Hedging Optimization Algorithms for Deregulated Electricity Markets

by

Michael R. Wagner

B.S. in Electrical Engineering and Computer Science
Massachusetts Institute of Technology (2000)

B.S. in Mathematics
Massachusetts Institute of Technology (2000)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2001

Author .....
Department of Electrical Engineering and Computer Science
May 15, 2001

Certified by .....
Marija Ilić
Senior Research Scientist
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Hedging Optimization Algorithms for Deregulated Electricity Markets

by

Michael R. Wagner

Submitted to the Department of Electrical Engineering and Computer Science
on May 15, 2001, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Recent trends in many U.S. states are to deregulate their electric power industry and markets with the desire to provide a more consumer-friendly environment than under regulation. However, deregulation also creates uncertainty and risk. It is this risk that we wish to address and contain. In this thesis, we review recently developed stochastic models of physical and financial aspects of deregulated electricity markets and research algorithms to utilize these models to hedge risk. First, we consider the issue of calibrating these models to historical data. Once the models are calibrated sufficiently, we discuss two major frameworks for hedging risk optimally. We begin by first developing a method for static hedging optimization, where we optimize a hedging strategy from a fixed point of time over a finite delivery period. Then we develop a more robust dynamic optimization, where the hedging strategy is continuously improved over a finite hedging period for a finite delivery period. A very lucid and recent motivation for the research in this thesis comes from California, where deregulation took place five years ago. Within the last year, the spot market behaved erratically, causing utility companies to plummet financially, ultimately resulting in many declaring bankruptcy and requiring the state of California to intervene so that California did not fall dark. The hedging optimization algorithms developed in this thesis could be used in deregulated electricity markets to possibly avoid a repetition of the situation that occurred in California.

Thesis Supervisor: Marija Ilić
Title: Senior Research Scientist

Para Uio.

Acknowledgments

I would like to express my deepest gratitude to my thesis advisor, Dr. Marija Ilić. I will always be in debt for her generous act of kindness during a period when I was lost academically. Marija also provided a warm family-like atmosphere in which her students are comfortable and productive. It was a privilege working with Dr. Ilić.

I would also like to thank Dr. Petter Skantze. The research in this thesis is derived directly from his Ph.D. dissertation. Petter and I collaborated extensively and a large portion of the research contained in this thesis is influenced by our interaction. Dr. Skantze's guidance during my research will not be forgotten.

I am also grateful to my fellow graduate students with whom I've had many lively discussions, both academic and otherwise. Particularly, I wish to thank Özge Gözüm for our "therapy sessions." I also wish to thank future graduate student David Matsa for his econometric insight.

Last but not least, I would like to thank all my friends and family. Their support during the course of this last year gave me the motivation to push harder. Specifically, I wish to thank my parents for always believing in me. However, this thesis would not have been possible without my grandfather, who first instilled in me the desire to attend MIT. It is to him I dedicate this thesis.

This research was supported by the Consortium on Competitive Electric Power Systems at the MIT Energy Laboratory through a research assistantship at the MIT Laboratory for Information and Decision Systems (LIDS).

Contents

List of Figures

Chapter 1

Introduction

Electricity markets in the United States have traditionally existed as monopolies and oligopolies. To prevent undesirable economic conditions, these electricity markets have traditionally been governmentally regulated. In 1978, the airline industry was deregulated, creating competition and a consumer-friendly airline market. The airline deregulation was arguably a success. In recent years, electricity markets have undergone a similar deregulation, attempting to create more economically desirable consumer markets, motivated by the airline deregulation. However, the electricity markets have not experienced the same success as the airline market.

The electricity market is unique and so is its deregulation. This market is a special case of a commodity market due to the hard to store nature of electricity. This fact creates a financial market that does not behave as traditional commodity markets do. Deregulation creates a system where the market functions as a spot market, with electricity spot prices set each hour. The market functions as follows. Generator entities and Load Serving entities (LSE) submit supply and demand bids to a central controlling Independent System Operator (ISO), which then creates an aggregate supply and demand curve, which in turn sets the equilibrium price and quantity of electric power to be traded. Dependent on the specific electricity market (such as California), 20-80% of total electric power is traded on the spot market. For more information, please see [6] and [12]. For simplicity, this thesis assumes the market is a daily spot market, rather than an hourly spot market.

The supply and demand bids are created based on forecasts of customer usage. However, occasionally, unexpected surges or drops in customer demand will result in monetary losses

for LSEs. For example, if customer demand for power drops unexpectedly to a very low level, the LSE is still required to buy the total amount that the LSE's bid requires, and loses the extra power not used by the customer, resulting in a monetary loss, perhaps very large. Also, if customer demand surges, then the LSE is required to purchase extra power to meet the extra customer demand from a standby generator, which will necessarily charge a much higher price. Again, the LSE will suffer financially.

The main focus of this thesis is on developing algorithms for hedging against these deviations from customer demand forecasts utilizing electricity forward market contracts. The use of forwards must be limited, however; if too much of the electricity load is supplied by forward contracts, the spot market will collapse, rendering the deregulated system a failure. Hence, we require bidding into the spot market, and will assume that hedging quantities of electricity will only be allowed to be a fraction of the total market electricity load. We will look at both static and dynamic hedging strategies and multiple methods for solving such problems. We will also discuss the advantages and disadvantages of these methods.

It is also interesting and motivating to look at a case study of deregulation. In 1996 the California Electricity Market underwent generation deregulation, attempting to create an almost perfect competitive market. The motivation was a guaranteed 20% cut in costs for residential and small business customers through 2002. During the first few years, the deregulation appeared to be going well. However, in the last few months of 2000, the economy boomed and power demand rose. This unexpected surge in consumer demand shocked the spot market - the deregulation was back-firing. The spot market has reached an equilibrium price that is a magnitude larger than the fixed price at which the LSEs are selling electricity to consumers. The large LSEs in California, PG&E and Southern California Edison, have been losing vast amounts of money in the last few months. With every megawatt of electricity these providers sell, they lose money. Once cornerstones of financial security, these large companies have recently been categorized junk bonds. Without government intervention, these LSEs who have no credit-worthiness, are not able to purchase power from generators to sell to the consumer. In the worst case scenario, California will plunge into darkness. Had a hedging strategy been implemented earlier, perhaps California energy providers would not be in the unfortunate situation they are in now. Perhaps future LSEs will use ideas presented in this thesis so that the unfortunate

situation existing in California will never rear its ugly head again.

1.1 Problem Formulation

In order to begin investigating meaningful hedging strategies for deregulated electricity markets, we must first provide a sufficient model for describing the dynamics of the market. We have chosen to implement a mathematical stochastic model in order to capture as much of the dynamics of the market as possible. With a mathematical model, we will have the liberty to implement hedging strategies and optimization methods mathematically as well, resulting in highly desirable results.

1.1.1 Stochastic Models of Electricity Load and Spot Price

A Four Factor stochastic model of electricity load and spot price has been developed in [19]. In this thesis, we will use a simplification of this model for simulations. The stochastic processes that represent the electricity load and spot price are defined as mean-reverting processes that attempt to capture seasonalities in electricity (see [19]). In particular, the model attempts to capture monthly, weekly, and daily trends (e.g. a surge in summer and winter months due to increased use of air conditioning and heaters, respectively). The actual simplified model used in this thesis is presented below. The parameters of these models are calibrated with historical data, and more information can be found in [19]. Superscripts indicate the process being modeled, either electricity load (L), or the supply sub-process (b). Subscripts indicate the time scale of the process (or parameter), either daily (d) or monthly (m). Also, note that a super-score indicates an average. The sigmas (σ) are the respective volatilities of the corresponding processes. The z variables are standard normal random variables used as seeds to the stochastic processes. The alpha (α) and beta (β) values are the mean-reverting parameters of the stochastic processes. The lambda's (λ) are drift factors and the mu's (μ) are average monthly shapes of the processes. The spot price (s) is given as a function of the two underlying stochastic processes (L and b). Again, more details can be found in [19].

Load stochastic model

$$L_{d+1} - L_d = \alpha(\overline{L}_m - L_d) + \sigma_d^L z_d^L \tag{1.1}$$

$$\overline{L}_m = \mu_m^L + \delta_m^L \tag{1.2}$$

$$\delta_{m+1}^L - \delta_m^L = \lambda^L + \sigma_m^{\delta^L} z_m^{\delta^L} \tag{1.3}$$

Price stochastic model

$$b_{d+1} - b_d = \beta(\overline{b}_m - b_d) + \sigma_d^b z_d^b \tag{1.4}$$

$$\overline{b}_m = \mu_m^b + \delta_m^b \tag{1.5}$$

$$\delta_{m+1}^b - \delta_m^b = \lambda^b + \sigma_m^{\delta^b} z_m^{\delta^b} \tag{1.6}$$

$$s_d = e^{aL_d + b_d} \tag{1.7}$$

These processes, once calibrated correctly with historical data, are able to produce simulations of load and spot price very closely approximating real data. However, historical calibration is not the only means to create meaningful parameters for these models. Other methods include extending historical calibration to incorporate projections. These projections include, but are not limited to: equipment outages, water and solar power contributions, etc.

However, historical calibration does have its limitations, especially since deregulation is in its infancy. As the pace of deregulation increases, existing market dynamics are bound to change drastically as well [10]. If the dynamics change sufficiently, historical calibration is all but useless. Another method to calibrate these models is to use forward curve and derivative information. Calibration will be fully discussed in the next chapter.

1.1.2 Value Functions

Using these models we are able to develop cash flow models as well. Under no other assumptions, we can represent our cash flow as such

$$U = \sum_{m=1}^{N} \sum_{d=mM+1}^{M(m+1)} l_d(R - s_d) \tag{1.8}$$

This equation simply is the total cash flow a LSE receives if it supplies customers with power at a fixed rate R over a hedging period. The hedging horizon used to calculate this flow is N months, with M days per month. If we look at the argument to the double summation, we see that it is the daily load multiplied by the difference in the cost of selling and buying a unit MW of electricity $(R - S_d)$. This is the cash flow for one day. Summing over all days in a month (the inner summation), we get the cash flow for one month. Once again taking a summation over all months in the hedging period (the outer summation), we have the total cash flow over the entire hedging period. Now we can consider another cash flow function, incorporating hedging with electricity forward contracts

$$V = \sum_{m=1}^{N} \sum_{d=mM+1}^{M(m+1)} l_d(R - s_d) + q_m(s_d - F_m) \tag{1.9}$$

The q variables are the quantities of monthly forward contracts of electricity. The F variables are the respective prices of the contracts. The additional term is the quantity of forward contracts bought multiplied by the difference of the spot price and the forward contract price. Since the forward contract quantity and price evolve on a monthly scale and the spot price evolves on a daily scale, we can interpret the summation of this term as a hedge against the volatility of the electricity market with the assumption that some power will be provided by the forward contacts at maturity. This is the equation we are going to optimize, using the quantities of contracts as the decision variables.

Chapter 2

Calibration

Now that we have reviewed the stochastic models for deregulated electricity markets, we must calibrate them to historical data so that they may be useful in our optimizations. There are many methods for calibrating a model to a given set of data. We will consider three different methods and discuss their applicability to our models. First, we shall discuss basic linear regressions under the assumption of time-scale separation between various subprocesses. Then we will develop maximum likelihood estimation using Kalman filtering. We will conclude with an analysis of the applicability of a specific econometric model into which our models can be manipulated.

Though we develop these calibration methodologies, we must also warn about calibrating to historical data, especially in deregulated electricity markets. Since deregulation is such a recent development, the market is very dynamic. Even the most accurate historical calibrations can be useless if the dynamics of the market changes sufficiently. We must always keep this in mind as we calibrate on historical data. In the future, once the deregulated markets mature and approach a steady-state, historical calibration will be much more reliable.

2.1 Linear Regression using Time-Scale Separation

In order to facilitate the calibration of the stochastic models to historical data, we first made a simplifying assumption that certain parameters evolve at different rates over different time scales than other parameters. For example, our model as presented previously, defines $\overline{L}_m$ and $\overline{b}_m$ as monthly parameters and the remaining as daily parameters. Though this may not

be reality, we assumed it so that the calibrations would be guaranteed to converge. Time-scale separation essentially broke down the calibration problem into two smaller, much easier to solve and independent, sub-calibration problems (monthly and daily parameter calibrations).

Once the time-scale separation was assumed, the monthly averages were simple to calculate. For example, to calculate the monthly average shape of load, we needed only to take averages over certain intervals of the historical data which corresponded to the different months. Mathematically,

$$\mu_m^L = \frac{1}{T_m - T_{m-1} + 1} \sum_{d=T_{m-1}}^{T_m} LoadData_d \tag{2.1}$$

and

$$\mu_m^b = \frac{1}{T_m - T_{m-1} + 1} \sum_{d=T_{m-1}}^{T_m} SupplyData_d \tag{2.2}$$

where $[T_{m-1}, T_m]$ is the time interval of the load data that corresponds to the month in question.

Once these monthly parameters were calculated, they were factored out. What remained were *daily* stochastic models where we ran principal component analysis followed by simple linear regressions to calculate the daily parameter values. The exact details may be found in [19]. However, a problem we encountered was lack of supply-side historical data. Calculating load parameters was straightforward since we had 18 years of historical data at our disposal. On the other hand, we had less than two years of price historical data, which limited the accuracy of our calibrations. But overall, time-scale separation linear regressions worked the best of all the methods we utilized, and its parameters are the ones used in all our results.

2.2 Maximum Likelihood Estimation utilizing Kalman Filtering

Another method we attempted to implement was maximum likelihood estimation coupled with Kalman filtering to estimate parameters of the stochastic model without assuming time-scale separation. Kalman filtering is usually used to estimate the state of a system,

though a modification can be implemented to estimate the parameters of our model. We began by first simplifying our load model to see if Kalman filtering would work. The simplified model was created by assuming zero monthly averages

$$L_{d+1} - L_d = -\alpha L_d + \sigma z_d \tag{2.3}$$

We begin by rewriting the stochastic models in a state-space formulation where L_d is our state, and M_d is our historical measurement of the state L_d

$$L_{d+1} = AL_d + w_d \tag{2.4}$$

$$M_d = L_d + \nu_d \tag{2.5}$$

where $w_d \sim N(0, Q)$ and $\nu_d \sim N(0, 0)$.

Notice that ν can be eliminated since we have perfect observations of our model and no observation noise. In most of the literature on Kalman filters, Q is assumed to be known. However, in [5] it is assumed unknown and a methodology is developed such that Q may be estimated. The methodology also allows us to estimate A. In theory, using this methodology, we will be capable of estimating $\alpha = 1 - A$ and $\sigma = \sqrt{Q}$. Essentially, we are estimating a state that we already know; however, a by-product of the estimation allows us to estimate values for parameters in our state-space description, which are a function of our stochastic model parameters.

