This is to certify that the

dissertation entitled

Neural Networks for Nonlinear Programming

presented by

Chia-Yiu Maa

has been accepted towards fulfillment
of the requirements for

Ph.D. degree in Electrical Engineering

[illegible]
Major professor

Date 2/4/91

DATE DUE	DATE DUE	DATE DUE
FEB 0 8 [illegible]		
JUN 1 2 [illegible]		

NEURAL NETWORKS FOR NONLINEAR PROGRAMMING

By

Chia-Yiu Maa

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Electrical Engineering

1991

ABSTRACT

NEURAL NETWORKS FOR NONLINEAR PROGRAMMING

By

Chia-Yiu Maa

Artificial neural networks (ANNs) for optimization are analyzed from the viewpoint of optimization theory. A unifying optimization network theory for linear programming, quadratic programming, convex programming, and nonlinear programming is derived. A 2-phase optimization network is proposed which can obtain both the exact solution, in contrast to the approximate solution by Kennedy and Chua's networks, as well as the corresponding Lagrange multipliers associated with each constraint. The quality of the solutions obtained by the optimization ANNs is quantified through simulation.

The applicability of the optimization ANNs for solving real-world problems is demonstrated with examples of the economic power dispatching problem and the optimal power flow problem. It is shown that the mapping technique of the optimization ANNs is simple and that they are able to handle various kinds of constraint sets. Furthermore, it is demonstrated that the optimization ANNs attain a better objective function value.

Overall, this work lays a solid groundwork for optimization ANNs in both theoretical and practical aspects.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dr. M. Shanblatt, my major advisor for his guidance and encouragement throughout the years of my graduate studies.

I also want to thank Dr. D. Reinhard, Dr. T. Grotjohn, Dr. R. Enbody, and Dr. J. Gardiner for their kindness to be my committee and their valuable comments and suggestions on this work.

Finally, I wish to dedicate this dissertation to my parents for getting me started right and continued support throughout my studies, my wife Mei-Ling for her patience, understanding, and love, and my lovely daughter Melody.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Conventional digital computers are very good at executing well-formulated sequences of instructions represented by the stored program. There are some tasks, however, which are very cumbersome to solve by conventional digital computers. These include vision, speech, pattern recognition with distorted data, and information retrieval where only partial input information is given. These tasks, on the other hand, are accomplished and performed well by the human brain. The basic processing elements of the human brain are neurons, which are electrochemical devices with response times in the range of milliseconds. In the human brain there are approximately 10^{11} neurons and each of them may be connected to thousands of other neurons. It is not yet well understood what interconnection structure organizes the neurons, nor how this massively parallel interconnected system (a biological neural network) interacts, stores and retrieves memory, and manipulates our thoughts.

In contrast, artificial neural networks (ANNs) are machine models of the biological neural networks with the aim of achieving human-like performance. Recently there has been a resurgence in the field of ANNs due to new network topologies (feed-forward multilayer network, Hopfield feedback network) and algorithms (back propagation, stochastic neural network), implementation techniques (digital VLSI technology, analog VLSI technology, electro-optics technology), and various emerging applications.

ANNs do not always outperform conventional (sequential or parallel) computers. Rather they provide a different approach to attack certain problems which are not easily solvable using conventional computers. The key characteristics of ANNs are listed in Table 1.1 and contrasted with the corresponding characteristics of conventional computers.

Table 1.1. Characteristics of ANNs and conventional computers.

Characteristics	ANNs	Conventional Computers
Memory	Distributed; Associative	Localized; Specific
Fault-Tolerance	Inherent	Not Inherent
Pattern Recognition Ability	Fast	Slow
Classification	Excellent	Poor
Partial Information Retrieval	Excellent	Poor
Error Correction Ability	Excellent	Poor
Learning Ability	Excellent	Poor
Math & Algorithmic Ability	Poor	Excellent
Timing Scheme	Asynchronous	**
Execution Mode	Highly Parallel	**
Processing Element	Simple Unit	**
Connectivity	High	**

** These characteristics are system-dependent.

The feature possessed by ANNs which differs most from conventional computers is that ANNs store information in their structure rather than in specific locations. All the parameters (connection weights, external biases, thresholds of neurons, initial states of neurons) collectively determine the information stored in the network. As a result, if some of the interconnections are disconnected or some of the neurons fail, the function of the network is preserved qualitatively. This provides the inherent ability of fault tolerance and sometimes the ability to retrieve the full output data pattern with only partial input information. For pattern recognition, correlations of the input patterns and output pattern are stored in the network. With a distorted or noisy input

ANNs do not always outperform conventional (sequential or parallel) computers. Rather they provide a different approach to attack certain problems which are not easily solvable using conventional computers. The key characteristics of ANNs are listed in Table 1.1 and contrasted with the corresponding characteristics of conventional computers.

Table 1.1. Characteristics of ANNs and conventional computers.

Characteristics	ANNs	Conventional Computers
Memory	Distributed; Associative	Localized; Specific
Fault-Tolerance	Inherent	Not Inherent
Pattern Recognition Ability	Fast	Slow
Classification	Excellent	Poor
Partial Information Retrieval	Excellent	Poor
Error Correction Ability	Excellent	Poor
Learning Ability	Excellent	Poor
Math & Algorithmic Ability	Poor	Excellent
Timing Scheme	Asynchronous	**
Execution Mode	Highly Parallel	**
Processing Element	Simple Unit	**
Connectivity	High	**

** These characteristics are system-dependent.

The feature possessed by ANNs which differs most from conventional computers is that ANNs store information in their structure rather than in specific locations. All the parameters (connection weights, external biases, thresholds of neurons, initial states of neurons) collectively determine the information stored in the network. As a result, if some of the interconnections are disconnected or some of the neurons fail, the function of the network is preserved qualitatively. This provides the inherent ability of fault tolerance and sometimes the ability to retrieve the full output data pattern with only partial input information. For pattern recognition, correlations of the input patterns and output pattern are stored in the network. With a distorted or noisy input pattern applied, a well-trained ANN is able to map it to an output whose corresponding input pattern best matches the applied one. Training, also called learning or adaptation, is the process determining the connection weights, usually over time, to improve performance. Massive parallelism is another feature possessed by neural networks which is necessary for high performance computation for applications like speech and pattern recognition.

Because of the immaturity of ANN research, various kinds of network structures have been proposed and tested for different applications. There is no agreement on which network best fits a particular application. Neither is there a unified way to classify the existing ANNs. One way, for example, is classifying them by topological groupings of single-layered or multi-layered; feed-forward or feedback; fully connected, nearest-neighbor connected, or hexagonally connected. ANNs can also be divided into two categories depending on whether their usage is neuroscience-oriented or engineering-oriented. The former tries to model simple nerve systems of some animals and implement the functions of the model through software or hardware. Then the implementation is used as a paradigm to validate and predict the behavior of the original nerve system. The latter aims at mimicking the biological neurons and their networks with some adaptation and modification based upon the available analytical methods and available implementation technology in order to exploit the decision-making functions.

For the engineering-oriented ANNs, one strong area of application is the solving of constrained optimization problems. The ANN which is most widely used and cited for such an application is the Hopfield network [1-7]. This is an one-layered, fully connected, feedback network. A often-adopted procedure to solve a specific problem using the Hopfield network includes the following steps:

i) select a mapping (representation, transformation, or encoding) such that the outputs of the network correspond to the solution to the problem;

ii) choose a proper energy function, bounded from below, whose minimum corresponds to a feasible solution to the problem;

iii) derive network connection weights and bias inputs, which properly embed the objective function and constrains of the problem into the network; and,

iv) choose initial values for the neurons in such a way that the network converges to a stable state which is a feasible solution to the problem.

Currently, each of the above steps is based on *ad hoc* procedures [6-44]. A lot of effort, however, is being placed on formulating rule-based methodologies to obtain parameters, derive energy functions, and choose initial values [12-13,18-24,45-50,57-60]. For a specific problem, there are various choices at each step of the above procedure, but, except in a few cases [18-19,23-24,58-60], most of the work reported to date does not guarantee, or at least analytically guarantee, that the state to which the network converges is a feasible solution to the problem.

1.1 Problem Statement

ANNs, particularly the Hopfield network, have been used to solve optimization problems such as linear programming, nonlinear programming, and dynamic programming. The stability and convergence of the Hopfield network is ensured due to its gradient descent nature [6]. But a reasonably formulated network, like the one used for linear programming is not sufficient to guarantee convergence to a feasible solution of the original problem. In fact, as will be shown in this dissertation, the linear programming network by Hopfield will converge to a point which is in general not close enough to an optimal solution. For certain networks, some states of convergence (local minima) may even turn out to be infeasible with respect to the original problem. An example of this phenomenon is the traveling salesman problem as

ii) choose a proper energy function, bounded from below, whose minimum corresponds to a feasible solution to the problem;

iii) derive network connection weights and bias inputs, which properly embed the objective function and constrains of the problem into the network; and,

iv) choose initial values for the neurons in such a way that the network converges to a stable state which is a feasible solution to the problem.

Currently, each of the above steps is based on *ad hoc* procedures [6-44]. A lot of effort, however, is being placed on formulating rule-based methodologies to obtain parameters, derive energy functions, and choose initial values [12-13,18-24,45-50,57-60]. For a specific problem, there are various choices at each step of the above procedure, but, except in a few cases [18-19,23-24,58-60], most of the work reported to date does not guarantee, or at least analytically guarantee, that the state to which the network converges is a feasible solution to the problem.

1.1 Problem Statement

ANNs, particularly the Hopfield network, have been used to solve optimization problems such as linear programming, nonlinear programming, and dynamic programming. The stability and convergence of the Hopfield network is ensured due to its gradient descent nature [6]. But a reasonably formulated network, like the one used for linear programming is not sufficient to guarantee convergence to a feasible solution of the original problem. In fact, as will be shown in this dissertation, the linear programming network by Hopfield will converge to a point which is in general not close enough to an optimal solution. For certain networks, some states of convergence (local minima) may even turn out to be infeasible with respect to the original problem. An example of this phenomenon is the traveling salesman problem as solved by the Hopfield network, where some converged states set up a traveling schedule to visit some cities more than once and not to visit some other cities at all.

To address this problem, a more thorough theoretical knowledge of the ANN characteristics and more knowledge of the relationship between the ANNs and optimization theory is needed. As a result of this shallow understanding, trial-and-error approaches are currently adopted in choosing the parameters of the network for solving various optimization problems. And what's worse is the fact that the set of parameters resulting from the trail-and-error procedures are case-dependent. Consequently, the application variety of ANNs is limited, especially for applications requiring real-time response. An urgent need is a more thorough establishment of the theoretical analysis for ANNs from the viewpoint of optimization theory. A key question is whether there is a general type network suitable for particular classes of optimization applications. If a general network is possible, then a proper procedure for mapping classes of optimization problems into this general network is desirable. Additionally, guidelines are needed to verify whether a given network will converge to the desired optimal solution(s) or not. If an exact solution is not achievable due to the finiteness of the network parameters, a network which will converge to an approximate solution is sought.

For fully exploring the computational power of ANNs in optimization problems, an investigation involving the following fundamental research tasks is to be done in this dissertation.

(1) Analysis of ANNs from the viewpoint of optimization theory.

(2) Developing generalized network(s) for solving basic optimization problems such as linear programming, quadratic programming, and nonlinear programming.

(3) Quantifying, through simulation, the potential advantages and disadvantages of the generalized network(s).

(4) Demonstrating the applicability of the generalized network(s) for solving some optimization problems requiring real-time response.

The problems to be demonstrated are the economic power dispatching problem and the optimal power flow problem.

1.2 Approach

The following plan is organized to achieve the goals of this research in a step-wise and overlapping fashion and to set the stage for subsequent developmental research further exploiting the anticipated results.

Task 1:	*Network Analysis*
Objective:	Various network formulations of the Hopfield model will be analyzed from the viewpoint of optimization theory in order to identify the reasons why and when the network succeeds or fails.
Approach:	The first step in this research is to analyze optimization formulations of the Hopfield model. Various network formulations reported in recent literature are classified into different categories to be analyzed in a systematic way from the viewpoint of optimization theory. The primary theories used in the analysis include the Kuhn-Tucker optimality conditions for constrained optimization and the penalty function methods which translate constrained optimization problems to unconstrained problems. Network formulations are translated to forms which can then be analyzed for optimality conditions and the other criteria mentioned above. Results of the analysis are compared with reported experiments and used to verify the adequateness of the network formulations in the

literature. Based on the analysis results, guidelines for checking the propriety of various network formulations are also sought.

Task 2: *Optimization Network Formulation*

Objective: Formulate basic optimization networks and their extensions, and develop, if possible, a generalized network structure.

Approach: With the analysis results obtained in Task 1, some optimization networks can be formulated in such a way that the stability, convergence, and optimality of convergence of these networks are theoretically assured. The network formulation is started on some basic problems like linear programming, quadratic programming, and nonlinear programming with affine (linear) constraints. Optimization networks for more complicated optimization problems, such as nonlinear programming with nonlinear constraints and other combinatorial optimization problems, are examined next as possible extensions of those basic networks. All the developed networks are checked with the previously derived guidelines to see whether they are legitimate. Based on the experience of formulating various networks, a generalized network is thus developed and a procedure for mapping classes of optimization problems into this network is sought.

Task 3: *Network Simulation*

Objective: Produce simulations of the optimization networks developed in the last task and provide some benchmark comparison parameters.

Approach: Simulation programs for each of the optimization networks are developed. For each optimization network there is a set of first order differential equations which can be solved by standard numerical analysis techniques. Simulation programs are used to provide performance metrics for comparison with other approaches. Metrics to be extracted include speed of convergence (throughput) and a measure of computation accuracy (quality). The simulation is also used to study the network sensitivity with respect to certain parameters, for example, the input resistance and capacitance of each neuron and the time increment used solved the differential equations. Based upon the simulation results, possible modification and re-formulation of the networks are taken by going back to Task 2 for the purpose of performance improvement.

Task 4: *Case Study*

Objective: Apply the developments in new optimization networks to real engineering optimization problems: economic power dispatching (EPD) and optimal power flow (OPF).

Approach: For the EPD problem, two cases, with and without the consideration of the transmission line losses, are solved by using the developed network formulations and their results are compared with the results obtained by other traditional methods. The formulation of the OPF is modified by delineating the various constraint types for a complete OPF so that this reduced model can be handled by the developed networks. Simulations of the OPF will be made thereafter and the results will be compared to benchmark OPF formulations.

1.3 Overview of the Dissertation

Chapter 2 covers the background knowledge pertaining to neural networks. It includes a biological review of neural networks, some simplified neuron models and network topologies, and a literature review of optimization-related work done on neural networks. Because of its pioneering role in applying ANN to optimization problem, the Hopfield feedback network is covered in Chapter 2 to briefly introduce the basic idea behind optimization neural networks.

Chapter 3 starts out with an introduction of the requisite mathematical background, followed by the analyses of three existing optimization neural networks. It ends with an in-depth study of the linear programming problem with hypercube feasible region as solving by the Kennedy and Chua's network [19].

The core of this dissertation is in Chapter 4 in which the theoretical results for various optimization networks are derived. The derivation of the linear programming network is given first. Then, the results are extended to the quadratic programming network and finally the nonlinear programming network. The least squares problem is shown to be solvable by the quadratic programming network. A two-phase network described last is capable of converging to the exact solution of a optimization problem and obtaining the corresponding Lagrange multipliers as well.

Chapter 5 gives the simulation results of various optimization problems using the developed network structures. Chapter 6 illustrates the applicability of the optimization network to real-world problems by two case studies: economic power dispatch and optimal power flow. The conclusion of this work is given in Chapter 7.

CHAPTER II

BACKGROUND

This chapter starts with a brief foundational review of neurobiology followed by some simplified models of neurons and different network interconnection topologies. A literature review of ANN applications to optimization problems is given next. The Hopfield feedback network is covered independently in the last section because of its pioneering role in applying ANN techniques to optimization problems.

2.1 Neurobiological Review

The major parts of a typical biological neuron include a *nucleus, cell body, axon hillock, axon, synapses*, and *dendrites* as shown in Figure 2.1. *Dendrites* are the receivers of incoming signal. When the incoming signal (a smooth varying analog voltage) reaches a certain value, the nerve cell fires and the *axon hillock* generates a pulse (action potential). The output of a typical neuron consists of a series of action potentials each about 1 millisecond long. If a neuron has a strong input, action potentials are generated at a high rate. If the input is weak or absent, action potentials are produced at a low rate. The mean rate of the generation of action potentials as a function of the input follows the form of a sigmoid function. The effective input usually refers to the short time average or running integral of an excitation; its frequency of generating action potentials is considered as the effective output [61].

The *axon* is very resistive and is in charge of transmitting action potentials to *synapses* through which a neuron interacts with up to thousands of other neurons. A synapse consists of two parts, the *presynaptic membrane* and the *postsynaptic membrane*. They are separated by a gap called the *synaptic cleft* which is about 500Å wide. Inside the presynaptic membrane there are small and diffusible molecules called *neurotransmitters* that are released into the cleft in response to an action potential. The released neurotransmitters diffuse to the postsynaptic membrane where they combine with certain receptor molecules causing a *depolarization* of the postsynaptic membrane. The depolarization signal is collected by the dendrite of the second neuron and sent to its cell body [62]. The response of the postsynaptic membrane is a graded response rather than a pulse. With an excitatory synapse the postsynaptic potential will be more positive, whereas with an inhibitory synapse the postsynaptic potential will be more negative.

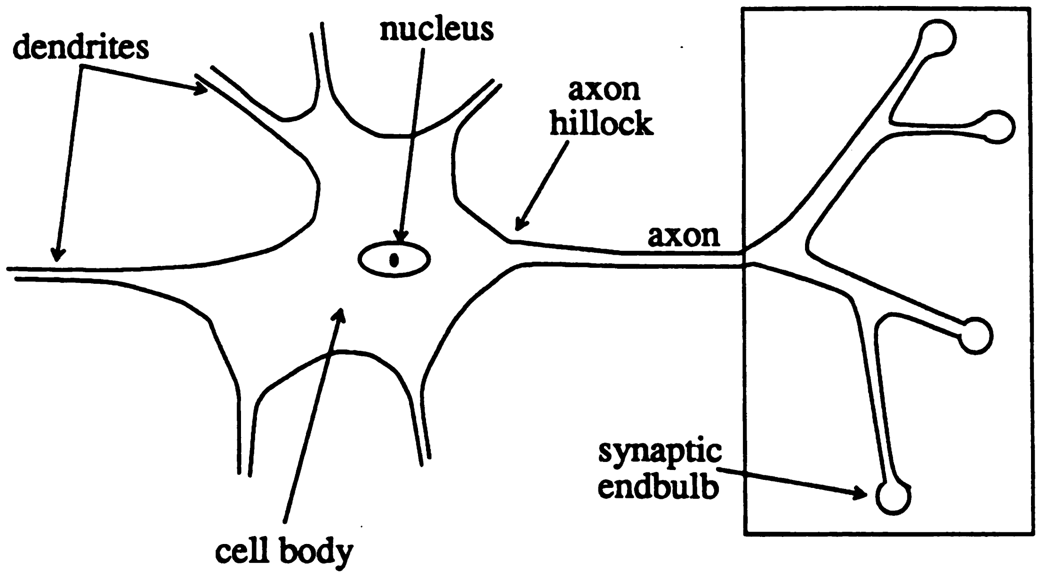


Figure 2.1. A biological neuron.

2.2 Simplified Neuron Models and Network Topologies

Neuron models constructed for the purpose of studying the function of the brain generally involve very complicated mechanisms and thus result in complex structures. These models fall into the category of neuroscience-oriented ANNs and are of primary interest to neurobiologists. On the other hand, engineering-oriented neuron models which are used for the purpose of improving the artificial (machine) processing of data or information are of primary interest in this work.

