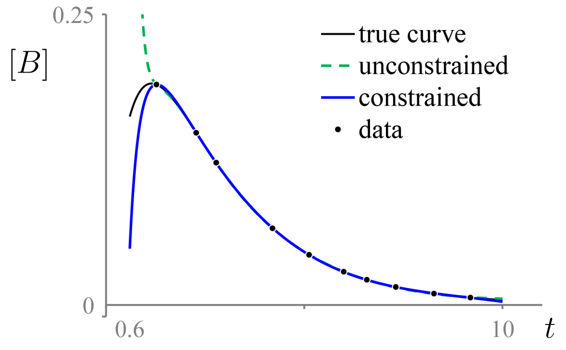


Figure 3.3: Model results for the concentration of B

Nonzero-order restrictions

If nonconstant restrictions on the response variable are known, they can be applied in a similar manner. These constraints rely more heavily upon the modeler's system knowledge than simple bounds, but can lead to increased modeling accuracy and robustness. Linear and nonlinear constraints of this form may come from initial and boundary conditions, mass and energy balances, and problem limits.

Often, knowledge exists for a system that is simpler than the system of interest. Often, that simple system represents a limit or worst-case scenario. In these cases, the simple system can be used to bound the model. For example, for a heat transfer system we can bound heat duty using information from a Carnot engine. If a simple model is available for a reactive system with no byproducts, that model can be used as a lower bound for certain concentrations. As long as the simple system represents a limit and not an approximation, it can be used to bound the output.

3.4.2 Restricting multiple responses

Additional relationships between response variables, z_k, $k = 1, 2, \ldots, n_{\text{resp}}$, may also be available to a modeler. One example of such a relationship is a mass or energy balance involving inlet and outlet flows where two or more flows are response models. Another example is the modeling of discretized state variables, such as a fluid velocity profile in a tube, that results in an intrinsic order of modeled variable values of the form: $\hat{z}_{k'} \leq \hat{z}_{k>k'}$. The feasible region for a simultaneous restriction of multiple response variables can be described as

$$\Omega(\mathcal{X}) := \{\beta \in \mathbb{R}^m : d(\hat{z}(x;\beta)) + h(x) \leq 0,\ x \in \mathcal{X}\}\,, \tag{3.5}$$

where d provides a function for the relationship among all responses $k = 1, 2, \ldots, n_{\text{resp}}$.

In the general case, the solution of this problem requires the simultaneous solution of all response models. As a result, this formulation is not compatible with `ALAMO`, which relies on efficiencies gained by independent treatment of model output variables. In the following subsections, we demonstrate a solution method for the general case and describe an adaptation of this method for the `ALAMO` framework.

General case

As mentioned above, a solution for the general case involves the regression of all response models simultaneously using (PI^{mult}):

$$(\text{PI}^{\text{mult}}) \quad \min_{\beta \in \mathcal{A} \cap \Omega(\mathcal{X}^l)} \quad g^{\text{mult}}(\beta_k)$$

where g^{mult} is the objective of the simultaneous regression problem. The algorithm is shown in Figure 3.4.

The fitting objective for the weighted linear least squares regression, using weighting factors w_K, results in the following formulation and fits each output variable simultaneously:

$$g^{\text{mult}}(\beta_k) = \sum_{k=1}^{n_{\text{resp}}} w_k \sum_{i=1}^{N} \left(z_{ki} - \sum_{j \in \mathcal{B}} \beta_{kj}\, X_{ij} \right)^2 \tag{3.6}$$

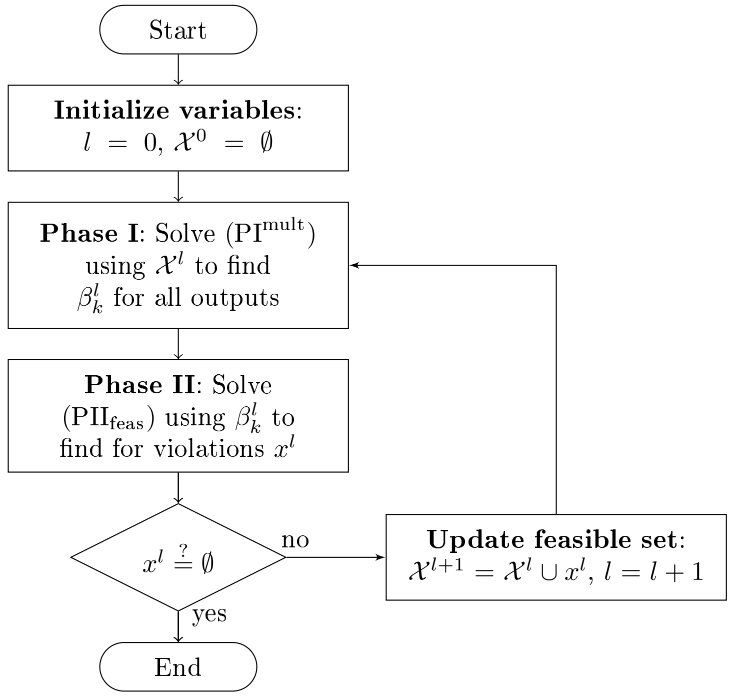

Start
Initialize variables: $l = 0, \mathcal{X}^0 = \emptyset$
Phase I: Solve ($\mathrm{PI}^{\mathrm{mult}}$) using $\mathcal{X}^l$ to find β_k^l for all outputs
Phase II: Solve ($\mathrm{PII}_{\mathrm{feas}}$) using β_k^l to find for violations x^l
$x^l \overset{?}{=} \emptyset$
no
yes
Update feasible set: $\mathcal{X}^{l+1} = \mathcal{X}^l \cup x^l$, $l = l + 1$
End

Figure 3.4: Restricting multiple responses for the general case

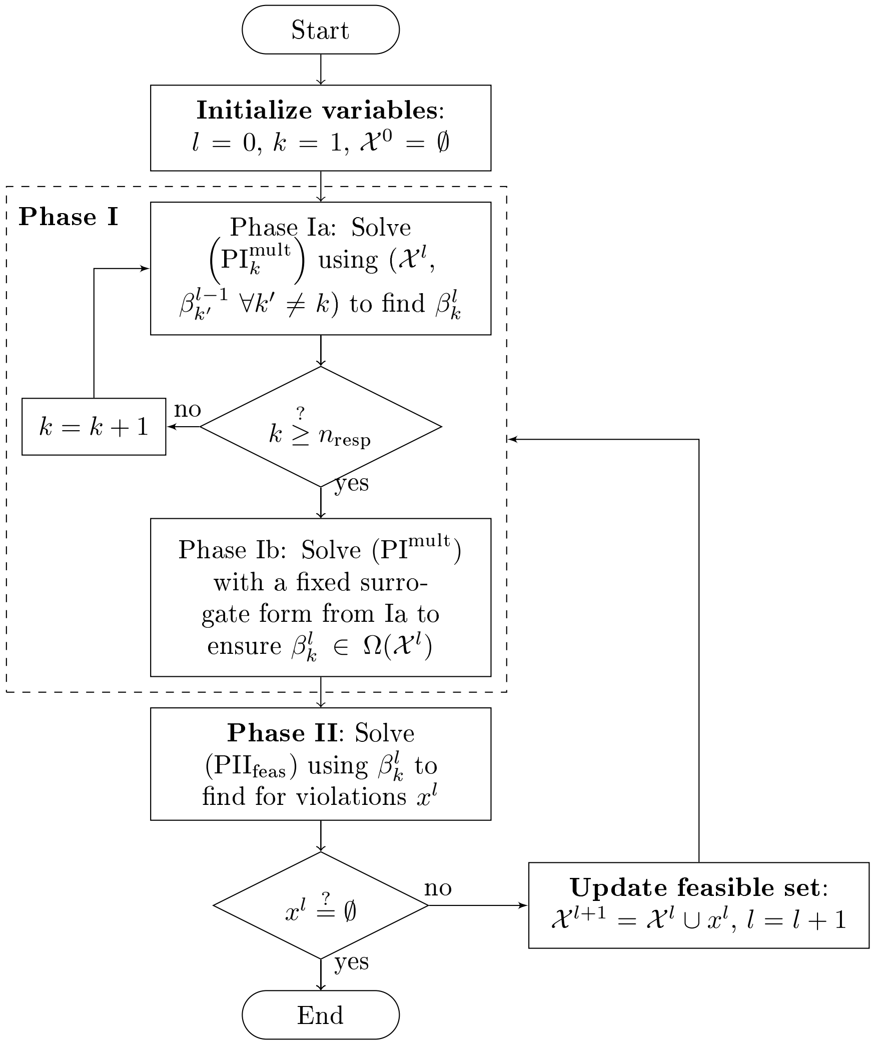
Adaptation for ALAMO

The `ALAMO` package does not require fixed functional forms for response variables. As a result, we can use `ALAMO` to enforce Equation 3.5 with a more flexible functional form. Using the `ALAMO` framework, we solve for each response $z_{k'}$ independently, using the previous iteration's, $l-1$, surrogate models for $z_{k \neq k'}$. The resulting Phase I and Phase II problems are included below:

$$\begin{aligned}
\left(\mathrm{PI}_{k'}^{\mathrm{mult}}\right) \quad & \min_{\beta_{k'} \in \mathcal{A}_{k'}} \quad g(\beta_{k'}) \\
& \text{s.t.} \quad d\left(\hat{z}_{k'}(x;\beta_{k'}), \hat{z}_{k \neq k'}(x;\beta_k^{l-1})\right) + h(x) \leq 0 \quad x \in \mathcal{X}^l
\end{aligned}$$

$$\begin{aligned}
\left(\mathrm{PII}_{\mathrm{feas}}^{\mathrm{mult}}\right) \quad & \max_{x} \quad d\left(\hat{z}_k(x;\beta_k^l)\right) + h(x) \\
& \text{s.t.} \quad x \in \left[x^{\mathrm{lo}}, x^{\mathrm{up}}\right]
\end{aligned}$$

where the form of g_k and $\mathcal{A}_k$ are given by (A) for response variables $k = 1, 2, \ldots, n_{\mathrm{resp}}$. In this case, Phase I requires the solution of $\left(\mathrm{PI}_{k'}^{\mathrm{mult}}\right)$ for each k. After the regression of



Start
Initialize variables: $l = 0$, $k = 1$, $\mathcal{X}^0 = \emptyset$
Phase I
Phase Ia: Solve $\left(\mathrm{PI}_k^{\mathrm{mult}}\right)$ using ($\mathcal{X}^l$, $\beta_{k'}^{l-1}\ \forall k' \neq k$) to find β_k^l
$k \overset{?}{\geq} n_{\mathrm{resp}}$
no
$k = k+1$
yes
Phase Ib: Solve ($\mathrm{PI}^{\mathrm{mult}}$) with a fixed surrogate form from Ia to ensure $\beta_k^l \in \Omega(\mathcal{X}^l)$
Phase II: Solve ($\mathrm{PII}_{\mathrm{feas}}$) using β_k^l to find for violations x^l
$x^l \overset{?}{=} \emptyset$
no
Update feasible set: $\mathcal{X}^{l+1} = \mathcal{X}^l \cup x^l$, $l = l+1$
yes
End

Figure 3.5: Extending restricting multiple responses to `ALAMO`

each individual response variable, it is possible that $\beta_k^l \notin \Omega(\mathcal{X}^l)$ for $k = 1, 2, \ldots, n_{\mathrm{resp}}$ since $\left(\mathrm{PI}_{k'}^{\mathrm{mult}}\right)$ is solved using previous models for responses $k \neq k'$. To ensure the feasibility of the resulting model combination after solving $\left(\mathrm{PI}_{k'}^{\mathrm{mult}}\right)$ for each k, we fix the functional form of each response and solve $\left(\mathrm{PII}_{\mathrm{feas}}^{\mathrm{mult}}\right)$ using linear least squares regression, as for the general case, using the objective from Equation 3.6. This approach is outlined in Figure 3.5. Using this solution approach, we ensure that $\beta_k^l \in \Omega(\mathcal{X}^l)$ at the end of Phase I and supply the resulting feasible solution to Phase II.

3.4.3 Restricting response derivatives

Frequently, it is useful to impose restrictions using pre-existing derivative or partial derivative information during the formulation of an empirical model. Although these constraints require more knowledge of the underlying system, they may result in an advantageous combination of first-principles theory and empirical data. For problems of this type, regression models must be once- or twice-differentiable, depending on the type of enforced constraints. Restrictions on the feasible regions of the first and second derivatives of each response model, $\hat{z}$, with respect to the predictors, x, have the following functional form.

$$\Omega(\mathcal{X}) := \{\beta \in \mathbb{R}^m : a^\intercal \nabla_x \hat{z} + h(x) \leq 0,\ x \in \mathcal{X}\} \tag{3.7}$$

$$\Omega(\mathcal{X}) := \left\{\beta \in \mathbb{R}^m : a^\intercal \nabla_x^2 \hat{z} + h(x) \leq 0,\ x \in \mathcal{X}\right\} \tag{3.8}$$

First derivative restrictions

Derivative constraints may represent the most elegant combination of empirical data, first principles, experience, and intuition. One ubiquitous example is the monotonicity of a response variable with respect to one or more predictors. Cumulative distributions, the entropy of an enclosed system, and gas pressure with respect to temperature under ideal or near ideal conditions are all examples of monotonic relationships. Derivative restrictions may also result from imposing initial or boundary conditions as shown in Section 3.4.4. Below, we include an example demonstrating the imposition of an upper bound on the magnitude of the gradient with the aim of model smoothing and the reduction of over-fitting.

Second derivative restrictions

Enforcing bounds on the curvature of a surrogate model can be somewhat more complicated. For some modeling applications, however, it can be extremely useful. Consider, for example, the preservation of convexity or concavity of a regression model; if data is sampled from an underlying convex distribution, the resulting model is also convex.

Enforcing derivative restrictions is particularly enticing when the resulting metamodels will be used in an optimization framework. To enforce such a condition, the modeler must have *a priori* knowledge of the convexity or concavity of the underlying data set. Additionally, the regression model, $\hat{z}(x)$, must be twice differentiable to enforce convexity. This type of restriction is enforced by requiring the Hessian matrix of partial second derivatives $H(\beta, x)$ to be positive-definite. This requirement is, in turn, enforced by restricting the determinant to be nonnegative:

$$\Omega(\mathcal{X}) := \{\beta \in \mathbb{R}^m : \det(H(\beta, x)) \geq 0,\ x \in \mathcal{X}\} \tag{3.9}$$

Consider, for example, that we desire to ensure the convexity of $\hat{z}(x_1, x_2) = \beta_0 + \beta_1\, x_1 + \beta_2\, x_2 + \beta_3\, x_1\, x_2^2 + \beta_4\, x_1^2\, x_2$ with the following Hessian matrix.

$$H(\beta, x_1, x_2) = \begin{pmatrix} 2\,\beta_4\, x_2 & 2\,\beta_3\, x_2 + 2\,\beta_4\, x_1 \\ 2\,\beta_3\, x_2 + 2\,\beta_4\, x_1 & 2\,\beta_3\, x_1 \end{pmatrix}$$

The enforcement of convexity on the feasible region of the regression problem results from a restriction of the determinant of H to the nonnegative space. This constraint is nonlinear and nonconvex in the β space. Using a least squares regression object, enforcing convexity would transform a quadratic problem into a quadratically constrained quadratic problem:

$$\Omega(\mathcal{X}) := \left\{\beta \in \mathbb{R}^5 : -\beta_3\,\beta_4\,x_1\,x_2 - \beta_4^2\,x_1^2 - \beta_3^2\,x_2^2 \geq 0,\ x \in \mathcal{X}\right\} \tag{3.10}$$

In the example below, we model data sampled from $z = x^2 - 0.4\,x + 0.04 + \epsilon$ over $x \in [-1, 1]$ where ϵ is sampled from a uniform random distribution, $\epsilon \in [-0.25, 0.25]$ using a potential regression model of the form $\hat{z}(x) = \beta_0 + \beta_1\,x + \beta_2\,x^2 + \beta_3\,x^3 + \beta_4\,x^4 + \beta_5\,x^5 + \beta_6\,x^6 + \beta_7\exp(x)$. The underlying distribution is convex; therefore, we desire a convex surrogate model. Beyond the desire to use this information to increase the accuracy of the surrogate model, there are many cases for which the imposition of favorable numerical properties onto a model may be beneficial. In this case, regression models used for optimization benefit from a convex surrogate.

To ensure convexity, we enforce a nonnegativity bound on the second derivative of the surrogate model: $\hat{z}''(x) = 2\beta_2 + 6\beta_3\,x + 12\,\beta_4^2 + 20\,\beta_5\,x^3 + 30\beta_6\,x^4 + \beta_7\exp(x)$. This results in the following Phase I constraints and Phase II subproblem.

Phase I:

$$2\beta_2 + 6\beta_3\,x + 12\,\beta_4\,x^2 + 20\,\beta_5\,x^3 + 30\beta_6\,x^4 + \beta_7\exp(x) \geq 0 \qquad x \in \mathcal{X}^l$$

Phase II:

$$\begin{aligned} \min_{x\in[-1,1]} \quad & f = 2\beta_2^l + 6\beta_3^l\,x + 12\,\beta_4^l\,x2 + 20\,\beta_5^l\,x^3 + 30\beta_6^l\,x^4 + \beta_7^l\exp(x) \\ \text{s.t.} \quad & f \leq -\epsilon \end{aligned}$$

We begin by solving the unconstrained fitting problem. Next, we compare this solution to the constrained case with convexity enforcement. To enhance the solution further, we investigate extending the enforcement domain (discussed in more detail in the next subsection). The training set consists of 25 points randomly sampled over $x \in [-1, 1]$ and the models are validated using a set of 200 randomly sampled points.

The unconstrained solution has a highly nonconvex region near the edges of the x-domain which can be seen in the models and second derivatives for all three cases in Figure 3.6. Models resulting from the unconstrained, convexity enforced, and extended enforcement domain methods are provided in Table 3.5 with corresponding training and test root mean squared errors.

Terms	Unconstrained regression models	Convexity enforced regression models	Extended domain enforcement
1	$\hat{z} = 1.097\,x^2$	no change	no change
2	$-0.363\,x + 1.023\,x^2$	no change	$-0.363\,x + 1.023\,x^2$
3	$-0.356\,x + 1.31\,x^2 - 0.581\,x^6$	$-0.361\,x + 1.15\,x^2 - 0.193\,x^4$	–
Root mean squared errors:			
Training	0.117	0.129	0.137
Test	0.104	0.0791	0.0796

Table 3.5: Models and errors found with unconstrained and enforced convexity

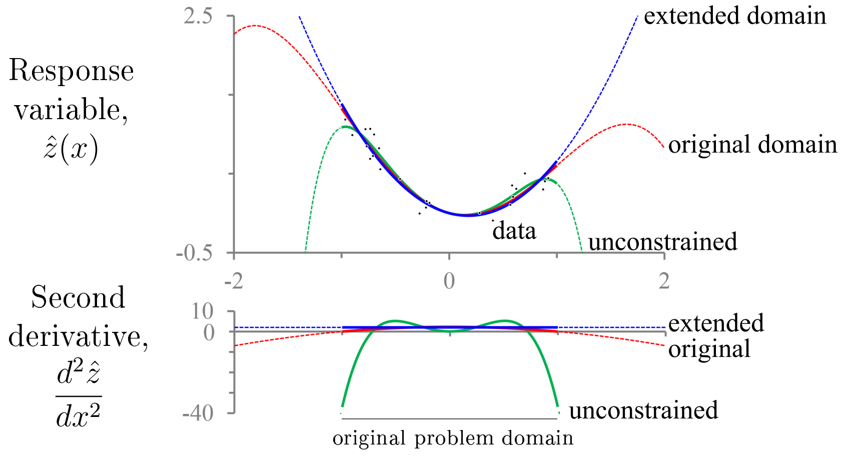


Figure 3.6: Convexity enforcement example

Enforcing convexity over the original sampling domain leads to a reduction in the test error of 24%. This occurs despite an increase in the training error of 10%. In both cases, a three-term model is selected. The solution is further improved by extending the constraint enforcement to $x \in [-10, 10]$. This results in a functional form that more closely matches the true function and a model simplification, or complexity reduction, without significant test error losses. The next subsection offers further details concerning accuracy improvement enabled by enforcing constraints beyond the original sampling domain.

