NOTESTINE, JESSICA GINEPRO. Sensitivity and Active Subspace Analysis for Agent-Based Models. (Under the Direction of Ralph Smith).

Agent-based modeling is a discrete stochastic framework used to simulate the actions and interactions of individual agents to recreate and predict complex phenomena. It has applications across many disciplines, including computer science, economics, biology, ecology, and social networks [15].

An agent-based model consists of a collection of agents and rules. Agents can take on a (typically) finite number of states. A collection of rules describes an agent’s actions and interactions with other agents. These rules determine an agent’s state at a given time based on its previous state and the current states of the other agents with whom it interacts. The rules in an agent-based model can be deterministic or stochastic.

Agent-based models often have many model parameters. Thus sensitivity analysis can play a large role in reducing model complexity by determining which parameters are most influential to a quantity of interest. We will discuss current sensitivity analysis methods for agent-based models and propose implementation of active subspace analysis for agent-based models. Current methods we will consider are one-at-a-time sensitivity analysis, Pearson correlation coefficients, Sobol’ sensitivity analysis, and Morris screening. We propose utilizing active subspaces to compute activity scores and create reduced-order surrogate models for agent-based models. Additionally, we utilize parameter subset selection to determine identifiable subsets of parameters at the nominal values.

To perform sensitivity analysis on agent-based models, surrogate models are often employed. It is common to use polynomial, Gaussian process, or neural network surrogates. We will compare the accuracy of these three choices for surrogate models and utilize them for sensitivity analysis of two chosen agent-based models.

We implement sensitivity analysis techniques on a small scale agent-based model with 2 and 5 parameters and compare results of active subspace analysis and parameter subset selection to those of current sensitivity analysis methods for agent-based models. We also implement this on a medium-scale model with 40 parameters and compare results of active subspace analysis and parameter subset selection with those of current sensitivity analysis methods employed for agent-based models.

One can employ Bayesian inference on surrogates of agent-based models to infer distributions of parameters. We propose utilizing surrogate models computed on active subspaces for Bayesian inference. We implement this on an agent-based model with 2 and 5
parameters and compare results with implementation on full-space surrogate models. Additionally, we utilize Bayesian inference on active subspaces for the influential parameters of a 40 parameter agent-based model.
Sensitivity and Active Subspace Analysis
for Agent-Based Models

by
Jessica Ginepro Notestine

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Mathematics

Raleigh, North Carolina

2022

APPROVED BY:

Mansoor Haider
Hien Tran
Brian Reich
Ralph Smith
Chair of Advisory Committee
DEDICATION

This work is dedicated to my family for their love and support in my academic endeavours, to my husband who always encouraged me when I was lacking confidence in my abilities, and to my sons that they may know they can accomplish great things when they put the work in.
BIOGRAPHY

Jessica Ginepro Notestine grew up in Western Massachusetts and received her Bachelors degree from Western New England University in 2014. She obtained her Masters degree from the University of Connecticut in 2016 and taught high school for two years. She then moved to Raleigh, North Carolina to pursue her PhD at North Carolina State University.
ACKNOWLEDGEMENTS

I would like to thank all those at North Carolina State University who have inspired me to grow as a mathematician. To my dissertation advisor Dr. Ralph Smith for guiding and encouraging me throughout the last three and a half years. You have been reliable and patient through my PhD journey. Thank you also for being a caring and supportive presence during the times when I needed to put being a mother ahead of working. Thank you to Dr. Mansoor Haider, Dr. Brian Reich, and Dr. Hien Tran for serving on my committee.

I would like to thank all those at Western New England University who first instilled a love for mathematics in me, especially Tom Hull who encouraged me to get into mathematics research and supported my goal of graduate school and continues to be a guiding source in my career.
# TABLE OF CONTENTS

Chapter 1  Introduction  ........................................................................................................... 1

Chapter 2  ODD Protocols and Summary of Models  ................................................................. 6
  2.1  ODD Protocol  .................................................................................................................. 6
  2.2  Rebellion NetLogo Model  ............................................................................................... 11
  2.3  Greenhouse NetLogo Model  ............................................................................................ 14

Chapter 3  Background Material on Surrogate Modeling  ......................................................... 22
  3.1  Multivariate Polynomial Regression  ................................................................................ 23
  3.2  Bayesian Information Criterion (BIC)  ............................................................................ 24
  3.3  Gaussian Process Regression  .......................................................................................... 25
  3.4  Neural Networks  ............................................................................................................ 29

Chapter 4  Background Material on Sensitivity Analysis and Active Subspace Methods  ........ 35
  4.1  Local Sensitivity Analysis  ............................................................................................... 36
    4.1.1  Normalized Gradient Approximation  ..................................................................... 37
  4.2  Parameter Subset Selection  ............................................................................................. 39
  4.3  Global Sensitivity Analysis  ............................................................................................. 40
    4.3.1  One-at-a-time Sensitivity Analysis  ......................................................................... 40
    4.3.2  Correlation Coefficients  ......................................................................................... 41
    4.3.3  Sobol’ Sensitivity Indices  ....................................................................................... 41
    4.3.4  Morris Screening  .................................................................................................... 46
  4.4  Active Subspace Analysis  ............................................................................................... 47
    4.4.1  Active Subspace Analysis for Linearly Parameterized Problems  ................. 49
    4.4.2  Active Subspace Analysis for Nonlinearly Parameterized Problems  ......... 49
    4.4.3  Activity Scores  ...................................................................................................... 52
    4.4.4  Time-Dependent Models  ....................................................................................... 52
  4.5  Energy Statistics  ............................................................................................................ 52

Chapter 5  Sensitivity and Active Subspace Analysis  ............................................................... 54
  5.1  Rebellion Model with \( p = 2 \) Parameters  ..................................................................... 54
    5.1.1  Direct Analysis  ....................................................................................................... 56
    5.1.2  Surrogate Models  ................................................................................................... 59
    5.1.3  Sensitivity Analysis  ................................................................................................. 66
  5.2  Rebellion Model with \( p = 5 \) Parameters  ..................................................................... 75
    5.2.1  Surrogate Models  ................................................................................................... 75
    5.2.2  Sensitivity Analysis  ................................................................................................. 82
  5.3  Greenhouse Model  ......................................................................................................... 92
    5.3.1  Surrogate Models  ................................................................................................... 94
    5.3.2  Sensitivity Analysis  ................................................................................................. 98
  5.4  Conclusions for Both Models  ....................................................................................... 109
Agent-based modeling is a discrete stochastic framework used to simulate the actions and interactions of individual agents to recreate and predict complex phenomena. It has applications across many disciplines, including computer science, economics, biology, ecology, and social networks [15].

An agent-based model consists of agents and rules. Agents can take on a (typically) finite number of states. A collection of rules describes an agent's actions and interactions with other agents. These rules determine an agent's state at a given time based on its previous state and the current states of the other agents with which it interacts. The rules in an agent-based model can be deterministic or stochastic. Agent-based models contain Markov processes at the micro-level, since each agent computes a new state based on its current state.

Agent-based models can be implemented via discrete event simulations or time-stepped simulations. Discrete event simulations are used to execute events in a specified order and do not require the use of equal time increments as we move forward in time. Time-stepped simulations require moving the simulation clock forward by the same amount at every time step. We focus on time-stepped simulations since they correspond with the design of NetLogo [37]. NetLogo is a program designed to run and gather data from agent-based models which we will utilize in this dissertation. For time-stepped simulations, we can
have synchronous and asynchronous updates. With synchronous updates, all the agents’ states are updated simultaneously. With asynchronous updates, one updates agents in a particular (potentially random) order. In NetLogo, we use the ask command to control agents, which updates agents in a random order for each time step [24].

To illustrate, consider the SIR (susceptible, infectious, recovered) model

\[
\begin{align*}
\frac{dS}{dt} &= \delta N - \delta S - \gamma k IS \\
S(0) &= S_0 \\
\frac{dI}{dt} &= \gamma k IS - (r + \delta)I \\
I(0) &= I_0 \\
\frac{dR}{dt} &= r I - \delta R \\
R(0) &= R_0
\end{align*}
\]

with the parameters \( \theta = [\gamma, k, r, \delta]^T \), which are the infection coefficient, interaction coefficient, recovery rate, and birth/death rate, respectively. This is an ordinary differential equation (ODE) model that encapsulates the spread of a disease. With this model, one typically sees an outburst of infections, followed by the majority of the population becoming recovered.

Similarly, one could consider the stochastic SIR model, where individuals are represented discretely and, rather than having continuous flow from \( S \) to \( I \) and \( I \) to \( R \), one has discrete transitions. The rules for the stochastic SIR model are given by Table 1.1. The stochastic SIR model is a continuous-time Markov process where the state space for \( S \) and \( I \) is \( \{0, 1, 2, \ldots\} \) [16]. The stochastic SIR model is similar to an agent-based model in that individuals are represented discretely. Further, each individual has a probability of moving from one state to another for a given time interval. However, the stochastic SIR model does not incorporate the interactions between individuals as is done in an agent-based model.

We now discuss an agent-based model based on the SIR model. First, the designer of the agent-based model would specify a map size corresponding to the area they want to

Table 1.1 Transition rules for the stochastic SIR model.

<table>
<thead>
<tr>
<th>Event</th>
<th>Transition</th>
<th>Rate at which event occurs</th>
<th>Probability of transition in ([t, t + dt])</th>
</tr>
</thead>
<tbody>
<tr>
<td>Birth</td>
<td>( S \rightarrow S + 1 )</td>
<td>( \mu N )</td>
<td>( \mu N dt )</td>
</tr>
<tr>
<td>Susceptible death</td>
<td>( S \rightarrow S - 1 )</td>
<td>( \mu S )</td>
<td>( \mu S dt )</td>
</tr>
<tr>
<td>Infection</td>
<td>( S \rightarrow S - 1, I \rightarrow I + 1 ) ( \beta SI/N )</td>
<td>( (\beta SI/N) dt )</td>
<td></td>
</tr>
<tr>
<td>Recovery</td>
<td>( I \rightarrow I - 1 )</td>
<td>( \gamma I )</td>
<td>( \gamma I dt )</td>
</tr>
<tr>
<td>Infectious death</td>
<td>( I \rightarrow I - 1 )</td>
<td>( \mu I )</td>
<td>( \mu I dt )</td>
</tr>
</tbody>
</table>
investigate. Next, the designer would create the number of agents they desire on their map. An agent in this case would be a person who could take on states: susceptible, infected, or recovered. A set of rules would then be determined. A basic set of rules for this model is:

- how far an individual can move in one time step,
- determination of an individual’s current state, based on its previous state and the current states of those around it.

For example, if a susceptible individual is interacting with many infected individuals, they are more likely to become infected. Additionally, an infected individual would recover at a specific rate. This could be modeled by requiring an individual to stay sick for a specific number of time steps or by giving them a probability of recovering at each time step. The first option would impose that all individuals recover at the same rate which is not the case in real world applications. In the second approach, one will have more variability in how fast individuals recover, but one can also end up with unrealistic cases where an individual recovers immediately or never recovers. Alternatively, one can combine these approaches by providing a standard recovery time plus some random deviation from that value. The modeler must weigh the benefits and pit falls of these options and choose which will suit their needs best.

We now discuss how models are designed and run in NetLogo. NetLogo uses turtle graphics, which is vector graphics using a relative cursor (the turtle) on a Cartesian plane. NetLogo has four types of agents: turtles, patches, links, and the observer. Patches make up the world and are setup in a grid format. Turtles can move around in the world made up of patches. Links connect individual turtles and the observer can be thought of as looking over the world that’s been setup [37]. Sometimes turtles are the agents that we’re interested in, such as in the SIR example, where we would want to track individuals. Another example of this case would be the DITCH - Diversity and Inter-ethnic marriage: Trust, Culture and Homophily - model discussed in [35], which is an agent-based model used to explore the percentage of cross-ethnic marriages in an urban environment. Patches can also be the agents of interest. An example of this can be found with the TRANSIMS - TRansportation ANalysis SIMulation System - model in [24], which models traffic flow in an urban environment. In this case, the patches are the agents representing the road and their possible states are occupied or unoccupied with a car.

We choose to utilize NetLogo since it is simple to work in and compatible to run medium to large scale models [1]. By medium to large, we indicate that NetLogo runs models with up to 40 parameters. We will utilize the ODD (“overview, design concepts, and details”) protocol from [40] described in detail in Section 2.1 to outline agent-based models.
In Chapter 2, we present the ODD protocol for describing agent-based models and summarize two models that will be used in the rest of the dissertation. In Chapter 3, we present background material on surrogate modeling and, in Chapter 4, we discuss background material on sensitivity analysis methods. In Chapter 5, we create surrogate models and apply sensitivity and active subspace analysis methods and compare results to existing methodologies applied to agent-based models. We use NetLogo 6.0.4 [55], R [39], and MATLAB [32] in our investigation. We utilize MATLAB for active subspace and sensitivity analysis. NetLogo has a built in tool called BehaviorSpace, which allows us to run a model multiple times and collect data for various parameter choices. This tool takes the Cartesian product of chosen parameter choices creating a grid format which is not ideal. To obtain a random sample, we employ the package RNetLogo [51] which allows one to run NetLogo models without a GUI and with random parameter choices. In Chapter 6, we present and apply the Delayed Rejection Adaptive Metropolis (DRAM) algorithm for Bayesian inference for the agent-based model described in Section 2.2.

A primary objective of this dissertation is to provide sensitivity analysis methods for agent-based models having up to \( p = 40 \) parameters. In particular, we provide an overview of current sensitivity analysis methods utilized and expand this to include active subspace methods. Further, we apply Bayesian inference on active subspaces to agent-based models to infer parameter distributions. We create surrogate models for agent-based models to improve computation time and to enable the use of deterministic models. We determine local parameter identifiability for the two agent-based models described in Chapter 2, which determines whether or not inputs \( \theta \) to a surrogate model can be uniquely determined from responses \( y = f(\theta) \). Further, we utilize the methods outlined in Chapter 5 for sensitivity analysis of the two agent-based model described in Chapter 2. The original contributions of this dissertation are:

- Comparing the use of neural networks and Gaussian processes as surrogate models for the chosen agent-based models. This is important because neural networks are quick to train and evaluate. As the dimension of the parameter space increases, the previously used Gaussian process model becomes slower and more computationally expensive. Neural networks help mitigate this "curse of dimensionality". Specifically, as the number of samples increases with a Gaussian process, the run-time complexity is \( O(n^3) \) due to the need for matrix inversion of large covariance matrices [45]. Neural networks do not require inversion of such matrices and handle larger data sets without the computational increase. Previous comparison of surrogate model choices for agent-based models is detailed in [4].
• Applying active subspace analysis to surrogate models for agent-based models to determine parameter influence via activity scores. This is valuable because activity scores are a derivative-based sensitivity analysis method, whereas many other methods, such as Sobol’ indices are variance-based. Further, Sobol’ indices identify a subset of influential parameters, while active subspaces identify a subspace of influential parameters. This permits parameter dimension reduction when parameter influence is on linear subspaces. As we will see in Chapter 5, activity scores also have a lower computational requirement than Sobol’ indices.

• Applying parameter subset selection to surrogate models for agent-based models to determine parameter identifiability at the nominal values.

• Performing Bayesian inference on active subspaces for surrogate models of agent-based models to determine parameter distributions. Bayesian inference for models becomes prohibitive for models with a large parameter space, due to the inclusion of nonidentifiable parameters. Recall, nonidentifiable parameters are parameters that cannot be uniquely determined from model responses. Often, one fixes noninfluential parameters and performs Bayesian inference on influential parameters only. Using active subspaces provides an alternative approach where Bayesian inference in performed on a lower dimensional subspace and mapped back to the full space.
In this Chapter, we detail the overview, design concepts, and details (ODD) protocol and use it to describe two agent-based models which will be utilized in the rest of the dissertation. We introduce the ODD protocol in Section 2.1 and detail the Rebellion and Greenhouse models in Sections 2.2 and 2.3.

2.1 ODD Protocol

The ODD protocol starts with three elements which provide an overview of what the model is about and how it is designed, followed by an element of design concepts that depicts the ABM’s essential characteristics. It ends with three elements that provide the details necessary to make the description complete [40]. These three stages and seven elements are outlined in Table 2.1.

The first element, *purpose and patterns*, is a statement of the question or problem addressed by the model: what system we are modeling, and what we are trying to learn about it.

The element *entities, state variables, and scales* provides an outline of the model, starting
Table 2.1 Overview of the ODD protocol for describing agent-based models.

<table>
<thead>
<tr>
<th>Elements of the ODD Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
</tr>
<tr>
<td>1. Purpose and Patterns</td>
</tr>
<tr>
<td>2. Entities, State Variables, and Scales</td>
</tr>
<tr>
<td>3. Process Overview and Scheduling</td>
</tr>
<tr>
<td>Design Concepts</td>
</tr>
<tr>
<td>4. Design concepts</td>
</tr>
<tr>
<td>• Basic Principles</td>
</tr>
<tr>
<td>• Emergence</td>
</tr>
<tr>
<td>• Adaptation</td>
</tr>
<tr>
<td>• Objectives</td>
</tr>
<tr>
<td>• Learning</td>
</tr>
<tr>
<td>• Prediction</td>
</tr>
<tr>
<td>• Sensing</td>
</tr>
<tr>
<td>• Interaction</td>
</tr>
<tr>
<td>• Stochasticity</td>
</tr>
<tr>
<td>• Collectives</td>
</tr>
<tr>
<td>• Observation</td>
</tr>
<tr>
<td>Details</td>
</tr>
<tr>
<td>5. Initialization</td>
</tr>
<tr>
<td>6. Input Data</td>
</tr>
<tr>
<td>7. Submodels</td>
</tr>
</tbody>
</table>

with what its entities are and what variables and attributes are used to characterize them. Entities refer to the objects represented in the model. Model entities are characterized by their state variables which describe their state at any time. We then define the temporal and spatial scales in our model. Most agent-based models use discrete time steps to represent the passage of time. The spatial scale of a model refers to the grid size utilized and the topology of the grid. In the SIR example from Chapter 1, the model entities are the individual people and their state variables would be their classification into susceptible, infected, or recovered. The spatial scale is usually a standard grid for the SIR example.

We follow this with the element process overview and scheduling which describes the processes that change the state variables of the model entities. When specifying the processes that are explicitly represented in the model, we also specify the model’s schedule: the order in which the processes are executed by the computer.

To describe the design concepts element, the key questions to ask for each component are provided below. These key questions were taken from Table 3.1 in [40]. We note that not all the concepts are important for all agent-based models. Some models may only need to address a few design concepts in detail.
Basic Principles

- What general concepts, theories, hypotheses, or modeling approaches underlie the model's design? How is the model related to previous thinking about the problem it addresses?

- How were these principles incorporated in the model's design? Does the model implement the principles in its design, or address them as a study topic, e.g., by evaluating and proposing alternatives to them?

Emergence

- What are the model's important results and outputs? Which of them emerge from mechanistic representation of the adaptive behavior of individuals, and which are imposed by rules that force the model to produce certain results?

Adaptation

- What adaptive behaviors for agents have and why? In what ways can they respond to changes in their environment and themselves? What decisions do they make?

- How are these behaviors modeled? Do submodels of adaptive behavior assume agents choose among alternatives by explicitly considering which is most likely to increase some specific objective (direct objective-seeking), or do they simply force agents to reproduce patterns observed in real systems (indirect objective seeking)?

Objectives

- For adaptive behaviors modeled as direct objective-seeking, what measure or agent objectives is used to rate decision alternatives? This objective measure is the agent's internal model of how it would benefit from each choice it might make. What elements of future success are in the objective measure? How does the objective measure represent processes that link adaptive behaviors to important variables of the agents and their environment?

- How were the variables and mechanisms in the objective measure chosen, considering the model's purpose and the real system it represents? How is the agent's current internal state considered in modeling decisions? Does the objective measure change as the agent changes?
Learning

• Do individuals change how they make adaptive decisions over time as a consequence of their experience? If so, how?

Prediction

• How do agents predict future conditions (environmental and internal) in their sub-models for adaptive behavior? What assumptions about, or mechanisms of, the real individuals being modeled were the basis for how prediction is modeled?

• How does simulated prediction make use of mechanisms such as memory, learning, or environmental cues? Or is prediction, “tacit”, i.e., only implied in simple rules for adaptive behavior?

Sensing

• What variables of their environment and themselves are agents assumed to sense and therefore be able to consider in their behavior? What is the basis for these assumptions?

• What sensing mechanisms are modeled explicitly, and which sensed variables are agents instead assumed simply to “know”?

• With what accuracy or uncertainty are agents assumed to “know” or sense which variables? Over what distances?

Interaction

• How do the model's agents interact? Do they interact directly with each other (e.g., does one agent directly change the state of others)? Or is interaction mediated, such as via competition for a resource?

• With which other agents does an agent interact?

• What real interaction mechanisms were the model's representation of interaction based on? At what spatial and temporal scales do they occur?
Stochasticity

- How are stochastic processes (based on pseudorandom numbers) used in the model and why? Are stochastic processes used: To initialize the model? Because it is believed important for some processes to be variable but unimportant to represent the causes of variability? To reproduce observed behaviors using empirically determined probabilities?

Collectives

- Are collectives - aggregations of agents that affect the state or behavior of member agents and are affected by their members - represented in the model?

- If so, how are collectives represented? Do they emerge from the behaviors of agents, or are agents given behavior submodels that impose the formation of collectives? Or are the collectives modeled as another type of agent with its own behaviors and state variables?

Observation

- What outputs from the model are needed to observe its internal dynamics as well as its system-level behavior? What tools (graphics, file output, data on individuals, etc.) are needed to obtain these outputs?

- What outputs and analyses are needed to test the model against the criteria for usefulness - usually, a set of patterns - defined in the “Purpose and patterns” element? What outputs are needed to solve the problem the model was designed for?

For the initialization element, initial conditions are provided that describe how to setup the model world at the beginning of the simulation.

Some models include environmental variables which are read into the model - as opposed to simulated. If a model requires this component, it is described in the input data element.

Lastly, in the submodels element all the major processes in the model are described in detail. Each process is considered to be a submodel.

In the next two sections, we utilize the ODD protocol to describe two agent-based models that will be utilized for the rest of the dissertation.
2.2 Rebellion NetLogo Model

We use a modified version of the Rebellion [56] model from the NetLogo library. This model is designed to track the rebellion of a subjugated population against a central authority. In design, it is similar to the model developed in [15], which tracks outbreaks of crime in a city.

Purpose and Patterns

This model was designed to simulate the rebellion of a subjugated population against a central authority. It is an adaptation of Joshua Epstein's model of civil violence [14, 56].

Entities, State Variables, and Scales

This model contains two types of agents: citizens and law enforcement officers. The environment is a 2-dimensional grid. The global parameters and citizen attributes are provided in Tables 2.2 and 2.3, respectively. Our goal is to apply sensitivity and active subspace analyses to the parameters in Table 2.2 to determine influential parameter subsets.

Additionally, we specify a grid size of $50 \times 50$, and global variables $k = 2.3$ and threshold $= 0.1$ which are the arrest probability constant and population threshold, respectively. The model includes an option to keep movement on or turn movement off.

Process Overview and Scheduling

The process schedule for each time step is as follows:

1. If movement is on, all citizens and law enforcement officers move to a random site within their vision: Rule M - to be described in submodels. If movement is off, only law enforcement officers move to a random site within their vision.

Table 2.2 Global parameters for the Rebellion model. The default values are based on realistic behaviors for this type of city.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value Range</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Government Legitimacy (L)</td>
<td>0-1</td>
<td>0.8</td>
</tr>
<tr>
<td>Initial Agent Density</td>
<td>0-100</td>
<td>70</td>
</tr>
<tr>
<td>Vision</td>
<td>0-10</td>
<td>7</td>
</tr>
<tr>
<td>Maximum Jail Term</td>
<td>0-50</td>
<td>30</td>
</tr>
<tr>
<td>Initial Officer Density</td>
<td>0-100</td>
<td>4</td>
</tr>
</tbody>
</table>
Table 2.3 Citizen attributes for the Rebellion model.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value/Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardship (H)</td>
<td>0-1</td>
</tr>
<tr>
<td>Risk Aversion</td>
<td>0-1</td>
</tr>
<tr>
<td>Grievance</td>
<td>H (1-L)</td>
</tr>
</tbody>
</table>

2. All citizens determine whether they should become active (rebellious) or remain quiet (peaceful/law-abiding): Rule A - to be described in submodels.

3. Law enforcement officers arrest a random active citizen within their vision: Rule C - to be described in submodels.

4. Jailed citizens get their term reduced at the end of each clock tick.

5. Update the display, advance the clock, and update plots.

Design Concepts

This model was designed to simulate the rebellion of a subjugated population against a central authority. It is an adaptation of Joshua Epstein’s model of civil violence. [14, 56] The outputs considered are the number of active (rebellious) citizens and the number of jailed citizens at each time step.

Citizens respond to the number of active citizens and the number of law enforcement officers within their vision. They use this information to determine whether they will become active or quiet at a given point in time. Law enforcement officers respond to the active citizens within their vision by arresting one, if any are present. Individuals do not change how they make decisions over time.

Initialization

The initialization of the Rebellion model is as follows:

1. The global variables \( k = 2.3 \) and threshold = 0.1 are set.

2. Each patch calculates its neighborhood. Doing this with the patches saves computation time at each time step when agents need to “see” within their vision radius.

3. Law enforcement officers are created and placed randomly on the world.
4. Citizens are created and placed randomly on the world. They have the following attributes set:

- Risk aversion is taken from a uniform distribution on $[0,1]$.
- Hardship is taken from a uniform distribution on $[0,1]$.
- All citizens are set to quiet at the start.
- All citizens have a current jail term of 0.

5. The clock is started and the initial state of the system is plotted.

**Input data**

No input data is provided in this model.

**Submodels**

Recall the global parameters for the Rebellion agent-based model from Table 2.2. Government legitimacy quantifies the perceived legitimacy of the government, initial agent density provides the proportion of patches that will be occupied by a citizen, vision quantifies how far an agent - citizen or officer - can see, maximum jail term is the maximum number of time steps a citizen can be jailed for, and initial officer density describes the proportion of patches that will be occupied by a law enforcement officer.

**Rule M (Move)**

Citizens and law enforcement officers move to an empty patch within their vision. Note that there is an option in this model to turn movement off. Doing so prevents citizens from moving; however, law enforcement officers will still be allowed to move around the world.

**Rule A (Determine Behavior)**

Let grievance, risk aversion, threshold, and estimated-arrest-probability be denoted by $G$, $R$, $T$, and $P$, respectively. Then if

$$G - R \cdot P > T,$$

a citizen’s state is set to active (rebellious). Recall, $G = H(1 - L)$, where $H$ is hardship and $L$ is government legitimacy.
**Figure 2.1** Red patches show the patches within the vision radius of the black patch.

**Estimated-arrest-probability**

Let $C$ be the number of law enforcement officers within a citizen's vision. Let $A$ be the number of active citizens within a citizen's vision. Then the estimated-arrest-probability is computed as

$$P = 1 - \exp(-k \cdot \lfloor C/A \rfloor).$$

**Rule C (Enforce)**

Law enforcement officers look for an active citizen within their vision, select one at random, and move to that citizen's location. The selected citizen's status is set to quiet, they are removed from the population, and assigned a jail term from $\mathcal{U}(0, J_{max})$, where $J_{max}$ is the maximum jail term global parameter.

**2.3 Greenhouse NetLogo Model**

**Purpose and Patterns**

Epistatic effects in a system refer to when the overall effect cannot be explained from the different components of the system but exists within the interaction between the components. The purpose of the model is to gain insight regarding the possible impact of epistatic effects on the performance of greenhouse owners and the effects different development styles have on the performance of greenhouses [21].
Entities, State Variables, and Scales

This model has one type of agent: greenhouse owners. Greenhouse owners attempt to upgrade their technologies to increase their control on the internal climate of the greenhouse. The greenhouse owners want to maximize their profit or yield.

Each greenhouse will either cultivate Veggies or Flowers. For both crop types, there is an ideal climate under which this crop type will produce the highest fraction of their “Potential yield”. The crop types have the properties summarized in Table 2.4.

Each greenhouse consists of four different technologies, each from a different Technology category. The technologies interact together to reach the internal climate where the crop production is optimal. Each technology affects a greenhouse’s temperature, CO2 levels, humidity, and light levels. The corresponding global parameters are provided in Table 2.5. Negative and positive values correspond to a negative and positive influence on that greenhouse aspect, respectively.

Every greenhouse owner forms an opinion about the products available on the market. This opinion depends on their own experience and that of their neighbours multiplied by a stubbornness factor.

Profits are added to a greenhouse owner’s bank balance and expenses on technologies

Table 2.4 Global parameters related to crop types for the Greenhouse model.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value Range</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Veggies</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ideal Temperature (in °C)</td>
<td>0-40</td>
<td>26</td>
</tr>
<tr>
<td>Ideal CO2 concentration (in ppm)</td>
<td>1000-2000</td>
<td>1500</td>
</tr>
<tr>
<td>Ideal humidity (in %)</td>
<td>0-100</td>
<td>75</td>
</tr>
<tr>
<td>Ideal Light Intensity (in lux)</td>
<td>1000-2000</td>
<td>1400</td>
</tr>
<tr>
<td>Potential yield number</td>
<td>0-1000</td>
<td>900</td>
</tr>
<tr>
<td>Market price (in €/crop unit)</td>
<td>0-20</td>
<td>15</td>
</tr>
<tr>
<td>Purchase price (in €/crop unit)</td>
<td>0-20</td>
<td>3</td>
</tr>
<tr>
<td><strong>Flowers</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ideal Temperature (in °C)</td>
<td>0-35</td>
<td>24</td>
</tr>
<tr>
<td>Ideal CO2 concentration (in ppm)</td>
<td>1000-2000</td>
<td>1200</td>
</tr>
<tr>
<td>Ideal humidity (in %)</td>
<td>0-100</td>
<td>87</td>
</tr>
<tr>
<td>Ideal Light Intensity (in lux)</td>
<td>1000-2000</td>
<td>1700</td>
</tr>
<tr>
<td>Potential yield number</td>
<td>0-1000</td>
<td>900</td>
</tr>
<tr>
<td>Market price (in €/crop unit)</td>
<td>0-20</td>
<td>15</td>
</tr>
<tr>
<td>Purchase price (in €/crop unit)</td>
<td>0-20</td>
<td>3</td>
</tr>
</tbody>
</table>
Table 2.5 Global parameters related to technologies for the Greenhouse model.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value Range</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Technology A (Heater)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temperature Min</td>
<td>0-1</td>
<td>0.5</td>
</tr>
<tr>
<td>CO2 Max</td>
<td>0-0.5</td>
<td>0.05</td>
</tr>
<tr>
<td>Humidity Min</td>
<td>-0.5-0</td>
<td>-0.05</td>
</tr>
<tr>
<td>Light Maximum</td>
<td>0-0.1</td>
<td>0</td>
</tr>
<tr>
<td><strong>Technology B (CO2 Technology)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temperature Max</td>
<td>0-0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>CO2 Min</td>
<td>0-1</td>
<td>0.5</td>
</tr>
<tr>
<td>Humidity Min</td>
<td>-0.3-0</td>
<td>-0.05</td>
</tr>
<tr>
<td>Light Min</td>
<td>-0.3-0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Technology C (Humidifier)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temperature Max</td>
<td>0-0.2</td>
<td>0.05</td>
</tr>
<tr>
<td>CO2 Max</td>
<td>0-0.1</td>
<td>0</td>
</tr>
<tr>
<td>Humidity Min</td>
<td>0-1</td>
<td>0.5</td>
</tr>
<tr>
<td>Light Min</td>
<td>-0.3-0</td>
<td>-0.05</td>
</tr>
<tr>
<td><strong>Technology D (Lights)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temperature Max</td>
<td>0-0.1</td>
<td>0.05</td>
</tr>
<tr>
<td>CO2 Max</td>
<td>0-0.2</td>
<td>0</td>
</tr>
<tr>
<td>Humidity Min</td>
<td>-0.05</td>
<td>-0.2-0</td>
</tr>
<tr>
<td>Light Min</td>
<td>0-1</td>
<td>0.5</td>
</tr>
</tbody>
</table>

and energy use are extracted from it. If the bank balance is negative, the greenhouse owner goes bankrupt and will be replaced. The parameters related to the Greenhouse owners are provided in Table 2.6.

