Energy Storage Applications of the Knowledge Gradient for Calibrating Continuous Parameters, Approximate Policy Iteration using Bellman Error Minimization with Instrumental Variables, and Covariance Matrix Estimation using an Errors-in-Variables Factor Model

Warren Robert Scott

A Dissertation
Presented to the Faculty of Princeton University in Candidacy for the Degree of Doctor of Philosophy

Recommended for Acceptance by the Department of Operations Research and Financial Engineering Adviser: Professor Warren B. Powell

June 2012
Abstract

We describe an adaptation of the knowledge gradient, originally developed for discrete ranking and selection problems, to the problem of calibrating continuous parameters for the purpose of tuning a simulator. The knowledge gradient for continuous parameters uses a continuous approximation of the expected value of a single measurement to guide the choice of where to collect information next. We show how to find the parameter setting that maximizes the expected value of a measurement by optimizing a continuous but nonconcave surface. We compare the method to sequential kriging for a series of test surfaces, and then demonstrate its performance in the calibration of an expensive industrial simulator.

We next describe an energy storage problem which combines energy from wind and the grid along with a battery to meet a stochastic load. We formulate the problem as an infinite horizon Markov decision process. We first discretize the state space and action space on a simplified version of the problem to get optimal solutions using exact value iteration. We then evaluate several approximate policy iteration algorithms and evaluate their performance. We show that Bellman error minimization with instrumental variables is equivalent to projected Bellman error minimization, previously believed to be two different policy evaluation algorithms. Furthermore, we provide a convergence proof for Bellman error minimization with instrumental variables under certain assumptions. We compare approximate policy iteration and direct policy search on the simplified benchmark problems along with the full continuous problems.

Finally, we describe a portfolio selection method for choosing virtual electricity contracts in the PJM electricity markets, contracts whose payoffs depend on the difference between the day-ahead and real-time locational marginal electricity prices in PJM. We propose an errors-in-variables factor model which is an extension of the classical capital asset pricing model. We show how the model can be used to estimate
the covariance matrix of the returns of assets. For US equities and PJM virtual contracts, we show the benefits of the portfolios produced with the new covariance estimation method.
Acknowledgements

I would like to thank Prof. Powell for his tireless help and guidance along problems that I felt were relevant, interesting, and challenging. I would also like to thank my fellow ORFE graduate students who helped motivate me and kept me inspired. A special thanks to my parents whose constant help and support kept me on the path to finishing. And of course, the highlight of my PhD experience has been meeting my wonderful wife Astrid.
To my wife and parents.
Contents

Abstract ......................................................... iii
Acknowledgements .............................................. v
List of Tables .................................................... xii
List of Figures ................................................... xiv

1 Introduction ................................................... 1
   1.1 The Knowledge Gradient for Calibrating Continuous Parameters 3
   1.2 Approximate Dynamic Programming for Energy Storage ........ 3
   1.3 Portfolio Selection and Covariance Matrix Estimation using an Errors-in-Variables Factor Model .... 4

2 The Knowledge Gradient for Continuous Parameters ............. 6
   2.1 Introduction .................................................. 6
   2.2 Literature Review ........................................... 8
   2.3 The Model .................................................... 10
      2.3.1 Covariance Structure .................................. 11
      2.3.2 Updating Equations .................................... 12
      2.3.3 Parameter Estimation .................................. 15
      2.3.4 The Knowledge-Gradient Policy ....................... 17
   2.4 The Knowledge Gradient for Continuous Parameters .......... 18
      2.4.1 Comparison to the Expected Improvement of EGO .......... 20
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4.2</td>
<td>Calculation of the Knowledge Gradient for Continuous Parameters</td>
<td>21</td>
</tr>
<tr>
<td>2.4.3</td>
<td>The Gradient of the Knowledge Gradient for Continuous Parameters</td>
<td>24</td>
</tr>
<tr>
<td>2.4.4</td>
<td>Maximizing the Knowledge Gradient for Continuous Parameters</td>
<td>27</td>
</tr>
<tr>
<td>2.4.5</td>
<td>The KGCP Policy</td>
<td>29</td>
</tr>
<tr>
<td>2.5</td>
<td>Convergence</td>
<td>30</td>
</tr>
<tr>
<td>2.6</td>
<td>Numerical Results</td>
<td>39</td>
</tr>
<tr>
<td>2.6.1</td>
<td>Standard Test Functions</td>
<td>40</td>
</tr>
<tr>
<td>2.6.2</td>
<td>Calibration Problem</td>
<td>42</td>
</tr>
<tr>
<td>2.6.3</td>
<td>Airline Calibration</td>
<td>44</td>
</tr>
<tr>
<td>2.6.4</td>
<td>Performance on Calibration Problem</td>
<td>45</td>
</tr>
<tr>
<td>2.7</td>
<td>Conclusion and Future Work</td>
<td>47</td>
</tr>
<tr>
<td>3</td>
<td>Approximate Dynamic Programming for Energy Storage with New Results on Instrumental Variables and Projected Bellman Errors</td>
<td>48</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>48</td>
</tr>
<tr>
<td>3.2</td>
<td>Models</td>
<td>51</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Wind</td>
<td>51</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Electricity Prices</td>
<td>52</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Electricity Demand</td>
<td>55</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Energy Storage</td>
<td>56</td>
</tr>
<tr>
<td>3.3</td>
<td>Dynamic Programming Problem</td>
<td>57</td>
</tr>
<tr>
<td>3.3.1</td>
<td>State Variable</td>
<td>57</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Decision Variables</td>
<td>57</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Exogenous Information Process</td>
<td>59</td>
</tr>
<tr>
<td>3.3.4</td>
<td>State Transition</td>
<td>59</td>
</tr>
<tr>
<td>3.3.5</td>
<td>Contribution and Objective</td>
<td>60</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Summary of Algorithm 1 from [36]</td>
<td>23</td>
</tr>
<tr>
<td>2.2</td>
<td>The KGCP Policy</td>
<td>29</td>
</tr>
<tr>
<td>2.3</td>
<td>Performance on Standard Test Functions. Each row summarizes 500</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>runs of each policy on the specified test function with the specified</td>
<td></td>
</tr>
<tr>
<td></td>
<td>observation noise variance. We define $\sigma(OC)$ as $\text{Std}(\mathbb{E}(OC))$ and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\text{Med}$ as the median $OC$.</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>Set of benchmark problems specifying the type (Full or Battery Arbitrage), the number of discretization levels for time (1=steady state),</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>resource, price, load (1=deterministic) and wind. The remaining</td>
<td></td>
</tr>
<tr>
<td></td>
<td>columns specify average maximum wind divided by the load, storage</td>
<td></td>
</tr>
<tr>
<td></td>
<td>capacity divided by hourly load, round trip efficiency (RTE), and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the maximum charge/discharge rate (C/10 means it takes hours to</td>
<td></td>
</tr>
<tr>
<td></td>
<td>charge/discharge).</td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td>Parameter settings for problems with continuous states. Problems 1-3</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>have time-dependent stochastic loads and electricity prices. Problems</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4-10 are steady-state.</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>The portfolio statistics of the portfolios shown in Figure 4.8 $\mu$ is the</td>
<td>107</td>
</tr>
<tr>
<td></td>
<td>annualized geometric return, and $\sigma$ is the annualized standard devia-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tion.</td>
<td></td>
</tr>
</tbody>
</table>
4.2 The portfolio statistics of the portfolios shown in Figure 4.10 which use portfolios chosen with errors-in-variables CAPM. $\mu$ is the annualized geometric return, and $\sigma$ is the annualized standard deviation.
List of Figures

2.1 Algorithm 1 is a scan line algorithm to re-express \( f(z) = \max_{i=0,...,n} a_i + b_i z \) as \( f(z) = \sum_{i=0}^{n}(\tilde{a}_i + \tilde{b}_i z)1_{[x_i,x_{i+1})}(z) \).

2.2 (a) The estimate of the function along with the 95% confidence intervals of the estimate after 4 observations. (b) The knowledge gradient for continuous parameters (KGCP) and exact knowledge gradient over a finely discretized set of decisions (KGCB) after 4 observations. (c) The estimate of the function after 9 observations. (d) The knowledge gradient after 9 observations.

2.3 A diagram of the points: \( x^{acc} \) is an accumulation point; \( x^{mult} \) is a point being measured multiple times; \( x^{near} \) is a point near to \( x^d \) we are considering to measure; \( x^d \) is an arbitrary fixed point in the open ball centered at \( x^{acc} \).

2.4 (a) The negative of the Branin function. (b) A contour plot of the negative Branin function. We will maximize the negative of the Branin function using noisy observations normally distributed around the true function.
2.5 (a) The estimate of the function after 6 observations. The actual observations are plotted as well. (b) The knowledge gradient for continuous parameters surface is plotted. The height is a measure of how much we expect the maximum of the estimate of the function to increase by measuring the corresponding decision. We choose the next sampling decision by finding the decision which maximizes the knowledge gradient for continuous parameters shown in 2.5(b).

2.6 (a) The estimate of the function after 20 observations. The actual observations are plotted as well. (b) The contour plot of the estimate of the function after 20 observations.

2.7 (a)-(c) show examples of Gaussian Processes with the given covariance parameters. (d)-(f) show the mean opportunity cost of the KGCP policy on the various Gaussian processes.

2.8 We show the performance of the algorithms on the calibration problem. The estimates of the objective of each algorithm are shown from five sample paths.

2.9 Frequency histograms showing where the AKG policy sampled in each dimension for a particular sample path.

3.1 The energy flow diagram.

3.2 The average PJM Real-Time price at the Western Hub as a function of the hour of the week.

3.3 The total ERCOT load over the first full week of 2010.

3.4 Summary of approximate policy iteration. The inner loop simulates transitions from a fixed policy in order to approximately evaluate the fixed policy. The outer loop improves the policy.
3.5 Performance as a percent of the benchmark optimization solution using API with instrumental variables, least-squares API, a myopic policy and direct policy search. ................................................................. 80

3.6 We plot a 10 day sample path of a policy produced by approximate policy iteration with instrumental variables Bellman error minimization using quadratic basis functions on Problem 1. (a) We plot the electricity price and resource level. (b) We plot a histogram of the resource level. ................................................................. 81

3.7 The algorithms use quadratic basis functions. We show the percentage of optimal along with 95% confidence intervals for the average percentage of optimal for Bellman error minimization using instrumental variables (IV) when only certain dimensions of the post-state are included in the post-state value function approximation. ....... 82

3.8 We plot the average objective of both approximate policy iteration algorithms on the continuous problems shown in Table 3.2. .............. 84

3.9 A sample path of the electricity spot price and resource level. The vertical lines correspond to midnight. ........................................ 84

4.1 (a) The CAPM estimate of $\beta$ for Chevron. (b) The CAPM estimate of $\beta$ for ExxonMobil (c) The residuals for the fit of Chevron’s $\beta$. (d) The residuals for the fit of ExxonMobil’s $\beta$. ...................... 97

4.2 (a) The errors-in-variables CAPM estimate of $\beta$ for Chevron. (b) The errors-in-variables CAPM estimate of $\beta$ for ExxonMobil. (c) The residuals for the fit of Chevron’s $\beta$. (d) The residuals for the fit of ExxonMobil’s $\beta$. ......................................................... 98
4.3 (a) The residuals between Chevron and ExxonMobil using CAPM as shown in Figures 4.1(a) and 4.1(b). (b) The residuals between Chevron and ExxonMobil using the errors-in-variables CAPM as shown in Figures 4.2(a) and 4.2(b).

4.4 The cumulative sum of the 1 AM real-time price minus the 1 AM day-ahead price for the PJM hubs. When the cumulative sum slopes upwards, the day-ahead price is lower than the real-time price on average. When the cumulative sum slopes downwards, the real-time price is lower than the day-ahead price on average.

4.5 The cumulative sum of the 5 PM real-time price minus the 5 PM day-ahead price for the PJM hubs.

4.6 The cumulative sum of the real-time price minus day-ahead price for a particular hour of the day at the Western Hub. Early in the morning, electricity is cheapest in the day-ahead market on average. In the middle of the day, electricity is cheapest in the real-time market on average.

4.7 The cumulative sum of the real-time price minus day-ahead price for each hour of the day at the New Jersey Hub.

4.8 Starting from one dollar, wealth paths different portfolios. A calibration length of 5 days was used for CAPM and errors-in-variables CAPM.

4.9 $\mu$ is the annualized geometric return, and $\sigma$ is the annualized volatility. We show the results using calibration lengths of (a) 5 days. (b) 20 days. (c) 100 days. (d) 500 days.

4.10 Starting from one dollar, wealth paths of portfolios chosen using errors-in-variables CAPM with different risk aversions, $A$.

4.11 The efficient frontier for the PJM model.
<table>
<thead>
<tr>
<th>Section</th>
<th>Content</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1</td>
<td>When there are errors in the regressors, instrumental variables can be used to solve the problem</td>
<td>130</td>
</tr>
<tr>
<td>B.2</td>
<td>The algorithms use linear basis functions. We show the performance of Bellman error minimization using instrumental variables (IV) and least-squares Bellman error minimization (LS) along with direct policy search (KGCP)</td>
<td>136</td>
</tr>
<tr>
<td>B.3</td>
<td>Third order basis functions. We show the performance of Bellman error minimization using instrumental variables (IV) and least-squares Bellman error minimization (LS).</td>
<td>137</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

We first consider the fundamental and widely applicable problem of tuning parameters of a simulator to achieve specific performance metrics. This simulation optimization problem requires searching over a continuous, multidimensional parameters space to find settings which produce good results. Each measurement is time consuming and yet produces noisy measurements. A balance between exploration and exploitation (global versus local search) must be achieved when sequentially deciding the parameter settings with which to run the simulator (see [71]). Sequentially running the simulator with random parameter settings or over a grid is typically highly inefficient. The knowledge gradient framework provides a measure for the value of running the simulator with a particular parameter setting and can be used to efficiently sequentially choose parameter settings to run the simulator with. In additional, we show that the knowledge gradient for continuous parameters has nice convergence theory. The knowledge gradient for continuous parameters has many applications, such as tuning the parameters of an airline simulator or optimizing the parameters of an energy storage policy.

We next consider the problem of using energy from the grid, with a stochastic price, along with wind energy and a battery to provide electricity to a stochastic load.
Inherent in this problem is the question of when to charge and discharge the battery in order to maximize profit. For the purpose of benchmarking, we formulate the problem as an infinite horizon discrete-time Markov decision process and discretize the state and action space. For the full energy storage problem, true optimal solutions cannot be computed, but approximate dynamic programming algorithms can be used to find good policies. For example, the concept of Bellman error minimization can be used inside an approximate policy iteration framework to find parametric value functions which approximately solve Bellman’s optimality equation. A fundamentally different approach is to use direct policy search. Direct policy search uses simulation optimization, such as the knowledge gradient for continuous parameters, to optimize a parametric policy over a vector of continuous parameters.

The final problem we consider is that of choosing a portfolio of virtual electricity contracts in the PJM network. Virtual electricity contracts, consisting of incs and decs, are distributed daily by PJM and can be used to hedge the risk of electricity spikes in the spot market along with volatile congestion prices. The payoff of these contracts is simply the difference between the day-ahead and real-time price at a particular node in the PJM network. However, portfolios of these contracts are more desirable because of the large volatility in the payoff of a single contract. Constructing covariance matrices is fundamental to many portfolio selection techniques dating back to Harry Markowitz (see [60]). We show how an errors-in-variables factor model can be used to construct a full rank covariance matrix. By backtesting mean-variance efficient portfolios, we can conclude whether our covariance matrix estimation technique adds value for choosing portfolios of virtual electricity contracts.
1.1 The Knowledge Gradient for Calibrating Continuous Parameters

In Chapter 2, we extend the concept of the correlated knowledge-gradient policy for ranking and selection of a finite set of alternatives to the case of continuous decision variables. We propose an approximate knowledge gradient for problems with continuous decision variables in the context of a Gaussian process regression model in a Bayesian setting, along with an algorithm to maximize the approximate knowledge gradient. In the problem class considered, we use the knowledge gradient for continuous parameters to sequentially choose where to sample an expensive noisy function in order to find the maximum quickly. We show that the knowledge gradient for continuous decisions is a generalization of the efficient global optimization algorithm proposed by Jones, Schonlau, and Welch. We then compare the method to sequential kriging on a series of test surfaces and demonstrate its performance on the calibration of an industrial airline simulator.

1.2 Approximate Dynamic Programming for Energy Storage

In Chapter 3, we address the problem of developing near-optimal policies for an energy system that combines energy from an exogenously varying supply (wind) and price (the grid) to serve time dependent and stochastic loads in the presence of a storage device. The goal is to consider three different types of uncertainty in a stochastic control problem that arises when using energy from renewables. With a few simplifications, the problem can be solved optimally using classical dynamic programming methods, but the full problem can only be solved approximately. We compare several approximate dynamic programming algorithms, including approxi-
mate policy iteration using least-squares Bellman error minimization, Bellman error minimization using instrumental variables, and least-squares projected Bellman error minimization. We show for the first time that Bellman error minimization using instrumental variables is mathematically equivalent to projected Bellman error minimization, previously thought to be fundamentally different algorithmic strategies. We show that Bellman error minimization using instrumental variables, implemented in an approximate policy iteration algorithm, significantly outperforms classical least-squares policy iteration, but underperforms direct policy search. All of these are tested using realistic data, and are compared against optimal benchmarks.

1.3 Portfolio Selection and Covariance Matrix Estimation using an Errors-in-Variables Factor Model

In Chapter 4, we propose a method for buying and selling portfolios of electricity contracts in the PJM real-time and day-ahead markets, where our focus is identifying robust portfolios that balance risk and return, independent of the decision of whether to be invested or not. In addition to determining the hourly day-ahead and real-time electricity prices for each location in the network, PJM provides a market for hedging instruments including virtual contracts, financial transmission rights, and up-to congestion contracts. The large costs of storing energy combined with the many sources of uncertainty in PJM’s supply, demand, and transmission create a unique and complicated relationship between the day-ahead and real-time prices for electricity. We first focus on estimating the covariance matrix of electricity prices at different locations using an errors-in-variables extension to a factor model, where we propose a
model that allows us to build a covariance matrix based on very short histories. We then use this covariance matrix to construct efficient portfolios of electricity contracts.
Chapter 2

The Knowledge Gradient for Continuous Parameters

2.1 Introduction

Our goal is to find the global maximum of a real valued continuous function that is expensive to compute and that can only be evaluated with uncertainty. We need an algorithm that can give satisfactory results with as few function evaluations as possible. For this reason, we are willing to spend extra time deciding where we would like to evaluate the function next. This problem arises in applications such as simulation optimization, the design of machinery, medical diagnostics, biosurveillance, and the design of business processes.

We extend the concept of the knowledge-gradient policy for correlated beliefs (KGCB) described in [37] and [36], originally developed to find the best of a finite set of alternatives, to problems where we are trying to maximize over a multidimensional set of continuous variables. The KGCB policy maximizes the marginal value of a single measurement and has produced very promising results in discrete ranking and selection problems without requiring the use of any tunable parameters. In [36]
the KGCB policy is used in a simulation optimization application to tune a set of continuous parameters which must be discretized to perform the search. However, the KGCB policy becomes computationally too expensive when it is necessary to discretize over a large multidimensional vector. We extend the knowledge gradient to multidimensional continuous problems, and then show that the knowledge gradient concept is at least competitive with, or outperforms, specialized algorithms for specific problems.

Although the concept for the knowledge gradient is very general, we choose to model the function to be optimized using Gaussian process regression with a squared exponential covariance function and model the noise in the observations as additive Gaussian noise. The knowledge gradient for continuous parameters (KGCP) policy that we propose extends the well known efficient global optimization algorithm in [45] to the case of noisy observations. When choosing a sampling decision, the KGCP accounts for the fact that an additional observation will update the regression function at unsampled decisions as well as at the sampling decision; the updated best decision will not necessarily be the current best decision or sampling decision.

This paper makes the following contributions: (1) We propose an approximation to the knowledge gradient for multidimensional continuous decision variables which can be efficiently computed; (2) We describe a gradient ascent algorithm that can be used to maximize the knowledge gradient for continuous parameters without resorting to discretization; (3) We prove that, under mild conditions, the knowledge gradient for continuous parameters policy applied to maximizing a continuous function with observation noise will cause the uncertainty in the regression model to disappear in the limit; (4) We examine the competitive performance with sequential kriging, a widely used algorithm which lacks our theoretical guarantees, on a series of test functions.

This paper is organized as follows. Section 2.2 reviews the literature for continuous global optimization problems. Section 2.3 describes the Bayesian model capturing our
prior belief in the function being optimized. We review the knowledge gradient for
discrete alternatives, which guides measurements by computing the marginal value of
information. Section 2.4 describes how the knowledge gradient can be computed for
continuous measurements. The knowledge gradient for continuous parameters is then
compared to the expected improvement in [45]. Our approach requires approximating
the knowledge gradient as a continuous function, and we derive a gradient ascent
algorithm for this purpose. In Section 2.5 we give mild conditions under which the
posterior variance at each decision in the regression model will go to zero almost surely
when using the knowledge gradient for continuous parameters policy for finding the
global maximum of a function with observation noise. Finally, Section 2.6 compares
the knowledge gradient for continuous parameters to sequential kriging optimization
[44], which is a popular algorithm for determining sequential measurements in the
presence of noise, on a set of test functions.

2.2 Literature Review

We briefly present and summarize some of the current approaches to maximizing
an expensive function with observation noise. The applications are vast, and multi-
ple research disciplines have addressed the problem. Simulation optimization covers
gradient-based methods (see [75], [18], [89], [39], and [76]), direct search methods
(see [45]), and metaheuristics (see [38]). The term model-based optimization can be
used to categorize the fields of trust regions (see [70], [21], [20], and [25]), response
surface methodology (see [9], [43], [65], [66], [67], and [74]), and the surrogate manage-
ment framework (see [7]). Finally, Bayesian global optimization consists of algorithms
which combine Bayesian models of the function with single-step look ahead criteria.

Bayesian global optimization takes a statistical approach to optimizing functions
efficiently (see [78]). One of the first approaches in the field is [52] that approxi-
mates a one-dimensional function with a Wiener process and uses a probability of improvement criterion to choose the next point to sample. \[93\] uses the probability of improvement concept for higher dimensions in the P-algorithm. \[102\] as well as \[63\] and \[56\] also use a one-dimensional Wiener process but then use expected improvement criteria to choose the next point to sample; they discuss convergence in the case of no observation noise. For the case of no observation noise, \[77\] introduces the popular DACE (design and analysis of computer experiments) kriging model to approximate the expensive function; a kriging model is a method of interpolation based on random spatial processes (see \[61\], \[23\], \[49\], and \[50\]) and is referred to as Gaussian process regression in computer science (see \[73\]). \[45\] presents the efficient global optimization (EGO) algorithm for optimizing expensive functions without noise which combines a kriging model with an expected improvement criterion (also see \[79\] and \[80\]). Work has been done in \[96\] to prove convergence for an expected improvement algorithm in the case of no observation noise if the true function comes from a reproducing kernel Hilbert space generated by the covariance function. Another example of Bayesian global optimization is \[42\] which combines radial basis interpolation and a utility function based on the uncertainty of the response surface weighted by how close the response surface’s value at that point is to a specified target value.

Recent extensions of Bayesian global optimization explicitly account for observation noise, although limited convergence theory has been developed for the following algorithms. In \[44\], sequential kriging optimization (SKO) combines a kriging model with an expected improvement criterion which accounts for noisy functions observations; the expected improvement criterion is weighted by a term that favors decisions with higher uncertainty. One challenge of SKO, like many other Bayesian global optimization algorithms, is maximizing the expected improvement criterion to find the next sampling decision; the Nelder-Mead simplex method is suggested. \[98\] and \[97\] present an informational approach to global optimization (IAGO) which combines
a kriging model, Monte Carlo, and other approximation techniques to estimate the
distribution of the global minimizer of the function after an additional observation.
The sampling decision is made by minimizing the entropy (which can be interpreted
as uncertainty) of the global minimizer. The approaches in [34] and [35] address the
issue of different levels of noise using an expected improvement criterion with kriging
models found in [23] which allow for noisy observations.

2.3 The Model

We consider the following optimization problem

$$\arg\max_{x \in X} \mu(x)$$  \hspace{1cm} (2.1)

where $x \in \mathbb{R}^p$ is a decision vector, $X$ is a compact feasible set of decisions, and
$\mu : \mathbb{R}^p \to \mathbb{R}$ is a continuous function we wish to maximize. Let $\hat{y}_{n+1}$ be the sam-
ple observation of the sampling decision $x^n$ for $n = 0, ..., N - 1$. The variance of an
observation, given $\mu$, at a decision $x$ is $\lambda(x)$, and we assume $\lambda : \mathbb{R}^p \to \mathbb{R}$ is continu-
ously differentiable over the domain $X$ and is known. In practice, the variance of the
observation noise is unknown but can be estimated. We assume $\hat{y}_{n+1}$ has a normal
distribution centered around the true function,

$$\hat{y}_{n+1} | \mu, x^n \sim \mathcal{N}(\mu(x^n), \lambda(x^n)),$$

and $\hat{y}_1, ..., \hat{y}_{N+1}$ are independent given $\mu$ and $x^0, ..., x^N$. (This assumption would be
violated if using the method of common random numbers (see [19]).) Our goal is to
sequentially choose $x^n$ at each iteration $n = 0, ..., N - 1$ in order to approach the
solution to (2.1) as quickly as possible.
Adopting a Bayesian framework, we start with some belief or information about the truth, $\mu$. We treat $\mu$ as a random variable and assign it a Gaussian process (GP) prior density. $\mu^n$ is our updated mean of our random variable, given $n$ observations. Then, for any $x^0, \ldots, x^n \in \mathcal{X}$, our a priori distribution is $[\mu(x^0), \ldots, \mu(x^n)]^T \sim \mathcal{N}(\mu^0([x^0, \ldots, x^n]), \Sigma^0([x^0, \ldots, x^n])$ where $\mu^0([x^0, \ldots, x^n]) = \mathbb{E}([\mu(x^0), \ldots, \mu(x^n)]^T)$ and $\Sigma^0([x^0, \ldots, x^n]) = \text{Cov}([\mu(x^0), \ldots, \mu(x^n)]^T)$. Next we define a filtration $\mathcal{F}^n$ where $\mathcal{F}^n$ is the sigma-algebra generated by $x^0, \hat{y}^1, \ldots, x^{n-1}, \hat{y}^n$. We define $\mu^n([x^0, \ldots, x^n]) = \mathbb{E}([\mu(x^0), \ldots, \mu(x^n)]^T|\mathcal{F}^n)$ and $\Sigma^n([x^0, \ldots, x^n]) = \text{Cov}([\mu(x^0), \ldots, \mu(x^n)]^T|\mathcal{F}^n)$ for $x^0, \ldots, x^n \in \mathcal{X}$. In addition we use the notation $\Sigma^n(x^0, x^1) = \text{Cov}(\mu(x^0), \mu(x^1)|\mathcal{F}^n)$.

The multivariate normal distribution is a natural conjugate family when the observations come from a normal distribution with known variance. This means our posterior is also multivariate normal. Hence, conditioned on $\mathcal{F}^n$, $[\mu(x^0), \ldots, \mu(x^n)]^T \sim \mathcal{N}(\mu^n([x^0, \ldots, x^n]), \Sigma^n([x^0, \ldots, x^n]))$. Next we explain a method to assign the initial covariance between $\mu(x^0)$ and $\mu(x^1)$.

### 2.3.1 Covariance Structure

In order to specify the covariance matrix for our a priori distribution of $\mu$ at $x^0, \ldots, x^n \in \mathcal{X}$, it is sufficient to specify a covariance function. Similar to [77] and [36], we assume a Gaussian covariance function. Letting $x^0$ and $x^1$ be arbitrary decisions in $\mathcal{X}$, we write,

$$\text{Cov}(\mu(x^0), \mu(x^1)) = \beta \exp(- \sum_{i=1}^{p} \alpha_i(x^0_i - x^1_i)^2), \quad \alpha > 0, \beta > 0, \quad (2.2)$$

where $\alpha \in \mathbb{R}^p$ is called the activity of $\mu$ and $\beta \in \mathbb{R}^1$ controls the uncertainty of our belief about $\mu$. The initial covariance function given in (2.2) is a metric, meaning the covariance of two decisions decreases as the distance between them increases. The parameter $\alpha_i$ for $i = 1, \ldots, p$ is called the activity in dimension $i$ and represents how smooth $\mu$ is in dimension $i$ (see [45]). For example, a very small $\alpha_i$ would make the
covariances bigger, indicating that $\mu$ is believed to be very smooth in dimension $i$. The key idea is that the true function should be positively correlated at nearby points. For example, if $\mu(x)$ is greater than $\mu^0(x)$, then, for small $\delta \in \mathbb{R}^p$, we should expect $\mu(x + \delta)$ to be greater than $\mu^0(x + \delta)$ as well, assuming $\mu$ is smooth. [73] explains that Gaussian processes with this covariance function are very smooth because they have mean square derivatives of all orders.