3.4.4 Enforcement domain expansion and contraction

In the previous subsections, we primarily restrict the response variables over the range of the original predictor variable; yet, as previously mentioned, it is possible to extend this range to include an *expected extrapolation range* or to contract this domain to enforce conditions over a subset of $\mathcal{X}$ (e.g., boundary or initial values, a smaller dimensional space, etc.).

Safe extrapolation

By expanding the enforcement domain, we aim to provide a *safe extrapolation.* While, in general, robust prediction techniques avoid extrapolation, engineers and scientists regularly use extrapolation to forecast results beyond an initial sample space [77]. Extrapolation, or using the regression model to predict beyond the range of the original data, increases the likelihood model error is introduced [77]. For these reasons, we propose the use of constrained regression over an expanded set $x \in \mathcal{X}_{\text{extrap}}$ with the aim of performing safe extrapolation. Safe extrapolation can be used in conjunction with any of the problem classes described in Section 3.4 to improve extrapolation accuracy and, often, improve accuracy within the original problem domain.

For further illustration, we revisit the example from Section 3.4.1 involving reactions in series in a batch reactor. Using the same data set and potential set of basis functions, we model the concentration of B over $x \in [0.6, 10]$ while enforcing bounds on $[B] \in [0, 1]$ over the expected extrapolation range of $x \in [0.1, 11]$.

Test errors for the unconstrained, constrained, and extended domain enforcement are provided in Table 3.7. By extending the enforcement domain, we are able to reduce the error by a factor of 55.2 over the unconstrained model and 7.61 over the original constrained models while maintaining a five-term solution. The models and corresponding test errors with model complexity from one to five terms are included in Table 3.6.

Table 3.6: Successive models for $[\hat{B}](t)$ with bounds enforced over $x \in [0.1, 11]$

Terms	Test error	Model, $[\hat{B}](t)$
*1	–	$0.261/t$
1	0.0378	$0.156/\sqrt{t}$
*2	–	$1.39/t^2 - 1.38/t^3$
*2	–	$-0.102 \log t + 0.219$
2	0.0282	$0.350/\sqrt{t} - 0.105$
3	0.0100	$-0.0609\, t + 0.00363\, t^2 + 0.263$
4	0.0044	$0.456\sqrt{t} - 0.281\, t + 0.0208\, t^2 - 0.000705\, t^3$
*5	–	$-1.29 \log t - 1.03/t + 0.0799/t^2 + 1.20\sqrt{t} - 0.0719\, t$
5:final	0.00155	$3.30 \cdot 10^{-6} \exp t + 0.458\sqrt{t} - 0.280\, t + 0.0180\, t^2 - 5.16 \cdot 10^{-5}\, t^4$

*Unsuccessful trial model

Table 3.7: Test error over the original and extended domains for each regression method

	Test error	
	Original domain $t \in 0.6, 10$	Extended domain $t \in 0.1, 0.6$ and $t \in 10, 11$
Unconstrained regression model	0.0856	0.951
Original domain constrained	0.0118	0.0788
Extended domain constrained	0.00155	0.00717

Figure 3.7 shows the true concentration curve, data, and the three models over both the original problem domain and extended enforcement domain. By extending the enforcement domain, we are able to better predict the model over the extended domain and the original problem domain. The extrapolation errors calculated from 500 additional validation points sampled over for each expansion domain: $t \in [0.1, 0.6]$ and $t \in [10, 11]$. These errors and the test error over the original problem domain are provided in Table 3.7.

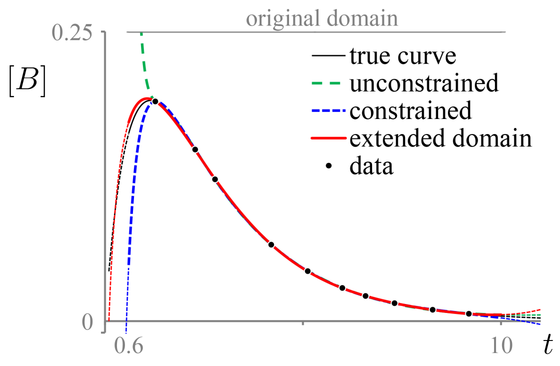


Figure 3.7: Model results for the concentration of B

As we increase the restriction on the regression on this problem, inverse bases functions become inactive in exchange for terms that are more favorable at small values of t. This is evident in both the original constrained models and the extended domain constrained models. Constraining the model in the x- and z-space forces the regression to select terms and parameter values (i.e., restrict β) without *a priori* knowledge of relationships among terms and the β space.

Boundary and Initial conditions

Enforcing boundary conditions while modeling empirical data often leads to a more physically consistent regression model. Boundary conditions are imposed on ordinary or partial differential equations and are categorized into three types: Dirichlet, Neumann, and Robin. Often, these conditions are enforced in a reduced dimensional space. For example, initial conditions provide specifications on the time domain, $t = 0$, without restricting the space domain. For these problems, we reduce the enforcement domain by one or more dimensions x_i to $\mathcal{X} \cap \{x_i = x_i^*\}$.

Dirichlet boundary conditions specify the value of the solution, z, at a fixed location, x_i^*, of the x-domain. Neumann boundary conditions specify the value of the gradient at a fixed location in at least one dimension, x_i^*. Finally, Robin boundary conditions specify a linear combination of function values and derivatives at a fixed location in the domain. The imposition of Dirichlet boundary conditions results in a feasible region that is similar to that provided by Equation 3.4, while Neumann and Robin conditions result in the form described by Equations 3.7 and 3.8.

Standard boundary conditions require semi-infinite equality constraints and enforcing these constraints may impede the convergence of the optimization solver. Instead, we permit an ϵ-tolerance on the slack of the equality as follows for $a, b \in \mathbb{R}^n$.

$$\Omega(\mathcal{X}) :=$$
$$\left\{ \beta \in \mathbb{R}^m : \begin{array}{ll} \hat{z}(x;\beta) + a^\intercal \nabla_x \hat{z} + b^\intercal \nabla_x^2 \hat{z} - h(x) \leq \epsilon_{\text{viol}} & \\ & x \in \mathcal{X} \cap \{x_i = x_i^*\} \\ \hat{z}(x;\beta) + a^\intercal \nabla_x \hat{z} + b^\intercal \nabla_x^2 \hat{z} - h(x) \geq -\epsilon_{\text{viol}} & \end{array} \right\}$$

These constraints allow a modeler to use both high fidelity, simulation data and first principles boundary conditions.

3.5 Computational experiments

In this section, we demonstrate efficacy of the proposed methods through a computational study and compare the accuracy of constraining the x- and z-space in contrast to unconstrained regression. For each instance, we use a fixed data set from which we generate a regression model three ways:

1. Unconstrained: Subset regression using `ALAMO`
2. Constrained: Constrained subset regression over the original problem domain using `ALAMO`
3. Extended domain constrained: Constrained subset regression over an extended or expected extrapolation problem domain using `ALAMO`.

The test set includes three types of underlying functions: chemical reactor models, normal distributions, and logistic problems that have upper and/or lower bounds. Details of each category are provided in Table 3.8. In total, the test set contains 120 problems. Ten functional forms from each category are generated using random parameter values sampled from a uniform distribution over the variable ranges specified in Table 3.8. Using a Latin hypercube design of experiments, four distinct data sets per functional form are generated and tested. The potential functional forms for each regression model include exponentials, logarithms, and several powers: ±0.5, ±1, ±2, ±3, ±4.

Table 3.8: Description of problem types in the computational test set

Problem type	Description	Data set sizes	Random parameters
Batch reactor	Batch reactor concentration of B, $A \xrightarrow{k_a} B \xrightarrow{k_b} C$, $[A]_0 = 1$, $[B]_0 = 0$, and $[C]_0 = 0$.	3, 10, 10, 10	$k_a \in [0.1, 2]$, $k_b \in [0.1, 2]$
Normal distribution	Normal distribution with mean μ and standard deviation σ	3, 10, 10, 10	$\mu \in [1, 10]$, $\sigma \in [1, 3]$
Logisitic curve	Logistic function with multiplier a and offset b, $1/\left(1 + \exp(-a(x-b))\right)$	3, 10, 10, 10	$a \in [0, 5]$, $b \in [3, 7]$

Specifications for Methods 2 and 3 are summarized in Table 3.9. The logistic and batch reactor test instances are bounded from above and below while the normal distribution is bounded from below.

Table 3.9: Constrained regression specifications

Problem type	Bounds enforced	Original problem domain	Extended problem domain
Batch reactor	$0 \leq [B] \leq [A]_0$	$t \in [0.6, 10]$	$t \in [0.5, 11]$
Normal distribution	$0 \leq z$	$x \in [1, 10]$	$x \in [0.5, 11]$
Logisitic curve	$0 \leq z \leq 1$	$x \in [1, 10]$	$x \in [0.5, 11]$

To compare the three methods, we use the error factor, defined by Equation 3.3, as a model quality metric. The cumulative distribution of error factors for the three methods is plotted in Figure 3.8. Here, four factors are considered:

1. Training error: error between the model and the points used to train the model

2. Original domain test error: error between the model and 1000 randomly sampled points over the original problem domain

3. Lower extended domain test error: error between the model and 500 randomly sampled points over the domain extended to smaller values of the predictor variables

4. Upper extended domain test error: error between the model and 500 randomly sampled points over the domain extended to larger values of the predictor variables.

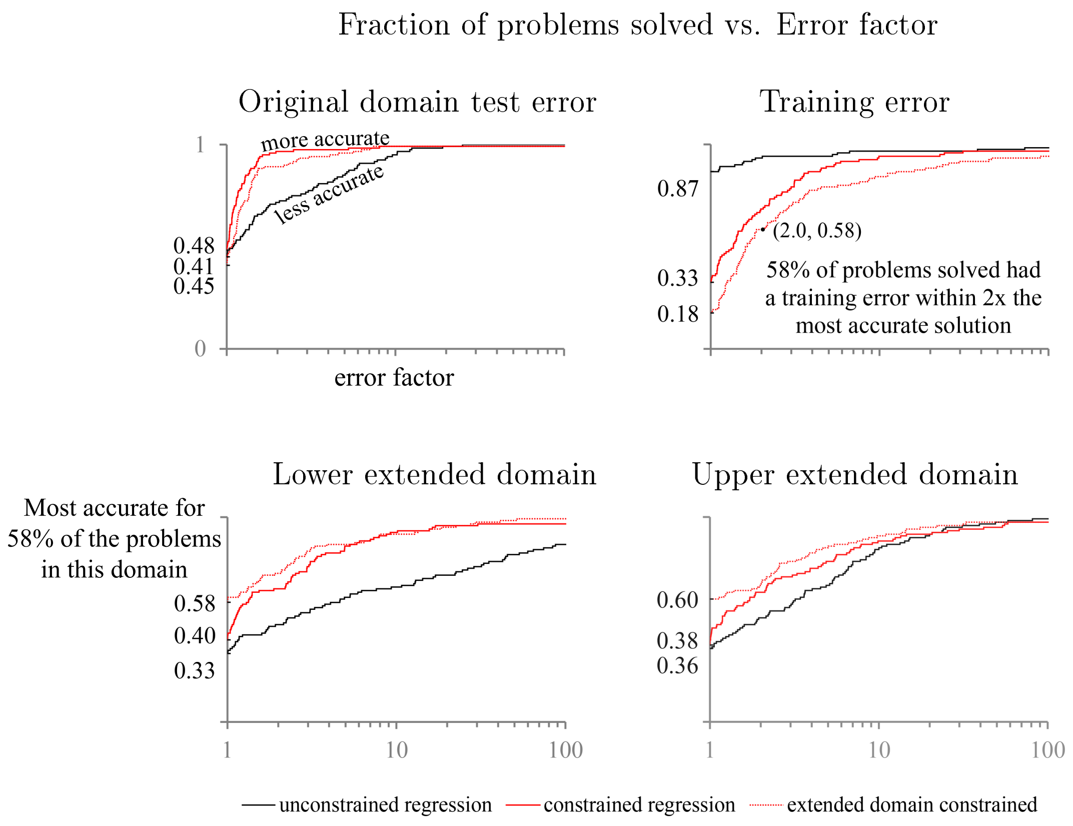


Figure 3.8: Computational results

The plots in Figure 3.8 depict a race-to-the-top metric. Therefore, curves that climb higher for smaller values of the error factor signify more accurate regression modeling. Since the primary purpose of these regression models is an accurate representation of true functional form, we first consider the test accuracy over the original sample space. In 48% of the problems, constrained regression is either the most accurate or equal to the most accurate over this domain and continues to be particularly dominant over the unconstrained regression models. Yet, the unconstrained regression models have significantly smaller training error, with errors at least as low as the smallest training error in 87% of the problems. In the final 13% of the test set, constrained regression generates more complex models than unconstrained regression. This superior prediction capability further motivates the addition of response variable bounds to improve model accuracy over a simple empirical data modeling technique.

By extending the bound enforcement domain as described in Table 3.9, the test error over both the lower and upper extrapolation range is improved to allow for safer model extrapolation. The extended domain constrained models demonstrate the highest accuracy for 58 and 60% of the problems for the lower and upper bounds, respectively. These extended domain methods are dominant over unconstrained models and are superior to constraining only the original problem space. In particular, there is benefit in the lower extrapolation of these extended constraints over the unconstrained case.

In order to achieve this increased model accuracy, constrained regression does require additional computation effort. Up to five points are added to each discretized Phase II problem. In most cases, if the trial model violates a bound, all five points are returned. However, some problems resulted in fewer than five violated points. The constrained regression statistics (feasible set size, total constrained regression iterations, and regression resolves required) for this problem set are provided in Table 3.10. Constraining the regression problem over the expanded domain requires more regression re-solves and iterations than constraining the original problem domain.

Table 3.10: Constrained regression solution statistics

	Points in the final feasible set		Constrained regression iteration		Additional regression problems solved	
	Mean	Range	Mean	Range	Mean	Range
Original domain	7.52	[0, 25]	1.93	[1, 3]	1.58	[0, 7]
Extended domain	10.7	[0, 35]	2.28	[1, 6]	2.50	[0, 18]

For the problems shown, constraints enforcing bounds on the response variables not only generate more physically realizable models, but use this information to build more accurate models and to enhance extrapolation accuracy.

3.6 Conclusion

The combination of data-driven modeling and theory-driven *a priori* knowledge results in higher quality of surrogate models and the effects are measured by both physical relevance and model accuracy. We introduce a novel approach to constrained regression: restricting the original problem in the space of predictor and response variables. Constraints of this *more intuitive* form can be used to reveal hidden relationships between regression coefficients that are otherwise unknown to the modeler. In particular, we describe several classes of problems that lead to intuitive constraints including zero-order bounds of response variables, thermodynamic limitations, safe extrapolation, boundary conditions, and enforcing favorable numerical properties on first and second derivatives. Through extensive testing, we demonstrate the capability of these restrictions to improve model accuracy in addition to ensuring adherence to physical limits.

We conclude that *a priori* information from first principles, intuition, and other system knowledge can be used to enhance data-driven modeling with many model selection and regression methods, from ordinary least squares regression to subset selection and regularized regression.

Chapter 4

A global optimization approach to symbolic regression

Symbolic regression methods generate expression trees that simultaneously define the functional form of a surrogate model in addition to regression parameter values. As a result, the regression problem can search many nonlinear functional forms using only the specification of simple mathematical operators such as addition, subtraction, multiplication, and division, among others. Currently, state-of-the-art symbolic regression methods leverage genetic algorithms and adaptive programming techniques. Genetic algorithms lack optimality certifications and are typically stochastic in nature. In contrast, we propose an optimization formulation for the rigorous deterministic optimization of the symbolic regression problem.

We present a disjunctive optimization formulation used to solve the symbolic regression problem as well as several performance-enhancing improvements. We demonstrate this symbolic regression technique using an illustrative example and compare our formulation and improvements in an array of experiments based upon two literature instances.

4.1 Introduction

Traditional regression and model selection methods seek to develop regression models with a pre-determined model structure or a set of alternative model structures. In contrast, symbolic regression aims to learn the functional forms and corresponding model parameters simultaneously [63, 64]. Traditional regression techniques limit the scope of model development to a fixed functional form, e.g. linear, quadratic, exponential, logarithmic, or some combination of pre-determined functions. Symbolic regression requires only the specification a set of operators and operands ($+$, $-$, $*$, $\div$, $\exp(\cdot)$, $\log(\cdot)$, $(\cdot)^2$, $(\cdot)^3$, $\sqrt{\cdot}$, x, constant, etc.). Increasing the flexibility of the regression data structure allows for the modeling of black boxes with unknown functional forms and results in a model that not only describes the data accurately, but can provide insights into the underlying processes. These methods have shown success over a wide range of applications: chemical systems [11, 72, 73], fluid dynamics [106], control engineering and system dynamics [9, 52], pattern classification [54], finance [20], and natural sciences [89].

Symbolic regression, also known as function identification and empirical design, begins with an idea that any expression can be written as an *expression tree* [63]. An expression tree recursively defines the order of operations in a function using operands at leaf nodes and operators for all other nodes. Expression trees need not be binary; however, we limit the scope of this work to binary expression trees.

To evaluate an expression tree, each operator is applied recursively starting at the root node. This tree structure encodes the order of operations, or parenthetical structure of the model's functional form, using the depth of each node. Figure 4.1 shows the expression tree for $\frac{2\,x_1}{5 - x_2}$. Often, small changes in the expression tree result in large changes in model structure. By changing only the root node operator, the resulting expression tree now represents $2\,x_1 + x_2 - 5$.

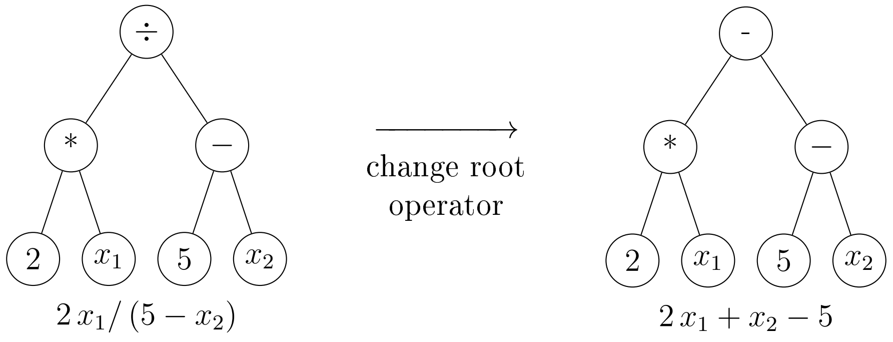


Figure 4.1: Expression tree examples

To generate an accurate regression model, symbolic regression minimizes an error metric, such as the sum of the squared error between response data and the root node value of the expression tree [108]. The structure, size, and activity or value of each node is chosen to minimize this overall error metric. The goal of this process is to optimize both the structure of the tree and the activity or value of each node simultaneously.

Symbolic regression is currently approached as an application of Genetic Programming (GP), the systematic method for training a computer to solve high-level problems by combining simpler functions and subroutines [64]. Genetic programming describes how computers learn specific tasks without a programmer's explicit instructions [63]. GP methods adjust and improve the expression or syntax tree using an iterative application of techniques that are common to evolutionary optimization algorithms.