There are five parameters related to external circumstances, provided in Table 2.7. The remaining global parameters are provided in Table 2.8 and their use in the model will be shown in the submodels.

**Process Overview and Scheduling**

The process schedule for each time step is as follows:

Table 2.6 Global parameters related to greenhouse owners for the Greenhouse model.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value Range</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stubbornness Factor</td>
<td>0.7-0.9</td>
<td>0.7</td>
</tr>
<tr>
<td>Initial Bank Account</td>
<td>0-100,000</td>
<td>70,000</td>
</tr>
</tbody>
</table>
Table 2.7 Global parameters related to external circumstances for the Greenhouse model.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value Range</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>External Temperature</td>
<td>0-100</td>
<td>17</td>
</tr>
<tr>
<td>External CO2</td>
<td>200-2000</td>
<td>400</td>
</tr>
<tr>
<td>External Humidity</td>
<td>50-100</td>
<td>74</td>
</tr>
<tr>
<td>External Light</td>
<td>800-2000</td>
<td>1000</td>
</tr>
<tr>
<td>Fuel Price</td>
<td>0.4-1.0</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Table 2.8 Remaining global parameters for the Greenhouse model.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value Range</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost Price Cooperative Multiplier</td>
<td>0-2</td>
<td>1.3</td>
</tr>
<tr>
<td>Influence Range Secondary</td>
<td>0-0.05</td>
<td>0.04</td>
</tr>
<tr>
<td>Cost Price Range</td>
<td>0-1000</td>
<td>400</td>
</tr>
<tr>
<td>Energy Use Min</td>
<td>0-5000</td>
<td>0</td>
</tr>
<tr>
<td>Cost Price Min</td>
<td>1000-1500</td>
<td>1500</td>
</tr>
<tr>
<td>Lifespan Min</td>
<td>0-4</td>
<td>3</td>
</tr>
<tr>
<td>Influence Range Main</td>
<td>0-0.1</td>
<td>0.02</td>
</tr>
<tr>
<td>Improve Same Tech Count</td>
<td>0-5</td>
<td>4</td>
</tr>
<tr>
<td>Energy Use Max</td>
<td>5,000-10,000</td>
<td>5,000</td>
</tr>
<tr>
<td>Cost Price Max</td>
<td>1500-2000</td>
<td>1500</td>
</tr>
<tr>
<td>Lifespan Max</td>
<td>4-10</td>
<td>5</td>
</tr>
</tbody>
</table>

1. Greenhouse owners adapt tech counters, calculate production, calculate total energy use, update their bank account, and revise opinions. Each of these actions corresponds to a submodel.

2. Technologies are improved where appropriate.


4. Technologies lifetime is adjusted for the step and the visual is updated.

**Design Concepts**

The purpose of the model is to gain insight into the possible impact of epistatic effects on the performance of greenhouse owners and the effects different development styles have on the performance of greenhouses. The sum of all actual production is the macroscopic quantity of interest; it's behavioral pattern emerges from the interaction between technologies. [21]

Greenhouse owners interact and respond to opinions of other Greenhouse owners, depending on the development style specified. The default development style is protected
development. They use these opinions, in addition to their own opinion, to update technologies.

**Initialization**

The initialization of the Greenhouse model is as follows:

1. The heaters, CO2 technologies, humidifiers, and lights are created.
2. The technology markets are listed out.
3. The improve same tech counter is set to a random integer between 0 and 4.
4. The greenhouse owners are created and greenhouses are setup.

**Input data**

No input data is provided in this model.

**Submodels**

**Create Greenhouse Network**

Taken from model library "Preferential Attachment", this submodel creates a network where a few "hubs" have lots of connections while everyone else only has a few. This is done by having new network members prefer to make a connection to the more popular existing members [57].

**Adapt Tech Counters**

This submodel keeps track of the number of technologies that are bought.

**Calculate Production**

First, the investments in crop seeds are extracted from the bank account (investments in seeds are: Potential Growth x Purchase Price). Then the sum of all tech influence numbers is calculated. The maximum total influence is 1. At 1, the ideal climate is the same as the internal climate. The relative climate range (CR) is calculated for temperature, CO2, humidity, and light:

\[
CR_r = 1 - (|E - I|/I - F|E - I|/I)
\]
where \( E \) refers to the external amount (of temperature/CO2/humidity/light), \( I \) refers to the ideal amount (of temperature/CO2/humidity/light), and \( F \) is an influence factor. Actual production is then calculated as the (veggies or flowers) potential yield multiplied by the mean of the \( CR_i \)'s. The relative production is the actual production divided by the potential production.

**Calculate Total Energy Use**

The total energy use is calculated as the sum of the the energy needed for each technology in a greenhouse.

**Update Bank Account**

- Set marginal revenue: (market price - purchase price).
- Set total revenue: marginal revenue \( \times \) actual production.
- Set energy costs: total energy use \( \times \) fuel price.
- Set bank account: bank account \( + \) total revenue - total energy costs.

If the greenhouse owner's budget is smaller than zero, they go bankrupt and resets their properties to the initial values, to represent a start-up. The total depreciation is calculated with the sum of all technology costs in use, divided by the initial lifetime of those technologies.

- Set profit: marginal revenue \( \times \) actual production - energy costs - depreciation of technologies.

**Revise Opinions**

For the products currently being used, Set Opinion \( = \) (Current Opinion \( + \) Actual Production / Potential yield) / 2.

**Improve Technologies**

First the tech to improve is chosen at random. The number of times a certain product is improved in a row is defined. If this number is less than ImproveSameTechCounter then we select a tech at random to update, implemented by the submodel SelectTechOptionRandom. Otherwise, if the development style is specified as open source cooperation, the originally chosen technology is improved via the submodel SelectTechOption2, and via the submodel SelectTechOption1, otherwise.
Select Tech Option 1

The highest main function of all the products is stored as MaxTech. This specifies which product is chosen. The submodel ImproveTech is then run.

Select Tech Option 2

The greenhouse owner looks for the best performing greenhouse, and selects their product of the chosen technology. The submodel ImproveTech is then run.

Select Tech Option Random

A random technology is chosen to improve. Then progresses as in submodel SelectTechOption2 if open source cooperation is specified as the development style and as in SelectTechOption1 if not.

Improve Tech

For every selected technology, if the the main function of one of the products of that selected technology is equal to the MaxTech, then the main function is improved and the secondary functions are randomly changed. The cost price of the adapted technology is increased. Otherwise, if the development style is protected development, replace the product with highest main function. Else replace the product with highest sum of all functions.

Share Opinions

Ask all link-neighbours with the same crop type to adapt their opinions based on the opinion of a specified greenhouse owner. For link-neighbour with the same crop type and particular opinion for product X is equal to 0, set $\text{Opinion} = \text{Stubbornness Factor} \times \text{Opinion of myself}$. For link-neighbour with the same crop type and particular opinion for product X is unequal to 0, set $\text{Opinion} = (\text{Opinion} + \text{Stubbornness Factor} \times \text{Opinion of myself}) / (1 + \text{Stubbornness Factor})$.

Purchase New Technologies

For every product check whether the lifespan is equal to zero. If so, replace this product by the product for which the greenhouse owner has the highest opinion. If the bank account is smaller than the cost price of this product, select from all the products which the greenhouse owner can afford. If this list is empty (this means that there is no affordable technology
available) the greenhouse owner is bankrupted and will be replaced by a new one. If this list isn't empty, select a random product and replace the over-aged product with this one.
Due to their stochastic nature, agent-based models are not well suited for sensitivity analysis, active subspace analysis, and Bayesian inference. Each run with the same parameter values and same starting conditions can produce different results. To make sensitivity analysis, active subspace analysis, and Bayesian inference possible, one must first create deterministic surrogate models. This permits application of sensitivity analysis techniques and enables computation of the gradient of the output of the model with respect to the model input parameters. Additionally, this improves computation time allowing for a large number of model evaluations.

In this Chapter, we present the necessary background material for constructing surrogate models. In Sections 3.1, 3.3, and 3.4, we present ordinary least squares regression for multivariate polynomials, Gaussian process regression, and neural networks, respectively. Ordinary least squares regression for multivariate polynomials is simple and fast to train and evaluate, but is not as accurate as neural networks and Gaussian processes. Gaussian processes are advantageous as they treat unresolved model behavior as noise and can yield statistical prediction intervals. Gaussian process regression suffers from the "curse of dimensionality" in that as the number of parameters and training data increases, the time
it takes to train and evaluate a Gaussian process increases rapidly. Neural networks are advantageous as they are a nonlinear modeling technique. Further, neural networks provide a better fit to data than polynomial surrogates do and do not suffer from the same significant increase in computation time as the number of parameters or data points increases that a Gaussian process does.

The observation model for the agent-based model is considered to be of the form

$$y = f(q) + \epsilon,$$  \hspace{1cm} (3.1)

where $f$ corresponds to a deterministic surrogate model, $q$ is the input to the model, and $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Note that $q$ is used as the input to the surrogate model to represent the parameters $\theta$ and, when appropriate, time $t$. That is, for time-dependent surrogate models, $q = [\theta^T, t]^T$. $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is a commonly made assumption. Further discussion about why this is appropriate for the two agent-based models we consider is provided in Chapter 5.

### 3.1 Multivariate Polynomial Regression

One method of surrogate modeling, which we utilize, is multivariate polynomial regression constructed via ordinary least squares optimization. We first summarize ordinary least squares (OLS) for a linear model. Consider a model

$$f(x) = \beta_0 + \beta_1 x$$

fit to data $(x_i, y_i)$, $i = 1, ..., n$. This requires the estimation of $\beta_0$ and $\beta_1$. It is assumed that

$$y_i = f(x_i) + \epsilon_i, \quad i = 1, ..., n,$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ are independent and identically distributed (iid). The likelihood function of this model is

$$\mathcal{L}(\beta_0, \beta_1, \sigma) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - f(x_i))^2}{2\sigma^2}}.$$  

Define the residual sum of squares as

$$\text{RSS}(\beta_0, \beta_1) = \sum_{i=1}^{n} (y_i - f(x_i))^2.$$
To find the minimizing values of $\beta_0$ and $\beta_1$, we take the derivative of RSS with respect to $\beta_0$ and $\beta_1$ and set them equal to zero to obtain

$$\frac{\partial \text{RSS}}{\partial \beta_0} = -2 \sum_{i=1}^{n} (y_i - f(x_i)) = 0,$$

$$\frac{\partial \text{RSS}}{\partial \beta_1} = -2 \sum_{i=1}^{n} x_i (y_i - f(x_i)) = 0.$$

Rearranging these two equations yields

$$\sum_{i=1}^{n} y_i = n \beta_0 + \beta_1 \sum_{i=1}^{n} x_i,$$

$$\sum_{i=1}^{n} x_i y_i = \beta_0 \sum_{i=1}^{n} x_i + \beta_1 \sum_{i=1}^{n} x_i^2,$$

which we solve to obtain the estimates

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i - n \bar{x} \bar{y}}{\sum_{i=1}^{n} x_i^2 - n \bar{x}^2},$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x},$$

where $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ and $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$.

This process is easily expanded to include multiple independent variables as well as to include general nonlinear polynomial regression [53]. Implementation in MATLAB will be performed via the MultiPolyRegress function provided in [7].

### 3.2 Bayesian Information Criterion (BIC)

When fitting a polynomial surrogate, the degree of the employed polynomial must be determined. One way to do this is to utilize the Bayesian Information Criterion (BIC). The Bayesian Information Criterion provides a measure of how well a model fits data while penalizing a model if it is overly complex. We choose to use the BIC over the more commonly know Akaike Information Criterion (AIC) as it penalizes model complexity more heavily. The formula for the AIC is

$$AIC = 2k - 2\ln(\mathcal{L}),$$

where $\mathcal{L} = \mathcal{L}(\hat{\theta})$ is the maximum value of the likelihood function of the model and $k$ is the number of parameters. The likelihood of a model is the probability density of the data, defined as a function of the parameters.
The BIC encapsulates a larger penalty for model complexity by multiplying the number of parameters by the logarithm of the number of observations. The formula for the BIC is

\[ BIC = k \ln n - 2 \ln(L), \]

where \( L = \mathcal{L}(\hat{\theta}) \) is the maximum value of the likelihood function of the model, \( k \) is the number of parameters, and \( n \) is the number of observations. In general, when comparing two models, the model with the lower BIC value is considered to better balance the model's ability to fit the data set and avoid over-fitting simultaneously [32, 33].

### 3.3 Gaussian Process Regression

A second method of surrogate modeling, which we employ, is Gaussian process (GP) regression, known as Kriging in spatial statistics. Recall equation (3.1). The construction of a Gaussian process surrogate is based on the assumption that model evaluations can be treated as realizations of a Gaussian process. This treats unresolved model behavior as noise and yields statistical prediction intervals at test inputs.

**Definition 3.3.1.** A Gaussian process with mean function \( m(q) \) and covariance function, or kernel, \( k(q, q') \) is defined as

\[
m(q) = \mathbb{E}[f(q)],
\]

\[
k(q, q') = \mathbb{E}[(f(q) - m(q))(f(q') - m(q'))].
\]

This is denoted by \( f(q) \sim GP(m(q), k(q, q')) \). Formally, a Gaussian Process is an infinite collection of random variables, any finite number of which have a joint Gaussian distribution [41].

Gaussian process surrogate model construction requires the following steps:

1. Specification of mean and covariance functions;
2. Inference of hyperparameters in \( m(q) \) and \( k(q, q') \);
3. Construction of joint prior and conditional distributions for training data and future observations.

**Mean and Covariance Functions**

The mean and covariance functions of a Gaussian process provide the structure for the Gaussian process. The mean function is specified to incorporate global trends or biases.
ordinary Kriging, the mean function is assumed to be constant with \( m(q) = \beta_0 \).

The choice of covariance function (or kernel) imposes smoothness properties and incorporates correlation structures. Three common choices for these covariance functions are the Squared Exponential, Matern 3/2, and Matern 5/2, given as

\[
\begin{align*}
    k_{SE}(q, q') &= \sigma_f^2 \exp \left[ -\frac{1}{2} \frac{(q - q')^T (q - q')}{\sigma^2} \right] \quad (3.2) \\
    k_{M3/2}(q, q') &= \sigma_f^2 \left( 1 + \frac{\sqrt{3} r}{\sigma} \right) \exp \left( -\frac{\sqrt{3} r}{\sigma} \right) \quad (3.3) \\
    k_{M5/2}(q, q') &= \sigma_f^2 \left( 1 + \frac{\sqrt{5} r}{\sigma} + \frac{5 r^2}{3 \sigma^2} \right) \exp \left( -\frac{\sqrt{5} r}{\sigma} \right), \quad (3.4)
\end{align*}
\]

respectively, where \( r = \sqrt{(q - q')^T (q - q')} \). The hyperparameters to be estimated are \( \sigma_f \) and \( \sigma_\ell \). For models with a large number of inputs, it is often beneficial to utilize different length scales for each predictor. The automatic relevance determination (ARD) Matern 3/2 kernel is given by

\[
k_{AM3/2}(q, q') = \sigma_f^2 \left( 1 + \sqrt{3} r \right) \exp\left( -\sqrt{3} r \right), \quad (3.5)
\]

where

\[
r = \sqrt{\sum_{m=1}^{p} \frac{(q_m - q'_m)^2}{\sigma_m^2}}.
\]

The ARD squared exponential and Matern 5/2 kernels are defined analogously [41].

The squared exponential covariance function is infinitely differentiable and thus provides smooth functions. For the Matern \( \nu \) covariance function, the process will be \( k \)-times differentiable for \( k < \nu \)[41]. That is, the Matern 3/2 kernel is once differentiable and the Matern 5/2 kernel is twice differentiable. We will consider all three of these kernels in Chapter 5.

**Inference of Hyperparameters**

Consider the observation model

\[
y_i = f(q_i) + \epsilon_i, \quad i = 1, ..., N, \quad (3.6)
\]

where \( \epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_0) \); that is, \( \epsilon_i \) are independently and identically distributed with a normal distribution with mean 0 and variance \( \sigma^2_0 \). Then

\[
y = [ f(q_1), ..., f(q_N) ] \sim \mathcal{N}(m, C + \sigma^2_0 I),
\]
where the symmetric positive semidefinite matrix $C \in \mathbb{R}^{M \times M}$ has entries $C_{ij} = k(q_i, q_j)$. The correlation matrix $R$ is defined by the relation $C = \sigma_f^2 R$. For the constant mean function $m = \beta_0 \mathbf{1}$, where $\mathbf{1}$ is the vector of all ones, the likelihood is
\[
L = \frac{1}{(2\pi)^{M/2}\sqrt{C + \sigma_0^2 I}} \exp \left( -\frac{1}{2} (y - m)^T (C + \sigma_0^2 I)^{-1} (y - m) \right)
\]
where $\delta = \frac{\sigma_0}{\sigma_f}$. To estimate $\beta_0$ and $\sigma^2$, the likelihood is maximized. This is equivalent to maximizing the log likelihood
\[
L = \frac{1}{(2\pi\sigma_f^2)^{M/2}|R + \delta^2 I|^{1/2}} \exp \left( -\frac{1}{2\sigma^2} (y - \beta_0 \mathbf{1})^T [R + \delta^2 I]^{-1} (y - \beta_0 \mathbf{1}) \right).
\]

The necessary conditions
\[
\frac{\partial L}{\partial \beta_0} = 0, \quad \frac{\partial L}{\partial \sigma^2} = 0
\]
are enforced to obtain the maximum likelihood estimates
\[
\beta_{0,MLE} = \frac{1}{1^T [R + \delta^2 I]^{-1} [y - \beta_{0,MLE} \mathbf{1}]} y
\]
\[
\sigma_{f,MLE}^2 = \frac{1}{M} [y - \beta_{0,MLE} \mathbf{1}]^T [R + \delta^2 I]^{-1} [y - \beta_{0,MLE} \mathbf{1}].
\]

Substituting (3.8) and (3.9) into (3.7) and removal of constant terms yields
\[
-2L(\sigma_f, \delta) = M \ln(\sigma_{f,MLE}^2(\sigma_f, \delta)) + \ln(|R(\sigma_f) + \delta^2 I|)
\]
which only depends on $\sigma_f$ and $\delta$ and can be minimized to obtain estimates for $\sigma_f$ and $\delta$.

**Predictions and Gradient of Gaussian Process Surrogate**

We first discuss how a Gaussian process makes predictions.

Let $f_\star$ be a random variable representing the function value at a point $q_\star$, $f_\star = f(q_\star)$. The expected function value at $q_\star$ is
\[
\mathbb{E}[f_\star|Q, y, q_\star] = \sum_{i=1}^n \alpha_i k(q_i, q_\star) + \beta_0,
\]
where \( y \) denotes the vector of training outputs, \( Q \) denotes the matrix of training inputs, \( \alpha = C^{-1}(y - \beta_0 1) \), \( \beta_0 \) is the constant mean value, and \( k \) is the covariance function. Note that in the implementation in MATLAB, \( \alpha \) is optimized so that \( \beta_0 = 0 \).

The expected gradient of the Gaussian process can be computed using the additive and constant multiple properties of the gradient to obtain

\[
\nabla_q \mathbb{E}[f|Q, y, q_*] = \nabla_q \sum_{i=1}^{n} \alpha_i k(q_i, q_*) = \sum_{i=1}^{n} \alpha_i \nabla_q k(q_*, q_i),
\]

(3.10)
e.g., [19]. Specifically, for the Matern 3/2 kernel, the expected gradient is provided by (3.10), where

\[
\nabla_q k(q_*, q_i) = \frac{3\sigma_f^2}{\sigma_i^2} (q_i - q_*) \exp \left( -\frac{\sqrt{3} r}{\sigma_i} \right),
\]

which can be used directly with computed weights to calculate the gradient for a Gaussian process surrogate [32].

**Prediction Intervals**

Let \( Q^* = [q_1^*, ..., q_M^*] \) be the \( p \times M \) matrix of test input values and denote \( f^* = [f(q_1^*), ..., f(q_M^*)] \). Then by equation (3.6), \( y^* = f^* + \epsilon^* \) where \( \epsilon^* = [\epsilon_1^*, ..., \epsilon_M]^T \). Define the \( N \times M \) and \( M \times M \) covariance matrices

\[
[C_*]_{ij} = k(q_i, q_j^*), \ [C_{**}]_{ij} = k(q_i^*, q_j^*).
\]

For the matrix of test inputs \( Q^* \), one generates a random Gaussian response from prior distribution

\[
f^* \sim \mathcal{N}(\beta_0 1^*, C_{**}).
\]

(3.11)

To enforce the structure provided by the training data, condition the joint prior distribution on the training responses and predictions

\[
\begin{bmatrix}
    y \\
    f^*
\end{bmatrix} = \begin{bmatrix}
    \beta_0 1 \\
    \beta_0 1^*
\end{bmatrix} \begin{bmatrix}
    C + \sigma_0^2 I & C_* \\
    C_* & C_{**}
\end{bmatrix},
\]

(3.12)

where \( C \) was defined previously as \( C_{ij} = k(q_i, q_j) \) for \( i, j, = 1, ..., N \), and observed training data is \( \{Q = [q_1, ..., q_N], y = [y_1, ..., y_N]\} \). Recall that for (3.6), \( \epsilon \sim \mathcal{N}(0, \sigma_0^2) \). Here \( 1 \) and \( 1^* \) are the \( N \times 1 \) and \( M \times 1 \) vectors of ones. This conditional predictive distribution is

\[
f^*|Q, Q^*, y \sim \mathcal{N}(\mathbb{E}[f^*], \text{cov}[f^*]),
\]

28
with mean and covariance

\[
E[F^*] = \beta_0 I^* + C_\ast^T (C + \sigma_0^2 I)^{-1} (y - \beta_0 I),
\]

\[
\text{cov}[F^*] = C_{ss} - C_\ast^T (C + \sigma_0^2 I)^{-1} C_\ast.
\]

This determines that the test observations are normally distributed,

\[
y_* | Q, Q^*, y \sim \mathcal{N}(E[y_*], \text{cov}[y_*]),
\]

where \(E[y_*] = E[F^*]\) and \(\text{cov}[y_*] = \text{cov}[F^*] + \sigma_0^2 I\). Full details of the derivation of the conditional predictive distribution are provided in [41, 47].

Recall that Gaussian processes are valuable as surrogate models as unresolved model behavior is treated as noise and they yield statistical prediction intervals at test inputs. However, Gaussian processes suffer from the "curse of dimensionality" where computation time increases rapidly as the number of parameters and training data points increases. Gaussian processes perform well on smaller data sets as they have less hyperparameters to train [41, 45].

### 3.4 Neural Networks

Neural networks are a nonlinear modeling technique useful for modeling complex relationships between inputs and outputs and commonly employed for pattern recognition. Neural networks are appropriate for nonlinear regression, as they can approximate any function to a desired accuracy. An explanation of this can be found in [34].

First, we define a neuron. A neuron takes inputs and produces an output by weighting individual inputs, summing them up, and adding a bias. It then passes this value through an activation function. Activation functions are typically sigmoidal and produce a numerical output. This process is depicted in Figure 3.1a. Here inputs are \(x_1\) and \(x_2\) and corresponding weights are \(w_1\) and \(w_2\). The net input function calculates \(z = w_1 x_1 + w_2 x_2 + b\), where \(b\) is a specified bias. This value is then passed through our activation function. A commonly used activation function is

\[
\sigma(z) = \frac{1}{1 + e^{-z}}
\]

depicted in Figure 3.1b, although other functions can be used.

A neural network is a network consisting of layers of neurons. The structure of such a network is depicted in Figure 3.2. The first layer of a neural network is the input layer, where inputs are passed to the first layer of neurons. The second layer is called a hidden layer,
as it is an intermediate step to producing the final output. Within the hidden layer, each neuron weights the inputs, adds the bias, and passes them through an activation function as previously described. The outputs from the hidden layer are then passed to the output layer, where the process is repeated, using the hidden layer outputs as the inputs to the output layer.

What we have described is a shallow neural network, where there is only one hidden layer. Note that it is possible to have more hidden layers in a neural network, and when this occurs the neural network is termed a deep neural network. For simplicity, we focus on shallow neural networks in our discussion and analysis. Additionally, we have that information flows only one way within the neural network. This type of neural network is called a feedforward neural network. Other neural networks, such as recurrent neural networks allow information flow through the network in both directions, and documentation for this can be found in [32]. Again, we focus on the more frequently used feedforward neural network. A feedforward neural network is faster to train, use, and manipulate as there are no feedback loops in the network.

To utilize neural networks, the first step is to define weights and biases, and choose which activation functions to use. Data is then passed through the neural network to train these weights and biases. The more training data available, the better results the neural network will provide. However, it is very easy to overfit data in a neural network. To control overfitting, training data is separated into three categories, training data, validation data, and test data. Training data is used to determine the weights and biases and validation data is used to adjust these values so as to not overfit the training data. The test data is then used as an assessment of how well the neural network fits, by using data not previously employed.
on the neural network. In MATLAB, we use the functions `feedforwardnet` and `train` to initialize and train the neural network from the Deep Learning Toolbox. With the `train` function, we use the default method of training our neural network weights and biases. For feedforward networks in MATLAB the training method is Levenberg-Marquardt backpropagation.

The Levenberg-Marquardt algorithm blends the steepest descent method and the Gauss-Newton algorithm, so we first describe these individually.

**Steepest Descent Algorithm**

The steepest descent algorithm, otherwise known as the error back-propagation algorithm, update rule for weights can be formulated as

\[
    w_{k+1} = w_k - \alpha g_k,
\]

where \( w \) is the vector of weights, \( \alpha \) is the step size, and \( g \) is the gradient of the error function with respect to the weights. Define \( e_{p,m} \) as the training error at output \( m \) when applying pattern \( p \), defined as \( e_{p,m} = d_{p,m} - o_{p,m} \) where \( d \) is the desired output vector and \( o \) is the actual output vector. The error function is defined as

\[
    E(x, w) = \frac{1}{2} \sum_{p=1}^{P} \sum_{m=1}^{M} e_{p,m}^2,
\]
where \( x \) is the input vector, \( w \) is the weight vector, and \( e_{p,m} \) is the training error at output \( m \). The gradient vector is
\[
g = \left[ \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_N} \right]^T,
\]
where \( N \) is the total number of weights in the network. This method has stable but slow convergence and low computational complexity, as only computation of the gradient is needed. The steepest descent algorithm can also jitter between values in certain cases since the step size \( \alpha \) is not updated.

A related method that is commonly employed for training neural networks is stochastic gradient descent. This method follows this process, but uses a random subset of the training data to update the weight vector at each time step, as opposed to using all data provided. See [2] for more details about stochastic gradient descent.

Newton's Method

The update rule for Newton's method is
\[
w_{k+1} = w_k - H_k^{-1}g_k,
\]
where \( H \) is the Hessian matrix
\[
H = \begin{bmatrix}
\frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_N} \\
\frac{\partial^2 E}{\partial w_2 \partial w_1} & \frac{\partial^2 E}{\partial w_2^2} & \cdots & \frac{\partial^2 E}{\partial w_2 \partial w_N} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 E}{\partial w_N \partial w_1} & \frac{\partial^2 E}{\partial w_N \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_N^2}
\end{bmatrix}.
\]
This comes from the assumption that the gradient components \( g_i \) are functions of the weights and that all weights are linearly independent. Under this assumption, Taylor series can be used to obtain first order approximations
\[
g_i \approx g_{i,0} + \frac{\partial g_i}{\partial w_1} \Delta w_1 + \frac{\partial g_i}{\partial w_2} \Delta w_2 + \cdots + \frac{\partial g_i}{\partial w_N} \Delta w_N,
\]
where \( i = 1, \ldots, N \). Using the definition of \( g_i \) to formulate in terms of the error function yields
\[
\frac{\partial g_i}{\partial w_j} = \frac{\partial}{\partial w_j} \left( \frac{\partial E}{\partial w_i} \right) = \frac{\partial^2 E}{\partial w_j \partial w_i}.
\]
Utilizing this and setting \( g_i = 0 \), we obtain \( \Delta w = -H^{-1}g \), and the resulting update relation. Newton's method has quadratic convergence but is unstable as small variations in the
initial estimate can produce significant changes in the final iterate value. Newton's method requires computation of the gradient and Hessian.

