A Modified Interior Point Method for Alternating Minimizations

Technical Report of the ISIS Group
at the University of Notre Dame
ISIS-94-005
March, 1994

Peter T. Szymanski and Michael Lemmon
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

Interdisciplinary Studies of Intelligent Systems

A Modified Interior Point Method for Alternating Minimizations

Peter T. Szymanski and Michael Lemmon *
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556
(219)-631-5480

Keywords: Supervisory Control, Interior Point, Constrained Optimization, Alternating Minimization

March 22, 1994

Abstract

Optimization of non-linear performance functionals subject to constraints is a typical problem in control and there exist many different optimization methods. These methods, however, can take a long time to converge to optimal solutions. This paper presents a *modified interior point algorithm* that optimizes a class of performance functionals possessing both linear and non-linear characteristics. The optimization is an *alternating minimization* (AM) of the linear and non-linear characteristics. The algorithm presented, based upon typical interior point methods from Linear Programming, is fast, requiring $O(\sqrt{n})$ iterations to find a local optimum, and computationally efficient requiring $O(n^{3.5})$ computations to solve the problem. Conditions are specified for the non-linear optimization to guarantee convergence of the algorithm to local optima. Results are presented demonstrating the rates and convergence to local minima. These AM methods are useful in the design of distributed supervisory controllers for large scale systems.

1 Introduction

Optimization of performance functionals plays a central role in the design of supervisory and intelligent controllers. The performance of a system is quantified by a function of the controls and control policy that defines the operation of the controller. Through appropriate choice

*The authors would like to acknowledge the partial financial support of the National Science Foundation (MSS-9216559)

of the controls and the control policy (which usually are constrained due to limited resources or limits on the controller's complexity), the performance functional can be minimized to produce an optimal constrained controller.

The problems that are typically encountered in these optimization problems can stem from the fact that some performance functionals are non-linear. Optimization techniques cannot find globally optimal solutions due to the non-convex nature of the non-linear functions. Also, the optimization techniques can be quite slow and inefficient for large scale problems.

This work deals with the optimization of a class of non-linear objective functions which arise from the supervisory control problem to be presented below. The types of non-linear optimization problems being addressed are those of the form

$$\begin{array}{ll} \text{Minimize} & f(\mathbf{x}, \theta) \\ \text{Subject to:} & \\ & A\mathbf{x} = \mathbf{b} \\ & g(\theta) \leq 0, \end{array} \tag{1}$$

where $A \in \mathbb{R}^{n \times n}$ and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$. It is assumed that $f(\mathbf{x}, \theta)$ is defined such that it can be written as a linear function of the form

$$f(\mathbf{x}, \theta) = f_L(\mathbf{x}) = \mathbf{c}(\theta)^T \mathbf{x}$$

and as a non-linear function of θ, $f_{NL}(\theta)$. The constraints on the problem are assumed to be linear in $\mathbf{x}$. No assumptions are made on the nature of the constraints for the non-linear problem.

Current methods for optimizing supervisory, intelligent control problems are varied. Work by Moed[1] uses genetic algorithms to find minimum entropy solutions for the organization level of Saridis' hierarchy of intelligent control[2]. There, genetic algorithms are used to produce solutions. However, genetic algorithms in general are heuristic in nature, depending upon the parameters chosen for reproduction, recombination, and mutation, and they may not find optimal solutions.

Expectation-Maximization (EM) approaches[3] have been used in the neural networks community to find locally optimal solutions for mixtures of expert networks[4, 5]. These approaches find sets of parameters that maximize the *log likelihood* function. They converge to locally optimal solutions and have linear convergence properties[6], but their convergence analysis exhibits rates which depend upon knowledge of the distance between the current and optimal solutions (which is usually unknown).

Alternating minimization techniques have been applied by Byrne[7] to find optimal Boltzmann machines. This approach minimizes a measure of information divergence alternately over two sets. The method finds solutions which are locally optimal, but the speed of convergence is not known.

This paper examines an alternative approach to solving the aforementioned class of non-linear optimization problems for the design of supervisory controllers. The approach is based upon interior point methods[8] for solving linear programming problems. These methods are fast, having $O(n^{3.5})$ computational complexity and are known to converge to optimal solutions in $O(\sqrt{n})$ iterations. Interior point methods are guaranteed to find optimal solutions to the linear programming problems.

The proposed method solves this class of non-linear problems using an alternating minimization approach. The method embeds a non-linear optimization within the steps of an interior point method that solves the linear optimization. The paper presents conditions on the minimization of the non-linear components that guarantee the convergence of the overall algorithm to locally optimal solutions. Solutions can be made arbitrarily close to the local optima. The algorithm, as it is applied to the supervisory control problem in this work, retains the computational complexity of $O(n^{3.5})$ of the original interior point algorithm for linear programming problems. It also demonstrates convergence to the local minima in $O(\sqrt{n})$ iterations.

The significance of this approach is that it provides an efficient algorithm for solving a large class of non-linear optimization problems found in numerous applications from code book design in source coding theory to parameter estimation in mixture networks or maximum likelihood estimation problems. This alternating minimization approach has polynomial complexity which means that the operation of the algorithm scales well with increasing problem dimensionality. Thus, as the dimensionality of the control problem becomes large, the algorithm will still work efficiently to find optimal controllers. The proposed approach also provides a general framework for solving control problems using barrier functions and interior point methods, which may be applicable to more general non-linear problems.

The remainder of the paper is as follows. Section 2 motivates the alternating minimization as the solution to a controller design problem. Section 3 details the basic interior point approach to optimization and describes the issue that must be addressed to allow its use as an alternating minimization technique. Section 4 outlines the alternating minimization interior point algorithm. Section 5 offers proofs validating the embedded non-linear optimization step. Simulations in Section 6 support the convergence of the algorithm and its convergence rate as a function of problem dimensionality. Finally, Section 7 discusses the results and motivates future areas of research.

2 Control Formulation

The types of non-linear optimization problems addressed by this work are motivated by the design of controllers that approximate some optimal control function. Assume that a controller has inputs and outputs which come from the sets Z and R, respectively. Z is a finite set of states corresponding to quantized inputs or states from a finite state automaton. Let $M = |Z|$ be the number of inputs states. $R \subset \mathbb{R}^n$ is a finite set of command vectors to the system's actuators. Further, there are probabilities associated with each set where $p_z = \Pr\{z \text{ occurs}\}$ and $q_r = \Pr\{r \text{ chosen}\}$.

It is assumed that there is a function which describes the desired command vectors for each input state. This information either exists as a function available to the system or as a set of data points which describes the function at various states in the task being performed. Let $Q^* : Z \to R$ be a function mapping the set of states into the set of controls. $Q^*(z) = r_z$ defines the desired or optimal control associated with z. This means that each z in the input set has an associated control vector that is optimal with respect to the system performance measure.

Given this problem description, assume that Q^* is to be approximated by the controller of Figure 1. The controller consists of K independent agents, each of which implements a

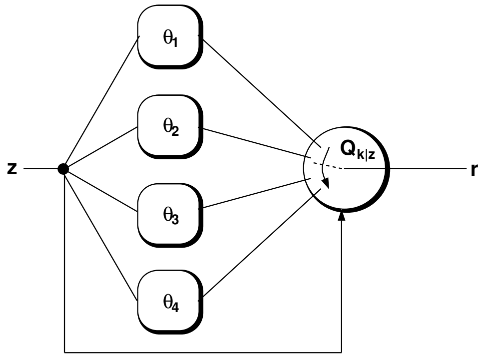

Figure 1: Basic Control Architecture

mapping from inputs to controls, $\theta_k(\cdot) : Z \to R$. For our purposes, $\theta_k(z) = \theta_k$, a constant n-dimensional control vector. Let θ_k be the k^{th} control and let controls be chosen for inputs using a set of probabilities, $Q_{k|z} = \Pr\{\text{agent } k \text{ chosen}|z \text{ occurred}\}$. The set of $Q_{k|z}$'s will be called the *control policy* for the agent. Assume that the controller is free to choose its set of controls and control policies.

For large problems, the cardinality of the set of optimal control vectors can be much larger than the number of agents available in the controller. The problem is the determination of the set of θ_k's (the controls) and the set of $Q_{k|z}$'s (the control policy) that approximates Q^* as well as possible. The optimization problem can be stated formally as

$$\begin{array}{ll} \text{Minimize} & \overline{D}(p, Q, \theta) = \sum_{z,k} p_z Q_{k|z} d(\theta_k, r_z) \\ \text{With respect to:} & Q_{k|z} \text{ and } \theta_k \\ \text{Subject to:} & \\ & \sum_k Q_{k|z} = 1 \\ & Q_{k|z} \geq 0, \theta_k \in \mathbb{R}^n, \end{array} \tag{2}$$

where $d(\theta_k, r_z)$ is a non-negative, convex error or performance measure between the chosen and optimal controls. The optimization problem is stated in its most general form as most of the results presented in succeeding sections do not rely on a specific form for $d(\theta_k, r_z)$. For this work, however, the error measure that will be used to demonstrate the approach is the squared error between the chosen and optimal controls

$$d(\theta_k, r_z) = ||\theta_k - r_z||^2,$$

where $||\theta_k - r_z||^2$ is the squared norm of the difference between the chosen and optimal controls. Results that do rely on the form of the measure are indicated as such.