The method first instructs us to go through a series of steps (details can of course be found in [5]) to derive the maximum likelihood function. We show an alternative derivation of the maximum likelihood function in terms of our simplified stochastic model parameters. The form of the maximum likelihood function is given in [5] as

$$J = \frac{1}{2} \sum_{t=1}^{T-1} (y_t - x_t)' N_{t+1|t}^{-1} (y_t - x_t) + log(det(N_{t+1|t})) \tag{2.6}$$

which can be manipulated into a function of our model parameters as follows

$$J = \frac{1}{2} \sum_{t=1}^{T-1} (y_t - x_t)' \frac{1}{P_{t+1|t}} (y_t - x_t) + log(det(P_{t+1|t})) \tag{2.7}$$

$$= \frac{1}{2}\sum_{t=1}^{T-1}\frac{(y_t - Ax_t)^2}{A^2P_t + Q} + log(A^2P_t + Q) \tag{2.8}$$

$$= \frac{1}{2}\sum_{t=1}^{T-1}\frac{(y_t - (1-\alpha)x_t)^2}{(1-\alpha)^2P_t + \sigma^2} + \log\left[(1-\alpha)^2P_t + \sigma^2\right] \tag{2.9}$$

where x_t is the estimate at time t, y_t is the actual measurement at time t, and P_t is the estimate error covariance matrix at time t. $N_{t+1|t}$ is the innovation matrix defined in [5] and $P_{t+1|t}$ is the conditional estimate error covariance matrix. The next step is to minimize the maximum likelihood function with respect to α and σ, which is presented in [5] by using Newton's Method

$$\Theta_{i+1} = \Theta_i - \rho_i H^{-1}(\Theta_i)\nabla J(\Theta_i) \tag{2.10}$$

where

$$\Theta = \begin{bmatrix} \alpha \\ \sigma \end{bmatrix} \tag{2.11}$$

and H is the Hessian of the maximum likelihood function

$$H(J(\alpha,\sigma)) = \begin{bmatrix} \frac{\partial^2 J(\alpha,\sigma)}{\partial\alpha^2} & \frac{\partial^2 J(\alpha,\sigma)}{\partial\alpha\partial\sigma} \\ \\ \frac{\partial^2 J(\alpha,\sigma)}{\partial\sigma\partial\alpha} & \frac{\partial^2 J(\alpha,\sigma)}{\partial\sigma^2} \end{bmatrix} \tag{2.12}$$

However, the implementation of Newton's method resulted in a singular Hessian matrix. We can not use Newton's method since that method requires us to invert the Hessian. We attempted to use a pseudo-inverse function, but that did not work either. We then tried using a gradient algorithm that did not require second order information. We implemented steepest descent, with and without a line search (whether or not the value of ρ_i is optimized)

$$\Theta_{i+1} = \Theta_i - \rho_i \nabla J(\Theta_i) \tag{2.13}$$

Though the procedure was able to run, we did not achieve any convergence results. However, according to [5], a singular Hessian is a very good indication that convergence was not possible. Another problem with using Kalman filtering is that the "innovation matrix," another matrix inverted in the estimation procedure, was singular as well. Given that

our simplified model was plagued with singularities, we decided that maximum likelihood estimation using Kalman filtering was not a good method to estimate the parameters of our complete model.

2.3 Econometric Model: Autoregressive Functions

The last method we investigated was using a specialized class of econometric models called autoregressive functions. These special functions are basically difference equations of the form

$$y_t = \phi_{t-1} y_{t-1} + \phi_{t-2} y_{t-2} + \cdots + \phi_{t-p} y_{t-p} + \delta + \epsilon_t \tag{2.14}$$

where δ is a constant and ϵ_t is a disturbance at time t. Assumptions behind this model are that the process is stationary

$$E[y_t] = E[y_{t-1}] = \cdots = \mu = \frac{\delta}{1 - \phi_1 - \cdots - \phi_p} \tag{2.15}$$

and

$$\sum \phi_i < \infty \tag{2.16}$$

which implies that μ is finite. Another assumption is that the ϵ_t are independent white noise random variables. Once these assumptions are met, we may then apply a maximum likelihood analysis to derive a log-likelihood function to optimize

$$L = -T log(\sigma_\epsilon) - \frac{\delta(\Phi_1, \cdots, \Phi_p)}{2\sigma_\epsilon^2} \tag{2.17}$$

where T is the number of observations and

$$\delta(\Phi_1, \cdots, \Phi_p) = \sum_t \epsilon_t^2 \tag{2.18}$$

and

$$\epsilon_t = \Theta(B) w_t, \epsilon_t \sim N(0, \sigma_\epsilon) \tag{2.19}$$

where $\Theta(B)$ is a lag operator for an autoregressive function of order p

$$\Theta(B) = (1 - \Phi_1 B - \Phi_2 B^2 - \cdots - \Phi_p B^p) \tag{2.20}$$

It turns out that this log-likelihood function can be minimized by applying multiple linear regressions on the original autoregressive function. Therefore, if we can manipulate our model into an autoregressive function, and if all assumptions hold, we will be able to calibrate our stochastic model using simple regression techniques. Let us reexamine our simplified, but second order, load model and derive an autoregressive model

$$L_{d+1} - L_d = -\alpha L_d + \sigma z_d \tag{2.21}$$

$$\delta_{d+1} - \delta_d = \sigma^\delta z_d^\delta \tag{2.22}$$

We will now define as in [16]

$$w_d = L_d - \delta_d \tag{2.23}$$

and derive an autoregressive model

$$w_{d+1} - \delta_{d+1} - w_d + \delta_d = -\alpha w_d + \alpha \delta_d + \sigma z_d \tag{2.24}$$

rearranging

$$\begin{aligned} w_{d+1} &= (1-\alpha) w_d + \delta_{d+1} + (\alpha - 1)\delta_d + \sigma z_d & (2.25) \\ &= (1-\alpha) w_d + \alpha \delta_d + \sigma^\delta z_d^\delta + \sigma z_d & (2.26) \end{aligned}$$

notice that

$$\alpha \delta_d = w_{d+1} - (1-\alpha) w_d - \sigma^\delta z_d^\delta - \sigma z_d \tag{2.27}$$

and it follows that

$$
\begin{aligned}
w_{d+2} &= (1-\alpha)w_{d+1} + \alpha\delta_{d+1} + \sigma^{\delta} z^{\delta}_{d+1} + \sigma z_{d+1} && (2.28)\\
&= (1-\alpha)w_{d+1} + \alpha(\delta_d + \sigma^{\delta} z^{\delta}_{d}) + \sigma^{\delta} z^{\delta}_{d+1} + \sigma z_{d+1} && (2.29)\\
&= (1-\alpha)w_{d+1} + \alpha\delta_d + \alpha\sigma^{\delta} z^{\delta}_{d} + \sigma^{\delta} z^{\delta}_{d+1} + \sigma z_{d+1} && (2.30)\\
&= (1-\alpha)w_{d+1} + w_{d+1} - (1-\alpha)w_d - \sigma^{\delta} z^{\delta}_{d} - \sigma z_d + \alpha\sigma^{\delta} z^{\delta}_{d} + && (2.31)\\
&\quad \sigma^{\delta} z^{\delta}_{d+1} + \sigma z_{d+1} \\
&= (2-\alpha)w_{d+1} + (\alpha-1)w_d + (\alpha-1)\sigma^{\delta} z^{\delta}_{d} + \sigma^{\delta} z^{\delta}_{d+1} - \sigma z_d + \sigma z_{d+1} && (2.32)
\end{aligned}
$$

which results in

$$
w_d = (2-\alpha)w_{d-1} + (\alpha-1)w_{d-2} + \epsilon_d \tag{2.33}
$$

where ϵ_d is zero mean noise. It would appear that autoregressive functions would be ideal to estimate the parameters of our model. However, a subtle assumption is violated in our derivation. To use an autoregressive parameter estimation, our noise ϵ_d must be white noise. This means that it must have zero mean (which it does) AND have independent time instances. However, we may notice that from our derivation above, ϵ_d depends on random seeds in two different time periods. Hence, different instances of noise are not independent, and the noise is not white noise. Thus, we conclude that autoregressive models are not well suited for calibrating our problem.

2.4 Conclusions on Calibration

After completing these different calibration analyses, we have come to the conclusion that certain simplifying assumptions must be made in order to attain convergence during model calibration. The results that follow are calibrated on the time-scale separation assumption. More complex calibration schemes failed to converge for a variety of reasons. Singularities plagued Kalman filtering calibration. While attempting to use econometric analysis, our stochastic model could not help but violate certain basic assumptions of the autoregressive model. Certainly, an open area of research is to investigate better methods and techniques for calibration so that our model's parameters can be estimated without making simplifying

assumptions. However, further research on our part is out of the scope of this thesis.

Chapter 3

Static Optimization Algorithms

In this chapter, we will examine two static optimization methods for maximizing the hedged cash flow V, given constraints on the risk and uncertainty derived from the volatile nature of the electricity markets. Both methods will use monthly electricity forward contracts as the decision variables. First, we will examine the maximization of the hedged value function with a Value at Risk constraint. Then we will examine the same optimization using a Mean Variance formulation. Discussions of the advantages and disadvantages will be included in the treatment of both methods. We will also present a treatment of the speculation sub-problem, where we concentrate on the behavior of the cash flow V as a function of quantities of monthly forward electricity contracts. In addition, complexity analyses of these methods will be presented in their treatment.

3.1 Value at Risk

The Value at Risk formulation attempts to limit the effect of market volatility by introducing a probabilistic constraint while maximizing the expected value of the cash flow function

$$\max_{q_m} E[V] \tag{3.1}$$

$$s.t.\ \Pr[V \leq V_{min}] \leq X \tag{3.2}$$

The constraining probability is defined by two parameters, V_{min} and X. V_{min} is a user-defined lower threshold for V, and X is a limiting probability. These two parameters can

be interpreted as the largest acceptable probability X (e.g. 1%) that the cash flow V drops below a critical value V_{min} (e.g. the minimum cash flow a company must have to avoid bankruptcy). Though simple to grasp intuitively, the analytical solution to this optimization problem can not be found in closed form, and we have attempted to optimize the hedging problem using simulations.

3.1.1 Visualizing the Search Space

Since we are dealing with a probabilistic constraint, we are not certain of the characteristics of the feasible region of the q_m variables. The feasible region is defined to be all portfolios of monthly forward contracts q such that the probabilistic Value at Risk constraint is not violated. It is important to note that the search space is of dimension N, since our decision variables are $q_1, \ldots, q_N$. We are interested in whether or not the feasible search space is convex. If it is, we may reference linear programming theory, and note that for a convex feasible search space and linear objective function, the optimal solution will always be an extreme point (i.e. a vertex of the feasible search space). If it is not, we would have to invoke a non-linear optimization algorithm in order to find an optimal solution. Simulated annealing (see [4]), a probabilistic algorithm, is a good candidate for solving our problem if we discover a non convex feasible region.

To begin visualizing the feasible search space, we make some simplifying assumptions. First we examine V as a one dimensional function (i.e. $V(q_1)$) so that we are only looking at the optimization over the first month. We first assume load is constant, then relax load to its stochastic model. We also assume the forward contract price (F_1) is equal to the expected value of the spot price, calculated from the first month's simulations and that R is greater than the expected value of the spot price. For these preliminary simulations, we assign V_{min} as 80% of the average value of V over all values of q_1, a reasonable minimum value. We were able to approximate the constraint probabilities by using simulations. First, we simulated T sets of load and spot price data. Then, for a given value of q_1 we record the number of simulations that violate V_{min}. Consolidating all this information we are able to create an approximate probability distribution as a function of q_1. Using these approximations, we are able to determine which values of q_1 will be feasible for use in our objective function. We also note very few discrepancies between a constant load and the stochastic model; we therefore include the model in further trials. We summarize the results in the following

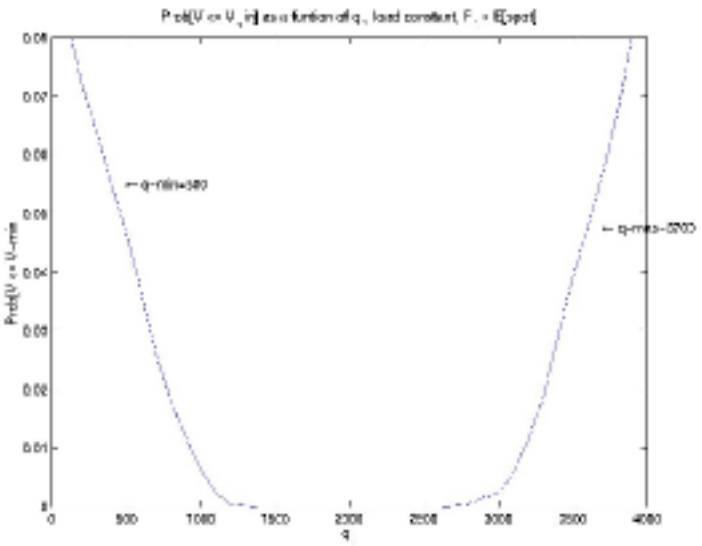

Figure 3-1: Probability[V(q1) $\leq V_{min}$] $\leq$ .05, holding load constant and assuming F1 = E[spot]

plots.

It follows from these graphs that the feasible search region for the first month's hedging optimization is convex (i.e. it is a closed interval, namely $[q_{1,min}, q_{1,max}]$). Since the feasible region is convex, we only need to compare the value of $V(q_{1,min})$ and $V(q_{1,max})$ and choose the maximum.

3.1.2 Forward Contract Price not equal to E[spot]

How does the feasible search space change if the forward contract price differs from the expected value of the spot price? In particular, we wish to ascertain that convexity is preserved. The following plots display the behavior of the feasible search space if we allow the forward contract price to be greater than or less than the expected value of the spot price, assuming stochastic load.

These plots show what should be clear intuitively. If the forward contract price is less than the expected price of the spot, one should try to buy as many forward contracts as possible, thus reducing the risk as the number of q rises. The opposite argument works when the forward contract price is greater than the expected spot price. We again notice that the feasible search spaces are convex: $[-\infty, q_{max}]$ and $[q_{min}, \infty]$, though with an infinite optimal value for q, namely $-\infty$ for F > E[spot] and ∞ for F < E[spot]. We also calculated

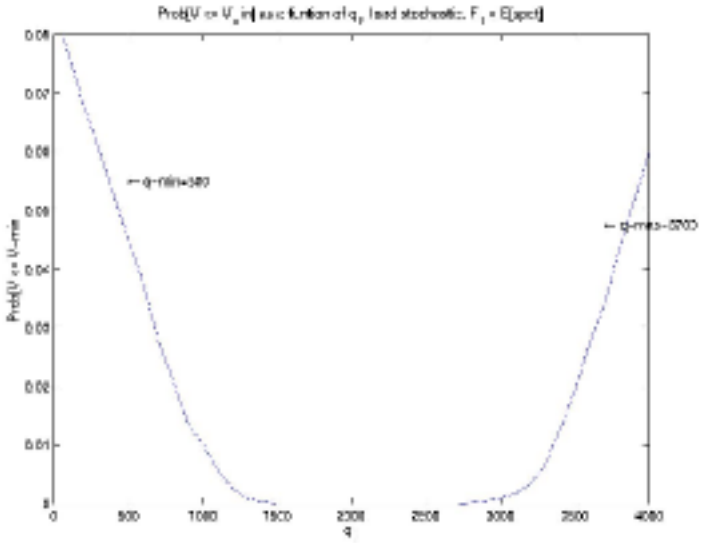

Figure 3-2: Probability[V(q1) $\leq V_{min}$] $\leq$ .05, relaxing load to its stochastic model assuming F1 = E[spot]

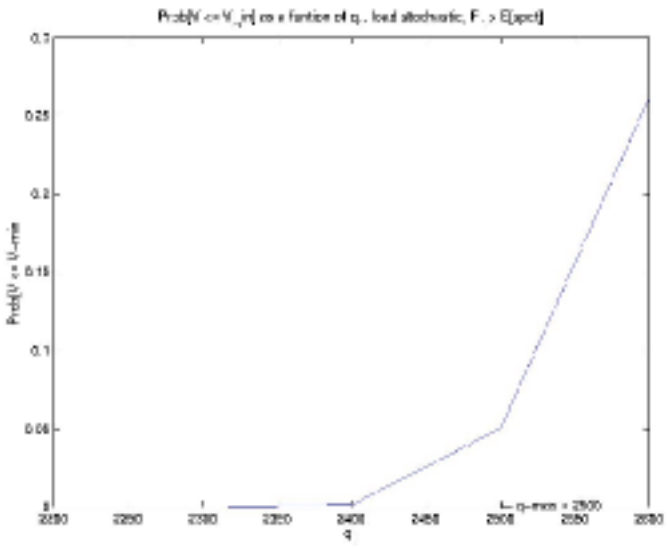

Figure 3-3: Probability[V(q1) $\leq V_{min}$] $\leq$ .05, relaxing load to its stochastic model and assuming F1 > E[spot]

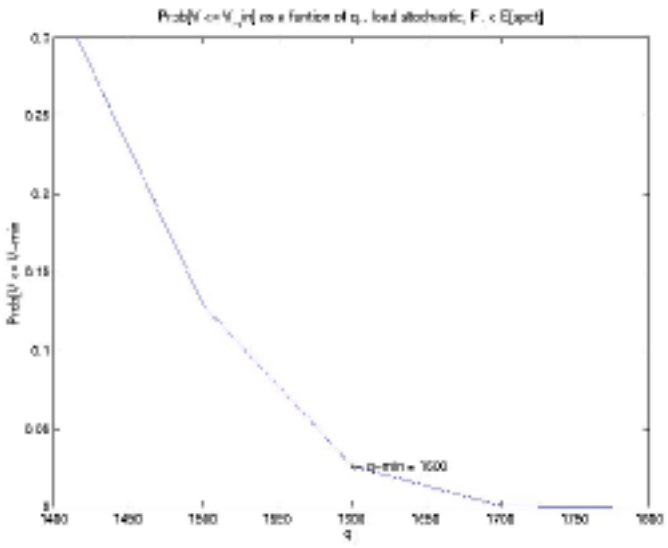

Figure 3-4: Probability[V(q1) $\leq V_{min}$] $\leq$ .05, relaxing load to its stochastic model and assuming F1 $<$ E[spot]

the variance of V as a function of q_1. We show the plot below. We conclude, that for one month, the feasible search spaces are convex.

