Making the Most of a Model: Interrogation Methods for Computer Codes

Authors: Adam Stein[a*], Kenneth Redus[b], Paul Fischbeck[a,c]

[a] Department of Engineering and Public Policy, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213, adamstein@cmu.edu
[b] Redus and Associates LLC, 189 Lafayette Drive, Suite C, Oak Ridge, TN USA 37830
[c] Department of Social and Decision Science, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213
*corresponding author

ABSTRACT

Computer code models have become the default tool for analysis in many areas of research and industry. The research for evaluating these models has been primarily focused on theoretical simulation methods and overlooks the application of these methods. Many existing and highly valued computer codes and models do not allow for simulation, uncertainty quantification, or other modern computing capabilities. These computer codes are generally comprised of two categories: those with limited operation by design and older legacy codes. Updating these software codes is not an option in many situations due to time constraints, cost, loss of skills needed to upgrade aging programming languages, lack of access to source code, and other constraints. This paper compiles and evaluates methods to systematically interrogate computer codes, including reduced iteration design of experiments (DoE) methods. It was determined that while several of these methods are routinely used in other fields, they have not been applied to computer code models. This paper discusses the challenges present when evaluating computer codes and offers a decision framework for selecting interrogation methods. An example case study application of a definitive screening design (DSD) to aerosol transport modeling using the Atmospheric Relative Concentrations in Building Wakes (ARCON96) computer code is provided to illustrate the use of the decision framework and application of DoE fractional factorial designs to computer codes.

Keywords: legacy code, design of experiment, uncertainty quantification, computer experiment, computer model, ARCON96

Competing Interest: The authors declare no competing interests exist.

Funding: This research was supported in part by the Pugh fellowship. The supporting organization was not directly involved with this research

1 INTRODUCTION

Computer code models have become the default tool for analysis in many areas of research and industry. Understanding how to effectively explore these models is useful to researchers and code operators alike. Computer code models are commonly used to calculate one result without mechanisms to consider uncertainty or sensitivity, leading to inefficient and incomplete analyses. Further, the application of theoretical statistical methods for evaluating these computer codes and models has been limited.

Proper experiment design is needed to ensure the model input factors are varied in a way that fully interrogates the model. Multiple iterations, or 'runs', are needed for sampling, uncertainty quantification, and reliability analyses. Simulation and similar methods have been the focus of research for this purpose [1]–[3]. These methods are generally applied to complex computer models that have been designed to be compatible with simulation. However, simulation is not compatible with the design and operation of most computer codes and models. Therefore, other analysis methods must be considered to interrogate these computer codes. To address this gap, this paper provides a framework for the selection of an analysis method, challenges that must be considered, and an example case study.

A computer code is defined in this paper as a computer program or software which contains a model representative of a system. A computer code model is the portion of the code that calculates a response output from inputs. The distinction seems obvious but is inconsistently defined in a large portion of the literature. Many computer codes were designed to complete a function, and only that function, as efficiently as possible. Older legacy codes were written when the computational cost was at a higher premium, using less advanced programming languages, necessitating simple design. Despite a significant reduction in computational costs and the development of more advanced computer languages, modern codes are still written with limited functionality. This is done to make it less costly to develop, easier to maintain, or simpler to validate. While these limitations were designed into the code for resource or functional efficiency, they also create challenges for evaluation. A large body of literature exists on operating computer code models, but little has been published on effective and appropriate methods to operate limited or legacy computer codes systematically. We present methods that could be applied to interrogate these common and valuable computer codes, along with a decision framework to select a method that will function with a specific computer code.

One largely unexplored approach that could be used to interrogate computer code models is the design of experiments (DoE). These methods have long been used to characterize physical models and are prevalent in many fields, including scientific studies and industrial quality control. Instead of prescribing parameters and running physical experiments, the process of getting data from most computer models more closely resembles *interrogation*. In this context, interrogation can be thought of as asking the code specific questions. In this paper we propose the application of factorial experiment designs to interrogate an existing model or code where other methods are not well suited.

This paper is organized into four main sections. First, the challenge of utilizing codes that are not designed for multiple simulation use is described. Second, applicable methods and tools are collected and discussed. Third, the application of experimental design to interrogate an existing computer code model is presented. Finally, a case study applying a definitive screening design (DSD) method to the aerosol transport modeling code ARCON96[1] is provided.

2 Current Challenges

There are many challenges to interrogating an existing code. These challenges are often overlooked in the literature and left to the operator to solve. Challenges vary widely based on the characteristics of the code being used, the model contained in the code, and the use case. Seven challenges are discussed in this section and solutions are presented where possible.

2.1 System Design and Opacity

Experiment design is used to structure experimental observations of a system. The observations are often used to make an experimentally determined empirical model. These models are often called extrinsic because they are defined from the external behavior of the system and not the internal function. A mechanistic model is not based on observations, but instead employ an understanding of the underlying phenomena that define the function of the model.

Model opacity refers to the level of knowledge that is available about the model contained in the computer code, as described in Figure 1. An extrinsic model, a mechanistic model, or a combination of the two, might be contained in an “opaque box” system[2]. Gray or clear box systems are where some or all of the internal function of the system is known, respectively. Open-box codes provide access to the code and model directly and usually also provide a way to interface with the code. A spreadsheet-based model is a good example of open-box. If the spreadsheet is later locked it would be an opaque box to a new operator.

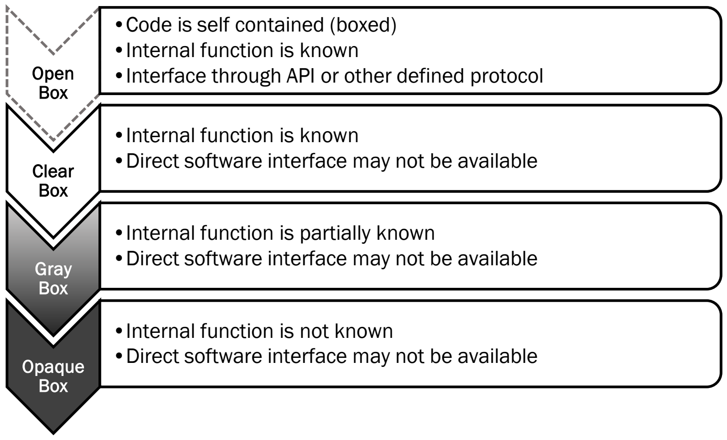
Open Box
• Code is self contained (boxed)
• Internal function is known
• Interface through API or other defined protocol
Clear Box
• Internal function is known
• Direct software interface may not be available
Gray Box
• Internal function is partially known
• Direct software interface may not be available
Opaque Box
• Internal function is not known
• Direct software interface may not be available

Figure 1: Hierarchy of code system knowledge and connectivity

[1] Atmospheric Relative Concentrations in Building Wakes (ARCON96) code is a US Nuclear Regulatory Commission (NRC) code typically used to calculate relative atmospheric concentrations at control room air intakes relative to plumes from hypothetical nuclear power plants accidental releases. As of this writing ARCON96 can be found at https://ramp.nrc-gateway.gov/content/arcon-overview
[2] Sometimes referred to as ‘black box’ and ‘white box’ systems

Connectivity to the model does not affect the function of the model directly but does limit the methods that can be used to interact with the code. The level of opacity is important for selecting the analysis strategy that is best suited to the use case.