It is easy to demonstrate that this problem falls into the class of non-linear problems described above. First, $\overline{D}(p, Q, \theta)$ can be written as a linear objective function,

$$\overline{D}_L(p, Q, \theta) = \mathbf{c}(\theta_k)^T \mathbf{x},$$

where $\mathbf{c}(\theta_k) = (p_1||\theta_1 - r_1||^2, p_2||\theta_1 - r_2||^2, \cdots, p_z||\theta_k - r_z||^2)^T$ and $\mathbf{x} = (Q_{1|1}, Q_{1|2}, \cdots, Q_{k|z})^T$. The constraints on the problem with respect to $\mathbf{x}$, which result from the fact that the $Q_{k|z}$'s are constrained to be probabilities, can be written as $A\mathbf{x} = \mathbf{b}$ where

$$A = [I_{M\times M} \cdots I_{M\times M}]$$

and $\mathbf{b} = (1, \cdots, 1)^T$. The problem can also be written as a non-linear, quadratic function

$$\overline{D}_{NL}(\theta_k) = \sum_{z,k} p_z Q_{k|z}(\theta_k - r_z)^T(\theta_k - r_z).$$

Here there are no constraints on the θ_k's. Thus, it is obvious that this problem is of the class of problems described above.

These two representations of the problem lead to the alternating minimization approach. It can be shown that $\overline{D}(p, Q, \theta_k)$ is non-convex with regard to both $Q_{k|z}$ and θ_k simultaneously, since the Hessian of $\overline{D}(p, Q_{k|z}, \theta_k)$ with respect to the complete parameter vector ($Q_{k|z}$ *and* θ_k) is indefinite. Therefore, any optimization technique will most likely produce a solution that is sub-optimal. It can be shown, however, that the function is convex in $Q_{k|z}$ (resulting from the fact that the problem is linear in nature) or in θ_k (proven in the appendix, relying on the quadratic nature of $\overline{D}(\cdot)$) separately. The conjecture is that utilizing the linear and quadratic natures of the function and iteratively optimizing with respect to each representation, the problem can be solved efficiently to produce locally optimal solutions.

Taking this approach results in the following minimization problems. The linear minimization problem is

$$\begin{array}{ll} \text{Minimize} & \overline{D}_L(\mathbf{x}) = \mathbf{c}(\theta_k)^T\mathbf{x} \\ \text{With respect to:} & \mathbf{x} \\ \text{Subject to:} & \\ & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0 \end{array} \tag{3}$$

and the quadratic minimization is

$$\begin{array}{ll} \text{Minimize} & \overline{D}_{NL}(\theta) = \sum_{k,z} p_z Q_{k|z}(\theta_k - r_z)^T(\theta_k - r_z) \\ \text{With respect to:} & \theta_k \in \mathbb{R}^n, \end{array} \tag{4}$$

where there are no constraints on θ_k. The overall problem is solved by first performing the minimization in (3) holding the θ_k's constant and then doing the minimization in (4) holding the $Q_{k|z}$'s constant. The first minimizes performance by changing the control policy while the second minimizes performance by choosing new controls. We show later in this paper that by repeatedly performing this alternating minimization, a locally optimal controller can be found.

The applications of this type of controller are numerous. One possible example of this type of problem could be the control of an industrial robot. Consider a robot that is capable of executing a finite number of controls specified by a set of command vectors, R. (Such commands could be to move the arm at different speeds, to different final positions, with different forces, etc..) The task the robot is to perform can be described by a number of trajectories or data points that map a finite set of input states for the robot, Z, into the set of controls, R. Such a mapping would be the Q^* mentioned above, where $Q^*(z) = r_z$ specifies the control for each input that produces an optimal response. Further assume that the robot is constrained by limited resources (e.g. limited memory or a limited number of microcontrollers) such that it cannot fully implement the Q^* that describes the optimal controls. In other words, it can only implement a finite number, K, of controls where K is much less than the cardinality or R. Thus, the problem becomes one of choosing a subset of controls $\theta_k \subset R$ and a control policy $Q_{k|z}$ to choose controls given the various inputs such that the performance of the approximation (the $Q_{k|z}$'s and the θ_k's) matches the optimal performance (Q^*) as closely as possible. A similar formulation for the control of robot arms with variable payloads can be found in [9].

Remark: This type of formulation is also important in *set point* control. Set point control is concerned with the determination of control parameters in systems which have widely varying responses and characteristics over the state space in which the system operates. Control is achieved by identifying a set of points or regions in the state space and then determining a different set of controls for each of those regions. In the approach proposed here, the different operating regions would correspond to the various inputs. The agents would correspond to various controls or gains available to the system regardless of state. The idea, then, is to determine which subset of controls and which mapping from the operating points to the controls results in the best overall performance of the system taken over all of the operating points.

3 Interior Points

The problem is to optimize a non-linear function subject to constraints. Here, the non-linear function is a performance functional particular to a supervisory control problem with constraints due to the limited set of controls which can be implemented. It was previously mentioned that obtaining a globally optimal solution to a non-linear problem such as this is not possible because of the likely non-convex nature of the function. Therefore, a method which produces locally optimal solutions in an efficient manner is desired. Using the observation that the objective function comes from the class of non-linear functions described in Section 1 leads to treatment of the problem as an alternating minimization procedure.

Treatment of the problem as an interior point procedure follows from the desire for a fast solution algorithm. It is known that linear programming problems can be solved in an extremely efficient manner using interior point methods[8]. Thus, it seems a natural extension to combine fast interior point methods for linear optimization with more standard approaches minimizing the non-linear characteristics of the problem. The goal of such a combination is to embed an optimization with respect to a function's non-linear characteristics into the standard interior point approach while retaining the latter's rate of convergence and convergence properties. Before detailing the alternating minimization approach, a description of the interior point method is in order.

Interior point schemes solve optimization problems by following a path in the interior of the set of feasible solutions to a problem. The *central path* is the set of points

$$\mathbf{x}^*(\alpha) = \arg\min_{\mathbf{x}} f_\alpha(\mathbf{x}), \tag{5}$$

where $\alpha \geq 0$ and $f_\alpha(\mathbf{x}) = \alpha \mathbf{c}^T\mathbf{x} + p(\mathbf{x})$ is an augmented cost function for a linear programming problem. $p(\mathbf{x}) = -\sum_i \log x_i$ is a logarithmic barrier function where x_i is the i^{th} component of $\mathbf{x}$. A logarithmic barrier function is one that increases in value as the distance to the barrier decreases, becoming infinite at the barrier. Such barrier functions keep solutions in the feasible region. Figure 2 depicts a polytope which is assumed to have barriers set up at the boundaries. The dashed lines depict a number of level curves corresponding to different heights of the barrier function. The figure also depicts the central path through the polytope, as well as two hyperplanes corresponding to a cost function which are tangent to the barrier function for two different values of α.

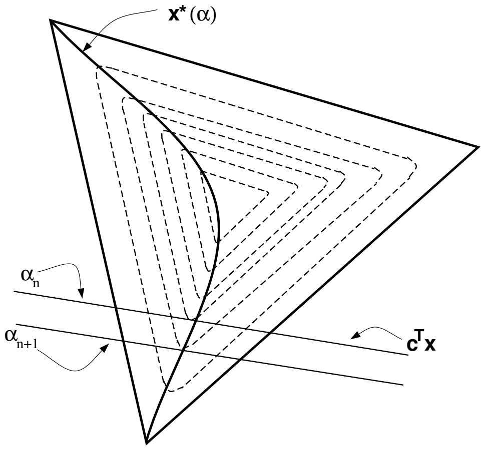

Figure 2: Barrier Functions and Cost Planes

The interior point, or path following, method operates as follows. An initial point for an $\alpha \geq 0$ is found, corresponding to a point on or near the central path. The point is a solution to $f_\alpha(\mathbf{x})$ subject to the constraints $A\mathbf{x} = \mathbf{b}$. The initial solution is a point at or near the point of tangency between the cost hyperplane and the barrier function. Starting with this point, α is increased, which effectively decreases the influence of the barrier functions. This moves the point of tangency between the hyperplane and the barrier function out toward the boundaries of the set. This new point of tangency, corresponding to this new value of α, is reached using gradient descent techniques from the current point. The increase in α, however, is made small to ensure that the distance to the next point of tangency is small. This keeps the next point in a region of quadratic convergence for a gradient descent or Newton-Raphson type update. The process is then repeated and continues until the cost hyperplane is moved arbitrarily close to the boundaries of the set. At this point, α is very large and the contribution from the barrier function is very small. The solution which is found is arbitrarily close to the solution which minimizes $\mathbf{c}^T\mathbf{x}$.

Interior point methods generally use gradient descent techniques to find points on or near the central path. The modified interior point method described in Section 4 is based upon the use of the Scaling Steepest Descent (SSD) algorithm[8] used in the solution of linear programming problems. The SSD algorithm is a gradient descent algorithm that finds new points by scaling the current point to a new coordinate system, projecting the scaled problem into the constraint space of the linear programming problem, performing the descent update on the point, and then scaling the result back to the original space. For a linear problem of the form

$$\begin{array}{ll} \text{Minimize} & d = \mathbf{c}^T\mathbf{x} \\ \text{With respect to:} & \mathbf{x} \\ \text{Subject to:} & \\ & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0. \end{array} \qquad (6)$$

The projection operation is defined as $P_A = I - A^T(AA^T)^{-1}A$ and projects the problem into the null space of A. The scaling transformation is of the form $\mathbf{x} = X\mathbf{y}$. For the interior point approaches, $X = \text{diag}(x_1, x_2, \ldots, x_n)$ scales each vector to a vector of all 1's.