3.1.3 Generalization to N dimensions and its Consequences

To solve the one month optimization, we performed T simulations of load and spot price data and then evaluated the objective function V over a range of q_1 for all T simulations.This exhaustive approach allowed us to find approximations to the constraining probabilities (equal to the constraining probabilities as $T \to \infty$). Referencing linear programming theory and the fact that the feasible search regions are convex, we then find extreme points, and choose the optimal value of V over the extreme points. In practice, we allow the range of q_1 to be discretized to a vector of size Q. This optimization algorithm is of order $O(TQ)$.

Suppose we wish to generalize this algorithm to N months. We would then need to simulate T sets of load and spot price data, and then evaluate the objective function V for all the $q_1, \cdots, q_N$ decision variables of size Q. We again would find extreme points (or boundary points if extreme points are ambiguous) of the feasible region and then choose the optimal N-tuple of the q variables that would maximize V. However, this algorithm is of order $O(TQ^N)$, which is very inefficient for any values of T, Q, and N greater than trivial values. This algorithm is outside the class of polynomial solvable problems.

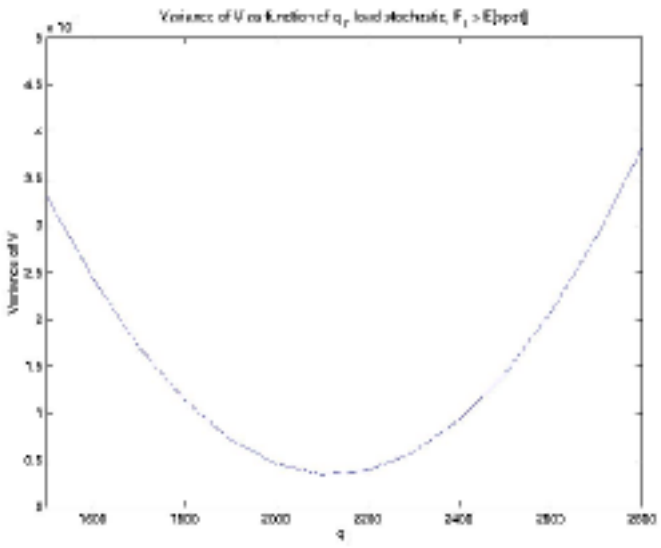

Figure 3-5: Variance of V(q1), with load assuming its stochastic model and assuming F1 = E[spot]

To give us intuition about the general problem, we simulate a two month optimization assuming forward prices (first and second month) are equal to the expected values of the spot prices calculated over the respective months. We are able to examine a 3D plot of the probability over a range of q_1 and q_2. We provide two views to provide more intuition and understanding.

We now examine the feasible region and notice that it still remains convex for a two month simulation. Since the feasible region is amorphous and there is no clear extreme points, we will consider the entire boundary as candidates for the objective function. Since we are considering a discretized range for the q variables, the boundary will necessarily be a finite set of values to evaluate. We are also curious about the variance as a function of the two monthly q variables, and we include this plot as well.

3.1.4 Generalization with Forward Prices not equal to E[spot]

As we examined in the one dimensional case, we investigate the behavior for the situation where the forward contract price in month one is less than the expected value of the spot price, while the forward contract price in month two is greater than the expected value of the spot price. The following plots show the resulting behavior and demonstrate that convexity is still maintained.

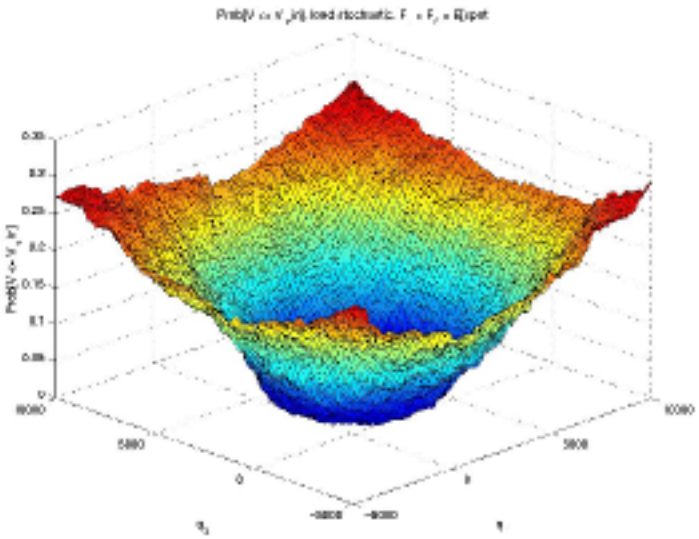

Figure 3-6: Probability[V(q1,q2) $\leq V_{min}$] $\leq$ .05, relaxing load to its stochastic model and assuming F1 = F2 = E[spot], view 1

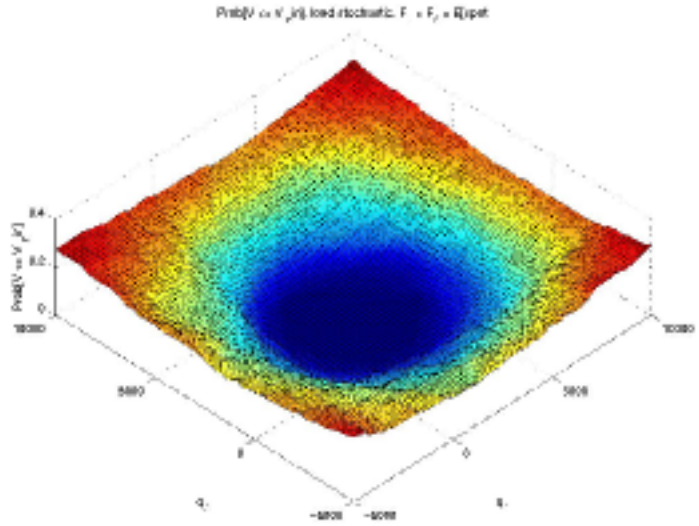

Figure 3-7: Probability[V(q1,q2) $\leq V_{min}$] $\leq$ .05, relaxing load to its stochastic model and assuming F1 = F2 = E[spot], view 2

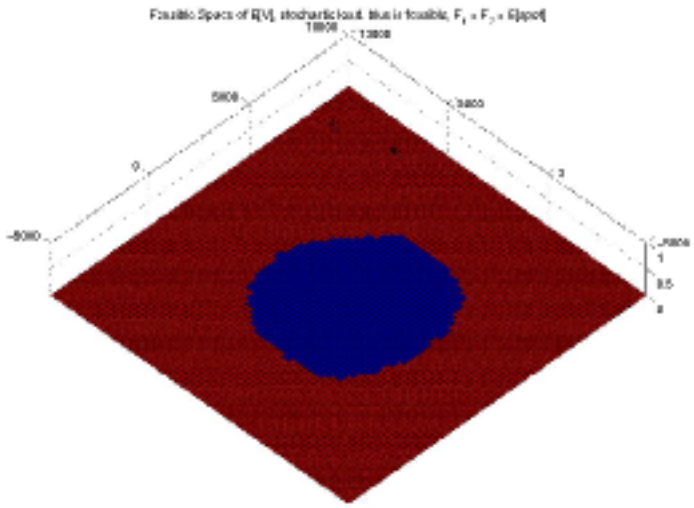

Figure 3-8: Feasible Region for V(q1,q2), relaxing load to its stochastic model and assuming F1 = F2 = E[spot]

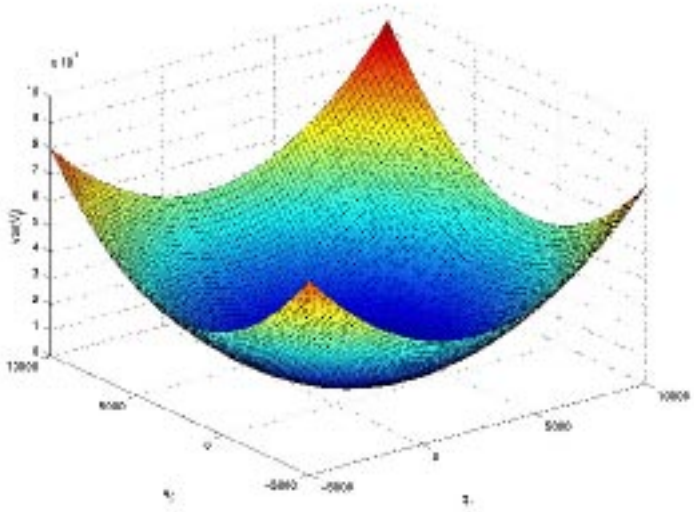

Figure 3-9: Variance of V(q1,q2), relaxing load to its stochastic model and assuming F1 = F2 = E[spot]

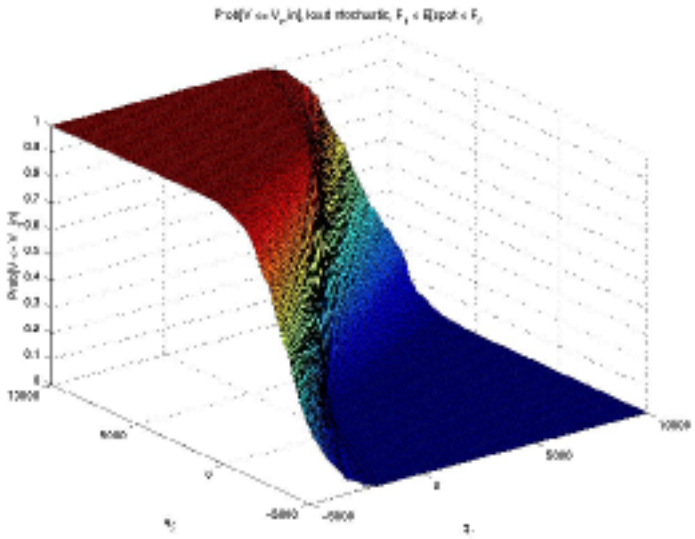

Figure 3-10: Probability[V(q1,q2) $\leq V_{min}$] $\leq$ .05, relaxing load to its stochastic model and assuming F1 $<$ E[spot] $<$ F2, view 1

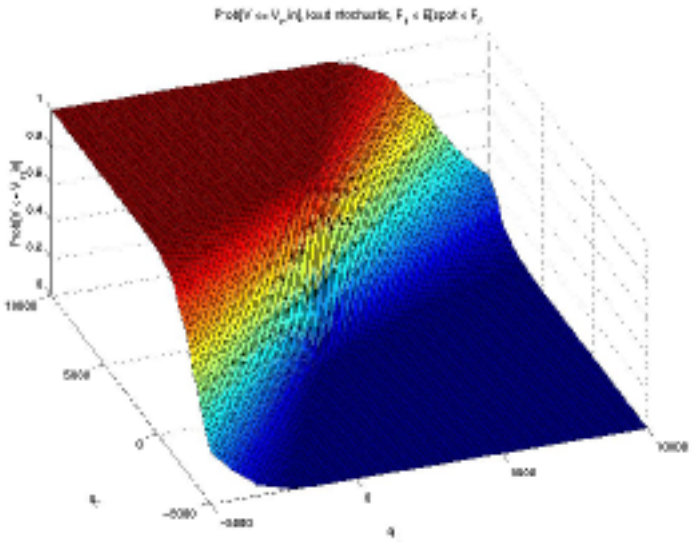

Figure 3-11: Probability[V(q1,q2) $\leq V_{min}$] $\leq$ .05, relaxing load to its stochastic model and assuming F1 $<$ E[spot] $<$ F2, view 2

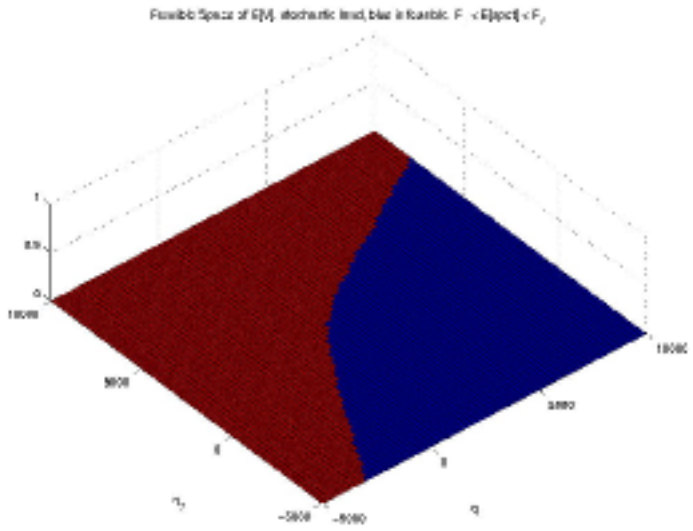

Figure 3-12: Feasible Region for V(q1,q2), relaxing load to its stochastic model and assuming F1 < E[spot] < F2

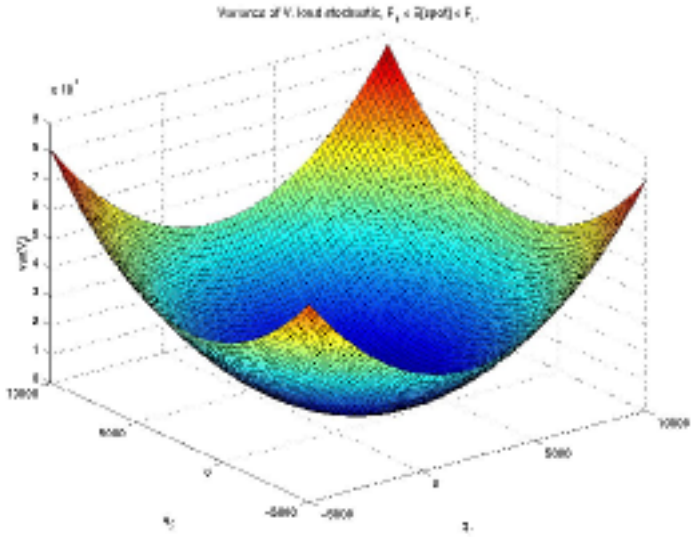

Figure 3-13: Variance of V(q1,q2), relaxing load to its stochastic model and assuming F1 < E[spot] < F2

3.1.5 Efficient Reformulation Attempt

Let us reexamine our objective function and probabilistic constraint. The objective function can be rewritten as

$$V = \sum_{k=M(1)+1}^{M(1+1)} l_k(R - s_k) + q_1(s_k - F_1) + \cdots + \sum_{k=MN+1}^{M(N+1)} l_k(R - s_k) + q_N(s_k - F_N) \quad (3.3)$$

We notice that V is linear in the q variables, and can be decoupled easily into N different functions V_i, $i = 1, \ldots, N$

$$V_i = \sum_{k=M(i)+1}^{M(i+1)} l_k(R - s_k) + q_i(s_k - F_i) \quad (3.4)$$

such that

$$V = \sum_{i=1}^{N} V_i \quad (3.5)$$

Therefore, V can be easily simplified to be considered on a monthly basis. However, the probabilistic constraint $Prob[V \leq V_{min}]$ is a constraint over all the months jointly. Unfortunately, there does not seem to be a way to decouple the probabilistic constraint into monthly components. Hence, the Value at Risk formulation optimization is intrinsically exponential in the number of months, a very undesirable characteristic. Thus, we wish to investigate other formulations such that the objective function and constraints can be decoupled to a monthly basis, thus eliminating the exponential computational factor. Therefore, we now examine the Mean Variance formulation.

