# Role of experimental data in validating and quantifying uncertainties in complex physical systems

by

James Robert Oreluk

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

and the Designated Emphasis

in

Computational and Data Science and Engineering

in the

Graduate Division

of the

University of California, Berkeley

Spring 2019

**Role of experimental data in validating and quantifying uncertainties in complex physical systems**

## Abstract

Role of experimental data in validating and quantifying uncertainties in complex physical systems

by

James Robert Oreluk

Doctor of Philosophy in Engineering - Mechanical Engineering

with Designated Emphasis
in
Computational and Data Science and Engineering

University of California, Berkeley

Professor Michael Frenklach, Co-chair

Professor Andrew Packard, Co-chair

Model validation is the process of evaluating how well a computational model represents reality. That is to say, does the model make predictions that adequately agree with the experimental evidence? Both model validation and uncertainty quantification have gained tremendous attention from researchers in engineering, physics, chemistry, and biology. Uncertainty quantification methods have been successfully applied to assessing model predictions of unmeasured quantities of interest and assisting in the development of computationally efficient, yet predictive, reduced-order models. In both cases, experimental data are incorporated into the analysis to refine the uncertainty estimate. However, with the amount of experimental data published and being generated through ongoing scientific endeavors, it is crucial to organize and integrate experimental data with the uncertainty quantification methods.

In this work, I develop tools for uncertainty quantification and construct a validation workflow that seamlessly integrates uncertainty quantification tools with an online database of chemical kinetics validation data. The first part of this dissertation discusses the need for structured experimental data, emphasizing its value towards model validation, and explore how online databases provide structure to data. An optimization-based framework for uncertainty quantification, *Bound-to-Bound Data Collaboration*, is employed throughout the dissertation to verify the compatibility of models with data. A novel strategy for surrogate modeling using Bound-to-Bound Data Collaboration is developed to guide the fitting procedure towards regions of the parameter space where the model predicts the data accurately. This technique is demonstrated in two simple examples and a solid-fuel combustion example. In the second part of this dissertation, three complex physics-based models are investigated, specifically $H_2/O_2$ combustion, a solid-fuel char oxidation model, and a semi-empirical quantum chemistry model. The efficacy of the validation workflow for developing predictive models, and the scientific insights uncovered from the analysis, is discussed.

To my parents, for instilling a sense of curiosity
To Sunjung, for the joy you bring to my life

# Contents

# List of Figures

# List of Tables

## Acknowledgments

Throughout my life, I have had the opportunity to be mentored by so many truly extraordinary and inspiring people, who have helped me become the researcher I am today. The following research would not have been possible without them.

First and foremost I would like to thank my advisors, Professor Michael Frenklach and Professor Andrew Packard. Their experience and guidance enabled me to think more critically about my research. Michael had taught me so much over these recent years but none more important than to be critical and skeptical of what is written before accepting it as truth. Michael had also made my transition from experimental to computational research as smooth as possible and through that transition, I found a passion for coding and predictive modeling. Andy taught me to become a better communicator and inspired me to be creative when trying to solve problems. I will always be thinking about how to break a problem down into a simpler (often lower-dimensional) form to easily illustrate a concept. It would require another dissertation to list all that I have learned from Michael and Andy over the past years, for the sake of brevity, I am sincerely grateful.

I am thankful for the funding support of the Department of Energy, National Nuclear Security Administration, under Award Number(s) DE-NA0002375. Ultimately the funding and resources used in this dissertation were provided by American taxpayers, to whom I am appreciative.

I wish to thank my dissertation committee, consisting of Michael Frenklach, Andrew Packard, Philip Stark, and Carlos Fernandez-Pello for their time, guidance, and comments.

Outside of Berkeley, I had the pleasure to be a part of the Carbon-Capture Multidisciplinary Simulation Center led by Professor Philip Smith at the University of Utah. Phil's enthusiasm for research is intoxicating which made the Center a wonderful group to be part of. From the Center, I also want to extend my thanks to Professor Sean Smith who had spent hours discussing various ideas for instrumentation modeling on the reduced char oxidation model. I also had the opportunity to work with Salvatore Iavarone who has been a wonderful collaborator and friend.

From Sandia National Laboratories, I would like to thank Dr. Habib Najm and Dr. Khachik Sargsyan for the insightful discussions of Bayesian inference, model error, and dynamical systems.

I would also like to acknowledge the support of professors and students I had worked with at the University of Illinois at Chicago (UIC), where I completed my undergraduate degree. My academic advisor at UIC, Professor Jeremiah Abiade, was a mentor by every meaning of the word. Professor Abiade opened my eyes to the possibility of graduate school that I am grateful for. Professor Constantine Megaridis for introducing me to microfluidic research and being supportive of my passions towards combustion science. During my time at Professor Megaridis's laboratory, I worked alongside Tom Schutzius who not only became a friend but showed me the tenacity required to be a successful graduate researcher. Lastly, I would not be where I am today without the help of Professor Kenneth Brezinsky. Hearing about Professor Brezinsky's shock-tube laboratory and his research experiences not only provided additional context and meaning to coursework but also *ignited* my interest in combustion science.

I was fortunate enough to have the opportunity to work with Professor Paul Ronney at the University of Southern California on developing additive manufactured microcombustors. I am thankful to Professor Ronney for the opportunity where I was able to reaffirm my interest in combustion science.

UC Berkeley has been an intellectually stimulating environment. This was not only due to the faculty but also the incredible graduate students and collaborators whom I had the opportunity to work alongside. They made the time at UC Berkeley, and especially the late hours in Hesse Hall, much more enjoyable. From our lab I want to give thanks to David Edwards for getting me started in computational quantum chemistry, Ravi Singh for countless discussions ranging from soot oxidation to predicting March Madness games, Arun Hegde for being a sounding board for uncertainty quantification strategies and methodologies, and Zhenyuan Liu for making our lab an enjoyable workplace. Outside of our lab, I greatly benefited from the professional and personal conversations I had with Xian Shi, Je Ir Ryu, Yulin Chen, Tiernan Casey, Anna Maria Valencia Lopez, Daniel Pineda, Alex Frank, Octavio Narvaez-Aroche, Kate Schweidel, Russell LaBrie, Miguel Sierra-Aznar, David Vuilleumier, Emmanuel Sin, Fredrik Grøvdal, and He Yin. I would also like to give thanks to Dr. Dmitry Zubarev for his help and insight in semi-empirical quantum chemistry methods and statistical learning theory.

Finally, I would like to thank my family. My parents, whose love and support made me who I am today. My sisters, Katie and Kari, for supporting me despite not calling as often as I should and for understanding when Christmas gifts arrived three weeks late. Last, and most importantly, I would like to thank my wife, Sunjung. Thank you for being so encouraging and patient throughout all the long nights and weekends.

# Chapter 1

# Introduction

## 1.1 Motivation

Experimentation has long been the predominant route for investigating scientific phenomena. However, with more sophisticated computational simulations that employ underlying physics-based models, more scientists today rely on simulations for their scientific endeavors. Physics-based models are now an integral part of engineering and are essential to the analysis of complex physical problems [1–4]. Such simulations are increasingly used as test-beds for new engineering designs and to support decision-making. The reliability of a computer simulation is of the utmost importance when using models for decision-making. It is natural for scientists to question these models, e.g., what evidence is there to trust predictions from the computational model? How accurate is the model prediction? Such questions are fundamental to the field of uncertainty quantification (UQ), where reliability and uncertainty are studied across various problems, e.g., model validation, code verification, parameter estimation, inverse problems, and prediction.

The U.S. Department of Energy had defined UQ as “the end-to-end study of the reliability of scientific inferences [5].” For any inference to be practical, it will demand that the model agrees adequately with reality, i.e., with experimental evidence. This assessment, termed *model validation*, quantifies the accuracy of a model’s output against experimental evidence [6, 7]. Model validation can ensure that a model is being used within its domain of applicability [8]. For a computational model to be useful does not require the model to be valid for all scenarios, only that it is valid in the scenarios for which it is being employed. Use of a computational model outside of the domain where it has been validated, i.e., extrapolating model predictions, is always a perilous task. Model validation (and UQ in general), is not meant to determine if a model is ‘true.’ UQ tells us that, “if you accept the validity of the model (to some quantified degree), then you must logically accept the validity of certain conclusions (to some quantified degree) [9].”

A common problem in UQ is the forward propagation of uncertainty in model prediction.

Model prediction determines the uncertainty (or error) associated with a model output due to the uncertainty in the model parameters. Model parameters are not known precisely, as they were measured with error. This lack of certainty propagates through the model into an uncertain prediction. Physics-based models and their associated set of model parameters attempt to represent reality. Therefore, it is crucial to refine the model parameters and reduce their related uncertainties to obtain a better understanding of reality.

Methods for computing the forward propagation vary depending on how the uncertainty is modeled. Uncertainty can be modeled as a probabilistic (stochastic) or deterministic quantity, and in the following section, these representations of uncertainty are discussed. The *inverse problem* is the reverse of the forward problem, where one infers the unknown parameters from a set of experimental measurements. Inverse problems, in general, are challenging to solve as they may not have a unique solution [10].

Model validation and UQ heavily depend on experimental evidence, which is often generated through scientific experimentation. Experimental results can be found in approximately 33,000 active peer-reviewed journals with nearly three million new scientific papers each year [11]. With the abundance of published content, discovering, extracting, and transforming a heterogeneous collection of experimental data into knowledge is challenging [12, 13]. It is clear to the scientific community that analyzing experimental data requires better organization [14, 15] as well as a platform for collaboration to help develop predictive models [16–18]. Here, a model is deemed "predictive" when the model prediction adequately agrees with experimental data.

In this dissertation, I aim to develop UQ tools to aid in the analysis of physics-based models by utilizing an online database of experimental data. These tools are demonstrated in three real-world applications, described in Section 1.2. In these examples we remind ourselves that a model can never truly be validated in the sense that it is certified to perform well in scenarios beyond the data. A model can only be assessed based on the experimental data (evidence) at hand. We can only determine the compatibility with the current evidence. As new information is obtained, a model deemed valid could become incompatible with the new evidence.

## Representation of uncertainty

Uncertainty can enter a complex physics-based model through various sources, e.g., model parameters, model form, experimental data [6]. Identifying these sources and their magnitudes are essential to a UQ analysis. Many methodologies have been developed to represent, combine, and propagate uncertainty, e.g., probability theory (frequentist [19] and Bayesian approaches [20–23]), interval propagation [24, 25], probability bound analysis [26], evidence theory [27, 28], fuzzy logic [29, 30], etc. In this dissertation, an optimization-based framework for UQ will be used where constraints specify parametric and experimental uncertainty. The details of this framework are discussed in Chapter 3. A review of a more general statistical framework for inverse problems is discussed in [31].

Both probabilistic and non-probabilistic methodologies for UQ have been studied extensively, and it cannot be stated that one method is superior to the other. There are advantages and disadvantages for each [32, 33], and these have been compared [34, 35]. Often, a problem's description of the uncertainty dictates the more suitable methodology. These methods should not add additional information that is not present, but also not ignore information that is present. For example, when representing uncertainty as a prior or constraint careful treatment is necessary as far more information than intended can be included in the analysis [36].

## 1.2 Organization of dissertation

This dissertation intends to illustrate how tools for UQ can be applied to practical engineering problems by combining the UQ analysis with access to experimental data. A validation workflow is demonstrated on three examples to quantify the agreement (or disagreement) between models and experimental data. The dissertation is organized as follows:

1. This first chapter summarizes the motivation for this dissertation. An outline of the challenges in computational modeling, decision-making, and prediction under uncertainty is given, with an emphasis on model validation.

2. The second chapter underlines the necessity of organized validation data, specifically storing experimental data with a common structure and the interlinking between the data records. A summary of the PrIMe Data Warehouse, which was developed to archive chemical kinetics experimental data, is provided. A collection of solid-fuels experiments was recently integrated into the Data Warehouse by building upon existing data models. Tools for access to the Data Warehouse and building stand-alone applications are introduced.

3. The third chapter focuses on a mathematical framework used for uncertainty quantification, called Bound-to-Bound Data Collaboration (B2BDC). I present this framework with a focus on model validation that is achieved through B2BDC's scalar consistency measure. A validation workflow is proposed that integrates the B2BDC framework with experimental data retrieved from the PrIMe Data Warehouse.

4. The fourth chapter introduces a novel strategy for developing piecewise polynomial surrogate models for B2BDC. This strategy utilizes experimental data during the fitting procedure to determine if the model and data are compatible over a subdomain. The efficacy of the strategy is demonstrated on a simple example and used in the analysis in Chapter 6.

The following three chapters demonstrate how the developed validation workflow and piecewise modeling strategy can be utilized to examine complex physical systems.

5. The fifth chapter explores a developed $H_2/O_2$ system and its compatibility with shock-tube and laminar flame speed experiments. Here, the computational model is known with high certainty; however, considerable uncertainty exists in the experimental measurements. The developed chemical kinetics mechanism was incompatible with a set of low-temperature, high-pressure shock-tube experiments. The developed mechanism, UQ analysis, and results are discussed. Reasons for the disagreement observed in the shock-tube experiments are discussed in light of recent experiments and analysis.
6. The sixth chapter analyzes the development of a reduced char oxidation model when both the char model and experimental data have significant uncertainty. The validation workflow was employed iteratively to construct a reduced-order model, by systematically invalidating potential model forms. With this novel strategy for piecewise modeling, I uncovered a potential issue with a widely used prior modeling assumption.
7. The seventh chapter studies the ability of a semi-empirical quantum chemistry model to predict the heats of formation of a series of alkanes. The experimental data have high precision, whereas the computational model is empirically based and uncertain. The quantum chemistry model, experimental data, and the analysis are discussed. A charge-based model discrepancy is explored as a potential means of rectifying the observed inconsistency that provides additional feedback to domain scientists.

The three systems examined using this methodology illustrate how the developed workflow can provide analysis and feedback when the model or data is uncertain.

8. The final chapter concludes by encapsulating the main findings and results from the three use cases. Current challenges for model validation and uncertainty quantification are presented, with some suggestions of promising approaches.

## 1.3 Contributions of dissertation

The work presented in this dissertation creates tools that combine UQ analysis with an organized collection of validation data. These tools and techniques are then applied to refine our knowledge and understanding of complex physics-based models. These contributions include:

- The creation of a validation workflow that connects the online platform of PrIMe with the B2BDC UQ framework. The workflow acts as a semi-automated and systematic procedure for model validation. The workflow was used to analyze three systems in this dissertation.

- Employing this validation workflow in the $H_2/O_2$ combustion model, identified a set of potentially problematic shock-tube experiments that span the high-pressure and low-temperature scenarios. Inconsistencies between model predictions and data under these different conditions called for a reconsideration of the experimental measurements, assumed uncertainties, and the idealized reactor used in the simulation.

- The B2BDC methodology was used to guide the development of piecewise polynomial surrogate models. A novel strategy for piecewise modeling is presented that combines piecewise surrogate modeling and B2BDC's scalar consistency measure. This strategy eliminates subdomains incompatible with experimental evidence. On the tested examples, a significant reduction in the overall number of function evaluations is achieved by targeting regions of the parameter space where the model adequately predicts the data.

# Chapter 2

# Providing structure to experimental data through PrIMe

Experimental data plays a critical role in model development and validation. A database of reliable experimental data can help validate models and ensure the preservation of historical data. The cloud-based infrastructure Process Informatics Model (PrIMe) was established to support the development of predictive chemical kinetic models, by enabling automated access to experimental data. Automated access to data requires the experimental data to be organized and have a common structure; both are provided by the PrIMe Data Warehouse, an online repository of chemical kinetics data and experiments. The PrIMe Data Warehouse, the newly developed data models for solid-fuels, and Warehouse APIs are reviewed in this chapter.

## 2.1 Introduction

In the combustion community, scientists perform experiments to gain understanding of the fundamental combustion processes. Apparatuses are built for experiments; instrumentation is used to measure the various physical phenomena, from flame propagation speeds to the formation or decay of radical species. Measurement data are essential for verifying the reliability of a developed chemical kinetics model through model validation or prediction. Additional experimental data can be helpful for comparison of a model's output at various scenarios. Added data can also be useful in model prediction, where the set of compatible model parameters is reduced, thus decreasing the uncertainty in a model prediction (as described in Section 3.3). Unfortunately, acquiring diverse data is not easy.

Published journal articles use tables and figures to report and visualize experimental data. Often, additional data are stored in the supplementary materials, which can be in a variety of file formats based on the experimenters' preferences, e.g., Excel spreadsheets, CSV files, plain-text files, etc. No one standard prevails. Data collection is the process of gathering relevant data across the published literature, which is an arduous task. However, data collected cannot be used immediately. Experimental data must be parsed and cleaned, extracting the pertinent information and fixing typographical errors. Each data set needs individual attention. Developing specialized parsing scripts to parse each data set when considering a substantial collection of data, quickly becomes a burden. After the data is parsed, additional issues may arise because of missing information, e.g., a data set may not have reported the measurement units. Without a standardized data format, searching for a sizable collection of experimental data introduces significant and avoidable challenges.

The necessity of a standardized combustion database containing (i) a complete record of the experiment, (ii) a machine-readable data format, and (iii) a common format for all records was addressed by the pioneering work of PrIMe Kinetics [13, 37]. PrIMe is a cloud-based infrastructure to help develop predictive chemical kinetics models through collaborative data. PrIMe is composed of two major components: the PrIMe Workflow Application and the PrIMe Data Warehouse. The PrIMe Workflow Application is a front-end interface with a variety of tools for users to discover, create, share, and analyze chemical models [38]. The PrIMe Data Warehouse is an online repository that archives combustion data and their associated uncertainties. Central to the PrIMe Data Warehouse are the PrIMe Data Models, which assign a common data format for archiving experimental data. Early work by Michael Frenklach and Zoran Djurisic [37] laid the foundation of the Data Warehouse and its data models, which will be extended to store a collection of solid-fuel data.

## 2.2 PrIMe Data Models

A data model is a set of standards that defines how data and its properties are organized and associated. PrIMe's Data Models introduced a standard on how combustion data should be organized [13]. All data records are encoded as an eXtensible Markup Language (XML) document, which supplies additional metadata information in the form of tag pairs enclosing the data. Each XML tag is labeled, allowing the data to be searched and giving structure to the experimental data.

Structured data labels each piece of information, making it more efficient to process and analyze. Source Code 2.1 shows an example of how XML metadata provides structure to data. In this example, a shock-tube with an inner diameter of 9.82 cm is encoded in XML.

Source Code 2.1: Example of XML Metadata

```xml
<property name="diameter" id="d_inner" units="cm" description="Inner diameter of
↪  shock-tube">
<value>9.82</value>
</property>
```

The `property` tag pair marks the start and end of an element. All information between the opening and closing `property` tags are associated with a single property element. Attributes relate additional information to a property element, e.g., `name`, `id`, `units`, and `description`. The `name` of the property is a diameter, `units` are centimeters, and the `id`, `d_inner`, denotes an inner diameter. Nested between the `property` tags is the child element, `value`, which contains the measurement value, 9.82. Collectively, the unstructured text description of "a shock-tube with an inner diameter of 9.82 cm" was encoded in XML in Source Code 2.1 by adding metadata information.

The *XML schema* governs the organization of the XML tags and determines which are valid. XML schemas are representations of the PrIMe Data Models, acting as a blueprint for all experimental records stored in the Data Warehouse. XML schemas can require specific information, e.g., reporting the measurement units of a property. All properties of an XML document must contain the measurement units; otherwise, the document is invalid.

Documents that adhere to the rules set by the data model are validated by the schema and can be archived in the PrIMe Data Warehouse. This validation process ensures that each data record complies with the standards set by the PrIMe Data Models. Records that have missing information, misspellings/typos, or are duplicate entries are invalid.

## 2.3 Organization of the PrIMe Data Warehouse

The PrIMe Data Warehouse is a central repository for archiving chemical kinetics data relevant to the development of predictive models. These documents include experimental data, the properties of the chemical species, or the chemical kinetic models. Archiving a document in the Data Warehouse requires more than just a data model; the organization of the documents is also essential. Documents in the PrIMe Data Warehouse are organized by *catalogs*, where each catalog belongs to a *collection*. Table 2.1 shows a list of the collections in the PrIMe Data Warehouse.

For each item listed in Table 2.1, there is an associated data model. Each bulleted item is a collection in the Data Warehouse. Non-bulleted items are auxiliary data records, which are XML documents containing supporting information to another record. A schematic illustrating the hierarchical organization of the Data Warehouse is shown in Figure 2.1.

Table 2.1: List of collections in PrIMe Data Warehouse

- Bibliography (`b`)
- Element (`e`)
- Chemical Species (`s`)
  - Transport Coefficient (`tr`)
  - Thermodynamic Coefficient (`thp`)
  - Chemical Analysis (`ca`)
- Reaction (`r`)
  - Reaction Rate (`rk`)
- Experiment (`e`)
- Chemical Model (`m`)
- Dataset (`d`)
  - Surrogate Model (`sm`)
- Data Attribute (`a`)
  - Instrument Model (`im`)
- Optimization Variable (`v`)
  - Optimization Variable Bounds (`vb`)

## Linking XML documents

Each XML document stored in the PrIMe Data Warehouse has only one copy, preventing repeated information from being archived. Linking (or cross-referencing) documents is possible when there is just a single XML document.

To exemplify the linking between records, we will consider an example of methane gas. Methane gas is a chemical species; therefore, it has an associated XML document in the `species` catalog. Each XML document is assigned a unique identifier, known as a PrIMe ID (`primeID`). The PrIMe ID for methane is `s00009193`. Any experiment, reaction, or transport property that is associated with or uses methane gas will reference methane by its PrIMe ID.

The letter preceding the eight digit number in the PrIMe ID specifies the catalog. For methane gas, the `s` corresponds to a chemical species record. Table 2.1 shows the three

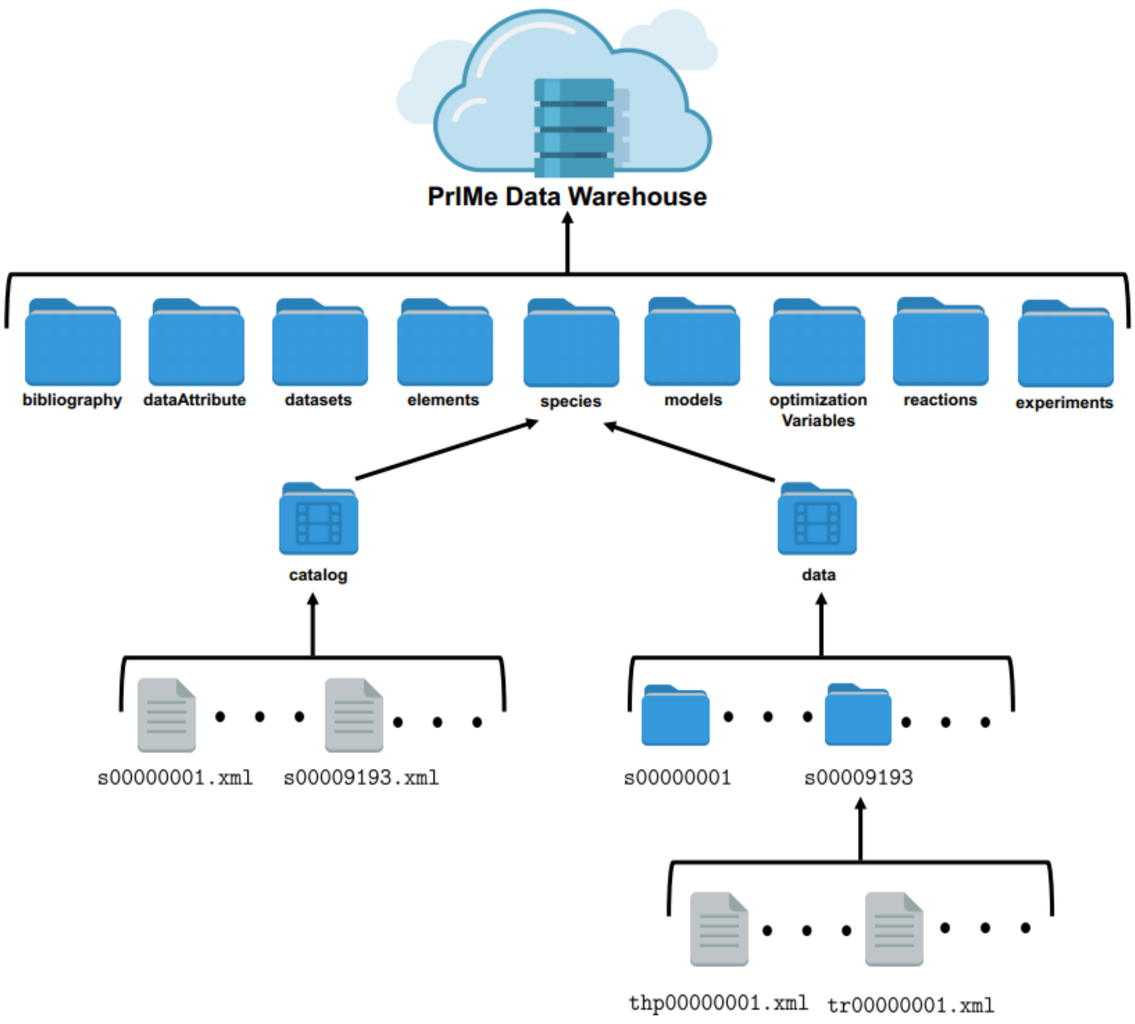


Figure 2.1: Schematic of the PrIMe Data Warehouse. Each collection (bibliography, dataAttribute, datasets, etc.) contains a catalog folder and data folder. The catalog folder contains all XML files for the particular collection. The data folder contains any additional information for the particular record. A thermodynamic polynomial XML file, `thp00000001.xml` and transport coefficient file, `tr00000001.xml` are shown in the data directory for species `s00009193`.

possible auxiliary files for a chemical species record: transport coefficient files, thermodynamic coefficient files, and chemical analysis files. Each auxiliary file provides additional data (properties) about the associated chemical species. Transport properties are stored in a transport coefficient XML document and thermodynamic properties are stored in the thermodynamic coefficient XML document. Chemical analysis was recently developed by the author and added to the PrIMe Data Warehouse to further describe solid-fuel data. Both transport and thermodynamic files are available for methane gas. A new XML document with its unique PrIMe ID will be created for each new transport or thermodynamic data set.

Source Code 2.2 shows a thermodynamic XML document for methane gas. For all XML documents, the opening tag defines the record; in this case, `thermodynamicPolynomials`. Inside the opening tag is the attribute `primeID`, which has the value `thp00000003`. This thermodynamic PrIMe ID is unique for the associated chemical species. The tag `speciesLink` connects the thermodynamic polynomial with an associated chemical species, `s00009193`; i.e., methane gas.

Similarly, the tag `bibliographyLink` connects the thermodynamic polynomial to a bibliography reference. A majority of the content in the document is between the `polynomial` tags, which define the NASA polynomials and their valid temperature ranges [39]. In this single XML document, then, all thermodynamic information is labeled and annotated with provenance information.

Source Code 2.2: Thermodynamic Polynomial XML

```xml
<thermodynamicPolynomials xmlns="http://purl.org/NET/prime/"
↪  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" type="nasa7"
↪  primeID="thp00000003" xsi:schemaLocation="http://purl.org/NET/prime/
↪  http://warehouse.primekinetics.org/schema/thermodynamicPolynomials.xsd">
<copyright>primekinetics.org 2006</copyright>
<bibliographyLink preferredKey="Burcat A., Ruscic B., 2006." primeID="b00014727"/>
<preferredKey group="prime">g 8/99 ANHARMONIC</preferredKey>
<speciesLink preferredKey="CH4" primeID="s00009193"/>
<referenceState>
  <Tref units="K">298.15</Tref>
  <Pref units="Pa">100000</Pref>
</referenceState>
<dfH units="J/mol">-74595.4242</dfH>
<polynomial>
  <validRange>
    <bound kind="lower" property="temperature" units="K">200.000</bound>
    <bound kind="upper" property="temperature" units="K">1000</bound>
  </validRange>
  <coefficient id="1" label="a1">5.1491</coefficient>
  <coefficient id="2" label="a2">-0.013662</coefficient>
  <coefficient id="3" label="a3">4.9145e-005</coefficient>
  <coefficient id="4" label="a4">-4.8425e-008</coefficient>
  <coefficient id="5" label="a5">1.666e-011</coefficient>
  <coefficient id="6" label="a6">-10246.5983</coefficient>
  <coefficient id="7" label="a7">-4.6385</coefficient>
</polynomial>
<polynomial>
  <validRange>
```

```
    <bound kind="lower" property="temperature" units="K">1000</bound>
    <bound kind="upper" property="temperature" units="K">6000.000</bound>
  </validRange>
  <coefficient id="1" label="a1">1.6533</coefficient>
  <coefficient id="2" label="a2">0.010026</coefficient>
  <coefficient id="3" label="a3">-3.3166e-006</coefficient>
  <coefficient id="4" label="a4">5.3648e-010</coefficient>
  <coefficient id="5" label="a5">-3.147e-014</coefficient>
  <coefficient id="6" label="a6">-10009.5936</coefficient>
  <coefficient id="7" label="a7">9.9051</coefficient>
</polynomial>
</thermodynamicPolynomials>
```

### Data enrichment and reproducibility

One benefit of the PrIMe Data Warehouse is the ability to archive complete documentation of the experiment conducted. Full documentation is essential when preserving historical data, which may contain known typographical errors that are commonly not corrected when published in a journal [40]. The community can enrich or add value to an experiment by maintaining an experimental record with the most up-to-date information, which is beneficial to all researchers.

Reproducibility is especially important when considering validation data that has been published in graphical figures. Recovering the exact measurement data from a figure can be challenging. Data is often extracted by using a plot digitizer, which is a tool for extracting numerical values from a digital figure. Human interaction with this tool creates inherent variability in the extracted numerical values. Two researchers claiming to use the "same" set of measurement data could use different extracted values, making their analyses difficult to reproduce.

The public XML documents in the PrIMe Data Warehouse can help demystify any ambiguity as to *what* experimental data was used in a study. Experimental documents stored in the Data Warehouse are easily accessible through the PrIMe Workflow Application [37, 38] or through the Warehouse API described in Section 2.5. Each document from the Data Warehouse can be referenced by their unique PrIMe ID, giving subsequent researchers a means of obtaining an exact copy of the data used in a study, allowing for the study to be reproduced. PrIMe strives to deliver both publicly available experimental data and software for developing predictive models, both of which are critical components to computational reproducibility [41].

## 2.4 Addition of solid-fuels experimental data

In Chapter 6, the validity of a reduced char oxidation model is analyzed against a set of validation data coming from a collection of solid-fuel experiments. At the beginning of our study, solid-fuel experimental data did not exist in the PrIMe Data Warehouse. Data models for chemical species are encoded by their chemical composition, i.e., the number of atoms of each element. A detailed description of the molecular composition of a solid-fuel is not attainable because of the complexity and inhomogeneity of the fuel. Often proximate or ultimate analyses are used to characterize the composition of a solid-fuel [42]. A proximate analysis abstracts the basic chemical properties, i.e., the percentage of moisture, ash, volatile matter, and fixed carbon. Elemental composition of the solid-fuel can be determined through the ultimate analysis, where the percentages of carbon, hydrogen, oxygen, nitrogen, and sulfur are measured. To complicate matters, the proximate and ultimate analysis can vary between solid-fuels of the same name. One example of this is Pittsburgh No. 8 coal, which is a typical medium sulfur high-volatile bituminous coal coming from the Pittsburgh Coal Seam in Pennsylvania and Ohio. A piece of Pittsburgh No. 8 coal used in an experiment conducted at Sandia National Laboratories [43] contained 6.9% ash, whereas another test at the International Flame Research Foundation [44] used a piece of Pittsburgh No. 8 containing 8.1% ash. Coal mined from one shaft can vary in its properties from coal in a nearby shaft.

Due to the variability of the chemical properties of a similar solid-fuel, it is essential to document the proximate and ultimate analyses of the fuel used in each experiment. The chemical species data models have been expanded to handle either a chemical composition specified by the number of atoms in each element, shown in Source Code 2.3, or by a secondary chemical analysis file, shown in Source Code 2.4. The chemical species XML document for methane gas is seen in Source Code 2.3. Methane gas can be specified by its atomic makeup. Therefore, Lines 15-18 report its composition; namely 1 carbon and 4 hydrogen atoms. Unlike methane, Black Thunder Coal, shown in Source Code 2.4, cannot be specified by an atomic makeup. Lines 8-10 show the chemical composition is described by a chemical analysis file, `ca00000001` (shown in Appendix A.2).

Source Code 2.3: Example of chemical species XML specified by atomic makeup

```
1   <chemicalSpecies xmlns="http://purl.org/NET/prime/"
    ↪  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" primeID="s00009193"
    ↪  xsi:schemaLocation="http://purl.org/NET/prime/
    ↪  http://warehouse.primekinetics.org/schema/species.xsd">
2     <copyright>primekinetics.org 2005-2018</copyright>
3     <content bibliography="b00014319" copyrighted="true" source="NIST">
4       Elements attributed to NIST are part of a collection copyrighted by NIST.
5     </content>
6     <preferredKey group="prime" type="formula">CH4</preferredKey>
7     <chemicalIdentifier>
8       <name source="NIST" type="CASRegistryNumber">74-82-8</name>
9       <name source="NIST" type="formula">CH4</name>
10      <name source="NIST">methane</name>
11      .
12      .
13      .
14    </chemicalIdentifier>
```

```
  <chemicalComposition>
    <atom symbol="C">1</atom>
    <atom symbol="H">4</atom>
  </chemicalComposition>
</chemicalSpecies>
```

Source Code 2.4: Example of chemical species XML specified by chemical analysis

```
<chemicalSpecies xmlns="http://purl.org/NET/prime/"
↪  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" primeID="s00011163"
↪  xsi:schemaLocation="http://purl.org/NET/prime/
↪  http://warehouse.primekinetics.org/schema/species.xsd">
  <copyright>primekinetics.org 2005-2018</copyright>
  <preferredKey>Black Thunder Seam, WY</preferredKey>
  <chemicalIdentifier>
    <name source="Sandia">Black Thunder Seam</name>
        <name type="FuelType">Subbituminous</name>
  </chemicalIdentifier>
  <chemicalComposition item="coal">
          <coal specifiedBy="chemicalAnalysis">ca00000001</coal>
  </chemicalComposition>
</chemicalSpecies>
```

Chemical analysis data models helped store data on 269 solid-fuels used in 2,710 experiments in the PrIMe Data Warehouse. A collection of experimental data was provided by Sandia National Laboratories and the International Flame Research Foundation. The flexibility of the PrIMe Data Models made the storage of solid-fuel data possible and gave a standard structure for all experimental records. In the following section, we discuss the newly developed Warehouse API and a front-end application for analyzing coal data.

## 2.5 PrIMe Data Warehouse API

Archiving combustion data in the PrIMe Data Warehouse preserves the data, but most practical applications require *access* to the data. There are two methods for retrieving experimental data from the Data Warehouse. The first method uses the PrIMe Workflow Application, a cloud-based interface for creating and executing projects on PrIMe [37]. Each PrIMe Workflow Application project is formed by a set of components. Each component is an interactive block of code that executes online and enables a user to perform a simulation or data analysis task.

A more versatile method for accessing the Data Warehouse is through the Warehouse API, available online at [45] and in Appendix A.2. The Warehouse API is a set of methods and objects written in MATLAB and Python and are used to communicate with the PrIMe Data Warehouse. The Warehouse API constructs an Elasticsearch query, which is then passed to and executed by the PrIMe server [46]. Results are then returned to the user. By standardizing the storage of experimental data, users can find all records of interest with a single search via the Warehouse API. The Warehouse API will be used throughout this dissertation to find quantities of interest and obtain all pertinent simulation information, i.e., input conditions, observed values, and the experimental bounds.

The PrIMe Data Models and API allow developers to create software that can interact with documents stored in the PrIMe Data Warehouse. Parsing relevant features from an XML document is simple, only requiring a couple of lines of code, since all experimental documents share a common structure. An example of an application that can be built using the Warehouse API is the Carbon Capture Multidisciplinary Simulation Center (CCMSC) Coal Database, which is discussed below.

### CCMSC Coal Database

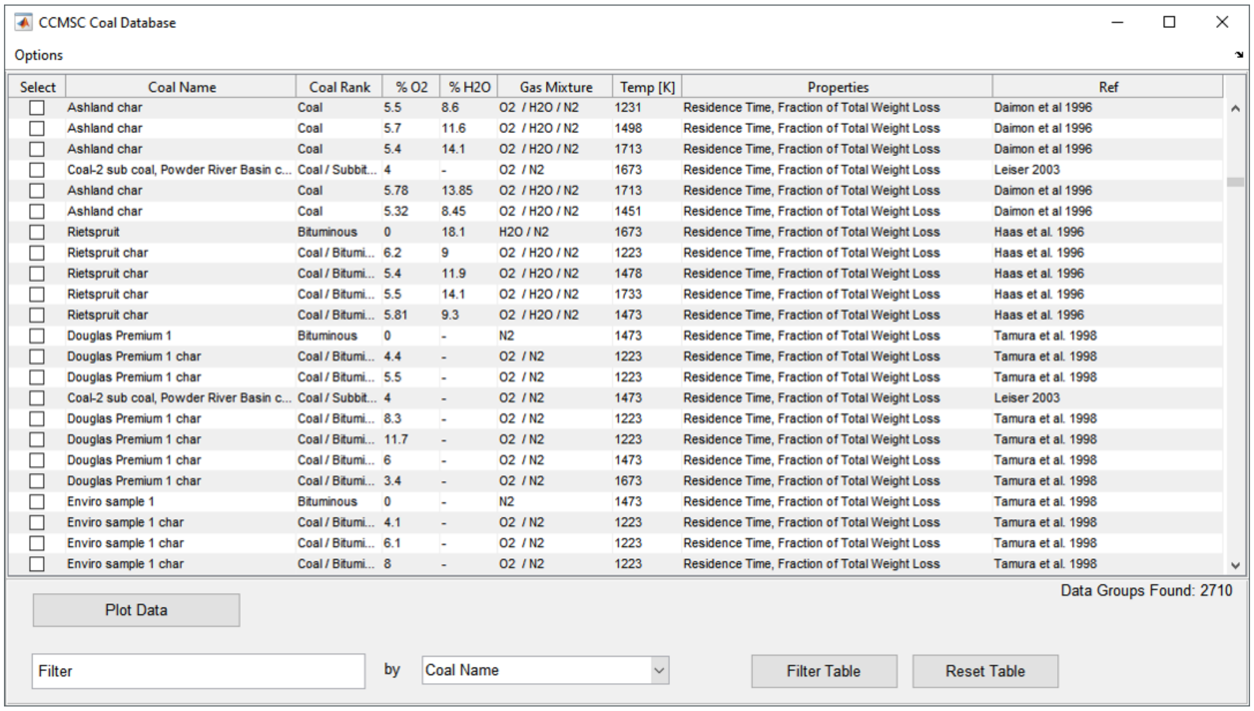
A significant amount of coal data was collected from a crowd-sourced effort to validate a reduced char oxidation model (discussed in Chapter 6). The culmination of this is the CCMSC Coal Database [47], a front-end application to the experimental data on coal (or char) in the PrIMe Data Warehouse. The CCMSC Coal Database is a stand-alone MATLAB application that uses the Warehouse API. Figure 2.2 shows the application's GUI, through which users can search for relevant coal experiments by coal name, coal rank, gas mixture, properties, etc. The interaction between the application and the Data Warehouse is through the Warehouse API. Search results are visualized within the application or data are saved locally.

Figure 2.3 shows two examples of data visualization in the Coal Database application. Figure 2.3(a) is experimental data of mass loss of Rietspruit char from the International Flame Research Foundation, while Figure 2.3(b) shows char combustion temperature from an experiment conducted at Sandia National Laboratories.

The CCMSC Coal Database application shows that a collection of experimental data stored in a standard, structured format can be efficiently queried, extracted, and analyzed, irrespective of where the data originated. The PrIMe Data Warehouse eliminated the difficulties in parsing an extensive collection of experimental data and is just one example of a chemical kinetics database. Alternative chemical kinetics databases that fulfill a similar role to the Data Warehouse are discussed in the following section.

## 2.6 Alternative chemical kinetics databases

This chapter gave an overview of how the PrIMe Data Warehouse and its data models create a common, structured data format for the archival of chemical kinetics data. It is important to acknowledge a few other recently developed chemical kinetics databases, where, in all examples, a new data format was developed to archive chemical kinetics data, with the


CCMSC Coal Database

Options

| Select | Coal Name | Coal Rank | % O2 | % H2O | Gas Mixture | Temp [K] | Properties | Ref |
|---|---|---|---|---|---|---|---|---|
| ☐ | Ashland char | Coal | 5.5 | 8.6 | O2 / H2O / N2 | 1231 | Residence Time, Fraction of Total Weight Loss | Daimon et al 1996 |
| ☐ | Ashland char | Coal | 5.7 | 11.6 | O2 / H2O / N2 | 1498 | Residence Time, Fraction of Total Weight Loss | Daimon et al 1996 |
| ☐ | Ashland char | Coal | 5.4 | 14.1 | O2 / H2O / N2 | 1713 | Residence Time, Fraction of Total Weight Loss | Daimon et al 1996 |
| ☐ | Coal-2 sub coal, Powder River Basin c... | Coal / Subbit... | 4 | - | O2 / N2 | 1673 | Residence Time, Fraction of Total Weight Loss | Leiser 2003 |
| ☐ | Ashland char | Coal | 5.78 | 13.85 | O2 / H2O / N2 | 1713 | Residence Time, Fraction of Total Weight Loss | Daimon et al 1996 |
| ☐ | Ashland char | Coal | 5.32 | 8.45 | O2 / H2O / N2 | 1451 | Residence Time, Fraction of Total Weight Loss | Daimon et al 1996 |
| ☐ | Rietspruit | Bituminous | 0 | 18.1 | H2O / N2 | 1673 | Residence Time, Fraction of Total Weight Loss | Haas et al. 1996 |
| ☐ | Rietspruit char | Coal / Bitumi... | 6.2 | 9 | O2 / H2O / N2 | 1223 | Residence Time, Fraction of Total Weight Loss | Haas et al. 1996 |
| ☐ | Rietspruit char | Coal / Bitumi... | 5.4 | 11.9 | O2 / H2O / N2 | 1478 | Residence Time, Fraction of Total Weight Loss | Haas et al. 1996 |
| ☐ | Rietspruit char | Coal / Bitumi... | 5.5 | 14.1 | O2 / H2O / N2 | 1733 | Residence Time, Fraction of Total Weight Loss | Haas et al. 1996 |
| ☐ | Rietspruit char | Coal / Bitumi... | 5.81 | 9.3 | O2 / H2O / N2 | 1473 | Residence Time, Fraction of Total Weight Loss | Haas et al. 1996 |
| ☐ | Douglas Premium 1 | Bituminous | 0 | - | N2 | 1473 | Residence Time, Fraction of Total Weight Loss | Tamura et al. 1998 |
| ☐ | Douglas Premium 1 char | Coal / Bitumi... | 4.4 | - | O2 / N2 | 1223 | Residence Time, Fraction of Total Weight Loss | Tamura et al. 1998 |
| ☐ | Douglas Premium 1 char | Coal / Bitumi... | 5.5 | - | O2 / N2 | 1223 | Residence Time, Fraction of Total Weight Loss | Tamura et al. 1998 |
| ☐ | Coal-2 sub coal, Powder River Basin c... | Coal / Subbit... | 4 | - | O2 / N2 | 1473 | Residence Time, Fraction of Total Weight Loss | Leiser 2003 |
| ☐ | Douglas Premium 1 char | Coal / Bitumi... | 8.3 | - | O2 / N2 | 1223 | Residence Time, Fraction of Total Weight Loss | Tamura et al. 1998 |
| ☐ | Douglas Premium 1 char | Coal / Bitumi... | 11.7 | - | O2 / N2 | 1223 | Residence Time, Fraction of Total Weight Loss | Tamura et al. 1998 |
| ☐ | Douglas Premium 1 char | Coal / Bitumi... | 6 | - | O2 / N2 | 1473 | Residence Time, Fraction of Total Weight Loss | Tamura et al. 1998 |
| ☐ | Douglas Premium 1 char | Coal / Bitumi... | 3.4 | - | O2 / N2 | 1673 | Residence Time, Fraction of Total Weight Loss | Tamura et al. 1998 |
| ☐ | Enviro sample 1 | Bituminous | 0 | - | N2 | 1473 | Residence Time, Fraction of Total Weight Loss | Tamura et al. 1998 |
| ☐ | Enviro sample 1 char | Coal / Bitumi... | 4.1 | - | O2 / N2 | 1223 | Residence Time, Fraction of Total Weight Loss | Tamura et al. 1998 |
| ☐ | Enviro sample 1 char | Coal / Bitumi... | 6.1 | - | O2 / N2 | 1223 | Residence Time, Fraction of Total Weight Loss | Tamura et al. 1998 |
| ☐ | Enviro sample 1 char | Coal / Bitumi... | 8 | - | O2 / N2 | 1223 | Residence Time, Fraction of Total Weight Loss | Tamura et al. 1998 |

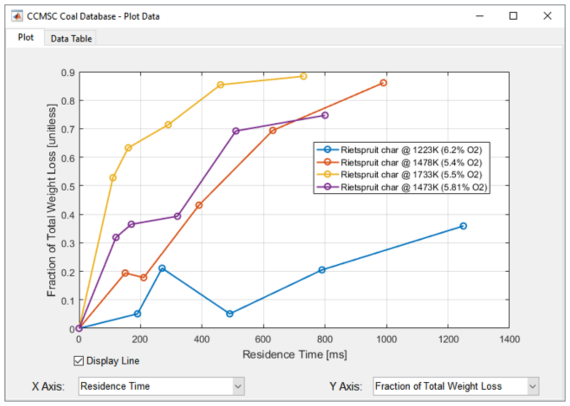
Data Groups Found: 2710

Figure 2.2: Snapshot of the stand-alone CCMSC Coal Database MATLAB application.

hopes of promoting wider adoption of the proposed format. The most interesting examples are ReSpecTh and ChemKED. ReSpecTh [49], launched in June 2015, has a data format that begins with the existing PrIMe Data Models (i.e., XML schemas) and makes a few modifications. The ReSpecTh database contains nearly 1,000 experiments, although many overlap with those already stored in PrIMe. Because of the inherent similarity in the formats, a comparison between PrIMe and ReSpecTh is made in Appendix A.3.

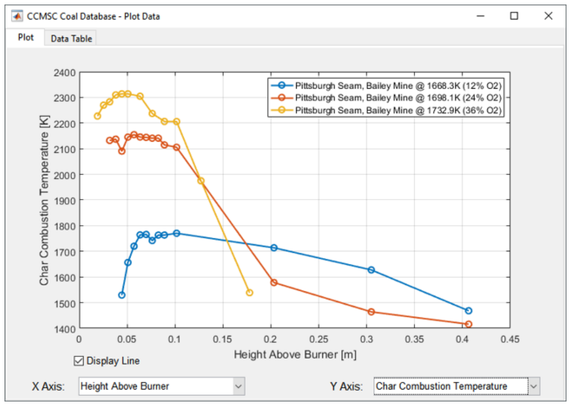
ChemKED launched in 2017 by Weber and Niemeyer [50]. Unlike PrIMe, ChemKED stores experimental data in YAML, which may be considered a cleaner format. ChemKED also launched with the Python package, PyKED, providing an interface with the ChemKED files. ChemKED has archived a collection of butanol, toluene, and n-heptane experiments. Although not an experimental data format, in 2013 Cloudflame [51, 52] provided an online infrastructure to execute numerical simulations and simplified searches of experimental data.

Irrespective of which chemical kinetics database a user adopts, the community has been moving towards archiving experimental data in a common format. The benefits of a combustion database outweigh the single, upfront cost of encoding a published journal article in the chosen format. Queries can be made across the entire database with no need to parse each journal article. Although many data formats have manifested in recent years, it is theoretically possible to convert from one data format to another, as long as a few necessary conditions are held. First, each experimental document has to contain full


CCMSC Coal Database - Plot Data
Plot
Data Table
Fraction of Total Weight Loss [unitless]
Residence Time [ms]
Rietspruit char @ 1223K (6.2% O2)
Rietspruit char @ 1478K (5.4% O2)
Rietspruit char @ 1733K (5.5% O2)
Rietspruit char @ 1473K (5.81% O2)
Display Line

X Axis: Residence Time
Y Axis: Fraction of Total Weight Loss

(a)


CCMSC Coal Database - Plot Data
Plot
Data Table
Char Combustion Temperature [K]
Height Above Burner [m]
Pittsburgh Seam, Bailey Mine @ 1668.3K (12% O2)
Pittsburgh Seam, Bailey Mine @ 1698.1K (24% O2)
Pittsburgh Seam, Bailey Mine @ 1732.9K (36% O2)
Display Line

X Axis: Height Above Burner
Y Axis: Char Combustion Temperature

(b)

Figure 2.3: Visualization of experimental data from the CCMSC Coal Database application. Panel a: the fraction of weight loss of Rietspruit char at various gas temperatures [48]. Panel b: char combustion temperature of Pittsburgh coal at three oxidation conditions [43].

information of the experiment conducted; e.g., measurement units, apparatus, conditions, bibliography reference, measured value, etc. Second, all information has to be distinctly labeled. Data format conversion — e.g., from PrIMe(A) to ChemKED(B) — would then be a mapping (A $\mapsto$ B) without loss of any information.

## 2.7 Summary

The PrIMe Data Warehouse and Data Models create a standard on how chemical kinetics data are encoded, organized, and archived. The flexibility of the PrIMe Data Models was highlighted when 2,710 solid-fuel experiments were added to the Data Warehouse. A new auxiliary data model was created to archive the proximate and ultimate analyses that are commonly used to characterize the chemical composition of solid-fuels. Researchers can interact with the entire collection of experimental data by using the Warehouse API. The CCMSC Coal Database demonstrated how the Warehouse API could also be used for application development. Experimental data stored in the Data Warehouse will be used as validation data throughout this dissertation.

# Chapter 3

# Bound-to-Bound Data Collaboration

Complex physics-based simulations are becoming increasingly common to support engineering analysis. Such computational models are used to test new designs and support decision-making. The accuracy of the information obtained from a computational simulation requires the model to be in good correspondence with nature, i.e., experimental evidence. In the previous chapter, we discussed a database that gives structure to experimental data; however, a database is only useful for predictive modeling if there exists a mathematical framework to quantify the agreement (or disagreement) between models and data. In this chapter, the uncertainty quantification framework called Bound-to-Bound Data Collaboration (B2BDC) will be discussed in the context of model validation and prediction.

## 3.1 Introduction

B2BDC is an optimization-based framework for uncertainty quantification, where uncertain model parameters are constrained by the combination of models and experimental data. This chapter will provide an overview of the B2BDC methodology. An emphasis is placed on the methodological details for model validation, precisely the consistency of a domain and model prediction. The methodology used here follows previous lab members works that derive many of the optimization problems shown below [53–56].

Let us define $\{M_e(x)\}_{e=1,\ldots,N}$ as a collection of polynomial surrogate models mapping from a common parameter space to various scalar-valued quantities of interest (QOIs). For a parameter vector $x \in \mathbb{R}^n$, the expression $M_e(x)$ evaluates the model prediction for the $e$-th quantity of interest (QOI). Uncertainty in the experimental observations of the QOIs are represented deterministically. These uncertainties come in B2BDC in the form of expertly assessed, interval bounds, $\{[\widetilde{L_e}, \widetilde{U_e}]\}_{e=1,\ldots,N}$.

Often, prior knowledge can confine the model parameters, $x$, to a set $\mathcal{H} \subset \mathbb{R}^n$. Prior knowledge can be represented as a set of constraints on either individual model parameters, e.g., $-1 \leq x_1 \leq 3$ or can represent relationships between parameters, e.g., $0 \leq 5x_1 + x_2 \leq 2$.

In B2BDC, polynomial surrogate models, $M_e(x)$, are used to approximate the often computationally expensive underlying model, $f(x, x_{s,e})$, where $x_{s,e}$ are the scenario parameters for the $e$-th QOI. Scenario parameters are the model parameters that are assumed to have no uncertainty. Approximating an underlying model with a surrogate model can cause a potential difference or error. Appendix B.2 discusses how surrogate modeling error is estimated. The fitting error of the $e$-th QOI, denoted $\epsilon_e$, will be accounted for in the analysis by adding the error to the associated experimental bounds, where $[L_e, U_e] = [\widetilde{L_e} - \epsilon_e, \widetilde{U_e} + \epsilon_e]$.

Constrained by both prior knowledge on $x$ and the experimental uncertainty ranges, $[L_e, U_e]$, each QOI has an associated feasible set of parameters for which model evaluations agree with the corresponding experimental bounds:

$$\mathcal{F}_e = \{x \in \mathcal{H} : L_e \leq M_e(x) \leq U_e, \text{ for } e = 1, 2, \ldots, N\}. \tag{3.1}$$

A *dataset* is the collection of model-data constraints and prior knowledge on $x$. Model parameters that satisfies all constraints in the dataset forms the feasible set of the dataset:

$$\mathcal{F} = \bigcap_{e=1}^{N} \mathcal{F}_e = \{x \in \mathcal{H} : L_e \leq M_e(x) \leq U_e, \text{ for } e = 1, 2, \ldots, N\}. \tag{3.2}$$

## 3.2 Dataset Consistency

Model validation in the B2BDC framework amounts to establishing the consistency of a dataset. If the feasible set $\mathcal{F}$ is nonempty, the models and data are said to be *consistent*. The *scalar consistency measure* can efficiently and provably quantify the consistency of a dataset [56]. The scalar consistency measure can be formulated as the following constrained optimization problem:

$$\begin{aligned} C_D = & \max_{\gamma \in \mathbb{R}, x \in \mathcal{H}} \gamma \\ & \text{s.t. } L_e + \frac{(U_e - L_e)}{2}\gamma \leq M_e(x) \leq U_e - \frac{(U_e - L_e)}{2}\gamma \\ & \text{for } e = 1, 2, \ldots, N, \end{aligned} \tag{3.3}$$

where "s.t." is an abbreviation for "subject to." $C_D$ is the consistency measure of the dataset, and $\gamma$ is the symmetrical tightening (or expansion) of the experimental bounds.

If $\gamma > 0$, the feasible set is non-empty as the experimental bounds can be tightened such that the feasible remains non-empty. The dataset is consistent as the models and data agree in the prior domain $x \in \mathcal{H}$. Alternatively if $\gamma < 0$, the feasible set is empty. This result tells us that the experimental bounds need to be expanded beyond their prescribed values, for any model parameter in $\mathcal{H}$ to agree with all model-data constraints. The dataset is then said to be *inconsistent*.

Exact calculation of the global maximum of the scalar consistency measure (Eq. 3.3) is difficult to attain. The scalar consistency measure is a nonconvex quadratically constrained quadratic program [57], a class of problems that are NP-Hard [58, 59]. Bounds on the value of $\gamma$ can be found by the inner ($\underline{\gamma}$) and outer ($\overline{\gamma}$) solutions to Eq. 3.3, where $\underline{\gamma} \leq \gamma \leq \overline{\gamma}$. To determine $\overline{\gamma}$, a convex relaxation to the original problem is necessary [56, 60]. A convex relaxation is a reformulation of an original optimization problem, into a convex problem, where the optimal value can be guaranteed to be the global solution [61]. With Eq. 3.3, the optimal value of the convex relaxation problem is guaranteed to be at least as large as the optimal value of the original problem, i.e., $\gamma \leq \overline{\gamma}$. Therefore, if $\overline{\gamma} < 0$, then it can be provably shown that the models and data are in disagreement over $x \in \mathcal{H}$.

If a dataset is inconsistent, elucidating the exact cause of the inconsistency can be challenging. The underlying model, the prescribed experimental bounds, or the prior domain of the model parameters may be at fault. One means of diagnosing potential dataset inconsistencies is through the sensitivities to the scalar consistency measure. Highly sensitive model-data constraints may indicate models and data that could potentially be responsible for the inconsistency [56]. Sensitivity to the scalar consistency measure was shown to be inefficient when many model-data constraints are contributing to an inconsistent dataset. For these cases, the *vector consistency measure* [62] was developed where individual relaxations are applied to each model-data constraint to identify a sparse resolution to the inconsistency. Throughout this dissertation, we will only require the application of the scalar consistency measure.

## 3.3 Model prediction

If the feasible set is non-empty, the dataset is consistent and a posterior analysis can occur by propagating all parametric uncertainties to model predictions. In B2BDC, the prediction of a particular model, $M_p(x)$, amounts to establishing bounds on the range of the prediction model, subject to model parameters in the feasible set, $\mathcal{F}$:

$$\left[\min_{x\in\mathcal{F}} M_p(x),\ \max_{x\in\mathcal{F}} M_p(x)\right]. \tag{3.4}$$

In Eq. 3.4, the prediction model need not be a member of the collection of models that define the feasible set, $\{M_e(x)\}_{e=1,\ldots,N}$. Often an analysis requires the posterior bounds on the range of an uncertain model parameter, which amounts to model prediction, where $M_p(x) = x_i$. Similar to the scalar consistency measure (Eq. 3.3), calculating the exact minimum or maximum value of Eq. 3.4 is NP-Hard; therefore, only inner and outer bounding solutions are computed [57]. In cases where validation data are not available to assess the predictivity of a dataset, an alternative approach is necessary. One technique, called *blind prediction*, uses the training data to evaluate the prediction of the model on unseen data. Appendix B.3 describes how blind prediction is implemented in the context of B2BDC.

## 3.4 Optimization methods

The feasible set, by definition, is a collection of model parameters that agree with the prior knowledge on $x$ and the model-data constraints. Any parameter in $\mathcal{F}$ is a viable option based on a deterministic view of uncertainty. In some applications, it becomes intractable to evaluate an underlying model with many samples from $\mathcal{F}$. In these problems, it's useful to determine a representative parameter from $\mathcal{F}$. However, we recognize and caution that a single model parameter will not characterize the uncertainty exhibited by the set $\mathcal{F}$. The following optimization methods have been proposed to find an optimal point subject to model parameters in $\mathcal{F}$.

One method, referred to as the LS-F method [63], employs the following objective function,

$$\phi_{LS} = \sum_{e=1}^{N} [w_e(M_e(x) - y_e)]^2, \tag{3.5}$$

where $w_e$ is the statistical weight and $y_e$ is the experimental measurement of the $e$-th QOI. The model parameter that minimizes this objective, i.e., $\min\limits_{x \in \mathcal{F}} \phi_{LS}$, is called the LS-F parameter. A non-linear optimization solver (MATLAB's *fmincon* [64]) can be used to obtain a solution to Eq. 3.5.

Another optimization method, named 1N-F [57] seeks the model parameter $x \in \mathcal{F}$ that is in an $\ell_1$ sense nearest to the nominal parameter value $x_{nom}$. This criterion might be chosen because, in some examples, the nominal parameter value has a special meaning. The nominal value could be associated with a scientific community's agreed-upon value. If the nominal value is infeasible, then the smallest deviation from this value would be of interest. The 1N-F method uses the following objective function,

$$\phi_{\mathrm{1N}} = ||x - x_{nom}||_1 = \sum_{i=1}^{n} |x_i - x_{nom,i}|, \tag{3.6}$$

where $x_i$ is the $i$-th component of the model parameter vector $x$. The model parameter that minimizes this objective, $\min\limits_{x \in \mathcal{F}} \phi_{1N}$ is the 1N-F parameter.