Nearness of points to the central path also plays an important role in interior point problems. Best performance of the interior point algorithm, which employs the SSD algorithm, is achieved when successive points are very near to the central path. If $\delta(\mathbf{x}, \alpha)$ is the distance to the central point in the projected and scaled problem and $\delta(\mathbf{x}, \alpha) < 1$, then the point is near the central path and the SSD algorithm exhibits quadratic convergence[8]. Once a point is near the central path, the interior point algorithm maintains the nearness of successive points through its choice of α.

The initial point must still be chosen to be near the central path through a process called *centralization*. Centralization is the process by which successive points are found that are near the central path, or where the first point is determined such that it is near the central path. In cases where the initial point is not near the central path, a feasible solution can be found and then centralized through repeated application of the SSD algorithm, or it can be computed analytically.

Convergence of the path following methods to optimal solutions is proven by bounding the distance between the current solution and the optimal solution. However, the value of the optimal solution is unknown and cannot be used. Fortunately, mathematical programming offers the *duality gap* as a bound on the distance to an optimal solution[10]. The duality gap is the difference between the current solution for a primal problem and the current solution for the dual problem. For a linear programming problem as in (6), the duality gap is $\Delta = \mathbf{c}^T\mathbf{x} - \mathbf{b}^T\mathbf{u}$ where $\mathbf{c}^T\mathbf{x}$ is the cost of the primal linear programming problem and $\mathbf{b}^T\mathbf{u}$ is the cost of the dual linear programming problem. In linear problems where the primal and dual problems are both feasible, strong duality states that the optimal primal and dual solutions are equal[10], which further implies that $\Delta \geq 0$. For the interior point methods, the duality gap has nice closed form expressions that are a function of α. Thus, they also provide a convenient stopping criterion for these algorithms.

While this procedure is straightforward for linear programming problems, difficulties are encountered when attempting to embed a non-linear optimization within the steps of the interior point algorithm. In an alternating minimization as is being proposed for (2), the interior point scheme optimizes $\overline{D}(p, Q, \theta)$ with respect to the control policy, $Q_{k|z}$. Then, a second minimization is done with respect to the controls, θ_k's. This second minimization is where the difficulties arise.

The difficulties stem from the rotation of the cost hyperplane due to the minimization with respect to θ_k. $\overline{D}(Q, \theta_k)$ can be rewritten as the linear function $\overline{D}_L(Q, \theta_k) = \mathbf{c}(\theta_k)^T\mathbf{x}$, where again $\mathbf{x}$ is the vector of probabilities as defined in Section 2. Any change that occurs in $\mathbf{c}(\theta_k)$ due to changes in θ_k rotates the cost plane about the current solution $\mathbf{x}$. Figure 3 shows how changing the controls can change the point of tangency between the current solution and the next central point. In the figure, $\mathbf{x}_n$ corresponds to the current solution which is also a point of tangency between the cost hyperplane, $\mathbf{c}_n^T\mathbf{x}$, and a level curve of the barrier function. $\mathbf{x}_{n+1}$ corresponds to the next central point, the point of tangency between the new cost hyperplane after rotation $\mathbf{c}_{n+1}^T\mathbf{x}$ and the barrier function. The problem is that if the hyperplane is rotated about $\mathbf{x}_n$ too much, the distance to the next central point δ may be too great, taking the next point outside the region of quadratic convergence. This is exactly the obstacle that must be overcome before an efficient interior point method can be developed for AM algorithms. Basically, this problem will be solved by determining how aggressively each leg of the optimization can be solved without moving the point too far from the central path.

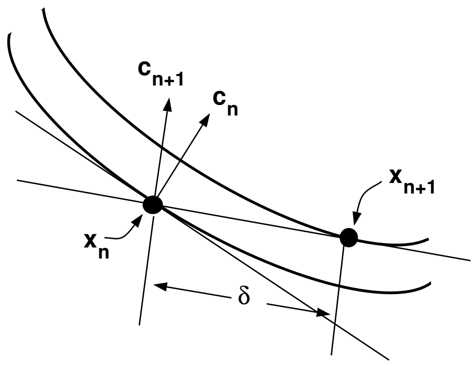

Figure 3: Effect of Changing Cost Planes

Remark: When the problem is formulated as it is for the control problem in Section 2, it consists of finding an approximation of Q^* through the choice of $Q_{k|z}$ and θ_k. For the industrial robot example, this approach assumes that the controller has available a description of Q^* (the set of data points that describes the trajectories) which it can use to guide its choice of parameters. Through the use of an interior point alternating minimization, the controller determines a good approximation for Q^* by alternately improving first its control policy ($Q_{k|z}$) and then its set of controls (θ_k) using the information contained in the data points describing the trajectories. In other words, it is determining a set of parameters that allow it to approximate the original function as closely as possible.

Such an approach is very similar to the estimation of distributions from incomplete or partial observations using the Expectation-Maximization (EM) algorithm[3]. An EM approach assumes an underlying family of densities, $f(\mathbf{x}|\phi)$, which describe the original data or function. The underlying densities depend upon the control parameter vector ϕ. The desire is to approximate this original family by a hypothesized family of densities, $g(\mathbf{y}|\phi)$. The idea is to use a set of observations $\mathcal{Y}$ to infer the structure and parameters of $\mathcal{X}$, corresponding to the original data set.

EM approaches have been implemented using alternating minimization procedures. These procedures operate by computing, at each step, estimates for the probabilities involved corresponding to the expectation step, and then computing maximum likelihood estimates on the parameters to perform the maximization step. Some examples of EM algorithms are Blahut's Algorithm from source coding theory[11], or the training methods used in the mixtures of experts approach from the neural networks community[4, 5].

4 Modified Interior Point Algorithm

The specific interior point alternating minimization algorithm proposed follows here. It is closely related to the path following methods described in [8]. A brief description of the steps common to the basic interior point algorithm, along with a more thorough description of the modification, follows here. The next section outlines the proofs of the steps which are specific to the modified interior point algorithm, while proofs of the basic interior point steps can be found in [8] and in the appendix.

Before presenting the algorithm, a declaration of the variables and quantities involved is in order. $\mathbf{x}^{(m)}$ and $\theta_k^{(m)}$ will denote the probability vector and choice of controls at iteration m. $\alpha^{(m)}$ denotes the weighting coefficient at time m. β is the update parameter for α and P_{AX}, X, and $\nabla_{\mathbf{x}} f_\alpha(\mathbf{x})$ are the projection matrix, the scaling transformation, and the gradient of $f()$ respectively. $\mathbf{c}(\theta_k^{(m)})$ is the cost vector corresponding to θ_k at time m and $\mathbf{c}(\theta_k)^{(m),*}$ is the minimizing cost vector at time m. Finally, ϵ is the tolerance that indicates how close a solution must be to a local optimum in order to terminate the algorithm.

Algorithm 4.1 (Modified Interior Point Algorithm)

`Initialize.`

  `Set` $m = 0$.

  `Centralize` $\mathbf{x}^{(m)}$ `by setting` $x_i = 1/K$.

  `Choose an initial set of` $\theta_k^{(m)}$`'s.`

  `Compute` $\alpha^{(m)}$ `such that` $\delta(\mathbf{x}^{(m)}, \alpha^{(m)}) = ||P_{AX} X \nabla_{\mathbf{x}} f_\alpha(\mathbf{x})|| \leq 0.5$.

repeat

  `Update` $\alpha^{(m+1)} = \alpha^{(m)}/\beta$, `where` $1/\beta \leq (1 + 0.1/\sqrt{n})$.

  `Find` $\mathbf{x}^{(m+1)}$ `by performing` $k \geq 2$ `iterations of the Scaling Steepest Descent (SSD) algorithm[8].`

  `Find` $\mathbf{c}(\theta_{\mathbf{k}}{}^{(m+1)}) = (1-\gamma)\mathbf{c}(\theta_{\mathbf{k}}{}^{(m)}) + \gamma \mathbf{c}(\theta_{\mathbf{k}}{}^{(m+1),*})$, `where` $\gamma \leq \gamma(\mathbf{x}^{(m+1)}, \alpha^{(m+1)}, \mathbf{c}(\theta_k^{(m)}))$.

  `Set m = m + 1`

until$(\Delta < \epsilon)$

The initialization portion of the algorithm consists of three basic steps. First, the problem is centralized by choosing an appropriate set of values for the initial probabilities. It can be shown that when $\alpha = 0$, the center of the barrier function occurs at the point where the probabilities are uniform ($x_i = 1/K$). Second, an initial set of controls (θ_k's) are chosen.

This is assumed to be some random choice of the controls over the control space R. Third, an initial value of α needs to be computed to ensure that the first central point is near the next central point (within $\delta(\mathbf{x}^{(0)}, \alpha) \leq 0.5$ of the next central point where $\delta(\mathbf{x}^{(0)}, \alpha)$ is a measure of the distance).

The main body of the algorithm consists of the **repeat-until** loop which consists of four steps. First, is the update of α. α is increased at every step through multiplication by a term of the form $1/\beta$. This is a simple geometric update of the weighting factor. The choice of β is chosen such that a single iteration of the SSD algorithm can be taken for each α. (This is shown in Section 5 in a lemma from [8].)

The second step of the body of the algorithm finds the next central point given the current point. It performs two or more iterations of the SSD algorithm to find the next point. This seemingly contradicts the statement in the last paragraph, but that is misleading. A single step would suffice if one were not doing alternating minimizations. However, since a non-linear optimization occurs next, the "extra" steps are needed, but they do not change the convergence properties of the algorithm.