2.2 Statistically Valid Results and Uncertainty

Computer models are often representative models for a real-world system. Even with calibration, verification, and validation a computer code rarely models a physical system perfectly because the input data to the system is necessarily a sample of the full data set. The data set used to design, test, and train the model is usually not included with the model. Further, only the most accurately documented models include variance and uncertainty quantification along with methods and formulas.

Opaque and gray box systems make it impossible to know the ability of the model to accurately describe the original system because the internal function and data that was used to define that structure is unknown. When using and interrogating computer models it is important to understand that you are subject to the assumptions, conventions, bias, and uncertainty in the computer model.

Model inputs are another source of uncertainty entering the system. When interrogating a model, it is common to use a range of input values regardless of the interrogation technique used. The input values should be selected carefully to ensure they are feasible values and are based on reasonable and logical assumptions. Selecting values that are too broad (i.e. beyond the range of the model) or too narrow can result in invalid responses. Not all codes have safeguards built in to avoid this situation, so it is prudent to be cautious.

2.3 Operational Design

The operational design of code operation must be considered when selecting statistical experimental design methods. The operational characteristics and both the ability and method of the operator to interface with the code is often overlooked. Codes that have a built-in simulation tool, or are easy to integrate with a separate simulation tool, have the lowest barrier to operator use. Uncertainty can be explored through extensive interrogation with methods discussed in Section 3.2. A single run code, one that is designed to provide a response in one operation, poses limitations. These codes are not easy to use in a batch or simulation manner. The operator interface design may require direct interaction that completely prevents the use of a separate simulation tool.

Many runs are needed even for the smallest experimental design. Single run codes require additional operator time to set up and run each iteration. Operator expertise can reduce the time needed for each run but is unlikely to match the time savings of a simulation tool. The barriers to simulation usually result in ‘normal’ operation as described in Section 3.1. However, these codes are usually good candidates for DoE methods, discussed in 3.4.

There are exceptions, such as finite element analysis models, that take extensive computational time to run and therefore might only be run once. In this case, it is important to quantify and reduce uncertainty as much as possible in the experiment design prior to simulation.

2.4 Code Alteration

Some codes are not designed in a way that allows multiple runs (i.e. ARCON96 in Section 5). It may not be possible to upgrade codes to operate for multiple iterations for a variety of reasons. For instance, the source code could be lost, necessary programming skills are lost, insufficient time or budget is available, and other possible reasons. Third-party codes often cannot be altered.

Simulation tools can be very powerful, but they are only useful when the code can be operated for multiple iterations and is compatible with the simulation tool. Many existing codes were not intended to be run remotely (i.e. by another code) and do not provide API or file access which simulation codes can leverage.

If the code is validated for function, that is it has been certified in some way the code functions and provides the expected response, then altering the code would likely void that validation. If the validation is important to the experiment, to certify the code is performing the function it was intended to perform, then altering the code is not an option.

2.5 Constrained Design Regions

Constrained design regions exist when factors are limited in combinations or ranges. They can exist for categorical or continuous factors. Mixtures are a common example where a formulation contains proportions of ingredients up to 100%, but some may only be possible in a limited range. Physical systems such as pressure vessels that can operate in a limited range of temperature and pressure combinations provide another example.

Care must be taken to only interrogate the model with feasible factor parameters. Two general methods exist to accomplish this need. Input factor scenarios can be carefully selected to avoid constrained design regions or techniques can be used to adapt analysis methods to address a constrained design region. Techniques vary by the analysis method used and should be considered prior to interrogation if a constrained design region exists.

2.6 New Use Cases

New use cases provide an opportunity to extend the value of an existing code without creating a new computer code. In some situations, the existing code is applicable but cannot be used as-is and a new operational method must be developed. Even if a modeler with the necessary experience is available, it may take significant time to design an operational method. One example of a new use case will be illustrated in the case study in Section 5 of this paper.

2.7 Replacement Value

The value of overcoming these challenges should be weighed against the option of replacing the existing computer code model with a more capable computer code. Computer codes require significant time and resources to develop. However, there are often co-benefits associated with developing a new computer code which makes the investment worthwhile, including simplified maintenance, the ability to add features, and uncertainty/sensitivity analysis capabilities. It may not be justifiable to develop a new code, such as when the existing code is used infrequently, there is a short deadline for a project, or no development funding exists. Replacement of gray and opaque box computer code models may not be possible due to insufficient understanding of the underlying foundation. When code verification and validation are necessary there will be additional development time and cost.

3 Methods and Tools

Selection of an interrogation method must consider the characteristics of the computer code and specific challenges, such as those discussed in Section 2. To explore an existing model or code there are five main options.

1. Normal operation
2. Scenario Analysis
3. Simulation tools
4. Design of Experiments
5. Reduced-form models

Development of an analysis plan requires an understanding of the computer model, characteristics and limitations of analysis design options, and the resources available. The goal of the analysis plan should be clearly understood prior to selecting a method that will achieve that goal. This is especially a concern when multiple methods are sufficient to analyze the computer code but may not result in the desired goal. Goals commonly include determining model response in a specified range of inputs but can also be to understand the processes and functions of the computer code model. The decision tool in Figure 2 is provided to guide the selection process. Further discussion of the analysis options is provided in the subsequent sections.