The simplest neuron model sums the weighted inputs and passes the result through a certain function. The outcome of the function, considered as the output of the neuron, branches out via weighted connections to the inputs of other neurons. Let α_i's be the inputs from other neurons, ω_{ij} be the weight of the connection from the output of neuron i to the input of neuron j, $f(\bullet)$ be the neural function, and y_j be the output of the neuron j. As mentioned in the previous section, the firing of action potentials of a neuron is a type of threshold mechanism. To model this mechanism requires two more variables, θ_j and ω_{0j}, denoting the threshold value and its weight for neuron j, respectively. This simplified representation of a neuron is shown in Figure 2.2.

From the computation viewpoint, this neuron model can be thought of as a processing element (PE). The function $f(\bullet)$ maps input values to a prespecified range and is generally of the following four types: linear, nonlinear ramp, step, and sigmoidal. These input/output relationships are shown in Figure 2.3. The sigmoidal function is the most pervasive because it is bounded, monotonic, nondecreasing and provides a graded, nonlinear response which most resembles a real neuron. The range of the sigmoidal function is sometimes changed from [0,1] to [-1,1], depending on the application, giving symmetry with respect to the origin.

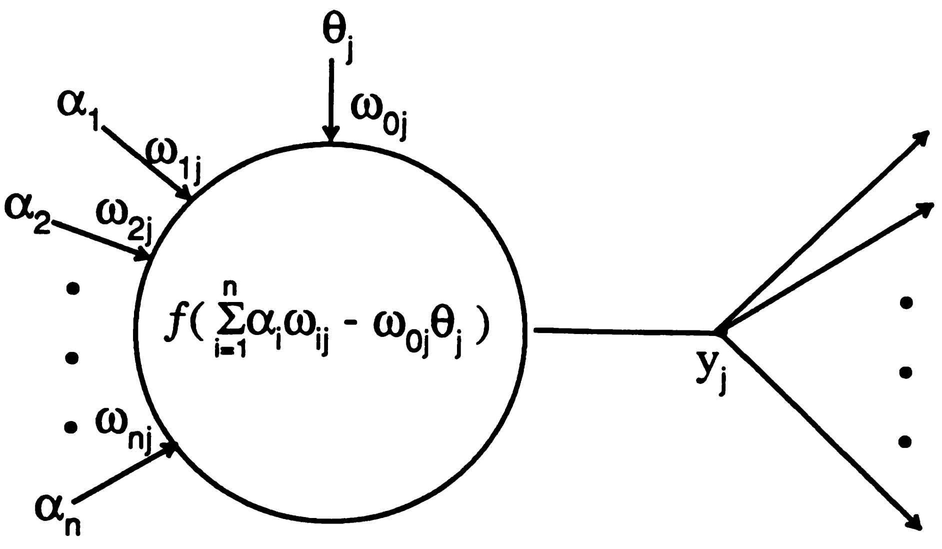
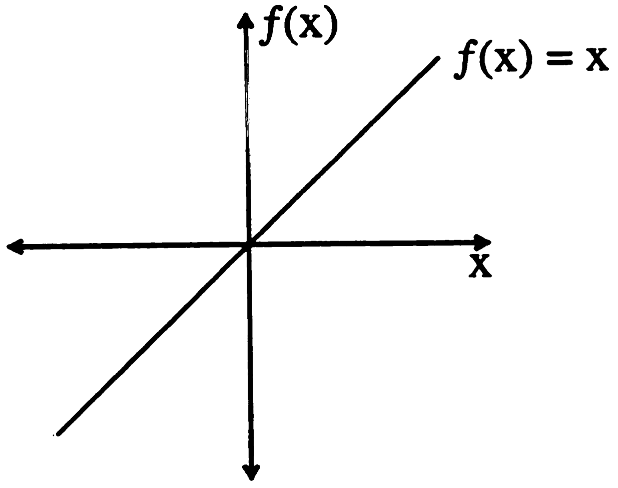


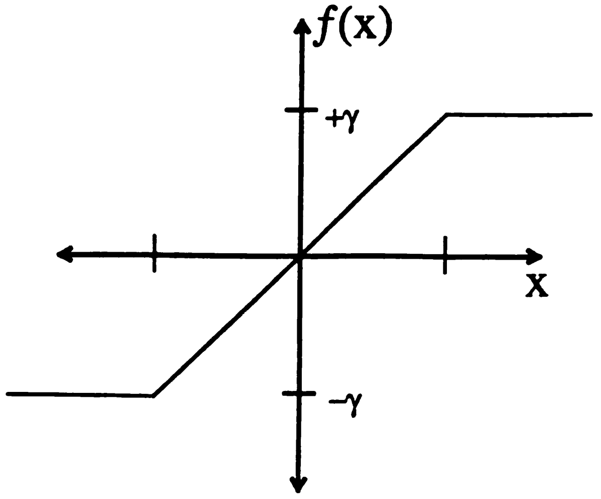
Figure 2.2. Simplified representation of a neuron.



(a)

Figure 2.3. Neuron response functions. (a) Linear function. (b) Nonlinear ramp function. (c) Step function. (d) Sigmoidal function.

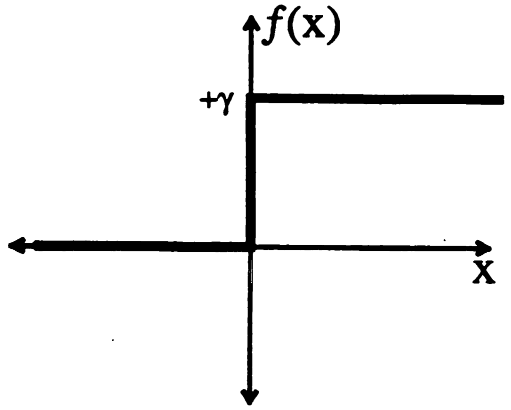
Figure 2.3. (cont'd.).



$$f(x) = \begin{cases} +\gamma & \text{if } x \geq \gamma \\ x & \text{if } |x| < \gamma \\ -\gamma & \text{if } x \leq -\gamma \end{cases}$$

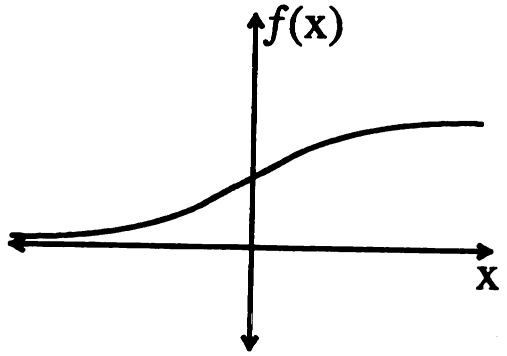
- γ is the saturation level

(b)



$$f(x) = \begin{cases} +\gamma & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

(c)

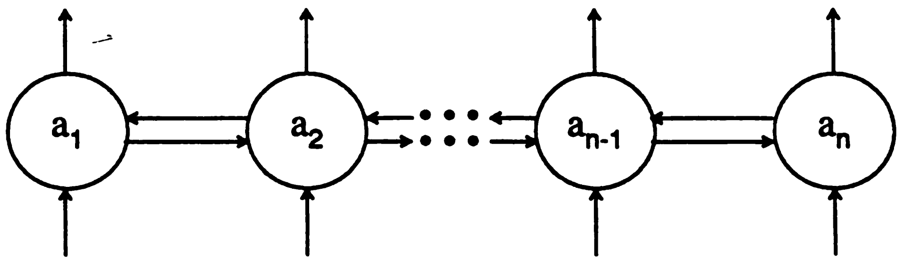


$$f(x) = \frac{1}{1 + e^{-x}}$$

(d)

The connection weight in the model corresponds to the correlating strength between the presynaptic potential and the postsynaptic potential. A positive weight models a excitatory synapse and a negative weight models a inhibitory synapse. A positive connection weight implies a positive correlation between the connected neurons or a rewarding relationship. A negative weight implies negative correlation or a punishing relationship.

The interconnection structure of neurons in human brain is very complicated but is thought not to be random. There is evidence showing that both the retina and cortex are organized into layers of cells with interconnections within and between layers. Connections within a layer are referred to as intra-field connections, lateral connections, or short-term memory (STM). Connections between layers are referred to as inter-field connections, field connections, or long-term memory (LTM). The intra-field connections are usually considered unidirectional while inter-field connections may propagate signals in a feed-forward and/or feedback direction. Three interconnection topologies are given in Figure 2.4. The networks shown in Figures 2.4 (b) and (c) can be extended to n layers.



(a)

Figure 2.4. Interconnection topologies. (a) One-layer laterally connected neural network. (b) Two-layer feed-forward neural network. (c) Two-layer feedback neural network.

As an example of the functional difference between these interconnections, consider the XOR problem [63]. This problem is not solvable using a single layer neural network as was discovered early on in the ANN research. It can now be solved easily using a multilayered trained network. This training, however, is nontrivial and it is still an open question on how to best train a complicated ANN. The connection patterns within and between the layers are not necessarily fully connected as is the case illustrated. These connections may be nearest-neighbor type connections, connected according to certain patterns, or randomly connected with fixed fan-in and fan-out. Other connection topologies, such as mesh, feature map, and three-dimensional arrays, have also been studied [47-49,61,64]. Though not shown in Figure 2.4, every connection is weighted.

2.3 Literature Review

According to the chronologically edited book by Anderson and Rosenfeld, which contains over 40 important historical papers in this field, the idea of modern neural networks can be traced back to as early as late 19th century [65]. Recent interest has been sparked mainly due to the works of Hopfield [4-7], Grossberg [66-69], Kohonen [61,70], McClelland [71-74], and Rumelhart [71-76]. An in-depth study of neural network research and applications up to 1987 has been done by DARPA [77]. In this study, the storage of a neural network is measured in terms of *interconnects*, and the speed of a network is described in terms of *interconnects-per-second*. Also in this study, interconnects versus interconnects-per-second charts have been used the first time to measure the computational capabilities of neural networks.

Tank and Hopfield first applied an ANN-based technique to solve optimization problems like the traveling salesman problem and linear programming [6-7]. In the case of linear programming, the objective function and inequality constraints are mapped into a closed-loop network in such a way that constraint violations loop back to adjust the states of the neurons. The overall energy functions of a network so designed decreases until it reaches a minimum. Under a high-gain limit assumption often placed on certain neurons, the corresponding output of the network then presumably approaches a solution to the original problem. However, this presumption is not true in general as will be seen in the next chapter.

Motivated by the work of Tank and Hopfield, an abundance of research has been done on applying the ANNs to other optimization problems. Some work has been done on the justification of the ANN model used for the traveling salesman problem and possible model modifications as well as extensions [8-13,78-79]. There is increasing interest in applying ANNs to various linear programming problems, such as integer linear programming and problems with equality constraints and to related topics such as nonlinear programming and dynamic programming [14-24,80-82].

For engineering-related optimization applications, ANNs have been developed to solve the placement and the routing problems in VLSI design [86-93], general computer-aided design [94-99], and power systems engineering problems like security monitoring, contingency classification, and economic power dispatching [25-26,100-109]. More general optimization-related applications using ANN-based techniques can be found in [27-46,110-117]. Most of the works cited, however, have not yet been vindicated theoretically, and this limits their applicability.

Some researchers have also sought to combine both the ANN models and another new optimization technique, namely, simulated annealing [118] in order to explore the merits of the two [18-24,51-54,119-122]. But the drawbacks are that the amount of time needed for the annealing process is too long plus there is no simple way to implement such models using any currently available technology.

Analysis is being carried out on some of the ANN models used for optimization-related applications [55-60,123-132]. But, except for a few papers reported to date [18-20,23-24], most of these analyses were not undertaken from the viewpoint of optimization theory. As a result, the network so designed may converge to a point which is useless to the original problem. Though the ANN model used in [20] was developed based on some optimization methods, they merely replaced the discrete procedures by a set of differential equations, which do not necessarily guarantee the stability nor the convergence of the network. We will justify our argument in the next chapter when the models in [6,19,24] are carefully studied under a unifying optimization ANN theory developed in the process of this work.

2.4 The Hopfield Model

The Hopfield model falls into the category of a one-layer, laterally connected network; it is sometimes referred to as the feedback network [133]. One of the major contributions of the Hopfield model is that it can be built with analog circuit components and is suitable for analog VLSI implementation [134-142]. In this model each neuron, with input u_i and output V_i, is modeled as an amplifier with a capacitive element C_i and a resistive element ρ_i at the input node. These components partially define the time constant of the neuron. The output of neuron j is connected to the input of neuron i via a finite conductance T_{ij}. This conductance models the synapse and is symmetric, i.e., $T_{ij} = T_{ji}$. Figure 2.5 illustrates the basic structure of a Hopfield neuron. The input-output relationship of the amplifier is sigmoidal. The excitatory synapse (T_{ij}>0) and the inhibitory synapse (T_{ij}<0) are implemented by connecting the conductance to the normal output and inverted output of amplifier j, respectively.

A general Hopfield network is shown in Figure 2.6. The rate of change of u_i is determined by the following equation derived using KCL.

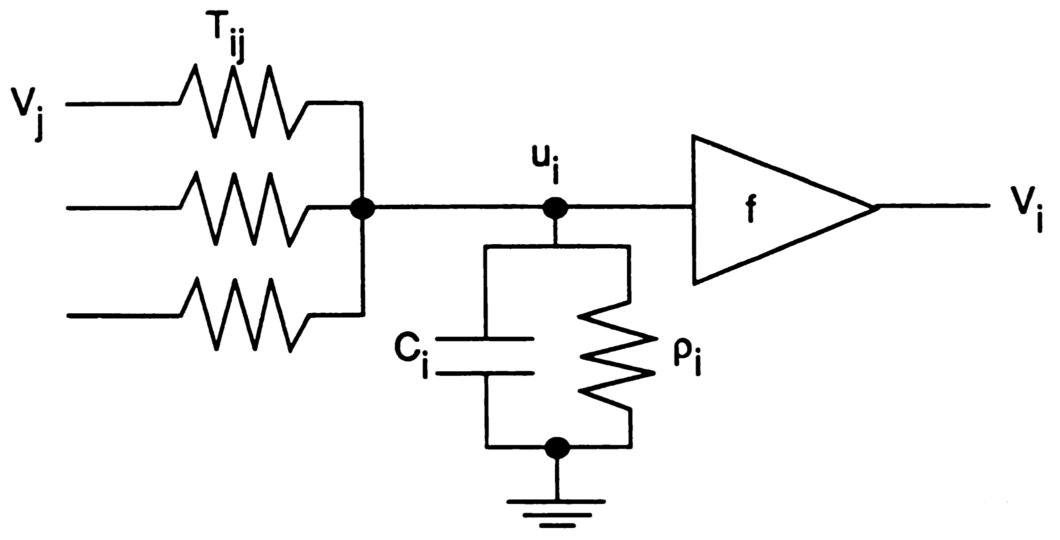


Figure 2.5. A basic neuron of the Hopfield model.

$$C_i(\frac{du_i}{dt}) = \sum_{j=1}^{n} T_{ij}(V_j - u_i) - \frac{u_i}{\rho_i} + I_i$$

$$= \sum_{j=1}^{n} T_{ij}V_j - (\sum_{j=1}^{n} T_{ij} + \frac{1}{\rho_i})u_i + I_i, \tag{2.1}$$

where

$$V_j = f_j(u_j). \tag{2.2}$$

Let $R_{ij} = \frac{1}{T_{ij}}$, $\frac{1}{R_i} = \frac{1}{\rho_i} + \sum_{j}^{n} \frac{1}{R_{ij}}$, and choose $f_j(\bullet) = f(\bullet)$ for all j. Then the above equations can be rewritten as

$$C_i(\frac{du_i}{dt}) = \sum_{j=1}^{n} T_{ij}V_j - \frac{u_i}{R_i} + I_i \tag{2.3}$$

where

$$V_j = f(u_j), \tag{2.4}$$

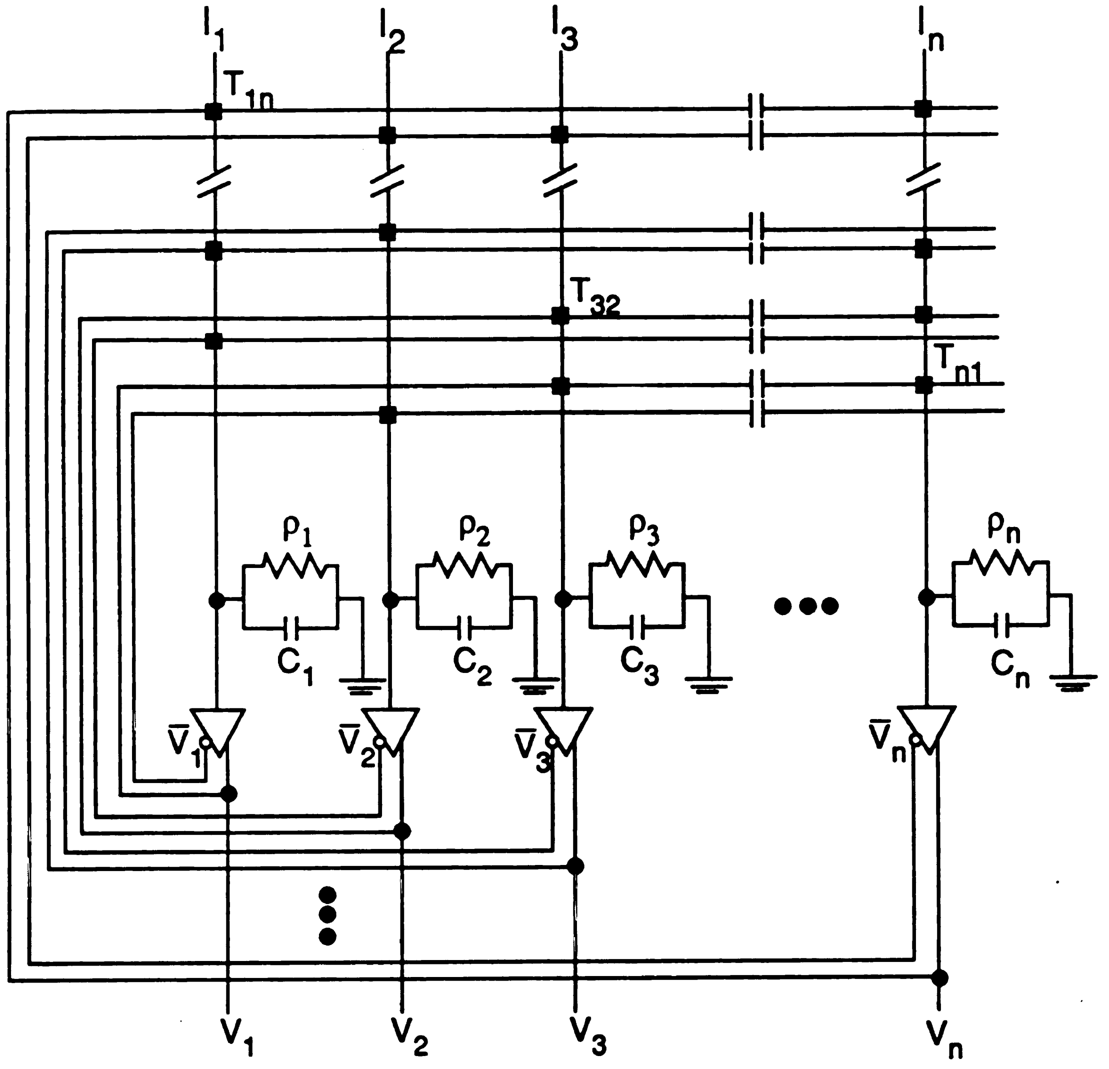


Figure 2.6. A general Hopfield network.

and

$$u_j = f^{-1}(V_j). \tag{2.5}$$

The first integral or the energy function of equation 2.3 is

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} T_{ij} V_i V_j - \sum_{i=1}^{n} I_i V_i + \sum_{i=1}^{n} \frac{1}{R_i} \int_0^{V_i} f^{-1}(\xi_i)\, d\xi_i \tag{2.6}$$

for $-\frac{\partial E}{\partial V_i} = C_i(\frac{du_i}{dt})$. The time derivative of the energy function can be found by applying the chain rule as

$$\begin{aligned}\frac{dE}{dt} &= \sum_{i=1}^{n} \frac{\partial E}{\partial V_i}\frac{dV_i}{dt} \\ &= \sum_{i=1}^{n} \frac{\partial E}{\partial V_i}\frac{dV_i}{du_i}\frac{du_i}{dt} \\ &= \sum_{i=1}^{n} \frac{\partial E}{\partial V_i}\frac{df(u_i)}{du_i}(-\frac{1}{C_i}\frac{\partial E}{\partial V_i}) \\ &= -\sum_{i=1}^{n} \frac{1}{C_i}\frac{df(u_i)}{du_i}\left[\frac{\partial E}{\partial V_i}\right]^2 . \end{aligned} \tag{2.7}$$

Since $f(u_i)$ is monotonically increasing, $\frac{dE}{dt} \le 0$ for all t. As a result, the value of the energy function is strictly decreasing and becomes zero only at equilibrium points at which $\frac{\partial E}{\partial V_i} = -C_i\frac{du_i}{dt} = 0$ for all i.