Updating the cost vector, which is the next step in the body, is the main modification to the basic interior point algorithm. Here, the cost vector is updated by taking a convex combination of the minimizing cost vector and the current cost vector. The minimizing cost vector results from a Newton-Raphson (NR) iteration that minimizes $\overline{D}(p, Q, \theta_k)$ with respect to θ_k holding the control policy constant. Since the function is quadratic, a single NR step produces the minimizing set of controls as

$$\theta_k^{(m+1)} = \frac{\sum_z p_z Q_{k|z}^{(m+1)} r_z}{\sum_z p_z Q_{k|z}^{(m+1)}}. \tag{7}$$

(For a proof of this step, see Theorem A.3 in the appendix.) The minimizing cost vector is the cost vector computed using this optimal set of controls.

Examination of the cost vector update step indicates that the update is constrained. A parameter, γ, controls the relative contributions of the current and minimizing cost vector. This parameter is required to control the rotation of the cost hyperplane. Recall from Section 3 that rotating the cost plane too much may cause the current solution to be too far from the next central point. Thus, constraining the contributions of the current and minimizing cost vectors guarantees that the distance is held within acceptable limits. The value for γ for each step that maintains acceptable distances is

$$\gamma \leq \frac{\delta_2 - \delta_1}{\alpha Q_{max} ||\mathbf{c}(\theta_k^{(m+1),*}) - \mathbf{c}(\theta_k^{(m)})||}$$

where $\delta_2 \leq 0.5$ and $0 < \delta_1 < \delta_2$. This is shown in the next section.

Finally, the last part of the algorithm is the check for convergence. Here, the duality gap, Δ, is checked against a predefined tolerance, ϵ. If the gap is small enough, the algorithm is terminated as the solution is within ϵ of a local minimum. If the gap is not small enough, m is incremented and the loop repeats. The formula that defines the duality gap is

$$\Delta = (n + \delta_2\sqrt{n})/\alpha^{(m)}.$$

This formula is derived in [8] and is repeated in the appendix.

5 Convergence

This section presents various theorems and lemmas relating to the operation of the modified interior point algorithm and the choice of the various parameters. It details mostly the modifications, but does present some results from [8] that are necessary to the explanation of the convergence results for the modified interior point approach.

Beginning with the initialization section of the algorithm, the following theorem examines the initial centralization of the probabilities. In typical interior point methods, the centralization of any point for a given value of α is not a trivial task. However, in this problem, there is the benefit of being restricted to the set of all pmf's. Centralization for this problem assumes that there is no contribution from the cost function ($\alpha = 0$) and one only needs to minimize the barrier function.

Theorem 5.1 *For $p(\mathbf{Q}) = \sum_{k,z} \log \frac{1}{Q_{k|z}}$ such that for all $z \in Z$, $\sum_{k=1}^{K} Q_{k|z} = 1$, $p(\mathbf{Q})$ is minimized by*

$$Q_{k|z} = 1/K.$$

Proof: Let $f(\mathbf{Q}) = p(\mathbf{Q}) + \sum_z \lambda_z[(\sum_k Q_{k|z}) - 1]$, where λ_z is a Lagrange multiplier. Taking the gradient of $f(\mathbf{Q})$ with respect to $Q_{k|z}$ yields

$$\partial f(\mathbf{Q})/\partial Q_{k|z} = -1/Q_{k|z} + \lambda_z.$$

For the point which minimizes $f(\mathbf{Q})$, $\nabla f(\mathbf{Q}) = 0$. The only way this can happen is for $\lambda_z = 1/Q_{k|z}$, implying that all of the $Q_{k|z}$'s are equal. Since the $Q_{k|z}$'s are probabilities, the only way this can occur is when $Q_{k|z} = 1/K$. $\square$

Thus, we can find the exact center of the barrier function in $O(n)$ time, where $n = K \cdot M$.

The convergence properties of the interior point approach depend upon the nearness of points to the central path. The algorithm determines a sequence of solutions (controls and control policies) which are found from previous solutions. If an initial point is near the central path, the following lemma guarantees that the point resulting from an application of the SSD algorithm will also be near the central path.

Lemma 5.1 (See [8]) *If the distance, $\delta(\mathbf{x}^{(m)}, \alpha^{(m)}) = ||\overline{h}(\mathbf{x}, \alpha)||$, to a central point for a given α is such that $\delta(\mathbf{x}, \alpha^{(m)}) \leq 0.5$, and α is updated as $\alpha^{(m+1)} = \alpha^{(m)}/\beta$ where $1/\beta < (1 + 0.1/\sqrt{n})$ then the SSD algorithm finds a new point such that $\delta(\mathbf{x}^{(m+1)}, \alpha^{(m+1)}) \leq 0.5$ in a single step.*

In the lemma, the condition $\delta(\mathbf{x}, \alpha) \leq 0.5$ guarantees that the control solution is close to the central path and is in a region where the SSD algorithm exhibits quadratic convergence. Thus, if the initial point meets this requirement, one iteration of the SSD algorithm for each value of α produces a sequence of solution points where each point is also close to the central path and in the region of quadratic convergence. It is this nearness to the central path that gives the original algorithm its speed.

The following corollary to Lemma 5.1, which was developed in this work, indicates how much closer the resulting point will be to a central point when more than one iteration of the SSD algorithm is taken.

Corollary 5.1 *Assuming the same conditions as in Lemma 5.1, k iterations of the SSD algorithm produce a point* $\mathbf{x}_k$ *such that* $\delta(\mathbf{x}_k, \alpha) \leq \delta_1 = (0.7)^{2^k}$.

Proof: It was shown in [8] that if $\delta(\mathbf{x}^{(m)}, \alpha^{(m)}) \leq 0.5$, then when α is updated as in Lemma 5.1 that $\delta(\mathbf{x}^{(m)}, \alpha^{(m+1)}) \leq 0.7$. Lemma 5.4 in [8] shows that $\delta(\mathbf{x}^{(m)}, \alpha^{(m+1)}) \leq 1$ is in the region of quadratic convergence for the SSD algorithm. Therefore, one iteration of the SSD algorithm from $\mathbf{x}^{(m)}$ produces a new point $\mathbf{x}_1^{(m+1)}$ where $\delta(\mathbf{x}_1^{(m+1)}, \alpha^{(m+1)}) \leq \delta(\mathbf{x}^{(m)}, \alpha^{(m+1)})^2$. But $\delta(\mathbf{x}^{(\mathbf{m})}, \alpha^{(m+1)}) \leq 0.7$ so $\delta(\mathbf{x}_1^{(m+1)}, \alpha^{(m+1)}) \leq (0.7)^2$. But the new point, $\mathbf{x}_1^{(m+1)}$ is already in the region of quadratic convergence so another step would produce $\delta(\mathbf{x}_2^{(m+1)}, \alpha^{(m+1)}) \leq \delta(\mathbf{x}_1^{(m+1)}, \alpha^{(m+1)})^2 \leq (0.49)^2 = (0.7)^{2^2}$. Proceeding inductively, assuming that $\delta(\mathbf{x}_k^{(m+1)}, \alpha^{(m+1)}) \leq (0.7)^{2^k}$, then $\delta(\mathbf{x}_{k+1}^{(m+1)}, \alpha^{(m+1)}) \leq \delta(\mathbf{x}_k^{(m+1)}, \alpha^{(m+1)})^2 \leq ((0.7)^{2^k})^2 = (0.7)^{2^{k+1}}$, thus proving the conjecture. $\square$.

The extra "closeness" that results from the additional SSD iterations comes into play when performing the update to the cost vectors.

The minimization of the cost functional in the update of the cost vectors results in a different $\mathbf{c}(\theta)$ for the linear problem. If the non-linear minimization changes $\mathbf{c}(\theta)$ too much, the distance to the next central point will be too great. In this case, the point is no longer in the region of quadratic convergence for the SSD algorithm and the method loses its speed. Thus, one wants to limit the change in the cost vector. Taking the update step to be the convex combination of the current and minimizing cost vector as in the modified algorithm, one can control the rotation of the cost hyperplane through the choice of γ. The following theorem, which is the main result of this paper, specifies the choice of γ to ensure that the point is still near the next central point after cost plane rotation.

Theorem 5.2 *Let* $\mathbf{c}(\theta_k^{(m)}) = \mathbf{c}^{(m)}$ *and* $\mathbf{c}(\theta_k^{(m),*}) = \mathbf{c}^{(m),*}$ *be the current cost and the minimizing cost vectors. Assume that* $\delta(\mathbf{x}, \alpha, \mathbf{c}^{(m)}) \leq \delta_1 < 0.5$ *and let* $\mathbf{c}^{(m+1)} = (1-\gamma)\mathbf{c}^{(m)} + \gamma\mathbf{c}^{(m+1),*}$. *If* γ *is chosen as*

$$\gamma \leq \frac{\delta_2 - \delta_1}{\alpha Q_{max}||\mathbf{c}^{*,(m+1)} - \mathbf{c}(\theta)^{(m)}||} \tag{8}$$

where $\delta_1 < \delta_2 = 0.5$, *then* $\delta(\mathbf{x}, \alpha, \mathbf{c}^{(m+1)}) \leq \delta_2 = 0.5$.