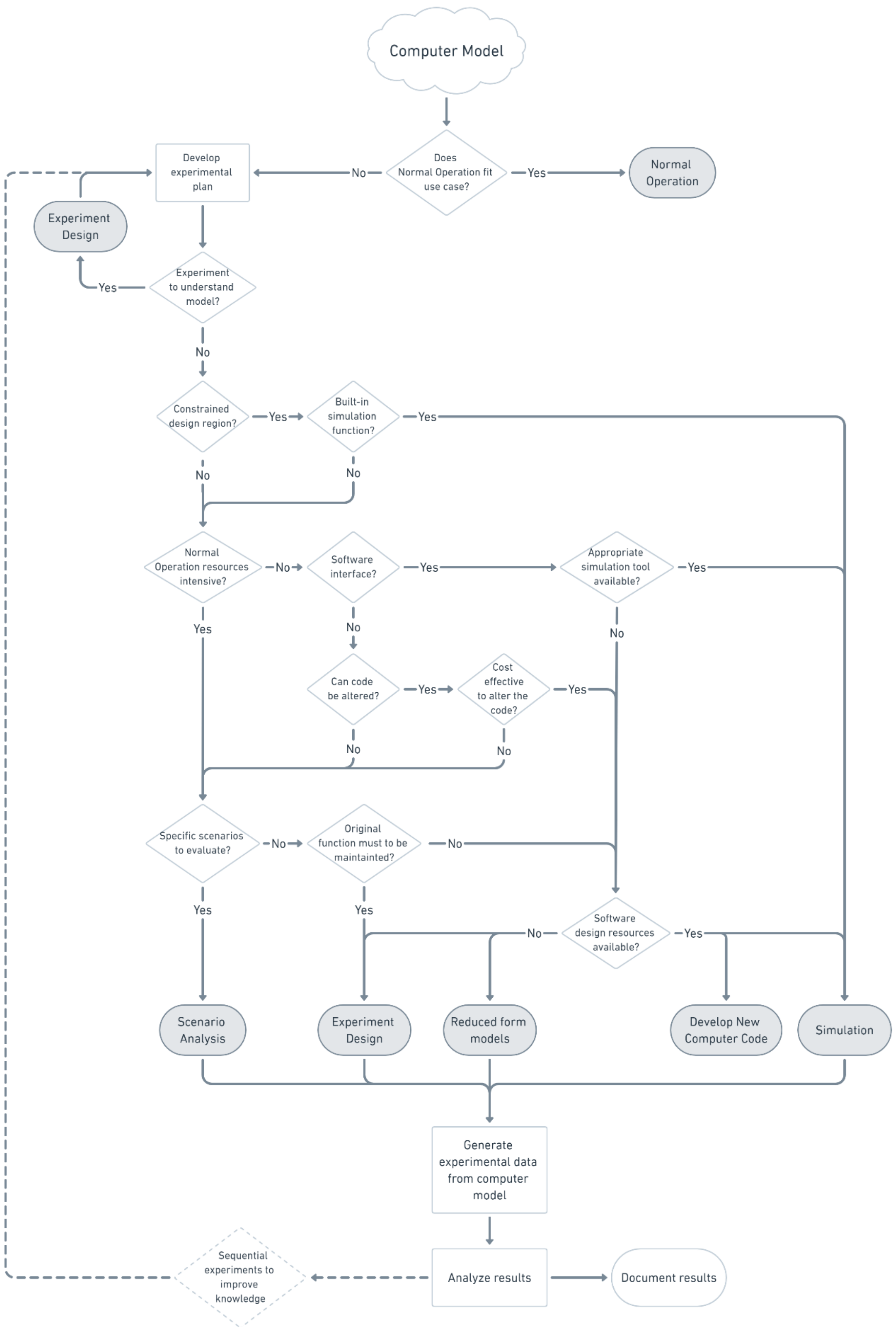
Computer Model
Does Normal Operation fit use case?
Yes
Normal Operation
No
Develop experimental plan
Experiment Design
Experiment to understand model?
Yes
No
Constrained design region?
Yes
Built-in simulation function?
Yes
No
No
Normal Operation resources intensive?
No
Software interface?
Yes
Appropriate simulation tool available?
Yes
Yes
No
No
Can code be altered?
Yes
Cost effective to alter the code?
Yes
No
No
Specific scenarios to evaluate?
No
Original function must to be maintainted?
No
Yes
Yes
Software design resources available?
No
Yes
Scenario Analysis
Experiment Design
Reduced form models
Develop New Computer Code
Simulation
Generate experimental data from computer model
Analyze results
Document results
Sequential experiments to improve knowledge

Figure 2: A decision framework for computer code analysis strategies

The following options assume a few points: The modeler has the skills necessary to operate the model. The methods described herein are to enable augmented use of the existing model. Factor levels can be set independently of the level of any other factor. Models can be deterministic or stochastic, but the input factors can be manipulated in a controlled way. It is assumed that the computer model being interrogated is a valid model or it would not be considered for further use.

3.1 Normal Operation

Normal operation is the usual practice but often limits model operation to an original use case. While the normal operation is common, in many cases it does not involve statistically exploring the code. Many operators simply use best assumption input values and run the code one time, avoiding any sensitivity or uncertainty analysis. This is especially true when uncertainty tools are not built into the model.

Using the model in normal operation avoids the need to invest in learning the new skills required for the other options. Under normal operation cases, the input factors should be well known. Fully exploring the model in normal operation requires a brute force approach to complete many runs, which is inefficient and often not feasible. If normal operation is selected as the appropriate strategy for the experiment, the analysis plan should include a detailed workflow checklist, quality assurance, and data management plan. The purpose is to reduce errors that will be difficult to detect later and to potentially reduce the time needed by streamlining the process.

3.2 Scenario Analysis

Scenario analysis is often called 'what-if' analysis and calculates a model response to understand one possible situation or scenario. This approach is used to make the analysis more tractable when the model is too complex for full interrogation of the design region, or when an understanding of specific scenarios is needed and the remainder of the design region is not of interest. For example, in risk analysis, this approach is commonly used when there are expected modes of failure in a system which can be used as the basis for scenarios. The number of runs needed is defined by the number of scenarios the operator chooses to explore.

Scenario analysis is similar to normal operation in many ways. The code is usually operated in the normal manner and inputs are varied. The analysis plan for a scenario analysis generally includes sets of inputs selected determined to be the best estimates of specific scenarios. Sensitivity analysis is occasionally employed and usually uses an OFAT method. Many DoE methods are also capable of scenario sensitivity analysis if the input boundary values are properly defined. Several of these DoE methods have the advantage of being able to detect factor interaction and higher-order effects.

3.3 Simulation

Simulation[3] allows for semi-automated iteration and interrogation of a computer model across the defined range of inputs. A simulation tool can be used to iterate a model for hundreds or thousands of runs if needed. These tools often provide a more extensive suite of modeling tools

[3] Simulation is commonly referred to as batch processing in industry applications. Batch processing may also refer to a mixture of simulation and scenario analysis

for uncertainty quantification. Consequently, it is desirable to fill the experimental region with points as efficiently as possible (i.e. space-filling designs) so that each run provides additional information and captures variability between input factors and the response. This is a desirable feature if the experimenter does not know the form of the model that is required and believes that interesting phenomena are likely to be found in different regions of the experimental space.

From an experimental design perspective, space-filling designs are often appropriate for deterministic computer models because interrogation points are spread systematically throughout the region of experimentation [2]. Space-filling designs generally do not contain any replicate runs. For a deterministic computer model, this is desirable because a single run of the computer model at an interrogation point provides all of the information about the response at that point. Methods for space-filling designs include Latin Hypercube design [4], [5], spherical packing maximin design [6], Bayesian probabilistic sensitivity [7], and Gaussian process [3].