If u_j in equation 2.4 is replaced by λu_j, where λ is a constant representing the neuron gain, then equation 2.5 becomes

$$u_i = \frac{1}{\lambda} f^{-1}(V_j). \tag{2.8}$$

Hopfield asserted that if λ is chosen to be large enough, then the third term on the right hand side of equation 2.6 is negligible compared to the other terms and thus can be dropped [5]. This leads to the following:

$$C_i(\frac{du_i}{dt}) = \sum_{j=1}^{n} T_{ij}V_j + I_i \tag{2.9}$$

and

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} T_{ij}V_iV_j - \sum_{i=1}^{n} I_iV_i . \tag{2.10}$$

Note that these two equations are valid only for the high gain limit assumption, that is, when λ is very large.

Equations 2.3 - 2.6 actually define a gradient system and thus guarantee no oscillations or any complicated behavior in the system [133]. Furthermore, it has been proven that such a system has only a finite number of isolated equilibria and they are bounded [143-144]. The network can thus be envisioned as a system which tends to find a path leading to a local minimum in the energy surface. This surface is collectively defined by the network parameters. Those isolated equilibrium points may correspond to memory patterns in associative memory, patterns in a pattern recognition problem, or locally optimal solutions to an optimization problem.

CHAPTER III

ANALYSIS OF NEURAL NETWORKS FOR OPTIMIZATION

The requisite mathematical background is given in the first section to introduce the notations and the basic theorems used in this dissertation. The ANN models described in [6,19,24] are studied in detail and justified by the theorems stated in Section 3.1. A thorough analysis of the linear programming neural network for problems with hypercube feasible region is covered in the last section. It serves to demonstrate the dynamics of the optimization network in [19].

3.1 Mathematical Background

The following notation and conventions are used throughout this dissertation. $X \subset R^n$ is said to be *convex* iff for any $a,b \in X$ implies $[a,b] \subset X$, where $[a,b]=\{x \in R^n \mid x=\lambda a+(1-\lambda)b,\ 0 \le \lambda \le 1\}$. Let $X \subset R^n$ be a nonempty convex set, then $f:X \rightarrow R$ is said to be *convex* iff $f(\lambda x+(1-\lambda)y) \le \lambda f(x)+(1-\lambda)f(y)$, for any $x,y \in X$ and for $0 \le \lambda \le 1$. The function $f:X \rightarrow R$ is *concave* if $-f$ is convex. An *affine* function $f:X \rightarrow R$ is a function which is convex and concave. K is a *cone* in R^n iff $\alpha x \in K$ for any $x \in K$ and for any $\alpha \ge 0$. K is a *convex cone* in R^d iff K is a cone and K is convex. Note that $x=\{Ay \mid y \ge 0\}$, where A is an $n \times m$ matrix, is a closed convex cone. H is a *hyperplane* in R^n iff there exists $a \in R^n$, $a \ne 0$, and $\alpha \in R$, such that $H = \{x \in R^d \mid \langle a, x \rangle = \alpha\}$, where $\langle \cdot,\cdot \rangle$ is the Euclidean inner product on R^n, and a is a normal vector of H. Hyperplanes H and H' are said to be *parallel* iff their normals are proportional. Let $H = H(a,\alpha) = \{x \in R^n \mid \langle a, x \rangle = \alpha\}$, and $a \neq 0$, then the corresponding closed half spaces are defined by $H_+(a,\alpha) = \{x \in R^n \mid \langle a, x \rangle \geq \alpha\}$, and $H_-(a,\alpha) = \{x \in R^n \mid \langle a, x \rangle \leq \alpha\}$. Note that the quadratic function

$$f(x) = \frac{1}{2} x^T A x + a^T x + b$$

is convex (strictly convex) on R^n iff A is positive semidefinite (positive definite), where A is a symmetric $n \times n$ matrix, $a \in R^n$, and $b \in R$.

Let the *program* $(P)^*$ be of the following form:

Minimize $f(x)$ subject to constraints

$$g_1(x) \leq 0, \ldots, g_r(x) \leq 0, \quad h_1(x) = 0, \ldots, h_m(x) = 0,$$

where f and the g_i's are functions on R^n and the h_j's are functions on R^n for $m \leq n$. (P) is said to be a *convex program* if f and the g_i's are convex functions on R^n and the h_j's are affine functions on R^n. A vector x is called a *feasible solution* to (P) iff x satisfies the $r+m$ constraints of (P). In other words, the feasibility set to (P) is the (possibly empty) set

$$K_o = K_1 \cap \ldots \cap K_r \cap L_1 \cap \ldots \cap L_m$$

where

$$K_i = \{x \mid g_i(x) \leq 0\}, \quad i = 1, \ldots, r,$$

and

---

* (P) and other capital letters, such as (LP), (QP), and (NP), when enclosed with parentheses represent mathematical programs.

$$L_j=\{x \mid h_j(x)=0\}, \quad j=1,\ldots,m.$$

When K_o is empty, (P) is said to be *infeasible*; otherwise, (P) is *feasible*. If (P) is a convex program, K_o is necessarily convex. For $x \in K_o$, the *binding set* at x is the set $I=\{i \mid g_i(x)=0\}$. Let $x \in K_o$; x is said to be a *regular point* if the gradients, $\nabla g_i(x)$, $\nabla h_j(x)$, $i \in I(x)$, $1 \le j \le m$, are linearly independent.

The following theorem is known as the Kuhn-Tucker optimality theorem. (For proof see [145].)

Theorem 3.1: Let (P) be a convex program in the notation above. Let $\bar{x}$ be a feasible solution to (P). Suppose each g_i and h_j is differentiable at $\bar{x}$. Assume further that $\bar{x}$ is a regular point. Then $\bar{x}$ is an optimal solution to (P) if and only if there exists $\lambda=[\lambda_1 \ldots \lambda_r]^T$ and $\mu=[\mu_1 \ldots \mu_m]^T$ together with $\bar{x}$ that satisfy:

(i) $\lambda_i \ge 0$, $g_i(\bar{x}) \le 0$, and $\lambda_i g_i(\bar{x})=0$, $i=1,\ldots,r$;

and

(ii) $\nabla f(\bar{x}) + \sum_{i=1}^{r} \lambda_i \nabla g_i(\bar{x}) + \sum_{j=1}^{m} \mu_j \nabla h_j(\bar{x}) = 0$. □

The variables λ_i and μ_j are known as *Lagrange multipliers*. Without the assumption of the convexity of the functions, the conditions (i) and (ii) are only the necessary conditions for $\bar{x}$ to be a local minimizer.

Theorem 3.2: Let (P) be a program in the notation above. Let $\bar{x}$ be a feasible solution to (P). Suppose each g_i is differentiable at $\bar{x}$. Assume further that $\bar{x}$ is a regular point. If $\bar{x}$ solves (P) locally, then there exists $\lambda=[\lambda_1 \ldots \lambda_r]^T$ and $\mu=[\mu_1 \ldots \mu_m]^T$ together with $\bar{x}$ that satisfy:

(i) $\lambda_i \ge 0$, $g_i(\bar{x}) \le 0$, and $\lambda_i g_i(\bar{x})=0$, $i=1,\ldots,r$;

and

(ii) $\nabla f(\bar{x}) + \sum_{i=1}^{r} \lambda_i \nabla g_i(\bar{x}) + \sum_{j=1}^{m} \mu_j \nabla h_j(\bar{x}) = 0.$ $\square$

For proof of above theorem see [146].

Define $\mu_{j^+} = \max\{\mu_j, 0\}$, and $\mu_{j^-} = \min\{\mu_j, 0\}$. Then, $\mu_j = \mu_{j^+} + \mu_{j^-}$ for $1 \le j \le m$. The corresponding term in condition (ii) above can be written as

$$\begin{aligned} \mu_j \nabla h_j &= (\mu_{j^+} + \mu_{j^-}) \nabla h_j \qquad (3.1) \\ &= \mu_{j^+} \nabla h_j - (-\mu_{j^-}) \nabla h_j \\ &= \mu_{j^+} \nabla h_j + (-\mu_{j^-})(-\nabla h_j) \\ &= \mu_{j^+} \nabla h_j + (-\mu_{j^-}) \nabla(-h_j). \end{aligned}$$

Since only one of μ_{j^+} or μ_{j^-} is nonzero and $h_j = 0$, condition (ii) can be extended by stipulating mutually exclusive terms for $h_j \le 0$ and $h_j \ge 0$.

$$\nabla f + \sum_{i=1}^{r} \lambda_i \nabla g_i + \sum_{j=1}^{m} \left[\mu_{j^+} \nabla h_j + (-\mu_{j^-}) \nabla(-h_j) \right] = 0. \qquad (3.2)$$

The Lagrange multipliers are now forced to be all non-negative. Let $g_{r+2j-1} = h_j$, $g_{r+2j} = -h_j$, $\lambda_{r+2j-1} = \mu_{j^+}$, and $\lambda_{r+2j} = -\mu_{j^-}$, the ***extended program*** (P') can now be expressed as:

Minimize $f(x)$ subject to $g_i(x) \le 0$, $1 \le i \le r+2m$.

Under these notations, Theorem 3.1 may be restated as the following corollary.

Corollary 3.1: Under the assumptions of Theorem 3.1 and the notations above, $\bar{x}$ is an optimal solution to (P') if and only if there exists $\lambda = [\lambda_1 \ldots \lambda_{r+2m}]^T$ together with $\bar{x}$ that satisfy

(i) $\lambda_i \ge 0$, $g_i(\bar{x}) \le 0$, and $\lambda_i g_i(\bar{x}) = 0$, $i = 1, \ldots, r$;

and

(ii) $\nabla f(\bar{x}) + \sum_{i=1}^{r+2m} \lambda_i \nabla g_i(\bar{x}) = 0.$ □

Next, define an *alternative binding set* to be $I'(x) = \{i \mid \lambda_i \neq 0,\ r+1 \le i \le r+2m\}$, then condition (ii) of Corollary 3.1 can be expressed as

$$\nabla f(\bar{x}) + \sum_{i \in I(\bar{x})} \lambda_i \nabla g_i(\bar{x}) + \sum_{i \in I'(\bar{x})} \lambda_i \nabla g_i(\bar{x}) = 0. \tag{3.3}$$

This implies that $-\nabla f(\bar{x})$ lies in the closed convex cone spanned by $\nabla g_i(\bar{x})$ for $i \in I(\bar{x}) \cup I'(\bar{x})$. If $\bar{x}$ is a regular point, then $-\nabla f(\bar{x})$ can be uniquely decomposed into a positive linear combination of ∇g_i, $i \in I(\bar{x}) \cup I'(\bar{x})$.

Define $g_i^+(x) = \max\{0, g_i(x)\}$, i.e., $g_i^+(x)$ is the magnitude of the violation of the $i$th constraint in (P') where $1 \le i \le r+2m$. The following theorem is known as the penalty function theorem. (For proof see [147].)

Theorem 3.3: Let (P') be the extended program stated above for $f \in C^1$ and $g_i \in C^1$ where $1 \le i \le r+2m$. Let $\{s_k\}_1^\infty$ be a nonnegative, strictly increasing sequence tending to infinity. Define the function

$$L(s, x) = f(x) + \frac{s}{2} \sum_{i=1}^{r+2m} (g_i^+(x))^2. \tag{3.4}$$

Let the minimizer of $L(s_k, x)$ be x_k. Then any limit point of the sequence $\{x_k\}_1^\infty$ is an optimal solution to (P') and, equivalently, to (P). Furthermore, if $x_k \to \bar{x}$ and $\bar{x}$ is a regular point, then $s_k g_i^+(x_k) \to \lambda_i$, which is the Lagrange multiplier associated with (P'). □

Note that the penalty function $L(s, x)$ is called the energy function in [6]. Later it will be shown that it is a qualified Lyapunov function for a neural network. The following corollary is a direct result from Theorem 3.3.

Corollary 3.2: Let the notations and assumptions be as defined in Theorem 3.3. Then given $\varepsilon > 0$, there exists a sufficient large s such that the minimizer of $L(s, x)$ lies in

$N(O, \varepsilon)$, where $N(O, \varepsilon) = \{x \mid \|x-\bar{x}\|<\varepsilon,\ \bar{x}\in O\}$ and O is the set of minimizers of (P'). □

3.2 Networks Analysis

There are currently three ANN models proposed for solving nonlinear programming problems [6,19,24]. Consider first only the case of linear programming. The linear program (LP) considered here is of the following form:

$$\text{Minimize } f(x) = a^T x$$

$$\text{subject to } g(x) = Dx - b \leq 0,$$

where D is an $m \times n$ matrix, $b \in R^m$, $a \in R^n$, and $x \in R^n$. If equality constraints are considered as well, then each of them can be replaced by two inequality constraints as shown in the last section so that the following discussion still holds. Note that $\nabla f(x) = a$ and $\nabla g(x) = D^T$. For simplicity in notation, denote $g^+ = [g_1^+ \dots g_m^+]^T$.

3.2.1 The Model by Tank and Hopfield

The network structure proposed by Tank and Hopfield for solving the linear programming problems is shown in Figure 3.1 [6]. a_i and b_i are implemented by current sources. The voltage outputs, x_i, on the upper right of the figure are the variables in the linear programming problem. The outputs, g_j^+, on the lower left of the figure, measure the constraint satisfaction (violation). s in the rectangles is a large positive constant.

Their model can be described in compact form by

$$\dot{x} = C^{-1}\left\{-\nabla f(x) - \nabla g(x) g^+(x) - \frac{1}{s} R^{-1} x\right\} s, \tag{3.5}$$

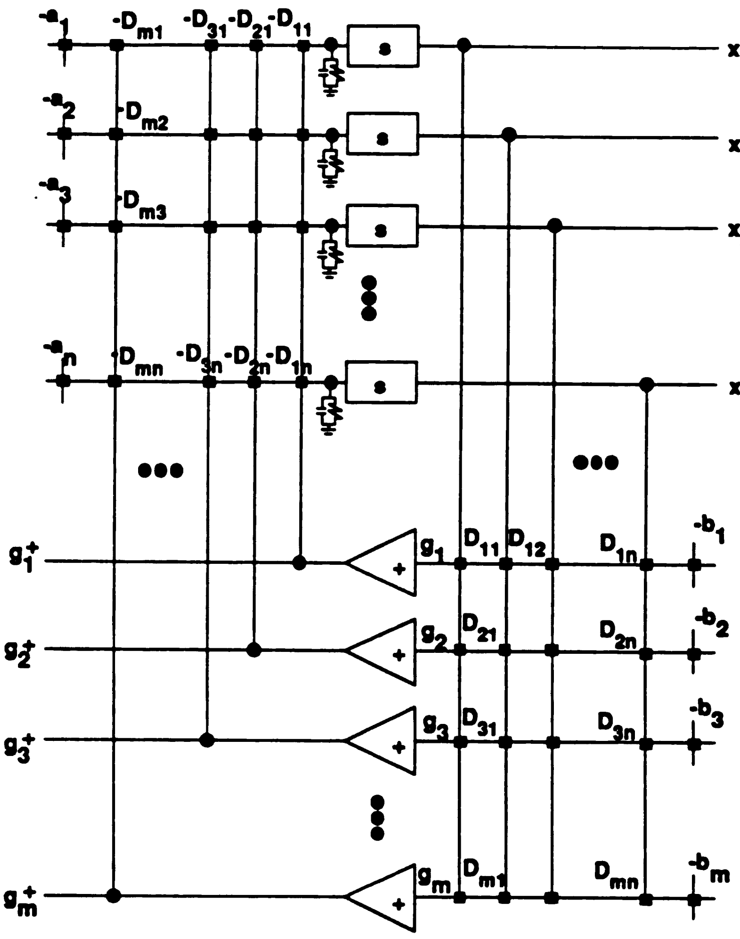


Figure 3.1. Linear programming network by Tank and Hopfield.

where C is a $n \times n$ diagonal matrix due to the self-capacitance of each amplifier. R is a $n \times n$ diagonal matrix with $\frac{1}{R_{ii}} = \sum_{j=1}^{m} -D_{ji} + \frac{1}{\rho_i}$ where $\frac{1}{\rho_i}$ the self-conductance of each amplifier. The so called energy function is chosen to be

$$E_1(x) = f(x) + \sum_{j=1}^{m} (g_j^+(x))^2 + \sum_{i=1}^{n} \frac{x_i^2}{2sR_i}. \tag{3.6}$$

Taking the derivative of $E_1(x)$ with respect to time yields

$$\frac{dE}{dt} = \sum_{i=1}^{n} \frac{\partial E}{\partial x_i} \frac{dx_i}{dt}$$

$$= (-C\dot{x})^T \dot{x}$$

$$= -\dot{x}^T C \dot{x} \le 0 \qquad (3.7)$$

for all t. Equality holds only at equilibrium points since C is a positive, diagonal matrix. Note that ∇f and ∇g are constant vectors for linear programming and $g^+(x)$ is just a vector. Replace $g^+(x)$ by a vector λ and take C to be the identity matrix, then the minimizer $\bar{x}$ of E_1 occurs when

$$-\nabla f(\bar{x}) - \nabla g(\bar{x})\lambda - \frac{1}{s}R^{-1}\bar{x} = 0. \qquad (3.8)$$

Comparing this equation with the condition (ii) of Theorem 3.2 shows that either the system described by equation 3.5 does not have an equilibrium or else, even if it does, the equilibrium would not be a solution to the program (LP).

Suppose s is sufficiently large, as suggested by Tank and Hopfield, the last terms of equation 3.5 and equation 3.8 can be neglected. In this case equation 3.8 can be viewed as fulfilling the necessary conditions of Theorem 3.2. But since λ_j $(=g_j^+)$ is required to be positive for some j, the equilibrium of equation 3.5 has to lie in the infeasible region of the program (LP). Depending on a particular program, the equilibrium may be quite far from the true minimizer of (LP) which is generally on a corner (or boundary) of the feasible region. This drawback makes their model unreliable in solving linear programming problems even though the model has a big advantage for hardware implementation.