Proof: First, we know by assumption that $\delta(\mathbf{x}, \alpha, \mathbf{c}^{(m)}) \leq \delta_1 < 0.5$. We are interested in defining γ such that $\delta(\mathbf{x}, \alpha, \mathbf{c}^{(m+1)}) \leq 0.5$. From [8], we know that $\delta(\mathbf{x}, \alpha, \mathbf{c}^{(m)}) = ||\mathbf{h}(\mathbf{x}, \alpha, c^{(m)})||$, the norm of the step taken in the SSD algorithm. Let $\mathbf{h}_1 = \mathbf{h}(\mathbf{x}, \alpha, \mathbf{c}^{(m)})$ and let $\mathbf{h}_2 = \mathbf{h}(\mathbf{x}, \alpha, \mathbf{c}^{(m+1)})$. Using the triangle inequality, it can be shown that

$$||\mathbf{h}_2|| \leq ||\mathbf{h}_2 - \mathbf{h}_1|| + ||\mathbf{h}_1||.$$

But by assumption, we know that $||\mathbf{h}_1|| \leq \delta_1$. Using this and the fact that $\mathbf{h}(\mathbf{x}, \alpha, \mathbf{c}(\theta_k)) = \alpha P_{AX} X \mathbf{c}(\theta_k) - \mathbf{e}_p$[8] yields

$$||\mathbf{h}_2|| \leq ||\alpha P_{AX} X(\mathbf{c}^{(m+1)} - \mathbf{c}^{(m)})|| + \delta_1.$$

Recognizing that $(\mathbf{c}^{(m+1)} - \mathbf{c}^{(m)}) = (1-\gamma)\mathbf{c}^{(m)} + \gamma\mathbf{c}^{(m+1),*} - \mathbf{c}^{(m)}$, results in

$$||\mathbf{h}_2|| \leq ||\alpha\gamma P_{AX} X(\mathbf{c}^{(m+1),*} - \mathbf{c}^{(m)})|| + \delta_1.$$

Since the projection of any vector into the null space of the constraints is less than or equal to the length of the vector itself, the following results

$$||\mathbf{h}_2|| \leq ||\alpha\gamma X(\mathbf{c}^{(m+1),*} - \mathbf{c}^{(m)})|| + \delta_1.$$

Now, extracting the constants from the norm, realizing that $||A\mathbf{x}|| \leq ||A||\ ||\mathbf{x}||$, and that $||X||$ is the maximum eigenvalue of X results in

$$||\mathbf{h}_2|| \leq \alpha\gamma Q_{max}||\mathbf{c}^{(m+1),*} - \mathbf{c}^{(m)}|| + \delta_1.$$

Plugging in the value of γ from (8) yields

$$||\mathbf{h}_2|| \leq \left[\frac{\alpha Q_{max}||\mathbf{c}^{(m+1),*} - \mathbf{c}^{(m)}||(\delta_2 - \delta_1)}{\alpha Q_{max}||\mathbf{c}^{(m+1),*} - \mathbf{c}^{(m)}||}\right] + \delta_1.$$

Simplifying produces the desired result that $||\mathbf{h}_2|| \leq \delta_2 \leq 0.5$ and the theorem is proved. □

This theorem says that if γ is chosen small enough, then it can be guaranteed that the distance to the new central point after cost hyperplane rotation will still place the current point in the region of quadratic convergence, provided that the current point was well within the region of quadratic convergence of the original central point. More simply put, if the solution point is close enough to the original central point, then the cost can be updated so that it will still be near the next central point. It is obvious, then, how the extra closeness provided by Corollary 5.1 is utilized by the update of the cost vectors. There is, however, a tradeoff implicit in the number of SSD steps taken. Smaller numbers of SSD steps take less computational time but produce δ_1's that are closer to $\delta_2 = 0.5$. Larger numbers of SSD steps take greater computational time, but the range for γ can be greater as δ_1 is getting smaller. In order to perform the alternating minimizations (that is, to have nonzero values of γ that allow minimization with respect to θ_k) , the following theorem shows that two SSD steps and a convex combination of the minimizing and current cost vectors will suffice to produce a new point which is close to the central path.

Theorem 5.3 *Let $\delta(\mathbf{x}^{(m)}, \alpha^{(m)}, \mathbf{c}(\theta_k)^{(m)}) \leq 0.5$ and let α be updated as in Lemma 5.1. Let γ be updated as in (8) and let $\mathbf{c}(\theta_k^{(m+1)}) = (1-\gamma)\mathbf{c}(\theta_k^{(m)}) + \gamma\mathbf{c}(\theta_k^{(m+1),*})$. Then, one iteration of the* **repeat-until** *loop of Algorithm 4.1 with two SSD iterations produces a new point, $\mathbf{x}^{(m+1)}$ such that $\delta(\mathbf{x}^{(m+1)}, \alpha^{(m+1)}, \mathbf{c}(\theta_k^{(m+1)}) \leq 0.5$.*

Proof: The proof proceeds in two steps, first examining the effects of two SSD steps on δ, and then examining the effects of the update of the cost vector on δ.

First, $\delta(\mathbf{x}^{(m)}, \alpha^{(m)}, \mathbf{c}(\theta_k)^{(m)}) \leq 0.5$ by assumption, meaning the current point is near the current central point. Since α is updated according to the conditions of Lemma 5.1, two iterations of the SSD algorithm produce a new point, where according to Corollary 5.1 $\delta(\mathbf{x}_2^{(m+1)}, \alpha^{(m+1)}, \mathbf{c}(\theta_k^{(m)})) \leq \delta_1 = (0.7)^4$. Thus, the point after two SSD iterations is guaranteed to be $\delta(\mathbf{x}_2^{(m+1)}, \alpha^{(m+1)}, \mathbf{c}(\theta_k^{(m)})) \leq 0.2401$.

Now, take this value to be the starting value for δ in the cost vector update. Then, $\delta(\mathbf{x}_2^{(m+1)}, \alpha^{(m+1)}, \mathbf{c}(\theta_k^{(m)}))$ satisfies the conditions of Theorem 5.2 in that it is strictly less than 0.5. Therefore, a valid γ can be chosen according to (8) where $\delta_2 = 0.5$ and δ_1 is the result from the two SSD steps. Using Theorem 5.2, the point after updating the cost vector is such that

$$\delta(\mathbf{x}^{(m+1)}, \alpha^{(m+1)}, \mathbf{c}(\theta_k^{(m+1)})) \leq 0.5,$$

and the theorem is proved. □

The speed of the modified interior point algorithm for the specific sort of controls problems described by (2) results from the speed of the basic interior point algorithm and the speed of the non-linear optimization for this particular problem. Since the non-linear function of θ_k is quadratic in θ_k, it can be solved via (7) in a single step using a Newton-Raphson approach. This step has a computational complexity of $O(n)$, which leads to the following theorem concerning the overall computational complexity of the modified interior point algorithm

Theorem 5.4 *The computational complexity of the modified interior point algorithm is* $O(n^{3.5})$.

Proof: The proof of this theorem follows in two steps. First, the number of iterations required to terminate the algorithm is $O(\sqrt{n})$. This follows from Lemma 6.2 and Lemma 7.2 in [8]. There, it is shown that the number of iterations of the basic algorithm necessary to produce a solution is $O(\sqrt{n})$ when α is updated as in Lemma 5.1.

The maximum computational complexity of any step in the basic algorithm is that associated with the matrix inversion in the SSD step, which is $O(n^3)$. Since the modification to the algorithm consists of the Newton-Raphson update which has $O(n)$ complexity, the maximum complexity step is still the $O(n^3)$ NR step. Thus the overall computational complexity of the algorithm is $O(n^{3.5})$. □

The last result details the convergence of the algorithm to locally optimal solutions. It relies on the fact that the duality gap bounds the distance to optimal solutions at each step. Since the duality gap does not increase in any of the steps, and is in fact a monotone decreasing sequence bounded below by zero, it says that the sequence converges to zero.

Theorem 5.5 *For the modified interior point algorithm defined above, the sequence of solutions,* $\{\mathbf{x}^{(m)}\}$, *converges to a locally optimal solution* $\mathbf{x}^*$.

Proof: The proof of the theorem relies on results concerning the duality gap which are found in [8] and which are repeated in the appendix. Mathematical programming defines the duality gap as $\Delta = \mathbf{c}(\theta_k)^T\mathbf{x} - \mathbf{b}^T\mathbf{u}$, the difference between primal and dual solutions. Since this problem deals with a sequence of linear programming problems, and since the set of feasible solutions for each problem is convex and bounded, we know by strong duality[10] that $d^* = \mathbf{c}(\theta_k)^T\mathbf{x}^* = v^* = \mathbf{b}^T\mathbf{u}^*$. From Lemma A.2, the duality gap for each problem in the sequence is

$$\Delta^{(m)} = (n + \epsilon\sqrt{n})/\alpha^{(m)}.$$

Note that in the algorithm, changing the cost vectors and the solutions does not change the value of the duality gap, as the duality gap is controlled by α. As long as the solutions remain close to the central path, the size of the duality gap is unaffected by changes to the solution and cost vectors. As $\alpha^{(m)} \to \infty$, $\Delta^{(m)} \to 0$. This implies that, $d^{(m)} \to d^*$ and $v^{(m)} \to v^*$. Thus, an optimum solution is found. However, due to the non-convex nature of the composite problem, all that can be said about the nature of the optimum is that it is locally optimal. $\square$

6 Simulations

Simulations were run that demonstrate the ability of the approach to approximate control functions for supervisory systems. The simulations presented here assume that a controller has available two agents with which to implement controls. It chooses the controls and determines a control policy based on a set of data points that represent the desired controls in response to the set of inputs. The problem set over which the simulations were run is similar to the data used in [6] and a sample of this data set appears in Figure 4. The data was drawn from the following equations: $f_1(x) = 0.8x - 1.0 + \mathrm{N}(0, \sigma)$ and $f_2(x) = 0.4x + 2.0 + \mathrm{N}(0, \sigma)$, where x is a uniformly distributed random variable $x \in [X_L, X_U]$ ($x \in [-1, 1]$ for $f_1(x)$ and $x \in [2, 4]$ for $f_2(x)$) and $\mathrm{N}(0, \sigma)$ is a normally distributed random variable with mean 0 and standard deviation of σ. Data points were chosen from $f_1(x)$ with a probability of 0.4 and from $f_2(x)$ with a probability of 0.6.