**Gauss-Newton Method**

The Gauss-Newton method is based on Newton's method with the goal of reducing the computational cost. The update rule for the Gauss-Newton method is

$$w_{k+1} = w_k - (J_k^T J_k)^{-1} J_k e_k,$$

where the error vector $e$ has the form

$$e = [e_{1,1}, e_{1,2}, \cdots, e_{1,M}, \cdots, e_{p,1}, e_{p,2}, \cdots, e_{p,M}]^T.$$

Here $M$ is the number of outputs from our network, and the Jacobian matrix has the form

$$J = \begin{bmatrix}
\frac{\partial e_{1,1}}{\partial w_1} & \frac{\partial e_{1,1}}{\partial w_2} & \cdots & \frac{\partial e_{1,1}}{\partial w_N} \\
\frac{\partial e_{1,2}}{\partial w_1} & \frac{\partial e_{1,2}}{\partial w_2} & \cdots & \frac{\partial e_{1,2}}{\partial w_N} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial e_{1,M}}{\partial w_1} & \frac{\partial e_{1,M}}{\partial w_2} & \cdots & \frac{\partial e_{1,M}}{\partial w_N} \\
\frac{\partial e_{p,1}}{\partial w_1} & \frac{\partial e_{p,1}}{\partial w_2} & \cdots & \frac{\partial e_{p,1}}{\partial w_N} \\
\frac{\partial e_{p,2}}{\partial w_1} & \frac{\partial e_{p,2}}{\partial w_2} & \cdots & \frac{\partial e_{p,2}}{\partial w_N} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial e_{p,M}}{\partial w_1} & \frac{\partial e_{p,M}}{\partial w_2} & \cdots & \frac{\partial e_{p,M}}{\partial w_N}
\end{bmatrix}.$$

This comes from the observation that $g = J e$ and that the element at the $i$th row and $j$th column of the Hessian matrix can be written as

$$H_{i,j} = \frac{\partial^2 E}{\partial w_i \partial w_j} = \sum_{p=1}^{P} \sum_{m=1}^{M} \frac{\partial e_{p,m}}{\partial w_i} \frac{\partial e_{p,m}}{\partial w_j} + \sum_{p=1}^{P} \sum_{m=1}^{M} \frac{\partial^2 e_{p,m}}{\partial w_i \partial w_j} e_{p,m}.$$  

From this it follows that the Hessian can be approximated as $H \approx J^T J$. Only the Jacobian is computed and similar to Newton's method, this provides fast but unstable convergence as an initial estimate being far from the desired value can cause this method to converge slowly or not at all [46].
Levenberg-Marquardt Algorithm

In the Levenberg-Marquardt algorithm, the approximation to the Hessian matrix used in the Gauss-Newton method is modified as

$$H \approx J^T J + \mu I$$

where $\mu$ is a positive constant and $I$ is the identity matrix. This is done so that the approximated Hessian matrix is invertible. Thus the update rule of the Levenberg-Marquardt algorithm is

$$w_{k+1} = w_k - (J_k^T J_k + \mu I)^{-1} J_k e_k.$$

The Levenberg-Marquardt algorithm is stable and fast and only requires computation of the Jacobian. It will hold similar to the steepest descent method when $\mu$ is large and to the Gauss-Newton method when $\mu$ is small. The inversion in this algorithm does not produce a problem as MATLAB is efficient at finding the solution to a matrix equation [32, 58].

The Levenberg-Marquardt Algorithm is used in MATLAB for medium problems. For large problems, stochastic gradient descent is often utilized [32].
Sensitivity analysis quantifies the variation in model responses as inputs are varied in a region of the admissible parameter space. We consider an observation model of the form

\[ y_i = f(t_i, \theta) + \epsilon_i, \ i = 1, ..., n \]  \hspace{1cm} (4.1)

where \( \theta \) denotes the parameters of the model and the observation errors \( \epsilon_i \) are iid and normally distributed, \( \epsilon \sim \mathcal{N}(0, \sigma^2) \). Here, \( t_i \) are value of an independent variable, such as time. Sensitivity analysis is performed to determine what the identifiable or influential parameters in a model are.

**Definition 4.0.1.** The parameters \( \theta \) in (4.1) are **identifiable** if they are uniquely determined by observations \( y_i \).

**Definition 4.0.2.** The parameter \( \theta_i \) in (4.1) is **statistically noninfluential** if random sampling from an appropriate distribution centered at \( \theta_i^* \) produces negligible variations in \( f \). That is, changes in the input \( \theta_i \) produce insignificant changes in \( f \).
Once sensitivity analysis is performed, one can fix noninfluential or nonidentifiable parameters in the model for subsequent model construction or statistical inference. Sensitivity analysis can also be used to assess whether a model is overly fragile with regard to various parameters.

Sensitivity analysis determines subsets of parameters that are influential. Active subspace analysis determines subspaces of parameters that are influential. This can be advantageous as it can be employed for parameter dimension reduction when parameter influence is on linear subspaces and it exploits use of stable linear algebra techniques as discussed in Section 4.4.

We discuss local sensitivity analysis in Section 4.1 and parameter subset selection in Section 4.2. In Section 4.3, we detail global sensitivity analysis methods, which include Pearson correlation coefficients, Sobol’ indices, and Morris screening. We discuss active subspace analysis in Section 4.4. Lastly, we summarize energy statistics in Section 4.5, which are a means to verifying sensitivity and active subspace analysis results.

4.1 Local Sensitivity Analysis

Local sensitivity analysis focuses on quantifying the variability of a response

\[ y = f(\theta), \quad \theta = [\theta_1, \ldots, \theta_p]^T, \]  

when parameters are perturbed about a nominal value \( \theta^* \). This is commonly defined by the partial derivatives \( \frac{\partial f}{\partial \theta_i}(\theta^*) \). This requires computation or approximation of the gradient of \( f \). Automatic differentiation, when available provides an exact value of the gradient, but is often computationally prohibitive or unavailable. When automatic differentiation is prohibitive, we use the forward finite difference approximation, defined by

\[
\frac{\partial f}{\partial \theta_i}(\theta^*) = \frac{f(\theta^* + \Delta \theta e_i) - f(\theta^*)}{\Delta \theta},
\]

where \( e_i \) is the \( i \)th canonical vector and \( \Delta \theta \) is generally chosen to be \( 10^{-6} \) to \( 10^{-8} \) of the nominal value of \( \theta_i \). Forward finite difference is simple to implement and quick to run, but can have numerical errors that one needs to be aware of. For example, for small values of \( \Delta \theta \), \( f(\theta^* + \Delta \theta e_i) \) and \( f(\theta^*) \) may agree to many significant digits, and so the subtraction of these suffers a loss of significant digits.

An alternative to forward finite difference is complex step approximations, described in [30, 31], which is not prone to numerical cancellation. We note, however, that the surrogate modeling software utilized in MATLAB for this dissertation do not directly accommodate...
complex arithmetic. Hence we employ finite-difference approximations when employing these surrogate models.

### 4.1.1 Normalized Gradient Approximation

Derivative-based sensitivity analysis methods are unit-based and can make it difficult to compare parameters with vastly different scales. As model inputs often have different units, using the partial derivatives in derivative-based sensitivity analysis may create an unfair comparison. Normalization of the gradient creates a unit-less gradient, which addresses this problem.

Consider surrogate models of the form

\[ f(t, \theta) : \mathbb{R} \times \mathbb{R}^p \rightarrow \mathbb{R}, \]

where \( \theta \in \mathbb{R}^p \). Note that some surrogate models have the time input whereas others do not. For generality, we include time in this description. To obtain parameters on the same scale, map the parameter space to \([0, 1]^p\). The scaled model is defined as

\[ h(t, x) = f(t, g(x)), \]

where

\[ g(x) = \text{diag}(\theta_u - \theta_\ell)x + \theta_\ell. \]

Here \( x \in [0, 1]^p \), \( g(x) \in \mathbb{R}^p \), and \( \theta_u \) and \( \theta_\ell \) are vectors containing the upper and lower limits for each parameter. Note that

\[ h(t, x) : \mathbb{R} \times [0, 1]^p \rightarrow \mathbb{R}, \]
\[ g(x) : [0, 1]^p \rightarrow \mathbb{R}. \]

For \( p = 1 \) with an non-time dependent model, we obtain

\[
h'(x) = \frac{h(x + \Delta) - h(x)}{\Delta} \\
= \frac{f(g(x + \Delta)) - f(g(x))}{\Delta} \\
\approx \frac{f(g(x)) + \Delta f'(g(x))g'(x) - f(g(x))}{\Delta} \\
\approx f'(g(x))g'(x).
\]
Generalizing this yields

$$\nabla h(t, x) = \text{diag}(\theta_u - \theta_f) \nabla f(t, g(x))$$

[5, 13, 25].

We provide an example to illustrate the necessity for normalization of the gradient. Consider the model

$$y = 10^6 \theta_1 + \theta_2,$$

where $\theta_1 \sim \mathcal{U}(0, 1)$ and $\theta_2 \sim \mathcal{U}(0, 10^6)$. Then

$$\frac{\partial y}{\partial \theta_1} = 10^6 \quad \text{and} \quad \frac{\partial y}{\partial \theta_2} = 1$$

for any value of $\theta = [\theta_1, \theta_2]^T$. Using active subspace methods as detailed in Section 4.4, would indicate that $\theta_1$ is influential and $\theta_2$ is noninfluential. Similarly, parameter subset selection as detailed in Section 4.2 would determine $\theta_1$ is identifiable and $\theta_2$ is nonidentifiable. However, these two parameters are on different orders of magnitude and one-at-a-time sensitivity analysis indicates that $\theta_1$ and $\theta_2$ have similar impacts on the response as shown in Figure 4.1. Normalizing this gradient, would provide

$$\frac{\partial y}{\partial x_1} = 10^6 \quad \text{and} \quad \frac{\partial y}{\partial x_2} = 1 \times 10^6 = 10^6,$$

indicating that these parameters have equivalent effects on the response variable.

![Figure 4.1](image-url)  

**Figure 4.1** One-at-a-time plots for the model $y = 10^6 \theta_1 + \theta_2$, where $\theta_1 \sim \mathcal{U}(0, 1)$ and $\theta_2 \sim \mathcal{U}(0, 10^6)$. Default values were taken to be $\theta^* = [0.5, 5 \times 10^5]^T$. 

38
4.2 Parameter Subset Selection

Consider the observation model

\[ y_i = f(t_i, \theta) + \epsilon_i , \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2) , \quad i = 1, ..., n, \]

with the associated sensitivity matrix

\[ \chi(\theta) = \begin{bmatrix} \frac{\partial y}{\partial \theta_1}(t_1, \theta) & \cdots & \frac{\partial y}{\partial \theta_p}(t_1, \theta) \\ \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial \theta_1}(t_n, \theta) & \cdots & \frac{\partial y}{\partial \theta_p}(t_n, \theta) \end{bmatrix}. \]  

(4.3)

In our example, \( t_i \) will denote time, but this can be any independent variable.

Consider the cost functional

\[ J(\theta) = \frac{1}{n} \sum_{i=1}^{n} [y_i - f(t_i, \theta)]^2. \]

At the optimal \( \theta^* \), we have that \( y_i \approx f(t_i, \theta^*) \). The Taylor expansion

\[ f(t_i, \theta^* + \Delta \theta) \approx f(t_i, \theta^*) + \nabla_\theta f(t_i, \theta^*) \Delta \theta \]

provides the approximation

\[ J(\theta^* + \Delta \theta) \approx \frac{1}{n} \sum_{i=1}^{n} [\nabla_\theta f(t_i, \theta^*) \Delta \theta]^2 = \frac{1}{n} \Delta \theta^T \chi^T \chi \Delta \theta. \]

Now, take \( \Delta \theta \) to be an eigenvector of \( \chi^T \chi \), corresponding to eigenvalue \( \lambda \), to obtain

\[ J(\theta^* + \Delta \theta) \approx \frac{\lambda}{n} ||\Delta \theta||_2^2. \]

Note that \( \chi^T \chi \) is symmetric positive semi-definite, so the eigenvalues are real and non-negative. Thus if \( \lambda \approx 0 \), the perturbations \( J(\theta^* + \Delta \theta) \) are also approximately 0, and the corresponding parameters are not identifiable at \( \theta^* \). This forms the basis for the local parameter subset selection algorithm provided in Algorithm 1 [8, 36, 38, 47]. Previous algorithms employed eigenvalues, while Algorithm 1 utilizes singular values as they are more numerically stable.
Algorithm 1 Local Parameter Subset Selection Algorithm from [36, 47]

1. Select space $\Gamma$ of admissible parameter values.
2. Set a threshold $\eta$, number of samples $M$, and set $j = p$.
3. For $k = 1, \ldots, M$
   
   (a) Obtain random samples $\theta^k \sim \mathcal{U}(\Gamma)$ and construct $\chi(\theta^k)$ as defined in (4.3).
   
   (b) Compute the ordered singular values $\sigma_1 \leq \sigma_2 \leq \cdots \leq \sigma_j$ of $\chi(\theta^k)$ by taking the SVD decomposition of $\chi(\theta^k)$.
   
   (c) If $\sigma_2^2 > \eta$, stop. All remaining inputs are identifiable. If $\sigma_1 < \eta$:
      
      i. Identify the component of $v_1$ having the largest magnitude, where $v_1$ is the singular vector associated with $\sigma_1$.
      
      ii. Remove the corresponding column in $\chi(\theta^k)$, set $j = j - 1$ and repeat 3b and 3c.

Remark: In practice, one often looks for a large gap in singular values $\sigma_i << \sigma_{i+1}$ to determine the stopping criteria. Once such a gap is no longer present, we obtain the desired results.

4.3 Global Sensitivity Analysis

Local sensitivity analysis uses small perturbations about nominal values to determine local influences of parameters. Global sensitivity analysis quantifies the variation in a model response as inputs are varied in an admissible parameter space. In this section, we discuss one-at-a-time sensitivity analysis, Pearson correlation coefficients, Sobol’ sensitivity indices, and Morris screening.

4.3.1 One-at-a-time Sensitivity Analysis

In one-at-a-time sensitivity analysis, all parameters are held constant at their nominal values except for one, which is varied across it’s feasible domain. The effect this has on the response or quantity of interest of the model is tracked during this process. This procedure is then repeated for each input. An example of this is depicted in Figure 4.1. This is one of the simplest sensitivity analysis methods. The main problem with this approach is that no parameter interaction is captured. This can provide problematic results when parameters have large interactions in a model. This method can be utilized directly for an agent-based model and on surrogate models for an agent-based model.
### 4.3.2 Correlation Coefficients

A simple way to observe statistical correlation between parameters and quantities of interest is via correlation coefficients. Consider the Pearson correlation coefficient, defined as

$$r_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

for determining the correlation between variables $X$ and $Y$. The standard deviations for $X$ and $Y$ are $\sigma_x$ and $\sigma_y$, respectively, and $\sigma_{xy}$ is the covariance between $X$ and $Y$. Pearson's correlation coefficient provides a value between -1 and 1, and is appropriate when observing a linear relationship between two random variables \[43\].

### 4.3.3 Sobol' Sensitivity Indices

Sobol' sensitivity indices apportion how uncertainty in the response is related to uncertainty in the inputs. They are variance-based and are advantageous because they do not require linearity or monotonicity as required with Pearson's correlation coefficient. Sobol' indices are considered a unit-less sensitivity measure. They are commonly applied to surrogate models of agent-based models for sensitivity analysis, such as in \[6, 52\]. For traditional Sobol' indices, parameters are assumed to be independent. Parameters in many applications are dependent or correlated, however, it is common in applications to assume parameters are independent and utilize the method detailed in this Section, usually with minimal ramifications on the results.

Consider the model,

$$y = f(\theta),$$

where $\theta = [\theta_1, ..., \theta_p]$. The sample space and probability density function for $\theta_k$ are $\Gamma_k$ and $\rho(\theta_k)$, respectively. Based on the assumption that parameters are independent, the sample space and probability density function for $\theta$ are

$$\Gamma = \prod_{k=1}^{p} \Gamma_k, \quad \rho(\theta) = \prod_{k=1}^{p} \rho(\theta_k),$$

respectively. The hierarchical expansion (or ANOVA decomposition) is

$$f(\theta) = f_0 + \sum_{i=1}^{p} f_i(\theta_i) + \sum_{1 \leq i < j \leq p} f_{ij}(\theta_i, \theta_j) + \cdots + f_{1,...,p}(\theta_1, ..., \theta_p),$$

where $f_0$ is the mean, $f_i(\theta_i)$ represents first order interactions, $f_{ij}(\theta_i, \theta_j)$ represents sec-
ond order interactions, through \( f_{1,...,p}(\theta_1,...,\theta_p) \) representing \( p^{th} \) order interactions. Under suitable conditions, the components can be formulated as

\[
f_0 = \int_{\Gamma} f(\theta)\rho(\theta) d\theta = \mathbb{E}[f(\theta)],
\]

\[
f_i(\theta_i) = \int_{\Gamma^{p-1}} f(\theta)\rho(\theta_{\sim i}) d\theta_{\sim i} - f_0 = \mathbb{E}[f(\theta)|\theta_i] - f_0,
\]

\[
f_{ij}(\theta_i, \theta_j) = \int_{\Gamma^{p-2}} f(\theta)\rho(\theta_{\sim\{i,j\}}) d\theta_{\sim\{i,j\}} - f_i(\theta_i) - f_j(\theta_j) + f_0 = \mathbb{E}[f(\theta)|\theta_i, \theta_j] - f_i(\theta_i) - f_j(\theta_j) - f_0,
\]

where \( \theta_{\sim i} \) and \( \theta_{\sim\{i,j\}} \) denotes the vector having all components of \( \theta \) except those in the \( i \) and the \( i \) and \( j \) components, respectively. That is,

\[
\theta_{\sim i} = [\theta_1, ..., \theta_{i-1}, \theta_{i+1}, ..., \theta_p].
\]

Note that components representing interactions have zero mean and are pairwise orthogonal.

The total variance \( D \) of the response \( Y \) is given by

\[
D = \text{var}(Y) = \int_{\Gamma} f^2(\theta)\rho(\theta) d\theta - f_0^2.
\]

By utilizing the ANOVA decomposition and the fact that the components have zero mean and are pairwise orthogonal, \( D \) can be formulated as

\[
D = \sum_{i=1}^{p} D_i + \sum_{1 \leq i < j \leq p} D_{ij} + \cdots + D_{1,...,p},
\]

where

\[
D_i = \int_{\Gamma} f_i^2(\theta_i)\rho(\theta_i) d\theta_i = \text{var}[f_i(\theta_i)],
\]

\[
D_{ij} = \int_{\Gamma} f_{ij}^2(\theta_i, \theta_j)\rho(\theta_i)\rho(\theta_j) d\theta_i d\theta_j.
\]

Higher order terms are defined similarly.
The Sobol’ indices are defined as
\[ S_i = \frac{D_i}{D}, \quad S_{ij} = \frac{D_{ij}}{D}, \quad S_{1,...,p} = \frac{D_{1,...,p}}{D} \]
and, by definition, they satisfy
\[ \sum_{i=1}^{p} S_i + \sum_{1 \leq i < j \leq p} S_{ij} + \cdots + S_{1,...,p} = 1. \]
Here \( S_i \) are termed the first order sensitivity indices and large values of \( S_i \) indicate variables that strongly influence the response. Additionally, we consider the total sensitivity indices which quantify the total effect of \( \theta_i \), including higher order interactions. With a second-order expansion, the total sensitivity indices are defined as
\[ S_{Ti} = S_i + \sum_{j=1}^{p} S_{ij}. \]
A probabilistic interpretation for the Sobol’ indices for \( y = f(\theta) \) is
\[ S_i = \frac{\text{var}[\mathbb{E}(Y|\theta_i)]}{\text{var}(Y)} \quad \text{and} \quad S_{Ti} = 1 - \frac{\text{var}[\mathbb{E}(Y|\theta_{\sim i})]}{\text{var}(Y)} = \frac{\mathbb{E}[\text{var}(Y|\theta_{\sim i})]}{\text{var}(Y)} \]
[25, 48]. We employ Algorithm 2 - a modification of the algorithm in [25], which incorporates the improvement detailed in [44], as given in [47] - to compute the indices.

Sobol’ indices assumes independent parameters. If parameters are dependent, then classic Sobol’ indices can be generalized as
\[ S_u = \frac{\text{cov}[f_u(\theta_u), f(\theta)]}{\text{var}[f(\theta)]}. \]
Here, \( u = \{i_1, ..., i_k\} \) is a subset of \( \{1, ..., p\} \) and \( \theta_u = \{\theta_{i_1}, ..., \theta_{i_k}\} \). This approach can be problematic in that the computation of the covariance requires specific prior knowledge about the joint probability density \( \rho(\theta) \), which is typically not known. Additionally, this can result in negative Sobol’ indices which obscures the interpretation of them as ranking influence of parameters [27]. We will not focus on these due to the required prior knowledge of parameters.
Algorithm 2 A modification of the algorithm in [25], which incorporates the improvement detailed in [44], as given in [47]

1. Draw \( M \) random numbers \( \theta_j^i \) and \( M \) random numbers \( \hat{\theta}_j^i \) from the parameter distribution for each parameter to create \( M \times p \) sample matrices

\[
A = \begin{bmatrix}
\theta_1^1 & \cdots & \theta_1^i & \cdots & \theta_1^p \\
\vdots & \cdots & \vdots & \cdots & \vdots \\
\theta_M^1 & \cdots & \theta_M^i & \cdots & \theta_M^p
\end{bmatrix}, \quad B = \begin{bmatrix}
\hat{\theta}_1^1 & \cdots & \hat{\theta}_1^i & \cdots & \hat{\theta}_1^p \\
\vdots & \cdots & \vdots & \cdots & \vdots \\
\hat{\theta}_M^1 & \cdots & \hat{\theta}_M^i & \cdots & \hat{\theta}_M^p
\end{bmatrix}.
\]

2. Define \( M \times p \) matrices

\[
C_i = \begin{bmatrix}
\theta_1^1 & \cdots & \hat{\theta}_1^i & \cdots & \theta_1^p \\
\vdots & \cdots & \vdots & \cdots & \vdots \\
\hat{\theta}_M^1 & \cdots & \hat{\theta}_M^i & \cdots & \theta_M^p
\end{bmatrix},
\]

to be identical to \( A \) except that the \( i^{th} \) column is taken from \( B \).

3. Append \( B \) to \( A \) to create the \( 2M \times p \) matrix

\[
D = \begin{bmatrix}
A \\
B
\end{bmatrix}.
\]

4. Compute \( M \times 1 \) vectors of model outputs

\[
y_A = f(A) , \quad y_B = f(B) , \quad y_{C_i} = f(C_i)
\]

and \( 2M \times 1 \) vector

\[
y_D = f(D)
\]

by evaluating the model at the input values in \( A, B, C_i, \) and \( D \). The evaluation of \( y_A, y_B, \) and \( y_{C_i} \) requires \( M(p + 2) \) model evaluations.

5. Approximate the first-order sensitivity indices via Monte Carlo integration as

\[
S_i = \frac{\text{var}[E(Y|q_i)]}{\text{var}(Y)} \approx \frac{1}{M} \frac{y_B^T y_{C_i} - y_B^T y_A}{y_D^T y_D - [E(y_D)]^2},
\]

and total indices as

\[
S_{T_i} = \frac{\text{E}[\text{var}(Y|q_{-i})]}{\text{var}(Y)} \approx \frac{1}{M} \frac{y_A^T y_A - 2y_A^T y_{C_i} + y_{C_i}^T y_{C_i}}{y_D^T y_D - [E(y_D)]^2},
\]

where \( E(y_D) \approx \frac{1}{2M} \sum_{j=1}^{2M} y_D^j \). Here \( y_D^j \) denotes the \( j^{th} \) element of \( f(D) \).
**Time-Dependent Sobol’ Indices**

Classic Sobol’ indices are useful for models that have a scalar response. For time-dependent models, Sobol’ indices can be computed independently at time values to quantify the influence of parameters over time. However, this excludes any temporal correlation. To address this, we introduce time-dependent Sobol’ indices from [3]. First, we define Sobol’ indices pointwise in time. Let \( u = \{i_1, i_2, \ldots, i_k\} \) be a subset of \( \{1, \ldots, p\} \). Denote the corresponding parameters by \( \theta_u = \{\theta_{i_1}, \ldots, \theta_{i_k}\} \). Then the hierarchical expansion or ANOVA decomposition is

\[
f(t, \theta) = f_0(t) + \sum_{k=1}^{p} \sum_{|u|=k} f_u(t, \theta_u),
\]

where \( f_0(t) = \mathbb{E}_{\theta}[f(t, \theta)] \) and

\[
f_u(t, \theta_u) = \mathbb{E}[f(t, \theta)|\theta_u] - \sum_{v \subset u} f_v(t, \theta_v).
\]

Note that this is a more compact formulation of the previous expansion. The partial and total variance are

\[
D_u(t) = \mathbb{E}[f_u^2(t, \theta_u)] = \text{var}[f_u(t, \theta_u)],
\]

\[
D(t) = \sum_{k=1}^{p} \sum_{|u|=k} D_u(t).
\]

The pointwise-in-time Sobol’ indices are

\[
S_u(t) = \frac{D_u(t)}{D(t)}.
\]

The generalized first order Sobol’ indices for \( \theta_u \) are then defined by

\[
S_u(T) = \frac{\int_0^T D_u(t) dt}{\int_0^T D(t) dt}.
\]

This can be approximated via quadrature by

\[
S_u(T) = \frac{\sum_{m=1}^{N_{\text{quad}}} D_u(t_m) w_m}{\sum_{m=1}^{N_{\text{quad}}} D(t_m) w_m},
\]

where \( \{t_m, w_m\}_{m=1}^{N_{\text{quad}}} \) are quadrature points and weights. Generalized total Sobol’ indices are defined analogously [3].
4.3.4 Morris Screening

Morris screening is a type of one-at-a-time sensitivity analysis. It is sometimes used with agent-based models, such as in [52]. We will utilize Morris screening to approximate the derivative-based global sensitivity measures (DGSM) given by the index

$$\mu_i^* = \int_{\Gamma} \left| \frac{\partial f}{\partial \theta_i}(\theta) \right| \rho(\theta) d\theta,$$

where $\Gamma$ is the parameter space. Morris screening partially eliminates the local nature of OAT methods by averaging over local derivatives to provide more global sensitivity measures. Note that the absolute value in $\mu_i^*$ avoids cancellation effects of derivatives which may change signs in the parameter space. To compute this index, we will utilize Monte Carlo approximations. Details are provided in Algorithm 3 taken from [25, 47, 48].

**Algorithm 3** Morris Screening Algorithm from [25, 47, 48]

1. Designate a step size $\Delta$ and number of samples $R$.
2. For $j = 1, ..., R$,
   (a) Randomly sample a value $\theta^j \in \Gamma = [0, 1]^p$.
   (b) Apply the finite-difference relation to compute the elementary effect associated with the $i^{th}$ parameter
   $$d^j_i = \frac{f(\theta^j + \Delta e_i) - f(\theta^j)}{\Delta}.$$  
   Here, we interpret $d^j_i$ as the finite difference approximation to the partial derivative at $\theta^j$.
3. Compute
   $$\mu_i^* = \frac{1}{R} \sum_{j=1}^{R} |d^j_i|$$
   $$\sigma_i^2 = \frac{1}{R-1} \sum_{j=1}^{R} (d^j_i - \mu_i)^2$$,  
   $$\mu_i = \frac{1}{R} \sum_{j=1}^{R} d^j_i$$
4. Noninfluential inputs are those with small values of $\mu_i^*$ and $\sigma_i$, and influential inputs are those with large values of $\mu_i^*$ and $\sigma_i$.  

46
We obtain two values of interest from Morris screening, $\mu_i^*$ and $\sigma_i^2$ as defined in Algorithm 3. The mean, $\mu_i^*$ quantifies the individual effect of the input on the output. The variance, $\sigma_i^2$ approximates the combined effects due to interactions with other inputs. Parameters with small values of both $\mu_i^*$ and $\sigma_i$ are noninfluential and parameters with large values of $\mu_i^*$ and $\sigma_i$ are influential. Unlike Sobol’ indices, $\mu_i^*$ and $\sigma_i$ do not quantify the relative influence of parameters.

4.4 Active Subspace Analysis

Local and global sensitivity analysis provides subsets of parameters that are influential to the model. Active subspace analysis provides linear subspaces of influential parameters. Further, active subspaces can be extended to quantify parameter sensitivities. In this section, we introduce these methods. First, we introduce the linear algebra concepts that are integral to this analysis.

Singular Value Decomposition

**Definition 4.4.1.** Let $A \in \mathbb{R}^{n \times p}$. The singular value decomposition (SVD) of $A$ is given by

$$A = U \Sigma V^T$$

where $U \in \mathbb{R}^{n \times n}$, $\Sigma \in \mathbb{R}^{n \times p}$, and $V \in \mathbb{R}^{p \times p}$. For $n > p$, $\Sigma = \begin{bmatrix} S & 0 \end{bmatrix}^T$, where

$$S = \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \ddots \\ & & & \sigma_p \end{bmatrix}, \quad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$$

is a diagonal matrix containing the singular values of $A$ in descending order. The columns of $U$ and $V$ contain the left and right singular vectors of $A$.

Let $r$ be the rank of $A$, determined by the number of nonzero singular values. In practice, we consider this as the number of singular values larger than a tolerance $\epsilon$. If $r < \min(n, p)$, $A$ is considered rank deficient and can be expressed as

$$A = U_r S_r V_r^T,$$

where $U_r \in \mathbb{R}^{n \times r}$ contains the first $r$ columns of $U$, $V_r \in \mathbb{R}^{p \times r}$ contains the first $r$ columns
of $V$, and
\[
S_r = \begin{bmatrix}
\sigma_1 \\
\vdots \\
\sigma_r
\end{bmatrix}.
\]

An alternate representation of $A$ that is often employed in software for computing purposes is the thin SVD of a matrix. Take our previous matrix $A \in \mathbb{R}^{n \times p}$, where $n > p$. Then we have
\[
A = U S V^T = U_p S V^T
\]
where $U_p \in \mathbb{R}^{n \times p}$ contains the first $p$ columns of $U$. The thin SVD is advantageous as it requires less storage.

**Eigendecomposition**

**Definition 4.4.2.** Let $A \in \mathbb{R}^{n \times n}$ be a square matrix. Assume $A$ has nondegenerate eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ with corresponding linearly independent eigenvectors $X_1, X_2, \ldots, X_n$. That is, the eigenspace of each eigenvalue has dimension one. Then
\[
A = P D P^{-1},
\]
where
\[
P = \begin{bmatrix} X_1 & X_2 & \ldots & X_n \end{bmatrix} \quad \text{and} \quad D = \begin{bmatrix} \lambda_1 & & \\
& \lambda_2 & \\
& & \ddots \\
& & & \lambda_n \end{bmatrix}.
\]

This decomposition can be easily verified as
\[
AP = A \begin{bmatrix} X_1 & X_2 & \ldots & X_n \end{bmatrix} = \begin{bmatrix} AX_1 & AX_2 & \ldots & AX_n \end{bmatrix} = \begin{bmatrix} \lambda_1 X_1 & \lambda_2 X_2 & \ldots & \lambda_n X_n \end{bmatrix} = \begin{bmatrix} X_1 & X_2 & \ldots & X_n \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\
& \lambda_2 & \\
& & \ddots \\
& & & \lambda_n \end{bmatrix} = PD;
\]
4.4.1 Active Subspace Analysis for Linearly Parameterized Problems

Example 1. To illustrate the idea of active subspaces, we consider the linear model

\[ y = 5\theta_1 + 2\theta_2 = \begin{bmatrix} 5 & 2 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = A\theta. \]

Then the unidentifiable and identifiable subspaces are

\[ N I(\theta) = N(A) = c \begin{bmatrix} 2 \\ -5 \end{bmatrix}, \quad c \in \mathbb{R} \]

\[ I(\theta) = R(A^T) = c \begin{bmatrix} 5 \\ 2 \end{bmatrix}, \quad c \in \mathbb{R}, \]

respectively. The identifiable or active subspace yields the direction where the function is changing the most and the inactive or unidentifiable subspace yields the direction in which the function is changing the least. In Example 1, the inactive subspace is the direction in which the function doesn't change at all. Note that we can formulate the inactive and active subspaces in terms of \( A^T A \) as

\[ N I(\theta) = N(A) = N(A^T A), \]

\[ I(\theta) = R(A) = R(A^T A) \]

[20].