A GP application begins with an initial population of individuals or randomly selected expression trees. These trees are compared by a fitness measure, the error metric. Individuals with high scores have a higher probability of being selected for the next iteration of crossover, mutation, and reproduction. The crossover step generally includes exchanging the subtrees of two individuals. Mutations for individual expression trees can include replacing a subtree with a random subtree, shrinkage mutations, or the random promotion of a subtree to a higher level. Both crossover and mutation are used for reproduction to define a new generation of offspring or population of expression trees [63].

The fundamental concepts behind symbolic regression are not restricted to GP. Almost exclusively, however, symbolic regression is performed using evolutionary algorithms—so much so that *symbolic regression* is often used interchangeably with *genetic programming*. Often, genetic algorithms can find very good solutions; however, there is no guarantee of either local or global optimality [23]. In fact, a publication by Korns states "in current state-of-the-art symbolic regression engines, accuracy is poor" [62] to describe the mismatch between the true and surrogate functional form found using published state-of-the-art symbolic regression techniques. Korns states that this is even a concern for one-basis models with minimal tree depth. To our knowledge, there has been no investigation that applies rigorous, global optimization methods to the symbolic regression problem.

In this chapter, we show that the symbolic regression problem can be formulated as a nonlinear nonconvex disjunctive program and can be solved to global optimality. Using this model, we describe techniques that improve surrogate model quality as well as the solution speed of the optimization model. The efficacy of these methods is demonstrated using standard literature test cases.

4.2 Proposed model

In this section, we describe the application of a nonconvex mixed-integer nonlinear disjunctive programming formulation to solve the symbolic regression problem using rigorous, global optimization.

4.2.1 Formulation

Figure 4.2 shows an expression tree with a depth of D and $N = 2^D - 1$ nodes. Each node n contains either an operand (x_d, constant, etc.) or an operator ($+$, $-$, $*$, $\div$, $\exp(\cdot)$, $\log(\cdot)$, $(\cdot)^2$, $(\cdot)^3$, $\sqrt{\cdot}$, x, constant, etc.) that acts on the left and/or right child, l_n and r_n, respectively.

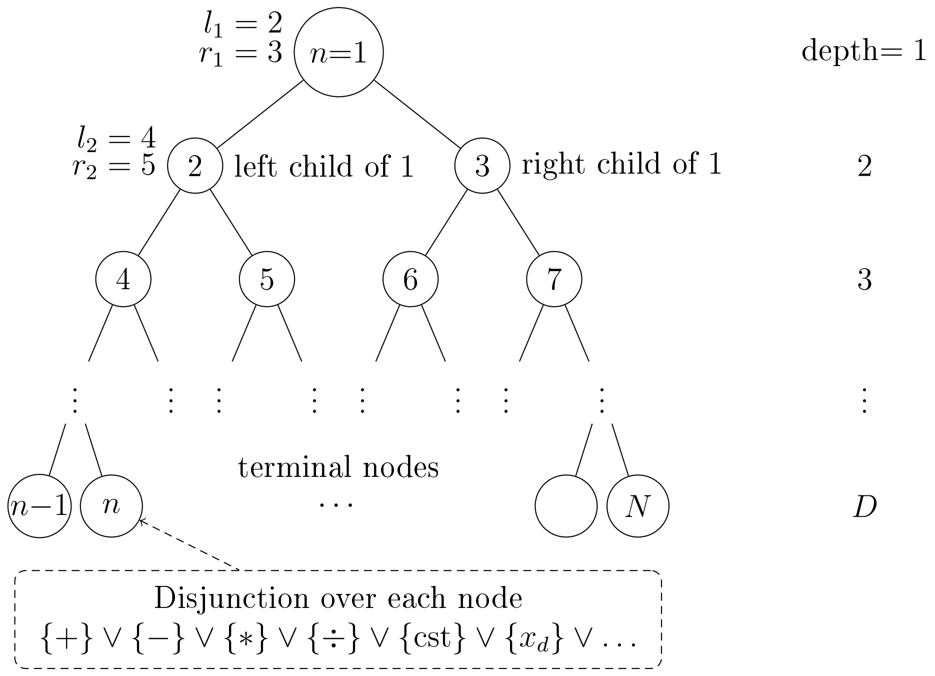


Figure 4.2: Binary tree with indices structure

Each node has a value v_{in} for all data points $i = 1, 2, \dots, n_{\text{data}}$, that is computed through the recursive application of the operators and operands of the children of node n. For each node, we apply a single disjuction that determines which operator or operand is active. A list and description of commonly used operators and operands is given in Table 4.1 along with other notation used in this model.

Using this tree and index assignment given in Figure 4.2, we formulate the following disjunctive optimization problem.

Table 4.1: Notation

Variable	Description	Defined for elements	Allowed values
Parameters			
z_i	Response z at data point i	$i = 1, 2, \ldots, n_{\text{data}}$	given
x_{id}	Predictor x_d at data point i	$i = 1, 2, \ldots, n_{\text{data}}$, $d = 1, 2, \ldots, n_{\text{pred}}$	given
Variables			
v_{in}	Value at node n and data point i	$i = 1, 2, \ldots, n_{\text{data}}$, $n = 1, 2, \ldots, N$	$\left[v_{in}^{\text{lo}}, v_{in}^{\text{up}}\right]$
y_n^o	Binary activity variable of operator o at node n	$o \in \mathcal{O}$, $n = 1, 2, \ldots, N$	$\{0, 1\}$
β_n	Model parameter for node n	$n = 1, 2, \ldots, N$	$\left[\beta_n^{\text{lo}}, \beta_n^{\text{up}}\right]$
Sets			
$\mathcal{O}$	Superset of simple operators	–	$\mathcal{O} \in \mathcal{B} \cup \mathcal{U} \cup \mathcal{L}$
$\mathcal{B}$	Binary operators	–	$\{+, -, *, \div, \ldots\}$
$\mathcal{U}$	Unary operators	–	$\{\exp, \log, \text{sqr}, \ldots\}$
$\mathcal{L}$	Leaf node types	–	$\bigcup_{d=1}^{D} \{x_d\} \cup \{\text{cst}\}$
$\mathcal{V}$	Variable operands	–	$\bigcup_{d=1}^{D} \{x_d\}$
TERM	Terminal nodes	–	$\{n; d = D\}$

$$\min \quad \sum_{i=1}^{n_{\text{data}}} (z_i - v_{i1})^2 \tag{4.1a}$$

$$\text{s.t.} \quad \bigvee_{o\in\mathcal{O}} \begin{bmatrix} y_n^o = 1 \\ \\ f_i^o\left(v_{in}, v_{il_n}, v_{ir_n}, x_{id}\right) = 0 \\ g_i^o\left(v_{in}, v_{il_n}, v_{ir_n}, x_{id}\right) \le 0 \\ i =, 1, 2, \ldots, n_{\text{data}} \end{bmatrix} \bigvee \begin{bmatrix} \sum_{o\in\mathcal{O}} y_n^o = 0 \\ v_{in} = 0 \\ i =, 1, 2, \ldots, n_{\text{data}} \end{bmatrix} \qquad n \notin \text{TERM} \tag{4.1b}$$

$$\bigvee_{o\in\mathcal{L}} \begin{bmatrix} y_n^o = 1 \\ \\ f_i^o\left(v_{in}, x_{id}\right) = 0 \\ g_i^o\left(v_{in}, x_{id}\right) \le 0 \\ i =, 1, 2, \ldots, n_{\text{data}} \end{bmatrix} \bigvee \begin{bmatrix} \sum_{o\in\mathcal{L}} y_n^o = 0 \\ v_{in} = 0 \\ i =, 1, 2, \ldots, n_{\text{data}} \end{bmatrix} \qquad n \in \text{TERM} \tag{4.1c}$$

$$\sum_{o\in\mathcal{O}} y_n^o \le 1 \qquad n = 1, 2, \ldots, N \tag{4.1d}$$

$$\sum_{o\in\mathcal{B}\cup\mathcal{U}} y_n^o \le y_{l_n}^o \qquad n = 1, 2, \ldots, N \tag{4.1e}$$

$$\sum_{o\in\mathcal{B}} y_n^o \le y_{r_n}^o \qquad n = 1, 2, \ldots, N \tag{4.1f}$$

$$\sum_{o\in\mathcal{U}\cup\mathcal{L}} y_n^o \le 1 - y_{r_n}^o \qquad n = 1, 2, \ldots, N \tag{4.1g}$$

$$\sum_{o\in\mathcal{L}} y_n^o \le 1 - y_{l_n}^o \qquad n = 1, 2, \ldots, N \tag{4.1h}$$

$$\sum_{o\in\mathcal{V}} y_n^o \ge 1 \tag{4.1i}$$

$$y_n^o \in \{0, 1\} \qquad n \notin \text{TERM} \qquad o \in \mathcal{O} \tag{4.1j}$$

$$y_n^o \in \{0, 1\} \qquad n \in \text{TERM} \qquad o \in \mathcal{L} \tag{4.1k}$$

$$v_{in} \in \left[v_{in}^{\text{lo}}, v_{in}^{\text{up}}\right] \qquad n = 1, 2, \ldots, N \qquad i = 1, 2, \ldots, n_{\text{data}} \tag{4.1l}$$

Here, we show a squared error minimization objective; however, this formulation can be easily expanded to other loss functions such as linear error, Mallow's C_p, or information criterion. We seek to minimize the squared error between response variable z_i based on the modeled value at the root node, v_{i1}, over the training data set $i = 1, 2, \ldots, n_{\text{data}}$. The model is a function of predictor variables x_d in n_{pred} dimensions. The disjunctions are modeled using binary variables y_n^o, where an operator or operand o is active at node n if $y_n^o = 1$ and inactive otherwise.

If active, each node can take on a value in one of three categories: binary operator, unary operator, or leaf node operand. This classification provides the basis for logical constraints. Each binary operator is applied using both children nodes while a unary operator uses only the value of the left node. The leaf node operands are specified as constants, cst, or set by the value of a predictor variable x. The specific application, f^o, of the operators and operands is given in Table 4.2. Additional requirements for an operator or operand are given by g^o. For division, $y_n^{\div} = 1$, $g^{\div}$ bounds the denominator, or right child, away from zero with a small number ϵ to avoid dividing by zero. The numerator is also bounded away from zero to avoid arbitrary solutions. We also require the left child to be strictly positive for both logarithms and square roots.

The disjunction for each node over $o \in \mathcal{O}$ defines which operator or operand is active at node n, the operation f_i^o that is applied, and the corresponding special restrictions g_i^o applied for each data point i. The first constraint, Equation 4.1b describes the disjunction over all nonterminal nodes $n \notin$ TERM. All operators and operands are available as non-terminal nodes. Terminal nodes $n \in$ TERM, however, are restricted to leaf node operands. Equation 4.1c shows the disjunction for terminal nodes. If no operators or operands are active, the value v_{in} is set to zero for each node. Equation 4.1d ensures that up to one operator or operand is active at each node. Logic constraints 4.1e–4.1h are used to ensure the correct activity of children nodes based on the operator and operand category. If any binary operator o is active at node n, Equation 4.1e and 4.1f ensure that both children nodes are active. Equations 4.1e and 4.1g certify the left child is active and the right child is inactive if any unary operator is activate at node n. Finally, if a leaf node operand is

Table 4.2: List of simple operators and operands

Description	Binary	$f_i^o = 0$	$g_i^o \leq 0$
Binary operations			
Addition	y_n^+	$v_{il_n} + v_{ir_n} - v_{in}$	
Subtraction	y_n^-	$v_{il_n} - v_{ir_n} - v_{in}$	
Multiplication	y_n^*	$v_{il_n} v_{ir_n} - v_{in}$	
Division	$y_n^{\div}$	$v_{il_n} - v_{in} v_{ir_n}$	$\epsilon - (v_{ir_n})^2$
			$\epsilon - (v_{il_n})^2$
Power	$y_n^{\wedge}$	$v_{il_n}{}^{\wedge} v_{ir_n} - v_{in}$	$\epsilon - v_{ir_n}$
Unary operations			
Exponential	$y_n^{\exp}$	$\exp(v_{il_n}) - v_{in}$	
Logarithm	$y_n^{\log}$	$v_{il_n} - \exp(v_{in})$	$\epsilon - v_{il_n}$
Square	y_n^{sqr}	$(v_{il_n})^2 - v_{in}$	
Cube	y_n^{cube}	$(v_{il_n})^3 - v_{in}$	
Square root	y_n^{sqrt}	$v_{il_n} - (v_{in})^2$	$\epsilon - v_{il_n}$
Leaf node operands			
Variable	$y_n^{x_d}$	$x_{id} - v_{in}$	
Constant	y_n^{cst}	$v_{i'n} - v_{in}, \forall i \neq i'$	

active, then Equations 4.1g and 4.1h ensure the inactivity of both children nodes. The final constraint, Equations 4.1i, ensures that the model is a function of at least one variable.

We model this nonconvex nonlinear disjunctive program (Equations 4.1a– 4.1l) using a big-M mixed-integer nonlinear programming (MINLP) formulation [41] as follows:

$$
\begin{array}{lll}
\min & \displaystyle\sum_{i=1}^{n_{\text{data}}} (z_i - v_{i1})^2 & \\
\text{s.t.} & f_i^o(v_{in}, v_{il_n}, v_{ir_n}, x_{id}) \le \overline{M}_{in}^o (1 - y_n^o) & n \notin \text{TERM},\ o \in \mathcal{O},\ i =, 1, 2, \ldots, n_{\text{data}} \\
& f_i^o(v_{in}, v_{il_n}, v_{ir_n}, x_{id}) \ge \underline{M}_{in}^o (1 - y_n^o) & n \notin \text{TERM},\ o \in \mathcal{O},\ i =, 1, 2, \ldots, n_{\text{data}} \\
& g_i^o(v_{in}, v_{il_n}, v_{ir_n}, x_{id}) \le G_{in}^o (1 - y_n^o) & n \notin \text{TERM},\ o \in \mathcal{O},\ i =, 1, 2, \ldots, n_{\text{data}} \\
& v_{in} \le v_{in}^{\text{up}} \displaystyle\sum_{o \in \mathcal{O}} y_n^o & n \notin \text{TERM},\ o \in \mathcal{O},\ i =, 1, 2, \ldots, n_{\text{data}} \\
& v_{in} \ge v_{in}^{\text{lo}} \displaystyle\sum_{o \in \mathcal{O}} y_n^o & n \notin \text{TERM},\ o \in \mathcal{O},\ i =, 1, 2, \ldots, n_{\text{data}} \\
& f_i^o(v_{in}, v_{il_n}, v_{ir_n}, x_{id}) \le \overline{M}_{in}^o (1 - y_n^o) & n \in \text{TERM},\ o \in \mathcal{L},\ i =, 1, 2, \ldots, n_{\text{data}} \\
& f_i^o(v_{in}, v_{il_n}, v_{ir_n}, x_{id}) \ge \underline{M}_{in}^o (1 - y_n^o) & n \in \text{TERM},\ o \in \mathcal{L},\ i =, 1, 2, \ldots, n_{\text{data}} \\
& g_i^o(v_{in}, v_{il_n}, v_{ir_n}, x_{id}) \le G_{in}^o (1 - y_n^o) & n \in \text{TERM},\ o \in \mathcal{L},\ i =, 1, 2, \ldots, n_{\text{data}} \\
& v_{in} \le v_{in}^{\text{up}} \displaystyle\sum_{o \in \mathcal{L}} y_n^o & n \in \text{TERM},\ o \in \mathcal{L},\ i =, 1, 2, \ldots, n_{\text{data}} \\
& v_{in} \ge v_{in}^{\text{lo}} \displaystyle\sum_{o \in \mathcal{L}} y_n^o & n \in \text{TERM},\ o \in \mathcal{L},\ i =, 1, 2, \ldots, n_{\text{data}} \\
& \displaystyle\sum_{o \in \mathcal{O}} y_n^o \le 1 & n = 1, 2, \ldots, N \\
& \displaystyle\sum_{o \in \mathcal{B} \cup \mathcal{U}} y_n^o \le y_{l_n}^o & n = 1, 2, \ldots, N \\
& \displaystyle\sum_{o \in \mathcal{B}} y_n^o \le y_{r_n}^o & n = 1, 2, \ldots, N \\
& \displaystyle\sum_{o \in \mathcal{U} \cup \mathcal{L}} y_n^o \le 1 - y_{r_n}^o & n = 1, 2, \ldots, N \\
& \displaystyle\sum_{o \in \mathcal{L}} y_n^o \le 1 - y_{l_n}^o & n = 1, 2, \ldots, N \\
& \displaystyle\sum_{o \in \mathcal{V}} y_n^o \ge 1 & \\
& y_n^o \in \{0, 1\} & n \notin \text{TERM},\ o \in \mathcal{O} \\
& y_n^o \in \{0, 1\} & n \in \text{TERM},\ o \in \mathcal{L} \\
& v_{in} \in \left[v_{in}^{\text{lo}}, v_{in}^{\text{up}}\right] & n = 1, 2, \ldots, N,\ i = 1, 2, \ldots, n_{\text{data}}
\end{array}
$$

In this formulation, we choose big-M values using interval arithmetic to find the upper $\overline{M}$ and lower bounds $\underline{M}$ for each constraint f_i^o at each node given limits on the values of each nodes v_{in}^{lo} and v_{in}^{up}. We define the upper bound G for g_i^o using a similar procedure. The following section includes an explanatory example using this formulation to solve a two-dimensional symbolic regression problem.

4.2.2 Illustrative example

To illustrate the MINLP formulation and its relationship to the structure of the expression tree, we detail a two-dimensional example problem. For conciseness, we will limit the problem to addition, subtraction, multiplication, and division with a maximum tree depth of three and $N = 7$. This results in $\mathcal{B} = \{+,\ -,\ *,\ \div\}$, $\mathcal{L} = \{x_1,\ x_2,\ \text{cst}\}$, and $\text{TERM} = \{4,\ 5,\ 6,\ 7\}$. The training set consists of ten data points on the curve $z = \dfrac{2\,x_1}{5 - x_2}$ over $0 \leq x_1, x_2 \leq 1$ sampled at random values for x_1 and x_2. The resulting symbolic regression model, including big-M constraints, has 70 continuous variables, 33 binary variables, and 1,378 constraints. The model is solved using the global optimization solver SCIP [102] with bounds for v_{in} for nodes $n > 1$ set to $[-100, 100]$. Root node bounds are determined assuming a model that remains within two standard deviations of the true value: $[v_{i1}^{\text{lo}}, v_{i,1}^{\text{up}}] = [z_i - 2\,\sigma, z_i + 2\,\sigma]$. The optimal expression tree is shown in Figure 4.3 and an instance i' of this tree at data point $(x_{i'1}, x_{i'2}) = (0.36, 0.60)$ is shown in i' Figure 4.4.