Simulation can also be a useful interrogation strategy for codes that contain stochastic models which do not return the same response with each run. A Monte Carlo simulation is a common design choice for interrogating stochastic models. In a Monte Carlo model, pseudo-random values are used for interrogation points instead of a structured space-filling design. Replicate runs can be useful in a stochastic model where a different response can occur with the same inputs. Monte Carlo models do not usually limit replicate runs but the occurrence of replication is usually by chance. Some tools provide options to intentionally replicate runs.

Simulation tools can be used in some cases to interface directly with existing models. Some can be used to 'control' existing codes by direct control of input text files or calling commands. When this option is available it can be a powerful way to interrogate a model. Significant effort is usually required to set up a simulation tool to interface with a code, even if that code is open-box. If the computer code is a closed system, it may not be possible to use a simulation tool. The time required to develop the interface, expertise of both the computer code being interrogated and the simulation tool, and limitation to the computer code model platforms that specialized simulation tools can interface should be considered. If a suitable tool is available, the investment may be a net benefit compared to alternative methods. While there are some open simulation tools, the majority are complex commercial products.

3.4 Design of Experiments

Design of Experiments (DoE) methods are used to interrogate a system or model in a systematic and statistically valid way. Many examples exist in the literature including full factorial [8], fractional factorial [9], split-plot [10], Plackett-Burman [11], Box-Behnken [12], optimal designs (D-criterion) [13], definitive screening design (DSD) [14], [15], augmented DSD [16]. Experiment designs can be selected to meet the needs or characteristics of the model. These characteristics may include factor interactions, nonlinearity, and constrained design regions. Tools exist to assist in experiment design using a range of methods.

Separation of control and noise factors is usually an important consideration in experimental design to enable reduction of variance. This is especially important for quality control in manufacturing. To address this, techniques such as randomization, blocking, and replication are often employed. Defined computer codes using deterministic models are expected to consistently provide the same results for a set vector of inputs without noise. Therefore, the extra consideration and techniques used to isolate and characterize the effect of systemic variance or noise are unnecessary with deterministic computer codes. Epistemic variance may still exist if the model contains stochastic factors.

The one factor at a time (OFAT) method involves the selection of a baseline for each factor and then successively varying each factor over its range with the other factors held constant at the baseline level. OFAT is analogous to sensitivity analysis used in many other fields and is very common for modelers to perform when testing model output. Factorial designs can provide more information or evaluate more factors with less or equal computational cost than OFAT experiments, making them more efficient [1]. Factorial designs are able to detect interaction effects, where changing the level of one factor changes the output effect caused by a second factor, something OFAT cannot detect [1].

Experimental design methods often offer a significant reduction in the number of runs needed compared to other approaches. It is attractive to limit the number of runs needed when each run is expensive in terms of resources; cost, time, limited opportunities to test. In the case of computer codes, the expense is usually in run setup or computation time. DSDs are inexpensive to implement, requiring only a minimum of $2m+1$ runs where m represents the number of factors [14]. Reducing workload and expense is beneficial even when the model that is being interrogated is readily available and open.

Screening designs are a family of experiment design methods that are used to identify the main effects (ME) of a system. A testing framework is used to ensure the set of possible factor combinations is explored in a sophisticated manner. Levels for each factor, often two or three levels per factor depending on the design, are selected for analysis. A particular design may be more appropriately matched to the application needed or desired results.

3.5 Reduced Form Model

Reduced form models have also been referred to as surrogate models [17] or metamodels [1]. Reduced form models are used when the existing model is too resource-intensive to operate in a simulation manner and DoE methods will not provide a sufficient understanding of the model response. Despite a body of literature on other applications, reduced-form models of computer codes have significant limitations for limited and legacy computer codes and should be considered as a last resort. This is especially true for validated codes that cannot be readily replaced by a surrogate model.

A reduced-form model involves making a new but simpler metamodel that attempts to provide the same results. To create a reduced-form model, an input dataset paired with output response is needed. Another analysis is used to generate computer model responses and a metamodel is developed from the model response. Factor combinations in the input dataset should be carefully selected to ensure the reduced-form model covers the needed design region of the original model. Methods such as fractional factorial design can be used, but OFAT is often used because the modeler does not have experience with DoE methods.

It is assumed that the computer code being interrogated is a valid model or it would not be considered for further use. Despite this assumption, the computer model is still a representation of the actual system and therefore does not perfectly describe the actual system. Extrapolating a reduced form model from the original computer code model further removes the response from the actual system. Without a complete understanding of the inner working of the code (i.e. opaque box), it is difficult to build a reduced-form model that is representative of the computer code model. If the model is well understood, and a reduced form model is only being considered to allow for simulation operation, it may be a better choice to build a new computer model.

Reduced-form models generally require follow up experiments or validation using existing data. The reduced form model is ideally trained from the same data that the original model was trained on. However, it is not common for the initial training data to be provided with the original model. Many runs (1,000-100,000) may be needed to sufficiently train and test the reduced-form model. The need for this many runs of the computer code to test and train is self-defeating if the goal were to create a less resource-intensive model for simulation. To verify and validate the reduced form model, a new test data set would be required. Without this step there the variance between the computer code model and the reduced form model can be determined but there is no way to know how well the reduced form model describes the original system.

4 APPLICATION OF FACTORIAL DESIGNS

The state of knowledge can be advanced through a sequence of staged experiments: screening, effect estimation, optimization, and mechanistic model. Factorial designs are well suited to screening and effect estimation, and some are useful for optimization. Creating a new mechanistic model is beyond the capabilities of a factorial experiment that is being applied to a surrogate model (i.e. computer code model instead of a physical system). Building a mechanistic model requires a deep understanding of the system operation and phenomena along with data to test and validate, similar to a reduced form model.

Factorial designs are usually employed in a design of experiments program aimed at collecting new data. Their most common use is in the earliest stages of experimentation when a large number of potentially important factors may affect a response of interest, and when the goal is to identify the generally fewer highly influential factors. Screening designs allow for a significant reduction in the number of runs needed compared to a simulation approach. Reducing the number of runs may offer a significant saving in time and cost.

Factorial designs are useful for interrogating legacy or complex computer codes in multiple ways. They allow the codes to be used in a 'non-intrusive' manner, not affecting the code or the results directly. This is especially important for codes that either cannot be modified, cannot be used with third-party simulation tools, or are opaque box third party codes that are not allowed to be changed.

As the number of factors m (input factors) increases the number of runs required increases to a point which makes full factorial designs sometimes impractical and inefficient. The sparsity principle states that most of the variability in a system or process output is due to a small number of inputs. Effect sparsity indicates that the number of active effects compared with active factors is relatively small. For example, for a problem with seven factors (input factors) each at two possible values, a full 2^7 design requires 128 runs, but the sparsity of effects means only a subset of the 7 factors and 21 two-way interactions (7 choose 2 combinations) are likely significant. As such, only a fraction of the complete 128 runs is required to obtain estimates on significant effects. Following this logic, reduced run designs have been developed to be more efficient in terms of design size.