### 2.3.2 Updating Equations

After the first $n$ sampling decisions, the distribution of $[\mu(x^0), ..., \mu(x^{n-1})]^T$ conditioned on $\mathcal{F}^n$ is multivariate normal and hence completely characterized by $\mu^n([x^0, ..., x^{n-1}])$ and $\Sigma^n([x^0, ..., x^{n-1}])$, which can be calculated as follows in [2.6] and [2.7]. For a fixed $n$, define the matrix $\Sigma^0 = \Sigma^0([x^0, ..., x^{n-1}])$ which can be calculated using [2.2]. Given the assumptions in our model, we can use the Kalman filter equations in [62] or equivalently the Gaussian process regression equations given in [73] to compute the posterior distribution of $\mu$ given $\mathcal{F}^n$. We calculate the measurement residual $\tilde{y}^n$ and the residual covariance $S^n$ as

$$\tilde{y}^n = \begin{bmatrix} \hat{y}^1 \\ \vdots \\ \hat{y}^n \end{bmatrix} - \begin{bmatrix} \mu^0(x^0) \\ \vdots \\ \mu^0(x^{n-1}) \end{bmatrix}, \quad (2.3)$$

$$S^n = \Sigma^0 + \text{Diagonal}([\lambda(x^0), ..., \lambda(x^{n-1})]). \quad (2.4)$$

We can then calculate the optimal Kalman gain using

$$K^n = \Sigma^0[S^n]^{-1}. \quad (2.5)$$

Note that if the minimum value of the observation noises, $\lambda_{\text{min}}$, is strictly positive, $[S^n]^{-1}$ is well defined because the minimum eigenvalue of $S^n$ is greater than or equal
to $\lambda^{\text{min}}$. Let $I_n$ be an $n \times n$ identity matrix. Finally, the updated expected values of $\mu$ at the first $n$ sampled points, and the covariance matrix between $\mu$ at the first $n$ sampled points, conditioned on $\mathcal{F}^n$, are given respectively by

$$
\begin{align*}
\begin{bmatrix}
\mu^n(x^0) \\
\vdots \\
\mu^n(x^{n-1})
\end{bmatrix}
&= 
\begin{bmatrix}
\mu^0(x^0) \\
\vdots \\
\mu^0(x^{n-1})
\end{bmatrix}
+ K^n \tilde{y}^n, \\
\Sigma^n & = (I_n - K^n) \Sigma^0.
\end{align*}
\tag{2.6}
\tag{2.7}
$$

The above equations update the distribution of $\mu$ at the first $n$ sampling decisions conditioned on $\mathcal{F}^n$, but we also need to update the distribution of $\mu(x)$ conditioned on $\mathcal{F}^n$, where $x \in X$ is an arbitrary decision variable that has not been sampled yet. We can do this with the following equations. Define $\bar{\Sigma}^0 = \Sigma^0([x^0, \ldots, x^{n-1}, x])$ and $\bar{\Sigma}^n = \Sigma^n([x^0, \ldots, x^{n-1}, x])$, and let $\bar{0}$ be a column vector of zeros. Our new optimal Kalman gain is given by

$$
\bar{K}^n = \bar{\Sigma}^0 egin{bmatrix} I_n \\ \bar{0}^T \end{bmatrix} [\bar{\Sigma}^n]^{-1}.
\tag{2.8}
$$

We can now update $\mu^0$ and $\bar{\Sigma}^0$ with the following equations,

$$
\begin{align*}
\begin{bmatrix}
\mu^n(x^0) \\
\vdots \\
\mu^n(x^{n-1}) \\
\mu^n(x)
\end{bmatrix}
&= 
\begin{bmatrix}
\mu^0(x^0) \\
\vdots \\
\mu^0(x^{n-1}) \\
\mu^0(x)
\end{bmatrix}
+ \bar{K}^n \tilde{y}^n, \\
\bar{\Sigma}^n & = (I_{n+1} - \bar{K}^n \begin{bmatrix} I_n & \bar{0} \end{bmatrix}) \bar{\Sigma}^0.
\end{align*}
\tag{2.9}
\tag{2.10}
$$
If we explicitly want the distribution of $\mu(x)$ conditioned on $\mathcal{F}^n$ at some arbitrary decision $x$ we can pull out the pertinent formulae from (2.9) and (2.10);

$$
\mu^n(x) = \mu^0(x) + \left[ \Sigma^0(x^0, x), \ldots, \Sigma^0(x^{n-1}, x) \right] [S^n]^{-1} \tilde{y}^n, \quad (2.11)
$$

$$
\Sigma^n(x, x) = \Sigma^0(x, x) - \left[ \Sigma^0(x^0, x), \ldots, \Sigma^0(x^{n-1}, x) \right] [S^n]^{-1} \begin{bmatrix} \Sigma^0(x^0, x) \\ \vdots \\ \Sigma^0(x^{n-1}, x) \end{bmatrix}, \quad (2.12)
$$

Equation (2.11) is a linear smoother if $\mu^0(x) = 0 \ \forall x$ and is referred to as Gaussian process regression (GPR) in [73] and regressing kriging in [35]. There are also recursive equations equivalent to (2.9) and (2.10) which update $\mu^n$ and $\Sigma^n$ (see [36]). [36] shows that after we have selected our sampling decision, $x^n$, but before we observe $\hat{y}^{n+1}$, our updated regression function is normally distributed conditioned on the information available at iteration $n$:

$$
\begin{bmatrix} 
\mu^{n+1}(x^0) \\
\vdots \\
\mu^{n+1}(x^{n-1}) \\
\mu^{n+1}(x^n) 
\end{bmatrix} = \begin{bmatrix} 
\mu^n(x^0) \\
\vdots \\
\mu^n(x^{n-1}) \\
\mu^n(x^n) 
\end{bmatrix} + \tilde{\sigma}(\Sigma^n, x^n) Z^{n+1}, \quad (2.13)
$$

where $Z^{n+1} = (\hat{y}^{n+1} - \mu^n(x^n)) / \sqrt{\lambda(x^n) + \Sigma^n(x^n, x^n)}$, with

$$
\tilde{\sigma}(\Sigma, x) \triangleq \frac{\Sigma e_x}{\sqrt{\lambda(x) + e^T_x \Sigma e_x}}; \quad (2.14)
$$

here $e_x$ is a column vector of zeros with a 1 at the row corresponding to decision $x$. It can be shown that $Z^{n+1} \sim \mathcal{N}(0, 1)$ because $\text{Var}(\hat{y}^{n+1} - \mu^n(x^n)|\mathcal{F}^n) = \lambda(x^n) + \Sigma^n(x^n, x^n)$. 

14
2.3.3 Parameter Estimation

In general we will not be given the parameters of the covariance function, $\alpha$ and $\beta$, the variance of observation noise, $\lambda()$, or the mean of the initial prior distribution on $\mu$, $\mu^0()$. We briefly outline how to obtain maximum likelihood estimates and maximum a posteriori estimates of the hyperparameters after we have $n$ observations.

Maximum Likelihood Estimation of $\alpha$, $\beta$, $\lambda$, and $\mu^0$

In the model presented in Section 2.3, the observations come from a multivariate normal distribution. Recalling that $\Sigma^n$ and $S^n$ depend on the parameters $\alpha \in \mathbb{R}^p$ and $\beta \in \mathbb{R}$, the likelihood function can easily be written as

$$L(y(\alpha, \beta, \lambda(x^0), ..., \lambda(x^{n-1}), \mu^0(x^0), ..., \mu^0(x^{n-1}))) = (2\pi)^{-n/2} |S^n|^{-1/2} \exp\left(-\frac{1}{2} (\hat{y} - \mu^0(x^0))^T (S^n)^{-1} (\hat{y} - \mu^0(x^0)) \right).$$

Depending on the problem, the parameters $\lambda(\cdot)$ and $\mu^0(\cdot)$ may already be known, or may need to be estimated. Now, if we assume that the variance of the observation noise, $\lambda(\cdot)$, is a constant $\lambda$ and $\mu^0(\cdot)$ is a constant $\mu^0$, we can write the likelihood function as:

$$L(y(\alpha, \beta, \lambda, \mu^0)) = (2\pi)^{-n/2} |\Sigma^n + \lambda I_n|^{-1/2} \exp\left(-\frac{1}{2} (\hat{y} - \mu^0 1)^T (\Sigma^n + \lambda I_n)^{-1} (\hat{y} - \mu^0 1) \right),$$

where $1$ is a $n \times 1$ column vector of ones and $\hat{y} = \left[ \hat{y}^1 \ldots \hat{y}^n \right]^T$. Note that in this case we are estimating $p + 3$ parameters using $n$ observations. We can write the
The log-likelihood function is given by:

\[
\ell_{\hat{y}}(\alpha, \beta, \lambda, \mu^0) = -\frac{n}{2} \ln(2\pi) - \frac{1}{2} \ln(|\Sigma^0 + \lambda I_n|) - \frac{1}{2} (\hat{y} - \mu^0 \mathbf{1})^T (\Sigma^0 + \lambda I_n)^{-1} (\hat{y} - \mu^0 \mathbf{1}).
\]

(2.16)

We can approximately maximize the likelihood over the parameters by using \texttt{patternsearch()} in Matlab started at multiple points chosen by a Latin hypercube sampling (LHS) design using the command \texttt{lhsdesign()}. Also, in the above log-likelihood we can easily solve for \(\mu^0\) in terms of \(\alpha, \beta, \) and \(\lambda\), giving us the estimate

\[
\hat{\mu}^0 = \frac{\hat{y}^T (\Sigma^0 + \lambda I_n)^{-1} \mathbf{1}}{1^T (\Sigma^0 + \lambda I_n)^{-1} 1}.
\]

Also, to prevent numerical issues, if \(|\Sigma^0 + \lambda I_n|\) is very small in (2.16), a useful equivalent expression to \(\ln(|\Sigma^0 + \lambda I_n|)\) is \(\text{trace} \left( \text{logm} (\Sigma^0 + \lambda I_n) \right)\) where \text{logm} is the matrix logarithm.

**Maximum a Posteriori Estimation**

If we have a prior distribution on the parameters it may be more appropriate to use the maximum a posteriori (MAP) estimates for the parameters which uses the prior knowledge about the parameters. Let \(\pi(\alpha, \beta, \lambda, \mu^0)\) be the prior density on \((\alpha, \beta, \lambda, \mu^0)\) and let \(\pi(\alpha, \beta, \lambda, \mu^0|\hat{y})\) be the posterior density on \((\alpha, \beta, \lambda, \mu^0)\) after observing \(\hat{y}\). We find the MAP estimates of the parameters by using \texttt{patternsearch()} in Matlab started multiple times to maximize the posterior distribution of the hyperparameters,

\[
(\hat{\alpha}, \hat{\beta}, \hat{\lambda}, \hat{\mu}^0) = \arg \max_{\alpha, \beta, \lambda, \mu^0} \pi(\alpha, \beta, \lambda, \mu^0|\hat{y})
= \arg \max_{\alpha, \beta, \lambda, \mu^0} \pi(\alpha, \beta, \lambda, \mu^0) L_{\hat{y}}(\alpha, \beta, \lambda, \mu^0).
\]
Robust Parameter Estimation

[81] presents a minimum distance estimator, $L_2E$, which estimates a density by minimizing an estimate of the integrated square error, $\int [f(x|\theta) - f(x|\theta_0)]^2 \, dx$, where $\theta_0$ are the true parameters of the parametric family. This parametric estimator is shown to be inherently robust with no tunable parameters. The $L_2E$ estimates can be obtained by minimizing the estimate of the integrated square error,

$$(\hat{\alpha}, \hat{\beta}, \hat{\lambda}, \hat{\mu}_0) = \arg \min_{\alpha, \beta, \lambda, \mu_0} 2^{-n} \pi^{-n/2} |\Sigma_0 + \lambda I_n|^{-1/2} - 2L_y(\alpha, \beta, \lambda, \mu_0).$$

This can be optimized with the same approach as optimizing the MLE estimates in Section 2.3.3.

2.3.4 The Knowledge-Gradient Policy

The knowledge-gradient policy as described in [36] for discrete $X$ is the policy which chooses the next sampling decision by maximizing the expected incremental value of a measurement. The knowledge gradient at $x$, which gives the expected incremental value of the information gained from a measurement at $x$, is defined as the following scalar field:

$$\nu^{KG,n}(x) \triangleq \mathbb{E}\left[ \max_{u \in X} \mu^{n+1}(u) \middle| \mathcal{F}^n, x^n = x \right] - \max_{u \in X} \mu^n(u). \quad (2.17)$$

The knowledge-gradient policy chooses the sampling decision at time $n$ by maximizing the knowledge gradient,

$$x^n \in \arg\max_{x \in X} \nu^{KG,n}(x). \quad (2.18)$$

By construction, the knowledge gradient policy is optimal for maximizing the maximum of the predictor of the GP if only one decision is remaining. [36] shows that in the case of a finite set of decisions, the knowledge gradient policy samples
every decision infinitely often as the number of sampling decisions goes to infinity; in other words, the knowledge gradient policy finds the best decision in the limit. In addition, [36] shows that the knowledge gradient policy is consistently competitive with or outperforms sequential kriging optimization (SKO) on several test functions.

The knowledge gradient can be explicitly computed when the feasible set of decisions, \( \mathcal{X} \), is finite (see [36]). In the case where \( \mathcal{X} \) is continuous, if \( p \) is small and \( \mathcal{X} \) is bounded, then \( \mathcal{X} \) can be discretized, allowing for the use of the technique in [36] for discrete decisions. However, the complexity of the calculation of this approximation of the knowledge gradient grows exponentially with the number of feasible decisions, \(|x|\), because we must use a dense \(|x| \times |x|\) covariance matrix in our calculation.

2.4 The Knowledge Gradient for Continuous Parameters

In this section we propose an approximation of the knowledge gradient that can be calculated and optimized when our feasible set of decisions is continuous. The approximation we propose can be calculated at a particular decision, \( x \), along with its gradient at \( x \), allowing us to use classical gradient-based search algorithms for maximizing the approximation. This strategy avoids the need to discretize the measurement space \( \mathcal{X} \) into a large number of points to be evaluated. Furthermore, it scales to multidimensional parameter spaces which would be impossible to discretize.

We form the knowledge gradient for continuous parameters (KGCP) by replacing the maximum over \( \mathcal{X} \subset \mathbb{R}^p \) with the maximum over \( x^0, \ldots, x^n \), the first \( n \) sampling decisions and the current sampling decision,

\[
\hat{\mu}^{KG,n}(x) \triangleq \mathbb{E} \left[ \max_{i=0,\ldots,n} \mu^{n+1}(x^i) \bigg| \mathcal{F}^n, x^n = x \right] - \max_{i=0,\ldots,n} \mu^n(x^i)|_{x^n=x}.
\]  

(2.19)
We define the knowledge gradient for continuous parameters policy, $\pi_{KGCP}$, as the policy which selects the next sampling decision by maximizing the knowledge gradient for continuous parameters,

$$x^n \in \arg\max_{x \in X} D_{KG,n}^C(x). \quad (2.20)$$

This approximation should improve as $n$ increases and the maximization is taken over more terms. The first remark is that the knowledge gradient for continuous parameters is nonnegative. The proof follows from Jensen’s inequality,

$$\mathbb{E} \left[ \max_{i=0,\ldots,n} \mu^{n+1}(x^i) \bigg| \mathcal{F}^n, x^n = x \right]$$

$$= \mathbb{E} \left[ \max_{i=0,\ldots,n} \mu^{n+1}(x^i) + \bar{\sigma}_i(\Sigma^n, x^n)Z^{n+1} \bigg| \mathcal{F}^n, x^n = x \right] \quad (2.21)$$

$$\geq \max_{i=0,\ldots,n} \mu^n(x^i)\big|_{x^n=x} + \bar{\sigma}_i(\Sigma^n, x^n)\mathbb{E} \left[ Z^{n+1} \bigg| \mathcal{F}^n, x^n = x \right] \quad (2.22)$$

$$= \max_{i=0,\ldots,n} \mu^n(x^i)\big|_{x^n=x}. \quad (2.23)$$

In (2.21) we substituted in the recursive update for $\mu^{n+1}(x^i)$ given in (2.13). $\bar{\sigma}_i(\Sigma, x)$ is the $i^{th}$ element of $\bar{\sigma}(\Sigma, x)$ which is defined in (2.14). In (2.22) we use Jensen’s inequality with the convex function $\phi(z) = \max_{i=0,\ldots,n} \mu^n(x^i) + \bar{\sigma}_i(\Sigma^n, x^n)z$ where $\mu^n(x^i)$ and $\bar{\sigma}_i(\Sigma^n, x^n)$ are constants since they are measurable with respect to $\mathcal{F}^n$.

Also, comparing the terms that depend on $x$ in the knowledge gradient and the knowledge gradient for continuous parameters, we easily see that

$$\mathbb{E} \left[ \max_{i=0,\ldots,n} \mu^{n+1}(x^i) \big| \mathcal{F}^n, x^n = x \right] \leq \mathbb{E} \left[ \max_{u \in \mathcal{X}} \mu^{n+1}(u) \big| \mathcal{F}^n, x^n = x \right]. \quad (2.23)$$

This fact follows trivially because the maximization in the left term is over a subset of the set maximized over in the right term. Initially, at time $n = 0$, the knowledge gradient for continuous parameters becomes

$$\nu_{KG,0}^C(x) = \mathbb{E}[\mu^1(x^0) | \mathcal{F}^0, x^0 = x] - \mu^0(x^0)|_{x^0=x} = \mu^0(x) - \bar{\mu}^0(x) = 0.$$
This shows the KGCP policy is indifferent about the first sampling decision. At time \( n = 1 \), (2.20) becomes

\[ x^1 \in \arg\max_{x \in X} \left( \mathbb{E}[\max_{i=0,1} \mu^2(x^i) | F^1, x^1 = x] - \max_{i=0,1} \mu^1(x^i) | x^1 = x \right). \]

At this point there is a trade-off between exploring and exploiting in our objective. Implicitly, the algorithm would like to exploit, or sample near a current maximum of \( \mu^n \); this seems likely to increase the maximum of \( \mu^n \). However, the algorithm would also like to explore, i.e. sample far away from any of the previous decisions; these decisions have more uncertainty and are less correlated with the current maximum of \( \mu^n \).

### 2.4.1 Comparison to the Expected Improvement of EGO

Efficient Global Optimization (EGO) is a method developed in [45] to optimize functions when there is no observation noise. For function maximization, EGO uses the expected improvement criterion, \( \mathbb{E}[I^n(x) | F^n] \), where the improvement given the information available at time \( n \) is defined to be the following random variable:

\[ I^n(x) = \max \left( \mu^{n+1}(x) - \max_{i=1,\ldots,n} y^i, 0 \right). \]

In [45], the EGO expected improvement is only defined in the case of no observation noise, where \( \lambda(\cdot) = 0 \). In this case, the knowledge gradient for continuous parameters is less than or equal to the EGO expected improvement criterion. In fact, if the second maximization term in the knowledge gradient for continuous parameters in (3.36) were over \( i = 0, \ldots, n - 1 \), the knowledge gradient for continuous parameters would be equivalent to the expected improvement in the case of no observation noise.
Proposition 2.4.1. In the case of no observation noise, $\tilde{\nu}^{KG,n}(x) \leq \mathbb{E}[I^n(x)|\mathcal{F}^n]$. Furthermore, $\mathbb{E}[I^n(x)|\mathcal{F}^n] = \mathbb{E}[\max_{i=0,...,n} \mu^{n+1}(x^i)|\mathcal{F}^n, x^n = x] - \max_{i=0,...,n-1} \mu^n(x^i)$.

Proof.

$$
\tilde{\nu}^{KG,n}(x)
= \mathbb{E}\left[\max_{i=0,...,n} \mu^{n+1}(x^i)|\mathcal{F}^n, x^n = x\right] - \max_{i=0,...,n} \mu^n(x^i)|_{x^n = x}
\leq \mathbb{E}\left[\max_{i=0,...,n} \mu^{n+1}(x^i)|\mathcal{F}^n, x^n = x\right] - \max_{i=0,...,n-1} \mu^n(x^i)
= \mathbb{E}\left[\max\left(\mu^{n+1}(x^n), \max_{i=0,...,n-1} \mu^n(x^i)\right)|\mathcal{F}^n, x^n = x\right] - \max_{i=0,...,n-1} \mu^n(x^i)
= \mathbb{E}\left[\max\left(\mu^{n+1}(x^n), \max_{i=1,...,n} \hat{y}^i\right)|\mathcal{F}^n, x^n = x\right] - \max_{i=1,...,n} \hat{y}^i
= \mathbb{E}\left[\max\left(\mu^{n+1}(x^n) - \max_{i=1,...,n} \hat{y}^i, 0\right)|\mathcal{F}^n, x^n = x\right]
= \mathbb{E}[I^n(x)|\mathcal{F}^n].
$$

(2.24)

In (2.24) we used the fact that, conditioned on $\mathcal{F}^n$, $\hat{y}^{i+1} = \mu^n(x^i) = \mu^{n+1}(x^i)$ for $i = 0,..., n - 1$ since there is no observation noise.

The EGO algorithm maximizes the expected improvement given in (2.25) at each iteration which is similar to maximizing the knowledge gradient for continuous parameters at each iteration when there is no observation noise.

2.4.2 Calculation of the Knowledge Gradient for Continuous Parameters

We will first show how to calculate the knowledge gradient for continuous parameters, and then derive the gradient of this continuous function that can be used in a steepest ascent algorithm. The knowledge gradient for continuous parameters in (3.36) can be efficiently calculated at a particular $x \in \mathcal{X}$ by using the two algorithms in [36], which we will now summarize. We define the pairs $(a_i, b_i)$ for $i = 0,..., n$ as the sorted pairs $(\mu^n(x^i), \tilde{\sigma}_i(\Sigma^n, x^n))$ conditioned on $\mathcal{F}^n$ and $x^n = x$ for $i = 0,..., n$. The pairs $(a_i, b_i)$
are sorted such that $b_i \leq b_{i+1}$ for $i = 0, ..., n - 1$. If there exists some $i \neq j$ such that $b_i = b_j$ and $a_i \leq a_j$, then the pair $(a_j, b_j)$ dominates $(a_i, b_i)$, and the pair $(a_i, b_i)$ is added to a list of initially dominated lines. The $a_i$’s are the intercepts and the $b_i$’s are the slopes of the lines in Figure 2.1(a). Furthermore we define $A^0$ as the index map such that $(a_i, b_i) = (\mu^n(x^{A^0}), \tilde{\sigma}_{A^0}(\Sigma^n, x^n))$. For a fixed $x^n = x$, $a_i$ and $b_i$ are $\mathcal{F}^n$ measurable and hence constants. We now simplify the first term in the knowledge gradient for continuous parameters,

$$
\mathbb{E} \left[ \max_{i=0, \ldots, n} \mu^{n+1}(x^i) \right| \mathcal{F}^n, x^n = x
$$

$$
= \mathbb{E} \left[ \max_{i=0, \ldots, n} \mu^n(x^i) + \tilde{\sigma}_i(\Sigma^n, x^n)Z^{n+1} \right| \mathcal{F}^n, x^n = x
$$

$$
(2.26)
$$

$$
= \mathbb{E} \left[ \max_{i=0, \ldots, n} a_i + b_iZ \right].
$$

(2.27)

In (2.26) we used the recursive update for $\mu^n(x^i)$ given in (2.13). We next summarize the two algorithms in [36] which show how to efficiently calculate the term in (2.27).

Algorithm 1 is a scan-line algorithm that replaces the maximization in (2.27) with a piecewise linear function using indicator functions. In Algorithm 1, $A^1$ is called the accept set and is a vector of indices which keeps track of all the $i$’s such that line $a_i + b_i z$ is part of the epigraph shown in Figure 2.1(a). We keep track of the values of $z$ where the lines intersect in a vector $c$. $c_{i+1}$ is the largest value of $z$ such that line $a_i + b_i z$ is part of the epigraph shown in Figure 2.1(a). In terms of the lines in the accept set $A^1$, $c_{1+A^1_i}$ is the intersection of $a_{A^1_i} + b_{A^1_i} z$ and $a_{A^1_{i+1}} + b_{A^1_{i+1}} z$. Solving for the $z$ such that these lines intersect we get $c_{1+A^1_i} = (a_{A^1_i} - a_{A^1_{i+1}})/(b_{A^1_{i+1}} - b_{A^1_i})$ for $i = 1, ..., \tilde{n}$, where $\tilde{n}$ is the length of $A^1$ minus one. Also we set $c_0 = -\infty$ and $c_{n+1} = +\infty$. For convenience, we define $\tilde{a}_i = a_{(A^1_i)}$, $\tilde{b}_i = b_{(A^1_i)}$, $\tilde{c}_{i+1} = c_{(1+A^1_i)}$, and $\tilde{c}_0 = -\infty$ for $i = 0, ..., \tilde{n}$. Algorithm 1 efficiently calculates constants $\tilde{c}_0, ..., \tilde{c}_{\tilde{n}+1}$ and the vector of indices, $A^1$, so that a function of the form $f(z) = \max_{i=0, \ldots, n} a_i + b_i z$
\begin{equation}
\begin{aligned}
(01) \quad c_0 &= -\infty, \ c_{n+1} = +\infty, \ A^1 = [0] \\
(02) \quad \textbf{for} \ i = 1 : n \\
(03) \quad \textbf{if} \ (a_i, b_i) \text{ not initially dominated} \\
(04) \quad \text{loopdone} = \text{false} \\
(05) \quad \textbf{while} \ \text{loopdone} == \text{false} \\
(06) \quad j = A^1(\text{end}) \\
(07) \quad c_{j+1} = (a_j - a_i)/(b_i - b_j) \\
(08) \quad \textbf{if} \ \text{length}(A^1) \neq 1 \ &\& c_{j+1} \leq c_{k+1} \text{ where } k = A^1(\text{end} - 1) \\
(09) \quad \text{Delete last element in } A^1. \\
(10) \quad \textbf{else} \ \text{add } i \text{ to the end of } A^1. \\
(11) \quad \text{loopdone} = \text{true} \\
(12) \quad \textbf{end} \\
(13) \quad \textbf{end} \\
(14) \quad \textbf{end} \\
(15) \quad \textbf{end}
\end{aligned}
\end{equation}

Table 2.1: Summary of Algorithm 1 from [36].

can be rewritten as $f(z) = \sum_{i=0}^{n} (a_{A^1_i} + b_{A^1_i} z) 1_{[\tilde{c}_i, \tilde{c}_{i+1})}(z)$. The algorithm is outlined in Figure 2.1 using the convention that the first index of a vector is zero.

(a) A visualization of Algorithm 1.  
(b) The output of Algorithm 1 with new indices.

Figure 2.1: Algorithm 1 is a scan line algorithm to re-express $f(z) = \max_{i=0,\ldots,n} a_i + b_i z$ as $f(z) = \sum_{i=0}^{n} (\tilde{a}_i + \tilde{b}_i z) 1_{[\tilde{c}_i, \tilde{c}_{i+1})}(z)$.  

23
Next, Algorithm 2 from [36] shows how to simplify the expectation in (2.28) to (2.29), which is something we can easily compute.

\[
\mathbb{E} \left[ \max_{i=0, \ldots, n} a_i + b_i Z \right] = \mathbb{E} \left[ \sum_{i=0}^{\tilde{n}} \left( a_{A_1^i} + b_{A_1^i} Z \right) 1_{[\tilde{c}_i, \tilde{c}_{i+1})}(Z) \right] = \sum_{i=0}^{\tilde{n}} \left[ a_{A_1^i} \mathbb{P} \left[ Z \in [\tilde{c}_i, \tilde{c}_{i+1}) \right] + b_{A_1^i} \mathbb{E} \left[ Z 1_{[\tilde{c}_i, \tilde{c}_{i+1})}(Z) \right] \right] = \sum_{i=0}^{\tilde{n}} \left[ a_{A_1^i} \left( \Phi(\tilde{c}_{i+1}) - \Phi(\tilde{c}_i) \right) + b_{A_1^i} \left( \phi(\tilde{c}_i) - \phi(\tilde{c}_{i+1}) \right) \right]. \tag{2.29}
\]

In (2.29), \(\phi(\cdot)\) and \(\Phi(\cdot)\) are the pdf and cdf of a standard normal random variable, respectively.

### 2.4.3 The Gradient of the Knowledge Gradient for Continuous Parameters

Next, we show how to calculate the gradient of the knowledge gradient for continuous parameters, \(\nabla_x \tilde{\nu}^{KG,n}(x)\), at a fixed \(x \in \mathcal{X}\). This will allow us to use gradient ascent to maximize the knowledge gradient for continuous parameters. Let \(A = A^0[A^1]\), meaning \(A_i = A^0[A^1]\); \(A\) is now a reordered index set. For example, if \(A^0 = [2, 1, 0]\) and \(A^1 = [0, 2, 1]\), then \(A = [2, 0, 1]\). \(A\) contains the indices \(i\) such that \((\mu^n(x^{A_0^i}) + \tilde{\sigma}_{A_0^i}(\tilde{\Sigma}^n, x^n))z\) is part of the epigraph of Figure 2.1(b) for some value of \(z\).

**Proposition 2.4.2.** The gradient of the first term in (3.36) is given by

\[
\nabla_x \mathbb{E} \left[ \max_{i=0, \ldots, n} \mu^{n+1}(x^i) \mid \mathcal{F}^n, x^n = x \right] = \sum_{i=0}^{\tilde{n}} \left[ (\nabla_x \mu^n(x^{A_1^i})) \left( \Phi(\tilde{c}_{i+1}) - \Phi(\tilde{c}_i) \right) + (\nabla_x \tilde{\sigma}_{A_1^i}(\tilde{\Sigma}^n, x^n)) \left( \phi(\tilde{c}_i) - \phi(\tilde{c}_{i+1}) \right) \right] + \sum_{i=0}^{\tilde{n}} \left[ \left( \mu^n(x^{A_1^i}) + \tilde{\sigma}_{A_1^i}(\tilde{\Sigma}^n, x^n) \tilde{c}_{i+1} \right) \nabla_x \tilde{c}_{i+1} - \left( \mu^n(x^{A_1^i}) + \tilde{\sigma}_{A_1^i}(\tilde{\Sigma}^n, x^n) \tilde{c}_i \right) \nabla_x \tilde{c}_i \right].
\]
Proof.

\[
\nabla_x \mathbb{E} \left[ \max_{i=0, \ldots, n} \mu^{n+1}(x^i) \right] \mathbb{F}^n, x^n = x
\]

\[
= \nabla_x \sum_{i=0}^{\tilde{n}} \left[ \mu^n(x^{A_i}) (\Phi(\tilde{c}_{i+1}) - \Phi(\tilde{c}_i)) + \tilde{\sigma}_A(\bar{\Sigma}^n, x^n)(\phi(\tilde{c}_i) - \phi(\tilde{c}_{i+1})) \right]
\]

\[
= \sum_{i=0}^{\tilde{n}} \left[ (\nabla_x \mu^n(x^{A_i})) (\Phi(\tilde{c}_{i+1}) - \Phi(\tilde{c}_i)) + (\nabla_x \tilde{\sigma}_A(\bar{\Sigma}^n, x^n))(\phi(\tilde{c}_i) - \phi(\tilde{c}_{i+1})) \right]
\]

\[
+ \sum_{i=0}^{\tilde{n}} \left[ \mu^n(x^{A_i}) \nabla_x \phi(\tilde{c}_{i+1}) - \Phi(\tilde{c}_i) + \tilde{\sigma}_A(\bar{\Sigma}^n, x^n) \nabla_x (\phi(\tilde{c}_i) - \phi(\tilde{c}_{i+1})) \right]
\]

\[
= \sum_{i=0}^{\tilde{n}} \left[ (\nabla_x \mu^n(x^{A_i})) (\Phi(\tilde{c}_{i+1}) - \Phi(\tilde{c}_i)) + (\nabla_x \tilde{\sigma}_A(\bar{\Sigma}^n, x^n))(\phi(\tilde{c}_i) - \phi(\tilde{c}_{i+1})) \right]
\]

\[
+ \sum_{i=0}^{\tilde{n}} \left[ (\mu^n(x^{A_i}) + \tilde{\sigma}_A(\bar{\Sigma}^n, x^n)\tilde{c}_{i+1}) \phi(\tilde{c}_{i+1}) \nabla_x \tilde{c}_{i+1} - (\mu^n(x^{A_i}) + \tilde{\sigma}_A(\bar{\Sigma}^n, x^n)\tilde{c}_i) \phi(\tilde{c}_i) \nabla_x \tilde{c}_i \right].
\]

Equation (2.30) is just the gradient of (2.29). In (2.31) we used the product rule because \( c_0, \ldots, c_{n+1} \) all depend on \( x^n \). In the last line we use the fact that \( \frac{\partial}{\partial x} \Phi(f(x)) = \phi(f(x)) \frac{\partial}{\partial x} f(x) \) and \( \frac{\partial}{\partial x} \phi(f(x)) = -\phi(f(x)) f(x) \frac{\partial}{\partial x} f(x) \) to differentiate the second term. The first term in the final equation is analogous to (2.29) with the scalars \( \mu^n(x^i) \) and \( \tilde{\sigma}_i(\bar{\Sigma}^n, x^n) \) replaced with the vectors \( \nabla_x \mu^n(x^i) \) and \( \nabla_x \tilde{\sigma}_i(\bar{\Sigma}^n, x^n) \).