3.2 Mean Variance

We can use Mean Variance to maximize the expected value of the cash flow while limiting the risk. Our first formulation, Value at Risk, explicitly introduced a probabilistic constraint in addition to the objective function. We can, however, produce the same essential constraint by eliminating the probabilistic constraint and adding another term to the objective function, namely a weighted variance. The Mean Variance formulation results

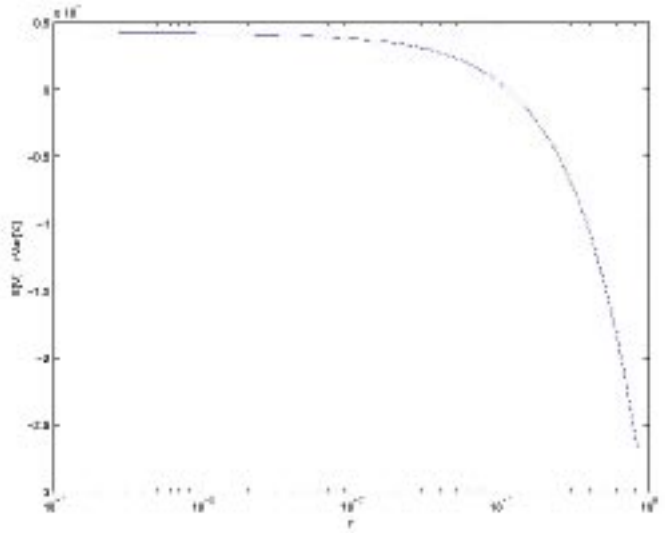

Figure 3-14: E[V] - rVar[V] plotted as a function of r, assuming F1 = E[spot]

$$\max_{q_m} E[V] - rVar[V] \tag{3.6}$$

However, we are uncertain what value of r we should use in our maximization. We therefore simulate the maximum value of the objective function as a function of r.

These plots provide us with very interesting information. We can determine the value of r for which the objective function is equal to zero. At this point, the components of the objective function are weighted equally, namely the expected value of V, which we want to maximize, and the variance of V, which we want to minimize. Once we know this threshold value of r and the behavior of the graph, we will be able to choose an r according to our preferences. Let us define this threshold value of r to be r*. If we choose a value of r less than r* we are putting more importance on the expected value of V and less emphasis on the variance. Using this information, management will be able to choose a value of r relevant to the situation at hand, and then look up the according q variables. It is also important to note the logarithmic scales of the graphs. Preliminary simulations also suggest this behavior of the objective function as a function of r over multiple months is very similar, if not identical, to the behavior over one month, shown previously. However, certain assumptions are made in determining the value of r in this fashion. This derivation is based on simulations over a finite set of q values. Therefore, our calculated r-curve will

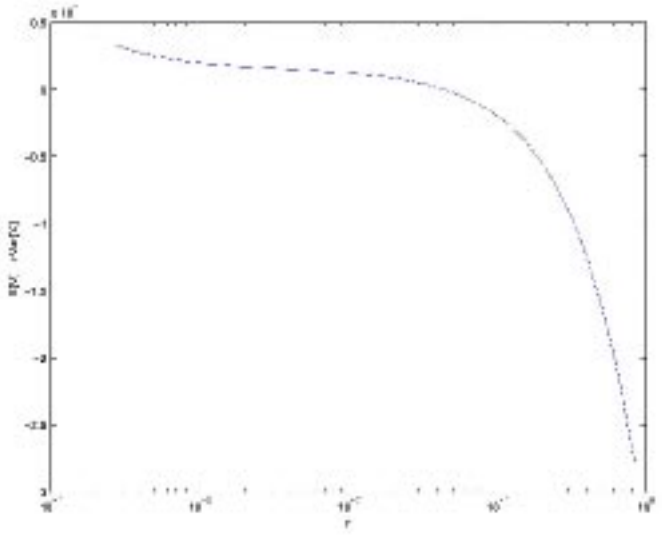

Figure 3-15: E[V] - rVar[V] plotted as a function of r, assuming F1 > E[spot]

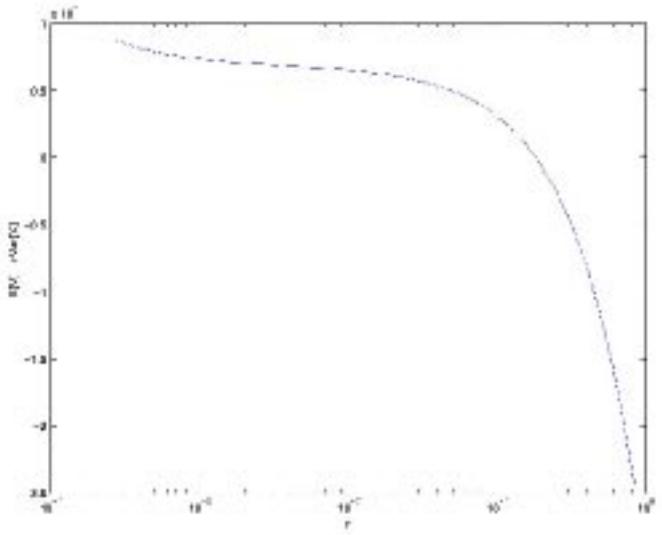

Figure 3-16: E[V] - rVar[V] plotted as a function of r, assuming F1 < E[spot]

be a function of the underlying set of q, chosen a priori. However, if the set of q is chosen so that it is representative of all feasible choices of q, then this calculation is valid. However, how this feasible set of q would be created is an open problem.

Given that we have a value of r, the mean variance formulation is comparable to the Value at Risk formulation. We showed earlier that the Value at Risk optimization solution is exponential in the number of months of the hedging period. The Mean Variance computation is also exponential in the number of months because, generally, we have no way to calculate the variance of V efficiently or in closed form, and must use simulations. However, a sub-problem of the hedging optimization can be solved very efficiently using the Mean Variance framework since a decoupling is possible.

3.2.1 Speculation Sub-Problem

Suppose we consider a simplification of the cash flow function. We are interested in the speculation problem, where only the terms dependent on the q decision variables are considered. We are speculating on how the forward contracts alone behave, and are not concerned with hedging

$$W = \sum_{m=1}^{N} \sum_{d=mM+1}^{M(m+1)} q_m(s_d - F_m) \tag{3.7}$$

Using this simplified cash flow function, we are able to solve the corresponding Mean Variance optimization very efficiently. We can rewrite W as follows

$$W = (c_1, c_2, \ldots, c_N) \bullet (q_1, q_2, \ldots, q_N) = C^T Q \tag{3.8}$$

where

$$c_m = \sum_{d=mM+1}^{M(m+1)} (s_d - F_m) \tag{3.9}$$

However, we are interested in the expected value of W, so we derive the following

$$E[W] = E[C^T Q] = E[C^T]Q = D^T Q \tag{3.10}$$

where

$$d_m = \frac{1}{T}\sum_{j=1}^{T}[\sum_{d=mM+1}^{M(m+1)}(s_d - F_m)] = \frac{\sum_{j=1}^{T} c_m}{T} \tag{3.11}$$

Computing d_m requires O(TM) steps to compute. Since there are N variables d_m, the number of steps to compute the vector D is O(TMN).

We now wish to represent the variance of W. Since we are working with the simplified function W, were are able to make the critical step in solving the optimization efficiently by writing the variance of W as follows (which is not possible if we are working with V)

$$Var[W] = Q^T \Sigma Q \tag{3.12}$$

where the covariance matrix is defined as

$$\begin{aligned}
\Sigma_{ij} &= Cov[\sum_{d=iM+1}^{M(i+1)}(s_d - F_i), \sum_{d=jM+1}^{M(j+1)}(s_d - F_j)] && (3.13)\\
&= E[(\sum_{d=iM+1}^{M(i+1)}(s_d - F_i) - d_i)(\sum_{d=jM+1}^{M(j+1)}(s_d - F_j) - d_j)] && (3.14)\\
&= E[(\sum_{d=iM+1}^{M(i+1)}(s_d - F_i))(\sum_{d=jM+1}^{M(j+1)}(s_d - F_j)) && (3.15)\\
&\quad -d_i\sum_{d=jM+1}^{M(j+1)}(s_d - F_j) - d_j\sum_{d=iM+1}^{M(i+1)}(s_d - F_i)]\\
&= E[\sum_{d=iM+1}^{M(i+1)} s_d \sum_{d=jM+1}^{M(j+1)} s_d - MF_i\sum_{d=jM+1}^{M(j+1)} s_d && (3.16)\\
&\quad -MF_j\sum_{d=iM+1}^{M(i+1)} s_d + M^2F_iF_j - d_i\sum_{d=jM+1}^{M(j+1)} s_d + d_iMF_j\\
&\quad -d_j\sum_{d=iM+1}^{M(i+1)} s_d + d_jMF_i + d_id_j]\\
&= E[\sum_{d=iM+1}^{M(i+1)} s_d \sum_{d=jM+1}^{M(j+1)} s_d] - MF_iE[\sum_{d=jM+1}^{M(j+1)} s_d] && (3.17)\\
&\quad -MF_jE[\sum_{d=iM+1}^{M(i+1)} s_d] + M^2F_iF_j - d_iE[\sum_{d=jM+1}^{M(j+1)} s_d] + d_iMF_j
\end{aligned}$$

$$
\begin{aligned}
& -d_j E[\sum_{d=iM+1}^{M(i+1)} s_d] + d_j M F_i + d_i d_j \\
= \; & E[\sum_{d=iM+1}^{M(i+1)} s_d \sum_{d=jM+1}^{M(j+1)} s_d] - (MF_i + d_i) E[\sum_{d=jM+1}^{M(j+1)} s_d] \\
& -(MF_j + d_j) E[\sum_{d=iM+1}^{M(i+1)} s_d] + M^2 F_i F_j + d_i M F_j + d_j M F_i + d_i d_j
\end{aligned} \tag{3.18}
$$

The remaining expectations in the covariance matrix can be found using simulations

$$
E[\sum_{d=iM+1}^{M(i+1)} s_d \sum_{d=jM+1}^{M(j+1)} s_d] = \frac{\sum_{k=1}^{T} [\sum_{d=iM+1}^{M(i+1)} s_d \sum_{d=jM+1}^{M(j+1)} s_d]}{T} \tag{3.19}
$$

$$
E[\sum_{d=iM+1}^{M(i+1)} s_d] = \frac{\sum_{k=1}^{T} [\sum_{d=iM+1}^{M(i+1)} s_d]}{T} \tag{3.20}
$$

$$
E[\sum_{d=jM+1}^{M(j+1)} s_d] = \frac{\sum_{k=1}^{T} [\sum_{d=jM+1}^{M(j+1)} s_d]}{T} \tag{3.21}
$$

The calculation of each of these expectations requires $O(TM^2)$, $O(TM)$ and $O(TM)$ steps respectively. Therefore, since the covariance matrix is an NxN matrix, the number of steps required to construct it is $O(TM^2N^2)$. Therefore, representing the speculation problem W in matrix form requires $O(TM^2N^2)$ steps in total.

Once the matrix representation of the speculation problem is formed, we have a quadratic objective function

$$
\max_{Q} D^T Q - r Q^T \Sigma Q \tag{3.22}
$$

This function can now be efficiently optimized using quadratic linear programming, a gradient method, or a modified simplex method as in [13].

3.3 Theoretical Connections Between Value at Risk and Mean Variance

A point of interest is the theoretical relationship between the Value at Risk and Mean Variance formulations. These relationships are of interest because future work could exploit these properties to create a mapping between Value at Risk and Mean Variance formulations. The Value at Risk formulation has two parameters, namely the threshold value of the objective function (V_{min}) and the limiting probability (X). This formulation is very intuitive, but computationally intractable. On the other hand, the Mean Variance formulation is more computationally tractable (speculation problem), though less intuitive, due to the ambiguity in choosing a reasonable value of r. An interesting open question is the possibility of a mapping from the Value at Risk parameters to the Mean Variance parameter r. Such a mapping would allow an intuitive formulation of the problem in a Value at Risk framework and a computationally tractable optimization using the mapped Mean Variance formulation, which could be solved efficiently using a variant of the speculation problem.

To begin, a review of the two formulations

$$\max_{q_m} E[V] \tag{3.23}$$

$$s.t.\ Pr[V \leq V_{min}] \leq X \tag{3.24}$$

and

$$\max_{q_m} E[V] - r_1 Var[V] \tag{3.25}$$

If we now perform a Lagrangian relaxation on the Value at Risk formulation, we notice

$$\max_{q_m} E[V] - r_2(Pr[V \leq V_{min}] - X) \tag{3.26}$$

which resembles the Mean Variance formulation very closely. Future work investigating this coincidence is likely to be fruitful.

Lastly, we would also like to point out another relationship between Value at Risk and Mean Variance. If the probability distribution that forms the Value at Risk constraint is

considered gaussian, we can also state (where $\overline{V}$ is the average value of V)

$$\Phi(\frac{V_{min} - \overline{V}}{\sqrt{Var[V]}}) = X \iff Var[V] = (\frac{V_{min} - \overline{V}}{\Phi^{-1}(X)})^2 \tag{3.27}$$

Consolidating these ideas would likely lead to very interesting and applicable research.

Chapter 4

Dynamic Optimization Algorithms

The analysis presented thus far concerns a static hedging methodology, where the optimization is calculated at one time instance for a finite horizon of several months. We will now examine a dynamic hedging methodology, where the optimization is calculated repeatedly over a hedging period for a finite horizon of several months. The motivation for this generalization is to take advantage of the value of information. Suppose that a high impact event (e.g. heat wave) occurs during the hedging period. We wish to use this information in subsequent hedges. Dynamic hedging will allow us to benefit from information we may gain during the hedging process.

4.1 Dynamic Hedging

Dynamic hedging is generalized from static hedging in that numerous hedges are performed over a period rather than one hedge at an instant. Also, static hedging results in a deterministic solution to the optimal portfolio, whereas dynamic hedging (in our solution) results in a time dependent optimal portfolio *policy*, where the true optimal portfolio is a function of the current state of the system (to be defined shortly). In the following figure, we clarify what we mean by a hedging period. Hedges can only take place at T discrete instances of the hedging period, while the delivery period is the M discrete instances that the hedges are trying to benefit.