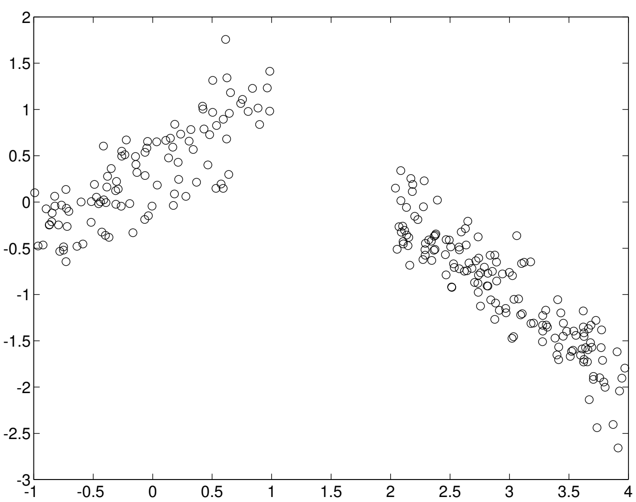

Figure 4: Data Set

Various sizes of input sets were tested for the control problem. The dimension of the problem was varied from $n = 10$ to $n = 100$. The variation in the size of the problem is used to check the convergence characteristics of the algorithm to see how they compared with the theoretical convergence rate of $O(\sqrt{n})$ iterations. The resulting data points and theoretical bound for these experiments appear in Figure 5. The data exhibit a linear characteristic in the log-log plot and match the predicted rate extremely well.

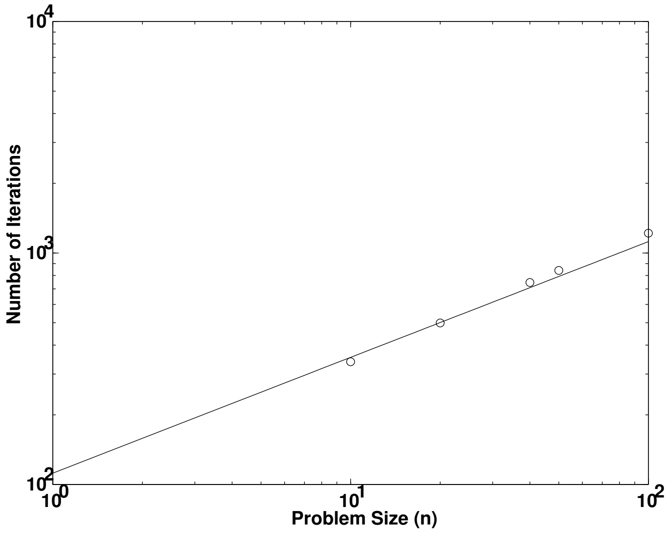

Figure 5: Convergence Rate

The effects of the value of γ on the final solution were also examined. A number of simulations were run on problem sizes from 10 to 50, with the results corresponding to $n = 40$ shown in Figure 6. The results indicate that as γ is increased, the performance measure for the problem gets worse. In this case, the data appear as a step. This suggests that for larger values of γ, the problem is unable to find good solutions either because the solutions get far from the central path or the cost hyperplanes rotate to cause the algorithm to converge to the center of the polytope which corresponds to the least complex control policy. Similar results were obtained for the other cases when $n = 10, 20, 50$.

The next set of results examines how the normalized distance varies over the solution of the problem from the initial point to the final solution. Various simulations were run to test this. The first set of simulations were those where the exact bound in (8) was used for γ. In these cases, the solution converged to the degenerate case of uniform probabilities. After the first iteration, $\delta(\mathbf{x}, \alpha)$ was approximately zero and γ shot up to one. The second set used a more heuristic approach for γ where

$$\gamma = \min(\gamma, 0.001\alpha).$$

Figure 7 displays two different curves where the top curve is the computed value for γ from (8) while the bottom curve is the value of γ from the above heuristic schedule. In this case, the solution is not degenerate ($Q_{k|z}$ is not uniform) and the θ_k's converge to distinct values. Figure 8 shows two different curves corresponding to $\delta(\mathbf{x}, \alpha)$ before updating α (bottom curve) and after (top curve). For the lower curve, $\delta()$ never exceeds 0.5, and for the upper curve, $\delta()$ never exceeds 0.7, which are the upper bounds for both cases. Thus, the results demonstrate that changing γ as indicated by (8) does not take the solution points too far from the next central point.

The final two results (Figures 9 and 10) show the convergence of the parameters (the $Q_{k|z}$'s and θ_k's) to their final values for a problem size where $n = 50$. In the case presented, the control policy converges to a maximum complexity solution. The maximum complexity

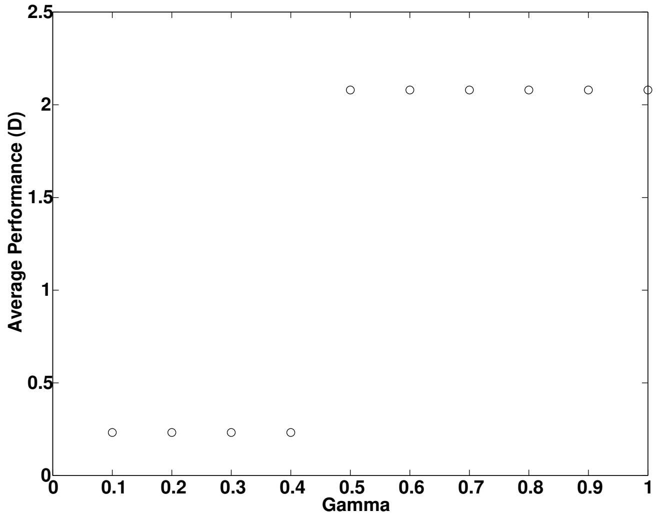

Figure 6: Effects of Gamma on Performance

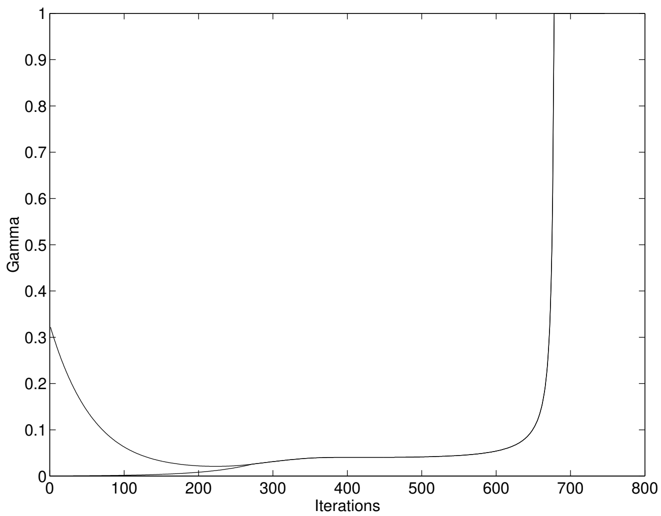

Figure 7: γ as a Function of α

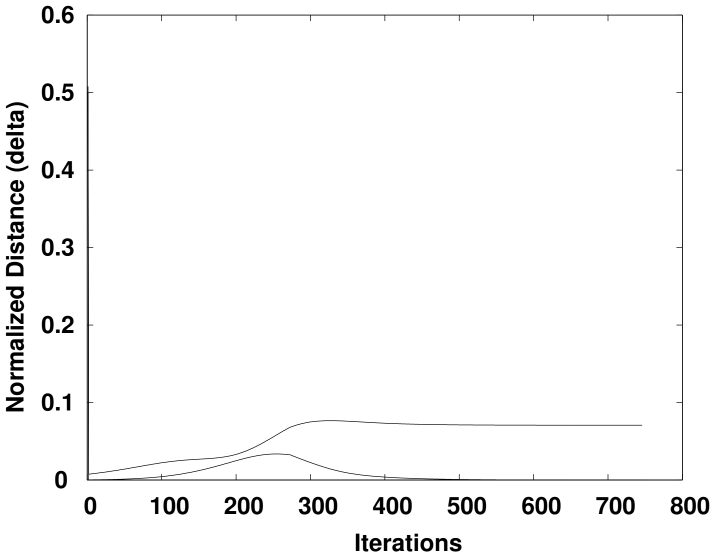

Figure 8: Normalized Distance vs Iterations

solution is one where the controller always chooses the control which best approximates its optimal control. The controls converge to values which best approximate the two different sets of points.

7 Discussion

The results demonstrate that the modified interior point algorithm is a viable approach for solving the type of non-linear optimization problems that can arise in supervisory control. By alternately minimizing over the set of controls and the control policy, an optimal controller is found that approximates the optimal control function. The method is fast for the case presented having a convergence rate of $O(\sqrt{n})$ iterations and a computational complexity of $O(n^{3.5})$ operations.