The calculation of \( \nabla_x \tilde{c}_i \) for \( i = 0, \ldots, \tilde{n} + 1 \) is relatively straightforward. An equivalent equation for the \( \tilde{c}_i \)'s which are output from Algorithm 1 is \( \tilde{c}_i = \frac{\tilde{a}_{i-1} - \tilde{a}_i}{b_i - b_{i-1}} \) for \( i = 1, \ldots, \tilde{n} \) with \( \tilde{c}_0 = -\infty \) and \( \tilde{c}_{\tilde{n}+1} = +\infty \). Then using the quotient rule we can calculate the following:

\[
\nabla_x \tilde{c}_i = \begin{cases} 
\frac{(\tilde{b}_i - \tilde{b}_{i-1})(\nabla \tilde{a}_{i-1} - \nabla \tilde{a}_i) - (\tilde{a}_{i-1} - \tilde{a}_i)(\nabla \tilde{b}_i - \nabla \tilde{b}_{i-1})}{(b_i - b_{i-1})^2}, & \text{for } i = 1, \ldots, \tilde{n} \\
0, & \text{for } i = 0, \tilde{n} + 1.
\end{cases}
\]

As long as we can calculate \( \nabla_x \mu^n(x^i) \) and \( \nabla_x \tilde{\sigma}_i(\bar{\Sigma}^n, x^n) \) for \( i = 0, \ldots, n \), we can calculate the expression in Proposition 2.4.2 and the gradient of the knowledge.
gradient for continuous parameters. The equations for these values are expressed in the next two lemmas.

**Lemma 2.4.1.**

\[
\nabla_{x^n} \mu^n(x^i) = \begin{cases} 
\vec{0}, & \text{if } i < n \\
\nabla_{x^n} \mu^0(x^n) + J^n[S^n]^{-1}\tilde{y}^n, & \text{if } i = n,
\end{cases}
\]

where we let \(J^n\) be the following matrix of first-order partial derivatives,

\[
J^n = \begin{bmatrix} 
\nabla_{x^n} \Sigma^0(x^0, x^n) & \cdots & \nabla_{x^n} \Sigma^0(x^{n-1}, x^n)
\end{bmatrix}
\]

(2.33)

\[
= 2 \begin{bmatrix} 
\alpha_1(x_1^0 - x_1^n)\Sigma^0(x^0, x^n) & \cdots & \alpha_1(x_1^{n-1} - x_1^n)\Sigma^0(x^{n-1}, x^n) \\
\vdots & \ddots & \vdots \\
\alpha_p(x_p^0 - x_p^n)\Sigma^0(x^0, x^n) & \cdots & \alpha_p(x_p^{n-1} - x_p^n)\Sigma^0(x^{n-1}, x^n)
\end{bmatrix}.
\]

(2.34)

**Proof:** Given in Appendix A.1

**Lemma 2.4.2.**

\[
\nabla_{x^n} \tilde{\sigma}_i(\Sigma^n, x^n) = \frac{B\nabla_{x^n} e^T_i \tilde{\Sigma}^n e_{x^n} - e^T_i \tilde{\Sigma}^n e_{x^n} \nabla_{x^n} B}{B^2},
\]

where \(B \equiv \sqrt{\lambda(x^n) + e^T_{x^n} \Sigma^n e_{x^n}}\) and

\[
\nabla_{x^n} e^T_i \tilde{\Sigma}^n e_{x^n} = \begin{cases} 
2\text{DIAG}(\alpha)(x^i - x^n)\Sigma^0(x^i, x^n) - J^n[S^n]^{-1}\Sigma^0 e_{x^i}, & \text{if } i < n \\
2J^n[S^n]^{-1} & \text{if } i = n
\end{cases}
\]

(26)
and

\[
\nabla x_n B = \frac{1}{2} (\lambda(x^n) + \Sigma^n(x^n, x^n))^{-\frac{1}{2}} \left( \nabla x_n \lambda(x^n) - 2J^n[S^n]^{-1} \begin{bmatrix} \Sigma^0(x^0, x^n) \\ \vdots \\ \Sigma^0(x^{n-1}, x^n) \end{bmatrix} \right).
\]

Proof: Given in Appendix A.2

### 2.4.4 Maximizing the Knowledge Gradient for Continuous Parameters

We begin by giving an illustrative example of the knowledge gradient for continuous parameters on a one-dimensional Gaussian process with normally distributed observation noise with a variance of 0.1. Figure 2.2(a) shows the results of the estimate of the function after four observations along with the actual observations. Figure 2.2(b) shows both the knowledge gradient for continuous parameters and the exact knowledge gradient over a finely discretized set of decisions. The knowledge gradient is larger at decisions with more uncertainty as well as points where the estimate of the function is larger. We can see that the knowledge gradient is nonconcave and seems to have local minima near previously sampled points. Furthermore, many of the local maxima appear to be approximately halfway between previously sampled points.

In Figure 2.2(c) and 2.2(d) we show the estimate of the function and knowledge gradient after nine observations. Again the knowledge gradient is not concave but many of the local maxima appear to be approximately halfway between previously sampled points. In higher dimensions, a gradient ascent algorithm started multiple times is appropriate for approximately maximizing a nonconcave continuous function.

We now have an objective that can be quickly evaluated along with its gradient at any decision $x$. We propose using a multi-start gradient ascent algorithm with con-
Figure 2.2: (a) The estimate of the function along with the 95% confidence intervals of the estimate after 4 observations. (b) The knowledge gradient for continuous parameters (KGCP) and exact knowledge gradient over a finely discretized set of decisions (KGCB) after 4 observations. (c) The estimate of the function after 9 observations. (d) The knowledge gradient after 9 observations.

Heuristically, as suggested above, there is likely to be a local maximum roughly halfway between two previously sampled points. Furthermore, we have a good guess at a starting step size that will keep our algorithm looking in the region between these two previously sampled points based on the distance between the two points. We can calculate all the midpoints between the set of sampled points and use them as starting points of our gradient ascent with a fixed step size chosen such that the magnitude of the first step is one fourth of the Euclidean distance between the two corresponding previously sampled points. We also choose to start the gradient ascent algorithm at the previously sampled decisions. These points are
likely to be very close to a local minimum and are thus reasonable starting locations for a gradient ascent algorithm, although a reasonable starting step size is more ambiguous. We can then take the maximum over all of the restarts to approximately get the overall maximum of the knowledge gradient for continuous parameters. We perform \( \binom{n}{2} + n \) restarts which may become computationally expensive at \( n \) grows large. Alternatively we could maximize KGCP over a set of candidate points chosen by an LHS design or use a genetic algorithm (see [35]). It is worth noting that it is not critical to get the exact maximum of the knowledge gradient for continuous parameters in order to determine the next sampling decision. There are likely several distinct points that are worth sampling and it may be acceptable if on one iteration the algorithm chooses a point which does not exactly maximize the knowledge gradient for continuous parameters.

### 2.4.5 The KGCP Policy

We now give an outline of the KGCP policy in Table 2.2. In line 2 we choose the sampling decision by maximizing the knowledge gradient for continuous parameters defined in (3.36). This maximization should be approximated by using the algorithm in Section 2.4.4. Also, the maximization in line 6 to find the implementation decision cannot be explicitly solved either. We approximate the solution using a multistart gradient ascent algorithm with the same starting points used in Section 2.4.4. The gradient of \( \mu^N(x) \) can be evaluated using Lemma 2.4.1. If no prior knowledge about
the parameters is available, an initial phase of sampling decisions chosen following a Latin hypercube design can be run before starting the KGCP policy as suggested in a similar context in [45].

In general we will not be given the parameters of the covariance function, $\alpha$ and $\beta$, the variance of observation noise, $\lambda()$, or the mean of the initial prior distribution on $\mu$, $\mu^0()$. If these parameters are not known, a step should be added before line 2 for estimating the covariance function parameters using MLE, maximum a posterior estimation (see [73]), or robust parameter estimation (see [81]). For example, we can approximately maximize the likelihood over the parameters by using patternsearch() in Matlab started at multiple points chosen by a Latin hypercube sampling (LHS) design using the command lhsdesign().

2.5 Convergence

In this section we show that, although the KGCP can be regarded as a near-sighted objective for finding the maximum of $\mu(x)$, the KGCP policy searches enough so that uncertainty of the regression function converges to zero almost surely for each decision as the number of sampling decisions and observations increases to infinity. Note that additional conditions would need to be specified before making the claim about the consistency of the posterior and finding the maximum of $\mu(x)$ almost surely in the limit. The proof is based on the fact that the knowledge gradient for continuous parameters of each decision converges to zero as the number of iterations of the algorithm goes to infinity. We then show that this implies that the conditional variance of $\mu$ at every observation converges to zero; in other words, we become certain of $\mu$ at every point. We define $\text{Var}^n[\cdot]$, $\text{Cov}^n[\cdot]$, and $\text{Corr}^n[\cdot]$ as $\text{Variance}[\cdot|\mathcal{F}^n]$, $\text{Covariance}[\cdot|\mathcal{F}^n]$, and $\text{Correlation}[\cdot|\mathcal{F}^n]$, respectively. For simplicity in this section
we assume the variance of the observation noise is a constant. Our presentation will need the following assumptions:

**Assumption 2.5.1.** \( \lambda(x) = \lambda > 0, \mu^0(x) = \mu^0 \), and the estimates of \( \alpha, \beta, \lambda, \) and \( \mu^0 \) are fixed.

**Assumption 2.5.2.** \( \limsup_{n \to \infty} |\mu^n(x) - \mu^n(u)| \) is bounded for every \( x, u \in X \) almost surely.

**Assumption 2.5.3.** For any \( x \neq u \), \( \exists c \) s.t. \( \limsup_{n \to \infty} |\text{Corr}^n[\mu(x), \mu(u)]| \leq c < 1 \) almost surely.

**Assumption 2.5.4.** We can exactly maximize the KGCP; \( x^n \in \argmax_{x \in X} \nu_{KG,n}(x) \).

**Proposition 2.5.1.** For every sample path, the knowledge gradient for continuous parameters of a decision \( x, \nu_{KG,n}(x) \), converges to zero if the conditional variance of \( \mu(x) \) converges to zero.

*Proof.* We first need an upper bound on the knowledge gradient for continuous parameters. We show in Appendix A.3 that

\[
\nu_{KG,n}(x) \leq \sqrt{2\beta \text{Var}^n[\mu(x)] / \pi \lambda}. \tag{2.35}
\]

Combining the fact that the knowledge gradient for continuous parameters is nonnegative and that the upper bound of the knowledge gradient for continuous parameters in (2.35) decreases to zero as \( \text{Var}^n[\mu(x)] \to 0 \), we obtain the desired result.

The next proposition provides a way to put an upper bound on the conditional variance of \( \mu \) near an accumulation point, \( x^{acc} \), of the sampling decisions. Figure 2.3 has a diagram of the points being considered. \( x^{acc} \) is an accumulation point of the sampling decisions. \( x^d \) is an arbitrary fixed point in an open ball centered around \( x^{acc} \) with radius \( \epsilon \); we are interested in \( \text{Var}[\mu(x^d)] \). \( x^{mult} \) is a point we consider measuring.
multiple times. \( x^{\text{near}} \) is a point which is closer to \( x^d \) than \( x^{\text{mult}} \) is close to \( x^d \) in terms of the initial covariance; formally, \( \Sigma^0(x^{\text{mult}}, x^d) \leq \Sigma^0(x^{\text{near}}, x^d) \). We denote an open ball centered at \( a \) with radius \( \epsilon \) as \( B(a, \epsilon) = \{ x | d(x, a) < \epsilon \} \).

Figure 2.3: A diagram of the points: \( x^{\text{acc}} \) is an accumulation point; \( x^{\text{mult}} \) is a point being measured multiple times; \( x^{\text{near}} \) is a point near to \( x^d \) we are considering to measure; \( x^d \) is an arbitrary fixed point in the open ball centered at \( x^{\text{acc}} \).

**Proposition 2.5.2.** Fix \( \epsilon > 0 \) and consider an arbitrary point \( x^d \in B(x^{\text{acc}}, \epsilon) \), where \( B(x^{\text{acc}}, \epsilon) \) is an open ball centered at \( x^{\text{acc}} \) with radius \( \epsilon \). If we have measured \( n \) points in the ball \( B(x^{\text{acc}}, \epsilon) \), an upper bound on the conditional variance of \( \mu(x^d) \) can be constructed by hypothetically measuring one particular point \( x^{\text{mult}} \) \( n \) times, where \( x^{\text{mult}} \) satisfies \( \Sigma^0(x^{\text{mult}}, x^d) \leq \Sigma^0(x, x^d), \forall x \in B(x^{\text{acc}}, \epsilon) \). Furthermore the upper bound on the conditional variance of \( \mu(x^d) \) is \( \beta - (\Sigma^0(x^{\text{mult}}, x^d))^2 \frac{n}{\beta n + \lambda} \) for every sample path.

**Proof.** Sketch of proof (See Appendix A.4 for full proof): We wish to find an upper bound on the conditional variance of \( \mu(x^d) \) which will converge to zero as \( n \to \infty \) and \( \epsilon \to 0 \). The ordering of the decision-observation pairs can be changed without altering the conditional variance of \( \mu(x^d) \), and the conditional variance of \( \mu(x^d) \) is a decreasing sequence. Therefore, after we have measured \( n \) points in \( B(x^{\text{acc}}, \epsilon) \), \( \max_{x_0, \ldots, x_{n-1} \in B(x^{\text{acc}}, \epsilon)} \text{Var}^n[\mu(x^d)] \) is an upper bound on the conditional variance of \( \mu(x^d) \); we have ignored the decisions outside of \( B(x^{\text{acc}}, \epsilon) \) because they would only lower the conditional variance more. We define the policy \( \pi^{\text{mult}} \) which sets \( x^0 = \...
\[ \ldots = x^{n-1} = x^{\text{mult}}. \] We can derive that under the policy \( \pi^{\text{mult}} \), \( \text{Var}^n[\mu(x)] = \beta - (\Sigma^0(x^{\text{mult}}, x))^2 \frac{n}{\beta n + \lambda}. \)

First consider the change \( \text{Var}^n[\mu(x^d)] - \text{Var}^{n+1}[\mu(x^d)] \) under \( \pi^{\text{mult}} \) if we have measured \( x^{\text{mult}} \) \( n \) times and then measure \( x^{\text{mult}} \) one more time. We define \( \beta_0 = \Sigma^0(x^{\text{mult}}, x^d) \). The decrease in the conditional variance of \( \mu(x^d) \) from measuring \( x^{\text{mult}} \) once more is

\[
\text{Var}^n[\mu(x^d)] - \text{Var}^{n+1}[\mu(x^d)] = \frac{\beta_0^2 \lambda}{((n+1)\beta + \lambda)(n\beta + \lambda)}, \tag{2.36}
\]

Second we consider measuring the change in \( \text{Var}^n[\mu(x^d)] - \text{Var}^{n+1}[\mu(x^d)] \) if we have measured \( x^{\text{mult}} \) \( n \) times and then measure \( x^{\text{near}} \) one time where \( x^{\text{near}} \) satisfies \( \Sigma^0(x^{\text{mult}}, x^d) \leq \Sigma^0(x^{\text{near}}, x^d) \). \( x^{\text{near}} \) can be thought of as a point close to \( x^d \) because \( \mu(x^{\text{near}}) \) has a higher initial covariance with \( \mu(x^d) \) than \( \mu(x^{\text{mult}}) \) does. We define \( \beta_1 = \Sigma^0(x^{\text{mult}}, x^{\text{near}}) \) and \( \beta_2 = \Sigma^0(x^{\text{near}}, x^d) \). Note that \( \beta_0 \leq \beta_2 \) and \( 0 < \beta_0, \beta_1, \beta_2 \leq \beta \); Figure 2.3 visually shows the relationships between the points. The decrease in the conditional variance of \( \mu(x^d) \) from measuring \( x^{\text{near}} \) is

\[
\text{Var}^n[\mu(x^d)] - \text{Var}^{n+1}[\mu(x^d)] = \left( \beta_2 - \frac{n\beta_0\beta_1}{n\beta + \lambda} \right)^2 \left( \beta - \frac{n\beta_0^2}{n\beta + \lambda} + \lambda \right)^{-1}. \tag{2.37}
\]

We want to show that if we have measured \( x^{\text{mult}} \) \( n \) times (and measured nothing else) that the amount we can lower the conditional variance of \( \mu(x^d) \) by observing \( x^{\text{mult}} \) again given in (2.36) is smaller than the amount given in (2.37) if we observe a new point \( x^{\text{near}} \). We verify this is true algebraically in Appendix A.4. We have shown that, for any \( n \geq 0 \), if we have sampled the decisions \( x^0, \ldots, x^{n-1} = x^{\text{mult}} \), then the additional decrease in the conditional variance of \( \mu(x^d) \) would be smallest by setting \( x^n = x^{\text{mult}} \). This is true for \( n = 0, 1, 2, \ldots \), so using an induction argument this proves that \( \max_{x_0, \ldots, x_{n-1} \in B(x^{\text{acc}}, \epsilon)} \text{Var}^n[\mu(x^d)] = \text{Var}^n[\mu(x^d)] \) under \( \pi^{\text{mult}} \), where \( x^{\text{mult}} \) satisfies \( \Sigma^0(x^{\text{mult}}, x^d) \leq \Sigma^0(x, x^d), \forall x \in B(x^{\text{acc}}, \epsilon) \). As explained
above, $\max_{x_0, \ldots, x_{n-1} \in B(x^{acc}, \epsilon)} Var^n[\mu(x^d)]$ is an upper bound on the conditional variance of $\mu(x^d)$ after we have measured $n$ points in $B(x^{acc}, \epsilon)$ (and possibly more points outside $B(x^{acc}, \epsilon)$). Under $\pi_{mult}$, $Var^n[\mu(x^d)] = \beta - (\Sigma^0(x^{mult}, x^d))^2 \frac{n}{\beta n + \lambda}$ which gives us the upper bound.

**Proposition 2.5.3.** Let $x^{acc}$ be an accumulation point of the sequence of sampling decisions $\{x^n\}_{n=0}^\infty$. Consider a point $x^d \in B(x^{acc}, \epsilon)$ using the Euclidean distance. Then $\lim_{n \to \infty} Var^n[\mu(x^d)] \leq \beta - \beta \exp(-8 \sum_{i=1}^p \alpha_i \epsilon^2)$ for every sample path.

**Proof.** We first show that $Var^n[\mu(x^d)]$ converges because it is a decreasing sequence that is bounded below by zero. If we measure $x^n$ at time $n$, the equation for the conditional variance becomes

$$
\Sigma^{n+1}(x^d, x^d) = \Sigma^n(x^d, x^d) - (\Sigma^n(x^d, x^d))^2 (\Sigma^n(x^n, x^n) + \lambda)^{-1}.
$$

The second term in (2.38) is clearly positive and thus $\Sigma^{n+1}(x^d, x^d) \leq \Sigma^n(x^d, x^d)$. Now, $n$ is arbitrary, so we can conclude that $Var^n[\mu(x^d)]$ is a decreasing sequence bounded below by zero. We define $Var^\infty[\mu(x^d)]$ as the limit of $Var^n[\mu(x^d)]$.

$x^{acc}$ is an accumulation point so for all $\epsilon > 0$ there are an infinite number of $n$ with $x^n \in B(x^{acc}, \epsilon)$. We now put an upper bound on $Var^n[\mu(x^d)]$. Under the policy $\pi_{mult}$ of only measuring $x^{mult}$ we can see

$$
\lim_{n \to \infty} Var^{\pi_{mult}, n}[\mu(x)] = \beta - \frac{(\Sigma^0(x^{mult}, x))^2}{\beta}.
$$

Let $\{k_n\}_{n=0}^\infty$ be a subsequence of natural numbers such that the policy $\pi$ chooses $x^{k_n} \in B(x^{acc}, \epsilon) \ \forall n$. Let $x^{mult}$ satisfy $\Sigma^0(x^{mult}, x^d) \leq \Sigma^0(x, x^d), \forall x \in B(x^{acc}, \epsilon)$. Using Proposition 2.5.2, we see that

$$
Var^{\pi, k_n}[\mu(x^d)] \leq Var^{\pi_{mult}, n}[\mu(x^d)] = \beta - (\Sigma^0(x^{mult}, x^d))^2 \frac{n}{\beta n + \lambda_0}.
$$

34
Now, letting \( n \) go to infinity we get

\[
\text{Var}_\infty[\mu(x^d)] = \lim_{n \to \infty} \text{Var}^{\pi,n}[\mu(x^d)] = \lim_{n \to \infty} \text{Var}^{\pi,k_n}[\mu(x^d)] \leq \beta - \frac{(\Sigma^0(x^\text{mult}, x^d))^2}{\beta}. \tag{2.40}
\]

This equation holds for any \( x^\text{mult} \) which satisfies \( \Sigma^0(x^\text{mult}, x^{acc}) \leq \Sigma^0(x, x^{acc}), \forall x \in B(x^{acc}, \epsilon) \) for a fixed \( \epsilon > 0 \). We next take the supremum over all such \( x^\text{mult} \) to obtain

\[
\text{Var}_\infty[\mu(x^d)] \leq \sup_{x \in B(x^{acc}, \epsilon)} \left( \beta - \frac{(\Sigma^0(x, x^d))^2}{\beta} \right)
= \beta - \frac{(\inf_{x \in B(x^{acc}, \epsilon)}(\Sigma^0(x, x^d))^2}{\beta}
\leq \beta - \frac{\inf_{x \in B(x^{acc}, \epsilon)}(\beta e^{-\sum_{i=1}^p \alpha_i (x_i - x^d_i)^2})^2}{\beta}
\leq \beta - \frac{(\beta e^{-\sum_{i=1}^p \alpha_i \epsilon^2})^2}{\beta} = \beta - \beta e^{-8 \sum_{i=1}^p \alpha_i \epsilon^2}. \tag{2.41}
\]

Equation (2.41) uses the fact that \((x_i - x^d_i)^2 \leq 4\epsilon^2\) because \(x, x^d \in B(x^{acc}, \epsilon)\) using the Euclidean distance.

**Corollary 2.5.1.** Since Proposition 2.5.3 was true for an arbitrary \( \epsilon > 0 \) and \( \lim_{\epsilon \to 0} \beta - \beta e^{-8 \sum_{i=1}^p \alpha_i \epsilon^2} = 0 \), we can conclude that \( \lim_{n \to \infty} \text{Var}^n[\mu(x^{acc})] = 0 \).

We now want to show that the knowledge gradient for continuous parameters of the points being sampled as \( n \) goes to infinity gets arbitrarily close to zero.

**Theorem 2.5.1.** Using the KGCP policy, \( \liminf_{n \to \infty} \sup_{x \in X} \hat{\nu}_{\text{KG}}^{n}(x) = 0 \) for every sample path.

**Proof.** Using equation (2.35) from the proof of Proposition 2.5.1 we put an upper bound on the knowledge gradient for continuous parameters at \( x^n \),

\[
\hat{\nu}_{\text{KG}}^{n}(x^n) \leq \frac{2}{\sqrt{2\pi}} \sqrt{\frac{\beta \text{Var}^n[\mu(x^n)]}{\lambda}}. \tag{2.42}
\]

First, the sequence of sampling decisions is a bounded sequence in \( \mathbb{R}^p \) and thus has an accumulation point, \( x^{\text{acc}} \). Also, the sequence \( \{\sup_{x \in X} \hat{\nu}_{\text{KG}}^{n}(x)\}_{n=0}^{\infty} \) is a non-
negative sequence because the knowledge gradient for continuous parameters is non-negative. Let \( \{k_n\}_{n=0}^{\infty} \) be a subsequence of natural numbers such that the KGCP policy chooses \( x^{k_n} \in B(x^{acc}, \epsilon) \) \( \forall n \). Now using Proposition 2.5.3 we write
\[
\lim_{n \to \infty} \text{Var}^n[\mu(x^{k_n})] \leq \beta - \beta e^{-8 \sum_{i=1}^{p} \alpha_i \epsilon^2} .
\]
Combining this with (2.42) we get,
\[
0 \leq \liminf_{n \to \infty} \bar{\nu}^{KG,k_n}(x^{k_n}) \leq \liminf_{n \to \infty} \frac{2}{\sqrt{2\pi}} \sqrt{\frac{\beta \text{Var}^{k_n}[\mu(x^{k_n})]}{\lambda}} \leq \frac{2}{\sqrt{2\pi}} \sqrt{\frac{\beta(\beta - \beta e^{-8 \sum_{i=1}^{p} \alpha_i \epsilon^2})}{\lambda}}.
\]
Since this equation was true for an arbitrary \( \epsilon > 0 \) and
\[
\lim_{\epsilon \to 0} \frac{2}{\sqrt{2\pi}} \sqrt{\frac{\beta(\beta - \beta e^{-8 \sum_{i=1}^{p} \alpha_i \epsilon^2})}{\lambda}} = 0,
\]
we can conclude that \( \liminf_{n \to \infty} \bar{\nu}^{KG,k_n}(x^{k_n}) = 0 \).
This implies that \( \liminf_{n \to \infty} \bar{\nu}^{KG,n}(x^n) = 0 \) as well because the \( \liminf \) of a sequence is less than or equal to the \( \liminf \) of one of its subsequences. Recalling that under the KGCP policy \( \bar{\nu}^{KG,n}(x^n) = \sup_{x \in X} \bar{\nu}^{KG,n}(x) \) by Assumption 2.5.4 and because \( \bar{\nu}^{KG,n}(x) \) is continuous and \( X \) is compact, we arrive at the desired result.

For the following theorems we need Assumption 2.5.2 that prevents the updated mean from approaching infinity or negative infinity. We need Assumption 2.5.3 which ensures the function does not become perfectly correlated at two different decisions; this seems intuitive but is not trivial to prove.

**Theorem 2.5.2.** If Assumptions 2.5.1, 2.5.2, 2.5.3, and 2.5.4 are satisfied and if
\[
\liminf_{n \to \infty} \sup_{x \in X} \bar{\nu}^{KG,n}(x) = 0,
\]
then \( \text{Var}^n(\mu(x)) \) converges to zero for all \( x \).
Proof.

\[ \hat{\nu}^{KG,n}(x) \]
\begin{align*}
\hat{\nu}^{KG,n}(x) & = E \left[ \max_{i=0,\ldots,n} \mu^{n+1}(x^i|\mathcal{F}^n, x^n = x) - \max_{i=0,\ldots,n} \mu^n(x^i|\mathcal{F}^n) \right] \\
& = E \max_{i=0,\ldots,n} \mu^{n+1}(x^i|\mathcal{F}^n, x^n = x) - \max(\mu^n(x^i), \mu^n(x)) \\
& \geq E \left[ \max(\mu^{n+1}(x^i), \mu^{n+1}(x)|\mathcal{F}^n) \right] - \max(\mu^n(x^i), \mu^n(x)) \\
& = E \left[ \max \left( \mu^n(x^i) + \hat{\sigma}_n(\Sigma^n, x)Z^{n+1}, \mu^n(x) + \hat{\sigma}_n(\Sigma^n, x)Z^{n+1} \right) \right] - \max(\mu^n(x^i), \mu^n(x)) \\
& = E \left[ \max \left( a_1 + b_1 Z^{n+1}, a_2 + b_2 Z^{n+1} \right) \right] - \max(a_1, a_2)
\end{align*}

(2.43)

\[ \begin{aligned}
\frac{\phi(a_2 - a_1)}{|b_1 - b_2|} + a_2 & \Phi \left( \frac{a_2 - a_1}{|b_1 - b_2|} \right) + a_1 \left( 1 - \Phi \left( \frac{a_2 - a_1}{|b_1 - b_2|} \right) \right) + |b_1 - b_2| \phi \left( \frac{a_2 - a_1}{|b_1 - b_2|} \right) - \max(a_1, a_2) \\
& = -|a_2 - a_1| \Phi \left( \frac{-|a_2 - a_1|}{|b_1 - b_2|} \right) + |b_1 - b_2| \phi \left( \frac{|a_2 - a_1|}{|b_1 - b_2|} \right).
\end{aligned} \]

(2.44)

In (2.43), we define \( i^* = \arg \max_{i=0,\ldots,n-1} \mu^n(x^i) \). In (2.44), for convenience, we define \( a_1 = \mu^n(x^{i^*}), a_2 = \hat{\sigma}_n(\Sigma^n, x), b_1 = \mu^n(x), \) and \( b_2 = \hat{\sigma}_n(\Sigma^n, x) \). The term in (2.45) is nonnegative and decreases as \( |a_2 - a_1| \) increases or \( |b_1 - b_2| \) decreases. Equation (2.45) holds for all decisions \( x \). Now, assume there is a decision \( x^{b_1} \) such that \( \lim_{n \to \infty} Var^n[\mu(x^{b_1})] = \epsilon_1 > 0 \). This limit exists because \( Var^n[\mu(x^{b_1})] \) is a decreasing sequence bounded below by zero as shown in (2.38). Then (2.45) becomes

\[ \begin{aligned}
\hat{\nu}^{KG,n}(x^{b_1}) & \geq -|\mu^n(x^{b_1}) - \mu^n(x^{i^*})| \Phi \left( \frac{-|\mu^n(x^{b_1}) - \mu^n(x^{i^*})|}{\hat{\sigma}_n(\Sigma^n, x^{b_1}) - \hat{\sigma}_n(\Sigma^n, x^{b_1})} \right) \\
& \quad + |\hat{\sigma}_n(\Sigma^n, x^{b_1}) - \hat{\sigma}_n(\Sigma^n, x^{b_1})| \phi \left( \frac{|\mu^n(x^{b_1}) - \mu^n(x^{i^*})|}{|\hat{\sigma}_n(\Sigma^n, x^{b_1}) - \hat{\sigma}_n(\Sigma^n, x^{b_1})|} \right).
\end{aligned} \]

(2.46)
Now by assumptions 2.5.2 and 2.5.3, \( \exists c_1, c_2 \) such that

\[
\limsup_{n \to \infty} |\mu^n(x) - \mu^n(x^*)| \le c_1 < \infty, \\
\limsup_{n \to \infty} Corr^n[\mu(x^{b_1}), \mu(x^*)] \le c_2 < 1.
\]

We can now put a lower bound on \( |\tilde{\sigma}_i^*(\Sigma^n, x^{b_1}) - \tilde{\sigma}_n(\Sigma^n, x^{b_1})| \).

\[
|\tilde{\sigma}_i^*(\Sigma^n, x^{b_1}) - \tilde{\sigma}_n(\Sigma^n, x^{b_1})| \\
= \frac{|Var^n[\mu(x^{b_1})] - Cov^n[\mu(x^{b_1}), \mu(x^*)]|}{\lambda + Var^n[\mu(x^{b_1})]} \\
\ge \frac{Var^n[\mu(x^{b_1})] - Corr^n[\mu(x^{b_1}), \mu(x^*)] \sqrt{Var^n[\mu(x^{b_1})]Var^n[\mu(x^*)]}}{\lambda + \beta} \\
\ge \frac{(1 - Corr^n[\mu(x^{b_1}), \mu(x^*)])\epsilon_1}{\lambda + \beta}.
\]

And now taking the limit inferior, we get

\[
\liminf_{n \to \infty} |\tilde{\sigma}_i^*(\Sigma^n, x^{b_1}) - \tilde{\sigma}_n(\Sigma^n, x^{b_1})| \ge \liminf_{n \to \infty} \frac{(1 - Corr^n[\mu(x^{b_1}), \mu(x^*)])\epsilon_1}{\lambda + \beta} \\
\ge \frac{c_2\epsilon_1}{\lambda + \beta} \\
= c_3 > 0.
\]

Going back to (2.46) and taking the limit inferior, we can now write

\[
\liminf_{n \to \infty} \tilde{\nu}_{KG,n}^{KG}(x^{b_1}) \ge -c_1 \Phi \left( \frac{-c_1}{c_3} \right) + c_3 \phi \left( \frac{c_1}{c_3} \right) > 0. \tag{2.47}
\]

By assumption the limit inferior of the supremum of the knowledge gradient for continuous parameters over all decisions is zero and thus (2.47) provides a contradiction.