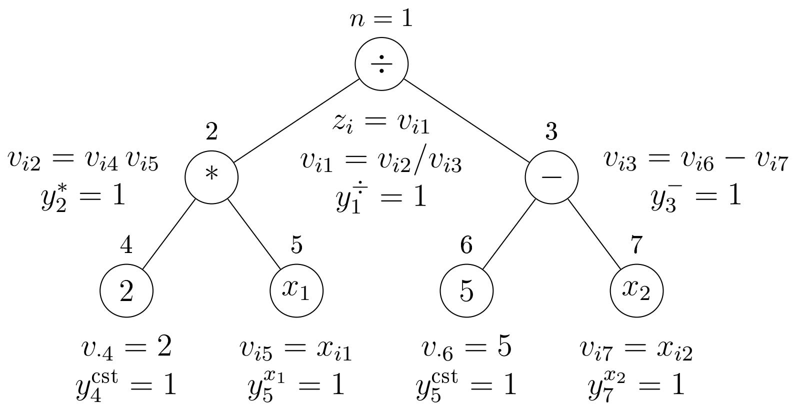


Figure 4.3: Expression tree for $\dfrac{2\,x_1}{5 - x_2}$ with relevant decision variable values

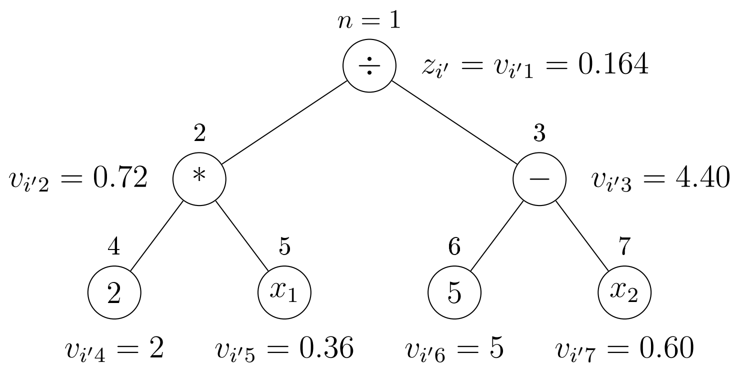


Figure 4.4: Expression tree for $\dfrac{2\,x_1}{5 - x_2}$ for $i = i'$ at $(x_{i'1}, x_{i'2}) = (0.36, 0.60)$

Typical symbolic regression methods utilize a pareto analysis [93] to analyze the trade-off between model complexity and model accuracy. Here we accomplish this by imposing an additional constraint on the total number of active nodes $\sum_{n=1}^{N} \sum_{o \in \mathcal{O}} y_n^o \leq T$, where T is increased over the range 1 to N. However, the same curve can be generated by minimizing the number of active nodes given an upper limit on the arror objective that can be successively increased from zero to a large value. In fact, some problems may solve more quickly using this objective rather than the error objective. The expression trees that result from parameterizing the solution with respect to T are shown in Figure 4.5.

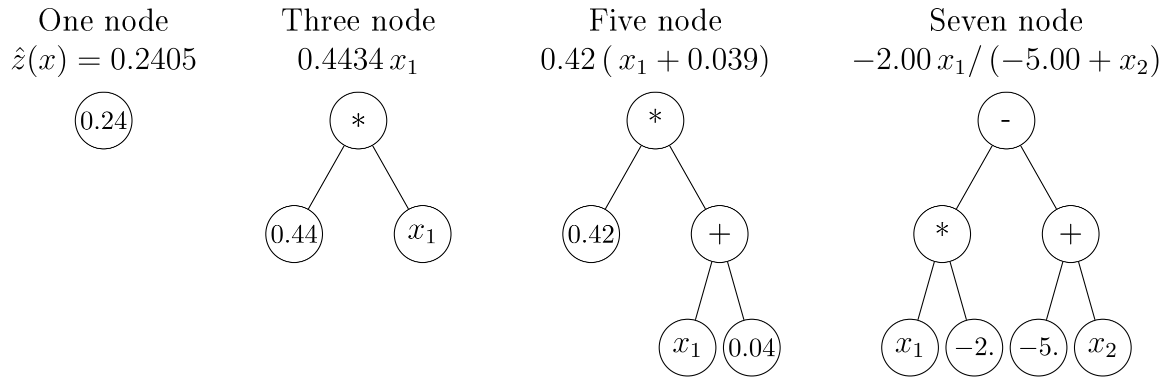


Figure 4.5: Expression trees with increasing complexity

Figure 4.6 shows the pareto curve of the sum of the squared error versus active nodes for this analysis. In general, desired models are selected using several criteria derived from the pareto front. For example, they may occur at the beginning of long flat plateaus or after a large cliff, here that might mean using the 3-node model $\hat{z}(x) = 0.4434\,x_1$. Alternatively, they may be chosen when the error reaches a desired level; in this case the error drops to zero with a 7-node model $\hat{z} = \dfrac{2\,x_1}{5 - x_2}$. The latter criterion would result in a match with the actual underlying function. However, models are often selected using a goodness-of-fit criterion such as Akaike information criterion [5], Bayesian information criterion [95], generalized information criterion [95], or cross-validation. For the expression trees in Figure 4.5, we solve the models using SCIP with a tightened constraint feasibility tolerance of $1 \cdot 10^{-10}$.

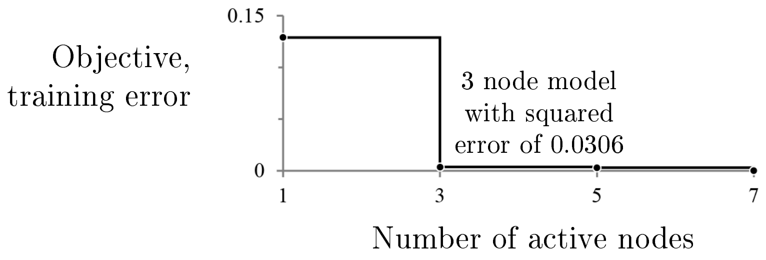


Figure 4.6: Pareto curve of error versus tree size for illustrative example

By solving this problem using symbolic regression, we are able to determine both the functional form and the values of associated modeling parameters that best fit the underlying data. Due to this algebraic formulation, we are able to solve this problem using a global optimization method to show that we arrive at the best possible model for each tree size.

4.3 Strengthening the formulation

In this section, we explore three methods to improve the model outlined in Section 4.2.1: redundancy elimination, symmetry-breaking cuts, and depth-based binary variable priorities.

Redundancy elimination

Often, there are multiple, equivalent formulations to express an underlying functional form. These can lead to degenerate or multiple solutions as well as increased computational resource requirements caused by comparisons between redundant solutions. For example, the following models are all equivalent:

$$\begin{aligned}
\text{Final :} \quad & \hat{z}(x) = 0.2\,(x + (-2)) \\
\text{Equivalent models :} \quad & \hat{z}_1(x) = 0.2\,(x + 2 - 4) \\
& \hat{z}_2(x) = 0.2\left(x - \sqrt{4}\right) \\
& \hat{z}_3(x) = 0.2\,(x - 2) \\
& \hat{z}_4(x) = \frac{x + (-2)}{5} \\
& \hat{z}_5(x) = 0.2\left(\sqrt{x^2} + (-2)\right).
\end{aligned}$$

To eliminate many unnecessary comparisons and enhance the performance of the optimization solver, we further constrain the symbolic regression problem by eliminating redundancies involving the parent node and its children. We exclude redundancies in three categories: the redundant manipulation of constants, redundant binary operations, inverse redundancies of unary operators.

To eliminate the unnecessary manipulation of constant terms, we include the following two logical constraints,

$$y^{\text{cst}}_{l_n} + y^{\text{cst}}_{r_n} \le 1 \qquad n \notin \text{TERM} \tag{4.2a}$$

$$y^{\text{cst}}_{l_n} \le 1 - \sum_{o \in \mathcal{U}} y^o_n \qquad n \notin \text{TERM} \tag{4.2b}$$

where constraint 4.2a eliminates all binary operations involving only constants and constraint 4.2b ensures unary operations do not have a constant argument. These two constraints result in the infeasibility of $\hat{z}_1$ and $\hat{z}_2$.

To avoid equivalent models that arise from a combination of addition and subtraction or multiplication and division, we restrict the subtraction and division operations as follows:

$$y^{-}_{n} + y^{\text{cst}}_{r_n} \le 1 \qquad n \notin \text{TERM} \tag{4.3a}$$

$$y^{\div}_{n} + y^{\text{cst}}_{r_n} \le 1 \qquad n \notin \text{TERM}. \tag{4.3b}$$

Constraint 4.3a allows for model $\hat{z}$ instead of $\hat{z}_3$ by disallowing the subtraction of a constant in favor of adding a negative number. Alternatively, we could eliminate this redundancy by disallowing the addition and subtraction of negative numbers. This would require constraining $v_{ir_n} \geq 0$ if $y_n^+ = 1$ or $y_n^- = 1$, however, and Equation 4.3a is a more favorable logical constraint that eliminates the same redundancy. Similarly, Equation 4.3b does not allow for dividing by a constant value which would eliminate model $\hat{z}_4$.

Lastly, we avoid performing nested inverse operations using

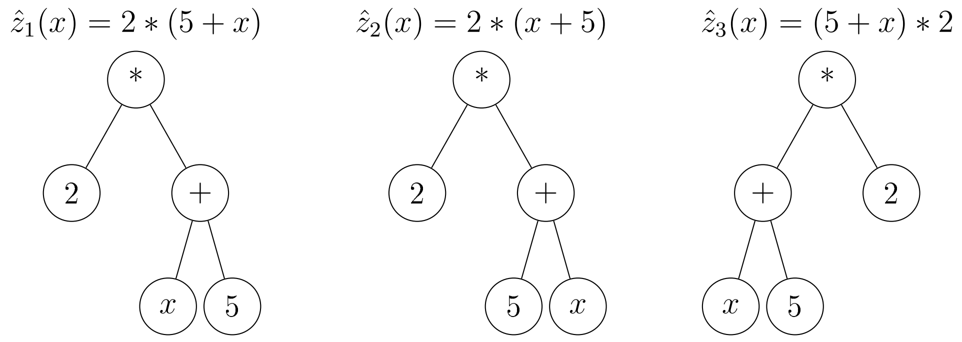
$$y_n^o + y_{l_n}^{o'} \leq 1 \qquad n \notin \text{TERM} \tag{4.4}$$

$$y_n^{o'} + y_{l_n}^o \leq 1 \qquad n \notin \text{TERM} \tag{4.5}$$

for all inverse unary operation pairs o and o', including squares and square roots and logarithms and exponentials. Constraints 4.4 and 4.5 eliminate $\hat{z}_5$ from the feasible model set.

Symmetry-breaking cuts

In addition to the redundant formulations above, many symmetries with expression trees lead to redundant solutions. In fact, for every symmetric operator, there is a corresponding tree symmetry. Figure 4.7 gives three equivalent trees with symmetries over order-independent operators $o \in \mathcal{I}$ where $\mathcal{I} := \{+, *\}$.



$\hat{z}_1(x) = 2 * (5 + x)$
$\hat{z}_2(x) = 2 * (x + 5)$
$\hat{z}_3(x) = (5 + x) * 2$

Figure 4.7: Three equivalent expression trees due to symmetric operators

Table 4.3: Binary priorities for the illustrative example

Binary variable		Priority
y_1^o	$o \in \mathcal{O}$	1
y_2^o, y_3^o	$o \in \mathcal{O}$	2
$y_4^o, y_5^o, y_6^o, y_7^o$	$o \in \mathcal{L}$	3

To eliminate these symmetries, we apply symmetry-breaking cuts

$$v_{i'l_n} - v_{i'r_n} \geq -M \sum_{o \in \mathcal{I}} y_n^o \tag{4.6}$$

for one data point $i = i'$ as a big-M constraint where M is set using interval arithmetic on the bounds of $v_{i'l_n}$ and $v_{i'r_n}$.

Node priorities

The solution speed of branching-based optimization methods is often sensitive to the priority given to integer or binary variables. For the symbolic regression problem, it is easy to see in Figure 4.1 that changes in node activity at higher tree depths have a greater impact on the final surrogate model while changes lower in the tree have less effect. To pass this information to the solver, we will set the priority of each node to its tree depth. For the illustrative example, we use the binary priorities listed in Table 4.3.

4.4 Demonstrations

In addition to the illustrative example, we have selected two problems that, at first appearance, are simple. However, they have been shown to be very challenging for standard symbolic regression packages [53, 101]. Standard symbolic regression packages employ genetic programming methods, which are inherently stochastic. Therefore, the metric used to compare the efficacy of these methods in the cited literature is the success rate as measured by the percentage of runs with a mean squared error less than some small tolerance.

For each instance, we provide results with and without redundancy constraints, symmetry-breaking cuts, and depth-based priority definitions from Section 4.3 including data for all eight combinations of these three elements. We also compare alternative problem objectives: minimizing the squared error and minimizing the number of active nodes for a given error tolerance. For these problems, we use a tolerance of zero.

All tests were performed using a `GAMS 24.2.2` interface to `SCIP 3.0` with `CPLEX 12.6.0.0`, `IPOPT 3.11`, `cppad 20120101.3`, and `GMP 4.2.2` on a 64-bit Genuine Intel E6420 2.13GHz processor.

4.4.1 Illustrative example

We continue with the illustrative example from Section 4.2.2, where we model the following function:

$$z(x) = \frac{2\,x_1}{5 - x_2}$$

using ten randomly sampled points in x_1 and x_2 from 0 to 1. We have already demonstrated the effectiveness of this formulation to determine this functional form of the model from these ten data points. In this section, we examine the efficacy of the three improvements introduced in Section 4.3.

This symbolic regression problem has 70 continuous and 33 binary variables. The base-case problem has 1,378 constraints which is increased by 15 when redundancy constraints are added and by 3 to include symmetry-breaking cuts.

To compare all eight combinations of constraints, we include solution times for the error minimization problem in Table 4.4. The results indicate that the most beneficial addition is the redundancy-eliminating constraint set. This is evidenced by a reduction of required computational time all four cases: from 710, 200, 270, and 68 seconds to 16, 40, 9, and 54 seconds, respectively. Adding redundancy-eliminating constraints leads to speed improvements of 43, 5.0, 28, and 1.3 times, where other conditions remain constant. Depth-based binary branching priorities speed up the solution by 2 to 5 times when redundancy constraints are enabled, and increase solution time when they are not used. The inclusion of symmetry-breaking constraints increases the solution time in most cases.

Table 4.4: Illustrative example – Solution time for the error minimization problem [seconds]

		Depth-based priorities	
Redundant constraints	Symmetry cuts	Use	Ignore
Use	Use	16	40
	Omit	9	54
Omit	Use	710	200
	Omit	270	68

We also investigate the active node minimization problem with resulting solution speeds summarized in Table 4.5. We observe that all additional elements are beneficial, both individually and when combined. Moreover, the solution improvement between the base case and the complete case, where the former has no additional elements included and the latter implements all three elements combined, is the most significant. The complete case solved in the shortest time, while the base case problem is not solved within the 1000 second time limit. The active node minimization problem solves, on average, 16 times faster than the error minimization problem for this example. The base case is the only notable exception because it could only be solved for the error minimization problem within the time limit.

Table 4.5: Illustrative example – Solution time for the active node minimization problem [seconds]

		Depth-based priorities	
Redundant constraints	Symmetry cuts	Use	Ignore
Use	Use	7	15
	Omit	11	13
Omit	Use	9	119
	Omit	12	>1000

On an individual basis, the most favorable additions to the error minimization problem are the redundancy-eliminating constraints. Additional improvement is observed with the inclusion of depth-based branching priorities. The same is true for the, easier to solve, active node minimization problem with the added freedom of depth-based priorities exhibiting improvements on an individual basis. Using the disjunction optimization formulation, we are able to identify and globally certify the best surrogate model using a set of simple operators and operands.

4.4.2 Example 2

For Example 2, we model the equation:

$$z(x) = x^2 + 100$$

from Keijzer [53] where the input x consists of 21 evenly spaced points between -1 and 1. Keijzer reports a success rate of 16% with an error tolerance of 0.001 on this simple problem using genetic programming. In other words, 16% of the trials return an expression tree model with a mean squared error less than 0.001.

We model the same data using addition, subtraction, multiplication, division, and square root operators applied to operands x and a constant term. This allows for the same solution flexibility the Keijzer allowed for in [53]. We bound each node value v_{in} for $n > 1$ to be within ±150, the same criteria used in the illustrative example to bound the root node values, and allow a maximum tree depth of four.

The resulting optimization model has 421 variables (105 binary and 315 continuous). The base problem contains 9,545 constraints, while redundancy-elimination requires 42 additional constraints and symmetry-breaking constraints require 7. Due to the nature of the operators in this problem, we are able to solve a nonconvex mixed-integer quadratically constrained program (MIQCP).

The optimal final expression tree is depicted in Figure 4.8. This tree represents the optimal solution for all eight cases; however, each case may result in different, but equivalent, trees primarily due to cases where redundant constraints and symmetry cuts are not used.

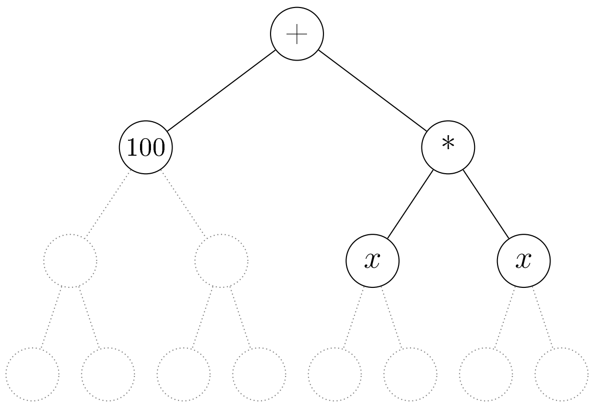


Figure 4.8: Optimal expression tree for Example 2

For this instance, the solution of the active node minimization problem is found to be more rapid and reliable. Table 4.6 shows the solution times for all eight cases. All cases are solved in less than one minute; whereas, for error minimization problem, only a single case could be solved within an hour. For this problem, using depth-based binary branching priorities has little effect on the solution speed and the addition of symmetry-breaking cuts was, in general, not beneficial. The redundancy-elimination constraints are beneficial in conjunction with symmetry-breaking cuts. In fact, the best performing case has no additional constraints, with or without branching priorities enabled. This is likely because the added constraints significantly reduce the feasible space. For this instances, feasible point identification is the primary difficulty due the zero tolerance specification on the error.

Table 4.6: Example 2 – Solution time for the active node minimization problem [seconds]

		Depth-based priorities	
Redundant constraints	Symmetry cuts	Use	Ignore
Use	Use	26	28
	Omit	14	10
Omit	Use	54	32
	Omit	5	5

When minimizing model error, the results convey the benefits of added constraints. Out of the eight cases tested, only one is solved to optimality within a one hour time limit;

the combination of redundancy constraints and depth-based priorities decreases the solution time to 893 seconds. The objective values after 3600 seconds are provided for each case in Table 4.7.

The solution methods employed by SCIP involve the generation of upper objective bounds on feasible solutions and certifiable lower objective bounds [102]. As the optimization routine progresses, the distance between these bounds closes until the algorithm converges. If the solver terminates before this, we compare progress by looking at the upper and lower objective bounds. The smaller this gap, the better the progress. Therefore, we report the best found objective values, or upper bounds, after 3600 seconds for each case in Table 4.7. For all cases, the lower bound is zero.

Table 4.7: Example 2 – Objective values for the error minimization problem found after 3600 seconds

		Depth-based priorities	
Redundant constraints	Symmetry cuts	Use	Ignore
Use	Use	2.24184	2.24327
	Omit	*0.00000	0.55022
Omit	Use	0.00006	0.55022
	Omit	0.00000	–

Missing entries indicate the absence of a feasible solution after 3600 seconds.
* Optimal solution found within time limit (after 893 seconds)

For these tests, we observe that using either redundancy or symmetry-breaking constraints is useful. A combination of these constraint classes, however, is not beneficial. Depth-based priorities showed improved solutions in every case. Using none of the added constraints or priorities prolongs feasible solution detection. Symmetry-breaking cuts appear to hinder this problem in three out of four tests.

After formulating this problem using a rigorous disjunctive model, we use a deterministic global optimization which identifies the exact functional form. The solution of this problem is also certified to be globally optimal and feasible. For this example, the best overall conditions combine redundancy-elimination constraints with depth-based priorities for the active node minimization problem. However, if we analyze the active node minimization problem independently, we see that tests without symmetry-breaking cuts are the best performers.

4.4.3 Example 3

Our third test instance is a problem from Uy et al. [101],

$$z(x) = x^3 + x^2 + x.$$

This, seemingly simple, function has been classified as a "difficult synthetic problem" and has been incorporated into the GP Benchmark library [71, 107]. Uy et al. reports a success rate of about 50% for an error tolerance of 0.01 on this problem using state-of-the-art symbolic regression implementations.