While the method is proven to converge to locally optimal solutions, the optimality of the resulting solutions depends to a great degree upon the choice of γ. Satisfying the bound in (8) guarantees that the algorithm will converge quickly to a locally optimal solution. However, it does not say anything about the nature of the solution. It is possible that in some cases, the resulting solution will be degenerate, meaning that the solution has converged to a solution where the control policy is a uniform distribution over the controls. This implies that the control policy uses no state information in choosing controls, which is not desirable in a good system. Additional work will address the improvements to the algorithm concerning appropriate schedules for γ to guarantee that degenerate solutions will not occur.

Another area which addresses degeneracy includes examination and greater use of the constrained optimization. In cases where degenerate solutions occur, additional barriers corresponding to added constraints on the problem may serve to eliminate the degeneracy.

The methods presented here are also connected to the operation of EM algorithms. These approaches try to maximize the log likelihood function for a set of probabilities. But the

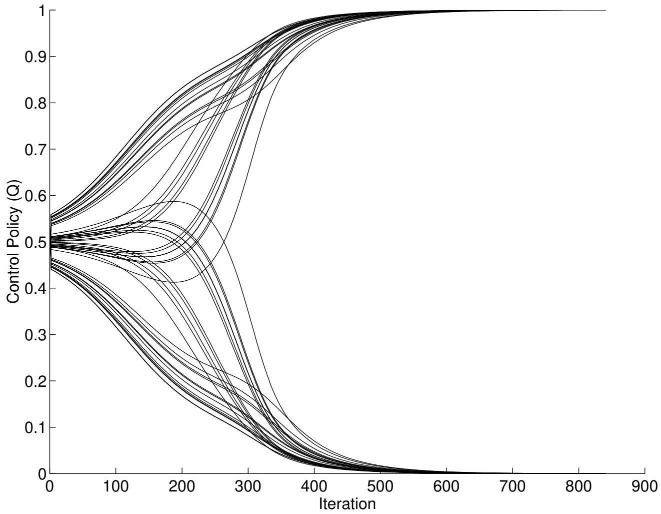

Figure 9: Convergence of Control Policy

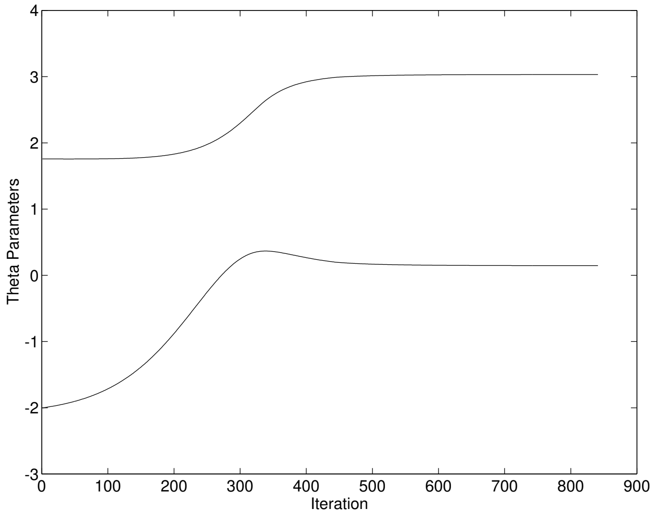

Figure 10: Convergence of Controls

log likelihood function is nothing more than a barrier function where the boundaries are set up where the probabilities are equal to zero. There is no explicit distortion measure, but it is "built in" to the probabilities by the types of distributions chosen. For the adaptive mixtures approach, the performance function is embedded into the Gaussians as the squared norm. Thus, it is not surprising that these approaches exhibit good convergence properties.

The alternating minimization approach presented here is also a general approach. The specific algorithm presented is tuned to the 2-norm and to supervisory control. However, the theorems presented do not depend upon the choice of the performance functional, nor on the optimization technique chosen. The method only relies on the knowledge of a minimizing cost vector. Thus, if a fast method exists for determining such a minimizing cost vector, then the method will still work and should be fast.

Future work will attempt to extend the basic algorithm to other areas. Since the convergence of the algorithm only depends upon the change of the cost vector, any non-linear optimization could be used, meaning that the approach need not be constrained to quadratic problems. Additionally, the performance measure need not be a Euclidean norm, it could be any "valid" measure. Use of different measures could allow the method to address optimization in neural networks or supervisory control of discrete event systems.

A Appendix

This appendix contains various proofs concerning the convexity of the component problems and the definition and form of the duality gap for the problem at hand.

First is a simple theorem proving the convexity of the performance functional with respect to θ_k.

Theorem A.1 *For constant p_z, $Q_{k|z}$, $\mathbf{r_z}$, where p_z and $Q_{k|z}$ are valid pmf's and $\mathbf{r_z}, \theta_\mathbf{k} \in \mathbb{R}^n$, the average distortion $\overline{D}(\theta) = \sum_{k,z} p_z Q_{k|z}(\theta_k - \mathbf{r}_z)^T(\theta_k - \mathbf{r}_z)$ is convex in θ.*

Proof: A twice differentiable function $F(\mathbf{x})$ is convex over a set S if and only if the Hessian of the function $\nabla^2 F(\mathbf{x})$ is positive semidefinite for each $\mathbf{x} \in S$. The gradient of $\overline{D}$ is the vector

$$\nabla\overline{D}(\theta) = \frac{\partial}{\partial\theta_{k,l}} = 2\sum_z p_z Q_{k|z}(\theta_{k,l} - \mathbf{r}_{z,l}). \tag{9}$$

Taking the derivative again yields the Hessian

$$\nabla^2\overline{D}(\theta) = \frac{\partial}{\partial\theta_{k,l}}\frac{\partial}{\partial\theta_{k,l}^T} = \mathrm{diag}(2\sum_z p_z Q_{k|z}). \tag{10}$$

The Hessian is a diagonal matrix with non-negative eigenvalues. Since the eigenvalues are all non-negative, the Hessian is positive semidefinite. Thus, the function is convex in θ. □

A similar proof is unnecessary to prove the convexity of the function with respect to the $Q_{k|z}$'s alone as the function is linear in the control policy and is therefore convex.

The next theorem proves the convexity of the augmented cost functional. This convexity allows the use of simple gradient descent to find the globally optimal solution for the cost functional.

Theorem A.2 *For constant p_z, $\mathbf{r}_z$, and θ_k where p_z is a valid pmf and $\mathbf{r}_z, \theta_k \in \mathbb{R}^n$ the auxiliary function for the linear programming problem $F(\mathbf{x}) = \alpha \mathbf{c}(\theta)^T \mathbf{x} + p(\mathbf{x})$ is convex in $\mathbf{x}$ where $\mathbf{x} = (Q_{1|1}, Q_{1|2}, \cdots, Q_{k|z})^T$, $p(\mathbf{x}) = -\sum_{k,z} \log Q_{k|z}$, and $\mathbf{c}(\theta) = (p_1||\theta_1 - \mathbf{r}_1||^2, \cdots, p_z||\theta_k - \mathbf{r}_z||^2)^T$ is the vector of cost coefficients.*

Proof: If the Hessian of $F(\mathbf{x})$ is positive semidefinite, then $F(\mathbf{x})$ is convex in $\mathbf{x}$. The gradient of $F(\mathbf{x})$ is

$$\nabla F(\mathbf{x}) = \alpha \mathbf{c}(\theta) - \mathbf{x}^{-1}$$

where $\mathbf{x}^{-1} = (1/Q_{1|1}, 1/Q_{1|2}, \cdots, 1/Q_{k|z}, \cdots)^T$. The Hessian of this function is

$$\nabla^2 F(\mathbf{x}) = \mathrm{diag}(\mathbf{x}^{-2})$$

where $\mathbf{x}^{-2} = (1/Q_{1|1}^2, 1/Q_{1|2}^2, \cdots, 1/Q_{k|z}^2, \cdots)^T$. Since the $Q_{k|z}$'s are all probabilities, they are non-negative. Thus, the Hessian is a diagonal matrix with non-negative eigenvalues and is positive semidefinite. Therefore, the function $F(\mathbf{x}) = \alpha \mathbf{c}(\theta)^T \mathbf{x} + p(\mathbf{x})$ is convex in $\mathbf{x}$. □

The Newton-Raphson update for the minimization of the cost functional with respect to θ_k is the subject of the following theorem. It shows how the NR step produces the update formula that appears in (7).

Theorem A.3 *Let $\overline{D}(p, Q_{k|z}, \theta_k) = \sum_{k,z} p_z Q_{k|z} ||\theta_k - r_z||^2$ and let p_z and $Q_{k|z}$ be held constant. If a Newton-Raphson step is used to minimize $\overline{D}(p, Q_{k|z}, \theta_k)$ with respect to θ_k, then the minimizing set of θ_k's is*

$$\theta_k^{(m+1)} = \frac{\sum_z p_z Q_{k|z} r_z}{\sum_z p_z Q_{k|z}}.$$

Proof: The form of a Newton-Raphson step that finds a θ_k which minimizes $\overline{D}(p, Q_{k|z}, \theta_k)$ is

$$\theta^{(m+1)} = \theta^{(m)} - [\nabla^2 \overline{D}]^{-1} \nabla \overline{D}$$

where θ is the vector of θ_k's, $\nabla \overline{D}$ is the gradient and $\nabla^2 \overline{D}$ is the Hessian of $\overline{D}$. The gradient and Hessian are found in (9) and (10), respectively. Substituting the appropriate values for a particular θ_k yields

$$\theta_k^{(m+1)} = \theta_k^{(m)} - \frac{2 \sum_z p_z Q_{k|z} (\theta_k^{(m)} - r_z)}{2 \sum_z p_z Q_{k|z}}.$$

Breaking apart the summation produces

$$\theta_k^{(m+1)} = \theta_k^{(m)} - \frac{\sum_z p_z Q_{k|z} \theta_k^{(m)}}{\sum_z p_z Q_{k|z}} + \frac{\sum_z p_z Q_{k|z} r_z}{\sum_z p_z Q_{k|z}},$$

and simplifying this result produces

$$\theta_k^{(m+1)} = \frac{\sum_z p_z Q_{k|z} r_z}{\sum_z p_z Q_{k|z}},$$

and the theorem is proved. □.