3.2.2 The Model by Kennedy and Chua

The model developed by Kennedy and Chua [19] is based on the Chua's previous work in [148]. The basic components in their network are integrators as shown in Figure 3.2. Their network formulation requires more hardware components to form

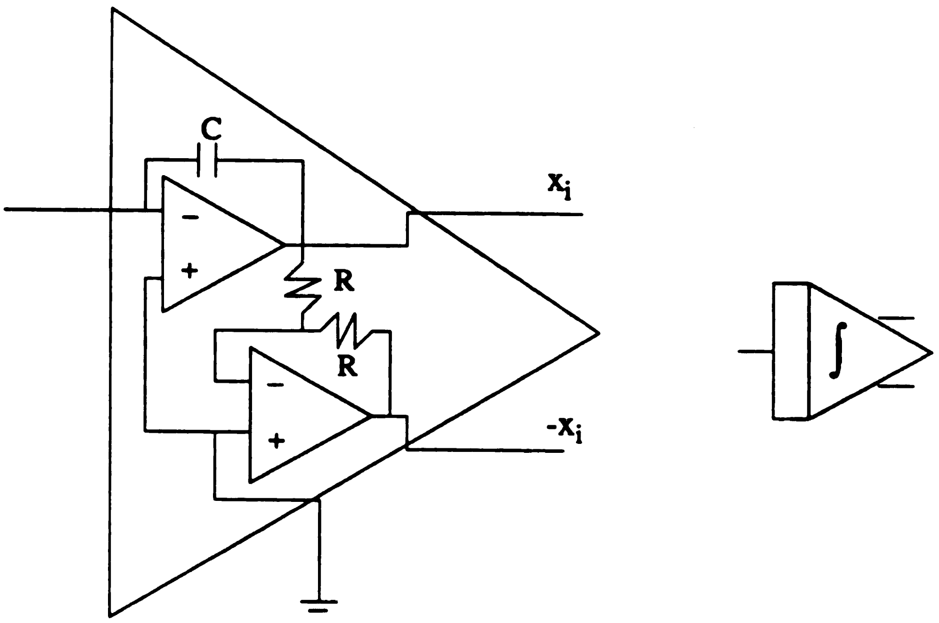


Figure 3.2 The integrator used in Kennedy and Chua's model.

the integrator in its analog circuit implementation when compared to the Tank and Hopfield network. But it is superior in that it circumvents the undesired terms in equations 3.5 and 3.6, namely, the terms due to self-conductance. Their model can be described by

$$\dot{x} = C^{-1}\left\{-\nabla f(x) - s\nabla g(x) g^{+}(x)\right\} \tag{3.9}$$

where C and s are defined as in equation 3.5. For argument sake, C is normally taken to be an identity matrix. This model has been used to solve both linear programming and quadratic programming problems. The corresponding energy function is

$$E_2(x) = f(x) + \frac{s}{2}\sum_{j=1}^{m}(g_j^+(x))^2. \qquad (3.10)$$

Kennedy and Chua showed that $E_2(x)$ is a Lyapunov function for the system of equation 3.9. This ensures that the system will converge to a stable equilibrium point without oscillation. Their network analysis is primarily based on the nonlinear circuit theory derived in [149].

Comparing equations 3.4 and 3.10 indicates that their work actually fulfills the penalty function method for a fixed penalty parameter. But they fail to justify the assumption required for the penalty function theorem (Theorem 3.3) to hold. Nor do they clarify the relations between the equilibrium point of the network and the true minimizer to the original program, since there could be more than one equilibrium point with respect to one minimizer unless the regularity of the minimizer is assumed. A straightforward analysis from the viewpoint of optimization theory is undertaken in the next chapter to establish a more sound theoretical foundation for Kennedy and Chua's network.

3.2.3 The Model by Rodriguez-Vazquez, et al.

The energy functions of the previous two networks are variations of penalty function methods, since they are formed by adding the cost functions with penalty terms. The penalty terms are derived by taking the magnitude of the constraint violation squared times a penalty parameter. According to the penalty function theorem, the true minimizer can only be obtained when the penalty parameter s is infinite. This is impossible to achieve in practice. To cope with this difficulty, Rodriguez-Vazquez, *et al.* proposed a network model which is formed by two mutually exclusive subsystems [24]. (Mutually exclusive here means only one of the two will contribute at a time.)

Let u_x be the feasibility index of x, i.e., u_x=1 if $x \in K_o$; otherwise, u_x=0. Their model can be expressed as

$$\dot{x} = -u_x \nabla f(x) - s \nabla g(x) g^+(x). \tag{3.11}$$

The corresponding energy function is

$$E_3(x) = u_x f(x) + \frac{s}{2} \sum_{j=1}^{m} (g_j^+(x))^2. \tag{3.12}$$

The trajectory of the system moves along $-\nabla f(x)$ if u_x=1; otherwise, it moves according to the negative gradients of the violated constraints. The combined effort of these two mutually exclusive subsystems forces the conglomerate trajectory to move toward the boundary of the feasible region. As long as it hits the boundary, the trajectory chatters around the boundary and, at the same time, it also approaches the optimum point. But a new problem arises: there is no equilibrium point in this system, since the condition (ii) of Theorem 3.2 can not be met. The trajectory bounces back and forth in a neighborhood of the minimizer, though the neighborhood can be made very small. The authors, however, have suggested that the system can be viewed as stable if the variation of the solution is bounded. They also suggested another model with the following system equation and energy function:

$$\dot{x} = -u_x \nabla f(x) - s \nabla g(x)(1-u_x). \tag{3.13}$$

and

$$E_3(x) = u_x f(x) + s \sum_{j=1}^{m} g_j^+(x). \tag{3.14}$$

But the same problem still exists for this model.

3.3 Linear Programming Network for Problems with Hypercube Feasible Region

The linear programming problem genre considered here is the minimization of a cost function

$$f(x) = a^T x \tag{3.15}$$

subject to $x \in [0,1]^n$, i.e., $0 \le x_i \le 1$ for every i, where x_i is the ith component of the n-vector x. Let 1_n and 0_n be n-vectors of all ones and zeros, respectively. Then the constraints can be represented in matrix form as

$$g(x) = \begin{bmatrix} -x \\ x - 1_n \end{bmatrix} = \begin{bmatrix} -I_n \\ I_n \end{bmatrix} x + \begin{bmatrix} 0_n \\ -1_n \end{bmatrix} = Dx + b \le 0. \tag{3.16}$$

$f(x)$ is a hyperplane in R^n and the feasible region is a unit hypercube in R^n. The general assumption is made on the system that $-a$ is not parallel to the normal vector of any hyperplane $g_j = 0$, where g_j is the jth component of $g(x)$. This ensures the uniqueness of the optimum point, or in this case the minimizer of the cost function. The minimizer of the cost function is one of the 2^n corners of the hypercube depending on the normal vector of the hyperplane.

Let $J = \{j \mid g_j > 0\}$ be the set of indices of the constraints which are violated. Let $k = card(J)$, the cardinality of J, i.e., the number of components of J. Then g_J^+ takes on the magnitudes of constraint violations. Consider next a network structure given by the piecewise linear differential equation

$$\dot{x} = -\left[D_J^T g_J^+ + a\right] = -\left[D_J^T (D_J x + b_J)^+ + a\right]. \tag{3.17}$$

This is similar to equation 3.5 without considering the term due to self-conductance. Note that this structure of the dynamical system described varies with time since J changes from time to time. D_J is a $k \times n$ matrix consisting of the jth rows of D, for every $j \in J$; similarly b_J consists of the jth components of b. Observe that since there are at most n constraint violations, D_J is at most an $n \times n$ matrix.

3.3.1 The equilibrium of the system

Under the assumption placed on $-a$, the minimizer of the cost function is at an intersection of n of $2n$ hyperplanes $g_j=0$. Recall that by the definition in Section 3.1, I is the set of indices of these hyperplanes. By the Kuhn-Tucker theorem for optimality, the necessary and sufficient condition for a point to be a minimizer is that

$$\nabla f + \sum_{j \in I} u_j \nabla g_j = 0 \tag{3.18}$$

where $u_j>0$. For any hypercube, the ∇g_j's are orthogonal for $j \in I$. Therefore, equation 3.18 can be viewed as decomposing ∇f into n orthogonal vectors ∇f_j's, each along the direction of $-\nabla g_j$.

Note that $\nabla f = a$ and $\nabla g_j = D_j^T$. Also, $(D_j x + b_j)^+$ measures the one-sided distance of x to $g_j=0$. If we denote $(D_j x + b_j)^+$ by w_j, then the equilibrium of equation 3.17 must satisfy

$$\nabla f + \sum_{j \in J} w_j \nabla g_j = 0. \tag{3.19}$$

Matching J with I and w_j with u_j shows that the equilibrium of the system fulfills the Kuhn-Tucker optimality condition. Furthermore the equilibrium is unique due to the orthogonality of $g_j=0$ for $j \in J$. Also observe that $w_j=(D_j x + b_j)^+>0$ for $j \in J$ indicating that the equilibrium lies in the infeasible region where n constraints are violated.

3.3.2 The initial state in the feasible region

When the initial state of the system is any point in the feasible region, then J is simply empty and the trajectory moves in the direction of $-a$ until it hits either one of the hyperplanes or an intersection of certain hyperplanes. Assume the former is the case. On the hyperplane $g_j=0$, the trajectory still follows the direction of $-a$ and thus it eventually enters into the other side of the hyperplane. At this time, D_J becomes a $1\times n$ matrix denoted by D_j (small j). The dynamics of the system are now described by

$$\dot{x} = -D_j^T(D_jx+b_j)^+-a. \tag{3.20}$$

$-D_j^T(D_jx+b_j)^+$ is a vector in the direction of $-D_j^T$ with magnitude proportional to $(D_jx+b_j)^+$. If we decompose $-a$ into two parts as $-a = a_{\hat{j}}+a_j$, where a_j is the projection of $-a$ in the direction of D_j^T, then equation 3.20 becomes

$$\dot{x} = \left[-D_j^T(D_jx+b_j)^++a_j\right] + a_{\hat{j}}. \tag{3.21}$$

As the system evolves in time, it reaches a point where the first two terms on the right hand side of equation 3.21 cancel each other. Thereafter, the trajectory rolls along with $a_{\hat{j}}$ until it hits another hyperplane.

When there is at least one constraint violation, similar to the case with the initial state in the infeasible region, $-a$ can be decomposed as $-a=\sum\limits_{j\in J}a_j+a_{\hat{J}}$, where a_j is the projection of $-a$ on the span of D_j^T for $j\in J$ and $a_{\hat{J}}$ is the portion of $-a$ not in this span. Note that a_j is fixed and unique for each $j\in J$ due to the orthogonality of the normal vectors D_j^T. J, however, is not fixed. The differential equation of the system for $k\geq 1$ can now be written as

$$\dot{x} = -\sum_{j\in J}D_j^T(D_jx+b_j)^+ + \sum_{j\in J}a_j + a_{\hat{J}}. \tag{3.22}$$

Extending the above argument to $k>1$ illustrates that as the trajectory moves from one side of a hyperplane $g_j=0$ to the other side of the hyperplane, the projection a_j will be continually reduced by $D_j^T(D_jx+b_j)^+$ until it becomes zero. Although the trajectory may hit another hyperplane, the orthogonality of the D_j^T's nonetheless guarantees that $a_j-D_j^T(D_jx+b_j)^+$ eventually will become zero. It is clear at this point that if a trajectory starting in the feasible region hits the intersection of certain hyperplanes, each a_j will be diminished gradually by the corresponding term $D_j^T(D_jx+b_j)^+$ and the result stated above still holds.

Since $-a$ is not parallel to any hyperplane by assumption, it can be decomposed into n orthogonal projections. As the trajectory moves from region to region, it will eventually enter a region where n of the $2n$ constraints are violated. In this region the trajectory settles to the unique equilibrium point on which $-a$ is fully represented by $-\sum_{j\in J} D_j^T(D_jx+b_j)^+$.

3.3.3 Example of a 2-dimensional hypercube

Due to the orthogonality of the hyperplanes, the trajectories of the system can be illustrated by considering the case of a 2-dimensional hypercube. In Figure 3.3, the vector $-a$ is shown on the upper right corner together with its two orthogonal projections. The intersection of the two dotted lines is the equilibrium point of the system. The broadened gray lines are trajectories of the system corresponding to different initial states.

Figure 3.4 gives several examples with the initial states in the infeasible region. The corresponding $-\sum_{j\in J} D_j^T(D_jx+b_j)^+$ with respect to each initial point is drawn by a solid line segment along with vector $-a$ shown by a dotted line segment. The direction of the trajectory changes whenever it crosses a hyperplane. Once the trajectory

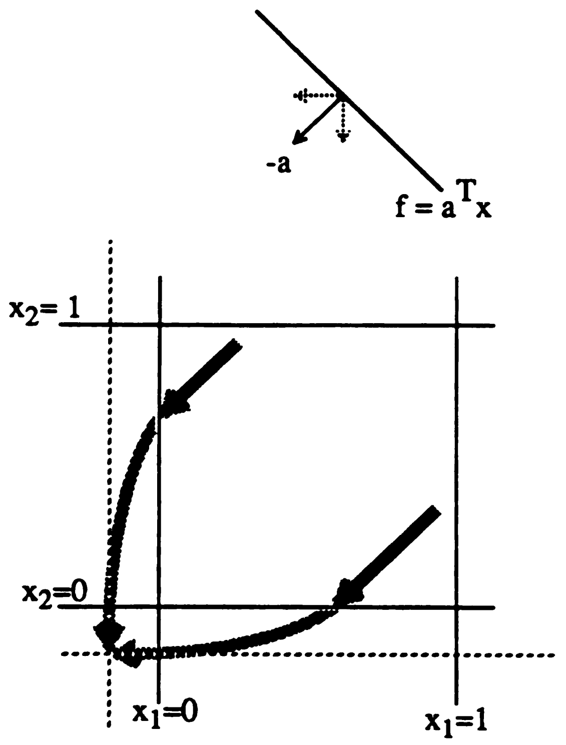


Figure 3.3. Trajectories starting in the feasible region.

reaches one of the dotted lines, i.e., one of the orthogonal projections of $-a$, it stays on the line and moves toward the equilibrium point. The equilibrium point will always lie in one of the four regions shown with light gray background in Figure 3.4. If we consider that the normal vectors of the hyperplanes separate these four regions from each other as bases of R^2, then there is only one region in which $-a$ will have positive coordinates. This is another interpretation of equation 3.17.

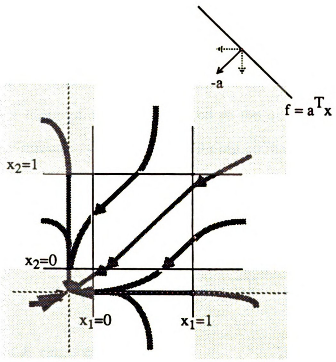


Figure 3.4. Trajectories starting in the infeasible region.

3.3.4 Further Analysis

Consider now the system defined by

$$\dot{x} = -s\left[D_J^T(D_Jx+b_J)^+\right]-a$$

$$= -s\left[\sum_{j\in J} D_j^T(D_jx+b_j)^+\right]-a \tag{3.23}$$

which is same as equation 3.9 but more explicitly expressed. When $s=1$, equation 3.23 is the same as equation 3.17. When $s>1$, denoting $v_j=s(D_jx+b_j)^+$ for $j\in J$, the equilibrium of equation 3.23 must satisfy

$$\nabla f + \sum_{j\in J} v_j \nabla g_j = 0. \tag{3.24}$$

The corresponding terms of equations 3.24 and 3.18, namely $I=J$ and $u_j=v_j$, can be matched. Since u_j is fixed for a given cost function, $(D_jx+b_j)^+=\frac{u_j}{s}$ is the equilibrium of equation 3.23. This means that the distance from the equilibrium point x to $g_j=0$ is reduced by a factor of s when compared to the case where $s=1$ (equation 3.19). By choosing s sufficiently large, the equilibrium point can thus be moved arbitrarily close to the intersection of the corresponding n hyperplanes, which is the minimizer of the cost function.

Next, the energy function E is defined as

$$E = f(x) + \frac{s}{2}\sum_{j\in J}\left[(D_jx+b_j)^+\right]^2. \tag{3.25}$$

The time derivative of E can is derived as

$$\begin{aligned}\frac{dE}{dt} &= \sum_i \frac{dE}{dx_i}\frac{dx_i}{dt}\\ &= \dot{x}^T\left[a+\sum_{j\in J} sD_j^T(D_jx+b_j)^+\right]\\ &= -\dot{x}^T\dot{x} < 0,\end{aligned} \tag{3.26}$$

for all $\dot{x}\neq 0$. Thus E is a Lyapunov function for the system. Hence the equilibrium is asymptotically stable by the asymptotical stability theorem [150]. In fact, it is also asymptotically stable in the global sense due to the unboundedness of $(D_jx+b_j)^+$ as $\|x\|\to\infty$.

For s sufficiently large, equation 3.25 is a form of the penalty function method. But, if we use the notation of v_j, then equation 3.25 becomes

$$E = f(x) + \frac{1}{2}\sum_{j\in J} v_j (D_jx+b_j)^+, \tag{3.27}$$

which is a form of the Lagrangian function method. So the forms of both methods are implicitly embedded in the network structure described by equation 3.23. As follows from the penalty function method theorem, the equilibrium of equation 3.23 approaches the minimizer of the cost function as $s \to \infty$. This, however, is impossible to implement in practice. A sufficiently large s will generally result in an equilibrium state which is a reasonably good approximation to the minimizer of the cost function.

A diagram of the trajectories of equation 3.23 for $s = \infty$ is shown in Figure 3.5 for a 2-dimensional hypercube. Whenever the trajectory lies in the infeasible region, it will be forced to move directly to either the closest hyperplane or the intersection of the hyperplanes if more than one constraint is violated. Then, it will move according to $-a$ to one of the hyperplanes $g_j = 0$, where $j \in I$. Once it reaches such a hyperplane, the trajectory slides on the hyperplane toward an intersection of hyperplanes with indices $j \in I$.

3.3.5 Extensions

The results of above analysis are directly applicable to cases where the hypercube is being scaled up or down, translated from the origin to any point, and/or rotated at any angle. It is also applicable to the region defined by $[l_1, u_1] \times \cdots \times [l_n, u_n]$ and its scaling, translation, and rotation, as long as the orthogonality of the hyperplanes ($g_j = 0$ for $j \in J$) is preserved. If the orthogonality is not preserved, the equilibrium of the system defined by equation 3.23 may not be unique, though the local asymptotic stability of the desired equilibrium still holds.

An extended theoretical argument for nonlinear programming in general is given in the next chapter.

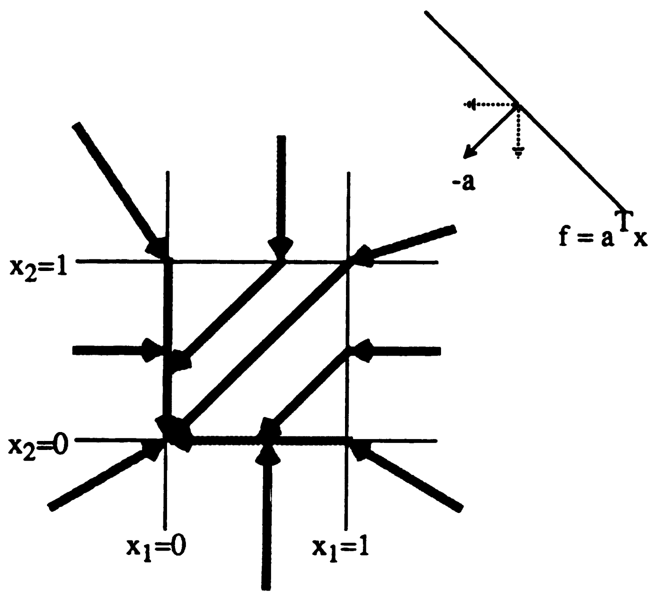


Figure 3.5. Trajectories for $s=\infty$.

CHAPTER IV

OPTIMIZATION NETWORK FORMULATION

A unifying mathematical framework for the Kennedy and Chua network for linear and quadratic programming is given in the first two sections followed by its extension to more complicated nonlinear programming problems. A two-phase optimization network model is then proposed which can obtain both the exact solution, in contrast to the approximate solution by Kennedy and Chua's network, as well as the corresponding Lagrange multipliers associated with each constraint.

4.1 Linear Programming Network Theory

Let the linear program (LP) be defined as in Section 3.2 and its objective functions is referred as $f_l(x)$. In this section, the network by Kennedy and Chua for linear programming is justified from the viewpoint of optimization theory. As has been pointed out in Section 3.2.2, $E_2(x)$ is precisely $L(s,x)$ for a fixed penalty parameter s. Thus we have the following proposition which is a restatement of Corollary 3.2.

Proposition 4.1: Let O be the set of minimizers of a feasible (LP). If O is bounded and contains only regular points, then given ε>0, there exists a sufficiently large s such that M, the set of the minimizers of the corresponding $E_2(x)$, satisfies

$\min\limits_{\bar{x}\in O}\|x-\bar{x}\|<\varepsilon$ for $x\in M$. □

Note that M is convex due to the fact that the sublevel set of a convex function, namely $E_2(x)$, is convex. Without the assumption of the boundedness of O, the proposition is still true. But when O is unbounded, any bounded subset of O is sufficient for obtaining a minimizer. This is due to the fact that if an (LP) has a finite optimum value, it must have a finite minimizer. Thus for O unbounded, we can place some additional, suitable bounding constraints into the original program so that the following discussion still holds.