## 3.5 Validation Workflow

The PrIMe Data Warehouse and the B2BDC methodology can be integrated into an end-to-end validation workflow. This workflow provides a semi-automated and systematic procedure for determining the validity of an underlying model. Domain scientists determine the choice of the underlying model and QOIs from the PrIMe Data Warehouse; therefore, the process is semi-automated. A collection of validation data from the PrIMe Data Warehouse is used to first develop and then constrain surrogate representations of QOIs from an underlying model. The B2BDC methodology then assesses the validity, i.e., consistency, of the models and data. This process is shown in Figure 3.1 and can be described in the following steps:

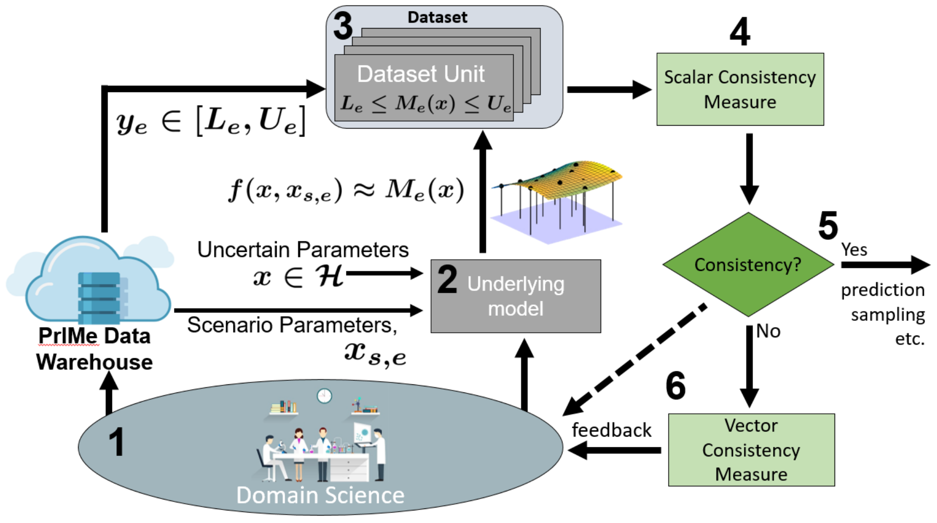


Figure 3.1: Model validation workflow that combines B2BDC methodology with the PrIMe Data Warehouse.

1. Domain science proposes an underlying computational model and a set of validation data, stored in the PrIMe Data Warehouse

2. For each QOI, the underlying model is evaluated using a scenario parameter obtained from the PrIMe Data Warehouse, and a polynomial surrogate model is developed

3. Each surrogate model is then constrained by the associated experimental bounds taken again from the Data Warehouse

4. The scalar consistency measure is calculated for the dataset

5. If the measure is consistent, then the model and data are in agreement

6. If the measure is inconsistent, feedback is returned to domain scientists to examine the source of the model-data disagreement (e.g., employing sensitivity to the consistency measure [56])

    a) Alternatively, the vector consistency measure can be evaluated to provide more refined feedback

The presented workflow is an end-to-end cycle to determine the validity of a model. An inadequate model can help inform scientists of the potential deficiencies or incompatibilities with the experimental data by following the feedback loop. The validation workflow then repeats with a new and improved model. This application of feedback is illustrated in Chapters 5 and 6, which helped identify incompatible data and guide the addition of new physics to a model. The validation workflow is central to the analysis conducted in this dissertation as it brings together tools for UQ analysis with access to an organized database of validation data.

## 3.6 Summary

The Bound-to-Bound Data Collaboration methodology was examined in this chapter for determining the consistency of an underlying model with experimental data. The scalar consistency measure determines if a feasible set of model parameters is empty or not. If the feasible set is empty, then there is disagreement between the model, experimental bounds, and the prior domain. Alternatively, if the feasible set is non-empty, the models and data agree over the domain. The B2BDC validation workflow is an end-to-end procedure for systematically determining and quantifying the agreement (or disagreement) between models and data. Three systems will be examined in this dissertation that use the validation workflow: an $H_2/O_2$ system, a reduced char oxidation model, and a semi-empirical quantum chemistry model, wherein all cases, experimental data comes from the PrIMe Data Warehouse. Before considering these examples, in the next chapter, I will develop a novel strategy for surrogate modeling that harnesses the scalar consistency measure.

# Chapter 4

# Developing piecewise surrogate models

The B2BDC methodology uses polynomial surrogate models in order to obtain provable optimization bounds. However, not all QOIs can be accurately characterized by a polynomial model. To improve the accuracy of a fixed-degree polynomial model, the prior domain will be decomposed into multiple disjoint partitions, each of which is represented by a polynomial model. This approach, known as piecewise modeling, can become computationally demanding if the underlying model is costly to assess. In this chapter, a novel strategy is developed using experimental data during the fitting procedure. The performance of this strategy is demonstrated in two examples, where the total number of function evaluations necessary to develop a piecewise surrogate model is reduced.

## 4.1 Introduction

The B2BDC methodology, as discussed in Chapter 3, employs polynomial surrogate models as an approximation of the underlying model for UQ. In previous studies [34, 63, 65, 66], the selected QOIs could be accurately represented by a quadratic surrogate model within their respective domains. Unfortunately, more recent applications of the B2BDC methodology [67, 68], which are discussed in Chapters 6 and 7, had encountered difficulties using a quadratic surrogate model. Non-polynomial QOIs and large prior domains resulted in substantial surrogate model fitting error, as defined in Appendix B.2. If a surrogate model has a large fitting error, then the surrogate is an inadequate representation of the underlying model over the specified domain. Any subsequent analysis is questionable, as the fitting error, which is propagated forward, is non-negligible. Ideally, the fitting error should be small enough that the UQ analysis will not change, regardless of whether or not it is propagated forward. However, when the fitting error is large, the analysis can be greatly affected.

An accurate representation of the underlying model may not be possible with a single polynomial model. We will consider developing piecewise surrogate models to approximate $f(x, x_{s,e})$, as initially described by Feeley [60]. Piecewise surrogate models are a collection of models that are developed on non-intersecting partitions of the prior domain. Collectively, these surrogate models can evaluate any value of the prior domain. Each *subdomain* (i.e., a single partition of the domain) is defined by a polynomial model. The collection of polynomial models can more accurately represent an underlying model over the prior domain by partitioning the domain into multiple disjoint subdomains. In the following section, the general strategy for piecewise modeling is outlined.

## 4.2 General strategy for piecewise modeling

Building a piecewise surrogate model can be a computationally demanding task, given the necessity for enough samples in each subdomain to construct a polynomial surrogate model. The general strategy for constructing piecewise surrogate models is described in Algorithm 1. In effect, the recursive algorithm will continue to partition a domain until each subdomain contains a polynomial surrogate model that has a fitting error less than a specified error tolerance. Algorithm 1 has a function, `fitOnDomain`, which handles many of the computational tasks; specifically, generating training samples in the domain, evaluating the underlying model for the training samples, fitting the surrogate model, and estimating the error. In most real-world applications, the bulk of the computational time is spent evaluating the underlying model for a set of training samples. On Line 4 of Algorithm 1, the function `partition` takes as input a domain, $\mathcal{D}$, and returns two disjoint subdomains, where $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$ and $\mathcal{D}_1 \cap \mathcal{D}_2 = \varnothing$. In Section 4.3, we discuss the details of this partitioning.

Constructing piecewise surrogate models is an effective strategy for developing models that fit within some specified error tolerance through domain partitioning. An accurate representation over the entire prior domain, $\mathcal{D}$, can be obtained. However, in the B2BDC framework the region of interest is the feasible set, which is only a subset of the prior domain. In practice, the feasible set is much smaller in volume compared to the prior domain. Therefore, significant computational effort is spent developing surrogates over the prior domain when only a fraction of volume is used in the analysis.

A new strategy for piecewise modeling can be formulated by incorporating each QOI's associated set of experimental bounds and the B2BDC's scalar consistency measure. The general piecewise modeling algorithm can be revised such that it focuses its computational efforts on developing accurate surrogate models for the feasible set. In other words, the new strategy will no longer partition a domain once it is proven incompatible with the data. This strategy unifies both the fitting procedure and experimental data through B2BDC's scalar consistency measure. In the following section, the details of this new strategy are described.

**Algorithm 1** General strategy for constructing piecewise surrogate models

```
1: procedure TREE = PIECEWISE(tree, @functionHandle, errTol,, $\mathcal{D}$);
   ▷ tree: array containing surrogate models objects (initially empty)
   ▷ @functionHandle: handle to underlying QOI model
   ▷ errTol: maximum tolerance for the surrogate model error
   ▷ $\mathcal{D}$: prior domain
2:     [model, error] = fitOnDomain(@functionHandle, $\mathcal{D}$);
       ▷ Returns a polynomial surrogate model object and the estimated error over the
   domain $\mathcal{D}$
3:     if error > errTol then
4:         [$\mathcal{D}_1$, $\mathcal{D}_2$] = partition($\mathcal{D}$);
           ▷ Partition $\mathcal{D}$ into two disjoint subdomains following a heuristic
5:          tree = piecewise(tree, @functionHandle, errTol, $\mathcal{D}_1$);
6:          tree = piecewise(tree, @functionHandle, errTol, $\mathcal{D}_2$);
7:     else
8:         tree = [tree; model];                         ▷ Save model object
9:     end if
10: end procedure
```

## 4.3 Novel strategy for piecewise modeling

In the B2BDC framework, each QOI model is constrained by an associated set of experimental bounds. The general strategy for piecewise modeling, shown in Section 4.2, made no use of the available experimental data during its fitting strategy. Here, the general piecewise modeling strategy and experimental data are combined to create a more efficient fitting strategy for sampling regions of interest.

This novel strategy is outlined in Algorithm 2, where the code (highlighted in red) has been added from the general piecewise modeling strategy. In Line 3, the function `checkScalarConsistency` is called after a surrogate model is constructed. This function calculates the scalar consistency measure (Eq. 3.3) to determine if the model and experimental bounds are in agreement across the model's domain. If the model and data are consistent, the boolean variable `consistent` is set to TRUE. On Line 4, the algorithm checks if the error is greater than the specified tolerance; if the domain is consistent, and if both statements are true, then the algorithm will continue to partition the current domain. This procedure will terminate once the surrogate model fitting error is less than the specified tolerance or once a subdomain is proven inconsistent.

The goal of this novel strategy is to obtain an accurate representation of the feasible set utilizing the fewest possible subdomains. Each subdomain requires a set of training samples to fit a polynomial surrogate model; few subdomains implies fewer evaluations of the (often computationally expensive) underlying model. It is challenging to determine the optimal partitioning of a domain such that the fewest number of subdomains are accurately characterized by a polynomial model. Initially, a coarse representation of the underlying model is available through the finite training samples. As additional training samples are evaluated, a more refined representation of the underlying model is obtained, which can help to identify polynomial behaved regions in the parameter space. Two partitioning strategies are discussed in the following section.

**Algorithm 2** Novel strategy for constructing piecewise surrogate models

```
1: procedure TREE = PIECEWISE(tree, @functionHandle, errTol,, $\mathcal{D}$, $[L, U]$);
     ▷ tree: array containing surrogate models (initially empty)
     ▷ @functionHandle: handle to underlying QOI model
     ▷ errTol: maximum tolerance for the surrogate model error
     ▷ $\mathcal{D}$: prior domain
     ▷ $[L, U]$ are experimental bounds of the @functionHandle
2:     [model, error] = fitOnDomain(@functionHandle, $\mathcal{D}$);
         ▷ Returns a polynomial surrogate model object and the estimated error over the domain $\mathcal{D}$.
3:     consistent = checkScalarConsistency(model, error, $[L, U]$)
4:     if error > errTol && consistent then
5:         [$\mathcal{D}_1$, $\mathcal{D}_2$] = partition($\mathcal{D}$);
             ▷ Partition $\mathcal{D}$ into two disjoint subdomains following a heuristic
6:         tree = piecewise(tree, @functionHandle, errTol, $\mathcal{D}_1$, $[L, U]$);
7:         tree = piecewise(tree, @functionHandle, errTol, $\mathcal{D}_2$, $[L, U]$);
8:     else
9:         tree = [tree; newModel];
10:    end if
11: end procedure
```

## Partitioning

*A priori*, is it not evident where a domain should be partitioned such that the resulting subdomains can be adequately represented by the given fidelity of a surrogate model. One of the simplest approaches is to decompose a domain following coordinate-aligned (axis-aligned) splits. Domains could also be decomposed using a multivariate hyperplane. For this dissertation, however, we will only consider coordinate-aligned partitions.

The process of partitioning a domain into two disjoint subdomains can be represented as a binary tree (shown in Fig. 4.1). The root node is the prior domain. It is decomposed into two child nodes. The general and novel strategies for piecewise surrogate modeling traverse the binary tree following a depth-first search [69]. A depth-first search builds out (and traverses) the left-most nodes of the binary tree first before considering the rightward nodes. This approach is illustrated in Figure 4.1. At iteration 0, Node 1 (i.e., the prior domain) is partitioned into two domains: Node 2 and Node 3. As the algorithm progresses, it begins by partitioning domains under Node 2. The domain associated with Node 3 is not considered until all subdomains of Node 2 are resolved.

The choice of which coordinate (model parameter) to split and where the partition occurs is dictated by a *partitioning rule*. A poorly chosen partitioning rule can lead to numerous subdomains which, as discussed before, requires many evaluations of the underlying model. A simple, albeit naïve, partitioning rule is to take the model parameter with the largest uncertainty range and partition it in half. This rule is referred to as the *uncertainty-based partitioning rule*. To illustrate, consider the following example where $\mathcal{D}$ is the prior domain and $x = [x_1; x_2]$,

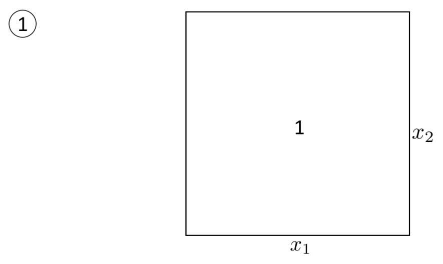
$$\mathcal{D} = \{x \in \mathbb{R}^2 : 0 \leq x_1 \leq 1, \;\; 0 \leq x_2 \leq 0.5\}. \tag{4.1}$$

Since $x_1$ has a larger uncertainty range than $x_2$, $x_1$ is selected as the parameter to partition. The selected model parameter will be divided in half, resulting in two disjoint subdomains, $\mathcal{D}_1$ and $\mathcal{D}_2$, where,

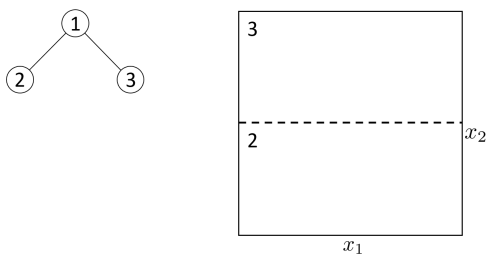
$$\begin{aligned}\mathcal{D}_1 &= \{x \in \mathbb{R}^2 : 0 \leq x_1 \leq 0.5, \;\; 0 \leq x_2 \leq 0.5\}\\ \mathcal{D}_2 &= \{x \in \mathbb{R}^2 : 0.5 < x_1 \leq 1, \;\; 0 \leq x_2 \leq 0.5\}.\end{aligned}$$

We demonstrate the performance of the uncertainty-based partitioning rule on two examples in Section 4.4. A less naïve rule has been proposed by Feeley [60], which begins by hypothetically splitting a model parameter in half, forming two subdomains, $\mathcal{D}_1$ and $\mathcal{D}_2$. A quadratic surrogate model is constructed for each of the two hypothetical subdomains. Coefficients for the quadratic model were obtained by minimizing the 2-norm of the training error. The test error was then estimated for each quadratic model. This process is then repeated for all model parameters. The hypothetical partition that yielded the smallest fitting error is selected as the model parameter by which the domain will be partitioned. This rule will be referred to as the *error-based partitioning rule*.

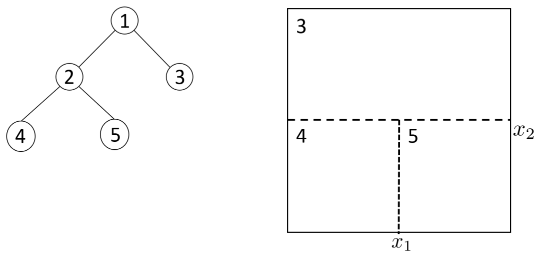
The error-based partitioning rule considers all possible model parameters and selects the best partition at the current iteration. We define "best" as the partition that yields the smallest estimated fitting error. This parameter is chosen because one of its two subdomains has the smallest fitting error compared to all other partitions. The motivation for selecting a model parameter based on the smallest fitting error is as follows. Subdomains with the smallest error may be less than the specified threshold; therefore, no further partitions would be necessary. If the fitting error is still above the threshold, potentially a fewer number of additional partitions would be required before a surrogate model is below the error threshold.



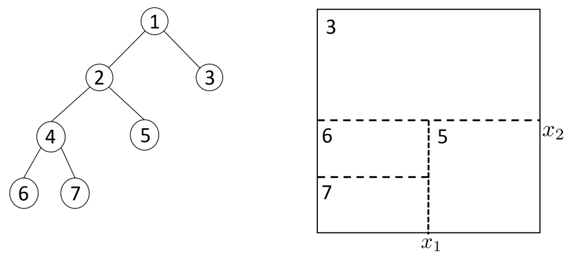
(a) Iteration 0



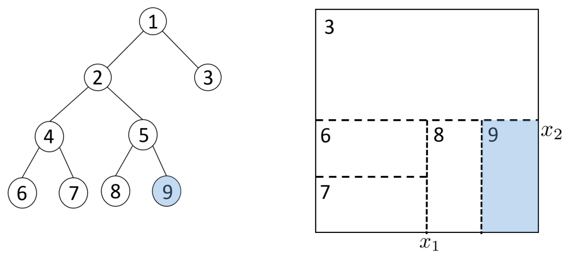
(b) Iteration 1



(c) Iteration 2



(d) Iteration 3



(e) Iteration 4

Figure 4.1: Example of a two-dimensional prior domain being partitioned into multiple, disjoint subdomains. Multiple iterations of the decomposition are shown with the associated binary tree. Each node of the tree corresponds with a domain. Parent nodes are partitioned into two children nodes. Based on Algorithm 1, if a surrogate model fits within the specified error tolerance, the domain will not be partitioned. The algorithm and tree traversal follows a depth-first search. In Figure 4.1(e), Node 3 will be considered by the algorithm after Node 9 is assessed.

Other parameter choices, by contrast, could require a larger number of partitions, as the error was larger.

## Combined information

The novel strategy for piecewise modeling integrates experimental data during the fitting procedure through B2BDC's scalar consistency measure. When considering a single QOI model, the consistency measure verifies that a single QOI model is in agreement with its experimental data, i.e., *self-consistency.* In other words, are there model parameters in the domain, such that a single QOI model can predict within the specified uncertainty bounds? If there are parameters that satisfy these constraints, then the subdomain is consistent.

In practice, the feasible set $\mathcal{F}_e$, formed by an individual model-data constraint, can be large in volume. It could take a considerable number of partitions before a subdomain does not contain part of the feasible set. The scalar consistency measure, as it was formulated in Eq. 3.3, is used to determine if a set of model parameters exists in the prior domain, where *all* QOI models and data are in agreement. The volume of the feasible set in the prior parameter space is significantly reduced when taking into account an increasing number of QOI models. Considering an arbitrary subdomain, a feasible set with a smaller volume is less likely to intersect the subdomain than a feasible set with a larger volume.

In many physical models, a single model evaluation can produce multiple model outputs. Each output can be associated with a different QOI, e.g., a species concentration profile. For such cases, it is recommended to construct all QOI models on the same subdomain and then evaluate the scalar consistency measure using all model-data constraints. There are various degrees in which one can validate a domain. The scalar consistency measure can be assessed for a single QOI, all available QOIs, or a particular batch of QOIs, e.g., only QOI models that have a fitting error of less than 10%. In this dissertation, we will determine the validity of a subdomain by using all available QOI models.

In this section, we discussed a few partitioning rules that are used to divide a domain. The error-based partitioning rule was based on the error assessment of a single QOI. When considering multiple QOI models, it is unclear how to select the "best" model parameter to partition. Each QOI may determine that a different model parameter is best to split. To clarify any ambiguity in this assessment, a single QOI will be assigned as the *template QOI.* The partitioning rule will follow the assigned template QOI to determine which model parameter to split. A subdomain will no longer be partitioned if the domain is inconsistent, or the template QOI's error (and all other QOI models) are less than the specified threshold.

The choice of template QOI and partitioning rule is problem-specific. We aim to reduce surrogate model fitting error; therefore, a natural choice for the template QOI is to use the QOI model that has the minimum fitting error over the prior domain. In the following section, a few examples are provided to demonstrate the novel piecewise modeling strategy. The piecewise modeling code used in this dissertation is available at https://github.com/oreluk/B2BDC-PiecewiseModels, which uses the B2BDC toolbox [70]. This code is also available in Appendix C.

## 4.4 Toy examples

In this section, two toy examples are used to illustrate the efficacy of the novel strategy for piecewise modeling and how the choice of partitioning rule affects the total number of subdomains. In the first example, a non-polynomial behaved test function is represented using piecewise quadratic surrogate models. The second example considers a chemical kinetics model with two QOIs to demonstrate how multiple QOIs can reduce the overall number of required subdomains. Both examples are demonstrations of how the piecewise modeling strategy can be applied in practice. This strategy is later applied in Chapter 6 to evaluate the validity of a coal combustion model.

### Branin function example

The Branin function [2] is a two-dimensional, non-quadratic test function, that will serve as a simple example for visualizing the piecewise modeling approach. We will show how a domain can be partitioned and how the proposed piecewise modeling strategy can efficiently develop surrogate models on consistent subdomains. The Branin function, $M(x_1, x_2)$, will be defined as:

$$M(x_1, x_2) = (x_2 - bx_1^2 + cx_1 - 6)^2 + 10(1 - t)\cos(x_1) + 10, \tag{4.2}$$

where $b = \frac{5.1}{4\pi^2}$, $c = 5\pi$, and $t = \frac{1}{8\pi}$. There are two uncertain model parameters, $x_1$ and $x_2$, that are part of the prior domain, $\mathcal{H}$, where $x_1, x_2 \in [1, 10]$. Figure 4.2 shows the Branin function over $\mathcal{H}$. The Branin function is constrained by the experimental bounds, $M(x_1, x_2) \in [9, 15]$. The resulting feasible set is shown in red.

A single quadratic model is insufficient in adequately representing $M(x_1, x_2)$ over the entire specified domain. Figure 4.3 shows a quadratic surrogate model in light-red compared to the Branin function in solid-blue. The *maximum relative error* was estimated via 10-fold cross validation, and was 440%. The maximum relative error is defined in Eq. B.2. In an effort to reduce this fitting error, the prior domain will be partitioned following one of the proposed partitioning rules.

#### Choice of partitioning rule

A partitioning rule dictates which model parameter to split at each iteration of the piecewise modeling algorithm. This directly affects the total number of subdomains required to fit a quadratic surrogate model within a specified error tolerance. One of the proposed partitioning rules had been based on the uncertainty length of the model parameters. This rule, named the uncertainty-based partition rule, would divide a model parameter with the largest uncertainty range in half, creating two subdomains.

For this example, the error tolerance will be set to 0.5% of the maximum relative error. Using the general strategy for piecewise modeling, shown in Algorithm 1, a subdomain will

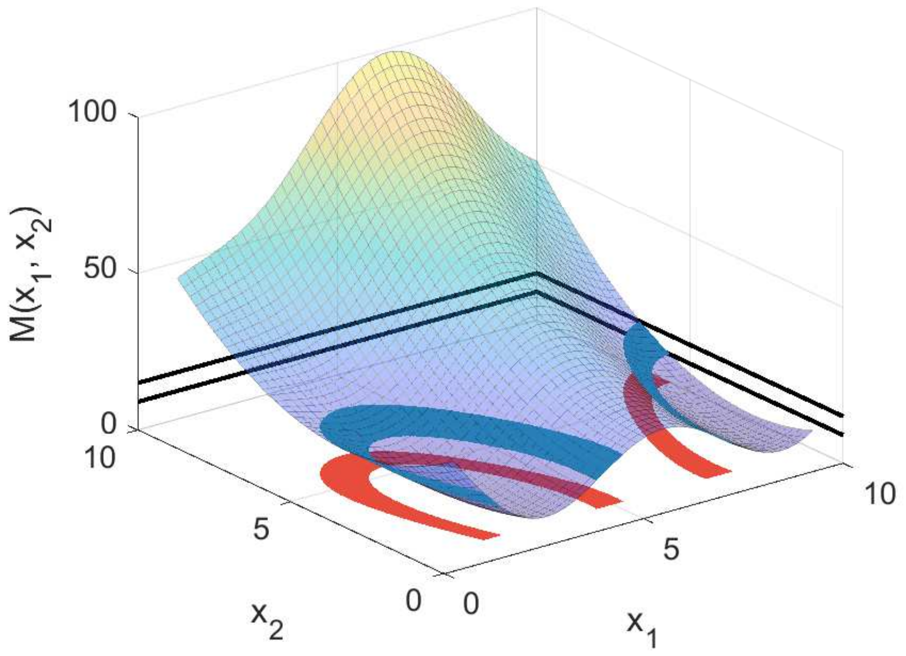


Figure 4.2: Transparent surface is the Branin function over the prior domain $x_1, x_2 \in [1, 10]$. The solid black lines are the experimental bounds $[9, 15]$. The solid blue region is where the model, $M(x_1, x_2)$ evaluates within experimental bounds. The red region is the feasible set, $\mathcal{F}$.

cease being partitioned when the estimated error is less than the tolerance. Figure 4.4(a) shows the 128 subdomains created using the general strategy. Each of the 128 subdomains was defined by a quadratic surrogate model by generating eight times more training samples than the number of coefficients. For a quadratic model, the number of coefficients, $n_{\text{coef}} = (n_{\text{param}} + 2)(n_{\text{param}} + 1)/2$, where the number of uncertainty model parameters, $n_{\text{param}}$ is equal to 2. Each of the 128 quadratic surrogate models has a maximum relative test error of less than 0.5%.

This result can be compared to the novel strategy for piecewise modeling, shown in Figure 4.4(b). A subdomain will cease being partitioned when the estimated error is less than the specified tolerance or the domain is proven incompatible with the experimental data. Using this strategy required only 71 subdomains to be developed. Subdomains that were proven inconsistent with the experimental data are shown in grey. Superimposed in Figure 4.4(b) is the feasible set (shown in red) from Figure 4.2. Each of the subdomains in white still fit within the specified error tolerance and collectively contain the feasible set. The novel strategy for piecewise modeling greatly reduced the total number of subdomains required when using the uncertainty-based partitioning rule.

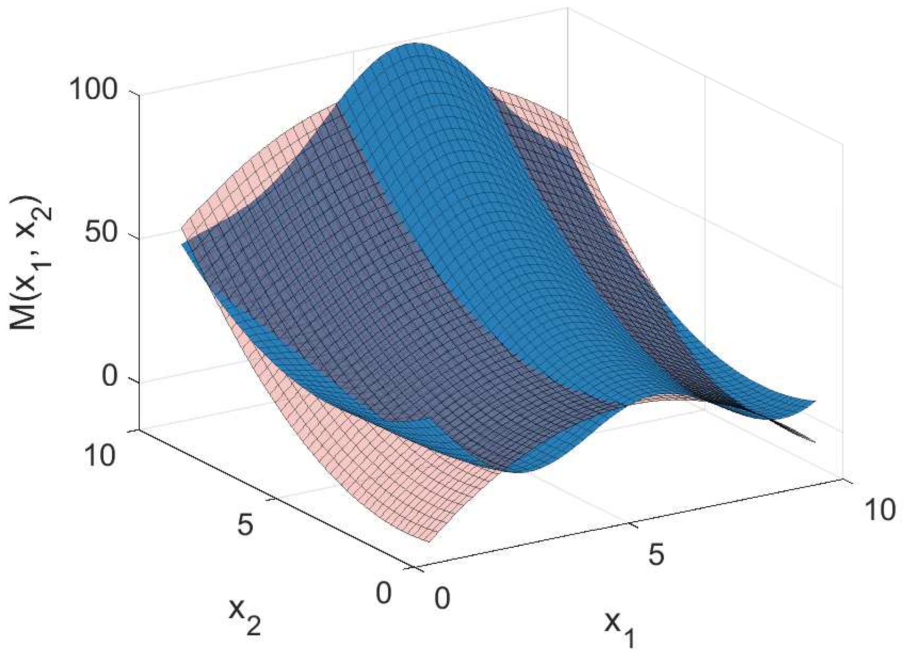


Figure 4.3: The Branin function (shown in blue) compared to a quadratic surrogate model (shown in light-red). The quadratic surrogate model was trained by minimizing the least-squares error across 120 training samples. The test error was estimated by 10-fold cross-validation, where the maximum relative error across the folds was 440%.

The uncertainty-based partitioning rule is easy to implement; however, it is not a well-motivated rule. To illustrate, consider the error-based partitioning rule for the Branin function. The error-based rule chooses a model parameter that yields the smallest estimated surrogate model error, details of which are discussed in Section 4.3. Figure 4.5 compares both the general and novel strategies for piecewise modeling following the error-based partitioning rule. Figure 4.5(a) shows the general strategy, where 44 subdomains were created. For each partition, the model parameter $x_1$ was selected to split, as it would result in subdomains with smaller fitting errors than $x_2$. This decision to always split $x_1$ can be better understood by considering Figure 4.2. For any constant $x_2$, the Branin function has an oscillatory behavior, due to the term, $\cos(x_1)$ in Eq. 4.2. Subdomains are fit with a quadratic surrogate model; therefore, any partition perpendicular to the $x_2$ coordinate would likely result in large errors due to the non-quadratic model response. Visualization of the quadratic surrogate models, fit over all subdomains at various iterations of the piecewise algorithm, are seen in Figure 4.6.

The novel strategy for piecewise modeling using an error-based partitioning rule is shown in Figure 4.5(b), where 43 subdomains were developed. A fewer number of

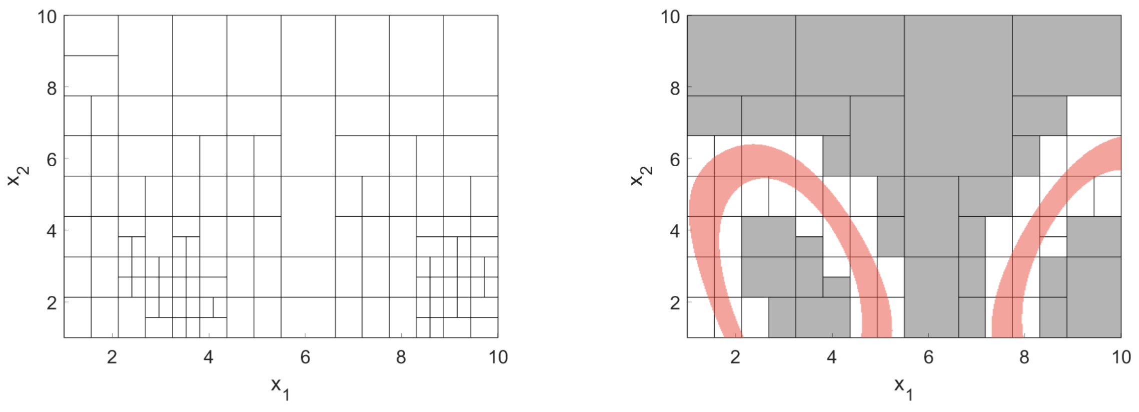


Figure 4.4: Comparing strategies for piecewise modeling using an uncertainty-based partitioning rule for the Branin function. Panel a: The general strategy for piecewise modeling resulted in 128 subdomains. Each subdomain has an associated quadratic surrogate model with less than 0.5% maximum relative error. Panel b: The novel strategy for piecewise modeling utilizing B2BDC's scalar consistency measure. Domains that are proven inconsistent are shown in grey. In total, only 71 subdomains were constructed. All consistent domains (shown in white) have an associated quadratic surrogate model with less than 0.5% maximum relative error.

Table 4.1: Number of subdomains for each partitioning rule and strategy to represent the Branin function with quadratic surrogate models

| Partitioning rule | General strategy | Novel strategy |
|---|---:|---:|
| Uncertainty-based | **128** | **71** |
| Error-based | 44 | 43 |

subdomains were necessary when using the novel strategy, as compared to the general strategy; however, the reduction in the number of subdomains was minimal (44 vs. 43). Both strategies and partitioning rules are summarized in Table 4.1. The error-based partitioning rule resulted in far fewer subdomains for both piecewise modeling strategies. Exploring the various partitioning rules is outside the scope of this dissertation; however, we have demonstrated that choice of partitioning rule can greatly affect the overall number of subdomains required. In the following section, we examine a simple chemical kinetics system and show how multiple QOIs are used to further reduce the number of subdomains.

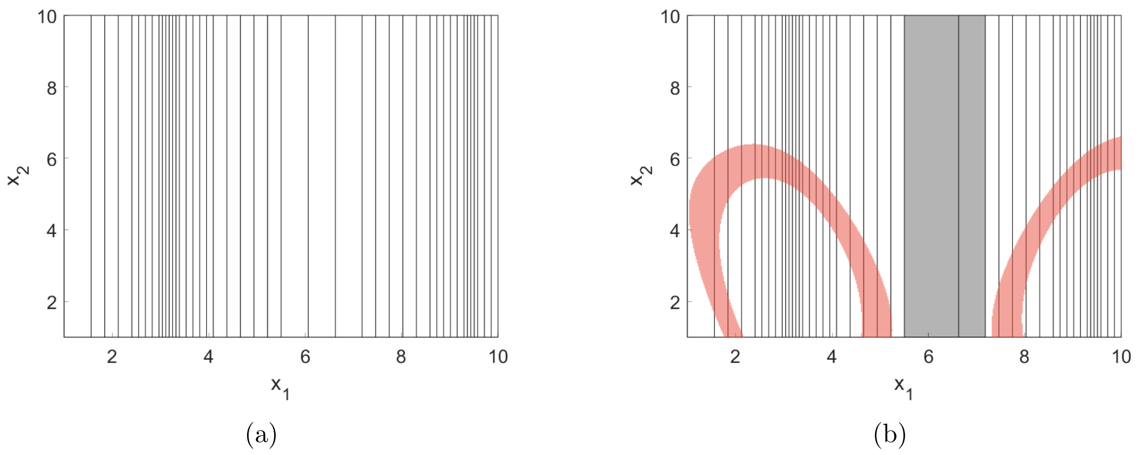
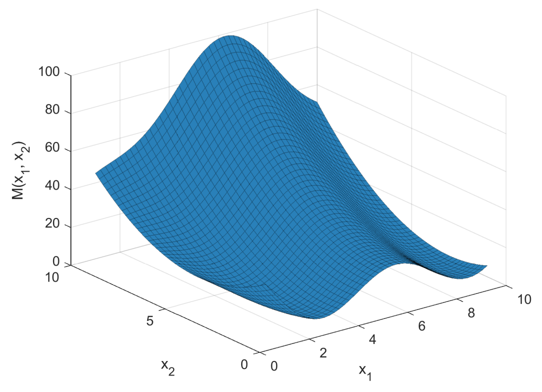
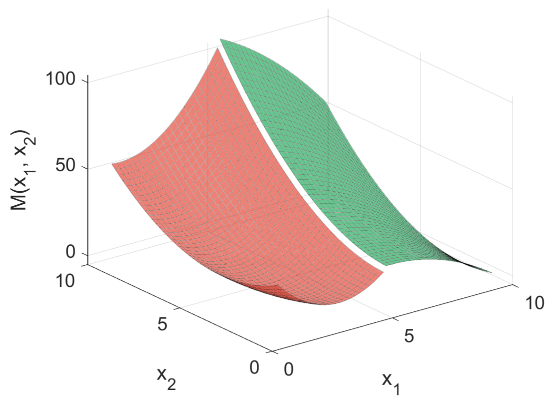


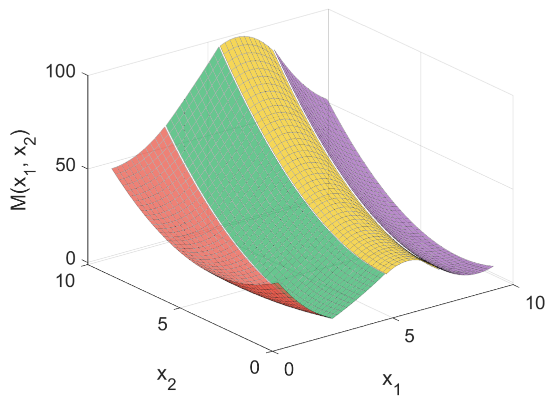
Figure 4.5: Comparing piecewise modeling strategies using an error-based partitioning rule on the Branin function. Panel a: The general strategy for piecewise modeling resulted in 44 subdomains. Panel b: The novel strategy for piecewise modeling utilizing B2BDC's scalar consistency measure resulted in 43 subdomains. Domains that are proven inconsistent are shown in grey. All consistent domains (shown in white) have an associated quadratic surrogate model with less than 0.5% maximum relative error.



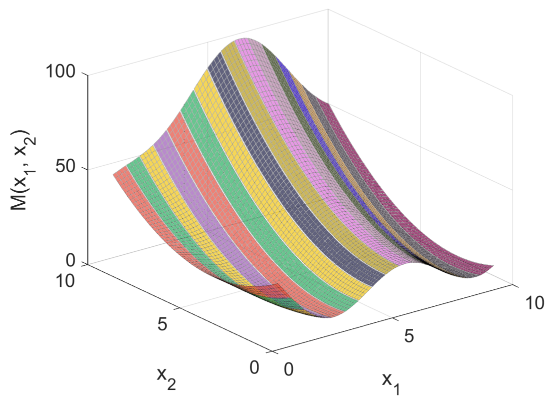
(a) Branin function



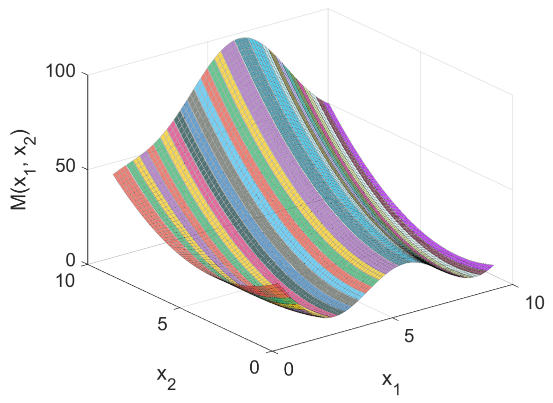
(b) 2 subdomains



(c) 4 subdomains



(d) 16 subdomains



(e) 32 subdomains

Figure 4.6: Visualization of the various iterations of the piecewise modeling strategy on the Branin function using an error-based partitioning rule. Each subdomain is represented by a quadratic surrogate model shown with a different color. Panel a: the Branin function. Panel b: 2 subdomains with a maximum relative error of 480%. Panel c: 4 subdomains with a maximum relative error of 320%. Panel d: 16 subdomains with a maximum relative error of 10%. Panel e: 32 subdomains with a maximum relative error of 2.8%.

## Chemical kinetics example

A simple chemical kinetics system composed of two consecutive, irreversible reactions is considered. This chemical kinetics example serves to illustrate how the combined information of multiple QOIs can be applied to reduce the size of the feasible set and thus, reduce the overall number of subdomains. Let us begin by considering the following chemical reactions,

$$\mathrm{A} \xrightarrow{k_1} \mathrm{B} \xrightarrow{k_2} \mathrm{C} \tag{4.3}$$

where A is the initial compound, which is converted into B at a rate of $k_1$. B is then converted into compound C at a rate of $k_2$. The rate of change in compounds are modeled by:

$$\begin{aligned} \frac{d[\mathrm{A}]}{dt} &= -\, k_1[\mathrm{A}] \\ \frac{d[\mathrm{B}]}{dt} &= k_1[\mathrm{A}] - k_2[\mathrm{B}] \\ \frac{d[\mathrm{C}]}{dt} &= k_2[\mathrm{B}] \end{aligned} \tag{4.4}$$

where [A], [B], and [C] are the concentrations of A, B, and C, respectively, at time $t$. $k_1$ and $k_2$ are the rate constants shown in Eq. 4.3. In the present example, $k_1$ and $k_2$ will be the uncertain model parameters, where $k_1 \in [0.5, 4]$ and $k_2 \in [0.2, 4]$ define the prior domain. There are two QOIs measured from the chemical kinetics system which will ultimately be compared to experimental data. The first QOI is the maximum concentration of compound B, which can be expressed as $M_1(k_1, k_2) = \max([\mathrm{B}]) = \frac{k_1}{k_2} e^{-k_1 t_{\max}}$. Experimental data for the first QOI is the interval $[0.45, 0.55]$. The second QOI, $M_2$, is the time to reach the maximum concentration of B, denoted as $t_{\max}$. The second QOI can be expressed by $M_2(k_1, k_2) = \frac{1}{k_2 - k_1} \ln \frac{k_2}{k_1}$, which is constrained by experimental bounds to $[0.55, 0.83]$. Figure 4.7 shows an example of the concentration profiles of [A], [B], and [C] as functions of time. The two QOIs, $M_1$ and $M_2$, are marked by the black cross, where $M_1$ is the value of the maximum concentration of [B] and $M_2$ is the time to reach the maximum concentration of [B].

Figure 4.8 shows a visualization of both QOIs across the prior parameter space. The feasible set associated with each QOI is highlighted in red. A single quadratic surrogate model is inadequate with respect to an error tolerance of 0.5% for representing either $M_1$ or $M_2$, where the maximum relative error was 6% and 18%, respectively. The feasible set $\mathcal{F}$, which is the intersection of parameters consistent with constraints for $M_1$ and $M_2$, is seen in Figure 4.9. Strategies for piecewise modeling were implemented on both $M_1$ and $M_2$ to evaluate the number of subdomains required to reach the goal of accurate surrogate models, i.e., less than 0.5% of the maximum relative error, on regions where the model and data agree.

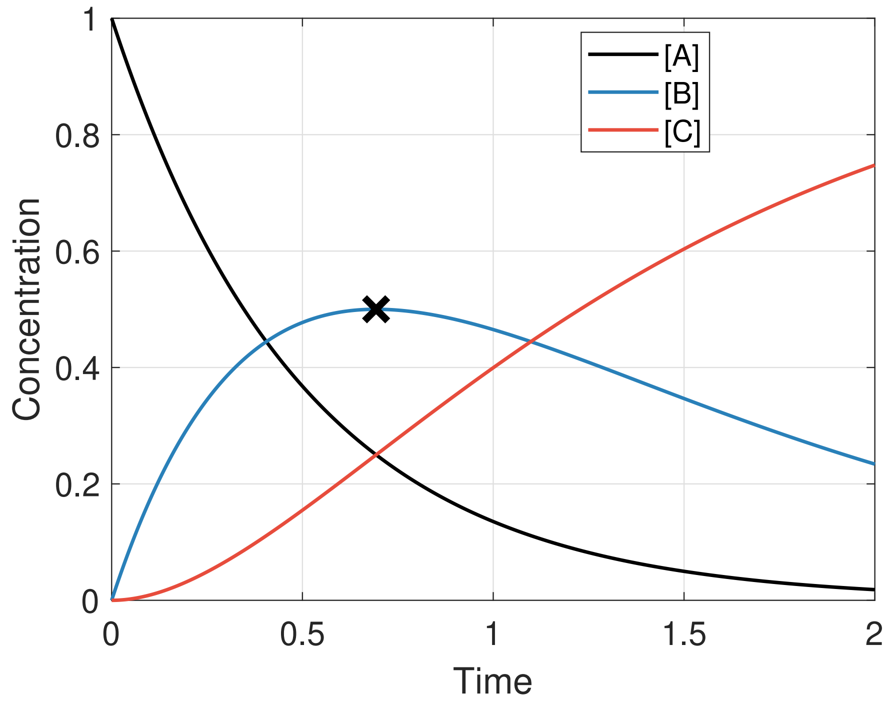
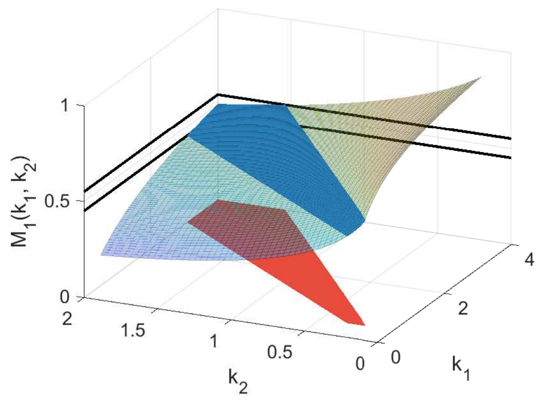


Figure 4.7: Concentration profiles of $[A], [B]$, and $[C]$ as a function of time. Initial concentration of $[A]_{t=0} = 1$ and $[B]_{t=0} = [C]_{t=0} = 0$. Rate constants where set as $k_1 = 2$ and $k_2 = 1$. The black cross is set at the maximum concentration of $[B]$, where $M_1(2,1) = 0.5$ and $M_2(2,1) = 0.693$.

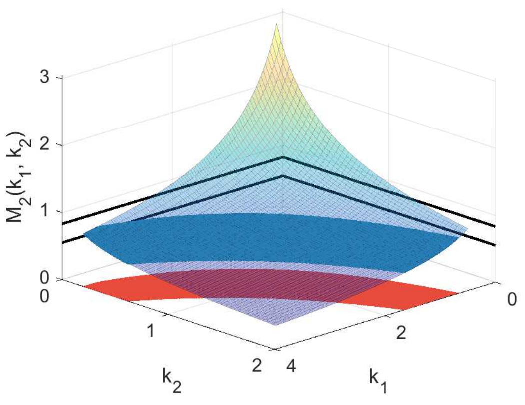
**Error-based partitioning rule**

In Section 4.4, the error-based partitioning rule led to a fewer number of subdomains compared to the uncertainty-based rule. Based on this result, we expect that the error-based partition rule will also provide fewer subdomains in the chemical kinetics example. The first QOI, $M_1$, using the general strategy for piecewise modeling, with a maximum relative error threshold of 0.5%, led to the construction of 16 subdomains. Recall that, the general strategy for piecewise modeling does not make use of the experimental data. The exact partitioning of the prior domain can be seen in Figure 4.10(a). The novel strategy for piecewise modeling, which uses the experimental data by calculating B2BDC's consistency measure, is shown in Figure 4.10(b), where the grey regions are provably inconsistent. In Figures 4.10(a) and 4.10(b), the white regions are defined by a quadratic surrogate model, which has a maximum relative error of less than 0.5%. For this example, using the novel strategy yielded two fewer subdomains.

The second QOI, $M_2$, is shown in Figure 4.11(a), where 41 subdomains were constructed using the general strategy for piecewise modeling. The novel strategy is shown in Figure 4.11(b), where only 29 subdomains were necessary — a reduction by twelve subdomains.



(a)



(b)

Figure 4.8: Response of $M_1$ and $M_2$ over the prior domain $k_1 \in [0.5, 4]$ and $k_2 \in [0.2, 4]$. Panel a: Surface shown is the maximum concentration of B, $M_1$. The black lines shown are the experimental bounds $[0.45, 0.55]$. The blue region is where the model, $M_1(k_1, k_2)$ evaluates within experimental bounds. The red region is the feasible set of parameters associated with $M_1$. Panel b: Surface shown is the time to reach a maximum concentration of B, $M_2$. The black lines shown are the experimental bounds $[0.55, 0.83]$. The blue region is where the model, $M_2(k_1, k_2)$ evaluates within experimental bounds. The red region is the feasible set associated with $M_2$.

Each subdomain was defined by a quadratic surrogate model by generating eight times more training samples than the number of coefficients in the surrogate model. When using the novel strategy, 576 evaluations of the underlying model were saved. In this example, each evaluation of the simulation was inexpensive. Thus, saving 576 runs was not that meaningful; however, when this strategy is applied to a larger-scale simulation, where each simulation takes many CPU-hours, these savings can be significant. Table 4.2 compares both strategies for piecewise modeling and partitioning rules.

Table 4.2: Number of subdomains for each partitioning rule and strategy to represent the two QOIs in the chemical kinetics example

| | Partitioning rule | General strategy | Novel strategy |
|---|---|---|---|
| $M_1(k_1, k_2)$ | Uncertainty-based | 20 | 16 |
| $M_1(k_1, k_2)$ | Error-based | 16 | 14 |
| | | | |
| $M_2(k_1, k_2)$ | Uncertainty-based | 43 | 29 |
| $M_2(k_1, k_2)$ | Error-based | 41 | 29 |

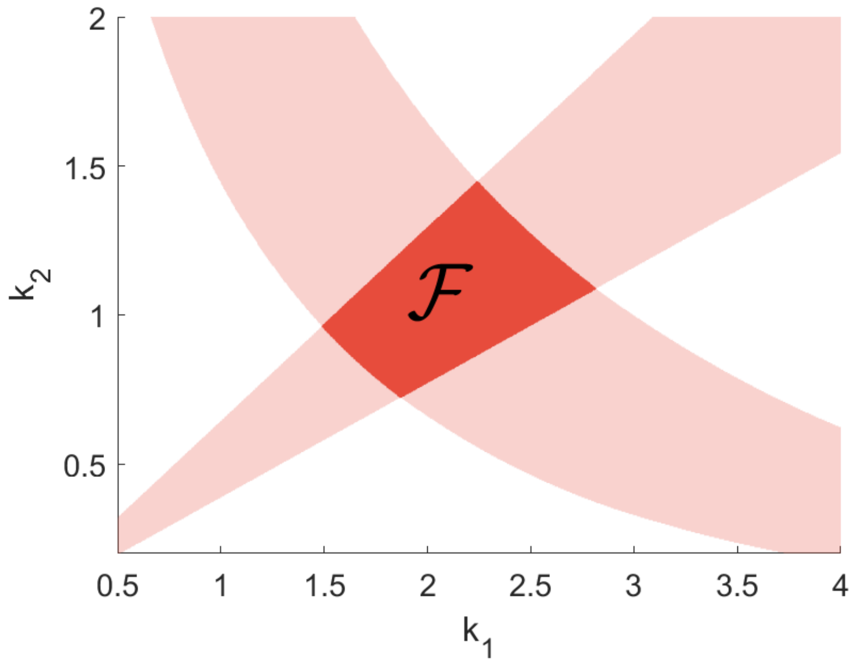


Figure 4.9: Feasible set of model parameters for the chemical kinetics example. Model parameters consistent with either $M_1$ or $M_2$ is shown in light red. The intersection of the light red regions is the feasible set, $\mathcal{F}$, shown in dark red.

## Consistency of a subdomain for all QOIs

When considering a single QOI, the scalar consistency measure only evaluates the compatibility of a single surrogate model with its associated experimental bounds, i.e., *self-consistency* [66]. Subdomains are consistent if there are model parameters that satisfy the imposed constraints (experimental bounds) on the QOI model, subject to the model parameters being in the subdomain. In practice, self-consistency is a weak constraint. The feasible set $\mathcal{F}_e$ is a large fraction of the prior domain $\mathcal{H}$; therefore, self-consistency is incapable of invalidating large volumes of the parameter space. The scalar consistency measure in its original formulation uses *all* QOI models. By adding additional constraints (in the form of more QOI models), the size of the feasible set is significantly reduced, as seen in Figure 4.9.

A model response for both $M_1$ and $M_2$ is obtained for each evaluation of the chemical kinetics model. Therefore, surrogate models for $M_1$ and $M_2$ can be built simultaneously without any additional model evaluations. With two QOI models, the strategies for piecewise modeling — specifically, the partitioning rule — become difficult to interpret. For this example, we will conduct the analysis twice, with $M_1$ and $M_2$ as the template QOI to

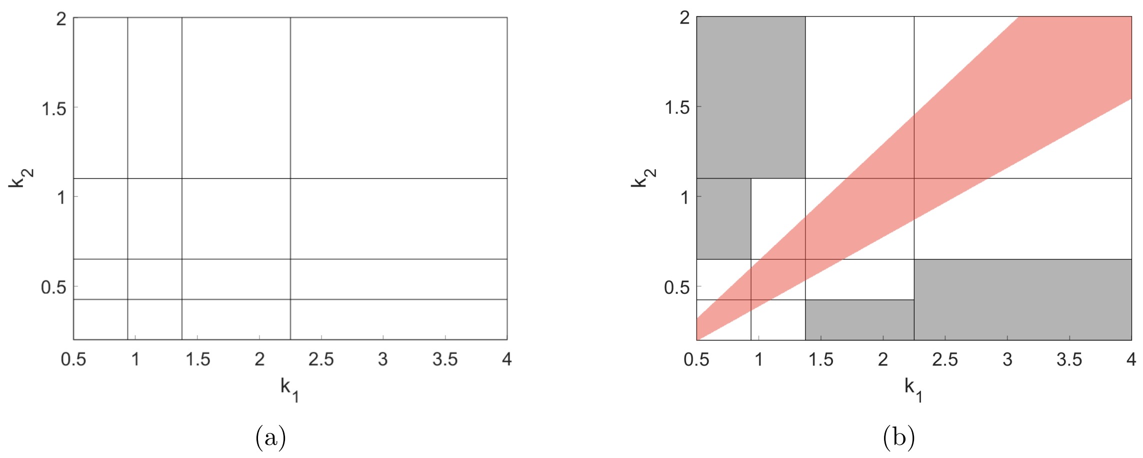


Figure 4.10: Comparing strategies for piecewise modeling of the first QOI, $M_1$, in the chemical kinetics example. The boundary for each subdomain is shown as the black lines where each subdomain was fit with a quadratic surrogate model. Each subdomain (shown in white) has an associated quadratic surrogate model with a maximum relative error of less than 0.5%. Panel a: The general strategy for piecewise modeling resulted in 16 subdomains. Panel b: The novel strategy for piecewise modeling resulted in 14 subdomains. Domains that are proven inconsistent are shown in grey. Shown in red is the feasible set associated with the experimental data for $M_1$.

examine its effect. Recall that, in the error-based partitioning rule, a model parameter is split based on a single QOI's error assessment. Therefore, for multiple QOIs, we designate one QOI as a "template" for all partitioning decisions.

First, $M_1$ was used as the template QOI to determine which model parameter to partition at each iteration. A quadratic surrogate model was developed for both QOIs over each subdomain, following the novel strategy in Algorithm 2. The scalar consistency measure was evaluated for all QOIs over each subdomain, and resulted in 11 subdomains. This result is illustrated in Figure 4.12(a), where, for each subdomain, there is a surrogate model for $M_1$ and $M_2$ that fits within the specified error tolerance. If we were to build a piecewise surrogate model for $M_1$ independently, it would require 16 subdomains. To exemplify this with regard to the number of samples, $4(n_{\text{param}}+1)(n_{\text{param}}+2)$ samples were used in each subdomain; thus, 768 samples are required to construct the 16 subdomains of $M_1$. Considering the approach described above, where the scalar consistency measure of the subdomain is calculated for all QOI models, only 528 samples were needed, a reduction of 31%. If we consider changing the template QOI from $M_1$ to $M_2$, 12 subdomains would be required, as shown in Figure 4.12(b). Building a piecewise model for $M_2$ independently required 41 subdomains to achieve a maximum relative error of less than 0.5% (Figure 4.11(a)). If we were to use $4(n_{\text{param}}+1)(n_{\text{param}}+2)$ samples per subdomain, 1,968 samples would be required for the 41 subdomains

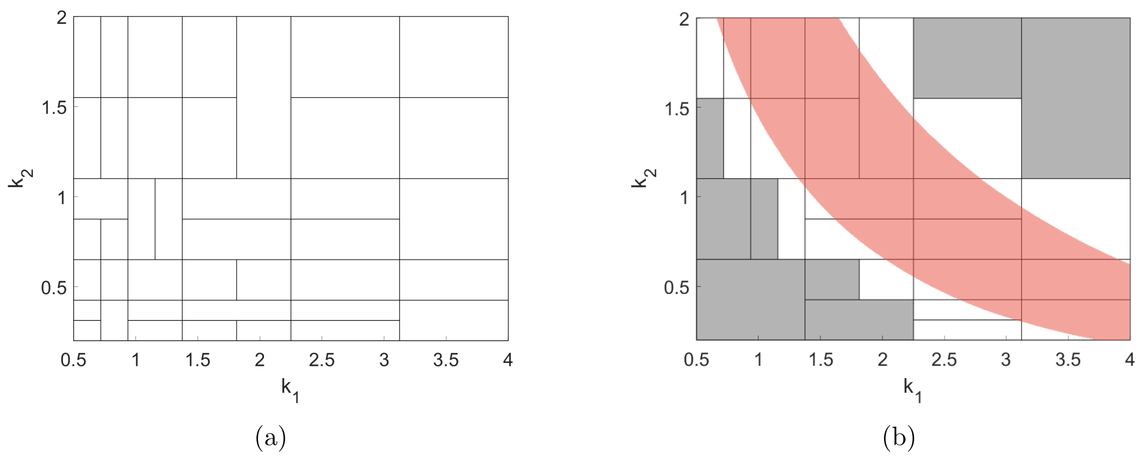


Figure 4.11: Comparing strategies for piecewise modeling of the second QOI, $M_2$, in the chemical kinetics example. The boundary for each subdomain is shown as the black lines where each subdomain was fit with a quadratic surrogate model. Each subdomain (shown in white) has an associated quadratic surrogate model with a maximum relative error of less than 0.5%. Panel a: The general strategy for piecewise modeling resulted in 41 subdomains. Panel b: The novel strategy for piecewise modeling resulted in 29 subdomains. Domains that are proven inconsistent are shown in grey. Shown in red is the feasible set associated with the experimental data for $M_2$.

of $M_2$. For the combined information approach described above, only 576 samples were needed, a reduction of 71%.

These results show that $M_1$ was a better choice for the template QOI. $M_1$ required fewer subdomains, therefore, fewer samples to obtain the same level of accuracy compared to $M_2$. However, in practice, it is not possible to compare these template QOI results. For this example, the maximum relative error over the prior domain was 6% for $M_1$ and 18% for $M_2$. We recommend that the template QOI is chosen based on the smallest fitting error over the prior domain, as discussed in Section 4.3.

## 4.5 Conclusion

The B2BDC methodology, discussed in Chapter 3, uses polynomial surrogate models in order to attain provable optimization bounds on prediction and consistency. Using such surrogate models to represent complex functions may lead to significant fitting errors. In this chapter, the discussed strategies for piecewise modeling retain these provable bounds from B2BDC, while widening the type of QOI models that can be accurately represented. In order to develop a more accurate surrogate model, however, more samples of the underlying model are required for either piecewise modeling strategy.

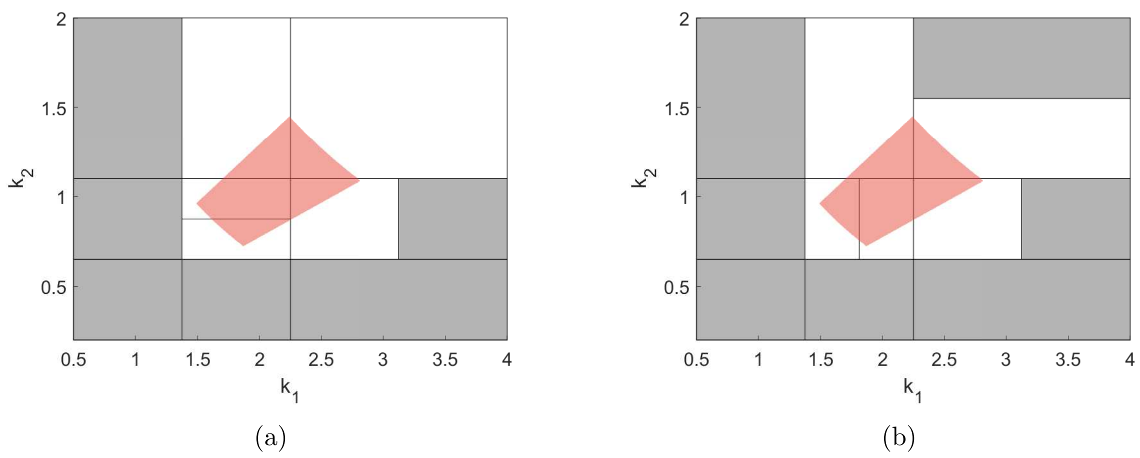


Figure 4.12: Novel strategy for piecewise modeling using multiple QOIs in the chemical kinetics example. Both QOI models were constructed on each subdomain. Domains shown in grey are proven inconsistent with the data. The feasible set was replotted on the domain in red. All consistent domains (shown in white) have a maximum relative error of less than 0.5% for both $M_1$ and $M_2$. Panel a: $M_1$ is used as the template QOI to guide the partitioning of the parameter space, resulting in 11 subdomains. Panel b: $M_2$ is used as the template QOI and 12 subdomains are constructed.

The general strategy for piecewise modeling was less efficient than the new strategy, creating many subdomains in regions of the parameter space that are not of interest to the analysis. A novel strategy was proposed, which develops an accurate representation in regions where the feasible set resides. If a subdomain is proven incompatible with the experimental data, the subdomain is neglected and no longer considered for partitioning by the algorithm. This novel strategy allows samples (i.e., computational effort) to be guided towards these regions of interest. Two toy examples were explored to demonstrate that the choice of partitioning rule and use of multiple QOI models to evaluate the scalar consistency measure can reduce the overall number of subdomains constructed. Increasing the dimensionality in both the uncertain model parameters and QOIs should demonstrate the efficacy of the novel strategy for piecewise modeling, where large portions of the prior domain can be proven inconsistent with the data, assuming the surrogate model fitting errors are accurate. In Chapter 6, a higher-dimensional example is examined that uses the developed piecewise modeling strategy.