4.4.2 Active Subspace Analysis for Nonlinearly Parameterized Problems

We now generalize this process to nonlinear models. To do this, consider the nonlinear model

\[ y = f(\theta), \quad (4.4) \]

where \( \theta = [\theta_1, ..., \theta_p]^T \). Begin by linearizing (4.4) via the gradient \( \nabla_\theta f \) and specifying a probability density function \( \rho(\theta) \). \( \rho(\theta) \) specifies the distribution of input parameters. We then use these to average the local sensitivity behavior over a supported region \( \Gamma \subseteq \mathbb{R}^p \).

Form the symmetric positive semi-definite matrix

\[ C = \int_{\Gamma} (\nabla f)(\nabla f)^T \rho d\theta, \]
under the assumption that $f$ is Lipschitz continuous and differentiable to ensure that the gradient is bounded. Since $C$ is symmetric positive semi-definite, it has an eigendecomposition

$$C = W\Lambda W^T, \quad \Lambda = \text{diag}(\lambda_1, ..., \lambda_p),$$

where $\lambda_1 \geq \cdots \geq \lambda_p \geq 0$ and $W = [w_1, ..., w_p]$ is an orthogonal matrix comprised of the orthonormal eigenvectors $w_i$ of $C$. Observe that

$$\lambda_i = w_i^T C w_i = \int_\Gamma (w_i^T \nabla f)^2 \rho \, d\theta.$$ 

This implies that for $\lambda_i$ close to zero, $f$ changes minimally along the direction specified by $w_i$. In practice, we look for a large gap in eigenvalues to partition the eigenvalue and eigenvector matrices. That is, we find $n < p$, where $\lambda_n \gg \lambda_{n+1}$, and we partition

$$\Lambda = \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \end{bmatrix}, \quad W = \begin{bmatrix} W_1 & W_2 \end{bmatrix},$$

where $\Lambda_1 = \text{diag}(\lambda_1, ..., \lambda_n)$ and $W_1 = [w_1, ..., w_n]$. We then define the active variables and inactive variables as

$$\xi = W_1^T \theta \in \mathbb{R}^n \quad \text{and} \quad \zeta = W_2^T \theta \in \mathbb{R}^{p-n},$$

respectively. The active subspace is the range of $\xi$ and the inactive subspace is the range of $\zeta$ [9].

In practice, we often do not know the distribution $\rho(\theta)$. In such an instance, a common choice is to take the uniform distribution over a 5-20% perturbation around the nominal values. Note that this also specifies $\Gamma$. Once $\rho(\theta)$ is specified, we utilize Monte Carlo Sampling to approximate the matrix $C$, previously defined. We present Algorithm 4 from [9, 47] for this purpose.

Once the active variables are determined, one can create surrogate models on the active subspace. This analysis is based on the assumption that parameters are mutually independent. Further, these eigenvalues and eigenvectors can be utilized to calculate activity scores, as defined next.
Algorithm 4 Monte Carlo Sampling to Approximate $C$ from [9, 47]

1. Independently sample $M$ values $\{\theta^j\}$ from the parameter distribution $\rho(\theta)$.
2. For each sample $\theta^j$, compute the gradient $\nabla_{\theta} f^j = \nabla_{\theta} f(\theta^j)$.
3. Compute an approximation to matrix $C$ as
   \[
   \hat{C} = \frac{1}{M} \sum_{j=1}^{M} (\nabla f^j)(\nabla f^j)^T.
   \]
4. Compute the eigendecomposition $\hat{C} = \hat{W}\hat{\Lambda}\hat{W}^T$. If there is a large gap in the eigenvalues; e.g., $\hat{\lambda}_n \gg \hat{\lambda}_{n+1}$, split the eigenvalue and eigenvector matrices as
   \[
   \hat{\Lambda} = \begin{bmatrix}
   \hat{\lambda}_1 \\
   \hat{\lambda}_2
   \end{bmatrix}, \quad \hat{W} = \begin{bmatrix}
   \hat{W}_1 & \hat{W}_2
   \end{bmatrix},
   \]
   where $\hat{W}_1 = [\hat{w}_1, ..., \hat{w}_n]$. Define the active variables as $\xi = \hat{W}_1^T \theta$ and the inactive variables as $\zeta = \hat{W}_2^T \theta$.

Remark: Note that step 4 utilizes eigenvalues. However, steps 3 and 4 can be adjusted to utilize the SVD decomposition instead. This is advantageous as it is more numerically stable. Define
   \[
   \hat{G} = \frac{1}{\sqrt{M}} \begin{bmatrix}
   \nabla f^1 & \cdots & \nabla f^M
   \end{bmatrix}.
   \]
   Then $\hat{C} = \hat{G}\hat{G}^T$, and computing the eigendecomposition of $\hat{C}$ is equivalent to computing the SVD decomposition
   \[
   \hat{G} = \hat{W}\sqrt{\hat{\Lambda}}\hat{V},
   \]
   as
   \[
   \hat{C} = \hat{G}\hat{G}^T = \hat{W}\sqrt{\hat{\Lambda}}\hat{V}\hat{V}^T\sqrt{\hat{\Lambda}}^T\hat{W}^T
   = \hat{W}\sqrt{\hat{\Lambda}}\sqrt{\hat{\Lambda}}^T\hat{W}^T
   = \hat{W}\hat{\Lambda}\hat{W}^T.
   \]
4.4.3 Activity Scores

Activity scores quantify the sensitivity of individual parameters. To calculate the activity score of parameter $\theta_i$, we compute

$$a_i = a_i(n) = \sum_{j=1}^{n} \lambda_j w_{ij}^2, \quad i = 1, ..., p,$$

where $w_{ij}$ is the $i^{th}$ component of the $j^{th}$ eigenvector $w_j$ in $W_1$ [10]. Note that we only take the summation over the active subspace. If we do not have an active subspace, it follows that $n = p$.

The total Sobol’ indices can be bounded by activity scores in the following manner:

$$S_{T_i} \leq \frac{1}{4\pi^2 V} (a_i(n) + \lambda_{n+1}).$$

Here $V = \text{Var}[f]$, $a_i(n)$ is the activity score utilizing an $n$ dimensional active subspace, and $\lambda_{n+1}$ is the $(n+1)$st eigenvalue of $C$ [10].

4.4.4 Time-Dependent Models

For time-dependent responses, there are two methods to extend activity scores. The first is to compute the matrix $C$ at individual time steps, independently. The second method is to incorporate the time derivative and compute

$$C = \frac{1}{T} \int \left( \int \begin{bmatrix} f_i(\theta, t)^2 & f_i(\theta, t) \nabla f(\theta, t)^T \\ f_i(\theta, t) \nabla f(\theta, t) & \nabla f(\theta, t)^T \nabla f(\theta, t)^T \end{bmatrix} \rho d\theta \right) dt.$$

Note that $C$ admits a block structure

$$C = \begin{bmatrix} a & b^T \\ b & D \end{bmatrix}, \quad a \in \mathbb{R}, \quad b \in \mathbb{R}^p, \quad D \in \mathbb{R}^{p \times p}.$$

One can construct a global subspace of $\mathbb{R}^p$ using the eigenvectors of the lower right block $D$, which is symmetric positive semidefinite. This would be equivalent to averaging a time dependent analog of $C$ over the time interval and computing its eigenvectors [11].

4.5 Energy Statistics

One needs to verify the determined influential and noninfluential parameter sets. This can be done qualitatively by taking random samples for the parameters in the model.
and plotting the distribution of the response. Repeat that process keeping noninfluential parameters fixed. If the two distributions appear to be the same, the results are accurate. Energy statistics provide a quantitative technique to determine if samples are from the same distribution.

Let \(X_1, \ldots, X_{n_1}\) and \(Y_1, \ldots, Y_{n_2}\) be \(p\)-dimensional random independent samples from distributions \(F_X\) and \(F_Y\), respectively. Note that these distributions need not be known. The null hypothesis is that \(H_0: F_X = F_Y\). That is, the two sets of samples come from the same distribution. We define the energy distance as

\[
\epsilon_{n_1, n_2}(X, Y) = \frac{2}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{m=1}^{n_2} |X_i - Y_m| - \frac{1}{n_1^2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_1} |X_i - X_j| - \frac{1}{n_2^2} \sum_{k=1}^{n_2} \sum_{m=1}^{n_2} |Y_k - Y_m|
\]

and test statistic

\[
T_{n_1, n_2} = \frac{n_1 n_2}{n_1 + n_2} \epsilon_{n_1, n_2}.
\]

Here \(|·|\) is the Euclidean norm. The test statistic sums the absolute difference between samples from the two distributions to quantify the distance between the two sets of samples. Small test values indicate that the null hypothesis \(H_0\) is true. As the distributions are unknown, we cannot directly compute critical or \(p\)-values for the hypothesis test. Instead, bootstrapping methods are utilized.

Let \(W = \{W_1, \ldots, W_n\} = \{X_1, \ldots, X_{n_1}, Y_1, \ldots, Y_{n_2}\}\) be the pooled sample where \(n = n_1 + n_2\). Sample without replacement \(M\) random samples \(W^{(k)}, k = 1, \ldots, M\), where \(M\) is chosen to ensure that \((M + 1)\alpha\) is an integer. Here, \(\alpha\) is a chosen significance level. For each bootstrap sample, compute the test statistic \(T_{n_1, n_2}^{(k)}\) based on the sample split

\[
A^{(k)} = \{W_1^{(k)}, \ldots, W_{n_1}^{(k)}\}, \quad B^{(k)} = \{W_{n_1+1}^{(k)}, \ldots, W_n^{(k)}\}.
\]

If the energy statistics \(T_{n_1, n_2}\) exceeds \(100(1 - \alpha)\%\) of the resampled replicates \(T_{n_1, n_2}^{(k)}\), then the null hypothesis is rejected. The critical or \(p\)-value at the \(\alpha\) significance level is defined as \(T_c\), where \(P(T_{n_1, n_2}^{(k)} \leq T_c) = 1 - \alpha\) [47, 49, 50].
In this Chapter, we apply the sensitivity analysis and active subspace methods detailed in Chapter 4 to surrogate models for the Rebellion and Greenhouse models described in Section 2.2 and 2.3, respectively. The goal of this analysis is to determine the influential and noninfluential parameters of each model with respect to the model response. Further, we compare the effectiveness of utilizing neural networks as surrogate models with that of polynomial and Gaussian process surrogates. Lastly, we utilize active subspace methods to calculate activity scores and verify their agreement with existing sensitivity analysis methodologies for agent-based models. We will consider the $p = 2$ and $p = 5$ parameter Rebellion model in Sections 5.1 and 5.2 and the $p = 40$ parameter Greenhouse model in Section 5.3.

5.1 Rebellion Model with $p = 2$ Parameters

In this section, we analyze the Rebellion model for $p = 2$ parameters, initial officer density and government legitimacy. Recall from Section 2.2, that initial officer density provides the number of officers per 100 grid spaces and has a default value of 4 and that government
legitimacy is the perceived legitimacy of the government from a citizen’s view and has a default value of 0.8. We keep movement turned off in the model to improve computation time as this does not have a significant effect on model behavior.

For a standard realization of the Rebellion model, we expect to see behavior of the type shown in Figure 5.1. That is, there should be an initial spike in criminal activity, followed by periods of peace and outbreaks of rebellious activity. Moreover, the number of jailed citizens will oscillate at a similar rate. Due to the stochastic nature of agent-based models, each run we have will give slightly different heights of activity and timing of outbreaks of rebellious activity. Note that we show 100 time steps in Figure 5.1 for understanding of the model. For our analyses, we focus solely on the first 20 time steps. This does not have a large impact on our sensitivity analysis and significantly decreases the required computation time.

We consider three responses from the Rebellion model: the number of actively rebellious citizens per 1000 citizens, the number of jailed citizens per 1000 citizens, and the sum of these two. We run the model over 20 time steps and take the average of these responses over those 20 time steps to obtain the quantities of interest (QoI). When taking the average of the sum of the number of active and jailed citizens, we obtain what is called the rate of violence in [15], which is the term we use in our discussion.

In Section 5.1.1, we outline direct analysis methods of the Rebellion model for $p = 2$ parameters. In Section 5.1.2, we create surrogate models for averaged-time and time-dependent responses to use in Section 5.1.3, for sensitivity and active subspace analysis.

![Figure 5.1 Example run of the Rebellion model with default parameter values given in Table 2.2.](image)
5.1.1 Direct Analysis

We first consider sensitivity analysis in the classical sense of how the quantities of interest are affected by variations in the parameters. We then investigate individual characteristics and how they may drive individual behavior. The methods we introduce in this section can be found in greater detail in [15, 40].

Model Parameter One-at-a-time Sensitivity Analysis

We begin with single-parameter sensitivity analysis, otherwise known as one-at-a-time sensitivity analysis detailed in Section 4.3.1. To do so, one parameter is varied at a time while holding all other parameters constant and the response of the quantity of interest to the variations is tracked.

When performing sensitivity analysis on a model, a common choice is to vary the parameters 5-20% about their nominal value. As an illustration, we vary initial officer density 20% about its nominal value of 4. Thus, a parameter range of 3.2-4.8 is considered. All other parameters are set at the default values in Table 5.2. The model is run 10 times for each of the 11 values of initial officer density taken to be equally spaced on the interval [3.2, 4.8]. We plot the results in Figures 5.2 and 5.3. For each response considered, we plot the mean, minimum, maximum, and standard deviation of the quantity of interest based on the runs’ results.

Figure 5.2 provides a sense of the spread of the data. Note that as initial officer density gets larger, all of the quantities of interest decrease. This correlates with our expectations: the more law enforcement present, the less likely citizens are to become active, resulting in less jailed citizens.

![Figure 5.2 One-at-a-time sensitivity analysis of the Rebellion model for averaged responses of (a) the number of active citizens per 1000 citizens, (b) the number of jailed citizens per 1000 citizens, and (c) the rate of violence.](image-url)
We utilize Figure 5.3 to look for patterns in the standard deviations. We observe a downward trend in the standard deviations relating to the averaged number of active citizens per 1000 citizens, and an upward trend in the standard deviations relating to the average number of jailed citizens per 1000 citizens. This is expected from the spread we observe in Figure 5.2. We now discuss why these trends appear. When initial officer density is set in the lower range of values, each law enforcement officer is arresting an active citizen at each turn, so there is not much variation in the number of jailed citizens but, depending on the location of the law enforcement officers, there will be variation in activity. When initial officer density is closer to 4.8, there are more law enforcement officers to arrest active citizens and discourage active behavior. However, if these law enforcement officers end up close to each other, as opposed to evenly spread across the world, then they are not able to arrest as many active citizens as if they were spread evenly throughout the world. There is no definitive trend to the standard deviations for the total number of active and jailed citizens per 1000 citizens.

In this section, we investigated how the quantities of interest of the model responded to changes in the parameter initial officer density. In general it is advantageous to do this with all parameters to get an idea of what quantities of interest react most to which parameters. The primary drawback to this approach is that we cannot observe any interactions between parameters by varying the parameters one-at-a-time.

**Multi-Parameter Sensitivity Analysis**

To address the disadvantage of one-at-a-time sensitivity analysis, one can create contour plots like those shown in Figure 5.4, where two parameters are varied simultaneously and the quantities of interest are observed. This is done with the parameters government
legitimacy and initial officer density. We vary both parameters 20% about their nominal values for ranges of \([0.64, 0.96]\) and \([3.2, 4.8]\).

Contour plots allow us to observe possible interactions between parameters. For example, in Figure 5.4a, we see that for government legitimacy values above 0.85, there is a low average number of active citizens regardless of the value of initial officer density. Clearly contour plots can be used to visualize interactions between two parameters. It is possible to utilize contour plots in a grid structure to add a third parameter - see [40] for details - but they are prohibitive as the number of parameters increases. Although contour plots can only help us visualize interactions between two or three variables at a time, this can still be a valuable tool to understanding how our model is working and for verifying that we are seeing behavior we would expect.

**Agent-Level Sensitivity Analysis**

One advantage of agent-based models is the capability to track individual behavior in addition to observing patterns that emerge on a system level. This can be particularly helpful in identifying how and where the complex dynamics of an agent-based model - and real-world phenomena - arise.

Figure 5.5 presents a scatter plot of the number of times an individual is arrested over 1000 time steps plotted against their hardship and risk aversion values, using the default values in the Rebellion model provided in Table 2.2. We scaled the model to a \(100 \times 100\) grid so that we have a larger number of agents to illustrate the emerging patterns. From Figure 5.5, we observe that citizens whose hardship is below 0.5 fall into the class of never actively rebellious, whereas citizens whose hardship is above 0.5 fall into the class of conditionally active or always active. A citizen who is always active spends the simulation alternating between jail time and active time, whereas a citizen who is conditionally active becomes
active or quiet depending on the behavior of other agents within the citizen’s vision. When hardship is greater than 0.5, a combination of a citizen’s hardship and risk aversion values will determine what class the citizen falls into.

Note that this can help us to understand system level patterns that emerge. For example, Figure 5.5 implies that there is a portion of the population that never becomes active, so we should never expect to see all citizen’s criminally active for the default parameter values. Additionally, the large portion of citizens that are conditionally active explains why we observe bursts of criminal activity alternating with periods of peace. Lastly, note that these observations are in line with the findings in [15].

5.1.2 Surrogate Models

To apply the sensitivity analysis methods of Chapter 4 to an agent-based model, we must first create surrogate models. This improves the computation efficiency necessary for a large number of runs and enables computation of the gradient by creating a deterministic model. We create time-dependent and averaged-time surrogate models. We utilize 3 different methods for creating surrogate models detailed in Chapter 3: multivariate polynomial regression via ordinary least squares, Gaussian process regression, and neural networks. Lastly, we compare results utilizing each surrogate model to ensure consistency.

The data comes from a 20% perturbation in the nominal values of initial officer density and government legitimacy. The model is run 200 times with random choices for these parameters in the perturbation ranges. Each run is 20 time steps. Values are averaged over 10 repetitions for each choices of parameters. The test data is obtained similarly and consists of 8 test data sets containing 25 runs each. Additionally, a second set of training
Table 5.1 Pearson correlation coefficients for quantities of interest in the Rebellion model with 2 parameters varied.

<table>
<thead>
<tr>
<th></th>
<th>Number of Active Citizens</th>
<th>Number of Jailed Citizens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate of Violence</td>
<td>0.9610</td>
<td>0.9851</td>
</tr>
<tr>
<td>Number of Jailed Citizens</td>
<td>0.8990</td>
<td></td>
</tr>
</tbody>
</table>

data is computed for the purpose of creating surrogate models on the active subspace. The same training and test data will be utilized for both the averaged-time surrogate models and the time-dependent surrogate models. For the averaged-time surrogate models, we consider averages of quantities of interest over the first 20 time steps. That is, we consider

$$y = \frac{1}{20} \int_0^{20} f(\theta, t) dt,$$

where $f(\theta, t)$ represents the time dependent response. In practice, we utilize trapezoidal rule with the computed training and test data to obtain scalar quantities of interest. We consider the same quantities of interest described in Section 5.1.1: the number of active citizens per 1000 citizens, the number of jailed citizens per 1000 citizens, and the sum of these two - or rate of violence.

We use normalized root mean squared error as a goodness of fit measure throughout this section. This is defined as

$$\frac{1}{N} \sum_{i=1}^{N} \sqrt{\frac{(y_i - \hat{y}_i)^2}{N}},$$

where $\bar{y}$ is the mean of our data set, $y_i$ are the observations at time $t_i$, and $\hat{y}_i$ are the corresponding predictions from the surrogate model.

Averaged-Time Surrogate Models

Prior to creating surrogate models, it is good practice to check correlations between quantities of interest to avoid redundancy in surrogate model creation. In Figure 5.6, we plot the correlations between quantities of interest and in Table 5.1, we provide the corresponding Pearson correlation coefficients. It is clear that the quantities of interest are highly correlated and so we focus on the rate of violence quantity of interest for sensitivity analysis. Indeed, if either of the other quantities of interest was considered, we would obtain similar results.

We recall from (4.1) that we assume normally distributed noise on the observations. We verify this assumption in Figure 5.7, where we plot a normalized histogram for 100 runs of
the Rebellion model utilizing the nominal parameter values given in Table 2.2. Here, we plot the quantity of interest: rate of violence.

We now present the three surrogate models used in our sensitivity analysis. Recall that the data comes from a 20% perturbation in the nominal values of initial officer density and government legitimacy, and the model is run for 200 randomly sampled choices for these parameters in the perturbation ranges. Each run is 20 time steps, and values are averaged over 10 repetitions for each choices of parameters. The test data is obtained similarly and consists of 8 test data sets containing 25 runs each.

**Polynomial Surrogate**

We begin with multivariate polynomial regression, as detailed in Section 3.1. In MATLAB we use the function `MultiPolyRegress` to construct the surrogate. To determine the

![Figure 5.6](image)

**Figure 5.6** Scatter plots depicting correlations between quantities of interest in the Rebellion model with 2 parameters varied.

![Figure 5.7](image)

**Figure 5.7** Normalized histogram for 100 runs of the Rebellion model with default parameter values from Table 2.2. We observe that the data fits a normal distribution with mean 656.53 and standard deviation 25.2009.
optimal order of polynomial, the degree polynomial chosen was that with the lowest BIC score. The BIC scores are provided in Table 5.2. The degree polynomials considered are those that do not produce a rank deficiency within polynomial terms, as the MultiPolyRegress utilizes a QR decomposition to determine the coefficients of the polynomial. We obtain a seventh degree polynomial

$$p(\theta_1, \theta_2) = \sum_{i=1}^{35} \alpha_i \phi_i(\theta_1, \theta_2),$$

where \( \phi_i(\theta_1, \theta_2) \) are basis functions and \( \alpha_i \) are associated weights. The full formula is provided in Appendix C.1. We provide a contour plot of the resulting polynomial surrogate in Figure 5.9a. The corresponding normalized root mean squared errors are given in Table 5.3 with full results provided in Table A.1. The goodness of fit to both the training and test data is provided in Figure B.1.

<table>
<thead>
<tr>
<th>Degree of Polynomial</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIC value</td>
<td>1550.6</td>
<td>1542.5</td>
<td>1350.6</td>
<td>1152.4</td>
<td>1175</td>
<td>1082.4</td>
<td>1042.9</td>
</tr>
</tbody>
</table>

**Gaussian Process Surrogate**

We now use Gaussian process regression as described in Section 3.3. To do so, we must specify the kernel function. Recall that the kernel determines the class of functions on which to perform regression. For this reason, we compute the Gaussian process regression using three different kernels, and employ the best fit choice. We considered the squared exponential, Matern 3/2, and Matern 5/2 kernels with different length scales for each predictor, as presented in Section 3.3. We plot the resulting surrogate models in Figure 5.10 and note that there is not a large graphical difference between the Gaussian processes produced by the different kernels. We note in Table 5.3, however that the Matern 3/2 kernel provides the lowest normalized root mean squared error for the training and test sets. Therefore we use the Matern 3/2 kernel for the Gaussian process surrogate model for the rest of the sensitivity analysis. The fitted Gaussian process surrogate utilizing the Matern 3/2 kernel has hyperparameters \([\sigma_1, \sigma_2, \sigma_f] = [0.2803, 13.6245, 327.6720]\). Full results are provided in Table A.2 and goodness of fit plots are provided in Figure B.2.
Neural Network Surrogate

We utilize neural networks for nonlinear regression as discussed in Section 3.4. Specifically, the regression is performed using a shallow feedforward neural network using 10 neurons in the hidden layer. The number of neurons in the hidden layer was determined in the following way. We ran a neural network regression using multiples of ten neurons in the hidden layer. When we obtained two consecutive models that behaved similarly, we took the smaller of the two models as our desired surrogate model.

The structure of the network is provided in Figure 5.8. The transfer function for the hidden layer is the tan-sigmoid transfer function given by 
\[
tansig(n) = \frac{2}{1 + \exp(-2 \cdot n)} - 1.
\]
The transfer function for the output layer is the identity function.

We plot the resulting model surface in Figure 5.9b and compute the normalized root mean squared errors in Table 5.3 with full results given in Table A.1. To avoid overfitting, the training data set is split into three parts to train the neural network. For this split, we use MATLAB’s default of 70% training, 15% validation, and 15% test. Table 5.3 implies this does a good job of avoiding heavy overfitting as the training data and test data normalized RMSE values are similar. The goodness of fit to both the training and test data is provided in Figure B.1.

We note from Table 5.3 that the Gaussian process regression incurs the most overfitting to the training data. Additionally, the neural network seems to provide the best fit to the test data sets.

The training times for each model and the computation time for evaluating the training data with the surrogate model are provided in Table 5.4. We observe that the polynomial surrogate takes the least amount of time to train and evaluate. Further, the neural network takes the most amount of time to train and evaluate here.

Figure 5.8 Neural network structure for the surrogate model for the 2-dimensional Rebellion model.
**Figure 5.9** (a) 2-D polynomial surrogate for Rebellion model data using a 7th degree polynomial and (b) 2-D neural network surrogate for Rebellion model using 10 neurons in the hidden layer.

**Figure 5.10** 2-D Gaussian process surrogate for Rebellion model using the (a) squared exponential, (b) Matern 3/2, and (c) Matern 5/2 kernels.

**Table 5.3** Normalized RMSE for 2-D surrogate models for the Rebellion model.

<table>
<thead>
<tr>
<th>Surrogate \ Set</th>
<th>Training Data</th>
<th>Mean of Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial</td>
<td>0.0240</td>
<td>0.0415</td>
</tr>
<tr>
<td>Gaussian Process (Squared Exponential Kernel)</td>
<td>0.0141</td>
<td>0.0189</td>
</tr>
<tr>
<td>Gaussian Process (Matern 3/2 Kernel)</td>
<td>0.0116</td>
<td>0.0166</td>
</tr>
<tr>
<td>Gaussian Process (Matern 5/2 Kernel)</td>
<td>0.0128</td>
<td>0.0174</td>
</tr>
<tr>
<td>Neural Network</td>
<td>0.0134</td>
<td>0.0148</td>
</tr>
</tbody>
</table>

**Time-Dependent Surrogate Models**

We now focus on creating a time-dependent surrogate model for the time-dependent rate of violence.

**Polynomial Surrogate**

We fit a polynomial surrogate with the same method as before and obtain a fifth degree
Table 5.4 Computational time for the 2-D averaged-time surrogate models for the Rebellion model, where training time provides the amount of time it takes to train each model and evaluation time provides the amount of time it takes to evaluate the training data utilizing that particular surrogate model.

<table>
<thead>
<tr>
<th>Surrogate</th>
<th>Computation Time</th>
<th>Training Time</th>
<th>Evaluation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial</td>
<td>0.078004</td>
<td>0.000850</td>
<td></td>
</tr>
<tr>
<td>Gaussian Process (Squared Exponential Kernel)</td>
<td>0.234975</td>
<td>0.014539</td>
<td></td>
</tr>
<tr>
<td>Gaussian Process (Matern 3/2 Kernel)</td>
<td>0.095283</td>
<td>0.006831</td>
<td></td>
</tr>
<tr>
<td>Gaussian Process (Matern 5/2 Kernel)</td>
<td>0.088154</td>
<td>0.008513</td>
<td></td>
</tr>
<tr>
<td>Neural Network</td>
<td>3.122076</td>
<td>0.095513</td>
<td></td>
</tr>
</tbody>
</table>

polynomial surrogate

\[ p(\theta_1, \theta_2, t) = \sum_{i=1}^{55} \alpha_i \phi_i(\theta_1, \theta_2, t), \]

where \( \phi_i(\theta_1, \theta_2, t) \) are basis functions and \( \alpha_i \) are associated weights. The full formula is provided in Appendix C.2. We provide a goodness of fit plot in Figure B.3.

**Gaussian Process Surrogate**

We fit a Gaussian process surrogate model using the ARD Matern 3/2 kernel, remaining consistent with the previous choice. The fitted Gaussian process surrogate utilizing the Matern 3/2 kernel has hyperparameters \([\sigma_1, \sigma_2, \sigma_3, \sigma_f] = [0.3890, 11.9817, 16.3342, 717.4624]\). A goodness of fit plot is provided in Figure B.3.

**Neural Network Surrogate**

We fit a neural network surrogate with 20 neurons in the hidden layer. Again, this value was determined in the manner previously described. The structure of the network is the same as in Figure 5.8, but with 20 neurons in the hidden layer. We plot a goodness of fit plot in Figure B.3.

Corresponding normalized RMSE values for all time dependent surrogate models are provided in Table 5.5 with full results provided in Table A.3. Note that the Gaussian process surrogate provides the closest fit to the agent-based model data. Further, the neural network provides a similar RMSE value as the Gaussian process.

We provide the computation time for training and evaluating the training data on the 2-D time-dependent surrogate models in Table 5.6. Note that the polynomial surrogate trains
and evaluates the fastest, while the Gaussian process surrogate trains and runs the slowest. From Tables 5.5 and 5.6, the neural network provides a balance between computational efficiency and accuracy in predictions.

### 5.1.3 Sensitivity Analysis

Here we perform sensitivity analysis on the 2-D surrogate models for the Rebellion model to determine if either parameter is noninfluential. We follow the order the sensitivity analysis methods are outlined in from Chapter 4. That is, we first perform local sensitivity analysis with parameter subset selection, followed by global sensitivity analysis utilizing one-at-a-time methods, Pearson correlation coefficients, Sobol’ indices, and Morris screening. We follow this up with active subspace analysis and lastly, verify the results and present our conclusions. The sensitivity analysis ranges for the parameters government legitimacy and initial officer density come from a 20% perturbation about their nominal values and are $[0.64, 0.96]$ and $[3.2, 4.8]$, respectively.