The heredity principle is commonly used when considering model selection, which can be strong or weak. Strong heredity implies that if a model includes a two-factor interaction, then its constituent main effects are included in the model. Robustness to assumptions of model heredity and sparsity is not uniform across all factorial designs [18].

Several advantages are associated with DSD compared to most factorial designs [14]. Main effects are completely independent of two-factor interactions. Two-factor interactions may be correlated but are not completely confounded by other two-factor interactions. Nonlinearity can be detected, and the responsible factors can be identified. Unlike most designs, quadratic effects are estimable, and the responsible factors can be identified. Augmented DSD can be used to include categorical factors [16]. If a classical response surface design is used with a subset of model factors, usually to reduce the number of runs needed, there is a risk of missing other important factors. Conversely, if a screening experiment is used to avoid missing important factors, interactions and quadratic effects will be missed. The DSD approach can be used for simultaneous screening and response surface exploration using quantitative factors with three levels. Using one of these designs with six or more three-level factors allows fitting the full quadratic model involving any subset of three three-level factors.

After a screening design is selected, factors need to be identified and feasible and reasonable levels need to be selected. Screening designs often use two levels for each factor, a high (+) and low (-) value. The “+” or “-“ representation is a DoE shorthand notation, and its key benefit is illustrating many combinations of many levels of many factors. Some screening designs use three or more levels for each factor to allow for the estimation of factor interactions and quadratic effects. Once the screen and levels are selected the factors should be entered into the code and run as specified by the screening design, as depicted in Table 1. Eleven factors are identified. Twelve runs of the code are performed where each run sets the factor value at its high (+) or low (-) value. For example, the first run of the code sets factors A, C, G, H, I, and K at the “high” values of the factor, and it sets factors B, D, E, F, and J at the “low” values of the factor. Contrast this to the 12th run of the code in which all values for factors A, B, C … K is set at the “low” value of each factor.

Level selections should be guided by the specific experimental design. Normally three levels for a factor are selected as the lower bound, midpoint, and upper bound. When there is more insight into the model operation (clear or gray box) alternate levels might produce more useful results. When a factor has a default or normal value that is close to the center it can be beneficial to use that value instead in a low, normal, high configuration. For example, if the voltage in a system has a lower bound of 10V and a high bound of 14V, the center would be 12V. If the system usually operates at 12.6V, which is greater than the midpoint of the range, it can be beneficial to use 12.6V in the analysis. Doing so allows the response to be directly usable by the operator as a ‘normal’ condition without additional analysis. However, some factorial designs require the use of a center value for statistical validity.

Table 1: An example 12-run Plackett-Burman Design for 11 factors. Input factors are alternated low (-) or high (+) systematically.

		Factors (*m*)										
		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**
Run	**1**	+	-	+	-	-	-	+	+	+	-	+
	2	+	+	-	+	-	-	-	+	+	+	-
	3	-	+	+	-	+	-	-	-	+	+	+
	4	+	-	+	+	-	+	-	-	-	+	+
	5	+	+	-	+	+	-	+	-	-	-	+
	6	+	+	+	-	+	+	-	+	-	-	-
	7	-	+	+	+	-	+	+	-	+	-	-
	8	-	-	+	+	+	-	+	+	-	+	-
	9	-	-	-	+	+	+	-	+	+	-	+
	10	+	-	-	-	+	+	+	-	+	+	-
	11	-	+	-	-	-	+	+	+	-	+	+
	12	-	-	-	-	-	-	-	-	-	-	-

Of keen interest in using DoE on codes is to select a design that incorporates as many key factors as possible. Although high statistical power is always required when stating that a factor indeed has a statistically significant effect on the response, Santos et. al. noted that statistical power does not prove a coefficient for a factor is properly estimated by the model generated by DoE [19]. Therefore, it is more important to select a design to match the factor than to have high statistical power. When using a DSD, if additional statistical power is desired beyond the minimum number of runs additional dummy factors can be added and then extra columns removed.

Blocking is a DoE technique used to control for variability from known and controllable factors [1]. Batches of raw material that may control some variation is an example of a controllable factor. Blocking is not necessary with a computer code model that is not stochastic and is therefore expected to provide reproducible consistent results.

Results can be used to explore intermediate values that were not used in the factor parameters. Experimental plans must be modified if a previously unknown but highly influential factor is discovered, the chosen range for one or more factors is infeasible, or factors cannot be varied independently. A screening experiment may not be able to lead directly to the desired state of knowledge. In these situations, the sequential experimentation strategy is sometimes abandoned in favor of a single design or set of experiments. The case study in section 5 provides an example of this situation.

5 Case Study: ARCON96

ARCON96 is a computer code used to calculate atmospheric relative concentrations (X/Q)[4] in building wakes [20]. The primary use of ARCON96 is in support of control room habitability calculations for nuclear power plants. The code is designed to input site parameters along with a year of continuous weather data for that same site. ARCON96 uses the weather data to calculate probabilistic X/Q atmospheric concentration values for that site and situation. A complex evidence-based model is behind that calculation. Documentation is available on the basic operation and inputs without underlying model flow diagrams, data, or a full set of equations making this a gray box code.

The case study analysis plan and code characteristics exemplify many of the challenges discussed in this paper. ARCON96 is a legacy code with operational capabilities that were limited by design. The internal operation of the code and model is known to some extent, which makes this a gray box code (Section 2.1). The documentation indicates that the model does a statistical analysis of input factors based on physical experiment data. Quantification of the uncertainty and variance between the physical data and the model is not provided (Section 2.2). The code operation is limited in function and ability to interface with other codes (Section 2.3). Code alteration is not possible due to the code not being available (Section 2.4). Further, the code is validated and widely used for nuclear facility regulation and licensing. Alteration of the code would require re-validation. A constrained design region may exist (Section 2.5). Finally, a new use case for ARCON96 is presented to address the changing needs of nuclear facility licensing without the need to alter the code or develop a new computer code (Section 2.6). Following the analysis method selection framework (Figure 2) an experimental design was selected for this case study, specifically a DSD design.

5.1 Different Use Case

The purpose of this case study is to use the already validated and unmodified code for an alternative use case: calculating X/Q values for many locations relative to the dispersion source. The X/Q values are a necessary component for defining the Low Population Zone and Exclusion Area Boundary for nuclear power plant siting. Typically, this process is completed with a different computer code called PAVAN. However, limitations in PAVAN do not allow for calculation near building wakes or near-field (very close) to the dispersion source. The ARCON96 code is designed to operate with these constraints, but not to consider a range of distances between the source and receptor. Therefore, an experiment must be designed to interrogate the ARCON96 model in a way that provides the needed response for this alternative use case. Analysis of the full design region and internal operation of ARCON96 is beyond the analysis plan for the presented use case and left for future work.