# Chapter 5

# Validating an $H_2/O_2$ reaction model

This chapter shows an application of the validation workflow (described in Chapter 3) using an $H_2/O_2$ kinetics mechanism. Shock-tube and laminar flame speed data from the PrIMe Data Warehouse are used with the B2BDC methodology to validate and optimize an elementary-reaction model. We focus on how the tools and analysis can improve our understanding of complex reaction models. The B2BDC analysis revealed a set of experiments inconsistent with a larger body of evidence from the community. The study also identified the importance of the $H_2 + O_2(1\Delta)=H + HO_2$ pathway in high-pressure flames.

## 5.1 Introduction

The kinetics of $H_2/O_2$ combustion has been used as an important scientific and technological subject. We consider $H_2/O_2$ reaction mechanisms because of their accumulated knowledge and ongoing research [38, 71–76]. These mechanisms represent a fundamental component of any hydrocarbon reaction model. As scientific inquiry improves our understanding of the kinetics, it has become clear that a systematic procedure to develop predictive models that can readily incorporate new information and provide feedback, is necessary.

Here, we apply the validation workflow on three $H_2/O_2$ reaction models. A reaction model is a collection of elementary chemical reactions, thermodynamic properties, and transport properties that adequately describes the transformation, energetics, and transport of a group of molecules and species. Throughout this chapter, the terms reaction model and reaction mechanism are used synonymously. The chemical reaction, $H + O_2 \xrightarrow{k_1} OH + O$, is an example of an elementary reaction, which converts atomic hydrogen and molecular oxygen to products at the rate $k_1$. The reaction rate, $k_1$, and its dependency on temperature can be expressed through the modified Arrhenius equation [77, 78],

$$k_1 = A_1 T^n e^{\frac{E_1}{RT}} \tag{5.1}$$

where $A_1$ is the pre-exponential factor, $E_1$ is the activation energy for the reaction, $R$ is the universal gas constant, $T$ is the temperature, and $n$ is the temperature dependence. When $n$ is zero, the original Arrhenius equation is recovered [77]. The point here is to emphasize the pre-exponential term $A_1$, which will encompass the rate-coefficient uncertainty in the present study. Each $H_2/O_2$ reaction mechanism has a set of uncertain model parameters that are the pre-exponential terms for the elementary reactions. Additional uncertain kinetic parameters can be incorporated into the B2BDC analysis, e.g., the activation energy (Chapter 6), collision efficiencies, transport properties; however, it is important for the experimental data (and selected QOIs) to have some sensitivity to these parameters, otherwise no information will be learned through the analysis.

### Reaction models

To test different aspects of the $H_2/O_2$ kinetics, three models were created from literature sources. The reaction model data (thermo, transport, rate constants) were archived in the PrIMe Data Warehouse and are available through the Warehouse API. The base reaction mechanism (Mechanism1) was taken from Burke et al. [72]. Reaction rates, thermodynamic, and transport properties for Mechanism1 can be found in Appendix D.1. In total, there are 27 elementary reaction rates; therefore, 27 uncertain model parameters for Mechanism1. This mechanism was chosen because of its high-quality performance when compared to shock-tube and flame data. Mechanism1 was primary used for code verification [6] of the validation workflow, ensuring that the numerical results shown in the original publication [72] could be replicated with PrIMe and the PrIMe Workflow Application components [38].

Mechanism2 consisted of Mechanism1 with a few critical parameter updates from Burke et al. [79] and Sellevåg et al. [80]. Mechanism2 is shown in Appendix D.2, which consists of 29 uncertain parameters. Lastly, Mechanism3 built on Mechanism2 by including chemistry for ozone, $O(^1D)$, $O_2(a^1\Delta g)$, and $OH^*$ whose reaction rates came from Konnov [75]. Mechanism3 has 58 uncertain model parameters.

A set of shock-tube and flame data are used to determine the validity of Mechanism3. The results shown in Section 5.5 are a proof-of-concept that the validation workflow can identify well-recognized inconsistencies (e.g., modeling low-temperature shock-tube ignition using a homogeneous, adiabatic, constant-volume reactor) and identify yet-unrecognized scientific interpretations (e.g., $O_2(a^1\Delta g)$ pathways in high-pressure flames).

## 5.2 Selection of QOIs and experimental data

QOIs are carefully selected to summarize a critical feature of an experiment. Some common examples include the ignition delay time, the peak value of a species profile, laminar flame speed, and time/location of a property. In the present study, we store QOIs in the PrIMe

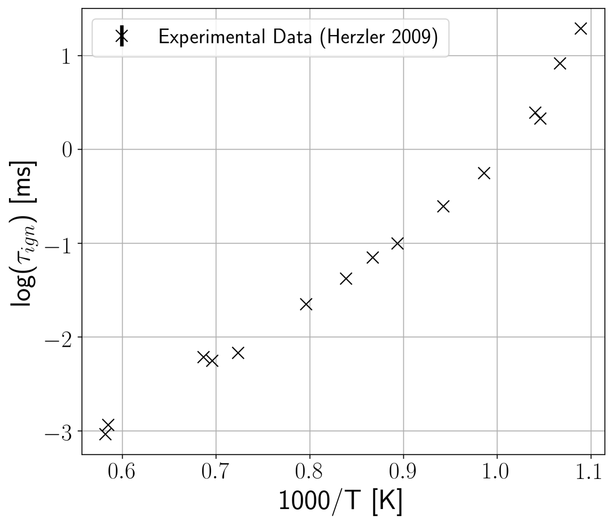
Data Warehouse as Data Attribute XML files, which fully describe the experimental feature, conditions, uncertainty bounds, and transformations required to obtain the QOI value.

Selecting appropriate QOIs for analysis is not a trivial task. Let us consider a shock-tube experiment conducted by Herzler and Naumann [81]. Figure 5.1(a) shows a plot of the ignition delay time ($\tau_{\text{ign}}$) vs. temperature ($T$) from an experimental with an initial mixture of hydrogen, oxygen, and argon at 1 atm. Each experimental measurement could be used as a QOI; however, this would require constructing a surrogate model for each data point. Using every data point as a QOI can be inefficient due to correlated or dependent measurements. Alternatively, Frenklach et al. [82, 83] proposed using a few representative QOIs to summarize the behavior from a set of data (shown in Figure 5.1).

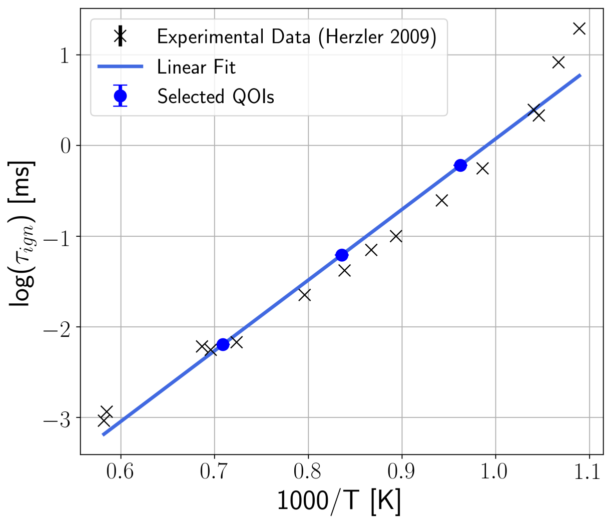
In this study, three QOIs will summarize each set of shock-tube data. First, a linear model is fit to the experimental data on a log($\tau_{\text{ign}}$)-vs.-1000/T plot by minimizing the least-squared error. Figure 5.1(b) shows the linear model as a blue line and the representative QOIs are the blue points at the quarter, half, and three-quarter temperatures. These points are the representative QOIs that summarize the shock-tube data set. The uncertainty associated with each of these QOIs was estimated by the standard error,

$$s = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2} \tag{5.2}$$

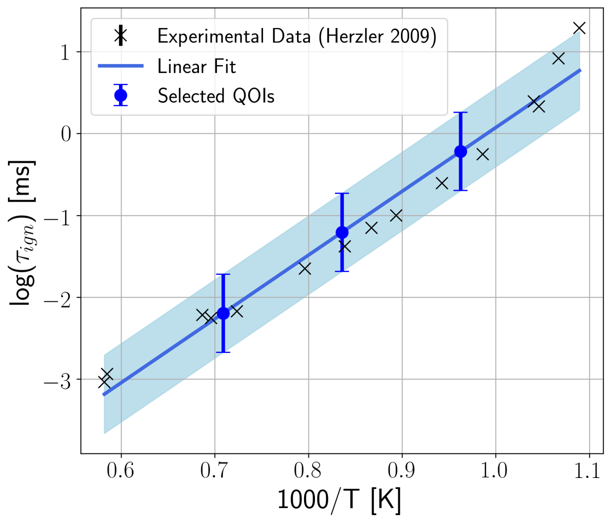
where $y_i$ is the experimental measurement, $\widehat{y_i}$ is the linear model estimate, and $n$ is the total number of data points in a given experimental data series. Figure 5.1(c) shows the estimated uncertainty, where $\pm 2s$ is the initial estimated experimental bounds. This procedure was used to derive 114 shock-tube QOIs across 38 conditions in 12 published studies. Two flame studies were also included in the validation data, accounting for ten QOIs (124 QOIs in total). The uncertainty bounds prescribed to the flame QOIs were based on expert opinion. The shock-tube and laminar flame speed QOIs are presented in Tables E.1 and E.2 of Appendix E.



(a)



(b)



(c)

Figure 5.1: Selecting representative QOIs for shock-tube experiments. A shock-tube experiment conducted by Herzler and Naumann [81] is used for illustration, where the initial composition of the gaseous mixture are described by mole fractions. The experiment was conducted at an initial pressure of 1 atm and an initial gas composition of 0.06 $H_2$, 0.03 $O_2$, and 0.90 Ar. Panel a: experimental data are plotted on a log ignition-delay time vs. 1000/T plot. Panel b: A linear model is fit to the experimental data, shown in blue. Three blue points are the QOIs selected to represent the experiment. These QOIs were selected at the quarter, half, and three-quarters temperature. Panel c: Error bars are the estimated experimental uncertainty (Eq. 5.2) shown as a light-blue region.

## 5.3 Selecting active variables

Each combustion mechanism has a different number of uncertain model parameters. In this study, the number of uncertain model parameters are equal to the number of elementary reactions. For example, Mechanism3 has 58 uncertain model parameters; however, a particular model response (QOI) will not be significantly affected by all model parameters. A sensitivity analysis can help identify the subset of model parameters that are most influential to the response. There are many approaches to sensitivity analysis [84]; e.g., local, variance-based [85], regression, etc. In this chapter, a global, regression-based sensitivity analysis will be performed [86]. For each QOI, a linear surrogate model is fit to the model response by using a fractional factorial design, specifically a Hadamard matrix [87, 88]. The regression coefficients from the linear model represent a measure of the sensitivity,

$$S_i = \frac{\partial f(x, x_{s,e})}{\partial x_i} \tag{5.3}$$

where $f(x, x_{s,e})$ is the underlying model for the $e$-th QOI. Coefficients with the largest magnitude have the highest sensitivity over the parameter space. We can normalize Eq. 5.3 by the following relation:

$$\widetilde{S}_i = S_i \frac{x_i}{f(x, x_{s,e})} = \frac{\partial f(x, x_{s,e})}{\partial x_i} \frac{x_i}{f(x, x_{s,e})} \tag{5.4}$$

In some physical systems, the most sensitive model parameter is known *a priori*. For example, in combustion, the $H + O_2$=$OH + O$ reaction rate has one of the highest sensitivities, but it is also the reaction rate with the least uncertainty. To account for the prior parameter uncertainty, an *impact factor* [82] is defined as the sensitivity, $S_i$, times the uncertainty. Model parameters that have large uncertainty and sensitivity may also have a significant impact on the QOI response. Parameters with large impact factors are selected as the *active variables* for a surrogate model. Each surrogate model will only depend on its associated set of active variables.

## 5.4 Evaluating shock-tube and flame QOIs

Simulations of shock-tube QOIs were performed with the PrIMe Plug Flow Reactor (PFR) component [38], available online via the ReactionLab toolbox [89]. Shock-tube experiments were modeled as a homogeneous, adiabatic, constant-volume reactor [90–92]. Input conditions for the simulation were passed from the PrIMe Data Warehouse to the PFR component.

Laminar flame speed simulations were performed using the online-based infrastructure, CloudFlame [51, 52, 93] that communicates with the PrIMe Workflow Application (PWA).

Simulation inputs are packaged in a PrIMe interface file (HDF5 file) that is passed to CloudFlame via a web-services application. CloudFlame then executes a Python script upon receipt and parses the HDF5 into appropriate input files. The script executes Cantera [94], an open-source combustion software, to run an isobaric, steady, one-dimensional, laminar premixed flame [90, 92]. After a simulation is complete, the Cantera output is parsed and saved into an HDF5 file that is passed back to the PWA. Approximately 4,700 premixed laminar flame speed simulations were conducted using this workflow module. Simulations were performed in parallel using 64 compute nodes on a cluster at KAUST.

## 5.5 Results and Discussion

For all mechanisms, the prior reaction rates were directly adopted from rate assessments and/or fundamental studies and either lie within the literature, reviewed uncertainties (e.g., $\mathrm{O} + \mathrm{H}_2 {=} \mathrm{OH} + \mathrm{H}$ [95]), or are based on more recent studies [79, 80, 96, 97]. The following analysis will focus on the assessment (and feedback) of the validation workflow for Mechanism3.

Each QOI was first analyzed for *self-inconsistency* before evaluating the scalar consistency measure for all 124 QOIs. A QOI is self-inconsistent when no point $x$ in $\mathcal{H}$ can produce an output value that falls within the associated uncertainty bounds. The $e$-th QOI is self-inconsistent if,

$$\mathcal{F}_e = \{x \in \mathcal{H} : L_e \leq M_e(x) \leq U_e\} = \varnothing.$$

Thirty-six of the shock-tube QOIs were self-inconsistent upon first analysis.

All uncertainty bounds for the shock-tube QOIs had followed the procedure described in Section 5.2. This method caused ten QOIs to have implausibly narrow uncertainty bounds (below 5%). After expanding the experimental uncertainty bounds of the self-inconsistent QOIs by a factor of 2, ten of the thirty-six QOIs became self-consistent. The remaining self-inconsistent QOIs were removed from the analysis, leaving them for future investigations.

Evaluating the scalar consistency measure for all remaining QOIs led to a consistency measure of $C_D = [-0.10, -0.02]$. To interpret, expanding all QOIs bounds by at least 2% to is necessary to resolve the model-data disagreement. Following the validation workflow (in Figure 3.1), the sensitivity of the consistency measure [56] was assessed for each QOI and parameter bound. Figure 5.2 shows the sensitivity of the consistency measure due to QOI bounds (indexed by the PrIMe ID). QOIs `a00000370` and `a00000460` have the highest sensitivity and are shock-tube experiments. Comparing this result to the sensitivity from the model parameters, shown in Figure 5.3, we observe that the sensitivities are nearly an order of magnitude smaller for the model parameters. A relative change greater than 5% was required to the upper or lower bound of 25 shock-tube QOIs to resolve the inconsistency. These QOIs were withheld from the analysis and present a future opportunity to re-evaluate their uncertainty bounds, surrogate approximations, and model-form uncertainty. The shock-tube QOIs that were removed from the final dataset are listed in Table E.1 with asterisks.

The scalar consistency measure was calculated on the final dataset and had a measure of $[0.03, 0.12]$. The LS-F optimization point, Eq. 3.5, was calculated using the consistent dataset. A new, optimized mechanism, called DynamicMech151203, was created using the LS-F point. The name "DynamicMech151203" was selected to show the reaction mechanism is merely a revision of Mechanism3 that can be improved by the combustion community through collaborative science. All mechanisms are provided in Appendix D as PrIMe XML files, HDF5 files, and Cantera input files.

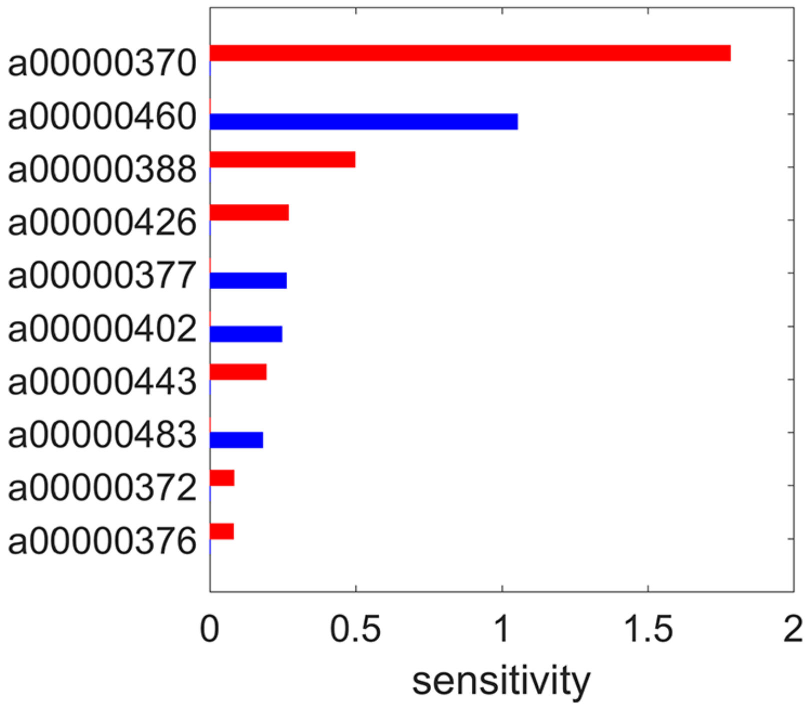


Figure 5.2: Normalized sensitivity of the consistency measure due to the QOI bounds. Blue bars indicate sensitivity to a QOI lower bound ($L_e$), and red bars are the upper bounds ($U_e$), where $e$ is the QOI index. The ten most sensitive QOIs are shock-tube data except `a00000483`, which is a flame speed experiment. Each experiment is indexed by its PrIMe ID, which is detailed in Appendix E.

## Discussion

The validation workflow, specifically the B2BDC analysis in the workflow, could determine a set of shock-tube experiments were at odds with a collection of experimental data. We

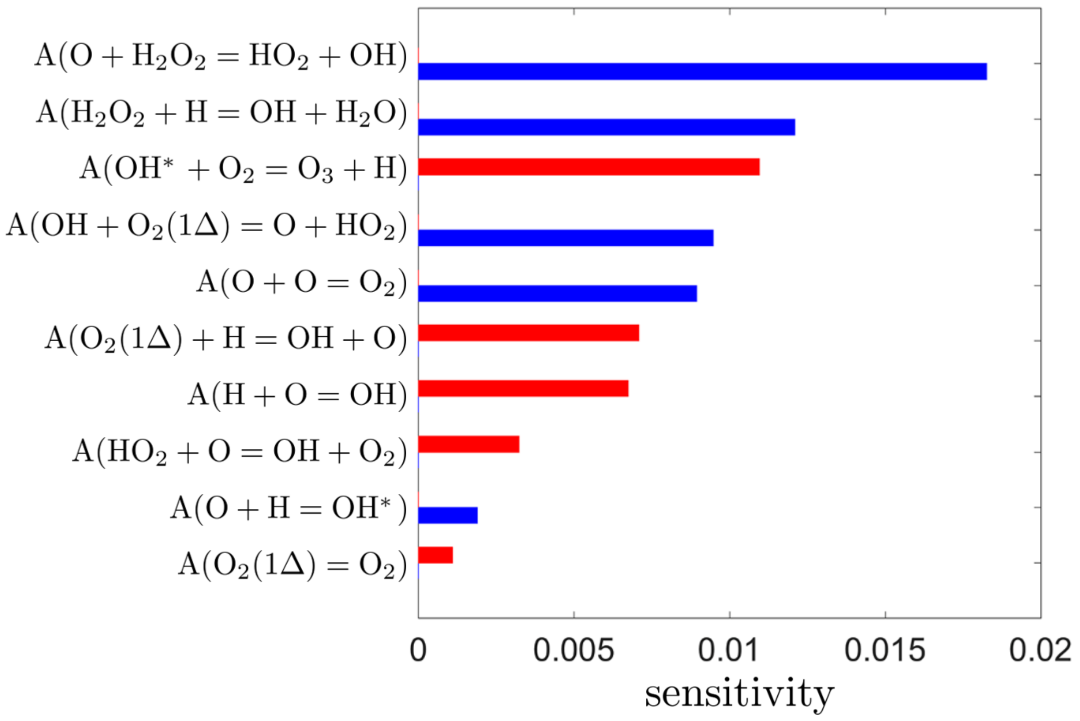


Figure 5.3: Normalized sensitivity of the consistency measure due to the parameter bounds. Blue bars indicate sensitivity to a parameter's lower bound ($l_i$), and red bars are the upper bounds ($u_i$), where $i$ is the parameter index. Sensitivities to the consistency measure for the experimental bounds (shown in Figure 5.2) are an order of magnitude larger than those of the parameter bounds. Therefore, we examined the experimental bounds to resolve the inconsistency.

had also observed an unexpected role of $O_2(a^1\Delta g)$ chemistry in high-pressure flames, both are discussed below.

## The role of $O_2(a^1\Delta g)$ chemistry

The validation workflow could reveal chemistry or kinetics, during the selection of active variables, which are potentially important considering the inclusion of excited species in $H_2/O_2$ combustion. Comparing Mechanism3 vs. Mechanism2, revealed significant differences in flame speed predictions for high-pressure, fuel-rich flames. Numerical results for the flame speed can be found in Table 5.1. The only difference between these mechanisms is the inclusion of electronically excited state chemistry from Konnov [75].

Figure 5.4 shows a ranked ordering of impact factors for the laminar flame speed QOI (`a00000484`). This flame experiment was conducted at a pressure of 15 atm with an equivalence ratio of 2.5, i.e., fuel-rich. It can be seen that $H_2 + O_2(1\Delta)$=$H + HO_2$ has

Table 5.1: Comparison of laminar flame speed (multicomponent transport with Soret effect) for Mechanism1, Mechanism2, Mechanism3, and DynamicMech151203 at their nominal parameter values.

| **Target PrIMe ID** | **Preferred Key** | **Pressure** [atm] | $\Phi$ | **Measured** [cm/s] | **Mech1** [cm/s] | **Mech2** [cm/s] | **Mech3** [cm/s] | **DynamicMech** [cm/s] |
|---|---|---|---|---|---|---|---|---|
| a00000476 | H2 F - 476 | 1 | 0.6 | 81.88 | 83.94 | 88.87 | 86.39 | 97.95 |
| a00000477 | H2 F - 477 | 1 | 1.65 | 280.21 | 280.28 | 284.83 | 280.84 | 286.38 |
| a00000478 | H2 F - 478 | 1 | 4 | 165.29 | 162.10 | 165.37 | 160.18 | 173.27 |
| a00000479 | H2 F - 479 | 20 | 1.5 | 80.9 | 76.84 | 81.72 | 75.67 | 106.17 |
| a00000480 | H2 F - 480 | 20 | 2 | 64.05 | 59.42 | 63.17 | 56.14 | 90.05 |
| a00000481 | H2 F - 481 | 20 | 0.85 | 18.35 | 24.29 | 27.91 | 26.28 | 43.49 |
| a00000482 | H2 F - 482 | 15 | 0.85 | 26.9 | 33.30 | 37.23 | 35.32 | 53.22 |
| a00000483 | H2 F - 483 | 10 | 0.85 | 45.3 | 46.28 | 50.34 | 48.08 | 65.91 |
| a00000484 | H2 F - 484 | 14 | 2.5 | 28.7 | 30.32 | **32.14** | **27.84** | 48.31 |
| a00000485 | H2 F - 485 | 25 | 2.5 | 27.7 | 32.04 | **34.16** | **29.28** | 53.54 |

amongst the largest impact factors. This reaction, precisely its reverse reaction, has been shown to inhibit high-pressure flames [98]. This inhibition is consistent with the numerical results in Table 5.1 showing a slower flame speed for Mechanism3 compared to Mechanism2.

Additional simulations were conducted to examine the effect of the $H + HO_2 {=} H_2 + O_2(1\Delta)$ pathway in high-pressure flames. A modified version of Mechanism2 was created that scaled the reaction rate for the ground-state reaction, $H + HO_2 {=} H_2 + O_2$, by the additional flux from the excited-state reaction. Simulations of high-pressure flames using the modified Mechanism2 yielded prediction within 1% to Mechanism3, confirming that the inhibition in flame speed of Mechanism3 was due to the excited-$O_2$ pathway. This result stems from the validation workflow that offers tools and feedback to domain experts. A more thorough investigation into the role of excited species in the $H_2/O_2$ mechanism is required; nevertheless, the B2BDC analysis could suggest an unexpected pathway in high-pressure flames.

## Shock-tube inconsistencies

Consistency analysis of the dataset revealed that several low-temperature (less than 1000 K), high-pressure (approximately $3 - 4$ atm) shock-tube ignition-delay times were self-inconsistent. In these experiments, the maximum concentration of OH was used as an indicator of ignition. Simulations of the shock-tube experiments were conducted using an idealized reactor (i.e., homogeneous, adiabatic, constant-volume) where it was found that the calculated ignition-delay time was more than 2.5 times larger than the experimental measurement. However, it is most likely that this disagreement does not originate from the data; rather, it stems from the assumptions in the reactor model. Studies have shown [99–102] that under these conditions a more sophisticated reactor model that incorporates

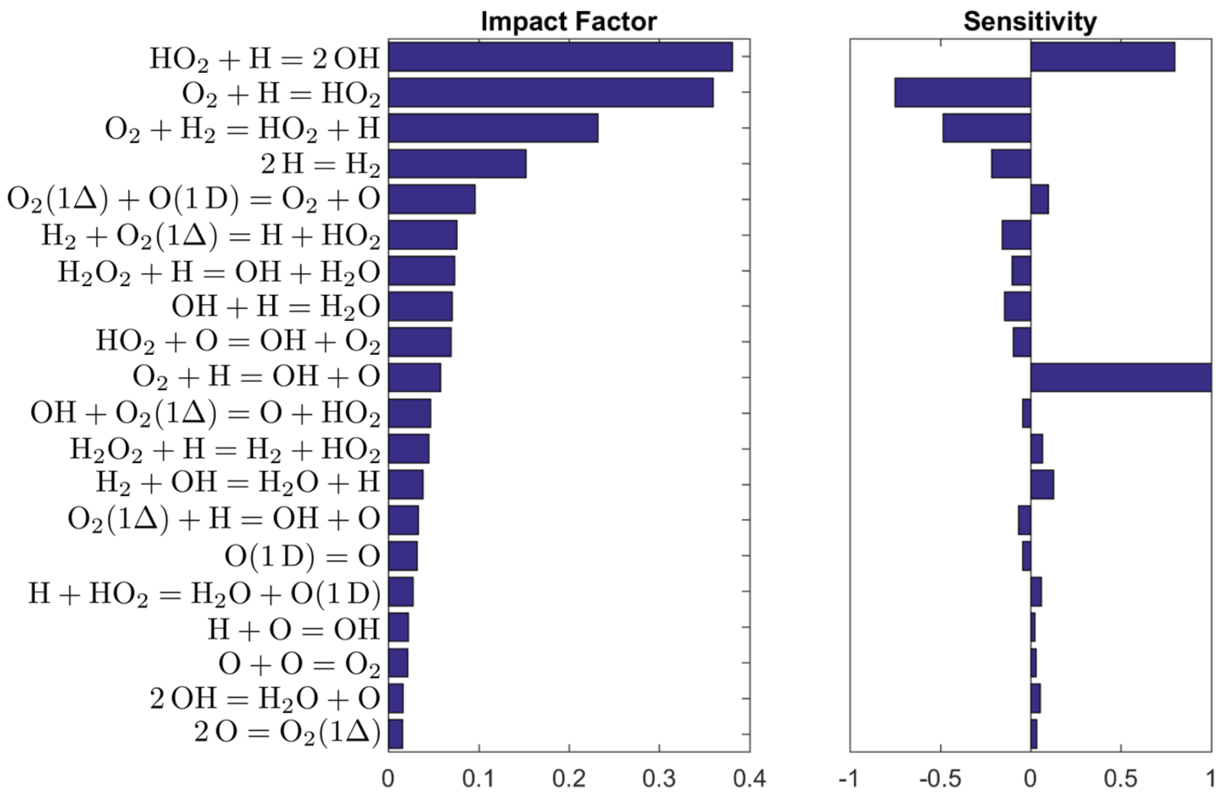


Figure 5.4: Sensitivity and ranked impact factor results for a laminar flame speed QOI (`a00000484`). The top 20 reactions are ranked by impact factor, defined as absolute sensitivity multiplied by uncertainty length.

heterogenous gas dynamics and heat-release effects should be implemented. The identified self-inconsistent QOIs should not be discarded as "bad data," instead they become valuable to the analysis, requiring a more complex reactor model.

## Performance of the optimized mechanism

A new optimized reaction mechanism (DynamicMech) was developed by finding the LS-F optimal model parameter (Eq. 3.5) from the feasible set of Mechanism3. The LS-F parameter minimizes the weighted least-squares error between model prediction and experimental data. In this study, the weights were set to unity.

All 114 shock-tube QOIs were calculated using the nominal reaction rate for all mechanisms. Table 5.2 shows the $\ell_2$-norm difference between model prediction $M(x_{nom})$, and the experimental data for the shock-tube QOIs, $\tau_{ign}$. Both $M(x_{nom})$ and $\tau_{ign}$ are vectors of length 114. DynamicMech outperforms the other mechanisms with a $\ell_2$-norm difference of 15.80; however, this result requires careful interpretation. Although

DynamicMech performed best in the prediction of shock-tubes, it was the worst in flame speed predictions (Table 5.1 and E.5). This bias in prediction towards the shock-tube experiments can be explained by the equal weights used in the LS-F parameter optimization and approximately ten times more shock-tube QOIs. Re-assessment of the weights in the LS-F optimization and their impact on the prediction of flame speed QOIs are subjects for future study.

Table 5.2: Comparison of the four reaction mechanisms prediction of the shock-tube QOIs. $\ell_2$-norm difference between model prediction (using the nominal model parameter) to the experimental data. The nominal model parameter for DynamicMech is the LS-F optimization point from Mechanism3.

| | $\|\|M(x_{nom}) - \tau_{\text{ign}}\|\|_2$ |
|---|---|
| Mechanism 1 | 29.40 |
| Mechanism 2 | 28.78 |
| Mechanism 3 | 29.09 |
| **DynamicMech151203** | **15.80** |

## 5.6 Conclusion

The validation workflow was applied to an $H_2/O_2$ reaction mechanism by using an extensive combustion data set with the UQ tools of B2BDC. Through this process, $O_2(a^1\Delta g)$ was identified as of potential importance in premixed laminar flame speed experiments. The B2BDC analysis also discovered a collection of shock-tube experiments that were self-inconsistent, requiring simulations beyond the idealized reactor model used in this study. The PrIMe cloud-infrastructure could easily add (or remove) QOIs from the analysis, in part from the integrated PrIMe Data Warehouse. The optimized reaction mechanism, DynamicMech, represents the first iteration of this process and is available on PrIMe. DynamicMech was created to be updated (much like a version of software) as new experimental data, prior knowledge, or new simulation codes are available. A manuscript has been written based on this material [66].

# Chapter 6

# Validating a reduced char oxidation model

A reduced char oxidation model is examined by applying the validation workflow of Chapter 3 and the piecewise modeling strategy of Chapter 4. Initial evaluation of the validation workflow determined that the reduced char oxidation model was consistent with a collection of validation data. However, the char particle temperature, the QOI, was challenging to accurately represent with a quadratic surrogate model, due to unphysically wide uncertainty ranges and large fitting errors. The latter, in turn, led to a consistent dataset; however, the surrogate models greatly deviated from the underlying model. To obtain more accurate fits, the piecewise modeling strategy was employed. This strategy, by fitting QOIs more accurately, was able to uncover the incompatibility of the char oxidation model and the experimental data. The physical reason turned out to be a widely used assumption of the initial particle diameter. Reconsidering the initial particle diameter as a distribution led to the opposite conclusion, that the model was consistent with the data. Thus, further examination of the input particle diameter is warranted.

## 6.1 Introduction

Improving our understanding of coal combustion is essential towards the development of cleaner, more efficient industrial boilers. Higher efficiency boilers can greatly increase thermal energy output compared to existing power plants, while also reducing harmful emissions (e.g., $CO_2$, $SO_2$, NOx, particulate matter, volatile organic compounds). Development of a new boiler design demands accurate (i.e., predictive) Computational Fluid Dynamics (CFD) simulations. Unfortunately, CFD simulations are often computationally expensive to evaluate. This can hinder any optimal design or UQ analysis for a new boiler design. Each CFD simulation is composed of various submodels. The multiphase flow, devolatilization, char oxidation, soot formation, ash transformation, and radiation heat transfer are each controlled by their own submodel. Selecting suitable submodels that can mitigate much of the computational effort, remains a challenge. Development of reduced-order submodels, which have been validated against a broad range of experiments, are necessary to lessen the computational effort while remaining predictive.

Although coal combustion models have been studied for years [103–108], it remains an ongoing challenge in engineering. The inhomogeneity in coal composition, outgassing of the volatile matter, pore structure, and annealing of the carbonaceous particle, are a few of the reasons this process remains difficult to investigate. Coal combustion [108, 109] can be simplified into the following steps: a coal particle first heats up, undergoes devolatilization, and then char oxidation.

This chapter focuses on the analysis and validation of the last step, char oxidation. The char oxidation model was developed by the Carbon-Capture Multidisciplinary Simulation Center (CCMSC) at the University of Utah [110]. This char oxidation model describes the mass transport and chemical reactions that occur between char carbon and three oxidizers: $O_2$, $H_2O$, and $CO_2$. A collection of experimental data stored in the PrIMe Data Warehouse (Chapter 2) is used with the B2BDC analysis (Chapter 3) to quantify the validity of the reduced-order char oxidation model. The validation data used throughout this study is described in Section 6.2. Section 6.4 describes the assigned prior uncertainty for the model parameters and QOIs. The results found by the B2BDC analysis and piecewise modeling strategy are explored in Section 6.7.

## 6.2 Experimental data

The reduced char oxidation model that is being analyzed is expected to be predictive in both conventional and oxy-fuel gas conditions. The CCMSC Coal Database, discussed in Section 2.5, was used to filter through the 2,710 solid-fuel experiments to determine a suitable set of validation data for these conditions. A collection of experiments were found from Hecht's thesis [111], which was carried out in a laminar-entrained flow reactor for three different coal types: Black Thunder, Illinois-6, and Utah Skyline. The ultimate and proximate analysis of these coals can be found in [111]. In this study, I will only focus on the analysis of a single coal type; namely Utah Skyline. The laminar-entrained flow reactor used for these experiments is shown in Figure 6.1 and is operated by Sandia National Laboratories (which is described in previous works [112–114]).

The particle temperature, size, and velocity were simultaneously measured at various heights above the burner, as detailed in [112]. Char particle temperature was measured by using a two-color pyrometry at 550 and 770 nm. The particle temperature will be used as the QOI in this study. Particle size was determined by a coded aperture approach discussed in [114]. Char particles were initially prepared by sieving the particles into six size bins: $53 - 63$ $\mu$m, $63 - 75$ $\mu$m, $75 - 90$ $\mu$m, $90 - 106$ $\mu$m, $106 - 125$ $\mu$m, and $125 - 150$ $\mu$m. An experiment was then conducted across twelve different gas environments for each of the initial particle bins.

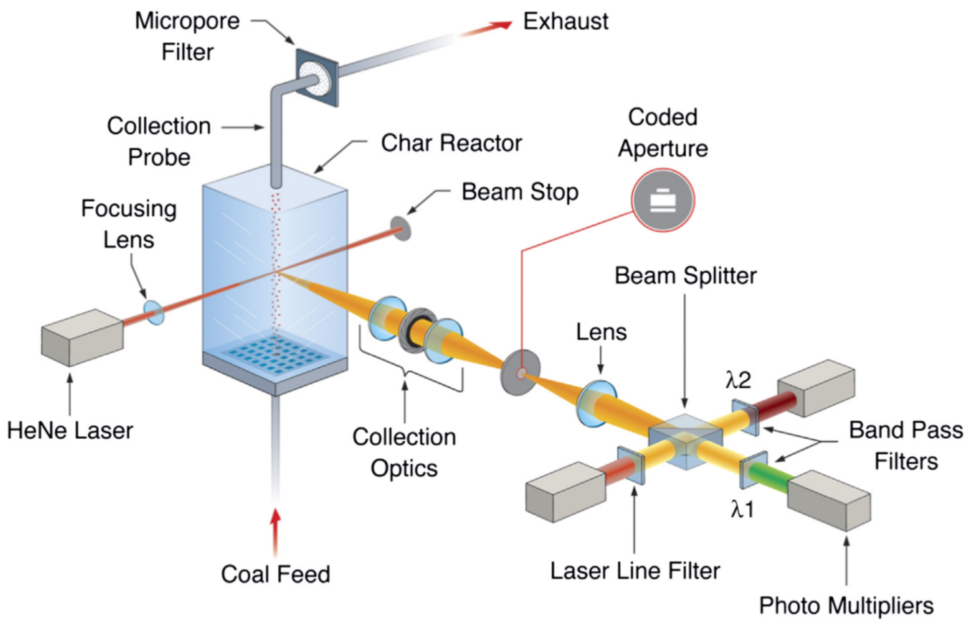


Figure 6.1: Optically accessible laminar entrained flow reactor. Char particles are fed into the center of a flat-flame Hencken burner. Char particles react as they travel upwards in a quartz chimney that is accessible by optical instrumentation.

Table F.1 lists the 72 experimental cases with their corresponding gas flow compositions, initial particle size bins, sampling heights, and associated PrIMe IDs. For each experimental case, there are various heights above the burner where a measurement had occurred. Approximately 100 char particles are reported as measured for each height [111]. There are a total of 399 gas conditions and heights with temperature measurements. These measurements are used to constrain the char oxidation model output.

## 6.3 Char oxidation physics and instrument models

The char oxidation model is composed of two submodels: the physics and instrument models [65, 115]. Figure 6.2 shows a block diagram of their relation, where the instrument model emulates the apparatus; i.e., a laminar entrained flow reactor and optical measurement process. The flow reactor and measurement process are detailed in Section 6.2. The physics model computes the reaction rate of the char reactions $r''_H$, and is a submodel of the instrument model. Calculated char kinetics are fed back into the instrument model to evaluate the following energy balance:

$$\frac{d_p \rho_p c_p v_p}{6} \frac{dT_p}{dz} = -S_{EXT} \varepsilon \sigma (T_p^4 - T_w^4) - S_{EXT} h^{\bullet} (T_p - T_g) + f_p r''_H \Delta h_{rxn} \tag{6.1}$$

where $T_p$ is the char particle temperature, $d_p$ is the char particle diameter, $\rho_p$ the char particle density, $c_p$ the particle specific heat, and $v_p$ the velocity of the particle. The first term on the right-hand side accounts for radiative heat transfer, the second term for convective heat transfer, and the third term for the heat release due to heterogeneous char reactions.

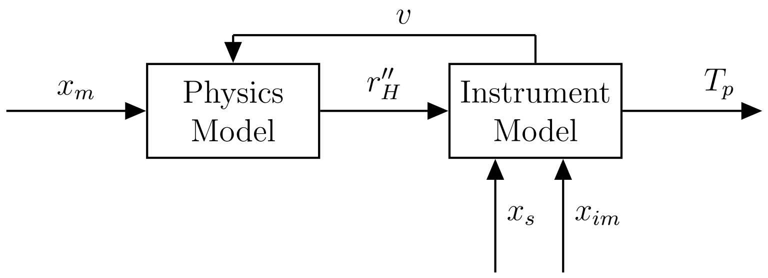


Figure 6.2: Block diagram of the char oxidation physics and instrument models. The output $T_p$ is used as the QOI for comparison with the experimental data. The model parameters $x_s$, $x_{im}$, and $x_m$ are the scenario, instrument, and physics model parameters, respectively. The char oxidation model has two internal parameters, the calculated char reaction rate $r''_H$ and the instrument state vector $v$.

The emissivity $\varepsilon$ of the char particle is an uncertain model parameter whose prior bounds are listed in Table 6.1. $S_{EXT}$, $\sigma$, and $T_w$ are the external surface area of the particle, the Stefan-Boltzmann constant, and the temperature of the reactor wall, respectively. These are known quantities; i.e., without any uncertainty. In the convective heat transfer term, $h^{\bullet}$ is the heat transfer coefficient corrected for mass transfer effects [116] and $T_g$ is the temperature of ambient gas; both are known parameters. In the final term, $f_p$ is the fraction of heat release that remains within the char particle, an uncertain model parameter. $\Delta h_{rxn}$ is the heat of reaction from the char gasification reaction and $r''_H$ is the reaction rate of char computed by the physics model, the details of which are discussed in [67].

Solving the ODE in Eq. 6.1 requires a realization of the uncertain model parameters $x$ and the scenario parameters $x_s$. The scenario parameter is assumed to have no uncertainty and its values are found in Table F.1; specifically, the gas condition, measurement height above the burner, and the initial particle size bin. Following the recommendation of domain scientists [117], the initial particle diameter was taken to be the average of the reported bin. The char oxidation model returns as an output the particle temperature at the specified measurement height.

## 6.4 Prior uncertainty on model parameters

The prior uncertainty for all uncertain model parameters is shown in Table 6.1. The uncertain model parameter vector $x$ can be divided into two groups, the physics model parameters $x_m$ and the instrument model parameters $x_{im}$, where, $x = [x_m; x_{im}]$. $f_p$, $\varepsilon$, and $\rho_{\mathrm{p0}}$, the initial density of the char particle are instrument model parameters. The pre-exponential and activation energies for the three Arrhenius oxidation reactions are physics model parameters. Both $S_{g0}$ and $\tau$ are physics model parameters that evaluate the diffusion through the porous char particle. $S_{g0}$ is the initial specific internal surface area of the particle and $\tau$ is the tortuosity of the particle [118].

The three char oxidation reactions are each defined by an Arrhenius expression (Eq. 5.1), where the temperature exponent, $n$ is equal to zero. Prior knowledge of the $O_2$ and $CO_2$ oxidation rates enabled domain scientists to define a prior correlated region. This correlated region is between the pre-exponential and activation energy of the $O_2$ reaction and another correlated region for the $CO_2$ reaction. These reactions are expressed by a rotated coordinate system, $\xi_1$ and $\xi_2$. This rotation can be described by:

$$\begin{aligned}
A_{\mathrm{O_2}} &= \exp(1.54\xi_{1,\mathrm{O_2}} - 0.308\xi_{2,\mathrm{O_2}} + 3.86)\\
\theta_{\mathrm{O_2}} &= 2.89\mathrm{E}3\ \xi_{1,\mathrm{O_2}} + 579\xi_{2,\mathrm{O_2}} + 1.03\mathrm{E}4\\
A_{\mathrm{CO_2}} &= \exp(0.326\xi_{1,\mathrm{CO_2}} - 0.0814\xi_{2,\mathrm{CO_2}} + 6.96)\\
\theta_{\mathrm{CO_2}} &= 4.04\mathrm{E}3\ \xi_{1,\mathrm{CO_2}} + 1.01\mathrm{E}3\ \xi_{2,\mathrm{CO_2}} + 2.42\mathrm{E}4
\end{aligned}$$

where $\theta \equiv E/R$ in the Arrhenius equation (Eq. 5.1).

Table 6.1: Uncertain model parameters and their associated prior bounds for the char oxidation model. The hypercube formed by the interval bounds is the prior parameter space $\mathcal{H}$.

| Parameter | $l_i$ | $u_i$ |
|---|---:|---:|
| $\varepsilon$ | 0.5 | 1.0 |
| $\rho_{\mathrm{p0}}$ | 200 | 800 |
| $f_{\mathrm{p}}$ | 0.4 | 1.0 |
| $\xi_{1,\mathrm{O_2}}$ | $-\sqrt{2}$ | $\sqrt{2}$ |
| $\xi_{2,\mathrm{O_2}}$ | $-\sqrt{2}$ | $\sqrt{2}$ |
| $\log(\mathrm{A_{H_2O}})$ | 4.61 | 9.21 |
| $\mathrm{E_{H_2O}}$ | 40000 | 100000 |
| $\xi_{1,\mathrm{CO_2}}$ | $-\sqrt{2}$ | $\sqrt{2}$ |
| $\xi_{2,\mathrm{CO_2}}$ | $-\sqrt{2}$ | $\sqrt{2}$ |
| $\log(\mathrm{S}_{g0})$ | 6.91 | 13.82 |
| $\tau$ | 1.0 | 5.0 |

## 6.5 Estimating the QOI bounds

In this analysis, a conservative estimation of the experimental bounds will be taken due to multiple sources of poorly understood or characterized uncertainty. In the energy balance shown in Eq. 6.1, the wall temperature $T_w$ was not experimentally measured and was set to 500 K in this analysis [114, 119]. Further, the wall temperature is most-likely a function of height $T_w(z)$, where the temperature is higher the closer to the burner. We also assume in the derivation of the energy balance that all char particles are spherical in morphology.

Within the instrument model, there is significant uncertainty that is unaccounted for; e.g., the particle velocity, sieving of the char particles, surrounding gas temperature, particle spectral emissivity, etc. There are also potentially unaccounted for errors in the instrumental model [115], which converts the measured voltage as a light intensity into a derived quantity (e.g., a particle temperature or diameter). Recent work by Xu et al. [120] had shown that there is significant variability in the emissivity of char within the visible light region ($390-710$nm), which would introduce larger than expected uncertainty in the two-color pyrometer measurement; and thus, larger uncertainty in the particle temperature measurements. Each char particle measured is inhomogeneous in its composition. The fraction of ash deposition on the char particle, or the amount of maceral matter, varies; thus, it is an additional sources of uncertainty. For these reasons, a conservative estimate of the uncertainty in the particle temperature was taken, which is defined by:

$$\mu_e = \text{mean}(T_{e,\text{data}}) \tag{6.2}$$

$$[L_e, U_e] = [\mu_e - \max(200,\ 2\sigma_{e,\text{data}}) \quad \mu_e + \max(200,\ 2\sigma_{e,\text{data}})] \tag{6.3}$$

where $\mu_e$ and $\sigma_{e,\text{data}}$ are the mean and standard deviation of the experimental measured particle temperature data ($T_{e,\text{data}}$) for the $e$-th QOI.

## 6.6 Iterative model development

The validation workflow was employed to develop a char oxidation model consistent with the experimental measurements. We begin by considering a single-film char oxidation model [114], which is widely used in modeling pulverized coal. The char oxidation model was developed by the CCMSC [110]. An initial application of the validation workflow determined that the proposed reduced char oxidation model was inconsistent with the experimental measurements described in Section 6.2.

Following the validation workflow, domain scientists at CCMSC have received feedback from the B2BDC analysis to assist in the development of the next reduced char oxidation model. The following char oxidation model was built upon the previous iteration and included additional physics (e.g., ash effects, annealing, porosity model, internal mass transfer) or modifications to the instrument model. The newly developed char oxidation model was then analyzed through another iteration of the validation workflow. Each iteration of the validation workflow amounts to changing the underlying model component; i.e., a function handle, as shown in Figure 3.1. Ultimately, after more than ten iterations of the workflow, a reduced char oxidation model was arrived at, and will be analyzed in the following sections. This reduced char oxidation model simulates the char particle as a porous medium, with both internal and external reactions and mass transfer. Derivation of the reduced char oxidation model are discussed in [67, 121].

## 6.7 Results

### Analysis with a fixed initial diameter

We begin our analysis by constructing quadratic surrogate models over the prior parameter space $\mathcal{H}$, whose bounds are shown in Table 6.1. A surrogate model was built for each QOI by employing a space-filling Latin Hypercube design [122] of 624 samples. The maximum estimated fitting error was calculated for each of the 399 QOIs, as defined in Eq. B.6. Figure 6.3 shows the estimated errors, where nearly all QOIs have an error larger than the experimentally reported uncertainty (200 K).

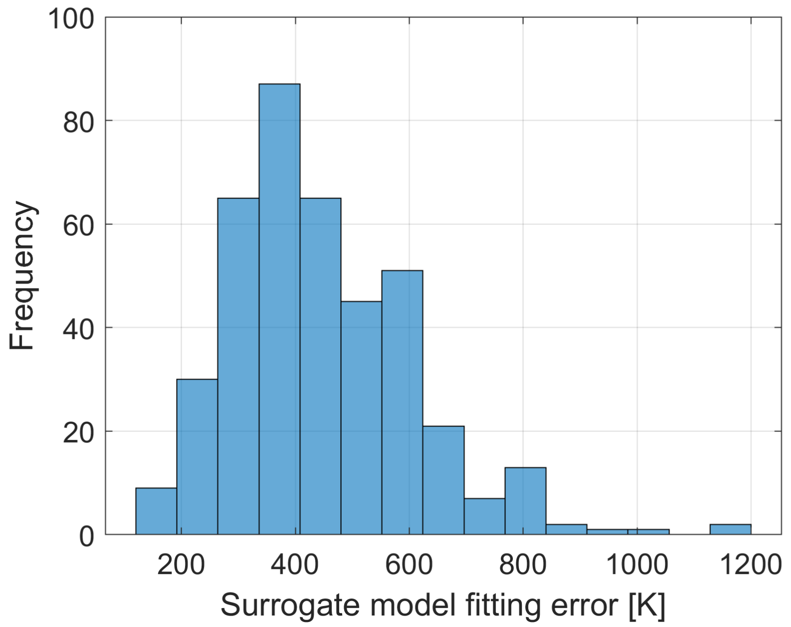


Figure 6.3: Histogram of the estimated surrogate model fitting error for each of the 399 QOIs. For each QOI, the estimated maximum error was evaluated by 10-fold cross-validation.

A dataset is formed after the surrogate models are developed over $\mathcal{H}$ by combining the surrogate models with the associated experimental bounds. Recall, the experimental bounds also carry the uncertainty from the estimated fitting error (Section 3.1). The dataset was found to be consistent with a scalar consistency measure of $C_D = [0.06, 0.15]$. However, this result does not instill much confidence, due to the significant fitting errors shown in Figure 6.3. The B2BDC analysis accounts for the fitting error by propagating forward the uncertainty, expanding the corresponding QOI bound, as described in Eq. B.4. Developing more accurate surrogate models would reduce this expansion and its effect on the UQ analysis.

In Chapter 4, a strategy for piecewise modeling was developed in order to construct accurate polynomial surrogate models on feasible regions of the parameter space. The scalar consistency measure was evaluated on each subdomain using all 399 QOIs. An error tolerance of 50 K was set in the analysis. The prior parameter space $\mathcal{H}$ was partitioned into 27 disjoint subdomains, requiring 12,655 evaluations of the char oxidation model. All 27 subdomains were proven inconsistent with the experimental data. As the surrogate model fitting error was improved, the scalar consistency analysis determined that the char oxidation model was at odds with the prior and experimental data.

Ideally, the general strategy for piecewise modeling (see Section 4.2) would be implemented, where the prior domain $\mathcal{H}$ is partitioned until all subdomains have a fitting error less than the specified error tolerance. This procedure, however, requires too many evaluations of the underlying model. Hence, the novel piecewise modeling strategy (see Section 4.3) was employed. In order to prove the underlying char oxidation model is inconsistent, it must be assumed that the estimated surrogate model fitting error is equal to or larger than the maximum error. Invalidating an underlying model using a surrogate model is described in Appendix B.1.

When constructing a surrogate model over a subdomain, all samples in the domain are used; therefore, the maximum estimated error, $\widehat{\epsilon}_{e,max}$, via 10-fold cross validation is larger (and more conservative) than the training error. However, while $\widehat{\epsilon}_{e,max}$ may be conservative, it still may not be larger than the maximum error, $\epsilon_{e,max}$, which is unknown. Computing the absolute maximum error $\epsilon_{e,max}$ is often intractable (discussed in Appendix B.2). Here, we will introduce an expansion factor, $K$, which increases our estimation of the surrogate model fitting error. This factor is aiming to achieve $K\,\widehat{\epsilon}_{e,max} \geq \epsilon_{e,max}$. The estimated error $\widehat{\epsilon}_{e,max}$ is increased by $K > 1$. However, there is no guarantee that a chosen value of $K$ will satisfy the inequality; rather, it illustrates the motivation for its inclusion. Employing the piecewise modeling strategy with a factor of 1.25 resulted in a larger number of subdomains, 43; yet, all subdomains were still inconsistent. Despite increasing the estimated fitting error, the char oxidation model and the experimental data were incompatible over the prior domain.

**Reexamining the initial particle diameter**

One of the modeling assumptions given to us (discussed in Section 6.3) was that the char oxidation was evaluated with a fixed particle diameter set to the average sieved bin. Let us consider the experimental measurement for one QOI shown in Figure 6.4. Here, the initial sieved bin is shaded in grey, $[53, 63]$ $\mu$m. When evaluating this QOI, an initial particle diameter of 58 $\mu$m was used; however, the vast majority of the measured particle diameters were larger than the initial diameter. Char particles were measured larger in diameter *after* burning.

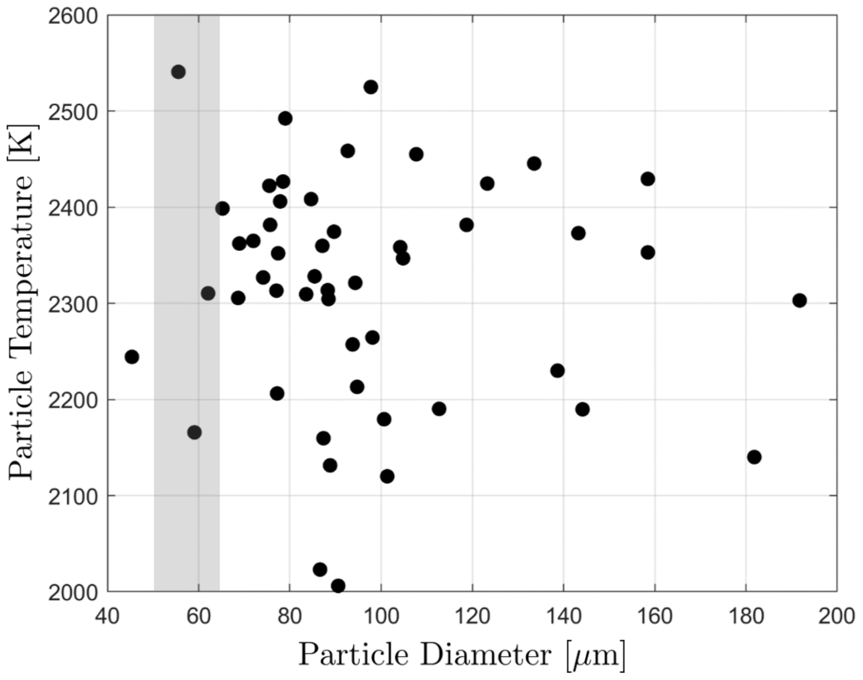


Figure 6.4: Measurement data for prepared Utah Skyline char from Hecht [111]. In a gas environment of 60% $O_2$, 16% $H_2O$, and 24% $CO_2$ both the particle temperature and diameter were measured at a height of 6.99 cm above the burner. The grey shaded region indicates the initial sieved particle bin of $58 \pm 5$ $\mu$m.

There are a few potential causes for an increase in the particle size. Coal particles undergo a swelling process when heated [123]; however, for this experiment, the coal particles were prepared chars, which would have minimal swelling [124]. Small particles can aggregate, which could explain a larger perceived size [125], but due to the low feed rate in terms of density (grams of coal per minute) with which char particles were fed into the flow reactor, potential aggregation of char particles would have been minimized. For these reasons, the char oxidation model should strictly decrease the particle diameter while burning. Nevertheless, the measurement data provides conflicting information, indicating larger than initial particle diameters. In the following section, we reassess the assumption of the initial diameter by considering an initial distribution of char particles.

## Analysis using a distribution of particles

The piecewise modeling strategy was able to reveal the disagreement between the char oxidation model, the prior, and experimental data. Reassessment of the initial modeling assumptions suggested that a fixed initial particle diameter could be the source of the inconsistency. Hence, we replace the input scenario parameter of a fixed initial particle diameter with an initial particle size distribution function (PSDF). The model response then becomes a distribution of particle temperatures, where the QOI is redefined as the average particle temperature response.

The initial PSDF had not been experimentally measured and is an unknown quantity. The initial probability density function can be characterized by introducing additional uncertain model parameters; e.g., specifying an uncertain mean and standard deviation. However, with the challenges in fitting a quadratic surrogate model to the QOI response, we chose not to increase the dimensionality of the problem.

This analysis is approximated by assuming one realization of the uncertain PSDF, specifying a mean and standard deviation value for a log-normal distribution. These values are obtained from the distribution of particle diameters that were measured at various heights in the burner [111]. These measurements occur after burning and are smaller than the true (but unknown) initial PSDF. The difference between the measured particle distribution and the initial distribution is controlled by the chemical reaction rates, which are uncertain. Another potential difficulty when considering a distribution is that not all particles are seen by the instrumentation, which is discussed in the following section.

### Modeling the measurement apparatus

Char particles were measured in the laminar-entrained flow reactor following an *in situ* method where the size, temperature, and velocity are recorded [112]. Particle size and temperature are measured by observing the transmitted light from the particle. The published experimental data, however, does not represent the entire distribution of char particles that entered the laminar-entrained flow reactor [114]. A subset of the particles that pass by the instrumentation are unmeasured or rejected due to an irregular/overlapping particle signals. The unmeasured particles are a consequence of the calibration process, which chooses an aperture in which bright burning particles are in focus. Futhermore, additional post-processing is used by experimenters where only approximately 100 particles are published as measured.

The particle temperature (or any particle property) can be attained at any height when evaluating a computational model; however, in order to mimic the measurement process, we will only consider simulated particles which emit enough light to be "seen" by the instrumentation. The light intensity of a char particle is calculated by:

$$I = C d_p T_p^4 \tag{6.4}$$

where $C$ is a calibration constant (set to unity), $d_p$ is the diameter of the spherical char particle, and $T_p$ is the particle temperature. A minimum light intensity was found by calculating Eq. 6.4 using the reported measurement data.

The average particle temperature, our new QOI, was calculated by taking the average temperature of particles whose light intensity is greater than the calculated minimum measured value. Particles that have a light intensity less than the minimum value are not included in the calculation. Figure 6.5 illustrates the type of constraint imposed by the minimum light intensity, which results in a subset of the simulated particles being unobserved.

### Surrogate modeling and results

Quadratic surrogate models were developed over $\mathcal{H}$ using the modified char oxidation model, which includes an assumed initial PSDF and light intensity model. Compared to the previous study, the QOI has changed from a single particle temperature to that of an average particle temperature. A histogram of the maximum estimated fitting error for all 399 gas conditions and measurement heights is shown in Figure 6.6. Comparing these results with the previous analysis in Figure 6.3 (using a fixed particle diameter) suggests that quadratic surrogate models better represent the average temperature QOI; however, the errors remain larger than the experimental uncertainty.

To attain more accurate surrogate models, the piecewise modeling strategy was employed, with an error tolerance of 100 K. Coefficients for the quadratic surrogate models were found by minimizing the 2-norm of the training sample error. Surrogate models were developed for each of the 187 subdomains, using 624 training samples in each subdomain. The scalar consistency measure found that 67 of the subdomains were consistent with the experimental data. The char oxidation model and the experimental data are consistent over the prior domain when using a distribution of particle diameters.