#### Parameter Subset Selection

We begin by applying parameter subset selection locally to the time-dependent surrogate models at the nominal values of the parameters government legitimacy and initial officer density. This means that at parameter value $\theta^* = [0.8, 4]^T$, the sensitivity matrix is of the

<table>
<thead>
<tr>
<th>Surrogate \ Set</th>
<th>Training Data</th>
<th>Mean of Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial</td>
<td>0.1028</td>
<td>0.1268</td>
</tr>
<tr>
<td>Gaussian Process (Matern 3/2 Kernel)</td>
<td>0.0176</td>
<td>0.0270</td>
</tr>
<tr>
<td>Neural Network</td>
<td>0.0251</td>
<td>0.0310</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Surrogate \ Computation Time</th>
<th>Training Time</th>
<th>Evaluation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial</td>
<td>0.217728</td>
<td>0.010482</td>
</tr>
<tr>
<td>Gaussian Process (Matern 3/2 Kernel)</td>
<td>251.462538</td>
<td>0.387969</td>
</tr>
<tr>
<td>Neural Network</td>
<td>4.841998</td>
<td>0.092707</td>
</tr>
</tbody>
</table>
form

\[ \chi = \begin{bmatrix}
\frac{\partial f}{\partial \theta_1}(1, \theta^*) & \frac{\partial f}{\partial \theta_2}(1, \theta^*) \\
\vdots & \vdots \\
\frac{\partial f}{\partial \theta_1}(20, \theta^*) & \frac{\partial f}{\partial \theta_2}(20, \theta^*)
\end{bmatrix}. \]

We then compute the SVD of \( \chi = USV^T \) for the three surrogate models

\[
S_P = \begin{bmatrix} 82.718 & 0 \\ 0 & 16092 \end{bmatrix}, \quad V_P = \begin{bmatrix} -0.013697 & 0.99991 \\ 0.99991 & 0.013697 \end{bmatrix}, \tag{5.1}
\]

\[
S_{GP} = \begin{bmatrix} 112.37 & 0 \\ 0 & 12652 \end{bmatrix}, \quad V_{GP} = \begin{bmatrix} -0.019054 & 0.99982 \\ 0.99982 & 0.019054 \end{bmatrix}, \tag{5.2}
\]

\[
S_{NN} = \begin{bmatrix} 127.72 & 0 \\ 0 & 13242 \end{bmatrix}, \quad V_{NN} = \begin{bmatrix} -0.021293 & 0.99977 \\ 0.99977 & 0.021293 \end{bmatrix}. \tag{5.3}
\]

Where (5.1), (5.2), and (5.3) are the computed \( S \) and \( V \) matrices from the SVD decomposition of \( \chi \) for the polynomial, Gaussian process, and neural network surrogates, respectively.

We note for each \( S \) matrix that the smallest singular value is not close to 0, but that \( \sigma_1 \ll \sigma_2 \). Thus we consider the largest component of \( v_1 \) to identify the parameter that is not identifiable. Here, that indicates that initial officer density is not identifiable.

**One-at-a-Time Sensitivity Analysis**

For one-at-a-time analysis, we consider all three surrogate models and data from the agent-based Rebellion model. Data from the agent-based model came from the averaged rate of violence over 10 repetitions for each parameter choice. Recall that for one-at-a-time sensitivity analysis, we hold all parameters constant at their default values except for one which we vary over it’s admissible parameter space. Here, we considered the default values of government legitimacy and initial officer density to be 0.8 and 4. The corresponding sensitivity analysis intervals considered were [0.64, 0.96] and [3.2, 4.8]. We plot the results of the one-at-a-time analysis in Figure 5.11, and observe that the parameter government legitimacy is more influential to the response. Further, we observe that all three surrogate models provide a reasonable fit to the data.

**Pearson Correlation**

We plot in Figure 5.12 a scatter plot of the training data for the surrogate models. We observe a much stronger relationship between government legitimacy and the rate of violence than between initial officer density and the rate of violence. This is supported in Table 5.7, where
we observe that the Pearson correlation coefficient computed as described in Section 4.3.2 for government legitimacy is close to 1 in absolute value, whereas the Pearson correlation coefficient for initial officer density is closer to 0 in absolute value. The Pearson correlation coefficients were computed with the MATLAB `corr` command.

**Figure 5.11** One-at-a-time sensitivity analysis of the 2-D averaged-time surrogate models for the Rebellion model for (a) government legitimacy and (b) initial officer density.

**Figure 5.12** Correlation between (a) government legitimacy and rate of violence and (b) initial officer density and rate of violence.

**Table 5.7** Pearson correlation coefficients for government legitimacy and initial officer density to the rate of violence.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Government Legitimacy</th>
<th>Initial Cop Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pearson Correlation Coefficient</td>
<td>-0.9752</td>
<td>-0.1593</td>
</tr>
</tbody>
</table>
Sobol’ Sensitivity Indices

We now compute Sobol’ indices using Algorithm 2 and report the first-order Sobol’ indices in Table 5.8 and the total Sobol’ indices in Table 5.9. These values were computed using a psuedo-random sample of $M = 10^5$ points. To determine the final value of $M$, we increment $M$ by powers of 10 until two consecutive powers of 10 provided similar results. The Sobol’ indices imply that our models are more sensitive to changes in government legitimacy than in initial officer density. Additionally, the Sobol’ indices are consistent for the three surrogate models. Note that $S_1 + S_2 \approx 1$ in Table 5.8, implying that the two parameters do not have large interaction terms.

Time-Dependent Sobol’ Indices

We plot the time-dependent Sobol’ indices in Figure 5.13, calculated using the method outlined in Section 4.3.3. They were computed using $10^5$ psuedo-random samples and indicate that government legitimacy is the most influential parameter for all times $t = 1$ to $t = 20$.

Morris Screening

The Morris screening results calculated via Algorithm 3 are provided in Tables 5.10 and 5.11. We used $R = 100$ samples and a step size of $\Delta = 10^{-4}$ multiplied by the nominal value of the parameters to obtain these values. Variation in the number of samples and step size used will not influence results. Table 5.10 identifies initial officer density as the least influential parameter. This result is consistent across all three surrogate models.

Table 5.8 First-order Sobol’ indices for parameters calculated using the surrogate models for the 2-D Rebellion model.

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$ (Government Legitimacy)</td>
<td>0.9865</td>
<td>0.9886</td>
<td>0.9889</td>
</tr>
<tr>
<td>$S_2$ (Initial Officer Density)</td>
<td>0.0075</td>
<td>0.0067</td>
<td>0.0063</td>
</tr>
</tbody>
</table>

Table 5.9 Total Sobol’ indices for parameters calculated using the surrogate models for the 2-D Rebellion model.

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_T_1$ (Government Legitimacy)</td>
<td>0.9900</td>
<td>0.9918</td>
<td>0.9924</td>
</tr>
<tr>
<td>$S_T_2$ (Initial Officer Density)</td>
<td>0.0097</td>
<td>0.0082</td>
<td>0.0079</td>
</tr>
</tbody>
</table>
Figure 5.13 Generalized Sobol’ indices where (a) and (d), (b) and (e), and (c) and (f) utilize the polynomial, Gaussian process, and neural network surrogates, respectively. The first row provides the generalized first-order Sobol’ indices and the second row provides the generalized total Sobol’ indices. Recall that $S_1$ corresponds to government legitimacy and $S_2$ corresponds to initial officer density.

Table 5.10 Mean $\mu^*_i$ from Morris screening performed using surrogate models for the 2-D Rebellion model.

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu^*_1$ (Government Legitimacy)</td>
<td>2222.8</td>
<td>1950.2</td>
<td>1943.1</td>
</tr>
<tr>
<td>$\mu^*_2$ (Initial Officer Density)</td>
<td>60.007</td>
<td>39.325</td>
<td>39.374</td>
</tr>
</tbody>
</table>

Table 5.11 Standard deviation $\sigma_i$ from Morris screening performed using surrogate models for the 2-D Rebellion model.

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_1$ (Government Legitimacy)</td>
<td>200.57</td>
<td>235.05</td>
<td>245.59</td>
</tr>
<tr>
<td>$\sigma_2$ (Initial Officer Density)</td>
<td>132.33</td>
<td>4.7789</td>
<td>5.3402</td>
</tr>
</tbody>
</table>

**Activity Scores**

We apply the active subspace methods detailed in Section 4.4, to the surrogate models to compute activity scores using the relation (4.5). We computed the active subspaces and activity scores using $10^4$ independent pseudo-random samples. This value was determined in the same manner as for the Sobol’ indices. The gradients were computed with automatic
differentiation on the polynomial and Gaussian process surrogates, and via finite difference methods on the neural network. For the finite difference methods, we employed a step size of $10^{-6}$ multiplied by the nominal value. We verified this choice of step size by varying the step size within the range of $10^{-4}$ and $10^{-8}$ and noting negligible changes in results. We used finite difference for the neural network gradient to circumvent the need to decompose the neural network structure in automatic differentiation.

The active subspaces were verified by creating a surrogate model on the active subspace and comparing its performance to the original model and surrogates. We plot these results in Figure 5.14. The activity scores in Table 5.12 demonstrate that the quantity of interest rate of violence is more sensitive to the parameter government legitimacy. Additionally, the activity scores have similar magnitudes for all surrogate models.

We verify that the activity score and total Sobol’ index bounds hold. Recall from Section 4.4, that the total Sobol’ indices can be bounded by activity scores in the following manner:

$$S_{T_i} \leq \frac{1}{4\pi^2 V}(a_i(n) + \lambda_{n+1}) = B_i.$$  

Here $V = \text{Var}[f]$, $a_i(n)$ is the activity score utilizing an $n$ dimensional active subspace, and $\lambda_{n+1}$ is the $(n+1)$st eigenvalue of $C$ [10]. We compute the values of $S_{T_i}$ and $B_i$ in Table 5.13. The bounds hold for the activity scores and total Sobol’ indices computed relating to the parameter government legitimacy but they do not hold for the parameter initial officer density.

![Figure 5.14](image)

**Figure 5.14** Active subspace verification for the surrogate models for the 2-D Rebellion model. (a), (b), and (c) correspond to the polynomial, Gaussian process, and neural network surrogates, respectively.

**Table 5.12** Activity scores for parameters calculated with surrogate models for the 2-D Rebellion model.

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$ (Government Legitimacy)</td>
<td>7.3069e+06</td>
<td>6.8907e+06</td>
<td>7.0956e+06</td>
</tr>
<tr>
<td>$a_2$ (Initial Officer Density)</td>
<td>1.3682e+03</td>
<td>1.6907e+03</td>
<td>1.8104e+03</td>
</tr>
</tbody>
</table>

71
Table 5.13 Total Sobol’ indices and activity score bounds for the 2-dimensional Rebellion model.

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Government Legitimacy</td>
<td>$S_{T_1}$</td>
<td>$B_{T_1}$</td>
<td>$S_{T_1}$</td>
</tr>
<tr>
<td></td>
<td>0.9900</td>
<td>3.3619</td>
<td>0.9886</td>
</tr>
<tr>
<td></td>
<td>$S_{T_1}$</td>
<td>$B_{T_1}$</td>
<td>$S_{T_1}$</td>
</tr>
<tr>
<td></td>
<td>0.9924</td>
<td>3.2737</td>
<td>0.9924</td>
</tr>
<tr>
<td>Initial Officer Density</td>
<td>$S_{T_2}$</td>
<td>$B_{T_2}$</td>
<td>$S_{T_2}$</td>
</tr>
<tr>
<td></td>
<td>0.0097</td>
<td>0.0006</td>
<td>0.0067</td>
</tr>
<tr>
<td></td>
<td>$S_{T_2}$</td>
<td>$B_{T_2}$</td>
<td>$S_{T_2}$</td>
</tr>
<tr>
<td></td>
<td>0.0008</td>
<td>0.0008</td>
<td>0.0008</td>
</tr>
</tbody>
</table>

density.

Upon computation of the covariance matrix, we observe that the parameters are not independent, which violates the assumption for utilizing Sobol’ indices and active subspaces. However, for this case, we still observe consistent results for the various methods. Note that active subspace methods are more robust to the violation of the assumption of independence than Sobol’ indices are.

**Time-Dependent Activity Scores**

We plot activity scores computed independently at time steps 1-20 in Figure 5.15. We observe that government legitimacy is again the more influential parameter at all time steps, except for $t = 1$ with the Gaussian process surrogate. These values were determined using $M = 10^4$ pseudo-random samples for the parameters at each time step. To facilitate computations, we computed the activity scores on the full space. This eludes the need to confirm the size of the active subspace at each time step.

The activity scores computed via the time-dependent surrogate models utilizing the block structure detailed in Section 4.4.4 can be seen in Table 5.14. These values were computed using $M = 10^4$ pseudo-random samples. We considered time as both a discrete and

![Figure 5.15](image-url) **Figure 5.15** Activity scores over time utilizing the (a) polynomial, (b) Gaussian process, and (c) neural network surrogates. Recall that $a_1$ corresponds to government legitimacy and $a_2$ corresponds to initial officer density.
continuous variable on the interval $[1, 20]$. When we considered time as a discrete variable, it was randomly sampled from the set $\{1, 2, ..., 20\}$. When we took time as a continuous variables, it was randomly sampled from the uniform distribution on $[1, 20]$. We note that for the the three surrogates, we get agreement with the activity scores computed using the averaged-time surrogates.

Table 5.14 Activity scores for parameters calculated with surrogate models for the 2-D time-dependent Rebellion model.

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$ (Government Legitimacy)</td>
<td>7.85e+06</td>
<td>8.341e+06</td>
<td>8.2977e+06</td>
</tr>
<tr>
<td>$a_2$ (Initial Officer Density)</td>
<td>5.1258e+03</td>
<td>2.613e+03</td>
<td>4.0806e+03</td>
</tr>
</tbody>
</table>

Time as a continuous variable

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$ (Government Legitimacy)</td>
<td>7.9605e+06</td>
<td>8.6707e+06</td>
<td>8.6187e+06</td>
</tr>
<tr>
<td>$a_2$ (Initial Officer Density)</td>
<td>4.9787e+03</td>
<td>2.7868e+03</td>
<td>4.1458e+03</td>
</tr>
</tbody>
</table>

Verification

We verify the results qualitatively by plotting the distribution of the response variable when both parameters are randomly sampled from the sensitivity analysis intervals $[0.64, 0.96]$ and $[3.2, 4.8]$ for government legitimacy and initial officer density, and when keeping initial officer density fixed. We utilize $10^4$ psuedo-random samples. We note in Figure 5.16 that the two distributions have strong agreement for all surrogate models.

Figure 5.16 Distribution of response when both parameters are randomly samples from the sensitivity analysis intervals and when keeping initial officer density (ICD) fixed, using the averaged-time (a) polynomial, (b) Gaussian process, and (c) neural network surrogate models for the 2-D Rebellion model.
We further verify our results by computing test statistics for each surrogate model and provide the results in Figure 5.17. These values were obtained with the same samples used for Figure 5.16. For the polynomial and neural network regression, the test statistic is smaller than both critical values and hence we do not reject the null hypothesis. This indicates that the two sets of samples are likely from the same distribution.

**Conclusion**

We conclude that government legitimacy is the most influential parameter for this model when considering only the parameters government legitimacy and initial officer density. This conclusion is supported by all sensitivity analysis and active subspace methods considered. The novel parts of this analysis was the use of a neural network surrogate model for an agent-based model, the use of activity scores, and the use of parameter subset selection. We note that the neural network surrogate model performed similarly to the Gaussian process and was more accurate than the polynomial surrogate. Further, the neural network saved significant computation time for the time-dependent surrogate models. The computation of activity scores requires significantly less evaluations than the computation of Sobol’ indices. Here, activity scores required computing the gradient at $10^4$ values, whereas Sobol’ indices required function evaluation at $10^5$ values. We note that all sensitivity analysis and active subspace analysis methods utilized assume parameter independence, which is not the case here. However the sensitivity analysis results remain consistent across all methods utilized, including computation of activity scores.

![Figure 5.17](image)

**Figure 5.17** Energy statistic results for the 2-D averaged-time (a) polynomial, (b) Gaussian process, and (c) neural network surrogates. Information included is the pooled set of ordered replicates, test statistics $T_{n_1,n_2}$, and critical values $T_c$ at 95% and 99% confidence levels.
5.2 Rebellion Model with \( p = 5 \) Parameters

We generate the data using a 20\% perturbation in the nominal values of all five parameters of the Rebellion model: initial officer density, initial agent density, vision, government legitimacy, and maximum jail term. We then create surrogate models based on these parameters, where \( \theta_1 \) is initial officer density, \( \theta_2 \) is initial agent density, \( \theta_3 \) is vision, \( \theta_4 \) is government legitimacy, and \( \theta_5 \) is maximum jail term. We run the model 500 times with random choices for these parameters randomly sampled from the uniform input distributions. Each run is 20 time steps. We average values over 10 repetitions for each choice of parameters. The test data is obtained similarly and consists of 10 test data sets containing 50 runs each. Additionally, we compute a second set of training data for the purpose of creating surrogate models on the active subspace. The same training and test data will be utilized for both the averaged-time surrogate models and the time-dependent surrogate models. We consider the same responses as described in Section 5.1.2: the number of actively rebellious citizens per 1000 citizens, the number of jailed citizens per 1000 citizens, and the rate of violence which is the sum of these two.

5.2.1 Surrogate Models

We again create multivariate polynomial, Gaussian process, and neural network surrogate models for the averaged-time and time-dependent responses using the techniques detailed in Chapter 3. We compare results utilizing each surrogate model to ensure consistency.

Averaged-Time Surrogate Models

Prior to creating surrogate models, we check the correlation between quantities of interest. We plot scatter plots of the correlations between quantities of interest in Figure 5.18. The corresponding Pearson correlation coefficients are provided in Table 5.15. These indicate that the quantities of interest are highly correlated and would produce similar results. We focus on the rate of violence quantity of interest. We note that the other quantities of interest yielded analogous results.
**Figure 5.18** Scatter plots depicting the correlation between quantities of interest in the Rebellion model with 5 parameters.

**Table 5.15** Pearson correlation coefficients for quantities of interest in the Rebellion model with 5 parameters.

<table>
<thead>
<tr>
<th></th>
<th>Number of Active Citizens</th>
<th>Number of Jailed Citizens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate of Violence</td>
<td>0.94634</td>
<td>0.97796</td>
</tr>
<tr>
<td>Number of Jailed Citizens</td>
<td>0.85799</td>
<td></td>
</tr>
</tbody>
</table>

**Polynomial Surrogate**

The degree of the polynomial surrogate is determined via the BIC scores to be degree 4. Again, we consider only the degree polynomials that do not produce rank deficiency within polynomial terms, i.e., we considered degrees 1-4. We provide the BIC scores for these degree polynomials in Table 5.16, where we observe that the fourth degree polynomial has the lowest BIC score.

We obtain a polynomial

$$p(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5) = \sum_{i=1}^{125} \alpha_i \phi_i(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5),$$

where $\phi_i(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$ are basis functions and $\alpha_i$ are associated weights. The full formula is provided in Appendix C.3. A goodness of fit plot is provided in Figure 5.19 and the corresponding normalized root mean squared error is provided in Table 5.17 with full results.

**Table 5.16** BIC scores for averaged-time polynomial surrogate models of the 5-D Rebellion model.

<table>
<thead>
<tr>
<th>Degree</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIC score</td>
<td>3.9828</td>
<td>4.0004</td>
<td>3.7309</td>
<td>3.5825</td>
</tr>
</tbody>
</table>
Figure 5.19 Goodness of fit of the fourth degree polynomial averaged-time surrogate model for the 5-D Rebellion model, where (a) shows the goodness of fit of the training data and (b) shows the goodness of fit of the test data.

provided in Table A.4.

**Gaussian Process Surrogate**

We compute the Gaussian process surrogate using three different kernels, the squared exponential, Matern 3/2, and Matern 5/2, as provided in Section 3.3. We use the versions of these with different correlation lengths for each predictor, as described in (3.5). The resulting fit of these models is provided in Figure 5.20 and the normalized RMSE is provided in Table 5.17 with full results provided in Table A.5. These fits indicate that the Matern 3/2 kernel provides the best fit surrogate model. Thus, we utilize the Gaussian process surrogate with the Matern 3/2 kernel for the sensitivity analysis. The kernel hyperparameters for the Gaussian process surrogate with the Matern 3/2 kernel is $[\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_f] = [7.2728, 264.3420, 16.2426, 0.2298, 549.6688, 303.7374]^T$.

**Neural Network Surrogate**

We fit a neural network surrogate with 10 neurons in the hidden layer. We determined the number of neurons in the hidden layer in the same manner as for the 2-D surrogate models. We provide goodness of fit plots in Figure 5.21 and normalized RMSE values in Table 5.17, with full results in Table A.4.

Note that Table 5.17 implies the Gaussian process model fits the data the best out of all the surrogate models, only slightly outperforming the neural network. Additionally, the
neural network does less overfitting to the training data. The polynomial surrogate model is the least effective at providing an accurate fit to the training and test data.

We provide computational times for training and evaluating the three surrogate models in Table 5.18. The polynomial surrogate trains and runs the fastest, but provides the least accurate fit to the training and test data. The Gaussian process surrogate model trains faster and evaluates faster than the neural network and has the best fit to the training and test data.

Table 5.17 Normalized RMSE for 5-D averaged-time surrogate models for the Rebellion model.

<table>
<thead>
<tr>
<th>Surrogate \ Set</th>
<th>Training Data</th>
<th>Mean of Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial</td>
<td>0.0521</td>
<td>0.0723</td>
</tr>
<tr>
<td>Gaussian Process (Squared Exponential Kernel)</td>
<td>0.0174</td>
<td>0.0293</td>
</tr>
<tr>
<td>Gaussian Process (Matern 3/2 Kernel)</td>
<td>0.0105</td>
<td>0.0270</td>
</tr>
<tr>
<td>Gaussian Process (Matern 5/2 Kernel)</td>
<td>0.0140</td>
<td>0.0274</td>
</tr>
<tr>
<td>Neural Network</td>
<td>0.0252</td>
<td>0.0286</td>
</tr>
</tbody>
</table>
Figure 5.21 Goodness of fit of the averaged-time neural network surrogate model for the 5-D Rebellion model, where (a) shows the goodness of fit of the training data and (b) shows the goodness of fit of the test data.

Table 5.18 Computation time for the 5-D averaged-time surrogate models for the Rebellion model, where training time is the amount of time (in seconds) it takes to train each model and evaluation time is the amount of time (in seconds) it takes to evaluate the training data utilizing that particular surrogate model.

<table>
<thead>
<tr>
<th>Surrogate</th>
<th>Computation Time</th>
<th>Training Time</th>
<th>Evaluation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial</td>
<td></td>
<td>0.044541</td>
<td>0.001164</td>
</tr>
<tr>
<td>Gaussian Process (Matern 3/2 Kernel)</td>
<td>1.112827</td>
<td>0.015294</td>
<td></td>
</tr>
<tr>
<td>Neural Network</td>
<td>3.130252</td>
<td>0.104787</td>
<td></td>
</tr>
</tbody>
</table>

Time Dependent Surrogate Models

We also compute time-dependent surrogate models for the 5-D Rebellion model for the time-dependent rate of violence quantity of interest.

Polynomial Surrogate

We follow the same procedure of considering polynomials of degrees that don't produce rank deficiencies in computation and take the degree with the lowest BIC score as the desired surrogate mode. The BIC values are calculated in Table 5.19 and we note that the third degree polynomial has the smallest BIC score. We obtain a polynomial

\[ p(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, t) = \sum_{i=1}^{83} \alpha_i \phi_i(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, t), \]

where \( \phi_i(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, t) \) are basis functions and \( \alpha_i \) are associated weights, via minimiz-
Table 5.19 BIC values for the time-dependent polynomial surrogate model for the 5-D Rebellion model.

<table>
<thead>
<tr>
<th>Degree of Polynomial</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIC value</td>
<td>6340</td>
<td>6275.9</td>
<td>6274.1</td>
<td>6696.8</td>
</tr>
</tbody>
</table>

ing the BIC score. The full formula is provided in Appendix C.4. A goodness of fit plot is provided in Figure 5.22 and the corresponding normalized root mean squared error is provided in Table 5.20 with full results provided in Table A.6.

**Gaussian Process Surrogate**

We compute the Gaussian process surrogate using the Matern 3/2 kernel with different length scales for each predictor. The goodness of fit of this model is provided in Figure 5.23 and the normalized RMSE is provided in Table 5.20 with full results provided in Table A.6. The kernel parameters for the Gaussian process surrogate are $[\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_f]^T = [13.602, 415.68, 26.645, 0.62829, 715.5, 36.666, 1872.3]^T$, where the first five components correspond to the length scales for the five parameter inputs, the sixth component is the length scale for the time input, and the last component corresponds to the signal standard deviation.

![Figure 5.22](image-url)  

**Figure 5.22** Goodness of fit of the third degree polynomial surrogate for the 5-D time-dependent Rebellion model. (a) Goodness of fit of the training data and (b) goodness of fit of the test data.
Neural Network Surrogate

We fit a neural network surrogate with 40 neurons in the hidden layer, determined as previously described. We provide goodness of fit plots in Figure 5.24 and normalized RMSE values in Table 5.20 with full results in Table A.6.

We observe from Tables 5.20 and 5.21 that the polynomial surrogate is the fastest model to train and evaluate, but has the highest RMSE values, and does not provide a close fit to the training or test data. Further, the neural network outperforms the Gaussian process in computation time for both training and evaluation and in the RMSE value on the test data.
Table 5.20 Normalized RMSE for the 5-D time-dependent surrogate models for the Rebellion model.

<table>
<thead>
<tr>
<th>Surrogate \ Set</th>
<th>Training Data</th>
<th>Mean of Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial</td>
<td>0.2218</td>
<td>0.2893</td>
</tr>
<tr>
<td>Gaussian Process (Matern 3/2 Kernel)</td>
<td>0.0026</td>
<td>0.0612</td>
</tr>
<tr>
<td>Neural Network</td>
<td>0.0324</td>
<td>0.0588</td>
</tr>
</tbody>
</table>

Table 5.21 Computation time for the 5-D time-dependent surrogate models for the Rebellion model, where training time refers to the amount of time (in seconds) it takes to train each model and evaluation time refers to the amount of time (in seconds) it takes to evaluate the training data utilizing that particular surrogate model.

<table>
<thead>
<tr>
<th>Surrogate \ Computation Time</th>
<th>Training Time</th>
<th>Evaluation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial</td>
<td>0.373915</td>
<td>0.010245</td>
</tr>
<tr>
<td>Gaussian Process</td>
<td>8763.988430</td>
<td>4.370799</td>
</tr>
<tr>
<td>Neural Network</td>
<td>14.667369</td>
<td>0.092156</td>
</tr>
</tbody>
</table>

sets. The Gaussian process performs the best on the training data set, indicating overfitting that is occurring. The neural network outperforms the Gaussian process in this case as the training data set is so large. Further, as the number of parameters in the model increases, the Gaussian process surrogate will require a larger increase in computation time than the neural network will.

5.2.2 Sensitivity Analysis

We perform sensitivity analysis on the 5-dimensional surrogate models for the Rebellion model to determine influential parameters. We begin with parameter subset selection to identify what parameters are identifiable locally. We then apply the global sensitivity analysis of Section 4.3 to perform one-at-a-time sensitivity analysis, and determine Pearson correlation coefficients, Sobol’ indices, and Morris screening. Additionally, we apply the active subspace techniques of Section 4.4. Lastly we verify the results from the sensitivity and active subspace analysis by utilizing energy statistics, detailed in Section 4.5.
Parameter Subset Selection

We perform parameter subset selection at the nominal values of the parameters \( \theta^* = [4, 70, 7, 0.8, 30]^T \). This means that at \( \theta^* \), the sensitivity matrix is

\[
\chi = \begin{bmatrix}
\frac{\partial f}{\partial \theta_1}(1, \theta^*) & \cdots & \frac{\partial f}{\partial \theta_5}(1, \theta^*) \\
\vdots & \ddots & \vdots \\
\frac{\partial f}{\partial \theta_1}(20, \theta^*) & \cdots & \frac{\partial f}{\partial \theta_5}(20, \theta^*)
\end{bmatrix}.
\]

We compute the SVD of \( \chi \) and progress in the same format as detailed in Algorithm 1 for each surrogate model. Here, we detail the steps and results for the time-dependent polynomial surrogate model and comment that the steps and results for the Gaussian process and neural network surrogate models are analogous.

For the time-dependent polynomial surrogate model, we compute singular values

\[
\sigma_1 = 3.935e-16 \leq 2.6926e^{-15} \leq 27.994 \leq 71.693 \leq 15563 = \sigma_5.
\]

As \( \sigma_1 \) is close to 0, we determine the largest component of \( v_1 \) is the component corresponding to maximum jail term. Thus maximum jail term is not identifiable and column 5 is removed from the sensitivity matrix. We recompute the SVD of \( \chi_{\sim 5} \), where \( \sim 5 \) indicates that column 5 has been removed. The computed singular values of \( \chi_{\sim 5} \) are

\[
\sigma_1 = 2.6898e^{-15} \leq 27.983 \leq 71.692 \leq 15563 = \sigma_4.
\]

Again, \( \sigma_1 \) is close to 0, so we identify the largest component of \( v_1 \) as the component corresponding to initial agent density and thus, column 2 is removed from \( \chi_{\sim 5} \). The computed singular values from the SVD of \( \chi_{\sim 2,5} \) are

\[
\sigma_1 = 27.939 \leq 71.649 \leq 15563 = \sigma_3.
\]

Here, \( \sigma_1 \) is not close to 0, but \( \sigma_1, \sigma_2 << \sigma_3 \), and thus, we again consider the largest component of \( v_1 \) to determine a unidentifiable parameter. This time, we obtain that the parameter vision is not identifiable. We repeat this process one more time with \( \chi_{\sim 2,3,5} \) to obtain the singular values

\[
\sigma_1 = 70.818 \ll 15563 = \sigma_2
\]

and determine the parameter initial officer density to be unidentifiable. Thus, there is only one identifiable parameter, and that is government legitimacy. Again, we note that the process is identical for the Gaussian process and neural network surrogates and that we
Figure 5.25 One-at-a-time plots for the 5 parameters of the Rebellion model for the averaged-time response. Plots include data from the agent-based model and the polynomial, Gaussian process, and neural network surrogate models.

observe the same result.

One-at-a-Time Sensitivity Analysis

We plot the results of one-at-a-time sensitivity analysis in Figure 5.25. Data from the agent-based model was computed by setting all but one parameter to their default values provided in Table 2.2. The parameter that was not fixed was varied about 20% of its nominal value. For each parameter set, the Rebellion model was run 10 times, and the average over these ten times was taken as the data point. We note that the Gaussian process and neural network surrogate models look identical in the plots and that the polynomial provides a slightly different fit to the data. Further, we observe from Figure 5.25, that the rate of violence response varies most as government legitimacy is varied.

Pearson Correlation

We provide a scatter plot of the training data for the surrogate models in Figure 5.26. We observe the strongest relationship between government legitimacy and the rate of violence. The other parameters appear less influential. This is supported in Table 5.22, where we observe that the Pearson correlation coefficient for government legitimacy is close to 1 in
Figure 5.26 Scatter plots depicting correlation between the rate of violence and (a) initial officer density, (b) initial agent density, (c) vision, (d) government legitimacy, and (e) maximum jail term.