**Corollary 2.5.2.** Under the KGCP Policy, if Assumptions 2.5.1, 2.5.2, 2.5.3, and 2.5.4 are satisfied, then \( \lim_{n \to \infty} Var^n[\mu(x)] = 0 \) for all \( x \).

**Proof.** Combining Theorem 2.5.1 and Theorem 2.5.2
2.6 Numerical Results

In this section we give an illustrative example of the KGCP policy as well as analyzing its performance on several standard test functions. We first illustrate the KGCP policy on the 2-dimensional Branin function and set the variance of the normally distributed observation noise to one ($\lambda = 1$). We plot the true Branin function in Figure 2.4. We stick with the more conservative convention of an initial LHS design using two times the number of dimensions plus two ($2p + 2$) used in [36] ([57] suggests using $10p$). After every observation we estimate the parameters ($\alpha$, $\beta$, $\lambda$, and $\mu^0$) with maximum likelihood estimation. Our estimate of the function after the initial 6 observations is shown in Figure 2.5(a) and the knowledge gradient for continuous parameters for each decision is shown in Figure 2.5(b). The knowledge gradient for continuous parameters is higher at decisions that have higher estimates or more uncertainty or both. At this point, after each observation, we update our estimate of the parameters and then choose our sampling decision by maximizing the knowledge gradient for continuous parameters. We repeat this several times, and Figure 2.6 shows the estimate of the function after 20 total observations chosen with the KGCP policy. Comparing these estimates with the true function shown in Figure 2.4, we
Figure 2.5: (a) The estimate of the function after 6 observations. The actual observations are plotted as well. (b) The knowledge gradient for continuous parameters surface is plotted. The height is a measure of how much we expect the maximum of the estimate of the function to increase by measuring the corresponding decision. We choose the next sampling decision by finding the decision which maximizes the knowledge gradient for continuous parameters shown in 2.5(b).

visually see that the policy has done a good job estimating the upper regions of the function as desired.

Figure 2.6: (a) The estimate of the function after 20 observations. The actual observations are plotted as well. (b) The contour plot of the estimate of the function after 20 observations.

2.6.1 Standard Test Functions

Next we compare the KGCP policy with sequential kriging optimization (SKO) from [44] on expensive functions with observation noise. We use the various test func-
tions used in [36], [45], and [44] as the true mean and add on normally distributed observation noise with variance $\lambda$. We define the opportunity cost as,

$$OC = \max_i \mu(i) - \mu(i^*),$$ (2.48)

where $i^* = \arg \max_i \mu^n(i)$, and Table 2.3 shows the performance on the different functions. These functions were designed to be minimized so the KGCP policy was applied to the negative of the functions. Each policy was run 500 times with the specified amount of observation noise. Table 2.3 gives the sample mean and sample standard deviation of the mean of the opportunity cost after 50 iterations for each policy. (To get the sample standard deviation of the opportunity cost you would multiply by $\sqrt{500}$). The means of the opportunity costs which are significantly better (using Welch’s t test at the .05 level (see [100])) are bolded. The results are given for different levels of noise; $\lambda$ is the variance of the normally distributed noise in the observations. Because a Gaussian process (GP) is only an approximation (a surrogate) for the preceding test functions, we next apply KGCP and SKO to functions that are guaranteed to be GP’s. Each GP row of Table 2.3 summarizes the results of running the policies on 500 GP’s created as follows: a function was generated from a 1-dimensional GP with the specified parameters of the covariance matrix in (2.2) over a 300 point grid on the interval $[0, 15]$. The standard deviation of each function, $\sigma$, is given as well to give a frame of reference for the values of $\lambda$. This number was created by taking the standard deviation of function values over a discretized grid. For all these runs (even the Gaussian process surfaces) an initial LHS design of $2p + 2$ function evaluations is used and maximum likelihood estimation is performed after each iteration to update the estimates of $\alpha$, $\beta$, $\lambda$, and $\mu^0$ (see [73]).

KGCP and SKO appear to have similar performance on Hartman 3 and Six Hump Camelback test functions. However, the KGCP policy does significantly better on the
Table 2.3: Performance on Standard Test Functions. Each row summarizes 500 runs of each policy on the specified test function with the specified observation noise variance. We define $\sigma(OC)$ as $\text{Std}(\mathbb{E}(OC))$ and $\text{Med}$ as the median $OC$.

Ackley 5 and Branin test functions, as well as most of the Gaussian process functions.

To get an idea of the rate of convergence of the KGCP policy, we plot the performance on the Gaussian processes in Figure 2.7. These promising simulations demonstrate that the KGCP algorithm is a very competitive policy.

### 2.6.2 Calibration Problem

We want to calibrate our simulator over $p$ continuous parameters, $x = [x_1, ..., x_p]^T$.

After the simulator runs with a fixed $x$, it outputs $q$ statistics of the company’s simulated operations, $G_1(x), ..., G_q(x)$; our observations of these statistics contain noise because there is randomness in the simulator. We want to calibrate the simulator against the company’s actual operations so we use the company’s true statistics which are $g_1, ..., g_q$. We want to find parameters $x$ such that $G_i(x)$ is close to $g_i$ for $i = 1, ..., q$.

Using a quadratic loss function, we can write our objective as maximizing the function
Figure 2.7: (a)-(c) show examples of Gaussian Processes with the given covariance parameters. (d)-(f) show the mean opportunity cost of the KGCP policy on the various Gaussian processes.

\[ \mu(x) \text{ given by} \]
\[ \mu(x) = - \sum_{i=1}^{q} (G_i(x) - g_i)^2. \]  \hspace{1cm} (2.49)

Our optimization problem can now be written

\[ \max_{x \in \mathcal{X}} \mu(x), \]  \hspace{1cm} (2.50)

where \( x \in \mathbb{R}^p \) and \( \mathcal{X} \) is a compact set. Our observations of \( \mu(x) \) contain noise since there is randomness in the simulator. We will sequentially choose parameter settings \( x^n \) for \( n = 0, \ldots, N - 1 \) where \( N \) is the number of times we are allowed to run the simulator due to time constraints. For \( n = 0, \ldots, N-1 \), after deciding to try parameter setting \( x^n \), we run the simulator which gives us a noisy observation of \( \mu(x^n) \) which we call \( \hat{y}^{n+1} \). We next describe a model for \( \mu(x) \).
2.6.3 Airline Calibration

We now demonstrate the approximate knowledge gradient policy on a particular calibration problem of the form given by equations (2.49) and (2.50). An industrial computer model simulates one month of a business jet’s operations which involves customers’ demands for airplanes, crew scheduling, ferrying, and unexpected airplane repairs; the model takes approximately fifteen minutes to simulate a month of the company’s operations once. After the program simulates the month of the operations one time, it outputs a set of statistics summarizing the behavior of the company over the month. The statistics of interest include the percentage of airplanes upgraded, $G_1(x)$, and the percentage of airplanes downgraded, $G_2(x)$; if the company cannot provide the exact airplane model requested by the customer, the company can upgrade or downgrade the customer’s demand by providing a better or worse plane (each airplane is given a quality number based on its size and age). The other important output statistics are three utilities ($G_3(x), G_4(x), G_5(x)$) which are observable percentages based on the airlines operations. In order for the simulator to be useful, these output statistics should match the previous month’s operations of the company fairly closely. The actual statistics of the company are given, and we call them our target goals, $g_1, ...., g_5$.

Before each run of the simulator, we may choose the parameter settings, $x$, of the simulator. For our particular problem we are allowed to set the maximum the plane can be downgraded, $x_1$, or upgraded, $x_2$. Parameters $x_3$ and $x_4$ are the simulator’s scaling factors for the penalties for downgrading and upgrading a customer’s demand; the company provides the customer with an airplane as similar to their request as possible. The last parameter, $x_5$, controls the simulator’s amount of ferrying (flying airplanes without passengers). The company can fly planes without passengers in order to strategically place airplanes and crew in more useful locations.
Our calibration problem requires solving the optimization problem in equation (2.50). Since each run of the simulator requires fifteen minutes, we limit ourselves to 50 iterations to determine the best parameter settings.

2.6.4 Performance on Calibration Problem

In this section we compare the algorithms for calibrating the airline simulator. Based on knowledge of plausible parameter values, we specify our domain by requiring $x_1 \in [0, 35.0]$, $x_2 \in [0, 4]$, $x_3 \in [20, 60]$, $x_4 \in [0, 4]$, $x_5 \in [0, 60]$, and the target goals for the output statistics in the objective are set to be $g_1 = 40$, $g_2 = 4$, $g_3 = 60$, $g_4 = 60$, and $g_5 = 60$. For the AKG and SKO, the first 12 observations were chosen by an LHS design in order to obtain MLE estimates of the parameters of the covariance function in equation (2.2) which are the variance of the observation noise, $\lambda$, and the initial mean, $\mu^0$.

We would like the objective given by equation (2.49), which is negative, to be as large as possible, although an objective of zero is not necessarily achievable by the simulator. For LHS, the implementation decision, $x^*$, is chosen simply as the sampling decision with the largest (best) observation. For AKG in these numerical tests the implementation decision is chosen in the same way as SKO; [44] gives a exponential utility maximization argument to suggest using $x^* = \arg \max_{x_0, \ldots, x_{n-1}} \mu^n(x) - \sqrt{\Sigma^n(x, x)}$ for the implementation decision. This avoids implementation decisions with very large uncertainty that $\max_x \mu^n(x)$ may occasionally give.

Figure 2.8 shows the results of the the algorithms calibrating the simulator, each with 50 observations from the simulator. Each algorithm was run five times, and the five sample paths of the objective are plotted. To produce Figure 2.8, the estimates of the objective are obtained after the algorithm has run by going back and running the simulator at each of the implementation decisions. The Latin hypercube sampling design appears to have a steadily increasing objective. SKO and AKG have similar
performances, although they both appear better than LHS. The output statistics, $G_1$ through $G_5$, are all on the scale of 0-100. Limited sample paths are available due to the time-consuming nature of the simulator, but we can visually conclude that the AKG policy is a competitive algorithm.

![Performance on Airline Calibration](image)

Figure 2.8: We show the performance of the algorithms on the calibration problem. The estimates of the objective of each algorithm are shown from five sample paths.

We now briefly examine the first sample path of the AKG algorithm by plotting where it sampled as well as how the output statistics of the implementation decisions improve as the number of sampled decisions increases. Figure 2.9 shows where the policy is sampling in each of the five dimensions. The final implementation decision for this sample path is $x = [20.4, 0.40, 44.5, 0.16, 51.68]^T$. Looking at the histograms in Figure 2.9, we see that for each dimension of $x$ the AKG policy has explored much of the decision space, but has also spent additional time sampling near the final implementation decision.
Figure 2.9: Frequency histograms showing where the AKG policy sampled in each dimension for a particular sample path.

2.7 Conclusion and Future Work

The knowledge gradient for continuous parameters is applicable to problems with continuous decision variables and observation noise and is similar to the expected improvement used in EGO when there is no observation noise. We presented a gradient ascent algorithm to approximately maximize the knowledge gradient for continuous parameters. The KGCP policy is very competitive with SKO and has nice convergence theory, giving conditions under which our uncertainty about the maximum of the expensive function with observation noise disappears. Extensions could include additional research with a priori distributions as well as additional approximations to speed up computations as the number of observations get large. Additional issues for further investigation are evaluating the algorithm on problems with larger dimensions, $p$, and applying the algorithm on problems with unequal variances in the observation noise.
Chapter 3

Approximate Dynamic Programming for Energy Storage with New Results on Instrumental Variables and Projected Bellman Errors

3.1 Introduction

Incorporating large amounts of energy from intermittent resources into the power grid creates many complications due to both variability and uncertainty. For example, if the wind power in the system drops suddenly, expensive ancillary services are required to satisfy the load. We also have to deal with electricity prices including time-varying contract prices as well as highly volatile spot prices. We need to manage our system to meet a time-varying load which has its own sources of uncertainty due to weather. Drawing on power from wind requires that we deal with an exogenously varying sup-
ply that introduces both short-term volatility with a daily cycle which is out of sync with loads. An electricity storage device can be used to mitigate the effects of the intermittency and uncertainty of wind as well as providing other services to a grid operator. Potential uses for an electricity storage device include electricity price arbitrage, generation capacity, ancillary services, transmission support, electricity service reliability, time-of-use energy cost management, regulation of energy production from renewables, and time-shifting of renewable energy (see [29]).

Many recent papers discuss the benefits of combining energy storage devices with renewables. [22] describes a virtual power plant which uses a dynamic programming algorithm to operate an energy storage facility and a wind farm. [88] describes the potential benefits of combining wind power with hydro storage. [41] and [92] discuss combining wind with compressed air energy storage. [86] investigates the potential value of a storage device in the PJM network used for arbitrage. [64] uses stochastic programming to operate a storage device which buys and sells in an electricity spot market. [46] discusses the value of the ability of electric vehicles to provide peak power, spinning reserves, and regulation. [103] examine a dual-threshold policy for a wind, storage, and transmission system. [54] discusses how approximate dynamic programming can be used to bound the value of natural gas storage, and [83] derives optimal policies for storing natural gas under certain assumptions on natural gas prices. [16] uses stochastic impulse control to operate a gas storage device. A thorough review of this growing literature is beyond the scope of this paper.

We address the problem of optimally controlling the power flows among a source with intermittent supply, a grid which offers infinite supply at a variable price, and a variable load in the presence of a storage device. A byproduct of this research will be the ability to estimate the economic value of storage for both long term investment as well as day-ahead tactical planning. The answers to these questions require knowing how the energy storage device will be used. In general, deciding how to charge
and discharge the storage device is a difficult problem to solve optimally due to the uncertainty in the wind, electricity prices, and electricity demand.

The primary contribution of the paper is the development of high quality, scalable algorithms for the near-optimal control of an energy storage device in the presence of complicating side variables such as prices, loads, and energy from renewables. We develop an optimal benchmark for a simplified problem and use this to evaluate an approximate policy iteration algorithm using least-squares policy iteration, an algorithmic strategy with strong theoretical support. We demonstrate the importance of using instrumental variables in this algorithmic strategy (see [27]; [47]; [87]; [8]). Recent research has also focused attention on the use of projected Bellman error minimization. We show for the first time that this is mathematically equivalent to Bellman error minimization when instrumental variables. Despite the strong theoretical support enjoyed by this algorithmic strategy, we also show that direct policy search still produces much better policies.

This paper is organized as follows. Section 3.2 gives a description and model of wind, electricity prices, electricity demand, and energy storage. Section 3.3 sets up the dynamic program that combines stochastic wind, stochastic electricity prices from the grid, and an energy storage device to satisfy a stochastic load. Section 3.4 summarizes approximate policy iteration for solving the dynamic program. Within policy iteration, we focus on several policy evaluation algorithms based on minimizing Bellman error: (1) instrumental variables Bellman error minimization, (2) least-squares projected Bellman error minimization, (3) instrumental variables projected Bellman error minimization. We show that these three policy evaluation algorithms are equivalent under certain full rank assumptions and converge when using off-policy sampling under certain conditions. Section 3.5 describes an alternative strategy to fit the parameters of a value function approximation using direct policy search. Finally, in Section 3.6 we analyze the performance of the approximate dynamic programming
policies on a series of simplified, discretized problems for which we have obtained an optimal benchmark, and then on the full, multidimensional problem with continuous variables. A byproduct of this research is a set of benchmark problems which can be used by the algorithmic community to test approximate algorithms with an exact solution and finally the full model.

3.2 Models

We wish to address the problem of combining power from the grid with stochastic prices, wind with stochastic supply, and storage to meet a stochastic demand for electricity as shown in Figure 3.1. We begin by describing the models we use for wind, electricity prices, electricity demand, and energy storage.

3.2.1 Wind

[11] suggests modeling the square root of the wind speed with an autoregressive (AR) model, while [18] suggests using a more general ARIMA model. Let $W_t$ be the wind
speed in (m/s). We define $Y_t$ to be the de-meaned square root of the wind speeds; $Y_t = \sqrt{W_t} - \mathbb{E}[\sqrt{W_t}]$. We use the wind speeds at Maryneal, TX every fifteen minutes to fit an AR model to $Y_t$. For the purpose of keeping the state space small we use an AR(1),

$$Y_t = \phi_1 Y_{t-\Delta t} + \epsilon_t,$$

(3.1)

where $\epsilon_t \sim \mathcal{N}(0, \sigma^2_\epsilon)$. Using the Yule-Walker equations (see [15]) and setting $\Delta t = 15$ minutes, we find the following estimates: $\mathbb{E}[\sqrt{W_t}] = 1.4781; \phi_1 = 0.7633; \sigma_\epsilon = 0.4020$. Now we can simulate $Y_t$ and then transform back to the corresponding wind speed $W_t$. Once we have the wind speed we can convert to the power produced by a wind turbine using a typical power curve equation (see [13]; [1]),

$$P_t = .5C_p \rho A W_t^3.$$

(3.2)

Here, $C_p$ is the power coefficient that is less than the Bentz limit of .593 (corresponding approximately to $C_p = .45$), $\rho$ is the density of air ($\rho = 1.225 kg/m^3$). $A$ is the area swept by the rotor blades of the turbine ($A = \pi 50^2 m^2$ for a typical turbine), $W_t$ is the velocity of the wind in m/s, and $P_t$ is the power output from the turbine in watts (1 watt = 1 kg $m^2$ /s). Typically there is a cut-in wind speed that is the minimum speed necessary to produce power, a rated wind speed (beyond which the wind turbine does not produce any extra power), and, finally, a very large speed called the cut-out wind speed, above which the turbine must be shut off.

### 3.2.2 Electricity Prices

In the PJM day-ahead market, PJM receives offers and bids for the next operating day, and at 4pm the day-ahead prices are determined with the scheduling, pricing, and dispatch program. In addition, there is an hourly real-time (spot) market that has even more extreme prices than the day-ahead market. The real-time prices at
the PJM Western Hub average $42.11 per MWh over 2009-2010, although the prices are occasionally negative and have a maximum of $362.90 per MWh. Figure 3.2 shows that the prices are lowest at night; they begin to increase around 5am and are typically the highest in the evening around 6pm.

Figure 3.2: The average PJM Real-Time price at the Western Hub as a function of the hour of the week.

We fit a jump diffusion process to the deseasonalized real-time electricity prices (see [17]). We first take the electricity prices, $P_t$, and convert to log prices,

$$Y_t = \log(P_t + c).$$

(3.3)

In Equation (3.3), we add a constant $c$ before taking the natural log to ensure we do not take the log of a negative number (we set $c$ to one minus the minimum value of $P_t$). We next calculate the deseasonalized log electricity prices, $Y_t^{ds} = Y_t - Y_t^s$, where $Y_t^s$ is the seasonal component and is a deterministic periodic function of $t$. When calibrating $Y_t^s$, we use an hour of week and month of year component. We then fit a jump diffusion process to the deseasonalized log prices,

$$dY_t^{ds} = \lambda(\mu - Y_t^{ds})dt + \sigma dW_t + dN_t,$$

(3.4)
where $\mu$ is the long term equilibrium price, $\lambda$ is the mean reversion rate, $W_t$ is a Brownian motion, and $N_t$ is the jump process. Discretizing, we can write

$$Y_{t+\Delta t}^{ds} - Y_{t}^{ds} = \lambda(\mu - Y_{t-\Delta t}^{ds})\Delta t + \sigma\sqrt{\Delta t}\epsilon_t + J_t,$$

(3.5)

where $\{\epsilon_{t+n\Delta t}\}_{n=0}^N$ are i.i.d. standard normal random variables, and $J_t$ is the jump over the interval $(t - \Delta t, t]$. If we were to model the jumps with a compound Poisson process, we could write $J_t = \sum_{k=0}^{X_t - X_{t-\Delta t}} J_{t,k}$, where $X_t$ is a Poisson process with intensity $\lambda^{Poisson}$ (hence the number of arrivals $X_t - X_{t-\Delta t} \sim$ Poisson($\lambda^{Poisson}\Delta t$)). However, for calibration purposes, [17] models the jumps as the i.i.d. process,

$$J_t = \epsilon_t^{jump}1(U_t < p^{jump}),$$

(3.6)

where $\epsilon_t^{jump}$ is the size of a jump, $U_t \sim$ unif$(0,1)$, and $p^{jump}$ is the probability of a jump over a time interval of $\Delta t$. We identify the nonzero jumps as in [17] by locating times where the absolute value of the return is more than three times the standard deviation of the returns. We can then fit $p^{jump}$ as the fraction of time jumps occur (we divide this estimate by two because most jumps are immediately followed by jumps in the opposite direction). In addition, we model $\{\epsilon_{t+n\Delta t}^{jump}\}_{n=0}^N$ as i.i.d. normal random variables with mean zero and standard deviation $\sigma^{jump}$.

At this point we can obtain estimates of $\lambda$, $\mu$, and $\sigma$ using least-squares linear regression on Equation (3.5); $Y_t^{ds} - Y_{t-\Delta t}^{ds}$ are the observations, and $\sigma\sqrt{\Delta t}\epsilon_t + J_t$ are the centered residuals. The variance of the residuals is,

$$\text{Var}(\sigma\sqrt{\Delta t}\epsilon_t + J_t) = \sigma^2\Delta t + \text{Var}(J_t) = \sigma^2\Delta t + p^{jump}(\sigma^{jump})^2,$$

(3.7)

which gives an equation which can be used for estimating $\sigma$. 

54
3.2.3 Electricity Demand

[28] outlines typical models for residential, commercial, and industrial power demand. Industrial power demand is relatively stable while residential power demand is highly dependent upon the temperature. For example, [68] models the load with a reflected Brownian motion that incorporates a seasonal component. [33] summarizes the main approaches to forecasting load such as an end-use model that incorporates appliances and customers, various regression models (based on temperature, time, and other factors), time series, and heuristics made by experts. [28] prefers the method of modeling the load as a function of temperature; additional factors could be used such as the temperature-humidity index and wind chill index (see [33]).

We use the actual total ERCOT energy loads every hour over 2010 (we can convert this to power by assuming the power consumption is constant over a time interval and using $E = P\Delta t$). The load clearly exhibits some hourly and daily features as shown in Figure 3.3.

![Figure 3.3: The total ERCOT load over the first full week of 2010.](image)

In general, the ERCOT load starts ramping up in the morning around 5am and peaks in the evening around 6pm, although the patterns vary greatly based on the
day of the week and the month of the year. We can deseasonalize the loads and then analyze the residuals of the loads. We write the deterministic seasonal component of the load, $m_t$, as the sum of an hour-of-week and monthly component. For any time, $t$, the hour of the week is an integer from 0, 1, ..., 167; zero corresponds to Sunday morning between 00:00:00 and 00:59:59, and 167 corresponds to Saturday night between 23:00:00 and 23:59:59 military time. To calculate the seasonal component, we calculate the average load over each of the hours of the week and call this the hour-of-week seasonal component, $m_{t\text{\ hour}}$. We take the residuals and calculate the average load over each of the months and call this the month-of-year seasonal component, $m_{t\text{\ month}}$. The residuals are then called the deseasonalized load, $D_{t}^{ds}$. We can write the decomposed load as,

$$D_t = m_t + D_{t}^{ds},$$

(3.8)

where the seasonal component $m_t$ is defined as

$$m_t = m_{t\text{\ hour}} + m_{t\text{\ month}}.$$

3.2.4 Energy Storage

[12] explains that stationary lead-acid batteries with tubular plates need a small amount of maintenance and can last up to 12 years when operated properly, costing approximately $300 per kWh of capacity. They typically have a round trip efficiency of 70% and a self-discharge rate of 2% per month and should not be discharged below 20% of their capacity (see [12]). The lifespan of the batteries can be maximized by limiting the depth of discharge to 15% per day. A typical lead-acid battery may have a C/10 maximum discharge rate, meaning it can be fully discharged over 10 hours using the maximum discharge rate (see [26]). In our work, we do not consider the effect of the storage rate on storage efficiency, as governed by Peukert’s Law (see [2]);
this would be a nice extension for handling lead-acid batteries, but is beyond the scope of our presentation.

### 3.3 Dynamic Programming Problem

We now address the problem of combining power from the grid with a stochastic price, wind with a stochastic supply, and storage to meet a stochastic demand for electricity. We call $D_t$ the total energy demand (in MWh) over the time period starting at $t - \Delta t$ and ending at $t$. This energy demand must be met at every time period from either wind energy, energy from the battery, or energy from the grid. We fix a time step, $\Delta t$, of fifteen minutes. The full model is described below.

#### 3.3.1 State Variable

The state variable, $S_t$, the fraction of the storage that is full ($R_t$), the current amount of wind energy in MWh ($E_t$), the current energy demand in MWh ($D_t$), and the current spot price of electricity to and from the grid in $$/MWh ($P_t$). We solve both steady-state and time-dependent applications. For time-dependent problems, we also include time $t$ in the state variable. We can write $S_t = (R_t, E_t, D_t, P_t)$.

#### 3.3.2 Decision Variables

For a fixed time $t$, the flows in Figure 3.1 can be represented by the vector $\{x_t^{WR}, x_t^{GR}, x_t^{RD}, x_t^{WD}, x_t^{GD}\}$, where $W$ refers to wind, $R$ refers to the battery resource, $G$ refers to the grid, and $D$ refers to the demand. At the wind node, the wind energy must either go to the storage or to the demand (we assume the storage can dissipate energy if necessary),

$$x_t^{WR} + x_t^{WD} = E_t^{\text{wind}}.$$
At the demand node, the energy demand is satisfied by the grid, the storage, and the wind,

\[ D_t = x_t^{GD} + \eta_{\text{discharge}} x_t^{RD} + x_t^{WD}. \]

Now we define the constants \( \Delta R_{\text{min}} \) and \( \Delta R_{\text{max}} \) as the minimum and maximum fraction of the storage you can charge over \( \Delta t \) (negative values correspond to discharging).

For example, if we have a lead acid battery with a \( C/10 \) maximum charge and discharge rate, and \( \Delta t = 15\text{min} \), then \( \Delta R_{\text{min}} = -1/40 \) and \( \Delta R_{\text{max}} = 1/40 \). Now, the feasible actions must satisfy,

\[
\frac{\Delta R_{\text{min}} R_{\text{capacity}}}{\eta_{\text{discharge}}} \leq x_t^{GR} \leq \frac{\Delta R_{\text{max}} R_{\text{capacity}}}{\eta_{\text{charge}}}, \quad (3.9)
\]

\[
0 \leq x_t^{RD} \leq \Delta R_{\text{max}} R_{\text{capacity}}. \quad (3.10)
\]

Equation (3.9) ensures that we do not charge or discharge the storage device faster than the storage device allows. In Equation (3.9) we could require \( 0 \leq x_t^{GR} \) if we did not want to allow selling from the storage to the grid. Equation (3.10) guarantees that we do not discharge the storage device faster than allowed when sending energy from the storage to demand. In our problem the demand must always be satisfied so it is easy to see how to optimally use the wind energy. We send as much wind as possible to demand, and the remaining wind is sent to the storage device for future use,

\[
x_t^{WD} = \min(E_t^{\text{wind}}, D_t), \quad (3.11)
\]

\[
x_t^{WR} = E_t^{\text{wind}} - x_t^{WD}, \quad (3.12)
\]

\[
x_t^{GD} = D_t - \eta_{\text{discharge}} x_t^{RD} - x_t^{WD}. \quad (3.13)
\]

Equations (3.12) and (3.13) are the flow constraints at the wind and demand node. Equations (3.11), (3.12), and (3.13) effectively reduce the size of our action space from
5 dimensions to 2 dimensions. In addition we require that $R_{t+\Delta t} > 0$ because the battery cannot go negative (in the case of lead-acid batteries we require $R_{t+\Delta t} > .2$ to prevent the battery from becoming fully discharged).