With the model and data consistent, the feasible set $\mathcal{F}$ is non-empty. A posterior analysis of the uncertain model parameters or the char oxidation model can then be conducted; e.g., sampling the feasible set or model prediction. A hit-and-run algorithm [126, 127] was used to obtain samples from the feasible set. Details of the sampling algorithm as it is applied for nonconvex quadratically constrained sets can be found in Algorithm 9.1 of Russi's thesis [57]. For each consistent subdomain, a sample of the feasible set was obtained. The union of all samples is shown in Figure 6.7. Each subplot has been rescaled to represent the prior uncertainty in the model parameters. Samples that are non-uniformly dispersed through the subplot indicate the learning that occurs from prior knowledge to posterior knowledge. The

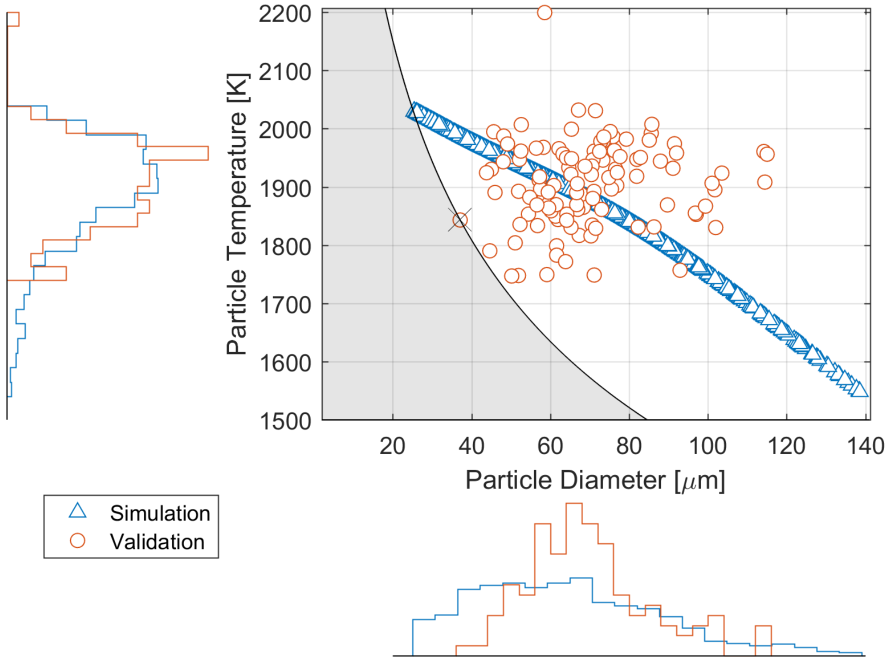


Figure 6.5: Visualization of the light intensity model enforcing a constraint on simulated particle temperature and size. Experimental data was acquired at a gas condition of 24% $O_2$, 14% $H_2O$, and 62% $N_2$ at a height of 3.81 cm above the burner. Experimental measurements are shown as red circles and the simulated char particles are shown (for one realization of $x$) as blue triangles. The grey shaded region is the constraint enforced by the light intensity model (Eq. 6.4). Particles in this region go "unseen" by the simulation model.

char oxidation model was used for blind prediction (described in Appendix B.3) for all 399 QOIs in the dataset. Prediction results for a single gas condition is shown in Figure 6.8. The black intervals are the prior experimental bounds and the red intervals are the blind model prediction from Eq. 3.4. Prediction results for all 12 gas conditions are reported in Appendix F.1.

## 6.8 Conclusion

This chapter examined a reduced char oxidation model by using the piecewise modeling strategy and B2BDC analysis. The validation workflow was applied in an iterative fashion to guide the development of the char oxidation model, which was ultimately found to be consistent with the prior knowledge and experimental measurements. However, the poor accuracy of the quadratic surrogate models introduced large uncertainty in the results.

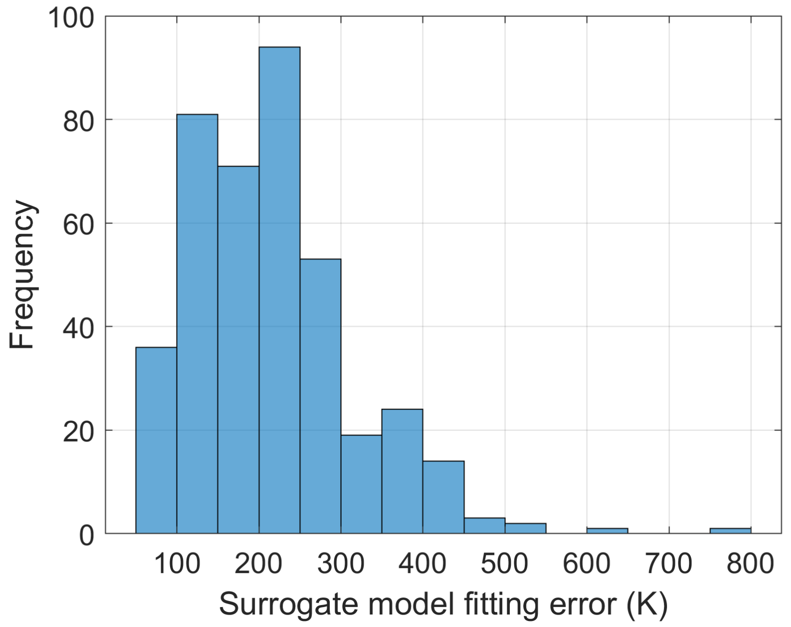


Figure 6.6: Histogram of the estimated surrogate model fitting error for each of the 399 QOIs. For each QOI, the estimated maximum error is shown, estimated by 10-fold cross validation.

Piecewise modeling was employed to refine the surrogate models and revealed the incompatibility between the char model and data, while also identifying a potential issue with the initial particle diameters.

A realization of the uncertain initial PSDF was selected and analyzed to evaluate the compatibility between the char oxidation model and the experimental data. The char oxidation model, using an initial PSDF and average temperature QOI, was found to be consistent with the experimental data. Different realizations of the uncertain initial PSDF, however, could lead to differing conclusions; i.e., the model and data are incompatible. The B2BDC analysis and piecewise modeling reached consistency, but doubts about the certainty of that conclusion persist. The results suggest that more work is necessary for characterizing the initial PSDF.

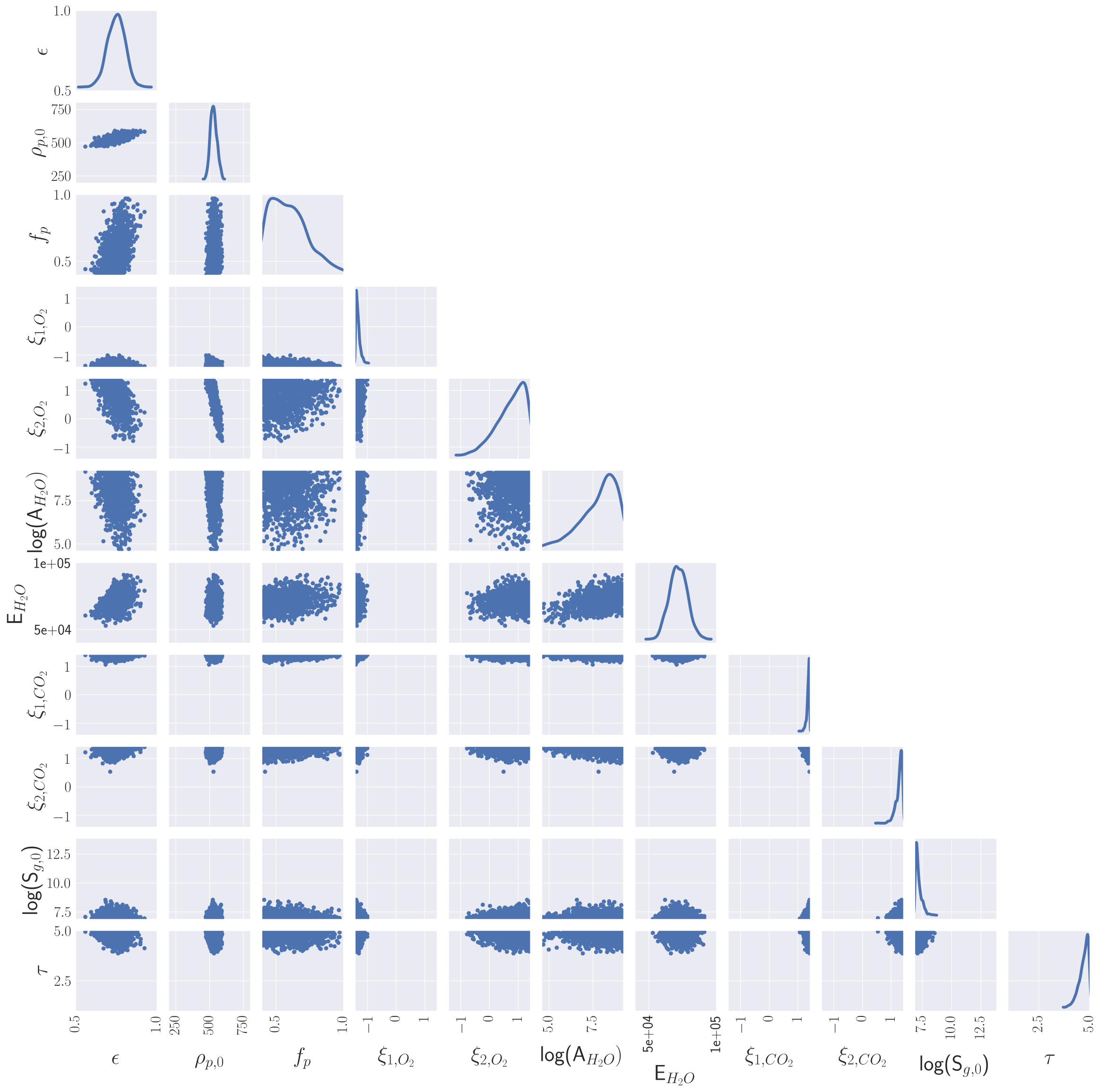


Figure 6.7: Pair-wise projection of the feasible model parameters. Each blue point is a model parameter vector $x \in \mathcal{F} \subset \mathbb{R}^{11}$. Shown on the diagonal is the marginal histogram of feasible samples from the corresponding parameter. Each sub-plot has been rescaled to the prior bounds listed in Table 6.1.

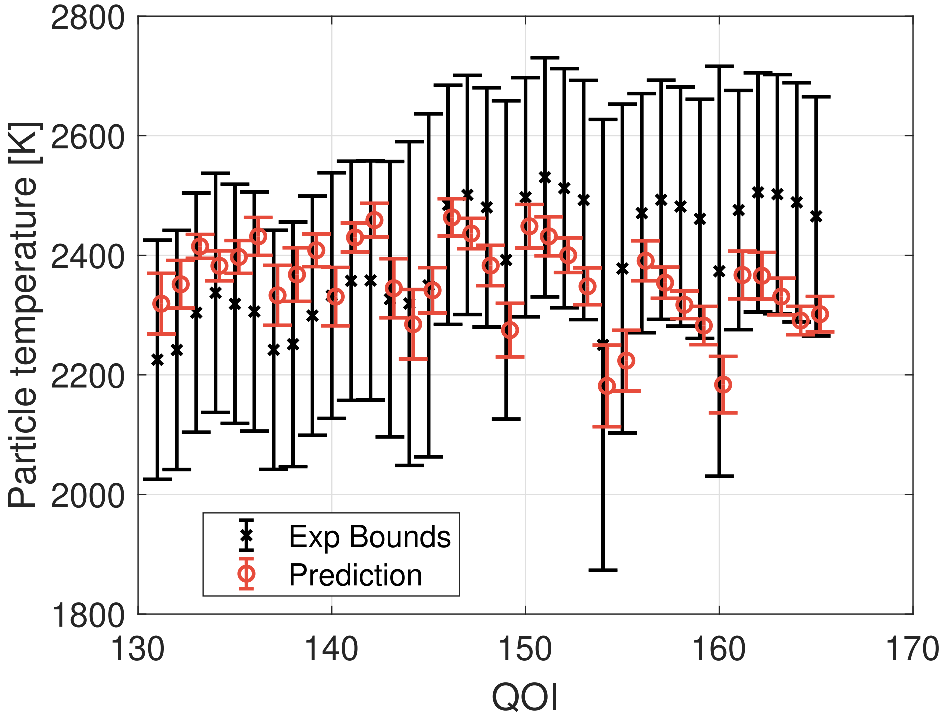


Figure 6.8: Blind prediction of QOIs at 60% $O_2$, 14% $H_2O$, and 26 % $CO_2$. Black interval is the experimental bounds. Red interval is the prediction of the $e$-th QOI (Eq. 3.4) using the feasible set excluding the $e$-th (Eq. B.7). Appendix F.1 contains similar figures for all 12 gas conditions.

# Chapter 7

# Validating a semi-empirical quantum chemistry model PM7

This chapter applies the B2BDC methodology to a semi-empirical quantum chemistry model PM7. The consistency of a homologous series of linear alkanes are examined, where the training data are *chemically accurate*, i.e., they have low uncertainty by the standards of computational chemistry. Potential difficulties using the developed piecewise modeling strategy from Chapter 4 are highlighted. Unlike the previous applications, the experimental data here are believed to be accurate, but there remains a great deal of uncertainty about the adequacy of the computational model itself. An additive model discrepancy is introduced as a potential means of rectifying the model-data inconsistency, diagnosing potential scenario dependencies, and providing additional feedback to domain scientists. The additive model discrepancy is tested on a set of cycloalkanes, that are energetically dissimilar to the training data.

## 7.1 Semi-empirical quantum chemistry

Quantum chemistry methods are the primary theoretical tools for analyzing energies, structures, and properties of molecules. Often quantum chemical methods are divided into two classes, molecular orbital (*ab initio* or semi-empirical) and density functional theory [128]. For molecular orbitals, and specifically *ab initio* methods (which means "from first principles"), one can determine the property for any molecule and state by the solution of the time-independent Schrödinger equation,

$$\widehat{H}\Psi_A = E_A\Psi_A \tag{7.1}$$

where $A$ is the state of interest. To solve Eq 7.1, the energy, $E$ and the wave function $\Psi$, based on the Hamiltonian operator $\widehat{H}$, needs to be found for the molecule of interest. However, accurate *ab initio* calculations are only possible for atoms and small molecules due to the considerable computational costs and knowledge of $\Psi$ [129]. To make the computation more tractable, the Born-Oppenheimer [130] and orbital [131] approximations make assumptions on the dynamics of the electron motion and form of $\Psi$. Following these assumptions, the Schrödinger equation can be simplified into the Hartree-Fock (HF) equations [132, 133], where an electron experiences a field of charge instead of many individual charges. The HF equations are solved through an iterative method called the self-consistent field method (SCF) [134]. SCF can calculate the energy of a molecule by the following procedure: one chooses an initial set of orbitals and calculates all the required electron overlap integrals, these results are used in the HF equations to compute a new set of orbitals. This process is repeated until the solution has converged [135]. For large molecules, the SCF procedure can still be too computationally demanding.

Further simplifications could be made by approximating the electron overlap integrals. These integrals can be fit to experimental data or, sometimes, are neglected [136]. Methods that follow these sets of approximations are known as semi-empirical [136, 137]. One of the lead developers in semi-empirical research was Michael Dewar [138, 139] whose work influences much of the computational software available today [140]. Semi-empirical quantum chemistry methods are similar to the *ab initio* methods but are computationally less demanding. The speed and straight-forward manner in which empirical data are incorporated into these methods are reasons why semi-empirical quantum chemistry finds a wide range of ubiquitous applications and sees ongoing development [141, 142].

Published alongside a semi-empirical method is a recommend (nominal) model parameter [139, 143, 144]. The nominal model parameter is found by minimizing the average prediction error across a diverse set of molecular data [144]. In this chapter, we go beyond an average prediction error analysis by accounting for both parametric and experimental uncertainties.

The semi-empirical quantum chemistry model PM7 [144] is investigated to determine its compatibility with a set of linear alkane training data, whose heats of formation are known to high precision. We will quantify the uncertainty in the model by using the B2BDC analysis. There are 27 adjustable model parameters in PM7 that are used to parameterize the interactions of carbon and hydrogen atoms. Our objective is to determine if there exists a set of PM7 parameter values that satisfies the uncertainty bounds of the training data, and if such a set exists, then quantify the uncertainty in model predictions.

## 7.2 Selection of QOIs and experimental data

Many approaches can be used to validate a computer model [6, 7]. Two key factors are the choice of validation data and the specific experimental feature to use as the QOI. A single homologous series, namely linear alkanes, will be considered in this chapter. This series was chosen due to its simplicity, accumulated knowledge, and that the number of uncertain model parameters does not change with the size of the molecule.

For a given parameterization, PM7 as a model provides many output responses; e.g., vibrational frequencies, ionization potential, HOMO-LUMO energies, the heat of formation, etc. The primary output value of PM7 is the standard heat of formation, which is denoted by $\Delta_f H^\circ_{298}$. A thermochemical network approach by Ruscic et al. [145] combines theory and experimental data to develop accurate thermochemical values for molecular properties, including members of linear alkanes. Table 7.1 shows the heats of formation of the homogeneous series, methane to octane, obtained from the thermochemical network approach. Table 7.1 also includes several large linear alkanes, specifically, decane, dodecane, tetradecane, and octadecane, whose heats of formation were obtained experimentally [146].

The heats of formation of linear alkanes will be used as our QOIs. The experimental bounds are taken as the reported heat of formation ± the uncertainty. Table 7.1 lists the training data, namely the heat of formation of methane, ethane to octane. Experimentally measured alkanes, i.e., decane, dodecane, tetradecane, and octadecane, are only used for prediction. For the sake of simplicity, each QOI model is indexed by the number of carbon atoms.

Table 7.1: List of the linear alkane QOIs, the associated index $e$ (equivalent to the number of carbon atoms in the linear alkane), the reported heat of formation, uncertainty, and reference

| $e$ | $f_e$ | $\Delta_f H^\circ_{298}$ **(kcal/mol)** | $2\sigma$ | **Ref** |
|---|---|---|---|---|
| 1 | $\Delta_f H^\circ_{298}(CH_4)$ | -17.81 | ±0.01 | [145] |
| 2 | $\Delta_f H^\circ_{298}(C_2H_6)$ | -20.06 | ±0.03 | [145] |
| 3 | $\Delta_f H^\circ_{298}(C_3H_8)$ | -25.10 | ±0.05 | [145] |
| 4 | $\Delta_f H^\circ_{298}(C_4H_{10})$ | -30.10 | ±0.06 | [145] |
| 5 | $\Delta_f H^\circ_{298}(C_5H_{12})$ | -34.98 | ±0.07 | [145] |
| 6 | $\Delta_f H^\circ_{298}(C_6H_{14})$ | -39.90 | ±0.08 | [145] |
| 7 | $\Delta_f H^\circ_{298}(C_7H_{16})$ | -44.82 | ±0.11 | [145] |
| 8 | $\Delta_f H^\circ_{298}(C_8H_{18})$ | -49.78 | ±0.16 | [145] |
| 10 | $\Delta_f H^\circ_{298}(C_{10}H_{22})$ | -59.68 | ±0.26 | [146] |
| 12 | $\Delta_f H^\circ_{298}(C_{12}H_{26})$ | -69.53 | ±0.33 | [146] |
| 14 | $\Delta_f H^\circ_{298}(C_{14}H_{30})$ | -79.37 | ±0.43 | [146] |
| 18 | $\Delta_f H^\circ_{298}(C_{18}H_{38})$ | -99.09 | ±0.65 | [146] |

## 7.3 Challenges in surrogate modeling

As aforementioned, the B2BDC methodology uses polynomial surrogate models to represent the response from an underlying simulation model. Using surrogate models makes tasks like optimization and sensitivity analysis much more efficient due to inexpensive evaluations. However, constructing surrogates models for the PM7 model was challenging with $\Delta_f H^\circ_{298}(C_4H_{10})$ serving as our example.

Experience has shown that surrogate models are only accurate (to within the experimental uncertainty) on small domains of the parameter space for the PM7 model. To illustrate, we compare the surrogate model fitting error of a quadratic polynomial and Gaussian process surrogate on a shrinking domain $\mathcal{H}_k = [x_{\text{nom}} \pm k \times \rho]$ where $k \in [0, 1]$, shown in Figure 7.1. The model parameter value, $x_{\text{nom}}$ is the nominal (pre-calibrated) set of model parameters for the PM7 model [144], $\rho$ was determined by changing the $i$-th component of the model parameter $x_{\text{nom}}$, with all other parameters remained fixed, such that the interval defined by $x_{\text{nom},i} + \rho_i$ produced a 10 kcal/mol change in $\Delta_f H^{\circ}_{298}(C_4H_{10})$. The nominal model parameter and $\rho$ are shown in Table 7.2.

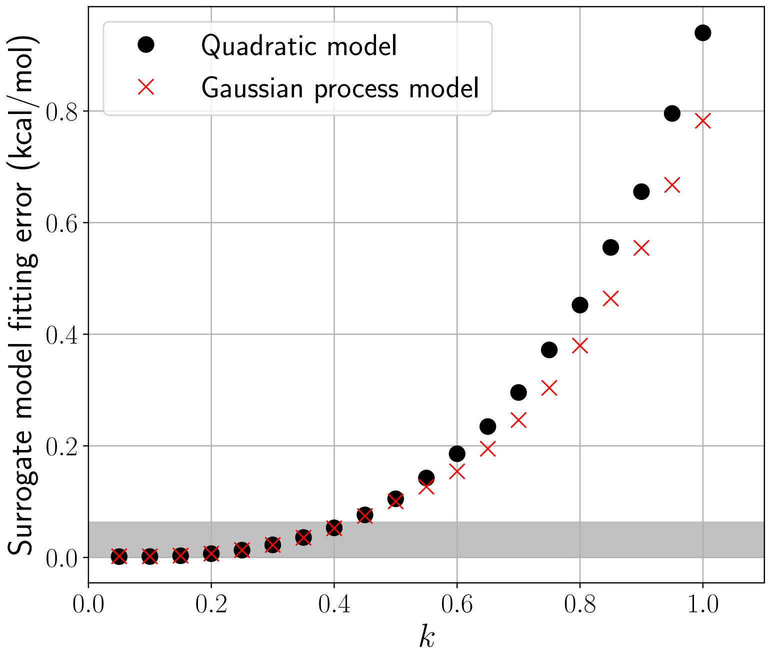


Figure 7.1: Surrogate model fitting error for the heat of formation of $C_4H_{10}$ evaluated on a shrinking domain $\mathcal{H}_k = [x_{\text{nom}} \pm k \times \rho]$, with $k \in [0, 1]$. The nominal PM7 value, $x_{\text{nom}}$, and $\rho$ are reported in Table 7.2. As the domain shrinks, by reducing $k$, the surrogate model fitting error decreases to a point where the error falls below the experimental uncertainty (shown in grey) of 0.06 kcal/mol. Comparison between a quadratic surrogate (black dots) and a Gaussian process model (red crosses) with a constant mean and a squared exponential covariance function is shown. 10-fold cross-validation was used to estimate the fitting error.

For each $k$, 7,500 Latin Hypercube samples [122] were generated in $\mathcal{H}_k$ to construct both surrogate models. The Gaussian process was implemented using MATLAB's *fitrgp* function [64]. To attain fitting error below our target, the experimental uncertainty of $\pm 0.06$ kcal/mol, the initial region needed to be reduced by the factor of $k = 0.4$. The reduced domain $\mathcal{H}_{0.4}$

Table 7.2: PM7 nominal parameter vector and the $\rho$ associated with a 10 kcal/mol change in the heat of formation of $C_4H_{10}$, where each parameter value was perturbed, one-at-a-time from its nominal value.

| Parameter | $x_{\text{nom}}$ | $\rho$ |
|---|---|---|
| $\text{USS}_\text{H}$ | -11.07 | 0.29033 |
| $\text{BETAS}_\text{H}$ | -8.3897 | 0.069276 |
| $\text{ZS}_\text{H}$ | 1.2602 | 0.017434 |
| $\text{GSS}_\text{H}$ | 14.15 | 0.36839 |
| $\text{FN11}_\text{H}$ | 0.17785 | 0.009959 |
| $\text{FN21}_\text{H}$ | 1.4287 | 0.33716 |
| $\text{FN31}_\text{H}$ | 0.99132 | 0.077376 |
| $\text{ALPB}_\text{H}$ | 4.0512 | 0.54877 |
| $\text{XFAC}_\text{H}$ | 2.8456 | 4.875 |
| $\text{USS}_\text{C}$ | -51.373 | 0.11427 |
| $\text{UPP}_\text{C}$ | -40.135 | 0.084482 |
| $\text{BETAS}_\text{C}$ | -14.415 | 0.12922 |
| $\text{BETAP}_\text{C}$ | -7.8937 | 0.067632 |
| $\text{ZS}_\text{C}$ | 1.9422 | 0.012886 |
| $\text{ZP}_\text{C}$ | 1.7087 | 0.007471 |
| $\text{GSS}_\text{C}$ | 12.347 | 0.13671 |
| $\text{GSP}_\text{C}$ | 11.933 | 0.18749 |
| $\text{GPP}_\text{C}$ | 10.452 | 0.8254 |
| $\text{GP2}_\text{C}$ | 9.3855 | 0.031369 |
| $\text{HSP}_\text{C}$ | 0.80263 | 0.11697 |
| $\text{FN11}_\text{C}$ | 0.045888 | 0.005604 |
| $\text{FN21}_\text{C}$ | 5.0371 | 1.5601 |
| $\text{FN31}_\text{C}$ | 1.5887 | 0.12723 |
| $\text{ALPB}_\text{HC}$ | 1.0387 | 0.007624 |
| $\text{XFAC}_\text{HC}$ | 0.20458 | 0.002002 |
| $\text{ALPC}_\text{C}$ | 2.6557 | 0.024061 |
| $\text{XFAC}_\text{C}$ | 0.93782 | 0.034605 |

is $5.5 \times 10^{10}$ times smaller by volume as compared to the original domain $\mathcal{H}_1$. Based on this result, I concluded that it would require an intractable number of surrogate models to explain the behavior of PM7 over $\mathcal{H}$ within the desired accuracy.

## Piecewise surrogate modeling

The developed piecewise modeling strategy (discussed in Chapter 4) can efficiently develop polynomial surrogate models on disjoint domains of the parameter space. This strategy is more efficient than traditional piecewise modeling by taking advantage of the experimental data. However, the piecewise modeling strategy was ineffective for the PM7 model.

The piecewise modeling strategy begins by fitting a quadratic surrogate model over each subdomain. As shown in Figure 7.1, quadratic surrogate models are only accurate on small domains; therefore, a significant amount of fitting error is propagated forward by expanding the associated experimental bounds. Evaluation of the scalar consistency measure then (in almost all cases) identifies the subdomain as consistent with the expanded experimental bounds. Constraints imposed by the experimental bounds are significantly relaxed when adding a fitting error which is nearly an order of magnitude larger than the experimental uncertainty. This result emphasizes that the developed piecewise modeling strategy can have difficulties invalidating large volumes of the prior domain when the fitting error is significant.

## 7.4 Direct sampling for consistency

Considering the challenges discussed in the preceding section, and due to the speed of PM7, this analysis will forgo the development of surrogate models. Both consistency and prediction will be assessed by directly sampling $\mathcal{H}$. A sample-based approach can only approximate the feasible set and provide an inner estimate to the model prediction. Proofs of inconsistency, as described in Chapter 3.2, are lost when using a sample-based approach.

The computational chemistry program MOPAC [147] was used to evaluate the PM7 model and associated model parameters. The input molecular geometries were obtained via optimization with the PM7 nominal parameter vector (Table 7.2), which remained frozen afterward, during the subsequent direct sampling. The freezing of the molecules geometries was done to avoid any unphysical reorganization or distortion. The consequences of freezing the molecular geometries with the nominal PM7 parameter vector are two-fold. First, using geometry optimization for a given parameter vector can (and usually does) result in different heats of formation as compared to the fixed geometry. Second, parameter vectors that are feasible with respect to the heats of formation, do not guarantee reaching a physically meaningful geometry. All feasible parameter vectors, which would produce unphysical geometries, should ultimately be removed from the feasible set.

The choice of the prior parameter domain, $\mathcal{H}$, determines all subsequent analysis and hence it needs to be well motivated. Our goal is to assess the predictivity of PM7 for large linear alkanes while being consistent with experimental data for small alkanes. Due to limited high-accuracy data from Ruscic et al. [145], heptane and octane ($C_7H_{16}$ and $C_8H_{18}$), the two largest alkanes from the training data, were set aside to assess predictivity. Although the physical properties of methane and ethane may represent the homologous series, the chemical properties, i.e., the energetics, are very different as seen through group additivity [148]. For this reason, methane and ethane were set aside from the analysis to not bias the result. Propane and butane were also not included in the initial region search since *a priori* it was not known if a feasible set exists. Based on these considerations, the determination of $\mathcal{H}$ was prioritized on feasible samples for pentane and hexane, which will be denoted as

$\mathcal{F}_{5:6}$. The subscript `5:6` is used to indicate a feasible set was formed with the 5th through the 6th QOIs; i.e., pentane and hexane.

To begin our search, a non-linear programming solver, *fmincon* [64], was used to minimize the following objective function, $\|f_{5:6}(x) - y_{5:6}\|_2^2$, where, $f_{5:6}(x)$ are the heats of formation of pentane and hexane computed by PM7 at the parameter vector $x$. $y_{5:6}$ are the reported heats of formation for pentane and hexane from Table 7.1. The local optimal parameter vector found, $x_{\text{opt}}$, was feasible for pentane and hexane; therefore, $x_{\text{opt}} \in \mathcal{F}_{5:6}$. A volume was taken around $x_{\text{opt}}$ to collect samples of $\mathcal{F}_{5:6}$. For each sampled point, the heats of formation of pentane and hexane were computed by PM7. $2 \times 10^5$ samples were generated by employing a space-filling Latin Hypercube design from the volume $[x_{\text{opt}} \pm x_{\text{opt}} \times 1 \times 10^{-3}]$ and 65 samples were found to be within $\mathcal{F}_{5:6}$.

Principal component analysis (PCA) [149] was conducted to identify a rotated coordinate system around the samples of $\mathcal{F}_{5:6}$, where all principal components were preserved. Considering the arbitrary choice of the volume used in the sampling of $\mathcal{F}_{5:6}$, each PCA direction was extended by ten times that of the $\mathcal{F}_{5:6}$ samples enabling a larger sample region to be considered. A Latin Hypercube design was taken to generate uniform samples in the rotated and extended space.

In total, 5.76 million samples were generated uniformly within the PCA-rotated volume. PM7 was used to evaluate the heats of formation of 9 linear alkanes, $CH_4$ to $C_9H_{20}$. Shown in Table 7.3 are the extreme parameter values from the generated samples. Rejection sampling was used to determine if a sample was feasible for each set of experimental bounds.

From the 5.76 million samples, there were 164,569 samples that were feasible for at least one alkane in the training set. The number of feasible samples found quickly dropped when considering feasibility with multiple alkanes. For example, 19,167 samples were feasible with at least two alkanes, 6,193 samples feasible with at least three alkanes, 3,110 samples feasible with at least four alkanes, 1,989 samples feasible with at least five alkanes, and 169 samples feasible with at least six alkanes.

## 7.5 Samples of the feasible sets

For each of the eight alkanes, a feasible parameter vector could be found that would predict its heat of formation within the experimental bounds. Feasible parameters were also found for all pairwise combinations of alkanes, i.e., a parameter vector could be found that would predict the heat of formation for any two alkanes within their respective experimental bounds. The percentage of feasible samples found for each pair of alkanes in the training data set is reported in Table 7.4. Methane had the smallest percentage of feasible samples, and the largest was for octane. This difference in proportions of feasible samples could be due to the uncertainty bounds associated to octane being nearly an order of magnitude larger than methane and the difference in the molecular structure of methane compared to all others

Table 7.3: Extreme parameter values from the search region and parameter vector of $\mathcal{F}_{2:8}$ found via global optimization with a genetic algorithm.

| Parameter | min($x$) | max($x$) | $x_{ga}$ |
|---|---:|---:|---:|
| $USS_H$ | -14.815 | -9.5475 | -14.577 |
| $BETAS_H$ | -10.444 | -6.383 | -7.3947 |
| $ZS_H$ | 0.85244 | 1.5053 | 1.1558 |
| $GSS_H$ | 10.953 | 15.822 | 15.096 |
| $FN11_H$ | 0.14962 | 0.23173 | 0.21973 |
| $FN21_H$ | 1.2161 | 1.4849 | 1.3555 |
| $FN31_H$ | 0.83875 | 1.0458 | 0.89818 |
| $ALPB_H$ | 3.9629 | 4.8467 | 4.0908 |
| $XFAC_H$ | 2.3077 | 2.7852 | 2.6264 |
| $USS_C$ | -52.553 | -45.61 | -50.896 |
| $UPP_C$ | -43.679 | -37.301 | -42.679 |
| $BETAS_C$ | -15.285 | -11.804 | -12.204 |
| $BETAP_C$ | -9.2246 | -6.7715 | -8.0123 |
| $ZS_C$ | 1.5914 | 2.2229 | 1.9807 |
| $ZP_C$ | 1.501 | 2.0466 | 1.5976 |
| $GSS_C$ | 10.314 | 13.662 | 12.653 |
| $GSP_C$ | 9.7585 | 13.092 | 10.104 |
| $GPP_C$ | 9.3945 | 12.464 | 9.8942 |
| $GP2_C$ | 8.6452 | 11.014 | 9.8562 |
| $HSP_C$ | 0.66956 | 0.77601 | 0.77597 |
| $FN11_C$ | 0.045528 | 0.055036 | 0.051751 |
| $FN21_C$ | 4.3632 | 5.1283 | 4.8582 |
| $FN31_C$ | 1.4298 | 1.7154 | 1.6624 |
| $ALPB_{HC}$ | 0.76974 | 1.1501 | 1.0878 |
| $XFAC_{HC}$ | 0.15181 | 0.21488 | 0.17728 |
| $ALPC_C$ | 2.3296 | 3.0615 | 2.5316 |
| $XFAC_C$ | 0.71959 | 0.96953 | 0.75887 |

in the homologous series. Table 7.4 shows the largest fraction of feasible samples were for $C_8H_{16}$, which is counterintuitive as the sampling efforts were focused on the feasible sets of $C_5H_{12}$ and $C_6H_{14}$.

From the generated samples, there was no single parameter vector that was able to simultaneously satisfy the uncertainty bounds of all the training data; i.e., $\mathcal{F}_{1:8} = \varnothing$. Thus, the PM7 model and the training data were found to be mutually inconsistent. The inability to find a parameter vector feasible for all alkanes does not prove the non-existence of such a point.

There were 4,020 samples of $\mathcal{F}_{5:6}$, the alkanes which we had focused our search on.

Table 7.4: Percentage of feasible points found (out of 5.76 million samples) for each alkane and pair of alkanes. Cells are colored corresponding to the numerical value, red indicating a larger value and green for a smaller value.

| | $CH_4$ | $C_2H_6$ | $C_3H_8$ | $C_4H_{10}$ | $C_5H_{12}$ | $C_6H_{14}$ | $C_7H_{16}$ | $C_8H_{18}$ |
|---|---|---|---|---|---|---|---|---|
| $CH_4$ | 0.2647 | 0.0036 | 0.0028 | 0.0027 | 0.0026 | 0.0019 | 0.0025 | 0.0030 |
| $C_2H_6$ | | 0.4253 | 0.0148 | 0.0113 | 0.0093 | 0.0077 | 0.0088 | 0.0100 |
| $C_3H_8$ | | | 0.4063 | 0.0318 | 0.018 | 0.0133 | 0.0140 | 0.0159 |
| $C_4H_{10}$ | | | | 0.4441 | 0.0572 | 0.0317 | 0.0297 | 0.0307 |
| $C_5H_{12}$ | | | | | 0.4172 | 0.0698 | 0.0501 | 0.0461 |
| $C_6H_{14}$ | | | | | | 0.3869 | 0.1070 | 0.0750 |
| $C_7H_{16}$ | | | | | | | 0.4723 | 0.1915 |
| $C_8H_{18}$ | | | | | | | | 0.5723 |

Prediction using the found feasible sets are shown in the following section. No samples were found from $\mathcal{F}_{2:8}$ from direct sampling. However, using a genetic algorithm via MATLAB's *ga* function (and nearly 750+ CPU hours), a single parameter vector $x_{ga}$, was obtained from $\mathcal{F}_{2:8}$ and is shown in Table 7.3. A similar approach was taken to obtain a sample from $\mathcal{F}_{1:8}$, although, none could be found.

## 7.6 Uncertainty in model predictions

In this section, we examine the uncertainty of predictions for larger alkanes using samples consistent with experimental data for smaller alkanes. Samples consistent with the experimental data for smaller alkanes were used for prediction of the larger alkane forming the histograms shown in Figure 7.2. The width of the histograms constitutes the prediction uncertainty. The predictions are only inner approximations as samples were only generated from a portion of the parameter space. Thus, the predicted uncertainty can be no smaller than what is presented.

Figure 7.2(a) depicts the prediction of the heat of formation of decane. In blue are samples from $\mathcal{F}_{5:6}$, i.e, parameter vectors that were feasible for both $C_5H_{12}$ and $C_6H_{14}$. The width of the distribution is 1.6 kcal/mol, which is larger than that of the experimental uncertainty of 0.52 kcal/mol (shown in grey) and larger than the chemical accuracy (1 kcal/mol). To further reduce the prediction uncertainty, additional constraints can be imposed by considering samples that are also feasible with smaller alkanes. Samples of $\mathcal{F}_{4:6}$ and $\mathcal{F}_{3:6}$, shown in red and yellow, respectively, are indeed capable of reducing the predicted uncertainty, and in the case of $\mathcal{F}_{3:6}$ producing a prediction 68% smaller than that of $\mathcal{F}_{5:6}$. Figures 7.2(b) and 7.2(c) show similar predictions for the heats of formation of dodecane and octadecane.

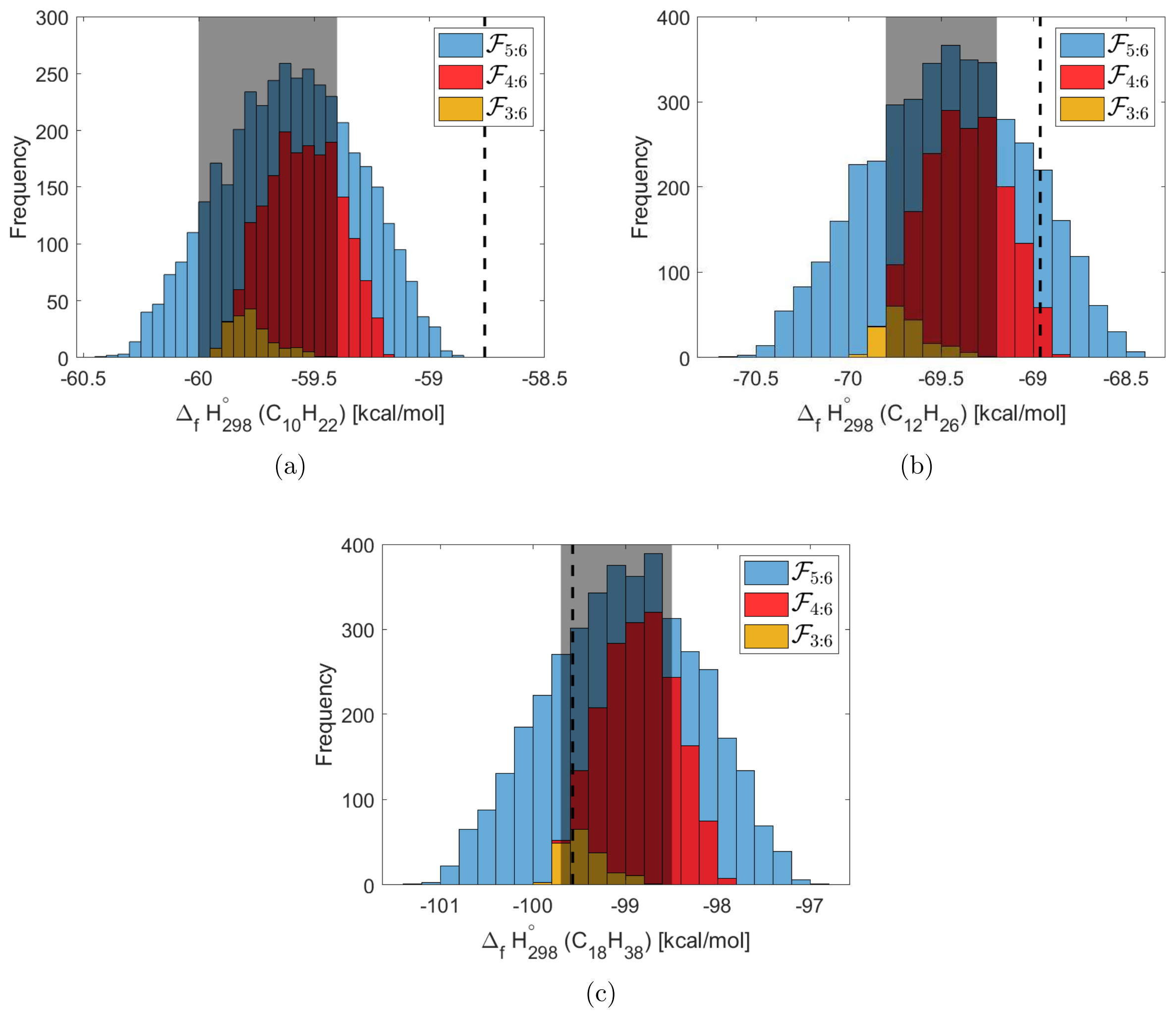


Figure 7.2: Heat of formation of a larger alkane, predicted from the feasible parameters of smaller alkanes. The grey shaded region is the experimental interval for the respective alkane from Table 7.1. The black vertical dashed line is the PM7 model prediction using the nominal model parameter, Table 7.2. In each panel, samples of $\mathcal{F}_{5:6}$, $\mathcal{F}_{4:6}$ and $\mathcal{F}_{3:6}$ are used. Panel a: the heat of formation of decane, Panel b: heat of formation of dodecane, and Panel c: the heat of formation of octadecane.

## 7.7 Classification of feasible points

The B2BDC feasible set is defined by polynomial surrogate models and constraints; however, because of the sample-based approach, the feasible set could not be represented with polynomial models and constraints. This well-defined representation of the feasible set was lost due to the inability to develop accurate surrogate models. An alternative approach is proposed to approximate the feasible set by using a model, trained on the sampled data.

In this section, the data collected via direct sampling are used to train a binary classifier that distinguishes a "feasible" class of parameter values from the "infeasible" ones. A feasible point could be associated with a specific feasibility label, such as $\mathcal{F}_{5:6}$, or any feasibility label. Given the unbalanced nature of the feasible and infeasible classes, i.e., over $5 \times 10^6$ samples with an infeasible label and only a few thousand samples belonging to any specific feasibility label, we choose the latter option and constructed the "feasible" class from all points that are feasible for at least one alkane in the training set.

A Random Forest classifier was selected as it can emulate the decision boundary for non-convex and even non-contiguous feasible sets. The classification was performed using the implementation in the Scikit-learn library [150], with equal misclassification costs. It is easy for one to verify that the PM7 feasible set is non-convex, a line segment between two feasible samples is not entirely feasible; however, non-contiguous feasible sets were not confirmed for PM7 as it is more challenging to prove. Both "feasible" and "infeasible" classes were down-sampled to $10^5$ to handle the class imbalance. Down-sampling was performed by using a randomly drawn $10^5$ samples from each class [151]. Performance of the classifier was evaluated by 5-fold cross-validation via the construction of the Receiver Operating Curve (ROC) and computation of the Area Under Curve (AUC), shown in Figure 7.3.

With an AUC value of 0.68, there is indeed a non-random structure in the feasible set recovered via direct sampling. Therefore, it is possible to train a classifier that fulfills the same function as the inequality constraints which defines the feasible set. Of course, this is only an approximation to the feasible set. A better performance metric of the classifier should be achieved for this classification-based feasible set to be practical. One route toward this goal is to obtain a sample in the parameter space that conveys a better representation of the spatial extents of the non-convex feasible set found via sampling on a Latin Hypercube. Samples near the boundary or surface of the feasible set (i.e., samples that straddle the decision boundary) are more informative for classification than a collection of samples from the interior of the set.

## 7.8 Diagnosing inconsistency with model discrepancy

The previous analysis did not provide evidence that the semi-empirical quantum chemistry model PM7 was consistent with a homologous series of linear alkanes; i.e., no model parameter could simultaneously predict the heat of formation of methane to octane within the accuracy of the training data. This result is not surprising given the empirical nature of PM7 discussed in Section 7.1. Given the high accuracy of the training data, one potential source of the inconsistency is the computational model itself.

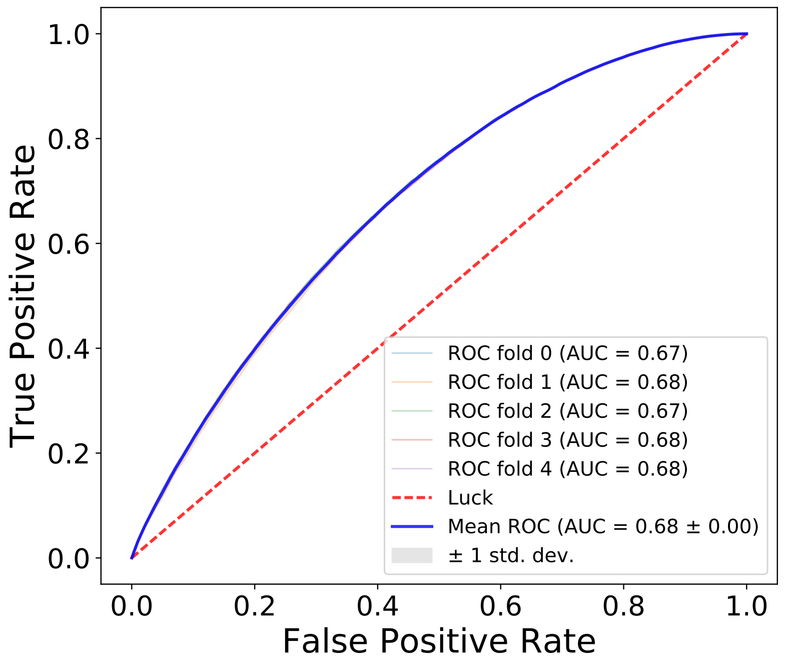


Figure 7.3: ROC and AUC for the binary classification (Random Forest model) of the feasibility of the points from the PM7 parameter space obtained via direct sampling. “Feasible” class includes points that are feasible for at least one training data point, “Infeasible” class includes points that are not feasible for any training data point. Equal misclassification costs were used. Both classes were down-sampled to $10^5$ samples. Classification performance was assessed by 5-fold cross-validation using $5 \times 10^3$ ensemble of random trees.

When a computational model differs from the physical process that generated the experimental data, a discrepancy or error exists. This difference can be compensated for by including a *model discrepancy* [22]. A model discrepancy function, $\delta$, also referred to in the literature as a model error [152] or model inadequacy [153], is a means of quantifying the observed disagreement between models and data. By far the most popular strategy for modeling a discrepancy is the Kennedy-O’Hagan (KO) approach [22], which adds a model discrepancy function to the model output. In the classical KO approach, the discrepancy function takes the form of a non-parametric model, e.g., a Gaussian process, and is a function of the scenario parameters.

More recent approaches to model discrepancy have gone beyond the additive KO approach. An embedded model discrepancy changes the input model parameters by incorporating a probabilistic model [154]. Intrusive embedding of the model discrepancy alters the underlying computational model [153]. Nevertheless, with compiled codes, e.g., MOPAC, invasive approaches are not amenable. Irrespective of how model discrepancy is accounted for, prediction outside of the validation data will always be risky.

## Implementation

Ideas from the KO approach, which are suitable for the B2BDC methodology, are implemented for the PM7 model. An additive model discrepancy will act as a diagnostic tool for identifying if a dependency in the scenario parameters could rectify the *a priori* model-data inconsistency. The model discrepancy will adjust the model output, $f_e(x)$, by,

$$f_e(x) + \delta(x, x_{s,e}) \tag{7.2}$$

where $\delta(x, x_{s,e})$ is a model of the discrepancy depending on the scenario parameters $x_{s,e}$, and in some instances the model parameter $x$. In this example, the scenario parameter $x_{s,e}$ is the geometry of the molecule, where each scenario is a different molecule. $\delta(x, x_{s,e})$ will be represented as a linear combination of basis functions; i.e., $\delta(x, x_{s,e}) = \sum_{i=1}^{m} c_i \phi(x, x_{s,e})$, where $\phi$ are basis functions and $c_i$ are unknown coefficients to the discrepancy function. This form of the $\delta(x, x_{s,e})$ is chosen, specifically a deterministic function with a linear dependence in $c_i$, such that it can be incorporated within the B2BDC framework with no additional modifications.

One known issue with the KO approach is its difficulty in identifying the correct calibrating model parameters without realistic prior information on the model discrepancy [155]. Modifying the computational model output by adding a model discrepancy often destroys essential physical properties of the model, leading to unphysical behavior in the output [154–156]. Introducing constraints on the discrepancy function might prevent unphysical behavior; however, the various types of constraints and their respective influence is outside the scope of this analysis.

We will use the model discrepancy function to determine if additional structured dependence, through $\delta(x, x_{s,e})$, could resolve the prior inconsistency in PM7 for predicting a set of linear alkanes. The following modifications are necessary to integrate the model discrepancy into the definition of a feasible set:

$$\widetilde{\mathcal{F}} = \bigcap_{e=1}^{N} \widetilde{\mathcal{F}_e} = \{(x, c) \in \mathcal{D} : L_e \leq f_e(x) + \sum_{i=1}^{m} c_i \phi(x, x_{s,e}) \leq U_e, \text{ for } e = 1, 2, \ldots, N.\} \tag{7.3}$$

where the model discrepancy parameter $c \in \mathbb{R}^m$, such that the prior, $\mathcal{D} \subset \mathbb{R}^{n+m}$, is for all uncertain model parameters and discrepancy parameters. We denote, $\widetilde{\mathcal{F}}$, as the feasible set including model discrepancy to differentiate it from the feasible set without discrepancy, $\mathcal{F}$.

## 7.9 Charge-based group correction

Choosing the basis functions $\phi(x, x_{s,e})$ to incorporate into the model discrepancy function is up to the modeler. The discrepancy function should be well-motivated and physically meaningful. We follow the work of Wang and Frenklach [157] which investigated the prediction of the standard heats of formation for benzenoid aromatics using the semi-empirical quantum chemistry model AM1 [139]. In their work, the semi-empirical quantum chemistry predictions deviated from the experimental measurements. A constant multiplier was introduced to each functional group [148] to account for the observed deviation between quantum chemistry prediction and data. A recent study by Allison et al. [158] had followed this approach by correcting for the systematic deviations in the prediction of polycyclic aromatic hydrocarbons by density functional theory and Gaussian-3 model chemistry methods.

Building on this logic, the discrepancy function will be based on a group correction. However, instead of a constant multiplier to each functional group, we will use another output from the PM7 model, namely the atomic charges. The electric charge of the carbon atoms will form the a vector of basis functions $\phi(x, x_{s,e})$ for the model discrepancy function. For straight-chained alkanes, there are two functional groups: methyl (C/C/H3) and methylene (C/C2/H2). We refer to these groups by the electric charge of the carbon atoms, denoted as $Q_{C3,j}(x)$ or $Q_{C2,k}(x)$. Figure 7.4 shows $Q_{C3,j}(x)$, the electric charge for the carbon atom in the methyl group. This charge was calculated by PM7 for the parameter vector $x$. Similarly, the electric charge associated with the carbon atom of a methylene group is shown in Figure 7.5, denoted as $Q_{C2,k}(x)$.

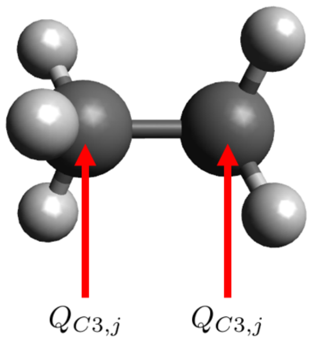


Figure 7.4: Illustration of ethane with the $Q_{C3,j}$ electric charges labeled.

The model discrepancy function based on group corrections of the electric charges will be:

$$\delta(x, x_{s,e}) = \sum_{i=1}^{m} c_i \phi_i(x, x_{s,e}) = c_0(x) + c_2(x) \sum_{k=1}^{n_k} Q_{C2,k}(x) + c_3(x) \sum_{j=1}^{n_j} Q_{C3,j}(x) \qquad (7.4)$$

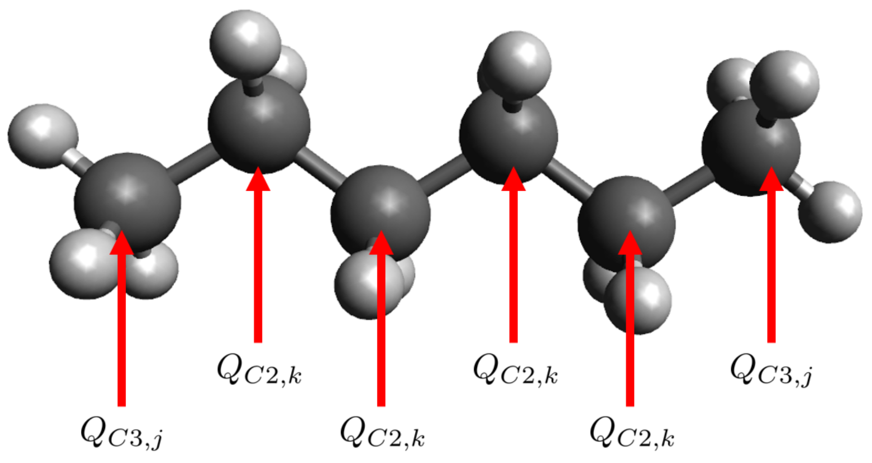


Figure 7.5: Illustration of n-hexane with all carbon atoms labeled with their associated electric charges, $Q_{C2,k}$ and $Q_{C3,j}$.

where $n_j$ is the total number of methyl (C/C/H3) groups, $n_k$ is the total number of methylene (C/C2/H2) groups in a molecule. Therefore,

$$\phi(x, x_{s,e}) = \left[1, \sum_{k=1}^{n_k} Q_{C2,k}(x), \sum_{j=1}^{n_j} Q_{C3,j}(x)\right].$$

Since the scenario parameter $x_{s,e}$ is the molecular structure, both $n_j$ and $n_k$ depends on $x_{s,e}$. To clarify the values taken by $n_k$ and $n_j$, all linear alkanes have two methyl groups, thus $n_j = 2$; whereas for cycloalkanes molecules that have no methyl groups, $n_j = 0$.

The model discrepancy parameter, $c_0$, enables the model discrepancy function to correct for a systematic bias; i.e., if the computational model under or overpredicts all QOIs by a fixed amount. Incorporating additional parameters to the discrepancy function increases its complexity and flexibility, enabling the model discrepancy to correct for more irregular differences. In the following section, we apply the additive model discrepancy function to a set of parameter vectors infeasible with PM7 for ethane to octane.

## 7.10 Model discrepancy results

The analysis begins by determining if the proposed model discrepancy function can resolve the prior model-data disagreement. Samples that are feasible with at least one alkane were used. If the parameter vector $x$ and the proposed model discrepancy function $\delta(x, x_{s,e})$ are consistent with the experimental data from Eq. 7.3, then there is a parameter $c = [c_0, c_2, c_3]$ such that, $f_e(x) + \sum_{i=1}^{3} c_i\phi_i(x, x_{s,e})$ evaluates the heat of formation within the experimental bounds for the training data. Calculation of the heat of formation from PM7 for the $e$-th QOI is denoted by $f_e(x)$. $5 \times 10^3$ samples were randomly drawn from the collection of 164,569 samples feasible with at least one alkane (Section 7.4). For each model parameter $x$, the following feasibility problem was evaluated:

$$\text{find } \; c \in \mathbb{R}^m \; \text{ s.t. } \; L_e \leq f_e(x) + \sum_{i=1}^{m} c_i \phi_i(x, x_{s,e}) \leq U_e \; \text{ for } e = 2, \ldots 8. \tag{7.5}$$

If Eq. 7.5 is feasible then there are coefficients $c$ such that the feasible set, $\widetilde{\mathcal{F}}$ is non-empty. From the $5 \times 10^3$ samples, 4,215 samples (84.3%) were feasible for some value of $c = [c_0, c_2, c_3]$. In other words, 84.3% of the model parameters considered, there exists a discrepancy coefficient $c$ that can correct the PM7 model output to within the uncertainty of the training data for ethane to octane. Methane was not included in the training set as it does not share any functional groups with the larger linear alkanes. Methane would require its own functional group for correction, which can be investigated in a later study. The additional flexibility from the model discrepancy function successfully brought the PM7 model to consistency with a set of linear alkane training data.

## Prediction with an additive model discrepancy

Predicting the heats of formation of large straight-chain alkanes using the model discrepancy function are shown in Figure 7.6. These prediction molecules are similar to that of the training data, although much longer. Prediction amounts to establishing the following interval,

$$\left[ \min_{(x,c) \in \widetilde{\mathcal{F}}} f_p(x) + \sum_{i=1}^{m} c_i \phi_i(x, x_{s,p}), \; \max_{(x,c) \in \widetilde{\mathcal{F}}} f_p(x) + \sum_{i=1}^{m} c_i \phi_i(x, x_{s,e}) \right], \tag{7.6}$$

where $x$ is a random sample of $5 \times 10^3$. Eq 7.6 is solved for each feasible model parameter $x$, thus, 4,215 intervals are found. The black intervals shown in Figure 7.6 represent the extrema across all 4,215 intervals.

Figure 7.6(a) shows the prediction of decane. The prediction interval is approximately 0.6 kcal/mol and overlaps with the experimental bounds from Table 7.1 shown in grey. Prediction using the nominal PM7 model parameter is shown as a black vertical dashed line that is at odds with the predicted interval and experimental bounds. In the cases considered, the prediction interval produced by PM7 and the charged-based discrepancy function was in agreement with the experimental bounds. As one may expect, the predicted uncertainty increases as we consider alkanes further from the training data. For example, octadecane ($C_{18}H_{38}$) has a prediction interval of approximately 2 kcal/mol, double that of the experimental bounds. These prediction results show that the PM7 model with a charge-based model discrepancy agrees qualitatively with the experimental data.

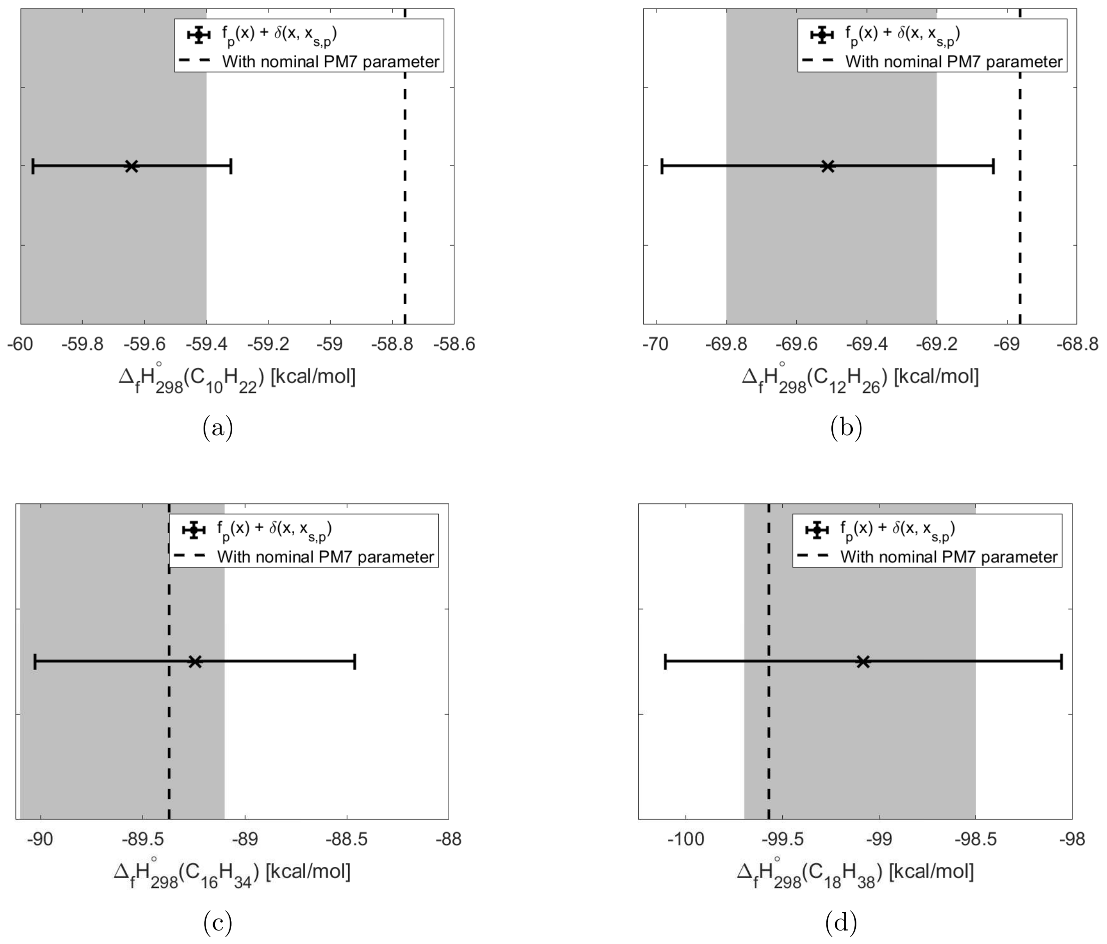


Figure 7.6: Prediction of the heat of formation using the PM7 model and an additive model discrepancy i.e., $f_p(x) + \delta(x, x_{s,p})$, for large straight-chained alkanes. The feasible set of model parameters were constrained by data for linear alkanes, ethane to octane. The black interval is the extrema of the prediction intervals from Eq. 7.6. Grey shaded regions are the experimental bounds reported in Table 7.1. The vertical black dashed line is the heat of formation evaluated by the nominal PM7 parameter value, $f_p(x_{\text{nom}})$.