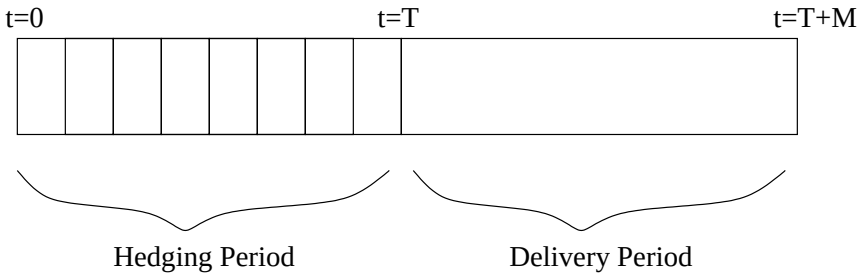

Figure 4-1: Dynamic Hedging and Delivery Periods

4.2 Dynamic Programming

A natural methodology to apply in solving the posed dynamic hedging problem is dynamic programming. Here we present an analysis and complexity bounds on the dynamic hedging problem solved by dynamic programming.

As before, we will use a hedged cash flow model. For the purposes of dynamic hedging, we will define the cash flow model only over the delivery period. In addition, the model will not be identical to the static model due to the extra forward contract variables corresponding to the discrete intervals in the hedging period. For simplicity, we will be assuming the delivery period is one month (with 30 daily delivery intervals). The model for the dynamic hedging cash flow model is

$$C = \sum_{d=T}^{T+M} [l_d(R - s_d) + \sum_{j=0}^{T} q_j(s_d - F_j)] \tag{4.1}$$

where j indexes the forward contract quantity decision variables during the hedging period. Following the general dynamic programming framework in [3], we define the state x_d of our hedging system to be the vector of stochastic variables l_d and b_d

$$x_d = \begin{bmatrix} l_d \\ b_d \end{bmatrix} \tag{4.2}$$

We define our decision u_d to be the policy of forward contracts bought in period d

$$u_d = q_d \tag{4.3}$$

Finally, we define our random disturbances w_d to be the standard normal random variable seeds to our stochastic models

$$w_d = \begin{bmatrix} z_d^l \\ z_d^b \end{bmatrix} \tag{4.4}$$

Given these standard dynamic programming variables, it is straightforward to define a reward *as seen from time d* as an expected value

$$g_d(x_d, q_d, w_d) = E[C|t = d] \tag{4.5}$$

Following the general dynamic programming algorithm in [3], we have

$$J_d(x_d) = \max_{q_d} E[g_d(x_d, q_d, w_d) + J_{d+1}(f(x_d, q_d, w_d))|t = d] \tag{4.6}$$

by letting $x_{d+1} = f(x_d, q_d, w_d)$, where f represents our stochastic models, we arrive at

$$J_d(x_d) = \max_{q_d}(E[C|t = d] + E[J_{d+1}(x_{d+1})|t = d]) \tag{4.7}$$

We now have the basic iteration step of the dynamic programming algorithm. We now only need the base case. Since dynamic programming is a backward running recursion, we need a starting value at time T. We may arrive at this value by applying the static hedging optimization at time T

$$J_T(x_T) = \max_{q_T} g_T(x_T, q_T, w_T) = \max_{q_T} E[C|t = T]] \tag{4.8}$$

Note that we use the static cash flow model for the base case optimization, since we are using the static hedging optimization. Note that $J_0(x_0)$ is the dynamic hedging optimization as seen from time 0 (i.e. the present time). Solving $J_0(x_0)$ is the actual solution to the dynamic hedging optimization, and is what we are working to calculate. We now have the basic dynamic programming algorithm for solving the dynamic hedging optimization.

4.3 Solution by Monte Carlo Simulation

Now that we have the algorithm for dynamic hedging, we must solve it. We propose to calculate the optimal portfolios by applying a Monte Carlo simulation methodology. We must keep in mind the motivation for using dynamic hedging: the value of information.

We proceed by splitting the problem into two hierarchical problems. We first consider the optimal solution for a hedging interval at a given time d, without regard to the interaction with other intervals or the hedging as a whole. We present a simulation approach to its solution and complexity bounds. We then combine these results with analysis of the entire dynamic hedging optimization, resulting in a methodology for the solution of the dynamic hedging optimization and global complexity bounds.

4.3.1 Constructed Isolated-Interval Solution

Define a new problem $f(d)$ as the expected cash flow over the delivery period as seen from time = d *with no other information*. Only this interval matters, and the behavior of all other intervals does not matter. Obviously this problem has no physical significance, but its construction will be very useful shortly

$$f(d) = E[C|t = d] \tag{4.9}$$

We solve this problem by applying Monte Carlo simulation. We first create a four dimensional matrix of cash flow related data, where the first two dimensions are sample ensemble values of the two stochastic models l_d and b_d. Let X be the sufficient number of sample ensembles needed so that our results are statistically significant. We have no theoretical bounds on X, and it poses a very interesting problem. However, for our purposes, we will assume X is sufficiently large enough. The third dimension is time, in which the stochastic processes evolve. This axis will begin at time d (the period we are in, which varies with the constructed problem) and end at the conclusion of the delivery period at time $T + M$. However, though the stochastic processes are simulated from time d, the calculation of the cash flow will only require matrix values for times greater than or equal to T (since the cash flow model is not defined for earlier times). The fourth dimension is a quantified range of the decision variable q_d, which is defined to be of size Q. This is shown graphically in Figure 4-2.

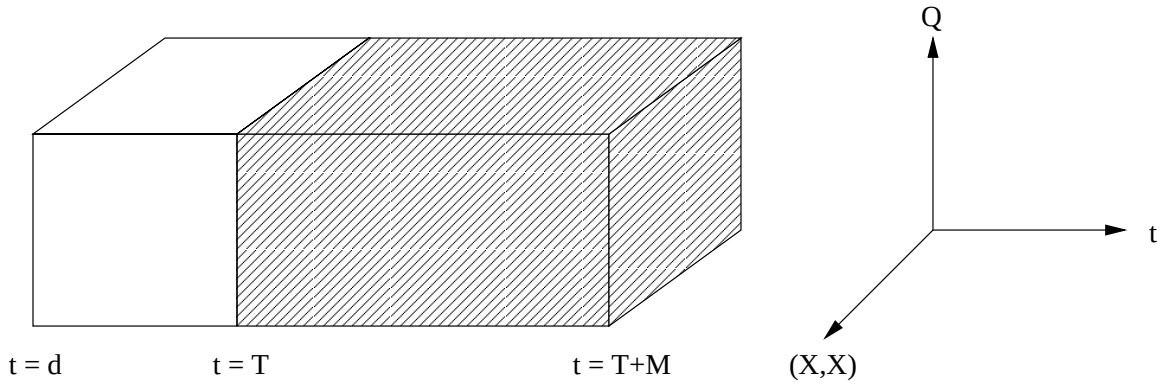

Figure 4-2: Graphical Representation of Data Matrix used to solve the Constructed Isolated-Interval Problem

Once we have this four dimensional matrix of data, we can take three-dimensional expectations over all the axes except the q_d axis (straightforward in MATLAB), resulting in a vector indexed by the forward contract quantity. It is a simple calculation to find the optimal forward contract quantity for this constructed problem

$$q^*_{d_i} = arg \max_{j=1..Q} E_{q_{d_j}}[DataVector] \tag{4.10}$$

4.3.2 Constructed Isolated-Interval Complexity Bounds

In order to solve this problem, we must create the four dimensional matrix of cash flow related data. The subsequent calculations are trivial in comparison. We must simulate X ensembles of both processes from time d to time $T + M$. Once this is completed, *for each* combination of the two ensemble sample values and times greater than or equal to T, we must calculate *for each* value of q_d the corresponding cash flow value, which is of order $O(M)$. Here we assume $q_{d'} = 0$ for $d' < d$ and $q_{d'}$ known for $d' > d$. During the global analysis, these assumptions will be validated. The overall complexity bound for creating the matrix of data, and the problem itself, is $O(MQX(T + M))$.

4.3.3 Global Dynamic Hedging Solution

Now that we have calculated complexity bounds on our constructed problem, we are well equipped to analyze the dynamic hedging algorithm. Using our constructed problem definition, we may rewrite the dynamic hedging algorithm as

$$J_d(x_d) = \max_{q_d}(f(d) + E[J_{d+1}(x_{d+1})|t = d]) \tag{4.11}$$

expanding this equation

$$\begin{aligned} J_d(x_d) \quad = \quad & \max_{q_d}(f(d) + E[\max_{q_{d+1}}(f(d+1) + E[\max_{q_{d+2}}(f(d+2) + \cdots \\ & + f(T)|t = T-1]|t = T-2]| \cdots |t = d]) \end{aligned} \tag{4.12}$$

It is clear that to solve the general dynamic hedging algorithm, at time d we must solve the constructed problem at least T-d times, with each subsequent solution becoming more intensive. For each solution of the $f(d)$ problem from times d to T, we create a matrix as before. However, calculating the cash flow requires knowledge of all values of the forward contract variables *for all* periods. Since dynamic programming is essentially a backward-running recursion algorithm, at time d, $q_{d'}$ will be know for $d' > d$. For $d' < d$, we define $q_{d'} = 0$ since the algorithm assumes nothing about the past (i.e. the algorithm assumes no hedging has taken place in the past). Consequently, we have validated the assumptions made in the previous section concerning forward contract variables that are not in the current period. To make use of the value of information, the motivation to use dynamic hedging, we will seed the processes at time d with the respective process ensemble averages at time d. In general, ensemble averages will differ at different times. This implies that $E[C|t = d_1] \neq E[C|t = d_2]$, which is the value of information that is so important. Applying this methodology, we will be able to solve for optimal forward contract quantity policies, which will result in a solution at least as good as the solution to the static hedging optimization.

4.3.4 Global Dynamic Hedging Complexity Bounds

Consider solving the dynamic hedging optimization in the N^{th} period. In addition to the static hedging optimization (whose complexity bound is "swallowed" by the dynamic hedging complexity bound) we must solve one $f(d)$ problem. Thus we have complexity $O(MQX(T+M))$ for the N^{th} period.

In the $(N-1)^{st}$ period, we must solve another f(d) problem. However, *for each* sample of $f(N-1)$ we must solve $f(N)$ again. Thus for the $(N-1)^{st}$ period, we have $O((MQX(T+M))^2)$ complexity.

Applying induction, it is clear that the complexity of the dynamic hedging at time 0, our original problem, is $O((MQX(T+M))^{N+1})$. Thus, our dynamic hedging optimization is NP-hard. Fortunately, we may apply approximate and neuro dynamic programming techniques to achieve reasonable performance in solving the dynamic hedging optimization.

4.4 Dynamic Hedging Incorporating Risk

Now that we have developed a simple dynamic hedging strategy, we notice that it may return a solution that carries too much risk due to the underlying volatility of the electricity markets. We now consider the dynamic hedging methodology taking into account risk, interpreted as variance of the cash flow. We define a new reward function to be a mean-variance metric

$$g_d(x_d, q_d, w_d) = E[C|t=d] - rVar[C|t=d] \tag{4.13}$$

where r is a measure of risk aversion. We are tempted to use the dynamic programming framework just derived using this new function as the reward function. However, [7] shows that this type of reward function does not satisfy the Bellman equation, rendering dynamic programming useless. The current literature on optimizing a mean-variance reward function suggests this problem is intrinsically difficult, perhaps NP-hard, though this latter conjecture has not been proven. An additional difficulty is that the hedging period and the delivery period (where our reward function is defined) are non-overlapping. The dynamic programming methodology in the previous section can account for this by introducing discount factors. These discount factors can also be used in the mean-variance case.

However, despite using discount factors, the mean-variance optimization problem remains very difficult.

The financial literature does solve this type of mean-variance optimization using only two stages. We conclude this thesis with the simulated solution of three versions of a two stage dynamic hedging problem. We first show results for the case where no risk is taken into account, and then results for different versions of risk (variance) metrics, as in [7].

4.4.1 Simulation Solution with no Penalty on Risk

We now present the simulated solution to the dynamic hedging problem which is solvable by dynamic programming. The reward in each interval of the hedging period is the expected value of cash flow in the delivery period given the decision of quantity of forward contracts bought in the interval. We consider only two hedging periods of one month each, with a one month delivery period. We present a plot of the expected cash flow, over the delivery period, as a function of the decisions in the first and second hedging intervals, q_1 and q_2. We also present the actual solution to this hedging problem: the optimal quantities of forward contracts to buy in each of the two hedging intervals and the optimal expected value of the cash flow over the delivery period. We may compare and contrast these solutions with the subsequent simulations and optimal values. The optimal solution is $(q_1^*, q_2^*) = (10000, 250)$ and the optimal expected cash flow value is $420,800$.

4.4.2 Simulation Solution with a Penalty on Risk

In [7], different metrics are given for quantifying risk. We will present simulated results with two of the metrics. The first metric is the standard variance interpretation of risk – in our optimization, risk is defined to be the variance of the cash flow over the decisions q_1 and q_2 *jointly*

$$Risk \sim Metric1 = Var[\sum_d \alpha^{d-1} R_d] \tag{4.14}$$

The second metric defines the risk to be the *sum* of the variance of the cash flow due to q_1 and the variance of the cash flow due to q_2

$$Risk \sim Metric2 = \sum_d \alpha^{d-1} Var[R_d] \tag{4.15}$$

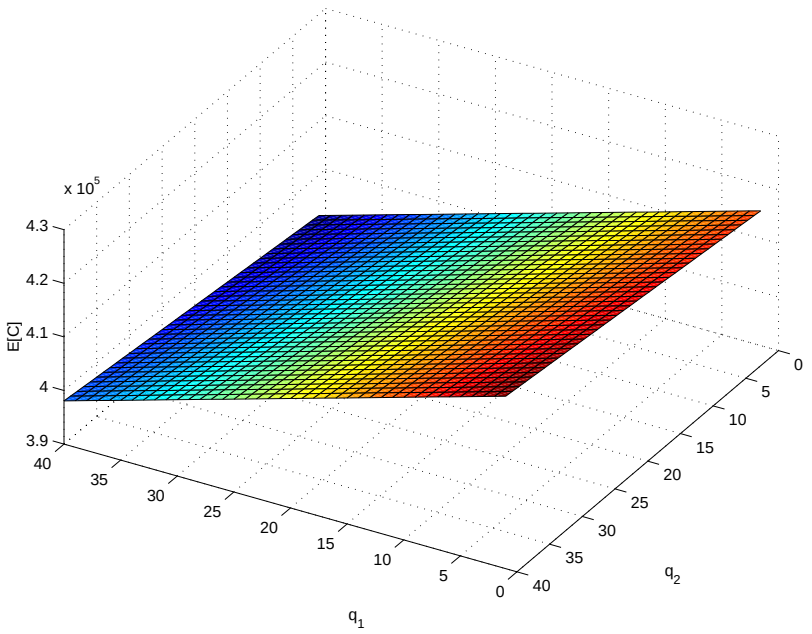

Figure 4-3: Expected Value of the Cash Flow over the Delivery Period as a function of Hedging Period Forward Contract Quantities

where R_d is the reward accumulated during the d^{th} hedging interval.