Table 5.22 Pearson correlation coefficients for rate of violence to initial officer density, initial agent density, vision, government legitimacy, and maximum jail term

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Pearson Correlation Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Officer Density</td>
<td>-0.0623</td>
</tr>
<tr>
<td>Initial Agent Density</td>
<td>0.0149</td>
</tr>
<tr>
<td>Vision</td>
<td>-0.0565</td>
</tr>
<tr>
<td>Government Legitimacy</td>
<td>-0.9712</td>
</tr>
<tr>
<td>Maximum Jail Term</td>
<td>0.0530</td>
</tr>
</tbody>
</table>

absolute value, while the others are close to 0 in absolute value.

Sobol’ Sensitivity Indices

We compute Sobol’ indices utilizing Algorithm 2 and report the first-order Sobol’ indices in Table 5.23 and the total Sobol’ indices in Table 5.24. These values were computed using a pseudo-random sample of $M = 10^6$ points. To find the value of $M$, we incremented $M$ by powers of 10 until two powers of 10 yielded similar results. The Sobol’ indices imply that the model is most sensitive to changes in the parameter government legitimacy. This result is consistent across all surrogate models.

We observe that $S_i \approx S_{Tj}$ for the five parameters and three surrogate models. Further,
Table 5.23 First-order Sobol’ indices for parameters calculated using the averaged-time surrogate models for the 5-D Rebellion model.

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$ (Initial Officer Density)</td>
<td>0.0071174</td>
<td>0.0075291</td>
<td>0.0068488</td>
</tr>
<tr>
<td>$S_2$ (Initial Agent Density)</td>
<td>0.0034936</td>
<td>0.003966</td>
<td>0.0038788</td>
</tr>
<tr>
<td>$S_3$ (Vision)</td>
<td>0.0012298</td>
<td>0.0015835</td>
<td>0.0014074</td>
</tr>
<tr>
<td>$S_4$ (Government Legitimacy)</td>
<td>0.98072</td>
<td>0.98079</td>
<td>0.9823</td>
</tr>
<tr>
<td>$S_5$ (Maximum Jail Term)</td>
<td>0.0001022</td>
<td>6.2464e-05</td>
<td>3.9193e-05</td>
</tr>
</tbody>
</table>

Table 5.24 Total Sobol’ indices for parameters calculated using the averaged-time surrogate models for the 5-D Rebellion model.

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{T1}$ (Initial Officer Density)</td>
<td>0.011733</td>
<td>0.011005</td>
<td>0.0095859</td>
</tr>
<tr>
<td>$S_{T2}$ (Initial Agent Density)</td>
<td>0.0062816</td>
<td>0.00564</td>
<td>0.0054397</td>
</tr>
<tr>
<td>$S_{T3}$ (Vision)</td>
<td>0.0041393</td>
<td>0.003705</td>
<td>0.0031603</td>
</tr>
<tr>
<td>$S_{T4}$ (Government Legitimacy)</td>
<td>0.98649</td>
<td>0.98711</td>
<td>0.98824</td>
</tr>
<tr>
<td>$S_{T5}$ (Maximum Jail Term)</td>
<td>0.0016311</td>
<td>0.00015437</td>
<td>0.00027146</td>
</tr>
</tbody>
</table>

$\sum_{i=1}^{5} S_i \approx 1$ for the three surrogate models. These two observations imply that there is minimal interaction between parameters. Thus the independence assumption made for Sobol’ analysis is not unreasonable.

**Time-Dependent Sobol’ Indices**

The generalized Sobol’ indices are computed using the methods detailed in Section 4.3.3 and are provided in Figure 5.27. They were computed using $10^4$ psuedo-random samples and indicate that government legitimacy is the most influential parameter for all time from $t = 1$ to $t = 20$. This is consistent with the results from the averaged-time Sobol’ indices.

**Morris Screening**

We plot Morris screening results computed via Algorithm 3 in Figure 5.28. We use $R = 100$ samples and a step size of $\Delta = 10^{-2}$ multiplied by the nominal value of the parameters to obtain these values. The absolute mean $\mu_i^*$, implies that parameter 4, government legitimacy, is the most influential parameter while other parameters are minimally influential. Note that the Gaussian process and neural network surrogate models provide consistent results, while the polynomial has similar rankings of parameters, but the value of $\mu_i^*$ for the polynomial surrogate is larger than that of the other two. We also note that the identification of government legitimacy as the only influential parameter is consistent with the Sobol’ 

86
The standard deviation $\sigma_i$ does not agree with this. We note that for various values of $R$ and various step sizes $\Delta$, the quantity $\mu^*_i$ remains relatively unchanged, while the quantity $\sigma_i$ changes drastically. The changes in $\sigma_i$ when varying either $R$ or the step size reorder the parameters that are considered influential. This is even the case for small step sizes, on the order of magnitude $10^{-6}$ to $10^{-8}$. Since the standard deviation $\sigma_i$ lead to ambiguity in the influence of parameters, we rely on $\mu^*_i$ to determine the influence of each parameter.

**Activity Scores**

We computed the active subspaces and activity scores using $M = 10^5$ psuedo-random samples. This value was determined in the manner as done with the Sobol’ indices. We compute the gradient via automatic differentiation on the polynomial and Gaussian process surrogate models. On the neural network surrogate model, we compute the gradient via forward finite difference with a step size of $10^{-6}$ multiplied by the nominal value of each parameter. We choose to use finite difference methods for the gradient of the neural network.
for the reasons outlines in Section 5.1.3. We utilized Algorithm 4 to compute the eigenvalues of the approximated matrix $C$ in Figure 5.29. We note that a 1-dimensional active subspace is appropriate here. A 1-dimensional active subspace is appropriate as there is a large gap - at least 3 orders of magnitude - between the first and second largest eigenvalues of $C$. We verify this 1-dimensional active subspace by creating surrogate models on the active subspace and comparing the predictions of the active subspace surrogate model with that of the full space surrogate model in Figure 5.30. The activity scores in Table 5.25 show that the quantity of interest rate of violence is most sensitive to changes to the parameter government legitimacy. Additionally, the three surrogate models have similar magnitudes in the activity scores. We confirm that the activity score and total Sobol’ index bounds hold. Recall the total

![Figure 5.28](image) (a) $\mu_i$ and (b) $\sigma_i$ from Morris screening performed using the averaged-time surrogate models for the 5-D Rebellion model.

![Figure 5.29](image) Eigenvalues of $C$ for active subspace computation computed using the (a) polynomial, (b) Gaussian process, and (c) neural network averaged-time surrogate models for the 5-D Rebellion model.
Table 5.25 Activity Scores for parameters calculated on the averaged-time surrogate models for the 5-D Rebellion model.

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$ (Initial Officer Density)</td>
<td>1955.5</td>
<td>9752.8</td>
<td>2213.8</td>
</tr>
<tr>
<td>$a_2$ (Initial Agent Density)</td>
<td>2.6557</td>
<td>0.13266</td>
<td>3.1631</td>
</tr>
<tr>
<td>$a_3$ (Vision)</td>
<td>192.43</td>
<td>2129.1</td>
<td>234.16</td>
</tr>
<tr>
<td>$a_4$ (Government Legitimacy)</td>
<td>6.793e+06</td>
<td>2.5031e+07</td>
<td>6.8686e+06</td>
</tr>
<tr>
<td>$a_5$ (Maximum Jail Term)</td>
<td>0.0025536</td>
<td>0.0094849</td>
<td>0.94781</td>
</tr>
</tbody>
</table>

Sobol’ index and activity score bound from (4.6). The computed values of $ST_i$ and $B_i$ are provided in Table 5.26, and the bound holds when $ST_i \leq B_i$. We again see that the bounds only hold for the total Sobol’ index and activity score for the parameter government legitimacy. Computation of the covariance matrix, again indicates that the parameters are not independent. Again, we note that Sobol’ analysis and active subspace analysis assumes independence of parameters but can often be utilized effectively when this assumption is violated. Sobol’ analysis is more sensitive than active subspace analysis to this assumption being falsely made. However, we still observe that the parameter government legitimacy is the most influential according to the Pearson correlation, Sobol’ indices, Morris screening, and activity scores.

**Time-Dependent Activity Scores**

Time-dependent activity scores computed independently at time steps 1-20 are provided in Figure 5.31. Government legitimacy is again the most influential parameter at all time steps. These values were determined using $M = 10^3$ psuedo-random samples for the parameters.

![Figure 5.30](image)

**Figure 5.30** Goodness of fit of 1-D active subspaces computed using the (a) polynomial, (b) Gaussian process, and (c) neural network averaged-time surrogate models for the 5-D Rebellion model.
at each time step. For computational reasons, we take $n = 5$ when computing (4.5) so that we evade the need to verify the active subspace at each time step.

Utilizing the block structure explained in Section 4.4.4, we compute activity scores using the time-dependent surrogates provided in Table 5.27. These values were computed using $M = 10^3$ pseudo-random samples. The time values were taken as random integer values between 1 and 20 and compared to results when taking random float values between 1 and 20. We obtain consistent results with previous methodologies and across the three surrogate models. The parameter government legitimacy is determined to be the most influential parameter.

**Table 5.26** Total Sobol’ indices and activity score bounds for parameters calculated on the averaged-time surrogate models for the 5-D Rebellion model.

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Officer Density</td>
<td>$S_{T_1}$</td>
<td>$B_1$</td>
<td>$S_{T_2}$  $B_1$</td>
</tr>
<tr>
<td></td>
<td>1.1733e-2</td>
<td>8.9225e-4</td>
<td>1.1005e-2  4.4603e-3</td>
</tr>
<tr>
<td>Initial Agent Density</td>
<td>$S_{T_2}$</td>
<td>$B_2$</td>
<td>$S_{T_2}$  $B_2$</td>
</tr>
<tr>
<td></td>
<td>6.2816e-3</td>
<td>1.2117e-6</td>
<td>5.64e-3  6.0669e-8</td>
</tr>
<tr>
<td>Vision</td>
<td>$S_{T_3}$</td>
<td>$B_3$</td>
<td>$S_{T_3}$  $B_3$</td>
</tr>
<tr>
<td></td>
<td>4.1393e-3</td>
<td>8.7803e-5</td>
<td>3.705e-3  9.7372e-4</td>
</tr>
<tr>
<td>Government Legitimacy</td>
<td>$S_{T_4}$</td>
<td>$B_4$</td>
<td>$S_{T_4}$  $B_4$</td>
</tr>
<tr>
<td></td>
<td>0.98649</td>
<td>3.0995</td>
<td>0.98711  11.448</td>
</tr>
<tr>
<td>Maximum Jail Term</td>
<td>$S_{T_5}$</td>
<td>$B_5$</td>
<td>$S_{T_5}$  $B_5$</td>
</tr>
<tr>
<td></td>
<td>1.6311e-3</td>
<td>1.1652e-9</td>
<td>1.5437e-4  4.3378e-9</td>
</tr>
</tbody>
</table>

**Figure 5.31** Activity scores over time utilizing the (a) polynomial, (b) Gaussian process, and (c) neural network surrogates, respectively. Recall that $a_1$ corresponds to initial officer density, $a_2$ corresponds to initial agent density, $a_3$ corresponds to vision, $a_4$ corresponds to government legitimacy, and $a_5$ corresponds to maximum jail term.
Table 5.27 Activity Scores for parameters calculated on the time-dependent surrogate models for the 5-D Rebellion model.

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time as a discrete variable</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a_1$ (Initial Officer Density)</td>
<td>3313.4</td>
<td>4.7057e+06</td>
<td>24569</td>
</tr>
<tr>
<td>$a_2$ (Initial Agent Density)</td>
<td>5.888</td>
<td>1787.9</td>
<td>8.0328</td>
</tr>
<tr>
<td>$a_3$ (Vision)</td>
<td>397.23</td>
<td>1.0395e+06</td>
<td>1013.7</td>
</tr>
<tr>
<td>$a_4$ (Government Legitimacy)</td>
<td>5.7405e+06</td>
<td>7.0304e+08</td>
<td>8.3392e+06</td>
</tr>
<tr>
<td>$a_5$ (Maximum Jail Term)</td>
<td>7.9378</td>
<td>108.51</td>
<td>11.254</td>
</tr>
<tr>
<td><strong>Time as a continuous variable</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a_1$ (Initial Officer Density)</td>
<td>3224.7</td>
<td>4.7225e+06</td>
<td>24679</td>
</tr>
<tr>
<td>$a_2$ (Initial Agent Density)</td>
<td>5.7352</td>
<td>1800.2</td>
<td>8.1594</td>
</tr>
<tr>
<td>$a_3$ (Vision)</td>
<td>388.47</td>
<td>1.0462e+06</td>
<td>1042.6</td>
</tr>
<tr>
<td>$a_4$ (Government Legitimacy)</td>
<td>5.8896e+06</td>
<td>7.0427e+08</td>
<td>8.7456e+06</td>
</tr>
<tr>
<td>$a_5$ (Maximum Jail Term)</td>
<td>7.9216</td>
<td>108.77</td>
<td>11.153</td>
</tr>
</tbody>
</table>

Figure 5.32 Distribution of response when all parameters are randomly sampled from input intervals and when fixing noninfluential parameters (NP), using the (a) polynomial, (b) Gaussian process, and (c) neural network averaged-time surrogate models for the 5-D Rebellion model. All parameters except government legitimacy are noninfluential.

Verification

We confirm our results qualitatively by plotting the distribution of our response variable when all five parameters are randomly sampled and when keeping all parameters except government legitimacy fixed. We utilize $10^4$ psuedo-random samples. We plot these distributions in Figure 5.32.

We verify our results quantitatively by computing test statistics for each surrogate and provide the results in Figure 5.33. These values were obtained using the same samples as were used for Figure 5.32. For the neural network and Gaussian process surrogate, the test statistic is smaller than both critical values and so we do not reject the null hypothesis. This indicates that the two sets of samples are from the same distribution. This does not occur
Figure 5.33 Energy statistic results for the (a) polynomial, (b) Gaussian process, and (c) neural network surrogates. Information includes the pooled set of ordered replicates, test statistics $T_{n_1,n_2}$, and critical values $T_c$ at 95% and 99% confidence levels.

for the polynomial surrogate. We note that the polynomial surrogate provided the worst fit to the agent-based model and so this result is not overly alarming.

Conclusion

We conclude that government legitimacy is the most influential parameter for this model when considering all five parameters. The original parts of the analysis were the use of a neural network surrogate model for an agent-based model, the use of parameter subset selection to determine identifiable parameters, and the use of activity scores for sensitivity analysis. The activity scores computation required significantly less evaluations than the Sobol’ indices computation. Here, Sobol’ indices required $10^6$ model evaluations, while activity scores required computation of the gradient $10^5$ times.

5.3 Greenhouse Model

In this section, we create surrogate models and perform sensitivity analysis for the Greenhouse model described in Section 2.3. As this model has 40 parameters it is ideal to identify a subset of noninfluential parameters that we can fix for the model. We construct data by considering a 10% perturbation about the nominal values, provided in Table 5.28. If the nominal value is an endpoint of the admissible range of values, we only consider a one-sided perturbation. For example, parameter 1, StubbornnessFactor has an admissible parameter range of 0.7 to 0.9, but the nominal value is 0.7. For parameter 1, the sensitivity analysis range will be $[0.7, 0.77]$ instead of $[0.63, 0.77]$.

Training data consists of 1000 psuedo-random parameter choices with 10 repetitions for each choice of parameters. We created the test data in the same manner with 10 test
Table 5.28 Parameters employed in the sensitivity analysis of the Greenhouse model. The parameter index specifies how we will reference each parameter. The parameter range is the full range of admissible values, whereas the SA range is the range for each parameter that we consider while implementing sensitivity analysis.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Index</th>
<th>Default Value</th>
<th>Parameter Range</th>
<th>SA Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>StubbornnessFactor</td>
<td>1</td>
<td>0.7</td>
<td>0.7-0.9</td>
<td>0.7-0.77</td>
</tr>
<tr>
<td>InitialBankAccount</td>
<td>2</td>
<td>70000</td>
<td>0-100,000</td>
<td>63,000-77,000</td>
</tr>
<tr>
<td>CostPriceCoopMult</td>
<td>3</td>
<td>1.3</td>
<td>0-2</td>
<td>1.17-1.43</td>
</tr>
<tr>
<td>InfluenceRangeSec</td>
<td>4</td>
<td>0.04</td>
<td>0-0.05</td>
<td>0.036-0.044</td>
</tr>
<tr>
<td>CostPriceRange</td>
<td>5</td>
<td>400</td>
<td>0-1000</td>
<td>360-440</td>
</tr>
<tr>
<td>InfluenceRangeMain</td>
<td>6</td>
<td>0.02</td>
<td>0-1</td>
<td>0.018-0.022</td>
</tr>
<tr>
<td>TechATempMin</td>
<td>7</td>
<td>0.5</td>
<td>0-1</td>
<td>0.45-0.55</td>
</tr>
<tr>
<td>TechACO2Max</td>
<td>8</td>
<td>0.05</td>
<td>0-0.5</td>
<td>0.045-0.055</td>
</tr>
<tr>
<td>TechAHumMin</td>
<td>9</td>
<td>-0.05</td>
<td>-0.5-0</td>
<td>-0.055-(-0.045)</td>
</tr>
<tr>
<td>TechALightMax</td>
<td>10</td>
<td>0</td>
<td>0-0.1</td>
<td>0-0.01</td>
</tr>
<tr>
<td>TechBTempMax</td>
<td>11</td>
<td>0.2</td>
<td>0-0.2</td>
<td>0.18-0.2</td>
</tr>
<tr>
<td>TechBCO2Min</td>
<td>12</td>
<td>0.5</td>
<td>0-1</td>
<td>0.45-0.55</td>
</tr>
<tr>
<td>TechBHumMin</td>
<td>13</td>
<td>-0.05</td>
<td>-0.3-0</td>
<td>-0.055-(-0.045)</td>
</tr>
<tr>
<td>TechBLightMin</td>
<td>14</td>
<td>0</td>
<td>-0.3-0</td>
<td>-0.03-0</td>
</tr>
<tr>
<td>TechCTempMax</td>
<td>15</td>
<td>0.05</td>
<td>0-0.2</td>
<td>0.045-0.055</td>
</tr>
<tr>
<td>TechCCO2Max</td>
<td>16</td>
<td>0</td>
<td>0-0.1</td>
<td>0-0.01</td>
</tr>
<tr>
<td>TechCHumMin</td>
<td>17</td>
<td>0.5</td>
<td>0-1</td>
<td>0.45-0.55</td>
</tr>
<tr>
<td>TechCLightMin</td>
<td>18</td>
<td>-0.05</td>
<td>-0.3-0</td>
<td>-0.055-(-0.045)</td>
</tr>
<tr>
<td>TechDTempMax</td>
<td>19</td>
<td>0.05</td>
<td>0-0.1</td>
<td>0.045-0.055</td>
</tr>
<tr>
<td>TechDCO2Max</td>
<td>20</td>
<td>0</td>
<td>0-0.2</td>
<td>0-0.02</td>
</tr>
<tr>
<td>TechDHumMin</td>
<td>21</td>
<td>-0.05</td>
<td>-0.2-0</td>
<td>-0.055-(-0.045)</td>
</tr>
<tr>
<td>TechDLightMin</td>
<td>22</td>
<td>0.5</td>
<td>0-1</td>
<td>0.45-0.55</td>
</tr>
<tr>
<td>VeggiesIdealTemp</td>
<td>23</td>
<td>26</td>
<td>0-40</td>
<td>23.4-28.6</td>
</tr>
<tr>
<td>VeggiesIdealCO2</td>
<td>24</td>
<td>1500</td>
<td>1000-2000</td>
<td>1350-1650</td>
</tr>
<tr>
<td>VeggiesIdealHum</td>
<td>25</td>
<td>75</td>
<td>0-100</td>
<td>67.5-82.5</td>
</tr>
<tr>
<td>VeggiesIdealLight</td>
<td>26</td>
<td>1400</td>
<td>1000-2000</td>
<td>1260-1540</td>
</tr>
<tr>
<td>VeggiesPotentialGrowth</td>
<td>27</td>
<td>900</td>
<td>0-1000</td>
<td>810-990</td>
</tr>
<tr>
<td>FlowersIdealTemp</td>
<td>28</td>
<td>24</td>
<td>0-35</td>
<td>21.6-26.4</td>
</tr>
<tr>
<td>FlowersIdealCO2</td>
<td>29</td>
<td>1200</td>
<td>1000-2000</td>
<td>1080-1320</td>
</tr>
<tr>
<td>FlowersIdealHum</td>
<td>30</td>
<td>87</td>
<td>0-100</td>
<td>78.3-95.7</td>
</tr>
<tr>
<td>FlowersIdealLight</td>
<td>31</td>
<td>1700</td>
<td>1000-2000</td>
<td>1530-1870</td>
</tr>
<tr>
<td>FlowersPotentialGrowth</td>
<td>32</td>
<td>900</td>
<td>0-1000</td>
<td>810-990</td>
</tr>
<tr>
<td>ExternalTemp</td>
<td>33</td>
<td>17</td>
<td>0-100</td>
<td>15.3-18.7</td>
</tr>
<tr>
<td>ExternalCO2</td>
<td>34</td>
<td>400</td>
<td>200-2000</td>
<td>360-440</td>
</tr>
<tr>
<td>ExternalHum</td>
<td>35</td>
<td>74</td>
<td>50-100</td>
<td>66.6-81.4</td>
</tr>
<tr>
<td>FuelPrice</td>
<td>36</td>
<td>0.8</td>
<td>0.4-1.0</td>
<td>0.72-0.88</td>
</tr>
<tr>
<td>VeggiesMarketPrice</td>
<td>37</td>
<td>15</td>
<td>0-20</td>
<td>13.5-16.5</td>
</tr>
<tr>
<td>FlowersMarketPrice</td>
<td>38</td>
<td>15</td>
<td>0-20</td>
<td>13.5-16.5</td>
</tr>
<tr>
<td>VeggiesPurchasePrice</td>
<td>39</td>
<td>3</td>
<td>0-20</td>
<td>2.7-3.3</td>
</tr>
<tr>
<td>FlowersPurchasePrice</td>
<td>40</td>
<td>3</td>
<td>0-20</td>
<td>2.7-3.3</td>
</tr>
</tbody>
</table>
data sets of 100 runs. Additionally, we created a second set of training data for the purpose of creating surrogate models on the active subspace. The same training and test data will be utilized for both the averaged-time surrogate models and the time-dependent surrogate models. The quantity of interest is the sum of all actual production.

We recall from (4.1) that we assume normally distributed noise on the observations. We verify this assumption in Figure 5.34, where we plot a normalized histogram for 1000 runs of the Greenhouse model utilizing the nominal parameter values given in Table 5.28.

5.3.1 Surrogate Models

Averaged-Time Surrogate Models

Here we construct Gaussian process and neural network surrogate models. Note that we do not use a polynomial surrogate model here. The Rebellion model analysis of Sections 5.1 and 5.2 indicated that a polynomial surrogate has poor agreement with agent-based model data. Further, the number of polynomial terms with 40 parameters would require solving a large linear system as the degree of the polynomial is increased and is prohibitive to compute. We compare results using each surrogate model to verify consistency. We first construct the surrogate models for the averaged-time response and recall that the quantity of interest is the average of the sum of all actual production over the first 100 time steps.

Gaussian Process Surrogate

We compute the Gaussian process surrogate using the Matern 3/2 kernel with different

Figure 5.34 Normalized histogram for 1000 runs of the Greenhouse model with default parameter values from Table 2.2. We observe that the data fits a normal distribution with mean 118072 and standard deviation 1118.51.
Table 5.29 Normalized RMSE for the averaged-time surrogate models for the Greenhouse model.

<table>
<thead>
<tr>
<th>Surrogate</th>
<th>Training Data</th>
<th>Mean of Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian Process</td>
<td>0.0025</td>
<td>0.0036</td>
</tr>
<tr>
<td>Neural Network</td>
<td>0.0035</td>
<td>0.0044</td>
</tr>
</tbody>
</table>

Table 5.30 Computation time for the averaged-time surrogate models for the Greenhouse model, where training time is the amount of time (in seconds) it takes to train each model and evaluation time is the amount of time (in seconds) it takes to evaluate the training data utilizing that particular surrogate model.

<table>
<thead>
<tr>
<th>Surrogate</th>
<th>Computation Time</th>
<th>Training Time</th>
<th>Evaluation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian Process</td>
<td>216.453052</td>
<td>0.161858</td>
<td></td>
</tr>
<tr>
<td>Neural Network</td>
<td>2.916970</td>
<td>0.121730</td>
<td></td>
</tr>
</tbody>
</table>

correlation scales for each parameter from (3.5). The resulting fit of this is provided in Figure 5.35 and the normalized RMSE is provided in Table 5.29 with full results provided in Table A.7. The kernel parameters for the Gaussian process surrogate are \([69.885, 9.7264e+10, 2.2429e+06, 1.3566e+06, 4.429e+08, 1.9034, 22.831, 2.5764, 2.5848e+08, 20.191, 26.995, 19.119, 20885, 18.916, 18.01, 8.858, 128.44, 1.2329e+06, 16.69, 3.5659, 55056, 34.84, 1922, 1.429e+05, 347.26, 16807, 3232.4, 1285.4, 83305, 2086.8, 53848, 3261.5, 508.62, 17580, 357.59, 12.949, 127.07, 126.69, 92.579, 68.31, 67816]^T\), where the first 40 components correspond to the different length scales for the 40 inputs, \(\sigma_i\), and the last component corresponds to the signal standard deviation, \(\sigma_f\).

**Neural Network Surrogate**

We fit a neural network surrogate with 10 neurons in the hidden layer. We determine the number of neurons in the hidden layer by iterating multiples of 10 neurons in the hidden layer. When two consecutive powers of 10 provided similar performance on the neural network surrogate as determined by the RMSE on the training and test data, the smaller of the two values was taken as the desired number of neurons for the hidden layer. We provide goodness of fit plots in Figure 5.35 and normalized RMSE values in Table 5.29 with full results provided in Table A.7.

Table 5.29 indicates that the Gaussian process surrogate model performs slightly better than the neural network. Both surrogate models do some overfitting to the training data as indicated by the difference in the RMSE values on the training and test data sets. Table 5.30 indicated that the neural network takes significantly less time to train and is generally faster than the Gaussian process for evaluation.
Time Dependent Surrogate Models

We also construct time-dependent Gaussian process and neural network surrogate models. We compare results using each surrogate model to verify consistency. We recall that the quantity of interest is the sum of all actual production over the first 100 time steps. With 1000 runs of 100 time steps, we have $10^5$ data points to possibly use in our training and test data sets. To improve computation time, we take a random sample of 2000 data points from the train and test data sets, and fit and test our model on that subset.

**Gaussian Process Surrogate**

We compute the Gaussian process surrogate model using the Matern 3/2 kernel with different length scales for each input provided by (3.5). We provide the fit of this surrogate

![Figure 5.35](image)

**Figure 5.35** Goodness of fit of the Gaussian process ((a) and (c)) and the neural network ((b) and (d)) surrogate models for the Greenhouse model. (a)-(b) Goodness of fit of the training data and (c)-(d) goodness of fit of the test data.
Table 5.31 Normalized RMSE for the time-dependent surrogate models for the Greenhouse model.

<table>
<thead>
<tr>
<th>Surrogate \ Set</th>
<th>Training Data</th>
<th>Mean of Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian Process</td>
<td>1.8183e-04</td>
<td>0.0048</td>
</tr>
<tr>
<td>Neural Network</td>
<td>0.0047</td>
<td>0.0067</td>
</tr>
</tbody>
</table>

Table 5.32 Computation time for the time-dependent surrogate models for the Greenhouse model, where training time is the amount of time (in seconds) it takes to train each model and evaluation time is the amount of time (in seconds) it takes to evaluate the training data utilizing that particular surrogate model.

<table>
<thead>
<tr>
<th>Surrogate \ Computation Time</th>
<th>Training Time</th>
<th>Evaluation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian Process</td>
<td>1277.668837</td>
<td>0.832271</td>
</tr>
<tr>
<td>Neural Network</td>
<td>2.941153</td>
<td>0.164749</td>
</tr>
</tbody>
</table>

We observe from Table 5.31 that the Gaussian process does significant overfitting. Further, the Gaussian process fits the training and test data the closest, however the Gaussian process does not overly outperform the neural network surrogate. We observe in Table 5.32 that the computation time needed for training and evaluating the Gaussian process is significantly larger than the neural network.

**Neural Network Surrogate**

We fit a neural network surrogate with 10 neurons in the hidden layer. We determined the number of neurons in the hidden layer in the format previously explained for the averaged-time neural network surrogate for the Greenhouse model. We plot the fit of this surrogate model in Figure 5.36 and compute the normalized RMSE values in Table 5.31 with full results provided in Table A.8. The kernel parameters for the Gaussian process surrogate are [0.86548, 4.5064e+09, 16.316, 0.18807, 6.7811e+06, 0.11339, 4.6409, 0.11135, 3284.1, 1.6057, 0.33168, 2.5493, 257.36, 3.6324, 357.86, 0.50448, 10.562, 8221, 2.0579, 1.342, 4502.3, 4.5749, 368.41, 21894, 102.66, 3942.5, 665.34, 276.35, 11704, 345.11, 7107.6, 735.21, 48.715, 2859.1, 93.957, 2.0955, 16.318, 21.214, 7.1981, 10.122, 606.12, 12231]T, where the first 40 components correspond to the different length scales for the 40 inputs, σi, the 41st component corresponds to the length scale for the time variable, σ41, and the last component corresponds to the signal standard deviation, σf.
5.3.2 Sensitivity Analysis

We perform sensitivity analysis on the surrogate models for the Greenhouse model. We begin with local sensitivity analysis via parameter subset selection, followed by global sensitivity analysis via one-at-a-time methods, Pearson’s correlation coefficients, Sobol’ indices, and Morris screening. We then use active subspace methods to compute activity scores and lastly verify the results and present the conclusions of these analyses.

Parameter Subset Selection

We apply parameter subset selection locally as described in Section 4.2 via Algorithm 1. We do so at the nominal values for each parameter, and call this value $\theta^*$. Note that this means

![Figure 5.36](image.png)

**Figure 5.36** Goodness of fit of the time-dependent Gaussian process ((a) and (c)) and neural network ((b) and (d)) surrogate models for the Greenhouse model, where (a)-(b) shows the goodness of fit of the training data and (c)-(d) shows the goodness of fit of the test data.
for each surrogate model and randomly selected parameter values $\theta^*$, we will compute

$$\chi = \begin{bmatrix}
\frac{\partial f}{\partial \theta_1}(1, \theta^*) & \cdots & \frac{\partial f}{\partial \theta_{40}}(1, \theta^*) \\
\vdots & \ddots & \vdots \\
\frac{\partial f}{\partial \theta_1}(100, \theta^*) & \cdots & \frac{\partial f}{\partial \theta_{40}}(100, \theta^*)
\end{bmatrix}.$$  

Utilizing the Gaussian process surrogate model, we determine four identifiable parameters: parameters 8, 27, 32, and 37. The singular values associated with this are

$$\sigma_1 = 231.19 \leq 732.87 \leq 7551.9 \leq 1.7771 \times 10^5 = \sigma_4.$$  

Parameter subset selection on the neural network will provide four identifiable parameters as well: parameters 2, 18, 27, and 32. The singular values associated with this are:

$$\sigma_1 = 1977.5 \leq 2669.6 \leq 7743.7 \leq 1.7653 \times 10^5 = \sigma_4.$$  

We note that both the Gaussian process and the neural network identify parameters 27 and 32 as identifiable.