## Prediction of cycloalkanes

In this section, we investigate if the learned discrepancy function can be transferred to a broader class of molecules; i.e., to make predictions of molecules dissimilar to the training data. In the previous section, we showed that model prediction with the learned model discrepancy produced intervals that were physically meaningful and coincided with the experimental bounds for large linear alkanes. In this section, the discrepancy function is tested on a set of small cycloalkanes that are different, specifically the strain energy, as compared to the training data.

Table 7.5: List of the cycloalkane QOIs, the reported heat of formation, and uncertainty [145]

| $f_e$ | $\mathbf{\Delta_f H^\circ_{298}}$ **(kcal/mol)** | $2\sigma$ |
|---|---|---|
| $\Delta_f H^\circ_{298}(C_3H_6)$ | 12.82 | $\pm0.11$ |
| $\Delta_f H^\circ_{298}(C_4H_8)$ | 6.65 | $\pm0.10$ |
| $\Delta_f H^\circ_{298}(C_5H_{10})$ | -18.55 | $\pm0.11$ |
| $\Delta_f H^\circ_{298}(C_6H_{12})$ | -29.45 | $\pm0.09$ |

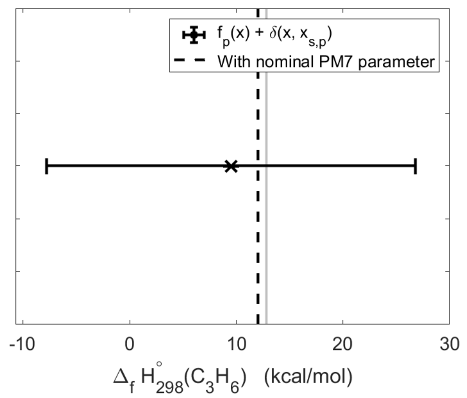
We begin by considering a set of cycloalkanes shown in Table 7.5. Due to the geometric and energetic differences between straight-chain alkanes and cycloalkanes, cyclohexane was included in the training data for $\widetilde{\mathcal{F}}$. Therefore, the feasible model parameters, $(x, c)$ must produce the heat of formation within the uncertainty bounds for linear alkanes (ethane to octane) and cyclohexane.

The same $5 \times 10^3$ samples of $x$ are used to evaluate the consistency of the model discrepancy function. Only 749 (14.98%) of the samples were feasible for the training data that included cyclohexane. Figure 7.7 shows the prediction interval for cyclopropane, cyclobutane, and cyclopentane using the feasible samples. Figures 7.7(a) and 7.7(b) show the prediction interval for both cyclopropane and cyclobutane that spans both positive and negative heats of formation, which is not physically meaningful. The predicted interval for these cycloalkanes spanned over 20 kcal/mol. This result illustrates that the learned model discrepancy was incapable of making predictions for small cycloalkane rings, which is to be expected.

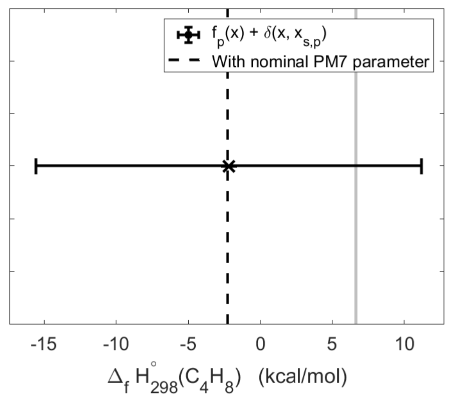
The discrepancy function had no means to learn about (or account for) strain energy that is present in the small cycloalkanes. Applying the discrepancy function on molecules dissimilar to the training data, unphysical results could occur. Investigating additional basis functions to account for the molecular strain that is present in cycloalkanes and including additional training data are topics for a future study.

## 7.11 Conclusion

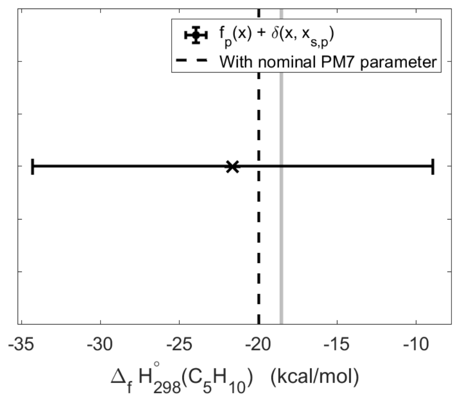
Uncertainty quantification of a semi-empirical quantum chemical model PM7 was performed by using a sample-based approach. The obtained results did not show evidence of model consistency, even in the “best-case” scenario, where we considered heats of formation of a small family of linear alkanes. The lack of evidence of model consistency with experimental data may imply that either the experimental data has some bias or the model has a deficiency. Given the quality of the experimental data, we tend to think the source of the model inconsistency resides with the PM7 model itself.



(a)



(b)



(c)

Figure 7.7: Prediction of the heat of formation using the PM7 model and an additive model discrepancy i.e., $f_p(x)+\delta(x, x_{s,p})$, for small cycloalkanes. The feasible set of model parameters was constrained by data for linear alkanes, ethane to octane, and cyclohexane. Grey shaded regions are experimental bounds reported in Table 7.5 and only serve for comparison. The horizontal black interval is the extrema of the predicted intervals from Eq. 7.6. The vertical black dashed line is the heat of formation evaluated by the nominal PM7 parameter value, $f_p(x_{\text{nom}})$.

One clear challenge in this study was the inability to accurately represent a significant volume of the parameter space with a polynomial surrogate model. This difficulty motivated the development of a tool-set for uncertainty quantification of models with high-dimensional parameter spaces that can have non-convex and, potentially, non-contiguous feasible sets. Incorporation of machine learning techniques with the B2BDC methodology was a viable strategy for efficiently tackling cases of this type. Attempts to learn the geometric structure of the identified feasible sets showed promising results. The possibility of accurate binary classification of feasible or infeasible classes suggests a route to sampling strategies that are more efficient than uniform, brute-force sampling and could rely on a semi-supervised model to search for feasible points.

Including a well-motivated, charge-based model discrepancy function could resolve the inconsistency between the PM7 model and a family of linear alkanes. Model discrepancy was demonstrated as a tool for providing additional feedback in the model validation workflow. However, solely addressing the model-data inconsistency was shown to be inadequate for making predictions. Additional basis functions (and data) are necessary when considering model predictions outside of the training data. A paper was published based on some work presented in this chapter [68].

# Chapter 8

# Conclusions and Future Work

## 8.1 Conclusions

This dissertation demonstrates that tools that utilize both UQ methodology and automated access to validation data are increasingly important for improving our understanding of complex physics-based models. The validation workflow shown was general in the sense it could be applied to any predictive model, using the cloud-based platform of PrIMe and the B2BDC UQ methodology. The presented workflow was effective in systematically evaluating the consistency of potential models, which was used in three cases: an $H_2/O_2$ system, a char oxidation model, and a semi-empirical quantum chemistry model.

The B2BDC analysis of an $H_2/O_2$ system revealed that a set of shock-tube measurements, spanning high-pressure and low-temperature scenarios, were incompatible with the reaction mechanism and experimental data. Here, the $H_2/O_2$ reaction mechanism was known with high certainty; however, the idealized reactor model, which emulates the shock-tube experiments, was judged as potentially inadequate for these scenarios.

During the investigation of a reduced char oxidation model, the validation workflow was applied in an iterative fashion which helped guide the development of a reduced-order model. In this analysis, both the experimental data and char oxidation model had significant uncertainty. The B2BDC analysis was able to uncover the importance of the initial particle diameter, specifically how a fixed initial particle diameter was inconsistent with the experimental data, whereas an initial distribution of particle diameters was consistent. This result suggested that additional work is necessary to characterize the initial particle diameters.

A semi-empirical quantum chemistry model, which is empirically based and uncertain, was incompatible with a set of experimental data known to high precision. The sample-based approach was shown to be effective in approximating the feasible set. Samples were then used to develop a binary classifier that constructed a decision boundary between feasible and infeasible samples in the parameter space. Since the semi-empirical quantum chemistry model was incompatible with a set of linear alkane data, an additive model discrepancy was explored to provide feedback to domain scientists and diagnose potential routes towards consistency.

For QOIs that were not well-represented with a single polynomial surrogate model, a novel strategy for piecewise modeling was presented that combines B2BDC's scalar consistency measure with piecewise modeling. This novel strategy was developed to eliminate domains of the parameter space that are incompatible with the experimental evidence. This strategy was successfully applied in several examples, reducing the total number of function evaluations necessary by refining only domains that contained the feasible set.

These cases served as examples of how the developed workflow and piecewise modeling strategy could provide analysis and feedback in situations where the amount of uncertainty present in the model and/or data varied. Our understanding of the world will always be limited by data and knowledge of the data-generating process. By developing tools that combine available experimental data with uncertainty quantification methods, we can improve our knowledge of complex physical models and assist in the development of predictive models.

## 8.2 Future Work

In Chapter 4, the piecewise modeling strategy developed multiple polynomial surrogate models, with each defined over a disjoint partition of the parameter space. This strategy could lead to numerous evaluations of an expensive computational model depending on the behavior of the model output and the specified error tolerance. Potentially a more flexible surrogate model, e.g., a Gaussian process, could accurately reproduce the model output over a partition. In such cases, the piecewise modeling strategy would then replace the expensive computational model with the Gaussian process model, where the Gaussian process will act as an intermediate surrogate model. The piecewise modeling strategy could also benefit from the exploration of more advanced partitioning schemes, specifically considering multivariate hyperplanes or Voronoi tessellations, which has been successfully applied by Rushdi et al. [159]. Currently, the piecewise modeling strategy partitions a domain following a greedy heuristic-based algorithm. This means for each partition, the optimal decision is made based on a heuristic. Non-greedy approaches, e.g., a look-ahead with $k$ iterations [160], could potentially reduce the total number of partitions required by taking a non-optimal decision at the current iteration. A partition decision is made that is optimal over the next $k$ iterations. Implementing these strategies would require many hypothetical partitions to be made, thus requiring numerous samples of the computational model, which would further necessitate the need for an intermediate surrogate model as discussed above.

Chapter 6 examined a reduced char oxidation model which included 399 QOIs. Many of the QOIs were poorly represented by quadratic surrogate models over the prior domain, which is why the piecewise modeling was employed. Potentially an iterative or wave-based approach, as described in the Bayesian History Matching literature [161, 162], could be used with the B2BDC methodology. In Chapter 7, a semi-empirical quantum chemistry model was investigated using uniform samples drawn from a rotated volume. Future research can investigate the utility of adaptive sampling methods [163, 164] and a larger prior domain. An additive charge-based model discrepancy was developed to account for the disagreement between the quantum chemistry model and a collection of experimental data. Potentially an embedded approach [154], which modifies the input parameters, could be of value as the model parameters are empirically based.

# Appendix A

# PrIMe Data Warehouse

## A.1 Chemical Analysis XML Document

As discussed in Section 2.4, the chemical composition of a solid-fuel is specified by a secondary chemical analysis document. Proximate and ultimate analyses of the solid-fuel are stored in the chemical analysis XML document. Managing the format of this document is the XML schema, shown in Source Code A.1. An example of a chemical analysis document can be found in Source Code A.2.

Source Code A.1: Developed schema for chemical analysis XML documents

```
<xs:schema xmlns="http://purl.org/NET/prime/"
           targetNamespace="http://purl.org/NET/prime/"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">

        <xs:annotation id="ID">
                <xs:documentation xml:lang="en">PrIMe Chemical Analysis XML
                ↪  schema</xs:documentation>
        </xs:annotation>

        <xs:annotation id="copyright">
                <xs:documentation xml:lang="en">primekinetics.org
                ↪  2005-2018</xs:documentation>
        </xs:annotation>

        <xs:annotation id="createdBy">
                <xs:documentation xml:lang="en">Thomas C. Allison, NIST, December
                ↪  2004</xs:documentation>
        </xs:annotation>

        <xs:annotation id="authoredBy">
                <xs:documentation xml:lang="en">Thomas C. Allison,
                ↪  NIST</xs:documentation>
```

```
              <xs:documentation xml:lang="en">Michael Frenklach, University of
              ↪  California at Berkeley</xs:documentation>
              <xs:documentation xml:lang="en">Zoran M. Djurisic, University of
              ↪  California at Berkeley</xs:documentation>
              <xs:documentation xml:lang="en">Devin R. Yeates, University of California
              ↪  at Berkeley</xs:documentation>
  <xs:documentation xml:lang="en">Jim Oreluk, University of California at
  ↪  Berkeley</xs:documentation>
      </xs:annotation>

      <xs:attributeGroup name="rootAttributes">
              <xs:attribute name="primeID" type="primeChemicalAnalysisIDType"
              ↪  use="required"/>
      </xs:attributeGroup>

      <xs:element name="chemicalAnalysis">
              <xs:complexType>
                      <xs:sequence>
                              <xs:element name="copyright"           type="xs:string"
                              ↪  minOccurs="0" maxOccurs="1"/>
                              <xs:element name="content"             type="contentType"
                              ↪  minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="bibliographyLink"    type="bibliographyLinkType"
      ↪  minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="speciesLink"         type="speciesLinkType"
      ↪  minOccurs="1" maxOccurs="unbounded"/>
      <xs:element name="property"            type="propertyType"
      ↪  minOccurs="0" maxOccurs="unbounded"/>
                              <xs:element name="additionalDataItem"
                              ↪  type="additionalDataItemType"  minOccurs="0"
                              ↪  maxOccurs="unbounded"/>
                      </xs:sequence>
                      <xs:attributeGroup ref="rootAttributes"/>
              </xs:complexType>
      </xs:element>

      <xs:simpleType name="primeBiblioIDType">
              <xs:restriction base="xs:string">
                      <xs:pattern value="b\d{8}"/>
              </xs:restriction>
      </xs:simpleType>

      <xs:simpleType name="primeSpeciesIDType">
              <xs:restriction base="xs:string">
                      <xs:pattern value="s\d{8}"/>
              </xs:restriction>
      </xs:simpleType>

<xs:simpleType name="primeChemicalAnalysisIDType">
  <xs:restriction base="xs:string">
```

```
    <xs:pattern value="ca\d{8}"/>
  </xs:restriction>
</xs:simpleType>

      <xs:complexType name="contentType">
              <xs:simpleContent>
                      <xs:extension base="attributableString">
                              <xs:attribute name="copyrighted"  type="xs:boolean"
                              ↪  use="optional"/>
                              <xs:attribute name="bibliography"
                              ↪  type="primeBiblioIDType" use="optional"/>
                      </xs:extension>
              </xs:simpleContent>
      </xs:complexType>

      <xs:complexType name="attributableString">
              <xs:simpleContent>
                      <xs:extension base="xs:string">
                              <xs:attribute name="source" type="xs:string"
                              ↪  use="optional"/>
                      </xs:extension>
              </xs:simpleContent>
      </xs:complexType>

      <xs:complexType name="speciesLinkType">
              <xs:attribute name="preferredKey" type="xs:string"
              ↪  use="required"/>
              <xs:attribute name="primeID"      type="primeSpeciesIDType"
              ↪  use="required"/>
      </xs:complexType>

<xs:complexType name="bibliographyLinkType">
  <xs:attribute name="preferredKey" type="xs:string"          use="optional"/>
  <xs:attribute name="primeID"      type="primeBiblioIDType" use="required"/>
</xs:complexType>

<xs:complexType name="propertyType">
  <xs:sequence>
    <xs:element name="value" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="uncertainty" type="uncertaintyType" minOccurs="0" maxOccurs="2"/>
    <xs:element name="component" type="componentType" minOccurs="0"
    ↪  maxOccurs="unbounded"/>
    <xs:element name="speciesLink" type="speciesLinkType" minOccurs="0"
    ↪  maxOccurs="unbounded"/>
    <xs:element name="derivedProperty" type="xs:string" minOccurs="0"
    ↪  maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="name" type="xs:string" use="required"/>
```

```
    <xs:attribute name="units" type="xs:string" use="optional"/>
    <xs:attribute name="description" type="xs:string" use="optional"/>
    <xs:attribute name="derivedPropertyExists" type="xs:string" use="optional"/>
    <xs:attribute name="kind" type="xs:string" use="optional"/>
  </xs:complexType>

  <xs:complexType name="uncertaintyType">
    <xs:simpleContent>
      <xs:extension base="xs:double">
        <xs:attribute name="bound"          type="xs:string" use="required"/>
        <xs:attribute name="kind"           type="xs:string" use="required"/>
        <xs:attribute name="transformation" type="xs:string" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="componentType">
    <xs:sequence>
      <xs:element name="speciesLink" type="speciesLinkType" minOccurs="1" maxOccurs="1"/>
      <xs:element name="amount" type="amountType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="uncertainty" type="uncertaintyType" minOccurs="0" maxOccurs="2"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="amountType">
    <xs:simpleContent>
      <xs:extension base="xs:double">
        <xs:attribute name="units" type="xs:string" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

        <xs:complexType name="additionalDataItemType">
                <xs:simpleContent>
                        <xs:extension base="xs:string">
                                <xs:attribute name="itemType"    type="itemTypeTypes"
                                ↪  use="required"/>
                                <xs:attribute name="description" type="xs:string"
                                ↪  use="optional"/>
                                <xs:attribute name="MIME"        type="xs:string"
                                ↪  use="optional"/>
                        </xs:extension>
                </xs:simpleContent>
        </xs:complexType>

        <xs:simpleType name="itemTypeTypes">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="URI"/>
                        <xs:enumeration value="file"/>
                        <xs:enumeration value="text"/>
                </xs:restriction>
        </xs:simpleType>
</xs:schema>
```

Source Code A.2: Example of a chemical analysis document for Pittsburgh No. 8 coal

```
<chemicalAnalysis xmlns="http://purl.org/NET/prime/"
↪  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" primeID="ca00000001"
↪  xsi:schemaLocation="http://purl.org/NET/prime/
↪  http://warehouse.primekinetics.org/schema/chemicalAnalysis.xsd">
  <copyright>primekinetics.org 2005-2015</copyright>
  <bibliographyLink preferredKey="Knill et al. 1988" primeID="b00019066" />
  <speciesLink preferredKey="Pittsburgh No. 8" primeID="s00010922" />
  <property description="Ultimate Analysis - daf C" name="fraction" label="C"
  ↪  units="unitless" kind="daf">
    <value>0.81426</value>
  </property>
  <property description="Ultimate Analysis - daf H" name="fraction" label="H"
  ↪  units="unitless" kind="daf">
    <value>0.05</value>
  </property>
  <property description="Ultimate Analysis - daf O" name="fraction" label="O"
  ↪  units="unitless" kind="daf">
    <value>0.10542</value>
  </property>
  <property description="Ultimate Analysis - daf N" name="fraction" label="N"
  ↪  units="unitless" kind="daf">
    <value>0.017068</value>
  </property>
  <property description="Ultimate Analysis - daf S" name="fraction" label="S"
  ↪  units="unitless" kind="daf">
    <value>0.0131</value>
  </property>
  <property description="Ultimate Analysis - daf Cl" name="fraction" label="Cl"
  ↪  units="unitless" kind="daf">
    <value>0.013052</value>
  </property>
  <property description="Proxmiate Analysis - dry Ash" name="fraction" label="ash"
  ↪  units="unitless" kind="dry">
    <value>0.081</value>
  </property>
  <property description="Proxmiate Analysis - daf Volatile Matter" name="fraction"
  ↪  label="VM_daf" units="unitless" kind="daf">
    <value>0.386</value>
  </property>
  <property description="High Volatile-Matter - daf" name="fraction" label="HVM_daf"
  ↪  units="unitless" kind="daf">
    <value>0.62</value>
  </property>
  <property description="Lower Calorific Value - dry Measured" name="LCV" label="LCV_dry
  ↪  Measured" kind="dry" units="MJ/kg">
    <value>29</value>
  </property>
  <property description="mass mean diamter" name="diameter" label="mass mean diamter"
  ↪  units="m">
    <value>80</value>
  </property>
</chemicalAnalysis>
```

## A.2 PrIMe 3.0 Warehouse API

An API was written for remotely interacting with data archived in the PrIMe Data Warehouse. The Warehouse API enables search queries, downloading particular documents, and extracting QOI targets from the PrIMe Data Warehouse. The Warehouse API was written in Python 3.6 can be found below and on https://github.com/oreluk/Warehouse-API.

In the Source Code A.3, the class `Warehouse` outlines the main methods for creating search queries. Queries in the PrIMe Data Warehouse are conducted via Elasticsearch that can be executed by the `Warehouse.search` method. Search queries are generated into a valid Elasticsearch message by `generateMessage.py`, shown in Source Code A.4.

Source Code A.3: PrIMe30.py

```python
import hashlib
from urllib.request import Request, urlopen
import urllib.parse
import json
import warnings

from generateMessage import * # Builds Elasticsearch messages


class Warehouse:
    def __init__(obj, user, password):
        #obj.Url = 'http://52.88.176.152:8080/'
        obj.Url = 'http://54.214.127.78:8080/'
        obj.user = user
        password = hashlib.md5(password.encode('utf-8')).hexdigest()

        urlreq = {'grant_type': 'password',
                  'username': obj.user,
                  'password': password}

        data = urllib.parse.urlencode(urlreq).encode('utf-8')

        headers = {'Content-Type': 'application/x-www-form-urlencoded'}
        request = Request(obj.Url + '/token', data=data, headers=headers)

        try:
            response_body = urlopen(request).read()
        except:
            print('Authentication failed.')
            raise

        obj.response_body = json.loads(bytes.decode(response_body))
        obj.token = 'Bearer ' + obj.response_body['access_token']
        print('User Authenticated')

    def search(obj, category, field, query1, query2=''):
        '''
        Generate ElasticSearch Queries for PrIMe 3.0
        SEARCH(OBJ, CATEGORY, FIELD, QUERY1, QUERY2)

        CATEGORY is a character string specifying the collection
        of the warehouse which will searched within. Default 'all'.
        Other valid collections are listed below:

        all
        bibliography
        dataAttribute
        dataset
        element
        experiment
        instrumentalModel
        model
        optimizationVariable
        optimizationVariableBounds
        reaction
        reactionRate
        species
        surrogateModel
        thermodynamicData
        transportData

        FIELD is a string specifying the field of the
        category which will be queried. Each category has its own
        unique fields but some fields, i.e. primeID , are shared
        amongst all categories.

        QUERY is a string containing a search query string.

        '''

        validCategories = ['all', 'bibliography', 'dataAttribute',
                           'dataset', 'element', 'experiment', 'instrumentalModel',
                           'model', 'optimizationVariable', 'optimizationVariableBounds',
                            'reaction', 'reactionRate', 'species', 'surrogateModel',
                            'thermodynamicData', 'transportData']

        s = [x.lower() for x in validCategories]
        if not any(x in category.lower() for x in s):
            warnings.warn(
                'Input category is not valid. Using the default catagory "all".')
            category = 'all'

        if (field.lower() != 'rates') and (category.lower() !=
        ↪  'optimizationVariable'.lower()):
            msg = generateMessage(category, field, query1)
        else:
            # optimizationVariable & rates take two inputs reactionID and rateID
            ↪  (query1,query2) as inputs.
            msg = generateMessage(category, field, query1, query2)

        searchUrl = obj.Url + 'api/v2/warehouse/search/raw/' + category
        headers = {'Content-Type': 'text/plain', 'Authorization': obj.token}
        jsonString = json.dumps(json.loads(msg))
        data = jsonString.encode('utf-8')

        request = Request(searchUrl, data=data, headers=headers)
        response = urlopen(request).read()
        return(json.loads(bytes.decode(response)))

    def getXml(obj, pathTo):
        # Accepts PATHTO as a string of the location of XML document in the PrIMe
        ↪  Warehouse
        #
        if isinstance(pathTo, str):
            # Download XML from specified location
            searchUrl = obj.Url + 'api/v2/warehouse/content?path=' + pathTo
            headers = {'Authorization': obj.token}
            request = Request(searchUrl, headers=headers)
            response = urlopen(request).read()
            return(bytes.decode(response))
        else:
            raise TypeError

    def getFile(obj, pathTo):
        # Saves file from Warehouse to current working directory(cwd)
        # PATHTO is a string of the location of file on the PrIMe Warehouse
        #
        # Example: wh.getFile('depository/species/catalog/s00009193.xml')
        #
        #
        import shutil

        if isinstance(pathTo, str):
            # Count number of entries in collection
            fileName = pathTo[-13:]
            searchUrl = obj.Url + 'api/v2/warehouse/content?path=' + pathTo
            headers = {'Authorization': obj.token}
            request = Request(searchUrl, headers=headers)
            response = urlopen(request).read()
            f = open(fileName, 'wb')
            f.write(response)
            print(fileName + ' was saved in the current working directory.')
        else:
            raise TypeError

    def getCount(obj, collection):
        # Returns the number of entries of a specified collection
        #
        # COLLECTION is a string of the location which will be counted.
        # Examples:
        #
        #     depository/species/catalog
        #     depository/bibliography/catalog
        #     depository/experiments/catalog
        #

        if isinstance(collection, str):
            # Count number of entries in collection
            searchUrl = obj.Url + 'api/v2/warehouse/search/all/count?path=' + collection
            headers = {'Authorization': obj.token}
            request = Request(searchUrl, headers=headers)
            response = urlopen(request).read()
            return(int(bytes.decode(response)))
        else:
            raise TypeError

    def exist(obj, pathTo):
        # Returns logical True or False depending if pathTo exists
        # pathTo is a string of the location of file on the PrIMe Warehouse
        #
        # Example:
        #     wh.exist('depository/bibliography/catalog/b00000033.xml')   File exists
        #     wh.exist('depository/bibliography/catalog/f99000000.xml')   File does not
        ↪  exist
        #
        if isinstance(pathTo, str):
            searchUrl = obj.Url + 'api/v2/warehouse/xml/exist?path=' + pathTo
            headers = {'Authorization': obj.token}
            request = Request(searchUrl, headers=headers)
            try:
                response = urlopen(request).read()
                print('File exists.')
                return(True)
            except:
                print('File does not exist.')
                return(False)
        else:
            raise TypeError

    def getList(obj, collection):
        # Returns a list of all files by the specified path
        # COLLECTION is a string specifiying the collection of the warehouse
        #
        # Example:
        #     wh.getList('depository/bibliography/catalog')
        #     wh.getList('depository/reaction/catalog')
        #
        if isinstance(collection, str):
            searchUrl = obj.Url + 'api/v2/warehouse/search/all?path=' + collection
            headers = {'Authorization': obj.token}
            request = Request(searchUrl, headers=headers)
            response = urlopen(request).read()
            return(json.loads(bytes.decode(response)))
        else:
            raise TypeError

    def getProperty(obj, path, prop):
        # Returns a property value from a document.
        # PATH is the location of a document on the PrIMe Warehouse
        # PROP is the field name of the property whose value will be returned
        #
        # Example:
        #     wh.findProp('depository/bibliography/catalog/b00000290.xml', 'year')
        #     wh.findProp('depository/experiments/catalog/x00001327.xml', 'kind')
        #     wh.findProp('depository/reactions/data/r00013869/rk00000036.xml', 'value')
        #
        if all(isinstance(item, str) for item in [path, prop]):
            searchUrl = obj.Url + 'api/v2/warehouse/fields?path=' + path + '&field=' +
            ↪  prop
            headers = {'Authorization': obj.token}
            request = Request(searchUrl, headers=headers)
            response = urlopen(request).read()
            return(json.loads(bytes.decode(response)))
        else:
            raise TypeError

    def getPropertyNames(obj, collection):
        # Returns possible property names for a given collection
        # COLLECTION is a string specifying the collection of the PrIMe Warehouse
        #
        # Example:
        #     wh.getPropertyNames('experiment')
        #     wh.getPropertyNames('bibliography')
        #     wh.getPropertyNames('element')
        #
        if isinstance(collection, str):
            searchUrl = obj.Url + 'api/v2/warehouse/fields?path=' + collection
            headers = {'Authorization': obj.token}
            request = Request(searchUrl, headers=headers)
            response = urlopen(request).read()
            return(json.loads(bytes.decode(response)))
        else:
            raise TypeError

    def getJSON(obj, category, pathTo):
        # GETJSON will match the pathTo with the ID of a JSON document and return the
        ↪  document
        #
        if isinstance(pathTo, str):
            startString = '{ "ids": { "values": "'
            endString = '" } }'
            msg = startString + pathTo + endString

            searchUrl = obj.Url + 'api/v2/warehouse/search/raw/details/' + category
            headers = {'Content-Type': 'text/plain',
                       'Authorization': obj.token}
            jsonString = json.dumps(json.loads(msg))
            data = jsonString.encode('utf-8')
            request = Request(searchUrl, data=data, headers=headers)
            response = urlopen(request).read()
            return(json.loads(bytes.decode(response)))
        else:
            raise TypeError

    def getBoundsFromOptVar(obj, vbPath):
        # Takes a optimization variable bounds path and returns upper and lower bounds
        #
        if isinstance(vbPath, str):
            f = obj.getJSON('optimizationVariableBounds', vbPath)
            vbDoc = f[0]

            lower = float(vbDoc['optimizationVariableLink']
                          ['bounds']['lower']['#text'])
            upper = float(vbDoc['optimizationVariableLink']
                          ['bounds']['upper']['#text'])
            return((lower, upper))
        else:
            raise TypeError

    def getModelBounds(obj, pathToModel):
        # returns reactionNames and bounds from the PrIMe database
        #
        import numpy as np
```

```python
        f = obj.getJSON('model', pathToModel)
        modelDoc = f[0]

        reactionKey = []
        rL = modelDoc['reactionSet']['reactionLink']
        reactionBounds = np.zeros((len(rL), 2))

        for i in range(0, len(rL)):
            reactionKey.append(rL[i]['reactionRateLink']['@preferredKey'])
            rID = rL[i]['@primeID']
            rkID = rL[i]['reactionRateLink']['@primeID']
            vResults = obj.search('optimizationVariable', 'rates', rID, rkID)

            if len(vResults) == 1:
                # single matching optimizationVariable Found
                variableID = vResults[0][-13:-4]
                vbResults = obj.search(
                    'optimizationVariableBounds', 'varlinkid', variableID)

                if len(vbResults) == 1:
                    # single results
                    reactionBounds[i, :] = obj.getBoundsFromOptVar(
                        vbResults[0])
                elif len(vbResults) == 2 and vbResults[0][-13:-4] == "vb00000000":
                    # two results but one is the vb0 file.
                    reactionBounds[i, :] = obj.getBoundsFromOptVar(
                        vbResults[1])
                else:
                    warnings.warn(
                        'Reaction :' + reactionKey[i] + ' has multiple bounds associated.
                        ↪  Setting bounds to (0,0).')
                    reactionBounds[i, :] = (0, 0)

            elif len(vResults) == 0:
                # variable not found
                reactionBounds[i, :] = (0.5, 2)
            else:
                raise ValueError(
                    'Results contain multiple matches. Unable to proceed.')

        return(reactionKey, reactionBounds)

    def getTarget(obj, pathToTarget):
        # Returns a dictionary of property information from the specified target
        #

        def parseCommonProperties(commonPropNode):
            # internal function for parsing common properties of experimental document
            #
            propNames = []
            propValues = []
            propUnits = []

            speciesName = []
            speciesID = []
            speciesUnits = []
            molFrac = []

            if isinstance(commonPropNode['property'], list):
                for i in range(0, len(commonPropNode['property'])):
                    pName = commonPropNode['property'][i]['@name']
                    if pName == 'initial composition':
                        # fill-in initial composition
                        compNodes = commonPropNode['property'][i]['component']
                        for j in range(0, len(compNodes)):
                            speciesName.append(
                                compNodes[j]['speciesLink']['@preferredKey'])
                            speciesID.append(
                                compNodes[j]['speciesLink']['@primeID'])
                            try:
                                speciesUnits.append(
                                    compNodes[j]['amount']['@units'])
                            except:
                                speciesUnits.append([])

                            try:
                                molFrac.append(
                                    float(compNodes[j]['amount']['#text']))
                            except:
                                molFrac.append([])
                    else:
                        propNames.append(pName)
                        propValues.append(
                            float(commonPropNode['property'][i]['value']['#text']))
                        propUnits.append(
                            commonPropNode['property'][i]['@units'])
            elif isinstance(commonPropNode['property'], dict):
                if commonPropNode['property']['@name'] == 'initial composition':
                    compNodes = expDoc['commonProperties']['property']['component']
                    for i in range(0, len(compNodes)):
                        speciesUnits.append(compNodes[i]['amount']['@units'])
                        molFrac.append(float(compNodes[i]['amount']['#text']))
                        speciesName.append(
                            compNodes[i]['speciesLink']['@preferredKey'])
                        speciesID.append(
                            compNodes[i]['speciesLink']['@primeID'])

            return(propNames, propValues, propUnits, speciesName, speciesID,
            ↪  speciesUnits, molFrac)

```

```
        # Get Target function
        if not isinstance(pathToTarget, str):
            raise TypeError

        f = obj.getJSON('dataAttribute', pathToTarget)
        datastore = f[0]
        #
        # Take JSON and Organize Data into Python Dictionary (qoi)
        qoi = {}

        obsNodes = datastore['dataAttributeValue']['observable']
        if isinstance(obsNodes, list):
            targetType =
            ↪  datastore['dataAttributeValue']['observable'][0]['property']['@name']
        elif isinstance(obsNodes, dict):
            targetType =
            ↪  datastore['dataAttributeValue']['observable']['property']['@name']

        if targetType == 'laminar flame speed' or targetType == 'flame speed':
            qoiType = 'flame speed'
        elif targetType == 'time' or targetType == 'ignition delay':
            qoiType = 'time'

        # Indicator Node
        indicatorNode = datastore['dataAttributeValue']['indicator']
        propNames = []
        propValues = []
        propUnits = []

        if isinstance(indicatorNode, list):
            for i in range(0, len(indicatorNode)):
                propNames.append(indicatorNode[i]['property']['@name'])
                propValues.append(
                    float(indicatorNode[i]['property']['value']['#text']))
                propUnits.append(indicatorNode[i]['property']['@units'])

        elif isinstance(indicatorNode, dict):
            propNames.append(indicatorNode['property']['@name'])
            propValues.append(
                float(indicatorNode['property']['value']['#text']))
            propUnits.append(indicatorNode['property']['@units'])

        # Observable Node Uncertainty
        obsNode = datastore['dataAttributeValue']['observable']
        if isinstance(obsNode, list):
            obsNode = datastore['dataAttributeValue']['observable'][0]
        elif isinstance(obsNode, dict):
            obsNode = datastore['dataAttributeValue']['observable']

        lowerBound = float(obsNode['bounds']['lower']['#text'])
        upperBound = float(obsNode['bounds']['upper']['#text'])
        uncKind = obsNode['bounds']['@kind']
        uncBounds = [lowerBound, upperBound]

        # Node information
        obsValue = float(obsNode['property']['value']['#text'])
        obsUnits = obsNode['property']['@units']
        if qoiType == 'time':
            obsInd = obsNode['@derivedBy']
            obsIndSpecies = obsNode['@speciesName']
            obsIndSpeciesID = obsNode['@speciesID']

        # propertyLink
        dataGroupID = datastore['propertyLink'][0]['@dataGroupID']

        # Load expDoc
        expID = datastore['propertyLink'][0]['@experimentPrimeID']
        pathToExp = 'depository/experiments/catalog/' + expID + '.xml'
        expF = obj.getJSON('experiment', pathToExp)
        expDoc = expF[0]

        # Experiment Information
        expKind = expDoc['apparatus']['kind']['#text']
        expID = expDoc['@primeID']

        # Parse Common Properties
        commonPropNode = expDoc['commonProperties']
        [pNames, pValues, pUnits, speciesName, speciesID, speciesUnits,
            molFrac] = parseCommonProperties(commonPropNode)

        # Copy Common Properties Over Indicator Properties if name matches
        for i in range(0, len(pNames)):
            for j in range(0, len(propNames)):
                if pNames[i] == propNames[j]:
                    propValues[j] = pValues[i]

        # Check if molFrac from Parse Common Properties is empty
        if not molFrac[0]:
            # if empty go through datagroup of expDoc
            # match obsValue with associated property name and
            # get all property and information from that dataGroup/dataPoint
            dpValues, dpUnits, dpNames = ([] for n in range(3))
            sNames, sUnits, sID, mF = ([] for n in range(4))

            # if multiple datagroups exist
            if isinstance(expDoc['dataGroup'], list):
                for k in range(0, len(expDoc['dataGroup'])):
                    if expDoc['dataGroup'][k]['@id'] == dataGroupID:
                        eDG = expDoc['dataGroup'][k]
            elif isinstance(expDoc['dataGroup'], dict):
                eDG = expDoc['dataGroup']
            else:
                raise TypeError

            for i in range(0, len(eDG['property'])):
                if eDG['property'][i]['@name'] == qoiType:
                    matchingPropID = eDG['property'][i]['@id']

                    for j in range(0, len(eDG['dataPoint'])):
                        if float(eDG['dataPoint'][j][matchingPropID]['#text']) ==
                        ↪  obsValue:
                            matchingDP = j

            for i in range(0, len(eDG['property'])):
                # get all information from dataPoint node
                cList = ['composition', 'concentration']
                s = [x.lower() for x in cList]
                if any(x in eDG['property'][i]['@name'] for x in s):

                    if len(speciesID) != 0:
                        # match speciesID with sID
                        for j in range(0, len(speciesID)):
                            speLID = eDG['property'][i]['speciesLink']['@primeID']
                            if speLID == speciesID[j]:
                                matchingID = eDG['property'][i]['@id']
                                speciesUnits[j] = eDG['property'][i]['@units']
                                molFrac[j] = float(
                                    eDG['dataPoint'][matchingDP][matchingID]['#text'])

                    else:
                        # no common property of composition
                        matchingID = eDG['property'][i]['@id']
                        sID.append(eDG['property'][i]
                                   ['speciesLink']['@primeID'])
                        sUnits.append(eDG['property'][i]['@units'])
                        sNames.append(eDG['property'][i]['@name'])
                        mF.append(
                            float(eDG['dataPoint'][matchingDP][matchingID]['#text']))

                else:
                    dpNames.append(eDG['property'][i]['@name'])
                    dpUnits.append(eDG['property'][i]['@units'])
                    pID = eDG['property'][i]['@id']
                    try:
                        dpValues.append(
                            float(eDG['dataPoint'][matchingDP][pID]['#text']))
                    except:
                        # if individual uncetainty is applied to a node there is a comma
                        ↪  delimination
                        # Example: 0.07,0.002
                        #
                        s = eDG['dataPoint'][matchingDP][pID]['#text'].split(
                            ',')
                        dpValues.append([float(s[0]), float(s[1])])

            # extend list with common property values
            propNames.extend(pNames)
            propValues.extend(pValues)
            propUnits.extend(pUnits)

            # If dp property name matches one of the common properties names.
            # Keep Common Property,  Delete DataPoint information.
            #
            # Common example of this is the ambient temperature of the enviroment in a flame experiment.
            # Common Properties has the temperature of enviroment while dataPoint may contain flame temperature
            # which is unused by simulation
            #
            # Remove if statement if you want all properties returned, including duplicate property names
            #
            if len(dpNames) != 0:
                for i in range(0, len(dpNames)):
                    for j in range(0, len(propNames)):
                        if dpNames[i] == propNames[j]:
                            dpNames.remove(dpNames[i])
                            dpUnits.remove(dpUnits[i])
                            dpValues.remove(dpValues[i])

            try:
                propNames.extend(dpNames)
                propValues.extend(dpValues)
                propUnits.extend(dpUnits)
            except:
                pass

            # add species information to speciesLists
            if len(sNames) != 0:
                speciesName.append(sNames)
                speciesID.append(sID)
                speciesUnits.append(sUnits)
                molFrac.append(mF)

        # Create new dictionary of QOI information
        qoi['indicator_name'] = propNames
        qoi['indicator_value'] = propValues
        qoi['indicator_units'] = propUnits
        qoi['preferredKey'] = datastore['preferredKey']['#text']
        qoi['observable_bounds'] = uncBounds
        qoi['observable_boundkind'] = uncKind
        qoi['observable_value'] = obsValue
        qoi['observable_units'] = obsUnits
        if qoiType == 'time':
            qoi['derivedBy'] = obsInd
            qoi['derivedBy_speciesName'] = obsIndSpecies
            qoi['derivedBy_speciesID'] = obsIndSpeciesID

        qoi['experiment_ID'] = expID
        qoi['experiment_type'] = expKind
        qoi['species_key'] = speciesName
        qoi['species_molFraction'] = molFrac
        qoi['species_primeID'] = speciesID
        qoi['species_units'] = speciesUnits

        return(qoi)
```

Source Code A.4: Generate Elasticsearch query message

```
def generateMessage(category, field, term, term2=''):
    # Generate Elasticsearch Message
    #
    # GENERATEMESSAGE(CATEGORY, FIELD, TERM) will take three character
    # strings and return an elastic search query.
    # CATEGORY is a character string specifying the type of
    # catagory will searched. Default 'all'. Other available
    # options are listed below:
    #
    # all
    # bibliography
    # dataAttribute
    # dataset
    # element
    # experiment
    # instrumentalModel
    # model
    # optimizationVariable
    # reactions
    # reactionRate
    # species
    # surrogateModel
    # thermodynamicData
    # transportData
    #
    # FIELD is a character string specifying the field to be searched for. This
    # often is a node name or attribute name in the document.
    #
    # TERM is a character string of the term which a query message will be
    # generated for.
```

```
#

def generateDefaultMessage(field, term):
    #
    # INTERNAL FUNCTION which will generated a default message from
    # MESSAGE = GENERATEDEFAULTMESSAGE(FIELD, TERM)
    #

    msg = ('[{ "query_string": { "fields": ["*' +
           field + '*"], "query": "' +
           term + '" }}]')
    return(msg)

# Conditional Statements
if category.lower() == 'species':
    if field.lower == 'preferredkey':
        msg = '[{ "match": "preferredKey.#text": "' + term + '"} }]'

    elif field.lower() == 'formula':
        msg = ('[ { "nested": { "path": "chemicalIdentifier.name",' +
               ' "query": { "bool": { "must": [ { "match": {' +
               ' "chemicalIdentifier.name.@type": "formula" } },' +
               ' { "match_phrase": { "chemicalIdentifier.name.#text": "' + term +
               '" } } ] } } } } ]')

    elif field.lower() == 'brutoformula':
        msg = ('[ { "nested": { "path": "chemicalIdentifier.name",' +
               ' "query": { "bool": { "must": [ { "match": {' +
               ' "chemicalIdentifier.name.@type": "formula" } },' +
               ' { "match_phrase": { "chemicalIdentifier.name.#text": "' + term +
               '" } } ] } } } } ]')

    elif field.lower() == 'caseregistrynumber':
        msg = (' [ { "nested": { "path": "chemicalIdentifier.name",' +
               ' "query": { "bool": { "must": [ { "match": {' +
               ' "chemicalIdentifier.name.@type": "CASRegistryNumber"' +
               ' } }, { "match_phrase": { "chemicalIdentifier.name.#text": "' + term
               ↪  +
               '"} } ] } } } } ]')

    elif field.lower() == 'inchi':
        msg = ('[ { nested: { path: "chemicalIdentifier.name", query: ' +
               ' { bool: { must: [ { match: { "chemicalIdentifier.name.@type":' +
               ' "InChI" } }, { match_phrase: { "chemicalIdentifier.name.#text": "' +
               ↪  term +
               '"} } ] } } } } ]')

    elif field.lower() == 'composition':
        # TODO LATER
        msg = generateDefaultMessage(field, term)

    elif field.lower() == 'name':
        msg = ('[ { "nested": { "path": "chemicalIdentifier.name", "query": ' +
               ' { "bool": { "must": [  { "match_phrase": {
               ↪  "chemicalIdentifier.name.#text": "' + term +
               '" } } ] } } } } ]')

    else:
        msg = generateDefaultMessage(field, term)

elif category.lower() == 'experiment':
    if field.lower() == 'additionaldataitem':
        msg = ('[ { "nested": { "path": "additionalDataItem", "query": {' +
               ' "bool": { "must": [ { "query_string": { "fields": ' +
               ' ["*additionalDataItem*"], "query": "' + term + '"} }' +
               ' ] } } } } ]')

    else:
        msg = generateDefaultMessage(field, term)

elif category.lower() == 'optimizationvariable':
    if field.lower() == 'rates':
        msg = ('[ { "match_phrase": {' +
               ' "variable.actualVariable.propertyLink.@reactionPrimeID":  "' + term
               ↪  +
               '" } }, { "match_phrase": {' +
               '"variable.actualVariable.propertyLink.@reactionRatePrimeID": "' +
               ↪  term2 +
               '" } } ]')
    else:
        msg = generateDefaultMessage(field, term)

elif category.lower() == 'optimizationvariablebounds':
    if field.lower() == 'varlinkid':
        msg = ('[ { "match": {' +
               ' "optimizationVariableLink.@primeID":  "' + term +
               '" } } ]')

    else:
        msg = generateDefaultMessage(field, term)

else:
    msg = generateDefaultMessage(field, term)

startString = '{ "bool": { "must": '
endString = '"must_not": { "match": { "PathTo": "*/_attic/*" } } } }'
msg = startString + msg + ',' + endString
return(msg)
```

## A.3 Comparison between databases

As discussed in Chapter 2.6, many new data formats had been developed in an effort to create wider adoption. The ReSpecTh database was developed by taking the PrIMe Data Models and remove all inter-linking between documents, in an attempt to condense all information into a self-contained document. This type of database is problematic, which will be highlighted in the following example.

One example of the interlinking between documents in the PrIMe Data Warehouse is with the bibliography references. A single bibliography XML document is created for each reference containing XML elements, e.g., title, year, author, DOI, journal. All experimental data, chemical rates, or reaction models which were derived from this bibliography record will link to the bibliography XML document, via its unique identifier. However, in the ReSpecTh database, all bibliography information is embedded into a single tag element. This tag contains the attributes, title and doi, and a long (unstructured) string containing all other information. It is important to note is there is no explicit labeling of the authors, journal, or year in this string.

Storing data in an unstructured string is inefficient for many reasons. First, all records which have data coming from the same bibliography reference will now have duplicate information for each bibliography record. Second, the storage of data in an unstructured string provides no benefit for search. The necessity for a common, structured, data format was to offer the ability to search and parse experimental data, which is destroyed if the data is stored in an unstructured string. While condensing information into a single document, the ReSpecTh format removed all capabilities to search for data by bibliography other than DOI, which was already present in the PrIMe Data Models. If a user wanted to find experiments by author, publication year, etc., searching through the ReSpecTh data format, would require parsing an unstructured data string, which is highly susceptible to errors. The ReSpecTh data format attempted to simplify the documentation of an experiment; however, introduced repeated information and an increased difficulty to find data.

# Appendix B

# Assessing surrogate models

## B.1 Invalidating the underlying model

Central to the B2BDC methodology is the notion of a feasible set. The scalar consistency measure (Eq. 3.3) can provide provable statements if the feasible set is empty, i.e., models and data are inconsistent over a particular domain [56, 60]. It is important to note that the B2BDC methodology in Chapter 3 provides provable statements of the inconsistency of the polynomial surrogate model $M_e(x)$, not the underlying model. To prove that the underlying model is inconsistent with experimental evidence, the following assumption is made.

Assume that the maximum absolute error between the underlying model, $f(x, x_{s,e})$, and its surrogate model, $M_e(x)$ for the $e$-th QOI is known. For simplicity, we assume that the model outputs and experimental bounds are positive real numbers, but in general, this condition is not necessary. The maximum absolute error is defined as:

$$\epsilon_{e,\max} = \max_{x \in \mathcal{H}} \left( |f(x, x_{s,e}) - M_e(x)| \right), \tag{B.1}$$

and the maximum absolute relative error can be defined as:

$$\epsilon_{e,\text{rel}} = \max_{x \in \mathcal{H}} \left( \left| \frac{f(x, x_{s,e}) - M_e(x)}{f(x, x_{s,e})} \right| \right). \tag{B.2}$$

With $\epsilon_{e,\max}$ known, the underlying model response can be bounded by:

$$M_e(x) - \epsilon_{e,\max} \leq f(x, x_{s,e}) \leq M_e(x) + \epsilon_{e,\max}. \tag{B.3}$$

The scalar-valued response of $f(x, x_{s,e})$ is contained within the interval $[M_e(x) - \epsilon_{e,\max}, M_e(x) + \epsilon_{e,\max}]$ which holds for $x \in \mathcal{H}$.

During any UQ analysis, whether it be model validation or QOI prediction, it is crucial to account and propagate forward all uncertainties in the analysis. This includes the uncertainty from representing the underlying model with a surrogate model. Ideally, the surrogate model fitting error is significantly less than the experimental uncertainty. This would indicate that including the fitting error is not expected to add a significant amount of uncertainty to the analysis. Nevertheless, to propagate the surrogate model fitting error forward, the error is incorporated into the respective experimental bounds,

$$[L_e, U_e] = [\widetilde{L_e} - \epsilon_{e,\max}, \widetilde{U_e} + \epsilon_{e,\max}], \tag{B.4}$$

where $[\widetilde{L_e}, \widetilde{U_e}]$ are the reported experimental bounds.

If the calculation of the scalar consistency measure (Eq 3.3) results in $\overline{\gamma} < 0$, it certifies the feasible set, $\mathcal{F} = \varnothing$. By incorporating $\epsilon_{e,\max}$ into the experimental bounds, the feasible set $\mathcal{F}$ can also be shown to be empty for the underlying model with the reported experimental bounds. Substituting Eq. B.4 into the definition of the feasible set:

$$\begin{aligned}\mathcal{F} &= \{x \in \mathcal{H} : L_e \le M_e(x) \le U_e, \ \text{ for } e = 1, 2, \ldots, N\}\\ &= \{x \in \mathcal{H} : \widetilde{L_e} - \epsilon_{e,\max} \le M_e(x) \le \widetilde{U_e} + \epsilon_{e,\max}, \ \text{ for } e = 1, 2, \ldots, N\}\\ &= \{x \in \mathcal{H} : \widetilde{L_e} \le M_e(x) + \epsilon_{e,\max},\\ &\qquad\qquad M_e(x) - \epsilon_{e,\max} \le \widetilde{U_e}, \ \text{ for } e = 1, 2, \ldots, N\}.\end{aligned}$$

From Eq. B.3,

$$\begin{aligned} f(x, x_{s,e}) &\le M_e(x) + \epsilon_{e,\max}\\ M_e(x) - \epsilon_{e,\max} &\le f(x, x_{s,e})\end{aligned}$$

with $f(x, x_{s,e})$ contained within the interval of $M_e(x) \pm \epsilon_{e,\max}$, the set defined by the underlying model and the reported experimental bounds,

$$\{x \in \mathcal{H} : \widetilde{L_e} \le f(x, x_{s,e}) \le \widetilde{U_e}, \ \text{ for } e = 1, 2, \ldots, N\} \subset \mathcal{F}.$$

The feasible set for the underlying model with the reported bounds is a subset of $\mathcal{F}$, the set defined by surrogate models with the maximum absolute error incorporated into the experimental bounds. This means any domain found to be inconsistent for the surrogate model is also inconsistent for the underlying model, assuming that the maximum absolute error between the two models was known.

## B.2 Estimating the surrogate model fitting error

Unfortunately, finding $\epsilon_{e,\max}$ would often necessitate solving an optimization problem, maximizing the absolute difference between $f_e(x)$ and $M_e(x)$. This procedure can be costly as each evaluation of $f_e(x)$ can be computationally expensive, making this calculation intractable. However, one can estimate the maximum absolute error by using a sample from the underlying model. Two common approaches for estimating a model's error with samples are by a hold-out set or through cross-validation. A hold-out set divides the sample into two sets, one for training the other for testing. Samples in the training set are used for fitting (also referred to as training) the model, and the samples in the test set are for assessing the accuracy of the model.

In $k$-fold cross-validation[165, 166], samples used in training are reused for testing. The sample data is represented as the following set, $\{(x_i, y_{e,i})\}_{i=1,\ldots,P}$, where $y_{e,i} = f_e(x_i)$ is the evaluation of the underlying model for the $e$-th QOI at sample $x_i$. The sample data is then divided into roughly $K$ equal parts, where each part is termed a *fold*, i.e., $k = 1, 2, \ldots, K$. To evaluate the fitting error of the $k$-th fold, we set aside the sample data associated with the $k$-th fold and construct a surrogate model, $Q^{-k}(x)$, from the remaining $K-1$ folds. The fitting error of the $k$-th fold can then evaluated by $\eta_{e,i} = \left|y_{e,i} - Q^{-k}(x_i)\right|_{i \in k\text{-th fold}}$.

After repeating for all $k$-folds, a surrogate model is built from the entire set of sample data $\{(x_i, y_{e,i})\}_{i=1,\ldots,P}$ and an estimate of the fitting error is made by,

$$\widehat{\epsilon}_e = \frac{1}{P}\sum_{i=1}^{P} \eta_{e,i} \tag{B.5}$$

In this dissertation, $k$-folds cross-validation will be used to estimate the surrogate model fitting error. Instead of using the mean absolute error shown in Eq B.5, a more conservative estimate of the fitting error is used by taking the maximum absolute error,

$$\widehat{\epsilon}_{e,\max} = \max_{i=1,2,\ldots,P} \eta_{e,i} \tag{B.6}$$

To restate, it is preferable that the maximum absolute error, $\epsilon_{e,\max}$ is negligibly small, although it is not a quantity one can directly control. In some applications, discussed in Chapter 6 & 7, $\widehat{\epsilon}_{e,\max}$ is much larger than the experimental uncertainty over the domain $\mathcal{H}$. For these examples, a new approach is necessary. Discussed in Chapter 4, a novel strategy was developed using piecewise surrogate models to decompose the domain $\mathcal{H}$ into subdomains that have a smaller fitting error.

## B.3 Blind prediction

A blind prediction in B2BDC is conducted by first removing the $j$-th QOI model from the collection of model-data constraints that defines the feasible set, forming the following feasible set:

$$\mathcal{F}_{\setminus j} = \bigcap_{\substack{e=1 \\ e \neq j}}^{N} \mathcal{F}_e = \{x \in \mathcal{H} : L_e \leq M_e(x) \leq U_e, \text{ for } e = 1, 2, \ldots, N\}. \tag{B.7}$$

where the set $\mathcal{F}_{\setminus j}$ is the intersection of all individual feasible sets excluding the $j$-th. The prediction of the $j$-th QOI model, $M_j(x)$, is "blind" or has no information of the $j$-th experimental measurement as it is was excluded from the set.

Blind prediction of the $j$-th QOI can be evaluated by Eq. 3.4 where $x \in \mathcal{F}_{\setminus j}$. Often a comparison is made between the blind prediction interval for the $j$-th QOI with the corresponding experimental bounds. The predicted interval and the experimental bounds will overlap if the set $\mathcal{F} \neq \varnothing$, i.e., the dataset is consistent. Conversely, if the feasible set with all QOIs is empty $\mathcal{F} = \varnothing$ and the set is non-empty with the $j$-th QOI removed, then the two intervals will appear disjoint.

# Appendix C

# Piecewise modeling for B2BDC Toolbox

The MATLAB based uncertainty quantification toolbox, Bound-to-Bound Data Collaboration, was expanded to include piecewise surrogate models. The `PiecewiseModel` class and the associated methods are listed below and are available at https://github.com/oreluk/B2BDC-PiecewiseModels.

The B2BDC Toolbox can be found at https://github.com/B2BDC/B2BDC, where commit `e9d501a` was used throughout the dissertation.