Take C to be I in equation 3.9 and rewrite it as

$$\dot{x} = -\nabla f(x) - s\left[\sum_{j=1}^{m} g_j^{+}(x)\nabla g_j(x)\right]. \tag{4.1}$$

The block diagram of the system of described by equation 4.1 is drawn in Figure 4.1. Define $sg_j^{+}=v_j$ and $J(x)=\{j\,|\,g_j^{+}(x)>0,\ 1\le j\le m\}$. The equilibrium of equation 4.1 occurs when

$$\begin{aligned} 0 &= \nabla f(\bar{x}) + s\sum_{j=1}^{m} g_j^{+}(\bar{x})\nabla g_j(\bar{x}) \\ &= \nabla f(\bar{x}) + \sum_{j\in J(\bar{x})} v_j\nabla g_j(\bar{x}). \end{aligned} \tag{4.2}$$

Since for linear programming problems, $\nabla g_j=d_j$, where d_j is the jth row of D, equation 4.2 can be expressed as

$$\nabla f(\bar{x}) + \sum_{j\in J(\bar{x})} v_j d_j = 0. \tag{4.3}$$

Proposition 4.2: If the (LP) is feasible with finite optimum value, then the system described by equation 4.1 has an equilibrium. □

Proof: Let $\bar{x}$ be a minimizer to the (LP). Then, according to Corollary 3.1, there exists $\lambda_j>0$ for $j\in I(\bar{x})$ such that

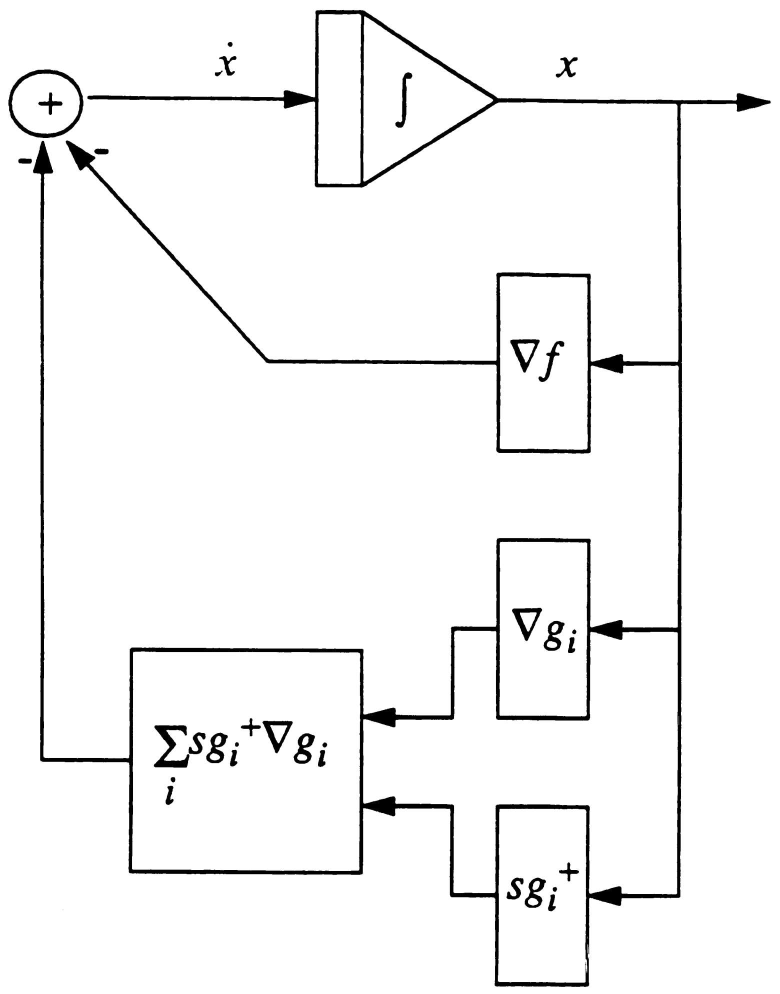


Figure 4.1. The block diagram of the system of equation 4.1.

$$a + \sum_{j \in I(\bar{x})} \lambda_j d_j = 0. \tag{4.4}$$

Choosing $J(\tilde{x}) = I(\bar{x})$ and $v_j = \lambda_j$, verifies the Proposition. □

Proposition 4.3: Let $\bar{x}$ be a solution to a feasible (LP) with a finite optimum value. If $\bar{x}$ is not regular, then the equilibrium of equation 4.1 corresponding to $\bar{x}$ is not unique. □

Proof: Since $\bar{x}$ is not regular, $\{d_j\}_{j\in I(\bar{x})}$ are linearly dependent. This implies that there exists two subsets of $I(\bar{x})$, say I' and I'' where $I'\neq I''$, such that

$$a + \sum_{j\in I'} \alpha_j d_j = 0, \text{ for } \alpha_j > 0, \tag{4.5}$$

and

$$a + \sum_{j\in I''} \beta_j d_j = 0, \text{ for } \beta_j > 0. \tag{4.6}$$

Let $\alpha_j = sg_j^+(x_1)$ for $j\in I'$ and $\beta_j = sg_j^+(x_2)$ for $j\in I''$. Since $I'\neq I''$, then there is at least one $\alpha_j \neq \beta_j$ and, consequently, at least one $g_j(x_1)\neq g_j(x_2)$. Thus the equilibrium corresponding to $\bar{x}$ is not unique. □

Note that even though the equilibrium points corresponding to $\bar{x}$ are not unique, they all result in same value of $E_2(x)$, since any equilibrium point of equation 4.1 is a minimizer to $E_2(x)$ as will be shown later.

In fact, it is observed that any convex combination of equations 4.5 and 4.6 satisfies the equilibrium condition. That is to say, for $0\leq\delta\leq 1$,

$$a + \delta \sum_{j\in I'} \alpha_j d_j + (1-\delta) \sum_{j\in I''} \beta_j d_j = 0. \tag{4.7}$$

Let $\hat{I} = I' \cup I''$ and $r = card(\hat{I})$. Denote α as the vector in R^r with its element equal to α_j if $j\in I'$, otherwise 0. Similarly, let β be the vector in R^r with its element equal to β_j if $j\in I''$, otherwise 0. Then, equation 4.7 implies that the line segment $[\alpha, \beta] \subset R_+^r$ is mapped to a single point, $-a$, in the closed convex cone $\{y \mid y = \sum_{j\in\hat{I}} \gamma_j d_j,\ \gamma_j > 0\}$. More generally, from equation 4.3 $-a$ lies in the closed convex cone $\{y \mid y = \sum_{j\in I(\bar{x})} \gamma_j d_j,\ \gamma_j > 0\}$ as seen from equation 4.4.

For linear programming $f(x)$ and $(g_j^+(x))^2$ are convex and continuously differentiable as is $E_2(x)$. Furthermore,

$$\frac{dE_2}{dt} = \sum_{j=1}^{n} \frac{\partial E_2}{\partial x_i} \frac{dx_i}{dt}$$

$$= \dot{x}^T \left[\nabla f + s \sum_{j=1}^{m} g_j^+ \nabla g_j \right]$$

$$= -\dot{x}^T \dot{x} \leq 0 \tag{4.8}$$

holds with equality only at the equilibrium $\tilde{x}$ of equation 4.1. Thus if the (LP) has a finite optimum value, $E_2(x)$ is a Lyapunov function of equation 4.1 and achieves its minimum at $\tilde{x}$.

Proposition 4.4: Let (LP) be a feasible program and $E_2(x)$ be correspondingly defined. Then the set of equilibrium points of equation 4.1 is the set of minimizers of $E_2(x)$. □

Proof: Let M_1 be the set of minimizers of $E_2(x)$ and M_2 the set of equilibrium points of equation 4.1. For $x \in M_2$, we have

$$\nabla E_2(x) = -\dot{x} = 0$$

and

$$\nabla^2 E_2(x) = \nabla g_J(x) \nabla g_J^T(x) \geq 0,$$

for $\nabla g_J = [\nabla g_j]_{j \in J(x)}$. Thus x satisfies the necessary condition of a minimum of $E_2(x)$. That is, $M_1 \subset M_2$. To show the converse, i.e., $M_2 \subset M_1$, we proceed as follows.

Since $M_1 \subset M_2$ and $M_1 \neq \emptyset$, there is at least one equilibrium point of equation 4.1, say $\tilde{x}$, which is a minimizer of E_2. Observe that $\nabla^2 E_2(\tilde{x})$ is strictly positive except along the direction of y at which

$$y^T \nabla g_J(\tilde{x}) \nabla g_J^T(\tilde{x}) y = 0. \tag{4.9}$$

So $\tilde{x}$ is a minimum except along the direction y. Therefore, $E_2(x)$ must be examined along the direction of y. Equation 4.9 implies $\nabla g_J^T(\tilde{x})y=0$, i.e., $y \in Null(\nabla g_J^T(\tilde{x}))$, where $Null(\nabla g_J^T(\tilde{x}))$ is the nullity of ∇g_J^T. Since $\nabla E_2(\tilde{x})=0$ implies $-a \in Range(\nabla g_J(\tilde{x}))$, we have $a^T y=0$ due to the fact that the linear subspace of $Null(\nabla g_J^T(\tilde{x}))$ is perpendicular to the linear subspace of $Range(\nabla g_J(\tilde{x}))$. Now

$$\begin{aligned}
\nabla E_2(\tilde{x}+y) &= a + \sum_{j \in J(\tilde{x}+y)} g_j^+(\tilde{x}+y)\nabla g_j(\tilde{x}+y) \\
&= a + \sum_{j \in J(\tilde{x})} g_j^+(\tilde{x}+y)\nabla g_j(\tilde{x}) \\
&= a + \sum_{j \in J(\tilde{x})} [\nabla g_j^T(\tilde{x}+y) - b]\nabla g_j(\tilde{x}) \\
&= a + \sum_{j \in J(\tilde{x})} [\nabla g_j^T\tilde{x} - b]\nabla g_j(\tilde{x}) \\
&= a + \sum_{j \in J(\tilde{x})} g_j^+(\tilde{x})\nabla g_j(\tilde{x}) \\
&= \nabla E_2(\tilde{x}) = 0,
\end{aligned} \tag{4.10}$$

if $J(\tilde{x}+y)=J(\tilde{x})$. Thus $\tilde{x}+y$, for $y \in Null(\nabla g_J^T(\tilde{x}))$, is an equilibrium of equation 4.1 as long as it does not evoke any new constraint violation. Moreover,

$$\begin{aligned}
E_2(\tilde{x}+y) &= a^T(\tilde{x}+y) + \frac{s}{2}\sum_{j \in J(\tilde{x}+y)} (g_j^+(\tilde{x}+y))^2 \\
&= a^T\tilde{x} + \frac{s}{2}\sum_{j \in J(\tilde{x})} (g_j^+(\tilde{x}+y))^2 \\
&= a^T\tilde{x} + \frac{s}{2}\sum_{j \in J(\tilde{x})} (\nabla g_j^T(\tilde{x}+y) - b)^2 \\
&= a^T\tilde{x} + \frac{s}{2}\sum_{j \in J(\tilde{x})} (\nabla g_j^T\tilde{x} - b)^2 \\
&= a^T\tilde{x} + \frac{s}{2}\sum_{j \in J(\tilde{x})} (g_j^+(\tilde{x}))^2
\end{aligned}$$

$$= E_2(\tilde{x}), \tag{4.11}$$

if $J(\tilde{x}+y)=J(\tilde{x})$. This implies $\tilde{x}+y$ is also a minimizer of E_2, and thus $M_2 \subset M_1$. □

Proposition 4.5: Let O be the set of minimizers of a feasible (LP). If O is bounded and contains only regular points, then there exists a sufficiently large s such that $J(\tilde{x})=I(\bar{x})$, for $\tilde{x}$ an equilibrium of equation 4.1 and $\bar{x}\in O$. □

Proof: From Proposition 4.4, $\tilde{x}$ is a minimizer of $E_2(x)$. According to Theorem 3.3, $\tilde{x}\to\bar{x}\in O$ and $v_j\to\lambda_j$ as $s\to\infty$. Since the convex cone spanned by $\nabla g_j(\bar{x})$ is closed, there exists a sufficiently large s such that $J(\tilde{x})=I(\bar{x})$. □

Note that O is necessarily convex. Also for $x_1, x_2\in O$, we have $I(x_1)=I(x_2)$ and the corresponding Lagrange multipliers λ_j are the same.

Proposition 4.6: If the unique minimizer $\bar{x}$ of a feasible (LP) is a regular point, then there exists a sufficiently large s such that the equilibrium of equation 4.1 $\tilde{x}$ with respect to $\bar{x}$ is unique. □

Proof: Let s_o be the parameter that satisfies Proposition 4.5 so that $J(\tilde{x})=I(\bar{x})$, for $\tilde{x}$ an equilibrium of equation 4.1. Since $\bar{x}$ is regular and unique, it implies that $card(I(\bar{x}))=n$, which furthermore implies $card(J(\tilde{x}))=n$ by the choice of s_o. In this case equation 4.2 is equivalent to solving a system of n linear equations for n unknowns, namely, the v_j's. Since $\nabla g_j(\tilde{x})$, for $j\in J(\tilde{x})$, are linearly independent by the choice of s_o, v is uniquely determined. But

$$v_j = s_o g_j^+(\tilde{x}) = s_o(\nabla g_j^T\tilde{x} - b_j), \tag{4.12}$$

so $\tilde{x}$ is uniquely determined, again by the linear independency of $\nabla g_j(\tilde{x})$, for $j\in J(\tilde{x})$. □

The above proposition considers only the case for which $card(I(\bar{x}))=n$. Suppose now that O contains more than one point, i.e., $card(I(\bar{x}))<n$. Let s_o be defined as in Proposition 4.6 and $\tilde{x}_o$ be a corresponding equilibrium of equation 4.1, then

$$a + s_o \sum_{j \in I(\overline{x})} g_j^+(\tilde{x}_o)d_j = 0. \tag{4.13}$$

As s changes continuously from s_o to ∞, $\tilde{x}$ moves continuously from $\tilde{x}_o$ to an $\overline{x} \in O$. If now s_0 is raised to s_1, the system moves from $\tilde{x}_o$ to $\tilde{x}_1$ so that

$$a + s_1 \sum_{j \in I(\overline{x})} g_j^+(\tilde{x}_1)d_j = 0. \tag{4.14}$$

Comparing equation 4.14 to 4.13, the constraint violation has been reduced by a factor of $\frac{s_o}{s_1}$. Also at the time of switching s_o to s_1 the system moves in the direction of a as can be seen by evaluating the dynamics of the system at $\tilde{x}_o$.

$$\begin{aligned} \dot{x}\,|_{\tilde{x}_o} &= -\left[a + s_1 \sum_{j \in I(\overline{x})} g_j^+(\tilde{x}_o)d_j\right] \\ &= -\left[(a + (s_o + s_1 - s_o) \sum_{j \in I(\overline{x})} g_j^+(\tilde{x}_o)d_j\right] \\ &= -(s_1 - s_o) \sum_{j \in I(\overline{x})} g_j^+(\tilde{x}_o)d_j \\ &= -(s_1 - s_o)\frac{-a}{s_o} \\ &= (\frac{s_1 - s_o}{s_o})a. \end{aligned} \tag{4.15}$$

Multiplying equation 4.13 by s_1 and equation 4.14 by s_o and taking their difference, we have

$$\begin{aligned} 0 &= (s_1 - s_o)a + s_1 s_o \sum_{j \in I(\overline{x})} (g_j^+(\tilde{x}_o) - g_j^+(\tilde{x}_1))d_j \\ &= (s_1 - s_o)a + s_1 s_o \sum_{j \in I(\overline{x})} (d_j^T\tilde{x}_o - b_j - d_j^T\tilde{x}_1 + b_j)d_j \\ &= (s_1 - s_o)a + s_1 s_o \sum_{j \in I(\overline{x})} d_j^T(\tilde{x}_o - \tilde{x}_1)d_j \end{aligned}$$

$$= (s_1 - s_o)a + s_1 s_o \sum_{j \in I(\bar{x})} d_j d_j^T(\tilde{x}_o - \tilde{x}_1). \tag{4.16}$$

If d_j is normalized, i.e., $\|d_j\|=1$, then $d_j d_j^T(\tilde{x}_o - \tilde{x}_1)$ is an orthogonal projection of $\tilde{x}_o - \tilde{x}_1$ along the direction of d_j. So equation 4.16 actually says that the sum of the projections of $\tilde{x}_o - \tilde{x}_1$ on the d_j's is in the direction of a. From equation 4.13 it is clear that $-a$ lies in the convex cone spanned by $\{d_j\}_{j \in I(\bar{x})}$. Also when switching s_o to s_1, the trajectory of the system will not move in any direction perpendicular the subspace spanned by $\{d_j\}_{j \in I(\bar{x})}$. Thus the following fact is noted:

Fact 4.1: Let $\tilde{x}_o$, $\tilde{x}_1$, $\bar{x}$, s_o, and s_1 be defined as above. If $\tilde{x}_o+y$ approaches $\bar{x}+y \in O$ as s changes from s_o to ∞, where y is perpendicular to $\{d_j\}_{j \in I(\bar{x})}$, then, if s_o is switched to s_1 at $x=\tilde{x}_o$, $\tilde{x}_1+y$ is the equilibrium of the system. □

The above fact can be geometrically interpreted. First the following definitions are helpful. Let $K \subset R^n$, then a hyperplane H is said to be a *supporting hyperplane* of K iff $K \subset H_+$ or $K \subset H_-$, and $cl(K) \cap H$ is nonempty, where $cl(K)$ is the closure of K. Let $K \subset R^n$ be closed and convex, then $F \subset K$ is called a *face* of K iff there exists a supporting hyperplane H of K such that $F = H \cap K$. This relationship is illustrated in Figure 4.2 in which K is represented by a gray background, H by a straight line, and F by a broadened line segment.

When there is more than one minimizer, the set of minimizers of a linear program (LP) actually forms a face. In Figure 4.3 the set of minimizers is represented by broadened line segments. The straight lines in the figure are the hyperplanes that define the feasible regions, which are shown with gray background. The equilibrium points of equation 4.1 are plotted by dotted line segments. There are three possible cases depending on whether the size of the equilibrium points of equation 4.1 is larger, smaller, or equal to the size of the set of the minimizers. (The size of a compact set S may be defined to be $\max_{x,y \in S}\|x-y\|$.) Three possible cases are illustrated in

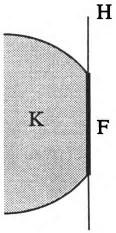


Figure 4.2. An illustration of the relationship among K, H, and F.

Figure 4.3. Any combination of these three cases is also possible. The arrows in Figure 4.3 depict the changing of the equilibrium set as the system is switched to a new s value.

Using the results obtained thus far we are able to conclude the following theorem. This is a very important result since it assures the complete stability of the network described by equation 4.1.