One-at-a-Time Sensitivity Analysis

We plot the one-at-a-time sensitivity analysis plots in Figures 5.37 and 5.38. We include data taken from the agent-based model, and the Gaussian process and neural network surrogate models. The data taken from the agent-based model was centered around the nominal values provided in Table 5.28. The y-axis scale was kept consistent across all plots. From these plots, we anticipate that parameters 27 and 32 are the most influential to the response. Further, we observe that the Gaussian process and neural network surrogates provide similar, but slightly different fits to the agent-based model.

Pearson Correlation

We compute the absolute value of the Pearson correlation coefficients for the averaged sum of all actual production to the parameters for the greenhouse model in Figure 5.39. The highest correlation comes from parameters 27 and 32 indicating that parameters 27 and 32 are the most influential to the quantity of interest.
Figure 5.37 OAT plot for Gaussian process and neural network surrogate models for the Greenhouse model for parameters 1-20.

Sobol' Sensitivity Indices

We compute Sobol' indices following Algorithm 2 from Section 4.3.3 and report the first order Sobol' indices in Figure 5.40 and the total Sobol' indices in Figure 5.41. These values were computed using a pseudo-random sample of \( M = 5 \times 10^6 \) points. The value of \( M \) was iterated until it became computationally prohibitive to iterate larger. There is not a large difference in the Sobol' indices computed with \( M = 5 \times 10^6 \) and \( M = 10^6 \). With \( M \leq 5 \times 10^6 \), some first order Sobol' indices are computed to have small negative values. The largest of these in absolute value is on the order of \( 10^{-4} \), and thus can be taken to be 0. The Sobol' indices imply that our model is most sensitive to changes in parameters 27 and 32,
Figure 5.38 OAT plot for Gaussian process and neural network surrogate models for the Greenhouse model for parameters 21-40.

VeggiesPotentialGrowth and FlowersPotentialGrowth. All other parameters are determined to be noninfluential. This result is consistent for both surrogate models.

Generalized Sobol’ Indices

We plot the generalized first-order and total Sobol’ indices, computed using the details in Section 4.3.3, in Figure 5.42. These values were computed using $10^6$ pseudo-random samples for the neural network. Parameters 27 and 32 are the most influential parameters for all times $t = 1$ to $t = 100$. Note that due to computational constraints, we only consider the neural network surrogate here. Recall from Table 5.32 that the Gaussian process takes 5 times
Figure 5.39 Pearson correlation coefficients for quantity of interest (averaged sum of all actual production) to parameters for the greenhouse model.

Figure 5.40 First-order Sobol' indices for parameters calculated using the (a) Gaussian process and (b) neural network averaged-time surrogate models for the Greenhouse model.

Figure 5.41 Total Sobol' indices for parameters calculated using the (a) Gaussian process and (b) neural network averaged-time surrogate models for the Greenhouse model.

longer than the neural network to evaluate, with minimal improvement on performance.
Figure 5.42 Time-dependent (a) first-order Sobol' indices and (b) total Sobol' indices utilizing the neural network surrogate for the Greenhouse model.

Figure 5.43 Absolute mean $\mu^*_i$ from Morris screening performed using the (a) Gaussian process and (b) neural network surrogate models for the Greenhouse model.

Morris Screening

We plot the Morris screening results computed using Algorithm 3 from Section 4.3.4 with normalized inputs in Figures 5.43 and 5.44. We use $R = 100$ samples and a step size of $\Delta = 10^{-4}$ to obtain these values. The computed quantities $\mu^*_i$ and $\sigma_i$ indicate that parameters 27 and 32 are the most influential. This is consistent with previous results from Pearson correlation coefficients and Sobol' indices. For various values of $R$ and various step sizes, the quantities remain relatively unchanged.

Activity Scores

We compute the active subspaces and activity scores using $M = 10^5$ psuedo-random samples. This value was determined in the same format as described for the Sobol' analysis. The normalized gradient was computed with forward finite difference methods for the neural
Figure 5.44 Standard deviation $\sigma_i$ from Morris screening performed using the (a) Gaussian process and (b) neural network surrogate models for the Greenhouse model.

network surrogate and automatic differentiation for the Gaussian process surrogate. With the finite difference methods a step size of $10^{-6}$ was utilized - utilizing a different step size would not change the results. We use finite difference methods for gradient approximation for the neural network to circumvent the need to decompose the neural network structure within MATLAB.

The results for the Gaussian process and the neural network in this section are different. We note from Figures 5.37 and 5.38 that the two surrogate models, while having a similar training and test RMSE values, provide a very different fit to the agent-based model. In this section, we will see some of these differences highlighted.

The eigenvalues of the approximated matrix $\mathbf{C}$, computed using Algorithm 4 from Section 4.4, are provided in Figure 5.45. We note that a 1-dimensional active subspace is appropriate for the Gaussian process surrogate and a 10-dimensional active subspace is appropriate for the neural network surrogate.

We verify these choices by creating surrogate models on the active subspaces and comparing the predictions of the active subspace surrogate model with that of the full space surrogate model. We conclude from Figure 5.46 that the chosen active subspaces are appropriate for this problem.

We plot activity scores in Figure 5.47, computed using equation (4.5). We observe that the most influential parameters are parameters 27 and 32 for the Gaussian process surrogate and parameters 2, 27, and 32 for the neural network surrogate. Since the neural network result includes parameter 2, we observe a slight deviation from the Pearson, Sobol’, and Morris results.

We verify that the activity score and total Sobol’ index bounds hold using the relation (4.6) from Section 4.4.3. The computed values of $S_{Ti}$ and $B_{Ti}$ are provided in Figure 5.48.
We see that the bounds hold for some but not all parameters. This occurs because the parameters are not independent which is a violation of the assumptions for Sobol’ and active subspace analysis. Recall that Sobol’ indices are more sensitive to deviations from this assumption.

**Time-Dependent Activity Scores**

We computed time-dependent activity scores independently at time steps 1-100 yielding the values plotted in Figure 5.49. This was computed using $10^5$ psuedo-random samples for the parameters. We observe that parameters 27 and 32 are determined as the most influential. We utilized the full space when computing activity scores instead of a smaller active subspace. This was done so that there was not a need to verify the dimension of the active subspace at each time step.

The activity scores computed via the time-dependent neural network surrogate utilizing the block structure explained in Section 4.4.4 are plotted in Figure 5.50. These values were computed using $10^5$ psuedo-random samples. The time values were taken as random integer values between 1 and 100 and compared to results when taking random float values between 1 and 100. We obtain consistent results as we had with the activity scores for the non-time dependent neural network surrogate. Again we only use the neural network surrogate model here due to computational time.

**Verification**

We verify our results qualitatively in Figure 5.51 by plotting the distribution of the response variable, the sum of all actual production, when all parameters are randomly sampled.
within 10% of their nominal values and when noninfluential parameters are fixed. We utilize $10^4$ psuedo-random samples. We observe that the original distribution of responses in Figure 5.51 appears consistent with the distribution of responses when noninfluential parameters are fixed.

We verify our results quantitatively by computing test statistics for both surrogate models and provide the results in Figure 5.52. These values were obtained using the same samples as were used for Figure 5.51. In Figure 5.52, we note that when we consider parameters 27 and 32 as the only influential parameters, the test statistic is larger than the critical value for the 95% confidence interval, but is contained in the 99% confidence interval for the neural network. Thus we do not reject the null hypothesis and conclude that these distributions are identical. We note that since these distributions are the same when only considering
parameters 27 and 32 as influential, they will also be identical for the subset of identifiable parameters identified in parameters subset selection, which included two more parameters. Similarly, the addition of parameter 2 to the influential set as indicated by the neural network activity scores will also not change results. For the Gaussian process surrogate model, we note that Figure 5.51 indicates that the two samples come from the same distribution but that Figure 5.52 indicates the opposite. Here, we still conclude that parameters 27 and 32 are the most influential as all sensitivity analysis methods and Figure 5.51 indicate this.

![Figure 5.48](image1.png)

**Figure 5.48** Total Sobol' indices and activity score bounds for parameters calculated for the (a) Gaussian process and (b) neural network surrogate models for the Greenhouse model.

![Figure 5.49](image2.png)

**Figure 5.49** Activity scores over time utilizing the neural network surrogate for the Greenhouse model.
Figure 5.50 Activity scores for parameters calculated on the neural network surrogate model for the Greenhouse model where (a) time is treated as a continuous variable and (b) time is treated as a discrete variable.

Figure 5.51 Distribution of response when all parameters are randomly sampled within 10% of their nominal value and when keeping all noninfluential parameters fixed, using the (a) Gaussian process and (b) neural network surrogate models for the Greenhouse model. Distributions labeled as NI fixed have parameters 27 and 32 as the influential parameters and all other parameters set at their default values.

Conclusion

We conclude that parameters 27 and 32 are the most influential for the Greenhouse model. We did not include the time-dependent Gaussian process surrogate model in the time-dependent activity scores and generalized Sobol’ analysis due to computation time. It takes the Gaussian process 0.80 seconds to compute 2000 pseudo-random samples while it only takes the neural network 0.17 second to compute the same samples.

We note that there are small differences in how the agent-based model data was fit by
the Gaussian process and neural network surrogate models in Figures 5.37 and 5.38. These differences become most apparent in the derivative-based sensitivity analysis methods, where we obtain additional parameters being identifiable or influential. The one-at-a-time plots would not indicate such an occurrence, and thus, these differences are coming from parameter interactions.

5.4 Conclusions for Both Models

The original contributions in this chapter were the use of neural networks as surrogate models for agent-based models, the use of parameter subset selection to determine local identifiability, and the use of activity scores to quantify the sensitivity of model outputs to fluctuations in model inputs. Previous sensitivity analysis methods applied to agent-based models includes Sobol’ indices and Morris screening as discussed in [6, 15, 28].

The accuracy of the neural network surrogates was similar to that of the Gaussian process surrogates as demonstrated via the normalized root mean squared error. Further, the neural network surrogates ran five times faster than the Gaussian process surrogate model on the Greenhouse model allowing for more efficient computation in the sensitivity analysis. This permitted larger random samples to be taken when computing Sobol’ indices and activity scores.

For both models, activity scores provided similar results to Sobol’ sensitivity analysis. Further, activity scores required smaller sample sizes to obtain consistent results. This means that activity scores require less model evaluations and are faster to run than Sobol’
sensitivity analysis. For the neural network surrogate models of the Greenhouse model, the neural network did provide a different subset of influential parameters. This was likely due to interaction terms, and would improve with more data. Generally, Gaussian process surrogate models are better for small data sets and neural networks perform better for large data sets [45].
In this Chapter, we utilize Bayesian inference to infer distributions for model parameters for the Rebellion model. Here, uncertainty in parameters is addressed by treating the parameters as random variables. In Section 6.1, we detail the method in which we obtain a posterior density for the parameters that best reflects the model and observations. In Section 6.2, we apply Bayesian inference to the surrogate models of the Rebellion model with $p = 2$ parameters and in Section 6.3, we apply Bayesian inference to the surrogate models of the Rebellion model with $p = 5$ parameters to obtain posterior distributions of the model parameters.

6.1 Background

Consider the observation model

$$Y_i = f_i(\theta) + \epsilon_i, \ i = 1, \ldots, n$$ \hspace{1cm} (6.1)
where \( Y_i \) are random variables representing measurements, \( f_i(\theta) \) is the model response at parameter value \( \theta = [\theta_1, \ldots, \theta_p]^T \), and \( \epsilon_i \sim \mathcal{N}(0, \sigma^2) \) are observation errors. Here, we assume that we have scalar-valued responses. We then construct the likelihood

\[
f(y|\theta) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\sum_{i=1}^{n} \frac{[Y_i - f_i(\theta)]^2}{2\sigma^2}\right),
\]

where \( y = [y_1, \ldots, y_n] \) is fixed and \( \theta \) varies over the admissible parameter space.

When prior information is not known, one should utilize a uniform prior to reflect the lack of prior knowledge. Gaussian priors are also a common choice for the reasons outlined in Section 6.1.5. Given a prior distribution \( \pi_0(\theta) \) calculate the posterior distribution

\[
\pi(\theta|y) = \frac{f(y|\theta)\pi_0(\theta)}{\int_{\mathbb{R}^p} f(y|\theta)\pi_0(\theta) d\theta}.
\]

For moderate or large \( p \), the calculation of the denominator is prohibitive using quadrature techniques. Thus, to compute the posterior distribution, one utilizes Bayesian Markov Chain Monte Carlo (MCMC) techniques [42, 47].

The strategy for MCMC techniques is as follows. Begin with current chain realization \( \theta^{k-1} \in \mathbb{R}^p \). Propose a new value \( \theta^* \) taken from a proposal distribution \( J(\theta^*|\theta^{k-1}) \), which specifies \( \theta^* \) based on \( \theta^{k-1} \). The choice for the proposal distribution \( J \) will be discussed for each algorithm. Compute a probability \( \alpha(\theta^{k-1}, \theta^*) \) based on the likelihood and specified prior distribution. Accept \( \theta^* \) with probability \( \alpha \) to obtain

\[
\theta^k = \begin{cases} 
\theta^* & \text{if accept} \\
\theta^{k-1} & \text{else}
\end{cases}.
\]

The posterior density is the stationary distribution for the chain. We begin by discussing the Metropolis algorithm.

### 6.1.1 Metropolis Algorithm

For the Metropolis algorithm, the proposal distribution is taken to be of the form \( \mathcal{N}(\theta^{k-1}, V) \) or \( \mathcal{N}(\theta^{k-1}, D) \), where \( V \) is the covariance matrix and \( D \) is a diagonal matrix which reflects the scale of each parameter. Take \( \alpha = \min\{1, r\} \), where

\[
r = \frac{\pi(\theta^*|y)}{\pi(\theta^{k-1}|y)} = \frac{f(y|\theta^*)\pi_0(\theta^*)}{f(y|\theta^{k-1})\pi_0(\theta^{k-1})}.
\]
Note that by taking the ratio \( r \), the need to compute the normalization constant is eliminated. We now extend the Metropolis algorithm to obtain the random walk Metropolis algorithm.

### 6.1.2 Random Walk Metropolis Algorithm

Specify the prior as as discussed previously. Two common choices of priors are uniform and Gaussian distributions. If no prior knowledge is known, we typically use a uniform or flat prior. Set the number of chain elements \( M \) and design parameters \( n_s \) and \( \sigma_s \). Determine the initial estimate for \( \theta \), \( \theta^0 \) by least squares and compute \( SS_{\theta^0} = \sum_{i=1}^{n} [y_i - f_i(\theta^0)]^2 \). This allows computation of the initial error variance estimate \( s_0^2 = \frac{SS_{\theta^0}}{n-p} \), where \( n \) is the number of observations and \( p \) is the number of parameters. Construct a covariance estimate \( V = s_0^2 \left[ \chi^T(\theta^0) \chi(\theta^0) \right]^{-1} \), where \( \chi_{ij}(\theta^0) = \frac{\partial f_i}{\partial \theta_j}(\theta^0) \). Then for \( k = 1, ..., M \), construct a candidate \( \theta^* \sim N(\theta^{k-1}, V) \) and compute \( SS_{\theta^*} = \sum_{i=1}^{n} [y_i - f_i(\theta^*)]^2 \). Accept \( \theta^* \) as \( \theta^k \) with probability \( \alpha = \min \{ 1, r \} \), where

\[
\rho = \exp(-\frac{SS_{\theta^*} - SS_{\theta^{k-1}}}{2s_{k-1}^2})(\frac{\pi_0(\theta^*)}{\pi_0(\theta^{k-1})}).
\]

Lastly, update \( s_k^2 \sim \text{Inv-gamma}(a_{val}, b_{val}) \) where \( a_{val} = 0.5(n_s + n) \) and \( b_{val} = 0.5(n_s\sigma_s^2 + SS_{\theta^*}) \).

The random walk Metropolis algorithm uses the same covariance matrix \( V \) for the whole chain and does not incorporate information learned about the posterior distribution as candidate parameters are accepted.

### 6.1.3 Adaptive Metropolis Algorithm

The adaptive Metropolis algorithm as discussed in [17, 48] incorporates information learned about the posterior distribution as candidate parameters are accepted by updating the covariance matrix after a nonadaptive period of length \( k_0 \) computed using the initial covariance matrix \( V = V_0 \).

Once chain values \( \theta^0, \theta^1, ..., \theta^{k-1} \) are computed, update the chain covariance matrix to be

\[
V_k = s_p \text{cov}(\theta^0, \theta^1, ..., \theta^{k-1}) + \epsilon I_p,
\]

where \( s_p \) is a design parameter dependent on the dimension of \( p \). The term \( \epsilon I_p, \epsilon \geq 0 \) ensures that \( V_k \) is positive definite. In theory, one can compute \( \text{cov}(\theta^0, \theta^1, ..., \theta^{k-1}) \) as

\[
\text{cov}(\theta^0, \theta^1, ..., \theta^{k-1}) = \frac{1}{k-1} \left[ \sum_{i=0}^{k-1} \theta^i(\theta^i)^T - k\bar{\theta}^k(\bar{\theta}^k)^T \right],
\]

113
where \( \tilde{\theta}^k = \frac{1}{k} \sum_{i=0}^{k-1} \theta^i \). However, in practice the recursive relation

\[
V_{k+1} = \frac{k-1}{k} V_k + \frac{S_k}{k} \left[ k \tilde{\theta}^{k-1}(\tilde{\theta}^{k-1})^T - (k + 1) \tilde{\theta}^k(\tilde{\theta}^k)^T + \theta^k(\theta^k)^T + \epsilon I_p \right]
\]

is used as it is more efficient. Similarly, the sample mean can be computed as

\[
\bar{\theta}^{k+1} = \frac{k}{k + 1} \bar{\theta}^k + \frac{1}{k + 1} \theta^k.
\]

### 6.1.4 Delayed Rejection

Another way to improve the Metropolis algorithm is delayed rejection [17, 48]. In the standard Metropolis algorithm, if chain candidate \( \theta^* \) is rejected, then the previous chain value \( \theta^{k-1} \) is retained. Using delayed rejection, one constructs alternative candidates \( \theta^{*j} \) if \( \theta^* \) is rejected. A second-stage candidate is chosen from

\[
J_2(\theta^{*2}|\theta^{k-1}, \theta^*) = \mathcal{N}(\theta^{k-1}, \gamma_2^2 V_k),
\]

where \( V_k \) is the covariance matrix from the adaptive algorithm and \( \gamma_2 < 1 \), increasing mixing by making the proposal function narrower than the original. Accept this second-stage candidate with probability

\[
\alpha_2(\theta^{k-1}, \theta^*, \theta^{*2}) = \min\{1, r_2\},
\]

where

\[
r_2 = \frac{\pi(\theta^{*2}|y) J(\theta^*|\theta^{*2}) J_2(\theta^{k-1}|\theta^{*2}, \theta^*) [1 - \alpha(\theta^*|\theta^{*2})]}{\pi(\theta^{k-1}|y) J(\theta^*|\theta^{k-1}) J_2(\theta^{*2}|\theta^{k-1}, \theta^*) [1 - \alpha(\theta^*|\theta^{k-1})]}.
\]

One can continue this process, producing third and higher-stage candidates, but software often defaults to creating only a second stage candidate. Convergence of this process is discussed in [17, 48].

We note that the delayed rejection alters the proposal function to improve mixing and the adaptive Metropolis algorithm provides feedback and reflects information learned about the posterior from previous chain iterates [42, 47]. These two techniques are complimentary and can employed jointly. For implementation of the DRAM algorithm, we utilize the MCMC Toolbox for MATLAB provided in [22] which is based on [17, 18]. The algorithm implemented by this toolbox is provided in algorithms 8.8 and 8.10 from [48].
Large parameter sets with nonidentifiable parameters can cause poor mixing and performance for the DRAM algorithm. To address this, one can apply Bayesian inference techniques on lower dimensional surrogate models created on active subspaces.

First compute the active subspace in the manner outlined in Section 3.4. One then specifies the full and active subspace prior distributions \( \pi_0(\theta) \) and \( \pi_0(\xi) \). A common choice is to employ Gaussian distributions for these priors as the projection of a Gaussian distribution onto an active subspace is still a Gaussian distribution [9]. Based on the model, construct the likelihood \( f(y|\xi) \) and use MCMC to compute \( M \) chain samples \( \xi^k \) for the active variables. Lastly, transform the active variables back to the physical space by sampling the inactive variables \( \zeta^k \) and computing

\[
\theta^k = \mathbf{W}_1 \xi^k + \mathbf{W}_2 \zeta^k.
\]

For full implementation details, including how to sample \( \zeta^k \), see Algorithm 5 taken from [47]. See [26] for additional details and [12] for an alternative implementation of MCMC on active subspaces.

Bayesian inference has been applied to agent-based models on the full space previously as seen in [23, 29]. We propose utilizing Bayesian inference on active subspaces for agent-based models to improve performance.

6.2 Rebellion Model with \( p = 2 \) Parameters

We implement the DRAM algorithm on the full space and active subspaces of the averaged-time 2-D Rebellion model surrogates. Recall the input parameters for these surrogate models were initial officer density and government legitimacy and the quantity of interest is the averaged rate of violence as explained in Section 5.1. We created polynomial, Gaussian process, and neural network surrogate models for this response in Section 5.1.2.

We create simulated data for each surrogate model by

\[
y_i = f(\theta^*) + \epsilon_i, \quad i = 1, \ldots, 100,
\]

where \( \theta^* = [4, 0.8]^T \) is the default parameter values and \( \epsilon_i \sim \mathcal{N}(0, 25^2) \). The noise estimate \( \epsilon_i \) is chosen based on Figure 5.7. The prior distribution for the parameters is taken to be \( \pi_0(\theta) \sim \mathcal{N}(\mu, \Sigma) \), where

\[
\mu = \begin{bmatrix} 4 \\ 0.8 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 1.3^2 & 0 \\ 0 & 0.067^2 \end{bmatrix}
\]

for the full space. The prior distribution for the active subspace is taken to be \( \pi_0(\xi) \sim \)
Algorithm 5 Bayesian Inference on Active Subspaces from [26, 47]

1. Consider the physical observation model (6.1).

2. Randomly sample \( N \) training values \( \theta^j \) from the admissible parameter space. Compute \( \widehat{W}_1 \) and \( \widehat{W}_2 \) via Algorithm 4 and project the training onto the active subspace to create \( \xi^j = \widehat{W}_1 \theta^j \). Fit a surrogate model \( g_i(\xi) \) to the training data \( \{\xi^j, f_i(\theta^j)\} \), \( j = 1, \ldots, N \). This can be done with polynomials, Gaussian processes, or neural networks. One should repeat this transformation process with test data to verify the surrogate model on the active subspace.

3. Specify the Gaussian full and active subspace prior distributions \( \pi_0(\theta) \) and \( \pi_0(\xi) \). For a Gaussian prior \( \pi_0(\theta) \sim \mathcal{N}(\mu, \Sigma) \), the active subspace prior will be \( \pi_0(\xi) \sim \mathcal{N}(\widehat{W}_1^T \mu, \widehat{W}_1^T \Sigma \widehat{W}_1) \).

4. Construct the likelihood

\[
f(y|\xi) = \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-SS_\xi/2\sigma^2}, \quad SS_\xi = \sum_{i=1}^n [y_i - g_i(\xi)]^2,
\]

based on (6.1).

5. Use the DRAM algorithm 12.14 and 12.16 from [47] to compute \( M \) chain samples \( \xi^k \) for the active variables.

6. Transform the active variables back to the physical space. Do so by sampling the inactive variables on the inactive subspace prior \( \pi_0(\xi) \sim \mathcal{N}(\widehat{W}_2^T \mu, \widehat{W}_2^T \Sigma \widehat{W}_2) \) and computing

\[
\theta^k = \widehat{W}_1 \xi^k + \widehat{W}_2 \xi^k.
\]
Table 6.1 Rejection rates from DRAM algorithm implemented with the averaged-time 2-D Rebellion model surrogates on the full and active subspaces.

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Space</td>
<td>0.8029</td>
<td>0.843</td>
<td>0.6618</td>
</tr>
<tr>
<td>Active Subspace</td>
<td>0.1101</td>
<td>0.1063</td>
<td>0.1049</td>
</tr>
</tbody>
</table>

$\mathcal{N}(\mu_a, \sigma_a^2)$, where $\mu_a = W_1^T \mu$ and $\sigma_a^2 = W_1^T \Sigma W_1$. Here, $W_1$ is the active variable transformation as described in Section 4.4. Recall that we utilize a 1-D active subspace. The number of simulations used is $10^5$.

The resulting rejection rates are provided in Table 6.1. We obtain a low rejection rate on the active subspace for all three surrogate models, but have a high rejection rate on the full space for the three surrogate models. This is consistent with our expectations from the goodness of fit of the surrogate models to the active subspace outline in Section 5.1.2. The corresponding chains are plotted in Figure 6.1.

The pairwise plots from the implementation of DRAM on the full space for each surrogate model are provided in Figure 6.2. We note that the pairwise plots are single-valued implying that the two parameters are algebraically related. This explains the high rejection rates for the DRAM implementation on the full space.

We plot the prior and posterior distributions for the parameters government legitimacy and initial officer density in Figure 6.3. The chains with a high rejection rate do not provide a trustworthy distribution as they do not provide proper mixing, but the results from the active subspace are reasonable. Further, for the Gaussian process and neural network surrogate models, there is agreement between the DRAM implementation on the full space and on the active subspace. This provides verification to the active subspace results. Lastly, we note that only the parameter distribution for government legitimacy is informed. This is expected from Section 5.1 which determined that government legitimacy was the only identifiable and influential parameter.

This implementation highlights the value of using DRAM on active subspaces as opposed to the full space. Specifically, inference on the active subspace provides proper mixing and reliable results, while inference on the full space does not provide proper mixing.

6.3 Rebellion Model with $p = 5$ Parameters

We implement the DRAM algorithm on the full space and active subspaces of the averaged-time 5-D Rebellion model surrogates. Recall the five parameters of this model are detailed in Table 2.2 and the quantity of interest is the averaged rate of violence. We create simulated
Figure 6.1 Chains from the DRAM algorithm implements on the surrogate models for the averaged-time 2-D Rebellion model. (a)-(f) are implementations on the full space and (g)-(l) are implementations on the active subspace.

data for each surrogate model by

\[ y_i = f(\theta^*) + \epsilon_i, \; i = 1, \ldots, 100 \]

where \( \theta^* = [4, 70, 7, 0.8, 30]^T \) provides the default parameter values and \( \epsilon_i \sim \mathcal{N}(0, 25^2) \). The noise estimate \( \epsilon_i \) is chosen based on Figure 5.7. The prior distribution for the parameters is
Figure 6.2 Pairwise plots of government legitimacy and initial officer density from the chains from the DRAM algorithm implemented on the surrogate models for the averaged-time 2-D Rebellion model. (a), (b), and (c) use the polynomial, Gaussian process, and neural network surrogates respectively.

Figure 6.3 Prior and posterior distributions for the parameters government legitimacy((a)-(c)) and initial officer density ((d)-(f)).

taken to be $\pi_0(\theta) \sim \mathcal{N}(\mu, \Sigma)$, where

$$
\mu = \begin{bmatrix} 4 \\ 70 \\ 7 \\ 0.8 \\ 30 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 1.3^2 & 10^2 \\ 10^2 & 1 \\ 1 & 0.067^2 \\ 0.067^2 & 6.7^2 \end{bmatrix}
$$
Table 6.2 Rejection rates from DRAM algorithm implemented with the averaged-time 5-D Rebellion model surrogates on the full and active subspaces.

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Space</td>
<td>0.9284</td>
<td>0.8762</td>
<td>0.8504</td>
</tr>
<tr>
<td>Active Subspace</td>
<td>0.1043</td>
<td>0.10398</td>
<td>0.1075</td>
</tr>
</tbody>
</table>

for the full space. The prior distribution for the active subspace is taken to be $\pi_0(\xi) \sim \mathcal{N}(\mu_a, \sigma_a^2)$, where $\mu_a = W_1^T \mu$ and $\sigma_a^2 = W_1^T \Sigma W_1$. Here, $W_1$ is the active variable transformation as described in Section 4.4. Recall that we utilize a 1-D active subspace here. The number of simulations used is $10^5$.

The resulting rejection rates are provided in Table 6.2. There is a high rejection rate on the full space and a low rejection rate on the active subspace for all surrogate models. This implies that the chains for the full space do not provide proper mixing, as seen in Figure 6.4. The neural network performs the best on the full space but still has a rejection rate of 85.04%. We also observe from Figure 6.5 that the chains on the active subspace do provide proper mixing and this is supported by the rejection rates in Table 6.2.

The pairwise plots from the implementation of DRAM on the full space for each surrogate model are plotted in Figures 6.6, 6.7, and 6.8. The only parameters that have a strong linear or nonlinear correlation are government legitimacy and initial officer density. This is true for all three surrogate models and is expected from the results in Section 6.2.

We plot the prior and posterior distributions for the parameters in Figure 6.9. The full space chain from the polynomial surrogate does not provide proper mixing, but the results from the active subspace are reasonable. For the Gaussian process and neural network surrogates, there is agreement between the full space and active subspace results and between all surrogate models. This consistency of results verifies the choice to use Bayesian inference on the active subspace.

For all three surrogate models, only the parameter government legitimacy is informed via DRAM. This is expected as government legitimacy was determined to be the only influential parameter in Section 5.2 demonstrated by all the considered methods.

6.4 Greenhouse Model

We now use Bayesian inference on the Greenhouse model to inform the distributions of influential parameters 27 and 32 - VeggiesPotentialGrowth and FlowersPotentialGrowth. The sensitivity analysis of Section 5.3 indicated that the Greenhouse model was only sensitive to changes in parameters 27 and 32. Thus, we create new surrogate models which
only take VeggiesPotentialGrowth and FlowersPotentialGrowth as inputs and produce the averaged-time sum of all actual production response. The fit of these models is provided in Figure 6.10 and the normalized RMSE is provided in Figure 6.3. We note that we get a similar fit and RMSE values for the two models.

We provide the plots of the chains in Figure 6.11 and the rejection rates in Table 6.4. We observe improper mixing when performing DRAM on the full space, but proper mixing when performing DRAM on the 1-D active subspace for both surrogate models.

We provide the pairwise plots for the DRAM implementation on the full space in Figure 6.12. We observe that the pairwise plot is single-valued, explaining the improper mixing of the chains on the full space. Further, this is an ideal situation to utilize DRAM on active subspaces.