Source Code C.1: PiecewiseModel.m

```
classdef PiecewiseModel < B2BDC.B2Bmodels.Model

    properties
        ModelTree = []; % Array of submodels
        Options = [];
        FunctionHandle = []; % Used to build piecewise model
        Type = [];
        ModelConsistency = []; % Index of consistent models in ModelTree
        %  each model in ModelTree
        ErrorFlag = 0; % Used to indicate a specific error in the
        %  construction of a pwm
    end

    properties (Hidden = true)
        Depth = 0; % Used in treeBuilder to determine current depth of node
    end

    methods
        function obj = PiecewiseModel(funcHandle, type, varList, data, option)
            % PiecewiseModel constructs a binary tree of surrogate models
            % where each leave is a new sub-domain of the original VARLIST.
            % Surrogate models are fit to FUNCHANDLE as specified by TYPE.
            % FUNCHANDLE is a function handle which takes as an input a
            % matrix of size nSamples-by-nVars and returns a vector of
            % size nSamples-by-1 of responses. ERRTOL and DEPTHTOL both
            % terminate branching of the binary tree. ERRTOL specifies an
            % absolute maximum error tolerance to be satisfied. MAXDEPTH
            % specifies how many branches from the root a node can span.
            % VARLIST is a VariableList specifying the original domain.
            % DATA is an optional structure input with fields of DATA.X and
            % DATA.y if the function has been evaluated
            % previous.

            % User supplied data
            if nargin > 3
                if ~isempty(data)
                    if ~isstruct(data)
                        error('Data must be supplied as a structure with X and y
                        ↪  fields');
                    end
                else
                    data.X = [];
                    data.y = [];
                end
            end

            if nargin > 4
                if isstruct(option)
                    obj.Options = option;
                else
                    error('Options for Piecewise model must be created by
                    ↪  piecewiseOptions()')
                end
            else
                obj.Options = piecewiseOptions;
            end

            if ~isa(funcHandle, 'function_handle')
                error('Must provide a function_handle to construct a PiecewiseModel');
            elseif ~ischar(type)
                error('Must provide a string specifying the type of models to fit for a
                ↪  PiecewiseModel (qinf/q2norm/rq)');
            elseif ~isa(varList, 'B2BDC.B2Bvariables.VariableList')
                error('Must provide a B2BDC.B2Bvariables.VariableList for constructions
                ↪  of a PiecewiseModel');
            end

            obj.FunctionHandle = funcHandle;
            obj.Type = type;
            obj.Variables = varList;
            obj.grow(varList, data);
```

```

        end

        function idx = findModelIndex(obj, X)
            % Takes in a matrix of design points X and returns a vector of
            % indices idx for the relevant models in ModelTree

            nSamples = size(X,1);
            nModels = length(obj.ModelTree);
            idx = zeros(nSamples,1);
            for ii = 1:nSamples
                for jj = 1:nModels
                    if obj.inDomain(obj.ModelTree(jj).Variables, X(ii,:))
                        idx(ii) = jj;
                        break;
                    end
                end
            end
        end

        function y = eval(obj, X)
            % Takes in an matrix of points, finds the B2BDC.B2Bmodels in
            % ModelTree associated with those points, and evaluates the
            % relevant models to return a vector of output.

            nSamples = size(X,1);
            idx = obj.findModelIndex(X);
            y = zeros(nSamples,1);

            % Loop over models and evaluate the data points
            modelIdx = unique(idx);
            for ii = 1:length(modelIdx)
                filter = idx == modelIdx(ii);
                y(filter) = obj.ModelTree(modelIdx(ii)).eval(X(filter, :));
            end
        end

        function y = length(obj)
            % Return number of models in obj.ModelTree
            y = length(obj.ModelTree);
        end

    end
end
```

Source Code C.2: grow.m

```
function obj = grow(obj, listOrIdx, data)
% OBJ = GROW(OBJ, LISTORIDX) will take a PiecewiseModel object OBJ and
% grow the tree at a LISTORIDX, where LISTORIDX is either a
% B2BDC.B2Bvariables.VariableList or an index of a pre-existing model.
% This allows a user to build a tree specified by the
% B2BDC.B2Bvariable.VariableList. If LISTORIDX is a numerical value
% specifying the model index to grow from, then the
% B2BDC.B2Bvariable.VariableList will be taken from the specified model.
%
% Grow is recursively called by branch when error tolerance, max depth
% tolerance or maximum number of models is not met.
%

%% Check inputs
if nargin < 3
    data.X = [];
    data.y = [];
end

if isa(listOrIdx, 'B2BDC.B2Bvariables.VariableList')
    if size(listOrIdx, 2) > 1
        % Array of VariableLists
        for ii = 1:size(listOrIdx, 2)
            obj.grow(listOrIdx(ii), data);
        end
        return
    else
        varList = listOrIdx;
        % TODO we should remove any model with this variable list
    end
elseif isnumeric(listOrIdx)
    % An index is only passed to grow when building/growing from an already
    %  completed PWM

    if listOrIdx > length(obj.ModelTree)
        error('Model Index exceeds number of Models in ModelTree.');
    end
    for ii = 1:numel(listOrIdx)
        varList = obj.ModelTree(listOrIdx(ii)).Variables;
        data = obj.ModelTree(listOrIdx(ii)).Data;

        % Remove model from tree
        obj.ModelTree(listOrIdx(ii)) = [];
        obj.ModelConsistency(listOrIdx(ii)) = [];

        obj.grow(varList, data);
    end
    return
end

%% Create output table

```

```
if obj.Depth == 0 && obj.Options.Verbose
    fprintf(['\n\t size(ModelTree) \t\t Tree Depth \t\t max(|Error|) \t\t Nodes Remaining
    ↪  \n', ...
        '\t----------------- \t\t------------- \t\t-------------- \t\t-----------------
        ↪  \n'])
end

%% Fit model of OBJ.TYPE on domain of VARLIST

[newModel, data] = obj.fitSubDomain(varList, data);

if strcmpi(obj.Options.ErrorType, 'absolute')
    modelError = newModel.ErrorStats.absMax;
elseif strcmpi(obj.Options.ErrorType, 'relative')
    modelError = newModel.ErrorStats.relMax;
else
    error('Option:ErrorType is unknown')
end

%% Check Self Consistency
if ~obj.ErrorFlag
    selfConsistent = true;
    if ~isempty(obj.Options.ExpBounds)
        opt = generateOpt();
        opt.Display = false;
        opt.AddFitError = true;
        dsTest = B2BDC.B2Bdataset.Dataset;
        dsUnit = generateDSunit('test', newModel, obj.Options.ExpBounds);
        dsTest.addDSunit(dsUnit);
        dsTest.isConsistent(opt);


        measure = dsTest.ConsistencyMeasure(2);
        if measure < 0
            selfConsistent = 0;
        else
            selfConsistent = 1;
        end
    end

    % to be used with 'approach2' code which will construct a dataset for multiple QOIs
    ↪  and
    % check for dataset consistency.
    if obj.Options.jointConsistency
         [selfConsistent, measure, modelError] = approach2(obj, newModel, data);
    end


    %% Decision to branch or save model

```

```
        s1 = modelError > obj.Options.ErrorTolerance;
        s2 = obj.Depth < obj.Options.MaxDepth;
        s3 = length(obj.ModelTree) < obj.Options.MaxNumberOfModels;

        if s1 && s2 && s3 && selfConsistent
            % Branch domain and increase depth
            obj.Depth = obj.Depth+1;
            printProgress(1);
            obj.branch(varList, data);
        else
            % Save model to ModelTree
            newModel.Data = data;
            obj.ModelTree = [obj.ModelTree newModel];

            if ~isempty(obj.Options.ExpBounds)
                obj.ModelConsistency = [obj.ModelConsistency measure];
            end
            printProgress(2);

            % Update PiecewiseModel error as maximum of error in ModelTree
            errStruct = [obj.ModelTree.ErrorStats];
            obj.ErrorStats.absMax = max([errStruct.absMax]);

            d = obj.ModelTree;
            save('currentPWM_progress', 'd');
        end
    else
        % Error Occured in FitSubDomain
        obj.ErrorFlag = 0;

        % Save model to ModelTree
        newModel.Data = data;
        obj.ModelTree = [obj.ModelTree newModel];

        if ~isempty(obj.Options.ExpBounds)
            obj.ModelConsistency = [obj.ModelConsistency -Inf];
        end
        printProgress(2);

        % Update PiecewiseModel error as maximum of error in ModelTree
        try
            errStruct = [obj.ModelTree.ErrorStats];
        catch
            keyboard
        end
        obj.ErrorStats.absMax = max([errStruct.absMax]);
        obj.ErrorStats.relMax = max([errStruct.relMax]);

        d = obj.ModelTree;
        save('currentPWM_progress', 'd');
```

```
end

%% Internal Function

    function printProgress(flag)
        % PRINTPROGRESS updates the progress of fitting the PiecewiseModel
        if flag == 1 % update current line of progress
            if obj.Options.Verbose
                rev = repmat(sprintf('\b'), 1, length(obj.Options.OutputMessage));
                obj.Options.OutputMessage = sprintf(['\t\t ', ...
                    num2str(length(obj.ModelTree)+1), '\t\t\t\t\t\t\t ', ...
                    num2str(obj.Depth), '/', num2str(obj.Options.MaxDepth), ...
                    '\t\t\t   ', num2str(round(modelError,3)), '\t\t\t\t', ...
                    num2str(obj.Options.remainingNodes-1), '\n']);
                fprintf([rev, obj.Options.OutputMessage]);
            end

        elseif flag == 2 % create a new line of progress
            nodesBelow = 2^(obj.Options.MaxDepth-obj.Depth);
            obj.Options.remainingNodes = obj.Options.remainingNodes - nodesBelow;

            if obj.Options.Verbose
                rev = repmat(sprintf('\b'), 1, length(obj.Options.OutputMessage));
                obj.Options.OutputMessage = sprintf(['\t\t ', ...
                    num2str(length(obj.ModelTree)), '\t\t\t\t\t\t\t ', ...
                    num2str(obj.Depth), '/', num2str(obj.Options.MaxDepth), ...
                    '\t\t\t   ', num2str(round(modelError,3)), '\t\t\t\t', ...
                    num2str(obj.Options.remainingNodes) ,'\n']);
                fprintf([rev, obj.Options.OutputMessage]);
                obj.Options.OutputMessage = '';
            end
        end
    end
end
```

Source Code C.3: fitSubDomain.m

```
function  [newModel, data] = fitSubDomain(obj, varList, data)
% NEWMODEL = fitSubDomain(OBJ, VARLIST, DATA)

% FITSUBDOMAIN will take in a PiecewiseModel OBJ and determine from TYPE
% how many samples are needed to fit on new sub-domain. TYPE is a string
% specifying the type of surrogate models used in GENERATEMODELBYFIT.
% Previous data coming from OBJ.DATA is recycled and only if new data is
% needed, FUNCHANDLE is evaluated on new X values to fit a model of TYPE.
% VARLIST is a B2BDC.B2Bvariables.VariableList specifying the new
% sub-domain to fit a surrogate model on.

if strcmpi(obj.Type, 'qinf') || strcmpi(obj.Type, 'q2norm')
```

```
    nSample = 4 * (length(varList) + 2) * (length(varList) + 1);
elseif strcmpi(obj.Type, 'rq')
    nSample = 4 * (length(varList) + 2) * (length(varList) + 1);
else
    error('Type not recognized.')
end

% Input Data is in Domain
if obj.Depth == 0
    oldData = data;
    inVarList = obj.inDomain(varList, oldData.X);
    idx = find(inVarList == 1);
    X = oldData.X(idx,:);
    y = oldData.y(idx,:);
else
    X = data.X;
    y = data.y;
end

nX = size(X,1);

if nX < nSample
    XNew = varList.makeLHSsample(nSample-nX);
    [yNew, XNew] = obj.FunctionHandle(XNew);

    X = [X; XNew];
    y = [y; yNew];


    %% If returned data is less than 75% of the samples we need
    % repeat sampling 5 times

    iter = 0;
    while length(y) < 0.75 * nSample && iter < 10
        iter = iter + 1;
        nX = size(X,1);
        XNew = varList.makeLHSsample(nSample-nX);
        [yNew, XNew] = obj.FunctionHandle(XNew);

        X = [X; XNew];
        y = [y; yNew];
    end

     %% Check if enough data is returned
    if length(y) < 0.75 * nSample
        obj.ErrorFlag = 1;
        nVar = length(varList);
        coef = zeros(nVar+1, nVar+1);
        coef(1,1) = Inf;
        newModel = B2BDC.B2Bmodels.QModel(coef, varList);
        newModel.ErrorStats.absMax = -1;
        newModel.ErrorStats.absAvg = -1;
        newModel.ErrorStats.relMax = -1;
        newModel.ErrorStats.relAvg = -1;
        newModel.ErrorStats.PhysicalUnits.Absolute = -1;
        newModel.ErrorStats.PhysicalUnits.Relative = -1;
        data = [];
        warning('Returned Data was less than 75% which was requested. Domain deemed
        ↪  invalid due to insufficent data')
        return
    end
end


%% for char oxidation we will only fit on qoi172 from y
if obj.Options.jointConsistency
    yOriginal = y;
    y = y(:,obj.Options.QOI);
end


%% Estimate Test Error: k-fold
% s = RandStream('mt19937ar','Seed',0);
% permIdx = randperm(s, size(X,1)); % Random ordering of X/y data

permIdx = randperm(size(X,1));

kFolds = 10;
nPerFold = floor(size(X,1)/kFolds); %
err = [];
relErr = [];

for k = 1:kFolds
    % kRange is defines elements for each fold
    if k == kFolds
        kRange = (k*nPerFold + 1) - nPerFold : size(X, 1);
    else
        kRange = (k*nPerFold + 1) - nPerFold : k*nPerFold;
    end

    % k-fold index is zero for elements in k-th fold
    kIndex = ones(length(y), 1);
    kIndex(kRange) = 0;

    % Filter X/y data for elements in k-fold index
    Xtrain = X(permIdx(logical(kIndex)),:);
    yTrain = y(permIdx(logical(kIndex)));

    Xtest = X(permIdx(~logical(kIndex)),:);
    yTest = y(permIdx(~logical(kIndex)));

    newModel = generateModelbyFit(Xtrain, yTrain, varList, obj.Type);
    yPred = newModel.eval(Xtest);
    e = abs(yPred - yTest);
    relE = abs(e./yTest);
    err = [err; e];
    relErr = [relErr; relE];
end

maxError = max(err);
maxRelativeError = max(relErr);
newModel = generateModelbyFit(X, y, varList, obj.Type);
newModel.ErrorStats.absMax = maxError * obj.Options.SafetyFactor;
newModel.ErrorStats.relMax = maxRelativeError * obj.Options.SafetyFactor;

%% Estimate Test Error: Hold Out set  70/30 split (train/test)
% percTraining = 0.7;
% trainingIdx = randperm(size(y,1));
% nTraining = floor(percTraining*size(y,1));
%
% xTrain = X(trainingIdx(1:nTraining), :);
% xTest = X(trainingIdx(nTraining+1:end), :);
%
% yTrain = y(trainingIdx(1:nTraining), :);
% yTest =  y(trainingIdx(nTraining+1:end), :);
%
% % Build Model
% newModel = generateModelbyFit(xTrain, yTrain, varList, obj.Type);
%
% % Evaluate Test Error
% ySurrogate = newModel.eval(xTest);
% absE = abs(ySurrogate - yTest);
%
% % Update Error
% newModel.ErrorStats.absMax = max(absE);
% newModel.ErrorStats.absAvg = mean(absE);
% relE = absE./yTest;
% newModel.ErrorStats.relMax = max(relE);
% newModel.ErrorStats.relAvg = mean(relE);




%% chemical kinetics example
if obj.Options.jointConsistency
    y = yOriginal;
end


%% Update Data
```

```


data.X = X;
data.y = y;
```

Source Code C.4: branch.m

```
function branch(obj, varList, data)
% BRANCH(OBJ, VARLIST) branches a binary tree into two sub-domains
% (children nodes) of VARLIST1 and VARLIST2. The VARLIST is divided by
% heuristics detailed in RULE.

[varList1, varList2, data] = obj.rule(varList, data); % Split VARLIST

% Divide data into two parts
inVarList = obj.inDomain(varList1, data.X);
data1.X = data.X(logical(inVarList), :);
data1.y = data.y(logical(inVarList), :);
data2.X = data.X(~inVarList, :);
data2.y = data.y(~inVarList, :);

obj.grow(varList1, data1); % Child Node 1
obj.grow(varList2, data2); % Child Node 2
obj.Depth = obj.Depth - 1;
```

Source Code C.5: rule.m

```
function [varList1, varList2, data] = rule(obj, varList, data)
% [VARLIST1, VARLIST2] = RULE(OBJ, VARLIST) heuristic rules for determining
% sub-domains of VARLIST.

% RULE is used to determine how and where to split VARLIST which is a
% B2BDC.B2Bvariables.VariableList into two sub-domains VARLIST1 and
% VARLIST2.

%% RULE: Split largest parameter uncertainty in half
%
% [varIdx, newBound1, newBound2, data] = obj.ruleSplitLargest(varList, data);

%% RULE: Split random variable in half
%
% [varIdx, newBound1, newBound2, data] = obj.ruleSplitRandom(varList, data);

%% RULE: Split each variable one at a time and fit with a 2norm quadratic
% model. Model with smallest 2norm error is the varIdx to be split in half.

% [varIdx, newBound1, newBound2, data] = obj.ruleSplitMinError(varList, data);

```

```
%% RULE: Split each variable by the golden ratio one-at-a-time
%
% [varIdx, newBound1, newBound2, data] = obj.ruleSplitMinErrorGolden(varList, data);

%% RULE: Split each variable one-at-a-time and fit with a 2 norm quadratic model.
% Model with smallest 2 norma error is the var to split, splits can occur at
% (1/4, 1/3, 1/2, 2/3, 3/4)
%
%
% [varIdx, newBound1, newBound2, data] = obj.ruleSplitMinErrorVariousPartitions(varList,
↪  data)

%% Rule: Split each variable one-at-a-time and fit with a 2norm quadratic
% k-fold is used to estimate error.

[varIdx, newBound1, newBound2, data] = obj.ruleSplitMinErrorKfold(varList, data);

%%

varList1 = varList.changeBound(newBound1, varIdx(1));
varList2 = varList.changeBound(newBound2, varIdx(1));
```

Source Code C.6: ruleSplitMinErrorKfold.m

```
function [varIdx, newBound1, newBound2, data] = ruleSplitMinErrorKfold(obj, varList,
↪  data)
oldData = data;

%
% Change Template QOI if consistency & error of Template is less than Error
% Tolerance.
%

if obj.Options.ChangeTemplateQOI == 1
    obj.Options.ChangeTemplateQOI = 0;
    ds = obj.Options.Dataset{length(obj.ModelTree) + 1};

    for i = 1:length(ds)
        absErr(i) = ds.DatasetUnits.Values(i).SurrogateModel.ErrorStats.absMax;
        relErr(i) = ds.DatasetUnits.Values(i).SurrogateModel.ErrorStats.relMax;
    end

    if strcmpi(obj.Options.ErrorType, 'absolute')
        qoi = find(max(absErr) == absErr);
    elseif strcmpi(obj.Options.ErrorType, 'relative')
        qoi = find(max(relErr) == relErr);
    end
else
    qoi = obj.Options.QOI;
```

```
end

%%

nVar = length(varList);
nSample = 2 * (nVar + 1) * (nVar + 2);
for i = 1:nVar
    var2Split = varList.Values(i);
    l = var2Split.LowerBound;
    u = var2Split.UpperBound;
    newBound1 = [l (u+l)/2];
    newBound2 = [(u+l)/2 u];

    % Fit Child 1
    v1 = varList.changeBound(newBound1, i);
    inVarList1 = obj.inDomain(v1, oldData.X);
    idx1 = find(inVarList1 == 1);
    X1 = oldData.X(idx1, :);
    y1 = oldData.y(idx1, :);
    nX = size(X1,1);


    if nX < nSample
        % do not have enough points...
        % add diag noise (regularizer)
        %         warning('Required more Samples in Rule')

        % for the time being...take a little more samples than required to be a
        ↪  determined system.
        X1new = v1.makeLHSsample(floor(1.1*(nSample - nX)));
        [y1New, X1new] = obj.FunctionHandle(X1new);

        y1 = [y1; y1New];
        X1 = [X1; X1new];

        % save new data generated
        oldData.X  = [oldData.X; X1new];
        oldData.y = [oldData.y; y1New];
    end



    %% Estimate Test Error: k-fold
    kFolds = 10;
    permIdx = randperm(size(X1,1));
    nPerFold = floor(size(X1,1)/kFolds); %
    err = [];
    relErr = [];

    for k = 1:kFolds
```

```
        % kRange is defines elements for each fold
        if k == kFolds
            kRange = (k*nPerFold + 1) - nPerFold : size(X1, 1);
        else
            kRange = (k*nPerFold + 1) - nPerFold : k*nPerFold;
        end

        % k-fold index is zero for elements in k-th fold
        kIndex = ones(size(X1,1), 1);
        kIndex(kRange) = 0;

        % Filter X/y data for elements in k-fold index
        try
            Xtrain = X1(permIdx(logical(kIndex)),:);

            yTrain = y1(permIdx(logical(kIndex)), qoi);

        catch
            keyboard
        end


        Xtest = X1(permIdx(~logical(kIndex)),:);
        yTest = y1(permIdx(~logical(kIndex)), qoi);

        newModel = generateModelbyFit(Xtrain, yTrain, v1, 'q2norm');
        yPred = newModel.eval(Xtest);
        e = abs(yPred - yTest);
        relE = abs(e./yTest);
        err = [err; e];
        relErr = [relErr; relE];
    end

    maxError = max(err);
    maxRelativeError = max(relErr);
    newModel1 = generateModelbyFit(X1, y1(:,qoi), v1, 'q2norm');
    newModel1.ErrorStats.absMax = maxError * obj.Options.SafetyFactor;
    newModel1.ErrorStats.relMax = maxRelativeError * obj.Options.SafetyFactor;

    %% Fit Child 2
    v2 = varList.changeBound(newBound2, i);
    inVarList2 = obj.inDomain(v2, oldData.X);
    idx2 = find(inVarList2 == 1);
    X2 = oldData.X(idx2,:);
    y2 = oldData.y(idx2,:);
    nX = size(X2,1);

    if nX < nSample
        % do not have enough points...
        % add diag noise (regularizer)
        %          warning('Required more samples in Rule')
        % for the time being...take a little more samples than required to be a
        ↪  determined system.
        X2new = v2.makeLHSsample(floor(1.1*(nSample - nX)));
        [y2New, X2new] = obj.FunctionHandle(X2new);

        y2 = [y2; y2New];
        X2 = [X2; X2new];

        % save new data generated
        oldData.X  = [oldData.X; X2new];
        oldData.y = [oldData.y; y2New];
    end



    %% Estimate Test Error: k-fold
    kFolds = 10;
    permIdx = randperm(size(X2,1));
    nPerFold = floor(size(X2,1)/kFolds); %
    err = [];
    relErr = [];

    for k = 1:kFolds
        % kRange is defines elements for each fold
        if k == kFolds
            kRange = (k*nPerFold + 1) - nPerFold : size(X2, 1);
        else
            kRange = (k*nPerFold + 1) - nPerFold : k*nPerFold;
        end

        % k-fold index is zero for elements in k-th fold
        kIndex = ones(size(X2,1), 1);
        kIndex(kRange) = 0;

        % Filter X/y data for elements in k-fold index
        try
            Xtrain = X2(permIdx(logical(kIndex)),:);

            yTrain = y2(permIdx(logical(kIndex)), qoi);
        catch
            keyboard
        end

        Xtest = X2(permIdx(~logical(kIndex)),:);
        yTest = y2(permIdx(~logical(kIndex)), qoi);

        newModel = generateModelbyFit(Xtrain, yTrain, v2, 'q2norm');
        yPred = newModel.eval(Xtest);
        e = abs(yPred - yTest);
        relE = abs(e./yTest);
        err = [err; e];
        relErr = [relErr; relE];
    end

    maxError = max(err);
    maxRelativeError = max(relErr);
    newModel2 = generateModelbyFit(X2, y2(:,qoi), v2, 'q2norm');
    newModel2.ErrorStats.absMax = maxError * obj.Options.SafetyFactor;
    newModel2.ErrorStats.relMax = maxRelativeError * obj.Options.SafetyFactor;

    minError(i) = min(newModel1.ErrorStats.absMax, newModel2.ErrorStats.absMax);
end

varIdx = find(min(minError) == minError);

var2Split = varList.Values(varIdx(1));
l = var2Split.LowerBound;
u = var2Split.UpperBound;
newBound1 = [l (u+l)/2];
newBound2 = [(u+l)/2 u];

data = oldData;
```

Source Code C.7: inDomain.m

```
function inD = inDomain(obj, varList, X)
% IND = INDOMAIN(OBJ, VARLIST, X) checks to see if X is in the domain of
% VARLIST.
%
% THIS FUNCTION MAY BE UNNECESSARY, VARLIST HAS IT'S OWN ISFEASIBLEPOINT IN
% B2BDC v0.8.

if isempty(X)
    inD = 0;
end

% This needs to change for linear constraints, also the function should be
% the equivalent of isFeasiblePoint but for variablesList objects rather
% than datasets.

H = varList.calBound;
nSamples = size(X,1);
inD = ones(nSamples,1);

for ii = 1:nSamples
    if any(H(:,1) > X(ii,:)')
        inD(ii) = 0;
    elseif any(H(:,2) < X(ii,:)')
```

```
        inD(ii) = 0;
    end
end
```

# Appendix D

# $H_2/O_2$ mechanisms

### Cantera Input

All calculations were conducted using each of the respective mechanism's HDF5 file found in the PrIMe Data Warehouse. XML, HDF5, and MAT files for all mechanisms are available online via the WarehouseAPI and at https://github.com/oreluk/h2o2_datasets/. Unfortunately, the HDF5 file format does not lend itself to text documents; therefore, all mechanisms were converted to CANTERA CTI files via CloudFlame's HDF5 to CTI converter [51].

## D.1 Mechanism1 - m00000007.cti

Source Code D.1: m00000007.cti

```
#-------------------------------------------------------------------------------
#HDF5
# CTI Generated : 2017-12-02 01:04:44
#-------------------------------------------------------------------------------

units(length='cm', time='s', quantity='mol', act_energy='cal/mol')


ideal_gas(name='gas',
          elements="Ar C H He N O",
          species="""Ar He N2 C(GR) CO
                     CO2 H H2 O O2
                     OH H2O HO2 H2O2""",
          reactions='all',
          initial_state=state(temperature= 298.15, pressure= 100000.0))

#-------------------------------------------------------------------------------
```

```
#Species data
#--------------------------------------------------------------------------------

species(name='Ar',
        atoms='Ar:1',
        thermo=(NASA([300.0, 1000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, -745.375,
                      4.366001]),
                NASA([1000.0, 5000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, -745.375,
                      4.366001])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 136.5,
                diam = 3.33))

species(name='He',
        atoms='He:1',
        thermo=(NASA([300.0, 1000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, -745.375,
                      0.9153488]),
                NASA([1000.0, 5000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, -745.375,
                      0.9153488])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 10.2,
                diam = 2.576))

species(name='N2',
        atoms='N:2',
        thermo=(NASA([300.0, 1000.0],
                     [3.298677, 0.00140824, -3.96322e-06,
                      5.64152e-09, -2.44486e-12, -1020.9,
                      3.950372]),
                NASA([1000.0, 5000.0],
                     [2.92664, 0.001487977, -5.68476e-07,
                      1.0097e-10, -6.75335e-15, -922.7977,
                      5.980528])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 97.53,
                diam = 3.621,
                polar = 1.76,
                rot_relax = 4.0))

species(name='C(GR)',
        atoms='C:1',
        thermo=(NASA([200.0, 1000.0],
                     [-0.31087, 0.0044035, 1.9039e-06,
                      -6.3855e-09, 2.9896e-12, -108.6508,
                      1.1138]),
                NASA([1000.0, 5000.0],
                     [1.4557, 0.0017171, -6.9758e-07,
                      1.3528e-10, -9.6765e-15, -695.128,
                      -8.5257])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 71.4,
                diam = 3.298))

species(name='CO',
        atoms='C:1 O:1',
        thermo=(NASA([300.0, 1000.0],
                     [3.262452, 0.001511941, -3.88176e-06,
                      5.58194e-09, -2.47495e-12, -14310.54,
                      4.848897]),
                NASA([1000.0, 5000.0],
                     [3.025078, 0.001442689, -5.63083e-07,
                      1.01858e-10, -6.91095e-15, -14268.35,
                      6.108218])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 98.1,
                diam = 3.65,
                polar = 1.95,
                rot_relax = 1.8))

species(name='CO2',
        atoms='C:1 O:2',
        thermo=(NASA([300.0, 1000.0],
                     [2.275725, 0.009922072, -1.04091e-05,
                      6.86669e-09, -2.11728e-12, -48373.14,
                      10.18849]),
                NASA([1000.0, 5000.0],
                     [4.453623, 0.003140169, -1.27841e-06,
                      2.394e-10, -1.66903e-14, -48966.96,
                      -0.9553959])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 244.0,
                diam = 3.763,
                polar = 2.65,
                rot_relax = 2.1))

species(name='H',
        atoms='H:1',
        thermo=(NASA([300.0, 1000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, 25471.63,
                      -0.4601176]),
                NASA([1000.0, 5000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, 25471.63,
                      -0.4601176])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 145.0,
                diam = 2.05))

species(name='H2',
        atoms='H:2',
        thermo=(NASA([300.0, 1000.0],
                     [3.298124, 0.000824944, -8.14302e-07,
                      -9.47543e-11, 4.13487e-13, -1012.521,
                      -3.294094]),
                NASA([1000.0, 5000.0],
                     [2.991423, 0.000700064, -5.63383e-08,
                      -9.23158e-12, 1.58275e-15, -835.034,
                      -1.35511])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 38.0,
                diam = 2.92,
                polar = 0.79,
                rot_relax = 280.0))

species(name='O',
        atoms='O:1',
        thermo=(NASA([300.0, 1000.0],
                     [2.946429, -0.001638166, 2.42103e-06,
                      -1.60284e-09, 3.8907e-13, 29147.64,
                      2.963995]),
                NASA([1000.0, 5000.0],
                     [2.54206, -2.75506e-05, -3.1028e-09,
                      4.55107e-12, -4.36805e-16, 29230.8,
                      4.920308])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 80.0,
                diam = 2.75))

species(name='O2',
        atoms='O:2',
        thermo=(NASA([300.0, 1000.0],
                     [3.212936, 0.001127486, -5.75615e-07,
                      1.31388e-09, -8.76855e-13, -1005.249,
                      6.034738]),
                NASA([1000.0, 5000.0],
                     [3.697578, 0.00061352, -1.25884e-07,
                      1.77528e-11, -1.13644e-15, -1233.93,
                      3.189166])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 107.4,
                diam = 3.458,
                polar = 1.6,
                rot_relax = 3.8))

species(name='OH',
        atoms='H:1 O:1',
        thermo=(NASA([200.0, 1000.0],
                     [4.12530561, -0.003225449, 6.52765e-06,
                      -5.79854e-09, 2.06237e-12, 3346.30913,
                      -0.69043296]),
                NASA([1000.0, 6000.0],
                     [2.86472886, 0.001056504, -2.59083e-07,
                      3.05219e-11, -1.33196e-15, 3683.62875,
                      5.70164073])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 80.0,
                diam = 2.75))

species(name='H2O',
        atoms='H:2 O:1',
        thermo=(NASA([300.0, 1000.0],
                     [3.386842, 0.003474982, -6.3547e-06,
                      6.96858e-09, -2.50659e-12, -30208.11,
                      2.590233]),
                NASA([1000.0, 5000.0],
                     [2.672146, 0.003056293, -8.73026e-07,
                      1.201e-10, -6.39162e-15, -29899.21,
                      6.862817])),
        transport = gas_transport(
                geom = "nonlinear",
                well_depth = 572.4,
                diam = 2.605,
                dipole = 1.844,
                rot_relax = 4.0))

species(name='HO2',
        atoms='H:1 O:2',
        thermo=(NASA([200.0, 1000.0],
                     [4.30179801, -0.004749121, 2.11583e-05,
                      -2.42764e-08, 9.29225e-12, 294.80804,
                      3.71666245]),
                NASA([1000.0, 3500.0],
                     [4.0172109, 0.00223982, -6.33658e-07,
                      1.14246e-10, -1.07909e-14, 111.856713,
                      3.78510215])),
        transport = gas_transport(
                geom = "nonlinear",
                well_depth = 107.4,
                diam = 3.458,
                rot_relax = 1.0))

species(name='H2O2',
        atoms='H:2 O:2',
        thermo=(NASA([300.0, 1000.0],
                     [3.388754, 0.006569226, -1.48501e-07,
                      -4.62581e-09, 2.47152e-12, -17663.15,
                      6.785363]),
                NASA([1000.0, 5000.0],
                     [4.573167, 0.004336136, -1.47469e-06,
                      2.3489e-10, -1.43165e-14, -18006.96,
                    0.501137])),
        transport = gas_transport(
                geom = "nonlinear",
                well_depth = 107.4,
                diam = 3.458,
                rot_relax = 3.8))


#-------------------------------------------------------------------------------
#Reaction data
#-------------------------------------------------------------------------------

#Reaction 1:
reaction('O2 + H  <=> OH + O', [1.04e+14, 0.0, 15309.835])

#Reaction 2:
reaction('H2 + O  <=> OH + H', [8.7923244e+14, 0.0, 19174.55],
         options='duplicate')

#Reaction 3:
reaction('H2 + O  <=> OH + H', [3.81803676e+12, 0.0, 7948.0],
         options='duplicate')

#Reaction 4:
reaction('H2 + OH  <=> H2O + H', [216194826.0, 1.51, 3429.562])

#Reaction 5:
reaction('OH + OH  <=> H2O + O', [33483.0984, 2.42, -1927.39])

#Reaction 6:
three_body_reaction('H + H + M <=> H2 + M', [3.143e+20, -1.806, 982.555066967],
         efficiencies='H2:2.5 H2O:12.0 CO:1.9 CO2:3.8 Ar:0.0 He:0.0 ')

#Reaction 7:
reaction('H + H + Ar <=> H2 + Ar', [4.011e+19, -1.506, 982.555066967])

#Reaction 8:
reaction('H + H + He <=> H2 + He', [4.011e+19, -1.506, 982.555066967])

#Reaction 9:
three_body_reaction('O + O + M <=> O2 + M', [6.16524893053e+15, -0.5, 0.0],
         efficiencies='H2:2.53 H2O:11.76 CO:1.88 CO2:3.82 Ar:0.0 He:0.0 ')

#Reaction 10:
```

```
reaction('O + O + Ar <=> O2 + Ar', [1.88584084934e+13, 0.0, -1788.3])

#Reaction 11:
reaction('O + O + He <=> O2 + He', [1.88584084934e+13, 0.0, -1788.3])

#Reaction 12:
three_body_reaction('H + O + M <=> OH + M', [4.71460212335e+18, -1.0, 0.0],
         efficiencies='H2:2.54 H2O:12.31 CO:1.92 CO2:3.77 He:0.75 Ar:0.75 ')

#Reaction 13:
three_body_reaction('OH + H + M <=> H2O + M', [1.483e+28, -3.798, 2706.52897774],
         efficiencies='H2:3.0 O2:1.5 N2:2.0 He:1.1 CO:1.9 CO2:3.8 H2O:0.0 ')

#Reaction 14:
reaction('OH + H + H2O <=> H2O + H2O', [2.46e+26, -2.916, 2096.18418054])

#Reaction 15:
falloff_reaction('O2 + H (+ M) <=> HO2 (+ M)',
                 kf=[4.65e+12, 0.44, 0.0],
                 kf0=[6.37e+20, -1.72, 524.568],
                 efficiencies='H2:1.99 O2:0.78 H2O:14.0 CO:1.9 CO2:3.8 Ar:0.67
                 ↪  He:0.8 ',
         falloff = Troe(A = 0.5, T3 = 1e-30, T1 = 1e+30, T2 = 0))

#Reaction 16:
reaction('O2 + H2  <=> HO2 + H', [1341000.0, 2.314, 53420.1782336])

#Reaction 17:
reaction('HO2 + H  <=> OH + OH', [7.08e+13, 0.0, 300.169572393])

#Reaction 18:
reaction('HO2 + O  <=> OH + O2', [28500000000.0, 1.0, -724.309178184])

#Reaction 19:
reaction('O2 + H2O  <=> HO2 + OH', [3.956e+13, 0.294, 69070.0191729])

#Reaction 20:
reaction('HO2 + HO2  <=> H2O2 + O2', [4.2e+14, 0.0, 11981.61],
         options='duplicate')

#Reaction 21:
reaction('HO2 + HO2  <=> H2O2 + O2', [130000000000.0, 0.0, -1629.34],
         options='duplicate')
```

```
#Reaction 22:
falloff_reaction('OH + OH (+ M) <=> H2O2 (+ M)',
                 kf=[45280000.0, 1.657, -1818.02704346],
                 kf0=[5.637e+19, -1.543, -1818.02704346],
                 efficiencies='H2:3.7 H2O:7.5 O2:1.2 H2O2:7.7 N2:1.5 He:0.65
                 ↪  CO:2.8 CO2:1.6 ',
         falloff = Troe(A = 0.43, T3 = 1e-30, T1 = 1e+30, T2 = 0))

#Reaction 23:
reaction('H2O2 + H  <=> OH + H2O', [2.408856e+13, 0.0, 3974.0])

#Reaction 24:
reaction('H2O2 + H  <=> H2 + HO2', [4.817712e+13, 0.0, 7948.0])

#Reaction 25:
reaction('O + H2O2  <=> HO2 + OH', [9635424.0, 2.0, 3974.0])

#Reaction 26:
reaction('OH + H2O2  <=> HO2 + H2O', [1.7378e+12, 0.0, 317.92],
         options='duplicate')

#Reaction 27:
reaction('OH + H2O2  <=> HO2 + H2O', [7.5858e+13, 0.0, 7272.42],
         options='duplicate')
```

## D.2 `Mechanism2 - m00000008.cti`

Source Code D.2: m00000008.cti

```
#-------------------------------------------------------------------------------
#HDF5
# CTI Generated : 2017-12-02 01:05:27
#-------------------------------------------------------------------------------

units(length='cm', time='s', quantity='mol', act_energy='cal/mol')


ideal_gas(name='gas',
          elements="Ar C H He N O",
          species="""Ar He N2 C(GR) CO
                     CO2 H H2 O O2
                     OH H2O HO2 H2O2""",
          reactions='all',
          initial_state=state(temperature= 298.15, pressure= 100000.0))

#-------------------------------------------------------------------------------
#Species data
#-------------------------------------------------------------------------------

species(name='Ar',
        atoms='Ar:1',
        thermo=(NASA([300.0, 1000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, -745.375,
                      4.366001]),
                NASA([1000.0, 5000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, -745.375,
                      4.366001])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 136.5,
                diam = 3.33))

species(name='He',
        atoms='He:1',
        thermo=(NASA([300.0, 1000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, -745.375,
                      0.9153488]),
                NASA([1000.0, 5000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, -745.375,
                      0.9153488])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 10.2,
                diam = 2.576))

species(name='N2',
        atoms='N:2',
        thermo=(NASA([300.0, 1000.0],
                     [3.298677, 0.00140824, -3.96322e-06,
                      5.64152e-09, -2.44486e-12, -1020.9,
                      3.950372]),
                NASA([1000.0, 5000.0],
                     [2.92664, 0.001487977, -5.68476e-07,
                      1.0097e-10, -6.75335e-15, -922.7977,
                      5.980528])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 97.53,
                diam = 3.621,
                polar = 1.76,
                rot_relax = 4.0))

species(name='C(GR)',
        atoms='C:1',
        thermo=(NASA([200.0, 1000.0],
                     [-0.31087, 0.0044035, 1.9039e-06,
                      -6.3855e-09, 2.9896e-12, -108.6508,
                      1.1138]),
                NASA([1000.0, 5000.0],
                     [1.4557, 0.0017171, -6.9758e-07,
                      1.3528e-10, -9.6765e-15, -695.128,
                      -8.5257])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 71.4,
                diam = 3.298))

species(name='CO',
        atoms='C:1 O:1',
        thermo=(NASA([300.0, 1000.0],
                     [3.262452, 0.001511941, -3.88176e-06,
                      5.58194e-09, -2.47495e-12, -14310.54,
                      4.848897]),
                NASA([1000.0, 5000.0],
                     [3.025078, 0.001442689, -5.63083e-07,
                      1.01858e-10, -6.91095e-15, -14268.35,
                      6.108218])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 98.1,
                diam = 3.65,
                polar = 1.95,
                rot_relax = 1.8))

species(name='CO2',
        atoms='C:1 O:2',
        thermo=(NASA([300.0, 1000.0],
                     [2.275725, 0.009922072, -1.04091e-05,
                      6.86669e-09, -2.11728e-12, -48373.14,
                      10.18849]),
                NASA([1000.0, 5000.0],
                     [4.453623, 0.003140169, -1.27841e-06,
                      2.394e-10, -1.66903e-14, -48966.96,
                      -0.9553959])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 244.0,
                diam = 3.763,
                polar = 2.65,
                rot_relax = 2.1))

species(name='H',
        atoms='H:1',
        thermo=(NASA([300.0, 1000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, 25471.63,
                      -0.4601176]),
                NASA([1000.0, 5000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, 25471.63,
                      -0.4601176])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 145.0,
                diam = 2.05))

species(name='H2',
        atoms='H:2',
        thermo=(NASA([300.0, 1000.0],
                     [3.298124, 0.000824944, -8.14302e-07,
                      -9.47543e-11, 4.13487e-13, -1012.521,
                      -3.294094]),
                NASA([1000.0, 5000.0],
                     [2.991423, 0.000700064, -5.63383e-08,
                      -9.23158e-12, 1.58275e-15, -835.034,
                      -1.35511])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 38.0,
                diam = 2.92,
                polar = 0.79,
                rot_relax = 280.0))

species(name='O',
        atoms='O:1',
        thermo=(NASA([300.0, 1000.0],
                     [2.946429, -0.001638166, 2.42103e-06,
                      -1.60284e-09, 3.8907e-13, 29147.64,
                      2.963995]),
                NASA([1000.0, 5000.0],
                     [2.54206, -2.75506e-05, -3.1028e-09,
                      4.55107e-12, -4.36805e-16, 29230.8,
                      4.920308])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 80.0,
                diam = 2.75))

species(name='O2',
        atoms='O:2',
        thermo=(NASA([300.0, 1000.0],
                     [3.212936, 0.001127486, -5.75615e-07,
                      1.31388e-09, -8.76855e-13, -1005.249,
                      6.034738]),
                NASA([1000.0, 5000.0],
                     [3.697578, 0.00061352, -1.25884e-07,
                      1.77528e-11, -1.13644e-15, -1233.93,
                      3.189166])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 107.4,
                diam = 3.458,
                polar = 1.6,
                rot_relax = 3.8))

species(name='OH',
        atoms='H:1 O:1',
        thermo=(NASA([200.0, 1000.0],
                     [4.12530561, -0.003225449, 6.52765e-06,
                      -5.79854e-09, 2.06237e-12, 3346.30913,
                      -0.69043296]),
                NASA([1000.0, 6000.0],
                     [2.86472886, 0.001056504, -2.59083e-07,
                      3.05219e-11, -1.33196e-15, 3683.62875,
                      5.70164073])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 80.0,
                diam = 2.75))

species(name='H2O',
        atoms='H:2 O:1',
        thermo=(NASA([300.0, 1000.0],
                     [3.386842, 0.003474982, -6.3547e-06,
                      6.96858e-09, -2.50659e-12, -30208.11,
                      2.590233]),
                NASA([1000.0, 5000.0],
                     [2.672146, 0.003056293, -8.73026e-07,
                      1.201e-10, -6.39162e-15, -29899.21,
                      6.862817])),
        transport = gas_transport(
                geom = "nonlinear",
                well_depth = 572.4,
                diam = 2.605,
                dipole = 1.844,
                rot_relax = 4.0))

species(name='HO2',
        atoms='H:1 O:2',
        thermo=(NASA([200.0, 1000.0],
                     [4.30179801, -0.004749121, 2.11583e-05,
                      -2.42764e-08, 9.29225e-12, 294.80804,
                      3.71666245]),
                NASA([1000.0, 3500.0],
                     [4.0172109, 0.00223982, -6.33658e-07,
                      1.14246e-10, -1.07909e-14, 111.856713,
                      3.78510215])),
        transport = gas_transport(
                geom = "nonlinear",
                well_depth = 107.4,
                diam = 3.458,
                rot_relax = 1.0))

species(name='H2O2',
        atoms='H:2 O:2',
        thermo=(NASA([300.0, 1000.0],
                     [3.388754, 0.006569226, -1.48501e-07,
                      -4.62581e-09, 2.47152e-12, -17663.15,
                      6.785363]),
                NASA([1000.0, 5000.0],
                     [4.573167, 0.004336136, -1.47469e-06,
                      2.3489e-10, -1.43165e-14, -18006.96,
                      0.501137])),
        transport = gas_transport(
                geom = "nonlinear",
                well_depth = 107.4,
                diam = 3.458,
                rot_relax = 3.8))


#-------------------------------------------------------------------------------
#Reaction data
#-------------------------------------------------------------------------------

#Reaction 1:
reaction('O2 + H  <=> OH + O', [1.04e+14, 0.0, 15309.835])

#Reaction 2:
reaction('H2 + O  <=> OH + H', [8.7923244e+14, 0.0, 19174.55],
         options='duplicate')

#Reaction 3:
reaction('H2 + O  <=> OH + H', [3.81803676e+12, 0.0, 7948.0],
         options='duplicate')

#Reaction 4:
```

```
reaction('H2 + OH  <=> H2O + H', [216194826.0, 1.51, 3429.562])

#Reaction 5:
reaction('OH + OH  <=> H2O + O', [2.83e+13, -0.764, -460.560180575],
         options='duplicate')

#Reaction 6:
reaction('OH + OH  <=> H2O + O', [12600.0, 2.5308, -1622.91682141],
         options='duplicate')

#Reaction 7:
three_body_reaction('H + H + M <=> H2 + M', [3.143e+20, -1.806, 982.555066967],
         efficiencies='H2:2.5 H2O:12.0 CO:1.9 CO2:3.8 Ar:0.0 He:0.0 ')

#Reaction 8:
reaction('H + H + Ar <=> H2 + Ar', [4.011e+19, -1.506, 982.555066967])

#Reaction 9:
reaction('H + H + He <=> H2 + He', [4.011e+19, -1.506, 982.555066967])

#Reaction 10:
three_body_reaction('O + O + M <=> O2 + M', [6.16524893053e+15, -0.5, 0.0],
         efficiencies='H2:2.53 H2O:11.76 CO:1.88 CO2:3.82 Ar:0.0 He:0.0 ')

#Reaction 11:
reaction('O + O + Ar <=> O2 + Ar', [1.88584084934e+13, 0.0, -1788.3])

#Reaction 12:
reaction('O + O + He <=> O2 + He', [1.88584084934e+13, 0.0, -1788.3])

#Reaction 13:
three_body_reaction('H + O + M <=> OH + M', [4.71460212335e+18, -1.0, 0.0],
         efficiencies='H2:2.54 H2O:12.31 CO:1.92 CO2:3.77 He:0.75 Ar:0.75 ')

#Reaction 14:
three_body_reaction('OH + H + M <=> H2O + M', [1.483e+28, -3.798, 2706.52897774],
         efficiencies='H2:3.0 O2:1.5 N2:2.0 He:1.1 CO:1.9 CO2:3.8 H2O:0.0 ')

#Reaction 15:
reaction('OH + H + H2O <=> H2O + H2O', [2.46e+26, -2.916, 2096.18418054])

#Reaction 16:
falloff_reaction('O2 + H (+ M) <=> HO2 (+ M)',
                 kf=[4.65e+12, 0.44, 0.0],
                 kf0=[6.37e+20, -1.72, 524.568],
                 efficiencies='H2:1.99 O2:0.78 H2O:14.0 CO:1.9 CO2:3.8 Ar:0.67
                 ↪  He:0.8 ',
         falloff = Troe(A = 0.5, T3 = 1e-30, T1 = 1e+30, T2 = 0))

#Reaction 17:
reaction('O2 + H2  <=> HO2 + H', [1341000.0, 2.314, 53420.1782336])

#Reaction 18:
reaction('HO2 + H  <=> OH + OH', [7.08e+13, 0.0, 300.169572393])

#Reaction 19:
reaction('HO2 + O  <=> OH + O2', [28500000000.0, 1.0, -724.309178184])

#Reaction 20:
reaction('O2 + H2O  <=> HO2 + OH', [2.642e+20, -2.194, 70150.6296335],
         options='duplicate')

#Reaction 21:
reaction('O2 + H2O  <=> HO2 + OH', [1656000000.0, 1.533, 68259.5613274],
         options='duplicate')

#Reaction 22:
reaction('HO2 + HO2  <=> H2O2 + O2', [1510.0, 2.6969, -3866.702],
         options='duplicate')

#Reaction 23:
reaction('HO2 + HO2  <=> H2O2 + O2', [2.5e+15, -1.461, -1469.7839],
         options='duplicate')

#Reaction 24:
falloff_reaction('OH + OH (+ M) <=> H2O2 (+ M)',
                 kf=[5.03e+12, 0.058, -634.68854952],
                 kf0=[5.68e+25, -3.358, 576.325578995],
                 efficiencies='CO2:2.99 N2:2.01 He:0.4 H2O:10.0 ',
         falloff = Troe(A = 0.55, T3 = 1e-30, T1 = 1e+30, T2 = 0))

#Reaction 25:
reaction('H2O2 + H  <=> OH + H2O', [2.408856e+13, 0.0, 3974.0])

#Reaction 26:
reaction('H2O2 + H  <=> H2 + HO2', [4.817712e+13, 0.0, 7948.0])

#Reaction 27:
```

```
reaction('O + H2O2  <=> HO2 + OH', [9635424.0, 2.0, 3974.0])

#Reaction 28:
reaction('OH + H2O2  <=> HO2 + H2O', [1.51e+14, -1.0553, -760.929866016],
         options='duplicate')

#Reaction 29:
reaction('OH + H2O2  <=> HO2 + H2O', [2100.0, 2.9565, -1358.7675977],
         options='duplicate')
```

## D.3 Mechanism3 - m00000009.cti

Source Code D.3: m00000009.cti

```
#-------------------------------------------------------------------------------
#HDF5
# CTI Generated : 2017-12-02 01:01:14
#-------------------------------------------------------------------------------

units(length='cm', time='s', quantity='mol', act_energy='cal/mol')


ideal_gas(name='gas',
          elements="Ar C H He N O",
          species="""Ar He N2 C(GR) CO
                     CO2 H H2 O O2
                     OH H2O HO2 H2O2 O2(^1Delta)
                     O(1D) O3 OH*""",
          reactions='all',
          initial_state=state(temperature= 298.15, pressure= 100000.0))

#-------------------------------------------------------------------------------
#Species data
#-------------------------------------------------------------------------------

species(name='Ar',
        atoms='Ar:1',
        thermo=(NASA([300.0, 1000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, -745.375,
                      4.366001]),
                NASA([1000.0, 5000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, -745.375,
                      4.366001])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 136.5,
                diam = 3.33))

species(name='He',
        atoms='He:1',
        thermo=(NASA([300.0, 1000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, -745.375,
                      0.9153488]),
                NASA([1000.0, 5000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, -745.375,
                      0.9153488])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 10.2,
                diam = 2.576))

species(name='N2',
        atoms='N:2',
        thermo=(NASA([300.0, 1000.0],
                     [3.298677, 0.00140824, -3.96322e-06,
                      5.64152e-09, -2.44486e-12, -1020.9,
                      3.950372]),
                NASA([1000.0, 5000.0],
                     [2.92664, 0.001487977, -5.68476e-07,
                      1.0097e-10, -6.75335e-15, -922.7977,
                      5.980528])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 97.53,
                diam = 3.621,
                polar = 1.76,
                rot_relax = 4.0))

species(name='C(GR)',
        atoms='C:1',
        thermo=(NASA([200.0, 1000.0],
                     [-0.31087, 0.0044035, 1.9039e-06,
                      -6.3855e-09, 2.9896e-12, -108.6508,
                      1.1138]),
                NASA([1000.0, 5000.0],
                     [1.4557, 0.0017171, -6.9758e-07,
                      1.3528e-10, -9.6765e-15, -695.128,
                      -8.5257])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 71.4,
                diam = 3.298))

species(name='CO',
        atoms='C:1 O:1',
        thermo=(NASA([300.0, 1000.0],
                     [3.262452, 0.001511941, -3.88176e-06,
                      5.58194e-09, -2.47495e-12, -14310.54,
                      4.848897]),
                NASA([1000.0, 5000.0],
                     [3.025078, 0.001442689, -5.63083e-07,
                      1.01858e-10, -6.91095e-15, -14268.35,
                      6.108218])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 98.1,
                diam = 3.65,
                polar = 1.95,
                rot_relax = 1.8))

species(name='CO2',
        atoms='C:1 O:2',
        thermo=(NASA([300.0, 1000.0],
                     [2.275725, 0.009922072, -1.04091e-05,
                      6.86669e-09, -2.11728e-12, -48373.14,
                      10.18849]),
                NASA([1000.0, 5000.0],
                     [4.453623, 0.003140169, -1.27841e-06,
                      2.394e-10, -1.66903e-14, -48966.96,
                      -0.9553959])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 244.0,
                diam = 3.763,
                polar = 2.65,
                rot_relax = 2.1))

species(name='H',
        atoms='H:1',
        thermo=(NASA([300.0, 1000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, 25471.63,
                      -0.4601176]),
                NASA([1000.0, 5000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, 25471.63,
                      -0.4601176])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 145.0,
                diam = 2.05))

species(name='H2',
        atoms='H:2',
        thermo=(NASA([300.0, 1000.0],
                     [3.298124, 0.000824944, -8.14302e-07,
                      -9.47543e-11, 4.13487e-13, -1012.521,
                      -3.294094]),
                NASA([1000.0, 5000.0],
                     [2.991423, 0.000700064, -5.63383e-08,
                      -9.23158e-12, 1.58275e-15, -835.034,
                      -1.35511])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 38.0,
                diam = 2.92,
                polar = 0.79,
                rot_relax = 280.0))

species(name='O',
        atoms='O:1',
        thermo=(NASA([300.0, 1000.0],
                     [2.946429, -0.001638166, 2.42103e-06,
                      -1.60284e-09, 3.8907e-13, 29147.64,
                      2.963995]),
                NASA([1000.0, 5000.0],
                     [2.54206, -2.75506e-05, -3.1028e-09,
                      4.55107e-12, -4.36805e-16, 29230.8,
                      4.920308])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 80.0,
                diam = 2.75))

species(name='O2',
        atoms='O:2',
        thermo=(NASA([300.0, 1000.0],
                     [3.212936, 0.001127486, -5.75615e-07,
                      1.31388e-09, -8.76855e-13, -1005.249,
                      6.034738]),
                NASA([1000.0, 5000.0],
                     [3.697578, 0.00061352, -1.25884e-07,
                      1.77528e-11, -1.13644e-15, -1233.93,
                      3.189166])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 107.4,
                diam = 3.458,
                polar = 1.6,
                rot_relax = 3.8))

species(name='OH',
        atoms='H:1 O:1',
        thermo=(NASA([200.0, 1000.0],
                     [4.12530561, -0.003225449, 6.52765e-06,
                      -5.79854e-09, 2.06237e-12, 3346.30913,
                      -0.69043296]),
                NASA([1000.0, 6000.0],
                     [2.86472886, 0.001056504, -2.59083e-07,
                      3.05219e-11, -1.33196e-15, 3683.62875,
                      5.70164073])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 80.0,
                diam = 2.75))

species(name='H2O',
        atoms='H:2 O:1',
        thermo=(NASA([300.0, 1000.0],
                     [3.386842, 0.003474982, -6.3547e-06,
                      6.96858e-09, -2.50659e-12, -30208.11,
                      2.590233]),
                NASA([1000.0, 5000.0],
                     [2.672146, 0.003056293, -8.73026e-07,
                      1.201e-10, -6.39162e-15, -29899.21,
                      6.862817])),
        transport = gas_transport(
                geom = "nonlinear",
                well_depth = 572.4,
                diam = 2.605,
                dipole = 1.844,
                rot_relax = 4.0))

species(name='HO2',
        atoms='H:1 O:2',
        thermo=(NASA([200.0, 1000.0],
                     [4.30179801, -0.004749121, 2.11583e-05,
                       -2.42764e-08, 9.29225e-12, 294.80804,
                       3.71666245]),
                NASA([1000.0, 3500.0],
                      [4.0172109, 0.00223982, -6.33658e-07,
                       1.14246e-10, -1.07909e-14, 111.856713,
                       3.78510215])),
        transport = gas_transport(
                geom = "nonlinear",
                well_depth = 107.4,
                diam = 3.458,
                rot_relax = 1.0))

species(name='H2O2',
        atoms='H:2 O:2',
        thermo=(NASA([300.0, 1000.0],
                      [3.388754, 0.006569226, -1.48501e-07,
                       -4.62581e-09, 2.47152e-12, -17663.15,
                       6.785363]),
                NASA([1000.0, 5000.0],
                      [4.573167, 0.004336136, -1.47469e-06,
                       2.3489e-10, -1.43165e-14, -18006.96,
                       0.501137])),
        transport = gas_transport(
                geom = "nonlinear",
                well_depth = 107.4,
                diam = 3.458,
                rot_relax = 3.8))

species(name='O2(^1Delta)',
        atoms='O:2',
        thermo=(NASA([200.0, 1000.0],
                      [3.78535371, -0.0032192854, 1.12323443e-05,
                       -1.17254068e-08, 4.17659585e-12, 10292.2572,
                       3.27320239]),
                NASA([1000.0, 6000.0],
                      [3.45852381, 0.00104045351, -2.79664041e-07,
                       3.11439672e-11, -8.55656058e-16, 10222.9063,
                       4.15264119])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 107.4,
                diam = 3.458,
                polar = 1.6,
                rot_relax = 3.8))
```

```

species(name='O(1D)',
        atoms='O:1',
        thermo=(NASA([200.0, 1000.0],
                     [2.49993786, 1.71935346e-07, -3.45215267e-10,
                      3.71342028e-13, -1.70964494e-16, 51996.5317,
                      4.61684555]),
                NASA([1000.0, 6000.0],
                     [2.49368475, 1.37617903e-05, -1.00401058e-08,
                      2.76012182e-12, -2.01597513e-16, 51998.6304,
                      4.6505095])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 80.0,
                diam = 2.75))

species(name='O3',
        atoms='O:3',
        thermo=(NASA([200.0, 1000.0],
                     [3.4074, 0.0020538, 1.3849e-05,
                      -2.2331e-08, 9.7607e-12, 15864.4979,
                      8.2825]),
                NASA([1000.0, 6000.0],
                     [12.3303, -0.011932, 7.9874e-06,
                      -1.7719e-09, 1.2608e-13, 12675.5831,
                      -40.8823])),
        transport = gas_transport(
                geom = "nonlinear",
                well_depth = 180.0,
                diam = 4.1,
                rot_relax = 2.0))

species(name='OH*',
        atoms='H:1 O:1',
        thermo=(NASA([300.0, 1000.0],
                     [3.46084428, 0.000501872172, -2.00254474e-06,
                      3.18901984e-09, -1.35451838e-12, 50734.9466,
                      1.73976415]),
                NASA([1000.0, 6000.0],
                     [2.7558292, 0.00139848756, -4.19428493e-07,
                      6.33453282e-11, -3.56042218e-15, 50975.1756,
                      5.62581429])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 80.0,
                diam = 2.75))


#-------------------------------------------------------------------------------
#Reaction data
#-------------------------------------------------------------------------------

#Reaction 1:
reaction('O2 + H  <=> OH + O', [1.04e+14, 0.0, 15309.835])

#Reaction 2:
reaction('H2 + O  <=> OH + H', [8.7923244e+14, 0.0, 19174.55],
         options='duplicate')

#Reaction 3:
reaction('H2 + O  <=> OH + H', [3.81803676e+12, 0.0, 7948.0],
         options='duplicate')

#Reaction 4:
reaction('H2 + OH  <=> H2O + H', [216194826.0, 1.51, 3429.562])

#Reaction 5:
reaction('OH + OH  <=> H2O + O', [2.83e+13, -0.764, -460.560180575],
         options='duplicate')

#Reaction 6:
reaction('OH + OH  <=> H2O + O', [12600.0, 2.5308, -1622.91682141],
         options='duplicate')

#Reaction 7:
three_body_reaction('H + H + M <=> H2 + M', [3.143e+20, -1.806, 982.555066967],
         efficiencies='H2:2.5 H2O:12.0 CO:1.9 CO2:3.8 Ar:0.0 He:0.0 ')

#Reaction 8:
reaction('H + H + Ar <=> H2 + Ar', [4.011e+19, -1.506, 982.555066967])

#Reaction 9:
reaction('H + H + He <=> H2 + He', [4.011e+19, -1.506, 982.555066967])

#Reaction 10:
three_body_reaction('O + O + M <=> O2 + M', [6.16524893053e+15, -0.5, 0.0],
         efficiencies='H2:2.53 H2O:11.76 CO:1.88 CO2:3.82 Ar:0.0 He:0.0 ')

```

```
#Reaction 11:
reaction('O + O + Ar <=> O2 + Ar', [1.88584084934e+13, 0.0, -1788.3])

#Reaction 12:
reaction('O + O + He <=> O2 + He', [1.88584084934e+13, 0.0, -1788.3])

#Reaction 13:
three_body_reaction('H + O + M <=> OH + M', [4.71460212335e+18, -1.0, 0.0],
         efficiencies='H2:2.54 H2O:12.31 CO:1.92 CO2:3.77 He:0.75 Ar:0.75 ')

#Reaction 14:
three_body_reaction('OH + H + M <=> H2O + M', [1.483e+28, -3.798, 2706.52897774],
         efficiencies='H2:3.0 O2:1.5 N2:2.0 He:1.1 CO:1.9 CO2:3.8 H2O:0.0 ')

#Reaction 15:
reaction('OH + H + H2O <=> H2O + H2O', [2.46e+26, -2.916, 2096.18418054])

#Reaction 16:
falloff_reaction('O2 + H (+ M) <=> HO2 (+ M)',
                 kf=[4.65e+12, 0.44, 0.0],
                 kf0=[6.37e+20, -1.72, 524.568],
                 efficiencies='H2:1.99 O2:0.78 H2O:14.0 CO:1.9 CO2:3.8 Ar:0.67
                 ↪  He:0.8 ',
         falloff = Troe(A = 0.5, T3 = 1e-30, T1 = 1e+30, T2 = 0))

#Reaction 17:
reaction('O2 + H2  <=> HO2 + H', [1341000.0, 2.314, 53420.1782336])

#Reaction 18:
reaction('HO2 + H  <=> OH + OH', [7.08e+13, 0.0, 300.169572393])

#Reaction 19:
reaction('HO2 + O  <=> OH + O2', [28500000000.0, 1.0, -724.309178184])

#Reaction 20:
reaction('O2 + H2O  <=> HO2 + OH', [2.642e+20, -2.194, 70150.6296335],
         options='duplicate')

#Reaction 21:
reaction('O2 + H2O  <=> HO2 + OH', [1656000000.0, 1.533, 68259.5613274],
         options='duplicate')

#Reaction 22:
reaction('HO2 + HO2  <=> H2O2 + O2', [1510.0, 2.6969, -3866.702],
        options='duplicate')

#Reaction 23:
reaction('HO2 + HO2  <=> H2O2 + O2', [2.5e+15, -1.461, -1469.7839],
        options='duplicate')

#Reaction 24:
falloff_reaction('OH + OH (+ M) <=> H2O2 (+ M)',
                kf=[5.03e+12, 0.058, -634.68854952],
                kf0=[5.68e+25, -3.358, 576.325578995],
                efficiencies='CO2:2.99 N2:2.01 He:0.4 H2O:10.0 ',
        falloff = Troe(A = 0.55, T3 = 1e-30, T1 = 1e+30, T2 = 0))

#Reaction 25:
reaction('H2O2 + H  <=> OH + H2O', [2.408856e+13, 0.0, 3974.0])

#Reaction 26:
reaction('H2O2 + H  <=> H2 + HO2', [4.817712e+13, 0.0, 7948.0])

#Reaction 27:
reaction('O + H2O2  <=> HO2 + OH', [9635424.0, 2.0, 3974.0])

#Reaction 28:
reaction('OH + H2O2  <=> HO2 + H2O', [1.51e+14, -1.0553, -760.929866016],
        options='duplicate')

#Reaction 29:
reaction('OH + H2O2  <=> HO2 + H2O', [2100.0, 2.9565, -1358.7675977],
        options='duplicate')

#Reaction 30:
three_body_reaction('O3 + M <=> O2 + O + M', [4054000.0, 0.0, 24416.256],
        efficiencies='Ar:0.51 O2:0.95 O3:2.5 O:4.0 ')

#Reaction 31:
reaction('O3 + O  <=> O2 + O2', [4.817712e+12, 0.0, 4093.22])

#Reaction 32:
reaction('O3 + O  <=> O2(^1Delta) + O2', [144531360000.0, 0.0, 4093.22])

#Reaction 33:
three_body_reaction('O + O + M <=> O2(^1Delta) + M', [7e+15, -1.0, 0.0],
        efficiencies='O:28.86 O2:8.0 H2O:5.0 O3:8.0 N2:2.0 ')