We model this function using all operators reported by Uy et al. with the exception of trigonometric functions due to solver incompatibilities with these functional forms. The operator and operand sets consists of $o \in \{+, -, *, \div, \exp, \log, x, \text{cst}\}$. We have also matched the range, training sample size, and sample method from [101] by randomly sampling 20 points from -1 to 1 for x. We bounded each node value for $n > 1$ to be within ± 25, used the same root node bound criteria as the illustrative example, and allowed a maximum tree depth of four.

The symbolic regression optimization model for this problem has 300 continuous and 120 binary variables with 9,047 constraints. The additional redundancy constraints and symmetry-breaking cuts add 83 and 7 constraints, respectively. The inclusion of the exponential and logarithmic unary operators constitutes a mixed-integer nonlinear problem formulation.

The final expression tree for the optimal case is not the expected eleven-node function $x * x * x + x * x + x$. Instead, the solution has factored out an x returning the nine-node function $x * (x + 1 + x * x)$. The nine-node model is illustrated in Figure 4.9.

By solving the active node minimization problem, the optimal solution is found after 1241 seconds using redundancy eliminating constraints and depth-based binary variable priorities.

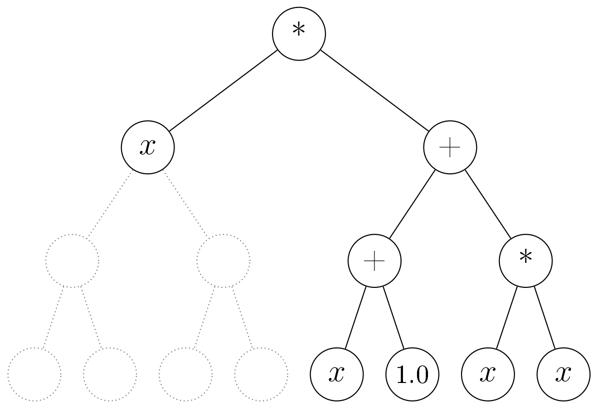


Figure 4.9: Optimal expression tree for Example 3

The upper and lower objective bounds after 2000 seconds are tabulated in Table 4.8. We observe that depth-based priorities and redundancy-eliminating constraints are beneficial both independently and when used in concert. The lower bound closure is improved or increased when using both the redundancy constraints and priorities. Most notably, depth-based priorities are necessary to find a feasible solution to this problem.

Since the true goal is to find accurate models, it may be sufficient to terminate when a feasible solution to the node minimization problem is found. For this problem, all feasible solutions are located between 175 and 184 seconds. In fact, the solution found at this point is feasible an optimal; the remaining solution time is spent certifying its global optimality.

Table 4.8: Example 2 – Objective values and lower bounds for the active node minimization problem found after 2000 seconds.

		Depth-based priorities	
		Use	Ignore
Redundant constraints	Symmetry cuts	[Lower bound, Best solution found]	
Use	Use	[8, 9]	[5, –]
	Omit	*[9, 9]	[5, –]
Omit	Use	[7, 9]	[5, –]
	Omit	[7, 9]	[5, –]

Missing entries indicate the absence of a feasible solution after 2000 seconds.
* Optimal solution found within time limit (after 1241 seconds)

No optimal solution is located after 3600 seconds when solving the error minimization problem; therefore, we list the best feasible objectives in Table 4.9. The lower bound for all cases is zero. Redundancy-eliminating constraints provide the best and most consistent reduction in the objective upper bound. The most successful conditions involved the use of both symmetry-breaking and redundancy-elimination constraints. Depth-based priorities appear to hinder the solution of this problem.

Table 4.9: Example 3 – Objective values for the error minimization problem found after 3600 seconds

		Depth-based priorities	
Redundant constraints	Symmetry cuts	Use	Ignore
Use	Use	2.07743	0.03054
	Omit	1.48318	0.34040
Omit	Use	7.03405	7.03405
	Omit	7.03405	7.03405

Missing entries indicate the absence of a feasible solution after 3600 seconds.

Similar to Example 2, we are able to certify the global optimality of the most accurate surrogate model. The proposed optimization formulation can be solved both rigorously and deterministically; therefore, we eliminate stochastic concerns that result in a 50% success rate on this problem when genetic methods are used [101]. The active node minimization problem outperforms the error minimization problem for this example. The best conditions for the active node minimization problem remain the use of redundancy-elimination cuts and depth-based branching priorities. The use of symmetry and redundancy-eliminating constraints is beneficial for the error minimization problem.

4.5 Conclusion

Symbolic regression methods select the structure of an expression tree as well as parameter values to flexibly define a nonlinear surrogate model. Conventional symbolic regression techniques use genetic algorithms to search for an optimal surrogate model. However, it has been shown that these methods often are unable to return an accurate solution with respect to model value mismatch and/or model form mismatch. We present a disjunctive optimization formulation to rigorously solve the symbolic regression problem and demonstrate its efficacy using a deterministic global optimization solver. The result is a symbolic regression method that is rigorous without being hindered by stochastic components that lead to low success rates.

We present and study solutions to a disjunctive symbolic regression formulation that identifies the exact modeling functional form and perfect model value matching. We find the active node minimization problem to be the strongest formulation after comparing this objective to error minimization. We also explore the benefits of adding redundancy-eliminating constraints imposed among a parent and its children nodes, cuts to break symmetries over symmetric operators, and the use of binary variable priorities based on the tree-depth of each node. The use of redundancy-constraints and depth-based priorities consistently improve the active node minimization problem. The results also indicate that, most often, the same is true for the error minimization problem; however, if poor performance is observed, symmetry-breaking cuts may be beneficial.

Chapter 5

Surrogate-based optimization of carbon capture processes

We apply surrogate modeling to address the design and optimization of post combustion carbon dioxide capture systems. The proposed methodology combines both a systems-based approach to process synthesis and high-fidelity, multi-scale process simulators for accurate representation of the chemical systems. These approaches are conventionally incompatible for rigorous, deterministic optimization as well as black-box optimization techniques. To overcome these obstacles, we utilize a surrogate-based optimization approach to construct simple, accurate, tailored surrogate models that serve as a proxy for high-fidelity simulators in a rigorous optimization framework.

In this chapter, we present a superstructure optimization strategy to design a flowsheet model as well as reactor geometries and operating conditions for a post combustion carbon capture process. We outline aspects of the underlying processes that define high-fidelity reactor simulators and detail surrogate modeling techniques used to preform the surrogate-based approach. Finally, we present surrogate modeling results that are currently used at the National Energy Technology Laboratory for superstructure optimization.

5.1 Introduction

Recent developments have introduced a variety of emerging technologies to capture CO_2 from point sources, such as power plants. As the number of technological combinations increases, a rapid screening process to compare best-case scenarios is necessary to identify the most promising selections. When ranking combinations of technologies, an impartial metric reflecting an optimal balance of trade-offs with respect to environmental and economic costs is required.

To leverage a full set of process and technology trade-offs, we use a systems-based approach to process synthesis. This approach ensures a fair screening of potential technologies by comparing each combination at its optimal set of process parameters: temperatures, pressures, geometries, flow rates, etc.. In this work, we utilize a systems-based approach to process synthesis for the design, or more specifically, the modification of an existing power plant. Such systematic numerical solution techniques have largely replaced intuitive, heuristic, and ad hoc structure development and are widely implemented in computer modeling systems, simulation packages, and optimization techniques [13].

This chapter uses a blend of new and traditional process synthesis techniques to enable promising concepts to be more quickly identified through rapid computational screening of devices and processes. By integrating high fidelity, multi-scale process simulations with advanced optimization software, we aim to rapidly screen new concepts and to promote more focused and effective pilot and demonstration scale projects.

Carbon capture and sequestration is one of the three primary options to reduce the total CO_2 emissions to the atmosphere [109] along with reducing energy use and switching to non-fossil fuels. Globally, fossil-fueled power plants lead all other industries in CO_2 emissions, accounting for 33-40% of the total [19, 94]; therefore, the Carbon Capture Simulation Initiative (CCSI) effort has focused on reducing emissions from the power industry. Carbon capture and sequestration begins with the separation of CO_2 from flue gas. The CO_2-rich gas stream is then compressed and injected into existing geological formations. The most energy intensive step in the process is the separation of the CO_2 from the flue gas stream [109]. By extension, this carbon capture step offers the most potential for improvement.

Advanced discrete optimization techniques address high-level flowsheet decisions, such as network configuration; reactor type selection; and number of required units. In these advanced formulations, continuous decisions representing flow rates, temperatures, and geometries are determined simultaneously. We implement a superstructure optimization approach to select carbon capture technologies and network configurations. A *superstructure of decisions* optimizes all synthesis decisions simultaneously to find the best embedded alternative within the superstructure formulation [13].

Traditional superstructure optimization formulations include algebraic objectives and constraints that relate decision variables through first principles models, mass and energy balances, individual unit models, and design constraints. However, for more complicated units and processes, these relationships may not be available in algebraic form with sufficient accuracy. For the units examined here, high-fidelity simulations are constructed to represent underlying chemical processes and providing access to rigorous thermodynamic packages. Typically, these so-called black box problems are optimized using derivative-free solvers. However, these methods lack the optimality certifications provided by derivative-based solvers. Additionally, they fail to meet the flexibility requirements of superstructure optimization for complicated systems.

To maximize the potential of our high-fidelity unit simulations, we extend these superstructure methods using the surrogate modeling software ALAMO [24] to generate a rich set of simple algebraic models based upon empirically derived simulation data. The resulting empirical models are tailored to include appropriate mathematical characteristics that ensure consistency with numerical simulators while enabling a rigorous superstructure formulation.

The remainder of the chapter is organized as follows. In the next section, we outline our superstructure formulation with specific attention to the input/output structure of units that are described using surrogate models. In Section 5.3, we detail the methods used to generate surrogate models. Details of the high-fidelity simulations used to represent reactor units and other process components are included in Section 5.4. Finally, provide computational results generated using high-fidelity simulations, surrogate modeling techniques, and process flowsheet optimization for an industrial carbon capture system.

5.2 Superstructure formulation

In this section, we describe a superstructure formulation used to (a) minimize the total cost including capital, operating, maintenance, utilities, and power expenses while (b) achieving a 90% capture target. The capture target is the percentage of CO_2 extracted from the flue gas into the CO_2-rich gas stream. A general schematic representing the optimization superstructure for a carbon capture process consisting of adsorbers and regenerators in series as well as other associated units is depicted in Figure 5.1.

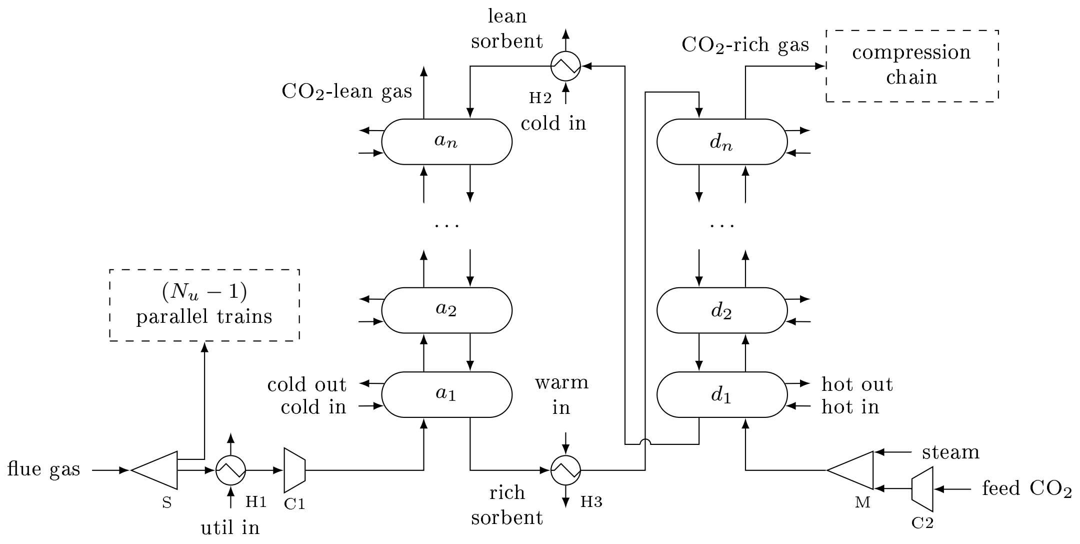


Figure 5.1: Superstructure carbon capture flow diagram

Flue gas from a 650MW pulverized coal power plant is split evenly between N_u identical carbon capture trains. For each carbon capture train, the flue gas is cooled and compressed using heat exchanger H1 and compressor C1. The flue gas stream is then contacted counter-currently with a CO_2-selective sorbent using a series of adsorbers, a_s for stages $s \in \{\text{ADS} = 1, 2, \ldots, n_{\text{ads}}\}$, resulting in a CO_2-lean or clean gas stream and a CO_2-rich sorbent stream. The rich sorbent stream is heated using exchanger H3 before regeneration in a series of desorbers, d_s for stages $s \in \{\text{REG} = 1, 2, \ldots, n_{\text{reg}}\}$, using compressed feed CO_2 and steam. Finally, the lean sorbent is cooled in exchanger H2 and recycled back into the adsorber chain.

In order to simplify the complexity of the optimization problem, surrogate models are used for the reactors, while first principle models are constructed to represent the dynamics the compressors, C1 and C2; heat exchangers, H1, H2, and H3; splitter, S; and mixer, M. The resulting algebraic equations can be combined to form a mixed-integer nonlinear problem (MINLP). The objective and constraints used to describe adsorbers, regenerators, and other associated units are detailed in the remainder of this section.

5.2.1 Objective: Cost of electricity

The goal of this optimization formulation is to select the optimal superstructure flowsheet, unit geometries, and operating conditions that minimize the estimated increase in the cost of electricity COE based on a 2007 650MW power plant with post combustion carbon extraction and the primary design constraint of 90% carbon capture.

The cost of electricity is estimated using the total overnight cost TOC, fixed operation and maintenance costs OC_{FIX}, variable operation and maintenance costs (including fuel) OC_{VAR}, and annual net megawatt-hours of power MWh generated at a 100% capacity factor CF [15]. Equation 5.1 summarizes the functional form of the optimization objective:

$$COE = \frac{CCF \cdot TOC + OC_{\text{FIX}} + CF \cdot OC_{\text{VAR}}}{CF \cdot MWh} + COE_{\text{TS\&M}}, \tag{5.1}$$

where CCF is the capital charge factor, CF is the plant capacity factor, and $COE_{\text{TS\&M}}$ is the cost for CO_2 transport, storage and monitoring. The operating costs are a function of equipment and unit geometries and specifications. These include reactor depth and diameter, the number of heat exchanger tubes and their dimensions, and the number of required reactor stages. Transportation costs for the major stages are estimated using cost correlations from [91] and a Lang factor is applied to obtain TOC. Fixed and variable operation and maintenance costs are calculated using process information or as a fraction of the TOC. For additional information regarding the correlations and functional forms used in the estimation of COE, we refer the reader to [76].

5.2.2 Constraint and variable descriptions

We employ several sets of integer variables to describe flowsheet decisions and define chemical process units. Discrete decisions and unit-specific constraints are detailed in this section. Unless otherwise stated, mass and component mass balances are enforced over each unit, heat exchanger areas are calculated using an estimated overall heat transfer coefficient with a log mean temperature driving force, and standard compression models are used to represent vapor phases [91]. Superscript and supscripts are denote streams or units, flow directions, and components of a given variable or property. In all descriptions, the temperatures, pressures, flow rates, component molar flow rates, and composition are denoted as T, P, F, C, and x or z, respectively.

Discrete decision variables

Discrete decisions involved in this super structure include: the number of parallel chains, the number of adsorber and regenerator stages, and technology specifications for each stage. The number of parallel trains, N_u, allows for the processing of large flue gas flow rates despite a 10m limit on the maximum diameter of each reactor. There are a maximum of n_{ads} adsorbers and n_{reg} regenerators permitted in the formulation. If a stage i is used in the flow sheet, its corresponding binary variable $y_i = 1$ for $i \in \text{ADS} \cup \text{REG}$ and is zero otherwise. Each adsorber a_s and regenerator d_s can take on one of two technologies included in this work: bubbling fluidized beds operating in either overflow and underflow modes. The following equation ensures that only one technology is used at any active stage:

$$\sum_{t \in \text{TN}} x_{it} = y_i \qquad i \in \text{ADS} \cup \text{REG}. \tag{5.2}$$

These technologies make up the technology set TN= $\{\text{overflow}, \text{underflow}\}$.

Adsorbers

An individual adsorber a_s at stage s uses a counter current configuration to contact gas and sorbent material. Figure 5.4 shows the variables associated with this reactor.

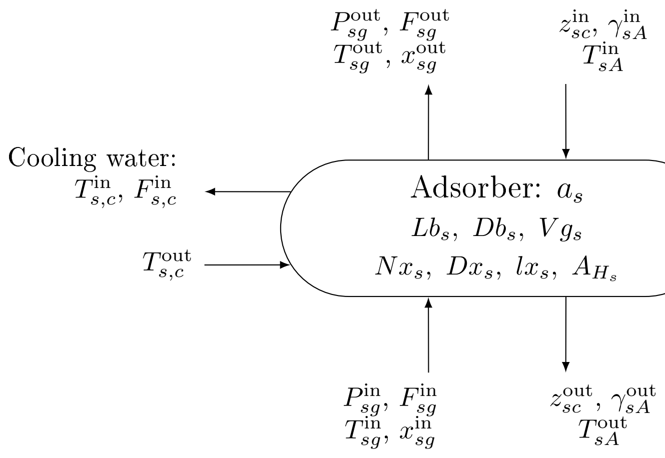

P_{sg}^{out}, F_{sg}^{out}
T_{sg}^{out}, x_{sg}^{out}
z_{sc}^{in}, γ_{sA}^{in}
T_{sA}^{in}
Cooling water:
$T_{s,c}^{\text{in}}$, $F_{s,c}^{\text{in}}$
$T_{s,c}^{\text{out}}$
Adsorber: a_s
$Lb_s,\ Db_s,\ Vg_s$
$Nx_s,\ Dx_s,\ lx_s,\ A_{H_s}$
P_{sg}^{in}, F_{sg}^{in}
T_{sg}^{in}, x_{sg}^{in}
z_{sc}^{out}, γ_{sA}^{out}
T_{sA}^{out}

Figure 5.2: Adsorber diagram

The gas, with properties denoted by a subscript g, flow upward while the sorbent stream, denoted with a subscript A, flows downward. Additionally, the reactor contains heat exchanger tubes, where cooling water properties are denoted with a subscript c. The adsorber has a bed depth Lb_s, a reactor diameter Db_s, a superficial gas velocity of Vg_s, and Nx_s heat exchanger tubes with a diameter of Dx_s and length of lx_s resulting in a heat exchanger area of A_{H_s}. The remainder of the input/output relationships are determined using surrogate models of simulated data. The specifications relating properties for each technology to inlet and geometric conditions are as follows:

$$\Omega = \sum_{t \in \text{TN}} x_{st} \hat{\Omega}_t \left(\phi_{\text{flue}}^{\text{gas,out}}, \phi_{s+1}^{\text{sorb,out}}, \phi_s^{\text{ads}} \right) \qquad s = 1 \tag{5.3}$$

$$\Omega = \sum_{t \in \text{TN}} x_{st} \hat{\Omega}_t \left(\phi_{\text{s}-1}^{\text{gas,out}}, \phi_{s+1}^{\text{sorb,out}}, \phi_s^{\text{ads}} \right) \qquad s \in \text{ADS} \setminus (1 \cup n_{\text{ads}}) \tag{5.4}$$

$$\Omega = \sum_{t \in \text{TN}} x_{st} \hat{\Omega}_t \left(\phi_{\text{s}-1}^{\text{gas,out}}, \phi_{d}^{\text{sorb,out}}, \phi_s^{\text{ads}}, T_{H2}^{\text{out}} \right) \qquad s = n_{\text{ads}}, d = 1 \tag{5.5}$$

where Ω is a proxy for each variable that requires a surrogate model $\hat{\Omega}$ including outlet temperatures, pressures, flow rates, and compositions as well as gas velocity, cooling water properties, and the number of heat exchanger tubes. For conciseness, we group the input variables into three categories: inlet gas properties, inlet sorbent properties, and unit specific properties. The inlet gas variable group $\phi_i^{\text{gas,out}}$ is comprised of gas properties leaving unit i including gas composition, x_{fc}, $fc = \{CO_2, H_2O, N_2\}$; pressure,P; temperature, T; and flow rate, F. The sorbent properties entering adsorber a_s and exiting unit i are part of variable group $\phi_i^{\text{gas,out}}$ including z_{sc}, $sc = \{HCO_3, H_2O, NH_2COO\}$; sorbent fraction from the top stage, γ; and sorbent temperature, T, as appropriate—excepting the sorbent temperature into $a_{n_{\text{ads}}}$ which results from the outlet temperature from H2, T_{H2}^{out}. Unit-specific properties collected in variable group ϕ_s^{ads} include cooling water flow rate and temperature, F_c^{in} and T_c^{in}; bed length and diameter, Lb and Db; diameter and length of heat exchanger tubes, Dx, and lx_s; and superficial gas velocity, Vg.