Theorem 4.1: Under the notations and assumptions of Proposition 4.1, then, given $\varepsilon>0$, there exists a sufficiently large s such that the system is completely stable and $\tilde{x}$ satisfies $\min_{\bar{x}\in O}\|\bar{x}-\tilde{x}\|<\varepsilon$. □

Proof: Let s_1 be the parameter that satisfies Proposition 4.5. Given $\varepsilon>0$, by Proposition 4.1 there exists a sufficiently large $s_2>s_1$ such that $\|\tilde{x}-\bar{x}\|<\varepsilon$, where $\tilde{x}$ is an equilibrium of equation 4.1 and $\bar{x}\in O$. Since O is bounded, by Proposition 4.1 the set M of equilibrium points of equation 4.1 is bounded. (Otherwise, for O bounded and M unbounded the conclusion of Proposition 4.1 would not be true.) Using $E_2(x)$ as the Lyapunov function for the system, LeSalle's Theorem [150] ensures that the system is

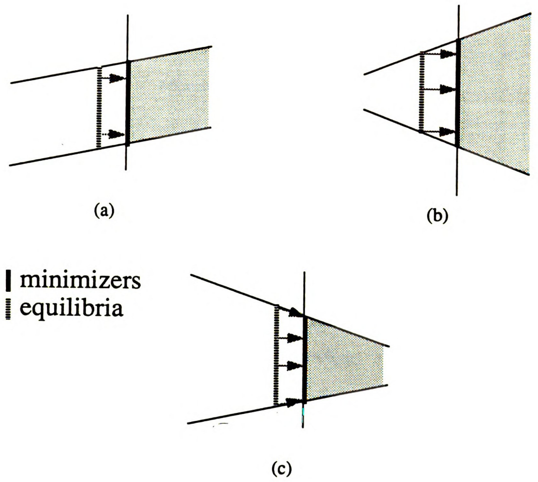


Figure 4.3. The relationship between the equilibrium set of equation 4.1 and the set of minimizers. (a) They are of the same size. (b) The equilibrium set is smaller in size than the set of the minimizers. (c) The equilibrium set is larger in size than the set of the minimizers.

completely stable in the sense that every trajectory will converge to a point in M without oscillation. □

4.2 Quadratic Programming Network Theory

Consider next the case of quadratic programming. Let the quadratic program (QP) be of the following form:

$$\text{Minimize } f_q(x) = \frac{1}{2}x^T Ax + a^T x$$

$$\text{subject to } g(x) = Dx - b \leq 0,$$

where A is a symmetric, positive semidefinite matrix. It is clear that $\nabla f_q(x) = Ax + a$. First the following lemma is needed.

Lemma 4.1: Let B be a symmetric, positive semidefinite matrix. Then $Bx=0$ if and only if $x^T Bx=0$. □

Proof: The only if part is clear, since for $Bx=0$, it follows that

$$x^T Bx = x^T(Bx) = 0.$$

The converse can be shown by using the Cauchy-Schwarz inequality; that is, if Q is symmetric, positive semidefinite then the following is true:

$$(x^T Qy)^2 \leq (x^T Qx)(y^T Qy), \tag{4.17}$$

for any x and y. By assumption $x^T Qx=0$, we have

$$0 \leq (x^T Qy)^2 \leq (x^T Qx)(y^T Qy) = 0, \tag{4.18}$$

for any value of y. This implies $|x^T Qy|=0$, for any y. But this implies $x^T Q=0$, i.e., $Qx=0$. The proof is complete. □

Similar to Proposition 4.1 for linear programming, we have the following proposition for quadratic programming which is a direct result of Corollary 3.2.

Proposition 4.7: Let O be the set of minimizers of a feasible (QP). If O is bounded and contains only regular points, then given $\varepsilon>0$, there exists a sufficiently large s

such that M, the set of the minimizers of the corresponding $E_2(x)$, satisfies $\min_{\bar{x}\in O}\|x-\bar{x}\|<\varepsilon$ for $x\in M$. □

Proposition 4.8: Let (QP) be a feasible program and $E_2(x)$ be correspondingly defined. Then the set of equilibrium points of equation 4.1 is the set of the minimizers of $E_2(x)$. □

Proof: Let M_1 be the set of the minimizers of $E_2(x)$ and M_2 the set of equilibrium points of equation 4.1. For $x\in M_2$, we have

$$\nabla E_2(x) = -\dot{x} = 0$$

and

$$\nabla^2 E_2(x) = A + \nabla g_J(x)\nabla g_J^T(x) \geq 0, \tag{4.19}$$

for $\nabla g_J=[\nabla g_j]_{j\in J(x)}$. If the A matrix in the definition of $f_q(x)$ is positive definite, then the Hessian of E_2 (equation 4.19) is positive definite. Thus the equilibrium of equation 4.1 is necessarily and sufficiently the unique minimizer of $E_2(x)$.

If, however, A is only positive semidefinite, then x satisfies only the necessary condition of a minimum. That is to say $M_1\subset M_2$. To show the converse, i.e., $M_2\subset M_1$, we proceed as follows.

Since $M_1\subset M_2$ and $M_1\neq\emptyset$, there is at least one equilibrium of equation 4.1, say $\tilde{x}$, which is a minimizer of E_2. Observe that $\nabla^2 E_2(\tilde{x})$ is strictly positive except along the direction of y at which

$$y^T(A + \nabla g_J(\tilde{x})\nabla g_J^T(\tilde{x}))y = 0. \tag{4.20}$$

So $\tilde{x}$ is a strict minimum except along the direction of y. We need only examine $E_2(x)$ along the direction of y. Equation 4.20 implies $\nabla g_J^T(\tilde{x})y=0$, i.e., $y\in Null(\nabla g_J^T(\tilde{x}))$. Equation 4.20 also implies $y^TAy=0$, and this implies $Ay=0$ by Lemma 4.1. Together we have $y\in Null(A)\cap Null(\nabla g_J^T(\tilde{x}))$. Also, since $\nabla E_2(\tilde{x})=0$

implies $-A\tilde{x}-a \in Range(\nabla g_J(\tilde{x}))$, we have

$$\begin{aligned} 0 &= y^T(A\tilde{x} + a) \\ &= y^T A\tilde{x} + y^T a \\ &= y^T a, \end{aligned} \tag{4.21}$$

due to the fact that $y \in Null(A) \cap Null(\nabla g_J^T(\tilde{x}))$ and that the linear subspace of $Null(\nabla g_J^T(\tilde{x}))$ is perpendicular to the linear subspace of $Range(\nabla g_J(\tilde{x}))$.

Now

$$\begin{aligned} \nabla E_2(\tilde{x}+y) &= A(\tilde{x} + y) + a + \sum_{j \in J(\tilde{x}+y)} g_j^+(\tilde{x}+y)\nabla g_j(\tilde{x}+y) \\ &= A\tilde{x} + a + \sum_{j \in J(\tilde{x})} g_j^+(\tilde{x}+y)\nabla g_j(\tilde{x}) \\ &= A\tilde{x} + a + \sum_{j \in J(\tilde{x})} [\nabla g_j^T(\tilde{x}+y) - b]\nabla g_j(\tilde{x}) \\ &= A\tilde{x} + a + \sum_{j \in J(\tilde{x})} [\nabla g_j^T\tilde{x} - b]\nabla g_j(\tilde{x}) \\ &= A\tilde{x} + a + \sum_{j \in J(\tilde{x})} g_j^+(\tilde{x})\nabla g_j(\tilde{x}) \\ &= \nabla E_2(\tilde{x}) = 0, \end{aligned} \tag{4.22}$$

if $J(\tilde{x}+y)=J(\tilde{x})$. Thus $\tilde{x}+y$, for $y \in Null(A) \cap Null(\nabla g_J^T(\tilde{x}))$, is an equilibrium of equation 4.1 as long as it does not evoke any new constraint violation. Moreover,

$$\begin{aligned} E_2(\tilde{x}+y) &= (\tilde{x}+y)^T A(\tilde{x}+y) + a^T(\tilde{x}+y) + \frac{s}{2} \sum_{j \in J(\tilde{x}+y)} (g_j^+(\tilde{x}+y))^2 \\ &= \tilde{x}^T A\tilde{x} + a^T\tilde{x} + \frac{s}{2} \sum_{j \in J(\tilde{x})} (g_j^+(\tilde{x}+y))^2 \\ &= \tilde{x}^T A\tilde{x} + a^T\tilde{x} + \frac{s}{2} \sum_{j \in J(\tilde{x})} (\nabla g_j^T(\tilde{x}+y) - b)^2 \\ &= \tilde{x}^T A\tilde{x} + a^T\tilde{x} + \frac{s}{2} \sum_{j \in J(\tilde{x})} (\nabla g_j^T\tilde{x} - b)^2 \end{aligned}$$

$$= \tilde{x}^T A\tilde{x} + a^T\tilde{x} + \frac{s}{2}\sum_{j\in J(\tilde{x})}(g_j^+(\tilde{x}))^2$$

$$= E_2(\tilde{x}), \tag{4.23}$$

if $J(\tilde{x}+y)=J(\tilde{x})$. This implies $\tilde{x}+y$ is also a minimizer of E_2, and thus $M_2 \subset M_1$. The proof is complete. □

Proposition 4.9: Let O be the set of minimizers of a feasible (QP). If O is bounded and contains only regular points, then there exists a sufficiently large s such that $J(\tilde{x})=I(\bar{x})$, for $\tilde{x}$ an equilibrium of equation 4.1 and $\bar{x}\in O$. □

Proof: Same proof as of Proposition 4.5. □

Proposition 4.10: If the unique minimizer $\bar{x}$ of a feasible program (QP) is a regular point, then there exists a sufficiently large s such that the equilibrium $\tilde{x}$ of equation 4.1 with respect to $\bar{x}$ is unique. □

Proof: Let s_o be the parameter that satisfies Proposition 4.9 so that $J(\tilde{x})=I(\bar{x})$, for $\tilde{x}$ an equilibrium of equation 4.1. Since $\bar{x}$ is regular and unique, it implies that $card(I(\bar{x}))=card(J(\tilde{x}))=n$ by the choice of s_o and $\nabla g^T_{J(\tilde{x})}$ is of full rank. The latter ensures that $\nabla g_{J(\tilde{x})}\nabla g^T_{J(\tilde{x})}$ is positive definite. Since $\tilde{x}$ is an equilibrium of equation 4.1, from equation 4.2 and using $J(\tilde{x})=I(\bar{x})$ we have

$$A\tilde{x} + a + s_o\sum_{j\in I(\bar{x})}\nabla g_j(\bar{x})(\nabla g_j^T\tilde{x} - b_j) = 0. \tag{4.24}$$

This is the same as

$$A\tilde{x} + a + s_o\nabla g_{I(\bar{x})}\left[\nabla g^T_{I(\bar{x})}\tilde{x} - b_J\right] = 0. \tag{4.25}$$

By rearranging the variable, we have

$$\left[A + s_o\nabla g_{I(\bar{x})}\nabla g^T_{I(\bar{x})}\right]\tilde{x} + a - s_o\nabla g_{I(\bar{x})}b_J = 0. \tag{4.26}$$

Observe that $A + s_o \nabla g_{I(\bar{x})} \nabla g_{I(\bar{x})}^T$ is positive definite, since A is positive semidefinite and $\nabla g_{J(\tilde{x})} \nabla g_{J(\tilde{x})}^T$ positive definite. Hence $\tilde{x}$ can be uniquely solved as

$$\tilde{x} = -\left[A + s_o \nabla g_{I(\bar{x})} \nabla g_{I(\bar{x})}^T\right]^{-1} \left[a - s_o \nabla g_{I(\bar{x})} b_J\right]. \ \square \qquad (4.27)$$

The above proposition considers only the case for which $card(I(\bar{x}))=n$. Suppose now that O contains more than one point, i.e., $card(I(\bar{x}))<n$. Let s_o be defined as in Proposition 4.10 and let $\tilde{x}_o$ be a corresponding equilibrium of equation 4.1, then

$$A\tilde{x}_o + a + s_o \nabla g_{I(\bar{x})}\left[\nabla g_{I(\bar{x})}^T \tilde{x}_o - b_J\right] = 0. \qquad (4.28)$$

As s changes continuously from s_o to ∞, $\tilde{x}$ moves continuously from $\tilde{x}_o$ to an $\bar{x} \in O$. If now we raise s_0 to s_1, the system moves from $\tilde{x}_o$ to $\tilde{x}_1$ so that

$$A\tilde{x}_1 + a + s_1 \nabla g_{I(\bar{x})}\left[\nabla g_{I(\bar{x})}^T \tilde{x}_1 - b_J\right] = 0. \qquad (4.29)$$

Assume that $\tilde{x}_o \to \tilde{x}_1 \to \bar{x}$ as $s_o \to s_1 \to \infty$. Let $y \in Null(A) \cap Null(\nabla g_J^T(\tilde{x}_o))$ and $J(\tilde{x}_o+y)=J(\tilde{x}_o)$. As the system of equation 4.1 is changed by raising s from s_o to s_1 at $x=\tilde{x}_o+y$, its dynamics are described by

$$\dot{x}\,|_{\tilde{x}_o+y} = A(\tilde{x}_o+y) + a + s_1 \sum_{j \in I(\bar{x})} \nabla g_j(\bar{x})(\nabla g_j^T(\tilde{x}_o+y) - b_j)$$

$$= A\tilde{x}_o + a + s_o \sum_{j \in I(\bar{x})} \nabla g_j(\bar{x})(\nabla g_j^T \tilde{x}_o - b_j). \qquad (4.30)$$

This is same as the dynamics of equation 4.1 when changing s_o to s_1 at $x=\tilde{x}_o$. By Proposition 4.8, $z^T a=0$ for any $z \in Null(A) \cap Null(\nabla g_J^T(\tilde{x}_o))$. It follows that

$$z^T \dot{x}\,|_{\tilde{x}_o+y} = z^T \left[A\tilde{x}_o + a + s_1 \sum_{j \in I(\bar{x})} \nabla g_j(\bar{x})(\nabla g_j^T \tilde{x}_o - b_j)\right]$$

$$= s_1 \sum_{j \in I(\bar{x})} z^T \nabla g_j(\bar{x})(\nabla g_j^T \tilde{x}_o - b_j)$$

$$= 0, \qquad (4.31)$$

since $z \in Null(\nabla g_j^T(\tilde{x}))$ implies that z is perpendicular to g_j for $j \in I(\bar{x})$. Equations 4.30 and 4.31 actually say that any vector in $Null(A) \cap Null(\nabla g_j^T(\tilde{x}))$ remains unchanged as the system evolves in time. This indicates the following fact.

Fact 4.2: Let $\tilde{x}_o$, $\tilde{x}_1$, $\bar{x}$, s_o, s_1, and y be defined as above. If $\tilde{x}_o+y$ approaches $\bar{x}+y \in O$ as $s_o \to \infty$, then, as s_o is switched to s_1 at $x=\tilde{x}_o$, the system obtains an equilibrium at $\tilde{x}_1+y$. □

Similar to the case of linear programming network, for the sake of practicality, the results for the quadratic programming network are summarized in the following theorem.

Theorem 4.2: Under the notations and assumptions of Proposition 4.7, then, given $\varepsilon>0$, there exists a sufficiently large s such that the system is completely stable and $\tilde{x}$ satisfies $\min_{\bar{x} \in O} \|\bar{x}-\tilde{x}\| < \varepsilon$. □

Proof: Basically the same proof as for Theorem 4.1. □

Note that the results of above argument still hold even if $I(\bar{x})$ is empty, i.e., no binding constraints. In this case the minizer lies in the interior of the feasibility set. This is a very strong result with a myriad of applications. In particular, the following two cases are examined.

Case (A): Solving $B_{n \times n} x = b$

B is assumed to be of full rank. This problem can be converted into minimizing $f(x) = \frac{1}{2}\|Bx-b\|^2$ over R^n. But

$$f(x) = \frac{1}{2}x^T B^T Bx - b^T Bx + \frac{1}{2}b^T b \tag{4.32}$$

and

$$\nabla f(x) = B^T(Bx-b). \tag{4.33}$$

Since $B^T B$ is symmetric and positive definite by the assumption placed on B, the optimization network formulation described by equation 4.1 for the case of quadratic programming can be applied. Define

$$\dot{x} = -\nabla f(x). \tag{4.34}$$

It follows that $\frac{df(x)}{dt} = -\|\nabla f(x)\|^2 \le 0$ with equality holds only at $Bx=b$. From Theorem 4.2 the unique equilibrium of equation 4.1, $x=B^{-1}b$, is globally asymptotically stable. Thus the problem is solved without actually calculating the inverse matrix of B.

Case (B): Solving $B_{m \times n} x = b$

In this case, assume $rank(B)=n<m$. This is a least squares problem. Similarly, we transform this problem into minimizing $f(x) = \frac{1}{2}\|Bx-b\|^2$ over R^n and define a system as equation 4.34. By the above theorems, the system converges asymptotically to an equilibrium $\tilde{x}$ which satisfies

$$0 = B^T(B\tilde{x}-b).$$

This implies that

$$\tilde{x} = (B^T B)^{-1} B^T b.$$

But this is exactly the least squares solution to $Bx=b$. Again the problem is solved without computing the inverse matrix.

The results derived can also be applied to the above two cases when $rank(B)<n$, though the equilibrium under such a condition is not unique. The above derivation is summarized as the following corollary.

Corollary 4.1: Let $B_{m \times n} x = b$, where B is a constant matrix, b a constant vector, and $n \le m$. If B is of full rank, then the dynamic system of equation 4.34 uniquely solves the problem $B_{m \times n} x = b$. If B is not of full rank, then depending on the initial states, the system of equation 4.34 approaches one of the solutions. □

Least squares problems with linear constraints can be solved as well by the same network formulation as described in equation 4.1. But the solution thus obtained is an approximation to the exact solution due to the finiteness of the penalty parameter s.

4.3 Nonlinear Programming Network Theory

Define the program (PP) to be

$$\text{Minimize } f_c(x) \text{ subject to } g(x)=Dx-b\leq 0,$$

where $f_c(x)$ is a C^1 convex function and $g(x)$ is similarly defined as in (QP).

Proposition 4.11: Let O be the set of minimizers of a feasible (PP). If O is bounded and contains only regular points, then given $\varepsilon>0$, there exists a sufficiently large s such that M, the set of the minimizers of the corresponding $E_2(x)$ satisfies $\min_{\bar{x}\in O}\|x-\bar{x}\|<\varepsilon$ for $x\in M$. □

Proof: The proof follows from Corollary 3.2. □

This proposition implies that there exists a sufficiently large s such that M is bounded. (Otherwise, there exists $x\in M$ and $\min_{\bar{x}\in O}\|x-\bar{x}\|>\varepsilon$ which thus it draws a contradiction.)

Proposition 4.12: Let (PP) be a feasible program and $E_2(x)$ be correspondingly defined. Then the equilibrium points of equation 4.1 are the minimizers of $E_2(x)$. □

Proof: Since $E_2(x)$ is convex and continuously differentiable on R^n, the critical points of $E_2(x)$ are the minimizers. This can be seen from Theorem 3.1 by taking λ and μ to be zero since there are no constraints for $E_2(x)$. But the critical points satisfy

$$0 = \nabla E_2(x) = -\dot{x}.$$

Thus the equilibrium points of equation 4.1 are just the minimizers of $E_2(x)$. □

It is known that if f is a C^2 (twice-continuously differentiable) real-valued function on an open convex set S in R^n, then f is convex if and only if its Hessian matrix

$$\nabla^2 f(x) = \left[\frac{\partial^2 f(x)}{\partial x_i \partial x_j} \right]$$

is positive semidefinite for every $x \in S$ [145]. Using this fact, the implications of Proposition 4.12 can be extended.

Proposition 4.13: Let (PP) be a feasible program and $E_2(x)$ be correspondingly defined. Assume that the Hessian matrix of $f_c(x)$ is positive definite. Then the equilibrium point of equation 4.1 is the unique minimizer of $E_2(x)$. □

Proof: By Proposition 4.12 the equilibrium of equation 4.1 is the minimizer of $E_2(x)$. Furthermore, we have that

$$\nabla^2 E_2(x) = \nabla^2 f_c(x) + \nabla g_J(x) \nabla g_J^T(x) \tag{4.35}$$

is positive definite, since $\nabla^2 f_c(x)$ is positive definite and $\nabla g_J(x) \nabla g_J^T(x)$ is positive semidefinite. Thus x is necessarily and sufficiently a strictly local minimizer of $E_2(x)$. Due to the convexity of $E_2(x)$, we can conclude that $\tilde{x}$ is the global (and thus unique) minimizer of $E_2(x)$. □

Proposition 4.14: Let O be the set of minimizers of a feasible (PP). If O is bounded and contains only regular points, then there exists a sufficiently large s such that $J(\tilde{x}) = I(\overline{x})$, where $\tilde{x}$ is an equilibrium of equation 4.1 and $\overline{x} \in O$. □

Proof: Same proof as of Proposition 4.5. □

Theorem 4.3: Under the notations and assumptions of Proposition 4.11, then, given $\varepsilon > 0$, there exists a sufficiently large s such that the system is completely stable and $\tilde{x}$ satisfies $\min_{\bar{x} \in O} \| \bar{x} - \tilde{x} \| < \varepsilon$. □

Proof: Let s_1 be the parameter that satisfies both Propositions 4.11 and 4.14. Since O is bounded, by Proposition 4.11 there exists a $s_2 > s_1$ such that the minimizers M of $E_2(x)$ are bounded. Using the fact that $E_2(x)$ is a Lyapunov function for the system of equation 4.1, LeSalle's theorem [150] ensures that the system is completely stable in the sense that every trajectory converges to a point in M without oscillation. □

Corollary 4.2: Under the notations and assumptions of Proposition 4.13, then, given $\varepsilon > 0$, there exists a sufficiently large s such that the unique equilibrium $\tilde{x}$ is globally asymptotically stable and $\| \bar{x} - \tilde{x} \| < \varepsilon$. □

Proof: Let s_1 be the parameter satisfies Proposition 4.11. Using the uniqueness of $\tilde{x}$ and applying Theorem 4.3, the result follows. □

The discussion in this section up to now is just an extension of the quadratic programming network with the objective function of the program allowed to be any C^1 convex function. The results obtained thus far apply to the program with linear equality constraints as well. In what follows, the discussion is extended to programs with nonlinear constraints, either equalities or inequalities.