Both metrics can incorporate a discount factor ($0 < \alpha < 1$) to account for the difference in the times of the hedging intervals. Again, these simulations are plots of the expected cash flow, over the delivery period, as a function of the decisions in the first and second hedging intervals, q_1 and q_2. We also give the optimal solutions and optimal cash flows for various values of the risk-aversion parameter r in the following table.

r	q_E^*	q_{V1}^*	q_{V2}^*	C_E^*	C_{V1}^*	C_{V2}^*
0	$(10^4, 250)$	-	-	$420,800$	-	-
0.05	-	$(10^4, 10^4)$	$(10^4, 10^4)$	-	$401,130$	$401,130$
0.5	-	$(10^4, 10^4)$	$(10^4, 10^4)$	-	$396,050$	$396,050$
5	-	$(10^4, 10^4)$	$(10^4, 10^4)$	-	$398,240$	$398,240$
50	-	$(10^4, 10^4)$	$(250, 10^4)$	-	$395,050$	$399,530$

These results allow us to draw sensible conclusions. Clearly, without a risk constraint ($r = 0$), our optimal cash flow given our range of forward contact variables is at a maximum. Once risk is taken into account ($r > 0$) the optimal cash flow decreases from the risk-free optimal cash flow value. As we weight the risk more heavily (increasing r), the optimal cash flow decreases. The slight discrepancy in the actual results stems from the intensive

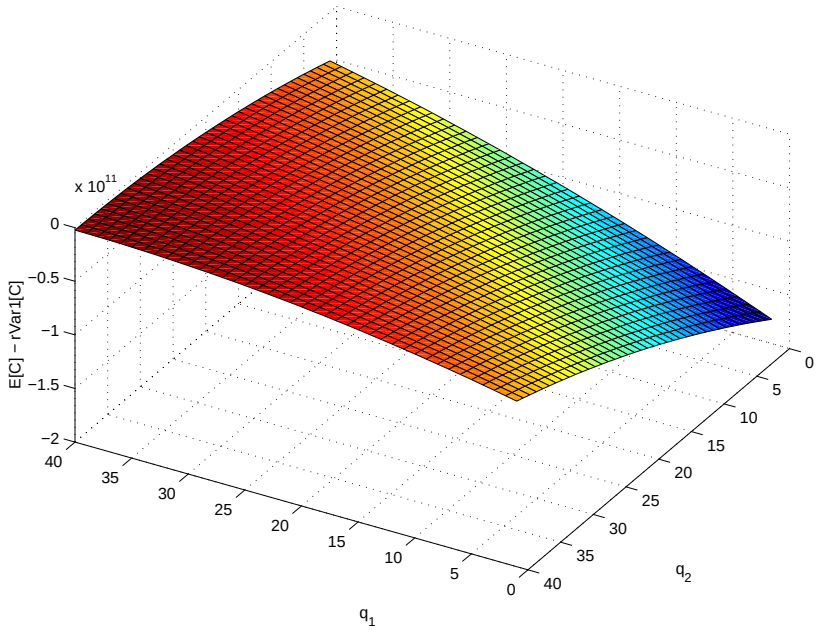

Figure 4-4: (Expected Value - rVariance1) of the Cash Flow over the Delivery Period as a function of Hedging Period Forward Contract Quantities

computational requirements of the solution from simulation. To achieve timely results, we were only able to simulate a small number of stochastic process runs. Thus, our solutions have an inherent error and over multiple runs, there will be discrepancies in the simulation results. However, the error is significantly small such that conclusions may still be drawn with some confidence (the error is approximately two orders of magnitude smaller than the actual cash flow value). It is also interesting to note that the difference in the two variance metrics is not apparent unless a very high value of r is used ($r = 50$). Thus, for most optimization applications, we conclude either variance metric is sufficient, and that the decoupled nature of the second metric may be exploited.

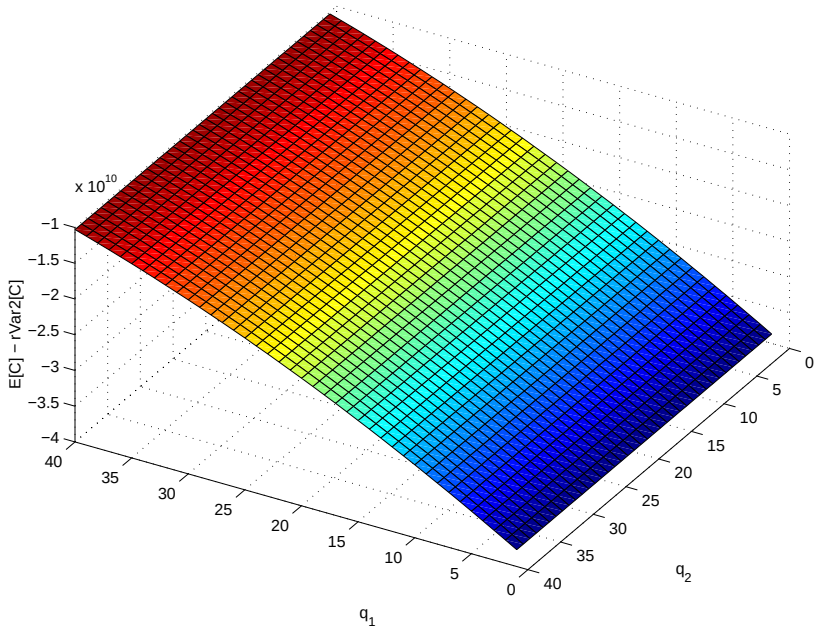

Figure 4-5: (Expected Value - rVariance2) of the Cash Flow over the Delivery Period as a function of Hedging Period Forward Contract Quantities

Chapter 5

Conclusion

Electricity market deregulation has been a popular trend in recent years. However, the resulting market volatilities are significant and this volatility translates into high risk for Load Serving entities. The current situation in California illustrates very clearly what can go wrong with deregulation and the need for hedging. In this thesis we have presented various hedging methodologies that may be used to combat and hedge Load Serving entities' risk in deregulated electricity markets.

In chapter one we reviewed stochastic models of electricity load and market spot price in deregulated electricity markets. These models contain parameters that capture seasonalities that are relevant in electricity markets. We also presented mathematical formulae for load serving entities' cash flow, both unhedged and hedged versions. These value functions, and variants thereof, are subsequently used as objective functions in optimizations.

In chapter two we discussed calibration issues relevant to the stochastic models of electricity load and spot price. We presented three different approaches to calibrating the models, along with discussions of the advantages and disadvantages of each method. We concluded with our recommendation of which approach to implement.

In chapter three we discussed static hedging optimization algorithms. This methodology is concerned with calculating a hedging portfolio once for a finite delivery period. We discussed two variants: Value at Risk and Mean Variance. We solved the Value at Risk static optimization using linear programming theory and simulations. Complexity bounds were also given. We discussed the Mean Variance static optimization from a theoretical point of view, pointed out some directions of research and showed a simulation solution method

for the speculation subproblem. We concluded with theoretical connections between the Value at Risk and Mean Variance frameworks and possible mappings from one to the other.

In chapter four we discussed dynamic hedging optimization algorithms. Dynamic hedging is the methodology of continuously calculating and updating a hedging portfolio over a finite hedging period for a finite delivery period. We presented a simulation-based dynamic programming solution when risk is neglected. Complexity bounds were also given. We next showed a non-dynamic programming simulation-based solution when risk in taken into account. We concluded with results of the dynamic programming simulation and the risk-sensitive simulations, using two different but related metrics for risk.

In conclusion, we are confident in stating that the intricate seasonality nature of electricity markets makes related analyses and optimizations very difficult. The stochastic models attempting to capture the characteristics of the market are complex and provide no closed form solutions to desired quantities. Subsequently, optimization techniques that require these models are in turn plagued by complexity. We are resigned to calculate simulation based solutions to optimization problems, which still require an exponential (in certain parameters) amount of time. Thus, simplifications are unfortunately necessary to calculate a solution to an optimization problem in a timely manner. Succinctly: since the electricity markets are driven by many different parameters, the stochastic models and optimizations must also account for these parameters, and we have a problem with a horrible curse of dimensionality. Techniques to overcome this curse must be applied if one wishes to achieve a meaningful solution to a hedging optimization. Future research must delve into investigating the applicability of these techniques to deregulated electricity markets.

5.1 Future Research on Static Optimization

We now conjecture about possible research directions that might allow us to overcome the curse of dimensionality. Certain optimization problems we have considered do happen to have special structure that may be exploited with further research. Consider the optimization problem (see [2], chapter 6)

$$\min_{q} \sum_{i=1}^{n} f_i(q_i) \tag{5.1}$$

$$s.t. \quad \sum_{i=1}^{n} g_{ij}(q_i) \leq 0, \ j = 1, \ldots, m \tag{5.2}$$

$$q_i \in Q_i, \ i = 1, \ldots, n \tag{5.3}$$

Notice that the objective function is separable in the q variables while the constraint is not. This is exactly isomorphic to our Value at Risk static optimization formulation, where

$$f_i(q_i) = V_i(q_i) \tag{5.4}$$

and

$$\sum_{i=1}^{n} f_i(q_i) = V(q) \tag{5.5}$$

as shown in chapter three. The constraints are

$$\sum_{i=1}^{n} g_{ij}(q_i) = P[V \leq V_{min}] - X \leq 0 \tag{5.6}$$

and

$$q_i \in Q_i, \ i = 1, \ldots, n \tag{5.7}$$

where Q_i is the set of possible values to which we have discretized our q variables. In our simulations, $Q_i = Q_j \ \forall i, j$.

The dual function of the above primal problem is

$$D(\mu) = \sum_{i=1}^{n} D_i(\mu) \tag{5.8}$$

where

$$D_i(\mu) = \inf_{q_i \in Q_i} \{f_i(q_i) + \sum_{j=1}^{m} \mu_j g_{ij}(q_i)\} \tag{5.9}$$

The dual problem is then

$$\max_{\mu \geq 0} D(\mu) \tag{5.10}$$

Notice that we have essentially decoupled the primal constraints in the dual space. Efficient algorithms may in turn be conceivable due to the decoupling. It is possible that the solution of the dual problem is equal to the solution of the original primal problem if there is no duality gap. We would consequently have a solution to the Value at Risk static optimization. Even if there is a duality gap, by the weak duality theorem in [2], the solution to the dual problem will provide a lower bound to the optimal solution of the static optimization.

5.2 Future Research on Dynamic Optimization

Dynamic hedging and the corresponding optimizations have proven more difficult than their static counterparts, not only in the research presented in this thesis, but also in the literature at large. Nonetheless, we propose a research direction that seems promising.

The stochastic models utilized in this thesis are inherently Markovian; clearly from the model equations themselves, we see that the values of L_{d+1} and b_{d+1} are only dependent on the immediately preceding values of L_d and b_d, respectively, and are independent of all other values of $L_{d'}$ and $b_{d'}$, $d' < d$. We propose constructing a semi-Markov process where the state transition times are deterministically equal to one, corresponding to the discrete nature of our models. The underlying Markov chain can be constructed using tree-building techniques in [9] using either simulations of our models or historical data. Since our models are mean-reverting, the constructed tree will approach a steady-state, which will be interpreted as the underlying Markov chain of the semi-Markov process. The technique in [9] also provides heuristics for calculating transitional probabilities of the Markov chain. Once we have a semi-Markov process, we may apply Markovian decision analysis techniques. We recommend [1] and [7].

Another related research direction to investigate is the application of periodically-varying Markov processes. These processes have transition probability matrices that vary periodically with successive transitions (i.e. $\exists N$ s.t. $P^N = P$, where N is the period; see [8]). Periodically-varying Markov processes are very powerful modeling tools and could very well be applied to deregulated electricity markets. Modeling seasonal *or* daily trends would correspond to having relevant transition probability matrices for each trend, respectively. However, since there are many different market characteristics evolving on different time

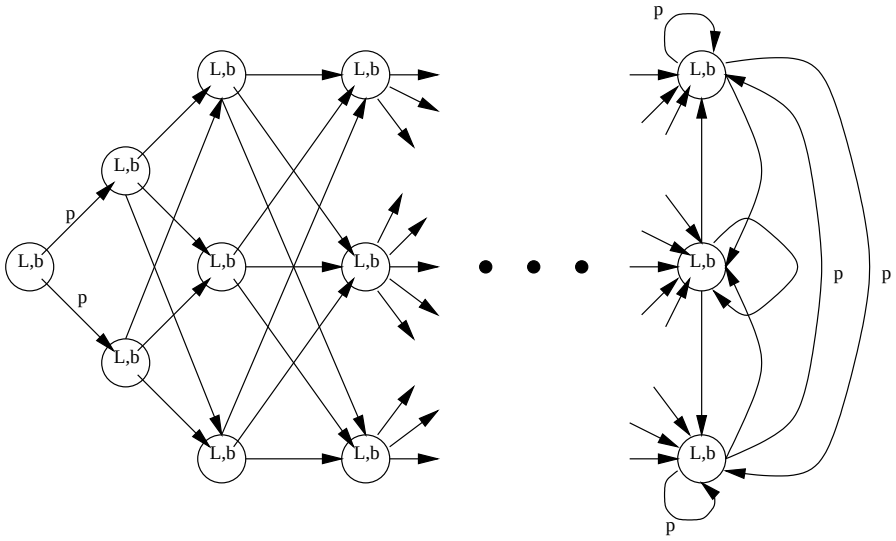

Figure 5-1: Tree Construction Results in a Steady State Markov Chain with (L,b) Pairs as States and p's as Transition Probabilities

scales (e.g. daily, weekly, and seasonal trends), the question remains how to unify the separate trend Markov processes into a single cohesive model. Questions also arise whether we could construct these new chains from our current model, or whether we need to build them independently, essentially creating new models of the deregulated electricity market dynamics.

Appendix A

Source Codes

Maximum Likelihood Code

```
% Parameter Estimation using Kalman Filter and Maximum Likelihood Estimation
% MLE Module
% This code is modified from code written by Andrej Gubina

function [mle, X, P, Y] = MLE(theta)

simplifiedModel;

y = Load;

% Initialization of unknown parameters vector theta
alpha           = theta(1);
sigma           = theta(2);

% Kalman Loop

%Initial values
A = 1 - alpha;
C = 1;
Q = sigma^2;

% Log Likelihood Function
J = 0;

% Initial estimate
x_t_t = y(1);
P_t_t = 1;

% information vectors to be returned
```

```
X = [];
P = [];

for t = 1:size(y,1)

  % Kalman Main loop
  x_tt_t    = A*x_t_t;
  P_tt_t    = A*P_t_t*transpose(A) + Q;
  N_tt_t    = P_tt_t;
  K         = P_tt_t*inv(N_tt_t);
  yhat_tt_t = y(t) - x_tt_t;
  x_tt_tt   = x_tt_t + K*yhat_tt_t;
  P_tt_tt   = [1 - K]*P_tt_t;

  X = [X; x_tt_tt];
  P = [P; P_tt_tt];

  % Criterion function J (Log Likelihood function)
  J     = J + transpose(yhat_tt_t)*inv(N_tt_t)*yhat_tt_t + log(abs(N_tt_t));
end;

mle = .5*J;
Y = y;
```

Maximum Likelihood Code with Measurement Noise

```
% Parameter Estimation using Kalman Filter and Maximum Likelihood Estimation
% MLE Module
% This code is modified from code written by Andrej Gubina
% INCLUSION OF MEASURE NOISE

function mle = MLE2(theta)

simplifiedModel;

y = Load;                                                          10

% Initialization of unknown parameters vector theta
alpha           = theta(1);
sigma           = theta(2);

% Kalman Loop

%Initial values
A = 1 - alpha;
Q = sigma^2;                                                       20
H = .01;

% Log Likelihood Function
J = 0;

% Initial estimate
x_t_t = y(1);
P_t_t = .1;

% information vectors to be returned                               30

X = [];
P = [];

T = size(y,1);

for t = 1:T

  % Kalman Main loop
  x_tt_t    = A*x_t_t;                                             40
  P_tt_t    = A*P_t_t*transpose(A) + Q;

  F_tt_t    = P_tt_t + H;
  K         = A*P_tt_t*inv(F_tt_t);

  x_tt_tt   = (A - K)*x_tt_t + K*y(t);
```

```
  P_tt_tt    = A*(P_tt_t - P_tt_t*inv(F_tt_t)*P_tt_t)*transpose(A) + Q;

  X = [X; x_tt_tt];
  P = [P; P_tt_tt];

  % Criterion function J (Log Likelihood function)
    J  = J - (1/2)*log(2*pi) - (1/2)*log(norm(F_tt_t)) - ...
         (1/2)*transpose(y(t) - x_tt_tt)*inv(F_tt_t)*(y(t) - x_tt_tt);
end;

Y = y;
mle = J;
```