We plot the prior and posterior distributions in Figure 6.13. We observe that the posteriors from Bayesian inference on the full space and active subspace are consistent, although the posterior from Bayesian inference on the full space is not smooth. We note that the resulting posteriors are consistent between both models and that both parameters VeggiesPotentialGrowth and FlowersPotentialGrowth are informed via Bayesian inference. This is expected from the sensitivity analysis performed in Section 2.3.

### 6.5 Conclusion

This chapter highlighted the value of performing Bayesian inference on active subspaces for agent-based models. We observed high rejection rates on Bayesian inference performed on the full space for the Rebellion model with $p = 2$ and $p = 5$ parameters. This was because not all parameters were identifiable. In particular, we only observed that government legitimacy was identifiable for the Rebellion model. Further, we observed high rejection rates on Bayesian inference performed on the full space for the Greenhouse model with $p = 2$ parameters as the parameters were single-valued.

We observed reasonable rejection rates for Bayesian inference on the active subspace for both models, and the corresponding posterior distributions were appropriate. Further, we note that the only parameter informed via Bayesian inference for the Rebellion model is

<table>
<thead>
<tr>
<th>Surrogate \ Set</th>
<th>Training Data</th>
<th>Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian Process</td>
<td>0.0028</td>
<td>0.0034</td>
</tr>
<tr>
<td>Neural Network</td>
<td>0.0029</td>
<td>0.0035</td>
</tr>
</tbody>
</table>

Table 6.3 Normalized RMSE for the 2-D averaged-time surrogate models for the Greenhouse model.
Table 6.4 Rejection rates from the DRAM algorithm implemented with the averaged-time 2-D Greenhouse model surrogates on the full and active subspaces.

<table>
<thead>
<tr>
<th>Surrogate Model</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Space</td>
<td>0.9468</td>
<td>0.9630</td>
</tr>
<tr>
<td>Active Subspace</td>
<td>0.2108</td>
<td>0.2612</td>
</tr>
</tbody>
</table>

The parameter that was indicated as influential in Sections 5.1 and 5.2.

Future work in this area includes application of Bayesian inference to the Greenhouse model with $p = 40$ parameters and exploring other prior distribution options. We utilized a wide Gaussian prior as the projection of the Gaussian prior onto an active subspace would remain Gaussian. An exploration into how other prior distributions perform on active subspaces and how to transform them would be valuable.
Figure 6.4 Chains from the DRAM algorithm implemented on the full space of the surrogate models for the averaged-time 5-D Rebellion model.
Figure 6.5 Chains from the DRAM algorithm implemented on the active subspaces of the surrogate models for the averaged-time 5-D Rebellion model.

Figure 6.6 Pairwise plots of parameters from the chain from the DRAM algorithm implemented on the polynomial surrogate model for the averaged-time 5-D Rebellion model.
Figure 6.7 Pairwise plots of parameters from the chain from the DRAM algorithm implemented on the Gaussian process surrogate model for the averaged-time 5-D Rebellion model.
Figure 6.8 Pairwise plots of parameters from the chain from the DRAM algorithm implemented on the neural network surrogate model for the averaged-time 5-D Rebellion model.
Figure 6.9 Prior and posterior distributions for the parameters initial officer density ((a)-(c)), initial agent density ((d)-(f)), vision ((g)-(i)), government legitimacy ((j)-(l)), and maximum jail term ((m)-(o)).
Figure 6.10 Goodness of fit of the Gaussian process ((a) and (c)) and the neural network ((b) and (d)) 2-D averaged-time surrogate models for the Greenhouse model. (a)-(b) Goodness of fit of the training data and (c)-(d) goodness of fit of the test data.
Figure 6.11 Chains from the DRAM algorithm implemented on the surrogate models for the averaged-time 2-D Greenhouse model. (a)-(d) are implementations on the full space and (e)-(f) are implementations on the 1-D active subspace.

Figure 6.12 Pairwise plots of VeggiesPotentialGrowth and FlowersPotentialGrowth from the chains from the DRAM algorithm implemented on the surrogate models for the averaged-time 2-D Greenhouse model.
Figure 6.13 Prior and posterior distributions for the parameters VeggiesPotentialGrowth ((a)-(b)) and FlowersPotentialGrowth ((c)-(d)).
7.1 Conclusions

The original contributions of this dissertation are the application of parameter subset selection and active subspace analysis to surrogate models for agent-based models, and Bayesian inference on active subspaces for surrogate models of agent-based models. Further, we provided a comparison of Gaussian processes and neural networks as surrogate models for the chosen agent-based models. Recall that active subspace analysis identifies subspaces of influential parameters whereas sensitivity analysis identifies subsets of influential parameters. Further, one can obtain a sensitivity measure - activity scores - from these active subspaces.

7.1.1 Comparison of Surrogate Models for Chosen Agent-Based Models

We utilized neural networks and Gaussian processes as surrogate models for the Rebellion and Greenhouse agent-based models described in Sections 2.2 and 2.3. From the analysis of Chapter 5, it became clear that neural networks provided a similar goodness of fit as Gaussian processes but did not suffer from the "curse of dimensionality." That is, as the number of parameters or the size of the training data set increased, the Gaussian process surrogate models became computationally costly due to the need for inversion of large covariance
matrices, whereas the neural network only incurred a small increase in computational time once trained.

7.1.2 Parameter Subset Selection for Agent-Based Models

In Chapter 5, we utilized parameter subset selection at the nominal values on the surrogate models for the Rebellion and Greenhouse models described in Sections 2.2 and 2.3. For the Rebellion model, we obtained consistent results with other sensitivity analysis methods. For the Greenhouse model, we had similar results, but identified 2 extra parameters as identifiable that were not influential according to the other sensitivity analysis methods. Parameter subset selection implemented at the nominal values is fast to run as it is a local sensitivity analysis method.

7.1.3 Active Subspace Analysis for Agent-Based Models

In Chapter 5, we utilized active subspace analysis methods on the surrogate models for the Rebellion and Greenhouse models described in Sections 2.2 and 2.3. Recall that active subspace analysis identifies subspaces of influential parameters, while sensitivity analysis identifies subsets of influential parameters. This is advantageous as it permits use of numerically stable decompositions, such as the SVD decomposition, and it can be used for parameter dimension reduction. We used the surrogate models to get the activity scores for each parameter. We compared the influence of each parameter according to the activity score to that of Morris and Sobol’ analysis. We found consistent results with the other methods utilized. Further, the computation of activity scores required less model computations than the computation of Sobol’ indices required.

7.1.4 Bayesian Inference on Active Subspaces for Agent-Based Models

In Chapter 6, we utilized Bayesian inference on active subspaces for the surrogate models of the Rebellion model described in Section 2.2 to determine posterior distributions for each parameter. We compared the results of this analysis with that of Bayesian inference performed on the full space. We obtained that the results from the active subspace provided better mixing and more comprehensive results. The posteriors from the full space implementation of DRAM were often unclear, as they had high rejection rates. Further, the results of DRAM on the full space agreed with the results of DRAM on the active subspace, when they were decipherable.
7.2 Future Work

Future work includes:

- Apply active subspace analysis to surrogate models for a wider variety of agent-based models to verify the consistency with Sobol’ analysis. It would be good to create surrogate models for more agent-based models and apply active subspace analysis to those surrogate models. Further, we would like to compare those results with those of Sobol’ analysis - compare computation time, results, and verify that the Sobol’ sensitivity index and activity score bound holds from (4.6).

- Apply Bayesian inference on active subspaces for an agent-based model with a moderate to large parameter space, such as the Greenhouse model with $p = 40$ parameters. We would like to apply the techniques of Chapter 6 to an agent-based model with a larger parameter space. This would include creating surrogate models, determining the active subspace, verifying the active subspace, creating a surrogate model on the active subspace, and then applying Bayesian inference on the surrogate model on the active subspace. This requires transformation of the prior to the active subspace and transformation of the posterior back to the full space.

- Investigate time-dependent active subspaces for Bayesian inference. It is ideal to perform Bayesian inference on a model with a time-dependent response. However, active subspaces require an averaged or fixed time response. While one can average the active subspace over time as described in Section 4.4, this often does not yield a proper active subspace. One alternative is determine active subspaces independently at discrete times and utilize that active subspace for that particular time. Ideas for this investigation include:
  
  - Utilizing the posterior from time $t_0$ for the prior at time $t_1$ and continuing this process for a specified number of time steps.
  - Creating a time-dependent active subspace, where the active subspace transformation is a function of time, and using that to transform the parameter space to the active subspace.
Bibliography


[37] Programming Guide. URL: https://ccl.northwestern.edu/netlogo/docs/programming.html.


In this appendix, we present data tables referenced in Chapter 5. Each table consists of full normalized RMSE results for the train and test data sets for the various surrogates fitted in Section 5.1, 5.2, and 5.3. The normalized RMSE values for all training and test data sets for the 2-D averaged-time polynomial and neural network surrogate models for the Rebellion model are provided in Table A.1.

The normalized RMSE values for all training and test data sets for the 2-D averaged-time polynomial and neural network surrogate models for the Rebellion model.

Table A.1 Normalized RMSE for 2-D averaged-time polynomial and neural network surrogate models for the Rebellion model.

<table>
<thead>
<tr>
<th>Set \ Surrogate</th>
<th>Polynomial</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train Data Set</td>
<td>0.0240</td>
<td>0.0134</td>
</tr>
<tr>
<td>Test Data Set 1</td>
<td>0.0365</td>
<td>0.0151</td>
</tr>
<tr>
<td>Test Data Set 2</td>
<td>0.0436</td>
<td>0.0170</td>
</tr>
<tr>
<td>Test Data Set 3</td>
<td>0.0527</td>
<td>0.0118</td>
</tr>
<tr>
<td>Test Data Set 4</td>
<td>0.0384</td>
<td>0.0132</td>
</tr>
<tr>
<td>Test Data Set 5</td>
<td>0.0299</td>
<td>0.0208</td>
</tr>
<tr>
<td>Test Data Set 6</td>
<td>0.0523</td>
<td>0.0091</td>
</tr>
<tr>
<td>Test Data Set 7</td>
<td>0.0312</td>
<td>0.0148</td>
</tr>
<tr>
<td>Test Data Set 8</td>
<td>0.0472</td>
<td>0.0171</td>
</tr>
</tbody>
</table>
Gaussian process surrogate models for the Rebellion model are detailed in Table A.2.

The normalized RMSE values for all training and test data set from the 2-D time-dependent polynomial, Gaussian process, and neural network surrogate models for the Rebellion model are provided in Table A.3.

The normalized RMSE values for all training and test data sets from the 5-D averaged-time polynomial and neural network surrogate models for the Rebellion model are provided in Table A.4.

The normalize RMSE values for the 5-D Gaussian process averaged-time surrogate models for the Rebellion model are provided in Table A.5.

The normalized RMSE values for the 5-D time-dependent polynomial, Gaussian process, and neural network surrogate models are provided in Table A.6.

The normalized RMSE values for all the training and test data sets for the averaged-time Gaussian process and neural network surrogate models for the Greenhouse model are provided in Table A.7.

**Table A.2** Normalized RMSE for 2-D averaged-time Gaussian process surrogate for the Rebellion model utilizing various kernels.

<table>
<thead>
<tr>
<th>Set \ Kernel</th>
<th>Squared Exponential</th>
<th>Matern 3/2</th>
<th>Matern 5/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train Data Set</td>
<td>0.0141</td>
<td>0.0116</td>
<td>0.0128</td>
</tr>
<tr>
<td>Test Data Set 1</td>
<td>0.0168</td>
<td>0.0137</td>
<td>0.0150</td>
</tr>
<tr>
<td>Test Data Set 2</td>
<td>0.0188</td>
<td>0.0181</td>
<td>0.0182</td>
</tr>
<tr>
<td>Test Data Set 3</td>
<td>0.0222</td>
<td>0.0187</td>
<td>0.0195</td>
</tr>
<tr>
<td>Test Data Set 4</td>
<td>0.0151</td>
<td>0.0138</td>
<td>0.0142</td>
</tr>
<tr>
<td>Test Data Set 5</td>
<td>0.0218</td>
<td>0.0201</td>
<td>0.0207</td>
</tr>
<tr>
<td>Test Data Set 6</td>
<td>0.0151</td>
<td>0.0114</td>
<td>0.0122</td>
</tr>
<tr>
<td>Test Data Set 7</td>
<td>0.0173</td>
<td>0.0136</td>
<td>0.0149</td>
</tr>
<tr>
<td>Test Data Set 8</td>
<td>0.0239</td>
<td>0.0233</td>
<td>0.0241</td>
</tr>
</tbody>
</table>

**Table A.3** Normalized RMSE for 2-D time-dependent surrogate models for the Rebellion model.

<table>
<thead>
<tr>
<th>Set \ Surrogate</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train Data Set</td>
<td>0.1028</td>
<td>0.0176</td>
<td>0.0251</td>
</tr>
<tr>
<td>Test Data Set 1</td>
<td>0.1153</td>
<td>0.0248</td>
<td>0.0298</td>
</tr>
<tr>
<td>Test Data Set 2</td>
<td>0.1489</td>
<td>0.0248</td>
<td>0.0354</td>
</tr>
<tr>
<td>Test Data Set 3</td>
<td>0.1471</td>
<td>0.0302</td>
<td>0.0341</td>
</tr>
<tr>
<td>Test Data Set 4</td>
<td>0.1143</td>
<td>0.0226</td>
<td>0.0271</td>
</tr>
<tr>
<td>Test Data Set 5</td>
<td>0.1163</td>
<td>0.0291</td>
<td>0.0306</td>
</tr>
<tr>
<td>Test Data Set 6</td>
<td>0.1229</td>
<td>0.0217</td>
<td>0.0270</td>
</tr>
<tr>
<td>Test Data Set 7</td>
<td>0.1131</td>
<td>0.0257</td>
<td>0.0283</td>
</tr>
<tr>
<td>Test Data Set 8</td>
<td>0.1367</td>
<td>0.0338</td>
<td>0.0353</td>
</tr>
</tbody>
</table>
The normalized RMSE values for all the training and test data sets for the time-dependent Gaussian process and neural network surrogate models for the Greenhouse model are provided in Table A.8.

### Table A.4
Normalized RMSE for 5-D averaged-time polynomial and neural network surrogates for the Rebellion model.

<table>
<thead>
<tr>
<th>Set</th>
<th>Surrogate</th>
<th>Polynomial</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train Data Set</td>
<td>0.0521</td>
<td>0.0252</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 1</td>
<td>0.0836</td>
<td>0.0262</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 2</td>
<td>0.0664</td>
<td>0.0364</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 3</td>
<td>0.0669</td>
<td>0.0295</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 4</td>
<td>0.0713</td>
<td>0.0200</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 5</td>
<td>0.0736</td>
<td>0.0221</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 6</td>
<td>0.0625</td>
<td>0.0246</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 7</td>
<td>0.0861</td>
<td>0.0454</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 8</td>
<td>0.0558</td>
<td>0.0229</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 9</td>
<td>0.0764</td>
<td>0.0348</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 10</td>
<td>0.0809</td>
<td>0.0236</td>
<td></td>
</tr>
</tbody>
</table>

### Table A.5
Normalized RMSE for 5-D Gaussian process averaged-time surrogate for the Rebellion model utilizing various kernels.

<table>
<thead>
<tr>
<th>Set</th>
<th>Kernel</th>
<th>Squared Exponential</th>
<th>Matern 3/2</th>
<th>Matern 5/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train Data Set</td>
<td>0.0174</td>
<td>0.0105</td>
<td>0.0140</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 1</td>
<td>0.0299</td>
<td>0.0285</td>
<td>0.0289</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 2</td>
<td>0.0440</td>
<td>0.0413</td>
<td>0.0420</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 3</td>
<td>0.0278</td>
<td>0.0232</td>
<td>0.0236</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 4</td>
<td>0.0284</td>
<td>0.0284</td>
<td>0.0282</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 5</td>
<td>0.0252</td>
<td>0.0271</td>
<td>0.0261</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 6</td>
<td>0.0214</td>
<td>0.0200</td>
<td>0.0200</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 7</td>
<td>0.0351</td>
<td>0.0309</td>
<td>0.0315</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 8</td>
<td>0.0189</td>
<td>0.0173</td>
<td>0.0176</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 9</td>
<td>0.0342</td>
<td>0.0305</td>
<td>0.0312</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 10</td>
<td>0.0278</td>
<td>0.0229</td>
<td>0.0245</td>
<td></td>
</tr>
</tbody>
</table>
Table A.6 Normalized RMSE for 5-D time-dependent surrogates for the Rebellion model.

<table>
<thead>
<tr>
<th>Set</th>
<th>Surrogate</th>
<th>Polynomial</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train Data Set</td>
<td>0.2218</td>
<td>0.0026</td>
<td>0.0324</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 1</td>
<td>0.3413</td>
<td>0.0689</td>
<td>0.0635</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 2</td>
<td>0.3253</td>
<td>0.0572</td>
<td>0.0606</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 3</td>
<td>0.3014</td>
<td>0.0669</td>
<td>0.0657</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 4</td>
<td>0.2941</td>
<td>0.0584</td>
<td>0.0590</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 5</td>
<td>0.2993</td>
<td>0.0658</td>
<td>0.0599</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 6</td>
<td>0.2899</td>
<td>0.0568</td>
<td>0.0569</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 7</td>
<td>0.2612</td>
<td>0.0644</td>
<td>0.0590</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 8</td>
<td>0.2600</td>
<td>0.0558</td>
<td>0.0548</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 9</td>
<td>0.2633</td>
<td>0.0633</td>
<td>0.0560</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 10</td>
<td>0.2575</td>
<td>0.0541</td>
<td>0.0530</td>
<td></td>
</tr>
</tbody>
</table>

Table A.7 Normalized RMSE for the averaged-time Gaussian process and neural network surrogates for the Greenhouse model.

<table>
<thead>
<tr>
<th>Set</th>
<th>Surrogate</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train Data Set</td>
<td>0.0025</td>
<td>0.0035</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 1</td>
<td>0.0037</td>
<td>0.0043</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 2</td>
<td>0.0034</td>
<td>0.0044</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 3</td>
<td>0.0030</td>
<td>0.0041</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 4</td>
<td>0.0035</td>
<td>0.0041</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 5</td>
<td>0.0040</td>
<td>0.0046</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 6</td>
<td>0.0040</td>
<td>0.0049</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 7</td>
<td>0.0035</td>
<td>0.0047</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 8</td>
<td>0.0036</td>
<td>0.0044</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 9</td>
<td>0.0038</td>
<td>0.0042</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 10</td>
<td>0.0036</td>
<td>0.0042</td>
<td></td>
</tr>
</tbody>
</table>

Table A.8 Normalized RMSE for the time-dependent Gaussian process and neural network surrogates for the Greenhouse model.

<table>
<thead>
<tr>
<th>Set</th>
<th>Surrogate</th>
<th>Gaussian Process</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train Data Set</td>
<td>1.8183e-04</td>
<td>0.0047</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 1</td>
<td>0.0047</td>
<td>0.0064</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 2</td>
<td>0.0046</td>
<td>0.0065</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 3</td>
<td>0.0049</td>
<td>0.0071</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 4</td>
<td>0.0049</td>
<td>0.0067</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 5</td>
<td>0.0052</td>
<td>0.0070</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 6</td>
<td>0.0051</td>
<td>0.0064</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 7</td>
<td>0.0048</td>
<td>0.0066</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 8</td>
<td>0.0049</td>
<td>0.0066</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 9</td>
<td>0.0045</td>
<td>0.0070</td>
<td></td>
</tr>
<tr>
<td>Test Data Set 10</td>
<td>0.0046</td>
<td>0.0063</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX

B

ADDITIONAL FIGURES

In this appendix, we Figures referenced in Chapter 5. Each figure consists of goodness of fit plots for the train and test data sets for the various surrogates fitted in Section 5.1. Figure B.1 depicts the goodness of fit of the polynomial and neural network surrogate models to the training and test data for the 2-D averaged-time response for the Rebellion model. Figure B.2 depicts the goodness of fit of the Gaussian process surrogate model to the training and test data for the 2-D averaged-time response for the Rebellion model utilizing the squared exponential kernel, the Matern 3/2 kernel, and the Matern 5/2 kernel. Lastly, Figure B.3 depicts the goodness of fit of the polynomial, Gaussian process, and neural network surrogate models to the training and test data for the 2-T time-dependent response for the Rebellion model.
Figure B.1 Goodness of fit of 2-D averaged-time polynomial ((a) and (c)) and neural network ((b) and (d)) surrogate for the Rebellion model, where (a)-(b) depict the goodness of fit of the training data and (c)-(d) depict the goodness of fit of the test data.
Figure B.2 Goodness of fit of 2-D averaged time Gaussian process surrogate for the Rebellion model utilizing the squared exponential ((a) and (d)), Matern $3/2$ ((b) and (e)), and Matern $5/2$ ((c) and (f)) kernel, where (a)-(c) depict the goodness of fit of the training data and (d)-(f) depict the goodness of fit of the test data.

Figure B.3 Visual of goodness of fit of 2-D time-dependent polynomial ((a) and (d)), Gaussian process ((b) and (e)), and neural network ((c) and (f)) surrogate models for the Rebellion model, where (a)-(c) shows the goodness of fit of the training data and (d)-(f) shows the goodness of fit of the test data.
In this appendix, we detail the averaged-time and time-dependent polynomial surrogate models for the Rebellion model with $p = 2$ and $p = 5$ parameters.

### C.1 2-D Rebellion Averaged-Time Polynomial Surrogate

The seventh degree polynomial surrogate model for the 2D averaged-time response of the Rebellion model is provided as

$$p(\theta_1, \theta_2) = \sum_{i=1}^{35} \alpha_i \phi_i(\theta_1, \theta_2)$$

-28924296.732264, 3009272.719273, -489499.04291747, 71172.647362641, -2841.4394446977, 2619.8374280728, -343.4310200715$^T$ and $\phi(\theta_1, \theta_2) = [1, \theta_1, \theta_2, \theta_1^2, \theta_1\theta_2, \theta_1^3, \theta_1^2\theta_2, \theta_1\theta_3, 
\theta_2, \theta_3, \theta_1\theta_4, \theta_1^2\theta_2, \theta_1\theta_5, \theta_2, \theta_3^2, \theta_1\theta_6, \theta_1\theta_7, \theta_2, \theta_3\theta_2, \theta_1^2\theta_2, \theta_2\theta_3, \theta_1\theta_8, \theta_1\theta_9, \theta_2, \theta_3\theta_2, \theta_1\theta_10, \theta_2, \theta_3\theta_2]$. The goodness of fit of this surrogate model is provided in Figure B.1 and the normalized RMSE values are provided in Table A.1.

### C.2 2-D Rebellion Time-Dependent Polynomial Surrogate

The fifth degree polynomial surrogate model for the 2-D time-dependent response of the Rebellion model is provided as

$$p(\theta_1, \theta_2, t) = \sum_{i=1}^{55} \alpha_i \phi_i(\theta_1, \theta_2, t)$$

where $\alpha = [432531.91207, -1905783.1915, -113743.87698, 3466364.4053, 272976.5206, 29016.565463, -3671529.9969, 44663.592614, -100938.92692, -1039.1535825, 2085564.8329, -181897.96861, 31636.235501, 12336.878428, -408.84708762, -537614.40654, 119468.26559, -20423.78485, 1514.9541507, -906.67461709, 51.942174358, -9816.1651348, 30585.88904, 2511.5693279, -21452.312401, -7366.4527945, -128.72767555, -5396.0258274, 5833.9657657, 535.95749413, -18.37564413, 7443.9232053, -1832.8366673, -183.77029418, -13.235758811, 1.8304527551, 242.58349717, -1400.8163035, -35.907596209, 1620.7296536, 73.339188269, 2.0416608522, -562.87633027, -8.7515944189, -3.968118908, 0.041256230652, 17.462842708, -5.1441919517, -0.27580809583, -5.3364222701, -0.68820164772, 0.013330281223, -0.63619580406, 0.34222709948, 0.016598181917, 0.0048994707324$^T$ and $\phi(\theta_1, \theta_2, t) = [1, \theta_1, \theta_2, \theta_1^2, \theta_1\theta_2, \theta_1^3, \theta_1\theta_3, \theta_2, \theta_3, \theta_1\theta_4, \theta_1^2\theta_2, \theta_1\theta_5, \theta_2, \theta_3^2, \theta_1\theta_6, \theta_1\theta_7, \theta_2, \theta_3\theta_2, \theta_1\theta_8, \theta_1\theta_9, \theta_2, \theta_3\theta_2, \theta_1\theta_10, \theta_2, \theta_3\theta_2]$. The goodness of fit of this surrogate model is provided in Figure B.3 and the normalized RMSE values are provided in Table A.3.

### C.3 5-D Rebellion Averaged-Time Polynomial Surrogate

The fifth degree polynomial surrogate model for the 5-D averaged-time response of the Rebellion model is provided as

$$p(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5) = \sum_{i=1}^{125} \alpha_i \phi_i(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$$
where $\alpha = [233214.05174, -8471.8209103, -530.08693921, -87.455019985, -898841.96697,\
-5920.7296231, 1741.0429581, -69.197718076, -3343.5549781, 4097.5174398, 1208.8495787,\
16.616703951, -12.930734165, -1513.5402521, 24.930185455, -128.69229521, 5400.3733048,\
372.21365998, 1785438.9799, 1609.8558777, 117.04320591, 67.75025477, 1.49132807,\
179.43605629, -545.14303051, -174.35400088, -1.8051192781, 24.44757897, 336.34779076,\
-1.3407882953, 197.52274454, 767.18478643, 11.545818341, -24696.520948, 107.92469828,\
-17.794291404, -0.053352331167, -1.0824104584, 0.3523893215, -0.25734429036,\
0.65495297907, 81.795011193, 0.40937952618, 401.6675051, 18.483266356, -0.41993856494,\
-1466490.6113, -2502.7352597, -11.863579363, -0.75688573726, -25.830682773, 0.10751800814,\
1.4581381578, 16.52067616, 10.597254581, 0.020707506916, -0.58400084505, 1.8044119239,\
-0.11933819086, 0.083759560738, -171.71328471, -0.8610963463, 974.62514888, -3.0060122813,\
1.0592802779, 0.0051194571138, 0.064177618239, -0.33415383316, 0.013684284054,\
-1.2607984651, -16.420106967, 0.03958194454, -91.999716508, -1.7680707534, 0.023385903059,\
-5.7322020677, -8.5668594971, 0.41994896065, 1209.2112874, -2.358689782, -0.2131693794,\
6317.4010007, 78.66456103, -1.2294208479, 0.10947434223, -3.378837227e-05,\
0.0033567713758, -0.001182346694, 0.00061472497297, 0.0052373997107, -0.025403902267,\
0.018770246903, 1.7970625922, -0.041015677345, 0.001663036533, 0.13394913899,\
2.2842186435, -0.033405209435, -22.514465167, -0.32720637942, -0.002438382211,\
-57.189753067, -2.3660800497, 0.003489030141, 0.001010128354, -1.0995903449,\
-20.752490658, 1.2402834657, 80.813811027, 9.8814718719, -0.11134435216, 3938.3833507,\
-48.63751302, -0.06533120253, 0.10538747998, 458245.84059, 286.34646124, 33.463761868,\
-0.41866833098, -0.0015707025084]^T and $\phi(\theta_1, \theta_2, \theta_4, \theta_5, \theta_3) = [1, \theta_1, \theta_2, \theta_4, \theta_5, \theta_3^2, \theta_2 \theta_3, \theta_2 \theta_4, \theta_2 \theta_5, \theta_2^2, \theta_2 \theta_3^2, \theta_2 \theta_4^2, \theta_2 \theta_5^2, \theta_3 \theta_4, \theta_3 \theta_5, \theta_4 \theta_5, \theta_1 \theta_2 \theta_3, \theta_1 \theta_2 \theta_4, \theta_1 \theta_2 \theta_5, \theta_1 \theta_3 \theta_4, \theta_1 \theta_3 \theta_5, \theta_1 \theta_4 \theta_5, \theta_2 \theta_3 \theta_4, \theta_2 \theta_3 \theta_5, \theta_2 \theta_4 \theta_5, \theta_3 \theta_4 \theta_5, \theta_1 \theta_2 \theta_3 \theta_4, \theta_1 \theta_2 \theta_3 \theta_5, \theta_1 \theta_2 \theta_4 \theta_5, \theta_1 \theta_3 \theta_4 \theta_5, \theta_2 \theta_3 \theta_4 \theta_5, \theta_1 \theta_2 \theta_3 \theta_4 \theta_5, \theta_1 \theta_2 \theta_4 \theta_5, \theta_1 \theta_3 \theta_4 \theta_5, \theta_2 \theta_3 \theta_4 \theta_5, \theta_1 \theta_2 \theta_3 \theta_4 \theta_5]$. The goodness of fit of this surrogate model is provided in Figure 5.19 and the normalized RMSE values are provided in Table A.4.
C.4 5-D Rebellion Time-Dependent Polynomial Surrogate

The third degree polynomial surrogate model for the 5-D time-dependent response of the Rebellion model is provided as

\[ p(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, t) = \sum_{i=1}^{83} \alpha_i \varphi_i(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, t) \]

where \( \alpha = [\ldots] \) and the normalized RMSE values are provided in Table A.6.
In this appendix, we present the results of running active subspace analysis on the Greenhouse model without normalizing the parameters to consistent intervals. Due to the differences in magnitude of some parameters, we will see that we get unreliable results highlighting the need for normalization of the gradient.

**Activity Scores**

The active subspaces and activity scores are computed using $M = 10^6$ pseudo-random samples. This value was determined in the same manner as the Sobol’ indices. The gradient was computed with finite difference methods using a step size of $10^{-6}$ multiplied by the nominal value and by automatic differentiation on the Gaussian process. The eigenvalues of the approximated matrix $C$ are provided in Figure D.1. We can clearly see a 10 dimensional active subspace is appropriate for the neural network surrogate. For the Gaussian process surrogate, we consider active subspaces varying in dimension from 1 to 12 and obtain that the lowest dimension, best fit is provided by the 11 dimensional active subspace.

We provide the active subspace verification for the neural network and Gaussian process surrogates in Figure D.2, by creating surrogate models on the active subspace and
Figure D.1 Eigenvalues for active subspace analysis of the Greenhouse model where the neural network surrogate is used in (a) and the Gaussian process surrogate is used in (b).

Comparing predictions of the active subspace surrogate model with that of the full space surrogate model.

We provide the activity scores in Figure D.3 and obtain different results from the different surrogates. For the neural network, we obtain that the most influential parameter is parameter 6 and for the Gaussian process, we obtain that the most influential parameter is parameter 34. These results are not consistent between surrogate models and are not consistent with the previous one-at-a-time analysis, Pearson correlation, nor Sobol’ indices.

Figure D.2 Active subspace verification for the (a) neural network and (b) Gaussian process surrogate.
Figure D.3 Activity scores computed using the (a) neural network and (b) Gaussian process surrogate.