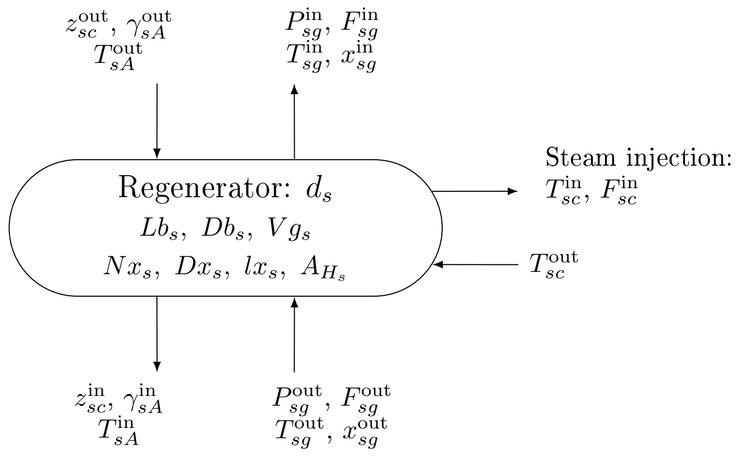
The area of heat change is calculated using the following formula:

$$A_{H_s} = \pi \, Nx_s \, Dx_s \, lx_s. \tag{5.6}$$

A detailed description of the technology used to accomplish gas-sorbent contacting is provided in Section 5.4.

Regenerators

Each regenerator d_s at stage s has a corresponding configuration to adsorber a_s and regenerates sorbent material for recycle. Figure 5.3 depicts the gas and sorbent properties, heat exchanger, and reactor geometries for the regenerator at stage s.



z_{sc}^{out}, γ_{sA}^{out}
T_{sA}^{out}
P_{sg}^{in}, F_{sg}^{in}
T_{sg}^{in}, x_{sg}^{in}
Steam injection:
T_{sc}^{in}, F_{sc}^{in}
Regenerator: d_s
$Lb_s,\ Db_s,\ Vg_s$
$Nx_s,\ Dx_s,\ lx_s,\ A_{H_s}$
T_{sc}^{out}
z_{sc}^{in}, γ_{sA}^{in}
T_{sA}^{in}
P_{sg}^{out}, F_{sg}^{out}
T_{sg}^{out}, x_{sg}^{out}

Figure 5.3: Regenerator diagram

The regenerator heat exchanger utilizes steam injection for reactor heating and is defined by a similar set of surrogate models:

$$\Omega = \sum_{t \in \text{TN}} x_{st} \hat{\Omega}_t \left(\phi_{\text{feed CO}_2}^{\text{gas,out}}, \phi_{s+1}^{\text{sorb,out}}, \phi_s^{\text{reg}} \right) \qquad s = 1 \tag{5.7}$$

$$\Omega = \sum_{t \in \text{TN}} x_{st} \hat{\Omega}_t \left(\phi_{\text{s}-1}^{\text{gas,out}}, \phi_{s+1}^{\text{sorb,out}}, \phi_s^{\text{reg}} \right) \qquad s \in \text{REG} \setminus (1 \cup n_{\text{reg}}) \tag{5.8}$$

$$\Omega = \sum_{t \in \text{TN}} x_{st} \hat{\Omega}_t \left(\phi_{\text{s}-1}^{\text{gas,out}}, \phi_{a}^{\text{sorb,out}}, \phi_s^{\text{reg}}, T_{H3}^{\text{out}} \right) \qquad s = n_{\text{reg}}, a = 1 \tag{5.9}$$

with variable grouping as detailed in the previous section. However, gas properties entering d_1 are defined by the steam and compressed feed CO_2 stream; while, sorbent properties into d_1 are defined by the outlet temperature from heat exchanger H3 and remaining sorbent properties from a_1. The heat exchange area is calculated using Equation 5.6.

Flue gas heat exchanger

The flue gas heat exchanger H1 is intended to cool the flue gas to a temperature that is more amenable to the process. The input/output structure of this exchanger is shown in Figure 5.4.

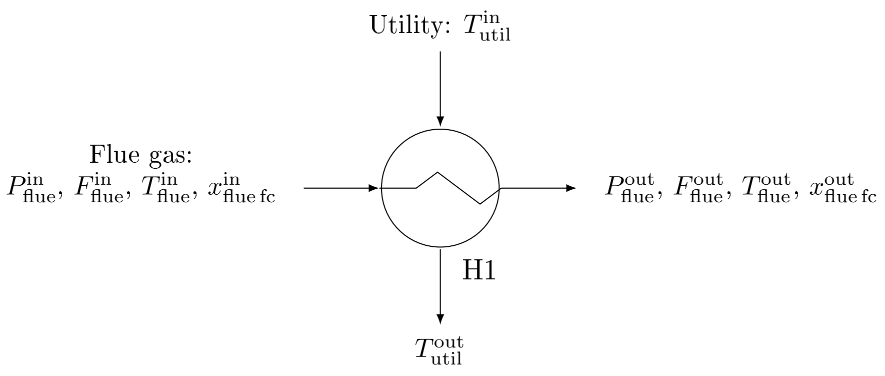

Utility: $T_{\text{util}}^{\text{in}}$
Flue gas:
$P_{\text{flue}}^{\text{in}}, F_{\text{flue}}^{\text{in}}, T_{\text{flue}}^{\text{in}}, x_{\text{flue fc}}^{\text{in}}$
$P_{\text{flue}}^{\text{out}}, F_{\text{flue}}^{\text{out}}, T_{\text{flue}}^{\text{out}}, x_{\text{flue fc}}^{\text{out}}$
H1
$T_{\text{util}}^{\text{out}}$

Figure 5.4: Flue gas heat exchanger diagram

The energy and hydrodynamic balances are described using existing empirical models. The flow rate of the cooling utility water is determined using [91]:

$$T_{\text{util}}^{\text{out}} F_{\text{util}} = T_{\text{in}}^{\text{out}} F_{\text{util}} + \frac{40683 \left(\frac{374 - T_{\text{flue}}^{\text{out}}}{274}\right)^{0.38} \left(C_{\text{H}_2\text{O}}^{\text{in}} - C_{\text{H}_2\text{O}}^{\text{out}}\right) \sum_{fc} cP_{fc} C_{fc}^{\text{in}} \left(T_{\text{flue}}^{\text{out}} - T_{\text{flue}}^{\text{in}}\right)}{cP_{H_2O}} \tag{5.10}$$

where cP_i is the heat capacoty of component i.

5.3 Surrogate model development

To incorporate detailed simulations of the adsorber and regenerator models into the super structure optimization framework, we require the generation of a set of algebraic surrogate models for each reactor and flow condition. To accomplish this, employ a software package ALAMO (Automated Learning of Algebraic Models for Optimization). We refer the reader to [24] for a more detailed description of these techniques.

To generate a set of surrogate models, the carbon capture reactor is simulated over a range of likely operating and design conditions. The resulting surrogate models are combined along with algebraic connectivity, superstructure logic, and design constraints to complete the superstructure optimization model. This approach has seen recent advances in chemical engineering design, where an existing modeling method is used to model components of a system. Henao and Maravelias [43] have used neural networks to approximate individual process units to formulate superstructure models. Kriging models have been used by Caballero and Grossmann[18] on disaggregated systems while more recent work models the entire process [28, 45, 79]. These models are large and complex and can oftentimes lead to intractable final superstructure optimization problems. Typically, these methods focus on developing models that are highly accurate. ALAMO develops models that are not only accurate but are also tailored to promote optimization of the final superstructure model. By considering the final purpose of the surrogate models, it is possible to identify functional forms that can be easily incorporated into larger optimization models without the difficulty of inherently complex surrogate models.

A three-step algorithm is used to identify surrogate models. After an initial design of experiments is generated and the simulation is sampled at these points, an initial surrogate model is generated using this data. To determine the accuracy of the surrogate, an error maximization adaptive sampling method is used to locate new areas of model mismatch. If the model is shown to be sufficiently accurate, the algorithm terminates. Otherwise, the newly sampled simulation points are added to the training set and the model is rebuilt.

Both the functional form and complexity of the surrogate model, $\hat{z}_k$, for each output variable z_k are unknown. However, we allow for large set of potential functional forms. ALAMO defines the set of simple basis functions $X_j(x),\ j \in \mathcal{B}$ that, when combined, serve as a flexible underlying functional form for the surrogate model. The set of simple basis functions $\mathcal{B}$ can be selected from physical principles, engineering experience, simple inspection, or statistical fitting functions. The final surrogate model for z_k becomes:

$$\hat{z}(x) = \sum_{j \in \mathcal{B}} \beta_j \, X_j(x) \tag{5.11}$$

where each basis function j is multiplied by a corresponding coefficient or regression parameter β_j. If the resulting model is comprised of every possible basis function available to ALAMO, the surrogate model would run into the same complexity obstacles as kriging and neural networks. Instead, a subset of basis functions is identified to remain in the final surrogate. The subset is chosen to minimize over-fitting of the model as well as to retain tractability of the final optimization problem.

To select the best subset model, ALAMO begins with a small number of allowed terms T. ALAMO solves the best T subset problem minimizing linear error using the mixed-integer linear problem (P):

$$\begin{array}{llll}
(P) & \min & \sum_{i=1}^{N} e_i & \\
& \text{s.t.} & e_i \geq z_i - \sum_{j\in\mathcal{B}} \beta_j \, X_j(x_i) & i = 1, 2, \ldots, N \\
& & e_i \geq \sum_{j\in\mathcal{B}} \beta_j \, X_j(x_i) - z_i & i = 1, 2, \ldots, N \\
& & \sum_{j\in\mathcal{B}} y_j = T & \\
& & \beta_j^{\text{lo}} y_j \leq \beta_j \leq \beta_j^{\text{up}} y_j & j \in \mathcal{B} \\
& & \beta_j \in [\beta_j^{\text{lo}}, \beta_j^{\text{up}}] & j \in \mathcal{B} \\
& & y_j \in \{0, 1\} & j \in \mathcal{B}.
\end{array}$$

where e_i is the absolute linear error between the model and the simulated data at each point i and the simple basis function, $X_j(x)$, $j \in \mathcal{B}$, are active when the corresponding binary variable $y_j = 1$ and inactive when $y_j = 0$. The size of the model, specified by the parameter T in first constraint, is increased until a goodness-of-fit measure, such as the corrected Akaike Information Criterion [46], worsens with an increase in model size.

For this work, we include more than 1000 basis functions in the basis set. With so large a basis set, (M) is difficult to solve to optimality for all but small values of T. Therefore, we use the best feasible model found after a predetermined solution time. This provides a heuristic solution to the best subset problem. Compared to other common alternative heuristic methods, we have seen vast improvements with this selection criteria. This is because (M) is able to find a near optimal solution by comparing the synergistic effects of all terms simultaneously without being hindered by correlated basis functions.

At this point, a traditional ALAMO application would use built-in error maximization sampling to determine new simulation points that serve to improve the surrogate models and define a stopping criterion. However, for the purposes of this work, we use a fixed data set.

Once the surrogate models for all process blocks are generated, they are used in conjunction with connectivity and design constraints to formulate a superstructure optimization problem to minimize the increased COE.

5.4 Process components

In this section, we describe the process units that will make up the superstructure formulation for the solid sorbent post combustion carbon capture process.

5.4.1 Carbon capture adsorber and regenerator models

Adsorber(s) and the regenerator(s) are the two primary units in a carbon capture system. The adsorber, or series of adsorbers, removes CO_2 from the flue gas stream through contact with a CO_2 selective sorbent. The outlet streams are the CO_2-lean flue gas stream and CO_2-rich sorbent stream this is be recycled after regeneration. The regenerator removes CO_2 from the sorbent resulting in desorbed sorbent that can be reused and a CO_2 stream that is compressed for storage.

In the context of this case study, the sorbent is an amine-impregnated, mesoporous sorbent developed at NETL (NETL 32D). This solid sorbent consists of a mixture of polyethyleneimine (PEI) and aminosilanes impregnated into the mesoporous structure of a silica substrate. CO_2 affinitty is achieved through chemical reactions between the amine sites within the sorbent and gaseous CO_2. The result is a formation of solid species bound within the sorbent. Unfortunately, it has been demonstrated that the presence of water has a significant effect on the equilibrium and kinetics of the CO_2 adsorption because the sorbent also shows a strong affinity for water [76]. Lee et al. [67] developed a lumped parameter kinetic model for the adsorption of both CO_2 and water onto this sorbent which takes into account differing reaction pathways and amine site limitations.

In this study, we focus on bubbling fluidized beds (BFB) to enable gas-solid contact over two regimes: (1) overflow and (2) underflow. In (1) the solid sorbent stream exits from the top of the reactor, while in (2) the outlet solid stream is configured at the bottom. Based on a screening analysis of available gas-solids contacting equipment, a detailed model for the this fluidized bed was developed in Aspen Custom Modeler (ACM) [2]. Gas-solids contacting equipment can be grouped into four broad categories differentiated by the way that gas and solids move through the system, each with unit-specific trade offs: fixed beds, moving beds, bubbling fluidized beds, and circulating fluidized beds. Bubbling fluidized bed reactors are used in a wide range of industrial processes, such as fluidized catalytic cracking and combustion of biomass [76]. Also promising are moving bed reactors which have been applied in several gas-solid contacting processes in the metallurgical, chemical, and petroleum industries [76]. Though several other technologies seem promising, we will limit our search to BFB reactors for the scope of this study.

The bubbling fluidized bed model used here was developed by Lee and Miller [68]. It expands upon the hydrodynamic model of Kunii and Levenspiel [66] to develop a complete one-dimensional model that is capable of modeling both adsorption and regeneration processes. The model includes immersed heat exchange tubes that modify the temperature of the solids bed and incorporate the effects of chemical reactions on hydrodynamics. The models were developed to capture sufficient behavioral details with the aim of providing accurate and predictive results while remaining computational tractability. In order to satisfy these competing requirements, one-dimensional models based on systems of partial differential equations were developed. One dimensional models neglect any radial variations that occur within the reactors due to the non-uniform distribution of gas and solids or wall effects; however, the computational complexity of these models is significantly decreased compared to two- or three- dimensional computational fluid dynamics models. The systems of partial differential equations used in the models addresses hydrodynamic behavior, interactions of the gas and solids, heat and mass transfer phenomena, and the kinetics of the adsorption and desorption reactions.

5.4.2 Power plant model

The carbon capture process is configured as a post combustion addition to a 650MW supercritical pulverized coal power plant. To power the carbon capture process, a parasitic load is applied to the powerplant. It is this loss of steam that produces the majority non-capital losses and, ultimately, increases in the cost of electricity discussed in Section 5.2.1. Derating of the power plant output due to steam extraction is estimated using Equation 5.12.

$$\text{net power, kW} = -420.42\dot{m} + 650300 \tag{5.12}$$

Equation 5.12 is a correlation relating the net power output to extracted steam using simulated data, where $\dot{m}$ is the mass flow rate of extracted 100psia steam in lb/s. Steam extraction occurs at the turbine IP/LP (intermediate/ low pressure) crossover and condensate returns to the deaerator. The power plant model, first created in Steam Pro [3], is exported to Thermoflex [4] to simulate the required modifications for steam extraction to power the carbon capture process.

5.4.3 CO_2 compression chain

Before the CO_2 stream can be transported for sequestration, it first needs to be compressed. The additional cost for this compression chain is included in the cost objective given in Section 5.2.1. CO_2 is compressed using an integral gear centrifugal compressor system—a type of compressor commonly used for CO_2 compression. Due to the relatively low speed of sound in CO_2, the pressure ratio in an integral gear compressor is approximately two. For this work, eight compression stages are used. Intercoolers are not present after the last two stages to avoid liquefaction of CO_2. The compressor model includes a glycol drying system capable of producing CO_2 with a very low water content. An ACM compressor chain model considers operating conditions when estimating the size, pressure ratio, and efficiency of each stage. Efficiency estimates are based on mass flow coefficient correlations [8]. Several constraints on compressor design and operating conditions are also incorporated in the model [70].

5.5 Results

For each of the four reactors studied, we generate a set of sample points and use them to identify sets of surrogate models of relevant response variables. These models are incorporated into the super structure formulation for optimization.

5.5.1 Simulation

Feasible input variable limits for all predictor variables are calculated after considering likely process operating conditions for adsorbers and regenerators. These limits are used both to define the sample and surrogate modeling ranges and to define the feasible space of the super structure formulation. Appropriate bounds are listed in Table 5.1.

While several input variable ranges are consistent for both adsorbers and regenerators, the majority of the operating conditions modeled separately. The most notable changes concern inlet gas properties and sorbent compositions due the the reverse nature of each reactor's purpose. Geometric property ranges remain similar, as do sorbent flow rates because these are recycled.

A total of 400 points are selected from a Latin hypercube design of experiments using the variable ranges provided in Table 5.1. Of the 400 points, 398, 400, 300, and 298 points were simulated successfully with overflow and underflow adsorbers and overflow and underflow regenerators, respectively. Sample points are generated from these simulations for each response variable listed in Table 5.2. The response ranges for each technology and reactor type are shown in Figure 5.5. The primary shifts between the adsorber and regenerator cases (shown in black and red, respectively) reside in the gas CO_2 composition, due to the respective purpose of each reactor, and the resulting temperatures due, in part, to the cooling of the absorber and heating of the regenerator. We observe a more significant response in

Table 5.1: Predictor variable descriptions and limits

Predictor variable	Description	Adsorber [min,max]	Regenerator [min,max]
Db	Reactor diameter, m	[10, 18]	[8, 16]
Lb	Reactor length, m	[2, 8]	no change
Dx	Heat exchanger tube diameter, m	[0.01, 0.04]	no change
lx	Heat exchanger tube length, m	[0.05, 0.55]	[0.05, 0.25]
F_g^{in}	Gas inlet flow rate, kmol/h	[4000, 9000]	[1000, 3000]
T_g^{in}	Gas inlet temperature, °C	[40, 100]	[140, 170]
x_{g,CO_2}^{in}	Gas inlet CO_2 molar composition	[0.02, 0.14]	[0.1, 0.4]
$x_{g,H_2O}^{\text{in}}{}^*$	Gas inlet H_2O molar composition	[0.02, 0.14]	–
P_g^{out}	Gas outlet pressure, bar	[1, 1.4]	[1, 1.3]
F_A^{in}	Sorbent mass flow rate, kg/h	[300000, 900000]	no change
T_A^{in}	Sorbent outlet temperature, °C	[50, 110]	[130, 150]
z_{A,HCO_3}^{in}	Sorbent outlet HCO_3 mass fraction, kmol/kg sorbent	[0, 0.4]	[0.1, 0.5]
$z_{A,HN_2COO}^{\text{in}}$	Sorbent outlet HN_2COO mass fraction, mol/kg sorbent	[0.2, 1.2]	[0.8, 1.8]
z_{A,H_2O}^{in}	Sorbent outlet H_2O mass fraction, mol/kg sorbent	[0.2, 1.2]	[0.3, 1.3]

* Not required to model regenerator simulations

the input variables and corresponding weight fractions in the adsorber compared to the regenerator. Overflow and underflow conditions are depicted in Figure 5.5 with dark and lighter bars. The most notable effects of this shift in flow regime are observed in the number of required heat exchanger tubes, flow rate, and outlet CO_2 composition. It is also worth noting that the heat exchanger outlet temperature is robust to changes in the inputs space for each reactor type and flow regime. The degree of freedom required for heat exchange is the exchanger flow rate for all cases—even the regenerator flow rate ranges an order of magnitude, although this may be difficult to discern in Figure 5.5.