```

```
#Reaction 34:
three_body_reaction('O2(^1Delta) + M <=> O2 + M', [1806642.0, 0.0, 397.4],
         efficiencies='Ar:0.01 He:0.01 CO2:0.01 H2O:3.3 H2:2.5 CO:5.67 O:43.33
         ↪  H:21833333.33 N2:0.0 ')

#Reaction 35:
three_body_reaction('O2(^1Delta) + O + M <=> O2 + O + M', [3.62661701796e+15,
↪  0.0, 0.0],
         efficiencies='Ar:0.63 ')

#Reaction 36:
reaction('O2(^1Delta) + O3  <=> O2 + O2 + O', [3.1315128e+13, 0.0, 5643.08])

#Reaction 37:
reaction('O2(^1Delta) + O(1D)  <=> O2 + O', [6.03e+12, 0.0, 0.0])

#Reaction 38:
reaction('O2 + O(1D)  <=> O2(^1Delta) + O', [1.9270848e+13, 0.0, -133.129])

#Reaction 39:
three_body_reaction('O(1D) + M <=> O + M', [481771200000.0, 0.0, 0.0],
         efficiencies='O2:5.83 O:10.0 H2O:3.0 N2:26.15 ')

#Reaction 40:
reaction('O(1D) + O3  <=> O2 + O + O', [7.226568e+13, 0.0, 0.0])

#Reaction 41:
reaction('O(1D) + O3  <=> O2 + O2', [7.226568e+13, 0.0, 0.0])

#Reaction 42:
reaction('H2 + O2(^1Delta)  <=> H + HO2', [616000.0, 2.335, 31097.5676999])

#Reaction 43:
reaction('O2(^1Delta) + H  <=> OH + O', [349886334.0, 1.45, 4510.49])

#Reaction 44:
three_body_reaction('H + O2(^1Delta) + M <=> HO2 + M', [9890000000.0, 2.03,
↪  3360.017],
         efficiencies='')

#Reaction 45:
reaction('HO2 + OH  <=> H2O + O2(^1Delta)', [109300.0, 1.707, 12542.0852998])

#Reaction 46:
```

```
reaction('OH + O2(^1Delta)  <=> O + HO2', [1.3e+13, 0.0, 34019.2182045])

#Reaction 47:
reaction('O3 + H  <=> OH + O2', [8.430996e+13, 0.0, 933.89])

#Reaction 48:
reaction('OH + O3  <=> HO2 + O2', [1.7e-12, 0.0, 1867.78])

#Reaction 49:
reaction('HO2 + O3  <=> OH + O2 + O2', [0.000584749794, 4.57, -1376.991])

#Reaction 50:
reaction('H + HO2  <=> H2O + O(1D)', [2.5e+12, 0.0, 300.169572393])

#Reaction 51:
reaction('O(1D) + H2  <=> OH + H', [8.129889e+13, 0.0, 0.0])

#Reaction 52:
reaction('O(1D) + H2O  <=> OH + OH', [1.0237638e+14, 0.0, -71.532])

#Reaction 53:
three_body_reaction('O + H + M <=> OH* + M', [1.5e+13, 0.0, 5974.52960908],
         efficiencies='O2:0.4 Ar:0.35 N2:0.4 H2O:6.5 ')

#Reaction 54:
three_body_reaction('OH* + M <=> OH + M', [21470000000.0, 0.5, 2061.1643971],
         efficiencies='O2:39.12 N2:5.03 H2O:137.87 H2:16.49 OH:69.86 H:69.86
         ↪  O:69.86 ')

#Reaction 55:
reaction('OH* + H2  <=> H2O + H', [2.6e+12, 0.5, -444.250967142])

#Reaction 56:
reaction('OH* + O2  <=> O3 + H', [252000000000.0, 0.5, -482.272446312])

#Reaction 57:
reaction('OH* + O2  <=> HO2 + O', [1.008e+12, 0.5, -482.272446312])

#Reaction 58:
reaction('OH* + H2O  <=> H2O2 + H', [2.96e+12, 0.5, -861.486672768])
```

## D.4 `DynamicMech151203 - m00000010.cti`

Source Code D.4: m00000010.cti

```
#-------------------------------------------------------------------------------
#HDF5
# CTI Generated : 2017-12-02 01:06:32
#-------------------------------------------------------------------------------

units(length='cm', time='s', quantity='mol', act_energy='cal/mol')


ideal_gas(name='gas',
          elements="Ar C H He N O",
          species="""Ar He N2 C(GR) CO
                     CO2 H H2 O O2
                     OH H2O HO2 H2O2 O2(^1Delta)
                     O(1D) O3 OH*""",
          reactions='all',
          initial_state=state(temperature= 298.15, pressure= 100000.0))

#-------------------------------------------------------------------------------
#Species data
#-------------------------------------------------------------------------------

species(name='Ar',
        atoms='Ar:1',
        thermo=(NASA([300.0, 1000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, -745.375,
                      4.366001]),
                NASA([1000.0, 5000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, -745.375,
                      4.366001])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 136.5,
                diam = 3.33))

species(name='He',
        atoms='He:1',
        thermo=(NASA([300.0, 1000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, -745.375,
                      0.9153488]),
                NASA([1000.0, 5000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, -745.375,
                      0.9153488])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 10.2,
                diam = 2.576))

species(name='N2',
        atoms='N:2',
        thermo=(NASA([300.0, 1000.0],
                     [3.298677, 0.00140824, -3.96322e-06,
                      5.64152e-09, -2.44486e-12, -1020.9,
                      3.950372]),
                NASA([1000.0, 5000.0],
                     [2.92664, 0.001487977, -5.68476e-07,
                      1.0097e-10, -6.75335e-15, -922.7977,
                      5.980528])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 97.53,
                diam = 3.621,
                polar = 1.76,
                rot_relax = 4.0))

species(name='C(GR)',
        atoms='C:1',
        thermo=(NASA([200.0, 1000.0],
                     [-0.31087, 0.0044035, 1.9039e-06,
                      -6.3855e-09, 2.9896e-12, -108.6508,
                      1.1138]),
                NASA([1000.0, 5000.0],
                     [1.4557, 0.0017171, -6.9758e-07,
                      1.3528e-10, -9.6765e-15, -695.128,
                      -8.5257])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 71.4,
                diam = 3.298))

species(name='CO',
        atoms='C:1 O:1',
        thermo=(NASA([300.0, 1000.0],
                     [3.262452, 0.001511941, -3.88176e-06,
                      5.58194e-09, -2.47495e-12, -14310.54,
                      4.848897]),
                NASA([1000.0, 5000.0],
                     [3.025078, 0.001442689, -5.63083e-07,
                      1.01858e-10, -6.91095e-15, -14268.35,
                      6.108218])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 98.1,
                diam = 3.65,
                polar = 1.95,
                rot_relax = 1.8))

species(name='CO2',
        atoms='C:1 O:2',
        thermo=(NASA([300.0, 1000.0],
                     [2.275725, 0.009922072, -1.04091e-05,
                      6.86669e-09, -2.11728e-12, -48373.14,
                      10.18849]),
                NASA([1000.0, 5000.0],
                     [4.453623, 0.003140169, -1.27841e-06,
                      2.394e-10, -1.66903e-14, -48966.96,
                      -0.9553959])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 244.0,
                diam = 3.763,
                polar = 2.65,
                rot_relax = 2.1))

species(name='H',
        atoms='H:1',
        thermo=(NASA([300.0, 1000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, 25471.63,
                      -0.4601176]),
                NASA([1000.0, 5000.0],
                     [2.5, 0.0, 0.0,
                      0.0, 0.0, 25471.63,
                      -0.4601176])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 145.0,
                diam = 2.05))

species(name='H2',
        atoms='H:2',
        thermo=(NASA([300.0, 1000.0],
                     [3.298124, 0.000824944, -8.14302e-07,
                      -9.47543e-11, 4.13487e-13, -1012.521,
                      -3.294094]),
                NASA([1000.0, 5000.0],
                     [2.991423, 0.000700064, -5.63383e-08,
                      -9.23158e-12, 1.58275e-15, -835.034,
                      -1.35511])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 38.0,
                diam = 2.92,
                polar = 0.79,
                rot_relax = 280.0))

species(name='O',
        atoms='O:1',
        thermo=(NASA([300.0, 1000.0],
                     [2.946429, -0.001638166, 2.42103e-06,
                      -1.60284e-09, 3.8907e-13, 29147.64,
                      2.963995]),
                NASA([1000.0, 5000.0],
                     [2.54206, -2.75506e-05, -3.1028e-09,
                      4.55107e-12, -4.36805e-16, 29230.8,
                      4.920308])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 80.0,
                diam = 2.75))

species(name='O2',
        atoms='O:2',
        thermo=(NASA([300.0, 1000.0],
                     [3.212936, 0.001127486, -5.75615e-07,
                      1.31388e-09, -8.76855e-13, -1005.249,
                      6.034738]),
                NASA([1000.0, 5000.0],
                     [3.697578, 0.00061352, -1.25884e-07,
                      1.77528e-11, -1.13644e-15, -1233.93,
                     3.189166])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 107.4,
                diam = 3.458,
                polar = 1.6,
                rot_relax = 3.8))

species(name='OH',
        atoms='H:1 O:1',
        thermo=(NASA([200.0, 1000.0],
                     [4.12530561, -0.003225449, 6.52765e-06,
                      -5.79854e-09, 2.06237e-12, 3346.30913,
                      -0.69043296]),
                NASA([1000.0, 6000.0],
                     [2.86472886, 0.001056504, -2.59083e-07,
                      3.05219e-11, -1.33196e-15, 3683.62875,
                      5.70164073])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 80.0,
                diam = 2.75))

species(name='H2O',
        atoms='H:2 O:1',
        thermo=(NASA([300.0, 1000.0],
                     [3.386842, 0.003474982, -6.3547e-06,
                      6.96858e-09, -2.50659e-12, -30208.11,
                      2.590233]),
                NASA([1000.0, 5000.0],
                     [2.672146, 0.003056293, -8.73026e-07,
                      1.201e-10, -6.39162e-15, -29899.21,
                      6.862817])),
        transport = gas_transport(
                geom = "nonlinear",
                well_depth = 572.4,
                diam = 2.605,
                dipole = 1.844,
                rot_relax = 4.0))

species(name='HO2',
        atoms='H:1 O:2',
        thermo=(NASA([200.0, 1000.0],
                     [4.30179801, -0.004749121, 2.11583e-05,
                   -2.42764e-08, 9.29225e-12, 294.80804,
                   3.71666245]),
             NASA([1000.0, 3500.0],
                  [4.0172109, 0.00223982, -6.33658e-07,
                   1.14246e-10, -1.07909e-14, 111.856713,
                   3.78510215])),
        transport = gas_transport(
                geom = "nonlinear",
                well_depth = 107.4,
                diam = 3.458,
                rot_relax = 1.0))

species(name='H2O2',
        atoms='H:2 O:2',
        thermo=(NASA([300.0, 1000.0],
                  [3.388754, 0.006569226, -1.48501e-07,
                   -4.62581e-09, 2.47152e-12, -17663.15,
                   6.785363]),
             NASA([1000.0, 5000.0],
                  [4.573167, 0.004336136, -1.47469e-06,
                   2.3489e-10, -1.43165e-14, -18006.96,
                   0.501137])),
        transport = gas_transport(
                geom = "nonlinear",
                well_depth = 107.4,
                diam = 3.458,
                rot_relax = 3.8))

species(name='O2(^1Delta)',
        atoms='O:2',
        thermo=(NASA([200.0, 1000.0],
                  [3.78535371, -0.0032192854, 1.12323443e-05,
                   -1.17254068e-08, 4.17659585e-12, 10292.2572,
                   3.27320239]),
             NASA([1000.0, 6000.0],
                  [3.45852381, 0.00104045351, -2.79664041e-07,
                   3.11439672e-11, -8.55656058e-16, 10222.9063,
                   4.15264119])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 107.4,
                diam = 3.458,
                polar = 1.6,
                rot_relax = 3.8))
```

```

species(name='O(1D)',
        atoms='O:1',
        thermo=(NASA([200.0, 1000.0],
                     [2.49993786, 1.71935346e-07, -3.45215267e-10,
                      3.71342028e-13, -1.70964494e-16, 51996.5317,
                      4.61684555]),
                NASA([1000.0, 6000.0],
                     [2.49368475, 1.37617903e-05, -1.00401058e-08,
                      2.76012182e-12, -2.01597513e-16, 51998.6304,
                      4.6505095])),
        transport = gas_transport(
                geom = "atom",
                well_depth = 80.0,
                diam = 2.75))

species(name='O3',
        atoms='O:3',
        thermo=(NASA([200.0, 1000.0],
                     [3.4074, 0.0020538, 1.3849e-05,
                      -2.2331e-08, 9.7607e-12, 15864.4979,
                      8.2825]),
                NASA([1000.0, 6000.0],
                     [12.3303, -0.011932, 7.9874e-06,
                      -1.7719e-09, 1.2608e-13, 12675.5831,
                      -40.8823])),
        transport = gas_transport(
                geom = "nonlinear",
                well_depth = 180.0,
                diam = 4.1,
                rot_relax = 2.0))

species(name='OH*',
        atoms='H:1 O:1',
        thermo=(NASA([300.0, 1000.0],
                     [3.46084428, 0.000501872172, -2.00254474e-06,
                      3.18901984e-09, -1.35451838e-12, 50734.9466,
                      1.73976415]),
                NASA([1000.0, 6000.0],
                     [2.7558292, 0.00139848756, -4.19428493e-07,
                      6.33453282e-11, -3.56042218e-15, 50975.1756,
                      5.62581429])),
        transport = gas_transport(
                geom = "linear",
                well_depth = 80.0,
                diam = 2.75))


#-------------------------------------------------------------------------------
#Reaction data
#-------------------------------------------------------------------------------

#Reaction 1:
reaction('O2 + H  <=> OH + O', [1.0426e+14, 0.0, 15309.835])

#Reaction 2:
reaction('H2 + O  <=> OH + H', [1.1163843132e+15, 0.0, 19174.55],
         options='duplicate')

#Reaction 3:
reaction('H2 + O  <=> OH + H', [4.8478829214e+12, 0.0, 7948.0],
         options='duplicate')

#Reaction 4:
reaction('H2 + OH  <=> H2O + H', [189661277.16, 1.51, 3429.562])

#Reaction 5:
reaction('OH + OH  <=> H2O + O', [2.83e+13, -0.764, -460.560180575],
         options='duplicate')

#Reaction 6:
reaction('OH + OH  <=> H2O + O', [12600.0, 2.5308, -1622.91682141],
         options='duplicate')

#Reaction 7:
three_body_reaction('H + H + M <=> H2 + M', [3.143e+20, -1.806, 982.555066967],
         efficiencies='H2:2.5 H2O:12.0 CO:1.9 CO2:3.8 Ar:0.0 He:0.0 ')

#Reaction 8:
reaction('H + H + Ar <=> H2 + Ar', [4.011e+19, -1.506, 982.555066967])

#Reaction 9:
reaction('H + H + He <=> H2 + He', [4.011e+19, -1.506, 982.555066967])

#Reaction 10:
three_body_reaction('O + O + M <=> O2 + M', [3.56641517546e+15, -0.5, 0.0],
         efficiencies='H2:2.53 H2O:11.76 CO:1.88 CO2:3.82 Ar:0.0 He:0.0 ')

```

```
#Reaction 11:
reaction('O + O + Ar <=> O2 + Ar', [1.090886399e+13, 0.0, -1788.3])

#Reaction 12:
reaction('O + O + He <=> O2 + He', [1.090886399e+13, 0.0, -1788.3])

#Reaction 13:
three_body_reaction('H + O + M <=> OH + M', [1.56619082538e+18, -1.0, 0.0],
         efficiencies='H2:2.54 H2O:12.31 CO:1.92 CO2:3.77 He:0.75 Ar:0.75 ')

#Reaction 14:
three_body_reaction('OH + H + M <=> H2O + M', [1.483e+28, -3.798, 2706.52897774],
         efficiencies='H2:3.0 O2:1.5 N2:2.0 He:1.1 CO:1.9 CO2:3.8 H2O:0.0 ')

#Reaction 15:
reaction('OH + H + H2O <=> H2O + H2O', [2.46e+26, -2.916, 2096.18418054])

#Reaction 16:
falloff_reaction('O2 + H (+ M) <=> HO2 (+ M)',
                 kf=[3.6934e+12, 0.44, 0.0],
                 kf0=[5.0595e+20, -1.72, 524.568],
                 efficiencies='H2:1.99 O2:0.78 H2O:14.0 CO:1.9 CO2:3.8 Ar:0.67
                 ↪  He:0.8 ',
         falloff = Troe(A = 0.5, T3 = 1e-30, T1 = 1e+30, T2 = 0))

#Reaction 17:
reaction('O2 + H2  <=> HO2 + H', [823890.0, 2.314, 53420.1782336])

#Reaction 18:
reaction('HO2 + H  <=> OH + OH', [1.1333e+14, 0.0, 300.169572393])

#Reaction 19:
reaction('HO2 + O  <=> OH + O2', [24613000000.0, 1.0, -724.309178184])

#Reaction 20:
reaction('O2 + H2O  <=> HO2 + OH', [2.642e+20, -2.194, 70150.6296335],
         options='duplicate')

#Reaction 21:
reaction('O2 + H2O  <=> HO2 + OH', [1656000000.0, 1.533, 68259.5613274],
         options='duplicate')

#Reaction 22:
reaction('HO2 + HO2  <=> H2O2 + O2', [1510.0, 2.6969, -3866.702],
         options='duplicate')

#Reaction 23:
reaction('HO2 + HO2  <=> H2O2 + O2', [2.5e+15, -1.461, -1469.7839],
         options='duplicate')

#Reaction 24:
falloff_reaction('OH + OH (+ M) <=> H2O2 (+ M)',
                 kf=[5.03e+12, 0.058, -634.68854952],
                 kf0=[5.68e+25, -3.358, 576.325578995],
                 efficiencies='CO2:2.99 N2:2.01 He:0.4 H2O:10.0 ',
         falloff = Troe(A = 0.55, T3 = 1e-30, T1 = 1e+30, T2 = 0))

#Reaction 25:
reaction('H2O2 + H  <=> OH + H2O', [3.9240866454e+13, 0.0, 3974.0])

#Reaction 26:
reaction('H2O2 + H  <=> H2 + HO2', [9.801635064e+13, 0.0, 7948.0])

#Reaction 27:
reaction('O + H2O2  <=> HO2 + OH', [5728380.0108, 2.0, 3974.0])

#Reaction 28:
reaction('OH + H2O2  <=> HO2 + H2O', [1.51e+14, -1.0553, -760.929866016],
         options='duplicate')

#Reaction 29:
reaction('OH + H2O2  <=> HO2 + H2O', [2100.0, 2.9565, -1358.7675977],
         options='duplicate')

#Reaction 30:
three_body_reaction('O3 + M <=> O2 + O + M', [3332100.0, 0.0, 24416.256],
         efficiencies='Ar:0.51 O2:0.95 O3:2.5 O:4.0 ')

#Reaction 31:
reaction('O3 + O  <=> O2 + O2', [5.5400074716e+12, 0.0, 4093.22])

#Reaction 32:
reaction('O3 + O  <=> O2(^1Delta) + O2', [433594080000.0, 0.0, 4093.22])

#Reaction 33:
three_body_reaction('O + O + M <=> O2(^1Delta) + M', [2.331e+15, -1.0, 0.0],
         efficiencies='O:28.86 O2:8.0 H2O:5.0 O3:8.0 N2:2.0 ')

```

```
#Reaction 34:
three_body_reaction('O2(^1Delta) + M <=> O2 + M', [1400809.9854, 0.0, 397.4],
          efficiencies='Ar:0.0 He:0.0 CO2:0.01 H2O:3.3 H2:2.5 CO:5.67 O:43.33
          ↪  H:21830531.79 N2:0.0 ')

#Reaction 35:
three_body_reaction('O2(^1Delta) + O + M <=> O2 + O + M', [3.62661701796e+15,
↪  0.0, 0.0],
          efficiencies='Ar:0.63 ')

#Reaction 36:
reaction('O2(^1Delta) + O3  <=> O2 + O2 + O', [3.1315128e+13, 0.0, 5643.08])

#Reaction 37:
reaction('O2(^1Delta) + O(1D)  <=> O2 + O', [6.03e+13, 0.0, 0.0])

#Reaction 38:
reaction('O2 + O(1D)  <=> O2(^1Delta) + O', [1.9270848e+13, 0.0, -133.129])

#Reaction 39:
three_body_reaction('O(1D) + M <=> O + M', [96354240000.0, 0.0, 0.0],
          efficiencies='O2:5.83 O:10.0 H2O:3.0 N2:26.15 ')

#Reaction 40:
reaction('O(1D) + O3  <=> O2 + O + O', [7.226568e+13, 0.0, 0.0])

#Reaction 41:
reaction('O(1D) + O3  <=> O2 + O2', [7.226568e+13, 0.0, 0.0])

#Reaction 42:
reaction('H2 + O2(^1Delta)  <=> H + HO2', [616000.0, 2.335, 31097.5676999])

#Reaction 43:
reaction('O2(^1Delta) + H  <=> OH + O', [146277780.6, 1.45, 4510.49])

#Reaction 44:
three_body_reaction('H + O2(^1Delta) + M <=> HO2 + M', [95016000000.0, 2.03,
↪  3360.017],
          efficiencies='')

#Reaction 45:
reaction('HO2 + OH  <=> H2O + O2(^1Delta)', [327900.0, 1.707, 12542.0852998])

#Reaction 46:
```

```
reaction('OH + O2(^1Delta)  <=> O + HO2', [5.8552e+12, 0.0, 34019.2182045])

#Reaction 47:
reaction('O3 + H  <=> OH + O2', [8.430996e+13, 0.0, 933.89])

#Reaction 48:
reaction('OH + O3  <=> HO2 + O2', [1.7e-12, 0.0, 1867.78])

#Reaction 49:
reaction('HO2 + O3  <=> OH + O2 + O2', [0.00036547163232, 4.57, -1376.991])

#Reaction 50:
reaction('H + HO2  <=> H2O + O(1D)', [1.9532e+12, 0.0, 300.169572393])

#Reaction 51:
reaction('O(1D) + H2  <=> OH + H', [8.129889e+13, 0.0, 0.0])

#Reaction 52:
reaction('O(1D) + H2O  <=> OH + OH', [1.0237638e+14, 0.0, -71.532])

#Reaction 53:
three_body_reaction('O + H + M <=> OH* + M', [1.2566e+13, 0.0, 5974.52960908],
         efficiencies='O2:0.4 Ar:0.35 N2:0.4 H2O:6.5 ')

#Reaction 54:
three_body_reaction('OH* + M <=> OH + M', [21470000000.0, 0.5, 2061.1643971],
         efficiencies='O2:39.12 N2:5.03 H2O:137.87 H2:16.49 OH:69.86 H:69.86
         ↪  O:69.86 ')

#Reaction 55:
reaction('OH* + H2  <=> H2O + H', [2.6e+12, 0.5, -444.250967142])

#Reaction 56:
reaction('OH* + O2  <=> O3 + H', [361410000000.0, 0.5, -482.272446312])

#Reaction 57:
reaction('OH* + O2  <=> HO2 + O', [672310000000.0, 0.5, -482.272446312])

#Reaction 58:
reaction('OH* + H2O  <=> H2O2 + H', [6.5803e+12, 0.5, -861.486672768])
```

# Appendix E

# $H_2/O_2$ QOIs and simulation results

## $H_2/O_2$ QOIs

**Table E.1** - Experimental shock-tube ignition-delay QOIs.

**Table E.2** - Experimental flame-speed QOIs.

## Ignition-delay simulations

**Table E.3** Comparison of ignition-delay results for Mechanism1, Mechanism2, Mechanism3, and DynamicMech151203.

## Flame simulations

**Table E.4** Comparison of flame-speed results (mixture-averaged transport) with CloudFlame-Cantera and CHEMKIN-PRO for Mechanism1.

**Table E.5** Comparison of the four reaction mechanisms prediction of the flame-speed QOIs vs. experimental data.

## E.1 $H_2/O_2$ QOIs

Table E.1: Experimental shock-tube ignition-delay QOIs.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | 2007 | 0.0426 | 0.0201 | 0.0903 | a00000451* | |
| 0.17 | 0.32 | 1768 | 0.0756 | 0.0365 | 0.1565 | a00000452* | [167] |
| | | 1580 | 0.1341 | 0.0637 | 0.2825 | a00000453* | |
| | | 2224 | 0.0662 | 0.0460 | 0.0951 | a00000448* | |
| 0.50 | 0.34 | 1969 | 0.1063 | 0.0750 | 0.1507 | a00000449* | [167] |
| | | 1767 | 0.1709 | 0.1184 | 0.2466 | a00000450* | |
| | | 2090 | 0.0461 | 0.0276 | 0.0768 | a00000445* | |
| 1.50 | 0.33 | 1838 | 0.0829 | 0.0511 | 0.1346 | a00000446* | [167] |
| | | 1640 | 0.1493 | 0.0912 | 0.2444 | a00000447* | |
| | | 1412 | 0.1392 | 0.0840 | 0.2306 | a00000382* | |
| 0.39 | 2.18 | 1318 | 0.2096 | 0.1280 | 0.3433 | a00000383* | [168] |
| | | 1237 | 0.3157 | 0.1916 | 0.5201 | a00000384* | |
| | | 1311 | 0.2494 | 0.1131 | 0.5500 | a00000385 | |
| 0.75 | 2.10 | 1266 | 0.3113 | 0.1423 | 0.6808 | a00000386* | [168] |
| | | 1223 | 0.3885 | 0.1740 | 0.8676 | a00000387* | |
| | | 1299 | 0.3125 | 0.1338 | 0.7296 | a00000454* | |
| 1.00 | 2.04 | 1257 | 0.3659 | 0.1626 | 0.8234 | a00000455* | [168] |
| | | 1218 | 0.4284 | 0.1916 | 0.9579 | a00000456* | |
| | | 1454 | 0.0273 | 0.0192 | 0.0390 | a00000370* | |
| 1.00 | 1.94 | 1269 | 0.0616 | 0.0438 | 0.0866 | a00000371* | [169] |
| | | 1126 | 0.1386 | 0.0988 | 0.1946 | a00000372* | |
| | | 1352 | 0.0498 | 0.0308 | 0.0806 | a00000373 | |
| 1.00 | 0.71 | 1180 | 0.0992 | 0.0621 | 0.1585 | a00000374 | [170] |
| | | 1047 | 0.1977 | 0.1234 | 0.3168 | a00000375 | |
| | | 1161 | 0.2021 | 0.0880 | 0.4637 | a00000460* | |
| 1.00 | 1.45 | 1027 | 0.4783 | 0.2087 | 1.0961 | a00000461* | [171] |
| | | 921 | 1.1324 | 0.4753 | 2.6977 | a00000462* | |
| | | 1023 | 0.2167 | 0.0997 | 0.4711 | a00000463* | |
| 1.00 | 2.35 | 974 | 0.4857 | 0.2323 | 1.0156 | a00000464 | [171] |
| | | 930 | 1.0888 | 0.5212 | 2.2747 | a00000465 | |
| | | 999 | 1.0014 | 0.7424 | 1.3508 | a00000388 | |
| 0.50 | 1.05 | 972 | 1.4996 | 1.1135 | 2.0195 | a00000389 | [81] |
| | | 947 | 2.2456 | 1.6521 | 3.0525 | a00000390 | |
| | | 1015 | 0.7533 | 0.4799 | 1.1826 | a00000391 | |
| 0.50 | 3.99 | 995 | 1.4457 | 0.9244 | 2.2609 | a00000392* | [81] |
| | | 976 | 2.7743 | 1.7571 | 4.3804 | a00000393* | |
| | | 1093 | 0.8912 | 0.7036 | 1.1288 | a00000394 | |
| 0.50 | 15.86 | 1067 | 1.5758 | 1.2490 | 1.9882 | a00000395 | [81] |

| **Equiv ratio $\Phi$** | **P [atm]** | **T [K]** | **$\tau_{\mathrm{ign}}$ [ms]** | **Uncertainty Bounds of $\tau_{\mathrm{ign}}$** | | **PrIMe ID** | **Ref** |
|---|---|---|---|---|---|---|---|
| | | 1042 | 2.7864 | 2.1811 | 3.5596 | a00000396 | |
| | | 1411 | 0.1112 | 0.0760 | 0.1628 | a00000415* | |
| 1.00 | 1.01 | 1197 | 0.2988 | 0.2062 | 0.4329 | a00000416 | [81] |
| | | 1039 | 0.8025 | 0.5445 | 1.1829 | a00000417 | |
| | | 1103 | 0.1333 | 0.0563 | 0.3153 | a00000418 | |
| 1.00 | 3.96 | 1052 | 0.3919 | 0.1708 | 0.8992 | a00000419 | [81] |
| | | 1005 | 1.1528 | 0.4936 | 2.6926 | a00000420 | |
| | | 1174 | 0.0505 | 0.0359 | 0.0711 | a00000421 | |
| 1.00 | 16.39 | 1115 | 0.2506 | 0.1786 | 0.3517 | a00000422 | [81] |
| | | 1063 | 1.2436 | 0.8793 | 1.7590 | a00000423* | |
| | | 1221 | 0.0739 | 0.0204 | 0.2675 | a00000439* | |
| 0.10 | 3.69 | 1108 | 0.3037 | 0.0866 | 1.0650 | a00000440 | [172] |
| | | 1014 | 1.2486 | 0.3554 | 4.3863 | a00000441* | |
| | | 1372 | 0.0410 | 0.0300 | 0.0560 | a00000427 | |
| 0.10 | 1.01 | 1162 | 0.1455 | 0.1085 | 0.1950 | a00000428 | [172] |
| | | 1007 | 0.5158 | 0.3859 | 0.6894 | a00000429 | |
| | | 1074 | 0.1147 | 0.0471 | 0.2792 | a00000430 | |
| 0.10 | 4.17 | 1022 | 0.4968 | 0.2112 | 1.1687 | a00000431* | [172] |
| | | 975 | 2.1530 | 0.9012 | 5.1438 | a00000432* | |
| | | 1113 | 0.3755 | 0.1473 | 0.9572 | a00000433* | |
| 0.10 | 17.78 | 1055 | 1.0493 | 0.4138 | 2.6609 | a00000434 | [172] |
| | | 1003 | 2.9324 | 1.1114 | 7.7368 | a00000435 | |
| | | 1608 | 0.0810 | 0.0279 | 0.2350 | a00000436 | |
| 0.10 | 0.94 | 1299 | 0.2626 | 0.0930 | 0.7416 | a00000437* | [172] |
| | | 1090 | 0.8520 | 0.3000 | 2.4195 | a00000438 | |
| | | 1192 | 0.1241 | 0.0579 | 0.2660 | a00000442 | |
| 0.10 | 16.31 | 1136 | 0.5109 | 0.2456 | 1.0629 | a00000443* | [172] |
| | | 1084 | 2.1028 | 1.0091 | 4.3820 | a00000444* | |
| | | 1534 | 0.0811 | 0.0509 | 0.1292 | a00000397 | |
| 0.50 | 0.99 | 1262 | 0.2668 | 0.1709 | 0.4167 | a00000398 | [172] |
| | | 1072 | 0.8779 | 0.5654 | 1.3630 | a00000399 | |
| | | 1183 | 0.0939 | 0.0417 | 0.2116 | a00000400 | |
| 0.50 | 4.08 | 1118 | 0.3172 | 0.1446 | 0.6960 | a00000401* | [172] |
| | | 1059 | 1.0718 | 0.4861 | 2.3631 | a00000402* | |
| | | 1192 | 0.1853 | 0.1101 | 0.3117 | a00000403 | |
| 0.50 | 15.57 | 1144 | 0.5729 | 0.3427 | 0.9576 | a00000404 | [172] |
| | | 1101 | 1.7710 | 1.0402 | 3.0153 | a00000405 | |
| | | 1623 | 0.0849 | 0.0280 | 0.2577 | a00000406* | |
| 3.99 | 1.06 | 1308 | 0.2522 | 0.0883 | 0.7208 | a00000407 | [172] |

| **Equiv ratio $\Phi$** | **P [atm]** | **T [K]** | $\tau_{\mathbf{ign}}$ **[ms]** | **Uncertainty Bounds of $\tau_{\mathbf{ign}}$** | | **PrIMe ID** | **Ref** |
|---|---|---|---|---|---|---|---|
| | | 1096 | 0.7491 | 0.2651 | 2.1165 | a00000408 | |
| | | 1297 | 0.0565 | 0.0136 | 0.2351 | a00000409 | |
| 3.99 | 3.98 | 1164 | 0.1954 | 0.0483 | 0.7897 | a00000410 | [172] |
| | | 1057 | 0.6756 | 0.1623 | 2.8127 | a00000411* | |
| | | 1143 | 0.1697 | 0.0979 | 0.2941 | a00000412 | |
| 3.99 | 14.87 | 1069 | 0.7237 | 0.4231 | 1.2381 | a00000413 | [172] |
| | | 1004 | 3.0870 | 1.7975 | 5.3016 | a00000414 | |
| | | 1009 | 0.2344 | 0.1048 | 0.5244 | a00000457 | |
| 0.42 | 3.28 | 972 | 0.8252 | 0.3762 | 1.8102 | a00000458* | [100] |
| | | 938 | 2.9051 | 1.2947 | 6.5184 | a00000459* | |
| | | 1062 | 0.3217 | 0.1535 | 0.6742 | a00000376 | |
| 1.00 | 3.56 | 1012 | 1.0194 | 0.4931 | 2.1072 | a00000377* | [100] |
| | | 966 | 3.2307 | 1.5479 | 6.7428 | a00000378* | |
| | | 1296 | 0.2278 | 0.1284 | 0.4041 | a00000466 | |
| 1.00 | 1.00 | 1184 | 0.3902 | 0.2287 | 0.6657 | a00000467 | [173] |
| | | 1089 | 0.6685 | 0.3927 | 1.1379 | a00000468 | |
| | | 1564 | 0.1707 | 0.1630 | 0.1787 | a00000472 | |
| 1.03 | 1.00 | 1411 | 0.3018 | 0.2893 | 0.3149 | a00000473 | [173] |
| | | 1286 | 0.5337 | 0.5118 | 0.5566 | a00000474* | |
| | | 1386 | 0.1634 | 0.1407 | 0.1898 | a00000469* | |
| 1.47 | 1.00 | 1280 | 0.2403 | 0.2089 | 0.2763 | a00000470* | [173] |
| | | 1190 | 0.3532 | 0.3071 | 0.4062 | a00000471* | |
| | | 1270 | 0.0263 | 0.0230 | 0.0300 | a00000361 | |
| 1.00 | 33.00 | 1242 | 0.0592 | 0.0519 | 0.0676 | a00000362 | [174] |
| | | 1215 | 0.1337 | 0.1168 | 0.1529 | a00000363 | |
| | | 1714 | 0.0235 | 0.0176 | 0.0315 | a00000364 | |
| 1.00 | 64.00 | 1578 | 0.0423 | 0.0320 | 0.0558 | a00000365 | [174] |
| | | 1461 | 0.0760 | 0.0569 | 0.1013 | a00000366 | |
| | | 1327 | 0.1031 | 0.0866 | 0.1226 | a00000367 | |
| 1.00 | 64.00 | 1311 | 0.1438 | 0.1220 | 0.1695 | a00000368* | [174] |
| | | 1295 | 0.2007 | 0.1695 | 0.2376 | a00000369* | |
| | | 1566 | 0.0668 | 0.0283 | 0.1576 | a00000424 | |
| 0.25 | 1.32 | 1365 | 0.1137 | 0.0502 | 0.2575 | a00000425 | [175] |
| | | 1210 | 0.1933 | 0.0856 | 0.4366 | a00000426* | |
| | | 1045 | 0.5000 | 0.3152 | 0.7931 | a00000379 | |
| 2.00 | 5.00 | 1016 | 1.3800 | 0.8768 | 2.1720 | a00000380 | [176] |
| | | 990 | 3.8100 | 2.3867 | 6.0823 | a00000381 | |

* Indicates target was not included in final, reduced dataset.

Table E.2: Experimental flame-speed QOIs.

| $\Phi$ | **Pressure [atm]** | **Flame speed [cm/s]** | **PrIMe ID** | **Ref** |
|---|---|---|---|---|
| 0.6 | 1 | 81.88 | a00000476 | [177] |
| 0.85 | 10 | 45.3 | a00000483 | |
| 0.85 | 15 | 26.9 | a00000482 | |
| 0.85 | 20 | 18.35 | a00000481 | |
| 1.5 | 20 | 80.9 | a00000479 | |
| 1.65 | 1 | 280.21 | a00000477 | |
| 2 | 20 | 64.05 | a00000480 | |
| 4 | 1 | 165.29 | a00000478 | |
| 2.5 | 15 | 28.7 | a00000484 | [98] |
| 2.5 | 25 | 27.7 | a00000485 | |

## E.2 Ignition-delay Simulations

Table E.3: Comparison of ignition-delay results for Mechanism1, Mechanism2, Mechanism3, and DynamicMech151203.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 2007 | 0.0426 | 0.2110 | 0.1787 | 0.2109 | 0.1668 | a00000451* | |
| 0.17 | 0.32 | 1768 | 0.0756 | 0.2906 | 0.2437 | 0.2486 | 0.2291 | a00000452* | [167] |
| | | 1580 | 0.1341 | 0.3530 | 0.3551 | 0.3526 | 0.3429 | a00000453* | |
| | | 2224 | 0.0662 | 0.1084 | 0.1086 | 0.1079 | 0.1089 | a00000448* | |
| 0.50 | 0.34 | 1969 | 0.1063 | 0.1859 | 0.1861 | 0.1871 | 0.1923 | a00000449* | [167] |
| | | 1767 | 0.1709 | 0.3106 | 0.3104 | 0.3084 | 0.3135 | a00000450* | |
| | | 2090 | 0.0461 | 0.1165 | 0.1158 | 0.1166 | 0.1201 | a00000445* | |
| 1.50 | 0.33 | 1838 | 0.0829 | 0.2030 | 0.2032 | 0.2018 | 0.2091 | a00000446* | [167] |
| | | 1640 | 0.1493 | 0.3469 | 0.3480 | 0.3454 | 0.3531 | a00000447* | |
| | | 1412 | 0.1392 | 0.3359 | 0.3327 | 0.3367 | 0.3274 | a00000382* | |
| 0.39 | 2.18 | 1318 | 0.2096 | 0.4930 | 0.4887 | 0.4865 | 0.4750 | a00000383* | [168] |
| | | 1237 | 0.3157 | 0.7197 | 0.7228 | 0.7179 | 0.6932 | a00000384* | |
| | | 1311 | 0.2494 | 0.6039 | 0.6004 | 0.6039 | 0.5884 | a00000385 | |
| 0.75 | 2.10 | 1266 | 0.3113 | 0.7451 | 0.7488 | 0.7479 | 0.7267 | a00000386* | [168] |
| | | 1223 | 0.3885 | 0.9293 | 0.9210 | 0.9330 | 0.9020 | a00000387* | |
| | | 1299 | 0.3125 | 0.7306 | 0.7347 | 0.7282 | 0.7221 | a00000454* | |
| 1.00 | 2.04 | 1257 | 0.3659 | 0.8974 | 0.8928 | 0.8980 | 0.8801 | a00000455* | [168] |
| | | 1218 | 0.4284 | 1.0875 | 1.0890 | 1.0930 | 1.0616 | a00000456* | |
| | | 1454 | 0.0273 | 0.0389 | 0.0388 | 0.0390 | 0.0387 | a00000370* | |
| 1.00 | 1.94 | 1269 | 0.0616 | 0.0839 | 0.0839 | 0.0843 | 0.0825 | a00000371* | [169] |
| | | 1126 | 0.1386 | 0.1906 | 0.1902 | 0.1904 | 0.1815 | a00000372* | |
| | | 1352 | 0.0498 | 0.0501 | 0.0502 | 0.0502 | 0.0498 | a00000373 | |
| 1.00 | 0.71 | 1180 | 0.0992 | 0.1132 | 0.1131 | 0.1131 | 0.1116 | a00000374 | [170] |
| | | 1047 | 0.1977 | 0.2670 | 0.2668 | 0.2674 | 0.2562 | a00000375 | |
| | | 1161 | 0.2021 | 0.0634 | 0.0634 | 0.0636 | 0.0616 | a00000460* | |
| 1.00 | 1.45 | 1027 | 0.4783 | 0.1871 | 0.1867 | 0.1875 | 0.1659 | a00000461* | [171] |
| | | 921 | 1.1324 | 5.1721 | 5.1747 | 5.1740 | 2.3170 | a00000462* | |
| | | 1023 | 0.2167 | 0.1847 | 0.1841 | 0.1854 | 0.1368 | a00000463* | |
| 1.00 | 2.35 | 974 | 0.4857 | 2.4202 | 2.3691 | 2.3883 | 0.4586 | a00000464 | [171] |
| | | 930 | 1.0888 | 5.3892 | 4.8630 | 5.3778 | 5.2808 | a00000465 | |

| **Equiv ratio Φ** | **P [atm]** | **T [K]** | $\tau_{\text{ign}}$ **[ms]** | **Mech1 [ms]** | **Mech2 [ms]** | **Mech3 [ms]** | **Dynamic Mech [ms]** | **PrIMe ID** | **Ref** |
|---|---|---|---|---|---|---|---|---|---|
| | | 999 | 1.0014 | 1.0716 | 1.0699 | 1.0739 | 0.9646 | a00000388 | |
| 0.50 | 1.05 | 972 | 1.4996 | 1.4500 | 1.4437 | 1.4532 | 1.2587 | a00000389 | [81] |
| | | 947 | 2.2456 | 2.0914 | 2.0814 | 2.0960 | 1.7005 | a00000390 | |
| | | 1015 | 0.7533 | 1.2968 | 1.2821 | 1.2986 | 0.4879 | a00000391 | |
| 0.50 | 3.99 | 995 | 1.4457 | 7.2147 | 7.2012 | 7.2180 | 1.0347 | a00000392* | [81] |
| | | 976 | 2.7743 | 13.2888 | 13.7560 | 13.8132 | 5.0978 | a00000393* | |
| | | 1093 | 0.8912 | 1.8995 | 1.8022 | 1.8478 | 0.7162 | a00000394 | |
| 0.50 | 15.86 | 1067 | 1.5758 | 3.8511 | 3.5931 | 3.6759 | 1.6318 | a00000395 | [81] |
| | | 1042 | 2.7864 | 7.2684 | 6.6861 | 6.8215 | 3.3953 | a00000396 | |
| | | 1411 | 0.1112 | 0.0988 | 0.0986 | 0.0984 | 0.0973 | a00000415* | |
| 1.00 | 1.01 | 1197 | 0.2988 | 0.2638 | 0.2646 | 0.2637 | 0.2585 | a00000416 | [81] |
| | | 1039 | 0.8025 | 0.7395 | 0.7378 | 0.7398 | 0.7000 | a00000417 | |
| | | 1103 | 0.1333 | 0.1557 | 0.1552 | 0.1562 | 0.1361 | a00000418 | |
| 1.00 | 3.96 | 1052 | 0.3919 | 0.3219 | 0.3207 | 0.3236 | 0.2380 | a00000419 | [81] |
| | | 1005 | 1.1528 | 3.1265 | 3.0969 | 3.1303 | 0.6613 | a00000420 | |
| | | 1174 | 0.0505 | 0.0910 | 0.0902 | 0.0918 | 0.0412 | a00000421 | |
| 1.00 | 16.39 | 1115 | 0.2506 | 0.7473 | 0.7310 | 0.7495 | 0.2676 | a00000422 | [81] |
| | | 1063 | 1.2436 | 3.1065 | 2.9523 | 3.0279 | 1.3318 | a00000423* | |
| | | 1221 | 0.0739 | 0.1253 | 0.1254 | 0.1254 | 0.1167 | a00000439* | |
| 0.10 | 3.69 | 1108 | 0.3037 | 0.3078 | 0.3033 | 0.3056 | 0.2539 | a00000440 | [172] |
| | | 1014 | 1.2486 | 2.2990 | 2.0732 | 2.1424 | 0.9107 | a00000441* | |
| | | 1372 | 0.0410 | 0.0429 | 0.0432 | 0.0428 | 0.0420 | a00000427 | |
| 0.10 | 1.01 | 1162 | 0.1455 | 0.1111 | 0.1111 | 0.1113 | 0.1061 | a00000428 | [172] |
| | | 1007 | 0.5158 | 0.3540 | 0.3500 | 0.3523 | 0.3057 | a00000429 | |
| | | 1074 | 0.1147 | 0.1022 | 0.1001 | 0.1010 | 0.0707 | a00000430 | |
| 0.10 | 4.17 | 1022 | 0.4968 | 0.9257 | 0.8077 | 0.8444 | 0.1932 | a00000431* | [172] |
| | | 975 | 2.1530 | 10.4534 | 10.5319 | 10.4979 | 8.6371 | a00000432* | |
| | | 1113 | 0.3755 | 1.0117 | 0.9951 | 1.0064 | 0.4486 | a00000433* | |

| **Equiv ratio $\Phi$** | **P [atm]** | **T [K]** | $\tau_{\mathrm{ign}}$ **[ms]** | **Mech1 [ms]** | **Mech2 [ms]** | **Mech3 [ms]** | **Dynamic Mech [ms]** | **PrIMe ID** | **Ref** |
|---|---|---|---|---|---|---|---|---|---|
| 0.10 | 17.78 | 1055 | 1.0493 | 4.2563 | 4.0069 | 4.0325 | 2.3928 | a00000434 | [172] |
| | | 1003 | 2.9324 | 14.6548 | 13.7606 | 13.7917 | 8.8267 | a00000435 | |
| | | 1608 | 0.0810 | 0.1121 | 0.1125 | 0.1126 | 0.1096 | a00000436 | |
| 0.10 | 0.94 | 1299 | 0.2626 | 0.3110 | 0.3118 | 0.3129 | 0.3044 | a00000437* | [172] |
| | | 1090 | 0.8520 | 0.9487 | 0.9445 | 0.9504 | 0.8847 | a00000438 | |
| | | 1192 | 0.1241 | 0.1439 | 0.1298 | 0.1330 | 0.0671 | a00000442 | |
| 0.10 | 16.31 | 1136 | 0.5109 | 1.5750 | 1.4266 | 1.4894 | 0.3127 | a00000443* | [172] |
| | | 1084 | 2.1028 | 7.2727 | 6.4609 | 6.5958 | 3.0409 | a00000444* | |
| | | 1534 | 0.0811 | 0.0702 | 0.0703 | 0.0699 | 0.0678 | a00000397 | |
| 0.50 | 0.99 | 1262 | 0.2668 | 0.2054 | 0.2071 | 0.2075 | 0.1997 | a00000398 | [172] |
| | | 1072 | 0.8779 | 0.6450 | 0.6468 | 0.6468 | 0.6013 | a00000399 | |
| | | 1183 | 0.0939 | 0.0925 | 0.0920 | 0.0926 | 0.0829 | a00000400 | |
| 0.50 | 4.08 | 1118 | 0.3172 | 0.1793 | 0.1774 | 0.1802 | 0.1436 | a00000401* | [172] |
| | | 1059 | 1.0718 | 0.6965 | 0.6833 | 0.6970 | 0.3286 | a00000402* | |
| | | 1192 | 0.1853 | 0.1801 | 0.1716 | 0.1767 | 0.0592 | a00000403 | |
| 0.50 | 15.57 | 1144 | 0.5729 | 0.8245 | 0.7407 | 0.7717 | 0.2880 | a00000404 | [172] |
| | | 1101 | 1.7710 | 2.5422 | 2.1622 | 2.2671 | 0.9766 | a00000405 | |
| | | 1623 | 0.0849 | 0.0611 | 0.0611 | 0.0611 | 0.0625 | a00000406* | |
| 3.99 | 1.06 | 1308 | 0.2522 | 0.2013 | 0.2014 | 0.2017 | 0.2034 | a00000407 | [172] |
| | | 1096 | 0.7491 | 0.6646 | 0.6626 | 0.6666 | 0.6550 | a00000408 | |
| | | 1297 | 0.0565 | 0.0590 | 0.0591 | 0.0589 | 0.0587 | a00000409 | |
| 3.99 | 3.98 | 1164 | 0.1954 | 0.1306 | 0.1306 | 0.1312 | 0.1244 | a00000410 | [172] |
| | | 1057 | 0.6756 | 0.4523 | 0.4513 | 0.4569 | 0.3392 | a00000411* | |
| | | 1143 | 0.1697 | 0.2319 | 0.2350 | 0.2398 | 0.1039 | a00000412 | |
| 3.99 | 14.87 | 1069 | 0.7237 | 1.6455 | 1.6456 | 1.7219 | 0.7081 | a00000413 | [172] |
| | | 1004 | 3.0870 | 8.6285 | 8.7015 | 8.9960 | 4.1206 | a00000414 | |
| | | 1009 | 0.2344 | 0.6922 | 0.6872 | 0.6961 | 0.1452 | a00000457 | |
| 0.42 | 3.28 | 972 | 0.8252 | 4.0392 | 4.0517 | 4.0382 | 2.4558 | a00000458* | [100] |

| **Equiv ratio Φ** | **P [atm]** | **T [K]** | $\tau_{\mathrm{ign}}$ **[ms]** | **Mech1 [ms]** | **Mech2 [ms]** | **Mech3 [ms]** | **Dynamic Mech [ms]** | **PrIMe ID** | **Ref** |
|---|---|---|---|---|---|---|---|---|---|
| | | 938 | 2.9051 | 14.3582 | 14.0180 | 13.9642 | 14.0914 | a00000459* | |
| | | 1062 | 0.3217 | 0.3889 | 0.3865 | 0.3902 | 0.3210 | a00000376 | |
| 1.00 | 3.56 | 1012 | 1.0194 | 1.2506 | 1.2431 | 1.2570 | 0.6748 | a00000377* | [100] |
| | | 966 | 3.2307 | 15.6339 | 15.6974 | 15.7585 | 6.5254 | a00000378* | |
| | | 1296 | 0.2278 | 0.3240 | 0.3266 | 0.3266 | 0.3198 | a00000466 | |
| 1.00 | 1.00 | 1184 | 0.3902 | 0.5702 | 0.5701 | 0.5700 | 0.5593 | a00000467 | [173] |
| | | 1089 | 0.6685 | 1.0229 | 1.0155 | 1.0192 | 0.9823 | a00000468 | |
| | | 1564 | 0.1707 | 0.3284 | 0.3261 | 0.3263 | 0.3292 | a00000472 | |
| 1.03 | 1.00 | 1411 | 0.3018 | 0.5864 | 0.5807 | 0.5857 | 0.5711 | a00000473 | [173] |
| | | 1286 | 0.5337 | 1.0302 | 1.0187 | 1.0205 | 0.9976 | a00000474* | |
| | | 1386 | 0.1634 | 0.2577 | 0.2605 | 0.2598 | 0.2589 | a00000469* | |
| 1.47 | 1.00 | 1280 | 0.2403 | 0.4156 | 0.4170 | 0.4168 | 0.4162 | a00000470* | [173] |
| | | 1190 | 0.3532 | 0.6628 | 0.6661 | 0.6676 | 0.6564 | a00000471* | |
| | | 1270 | 0.0263 | 0.0534 | 0.0529 | 0.0543 | 0.0323 | a00000361 | |
| 1.00 | 33.00 | 1242 | 0.0592 | 0.1064 | 0.1064 | 0.1093 | 0.0497 | a00000362 | [174] |
| | | 1215 | 0.1337 | 0.2175 | 0.2185 | 0.2253 | 0.0901 | a00000363 | |
| | | 1714 | 0.0235 | 0.0372 | 0.0370 | 0.0375 | 0.0369 | a00000364 | |
| 1.00 | 64.00 | 1578 | 0.0423 | 0.0597 | 0.0596 | 0.0600 | 0.0579 | a00000365 | [174] |
| | | 1461 | 0.0760 | 0.1069 | 0.1063 | 0.1083 | 0.0962 | a00000366 | |
| | | 1327 | 0.1031 | 0.1136 | 0.1154 | 0.1208 | 0.0644 | a00000367 | |
| 1.00 | 64.00 | 1311 | 0.1438 | 0.1549 | 0.1582 | 0.1681 | 0.0818 | a00000368* | [174] |
| | | 1295 | 0.2007 | 0.2172 | 0.2202 | 0.2372 | 0.1067 | a00000369* | |
| | | 1566 | 0.0668 | 0.1112 | 0.1125 | 0.1123 | 0.1077 | a00000424 | |
| 0.25 | 1.32 | 1365 | 0.1137 | 0.2230 | 0.2220 | 0.2232 | 0.2169 | a00000425 | [175] |
| | | 1210 | 0.1933 | 0.4531 | 0.4490 | 0.4501 | 0.4364 | a00000426* | |
| | | 1045 | 0.5000 | 0.7352 | 0.7328 | 0.7414 | 0.3524 | a00000379 | |
| 2.00 | 5.00 | 1016 | 1.3800 | 3.7181 | 3.7042 | 3.7514 | 0.9310 | a00000380 | [176] |
| | | 990 | 3.8100 | 11.8584 | 11.8627 | 12.1387 | 3.7999 | a00000381 | |

| **Equiv ratio $\Phi$** | **P [atm]** | **T [K]** | $\tau_{\text{ign}}$ **[ms]** | **Mech1 [ms]** | **Mech2 [ms]** | **Mech3 [ms]** | **Dynamic Mech [ms]** | **PrIMe ID** | **Ref** |
|---|---|---|---|---|---|---|---|---|---|

* Indicates target was not included in final, reduced dataset.

## E.3 Flame Simulations

Table E.4: Comparison of flame speed results (mixture-averaged transport) with CloudFlame-Cantera and CHEMKIN-PRO for Mechanism1.

| **Target PrIMe ID** | **Preferred Key** | **Pressure [atm]** | $\Phi$ | **Mechanism1 CLOUDFLAME-CANTERA [cm/s]** | **Mechanism1 CHEMKIN PRO [cm/s]** |
|---|---|---|---|---|---|
| a00000476 | H2 F - 476 | 1 | 0.6 | 117.41 | 117.86 |
| a00000477 | H2 F - 477 | 1 | 1.65 | 324.76 | 329.36 |
| a00000478 | H2 F - 478 | 1 | 4 | 125.08 | 126.03 |
| a00000479 | H2 F - 479 | 20 | 1.5 | 94.83 | 94.59 |
| a00000480 | H2 F - 480 | 20 | 2 | 73.74 | 73.29 |
| a00000481 | H2 F - 481 | 20 | 0.85 | 36.76 | 36.79 |
| a00000482 | H2 F - 482 | 15 | 0.85 | 47.32 | 47.30 |
| a00000483 | H2 F - 483 | 10 | 0.85 | 61.58 | 61.57 |
| a00000484 | H2 F - 484 | 14 | 2.5 | 38.16 | 37.95 |
| a00000485 | H2 F - 485 | 25 | 2.5 | 37.84 | 37.61 |

Table E.5: Comparison of the four reaction mechanisms prediction of the flame-speed QOIs vs. experimental data. $\ell_2$-norm difference between model prediction (using the nominal model parameter) to the experimental data.

| | $\|\|M(x_{nom}) - S_l\|\|_2$ |
|---|---|
| Mechanism 1 | 12.29 |
| Mechanism 2 | 18.67 |
| Mechanism 3 | 16.78 |
| **DynamicMech151203** | **67.30** |

# Appendix F

# Char oxidation QOIs and predictions

Experiments conducted in a laminar-entrained flow reactor at Sandia National Laboratories [111] were used as QOIs for the char oxidation study of Chapter 6). Gas temperature profiles and gas velocity profiles used to guide simulations are available in [67]. Proximate analysis of the prepared chars used in the following experimental cases are found in Table F.2.

Table F.1 reports a list of the experimental cases investigated in this work, along with corresponding gas flow compositions, initial particle size bins, sampling heights, and PrIMe database IDs. Heights which were sampled more than once are identified by a repeated height above the burner value.