The model contained in ARCON96 is designed to use empirical and representative weather data to calculate probabilistic X/Q values for a point-to-point dispersion (e.g. source to control room

[4] The X/Q is defined to be the relative atmospheric concentration at a receptor location per unit release rate of the material at a release location upwind of the receptor. For material given in terms of Ci, for example, the X/Q has units of *Ci/m³* per *Ci/s*. This is normally condensed to *s/m³* with the units associated with the material (Ci, g, mg, etc.) canceling and, therefore, being completely arbitrary.

intake). The model inside ARCON96 can be leveraged to compute X/Q values for a range of distances, weather patterns, and other site characteristics.

5.2 OPERATION

ARCON96 is designed for single run operation. Input factors are considered fixed, rather than random effects. Stochastic inputs are not used and outputs are deterministic and repeatable based on the inputs. ARCON96 was developed in 1996 and does not reflect contemporary software engineering. A user interface exists but is not operable with a modern 64-bit operating system, therefore operation is performed through the command line. Input consists of complex text files with a very specific format, and therefore error-prone and labor-intensive. Direct text file manipulation for inputs and output makes it easy to overlook a small detail and overwrite prior runs. This is not a large concern for the normal use case of one run, but the risk of error and effort is much higher if a large number of runs are needed. Experience from this case study suggests an experienced operator could create approximately one input file every 5-7 minutes. Once set up, ARCON96 completes an iteration of the code in less than a minute. An additional 3 minutes is needed to manually copy results from the output files and place them into an appropriate analysis tool, for a total of approximately 9-11 minutes per run.

Attempting to create a reduced form model (i.e. replacing the code with a representative model) that approximates ARCON96 would be a major undertaking. The ARCON96 code is managed under strict regulatory guidelines and not available for augmentation. The potential is further limited by the lack of access to the full experimental data sets used to build the original model. In this circumstance, the largest barrier to augmenting or replacing the existing model is regulatory acceptability. ARCON96 is a validated code that has been accepted for regulatory use. Using the code in an unaltered state allows the operator to avoid the significant delay and, depending on the situation, a significant cost for validation of a new model. DoE methods enable the use of ARCON96 in an unaltered state while significantly reducing the number of runs and therefore time and cost needed.

5.3 EXPERIMENT DESIGN

This case study utilizes DSD to interrogate the ARCON96 code. Factor parameters are defined to match the previously described use case and described in detail in the SI [21]. Simplifying assumptions are used in some cases to give bounding values, such as defining the weather files so that the wind always blows at the receptor. A constrained design region has been avoided through an understanding of the use case and proper and feasible parameter selection.

Normal operation of ARCON96 uses weather data (stability class, wind speed, wind direction) for a specific site. This data includes a natural variation that inhibits the ability of the experiment design to find the impact on system response from these factors. If significant factor interactions are present the weather variation may skew their estimated effects as well. To avoid that conflict, these factors are controlled as inputs. Although a full year of a single stability class, wind speed, and wind direction aimed directly at a receptor is unrealistic, it allows for control of the model to the extent needed for the experimental design. The response is expected to result in conservative values (higher) for all runs.

Table 2: Experiment design and results for ARCON96 model interrogation study

		Factor								Response (s/m^3)		
		A	B	C	D	E	F	G	H	X/Q 0-2 hours	X/Q 2-8 hours	X/Q 8-24 hours
Run	1	0	-	+	+	-	+	+	+	8.28E-05	8.28E-05	5.28E-05
	2	0	+	-	-	+	-	-	-	5.22E-03	5.22E-03	3.33E-03
	3	-	0	-	+	+	+	+	-	2.50E-05	2.50E-05	1.60E-05
	4	+	0	+	-	-	-	-	+	1.09E-02	1.09E-02	5.87E-03
	5	-	-	0	+	+	-	-	+	4.35E-05	4.35E-05	2.77E-05
	6	+	+	0	-	-	+	+	-	7.55E-03	7.55E-03	4.06E-03
	7	+	-	+	0	+	+	-	-	3.96E-04	3.96E-04	2.13E-04
	8	-	+	-	0	-	-	+	+	6.89E-05	6.89E-05	4.39E-05
	9	-	-	+	-	0	-	+	-	7.55E-03	7.55E-03	4.82E-03
	10	+	+	-	+	0	+	-	+	5.73E-05	5.73E-05	3.65E-05
	11	+	-	-	-	+	0	+	+	9.96E-03	9.96E-03	5.35E-03
	12	-	+	+	+	-	0	-	-	1.20E-05	1.20E-05	7.64E-06
	13	-	+	+	-	+	+	0	+	1.31E-03	1.31E-03	8.37E-04
	14	+	-	-	+	-	-	0	-	9.96E-05	9.96E-05	6.35E-05
	15	+	+	+	+	+	-	+	0	4.77E-05	4.77E-05	3.04E-05
	16	-	-	-	-	-	+	-	0	5.73E-03	5.73E-03	3.08E-03
	17	0	0	0	0	0	0	0	0	4.77E-04	4.77E-04	3.04E-04

DSD uses three levels, usually low, middle, and high values, to provide estimates of main effects as well as two-factor interactions and quadratic effects while remaining unbiased by second-order effects. Only one more than twice as many runs as there are factors (*2m+1*) are required and confounding of any pair of second-order effects is avoided. Designs having six factors or more allow for the estimation of the full quadratic model in any three factors. The resulting design includes 8 factors with 3 levels each. A full factorial design with three levels would require 3^8=6,561 runs. If more than three levels are needed per factor the number of runs needed increases rapidly. The DSD described in

Table 2 reduces the needed runs to 17.

Extrapolating the estimated time to complete a run in the prior section, a full factorial design would take 985-1200 hours and DSD would take roughly 2.5 hours. This assumes an experienced operator and no errors. When a large number of runs are required (e.g. full factorial) simulation may be used to automate the process and reduce labor time. The structure of input and output text files is compatible with some simulation tools. However, the relatively low time needed for the DSD indicates a setup of a simulation tool for this use is not resource-efficient; it would take longer to set up the simulation than would be saved by eliminating the operator.

Blocking techniques are possible with a DSD but are not needed for this experiment. The ARCON96 model provides deterministic and repeatable results without the kind of variation where a blocking technique would be required.

5.4 Results

System response is provided for three separate time intervals shown in

Table 2. Values for 0-2 hours and 2-8 hours are identical. This indicates that the experimental system response is invariable for these time periods across a wide range of factors. It should not be assumed however that the model itself is invariable without an explicit understanding of the model (i.e. clear or open box). The response may be very different if one of the factors that were fixed for this use case is altered.

A forward stepwise regression approach is recommended for the analysis of DSD experiment responses [14]. Results of the stepwise regression for the 8-24 hour time period were calculated using the R programming language and the Design and Analysis of Experiments with R (daewr) package [22], and are shown in Table 3.