### 3.3.3 Exogenous Information Process

We define the exogenous information process as the random changes in the state of the system, $W_{t+\Delta t} = \{\hat{E}_{t+\Delta t}, \hat{D}_{t+\Delta t}, \hat{P}_{t+\Delta t}\}$, which refer to exogenous changes in the energy from the wind $E_t$, loads $D_t$ and electricity spot prices $P_t$. These exogenous changes may be state dependent as well as time dependent.

### 3.3.4 State Transition

We write the state transition function as, $S_{t+\Delta t} = S^M(S_t, x_t, W_{t+\Delta t})$. The updated state variables can be written,

\[
\begin{align*}
E_{t+\Delta t} &= E_t + \hat{E}_{t+\Delta t}, \\
D_{t+\Delta t} &= D_t + \hat{D}_{t+\Delta t}, \\
P_{t+\Delta t} &= P_t + \hat{P}_{t+\Delta t}.
\end{align*}
\]

We assume extra energy can be dissipated at the storage device, and our next resource state can be computed,

\[
R_{t+\Delta t}(x_t) = \min \left( \frac{R_t R_{\text{capacity}} + \eta_{\text{charge}} (x_t^{GR} + x_t^{WR}) - x_t^{RD}}{R_{\text{capacity}}}, 1 \right).
\]
3.3.5 Contribution and Objective

The contribution function is simply the dollar value of energy sold minus the amount bought from the grid,

\[ C(S_t, x_t) = P_t D_t - P_t(x_t^{GR} + x_t^{GD}). \]

We consider the ergodic infinite horizon problem where the goal is to find the policy, \( X^\pi(S_t) \), which maximizes the expected discounted future rewards,

\[ \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t C(S_t, X^\pi(S_t)) \right]. \]  \hspace{1cm} (3.14)

In our model, the policy \( X^\pi(S_t) \) may be stationary (where the function does not vary over time) or time-dependent, as might occur when the function depends on the time of day. Time-dependent functions may be written using \( X^\pi_t(S_t) \), or by modifying the state variable to include time (from which we might compute hour of day).

3.4 Approximate Policy Iteration

The objective function (Equation (3.14)) can be solved approximately using several algorithmic strategies such as rolling horizon procedures (also known as model predictive control), stochastic programming, or some form of parameterized policy, but we are going to focus on policies based on value function approximations. To do this, we start with Bellman’s optimality equation, which allows us to write

\[ V_t(S_t) = \max_x C(S_t, x) + \gamma \mathbb{E}\{V_{t+1}(S_{t+1})|S_t\}, \]

where \( S_{t+1} = S^M_t(S_t, x, W_{t+1}) \) and the expectation is over the random variable \( W_{t+1} \).

Since the state variable is multidimensional and continuous, this equation cannot
be solved exactly, and as a result a large field of research has evolved to develop approximations of the value function (see [71]; [5]; [90]; [72]).

We focus on the most widely studied class of approximations that uses a linear model with pre-specified basis functions. We further take advantage of the strategy of using a post-decision state variable, denoted $S^x_t$, which is the state immediately after a decision but before any exogenous information has been revealed. For our problem, the post-decision state is given by

$$S^x_t = (R_{t+\Delta t}(x), E_t, D_t, P_t).$$

We then approximate the value function using the general form

$$\bar{V}_t(S^x_t) = \sum_{f \in F} \theta_f \phi_f(S^x_t),$$

(3.15)

where $(\phi_f(s))_{f \in F}$ is a user-specified set of features.

In approximate policy iteration there is typically an outer loop called policy improvement along with an inner loop where policy evaluation is run for a fixed policy. For approximate policy iteration algorithms, the policy improvement loop is fairly standard, but there are many variations of the policy evaluation algorithm.

In the remainder of this section, we review a class of algorithms based on a concept known in the reinforcement learning community known as least squares temporal difference (LSTD) learning. We review the theory, and then describe the algorithm based on Bellman error minimization. We next present two variants: the first uses the concept of projected Bellman error minimization, and the second uses instrumental variables, after which we demonstrate that these two methods are equivalent.
3.4.1 Theory

Many approximate dynamic programming algorithms can be classified under the category of projected equation methods (see [40]; [99]; [10]; [51]; [53]; [4]). Much of the literature focuses on finite states, and typically these methods attempt to find the best value function within a class of value functions. For example, if a linear model (also called linear architecture) is used to approximate the value function, the objective may be to find the weights which minimize the $L_2$ norm of the Bellman residual.

We focus on the approximating the post-decision value function ([53] approximates the pre-decision value function). Letting $S^x$ be the post-decision state space, this can be written

$$
\min_{\theta} \sum_{s \in S^x} \left( V(s) - \sum_{f \in F} \theta_f \phi_f(s) \right)^2 = \min_{\theta} \| V - \Phi \theta \|_2^2, \quad (3.16)
$$

where $\Phi$ is a matrix of fixed basis functions (each row corresponds to a state, and each column corresponds to a basis function), and $\theta$ is a column vector of weights.

Using a linear model for the value function, [10] presents the least-squares temporal difference learning algorithm for the policy evaluation of a fixed policy which will be presented below. The weights chosen with the least-squares approach will converge with probability one to the true weights if the correct basis functions are used (the true value function happens to be in the span of the basis functions) and a few other technical conditions are satisfied.

Also assuming finite states and actions, [53] introduces least-squares policy iteration which approximates the value of state-action pairs ($Q$-factors) with a linear model. When doing policy evaluation, they choose to use least-squares to minimize the fixed-point approximation error instead of the Bellman residual. The paper references the approximate policy iteration theory from [5] which bounds the $L_\infty$ norm of the difference between the true and approximated $Q$-factors.
[10] explains that TD(\(\lambda\)) uses information inefficiently relative to the Least-Squares approach to TD policy evaluation (LSTD). The LSTD policy evaluation algorithm described in [10] is an on-policy algorithm which approximates the true value function with a linear model with fixed basis functions. The algorithm uses instrumental variables to obtain an estimate of the value function which converges with probability one as the number of transitions increases to infinity.

[53] expands upon the LSTD algorithm from [10] by using a linear architecture to approximate the value function over the higher dimension state-action pairs. Furthermore, they give the geometric interpretation of several different methods of approximately solving Bellman’s equation. Once the value function in Bellman’s equation has been replaced by a linear model, Bellman’s equation is typically an over-determined system which cannot be solved exactly. When solving for the weights of the value function, the Bellman residuals can be minimized in a least-squares or weighted least-squares sense (Bellman error minimizing approximation). An alternative approach is to project the Bellman residuals down into the space spanned by the basis functions of the value function and then minimize the Bellman residuals. [53] explains that in general the approximate value function is a fixed point of the projected Bellman operator, not the Bellman operator (see [24] for a nice discussion).

3.4.2 Algorithm

We first summarize approximate policy iteration based on Bellman error minimization (see [10], [53]). We use a modified version of Bellman’s equation based on the post-decision state variable (see [71], [4]). Typically, Bellman’s equation for an infinite horizon problem is written around the pre-decision value function,

\[
V(S_t) = \max_x \mathbb{E}[C(S_t, x) + \gamma V(S_{t+1})|S_t].
\]  

(3.17)
The post-decision state, $S^x_t$, is the state immediately after being in the pre-decision state $S_t$ and taking the action $x$, but before you observe the randomness from the state transition or receive the contribution (see [71] [Chapter 4] for a thorough discussion of post-decision states). The post-decision value $V^x(S^x_t)$ is the value of being in post-decision state $S^x_t$ and is defined as $V^x(S^x_t) = \mathbb{E}[V(S_{t+1})|S^x_t]$. Equation (3.17) can be written as

$$V(S_t) = \max_x \{C(S_t, x) + \gamma V^x(S^x_t)\}.$$  

Using only post-decision states, Bellman’s equation can be written as

$$V^x(S^x_{t-1}) = \mathbb{E}[\max_x \{C(S_t, x) + \gamma V^x(S^x_t)\}|S^x_{t-1}].$$  \hspace{1cm} (3.18)

In addition to bringing the expectation outside of the maximum in Bellman’s equation, the post-decision value function has the advantage that the post-decision state is often of lower dimension than the pre-decision state.

Next, we replace the post-decision value function with a parametric linear model, $V^x(S^x_t) = \phi(S^x_t)^T\theta$, where $\phi(\cdot)$ is a column vector of pre-determined, real-valued basis functions, $\phi_1(\cdot), \ldots, \phi_k(\cdot)$, and $\theta$ is a column vector of weights for the basis functions. Plugging this approximation into Equation (3.18) for a fixed policy $\pi$ we get

$$\phi(S^x_{t-1})^T\theta = \mathbb{E}[C(S_t, X^\pi(S_t|\theta)) + \gamma \phi(S^x_t)^T\theta|S^x_{t-1}].$$  \hspace{1cm} (3.19)

If we could find a value of the $\theta$ where this equation were exactly satisfied for all states, we would have the true value function for the policy $X^\pi(S_t|\theta)$. In general, we are only able to find a value of $\theta$ which approximately solves Equation (3.19). We outline the approximate policy iteration algorithm in Figure 3.4 which combines an inner loop which performs policy evaluation for a fixed policy with an outer loop.
which improves the policy. We now summarize several techniques for finding $\theta$ which approximately solves Equation (3.19).

### 3.4.3 Policy Evaluation using Bellman Error Minimization

We draw on the foundation provided in [10], but adapted for the post-decision state in [58]. We focus on the off-policy case where a set of post-decision states, $\{S_t^x\}_{i=1}^n$, are generated randomly and then, for each sample, $i = 1, ..., n$, we simulate the contribution and next post-decision state, $\{S_t^x\}_i$. We rewrite Equation (3.19) as

\[
C(S_t, X^\pi(S_t|\theta)) = \left( \phi(S_{t-1}^x) - \gamma \mathbb{E}[\phi(S_t^x)|S_{t-1}^x] \right)^T \theta + C(S_t, X^\pi(S_t|\theta)) - \mathbb{E}[C(S_t, X^\pi(S_t|\theta))|S_{t-1}^x].
\]

(3.20)

This is now in the form of a linear regression problem. Using simulation, we are able to get observations of $C(S_t, X^\pi(S_t|\theta))$ and $\left( \phi(S_{t-1}^x) - \gamma \mathbb{E}[\phi(S_t^x)|S_{t-1}^x] \right)^T$ in Equation (3.20). We can write this in matrix form as

\[
\begin{align*}
C_t &= (\Phi_{t-1} - \gamma \Phi_t) \theta + C_t - \bar{C}_t, \\
\Phi_{t-1} &= \begin{bmatrix} 
\phi(S_{t-1}^x_1)^T \\
\vdots \\
\phi(S_{t-1}^x_n)^T 
\end{bmatrix}, \\
C_t &= \begin{bmatrix} 
C(\{S_t\}_1, \pi(\{S_t\}_1)) \\
\vdots \\
C(\{S_t\}_n, \pi(\{S_t\}_n)) 
\end{bmatrix}.
\end{align*}
\]

(3.21)

(3.22)

(3.23)
\[
\Phi_t = \begin{bmatrix}
\phi(S^x_t) \\
\vdots \\
\phi(S^x_n)
\end{bmatrix},
\] (3.24)

and

\[
\bar{C}_t = \begin{bmatrix}
\mathbb{E}[C(S_t, \pi^\pi(S_t|\theta)|\{S^x_{t-1}\})] \\
\vdots \\
\mathbb{E}[C(S_t, \pi^\pi(S_t|\theta)|\{S^x_n\})]
\end{bmatrix},
\] (3.25)

We have used subscripts \(t-1\) and \(t\) to explicitly keep track of which vectors are known at time \(t - 1\) and \(t\), respectively. We refer to \(C_t - \bar{C}_t\) as the Bellman errors or Bellman residuals, although the terms may be defined slightly differently in other contexts.

For least-squares Bellman error minimization, the objective is to minimize the \(L_2\) norm of the Bellman errors in Equation (3.21), \(\frac{1}{n}(C_t - \bar{C}_t)^T(C_t - \bar{C}_t)\). Throughout this paper we use the following assumption which assumes the basis functions are linearly independent and certain matrices have full rank:

**Assumption 3.4.1.** \(\Phi_{t-1}, (\Phi_{t-1} - \gamma \Phi_t), \) and \((\Phi_{t-1})^T(\Phi_{t-1} - \gamma \Phi_t)\) have full column rank, and \(k \leq n\).

These assumptions can be interpreted as needing to visit enough different states such that the model can identified. The typical least-squares equation yields the following estimator for \(\theta\) which we refer to as least-squares Bellman error minimization,

\[
\hat{\theta} = \left[(\Phi_{t-1} - \gamma \Phi_t)^T(\Phi_{t-1} - \gamma \Phi_t)\right]^{-1}(\Phi_{t-1} - \gamma \Phi_t)^T C_t.
\] (3.26)

The matrix of regressors, \((\Phi_{t-1} - \gamma \Phi_t)\), is not deterministic \((\Phi_t\) is not deterministic because we cannot calculate \(\mathbb{E}[\phi(S^x_t)|S^x_{t-1}]\)); we can only simulate \(\phi(S^x_t)\) given \(S^x_{t-1}\),
Approximate Policy Iteration

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Initialize ( \theta ).</td>
</tr>
<tr>
<td>02</td>
<td>for ( j = 1:M ) (Policy Improvement Loop)</td>
</tr>
<tr>
<td>03</td>
<td>Define the policy ( X^\pi(S_t</td>
</tr>
<tr>
<td>04</td>
<td>for ( i = 1:N ) (Policy Evaluation Loop)</td>
</tr>
<tr>
<td>05</td>
<td>Simulate a random post-decision state, ( S_{t-1}^x ).</td>
</tr>
<tr>
<td>06</td>
<td>Record ( \phi(S_{t-1}^x) ).</td>
</tr>
<tr>
<td>07</td>
<td>Simulate the state transition to get ( S_t ).</td>
</tr>
<tr>
<td>08</td>
<td>Determine the decision, ( x = X^\pi(S_t</td>
</tr>
<tr>
<td>09</td>
<td>Record ( C_{t,i} = C(S_t, x) ).</td>
</tr>
<tr>
<td>10</td>
<td>Record ( \phi(S_t^x) ), the observation of ( \mathbb{E}[\phi(S_t^x)</td>
</tr>
<tr>
<td>11</td>
<td>End</td>
</tr>
<tr>
<td>12</td>
<td>Update ( \theta ) with Equation (3.26), (3.27), (3.30), or (3.31). (Policy Evaluation)</td>
</tr>
<tr>
<td>13</td>
<td>End</td>
</tr>
</tbody>
</table>

Figure 3.4: Summary of approximate policy iteration. The inner loop simulates transitions from a fixed policy in order to approximately evaluate the fixed policy. The outer loop improves the policy.

and, as a result, the least-squares estimator for \( \theta \) will typically be inconsistent. Due to the structure of the problem, we use the method of instrumental variables instead (see [10] and [58]). An instrumental variable is a variable that is correlated with the regressors, but uncorrelated with the errors in the regressors and the observations (see Appendix B.1 or [27]; [47]; [87]; [8]). This results in what we call instrumental variables Bellman error minimization (called LSTD in [10]),

\[
\hat{\theta} = [(\Phi_{t-1})^T(\Phi_{t-1} - \gamma \Phi_t)]^{-1}(\Phi_{t-1})^T C_t.
\] (3.27)

[10] gives conditions such that Equation (3.27) is a consistent estimator (\( \lim_{n \to \infty} \hat{\theta} = \theta \) with probability one) for the on-policy case. The proof references the consistency properties of the method of instrumental variables by showing that the columns of \( \Phi^n \) are appropriate instrumental variables (see Appendix B.1). One interesting comment is that the matrix \( [(\Phi_{t-1})^T(\Phi_{t-1} - \gamma \Phi_t)] \) could have negative eigenvalues, unlike \( [(\Phi_{t-1} - \gamma \Phi_t)^T(\Phi_{t-1} - \gamma \Phi_t)] \).
3.4.4 Policy Evaluation using Projected Bellman Error Minimization

Again we start with the rearranged form of Bellman’s equation using post-decision states in matrix form (see Equation (3.21)),

\[ C_t = (\Phi_{t-1} - \gamma \Phi_t)\theta + (C_t - \bar{C}_{t-1}). \] (3.28)

The idea of projected Bellman error minimization (also called least-squares fixed-point approximation in [53]) is to first project the Bellman errors into the space spanned by the basis functions of the value function and then minimize them (see [53] and [91]). Projecting the left and right hand sides of Equation (3.28) down into the space spanned by \( \Phi_{t-1} \) (with respect to the \( L_2 \) norm), we get

\[ \Pi_{t-1}C = \Pi_{t-1}(\Phi_{t-1} - \gamma \Phi_t)\theta + \Pi_{t-1}(C_t - \bar{C}_{t-1}), \] (3.29)

where \( \Pi_{t-1} = \Phi_{t-1}((\Phi_{t-1})^T\Phi_{t-1})^{-1}(\Phi_{t-1})^T \) is the projection operator onto the space spanned by the basis functions (see [95] for the original derivation of this mapping or [71] [Section 8.2.3] ). For completeness, we note that \( \Pi_{t-1} \) is the \( L_2 \) projection which solves the problem, \( \min_{\theta} \|\Phi_{t-1}\theta - b\|_2 = \|\Pi_{t-1}b - b\|_2 \). In other words, if you want to get from an arbitrary vector \( b \), to the closest vector (in the \( L_2 \) sense) that is in the span of the columns of \( \Phi_{t-1} \), just apply the projection \( \Pi_{t-1} \) to the vector \( b \). By Assumption 3.4.1 \( \Phi_{t-1} \) has full column rank so \( \Pi_{t-1} \) is well defined.

Typically, Equation (3.29) is an over-determined set of equations. Taking a least squares approach, we find \( \theta \) by minimizing the norm of the Projected Bellman Error

\[ \min_{\theta} \|\Pi_{t-1}(C_t - \bar{C}_{t-1})\|_2 = \min_{\theta} \|\Pi_{t-1}C - \Pi_{t-1}(\Phi_{t-1} - \gamma \Phi_t)\theta\|_2. \]
The least-squares estimator of \( \theta \) yields what we refer to as least-squares projected Bellman error minimization,

\[
\hat{\theta} = \left[ (\Pi_{t-1}(\Phi_{t-1} - \gamma \Phi_t))^T (\Pi_{t-1}(\Phi_{t-1} - \gamma \Phi_t)) \right]^{-1} (\Pi_{t-1}(\Phi_{t-1} - \gamma \Phi_t))^T \Pi_{t-1} C_t.
\]

(3.30)

However, this is the classic errors-in-variables model due to the randomness in our observations \( \Phi_t \), and instrumental variables can be used to construct a consistent estimator for \( \theta \) (see Appendix B.1.1). We show \( \Phi_{t-1} \) can be used as instrumental variables as before in Equation (3.27), and the proof is similar to that in [10]. The resulting estimator is what we call the projected Bellman error minimization with instrumental variables,

\[
\hat{\theta} = \left[ (\Phi_{t-1})^T \Pi_{t-1}(\Phi_{t-1} - \gamma \Phi_t) \right]^{-1} (\Phi_{t-1})^T \Pi_{t-1} C_t.
\]

(3.31)

For completeness, we note that \( \Pi_{t-1} \Phi_{t-1} \) could have been used for the instrumental variables instead of \( \Phi_{t-1} \), but linear algebra can be used to show the estimator would be equivalent to Equation (3.31).

### 3.4.5 Consistency of Projected Bellman Error Minimization with Instrumental Variables

We show that projected Bellman error minimization with instrumental variables is consistent (converges in probability to the true weights); the result holds even when the state space is continuous or the discount factor is one. For notation consistent with Appendix B.1, we let \( X = \Pi_{t-1}(\Phi_{t-1} - \gamma \mathbb{E}[\gamma \Phi_t | \{S_{t-1}^x \}]) \), \( X' = \Pi_{t-1}(\Phi_{t-1} - \gamma \Phi_t) \), \( X'' = X' - X \), \( Y'' = \Pi_{t-1}(C_t - \bar{C}_{t-1}) \), and \( Z = \Phi_{t-1} \). [10] proves a similar result for the on-policy case.
We first assume that the covariance matrix between the instrumental variables and regressors has full rank, and we assume we restrict ourselves to the off-policy case:

**Assumption 3.4.2.** $\Sigma$ has full rank $k$, where $\Sigma_{jl} = \text{Cov}[Z_j, X_l]$.

**Assumption 3.4.3.** The rows of $\Phi_{t-1}$ are i.i.d. (off-policy).

The following assumptions will be necessary to use the law of large numbers, which guarantees sample means converge to their true means:

**Assumption 3.4.4.** $E[|Z_{ij}Y''_i|] < \infty$, $\forall j = 1, ..., k$.

**Assumption 3.4.5.** $E[|Z_{ij}X''_{il}|] < \infty$, $\forall j, l \in \{1, ..., k\}$.

**Corollary 3.4.1.** Under Assumptions 3.4.1, 3.4.2, 3.4.3, 3.4.4, 3.4.5, $\hat{\theta} = (\Phi_{t-1}^T \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t))^{-1} (\Phi_{t-1}^T \Pi_{t-1} C_t)$ is a consistent estimator for $\theta$ defined in Equation (3.29).

The proof follows from Proposition B.1.1 in Appendix B.1. The following lemmas show that the assumptions in Appendix B.1 for Proposition B.1.1 hold. We first show Assumption B.1.1 holds,

**Lemma 3.4.1.** $E[Y''_i] = 0$, $\forall i$.

*Proof:* See Appendix B.2.1

We next show Assumption B.1.2 holds, which states that the mean of the noise in the observation of the explanatory variables is zero.

**Lemma 3.4.2.** $E[X''_{ij}] = 0$, $\forall i, j$.

*Proof:* See Appendix B.2.2

We next show Assumption B.1.3 holds, which states that the instrumental variables are uncorrelated with the noise in the observations of the response variable.
Lemma 3.4.3. \( \text{Cov}[Z_{ij}, Y''_i] = 0, \quad \forall i, j. \)

Proof: See Appendix B.2.3.

We define \( e_i \) as a column vector of zeros with a one at the \( i \)’th position. We next show Assumption B.1.4 holds,

Lemma 3.4.4. \( \text{Cov}[Z_{ij}, X''_{il}] = 0, \quad \forall i, j, l. \)

Proof: See Appendix B.2.4.

Finally, Assumption B.1.5 holds by Assumption 3.4.2 and Assumptions B.1.6, B.1.7, and B.1.8 follow trivially from Assumptions 3.4.2, 3.4.3, 3.4.4, and 3.4.5 by the law of large numbers (see Appendix B.1). Therefore Proposition B.1.1 applies. \( \square \)

One interesting comment is that this proof holds even if the discount factor \( \gamma = 1 \).

However, for Assumption 3.4.1 to hold when \( \gamma = 1 \), it is not hard to see that a constant basis function cannot be used because \( (\Phi_{t-1} - \gamma \Phi_t) \) would not have full column rank.

3.4.6 Equivalence of Instrumental Variable Bellman Error Minimization and Projected Bellman Error Minimization

In Section 3.4.3 we summarized least-squares Bellman error minimization (Equation (3.26)) and instrumental variables Bellman error minimization (Equation (3.27)).

In Section 3.4.4 we summarized least-squares projected Bellman error minimization (Equation (3.30)) and instrumental variables projected Bellman error minimization (Equation (3.31)). It turns out instrumental variables Bellman error minimization, least-squares projected Bellman error minimization, and instrumental variables projected Bellman error minimization are equivalent.
Theorem 3.4.1. Under Assumption 3.4.1, the following policy evaluation algorithms are equivalent:

**Instrumental Variables Bellman Error Minimization (LSTD)**
\[
\hat{\theta} = [(\Phi_{t-1})^T(\Phi_{t-1} - \gamma \Phi_t)]^{-1}(\Phi_{t-1})^T C_t,
\]

**Least-Squares Projected Bellman Error Minimization (Least-Squares Fixed Point Approx.)**
\[
\hat{\theta} = \left[(\Pi_{t-1}(\Phi_{t-1} - \gamma \Phi_t))^T(\Pi_{t-1}(\Phi_{t-1} - \gamma \Phi_t))\right]^{-1}(\Pi_{t-1}(\Phi_{t-1} - \gamma \Phi_t))^T \Pi_{t-1} C_t,
\]

**Instrumental Variables Projected Bellman Error Minimization**
\[
\hat{\theta} = [(\Phi_{t-1})^T \Pi_{t-1}(\Phi_{t-1} - \gamma \Phi_t)]^{-1}(\Phi_{t-1})^T \Pi_{t-1} C_t.
\]

*Proof: See Appendix B.3*

### 3.4.7 On-Policy Versus Off-Policy

For evaluating a fixed policy, [94] proves that off-policy TD(\(\lambda\)) algorithms with a linear function approximation of the value function may not converge. In this case, off-policy means that the distribution of the states visited during a single infinite trajectory is not equal to the distribution of the states visited if we followed the fixed policy of interest. For a fixed policy, [94] gives conditions under which on-policy temporal-difference learning will converge to the true value function projected onto the space of value function approximations (with respect to a weighted norm).

[10] gives conditions for on-policy policy evaluation based on Bellman error minimization to converge to a fixed value function when using a linear model for the value function. [53] explains that on-policy LSTD biases the value function and may do a very poor job of fitting the value function at states that are rarely visited. Another major disadvantage of on-policy is that, if the policy does not explore enough states, Assumption 3.4.1 may not hold. An important point to keep in mind is that the
value of the greedy policy determined by the final value function approximation may be significantly different from the true value function.

### 3.5 Direct Policy Search

An alternative to Bellman error minimization for finding the regression vector $\theta$ is direct policy search. As before, we consider policies parameterized by $\theta$ of the form,

$$X^\pi(S_t|\theta) = \arg\max_x [C(S_t, x) + \gamma \phi(S^x_t)^T \theta],$$

where the post-decision value function $V(S^x)$ has been replaced by the linear model $\phi(S^x)^T \theta$. The goal of dynamic programming is to find a value function which satisfies Bellman’s equation; the optimal post-decision value function easily translates into an optimal policy which maps a state to an action (this may not be true for pre-decision value functions). Unlike policy iteration or value iteration, the objective of direct policy search is not necessarily to find a value function that is close to the true value function (with respect to some norm); our objective is to find a value of $\theta$ for which the policy $X^\pi(s|\theta)$ performs well. Additionally, we only need to consider features which are a function of the decisions; for this reason, the “value function approximation” is typically much simpler than what is required if we use Bellman error minimization. The challenge for direct policy search arises as the dimension of $\theta$ grows; randomly trying different values of $\theta$ is highly inefficient. However, direct policy search can use classic stochastic optimization algorithms to sequentially choose policies to simulate.
3.5.1 The Knowledge Gradient for Direct Policy Search

Our objective is to find a value of $\theta$ which solves the following stochastic optimization problem,

$$
\max_{\theta} V^\pi(S_0),
$$

(3.35)

given the policy $X^\pi(S_t|\theta)$. For a fixed value of $\theta$ we can obtain a noisy observation of the objective in Equation (3.35) by simulating $\hat{V}^\pi(S_0) = C_0(S_0, X^\pi(S_0|\theta)) + \gamma_1 C_1(S_1, X^\pi(S_1|\theta)) + \gamma_2 C_2(S_2, X^\pi(S_2|\theta)) + \cdots$. We can sequentially simulate the value for many different values of $\theta$ before determining which value of $\theta$ gives the best policy, $X^\pi(S_t|\theta)$. Unfortunately, the optimization problem given by Equation (3.35) is typically non-convex and non-separable. When the dimension of $\theta$ is small, the knowledge gradient for continuous parameters (KGCP) policy has been shown to work well for efficiently optimizing $\theta$ (see [82]).

The KGCP policy for optimizing $\theta$ combines a model of $\mu(\theta) = V^\pi(S)$ with a criterion which chooses the next value of $\theta$ for which a noisy observation of $\mu(\theta)$ will be simulated. In particular, the objective $\mu(\theta)$ is modeled using Gaussian process regression which can be viewed as a linear smoother. The KGCP quantifies how much we expect the maximum of the objective to increase by getting an additional noisy observation of $\mu(\theta)$ at a particular value of $\theta$. More formally, we let $\mathcal{F}^n$ be the sigma-algebra generated by $\theta^0, \ldots, \theta^{n-1}$ and the corresponding noisy observations of $\mu(\theta^0), \ldots, \mu(\theta^{n-1})$. $\mu^n(\theta)$ is the updated Gaussian process regression function after $n$ observations (see [?]). The KGCP is defined as

$$
\bar{\nu}^{KG,n}(\theta) \triangleq \mathbb{E} \left[ \max_{i=0,\ldots,n} \mu^{n+1}(\theta^i) \bigg| \mathcal{F}^n, \theta^n = \theta \right] - \max_{i=0,\ldots,n} \mu^n(\theta^i)|_{\theta^m=\theta}.
$$

In the Gaussian process regression framework, $\mu^{n+1}(\theta)$ given $\mathcal{F}^n$ is normally distributed for each value of $\theta$, and the KGCP can be calculated exactly (see [?]). The KGCP can be viewed as a generalization of the expected improvement criterion from
to the case with noisy observations (see [?]). The next sampling decision will be chosen to maximize the KGCP,

$$\theta^* \in \arg \max_{\theta} \bar{\nu}^{KG,n}(\theta).$$

After $N$ observations, the implementation decision (the value of $\theta$ we believe is best) can be chosen by maximizing the regression function,

$$\theta^* \in \arg \max_{\theta} \mu^N(\theta).$$

One additional challenge for using direct policy search is determining the feasible domain for $\theta$; the domain of $\theta$ is typically restricted to a hypercube or simplex, because a true global search over all of $\mathcal{R}^k$ without any structure is typically an arbitrarily hard problem even with smoothness assumptions. The value of $\theta$ which maximizes Equation (3.35) produces the best policy within the class of polices, $X^\pi(S_t|\theta)$. Direct policy search has the potential to choose the best $\theta$ of any algorithm when choosing a policy $X^\pi(S_t|\theta)$, although in practice there is always a limited budget (primarily due to time) of how many policies we can simulate.