Table 5.2: Response variable descriptions and limits

Response variable	Description
Nx	Number of heat exchanger tubes
Vg	Superficial gas velocity, m/s
$P_{\text{gas}}^{\text{in}}$	Gas inlet pressure, bar
$F_{\text{gas}}^{\text{out}}$	Gas outlet flow rate, kmol/h
$T_{\text{gas}}^{\text{out}}$	Gas outlet temperature, °C
$z_{\text{gas},\ CO_2}^{\text{out}}$	Gas outlet CO_2 molar composition
$z_{\text{gas},\ H_2O}^{\text{out}}$*	Gas outlet H_2O molar composition
F_{hx}	Heat exchanger outlet flow rate, kmol/h
$T_{\text{hx}}^{\text{out}}$	Heat exchanger outlet temperature, °C
$T_{\text{solid}}^{\text{out}}$	Sorbent outlet temperature, °C
$w_{\text{solid},\ HCO_3}^{\text{out}}$	Sorbent outlet HCO_3 weight fraction, mol/kg sorbent
$w_{\text{solid},\ NH_2COO}^{\text{out}}$	Sorbent outlet HN_2COO weight fraction, mol/kg sorbent
$w_{\text{solid},\ H_2O}^{\text{out}}$	Sorbent outlet H_2O weight fraction, mol/kg sorbent

* Values did not require simulation or model generation for regenerator cases

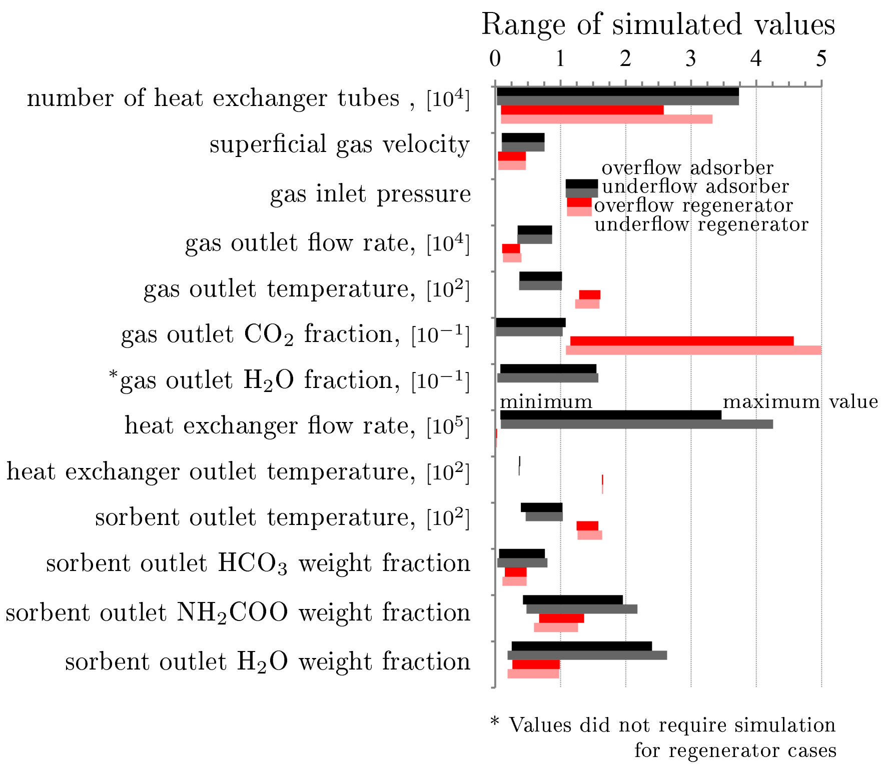


Figure 5.5: Range of simulated values

5.5.2 Surrogate models

Surrogate models are generated be the ALAMO [24] software using the 298-400 points sampled for each flow and reactor type. Functional forms, including monomial powers; multinomial power; and ratios, as permitted as follows.

Monomials:	x_d^α	$d = 1, 2, \ldots, n_{\text{pred}}$	$\alpha = \{\pm 0.5, \pm 1, \pm 2, \pm 3, \pm 4\}$
Multinomials:	$(x_d\, x_{d'})^\alpha$	$d, d' = 1, 2, \ldots, n_{\text{pred}}$ $d \neq d'$	$\alpha = \{-2, -1, -0.5, 0.5, 1, 2\}$
Ratios:	$\left(\dfrac{x_d}{x_{d'}}\right)^\alpha$	$d, d' = 1, 2, \ldots, n_{\text{pred}}$ $d \neq d'$	$\alpha = \{-2, -1, -0.5, 0.5, 1, 2\}.$

Additionally, we include logarithmic, exponential, an constant terms in the basis set. The result is a set of potential basis functions composed of 1,071 terms for the 13-input regenerator simulations and 1,140 for the 14-input adsorber instances.

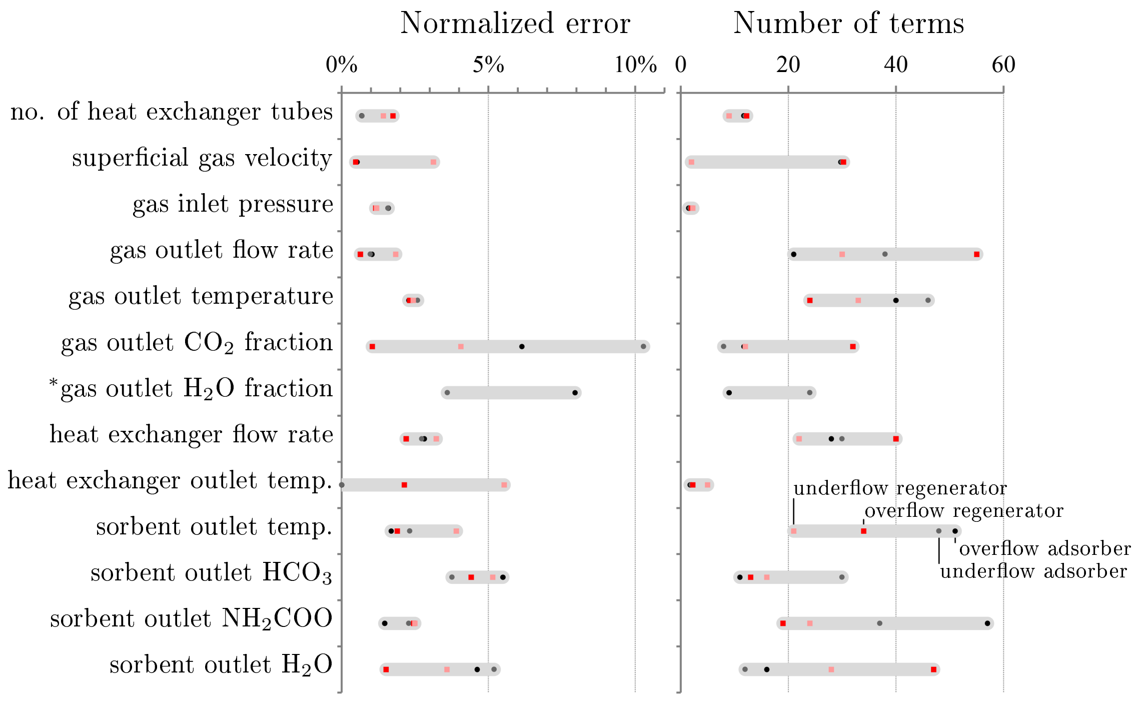
During model generation, we solve Problem (P) for the best-T-subset of terms and allow the number of terms, T, to increase. For larger value of T, we employ a heuristic solution due to increasing problem complexity. We impose a solution time limit on this problem step that may result in either an optimal solution that has not been proven globally or a strong, yet suboptimal solution. To ensure a strong solution, we utilize forward selection to provide an initial guess for (P). This involves choosing a T-term model by adding the best single term to the $T - 1$ model as measured by the objective of (P).

During an initial screening, several outputs are more difficult to model. For these problems, we impose a nonegativity constraint on the output variable, using the techniques described in Chapter 3, to aid in the rapid selection of strong models. For the overflow adsorber, we impose this condition on the superficial gas velocity, outlet gas compositions, and sorbent weight fractions of HCO_3 and water. While the underflow regenerator do not require these constraints, the overflow regenerator and underflow adsorber required the enforcement nonnegativity bounds on the superficial gas velocity, outlet gas compositions, and sorbent weight fractions of HCO_3 and water. For these more challenging problems, the solution time is also relaxed from 300 to 1000 seconds.

In Figure 5.6, we plot the normalized root mean square error between the sampled data and surrogate model as calculate using:

$$\text{Normalized error} = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(z_i - \hat{z}(x))^2}}{\max_i(z_i) - \min_i(z_i)}. \tag{5.13}$$

In addition to quantifying model accuracy, Figure 5.6 includes the number of terms in each response model. Using these results, we can compare several modeling outcomes in terms of



Note: Results with equal number of terms were separated by 0.25 for visualization
* Values did not require simulation for regenerator cases

Figure 5.6: Surrogate modeling results

both ease of modeling and model quality. For our purposes, we consider more parsimonious surrogates easier to model. For example, inlet gas pressure is easy to model since two terms are required for all cases. In contrast, the carbonate weight fraction in the sorbent requires 19-57 terms and is, therefore, more difficult to model. Both of these cases, however, result in high quality models with only 1.1-2.5% normalized error. In contrast, the heat exchanger outlet temperature, which only required 2-5 terms, results in models with a normalized error of up to 5.5%. This is because, as discussed previously, the magnitude of the data range for the heat exchanger outlet temperature is very small and, consequently, relatively insignificant root mean squared errors (RMSE) are inflated deceptively after normalization.

Table 5.3 provides a summary of resulting normalized errors and model sizes for each flow and reactor type. For all generated models, we find a normalized error in the range 0.00319-10.3% and a model size of 2-57 terms, or 0.171-5.14% of the potential basis set size.

Table 5.3: Summary of surrogate modeling results

Reactor type	Flow regime	Normalized error, [%]	Number of terms, [% of original terms]
Adsorber	overflow	2.5 ± 0.4	2.0 ± 0.3
	underflow	3.2 ± 0.6	2.0 ± 0.3
Regenerator	overflow	1.8 ± 0.2	2.4 ± 0.3
	underflow	3.2 ± 0.3	1.6 ± 0.2

For illustration, consider one of the smaller model sets: inlet gas pressure, P_g^{in}. The four generated models are listed with the sampled ranges and RMSE in Table 5.4. All

Table 5.4: Inlet pressure models

Flow regime	Range of P_g^{in}	Error, RMSE	Model, $\hat{P}_g^{\text{in}}$
Adsorbers:			
Overflow	[1.0784,1.5762]	0.00791	$0.5217\exp(P_g^{\text{out}}/1.4) + 0.016063\,Lb\,P_g^{\text{out}}$
Underflow	[1.0786,1.5763]	0.00791	$0.5216\exp(P_g^{\text{out}}/1.4) + 0.016096\,Lb\,P_g^{\text{out}}$
Regenerators:			
Overflow	[1.0996,1.4781]	0.00440	$0.4852\exp(P_g^{\text{out}}/1.4) + 0.018412\,Lb\,P_g^{\text{out}}$
Underflow	[1.0995,1.4783]	0.00451	$0.4851\exp(P_g^{\text{out}}/1.4) + 0.018346\,Lb\,P_g^{\text{out}}$

four models have the same functional form and modeling errors less than 0.00791 or 1.59% normalized error. In each case, the subset selection methods in ALAMO model inlet pressure as a function of only the outlet pressure and the bed depth by reducing the input space to two dimensions from 13 and 14. Furthermore, we can conclude that the pressure responses for both adsorber flow regimes are very similar because the sampled data ranges differ by less than 0.02% and the modeling coefficients differ by less than 0.3%. A similar case can be made for the regenerator flow regimes, with range and coefficient differences within 0.02 and 0.4%.

Moreover, we can infer that the behavior of the inlet pressure for the adsorber and regnerator is, in fact, different as evidenced by a noticeable shift in model coefficients. Over the regenerator ranges of Lb and P_g^{out} (the more restricted of the two pairs of ranges), the inlet pressure is always greater in the adsorber than the regenerator. These important details are not readily observed in the original 13- to 14-dimensional problem space and may not be obvious upon initial inspection of the variable ranges.

A superstructure optimization formulation, composed of the surrogate-based models introduced in this chapter, is used by the National Energy Technology Laboratory (NETL) as part of the CCSI. CCSI was initiated in 2010 to develop and deploy stage-of-the-art simulation, modeling, and optimization tools with the aim of accelerating the development of carbon capture technology [76]. The core goals of the CCSI group are to (1) identify promising concepts more quickly through a rapid computational screening of devices and processes; (2) reduce the time required to design and troubleshoot new devices and processes; (3) quantify the technical risk in taking technology from laboratory-scale to commercial-scale; and (4) stabilize deployment costs by replacing some physical operational testing with virtual power plant simulations. Models developed through our work play a critical role in enabling goal (1), where this supertructure and sets of surrogate models are used to identify, compare, and design promising combinations of carbon capture technologies.

5.6 Conclusion

The use of surrogate models in a super structure formulation enables the simultaneous and efficient screening of several carbon capture technology alternatives. By generating tailored surrogate models using ALAMO, we benefit from the increased accuracy of high fidelity simulations while leveraging advanced discrete optimization solvers. As a result, we enable the comparison of best-case scenarios using accurate representations of process trends. In this chapter, we outline a superstructure formulation used to design industrial carbon capture networks. Moreover, we demonstrate the effective comparison and screening of alternative technologies afforded by the proposed modeling techniques and detail methodologies to simulate and model the underlying processes.

We conclude that generating tailored algebraic surrogate models of black-box processes offers a strong balance between optimization rigor and simulation fidelity.

Chapter 6

Concluding remarks

6.1 Summary of work

In this section, we summarize the main contributions and accomplishments of this thesis.

Learning surrogate models for simulation-based optimization

In Chapter 2, we focus on a methodology to solve problems using high-fidelity simulators by identifying a set of surrogate models that are tailored for algebraic optimization. We present a novel algorithm for surrogate model development. Surrogate models are identified using an integer optimization approach to best subset selection given a large set of potential basis functions. Surrogate models are iteratively interrogated and improved using an adaptive sampling routine: error maximization sampling. The resulting surrogate models are incorporated into a rigorous optimization framework to enable the efficient discovery of optimal solutions.

We find that the proposed algorithms are superior to common machine learning methods in three desired categories: accuracy, parsimony, and efficiency. Improvements test error, model size, and data set size are demonstrated through computational experiments. Finally, the efficacy of integrating the resulting surrogate models into an optimization framework is demonstrated on a real world simulation problem. We conclude that the simple models generated using the proposed method for surrogate-based optimization enable a practical and effective combination of high fidelity simulators and rigorous optimization routines.

Constraining regression problems in the predictor and response variable domains

In Chapter 3, we extend data-driven regression methods to incorporate *a priori* theory-based knowledge. We introduce a class of constraints derived from predictor and response relationships that can be generally incorporated into regression problems. By restricting the regression problem using constraints from first principles and intuition, we simultaneously increase the accuracy and physical feasibility of the resulting surrogate models. The underlying knowledge for these constraints is based upon intuitive bounds on response outputs, thermodynamic limitations, ensuring the consistency of numerical properties, and ubiquitous boundary conditions on differential systems.

We propose a set of semi-infinite constraints to enforce these conditions during regression modeling. These constraints are used to reveal nonintuitive relationships within the regression parameter set to strengthen regression models through increased accuracy, more reliable extrapolation, and physical realizability. We describe several sources and demonstrate several classes of regression restrictions using illustrative examples and computational studies.

A global optimization approach to symbolic regression

In Chapter 4, we expand upon the freedom of nonlinear functional forms introduced in Chapter 2 by exploring a global optimization formulation to solve the symbolic regression problem. We propose a disjunctive optimization formulation to locate and certify global solutions to the symbolic regression problem. By employing a rigorous, deterministic optimization solver, we improve the success rate over current state-of-the-art symbolic regression methods that utilize genetic algorithms and adaptive programming techniques to search for this structure. Our symbolic regression implementation discovers and certifies global solutions that exactly match the model form and all model values.

In addition to a baseline optimization formulation, we propose several improvements to the formulation: redundancy eliminating constraints, symmetry breaking cuts, and depth-based binary variable priorities. We also compare active node minimization and error minimization. The best performance was observed when solving the active node minimization problem formulated using redundancy eliminating constraints imposed among a parent and its children nodes and binary variable priorities based on the tree-depth of each node.

Surrogate-based optimization of carbon capture processes

In Chapter 5, we apply techniques introduced in previous chapters to design and optimize post combustion carbon dioxide capture systems. As a result, we employ both a systems-based approach to process synthesis for optimal design and high-fidelity process simulators that provide an accurate representation of the chemical processes. Surrogate-based optimization combines the modeling methods introduced in previous chapters with a rigorous superstructure optimization formulation.

We outline the components of the superstructure, describe the details of the simulator models, and outline the surrogate modeling methods employed. Lastly, we present results from these methods which are currently used at the National Energy Technology Laboratory to compare and design industrial carbon capture systems.

6.2 Contributions

The major contributions of this thesis are summarized as follows:

Learning surrogate models

- We have developed a novel algorithm for the black-box surrogate modeling of high-fidelity models for incorporation into optimization problems.
- We have automated surrogate model builder to generate large sets of potential basis functions and select the best subset using a mixed-integer optimization problem.

- We proposed an adaptive sampling approach that expands on current methods, that use surrogate model uncertainty metrics to select points, to search the problem space for areas of model mismatch for an arbitrary surrogate model form.
- We establish model accuracy and simplicity as well as modeling efficiency, with respect to data sampling, through computational studies.
- The efficacy of proposed methodology is demonstrated using an optimization framework on a carbon capture case study.

Constrained regression

- We developed a general constrained regression formulation that enforces relationships among predictor and response variables for regression problems with arbitrary functional forms.
- A family of constraints is listed that can be used to enforce physical limitations and *a priori* knowledge on regression problems.
- We demonstrate an implementation of these methods to increase the accuracy, physical realizability, and extrapolation potential of regression models using several illustrative examples and computational experiments.

Rigorous symbolic regression

- We formulate a rigorous optimization model to find global, certifiable solutions to the symbolic regression problem.
- We list improvements to the formulation including alternate objectives, redundancy eliminating constraints, symmetry-breaking cuts, and depth-based binary variable priorities.
- The efficacy of this formulation is demonstrated on several examples from the symbolic regression literature.

Optimization of carbon capture processes

- We outline a superstructure formulation that uses a surrogate-based approach to screen carbon capture technologies for the design of a post combustion carbon capture process.
- We detail surrogate modeling results and a superstructure optimization framework that is currently in use at the National Energy Technology Laboratory for optimization and design.

6.3 Future work

In this section, we outline interesting areas for future research.

Model form identification

In Chapter 2, we use a set of predetermined basis functions to serve as a flexible underlying functional form for the surrogate model. Alternatively, in Chapter 4, we consider a more flexible definition of the nonlinear functional form that comprises a surrogate model. We expect there is great potential in the combination of these two methods.