Table 3: Stepwise regression results for the experimental model for 8-24 hour time period based on runs 1-17

Factor	Estimate	Std. Error	t value	P_r (>|t|)
(Intercept)	3.18E-04	5.31E-05	5.988	3.91E-03
A	4.85E-04	1.66E-05	29.26	8.12E-06
E	-2.41E-04	1.66E-05	-14.536	1.30E-04
A:E	1.67E-04	2.70E-05	6.175	3.49E-03
B	-3.76E-04	1.66E-05	-22.67	2.24E-05
G	1.29E-04	1.66E-05	7.791	1.46E-03
B:G	-1.37E-04	2.12E-05	-6.488	2.91E-03
D	-1.94E-03	1.66E-05	-116.815	3.22E-08
D:G	-3.97E-04	2.08E-05	-19.105	4.42E-05
B^2	-1.56E-03	5.35E-05	-29.143	8.25E-06
G^2	1.33E-03	4.71E-05	28.259	9.33E-06
D^2	1.85E-03	5.85E-05	31.638	5.95E-06
F	-4.21E-04	1.66E-05	-25.378	1.43E-05
Residual standard error: 6.203e-05 on 4 degrees of freedom				
Multiple R-squared: 0.9998			Adjusted R-squared: 0.9992	
F-statistic: 1681 on 12 and 4 DF			p-value: 8.249e-07	

Suppose a simple OFAT sensitivity analysis was used to evaluate the model in a normal operation manner. The results would have suggested the X/Q could be minimized by increasing receptor height (factor E). However, this method would not have discovered the positive interaction effects with stability class (factor A).

Similarly, distance to the receptor (factor D) is an order of magnitude larger in absolute terms, exhibits a second-order response, and interacts with several other factors. The alternative use case presented here is to determine the X/Q values for a range of distances. ARCON96 is designed to evaluate X/Q values for a single factor configuration. The finding of factor interaction and non-linear higher-order effects is important to the alternative use case. This example clearly illustrates how a simple OFAT approach would not have provided sufficient insight into the model response caused by the interaction of factors and higher-order effects.

5.5 ADDITIONAL RUNS TO ADDRESS CHALLENGES

Additional runs can be used to further explore the model response. Stability classes represent binned ranges of temperature differentials and are identified A through G. The alpha input is not continuous, although the underlying values built into the model are numeric. Therefore, the regression is valid but not useful for modeling, as a categorical value cannot be used to calculate a result from a regression model. A surrogate numerical value (e.g. 1-7) also cannot be used because it may not match the underlying values in the opaque box model. Therefore, the best that can be achieved is to verify the results using additional runs (run 18 & 19) as shown in Table 4. The additional response values confirm the results in Table 3, indicating that stability class G (i.e. extremely stable) atmospheric conditions result in higher concentrations at the receptor. That result is in line with intuition. If the air is stable it is less likely that particles will be scattered and more likely that they will flow directly forward to the receptor. Now that the result is verified it can be used at least to bound the max and min combinations, which satisfies the needs of this use case.

Table 4: Additional runs 18 & 19 to check direction and magnitude of response due to stability class (factor A). Runs 20 & 21 used as upper and lower bounds.

		Factor								Response (s/m^3)		
		A	B	C	D	E	F	G	H	X/Q 0-2 hours	X/Q 2-8 hours	X/Q 8-24 hours
Runs	18	-	0	0	0	0	0	0	0	1.58E-04	1.58E-04	1.01E-04
	19	+	0	0	0	0	0	0	0	8.28E-04	8.28E-04	5.28E-04
	20	+	-	-	-	-	-	+	-	1.20E-02	1.20E-02	7.64E-03
	21	-	+	+	+	+	+	-	+	1.20E-05	1.20E-05	6.44E-06

Upper and lower estimate scenarios (run 20 & 21) were based on the sign of each factor in Table 3. The runs result in higher and lower responses as expected. It is important to note that the responses were very similar to runs 4 & 12 respectively which have very different scenario designs. This finding adds further support to the complexity of the system that has already been shown through the stepwise regression analysis.

At this point, there is sufficient understanding of the model for operators that have weather data for a specific site or only need very conservative results. If a regression model is needed as a tool to estimate the response, then multiple experiments are needed.

5.6 Estimation Experiment Sets

Completing additional experiments is dependent on operator needs and outside the scope of this paper. Instead of providing direct results, a roadmap is presented for future use.

If site weather is available only one additional experiment is needed. If real data is not available, it may be possible to create representative data files based on nearby weather stations or regional averages. The weather file removes two factors from the input. The resulting 6 factor DSD design (Table 6) will provide a predictive model using 13 runs. Knowledge of the site surface roughness and building area can further limit the number of factors to 4, requiring only 9 runs.

Table 6: DSD design for 6 factors

		Factor					
		A	B	C	D	E	F
Run	1	0	+	-	-	-	-
	2	0	-	+	+	+	+
	3	+	0	-	+	+	-
	4	-	0	+	-	-	+
	5	-	-	0	+	-	-
	6	+	+	0	-	+	+
	7	-	+	+	0	+	-
	8	+	-	-	0	-	+
	9	+	-	+	-	0	-
	10	-	+	-	+	0	+
	11	+	+	+	+	-	0
	12	-	-	-	-	+	0
	13	0	0	0	0	0	0

Table 5: DSD Design for 7 factors.

		Factor						
		A	B	C	D	E	F	G
Run	1	0	+	-	+	-	+	-
	2	0	-	+	-	+	-	+
	3	-	0	+	-	+	+	-
	4	+	0	-	+	-	-	+
	5	+	-	0	+	+	+	+
	6	-	+	0	-	-	-	-
	7	+	-	-	0	+	-	-
	8	-	+	+	0	-	+	+
	9	-	-	+	+	0	-	-
	10	+	+	-	-	0	+	+
	11	-	+	-	+	+	0	+
	12	+	-	+	-	-	0	-
	13	+	+	+	+	+	-	0
	14	-	-	-	-	-	+	0
	15	0	0	0	0	0	0	0

If the weather data approach is not possible then a different approach is needed to avoid the obstacle presented by the stability class factor. In the case where only conservative results are needed then stability class can be fixed as 'G' in all input files and a 7 factor DSD (Table 5) can be used.

Multiple experiments can be used to create models for each stability class if needed, at a cost of 15 runs per stability class. Time to complete multiple experiments may be reduced in the case of ARCON96 because the text input files could be copied and reused with only the stability class and output file name requiring alteration. Once these models are created and runs completed, they will not need to be re-run unless factor levels fall outside of the range used in the DSD.

5.7 Discussion

Using the ARCON96 model for an alternative use case is possible but requires careful planning. The DSD approach provides reliable results using a significantly reduced number of runs. Additional runs can be used either for confirmation or a deeper understanding of the model.