### 3.6 Numerical Experiments

Our main objective is to compare approximate policy iteration (API) with least-squares Bellman error minimization to API with instrumental variables Bellman error minimization to see if instrumental variables add value in practice. We first compare the algorithms on discretized benchmark problems with known solutions so we can report how well they perform relative to optimal. Additionally we run direct policy search on the discretized benchmark problems to see if we can find an even better pol-
Finally, we run approximate policy iteration on a problem with a state consisting of five continuous dimensions and actions consisting of five continuous dimensions.

### 3.6.1 Creating Benchmark Problems

We first consider a finite, discretized state and action space with a fixed probability transition matrix. One solution technique for finding the exact solution is value iteration (see [72]). $V^0(S)$ is initialized to a constant for all $S$, and at each iteration, $n$, the algorithm updates the values of each state,

$$V^n(s) = \max_x \{ C(s, x) + \gamma \sum_{s'} V^{n-1}(s') P(s'|s, x) \}, \forall s \in S.$$  \hspace{1cm} (3.36)

The algorithm will converge to the true value function of the optimal policy which satisfies Bellman’s equation,

$$V(S) = \max_x \{ C(S, x) + \gamma \mathbb{E}[V(S'(S, x))|S] \}. \hspace{1cm} (3.37)$$

We discretized the state space in the benchmark test problems and then created fixed probability transition matrices for the exogenous information process in order to create a true discrete process (this is different than simply simulating a continuous process and then discretizing as you progress).

In Table 3.1 we summarize a list of the benchmark problems described in Section 3.3 with exact solutions. “Full” refers to the problem shown in Figure 3.1 with energy from wind and the grid serving a load. “BA” refers to a battery arbitrage problem without wind or a load, where you buy and sell electricity from the grid using storage. We include how finely each state variable is discretized (the size of the state space for a particular problem is the product of each of the discretization levels). We then list the wind capacity divided by the load, the storage capacity divided by the load over an hour, the round trip efficiency (RTE) of the storage device, and
the max charge and discharge rate of the storage device. For example, C/10 means the storage device can be fully charged or discharged in 10 hours. The transition matrix of the electricity prices was fit using the PJM Western Hub real time prices (with and without time of day). The transition matrix of the load was fit using the load of the PJM Mid-Atlantic Region (with time of day). The transition matrix for the wind was fit using data from wind speeds near the Sweetwater Wind Farm. For Problems 1 – 16 the state space is resource level, wind energy, and electricity price, \( S_t = (R_t, E_t, P_t) \) (time and demand are fixed). In these problems, the load is held constant in order to keep the benchmark problems computationally tractable (exact value iteration, even for this simplified problem, requires approximately 2 weeks on a 3Ghz processor). Later, we demonstrate the approximate algorithms on problems with stochastic, time-dependent loads. For Problems 17 – 20, the state space is the time-of-day, \( \tau_t \), (1-96 corresponding to fifteen minute intervals in a day), the resource level, and the electricity price, giving us the state variable \( S_t = (\tau_t, R_t, P_t) \). \( \Delta t \) is fixed to fifteen minutes for all the problems. We use a discount factor, \( \gamma = .999 \). We found that discount factors of \( \gamma = .99 \) or smaller produce policies that are relatively myopic, and do not allow us to hold energy in storage for extended periods.

### 3.6.2 Comparing to the Benchmark

In order to choose how long to run the inner policy evaluation loop and outer policy improvement loop (see Figure 3.4), we ran approximate policy iteration using instrumental variables Bellman error minimization several times on one of the problems. For the test problems, we found most of the improvement has occurred before \( M = 30 \) policy improvements and policy evaluations of length \( N = 5000 \).

In Figure 3.5 we compare approximate policy iteration with instrumental variables Bellman error minimization, approximate policy iteration with least-squares Bellman error minimization, and direct policy search based on KGCP (described in Section
Table 3.1: Set of benchmark problems specifying the type (Full or Battery Arbitrage), the number of discretization levels for time (1=steady state), resource, price, load (1=deterministic) and wind. The remaining columns specify average maximum wind divided by the load, storage capacity divided by hourly load, round trip efficiency (RTE), and the maximum charge/discharge rate (C/10 means it takes hours to charge/discharge).

In addition, we show the performance of the myopic policy which discharges the battery as quickly as possible and then leaves it empty. The value of the myopic policy is still positive due to the wind power as well as the initial energy in the battery. In Figure 3.5 approximate policy iteration with instrumental variables Bellman error minimization and least-squares Bellman error minimization use quadratic basis functions, and we run each algorithm 100 times. For each run of the algorithms, the final policies produced by each algorithm are then evaluated on the same sample path, $\omega \in \Omega$, where $\omega$ is generated from the discretized exogenous information process. We
then record the average percent of optimal and the standard deviation of the average percent of optimal across the 100 runs. The average percentage of optimal for a policy $\pi$ is computed as

$$\text{% of optimal} = \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \frac{\hat{F}^\pi(\omega)}{V^*(S_0(\omega))},$$

where $\omega$ is a sample path of the randomness in the state transitions, and $S_0(\omega)$ is the starting state which has been randomly generated from a uniform distribution. $\hat{F}^\pi(\omega)$ is a realization of value of the policy $\pi$ run on the sample path $\omega$, starting at the state $S_0(\omega)$, and $V^*(S_0(\omega))$ is the true value of the optimal policy for state $S_0(\omega)$ which is computed using Equation (3.36). We ran the approximate policy iteration with other sets of basis functions (first-order, third-order, fourth-order), but quadratic basis functions performed the best (see Appendix B.4).

When we perform direct policy search using KGCP, we budget ourselves to simulating 50 sequentially chosen policies, after which the KGCP algorithm must choose what it believes to be the best policy. This is repeated 100 times and the average percent of optimal and standard deviation of the average percent of optimal are given in Figure 3.5. Direct policy search produced solutions that were on average 91.8 percent of optimal, and were always at least 70 percent of optimal, for problems 1 through 16. One interesting observation is that direct policy search performed at least 70% of optimal for problems 1 through 16, which suggests direct policy search is robust. In particular, direct policy search did much better on many of the problems and should only improve if the algorithm is allowed to run longer (although the algorithm becomes very time consuming). However, direct policy search quickly becomes intractable as the number of basis functions increases. Choosing the search domain for direct policy search is another significant complication as the number of basis functions increases. We suggest using approximate policy iteration to find good values of the regression parameters, and then use direct policy search to improve the policy in the region of the fitted regression parameters.
Figure 3.5: Performance as a percent of the benchmark optimization solution using API with instrumental variables, least-squares API, a myopic policy and direct policy search.

In Figure 3.6 we show a sample path of a policy produced by approximate policy iteration on Problem 1 in Table 3.1. We see that the resource is charged when electricity prices are low and discharged when electricity prices are high. We also note that the battery fully discharges (down to 20 percent) relatively infrequently.

One way to reduce the number of basis functions used by the algorithms is to ignore dimensions of the post-decision state when constructing the value function approximation. In Figure 3.7 we show the results using three value function approximations: 1) resource level, wind power and electricity price, 2) resource level only, and 3) resource level and electricity price. We observe that using the resource level alone for the domain of the post-decision value function appears to do quite poorly.
Figure 3.6: We plot a 10 day sample path of a policy produced by approximate policy iteration with instrumental variables Bellman error minimization using quadratic basis functions on Problem 1. (a) We plot the electricity price and resource level. (b) We plot a histogram of the resource level.

for most problems. Using both resource level and electricity price appears to do fairly well overall, although using all the dimensions of the state variable appears to do the best. For certain problems, it may actually be advantageous to leave variables out of the state space in order to have a smaller number of weights to estimate for the value function.
Figure 3.7: The algorithms use quadratic basis functions. We show the percentage of optimal along with 95% confidence intervals for the average percentage of optimal for Bellman error minimization using instrumental variables (IV) when only certain dimensions of the post-state are included in the post-state value function approximation.

### 3.6.3 A Continuous Problem

In this section we consider problems with continuous states as well as a larger state space. We compare both approximate policy iteration algorithms on the continuous problems described in Table 3.2 although an optimal solution will no longer be available. These problems now have continuous states and continuous actions and the state transitions correspond the models in Section 3.2. The electricity prices and loads are now time-dependent and stochastic for Problems 1-3. Problems 4-10 are continuous steady-state problems.
Table 3.2: Parameter settings for problems with continuous states. Problems 1-3 have time-dependent stochastic loads and electricity prices. Problems 4-10 are steady-state.

Figure 3.8 shows that least-squares Bellman error minimization performs very poorly and the instrumental variables do indeed add value. Although all the dimensions of the state variable and action space are difficult to visualize, in Figure 3.9 we use a policy produced by approximate policy iteration with instrumental variables Bellman error minimization to show the electricity price and the percent of the storage which is full on one particular sample path. We can see that the policy tends to start charging the battery at night when electricity prices are low and then discharges the battery throughout the day when electricity prices are higher. Approximate policy iteration with instrumental variables Bellman error minimization is a promising scalable algorithm which is designed for problems where the states are continuous and the transition probabilities are unknown.

3.7 Conclusion

We have created a series of difficult benchmark problems that arise in a class of energy storage problems that represent a difficult algorithmic challenge with respect to identifying good control policies. The idea is to create (discretized) problems
Figure 3.8: We plot the average objective of both approximate policy iteration algorithms on the continuous problems shown in Table 3.2.

Figure 3.9: A sample path of the electricity spot price and resource level. The vertical lines correspond to midnight.

that can be solved optimally, use these benchmark problems to evaluate scalable approximation algorithms which can then be used on more complex problems.

We quickly found that considerable care has to be used in creating benchmark problems. For example, we found that using a discount factor of .99 produced problems where myopic policies worked well. As a result, substantial errors in the value function approximation still produced results that were within a few percent of op-
timal. The same result occurred if the battery was small relative to the amount of available wind energy. Our problems were chosen both to model realistic systems, but also to provide an algorithmic challenge, as evidenced by the poor performance of a myopic policy.

We compared three strategies based on Bellman error minimization (classical least squares approximate policy iteration, and variants that use instrumental variables and projected Bellman error minimization), to one based on direct policy search. This work produced several surprising results. First, we found that the performance using instrumental variables and projected Bellman error were not just similar - they were the same, an observation that led to a mathematical proof of this result. Second, we were somewhat surprised and impressed at how much better Bellman error minimization performed using instrumental variables, a technique that does not seem to be widely used in the reinforcement learning literature and virtually unknown in other ADP communities. But third, we were also surprised and a bit disappointed at how poorly Bellman error minimization, even with instrumental variables, worked relative to both the optimal solution as well as the performance of direct policy search.

This research suggests that direct policy search should be used, perhaps in conjunction with approximate policy iteration. The challenge is that in its derivative-free form, it does not scale easily to large numbers of parameters. This may be a major limitation in time-dependent applications where we may need to estimate a different set of parameters for each time period.
Chapter 4

Portfolio Selection and Covariance Matrix Estimation using an Errors-in-Variables Factor Model with an Application to the PJM Electricity Market

4.1 Introduction

PJM is an independent and profit-neutral regional transmission organization that is responsible for the grid operation and reliability for most of the northeast United States (see [69]). PJM sets the locational marginal prices (LMP’s) taking into account the sizes and locations of the electricity generation plants and loads, along with transmission congestion. The day-ahead PJM market determines the day-ahead electricity price at each node for each hour of the day. All bids and offers for electricity (including virtual bids from hedgers or speculators) are submitted by noon the
day-ahead. At 4 pm PJM determines the next day’s day-ahead LMP’s. Virtual bids that clear in the day-ahead market are purely financial contracts, typically referred to as “incs” (increments) or “decs” (decrements) for selling or buying electricity in the day-ahead market, respectively. The payoff of these contracts is the difference between the real-time and day-ahead price, minus a reserve fee.

PJM auctions off financial transmission rights (FTR’s) monthly and annually. FTR’s force the contract owner into a sequence of cash flows based on the difference in the congestion prices of two nodes in the PJM network. In addition, PJM sells up-to-congestion contracts, which can be used to hedge (or bet) whether the congestion price between nodes will increase or decrease from the day-ahead market to the real-time market. Without hedging, these contracts can be very risky due to the heavy-tailed real-time electricity prices. In this paper, we use covariance matrices and portfolio theory to control the risk of the portfolio of financial contracts.

Covariance matrices have been fundamental to choosing diversified portfolios of assets dating back to Harry Markowitz (see [60]). However, the sample covariance matrix is often ill-conditioned and typically not appropriate for Markowitz portfolio theory. Additionally, when calculating the covariance matrix of non-stationary random variables, estimation with data that is too old may not effectively capture the current market conditions. The use of factor models to estimate large covariance matrices of asset returns dates back to William Sharpe (see [85]). The most well-known factor models for capital assets are the capital asset pricing model, which uses excess market returns as the only factor (see [84]), and the Fama-French 3-factor model (see [30]), which uses three factors: excess market returns; small minus big market capitalization; high minus low book-to-market ratio.

This paper makes the following contributions: (1) We propose an errors-in-variables extension of the capital asset pricing model (CAPM) that accounts for errors in the observations of the market returns. (2) We show how the errors-in-
variables version of CAPM can be used to estimate the covariance matrix of the returns of assets, especially when we use relatively short histories so that we can capture short-term changes in volatility. (3) We evaluate portfolios optimized using the covariance matrices produced by CAPM, errors-in-variables CAPM, and the Fama-French three-factor model on the S&P 100. (4) We modify the Markowitz portfolio framework for buying and selling electricity in the PJM day-ahead and real-time electricity markets. (5) We evaluate portfolios optimized using covariance matrices produced by CAPM and errors-in-variables CAPM in the PJM electricity market.

This paper is organized as follows. Section 4.2 summarizes how factor models are used to estimate covariance matrices. We present an errors-in-variables factor model which assumes there are errors in the observations of the market returns, which can be used to construct a covariance matrix. Section 4.3 summarizes the typical Markowitz portfolio which requires a covariance matrix for the returns of the assets. We present an extension of the Markowitz portfolio which can be used for choosing portfolios of electricity contracts. In Section 4.4 we show that the covariance matrix constructed with the errors-in-variables factor model has benefits for choosing portfolios of U.S. equities as well as portfolios of electricity contracts.

4.2 Estimating Covariance Matrices

In this section, we show how CAPM and factor models can be used to estimate covariance matrices. We then propose an extension to CAPM which we call the errors-in-variables CAPM, which allows for errors in the observations of the market returns. We show how the errors-in-variables CAPM can be used to estimate the covariance matrix of the returns of assets, even if we use relatively short histories.
4.2.1 The Capital Asset Pricing Model

The capital asset pricing model (CAPM) described in [84], [55], and [6] relates the excess return of an asset to the excess return of the market portfolio under certain market equilibrium conditions. Assume we have \( N \) assets. Let \( Y_j \) be the excess return of asset \( j \) where \( j = 1, ..., N \), and let \( Y^m \) be the excess return of the market. Assuming each investor invests in a mean-variance optimal portfolio, CAPM states

\[
Y_j = \alpha_j + \beta_j Y^m + \epsilon_j, \quad (4.1)
\]

where \( E[\epsilon_j] = 0 \), and is independent from \( Y^m \). CAPM concludes that \( \alpha_j = 0 \), for \( j = 1, ...N \). Assuming \( \alpha_j \) and \( \beta_j \) are deterministic, it follows from Equation (4.1) that

\[
\text{Cov}[Y_j, Y^m] = \beta_j \text{Var}[Y^m]. \quad (4.2)
\]

Replacing the true covariance with the sample covariance, this gives an equation for estimating \( \beta_j \) which is equivalent to the least-squares estimate of \( \beta_j \),

\[
\hat{\beta}_j = \frac{\text{Cov}[Y_j, Y^m]}{\text{Var}[Y^m]}. \quad (4.3)
\]

One powerful application of CAPM is building covariance matrices, where we can write

\[
\text{Cov}[Y_i, Y_j] = \text{Cov}[\alpha_i + \beta_i Y^m + \epsilon_i, \alpha_j + \beta_j Y^m + \epsilon_j]
\]

\[
= \text{Cov}[\beta_i Y^m + \epsilon_i, \beta_j Y^m + \epsilon_j]
\]

\[
= \text{Cov}[\beta_i Y^m, \beta_j Y^m] + \text{Cov}[\beta_i Y^m, \epsilon_j] + \text{Cov}[\beta_j Y^m, \epsilon_i] + \text{Cov}[\epsilon_i, \epsilon_j]
\]

\[
= \beta_i \beta_j \text{Var}[Y^m] + \beta_i 0 + \beta_j 0 + 0
\]

\[
= \beta_i \beta_j \text{Var}[Y^m] + \text{Cov}[\epsilon_i, \epsilon_j]. \quad (4.4)
\]
It is typically assumed $\text{Cov}[\epsilon_i, \epsilon_j] = 0, \ \forall i \neq j$ (see [85]). In matrix notation with $N$ assets,

$$
\begin{align*}
\begin{bmatrix}
Y
\end{bmatrix}_{N \times 1} &= \begin{bmatrix}
\alpha
\end{bmatrix}_{N \times 1} + \begin{bmatrix}
\beta
\end{bmatrix}_{N \times 1} \begin{bmatrix}
Y^m
\end{bmatrix}_{1 \times 1} + \begin{bmatrix}
\epsilon
\end{bmatrix}_{N \times 1}.
\end{align*}
\tag{4.5}
$$

and

$$
\text{Cov}[Y] = \beta \beta^T \text{Var}[Y^m] + \text{Var}[\epsilon],
\tag{4.6}
$$

where $\text{Var}[\epsilon]$ is diagonal by assumption. The estimator given by Equation (4.6) is always positive semidefinite and guaranteed to be positive definite when $\text{Var}[\epsilon]$ has full rank. This is a major advantage over the sample covariance matrix which can easily be rank deficient. However, it is well known that difficulties may arise when using Equation (4.3) to estimate $\beta$. [3] says that, in practice, fundamental changes in an asset as well as specific events that affect an asset may distort historical values of $\beta$. For this reason we would like to create a more robust version of CAPM that allows for errors in the market returns, which also allows us to use more recent histories that capture short term changes in volatility. First, we summarize factor models (CAPM is a single-factor model).

### 4.2.2 Factor Models

A factor model decomposes the excess returns of the $N$ assets, $\begin{bmatrix}
Y
\end{bmatrix}_{N \times 1}$, into linear combinations of $K$ factors plus noise (see [30], [32], [31]),

$$
\begin{align*}
\begin{bmatrix}
Y
\end{bmatrix}_{N \times 1} &= \begin{bmatrix}
B
\end{bmatrix}_{N \times K} \begin{bmatrix}
F
\end{bmatrix}_{K \times 1} + \begin{bmatrix}
\epsilon
\end{bmatrix}_{N \times 1}.
\end{align*}
$$
Assuming $B$ is deterministic and each of the factors is uncorrelated with $\epsilon$, the variance of the excess returns can be written,

\[
\text{Cov}[Y] = \text{Cov}[BF + \epsilon] = \text{Cov}[BF] + \text{Cov}[\epsilon] = BCov[F]B^T + \text{Cov}[\epsilon].
\]

The covariance matrix of the factors, $\text{Cov}[F]$, can be approximated with the sample covariance matrix. As in [85], the covariance matrix of the residuals, $\text{Cov}[\epsilon]$, can be approximated with a diagonal matrix. In the Fama-French 3-factor model, the excess market return, small minus big market capitalization, and the high minus low book-to-market ratio are the three factors used for estimating the covariance matrix of the returns of US equities (see [30]).

### 4.2.3 CAPM with Instrumental Variables

We now formulate our model differently than Equation (4.1). We first assume the true excess return of asset $j$ has a linear relationship with the true excess market return, where $j = 1, \ldots, N$,

\[
Y_j = \alpha_j + \beta_j Y^m.
\]  

(4.7)

We next assume we do not observe $Y_j$, the true excess return of asset $j$; we only observe $Y'_j$, which is equal to the true excess return of asset $j$ plus the noise $Y''_j$,

\[
Y'_j = Y_j + Y''_j.
\]  

(4.8)

Analogously, we next assume we do not observe $Y^m$, the true excess return of the market; we only observe $Y^{m'}$, which is equal to the true excess return of the market
plus the noise $Y^{m''}$,

$$Y^{m'} = Y^{m} + Y^{m''}. \quad (4.9)$$

To summarize, at each time period, we are only able to observe $Y^{m'}, Y_1', Y_2', ..., Y_N'$. In Section 4.2.1 we assumed we could observe the true excess return of the market; now we assume we can only observe a noisy version of the excess return of the market.

Using Equations (4.7), (4.8), and (4.9) we are able to derive an analogous estimator to Equation (4.6) for estimating the covariance matrix of assets,

$$\text{Cov}[Y_i', Y_j'] = \text{Cov}[\alpha_i + \beta_i Y^{m} + Y_i'', \alpha_j + \beta_j Y^{m} + Y_j'']$$

$$= \text{Cov}[\beta_i Y^{m} + Y_i'', \beta_j Y^{m} + Y_j'']$$

$$= \text{Cov}[\beta_i Y^{m}, \beta_j Y^{m}] + \text{Cov}[\beta_i Y^{m}, Y_j''] + \text{Cov}[\beta_j Y^{m}, Y_i''] + \text{Cov}[Y_i'', Y_j'']$$

$$= \beta_i \beta_j \text{Var}[Y^{m}] + \beta_i \text{Cov}[Y^{m}, Y_j''] + \beta_j \text{Cov}[Y^{m}, Y_i''] + \text{Cov}[Y_i'', Y_j'']$$

$$= \beta_i \beta_j \text{Var}[Y^{m}] + \text{Cov}[Y_i'', Y_j''],$$

where we assumed $Y^{m}$ and $Y_j''$ were uncorrelated for each asset $j = 1, ..., N$. This can be written in matrix form as

$$\text{Cov}[Y'] = \beta \beta^T \text{Var}[Y^{m}] + \text{Var}[Y'']. \quad (4.10)$$

In CAPM described in Section 4.2.1, estimating $\text{Var}[Y^{m}]$ is straightforward using the sample variance, because we assumed we could observe $Y^{m}$. In this section, we assume we only observe $Y'^m$ and hence we must first estimate $Y^{m}$ before we can estimate $\text{Var}[Y^{m}]$.

**Estimating $\beta$**

The method of instrumental variables can be used to estimate $\beta$ in the errors-in-variables model given in Equations (4.7), (4.8), and (4.9) (see [27]; [47]; [87]). An
instrumental variable, $Z$, should be correlated with the market return, $Y^m$, but uncorrelated with the errors in the observations of $Y^m$ and $Y_j$. We assume we have $M$ observations of the returns of $N$ different assets. The following equation gives the instrumental variables estimate of $\beta_j$ for each asset $j = 1, ..., N$:

$$Z^T Y^m \hat{\beta}_j = Z^T Y^m_{j}.$$  \hfill (4.11)

If $\lim_{M \to \infty} \frac{1}{M} Z^T Y_j^{m''} = 0$, $\lim_{M \to \infty} \frac{1}{M} Z^T Y_j'' = 0$, and $\lim_{M \to \infty} \frac{1}{M} Z^T Y^m = \Sigma$, then Equation (4.11) yields a consistent estimate for $\beta_j$,  

$$Z^T (Y^m + Y^{m''}) \hat{\beta}_j = Z^T (Y^m \beta_j + Y_j''),$$

$$(Z^T Y^m + Z^T Y^{m''}) \hat{\beta}_j = Z^T Y^m \beta_j + Z^T Y_j''.$$

Now, taking the limit as $M$ goes to infinity,

$$\lim_{M \to \infty} \frac{1}{M} \left( Z^T Y^m + Z^T Y^{m''} \right) \hat{\beta}_j = \lim_{M \to \infty} \frac{1}{M} \left( Z^T Y^m \beta_j + Z^T Y_j'' \right),$$

$$\lim_{M \to \infty} \frac{1}{M} Z^T Y^m \hat{\beta}_j = \lim_{M \to \infty} \frac{1}{M} Z^T Y^m \beta_j,$$

$$\Sigma \left( \lim_{M \to \infty} \hat{\beta}_j \right) = \Sigma \beta_j,$$

$$\lim_{M \to \infty} \hat{\beta}_j = \beta_j.$$

For the numerical work in this paper, we use the returns of an equal-weighted portfolio as a reasonable instrumental variable.

**Estimating $Y^m$**

We first rewrite Equations (4.7), (4.8), (4.9) using an additional index for each of the $M$ observations. Letting $i = 1, ..., M$ be the index for the observation and $j = 1, ..., N$ be the index for the asset, we can write
\[ Y_{ij} = \alpha_j + \beta_j Y^m_i, \quad (4.12) \]

\[ Y'_{ij} = Y_{ij} + Y''_{ij}, \quad (4.13) \]

\[ Y^{m'}_i = Y^m_i + Y''^m_i. \quad (4.14) \]

In order to obtain estimates for \( Y^m \) in closed form, we make the following assumptions:

**Assumption 4.2.1.** \( \{Y'^{m''}_i\}_{i=1}^M \) are i.i.d. and

\[ Y'^{m''}_i \sim \mathcal{N}(0, (\sigma^m)^2), \quad i = 1, \ldots, M. \]

**Assumption 4.2.2.** \( \{Y''_{ij}\}_{i=1}^M \) are i.i.d. and

\[ Y''_{ij} \sim \mathcal{N}(0, (\sigma_j)^2), \quad i = 1, \ldots, M, j = 1, \ldots, N. \]

**Assumption 4.2.3.**

\[ \sigma^m = \sigma_1 = \cdots = \sigma_N. \]

**Assumption 4.2.4.**

\[ \alpha_1 = \cdots = \alpha_N = 0. \]
In order to estimate the market returns, \( Y_1^m, \ldots, Y_M^m \), we first write the log-likelihood:

\[
l(Y_1^m, \ldots, Y_M^m) = \sum_{i=1}^{M} \left[ -\frac{1}{2} \ln(2\pi(\sigma^m)^2) - \frac{(Y_i^m - Y_i^m)^2}{2(\sigma^m)^2} \right] + \sum_{j=1}^{N} \sum_{i=1}^{M} \left[ -\frac{1}{2} \ln(2\pi(\sigma_j)^2) - \frac{(Y_{ij}^m - Y_{ij})^2}{2(\sigma_j)^2} \right]
\]

Now, maximizing this expression with respect to \( Y_1^m, \ldots, Y_M^m \) we get,

\[
\arg \max_{Y_1^m, \ldots, Y_M^m} \left[ \sum_{i=1}^{M} -\frac{(Y_i^m - Y_i^m)^2}{2(\sigma^m)^2} \right] + \sum_{j=1}^{N} \sum_{i=1}^{M} -\frac{(Y_{ij}^m - Y_{ij})^2}{2(\sigma_j)^2} \]

Now taking the derivative with respect to \( Y_i^m \) and setting it to zero, we get

\[
2(Y_i^m - Y_i^m) + \sum_{j=1}^{N} 2(Y_{ij}^m - \beta_j Y_i^m) = 0,
\]

\[
Y_i^m - Y_i^m + \sum_{j=1}^{N} Y_{ij}^m - \sum_{j=1}^{N} \beta_j Y_i^m = 0,
\]

\[
Y_i^m + \sum_{j=1}^{N} \beta_j = Y_i^m + \sum_{j=1}^{N} Y_{ij}^m,
\]

\[
Y_i^m = \frac{Y_i^m + \sum_{j=1}^{N} Y_{ij}^m}{1 + \sum_{j=1}^{N} \beta_j}.
\] (4.15)

This equation gives an estimator for the true market excess return \( Y_i^m \) at each period \( i = 1, \ldots, M \). Note that the estimate of the scalar \( Y_i^m \) depends on the \( \beta \) for each asset as well as the observed excess return for each asset in period \( i \), \( Y_{i1}^m, \ldots, Y_{iN}^m \). This is
fundamentally very different from classical CAPM, because now the regression for each asset depends on the regression of all the other assets as well.

**Visualizing the Difference**

CAPM treats the market returns as noise-free, but our errors-in-variables CAPM explicitly assumes the market returns have noise. When there are errors in the explanatory variable, \([27]\) and \([47]\) show that least-squares linear regression typically yields an estimate of the slope which is too small in magnitude. This may correspond to the empirical observation that the CAPM estimate of \(\beta\) is typically too small in magnitude (see \([3]\)). In Figure 4.1(a) and 4.1(b) we show the CAPM estimates of \(\beta\) for Chevron and ExxonMobil using daily data over 50 days. The excess returns of the market are shown on the horizontal axis and the excess returns of Chevron and ExxonMobil are shown on the vertical axis, respectively. The residuals are drawn vertically because there is no noise in the market returns.

In Figure 4.2(a) and 4.2(b) we show the errors-in-variables CAPM estimates of \(\beta\) for Chevron and ExxonMobil. The excess returns of the market are shown on the horizontal axis and the excess returns of Chevron and ExxonMobil are shown on the y-axis, respectively. The residuals are no longer drawn vertically because there is noise in the market returns. The residuals are now estimated using the equations described in Sections 4.2.3 and 4.2.3. The estimates of \(\beta\) are now larger as predicted by the theory. In particular, if the estimates of \(\beta\) are too small in magnitude, we should expect the residuals to be correlated across assets. Figure 4.3 shows the residuals for Chevron and ExxonMobil using CAPM, and they appear to be positively correlated. Figure 4.3(b) shows the residuals for Chevron and ExxonMobil using the errors-in-variables CAPM. Visually, the residuals appear to be smaller and less correlated, as desired.
Figure 4.1: (a) The CAPM estimate of $\beta$ for Chevron. (b) The CAPM estimate of $\beta$ for ExxonMobil (c) The residuals for the fit of Chevron’s $\beta$. (d) The residuals for the fit of ExxonMobil’s $\beta$.

4.3 Portfolio Selection

[60] discusses the history of portfolio theory which started with the idea of diversification and evolved into the concept of mean-variance efficiency in [59]. In the presence of a risk-free asset and shorting, it is well known that the optimal Markowitz portfolio can be calculated in closed form. Below we briefly summarize the main points (see [14]).
Figure 4.2: (a) The errors-in-variables CAPM estimate of $\beta$ for Chevron. (b) The errors-in-variables CAPM estimate of $\beta$ for ExxonMobil. (c) The residuals for the fit of Chevron’s $\beta$. (d) The residuals for the fit of ExxonMobil’s $\beta$.