Consider next the case of convex programming.

Lemma 4.2: Let (CP) be a convex program. Then $E_2(x)$ is a convex function. □

Proof: By definition,

$$E_2(x) = f(x) + \frac{s}{2} \left[\sum_{i=1}^{r} (g_i^+(x))^2 + \sum_{j=1}^{m} (h_j(x))^2 \right], \tag{4.36}$$

where f and the g_i's are C^1 convex functions on R^n and the h_j's are affine functions on R^n. The Hessian matrix of the second term on the right hand side of equation

4.36 is

$$\nabla^2\left[\frac{s}{2}\sum_{i=1}^{r}(g_i^+(x))^2\right] = s\left[\nabla g_J \nabla g_J^T + \sum_{i\in J} g_i^+ \nabla^2 g_i\right],$$

which is clearly positive semidefinite. Thus $\sum_{i=1}^{r}(g_i^+(x))^2$ is a convex function.

Since each h_j is an affine function, they can each be expressed as

$$c_j^T x + e_j$$

for some vector c_j and constant e_j, $1 \le j \le m$. By checking the positive semidefiniteness of the Hessian matrix, it can be shown that the last term of equation 4.34 is also a convex function.

Now $E_2(x)$ is a sum of three convex functions, so it must be a convex functions as well. □

If $E_2(x)$ for some (CP) is bounded below, then by the formulation of equation 4.1 the system will converge to a minimizer of $E_2(x)$. So the results obtained for (PP) still hold for the case of convex programs.

Proposition 4.15: If the basic program (PP) is replaced by a convex program (CP) in Propositions 4.11 - 4.14, Theorem 4.3, and Corollary 4.2 (with the assumption that the minimizers of (CP) are bounded and contain only regular points), then their results still hold. □

Proposition 4.14 implies that for a sufficiently large s, the violated constraints at an equilibrium $\tilde{x}$ of equation 4.1 are the same as the binding constraints at a minimizer $\bar{x}$. For (PP), if $J(\tilde{x}) = I(\bar{x})$, then

$$\nabla g_{j\in J(\tilde{x})} = \nabla g_{j\in I(\bar{x})} \tag{4.37}$$

since the gradient of a violated constraint is a constant vector. But this is not always true for the case of convex programming. The gradient of a violated constraint for a convex program (CP) is a continuous vector function rather than a constant vector. It is due to this continuity and the assumption of regularity of the minimizers that the conclusion of Theorem 3.3 holds. For a convex program, it is only guaranteed that equation 4.37 holds when $s=\infty$. (For further discussion see Chapter 12 of [151].)

All the programs discussed thus far consider equality constraints (h_j's) with only affine forms. The strategy to map such constraints into the optimization network is to replace them by two inequality constraints, $h_j \geq 0$ and $h_j \leq 0$. Since each of these two inequalities is again a convex function, the results of Theorem 3.1 are applicable. In order to solve more general problems by the optimization network technique, we want to relax the restriction on the affineness of equality constraints. First the following theorem is needed which is an extension of Theorem 3.1. (For proof see [146]).

Theorem 4.4: Let (P) be a program in the notation described in Section 3.1. Let $\bar{x}$ be a feasible solution to (P). Suppose that $\bar{x}$ is a regular point. Further suppose that f, g_i, $i \in I(\bar{x})$, and h_j, $1 \leq j \leq m$, are convex and all are differentiable at $\bar{x}$. Then $\bar{x}$ is a global optimal solution to (P) if and only if there exists $\lambda=[\lambda_1 \ldots \lambda_r]^T \geq 0$ and $\mu=[\mu_1 \ldots \mu_m]^T > 0$ together with $\bar{x}$ that satisfy

(i) $\lambda_i g_i(\bar{x}) = 0$ for $i=1,\ldots,r$

and

(ii) $\nabla f(\bar{x}) + \sum_{i=1}^{r} \lambda_i \nabla g_i(\bar{x}) + \sum_{j=1}^{m} \mu_j \nabla h_j(\bar{x}) = 0.$ □

Corollary 4.3: Let (P) be a program with the same notations and assumptions of Theorem 4.4 except that h_j, $1 \leq j \leq m$, are all concave. Then $\bar{x}$ is a global optimal solution to (P) if and only if there exists $\lambda=[\lambda_1 \ldots \lambda_r]^T \geq 0$ and $\mu=[\mu_1 \ldots \mu_m]^T < 0$ together with $\bar{x}$ such that the conditions (i) and (ii) of Theorem 4.4 hold. □

Proof: Since $-h_j(x)$ is convex for all j and

$$\mu_j \nabla(-h_j) = (-\mu_j) \nabla h_j,$$

the result follows by applying Theorem 4.4. □

Since the h_j's are not assumed to be affine, the minimizers of the problem may be isolated, rather than a convex set. If h_j is a convex function, then $h_j \le 0$ defines a convex set. Similarly, $-h_j \le 0$ defines a convex set for h_j concave. Now using the notation of a extended program, i.e., letting $g_{r+2j-1} = h_j$ and $g_{r+2j} = -h_j$, leads to the following corollary.

Corollary 4.4: Let (P') be the extended program stated above for convex functions $f \in C^1$ and $g_i \in C^1$, $1 \le i \le r$, and convex or concave functions $h_j \in C^1$, $1 \le j \le m$. Let $L(s, x)$ and s be correspondingly defined. Let x_k be a minimizer of $L(s_k, x)$. Assume $x_k \to \bar{x}$ for some $\bar{x} \in O$, where O is the set of minimizers of (P'). Suppose O is bounded and contains only regular points. Then for h_j convex, $s_k g^+_{r+2j-1}(x_k) \to \mu_j$ and $s_k g^+_{r+2j}(x_k) \to 0$; for h_j concave, $s_k g^+_{r+2j-1}(x_k) \to 0$ and $s_k g^+_{r+2j}(x_k) \to (-\mu_j)$, where u_j is the corresponding Lagrange multiplier at $\bar{x}$. □

Proof: Rewrite equation 3.4 as

$$L(s, x) = f(x) + \frac{s}{2}\left[\sum_{i=1}^{r}(g_i^+(x))^2 + \sum_{j=1}^{m}(g^+_{r+2j-1}(x))^2 + \sum_{j=1}^{m}(g^+_{r+2j}(x))^2\right]. \quad (4.38)$$

Applying Theorems 3.3 and 4.4, it follows that if h_j is convex then $s_k g^+_{r+2j-1}(x_k) \to \mu_j$ and $s_k g^+_{r+2j}(x_k) \to 0$. By applying Theorem 3.3 and Corollary 4.3 it follows that if h_j is concave, $s_k g^+_{r+2j-1}(x_k) \to 0$ and $s_k g^+_{r+2j}(x_k) \to (-\mu_j)$. □

In fact, since $g^+_{r+2j-1}(x)$ and $g^+_{r+2j}(x)$ are continuous and mutually exclusive for $1 \le j \le m$, there exists a sufficiently large s_k such that $g^+_{r+2j-1}(x_k) > 0$ and $g^+_{r+2j}(x_k) = 0$ for h_j convex, and $g^+_{r+2j-1}(x_k) = 0$ and $g^+_{r+2j}(x_k) > 0$ for h_j concave.

Proposition 4.16: Let s_k be the parameter stated above. Then $L(s_k, x)$ is a convex function in a neighborhood of x_k. □

Proof: Due to the choice of s_k and the continuity of $\{g_i^+\}_{r+1}^{r+2m}$, there exists a neighborhood N of x_k so that for $x \in N$, $s_k g_{r+2j-1}^+(x) > 0$ and $s_k g_{r+2j}^+(x) = 0$ if h_j is convex; $s_k g_{r+2j-1}^+(x) = 0$ and $s_k g_{r+2j}^+(x) > 0$ if h_j is concave. Now for $x \in N$, if h_j is convex,

$$(g_{r+2j-1}^+(x))^2 + (g_{r+2j}^+(x))^2 = (g_{r+2j-1}^+(x))^2 = (h_j^+(x))^2$$

is convex since h_j^+ is convex. Similarly for $x \in N$, if h_j is concave,

$$(g_{r+2j-1}^+(x))^2 + (g_{r+2j}^+(x))^2 = (g_{r+2j}^+(x))^2 = ((-h_j)^+(x))^2$$

is convex since $(-h_j)^+$ is convex. As a sum of some convex functions, it follows that $L(s_k, x)$ is convex. □

To see that h_j^+ is convex provided h_j is convex, consider the following lemma.

Lemma 4.3: Let $g(x)$ be a convex function on a nonempty convex set $X \subset R^n$, then

$$g^+(x) = \begin{cases} g(x) & \text{if } g(x)>0 \\ 0 & \text{if } g(x)\le 0 \end{cases}$$

is a convex function over X. □

Proof: Choose $x, y \in X$. By the convexity of g, it follows that for $0 \le \lambda \le 1$

$$\begin{aligned} g(\lambda x+(1-\lambda)y) &\le \lambda g(x) + (1-\lambda) g(y) \\ &\le \lambda g^+(x) + (1-\lambda) g^+(y). \end{aligned}$$

If $g(\lambda x+(1-\lambda)y)>0$, then $g^+(\lambda x+(1-\lambda)y) = g(\lambda x+(1-\lambda)y)$, and we are done. If $g(\lambda x+(1-\lambda)y) \le 0$, then

$$g^+(\lambda x+(1-\lambda)y) = 0 \le \lambda g^+(x) + (1-\lambda) g^+(y)$$

and the proof is complete. □

The neighborhood mentioned in Proposition 4.16 need not to be a R^n ball around x_k, i.e., $B_\varepsilon = \{x \in R^n \mid \|x-x_k\| < \varepsilon\}$. In fact, it may be any shape convex set in which $L(s_k, x)$ is convex. By Proposition 4.16 there exists an $s>s_k$ and a neighborhood N such that $E_2(x)$ is convex over N. Thus if the boundedness of $E_2(x)$ is assumed, then the network formulation of equation 4.1 is again useful to converge locally to one of the minimizers of $E_2(x)$ provided the starting point lies in N. If the set O of minimizers is isolated, there is more than one neighborhood and $E_2(x)$ is convex over each one. To restrict the discussion to the local stability of equation 4.1 over a particular neighborhood, assume there is only one isolated minimizer set, and correspondingly one such neighborhood.

Proposition 4.17: Let (P') be a program in its extended form such that $f \in C^1$ and $g_i \in C^1$, $1 \le i \le r$, are convex functions, and $h_j \in C^1$, $1 \le j \le m$, are convex or concave functions. Let N be a neighborhood that satisfies Proposition 4.16. Suppose that the set of minimizers of (P') is bounded and contains only regular points. Then if the program (PP) is replaced by a program (P) in Propositions 4.11 - 4.14, Theorem 4.3, and Corollary 4.2, the results still hold locally over N. □

Although in Proposition 4.17 the stability of equation 4.1 is guaranteed only locally, in practice the N neighborhood can be very large as will be shown in some examples. Also in Proposition 4.17 the convexity and concavity are assumed throughout R^n, but this is not absolutely necessary so for the proposition to hold. It is sufficient to assume that the convexity and concavity of the corresponding functions hold on a nonempty open set, say $\Omega \subset R^n$, and restrict all discussion to Ω rather than R^n. That is to

$$
\begin{aligned}
&\text{Minimize } f(x) \\
&\text{subject to } g_i(x) \le 0, \text{ for } i=1 \text{ to } r, \\
&\qquad h_j(x)=0, \text{ for } j=1 \text{ to } m,
\end{aligned}
$$

and

$$x \in \Omega,$$

where f and g_i's are convex, differentiable on the open set Ω, and h_j's are convex or concave, differentiable on Ω.

Let $f: X \to R$, where X is a nonempty convex set in R^n. The function f is said to be *quasiconvex* if, for each x_1 and $x_2 \in X$, the following inequality is true:

$$f(\lambda x_1 + (1-\lambda)x_2) \le \max\{f(x_1), f(x_2)\} \quad \text{for each } \lambda \in (0,1).$$

The function f is said to be *quasiconcave* if $-f$ is quasiconvex. Let S be a nonempty open set in R^n, and let $g: S \to R$ be differentiable on S. The function g is said to be *pseudoconvex* if for each $x_1, x_2 \in S$ with $\nabla g(x_1)^T(x_2-x_1) \ge 0$ we have $g(x_2) \ge g(x_1)$, or equivalently, if $g(x_2) < g(x_1)$ then $\nabla g(x_1)^T(x_2-x_1) < 0$. The function g is said to be *pseudoconcave* if $-g$ is pseudoconvex. The pseudoconvexity of f ensures that if $\nabla f(\bar{x})=0$, then $\bar{x}$ is a global minimum of f. Figure 4.4 shows some examples of quasiconvex and pseudoconvex functions. Both quasiconvex and pseudoconvex functions assure that there are only global minimizers. But quasiconvex functions may have saddle points and pseudoconvex functions may not. Therefore, for pseudoconvex functions, the point at which the gradient vanishes is a global minimizer. Note that under differentiability, convexity implies pseudoconvexity, and under lower semi-continuity, pseudoconvexity implies quasiconvexity.

Thus, for a program (P) the results of Theorem 4.4 still hold by properly relaxing the convexity assumptions.

Theorem 4.5: Let (P) be a program in the notation described in Section 3.1. Let $\bar{x}$ be a feasible solution to (P). Suppose that $\bar{x}$ is a regular point. Further suppose that f is pseudoconvex, g_i is quasiconvex for $i \in I(\bar{x})$, and h_j is quasiconvex for $1 \le j \le m$. And all are differentiable at $\bar{x}$. Then $\bar{x}$ is a global optimal solution to (P) if and only if there exists $\lambda=[\lambda_1 \ldots \lambda_r]^T \ge 0$ and $\mu=[\mu_1 \ldots \mu_m]^T > 0$ together with $\bar{x}$ that satisfy

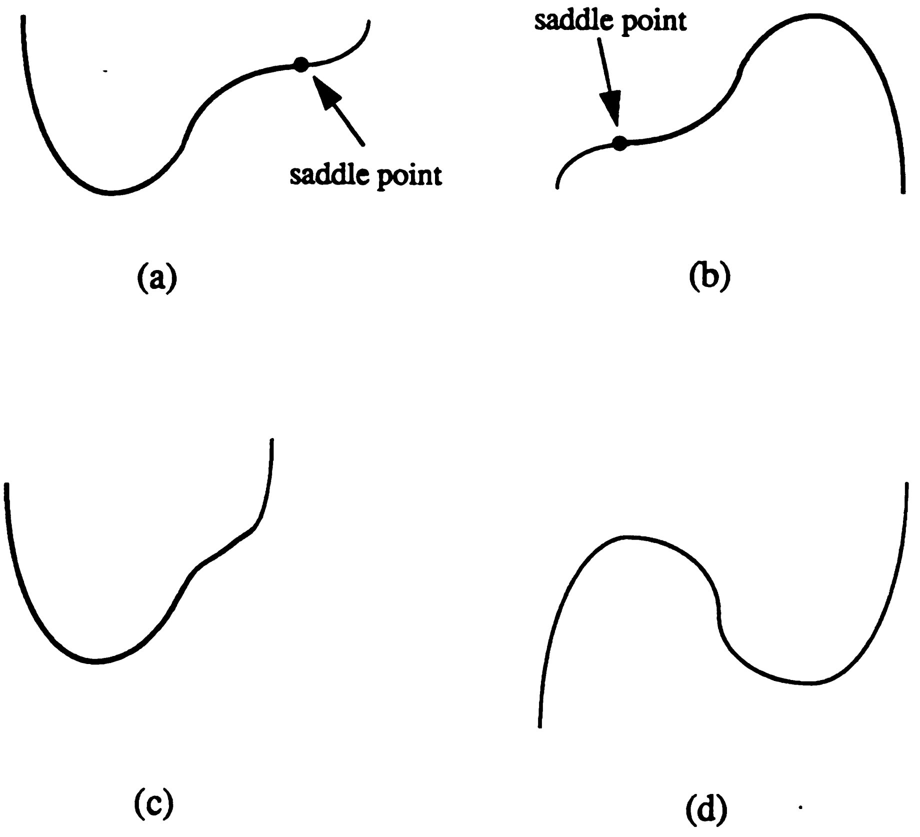


Figure 4.4. (a) Quasiconvex. (b) Quasiconcave. (c) Pseudoconvex. (d) Not Pseudoconvex and not quasiconvex.

(i) $\lambda_i g_i(\bar{x}) = 0$ for $i=1,\dots,r$

and

(ii) $\nabla f(\bar{x}) + \sum_{i=1}^{r}\lambda_i \nabla g_i(\bar{x}) + \sum_{j=1}^{m}\mu_j \nabla h_j(\bar{x}) = 0$. □

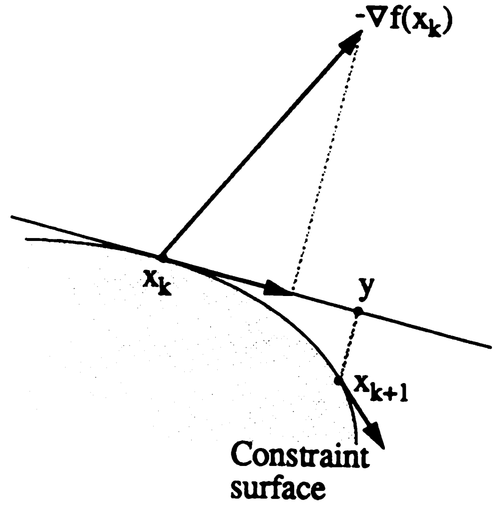
Corollary 4.5: Let (P) be a program with same notations and assumptions as in Theorem 4.5 except that h_j, $1 \le j \le m$, are all quasiconcave. Then $\bar{x}$ is a global optimal solution to (P) if and only if there exists $\lambda=[\lambda_1 \dots \lambda_r]^T \ge 0$ and $\mu=[\mu_1 \dots \mu_m]^T < 0$ together with $\bar{x}$ such that the conditions (i) and (ii) of Theorem 4.5 hold. □

Proof: Same proof as Corollary 4.3. □

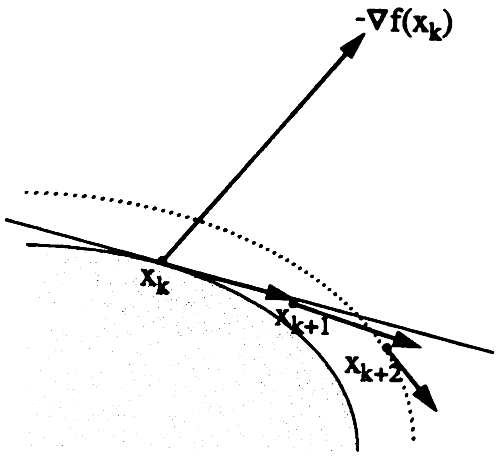
Now following a similar argument used to derive Proposition 4.17 it can be shown that the network formulation of equation 4.1 is also useful to obtain (locally) an approximate solution to (P) with proper pseudo- and quasi-convexity assumptions.