Kalman Filtering Script

```
% Parameter Estimation using Kalman Filter and Maximum Likelihood Estimation
% Kalman Filter Module
% This code is modified from code written by Andrej Gubina
% Used matlab black-box function "fminunc"

old = 1000;
new = 1001;
theta = [1;1];
conv = [];
A = [];
S = [];
J = 0;
X = [];
P = [];

for i = 1:100

conv = [conv; old - new];

  old = new;

  % Run Kalman Filter
  [J,X,P,Y] = MLE(theta);
  new = J;

  sup = size(X,1);
  a = theta(1); % alpha
  s = theta(2); % sigma

  % initialize all derivatives to zero for each iteration
  d_a = 0;
  d_s = 0;
  d_a_a = 0;
  d_s_s = 0;
  d_a_s = 0;

  % Derivatives of J

  for i = 1:sup

    d_a = d_a + ([(a^2 - 2*a + 1)*P(i) + s^2]*[a*X(i) + Y(i) - X(i)]*[2*X(i)] - ...
      [a*X(i) + Y(i) - X(i)]^2*[2*a*P(i) - 2*P(i)])/[(a^2 - 2*a + 1)*P(i) + ...
      s^2]^2 + [2*a*P(i) - 2*P(i)]/[(a^2 - 2*a + 1)*P(i) + s^2];

    d_s = d_s - [a*X(i) + Y(i) - X(i)]^2*[2*s]/[(a^2 - 2*a + 1)*P(i) + s^2]^2 + ...
      [2*s]/[(a^2 - 2*a + 1)*P(i) + s^2];
```

```
  d_s_s = d_s_s + (-2*[(a^2 - 2*a + 1)*P(i) + s^2]^2*[a*X(i) + Y(i) - X(i)] + ...
    [2*s]*[a*X(i) + Y(i) - X(i)]^2*[2]*[(a^2 - 2*a + 1)*P(i) + ...
    s^2]*[2*s])/[(a^2 - 2*a + 1)*P(i) + s^2]^4 + (2*[(a^2 - 2*a + 1)*P(i) + ...
    s^2] - 4*s^2)/[(a^2 - 2*a + 1)*P(i) + s^2]^2;

  d_a_s = d_a_s + ([(a^2 - 2*a + 1)*P(i) + s^2]^2*[2*s]*[a*X(i) + Y(i) - ...
    X(i)]*[2*X(i)] - 4*s*[(a^2 - 2*a + 1)*P(i) + s^2]*([(a^2 - 2*a + ...
    1)*P(i) + s^2]*[a*X(i) + Y(i) - X(i)]*[2*X(i)] - [a*X(i) + Y(i) - ...
    X(i)]^2*[2*a*P(i) - 2*P(i)]))/[(a^2 - 2*a + 1)*P(i) + s^2]^4 - ...
    [2*a*P(i) - 2*P(i)]*[2*s]/[(a^2 - 2*a + 1)*P(i) + s^2]^2;

  d_a_a = d_a_a + ([(a^2 - 2*a + 1)*P(i) + s^2]^2*([(a^2 - 2*a + 1)*P(i) + ...
    s^2]*[2*X(i)*X(i)] + [a*X(i) + Y(i) - X(i)]*[2*X(i)]*[2*a*P(i) - ...
    2*P(i)] - [a*X(i) + Y(i) - X(i)]^2*[2*P(i)] - [2*a*P(i) - ...
    2*P(i)]*[2]*[a*X(i) + Y(i) - X(i)]*[X(i)]) - ([(a^2 - 2*a + 1)*P(i) + ...
    s^2]*[a*X(i) + Y(i) - X(i)]*[2*X(i)] - [a*X(i) + Y(i) - ...
    X(i)]^2*[2*a*P(i) - 2*P(i)])*[2]*[(a^2 - 2*a + 1)*P(i) + s^2]*[2*a*P(i) - ...
    2*P(i)])/[(a^2 - 2*a + 1)*P(i) + s^2]^4 + ([(a^2 - 2*a + 1)*P(i) + ...
    s^2]*[2*P(i)] - [2*a*P(i) - 2*P(i)]^2)/[(a^2 - 2*a + 1)*P(i) + s^2]^2;
end;

d_a = d_a/2;
d_s = d_s/2;
d_a_a = d_a_a/2;
d_s_s = d_s_s/2;
d_a_s = d_a_s/2;

% Newtons Method, applied analytically

gradient = [d_a; d_s];
hessian = [d_a_a d_a_s; d_a_s d_s_s];

% Steepest Descent, with line-search

au = 10;
al = 0;
rho = (al + au)/2;

d1 = d_a;
d2 = d_s;

a = a + rho*d1;
s = s + rho*d2;

for i = 1:sup

  d_a = d_a + ([(a^2 - 2*a + 1)*P(i) + s^2]*[a*X(i) + Y(i) - X(i)]*[2*X(i)] - ...
```

```
      [a*X(i) + Y(i) - X(i)]^2*[2*a*P(i) - 2*P(i)])/[(a^2 - 2*a + 1)*P(i) + ...
      s^2]^2 + [2*a*P(i) - 2*P(i)]/[(a^2 - 2*a + 1)*P(i) + s^2];

    d_a = d_a/2;

    d_s = d_s - [a*X(i) + Y(i) - X(i)]^2*[2*s]/[(a^2 - 2*a + 1)*P(i) + s^2]^2 + ...
      [2*s]/[(a^2 - 2*a + 1)*P(i) + s^2];

    d_s = d_s/2;
  end;

  h_prime = transpose([d_a; d_s])*[d1;d2];

for j = 1:20

    if h_prime < 0
      al = rho;
    else
      au = rho;
    end;

    rho = (al + au)/2;

    a = a + rho*d1;
    s = s + rho*d2;

  for i = 1:sup

    d_a = d_a + ([(a^2 - 2*a + 1)*P(i) + s^2]*[a*X(i) + Y(i) - X(i)]*[2*X(i)] - ...
      [a*X(i) + Y(i) - X(i)]^2*[2*a*P(i) - 2*P(i)])/[(a^2 - 2*a + 1)*P(i) + ...
      s^2]^2 + [2*a*P(i) - 2*P(i)]/[(a^2 - 2*a + 1)*P(i) + s^2];

    d_a = d_a/2;

    d_s = d_s - [a*X(i) + Y(i) - X(i)]^2*[2*s]/[(a^2 - 2*a + 1)*P(i) + s^2]^2 + ...
      [2*s]/[(a^2 - 2*a + 1)*P(i) + s^2];

    d_s = d_s/2;
  end;

  h_prime = transpose([d_a; d_s])*[d1;d2];

  end;

  theta = theta - rho*gradient;

  A = [A; theta(1)];
  S = [S; theta(2)];
```

```
end;

subplot(2,2,1);
plot(conv);
title('Difference between updated versions of J function');

subplot(2,2,2);                                                    150
plot(A);
title('alpha');

subplot(2,2,3);
plot(S);
title('sigma');

alpha = theta(1)
sigma = theta(2)
                                                                   160
trueAlpha = .25
trueSigma = .75
```

Stochastic Model Parameters Script

```
MonthSigma_mLSigma_mb = [1,         1240.9526,    0.043193865;  ...
                         2,         1165.019,     0.034697878;  ...
                         3,         1249.7662,    0.037580555;  ...
                         4,         1449.6568,    0.045197089;  ...
                         5,         1578.9004,    0.11314249;   ...
                         6,         1944.4605,    0.086317638;  ...
                         7,         2188.7946,    0.090956508;  ...
                         8,         2078.4759,    0.10340528;   ...
                         9,         1904.5981,    0.082606587;  ...
                         10,        1458.5315,    0.064308124;  ...
                         11,        1519.1309,    0.076639634;  ...
                         12,        1450.7796,    0.071693513];

aalphabetaLambda_LLambda_bSigma_DLSigma_Db = [4.559968e-05, 0.6310146, ...
0.47896373, 0.18003582, 6.40329e-06, 171.66883, 0.00048277646];


mu_m = [22789; 22267; 22303; 22436; 23160; 25349; 28218; 29160; 26780; ...
        24117; 23335; 23292];

mu_b = [0.3903; 0.3914; 0.3906; 0.3911; 0.3874; 0.3835; 0.3842; 0.3859; ...
        0.3882; 0.3934; 0.3914; 0.3907];
```

Stochastic Model Monthly Shape Calculation Script

```
% Script to compute monthly shapes of data
% Load data format: [year, month, day, weekday, load, month_avg_load,
% delta_load] years 1994-2000
% Price data format: [year, month, day, weekday, b, month_avg_b, delta_b]
% years 1998-2000

% Read in raw data
loadFID = fopen('calpx_load_7_wk16_ymd_avg.txt','r');
supplyFID = fopen('b_calpx_3a.txt','r');

loadDataRaw = fscanf(loadFID, '%f');
supplyDataRaw = fscanf(supplyFID, '%f');

fclose(loadFID);
fclose(supplyFID);

% create ordered matrix out of raw data

loadDim = size(loadData);
supplyDim = size(supplyData);

loadMatrix = zeros(loadDim(1)/7,7);
supplyMatrix = zeros(supplyDim(1)/7,7);

for i=1:loadDim(1)
  if mod(i,7) == 0
    loadMatrix(floor(i/7)+1,7) = loadDataRaw(i);
  else
    loadMatrix(floor(i/7)+1,mod(i,7)) = loadDataRaw(i);
  end;
end;

for i=1:supplyDim(1)
  if mod(i,7) == 0
    supplyMatrix(floor(i/7)+1,7) = supplyDataRaw(i);
  else
    supplyMatrix(floor(i/7)+1,mod(i,7)) = supplyDataRaw(i);
  end;
end;

% Calculate mu_m: monthly shape of load -> average for a given month over all years

mu_m = zeros(12,1);
count_m = zeros(12,1);

for i=1:loadDim(1)/7
```

```
  for j=1:12
    if loadMatrix(i,2) == j
      mu_m(j) = mu_m(j) + loadMatrix(i,6);
      count_m(j) = count_m(j) + 1;
    end;
  end;
end;

mu_b = zeros(12,1);
count_b = zeros(12,1);

for i=1:supplyDim(1)/7
  for j=1:12
    if supplyMatrix(i,2) == j
      mu_b(j) = mu_b(j) + supplyMatrix(i,6);
      count_b(j) = count_b(j) + 1;
    end;
  end;
end;

for k=1:12
  mu_m(k) = mu_m(k)/count_m(k);
  mu_b(k) = mu_b(k)/count_b(k);
end;
```

Stochastic Model Script

```
% Stochastic model for generating load and supply data

parameters;

a = aalphabetaLambda_LLambda_bSigma_DLSigma_Db(1);
alpha = aalphabetaLambda_LLambda_bSigma_DLSigma_Db(2);
beta = aalphabetaLambda_LLambda_bSigma_DLSigma_Db(3);
lambda_L = aalphabetaLambda_LLambda_bSigma_DLSigma_Db(4);
lambda_b = aalphabetaLambda_LLambda_bSigma_DLSigma_Db(5);
sigma_dL = aalphabetaLambda_LLambda_bSigma_DLSigma_Db(6);
sigma_db = aalphabetaLambda_LLambda_bSigma_DLSigma_Db(7);


% length of generated data, in months
% need to change in searchSpace.m and variance.m as well
N = 2;

% Load data
delta_L_y = zeros(N,1);
delta_L_y(1) = lambda_L;

% m is for months
for m=2:N
  delta_L_y(m) = delta_L_y(m-1) + lambda_L + sigma_dL*normrnd(0,1);
end;

monthlyLoadAvg = zeros(N,1);

for m=1:N
  if mod(m,12) == 0
    monthlyLoadAvg(m) = mu_m(12) + delta_L_y(m);
  else
    monthlyLoadAvg(m) = mu_m(mod(m,12)) + delta_L_y(m);
  end;
end;

% Daily load, assuming 30 days per month

dailyLoad = zeros(30*N,1);
dailyLoad(1) = monthlyLoadAvg(1);

for d=2:30*N
  dailyLoad(d) = dailyLoad(d-1) + alpha*(monthlyLoadAvg(floor((d-1)/30)+1) - ...
    dailyLoad(d-1)) + MonthSigma_mLSigma_mb(floor((d-1)/30)+1,2)*normrnd(0,1);
end;
```

```
% Supply (price) data

delta_b_y = zeros(N,1);
delta_b_y(1) = lambda_b;

% m is for months
for m=2:N
  delta_b_y(m) = delta_b_y(m-1) + lambda_b + sigma_db*normrnd(0,1);
end;

monthlySupplyAvg = zeros(N,1);

for m=1:N
  if mod(m,12) == 0
    monthlySupplyAvg(m) = mu_b(12) + delta_b_y(m);
  else
    monthlySupplyAvg(m) = mu_b(mod(m,12)) + delta_b_y(m);
  end;
end;

% Daily load, assuming 30 days per month

dailySupply = zeros(30*N,1);
dailySupply(1) = monthlySupplyAvg(1);

for d=2:30*N
  dailySupply(d) = dailySupply(d-1) + beta*(monthlySupplyAvg(floor((d-1)/30) + ...
    1) - dailySupply(d-1)) + MonthSigma_mLSigma_mb(floor((d-1)/30) + ...
    1,3)*normrnd(0,1);
end;

load = dailyLoad;

spot = zeros(30*N,1);

for i=1:30*N
  spot(i)= exp(a*load(i) + dailySupply(i));
end;
```

One Month Search Space Script

```
% Attempt to view search space for V(q_1)
% Runtime: O(TN) where T is number of simulations
% and N is size of q_1 vector

tic;

% months; need to change in model.m as well
N = 1;

% Generate T new load and spot data to calculate probabilities using
% stochastic model

T = 500;

% Generate T sets of new load and spot data
load_generated = [];
spot_generated = [];

for i = 1:T
  model;
  load_generated = [load_generated, load];
  spot_generated = [spot_generated, spot];
end;

average_spot1 = mean(mean(spot_generated));

size = 15000;
div = 100;
q_1 = [0:div:size];
q_1 = q_1 - 5000*ones(1,size/div+1);
R = 1.15*average_spot1;
F_1 = 1.1*average_spot1;

% T scenarios of V(q_1): L is constant, l is stochastic
V_L = zeros(size/div+1,T);
V_l = zeros(size/div+1,T);

% Load constant
L = 2000;

for j = 1:T
  for i = 1:size/div+1
    for k = 1:30
      V_L(i,j) = V_L(i,j) + L*(R - spot_generated(k,j)) + ...
                    q_1(i)*(spot_generated(k,j) - F_1);
    end;
```

```
  end;
end;

for j = 1:T
  for i = 1:size/div+1
    for k = 1:30
      V_l(i,j) = V_l(i,j) + .1*load_generated(k,j)*(R - spot_generated(k,j)) + ...
                    q_1(i)*(spot_generated(k,j) - F_1);
    end;
  end;
end;

V_var_L = var(transpose(V_L));
V_var_l = var(transpose(V_l));

toc;

figure(1);
plot(q_1,V_var_L);
xlabel('q_1');
ylabel('Variance of V');
title('Variance of V as function of q_1, load constant, F_1 = E[spot]');

figure(2);
plot(q_1,V_var_l);
xlabel('q_1');
ylabel('Variance of V');
title('Variance of V as function of q_1, load stochastic, F_1 < E[spot]');

M_L = zeros(size/div+1,1);
M_l = zeros(size/div+1,11);

V_mu_L = mean(mean(V_L));
V_mu_l = mean(mean(V_l));

% Critical Values
V_min_L = .8*V_mu_L;
V_min_l = .8*V_mu_l;
X = .05;

% Count the number of scenarios where V_dynamic < V_min
for k=1:T
  for i=1:size/div+1
    if V_L(i,k) < V_min_L
      M_L(i) = M_L(i) + 1;
    end;
    if V_l(i,k) < V_min_l
      M_l(i) = M_l(i) + 1;
```

```
    end;
  end;
end;

% Create probabilities for (q_1, q_2) pairs
M_L = M_L./T;
M_l = M_l./T;

figure(3);
plot(q_1,M_L);
xlabel('q_1');
ylabel('Prob[V <= V-min');
title('Prob[V <= V_min] as a funtion of q_1, load constant, F_1 = E[spot]');

figure(4);
plot(q_1,M_l);
xlabel('q_1');
ylabel('Prob[V <= V-min');
title('Prob[V <= V_min] as a funtion of q_1, load stochastic, F_1 < E[spot]');

% Show maximum value of objective function as function of r

R = [];
r = [];

for i = 1:45
  r = [r; 1.2^(-i)];
end;

% Find expected value of V over all simulations as function of q variables

E_V_l = mean(transpose(V_l));

for i = 1:45
  R = [R; max(E_V_l - r(i).*V_var_l)];
end;

figure(5);
semilogx(r,R);
xlabel('r');
ylabel('E[V] - rVar[V]');
```