### 4.3.1 Markowitz with a Risk-free Asset

[59] and [85] describe a method for choosing a portfolio of assets in the presence of a risk-free asset. Let $\bar{r}$ be an $N \times 1$ column vector of the expected returns over one period of the $N$ risky assets, and assume the risk-free return over one period is $r_0$. Let $\bar{Y}$ be an $N \times 1$ column vector of the expected excess returns of the $N$ risky assets, $\bar{Y}_j = \bar{r}_j - r_0$, $j = 1, ..., N$. Let $\Sigma$ be the $N \times N$ covariance matrix of the returns of the $N$ risky assets over the next time period. Let $\alpha$ be an $N \times 1$ column vector where $\alpha_j$ represents the proportion of total wealth invested in asset $j$, $j = 1, ..., N$. 
Figure 4.3: (a) The residuals between Chevron and ExxonMobil using CAPM as shown in Figures 4.1(a) and 4.1(b) (b) The residuals between Chevron and ExxonMobil using the errors-in-variables CAPM as shown in Figures 4.2(a) and 4.2(b)

In addition, let $\alpha_0$ be the proportion of total wealth invested in the risk-free asset, and assume $\alpha_0 + \alpha_1 + \cdots + \alpha_N = 1$.

The expected return of the portfolio $\alpha$ after one period can be written

$$\mu(\alpha) = \alpha_0 r_0 + \alpha_1 \bar{r}_1 + \cdots + \alpha_N \bar{r}_N$$

$$= r_0 - r_0 (\alpha_0 + \alpha_1 + \cdots + \alpha_N) + \alpha_1 \bar{r}_1 + \cdots + \alpha_N \bar{r}_N$$

$$= r_0 + \alpha_1 (\bar{r}_1 - r_0) + \cdots + \alpha_N (\bar{r}_N - r_0)$$

$$= r_0 + \alpha^T \bar{Y}.$$

The variance of the portfolio $\alpha$ can be calculated using

$$\sigma^2(\alpha) = \alpha^T \Sigma \alpha.$$
Letting $A$ be a measure of risk aversion, the Markowitz portfolio is chosen by maximizing the expected return of the portfolio minus a penalty for risk,

$$\max_{\alpha} \left[ \mu(\alpha) - \frac{A}{2} \sigma^2(\alpha) \right] = \max_{\alpha} \left[ r_0 + \alpha^T \bar{Y} - \frac{A}{2} \alpha^T \Sigma \alpha \right].$$

Now, setting the gradient with respect to $\alpha$ equal to zero, we obtain

$$\bar{Y} - A \Sigma \alpha = \vec{0},$$
$$\alpha = \frac{1}{A} \Sigma^{-1} \bar{Y},$$

and $\alpha_0 = 1 - \alpha_1 - \cdots - \alpha_N$.

### 4.3.2 Without a Risk-free Asset

Without a risk-free asset, the expected return of the portfolio $\alpha$ over one period can now be written as

$$\mu(\alpha) = \alpha_1 \bar{r}_1 + \cdots + \alpha_N \bar{r}_N = \alpha^T \bar{r}.$$

We now require $\alpha_1 + \cdots + \alpha_N = 1$. The equation for the variance of the return of portfolio $\alpha$ is still

$$\sigma^2(\alpha) = \alpha^T \Sigma \alpha.$$

We can now formulate the problem of finding $\alpha$ as the following optimization problem:

$$\max_{\alpha} \left[ \mu(\alpha) - \frac{A}{2} \sigma^2(\alpha) \right]$$

subject to $\alpha_1 + \cdots + \alpha_N = 1$.
4.3.3 Extension for Electricity Markets

The relationship between the day-ahead and real-time electricity prices at the PJM hubs varies greatly by the hub and the hour of the day. To illustrate this, we plot the cumulative sum of the real-time prices minus the day-ahead prices ($ per MWh) for a fixed location and a fixed hour of the day. In Figures 4.4 and 4.5, we show the cumulative sum of these price differences in order to determine whether electricity is more expensive in the day-ahead or real-time market on average. In Figure 4.4, we see that, for the off-peak hour of 1 a.m., the relationship between the day-ahead and real-time prices vary greatly by the hub, and no general conclusion can be made about whether electricity is more expensive in the day-ahead or real-time market. In Figure 4.5, for the on-peak hour of 5 p.m., the dynamics are slightly different than the off-peak hour, but the conclusions are similar. Each hub has different dynamics, and the relationship between the day-ahead market and real-time market is complicated. For example, Figure 4.4 shows that for the New Jersey Hub at 1 a.m., electricity is cheaper in the day-ahead market on average. However, Figure 4.5 shows that for the New Jersey Hub at 5 p.m., on average, electricity is cheaper in the real-time market.

Next, we examine a single hub and plot the cumulative sum of the real-time price minus the day-ahead price by the hour of day. In Figure 4.6, we show the cumulative sum of the differences in day-ahead and real-time prices for each hour of the day, showing a systematic discrepancy by hour of day. This is even more evident at the New Jersey Hub shown in Figure 4.7. From 1 a.m. to 4 p.m., electricity is cheapest in the day-ahead market on average. However, from 5 p.m. to midnight, electricity is cheapest in the real-time market on average.

We will now attempt to model the difference in the day-ahead and real-time electricity price by the hour of the day and the hub. The number of assets, $N$, is now equal to the number of hub locations multiplied by 24. We will treat each hour of day
at each location as a separate asset. Let $P^{DA}$ be an $N \times 1$ vector of the day-ahead prices at each location and each hour of day. Let $\alpha$ by an $N \times 1$ vector where $\alpha_j$ represents the amount of electricity we buy at location and time $j$ on a fixed day, $j = 1, \ldots, N$. If $\alpha_j$ is negative, then we sell electricity in the day-ahead. We assume we start with one dollar and set the cost of buying electricity in the day ahead market to zero (self-financing),

$$\alpha_1 P_{1}^{DA} + \cdots + \alpha_N P_{N}^{DA} = \alpha^T P^{DA} = 0.$$ 

We enter these contracts the day before the electricity is actually transmitted. The next day we can liquidate our positions in the real-time market. Let $P^{RT}$ be the $N \times 1$ vector of the real-time prices at each location and each hour of day. $P^{RT}$ corresponds
to $P^{DA}$, although $P^{RT}$ is still unknown when we choose our portfolio $\alpha$. We will first assume $P^{DA}$ is known when we choose our portfolio $\alpha$.

Let $r$ be the $N \times 1$ vector of returns. The return $r_j$ can now be defined as simply $r_j = P_j^{RT} - P_j^{DA}, j = 1, \ldots, N$. Let $\bar{r}$ be the expected value of the return vector ($P^{DA}$ is known but $P^{RT}$ is random). The expected return of our portfolio can now be written as,

$$\mu(\alpha) = \alpha_1 \bar{r}_1 + \cdots + \alpha_N \bar{r}_N. \tag{4.17}$$

Let $\Sigma$ be the covariance matrix of the return vector $r$. The variance of the portfolio is

$$\sigma^2(\alpha) = \alpha^T \Sigma \alpha,$$
Figure 4.6: The cumulative sum of the real-time price minus day-ahead price for a particular hour of the day at the Western Hub. Early in the morning, electricity is cheapest in the day-ahead market on average. In the middle of the day, electricity is cheapest in the real-time market on average.

and we can choose our portfolio by maximizing the following objective,

$$\max_{\alpha} \left[ \mu(\alpha) - \frac{A}{2} \sigma^2(\alpha) \right]$$

subject to $\alpha^T P^{DA} = 0$. Using the definition of $\mu(\alpha)$ in Equation (4.17), this problem becomes

$$\max_{\alpha} \left[ \alpha^T \bar{r} - \frac{A}{2} \alpha^T \Sigma \alpha \right]$$

subject to $\alpha^T P^{DA} = 0$. The self-financing constraint that $\alpha_1 P^{DA}_1 + \cdots + \alpha_N P^{DA}_N = 0$ requires knowing the day-ahead prices, which are not known when choosing the portfolio. If all the day-ahead prices were equivalent at all locations and times, this constraint would be equivalent to $\alpha_1 + \cdots + \alpha_N = 0$. In reality, the day-ahead prices are not equivalent, but this is a reasonable approximation to get a portfolio.
Figure 4.7: The cumulative sum of the real-time price minus day-ahead price for each hour of the day at the New Jersey Hub.

that is almost self-financing. In reality, it means that the number of megawatt-hours we buy in the day-ahead market equals the number of megawatt-hours we sell in the day-ahead market. Overall, we are not making a directional bet about whether real-time electricity prices are going to be higher or lower than anticipated. In fact, these directional bets tend to have a very high variance due to the heavy-tails of the real-time prices. Overall our problem becomes,

\[
\max_{\alpha} \left[ \alpha^T \bar{r} - \frac{A}{2} \alpha^T \Sigma \alpha \right]
\]

subject to \( \alpha_1 + \cdots + \alpha_N = 0 \).

4.4 Numerical Experiments

We compare our covariance matrix estimation technique described in Section 4.2.3 with the CAPM covariance matrix estimation in Section 4.2.1. One of the typical
ways to evaluate a portfolio is to look at the Sharpe ratio, the annualized excess return divided by the annualized standard deviation. Additionally, the maximum draw-down gives an idea of how much a portfolio could potentially lose. The portfolio selection methods described in Section 4.3.1 require covariance matrices. We show the results of both covariance matrix estimation techniques for investing in U.S. equities and electricity in the PJM grid, in terms of how well they choose portfolios.

### 4.4.1 S&P 100

To see if the errors-in-variables version of CAPM (Equation (4.10)) has benefits over traditional CAPM (Equation (4.6)) for estimating covariance matrices, we evaluate their performance in terms of the quality of Markowitz portfolios produced by each method. We use four-week treasury bills as a proxy for the risky-free asset, ignoring transaction costs and allowing shorting.

Figure 4.8 shows wealth paths of various portfolios over the period of 2004 to 2011, starting from 1 dollar. The market portfolio is the S&P 500 index. The equal-weighted portfolio puts equal weight on each of the S&P 100 stocks, re-balancing daily. In this figure, CAPM refers to a Markowitz portfolio of the S&P 100 stocks re-balanced daily where the covariance matrix is estimated using Equation (4.6) with a calibration length of 5 days. The Fama-French method refers to a Markowitz portfolio of the S&P 100 stocks re-balanced daily where the covariance matrix is estimated using the Fama-French 3-factor model with a calibration length of 5 days. The errors-in-variables CAPM method refers to a Markowitz portfolio of the S&P 100 stocks re-balanced daily where the covariance matrix is estimated using Equation (4.10) with a calibration length of 5 days. In Table 4.1 we show the statistics of the various wealth paths of Figure 4.8. In this case, the errors-in-variables CAPM Markowitz portfolio has a higher Sharpe ratio than the traditional CAPM Markowitz portfolio.
Figure 4.8: Starting from one dollar, wealth paths different portfolios. A calibration length of 5 days was used for CAPM and errors-in-variables CAPM.

<table>
<thead>
<tr>
<th>Portfolio</th>
<th>$\mu$</th>
<th>$\sigma$</th>
<th>Sharpe Ratio</th>
<th>Max Drawdown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Riskfree</td>
<td>.019</td>
<td>.001</td>
<td>.000</td>
<td>.000</td>
</tr>
<tr>
<td>Market</td>
<td>-.004</td>
<td>.218</td>
<td>-.106</td>
<td>.565</td>
</tr>
<tr>
<td>Equal Weight</td>
<td>.038</td>
<td>.233</td>
<td>.080</td>
<td>.539</td>
</tr>
<tr>
<td>CAPM</td>
<td>.302</td>
<td>.258</td>
<td>1.099</td>
<td>.249</td>
</tr>
<tr>
<td>Fama-French 3-Factor</td>
<td>.140</td>
<td>.264</td>
<td>.460</td>
<td>.341</td>
</tr>
<tr>
<td>Errors-in-Variables CAPM</td>
<td>.377</td>
<td>.256</td>
<td>1.399</td>
<td>.341</td>
</tr>
</tbody>
</table>

Table 4.1: The portfolio statistics of the portfolios shown in Figure 4.8. $\mu$ is the annualized geometric return, and $\sigma$ is the annualized standard deviation.

To perform a fair comparison of the two covariance matrix estimation techniques, we next vary the risk aversion, $A$, in Equation (4.16), and create an efficient frontier. In Figure 4.9, we plot the efficient frontier for each of the portfolio selection methods. For a fixed value of $A$, we back-test each of the portfolio selection methods based on CAPM, the Fama-French 3-factor model, and errors-in-variables CAPM. The portfolio called market portfolio with risk-free is a portfolio which only uses the risk-free asset and the market portfolio. Each portfolio is updated daily, and transaction costs are ignored. We then calculate the annualized return and standard deviation of each
portfolio. We then repeat this for many values of $A$ in order to plot an efficient frontier. Figure 4.9 shows that the portfolio with errors-in-variables CAPM appears to outperform the portfolio with CAPM for each calibration length. For the calibration lengths 20 and 100 days, the portfolio using the Fama-French covariance matrix performs best. However, for the short calibration of length 5 days and the long calibration of length 500 days, the portfolio using the errors-in-variables CAPM has the highest Sharpe ratio. The highest Sharpe ratios are obtained when calibrating over 5 days, likely due to the fact that financial markets are non-stationary.

Figure 4.9: $\mu$ is the annualized geometric return, and $\sigma$ is the annualized volatility. We show the results using calibration lengths of (a) 5 days. (b) 20 days. (c) 100 days. (d) 500 days.
4.4.2 PJM

Next, we compare both covariance matrix estimation techniques, CAPM and errors-in-variables CAPM, for estimating the covariance matrix of the electricity market returns at each of the PJM hubs as described in Section 4.3.3. In Figure 4.10, we plot the wealth paths of the portfolios produced using errors-in-variables CAPM with various values for the risk aversion parameter, $A$, in Equation (4.18). In addition, we show the annualized return, annualized volatility, Sharpe ratio, and maximum drawdown for each of the portfolios in Table 4.2. We see that $A$ can be adjusted to control the annualized volatility and the maximum drawdown.

![Figure 4.10](image)

Figure 4.10: Starting from one dollar, wealth paths of portfolios chosen using errors-in-variables CAPM with different risk aversions, $A$.

In Figure 4.11 for many values of the risk aversion parameter $A$, we show the performance of the portfolios constructed using CAPM and errors-in-variables CAPM. As before, when estimating covariance matrices over only 5 days of data, portfolios using errors-in-variables CAPM produce higher Sharpe ratios than those with CAPM. For calibration lengths of 20, 100, and 500 days, the portfolios had very similar per-
Table 4.2: The portfolio statistics of the portfolios shown in Figure 4.10 which use portfolios chosen with errors-in-variables CAPM. \( \mu \) is the annualized geometric return, and \( \sigma \) is the annualized standard deviation.

Formances for each covariance estimation techniques. In the non-stationary settings of choosing PJM and S&P 100 portfolios, the errors-in-variables CAPM appears to do a better job estimating covariance matrices with a small amount of calibration data, resulting in portfolios with improved Sharpe ratios.

Figure 4.11: The efficient frontier for the PJM model.
4.5 Conclusion

We began with the intent to estimate covariance matrices of the returns on assets in order to choose portfolios and manage risk. Factor models provide a simple and numerically efficient way to estimate full rank covariance matrices. However, the classical way to estimate factor loadings defines the residuals as the vertical distance between the asset returns and the regression line. A natural extension is to define the residuals as the Euclidean distance between the asset returns and the regression line (total least squares). One downside of using the Euclidean distance to measure the size of the residuals is that it gives equal weights to each dimension, which may not be appropriate. In addition, the regression for each asset can no longer be estimated individually, because the fitted factor values should be the same across assets.

Allowing for errors in the factors (the market returns), we wrote down an errors-in-variables extension to CAPM and made assumptions, which allowed us to estimate the covariance matrix of the asset returns in closed form. We used the method of instrumental variable to calculate the $\beta$’s and maximum likelihood estimation to estimate the true market returns. We visually showed that the residuals of the assets appeared to be smaller and less correlated across assets, compared with the residuals in the traditional CAPM.

In our numerical work, we showed that Markowitz portfolios constructed with covariance matrices estimated with our errors-in-variables CAPM had higher Sharpe ratios than traditional CAPM. In particular, portfolios constructed using covariance matrices over the very short period of five days benefited greatly by the new covariance estimation technique and had surprisingly high Sharpe ratios. In the portfolios of PJM electricity contracts, making the number of megawatt hours bought in the day-ahead market sum to zero appeared to be very important in preventing large losses due to spikes in electricity prices. Overall, the new covariance matrix estimation
technique appeared to be very effective in controlling the variance of portfolios of virtual electricity contracts.
Chapter 5

Conclusion

We began by extending the knowledge gradient for calibrating continuous parameters. We showed that the inherently myopic method has nice convergence theory along with promising numerical results. We used the method to calibrate an airline simulator and then in the energy storage application.

The energy storage problem used approximate policy iteration to determine how to operate a storage device in the presence of a stochastic electricity price, volatile and intermittent wind power, and a stochastic load. For policy evaluation, we showed that off-policy Bellman error minimization with instrumental variables converges, and, furthermore, we showed it is actual equivalent to projected Bellman error minimization. We showed that direct policy search with basis functions actually had better performance than approximate policy iteration, although direct policy search does not easily extend to high dimensions. One challenge is choosing the search domain and another is the fundamental difficulty of optimizing a non-convex function with very little structure over a high dimensional space. A natural solution would be to use approximate policy iteration as a starting point for direct policy search.

We then used the method of instrumental variables, applicable to errors-in-variables models, and applied it to covariance matrix estimation. If the observations
of the market returns have noise, we can use the errors-in-variables extension of a factor model to estimate covariance matrices. We showed that we could get improved portfolios of PJM virtual contracts by using the errors-in-variables covariance estimation technique. We used a single factor in our model, but this could be generalized to multiple factors.
Appendix A

KGCP Appendix

A.1 Computing $\nabla_{x^n} \mu^n(x^i)$

If $i < n$ then $\mu^n_{x^i}$ does not depend on $x^n$ so $\nabla_{x^n} \mu^n(x^i) = 0$. Now consider when $i = n$. We start with equation (2.9) for $\mu^n(x^n)$ where $x^n$ has not been sampled and then simplify.

$$
\mu^n(x^n) = \mu^0(x^n) + \epsilon_{n+1}^T \Sigma^0 - \begin{bmatrix} I_n \\ \vec{y}^T \end{bmatrix} [S^n]^{-1} \vec{y}^n
$$

$$
= \mu^0(x^n) + \left[ \Sigma^0(x^0, x^n), \cdots, \Sigma^0(x^{n-1}, x^n) \right] [S^n]^{-1} \vec{y}^n
$$

Now, because $[S^n]^{-1} \vec{y}^n$ does not depend on the decision $x^n$, we can easily take the gradient.

$$
\nabla_{x^n} \mu^n(x^n) = \nabla_{x^n} \mu^0(x^n) + \left[ \nabla_{x^n} \Sigma^0(x^0, x^n), \cdots, \nabla_{x^n} \Sigma^0(x^{n-1}, x^n) \right] [S^n]^{-1} \vec{y}^n
$$

$$
= \nabla_{x^n} \mu^0(x^n) + J^n [S^n]^{-1} \vec{y}^n. \quad (A.1)
$$
where we \( J^n \) is defined in (2.33). When going from (2.33) to (2.34) we used the fact that the covariance function was of the form specified in (2.2).

A.2 Computing \( \nabla_{x^n} \tilde{\sigma}_i(\Sigma^n, x^n) \)

First, recall that

\[
\tilde{\sigma}_i(\Sigma^n, x^n) = \frac{e_i^T \Sigma^n e_{x^n}}{\sqrt{\lambda(x^n) + e_i^T \Sigma^n e_{x^n}}}, \quad i = 0, \ldots, n. \quad (A.2)
\]

After we derive the gradient of the numerator and denominator of this equation, we can find the gradient of (A.2) by using the quotient rule for differentiation.

\[
\nabla_{x^n} \tilde{\sigma}_i(\Sigma^n, x^n) = \frac{\sqrt{\lambda(x^n) + e_i^T \Sigma^n e_{x^n}} \nabla_{x^n} e_i^T \Sigma^n e_{x^n} - e_i^T \Sigma^n e_{x^n} \nabla_{x^n} \sqrt{\lambda(x^n)} + e_i^T \Sigma^n e_{x^n}}{|\lambda(x^n) + e_i^T \Sigma^n e_{x^n}|}
\]
A.2.1 The Numerator

First we consider the numerator of (A.2).

\[ e^T_x \Sigma^n e_n x = e^T_x (I - \bar{K}^n) \begin{bmatrix} I_n & 0 \end{bmatrix} \Sigma^0 e_n x \]  
\[ = e^T_x \Sigma^0 e_n x - e^T_x \bar{K}^n \begin{bmatrix} I_n & 0 \end{bmatrix} \Sigma^0 e_n x \]  
\[ = \Sigma^0 (x^i, x^n) - e^T_x \Sigma^0 \begin{bmatrix} I_n \cr \bar{0}^T \end{bmatrix} \begin{bmatrix} [S^n]^{-1} [I_n & 0] \Sigma^0 e_n x \end{bmatrix} \]  
\[ = \Sigma^0 (x^i, x^n) - \left[ \Sigma^0 (x^0, x^i), \ldots, \Sigma^0 (x^{n-1}, x^i) \right] [S^n]^{-1} \begin{bmatrix} \Sigma^0 (x^0, x^n) \\ \vdots \\ \Sigma^0 (x^{n-1}, x^n) \\ \Sigma^0 (x^{n-1}, x^i) \end{bmatrix} \]  
\[ = \Sigma^0 (x^i, x^n) - \left[ \Sigma^0 (x^0, x^n), \ldots, \Sigma^0 (x^{n-1}, x^n) \right] [S^n]^{-1} \begin{bmatrix} \Sigma^0 (x^0, x^i) \\ \vdots \\ \Sigma^0 (x^{n-1}, x^i) \end{bmatrix} \]  
\[ = \Sigma^0 (x^i, x^n) - \left[ \Sigma^0 (x^0, x^n), \ldots, \Sigma^0 (x^{n-1}, x^n) \right] [S^n]^{-1} \Sigma^0 e_x^i. \]  
\[ = 2 \text{diag}(\alpha) * (x^i - x^n) \Sigma^0 (x^i, x^n) - J^i [S^n]^{-1} \Sigma^0 e_x^i. \]  
\[ = 2 \text{diag}(\alpha) * (x^i - x^n) \Sigma^0 (x^i, x^n) - J^i [S^n]^{-1} \Sigma^0 e_x^i. \]  

In (A.3) we used the definition of \( \Sigma^a \) in (2.10). From (A.4) to (A.6) we just inserted the definition of \( \bar{K}^n \) given in (2.8). Going from (A.6) to (A.7) we took the transpose of the last term which is a scalar and used the fact that \( [S^n]^{-1} \) is symmetric. We first consider the case where \( i < n \). In this case \( [S^n]^{-1} \) does not depend on \( x^n \) so we can easily compute the gradient,

\[ \nabla_{x^n} e^T_x \Sigma^n e_n x = \nabla_{x^n} \Sigma^0 (x^i, x^n) - \left[ \nabla_{x^n} \Sigma^0 (x^0, x^n), \ldots, \nabla_{x^n} \Sigma^0 (x^{n-1}, x^n) \right] [S^n]^{-1} \Sigma^0 e_x^i \]
\[ = 2 \text{diag}(\alpha) * (x^i - x^n) \Sigma^0 (x^i, x^n) - J^i [S^n]^{-1} \Sigma^0 e_x^i. \]  
\[ = 2 \text{diag}(\alpha) * (x^i - x^n) \Sigma^0 (x^i, x^n) - J^i [S^n]^{-1} \Sigma^0 e_x^i. \]  

117
Now we consider the case where $i = n$. Using standard matrix differentiation, we can compute the gradient.

\[
\nabla_x e_n^T \Sigma_n e_x = \begin{bmatrix}
0 - 2 \left[ \frac{\partial}{\partial x_1} \Sigma^0(x^0, x^n), \ldots, \frac{\partial}{\partial x_n} \Sigma^0(x^n-1, x^n) \right] [S^n]^{-1} \\
0 - 2 \left[ \frac{\partial}{\partial x_{p}} \Sigma^0(x^0, x^n), \ldots, \frac{\partial}{\partial x_{n}} \Sigma^0(x^n-1, x^n) \right] [S^n]^{-1}
\end{bmatrix}
\]

\[
= -2 \begin{bmatrix}
\nabla_x \Sigma^0(x^0, x^n), \ldots, \nabla_x \Sigma^0(x^n-1, x^n)
\end{bmatrix} [S^n]^{-1}
\]

\[
= -2 J^n [S^n]^{-1} \begin{bmatrix}
\Sigma^0(x^0, x^n) \\
\Sigma^0(x^n-1, x^n)
\end{bmatrix}
\]
A.2.2 The Denominator

Now we consider the denominator of (A.2).

\[
\sqrt{\lambda(x^n) + e_{x^n}^T \Sigma^n e_{x^n}}
\]

\[
= \sqrt{\lambda(x^n) + e_{x^n}^T (I - K^n) \left[ I_n \mid \bar{0} \right] \Sigma^n} \tag{A.9}
\]

\[
= \sqrt{\lambda(x^n) + \Sigma^0(x_n, x^n) - e_{x^n}^T \bar{K} \left[ I_n \mid \bar{0} \right] \Sigma^0 e_{x^n}} \tag{A.10}
\]

\[
= \sqrt{\lambda(x^n) + \Sigma^0(x_n, x^n) - e_{x^n}^T \bar{K} \left[ I_n \mid \bar{0} \right] \Sigma^0 e_{x^n}} \tag{A.11}
\]

In (A.9) we inserted the definition of \( \Sigma^n \) given in (2.10). Going from (A.10) to (A.11) we inserted the definition of \( K^n \) given in (2.8). Now we take the gradient.

\[
\nabla_{x^n} \sqrt{\lambda(x^n) + e_{x^n}^T \Sigma^n e_{x^n}}
\]

\[
= \frac{1}{2} (\lambda(x^n) + \Sigma^0(x_n, x^n))^{-\frac{1}{2}} \left( \frac{\partial}{\partial g(x^n)} \lambda(x^n) - 2 \left[ \frac{\partial}{\partial g(x^n)} \Sigma^0(x_n, x^n) \right] \right)
\]

\[
= \frac{1}{2} (\lambda(x^n) + \Sigma^0(x_n, x^n))^{-\frac{1}{2}} \left( \nabla_{x^n} \lambda(x^n) - 2 \left[ \nabla_{x^n} \Sigma^0(x_n, x^n) \right] \right)
\]

\[
= \frac{1}{2} (\lambda(x^n) + \Sigma^0(x_n, x^n))^{-\frac{1}{2}} \left( \nabla_{x^n} \lambda(x^n) - 2 \nabla_{x^n} \Sigma^0(x_n, x^n) \right)
\]

In (A.11) we inserted the definition of \( \bar{K} \) given in (2.10). Going from (A.10) to (A.11) we inserted the definition of \( \bar{K} \) given in (2.8). Now we take the gradient.
A.3 Proof of Proposition 2.5.1

We derive the upper bound of the knowledge gradient for continuous parameters given in (2.35), starting with

\[ E \left[ \max_{i=0,...,n} \mu^{n+1}(x^i) \left| F^n, x^n = x \right. \right] \]  \hspace{1cm} (A.12)

\[ = E \left[ \max_{i=0,...,n} \mu^n(x^i) + \tilde{\sigma}_i(\tilde{\Sigma}^n, x^n) Z^{n+1} \left| F^n, x^n = x \right. \right] \]  \hspace{1cm} (A.13)

\[ \leq \max_{i=0,...,n} \mu^n(x^i) + E \left[ \max_{j=0,...,n} \tilde{\sigma}_j(\tilde{\Sigma}^n, x^n) Z^{n+1} \left| F^n, x^n = x \right. \right] \]  \hspace{1cm} (A.14)

We now need an upper bound on \( |\tilde{\sigma}_j(\tilde{\Sigma}^n, x^n)| \) in (A.14). We just note that

\[ |\tilde{\sigma}_j(\tilde{\Sigma}^n, x^n)| = \frac{e_{x^j} \tilde{\Sigma}^n e_{x^n}}{\sqrt{\lambda + e^T_{x^n} \tilde{\Sigma}^n e_{x^n}}} \]

\[ = \frac{\text{Cov}^{n}[\mu(x^j), \mu(x^n)]}{\sqrt{\lambda + \text{Var}^n[\mu(x^n)]}} \]

\[ = \frac{\text{Corr}^{n}[\mu(j), \mu(x^n)] \sqrt{\text{Var}^n[\mu(x^j)] \text{Var}^n[\mu(x^n)]}}{\sqrt{\lambda + \text{Var}^n[\mu(x^n)]}} \]

\[ \leq \frac{\sqrt{\text{Var}^n[\mu(x^j)] \text{Var}^n[\mu(x^n)]}}{\sqrt{\lambda}} \]

\[ = \sqrt{\frac{\text{Var}^n[\mu(x^j)] \text{Var}^n[\mu(x^n)]}{\lambda}}. \]  \hspace{1cm} (A.15)

Combining (A.14) and (A.15) we have an upper bound on the knowledge gradient for continuous parameters.

\[ \hat{\nu}^{KG,n}(x) \leq \frac{2}{\sqrt{2\pi}} \max_{j=0,...,n} \sqrt{\frac{\text{Var}^n[\mu(x^j)] \text{Var}^n[\mu(x^n)]}{\lambda}} \leq \sqrt{\frac{2\beta \text{Var}^n[\mu(x^n)]}{\pi \lambda}} = \sqrt{\frac{2\beta \text{Var}^n[\mu(x)]}{\pi \lambda}} \]  \hspace{1cm} (A.16)
The knowledge gradient for continuous parameters is nonnegative and the above upper bound on the knowledge gradient for continuous parameters of a decision \( x \) converges to zero as the conditional variance of \( \mu(x) \) converges to zero.