The next set of results detail the various relationships concerning the duality gap. The first theorem is fairly straightforward and defines the duality gap for a standard primal-dual linear programming problem.

Theorem A.4 *For the primal LP problem*

$$\begin{array}{ll} \textit{Minimize} & \mathbf{c}(\theta)^T\mathbf{x} \\ \textit{Subject to:} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0 \end{array}$$

and its associated dual problem

$$\begin{array}{ll} \textit{Maximize} & \mathbf{b}^T\mathbf{u} \\ \textit{Subject to:} & A^T\mathbf{u} + \mathbf{z} = \mathbf{c}(\theta) \\ & \mathbf{z} \geq 0 \end{array}$$

the duality gap is $\Delta = \mathbf{z}^T\mathbf{x}$.

Proof: The duality gap is defined as $\Delta = \mathbf{c}(\theta)^T\mathbf{x} - \mathbf{b}^T\mathbf{u}$. From the dual problem, one sees that $\mathbf{c}(\theta) = A^T\mathbf{u} + \mathbf{z}$. Substituting this for $\mathbf{c}(\theta)$ in the duality gap expression yields $\Delta = \mathbf{z}^T\mathbf{x} + \mathbf{u}^T A\mathbf{x} - \mathbf{u}^T\mathbf{b}$. Simplifying this as $\Delta = \mathbf{z}^T\mathbf{x} + \mathbf{u}^T(A\mathbf{x} - \mathbf{b})$ and realizing that $A\mathbf{x} = \mathbf{b}$ (due to the constraints from the primal LP) results in

$$\Delta = \mathbf{z}^T\mathbf{x} + \mathbf{u}^T\mathbf{0} = \mathbf{z}^T\mathbf{x},$$

thus proving the theorem. □

The second result details conditions on the choice of feasible dual slacks ($\mathbf{z}$) for the dual problem, which are used in deriving the closed form solution for the duality gap.

Lemma A.1 (Lemma 3.1 in [8]) *In order for* $\mathbf{z}$ *to be a feasible dual slack,* $P_A\mathbf{z} = P_A\mathbf{c}(\theta_k)$*, where* P_A *is a transformation that projects a vector into the null space of* A.

Proof: From the dual problem, we know that $A^T\mathbf{u} + \mathbf{z} = \mathbf{c}(\theta_k)$. Rearranging produces the vector $\mathbf{c}(\theta_k) - \mathbf{z} = A^T\mathbf{u}$. The vector, $\mathbf{c}(\theta_k) - \mathbf{z}$ can be decomposed into components in $\mathcal{N}(A)$ and $\mathcal{R}(A^T)$.

$$\mathbf{c}(\theta_k) - \mathbf{z} = P_A(\mathbf{c}(\theta_k) - \mathbf{z}) + \tilde{P}_A(\mathbf{c}(\theta_k) - \mathbf{z})$$

But we know from $\mathbf{c}(\theta_k) - \mathbf{z} = A^T\mathbf{u}$ that $\mathbf{c}(\theta_k) - \mathbf{z}$ is already in the range space of A^T and therefore $\mathbf{c}(\theta_k) - \mathbf{z} = \tilde{P}_A(\mathbf{c}(\theta_k) - \mathbf{z})$. This requires that the projection into the null space is zero, $P_A(\mathbf{c}(\theta_k) - \mathbf{z}) = 0$, leading to the conclusion that $P_A\mathbf{c}(\theta_k) = P_A\mathbf{z}$. □

The next lemma uses the conditions of the previous lemma to derive the closed form expression for the duality gap which is used as the termination condition in the modified interior point algorithm.

Lemma A.2 (Lemma 5.1 in [8]) *Given an $\alpha > 0$ and $||\mathbf{h}(\mathbf{x}, \alpha)|| \leq \epsilon$, a choice for a feasible dual slack is*

$$\mathbf{z} = X^{-1}\frac{\mathbf{e} + \mathbf{h}(\mathbf{x}, \alpha)}{\alpha},$$

where $X = diag(\mathbf{x})$ is a scaling transformation and $\mathbf{h}(\mathbf{x}, \alpha) = P_{AX}X\nabla F(\mathbf{x})$ is the scaled gradient projected into the null space of AX. The duality gap is

$$\mathbf{z}^T\mathbf{x} = \frac{n + \mathbf{e}^T\mathbf{h}(\mathbf{x}, \alpha)}{\alpha} \leq \frac{n + \epsilon\sqrt{n}}{\alpha}. \tag{11}$$

Proof: We know from Lemma A.1 that the projection of the dual slacks must equal the projection of the cost vector for the slacks to be feasible. Let $\tilde{\mathbf{z}} = X\mathbf{z}$. Projecting $\tilde{\mathbf{z}}$ into the null space of AX yields

$$P_{AX}\tilde{\mathbf{z}} = (P_{AX}\mathbf{e} + P_{AX}\mathbf{h}(\mathbf{x}, \alpha))/\alpha.$$

But $\mathbf{h}(\mathbf{x}, \alpha)$ is already in the null space of AX, therefore $P_{AX}\mathbf{h}(\mathbf{x}, \alpha) = \mathbf{h}(\mathbf{x}, \alpha)$. $\mathbf{h}(\mathbf{x}, \alpha) = P_{AX}X\alpha\mathbf{c}(\theta_k) - P_{AX}\mathbf{e}$, since $\mathbf{h}(\mathbf{x}, \alpha)$ is the projected and scaled gradient. Substituting this back into the expression for $P_{AX}\tilde{\mathbf{z}}$ yields

$$P_{AX}\tilde{\mathbf{z}} = (P_{AX}\mathbf{e} + \alpha P_{AX}X\mathbf{c}(\theta_k) - P_{AX}\mathbf{e})/\alpha = P_{AX}X\mathbf{c}(\theta_k).$$

But $P_{AX}X\mathbf{c}(\theta_k)$ is the projected cost for the scaled problem. Therefore, by Lemma A.1, $\tilde{\mathbf{z}}$ is a feasible dual slack for the scaled problem. This implies that $\mathbf{z}$ is a feasible dual slack for the original problem.

Substitution of this value of $\mathbf{z}$ into the expression for the duality gap yields

$$\mathbf{z}^T\mathbf{x} = \mathbf{x}^T\mathbf{z} = \mathbf{x}^T X^{-1}(\mathbf{e} + \mathbf{h}(\mathbf{x}, \alpha))/\alpha$$

But $\mathbf{x}^T X^{-1} = \mathbf{e}$ and $\mathbf{e}^T\mathbf{e} = n$ producing

$$\mathbf{z}^T\mathbf{x} = (n + \mathbf{e}^T\mathbf{h}(\mathbf{x}, \alpha))/\alpha$$

Here $||\mathbf{e}^T\mathbf{h}(\mathbf{x}, \alpha)|| \leq ||\mathbf{e}|| \cdot ||\mathbf{h}(\mathbf{x}, \alpha)|| = \sqrt{n} \cdot \epsilon$. Thus,

$$\Delta \leq (n + \epsilon\sqrt{n})/\alpha.$$

□

References

[1] M. C. Moed and G. N. Saridis, "A Boltzmann machine for the organization of intelligent machines," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 5, pp. 1094–1102, 1990.

[2] G. N. Saridis and K. P. Valavanis, "Analytical design of intelligent machines," *Automatica*, vol. 24, no. 2, pp. 123–133, 1988.

[3] A. Dempster, N. Laird, and D.B.Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 39, pp. 1–38, 1977.

[4] R. A. Jacobs *et al.*, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, pp. 79–87, 1991.

[5] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," Tech. Rep. 9301, MIT Computational Cognitive Science, Apr. 1993.

[6] M. I. Jordan and L. Xu, "Convergence results for the EM approach to mixtures of experts architectures," Tech. Rep. 9303, MIT Computational Cognitive Science, Sept. 1993.

[7] W. Byrne, "Alternating minimization and Boltzmann machine learning," *IEEE Transactions on Neural Networks*, vol. 3, pp. 612–620, July 1992.

[8] C. C. Gonzaga, "Path-following methods for linear programming," *SIAM Review*, vol. 34, pp. 167–224, June 1992.

[9] R. A. Jacobs and M. I. Jordan, "A modular connectionist architecture for learning piecewise control strategies," in *Proceedings of the American Control Conference*, vol. 2, (Boston, Massachusetts), pp. 1597–1602, 1991.

[10] M. S. Bazaraa, H. D. Sherali, and C. Shetty, *Nonlinear Programming: Theory and Algorithms*. New York: John Wiley and Sons, Inc., 1993.

[11] R. E. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE Transactions on Information Theory*, vol. IT-18, pp. 460–473, July 1972.