Table F.1: List of experimental cases with corresponding gas flow compositions, initial particle size bins, sampling heights (above the burner) and PrIMe database IDs

| **Exp case** | **Gas flow composition** | | | | **Particle size bin** [$\mu m$] | **Height above Burner** [cm] | **PrIMe ID** |
|---|---|---|---|---|---|---|---|
| | $\mathbf{y}_{O_2}$ | $\mathbf{y}_{H_2O}$ | $\mathbf{y}_{CO_2}$ | $\mathbf{y}_{N_2}$ | | | |
| 1 | 0.24 | 0.14 | 0.62 | 0.00 | 53-63 | 3.81, 5.08, 6.35, 7.62, 8.89, 10.16, 11.43 | `x00001418` |
| 2 | 0.24 | 0.14 | 0.62 | 0.00 | 63-75 | 3.81, 5.08, 6.35, 7.62, 8.89, 10.16, 11.43 | `x00001419` |
| 3 | 0.24 | 0.14 | 0.62 | 0.00 | 75-90 | 5.08, 7.62, 10.16, 12.7, 15.24 | `x00001420` |
| 4 | 0.24 | 0.14 | 0.62 | 0.00 | 90-106 | 5.08, 7.62, 10.16, 12.7, 15.24, 17.78 | `x00001421` |
| 5 | 0.24 | 0.14 | 0.62 | 0.00 | 106-125 | 7.62, 10.16, 12.7, 15.24, 17.78, 20.32 | `x00001422` |
| 6 | 0.24 | 0.14 | 0.62 | 0.00 | 125-150 | 7.62, 12.7, 17.78, 22.86 | `x00001423` |
| 7 | 0.36 | 0.10 | 0.54 | 0.00 | 53-63 | 2.54, 3.81, 5.08, 6.35, 7.62 | `x00001436` |
| 8 | 0.36 | 0.10 | 0.54 | 0.00 | 63-75 | 3.81, 5.08, 6.35, 7.62, 8.89 | `x00001437` |

**Table F.1 Continued From Previous Page**

| Exp case | Gas flow composition | | | | Particle size bin [$\mu m$] | Height above Burner [cm] | PrIMe ID |
|---|---|---|---|---|---|---|---|
| | $y_{O_2}$ | $y_{H_2O}$ | $y_{CO_2}$ | $y_{N_2}$ | | | |
| 9 | 0.36 | 0.10 | 0.54 | 0.00 | 75-90 | 3.81, 5.08, 6.35, 7.62, 8.89, 10.16 | x00001438 |
| 10 | 0.36 | 0.10 | 0.54 | 0.00 | 90-106 | 5.08, 7.62, 10.16, 12.7 | x00001439 |
| 11 | 0.36 | 0.10 | 0.54 | 0.00 | 106-125 | 7.62, 10.16, 12.7, 15.24, 17.78 | x00001440 |
| 12 | 0.36 | 0.10 | 0.54 | 0.00 | 125-150 | 7.62, 10.16, 12.7, 15.24, 17.78 | x00001441 |
| 13 | 0.36 | 0.14 | 0.50 | 0.00 | 53-63 | 2.54, 3.81, 5.08, 6.35, 7.62 | x00001454 |
| 14 | 0.36 | 0.14 | 0.50 | 0.00 | 63-75 | 3.81, 5.08, 6.35, 7.62, 8.89 | x00001455 |
| 15 | 0.36 | 0.14 | 0.50 | 0.00 | 75-90 | 5.08, 6.35, 7.62, 8.89, 10.16, 11.43 | x00001456 |
| 16 | 0.36 | 0.14 | 0.50 | 0.00 | 90-106 | 5.08, 7.62, 10.16, 12.7 | x00001457 |
| 17 | 0.36 | 0.14 | 0.50 | 0.00 | 106-125 | 5.08, 7.62, 10.16, 12.7, 12.7, 15.24 | x00001458 |
| 18 | 0.36 | 0.14 | 0.50 | 0.00 | 125-150 | 5.08, 7.62, 10.16, 12.7, 15.24, 17.78 | x00001459 |
| 19 | 0.60 | 0.10 | 0.30 | 0.00 | 53-63 | 2.54, 3.175, 3.81, 4.445, 5.08, 5.715 | x00001472 |
| 20 | 0.60 | 0.10 | 0.30 | 0.00 | 63-75 | 3.81, 4.445, 5.08, 5.715, 6.35, 6.985 | x00001473 |
| 21 | 0.60 | 0.10 | 0.30 | 0.00 | 75-90 | 3.81, 5.08, 6.35, 7.62, 8.89 | x00001474 |
| 22 | 0.60 | 0.10 | 0.30 | 0.00 | 90-106 | 5.08, 6.35, 7.62, 8.89, 10.16 | x00001475 |
| 23 | 0.60 | 0.10 | 0.30 | 0.00 | 106-125 | 6.35, 7.62, 8.89, 10.16, 11.43 | x00001476 |
| 24 | 0.60 | 0.10 | 0.30 | 0.00 | 125-150 | 6.35, 7.62, 8.89, 10.16, 11.43, 12.7 | x00001477 |
| 25 | 0.60 | 0.14 | 0.26 | 0.00 | 53-63 | 3.175, 3.81, 4.445, 5.08, 5.715, 6.35 | x00001490 |
| 26 | 0.60 | 0.14 | 0.26 | 0.00 | 63-75 | 3.175, 3.81, 4.445, 5.08, 5.715, 6.35, 6.985 | x00001491 |
| 27 | 0.60 | 0.14 | 0.26 | 0.00 | 75-90 | 3.81, 5.08, 6.35, 7.62, 8.89 | x00001492 |
| 28 | 0.60 | 0.14 | 0.26 | 0.00 | 90-106 | 5.08, 6.35, 7.62, 8.89, 10.16 | x00001493 |

**Table F.1 Continued From Previous Page**

| Exp case | Gas flow composition | | | | Particle size bin [$\mu m$] | Height above Burner [cm] | PrIMe ID |
|---|---|---|---|---|---|---|---|
| | $y_{O_2}$ | $y_{H_2O}$ | $y_{CO_2}$ | $y_{N_2}$ | | | |
| 29 | 0.60 | 0.14 | 0.26 | 0.00 | 106-125 | 5.08, 6.35, 7.62, 8.89, 10.16, 11.43 | x00001494 |
| 30 | 0.60 | 0.14 | 0.26 | 0.00 | 125-150 | 6.35, 7.62, 8.89, 10.16, 11.43, 12.7 | x00001495 |
| 31 | 0.60 | 0.16 | 0.24 | 0.00 | 53-63 | 3.175, 3.81, 4.445, 5.08, 5.715, 6.35, 6.985 | x00001508 |
| 32 | 0.60 | 0.16 | 0.24 | 0.00 | 63-75 | 3.81, 4.445, 5.08, 5.715, 6.35, 6.985, 7.62 | x00001509 |
| 33 | 0.60 | 0.16 | 0.24 | 0.00 | 75-90 | 3.81, 5.08, 6.35, 7.62, 8.89 | x00001510 |
| 34 | 0.60 | 0.16 | 0.24 | 0.00 | 90-106 | 5.08, 6.35, 7.62, 8.89, 10.16 | x00001511 |
| 35 | 0.60 | 0.16 | 0.24 | 0.00 | 106-125 | 5.08, 6.35, 7.62, 8.89, 10.16, 11.43 | x00001512 |
| 36 | 0.60 | 0.16 | 0.24 | 0.00 | 125-150 | 5.08, 6.35, 7.62, 8.89, 10.16, 11.43 | x00001513 |
| 37 | 0.24 | 0.14 | 0.00 | 0.62 | 53-63 | 2.54, 2.54, 3.81, 5.08, 6.35 | x00001526 |
| 38 | 0.24 | 0.14 | 0.00 | 0.62 | 63-75 | 2.54, 3.81, 5.08, 6.35, 7.62, 8.89, 10.16, 11.43 | x00001527 |
| 39 | 0.24 | 0.14 | 0.00 | 0.62 | 75-90 | 5.08, 7.62, 10.16, 12.7, 15.24 | x00001528 |
| 40 | 0.24 | 0.14 | 0.00 | 0.62 | 90-106 | 5.08, 7.62, 10.16, 12.7, 15.24, 17.78 | x00001529 |
| 41 | 0.24 | 0.14 | 0.00 | 0.62 | 106-125 | 7.62, 10.16, 12.7, 15.24, 17.78, 20.32 | x00001530 |
| 42 | 0.24 | 0.14 | 0.00 | 0.62 | 125-150 | 7.62, 10.16, 10.16, 12.7, 15.24, 17.78, 20.32, 22.86 | x00001531 |
| 43 | 0.36 | 0.10 | 0.00 | 0.54 | 53-63 | 2.54, 3.81, 5.08, 6.35, 7.62 | x00001544 |
| 44 | 0.36 | 0.10 | 0.00 | 0.54 | 63-75 | 3.81, 5.08, 6.35, 7.62, 8.89 | x00001545 |
| 45 | 0.36 | 0.10 | 0.00 | 0.54 | 75-90 | 3.81, 5.08, 6.35, 7.62, 8.89, 10.16 | x00001546 |
| 46 | 0.36 | 0.10 | 0.00 | 0.54 | 90-106 | 5.08, 7.62, 10.16, 12.7 | x00001547 |
| 47 | 0.36 | 0.10 | 0.00 | 0.54 | 106-125 | 5.08, 7.62, 10.16, 12.7, 15.24 | x00001548 |
| 48 | 0.36 | 0.10 | 0.00 | 0.54 | 125-150 | 5.08, 5.08, 7.62, 10.16, 12.7, 15.24, 17.78 | x00001549 |

**Table F.1 Continued From Previous Page**

| Exp case | Gas flow composition | | | | Particle size bin [$\mu m$] | Height above Burner [cm] | PrIMe ID |
|---|---|---|---|---|---|---|---|
| | $y_{O_2}$ | $y_{H_2O}$ | $y_{CO_2}$ | $y_{N_2}$ | | | |
| 49 | 0.36 | 0.14 | 0.00 | 0.50 | 53-63 | 2.54, 3.81, 5.08, 6.35, 7.62 | x00001562 |
| 50 | 0.36 | 0.14 | 0.00 | 0.50 | 63-75 | 2.54, 3.81, 5.08, 6.35, 7.62, 8.89 | x00001563 |
| 51 | 0.36 | 0.14 | 0.00 | 0.50 | 75-90 | 3.81, 5.08, 6.35, 7.62, 8.89, 10.16 | x00001564 |
| 52 | 0.36 | 0.14 | 0.00 | 0.50 | 90-106 | 5.08, 7.62, 10.16, 12.7 | x00001565 |
| 53 | 0.36 | 0.14 | 0.00 | 0.50 | 106-125 | 5.08, 7.62, 10.16, 12.7, 15.24 | x00001566 |
| 54 | 0.36 | 0.14 | 0.00 | 0.50 | 125-150 | 5.08, 5.08, 7.62, 7.62, 10.16, 12.7, 15.24 | x00001567 |
| 55 | 0.60 | 0.10 | 0.00 | 0.30 | 53-63 | 1.905, 2.54, 3.175, 3.81, 4.445, 5.08 | x00001580 |
| 56 | 0.60 | 0.10 | 0.00 | 0.30 | 63-75 | 3.175, 3.81, 4.445, 5.08, 5.715, 6.35 | x00001581 |
| 57 | 0.60 | 0.10 | 0.00 | 0.30 | 75-90 | 5.08, 5.715, 6.35, 6.985, 7.62, 8.255 | x00001582 |
| 58 | 0.60 | 0.10 | 0.00 | 0.30 | 90-106 | 5.08, 6.35, 7.62, 8.89 | x00001583 |
| 59 | 0.60 | 0.10 | 0.00 | 0.30 | 106-125 | 5.08, 6.35, 7.62, 8.89 | x00001584 |
| 60 | 0.60 | 0.10 | 0.00 | 0.30 | 125-150 | 5.08, 6.35, 7.62, 8.89, 10.16 | x00001585 |
| 61 | 0.60 | 0.14 | 0.00 | 0.26 | 53-63 | 1.27, 1.905, 3.175, 3.81, 2.54 | x00001598 |
| 62 | 0.60 | 0.14 | 0.00 | 0.26 | 63-75 | 3.175, 3.81, 4.445, 4.445, 2.54, 5.08 | x00001599 |
| 63 | 0.60 | 0.14 | 0.00 | 0.26 | 75-90 | 3.81, 4.445, 5.715, 6.35, 6.985, 5.08 | x00001600 |
| 64 | 0.60 | 0.14 | 0.00 | 0.26 | 90-106 | 5.715, 6.35, 6.985, 8.255, 8.89, 7.62 | x00001601 |
| 65 | 0.60 | 0.14 | 0.00 | 0.26 | 106-125 | 6.35, 5.08, 8.89, 7.62, 10.16 | x00001602 |
| 66 | 0.60 | 0.14 | 0.00 | 0.26 | 125-150 | 6.35, 5.08, 8.89, 7.62, 10.16 | x00001603 |
| 67 | 0.60 | 0.16 | 0.00 | 0.24 | 53-63 | 1.905, 2.54, 3.175, 3.81, 4.445, 5.08 | x00001616 |
| 68 | 0.60 | 0.16 | 0.00 | 0.24 | 63-75 | 2.54, 3.175, 3.81, 4.445, 5.08, 5.715 | x00001617 |
| 69 | 0.60 | 0.16 | 0.00 | 0.24 | 75-90 | 4.445, 5.08, 5.715, 6.35, 6.985 | x00001618 |

**Table F.1 Continued From Previous Page**

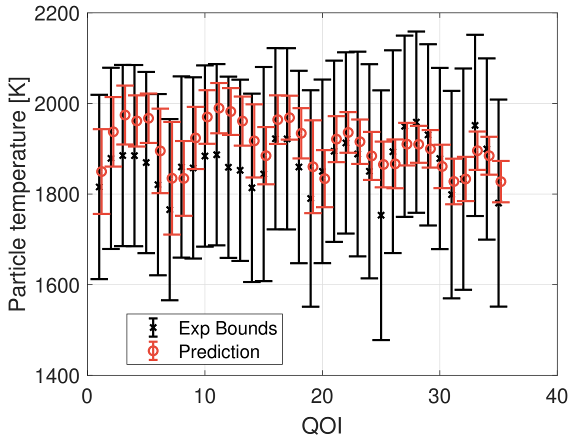
| **Exp case** | **Gas flow composition** | | | | **Particle size bin** [$\mu m$] | **Height above Burner** [cm] | **PrIMe ID** |
|---|---|---|---|---|---|---|---|
| | $\mathbf{y}_{O_2}$ | $\mathbf{y}_{H_2O}$ | $\mathbf{y}_{CO_2}$ | $\mathbf{y}_{N_2}$ | | | |
| 70 | 0.60 | 0.16 | 0.00 | 0.24 | 90-106 | 3.81, 5.08, 6.35, 7.62, 8.89 | x00001619 |
| 71 | 0.60 | 0.16 | 0.00 | 0.24 | 106-125 | 6.35, 7.62, 8.89, 10.16 | x00001620 |
| 72 | 0.60 | 0.16 | 0.00 | 0.24 | 125-150 | 5.08, 6.35, 7.62, 8.89, 10.16 | x00001621 |

Table F.2: Proximate analysis of the prepared chars. TGA measurements were made after the sieving and before injection of the chars in the entrained flow reactor. All measurements are wt.% db

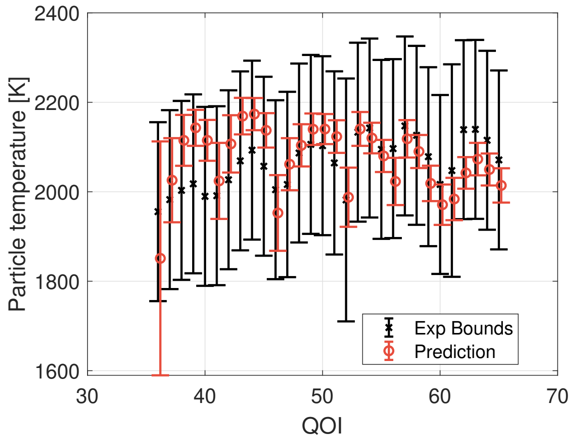
| | 53-63 $\mu$m | 63-75 $\mu$m | 75-90 $\mu$m | 90-106 $\mu$m | 106-125 $\mu$m | 125-150 $\mu$m |
|---|---|---|---|---|---|---|
| Volatiles | 6.83 | 6.35 | 4.61 | 4.82 | 3.05 | 3.53 |
| Fixed C | 71.62 | 76.59 | 73.86 | 78.50 | 64.86 | 67.32 |
| Ash | 21.55 | 17.07 | 21.53 | 16.68 | 32.09 | 29.15 |

## F.1 Interval prediction results

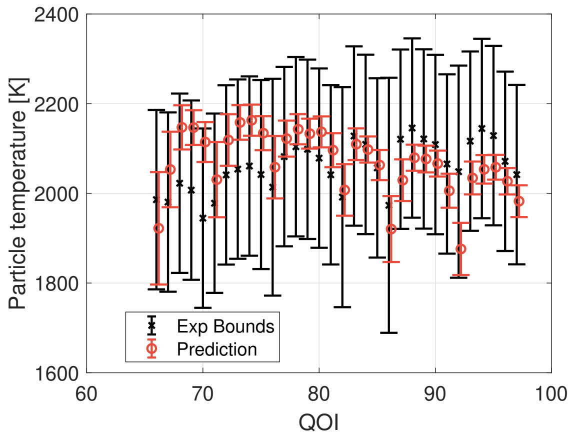
Figures F.1 - F.3 are the model predictions using a reduced char oxidation model with an initial particle size distribution function and light intensity model from Section 6.7. Surrogate model errors have not been applied to the experimental bounds, thus a few prediction intervals appear disjoint from the experimental data, despite the dataset being consistent when the experimental data is included.



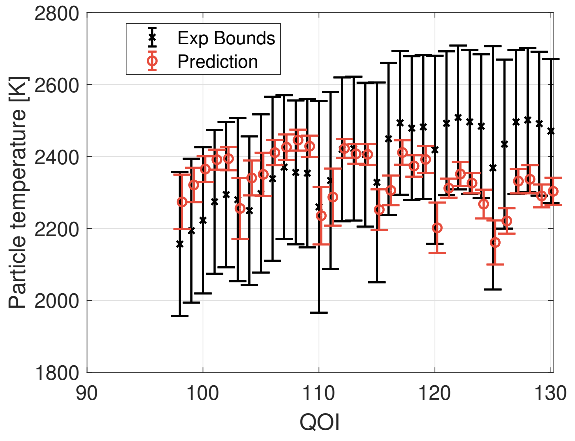
(a) 24% $O_2$, 14% $H_2O$, 62% $CO_2$



(b) 36% $O_2$, 10% $H_2O$, 54% $CO_2$

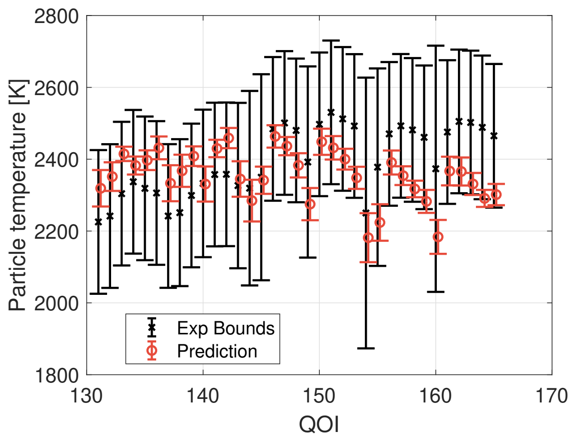


(c) 36% $O_2$, 14% $H_2O$, 50% $CO_2$

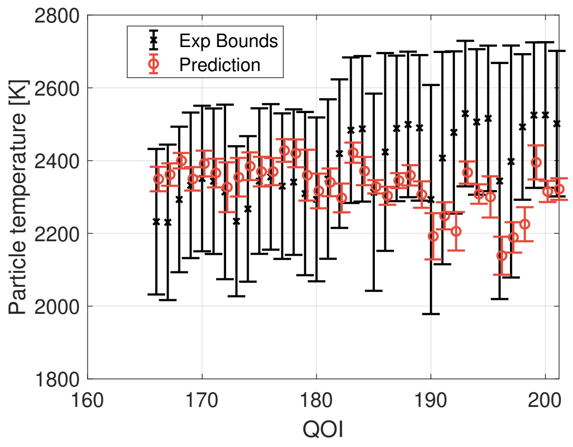


(d) 60% $O_2$, 10% $H_2O$, 30% $CO_2$

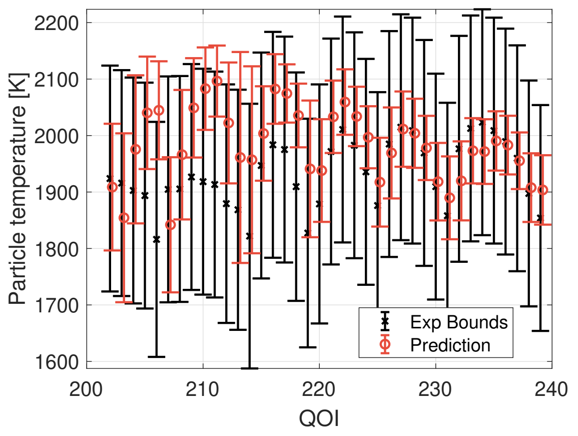
Figure F.1: Blind prediction of QOIs at gas conditions 1-4. The black intervals are the experimental bounds. The red intervals are the prediction of the $j$-th QOI (Eq. 3.4) using the feasible set excluding the $j$-th (Eq. B.7).



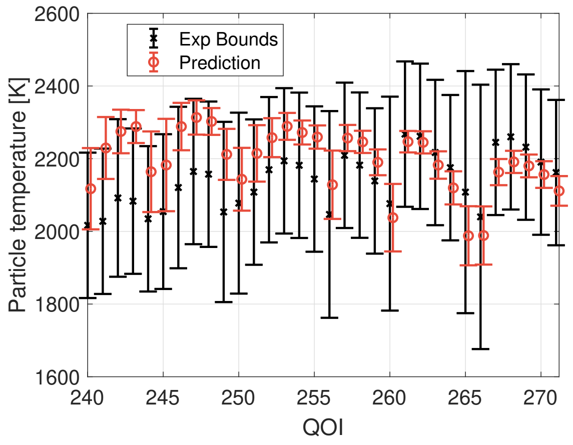
(a) 60% $O_2$, 14% $H_2O$, 26% $CO_2$



(b) 60% $O_2$, 16% $H_2O$, 24% $CO_2$

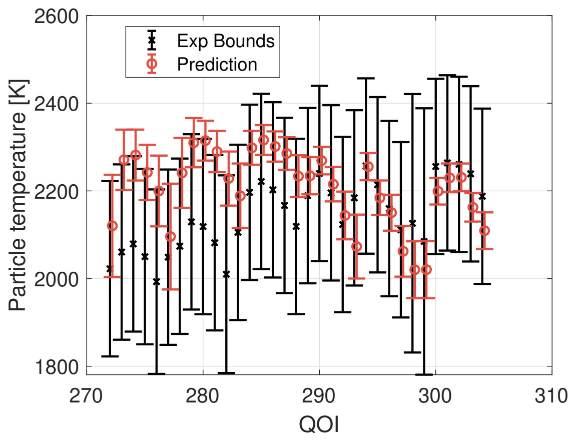


(c) 24% $O_2$, 14% $H_2O$, 62% $N_2$

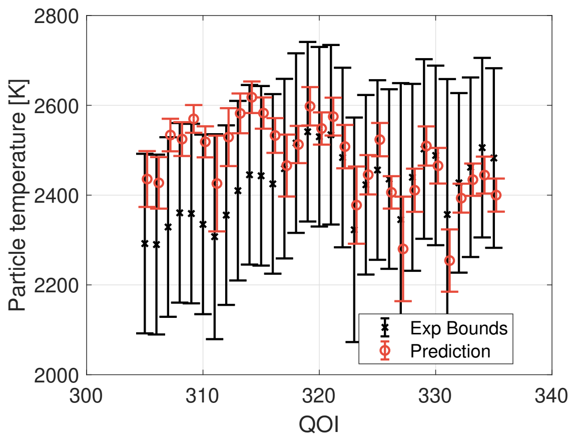


(d) 36% $O_2$, 10% $H_2O$, 54% $N_2$

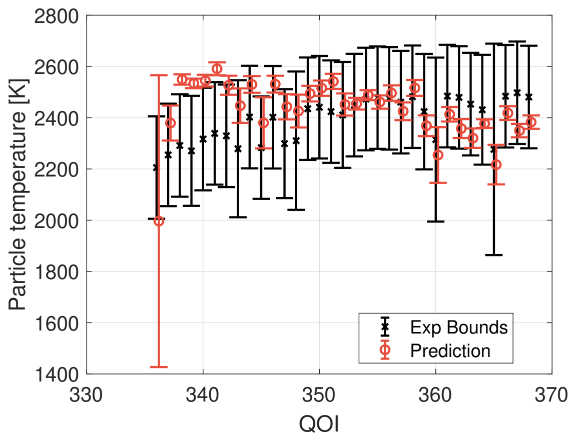
Figure F.2: Blind prediction of QOIs at gas conditions 5-8. The black intervals are the experimental bounds. The red intervals are the prediction of the $j$-th QOI (Eq. 3.4) using the feasible set excluding the $j$-th (Eq. B.7).



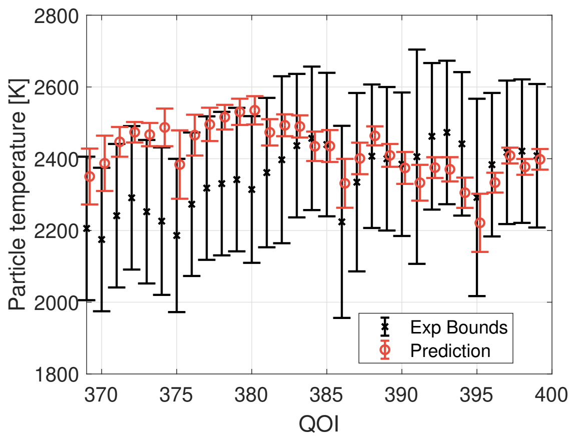
(a) 36% $O_2$, 14% $H_2O$, 50% $N_2$



(b) 60% $O_2$, 10% $H_2O$, 30% $N_2$



(c) 60% $O_2$, 14% $H_2O$, 26% $N_2$



(d) 60% $O_2$, 16% $H_2O$, 24% $CO_2$

Figure F.3: Blind prediction of QOIs at gas conditions 9-12. The black intervals are the experimental bounds. The red intervals are the prediction of the $j$-th QOI (Eq. 3.4) using the feasible set excluding the $j$-th (Eq. B.7).

# Bibliography

[1] Jerome Sacks, William J Welch, Toby J Mitchell, and Henry P Wynn. Design and analysis of computer experiments. *Statistical science*, pages 409–423, 1989. doi:10.1214/ss/1177012413.

[2] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13 (4):455–492, 1998. doi:10.1023/A:1008306431147.

[3] Nestor V Queipo, Raphael T Haftka, Wei Shyy, Tushar Goel, Rajkumar Vaidyanathan, and P Kevin Tucker. Surrogate-based analysis and optimization. *Progress in aerospace sciences*, 41(1):1–28, 2005. doi:10.1016/j.paerosci.2005.02.001.

[4] Alexander IJ Forrester and Andy J Keane. Recent advances in surrogate-based optimization. *Progress in aerospace sciences*, 45(1-3):50–79, 2009. doi:10.1016/j.paerosci.2008.11.001.

[5] U.S. Department of Energy. Scientific Grand Challenges for National Security: The Role of Computing at the Extreme Scale. *Technical Report*, 2009.

[6] William L Oberkampf and Christopher J Roy. *Verification and validation in scientific computing.* Cambridge University Press, 2010. doi:10.1017/CBO9780511760396.

[7] Maria J Bayarri, James O Berger, Rui Paulo, Jerry Sacks, John A Cafeo, James Cavendish, Chin-Hsu Lin, and Jian Tu. A framework for validation of computer models. *Technometrics*, 49(2):138–154, 2007. doi:10.1198/004017007000000092.

[8] Robert G Sargent. Verification and validation of simulation models. In *Proceedings of the 2010 Winter Simulation Conference*, pages 166–183. IEEE, 2010. doi:10.1109/WSC.2010.5679166.

[9] Timothy John Sullivan. *Introduction to uncertainty quantification.* Springer, 2015. doi:10.1007/978-3-319-23395-6.

[10] Albert Tarantola. *Inverse problem theory and methods for model parameter estimation.* SIAM, 2005. doi:10.1137/1.9780898717921.

[11] R Johnson, A Watkinson, and M Mabe. The STM Report: an overview of scientific and scholarly publishing, 2018.

[12] Joan Esnayra, Robert Pool, National Research Council, et al. *Bioinformatics: Converting Data to Knowledge: Workshop Summary*. National Academies Press, 2000. doi:10.17226/9990.

[13] Michael Frenklach. Transforming data into knowledge—Process informatics for combustion chemistry. *Proc. Combust. Inst.*, 31(1):125–140, 2007. doi:10.1016/j.proci.2006.08.121.

[14] David S Roos. Bioinformatics–trying to swim in a sea of data. *Science*, 291(5507): 1260–1261, 2001. doi:10.1126/science.291.5507.1260.

[15] Jim Gray, David T Liu, Maria Nieto-Santisteban, Alex Szalay, David J DeWitt, and Gerd Heber. Scientific data management in the coming decade. *Acm Sigmod Record*, 34(4):34–41, 2005. doi:10.1145/1107499.1107503.

[16] Karen L Schuchardt, James D Myers, and Eric G Stephan. A web-based data architecture for problem-solving environments: Application of distributed authoring and versioning to the extensible computational chemistry environment. *Cluster Computing*, 5(3):287–296, 2002. doi:10.1023/A:1015625205311.

[17] George Chin Jr and Carina S Lansing. Capturing and supporting contexts for scientific data sharing via the biological sciences collaboratory. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 409–418. ACM, 2004. doi:10.1145/1031607.1031677.

[18] James D Myers, Thomas C Allison, Sandra Bittner, Brett Didier, Michael Frenklach, William H Green, Yen-Ling Ho, John Hewson, Wendy Koegler, Carina Lansing, et al. A collaborative informatics infrastructure for multi-scale science. *Cluster Computing*, 8(4):243–253, 2005. doi:10.1007/s10586-005-4092-4.

[19] Raymond KW Wong, Curtis B Storlie, and Thomas CM Lee. A frequentist approach to computer model calibration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(2):635–648, 2017. doi:10.1111/rssb.12182.

[20] Thomas Bayes. LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFR S. *Philosophical transactions of the Royal Society of London*, 53:370–418, 1763. doi:10.1098/rstl.1763.0053.

[21] Devinderjit Sivia and John Skilling. *Data analysis: a Bayesian tutorial*. Oxford University Press, 2006.

[22] Marc C Kennedy and Anthony O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001. doi:10.1111/1467-9868.00294.

[23] Andrew Gelman, Hal S Stern, John B Carlin, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 2013.

[24] Ramon E Moore. *Interval analysis*, volume 4. Prentice-Hall Englewood Cliffs, NJ, 1966.

[25] Scott Ferson, Vladik Kreinovich, Janos Hajagos, William Oberkampf, and Lev Ginzburg. Experimental uncertainty estimation and statistics for data having interval uncertainty. *Sandia National Laboratories, Report SAND2007-0939*, 162, 2007.

[26] Scott Ferson and Lev R Ginzburg. Different methods are needed to propagate ignorance and variability. *Reliability Engineering & System Safety*, 54(2-3):133–144, 1996. doi:10.1016/S0951-8320(96)00071-3.

[27] AP Dempster. Upper and lower probabilities induced by a multivalued mapping. *The Annals of Mathematical Statistics*, 38(2):325–339, 1967.

[28] Glenn Shafer. *A mathematical theory of evidence*, volume 42. Princeton University press, 1976.

[29] Lotfi Asker Zadeh. The role of fuzzy logic in the management of uncertainty in expert systems. *Fuzzy sets and systems*, 11(1-3):199–227, 1983. doi:10.1016/S0165-0114(83)80081-5.

[30] Hans-Jürgen Zimmermann. *Fuzzy set theory and its applications*. Springer Science & Business Media, 2011. doi:10.1007/978-94-015-7949-0.

[31] Steven N Evans and Philip B Stark. Inverse problems as statistics. *Inverse problems*, 18(4):R55, 2002. doi:10.1088/0266-5611/18/4/201.

[32] Philip B Stark and Luis Tenorio. A primer of Frequentist and Bayesian inference in inverse problems. *Large-Scale Inverse Problems and Quantification of Uncertainty*, pages 9–32, 2010. doi:10.1002/9780470685853.ch2.

[33] Andrew Gelman et al. Objections to Bayesian statistics. *Bayesian Analysis*, 3(3): 445–449, 2008. doi:10.1214/08-BA318.

[34] Michael Frenklach, Andrew Packard, Gonzalo Garcia-Donato, Rui Paulo, and Jerome Sacks. Comparison of statistical and deterministic frameworks of uncertainty quantification. *SIAM/ASA Journal Uncertainty Quantif.*, 4(1):875–901, 2016. doi:10.1137/15M1019131.

[35] Matteo Broggi, Matthias Faes, Edoardo Patelli, Yves Govers, David Moens, and Michael Beer. Comparison of Bayesian and interval uncertainty quantification: Application to the AIRMOD test structure. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2017. doi:10.1109/SSCI.2017.8280882.

[36] Philip B. Stark. Constraints versus priors. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):586–598, 2015. doi:10.1137/130920721.

[37] PrIMe: Process Informatics Model, available at `http://www.primekinetics.org/`, 2006.

[38] Xiaoqing You, Andrew Packard, and Michael Frenklach. Process informatics tools for predictive modeling: Hydrogen combustion. *International Journal of Chemical Kinetics*, 44(2):101–116, 2012. doi:10.1002/kin.20627.

[39] Bonnie J McBride, Sanford Gordon, and Martin A Reno. Coefficients for calculating thermodynamic and transport properties of individual species. *Technical Report*, 1993.

[40] Jonathan D. Linton. All journals need to correct errors. *Nature*, 504, Dec 2013. doi:10.1038/504033d.

[41] Justin Kitzes, Daniel Turek, and Fatma Deniz. *The practice of reproducible research: case studies and lessons from the data-intensive sciences*. Univ of California Press, 2017.

[42] James G Speight. *Handbook of coal analysis*. John Wiley & Sons, 2015.

[43] Christopher R Shaddix and Alejandro Molina. Particle imaging of ignition and devolatilization of pulverized coal during oxy-fuel combustion. *Proceedings of the Combustion Institute*, 32(2):2091–2098, 2009. doi:10.1016/j.proci.2008.06.157.

[44] KJ Knill, TFJ Maalman, and ME Morgan. Development of combustion characterization technique for high volatile bituminous coals. *IFRF Doc No F*, 88, 1989.

[45] James Oreluk. Warehouse-API. `https://github.com/oreluk/Warehouse-API`, 2019.

[46] Clinton Gormley and Zachary Tong. *Elasticsearch: The definitive guide: A distributed real-time search and analytics engine*. O'Reilly Media, Inc., 2015.

[47] James Oreluk. coalDB. `https://github.com/oreluk/coalDB`, 2019.

[48] J.H.P. Haas, J. Daimon, and G. Gallager. Characterisation of weight loss and char burnout behavior for Rietspruit coal. *Technical Report*, 1996.

[49] ReSpecTh, available at `http://respecth.chem.elte.hu/respecth/`, 2015.

[50] Bryan W Weber and Kyle E Niemeyer. ChemKED: A human-and machine-readable data standard for chemical kinetics experiments. *International Journal of Chemical Kinetics*, 50(3):135–148, 2018. doi:10.1002/kin.21142.

[51] CloudFlame available at `https://cloudflame.kaust.edu.sa/`, 2013.

[52] Z. Reyno-Chiasson, N. Nettyam, G.L. Goteng, M. Speight, B.J. Lee, S. Baskaran, J. Oreluk, A. Farooq, H.G. Im, M. Frenklach, et al. CloudFlame and PrIMe: accelerating combustion research in the cloud. In *9th Int. Conf. on Chemical Kinetics, Ghent, Belgium*, 2015.

[53] Michael Frenklach, Andrew Packard, and Pete Seiler. Prediction uncertainty from models and data. In *Proceedings of the American Control Conference*, volume 5, pages 4135–4140. IEEE, 2002. doi:10.1109/ACC.2002.1024578.

[54] Pete Seiler, Michael Frenklach, Andrew Packard, and Ryan Feeley. Numerical approaches for collaborative data processing. *Optimization and Engineering*, 7(4): 459–478, 2006. doi:10.1007/s11081-006-0350-4.

[55] Trent Russi, Andy Packard, and Michael Frenklach. Uncertainty quantification: Making predictions of complex reaction systems reliable. *Chemical Physics Letters*, 499(1-3):1–8, 2010. doi:10.1016/j.cplett.2010.09.009.

[56] Ryan Feeley, Pete Seiler, Andrew Packard, and Michael Frenklach. Consistency of a reaction dataset. *Journal of Physical Chemistry A*, 108(44):9573–9583, 2004. doi:10.1021/jp047524w.

[57] Trent Michael Russi. *Uncertainty quantification with experimental data and complex system models*. PhD thesis, University of California, Berkeley, 2010.

[58] Panos M Pardalos and Stephen A Vavasis. Quadratic programming with one negative eigenvalue is NP-hard. *Journal of Global Optimization*, 1(1):15–22, 1991. doi:10.1007/BF00120662.

[59] Sartaj Sahni. Computationally related problems. *SIAM Journal on Computing*, 3(4): 262–279, 1974. doi:10.1137/0203021.

[60] Ryan Patrick Feeley. *Fighting the curse of dimensionality: A method for model validation and uncertainty propagation for complex simulation models*. PhD thesis, University of California, Berkeley, 2008.

[61] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[62] Arun Hegde, Wenyu Li, James Oreluk, Andrew Packard, and Michael Frenklach. Consistency analysis for massively inconsistent datasets in Bound-to-Bound Data Collaboration. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):429–456, 2018. doi:10.1137/16M1110005.

[63] Xiaoqing You, Trent Russi, Andrew Packard, and Michael Frenklach. Optimization of combustion kinetic models on a feasible set. *Proceedings of the Combustion Institute*, 33(1):509–516, 2011. doi:10.1016/j.proci.2010.05.016.

[64] Statistics and Machine Learning Toolbox, Parallel Computing Toolbox, and Optimization Toolbox Release 2017b. The MathWorks Inc. *Natick, MA*, 2002.

[65] Devin R Yeates, Wenjun Li, Phillip R Westmoreland, William Speight, Trent Russi, Andrew Packard, and Michael Frenklach. Integrated data-model analysis facilitated by an Instrumental Model. *Proceedings of the Combustion Institute*, 35(1):597–605, 2015. doi:10.1016/j.proci.2014.05.090.

[66] James Oreluk, Craig D Needham, Sathya Baskaran, S Mani Sarathy, Michael P Burke, Richard H West, Michael Frenklach, and Phillip R Westmoreland. Dynamic Chemical Model for $H_2/O_2$ Combustion Developed Through a Community Workflow. *arXiv preprint arXiv:1801.10093*, 2018.

[67] Salvatore Iavarone, James Oreluk, Sean T Smith, Arun Hegde, Wenyu Li, Andrew Packard, Michael Frenklach, Philip J Smith, Francesco Contino, and Alessandro Parente. Application of Bound-to-Bound Data Collaboration approach for development and uncertainty quantification of a reduced char combustion model. *Fuel*, 232:769–779, 2018. doi:10.1016/j.fuel.2018.05.113.

[68] James Oreluk, Zhenyuan Liu, Arun Hegde, Wenyu Li, Andrew Packard, Michael Frenklach, and Dmitry Zubarev. Diagnostics of data-driven models: Uncertainty quantification of PM7 semi-empirical quantum chemical method. *Scientific Reports*, 8(1):13248, September 2018. ISSN 2045-2322. URL `https://doi.org/10.1038/s41598-018-31677-y`.

[69] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972. doi:10.1137/0201010.

[70] Andrew Packard, Michael Frenklach, Wenyu Li, James Oreluk, and Arun Hedge. B2BDC Toolbox. `https://github.com/B2BDC/B2BDC`, 2016.

[71] Zekai Hong, David F Davidson, and Ronald K Hanson. An improved $H_2/O_2$ mechanism based on recent shock tube/laser absorption measurements. *Combustion and Flame*, 158(4):633–644, 2011. doi:10.1016/j.combustflame.2010.10.002.

[72] Michael P Burke, Marcos Chaos, Yiguang Ju, Frederick L Dryer, and Stephen J Klippenstein. Comprehensive $H_2/O_2$ kinetic model for high-pressure combustion. *International Journal of Chemical Kinetics*, 44(7):444–474, 2012. doi:10.1002/kin.20603.

[73] Alan Kéromnès, Wayne K Metcalfe, Karl A Heufer, Nicola Donohoe, Apurba K Das, Chih-Jen Sung, Jürgen Herzler, Clemens Naumann, Peter Griebel, Olivier Mathieu, et al. An experimental and detailed chemical kinetic modeling study of hydrogen and syngas mixture oxidation at elevated pressures. *Combustion and Flame*, 160(6): 995–1011, 2013. doi:10.1016/j.combustflame.2013.01.001.

[74] Tamás Varga, Tibor Nagy, C Olm, I Gy Zsély, R Pálvölgyi, É Valkó, G Vincze, M Cserháti, HJ Curran, and T Turányi. Optimization of a hydrogen combustion mechanism using both direct and indirect measurements. *Proceedings of the Combustion Institute*, 35(1):589–596, 2015. doi:10.1016/j.proci.2014.06.071.

[75] Alexander A Konnov. On the role of excited species in hydrogen combustion. *Combustion and Flame*, 162(10):3755–3772, 2015. doi:10.1016/j.combustflame.2015.07.014.

[76] Alexander A Konnov. Yet another kinetic mechanism for hydrogen combustion. *Combustion and Flame*, 203:14–22, 2019. doi:10.1016/j.combustflame.2019.01.032.

[77] Svante Arrhenius. Über die reaktionsgeschwindigkeit bei der inversion von rohrzucker durch säuren. *Zeitschrift für physikalische Chemie*, 4(1):226–248, 1889. doi:10.1515/zpch-1889-0416.

[78] Jeffrey I Steinfeld, Joseph Salvadore Francisco, and William L Hase. *Chemical kinetics and dynamics*. Pearson, 1989.

[79] Michael P Burke, Stephen J Klippenstein, and Lawrence B Harding. A quantitative explanation for the apparent anomalous temperature dependence of $OH+HO_2=H_2O+O_2$ through multi-scale modeling. *Proceedings of the Combustion Institute*, 34(1):547–555, 2013. doi:10.1016/j.proci.2012.05.041.

[80] Stig R Sellevåg, Yuri Georgievskii, and James A Miller. Kinetics of the gas-phase recombination reaction of hydroxyl radicals to form hydrogen peroxide. *The Journal of Physical Chemistry A*, 113(16):4457–4467, 2009. doi:10.1021/jp8110524.

[81] Jürgen Herzler and Clemens Naumann. Shock-tube study of the ignition of methane/ethane/hydrogen mixtures with hydrogen contents from 0% to 100% at different pressures. *Proceedings of the combustion institute*, 32(1):213–220, 2009. doi:10.1016/j.proci.2008.07.034.

[82] Michael Frenklach, Hai Wang, and Martin J Rabinowitz. Optimization and analysis of large chemical kinetic mechanisms using the solution mapping methodcombustion of methane. *Progress in Energy and Combustion Science*, 18(1):47–73, 1992. doi:10.1016/0360-1285(92)90032-V.

[83] Michael Frenklach, Andrew Packard, and Ryan Feeley. Optimization of reaction models with solution mapping. *Comprehensive Chemical Kinetics*, 42:243, 2007. doi:10.1016/S0069-8040(07)42006-4.

[84] Andrea Saltelli, Stefano Tarantola, Francesca Campolongo, and Marco Ratto. *Sensitivity analysis in practice: a guide to assessing scientific models*. Wiley, 2004.

[85] Ilya M Sobol. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and computers in simulation*, 55(1-3):271–280, 2001. doi:10.1016/S0378-4754(00)00270-6.

[86] Michael Frenklach. Modeling. In William C. Gardiner, editor, *Combustion Chemistry*, pages 423–453. Springer New York, New York, NY, 1984. ISBN 978-1-4684-0186-8. doi:10.1007/978-1-4684-0186-8_7.

[87] Jacques Hadamard. Resolution d'une question relative aux determinants. *Bull. des sciences math.*, 2:240–246, 1893.

[88] Robin L Plackett and J Peter Burman. The design of optimum multifactorial experiments. *Biometrika*, pages 305–325, 1946. doi:10.2307/2332195.

[89] Michael Frenklach, William Speight, and James Oreluk. ReactionLab. `https://github.com/PrIMeKinetics/ReactionLab`, 2019.

[90] William Cecil Gardiner and Alexander Burcat. *Combustion chemistry*. Springer, 1984. doi:10.1007/978-1-4684-0186-8.

[91] William Cecil Gardiner et al. *Rates and mechanisms of chemical reactions*. Benjamin, 1969.

[92] Forman A Williams. *Combustion theory*. CRC Press, 2018. doi:10.1201/9780429494055.

[93] G.L. Goteng, M Speight, N Nettyam, A Farooq, M Frenklach, and S.M. Sarathy. A hybrid cloud system for combustion kinetics simulation. In *23rd International Symposium on Gas Kinetics and Related Phenomena, Hungary*, 2014.

[94] David G. Goodwin, Raymond L. Speth, Harry K. Moffat, and Bryan W. Weber. Cantera: An object-oriented software toolkit for chemical kinetics, thermodynamics, and transport processes. `https://www.cantera.org`, 2018. Version 2.4.0.

[95] DL Baulch, CT Bowman, CJ Cobos, RA Cox, Th Just, JA Kerr, MJ Pilling, D Stocker, J Troe, W Tsang, et al. Evaluated kinetic data for combustion modeling: Supplement II. *Journal of physical and chemical reference data*, 34(3):757–1397, 2005. doi:10.1063/1.1748524.

[96] A Fernández-Ramos and AJC Varandas. A VTST study of the H + $O_3$ and O + $HO_2$ reactions using a six-dimensional DMBE potential energy surface for ground state $HO_3$. *The Journal of Physical Chemistry A*, 106(16):4077–4083, 2002. doi:10.1021/jp014120k.

[97] NK Srinivasan and JV Michael. The thermal decomposition of water. *International journal of chemical kinetics*, 38(3):211–219, 2006. doi:10.1002/kin.20172.

[98] Michael P Burke, Marcos Chaos, Frederick L Dryer, and Yiguang Ju. Negative pressure dependence of mass burning rates of $H_2$/CO/$O_2$/diluent flames at low flame temperatures. *Combustion and Flame*, 157(4):618–631, 2010. doi:10.1016/j.combustflame.2009.08.009.

[99] JW Meyer and AK Oppenheim. On the shock-induced ignition of explosive gases. In *Symposium (International) on Combustion*, volume 13, pages 1153–1164. Elsevier, 1971. doi:10.1016/S0082-0784(71)80112-1.

[100] GA Pang, DF Davidson, and RK Hanson. Experimental study and modeling of shock tube ignition delay times for hydrogen–oxygen–argon mixtures at low temperatures. *Proceedings of the combustion institute*, 32(1):181–188, 2009. doi:10.1016/j.proci.2008.06.014.

[101] Matthias Ihme, Yong Sun, and Ralf Deiterding. Detailed simulations of shock-bifurcation and ignition of an argon-diluted hydrogen/oxygen mixture in a shock tube. In *29th International Symposium on Shock Waves 1*, pages 209–214. Springer, 2015. doi:10.2514/6.2013-538.

[102] Kevin P Grogan and Matthias Ihme. Regimes describing shock boundary layer interaction and ignition in shock tubes. *Proceedings of the Combustion Institute*, 36 (2):2927–2935, 2017. doi:10.1016/j.proci.2016.06.078.

[103] IW Smith. The combustion rates of coal chars: a review. In *Symposium (International) on combustion*, volume 19, pages 1045–1065. Elsevier, 1982. doi:10.1016/S0082-0784(82)80281-6.

[104] J-L Su and DD Perlmutter. Effect of pore structure on char oxidation kinetics. *AIChE journal*, 31(6):973–981, 1985. doi:10.1002/aic.690310614.

[105] Robert H Hurt and Joseph M Calo. Semi-global intrinsic kinetics for char combustion modeling. *Combustion and flame*, 125(3):1138–1149, 2001. doi:10.1016/S0010-2180(01)00234-6.

[106] Reginald E Mitchell. An intrinsic kinetics-based, particle-population balance model for char oxidation during pulverized coal combustion. *Proceedings of the Combustion Institute*, 28(2):2261–2270, 2000. doi:10.1016/S0082-0784(00)80636-0.

[107] Liqiang Ma and Reginald Mitchell. Modeling char oxidation behavior under zone II burning conditions at elevated pressures. *Combustion and Flame*, 156(1):37–50, 2009. doi:10.1016/j.combustflame.2008.06.015.

[108] L Douglas Smoot and Philip J Smith. *Coal combustion and gasification*. Springer, 2013.

[109] K Lee Smith, L Douglas Smoot, Thomas H Fletcher, and Ronald J Pugmire. *The structure and reaction processes of coal*. Springer, 2013.

[110] Carbon Capture Multidisciplinary Simulation Center, available at `http://ccmsc.utah.edu/`, 2013.

[111] Ethan S Hecht. *Single particle studies of pulverized coal oxy-combustion*. PhD thesis, The University of Utah, 2013.

[112] DA Tichenor, RE Mitchell, KR Hencken, and S Niksa. Simultaneous in situ measurement of the size, temperature and velocity of particles in a combustion environment. In *Symposium (International) on Combustion*, volume 20, pages 1213–1221. Elsevier, 1985. doi:10.1016/S0082-0784(85)80610-X.

[113] Reginald E Mitchell. Experimentally determined overall burning rates of coal chars. *Combustion science and technology*, 53(2-3):165–186, 1987. doi:10.1080/00102208708947025.

[114] Jeffrey J Murphy and Christopher R Shaddix. Combustion kinetics of coal chars in oxygen-enriched environments. *Combustion and flame*, 144(4):710–729, 2006. doi:10.1016/j.combustflame.2005.08.039.

[115] Devin Rodney Yeates. *The Instrumental Model*. PhD thesis, University of California, Berkeley, 2011.

[116] R Byron Bird, Warren E Stewart, and Edwin N Lightfoot. *Transport phenomena*. John Wiley & Sons, 2007.

[117] Sean T Smith and Salvatore Iavarone. Personal Communication, June 2016.

[118] Suresh K Bhatia and DD Perlmutter. A random pore model for fluid-solid reactions: I. Isothermal, kinetic control. *AIChE Journal*, 26(3):379–386, 1980.

[119] Ethan S Hecht, Christopher R Shaddix, Manfred Geier, Alejandro Molina, and Brian S Haynes. Effect of $CO_2$ and steam gasification reactions on the oxy-combustion of pulverized coal char. *Combustion and Flame*, 159(11):3437–3447, 2012. doi:10.1016/j.combustflame.2012.06.009.

[120] Yang Xu, Shuiqing Li, Ye Yuan, and Qiang Yao. Measurement on the surface temperature of dispersed chars in a flat-flame burner using modified RGB pyrometry. *Energy & Fuels*, 31:2228–2235, 2016. doi:10.1021/acs.energyfuels.6b02203.

[121] Salvatore Iavarone, Benjamin Isaac, Sean Smith, Philip J Smith, Francesco Contino, and Alessandro Parente. Collaboration of simulations and experiments for development and uncertainty quantification of a reduced char combustion model. *Energy Procedia*, 120:500–507, 2017. doi:10.1016/j.egypro.2017.07.183.

[122] Michael D McKay, Richard J Beckman, and William J Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000. doi:10.1080/00401706.1979.10489755.

[123] PF Melia and CT Bowman. A three zone model for coal particle swelling. *Combustion science and technology*, 31(3-4):195–201, 1983. doi:10.1080/00102208308923640.

[124] Thomas K Gale, Calvin H Bartholomew, and Thomas H Fletcher. Decreases in the swelling and porosity of bituminous coals during devolatilization at high heating rates. *Combustion and Flame*, 100(1-2):94–100, 1995. doi:10.1016/0010-2180(94)00071-Y.

[125] Menachem Elimelech, John Gregory, and Xiadong Jia. *Particle deposition and aggregation: measurement, modelling and simulation*. Butterworth-Heinemann, 2013. doi:10.1016/B978-0-7506-7024-1.X5000-6.

[126] Valentin F Turchin. On the computation of multidimensional integrals by the Monte-Carlo method. *Theory of Probability & Its Applications*, 16(4):720–724, 1971. doi:10.1137/1116083.

[127] Robert L Smith. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984. doi:10.1287/opre.32.6.1296.

[128] Robert G Parr. Density functional theory of atoms and molecules. In *Horizons of Quantum Chemistry*, pages 5–15. Springer, 1980. doi:10.1007/978-94-009-9027-2_2.

[129] Richard A Friesner. Ab initio quantum chemistry: Methodology and applications. *Proceedings of the National Academy of Sciences*, 102(19):6648–6653, 2005. doi:10.1073/pnas.0408036102.

[130] Jean-Michel Combes, Pierre Duclos, and Ruedi Seiler. The Born-Oppenheimer approximation. In *Rigorous atomic and molecular physics*, pages 185–213. Springer, 1981. doi:10.1007/978-1-4613-3350-0_5.

[131] Attila Szabo and Neil S Ostlund. *Modern quantum chemistry: introduction to advanced electronic structure theory*. Dover Publications, 2012.

[132] Douglas R Hartree. The wave mechanics of an atom with a non-Coulomb central field. part I. Theory and methods. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 24, pages 89–110. Cambridge University Press, 1928. doi:10.1017/S0305004100011919.

[133] Vladimir Fock. Näherungsmethode zur Lösung des quantenmechanischen Mehrkörperproblems. *Zeitschrift für Physik*, 61(1-2):126–148, 1930. doi:10.1007/BF01340294.

[134] CCJ Roothaan. Self-consistent field theory for open shells of electronic systems. *Reviews of modern physics*, 32(2):179, 1960. doi:10.1103/RevModPhys.32.179.

[135] EJ Baerends, DE Ellis, and P Ros. Self-consistent molecular Hartree–Fock–Slater calculations I. the computational procedure. *Chemical Physics*, 2(1):41–51, 1973. doi:10.1016/0301-0104(73)80059-X.

[136] John A Pople and David L Beveridge. *Molecular orbital theory*. McGraw-Hill, 1970.

[137] Walter Thiel. Semiempirical methods: current status and perspectives. *Tetrahedron*, 44(24):7393–7408, 1988. doi:10.1016/S0040-4020(01)86235-9.

[138] Michael JS Dewar and Walter Thiel. Ground states of molecules. 38. the MNDO method. approximations and parameters. *Journal of the American Chemical Society*, 99(15):4899–4907, 1977. doi:10.1021/ja00457a004.

[139] Michael JS Dewar, Eve G Zoebisch, Eamonn F Healy, and James JP Stewart. Development and use of quantum mechanical molecular models. 76. AM1: a new general purpose quantum mechanical molecular model. *Journal of the American Chemical Society*, 107(13):3902–3909, 1985. doi:10.1021/ja00299a024.

[140] James JP Stewart. MOPAC2016. Stewart Computational Chemistry, Colorado Springs, CO, USA, available at http://openmopac.net, 2016.

[141] Pavlo O Dral, Xin Wu, Lasse Spörkel, Axel Koslowski, Wolfgang Weber, Rainer Steiger, Mirjam Scholten, and Walter Thiel. Semiempirical quantum-chemical orthogonalization-corrected methods: Theory, implementation, and parameters. *Journal of Chemical Theory and Computation*, 12(3):1082–1096, 2016. doi:10.1021/acs.jctc.5b01046.

[142] Pavlo O Dral, Xin Wu, Lasse Spörkel, Axel Koslowski, and Walter Thiel. Semiempirical quantum-chemical orthogonalization-corrected methods: Benchmarks for ground-state properties. *Journal of Chemical Theory and Computation*, 12(3):1097–1120, 2016. doi:10.1021/acs.jctc.5b01047.

[143] Richard C Bingham, Michael JS Dewar, and Donald H Lo. Ground states of molecules. xxvi. mindo/3 calculations for hydrocarbons. *Journal of the American Chemical Society*, 97(6):1294–1301, 1975. doi:10.1021/ja00839a002.

[144] James J.P. Stewart. Optimization of parameters for semiempirical methods VI: more modifications to the NDDO approximations and re-optimization of parameters. *Journal of Molecular Modeling*, 19(1):1–32, 2013. doi:10.1007/s00894-012-1667-x.

[145] B Ruscic and D.H. Bross. Active thermochemical tables (ATcT) values based on ver. 1.122 of the thermochemical network, 2016. *avaliable at ATcT.anl.gov*, 2017.

[146] EW Lemmon, MO McLinden, DG Friend, P Linstrom, and W Mallard. NIST chemistry webbook, NIST standard reference database number 69. *National Institute of Standards and Technology, Gaithersburg*, 2011.

[147] James J.P. Stewart. MOPAC2016. Stewart Computational Chemistry, Colorado Springs, CO. *available at http://openmopac.net*, 2016.

[148] Sidney W Benson, FR Cruickshank, DM Golden, Gilbert R Haugen, HE O'Neal, AS Rodgers, Robert Shaw, and R Walsh. Additivity rules for the estimation of thermochemical properties. *Chemical Reviews*, 69(3):279–324, 1969. doi:10.1021/cr60259a002.

[149] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: Computational Statistics*, 2(4):433–459, 2010. doi:10.1002/wics.101.

[150] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[151] Christopher M Bishop. *Pattern recognition and machine learning.* Springer, 2006.

[152] K Sargsyan, HN Najm, and R Ghanem. On the statistical calibration of physical models. *International Journal of Chemical Kinetics*, 47(4):246–276, 2015. doi:10.1002/kin.20906.

[153] Rebecca E Morrison, Todd A Oliver, and Robert D Moser. Representing model inadequacy: A stochastic operator approach. *SIAM/ASA Journal on Uncertainty Quantification*, 6(2):457–496, 2018. doi:10.1137/16M1106419.

[154] Khachik Sargsyan, Xun Huan, and Habib N Najm. Embedded model error representation for Bayesian model calibration. *arXiv preprint arXiv:1801.06768*, 2018.

[155] Jennỳ Brynjarsdóttir and Anthony O'Hagan. Learning about physical parameters: The importance of model discrepancy. *Inverse Problems*, 30(11):114007, 2014. doi:10.1088/0266-5611/30/11/114007.

[156] Yanyan He and Dongbin Xiu. Numerical strategy for model correction using physical constraints. *Journal of Computational Physics*, 313:617–634, 2016. doi:10.1016/j.jcp.2016.02.054.

[157] Hai Wang and Michael Frenklach. Enthalpies of formation of benzenoid aromatic molecules and radicals. *The Journal of physical chemistry*, 97(15):3867–3874, 1993. doi:10.1021/j100117a038.

[158] Thomas C Allison and Donald R Burgess Jr. High-quality thermochemistry data on polycyclic aromatic hydrocarbons via quantum chemistry. *Polycyclic Aromatic Compounds*, 35(1):16–31, 2015. doi:10.1080/10406638.2014.892890.

[159] Ahmad Rushdi, Laura P Swiler, Eric T Phipps, Marta D'Elia, and Mohamed S Ebeida. VPS: Voronoi Piecewise Surrogate models for high-dimensional data fitting. *International Journal for Uncertainty Quantification*, 7(1), 2017. doi:10.1615/Int.J.UncertaintyQuantification.2016018697.

[160] UK Sarkar, PP Chakrabarti, Sujoy Ghose, and SC DeSarkar. Improving greedy algorithms by lookahead-search. *Journal of Algorithms*, 16(1):1–23, 1994. doi:10.1006/jagm.1994.1001.

[161] I. Vernon, M. Goldstein, and R. Bower. Galaxy formation: a Bayesian uncertainty analysis. *Bayesian Analysis*, 5(4):619–670, 2010. doi:10.1214/10-BA524.

[162] I. Vernon, M. Goldstein, and R. Bower. Galaxy formation: Bayesian history matching for the observable universe. *Statistical Science*, 29(1):81–90, 2014. doi:10.1214/12-STS412.

[163] Christian G Bucher. Adaptive sampling — an iterative fast monte carlo procedure. *Structural safety*, 5(2):119–126, 1988.

[164] Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223–242, 04 2001. doi:10.2307/3318737.

[165] Frederick Mosteller and John W Tukey. Data analysis, including statistics. *Handbook of social psychology*, 2:80–203, 1968.

[166] T Hastie, R Tibshirani, and J Friedman. *The Elements of Statistical Learning, vol. 2.* Springer Series in Statistics, 2009. doi:10.1007/978-0-387-84858-7.

[167] T Asaba, WC Gardiner, and RF Stubbeman. Shock-tube study of the hydrogen-oxygen reaction. In *Symposium (International) on Combustion*, volume 10, pages 295–302. Elsevier, 1965. doi:10.1016/S0082-0784(65)80175-8.

[168] N Chaumeix, S Pichon, F Lafosse, and C.E. Paillard. Role of chemical kinetics on the detonation properties of hydrogen/natural gas/air mixtures. *International Journal of Hydrogen Energy*, 32(13):2216–2226, 2007. doi:10.1016/j.ijhydene.2007.04.008.

[169] RK Cheng and AK Oppenheim. Autoignition in methane hydrogen mixtures. *Combustion and Flame*, 58(2):125–139, 1984. doi:10.1016/0010-2180(84)90088-9.

[170] A Cohen and J Larsen. BRL report no. 1386. *Ballistics Research Laboratories, Aberdeen, Maryland*, 1967.

[171] S Fujimoto and M Suzuki. The induction period of hydrogenoxygen and methane-oxygen mixtures in a shock tube. *Memoirs Defense Academy, Japan*, 8(3):1037–1046, 1967.

[172] Clemens Naumann, Jürgen Herzler, Peter Griebel, Henry Curran, Alan Kéromnès, and Ioannis Mantzaras. Results of ignition delay times for hydrogen-rich and syngas fuel mixtures measured, Deliverable 1.1.3, H2-IGCC. *Technical Report*, 2011.

[173] EL Petersen, DM Kalitan, and MJA Rickard. Chemical kinetics of OH$^*$ chemiluminescence in high-temperature reacting flows. In *Proceedings of the Joint Meeting of the US Sections of the Combustion Institute*, volume 3, 2003.

[174] EL Petersen, DF Davidson, M Roehrig, and RK Hanson. Shock-induced ignition of high-pressure $H_2$-$O_2$-Ar and $CH_4$-$O_2$-Ar mixtures. In *AIAA, ASME, SAE, and ASEE, Joint Propulsion Conference and Exhibit, 31 st, San Diego, CA*, 1995.

[175] GL Schott and JL Kinsey. Kinetic studies of Hydroxyl radicals in shock waves. II. Induction times in the Hydrogen-Oxygen reaction. *The Journal of Chemical Physics*, 29(5):1177–1182, 1958. doi:10.1063/1.1744674.

[176] Gordon B Skinner and Gordon H Ringrose. Ignition delays of a Hydrogen—Oxygen—Argon mixture at relatively low temperatures. *The Journal of Chemical Physics*, 42 (6):2190–2192, 1965. doi:10.1063/1.1696266.

[177] SD Tse, DL Zhu, and CK Law. Morphology and burning rates of expanding spherical flames in $H_2$/$O_2$/inert mixtures up to 60 atmospheres. *Proceedings of the Combustion Institute*, 28(2):1793–1800, 2000. doi:10.1016/S0082-0784(00)80581-0.