### A.4 Proof of Proposition 2.5.2

We derive how the conditional variance of \( \mu(x^d) \) decreases if we repeatedly measure a particular point \( x^{\text{mult}} \) \( n \) times with noise variance \( \lambda \) for each observation. We define
the policy $\pi^{\text{mult}}$ which sets $x^0 = \cdots = x^{n-1} = x^{\text{mult}}$. Under this policy we see,

$$
\Sigma^n(x,x) = e^T \Sigma^n e_x
$$

$$
= e^T_x (I - \tilde{K}_n) \left[ I_n \mid 0 \right] \Sigma^0 e_x
$$

$$
= \Sigma^0(x,x) - e^T_x \tilde{K}_n \left[ I_n \mid 0 \right] \Sigma^0 e_x
$$

(a.17)

$$
= \Sigma^0(x,x) - e^T_x \Sigma^0 \left[ I_n \mid [S^n]^{-1} \left[ I_n \mid 0 \right] \tilde{K}_n \tilde{\sigma} \right]
$$

(a.18)

$$
= \Sigma^0(x,x) - \left[ \Sigma^0(x^0,x), \ldots, \Sigma^0(x^{n-1},x) \right] [S^n]^{-1} \left[ \Sigma^0(x^0,x) \right.

\[ \vdots \]

\[ \Sigma^0(x^{n-1},x) \]

(a.19)

$$
= \beta - \left[ \Sigma^0(x^{\text{mult}},x), \ldots, \Sigma^0(x^{\text{mult}},x) \right] \left[ \begin{array}{cccc}
\beta & \cdots & \beta \\
\vdots & \ddots & \vdots \\
\beta & \cdots & \beta
\end{array} \right]^{-1} \left[ \begin{array}{c}
\Sigma^0(x^{\text{mult}},x) \\
\vdots \\
\Sigma^0(x^{\text{mult}},x)
\end{array} \right]
$$

(a.20)

$$
= \beta - (\Sigma^0(x^{\text{mult}},x))^2 e^T \beta \left[ \begin{array}{cc}
1 & \cdots \\
\vdots & \ddots \\
1 & \cdots
\end{array} \right] + \lambda I_n \beta
$$

$$
= \beta - (\Sigma^0(x^{\text{mult}},x))^2 \frac{n}{\beta n + \lambda}.
$$

(a.21)

In (a.17) we insert the definition of $\tilde{\Sigma}$ given in (2.10). In (a.18) we insert the definition of $\tilde{K}_n$ given in (2.8). $[S^n]^{-1}$ is positive semi-definite, so the second term in (a.19) is nonnegative. In (a.20) $e$ is a column vector of ones, and we simplify the
expression using the definition of the inverse of $S^n$,

$$
[S^n]^{-1} \begin{bmatrix}
\beta & \cdots & 1 \\
\vdots & \ddots & \vdots \\
1 & \cdots & 1
\end{bmatrix} + \lambda I_n = I_n,
$$

$$
e^T[S^n]^{-1} \begin{bmatrix}
\beta & \cdots & 1 \\
\vdots & \ddots & \vdots \\
1 & \cdots & 1
\end{bmatrix} e = e^T I_n e,
$$

$$
e^T[S^n]^{-1} [\beta n + \lambda e] = n,
$$

$$
e^T[S^n]^{-1} e = \frac{n}{\beta n + \lambda}.
$$

(A.22)

First consider the change $\text{Var}^n[\mu(x^d)] - \text{Var}^{n+1}[\mu(x^d)]$ if we have measured $x^\text{mult}$ $n$ times and then measure $x^\text{mult}$ one more time. We use (A.21) and assume $\Sigma^0(x, x) = \beta, \forall x$. Also, define $\beta_0 = \Sigma^0(x^\text{mult}, x^d)$. The decrease in the conditional variance of $\mu(x^d)$ from measuring $x^\text{mult}$ once more is

$$
\text{Var}^n[\mu(x^d)] - \text{Var}^{n+1}[\mu(x^d)]
= (\beta - B^2_0 n(n\beta + \lambda)^{-1}) - (\beta - B^2_0 (n+1)((n+1)\beta + \lambda)^{-1})
= B^2_0 (n+1)((n+1)\beta + \lambda)^{-1} - B^2_0 n(n\beta + \lambda)^{-1}
= \frac{B^2_0 (n+1)(n\beta + \lambda) - B^2_0 n((n+1)\beta + \lambda)}{((n+1)\beta + \lambda)(n\beta + \lambda)}
= \frac{\beta_0^2 \lambda}{((n+1)\beta + \lambda)(n\beta + \lambda)}.
$$

(A.23)

In (A.23) we just used (A.21) which gives an expression for $\text{Var}^n[\mu(x)]$ if we measure $x^\text{mult}$ $n$ times and nothing else. Second we consider measuring the change in $\text{Var}^n[\hat{\mu}(x^d)] - \text{Var}^{n+1}[\hat{\mu}(x^d)]$ if we have measured $x^\text{mult}$ $n$ times and then measure $x^\text{near}$ one time, where $x^\text{near} \in B(x^\text{acc}, \epsilon)$ and satisfies $\Sigma^0(x^\text{mult}, x^d) \leq \Sigma^0(x^\text{near}, x^d)$. We define $\beta_1 = \Sigma^0(x^\text{mult}, x^\text{near})$ and $\beta_2 = \Sigma^0(x^\text{near}, x^d)$. Note that $\beta_0 \leq \beta_2$ and
0 < β₀, β₁, β₂ ≤ β.

\[ \Sigma^{n+1}(x^d, x^d) = \Sigma^{n}(x^{near}, x^d) - \Sigma^{n}(x^{near}, x^d) (\Sigma^{n}(x^{near}, x^{near}) + \lambda)^{-1} \Sigma^{n}(x^{near}, x^d) \]  \hspace{1cm} \text{(A.25)}

\[ \Sigma^{n}(x^d, x^d) = \Sigma^{n}(x^{near}, x^d) \left( \beta^2 - \frac{n\beta_0\beta_1}{n\beta + \lambda} \right)^2 \left( \beta - \frac{n\beta_1^2}{n\beta + \lambda} + \lambda \right)^{-1} \]  \hspace{1cm} \text{(A.26)}

In (A.25) we use the recursive equation for updating the conditional variance. In (A.26) we plugged in the equation for \( \Sigma^{n}(x^{near}, x^d) \) which is derived in the same way as (A.21). Equivalently we can write

\[ \text{Var}^n[\mu(x^d)] - \text{Var}^{n+1}[\mu(x^d)] = \left( \beta_2 - \frac{n\beta_0\beta_1}{n\beta + \lambda} \right)^2 \left( \beta - \frac{n\beta_1^2}{n\beta + \lambda} + \lambda \right)^{-1} \]  \hspace{1cm} \text{(A.27)}

We now want to show that if we have measured \( x^{\text{mult}} \) \( n \) times that the amount we can lower the conditional variance of \( \mu(x^d) \) by observing \( x^{\text{mult}} \) again given in (A.24)
is smaller than the amount given in (A.27) if we observe a new point \( x^\text{near} \).

\[
\begin{align*}
\left(\beta_2 - \frac{n\beta_0\beta_1}{n\beta + \lambda}\right)^2 \left(\beta - \frac{n\beta_1^2}{n\beta + \lambda}\right)^{-1} &= \left(\frac{\beta_2(n\beta + \lambda) - n\beta_0\beta_1}{n\beta + \lambda}\right)^2 \left(\frac{(\beta + \lambda)(n\beta + \lambda) - n\beta_1^2}{n\beta + \lambda}\right)^{-1} \\
&= \frac{(\beta_2(n\beta + \lambda) - n\beta_0\beta_1)^2}{(n\beta + \lambda) ((\beta + \lambda)(n\beta + \lambda) - n\beta_1^2)} \\
&\geq \frac{(\beta_0(n\beta + \lambda) - n\beta_0\beta_1)^2}{(n\beta + \lambda) ((\beta + \lambda)(n\beta + \lambda) - n\beta_1^2)} \\
&\geq \frac{\beta_0^2 \lambda^2}{(n\beta + \lambda)(n\beta\lambda + \beta\lambda + \lambda^2)} \\
&= \frac{\beta_0^2 \lambda}{(n\beta + \lambda)((n + 1)\beta + \lambda)} \\
&\text{In (A.28) we replaced } \beta_2 \text{ with the smaller } \beta_0. \text{ This is valid because the overall term is positive and the numerator is nonnegative because } \beta_0 \leq \beta_2 \text{ and } \beta_1 \leq \beta. \text{ In (A.29) we replaced } \beta_1 \text{ with the larger } \beta. \text{ This is valid because the derivative of (A.28) with respect to } \beta_1 \text{ is negative. Using the quotient rule the derivative of (A.28) with respect to } \beta_1 \text{ becomes:}
\end{align*}
\]

\[
\begin{align*}
&\frac{(n\beta + \lambda) ((\beta + \lambda)(n\beta + \lambda) - n\beta_1^2)}{2(n\beta + \lambda) - n\beta_0\beta_1)(n\beta_0 - \beta_0) - (n\beta_0 + \lambda) - n\beta_0\beta_1)^2 (n\beta + \lambda)(-2n\beta_1)
\end{align*}
\]

\[
= \frac{2n(n\beta + \lambda)e^{-2} ((\beta + \lambda)(n\beta + \lambda) - n\beta_1^2)(n\beta_0 + \lambda) - n\beta_0\beta_1)(-2n\beta_1)}{2(n\beta + \lambda)e^{-2} ((\beta + \lambda)(n\beta + \lambda) - n\beta_1^2)(n\beta_0 + \lambda) - n\beta_0\beta_1)(-2n\beta_1)}
\]

\[
= \frac{2n(n\beta + \lambda)e^{-2} ((\beta + \lambda)(n\beta + \lambda) - n\beta_1^2)(n\beta + \lambda - n\beta_1)(n\beta + \lambda - n\beta_1)}{2(n\beta + \lambda)e^{-2} ((\beta + \lambda)(n\beta + \lambda) - n\beta_1^2)(n\beta + \lambda - n\beta_1)(n\beta + \lambda - n\beta_1)}
\]

\[
= 2n(n\beta + \lambda)e^{-2} (n\beta + \lambda - n\beta_1)(n\beta + \lambda - n\beta_1)(n\beta + \lambda - n\beta_1)
\]

\[
\leq 0.
\]

We have now shown that if we have measured \( x^\text{mult} n \) times that the amount we can lower the conditional variance of \( \mu(x^d) \) by observing \( x^\text{mult} \) again given in (A.24) is smaller than the amount given in (A.27) if we observe a new point.
This is true for $n = 0, 1, 2, \ldots$ so using an induction argument we see

$$\max_{x_0, \ldots, x_{n-1} \in B(x^\text{acc}, \epsilon)} \text{Var}^n[\mu(x^d)]$$

equals $\text{Var}^n[\mu(x^d)]$ under $\pi^{\text{mult}}$. 

$x^{\text{near}}$. 

126
Appendix B

Energy Storage Appendix

B.1 The Instrumental Variable Method

The instrumental variable method is a well known technique for dealing with errors in the explanatory variables (errors-in-variables) of a regression problem. In this section we summarize explanatory variables and the consistency properties mentioned in [27], [10], [101], and [47]. We consider the linear model in matrix form:

\[ Y_i = \sum_{j=1}^{k} X_{ij} \beta_j \quad i = 1, \ldots, n, \]  

or

\[ Y = X \beta, \]

where \( Y \) is a \( n \times 1 \) vector of the response variable, \( X \) is a \( n \times k \) matrix of explanatory variables, and \( \beta \) is a \( k \times 1 \) vector of weights. Suppose we can only observe \( X' \) and \( Y' \), not the true values \( X \) and \( Y \). Following similar notation as [27],

\[ X_{ij}' = X_{ij} + X_{ij}'', \]
\[ Y_i' = Y_i + Y_i''., \]
and in matrix form,

\[ X' = X + X'', \quad \text{(B.2)} \]
\[ Y' = Y + Y'', \quad \text{(B.3)} \]

where \( X'' \) and \( Y'' \) are the errors in the observed values of \( X \) and \( Y \). Our linear regression model can now be written as:

\[ Y' = X\beta + Y''. \]

As explained in [47], we treat \( X \) and \( Y \) as random variables. Unlike a standard linear regression problem, Equation (B.1) is a structural relation which relates two random variables (\( X \) is not deterministic). This is different than a regression line which gives a functional relationship that relates the mean of the dependant variable to the regressor variable (see [47]).

The first assumptions are that the noise in \( X \) and \( Y \) have mean zero,

**Assumption B.1.1.** \( \mathbb{E}[Y'_{i}] = 0 \), \( i = 1, ..., n \).

**Assumption B.1.2.** \( \mathbb{E}[X'_{ij}] = 0 \), \( \forall i, j \).

### B.1.1 Example of Bias in Ordinary Least Squares

[47] and [27] show that least squares regression encounters problems with the model given by Equations (B.1), (B.2), (B.3). The source of the problem is the correlation between the \( X' \) and \( X'' \), since the observation of \( X \) is typically correlated with the error in \( X \). If \( \beta \) is a scalar \( (k = 1) \), this is easy to show. Starting with the least
squares estimate of $\beta_1$,

\[
\hat{\beta}_1 = (X^T X)^{-1} X^T Y' \\
\hat{\beta}_1 = \frac{\sum_{i=1}^{n} X_{i1}' Y_i'}{\sum_{i=1}^{n} (X_{i1}')^2} \\
= \frac{\sum_{i=1}^{n} X_{i1}' (X_{i1} \beta_1 + Y_i'')}{\sum_{i=1}^{n} (X_{i1}')^2} \\
= \frac{\sum_{i=1}^{n} X_{i1}' ((X_{i1}' - X_{i1}'' \beta_1 + Y_i''))}{\sum_{i=1}^{n} (X_{i1}')^2} \\
= \beta_1 - \beta_1 \frac{\sum_{i=1}^{n} X_{i1}' X_{i1}''}{\sum_{i=1}^{n} (X_{i1}')^2} + \frac{\sum_{i=1}^{n} X_{i1}' Y_i''}{\sum_{i=1}^{n} (X_{i1}')^2} \\
\text{(B.6)}
\]

In Equation (B.4) we substituted in Equation (B.1) and (B.3). In Equation (B.5) we used Equation (B.2). Now taking the limit as $n$ goes to infinity, Equation (B.6) becomes

\[
\lim_{n \to \infty} \hat{\beta}_1 = \beta_1 - \beta_1 \lim_{n \to \infty} \left( \frac{\sum_{i=1}^{n} X_{i1}' X_{i1}''}{\sum_{i=1}^{n} (X_{i1}')^2} \right) + \lim_{n \to \infty} \left( \frac{\sum_{i=1}^{n} X_{i1}' Y_i''}{\sum_{i=1}^{n} (X_{i1}')^2} \right) \\
= \beta_1 - \beta_1 \lim_{n \to \infty} \left( \frac{\sum_{i=1}^{n} X_{i1}' X_{i1}''}{\sum_{i=1}^{n} (X_{i1}')^2} \right) + \frac{\text{Cov}[X_{i1}', Y_i'']}{\text{E}[(X_{i1}')^2]} \\
= \beta_1 - \beta_1 \lim_{n \to \infty} \left( \frac{\sum_{i=1}^{n} X_{i1}' X_{i1}''}{\sum_{i=1}^{n} (X_{i1}')^2} \right) . \\
\text{(B.7)}
\]

Equation (B.8) holds if Cov$[X_{i1}', Y_i''] = 0$. For many problems, $X_{i1}'$ and $X_{i1}''$ are positively correlated. Hence Equation (B.8) shows that typically the least squares estimate of $\beta_1$ is inconsistent and too small in magnitude for these problems. This problem can be overcome if an instrumental variable is available.

In Figure B.1 we generated the regressors $X$ and $Z$ from a multivariate normal distribution with correlation 0.7. We then added independent Gaussian noise to $X$, and $Y$, where $Y = X \beta$. The various regression techniques are plotted Figure B.1. The errors in the $X$ have violated the assumptions necessary for least-squares, and the least-squares regression line is too flat, as the theory predicts. The least-absolute-
deviations regression (L1) is also too flat in this example. The instrumental variables method is consistent for this problem and this can be observed in the figure.

Figure B.1: When there are errors in the regressors, instrumental variables can be used to solve the problem.

If an instrumental variable is known to exist, why not just add it as an additional regressor? If our main goal is to determine the value of $\beta$, adding dimensions to the regression problem could cause $\beta$ to lose its meaning. As we see below, a properly chosen instrumental variable can yield a consistent estimator for $\beta$.

B.1.2 Consistency of Estimate using Instrumental Variables

An instrumental variable, $Z_j$, should be correlated with the true $X_j$ but uncorrelated with the errors in the observations of $X$ and $Y$ (see [27] and [47]). We use the notation $X_j$ to indicate the $j$’th column of $X$. Let $\Sigma$ be the $k \times k$ matrix where $\Sigma_{jl} = \text{Cov}[Z_j, X_l]$. Below we extend the consistency proof from [47] to use multiple instrumental variables ($k > 1$). We assume an instrumental variable exists with the following properties:

Assumption B.1.3. $\text{Cov}[Z_{ij}, Y''_i] = 0, \quad \forall i, j.$

Assumption B.1.4. $\text{Cov}[Z_{ij}, X''_{il}] = 0, \quad \forall i \in \{1, \ldots, n\}, \quad j, l \in \{1, \ldots, k\}.$
Assumption B.1.5. $\Sigma$ has full rank $k$, where $\Sigma_{jl} = \text{Cov}[Z_j, X_l]$.

Assumption B.1.6. $\lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} Z_{ij} Y_i'' = 0, \quad \forall j = 1, \ldots, k$.

Assumption B.1.7. $\lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} Z_{ij} X_{il}'' = 0, \quad \forall j, l \in \{1, \ldots, k\}$.

Assumption B.1.8. $\lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} Z_{ij} X_{il} = \text{Cov}[Z_j, X_l], \quad \forall j, l \in \{1, \ldots, k\}$.

Assumptions B.1.6, B.1.7, and B.1.8 do not follow trivially from assumptions B.1.3, B.1.4, B.1.5 without additional assumptions such as independence across the $n$ observations (because the law of large numbers does not apply). The method of instrumental variables defines the estimator $\hat{\beta}$ as the solution to

$$Z^T X' \hat{\beta} = Z^T Y', \quad (B.9)$$

where $Z$ is a $n \times k$ matrix. Note that $\hat{\beta}$ is uniquely defined when $Z^T X'$ has full rank $k$.

Proposition B.1.1. For the model given by Equations (B.1), (B.2), (B.3), if Assumptions B.1.1, B.1.2, B.1.3, B.1.4, B.1.5, B.1.6, B.1.7, B.1.8 hold, then $\hat{\beta} = [Z^T X']^{-1} Z^T Y'$ is a consistent estimator of $\beta$.

Simplifying Equation (B.9) we get

$$Z^T (X + X'') \hat{\beta} = Z^T (X \beta + Y''), \quad (B.10)$$

$$(Z^T X + Z^T X'') \hat{\beta} = Z^T X \beta + Z^T Y''.$$
In Equation (B.10), we used Equations (B.1), (B.2), and (B.3). Now, taking the limit as \( n \) goes to infinity,

\[
\lim_{n \to \infty} \frac{1}{n} \left( Z^T X + Z^T X'' \right) \hat{\beta} = \lim_{n \to \infty} \frac{1}{n} \left( Z^T X \beta + Z^T Y'' \right) \quad (B.11)
\]

\[
\lim_{n \to \infty} \frac{1}{n} Z^T X \hat{\beta} = \lim_{n \to \infty} \frac{1}{n} Z^T X \beta \quad (B.12)
\]

\[
\Sigma \left( \lim_{n \to \infty} \hat{\beta} \right) = \Sigma \beta \quad (B.13)
\]

\[
\lim_{n \to \infty} \hat{\beta} = \beta. \quad (B.14)
\]

In Equation (B.12), we used Assumptions B.1.3 and B.1.6 which imply \( \lim_{n \to \infty} \frac{1}{n} Z^T Y'' = \vec{0} \) and Assumptions B.1.4 and B.1.7 which imply \( \lim_{n \to \infty} \frac{1}{n} Z^T X'' = 0 \). We also used Slutsky’s theorem when taking the limit. In Equation (B.13) the sample covariances converge in probability to the true covariances by Assumption B.1.8. Getting to Equation (B.14), we used Assumption B.1.5 which ensures that \( \hat{\beta} \) is unique.

\[
\square
\]

### B.2 Proof of Lemmas in Section 3.4.5

#### B.2.1 Proof of Lemma 3.4.1

\[
\mathbb{E}[Y''] = \mathbb{E}[\Pi_{t-1}(C_t - \bar{C}_{t-1})] \quad (B.15)
\]

\[
= \mathbb{E} \left[ \mathbb{E}[\Pi_{t-1}(C_t - \bar{C}_{t-1}) \mid S^z_{t-1}] \right] \quad (B.16)
\]

\[
= \mathbb{E}[\Pi_{t-1}] \mathbb{E}[(C_t - \bar{C}_{t-1}) \mid S^z_{t-1}] \bigg|_{= \vec{0}} \quad (B.17)
\]

\[
= \vec{0}. \quad (B.18)
\]
B.2.2 Proof of Lemma 3.4.2

\[ E[X'''] = E[X' - X] \]  \hspace{1cm} \text{(B.19)}

\[ = E[\Pi_{t-1}(\Phi_{t-1} - \gamma \Phi_t) - \Pi_{t-1}(\Phi_{t-1} - E[\gamma \Phi_t \mid \{S_{t-1}^x\})]] \]  \hspace{1cm} \text{(B.20)}

\[ = \gamma E[\Pi_{t-1}(E[\Phi_t \mid \{S_{t-1}^x\}) - \Phi_t)] \]  \hspace{1cm} \text{(B.21)}

\[ = \gamma E[E[\Pi_{t-1}(E[\Phi_t \mid \{S_{t-1}^x\}) - \Phi_t]) \mid \{S_{t-1}^x\})]] \]  \hspace{1cm} \text{(B.22)}

\[ = \gamma E[E[\Pi_{t-1}(E[\Phi_t \mid \{S_{t-1}^x\}) - \Phi_t]) \mid \{S_{t-1}^x\})]] \]  \hspace{1cm} \text{(B.23)}

\[ = \gamma E[\Pi_{t-1}(E[\Phi_t \mid \{S_{t-1}^x\}) - E[\Phi_t \mid \{S_{t-1}^x\})]]. \]  \hspace{1cm} \text{(B.24)}

\[ = 0. \]  \hspace{1cm} \text{(B.25)}

B.2.3 Proof of Lemma 3.4.3

\[ \text{Cov}[Z_{ij}, Y_i''''] = E[Z_{ij} Y_i'''] - E[Z_{ij}] E[Y_i'''] \]  \hspace{1cm} \text{(B.26)}

\[ = E[\Phi_{t-1,ij} \{\Pi_{t-1}(C_t - \bar{C}_{t-1})\}] \]  \hspace{1cm} \text{(B.27)}

\[ = E[\Phi_{t-1,ij}] e_i^T \Pi_{t-1}(C_t - \bar{C}_{t-1}) \]  \hspace{1cm} \text{(B.28)}

\[ = E \left[ E[\Phi_{t-1,ij}] e_i^T \Pi_{t-1}(C_t - \bar{C}_{t-1}) \mid \{S_{t-1}^x\})] \right] \]  \hspace{1cm} \text{(B.29)}

\[ = E[\Phi_{t-1,ij}] e_i^T \Pi_{t-1} E[C_t - \bar{C}_{t-1} \mid \{S_{t-1}^x\})] \]  \hspace{1cm} \text{(B.30)}

\[ = 0. \]  \hspace{1cm} \text{(B.31)}
B.2.4 Proof of Lemma 3.4.4

\[
\text{Cov}[Z_{ij}, X_{it}'] = \mathbb{E}[Z_{ij}X_{it}'] - \mathbb{E}[Z_{ij}]\mathbb{E}[X_{it}'] = 0
\] (B.32)

\[
= \mathbb{E}[Z_{ij}(X_{it} - X_{il})]
\] (B.33)

\[
= \mathbb{E}[Z_{ij}e_i^T(X' - X)e_l]
\] (B.34)

\[
= \gamma\mathbb{E}[\Phi_{t-1,ij}e_i^T\Pi_{t-1}(\mathbb{E}[\Phi_t|\{S_{t-1}^x\}] - \Phi_t)e_l]
\] (B.35)

\[
= \gamma\mathbb{E}[\mathbb{E}[\Phi_{t-1,ij}e_i^T\Pi_{t-1}\mathbb{E}[\Phi_t|\{S_{t-1}^x\}] - \Phi_t|\{S_{t-1}^x\}]e_l]
\] (B.36)

\[
= \gamma\mathbb{E}[\Phi_{t-1,ij}e_i^T\Pi_{t-1}\mathbb{E}[\mathbb{E}[\Phi_t|\{S_{t-1}^x\}] - \mathbb{E}[\Phi_t|\{S_{t-1}^x\}]|e_l]
\] (B.37)

\[
= \gamma\mathbb{E}[\Phi_{t-1,ij}e_i^T\Pi_{t-1}\mathbb{E}[\Phi_t|\{S_{t-1}^x\}]e_l]
\] (B.38)

\[
= 0.
\] (B.39)

B.3 Proof of Theorem 3.4.1

We first show Equations (3.32) and (3.34) are equivalent. Starting with Equation (3.34) and recalling that, by definition, \(\Pi_{t-1} = \Phi_{t-1}((\Phi_{t-1})^T\Phi_{t-1})^{-1}(\Phi_{t-1})^T\), we can write

\[
[(\Phi_{t-1})^T\Pi_{t-1}(\Phi_{t-1} - \gamma\Phi_t)]^{-1}(\Phi_{t-1})^T\Pi_{t-1}C_t
\]

\[
= \left[(\Phi_{t-1})^T\left(\Pi_{t-1}\Phi_{t-1} - \gamma\Pi_{t-1}\Phi_t\right)\right]^{-1}(\Phi_{t-1})^T\Phi_{t-1}(\Phi_{t-1})^T\Phi_{t-1}^{-1}(\Phi_{t-1})^T\Pi_{t-1}C_t
\]

\[
= \left[(\Phi_{t-1})^T\Phi_{t-1} - \gamma(\Phi_{t-1})^T\Pi_{t-1}\Phi_t\right]^{-1}(\Phi_{t-1})^T\Pi_{t-1}C_t
\]

\[
= \left[(\Phi_{t-1})^T\Phi_{t-1} - \gamma(\Phi_{t-1})^T\Phi_{t-1}(\Phi_{t-1})^T\Phi_{t-1}^{-1}(\Phi_{t-1})^T\Phi_t\right]^{-1}(\Phi_{t-1})^T\Pi_{t-1}C_t
\]

\[
= [((\Phi_{t-1})^T\Phi_{t-1} - \gamma(\Phi_{t-1})^T\Phi_t)^{-1}(\Phi_{t-1})^T\Pi_{t-1}C_t
\]

\[
= [((\Phi_{t-1})^T\Phi_{t-1} - \gamma(\Phi_{t-1})^T\Phi_t)]^{-1}(\Phi_{t-1})^T\Pi_{t-1}C_t
\]

\[
= [((\Phi_{t-1})^T\Phi_{t-1} - \gamma(\Phi_{t-1})^T\Phi_t)]^{-1}(\Phi_{t-1})^T\Pi_{t-1}C_t
\]

\[
= [((\Phi_{t-1})^T(\Phi_{t-1} - \gamma\Phi_t)]^{-1}(\Phi_{t-1})^T\Pi_{t-1}C_t.
\]

134
Hence Equations (3.32) and (3.34) are equivalent. Next we show Equations (3.32) and (3.33) are equivalent. We start by writing

\[(\Phi_{t-1} - \gamma \Phi_t)^T \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \]
\[= (\Phi_{t-1} - \gamma \Phi_t)^T \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \]
\[\Rightarrow (\Phi_{t-1} - \gamma \Phi_t)^T \Phi_{t-1} \left[ (\Phi_{t-1})^T \Phi_{t-1} \right]^{-1} (\Phi_{t-1} - \gamma \Phi_t) \]
\[= (\Phi_{t-1} - \gamma \Phi_t)^T \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \]
\[\Rightarrow (\Phi_{t-1} - \gamma \Phi_t)^T \Phi_{t-1} \left[ (\Phi_{t-1})^T \Phi_{t-1} \right]^{-1} (\Phi_{t-1} - \gamma \Phi_t) \]
\[= (\Phi_{t-1} - \gamma \Phi_t)^T \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \]
\[\Rightarrow \left( \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \right)^T \]
\[= (\Phi_{t-1} - \gamma \Phi_t)^T \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \]
\[\Rightarrow \left[ (\Phi_{t-1} - \gamma \Phi_t)^T \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \right]^{-1} \left( \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \right)^T \]
\[= (\Phi_{t-1} - \gamma \Phi_t)^T \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \]
\[\Rightarrow \left( \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \right)^T \]
\[= \left[ (\Phi_{t-1} - \gamma \Phi_t)^T \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \right]^{-1} \left( \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \right)^T \]
\[= \left[ (\Phi_{t-1} - \gamma \Phi_t)^T (\Pi_{t-1})^T \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \right]^{-1} (\Phi_{t-1} - \gamma \Phi_t)^T \left( \Pi_{t-1} \right)^T \]
\[= \left[ (\Phi_{t-1} - \gamma \Phi_t)^T \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \right]^{-1} (\Phi_{t-1} - \gamma \Phi_t)^T \]
\[= \left[ (\Phi_{t-1} - \gamma \Phi_t)^T \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \right]^{-1} (\Phi_{t-1} - \gamma \Phi_t)^T \]
\[= \left[ (\Phi_{t-1} - \gamma \Phi_t)^T (\Pi_{t-1})^T \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \right]^{-1} (\Phi_{t-1} - \gamma \Phi_t)^T \]
\[= \left[ (\Phi_{t-1} - \gamma \Phi_t)^T \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \right]^{-1} (\Phi_{t-1} - \gamma \Phi_t)^T \]
\[= \left[ (\Phi_{t-1} - \gamma \Phi_t)^T (\Pi_{t-1})^T \Pi_{t-1} (\Phi_{t-1} - \gamma \Phi_t) \right]^{-1} (\Phi_{t-1} - \gamma \Phi_t)^T \]
\[= \left[ (\Phi_{t-1} - \gamma \Phi_t)^T (\Phi_{t-1} - \gamma \Phi_t) \right]^{-1} (\Phi_{t-1} - \gamma \Phi_t)^T \]
\[= \left[ (\Phi_{t-1} - \gamma \Phi_t)^T (\Phi_{t-1} - \gamma \Phi_t) \right]^{-1} (\Phi_{t-1} - \gamma \Phi_t)^T \]

Equation (B.40) uses the fact that \((\Pi_{t-1})^T = \Pi_{t-1}\). Hence Equations (3.32), (3.33), and (3.34) are equivalent.
B.4 Performance of Algorithms with Different Basis Functions

Figure B.2 and B.3 show the performance of the approximate dynamic programming algorithms using first order and third order basis functions, respectively.

Figure B.2: The algorithms use linear basis functions. We show the performance of Bellman error minimization using instrumental variables (IV) and least-squares Bellman error minimization (LS) along with direct policy search (KGCP).
Figure B.3: Third order basis functions. We show the performance of Bellman error minimization using instrumental variables (IV) and least-squares Bellman error minimization (LS).
Bibliography


142