Two Month Search Space Script

```
% Attempt to view search space for V(q_1,q_2)
% Runtime: O(TN^2) where T is number of simulations
% and N is size of (q_1,q_2) grid

tic;

% months; need to change in model.m as well
N = 2;

% Attempt to view searchSpace

% Generate T new load and spot data to calculate
% probabilities using stochastic model

T =300;

% Generate T sets of new load and spot data
load_generated = [];
spot_generated = [];

for i = 1:T
  model;
  load_generated = [load_generated, load];
  spot_generated = [spot_generated, spot];
end;

disp('T simulations generated');
toc;

average_spot1 = mean(mean(spot_generated(1:15*N,:)));
average_spot2 = mean(mean(spot_generated(15*N+1:30*N,:)));
average_spot = mean([average_spot1; average_spot2]);

size = 15000;
div = 200;
[q_1, q_2] =meshgrid((-1*size/3):div:(2*size)/3);
R = 1.15*average_spot;
F_1 = average_spot1;
F_2 = average_spot2;

% T scenarios of V(q_1,q_2) with load L constant or load l stochastic
V_L = zeros(size/div+1,size/div+1,T);
V_l = zeros(size/div+1,size/div+1,T);

% Constant value for load
L = 2000;
```

```
for k = 1:T
  for i = 1:size/div+1
    for j = 1:size/div+1                                                    50
      for n1 = 1:15*N
        V_L(i,j,k) = V_L(i,j,k) + L*(R - spot_generated(n1,k)) + ...
                     q_1(i,j)*(spot_generated(n1,k) - F_1);
      end;
      for n2 = (15*N+1):30*N
        V_L(i,j,k) = V_L(i,j,k) + L*(R - spot_generated(n2,k)) + ...
                     q_2(i,j)*(spot_generated(n2,k) - F_2);
      end;
    end;
  end;                                                                      60
  disp(k);
  disp(' th iteration out of ');
  disp(T);
  disp(' for computation of V_L');
end;

disp('V_L computed');
toc;

for k = 1:T                                                                 70
  for i = 1:size/div+1
    for j = 1:size/div+1
      for n1 = 1:15*N
        V_l(i,j,k) = V_l(i,j,k) + .1*load_generated(n1,k)*(R - ...
          spot_generated(n1,k)) + q_1(i,j)*(spot_generated(n1,k) - F_1);
      end;
      for n2 = (15*N+1):30*N
        V_l(i,j,k) = V_l(i,j,k) + .1*load_generated(n2,k)*(R - ...
          spot_generated(n2,k)) + q_2(i,j)*(spot_generated(n2,k) - F_2);
      end;                                                                  80
    end;
  end;
  disp(k);
  disp(' th iteration out of ');
  disp(T);
  disp(' for V_l');
end;

disp('V_l computed');
toc;                                                                        90

% Create color constraints revealing feasible search space
C_L = zeros(size/div+1,size/div+1);
C_l = zeros(size/div+1,size/div+1);
```

```
% Create probability matrices
M_L = zeros(size/div+1,size/div+1);
M_l = zeros(size/div+1,size/div+1);

% statistical variable to compute distribution of V_L and V_l
V_stat_L = zeros(T*(size/div+1)^2,1);
V_stat_l = zeros(T*(size/div+1)^2,1);
cnt_L = 0;
cnt_l = 0;

for k=1:T
  for i=1:size/div+1
    for j=1:size/div+1
      cnt_L = cnt_L + 1;
      cnt_l = cnt_l + 1;
      V_stat_L(cnt_L) = V_L(i,j,k);
      V_stat_l(cnt_l) = V_l(i,j,k);
    end;
  end;
end;

disp('V_stat_L and V_stat_l computed');
toc;

V_mu_L = mean(V_stat_L);
V_mu_l = mean(V_stat_l);

% Critical Values
V_min_L = .8*V_mu_L;
V_min_l = .8*V_mu_l;
X = .05;

% Count the number of scenarios where V < V_min
for k=1:T
  for i=1:size/div+1
    for j=1:size/div+1
      if V_L(i,j,k) < V_min_L
        M_L(i,j) = M_L(i,j) + 1;
      end;
      if V_l(i,j,k) < V_min_l
        M_l(i,j) = M_l(i,j) + 1;
      end;
    end;
  end;
end;

% Create probabilities for (q_1, q_2) pairs
```

```
M_L = M_L./T;
M_l = M_l./T;

disp('M_L and M_l computed');
toc;

for i=1:size/div+1
  for j=1:size/div+1                                                    150
    if M_L(i,j) < X
      C_L(i,j) = 1;
    else
      C_L(i,j) = 2;
    end;
    if M_l(i,j) < X
      C_l(i,j) = 1;
    else
      C_l(i,j) = 2;
    end;                                                                160
  end;
end;

disp('C_L and C_l computed');
toc;

% average along third (k) dimension
V_average_L = mean(V_L,3);
V_average_l = mean(V_l,3);
                                                                        170
disp('Calculations complete; plots following');
toc;

figure(1);
surf(q_1,q_2,M_L);
xlabel('q_1');
ylabel('q_2');
zlabel('Prob[V <= V_min]');
title('Prob[V <= V_min], load constant, F_1 < E[spot] < F_2');
                                                                        180
figure(2);
surf(q_1,q_2,M_l);
xlabel('q_1');
ylabel('q_2');
zlabel('Prob[V <= V_min]');
title('Prob[V <= V_min], load stochastic, F_1 < E[spot < F_2');

figure(3);
surf(ones(size/div+1),C_L);
xlabel('q_1');                                                          190
```

```
ylabel('q_2');
title('Feasible Space of E[V], constant load, blue is feasible, ...
      F_1 < E[spot] < F_2');

figure(4);
surf(q_1,q_2,zeros(size/div+1),C_l);
xlabel('q_1');
ylabel('q_2');
title('Feasible Space of E[V], stochastic load, blue is feasible, ...
      F_1 < E[spot] < F_2');                                          200

V_var_L = zeros(size/div+1,size/div+1);
V_var_l = zeros(size/div+1,size/div+1);

for i=1:size/div+1
  for j=1:size/div+1
    V_var_L(i,j) = var(V_L(i,j,:));
    V_var_l(i,j) = var(V_l(i,j,:));
  end;
end;                                                              210

disp('variances calculated');

figure(7);
surf(q_1,q_2,V_var_L);
xlabel('q_1');
ylabel('q_2');
zlabel('var(V_L)')
title('Variance of V, load constant, F_1 < E[spot] < F_2');
                                                                  220
figure(8);
surf(q_1,q_2,V_var_l);
xlabel('q_1');
ylabel('q_2');
zlabel('var(V_l)')
title('Variance of V, load stochastic, F_1 < E[spot] < F_2');

% Show maximum value of objective function as function of r

R = [];                                                           230
r = [];

for i = 1:45
  r = [r; 1.2^(-i)];
end;

% Find expected value of V over all simulations as function of q variables
```

```
E_V_l = mean(V_l,3);

for i = 1:45
  R = [R; max(max(E_V_l - r(i).*V_var_l))];
end;

figure(5);
semilogx(r,R);
xlabel('r');
ylabel('E[V] - rVar[V]');
```

Simplified Model Script (Calibration)

```
% Simplified stochastic model for generating load data

alpha = .25;
sigma = .75;

% length of generated data, in months
% need to change in searchSpace.m and variance.m as well
N = 2;

% Daily load, assuming 30 days per month

dailyLoad = zeros(30*N,1);
dailyLoad(1) = 0;

for d=2:30*N
  dailyLoad(d) = (1 - alpha)*dailyLoad(d-1) + sigma*normrnd(0,1);
end;

Load = dailyLoad;
```

Dynamic Hedging Script

```
% Simulations for Dynamic Hedging, two stages, three simulations:
% 1 - No variance
% 2 - variance #1
% 3 - variance #2

% months; need to change in model.m as well
% month 1, 2 = 2-stage hedging period; month 3 = delivery period
N = 3;

% Generate T new load and spot data using stochastic model

T = 100;

% Generate T sets of new load and spot data
load_generated = [];
spot_generated = [];

for i = 1:T
  model;
  load_generated = [load_generated, load];
  spot_generated = [spot_generated, spot];
end;

disp('T simulations generated');


average_spot1 = mean(mean(spot_generated(1:30,:)));
average_spot2 = mean(mean(spot_generated(31:60,:)));
average_spot3 = mean(mean(spot_generated(61:90,:)));
average_spot = mean([average_spot1; average_spot2; average_spot3]);

size = 10000;
div = 250;
[q_1, q_2] =meshgrid(0:div:size);
R = 1.15*average_spot;
F_1 = average_spot1;
F_2 = average_spot2;
F_3 = average_spot3;

% T scenarios of C(q_1,q_2)

C = zeros(size/div,size/div,T);
firstStage = zeros(size/div,size/div,T);
secondStage = zeros(size/div,size/div,T);

for k = 1:T
```

```
  for i = 1:size/div
    for j = 1:size/div
      for n = 61:90
        C(i,j,k) = C(i,j,k) + load_generated(n,k)*(R - spot_generated(n,k)) + ...
                   (q_1(i,j) + q_2(i,j))*spot_generated(n,k) - (q_1(i,j)*F_1 + ...
                   q_2(i,j)*F_2);
        firstStage(i,j,k) = firstStage(i,j,k) + load_generated(n,k)*(R - ...
                            spot_generated(n,k)) + q_1(i,j)*spot_generated(n,k) - ...
                            q_1(i,j)*F_1;
        secondStage(i,j,k) = secondStage(i,j,k) + load_generated(n,k)*(R - ...
                             spot_generated(n,k)) + q_2(i,j)*spot_generated(n,k) - ...
                             q_2(i,j)*F_2;
      end;
    end;
  end;
  disp(k);
  disp(' th iteration out of ');
  disp(T);
  disp(' for C');
end;

disp('C computed');

% Find the expected values and arguments (indices that give the q_1 and q_2 values)

EV = mean(C,3);
[y,i] = max(EV);
[z,j] = max(y);

Q1 = i(j)*div;
Q2 = j*div;
optVal_E = z;

figure(1);
surf(EV);
view([-150,50]);
xlabel('q_1');
ylabel('q_2');
zlabel('E[C]');

% First Variance metric

VAR1 = zeros(size/div,size/div);

for i = 1:size/div
  for j = 1:size/div
    VAR1(i,j) = var(C(i,j,:));
  end;
```

```
end;

r = 5;

V1 = EV - r*VAR1;
                                                                    100
[y,i] = max(V1);
[z,j] = max(y);

Q1_V1 = i(j)*div;
Q2_V1 = j*div;
optVal_V1 = EV(i(j),j);

figure(2);
surf(V1);
view([-150,50]);                                                    110
xlabel('q_1');
ylabel('q_2');
zlabel('E[C] - rVar1[C]');

% Second Variance Metric

sub1VAR2 = zeros(size/div,size/div);

for i = 1:size/div
  for j = 1:size/div                                                120
    sub1VAR2(i,j) = var(firstStage(i,j,:));
  end;
end;

sub2VAR2 = zeros(size/div,size/div);

for i = 1:size/div
  for j = 1:size/div
    sub2VAR2(i,j) = var(firstStage(i,j,:));
  end;                                                              130
end;

alpha = .8;

VAR2 = alpha*sub1VAR2 + sub2VAR2;

V2 = EV - r*VAR2;

[y,i] = max(V2);
[z,j] = max(y);                                                     140

Q1_V2 = i(j)*div;
```

```
Q2_V2 = j*div;
optVal_V2 = EV(i(j),j);

figure(3);
surf(V2);
view([-150,50]);
xlabel('q_1');
ylabel('q_2');                                                    150
zlabel('E[C] - rVar2[C]');

Q = [Q1, Q2; Q1_V1, Q2_V1; Q1_V2, Q2_V2]
OptimalValues = [optVal_E; optVal_V1; optVal_V2]
```

Bibliography

[1] Melike Baykal-Gürsoy and Keith W. Ross. Variability sensitive markov decision processes. *Mathematics of Operations Research*, 17(3), August 1992.

[2] Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, Massachusetts, second edition, 1999.

[3] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, Massachusetts, second edition, 2000.

[4] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, Belmont, Massachusetts, 1997.

[5] Edwin Burmeister and Kent D. Wall. Kalman filtering estimation of unobserved rational expectations with an application to the german hyperinflation. *Journal of Econometrics*, April 1982.

[6] California Power Exchange. *California's New Electricity Market - The Basics: How the PX Works*, third edition, March 1998. http://www.calpx.com.

[7] Jerzy A. Filar, L.C.M. Kallenberg, and Huey-Miin Lee. Variance-penalized markov decision processes. *Mathematics of Operations Research*, 14(1), February 1989.

[8] Ronald A. Howard. *Dynamic Probabilistic Systems: Markov Models.* John Wiley & Sons, Inc., 1971.

[9] John C. Hull. *Options, Futures, and other Derivatives.* Prentice Hall, fourth edition, 1999.

[10] Marija Ilić, Petter Skantze, and Poonsaeng Visudhiphan. Electricity troubles in california: Who's next? *IEEE Spectrum*, February 2001.

[11] Corwin Joy. Pricing, modeling and managing physical power derivatives. Technical Report, Positron Energy Consulting, 531 Frasier Street, Houston, Texas 77007, September 1998.

[12] James G. Kritikson. *California Electricity Primer*.

[13] Kah-Hoe Ng and Gerald B. Sheblé, editors. *Exploring Risk Management Tools*, Conference on Computational Intelligence for Financial Engineering (CIFEr), New York, 2000. IEEE/IAFE/INFORMS.

[14] Robert S. Pindyck and Daniel L. Rubinfeld. *Econometric Models and Economic Forecasts*. Irwin McGraw-Hill, fourth edition, 1998.

[15] Eduardo S. Schwartz. The stochastic behavior of commodity prices: Implications for valuation and hedging. *The Journal of Finance*, 52(3), July 1997.

[16] Petter Skantze. *A Fundamental Approach to Valuation, Hedging and Speculation in Deregulated Electricity Markets*. PhD thesis, Massachusetts Institute of Technology, June 2001.

[17] Petter Skantze and Marija Ilić. The joint dynamics of electricity spot and forward markets: Implications on formulating dynamic hedging strategies. Technical Report EL 00-005, MIT Energy Laboratory, November 2000.

[18] Petter Skantze and Marija Ilić. Investment dynamics and long term price trends in competitive electricity markets. 2001. To be presented in the 2001 IFAC SME conference.

[19] Petter Skantze, Marija Ilić, and Andrej Gubina. Bid-based stochastic model for electricity prices: The impact of fundamental drivers on market dynamics. Technical Report EL 00-004, MIT Energy Laboratory, November 2000.

[20] Stanislav Uryasev. Conditional value-at-risk: Optimization algorithms and applications. *Financial Engineering News*, 14, February 2000.

[21] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical Report 95-041, Department of Computer Science, University of North Carolina at Chapel Hill, June 2000.