The subset regression methods of Chapter 2 provide strong models that can be updated using symbolic regression. The subset regression surrogate models would make strong starting points for the symbolic regression problem. Additional information from the subset regression models can be used to help better bound the variables of the symbolic regression problem.

Symbolic regression may be used to enrich the basis set used in subset regression. Promising terms found using symbolic regression could be used to seed the subset regression problem. Furthermore, symbolic regression can aid in closing the gap between the underlying function and the final subset regression model.

Best subset regression

The `ALAMO` implementation described in Chapter 2 involves the solution of the best subset problem and error maximization sampling. However, the majority of the algorithmic time, that is not devoted to solving simulations, is dedicated to solving the best subset regression problem. The optimization approach to this problem is significantly faster and scales more favorably than the enumerative alternative, yet there is still room for improvement.

The current best subset problem is solved by parameterizing with respect to the number of terms allowed in the model. A closer examination of the benefits of simultaneously solving for the model size and model form could provide desirable solution time improvements. Additionally, research into cardinality constrained mixed-integer problems may yield promising formulations.

Alternate sampling methods

We propose an error maximization sampling technique in Chapter 2 to perform an iterative design of experiments aimed at locating areas in the surrogate model with high model mismatch. This method is the primary source of empirical data for the surrogate modeling problem. Then in Chapter 3, we introduce the constrained regression methods that allow for the inclusion of theory-based information into the empirical models. These methods are both aimed at improving a surrogate model by introducing strong information to the modeling problem. However, the two methods work independently.

During the solution of the constrained regression problem, we locate regions in the surrogate model that violate prespecified physical limits: regressors bounds, nonconvexities, thermodynamic limits, etc.. Often, these violated areas are also weakly represented regions of the surrogate model. An exploration into the incorporation of data from this region for error maximization sampling may prove interesting. It may also prove useful to investigate other fixed and iterative sampling methods.

Certified surrogate-based optimization results

The surrogate-based optimization approach discussed in Chapters 2 and 5 involves the (1) surrogate modeling of any black-box aspects of the optimization problem, (2) the incorporation and optimization of an algebraic optimization problem using surrogate models as proxy for black-box segments, and (3) the verification of the solution. In this thesis, we focus on steps (1) and (2). In Chapter 2, we verify the pareto curve of the case study through repeated simulation sampling. However, this method is not always reliable and may be computationally costly.

Sensitivity analyses can be implemented to provide estimates of first- and second-derivative values by repeated sampling of the simulation. Additionally, trust region methods can be used to guarantees convergence to the original optimization problem, provided that we accept moderate assumptions on the surrogate model and sampling data [14]. We expect these methods can provide a stronger alternative to (3).

Bibliography

[1] NETL Power Systems Financial Model Version 5.0. Available at `http://www.netl.doe.gov/business/solicitations/ssc2008/references/PSFM%20User%20Guide.pdf`, 2008.

[2] Aspen Custom Modeler, April 2014. `https://www.aspentech.com/products/aspen-custom-modeler.aspx`.

[3] Conventional steam cycle design program to create cycle heat balance and physical equipment needed to realize it, April 2014. `http://www.thermoflow.com/convsteamcycle_STP.html`.

[4] Fully-flexible design and simulation of conventional steam plants, combined cycles, and other thermal power systems, April 2014. `http://www.thermoflow.com/convsteamcycle_TFX.html`.

[5] H. Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19:716–723, 1974.

[6] A. C. Antoulas, D. C. Sorensen, and S. Gugercin. A survey of model reduction methods for large-scale systems. *Contemporary Mathematics*, 280:193–218, 2001.

[7] J. April, F. Glover, J. P. Kelly, and M. Laguna. Practical introduction to simulation optimization. In *Simulation Conference, 2003. Proceedings of the 2003 Winter*, volume 1, pages 71–78, 2003.

[8] R. H. Aungier. Centrifugal compressors: a strategy for aerodynamic design and analysis. Technical report, ASME Press, 2000.

[9] P. Balasubramaniam and A. V. A. Kumar. Solution of matrix Riccati differential equation for nonlinear singular system using genetic programming. *Genetic Programming and Evolvable Machines*, 10(1):71–89, 2008.

[10] Y. Bard. *Nonlinear Parameter Estimation.* Academic Press, 1974.

[11] K. D. Bettenhausen, P. Marenbach, S. Freyer, H. Rettenmaier, and U. Nieken. Self-organizing structured modelling of a biotechnological fed-batch fermentation by means of genetic programming. In *Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995. GALESIA. First International Conference on (Conf. Publ. No. 414)*, pages 481–486, 1995.

[12] L. T. Biegler, I. E. Grossmann, and A. W. Westerberg. A note on approximation techniques used for process optimization. *Computers & Chemical Engineering*, 9:201–206, 1985.

[13] L. T. Biegler, I. E. Grossmann, and A. W. Westerberg. *Systematic Methods for Chemical Process Design*. Prentice Hall, 1997.

[14] L. T. Biegler, Y. Lang, and W. Lin. Multi-scale optimization for process systems engineering. *Computers & Chemical Engineering*, 60:17–30, 2014.

[15] J. B. Black, J. L. Haslbeck, A. P. Jones, W. L. Lundberg, and V. Shah. Cost and performance of PC and IGCC Plants for a range of carbon dioxide capture. Technical report, DOE/NETL, 2011.

[16] G. E. P. Box, J. S. Hunter, and W. G. Hunter. *Statistics for experimenters: Design, Innovation, and Discovery, 2nd Edition*. Wiley, 2005.

[17] K. P. Burnham and D. R. Anderson. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer, 2002.

[18] J. Caballero and I. E. Grossmann. An algorithm for the use of surrogate models in modular flowsheet optimization. *AIChE Journal*, 54:2633–2650, 2008.

[19] R. Carapellucci and A. Milazzo. Membrane systems for CO2 capture and their integration with gas turbine plants. In *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, volume 217, pages 505–517. SAGE Publications, 2003.

[20] S.-H. Chen. *Genetic Algorithms and Genetic Programming in Computational Finance*. Springer, 2002.

[21] W. Chen, Z. Shao, and J. Qian. Interfacing IPOPT with ASPEN open solvers and CAPE-OPEN. In *10th International Symposium on Process Systems Engineering: Part A*, volume 27, pages 201–206. Elsevier, 2009.

[22] A. R. Conn, K. Scheinberg, and L. N. Vicente. Geometry of interpolation sets in derivative-free optimization. *Mathematical Programming*, 111:141–172, 2008.

[23] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics, 2009.

[24] A. Cozad, N. V. Sahinidis, and D. C. Miller. Learning surrogate models for simulation-based optimization. *AIChE Journal*, 2014. doi: 10.1002/aic.14418, 2014.

[25] N. Cressie. Spatial prediction and ordinary kriging. *Mathematical Geology*, 20:405–421, 1988.

[26] K. Crombecq, E. Laermans, and T. Dhaene. Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling. *European Journal of Operational Research*, 214:683–696, 2011.

[27] K. Crombecq, L. D. Tommasi, D. Gorissen, and T. Dhaene. A Novel Sequential Design Strategy for Global Surrogate Modeling. In *Winter Simulation Conference*, volume 1, pages 731–742, 2009.

[28] E. Davis and M. Ierapetritou. A kriging method for the solution of nonlinear programs with black-box functions. *AIChE Journal*, 53:2001–2012, 2007.

[29] A. Drud, ARKI Consulting and Development. CONOPT 3 Solver Manual. Available at `http://www.gams.com/dd/docs/solvers/conopt.pdf`, 2003.

[30] L. B. Evans, J. F. Boston, H. I. Britt, P. W. Gallier, P. K. Gupta, B. Joseph, V. Mahalec, E. Ng, W. D. Seider, and H. Yagi. ASPEN: An advanced system for process engineering. *Computers & Chemical Engineering*, 3:319–327, 1979.

[31] K. T. Fang and D. K. J. Lin. Uniform experimental designs and their applications in industry. *Handbook of Statistics*, 22:131–170, 2003.

[32] M. C. Fu. Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing*, 14:192–215, 2002.

[33] M. C. Fu, F. W. Glover, and J. April. Simulation optimization: A review, new developments, and applications. In *Simulation Conference, 2005 Proceedings of the Winter*, page 13, 2005.

[34] C. Gatu and E. J. Kontoghiorghes. Parallel algorithms for computing all possible subset regression models using the QR decomposition. *Parallel Computing*, 29:505–521, 2003.

[35] C. Gatu, P. I. Yanev, and E. J. Kontoghiorghes. A graph approach to generate all possible regression submodels. *Computational Statistics*, 52:799–815, 2007.

[36] J.D. Geest, T. Dhaene, N. Fach, and D. D. Zutter. Adaptive CAD-model building algorithm for general planar microwave structures. *IEEE Transactions on Microwave Theory and Techniques. Special Issue on Multilayer Microwave Circuits*, 9:1801–1809, 1999.

[37] D. I. Gibbons and G. C. McDonald. Constrained regression estimates of technology effects on fuel economy. *Journal of quality technology*, 31:235–245, 1999.

[38] P. E. Gill, W. Murray, and M. A. Saunders. User's Guide for SNOPT Version 7: Software for large-scale nonlinear programming. Available at `http://www.sbsi-sol-optimize.com/manuals/SNOPT%20Manual.pdf`, 2008.

[39] M. A. Goberna and M. A. López. Linear semi-infinite programming theory: An updated survey. *European Journal of Operational Research*, 143:390–405, 2002.

[40] D. Gorissen, K. Crombecq, I. Couckuyt, T. Dhaene, and P. Demeester. A surrogate modeling and adaptive sampling toolbox for computer based design. *Journal of Machine Learning Research*, 11:2051–2055, 2010.

[41] I. E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, 3:227–252, 2002.

[42] M. Hassoun. *Fundamentals of Artificial Neural Networks.* MIT Press, Cambridge, MA, 1995.

[43] C. A. Henao and C. T. Maravelias. Surrogate-based superstructure optimization framework. *AIChE Journal*, 57:1216–1232, 2011.

[44] R. Hettich and K. O. Kortanek. Semi-infinite programming: theory, methods, and applications. *SIAM Review*, 35:380–429, 1993.

[45] D. Huang, T. Allen, W. Notz, and N. Zeng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization*, 34:441–466, 2006.

[46] C. M. Hurvich and C. L. Tsai. A corrected Akaike information criterion for vector autoregressive model selection. *Journal of Time Series Analysis*, 14:271–279, 1993.

[47] W. Huyer and A. Neumaier. SNOBFIT—Stable Noisy Optimization by Branch and Fit. *ACM Transactions on Mathematical Software*, 35, 2008.

[48] F. John. Extremum Problems with Inequalities as Subsidiary Condition. In *Studies and Essays*, Courant Anniversary Volume, pages 187–204. Interscience, 1948.

[49] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2001.

[50] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[51] G. G. Judge and T. Takayama. Inequality restrictions in regression analysis. *Journal of the American Statistical Association*, 61:166–181, 1966.

[52] M. A. Keane, J. R. Koza, and J. P. Rice. Finding an impulse response function using genetic programming. In *American Control Conference, IEEE*, volume 1, pages 2345–2350, 1993.

[53] M. Keijzer. Improving symbolic regression with interval arithmetic and linear scaling. In *Genetic Programming*, pages 70–82. Springer, 2003.

[54] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal. Application of genetic programming for multicategory pattern classification. *Evolutionary Computation, IEEE Transactions on*, 4(3):242–258, 2000.

[55] P. S. Knopov and A. S. Korkhin. *Regression Analysis Under A Priori Parameter Restrictions.* Springer, 2011.

[56] A. S. Korkhin. Certain properties of the estimates of the regression parameters under a priori constraint-inequalities. *Cybernetics*, 21:858–870, 1985.

[57] A. S. Korkhin. Parameter estimation accuracy for nonlinear regression with nonlinear constraints. *Cybernetics and Systems Analysis*, 34:663–672, 1998.

[58] A. S. Korkhin. Solution of problems of the nonlinear least-squares method with nonlinear constraints. *Journal of Automatization and Informatic Sciences*, 6:110–120, 1999.

[59] A. S. Korkhin. Estimation accuracy of linear regression parameters with regard to inequality constraints based on a truncated matrix of mean square errors of parameter estimates. *Cybernetics and Systems Analysis*, 38:900–903, 2002.

[60] A. S. Korkhin. Determining sample characteristics and their asymptotic linear regression properties estimated using inequality constraints. *Cybernetics and Systems Analysis*, 41:445–456, 2005.

[61] A. S. Korkhin. Using a priori information in regression analysis. *Cybernetics and Systems Analysis*, 91:41–54, 2013.

[62] M. F. Korns. Accuracy in Symbolic Regression. In *Genetic Programming Theory and Practice IX*, pages 129–151. Springer, 2011.

[63] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[64] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Press, 1994.

[65] D. G. Krige. A statistical approach to some mine valuations and allied problems at the Witwatersrand. *Masters Thesis, University of Witwatersrand,South Africa*, 1951.

[66] D. Kunii and O. Levenspiel. *Fluidization Engineering.* 1991.

[67] A. Lee, D. Mebane, D. J. Fauth, and D. C. Miller. A Model for the Adsorption Kinetics of CO_2 on Amine-Impregnated Mesoporous Sorbents in the Presence of Water. In *28th International Pittsburgh Coal Conference*, volume 1, 2011.

[68] A. Lee and D. C. Miller. A one-dimensional, three region model for a bubbling fluidised bed adsorber. *Ind. Eng. Chem. Res.*, 52:469–484, 2013.

[69] C. K. Liew. Inequality constrained least-squares estimation. *Journal of the American Statistical Association*, 71:746–751, 1976.

[70] K. H. Lüdtke. *Process Centrifugal Compressors: Basics, Function, Operation, Design, Application.* 2004.

[71] J. McDermott, U.-M. O'Reilly, S. Luke, and D. White. Problem classification, April 2014. `http://www.gpbenchmarks.org/wiki/`.

[72] B. McKay, M. Willis, and G. Barton. Steady-state modelling of chemical process systems using genetic programming. *Computers & Chemical Engineering*, 21:981–996, 1997.

[73] B. McKay, M. Willis, D. Searson, and G. Montague. Non-linear continuum regression using genetic programming. *GECCO*, 2:1106–1111, 1999.

[74] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21:239–245, 1979.

[75] A. D. McQuarrie and C. L. Tsai. *Regression and Time Series Model Selection*. World Scientific Publishing Company, 1998.

[76] D. C. Miller, N. V. Sahinidis, A. Cozad, A. Lee, H. Kim, J. Morinelly, J. Eslick, and Z. Yuan. Computational Tools for Accelerating Carbon Capture Process Development. In *The 38th International Technical Conference on Clean Coal & Fuel Systems*, volume 1, 2013.

[77] D. C. Montgomery, E. A. Peck, and G. G. Vining. *Introduction to Linear Regression Analysis*. Wiley, 2012.

[78] A. Neumaier. MINQ–General definite and bound constrained indefinite quadratic programming. Available at `http://www.mat.univie.ac.at/~neum/software/minq/`, 1998.

[79] K. Palmer and M. Realff. Metamodeling approach to optimization of steady-state flowsheet simulations: Model generation. *Chemical Engineering Research and Design*, 80:760–772, 2002.

[80] C. C. Pantelides. SpeedUp–recent advances in process simulation. *Computers & Chemical Engineering*, 12:745–755, 1988.

[81] F. Provost, D. Jensen, and T. Oats. Efficient progressive sampling. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 1, pages 23–32, 1999.

[82] F. Pukelsheim. *Optimal Design of Experiments*. Society for Industrial and Applied Mathematics, 2006.

[83] C. R. Rao. *Linear Statistical Inference and Its Applications*. Wiley, 1965.

[84] R. Reemtsen and S. Görner. Numerical Methods for Semi-Infinite Programming: A Survey. In R. Reemtsen and J. J. Rückmann, editors, *Semi-Infinite Programming*, volume 25 of *Nonconvex Optimization and Its Applications*, pages 195–275. Springer US, 1998.

[85] R. Reemtsen and J. J. Rückmann. *Numerical Methods for Semi-Infinite Programming: A Survey*. Springer, 1998.

[86] G. Rezk. Inequality restrictions in regression analysis. *Journal of Development Economics*, 71:746–751, 1976.

[87] L. M. Rios and N. V. Sahinidis. Derivative-free optimization: A review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56:1247–1293, 2013.

[88] N. V. Sahinidis. BARON, User's Manual. Available at `http://www.gams.com/dd/docs/solvers/baron.pdf`, 2014.

[89] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science*, 324:81–85, 2009.

[90] J. D. Seader, W. D. Seider, A. C. Pauls, and R. R. Hughes. *FLOWTRAN simulation: An introduction.* CACHE, 1977.

[91] W. D. Seider, J. D. Seader, D. R. Lewin, and S. Widagdo. *Product and Process Design Principles: Synthesis, Analysis and Design.* 3rd Ed. Wiley, 2008.

[92] T. W. Simpson, J. Peplinski, P. N. Koch, and J. K. Allen. Metamodels for computer-based engineering design: Survey and recommendations. *Engineering with Computers*, 17:129–150, 2001.

[93] G. F. Smits and M. Kotanchek. Pareto-front exploitation in symbolic regression. In *Genetic programming theory and practice II*, pages 283–299. Springer, 2005.

[94] C. Stewart and M.-A. Hessami. A study of methods of carbon dioxide capture and sequestration—-the sustainability of a photosynthetic bioreactor approach. *Energy Conversion and Management*, 46:403–420, 2005.

[95] P. Stoica and Y. Selén. Model-order selection: A review of information criterion rules. *IEEE Signal Processing Magazine*, 21:36–47, 2004.

[96] M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103:225–249, 2005.

[97] M. Thompson. Some results on the statistical properties of an inequality constraints least squares estimator in a linear model with two regressors. *Journal of Econometrics*, 19:215–231, 1982.

[98] S. K. Thompson. *Sampling.* John Wiley & Sons, Inc., 2002.

[99] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58:267–288, 1996.

[100] R. Turton, R. C. Bailie, W. B. Whiting, and J. A. Shaeiwitz. *Modeling and Simulation in Chemical Engineering.* Pearson Education, 2008.

[101] N. Q. Uy, N. X. Hoai, M. O'Neill, R. I. McKay, and E. Galván-López. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12:91–119, 2011.

[102] S. Vigerske. SCIP, User's Manual. Available at `http://scip.zib.de/doc/html/index.shtml`, 2014.

[103] J. Von zur Gathen and M. Sieveking. A bound on solutions of linear integer equalities and inequalities. *Proceedings of the American Mathematical Society*, 72:155–158, 1978.

[104] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.

[105] G. G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129:370–380, 2007.

[106] A. H. Watson and I. C. Parmee. Identification of fluid systems using genetic programming. In *Proceedings of the Second Online Workshop on Evolutionary Computation*, pages 45–48, 1996.

[107] D. R. White, J. McDermott, M. Castelli, L. Manzoni, B. W. Goldman, G. Kronberger, W. Jaśkowski, U.-M. O'Reilly, and S. Luke. Better GP benchmarks: community survey results and proposals. *Genetic Programming and Evolvable Machines*, 14:3–29, 2013.

[108] M. J. Willis, H. G. Hiden, P. Marenbach, B. McKay, and G. A. Montague. Genetic programming: An introduction and survey of applications. In *IEEE Conference Publications*, volume 1, pages 314–319, 1997.

[109] H. Yang, Z. Xu, M. Fan, R. Gupta, R. B. Slimane, Bland A. E, and I. Wright. Progress in carbon dioxide separation and capture: A review. *Journal of Environmental Sciences*, 20:14–27, 2008.

[110] S. Žaković and B. Rustem. Semi-infinite programming and applications to minimax problems. *Annals of Operations Research*, 124:81–110, 2002.