The DSD analysis provides insight into the complex inner working of the ARCON96 model that are not provided in the documentation and are not apparent during model use. When ARCON96 is used for the originally designed use case the majority of factors are fixed in value and therefore limit concern that results that are not representative of the true response. The major exception is weather data (e.g. stability class, wind speed, wind direction) which vary throughout the year. This data is usually provided directly to the model which removes these factors as user inputs altogether.

Techniques can be used to address categorical factors with a continuous response as is the case with stability class. However, the presented technique limits the use of stepwise regression results. Now that the ARCON96 is better understood, the stability class factor can either be fixed to represent a conservative scenario or replaced with real weather data relevant to the site. The former can provide a stepwise regression model that can be used for prediction and the latter provides a more realistic response for a single location.

6 CONCLUSIONS

Interrogation of computer codes is a complex process and multiple methods exist. Defaulting to normal operation or one factor at a time analysis can provide an insufficient understanding of the model. Selections of a method are based on operational characteristics, resources, time, and the required outcome. A decision framework was developed to guide analysis method selection based on code characteristics and challenges which may be present.

The application of experimental designs to computer code interrogation provides several benefits. DoE can be used to effectively replace normal operation and provide enhanced results. A more complete understanding of the model response can be achieved with a systematic design. Typically, experimental designs are used to limit the number of experiment runs required to interrogate a model and estimate main effects, factor interaction, and higher-order effects. This approach is useful for legacy or complex codes, particularly ones that are not upgradeable for a variety of reasons. In some situations, the code can be extended to new use cases without changing the code. This is also important for validated codes that cannot be altered.

A novel application of experimental design to interrogate an existing model or computer code was presented in the form of a case study using the code ARCON96. This code was designed more than 20 years ago. When the experiment design is carefully designed it enables the use of an existing code for alternative use cases. While the operation of the code has several limitations, the underlying complex model is still valuable. Many of these limitations were addressed and overcome through experimental design techniques while maintaining the integrity of code validation.

REFERENCES

[1] D. C. Montgomery, *Design and analysis of experiments*, Eighth edition. Hoboken, NJ: John Wiley & Sons, Inc, 2013.

[2] T. J. Santner, B. J. Williams, and W. I. Notz, *The Design and Analysis of Computer Experiments*. New York, NY: Springer New York, 2018.

[3] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and Analysis of Computer Experiments," *Stat. Sci.*, vol. 4, no. 4, pp. 409–423, Nov. 1989, doi: 10/fn276g.

[4] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, vol. 21, no. 2, pp. 239–245, May 1979, doi: 10/gfzncd.

[5] Z. Wu, D. Wang, P. N. Okolo, K. Zhao, and W. Zhang, "Efficient space-filling and near-orthogonality sequential Latin hypercube for computer experiments," *Comput. Methods Appl. Mech. Eng.*, vol. 324, pp. 348–365, Sep. 2017, doi: 10/gbt7h8.

[6] L. Pronzato and W. G. Müller, "Design of computer experiments: space filling and beyond," *Stat. Comput.*, vol. 22, no. 3, pp. 681–701, May 2012, doi: 10/bmgpcq.

[7] J. E. Oakley and A. O'Hagan, "Probabilistic sensitivity analysis of complex models: a Bayesian approach," *J. R. Stat. Soc. Ser. B Stat. Methodol.*, vol. 66, no. 3, pp. 751–769, 2004, doi: 10/fh9mqk.

[8] S. R. A. Fisher, *The Design of Experiments*, 3rd ed. Edinburgh: Oliver and Boyd, 1942.

[9] D. J. Finney, "The Fractional Replication of Factorial Arrangements," *Ann. Eugen.*, vol. 12, no. 1, pp. 291–301, 1943, doi: 10/fqpn2r.

[10] B. Jones and C. J. Nachtsheim, "Split-Plot Designs: What, Why, and How," *J. Qual. Technol.*, vol. 41, no. 4, pp. 340–361, Oct. 2009, doi: 10/ggz95w.

[11] R. L. Plackett and J. P. Burman, "The Design of Optimum Multifactorial Experiments," *Biometrika*, vol. 33, no. 4, pp. 305–325, Jun. 1946, doi: 10/fmhkrj.

[12] G. E. P. Box and K. B. Wilson, "On the Experimental Attainment of Optimum Conditions," *J. R. Stat. Soc. Ser. B Methodol.*, vol. 13, no. 1, pp. 1–45, 1951.

[13] A. C. Atkinson and A. N. Donev, "The Construction of Exact D-Optimum Experimental Designs with Application to Blocking Response Surface Designs," *Biometrika*, vol. 76, no. 3, pp. 515–526, 1989, doi: 10/c66dhx.

[14] B. Jones and C. J. Nachtsheim, "A Class of Three-Level Designs for Definitive Screening in the Presence of Second-Order Effects," *J. Qual. Technol.*, vol. 43, no. 1, pp. 1–15, Jan. 2011, doi: 10/ggxpt6.

[15] B. Jones and C. J. Nachtsheim, "Effective Design-Based Model Selection for Definitive Screening Designs," *Technometrics*, vol. 59, no. 3, pp. 319–329, 2017, doi: 10.1080/00401706.2016.1234979.

[16] B. Jones and C. J. Nachtsheim, "Definitive Screening Designs with Added Two-Level Categorical Factors," *J. Qual. Technol.*, vol. 45, no. 2, 2013, doi: 10.1080/00224065.2013.11917921.

[17] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. Kevin Tucker, "Surrogate-based analysis and optimization," *Prog. Aerosp. Sci.*, vol. 41, no. 1, pp. 1–28, Jan. 2005, doi: 10/b2hpf3.

[18] S. Dougherty, J. R. Simpson, R. R. Hill, J. J. Pignatiello, and E. D. White, "Effect of Heredity and Sparsity on Second-Order Screening Design Performance," *Qual. Reliab. Eng. Int.*, vol. 31, no. 3, pp. 355–368, 2015, doi: 10/f657ng.

[19] C. P. Santos, T. J. Rato, and M. S. Reis, "Design of Experiments: A comparison study from the non-expert user's perspective," *J. Chemom.*, vol. 33, no. 1, p. e3087, 2019, doi: 10/ggbszp.

[20] J. V. Ramsdell, Jr. and C. A Simonen, “Atmospheric Relative Concentrations in Building Wakes,” Pacific Northwest National Laboratory, NUREG/CR-6331, 1997. Accessed: May 30, 2020. [Online]. Available: https://www.nrc.gov/docs/ML1721/ML17213A190.pdf.
[21] Adam Stein, “Dataset - Interrogation of ARCON96 using a Definitive Screening Design,” vol. 1, Oct. 2020, doi: 10.17632/4fcstkrrxm.1.
[22] J. Lawson, *Design and analysis of experiments with R*. Boca Raton: CRC Press, Taylor & Francis Group, 2014.