Corollary 4.6: Let (P') be an extended program such that $f \in C^1$ is a pseudoconvex C^1 function, g_i is a quasiconvex C^1 function for $1 \le i \le r$, and $h_j \in C^1$ is either quasiconvex or quasiconcave for $1 \le i \le m$. Let N be a neighborhood that satisfies Proposition 4.16. Suppose that the set of minimizers of (P') is bounded and contains only regular points. Then if the program (PP) is replaced by a program (P) in Propositions 4.11 - 4.14, Theorem 4.3, and Corollary 4.2, the results still hold locally over N. □

Note that the network formulation of equation 4.1 may be considered as a continuous approximation of the *gradient projection method* [147]. Figure 4.5(a) depicts the idea of the gradient projection method. The negative gradient of the objective function is projected onto the tangent surface of the active constraint set in order to find a point y. Then a new point x_{k+1} is found along the direction perpendicular to the tangent plane of x_k. Figure 4.5(b) illustrates the dynamics of equation 4.1 when the trajectory is on the boundary of the feasible region. The linear programming problem with hypercube feasible region described in Section 3.3 is a good example for illustrating the similarity between the gradient projection method and the dynamics of equation 4.1.



(a)



(b)

Figure 4.5. (a) Gradient projection method [147]. (b) The dynamics of equation 4.1 on the boundary of the feasible region.

4.4 A Two-Phase Optimization Network

In previous sections of this chapter it has shown that there exists a sufficiently large s so the network formulation as described by equation 4.1 is guaranteed to converge to an approximate solution for a large class of nonlinear programming problems. In what follows, a two-phase optimization network model, which can obtain both the exact solution to the problem as well as the corresponding Lagrange multipliers associated to each constraint is proposed. For linear programming problems, the network solves both the primal and the dual problem exactly.

For the sake of argument, assume, unless otherwise explained, that the program (P) considered in this section is a convex program for which $f \in C^1$ and $g_i \in C^1$, $1 \le i \le r$, are convex functions, and $h_j \in C^1$, $1 \le j \le m$, are affine functions.. It is clear that if the set M of minimizers of (P) contains only regular points, for $\bar{x} \in M$ it follows that $q \le card(I(\bar{x})) \le n$. The penalty function used here is

$$L(s,x) = f(x) + \frac{s}{2}\left[\sum_{i=1}^{r}(g_i^+(x))^2 + \sum_{j=1}^{m} h_j^2(x)\right]. \tag{4.39}$$

It is formed based on the program (P) rather than on its extended form.

The block diagram of a two-phase optimization network is shown in Figure 4.6. The network operates under different dynamics as the phase is changed by a predetermined timing switch. For $0 \le t < t_1$, the network operates according to the following dynamics:

$$\dot{x} = -\nabla f(x) - s\left[\nabla g_J(x) g_J^+(x) + \nabla h(x) h(x)\right]. \tag{4.40}$$

It is just in this case the same as the system described by equation 4.1 except that the sign of h_j for $1 \le j \le m$ may be either positive or negative. The subsystems within the two large rectangles do not contribute for $t < t_1$. When $t \ge t_1$, the dynamics of the

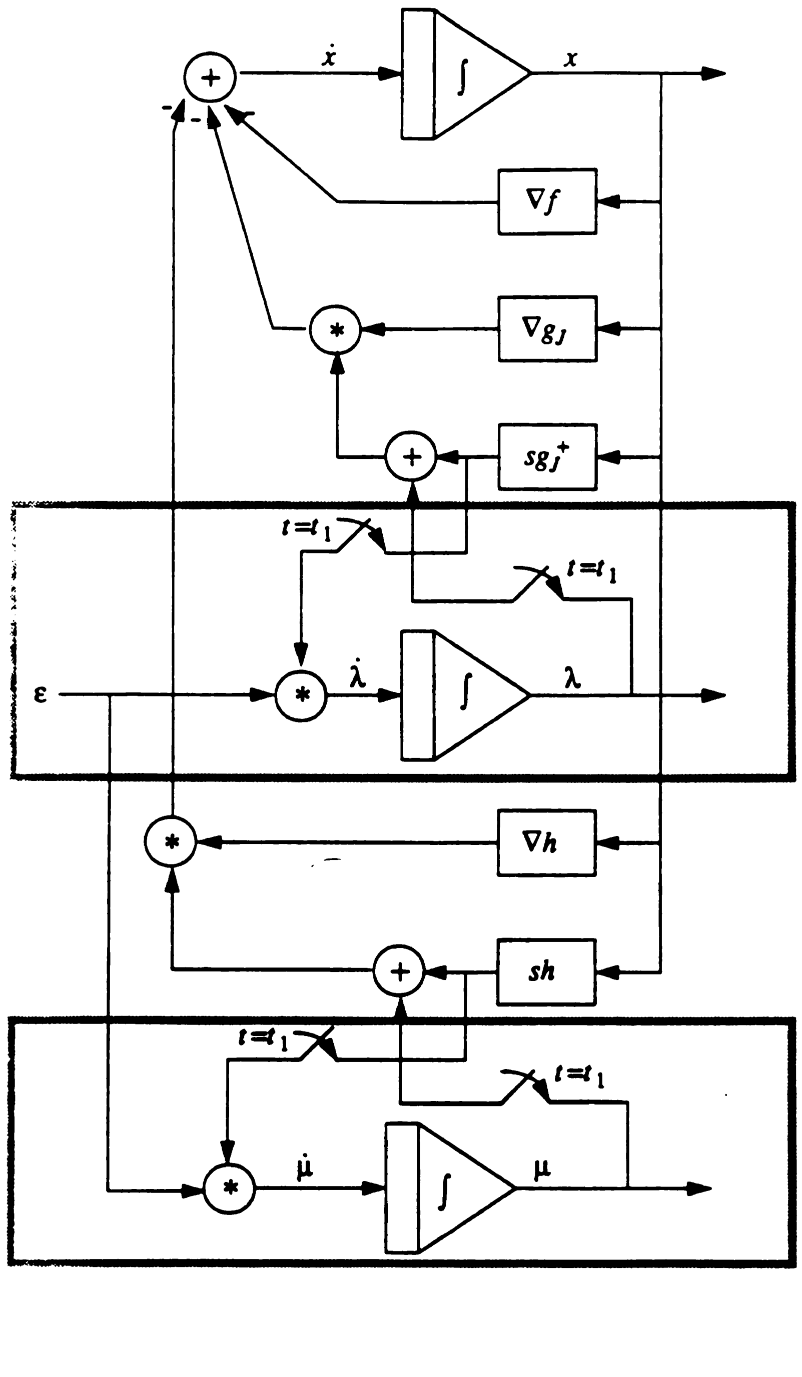


Figure 4.6. The block diagram of the two-phase optimization network.

network becomes

$$\dot{x} = -\nabla f(x) - \nabla g_J(sg_J^+ + \lambda) - \nabla h(sh + \mu), \tag{4.41}$$

$$\dot{\lambda} = \varepsilon(sg_J^+), \tag{4.42}$$

and

$$\dot{\mu} = \varepsilon(sh). \tag{4.43}$$

where ε is a small positive constant. For this network there is no restriction on the initial condition of x while the initial values of λ and μ are set to be zero vectors.

Due to the network formulation it is easy to check that the equilibrium of the system occurs when $g_J^+(x) = 0$, $h(x) = 0$, $\lambda > 0$, and

$$\nabla f(x) + \lambda \nabla g_J(x) + \mu \nabla h(x) = 0. \tag{4.44}$$

But this actually satisfies the optimality conditions of Theorem 3.3, and thus a equilibrium point of the two-phase network is nothing but a global minimizer to a convex program (P).

The rationale behind the two-phase network formulation is the following. In phase 1 $(t<t_1)$ it follows from Theorem 3.3 and Proposition 4.15 that for a sufficiently large s, equation 4.40 converges to an equilibrium $\tilde{x}$ at which $sg_i^+(\tilde{x})$ and $sh_j(\tilde{x})$ are very close to λ_i and μ_j, respectively, where λ_i and μ_j are the corresponding Lagrange multipliers defined in Theorem 3.1. Assume s is chosen such that $J(\tilde{x}) = I(\bar{x})$ for $\bar{x} \in O$, the minimizers of (P). By choosing t_1 properly, the trajectory of the system can be assumed to be within a small neighborhood of $\tilde{x}$, say $B(\tilde{x},\delta) = \{x \in R^n \mid \|x - \tilde{x}\| < \delta\}$, such that the approximation of λ_i and μ_j by $sg_i^+(x)$ and $sh_j(x)$ respectively is qualitatively preserved for $x \in B(\tilde{x},\delta)$.

In Phase 2 $(t \geq t_1)$, the network begins to shift the directional vector $sg_i^+(x)$ gradually to λ_i, and $sh_j(x)$ to μ_j. By imposing a small vector ε, the updating of equations 4.42 and 4.43 is comparatively much slower than that of equation 4.41. Approximation of such dynamics is possible by considering λ and μ to be fixed. Then it can be seen that equation 4.41 is seeking a minimum point of the *augmented Lagrangian function*

$$L_a(s, x) = f(x) + \lambda^T g(x) + \mu^T h(x) + \frac{s}{2}\left[||g^+(x)||^2 + ||h(x)||^2\right]. \quad (4.45)$$

With a small enough ε, the stability of equation 4.41 is qualitatively preserved and the system is driven toward a equilibrium point, which is a minimizer of (P).

In practice ε may be chosen to be $\frac{1}{s}$ leaving the network with only one preselected parameter. But using a ε independent of s gives more freedom to control the dynamics of the network. Also if the initial condition of x is in the feasible region, simulation results show that phase 2 of the network alone is sufficient to ensure the convergence to a minimizer of a convex program (P). For a more general program such as those covered in the last section, the convergence is only locally assured.

The network formulation proposed here is similar to the *multiplier method*, or the *augmented Lagrangian method* [147,152]. The multiplier method for the equality constrained problem (EP)

$$\text{minimize } f(x)$$

$$\text{subject to } h(x) = 0$$

is to transform the problem into a successive process. Within each iteration an x_k is sought to minimize

$$l_s(x, \mu_k) = f(x) + \mu_k^T h(x) + \frac{s}{2}||h(x)||^2. \quad (4.46)$$

If x_k is found, then μ_k is updated according to

$$\mu_{k+1} = \mu_k + sh(x_x). \quad (4.47)$$

Then, an x_{k+1} is sought to minimize $l_s(x, \mu_{k+1})$, and so on. Equation 4.43 is the same, in essence, as equation 4.47 with the former being a continuous approximation to the latter.

In practice, the multiplier method converts the inequality constraint $g_i(x) \leq 0$ to an equivalent equality constraint $g_i(x)+z_i = 0$ by adding a dummy nonnegative variable z_i. Then the problem is solved successively by finding x_k and $z_k \geq 0$ that minimize

$$f(x) + \lambda_k^T[g(x) + z] + \frac{s}{2}||g(x) + z||^2, \tag{4.48}$$

and updating λ_k similar to equation 4.47. Finally, a mapping scheme similar to the two-phase network can be formed to deal with only equality constraints. The state variables of such a network include x, z, λ, and μ.

CHAPTER V

NETWORK SIMULATION

To demonstrate the behavior of the networks proposed in the last chapter and to validate their properties, some examples for various problems have been performed using the ACSL (Advanced Continuous Simulation Language) software package (Sun version). The numerical algorithm used to integrate equations 4.1 and 4.38 to 4.41 is a 4-th order Runge-Kutta method.

5.1 Linear Programming

For linear programming, consider the following problem (LP_1) taken from [6]:

$$\text{Minimize } f(x) = -x_1 - x_2$$

subject to

$$g_1(x) = \frac{5}{12}x_1 - x_2 - \frac{35}{12} \leq 0,$$

$$g_2(x) = \frac{5}{2}x_1 + x_2 - \frac{35}{2} \leq 0,$$

$$g_3(x) = -x_1 - 5 \leq 0,$$

$$g_4(x) = x_2 - 5 \leq 0.$$

The corresponding feasible region is the gray area in Figure 5.1.

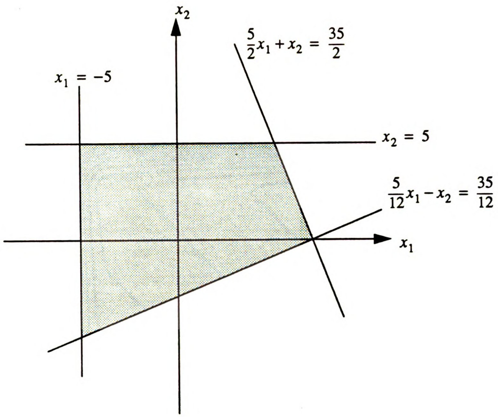

x_2
$\frac{5}{2}x_1 + x_2 = \frac{35}{2}$
$x_1 = -5$
$x_2 = 5$
$\frac{5}{12}x_1 - x_2 = \frac{35}{12}$
x_1

Figure 5.1. The feasible region of (LP_1).

It is easy to verify that the optimal solution to this problem is $x=(5.0, 5.0)^T$ and the corresponding Lagrange multipliers are $\lambda_1=0$, $\lambda_2=0.4$, $\lambda_3=0$, and $\lambda_4=0.6$. To illustrate the variation of $E_2(x)$ with respect to different values of s, the contours of $E_2(x)$ for $s=1$ and $s=10$ are shown in Figure 5.2. Comparing Figures 5.3(b) and (d), we see that the larger the value of s, the closer the minimizer of $E_2(x)$ to the optimal solution. This is implied by Theorem 3.3. Since $E_2(x)$ is radially unbounded, the unique equilibrium of equation 4.1, i.e., the minimizer of $E_2(x)$ is thus globally asymptotically stable for some large s. For this example, $s \geq 1$ is more than sufficient. The equilibrium point of the network and the terms $sg_j^+(x)$ are given in Table 5.1 for

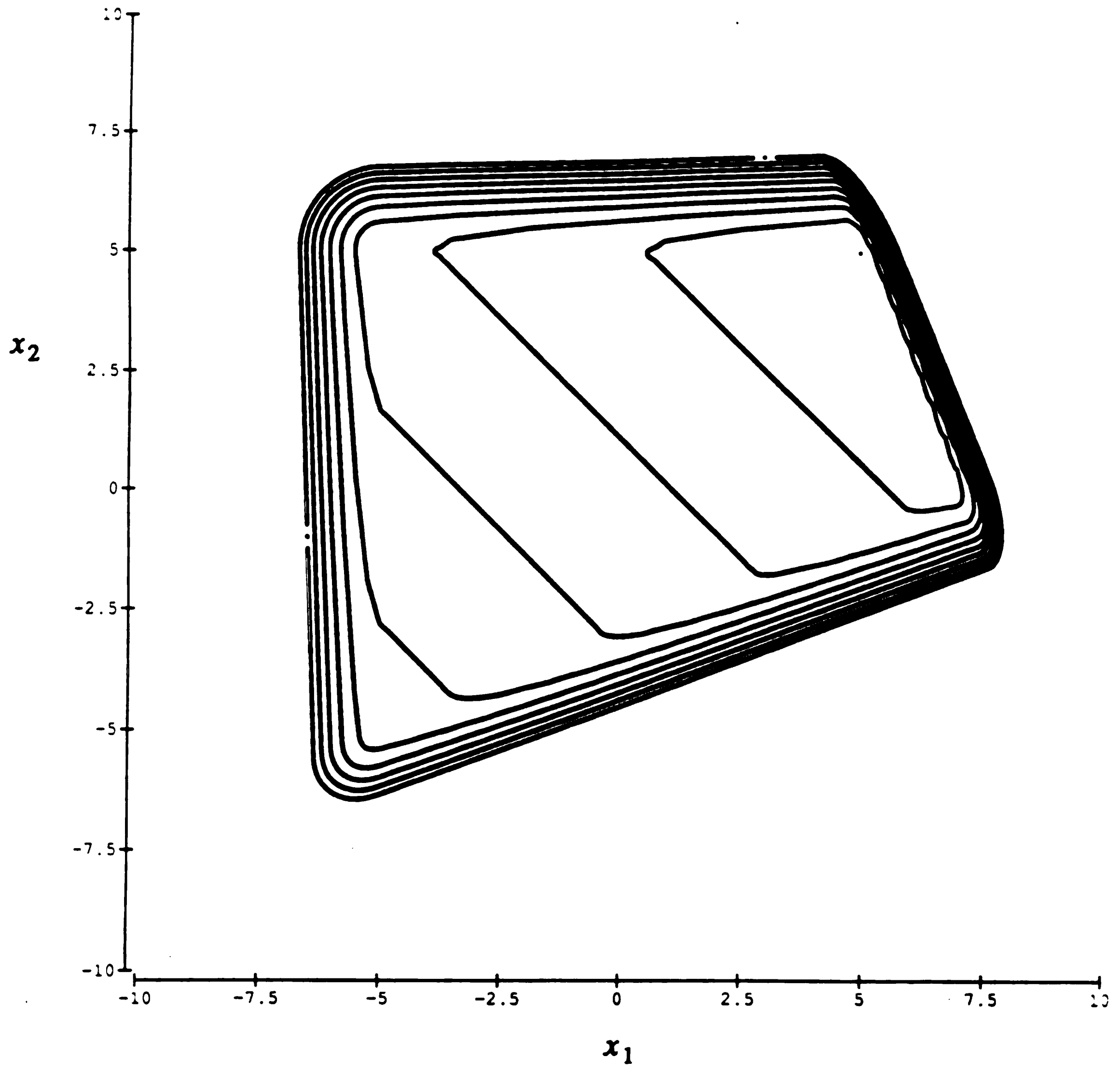


Figure 5.2. Contours of $E_2(x)$ for (LP_1). (a) The contour for s=10.

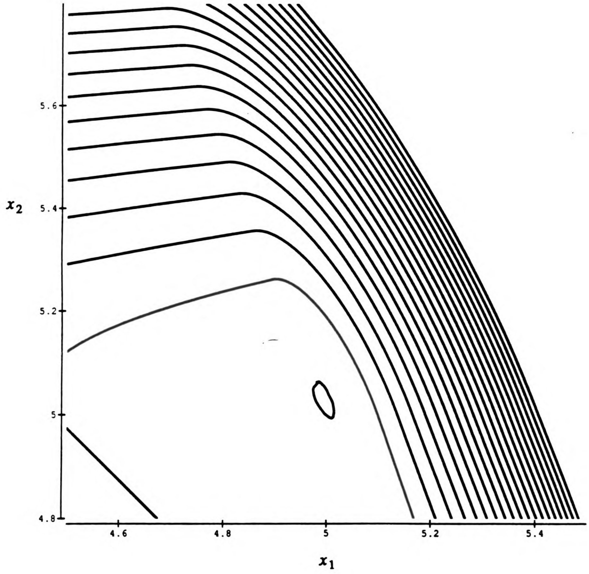


Figure 5.2. (cont'd.) (b) Detailed plot of (a) around $\bar{x}$=(5, 5).

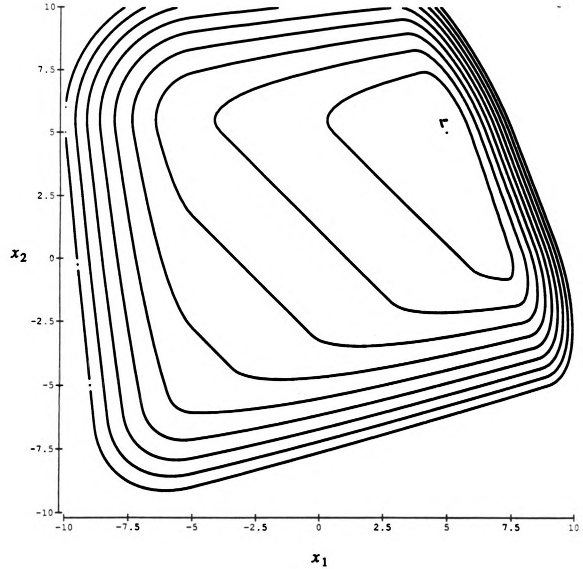


Figure 5.2. (cont'd.) (c) The contour for s=1.

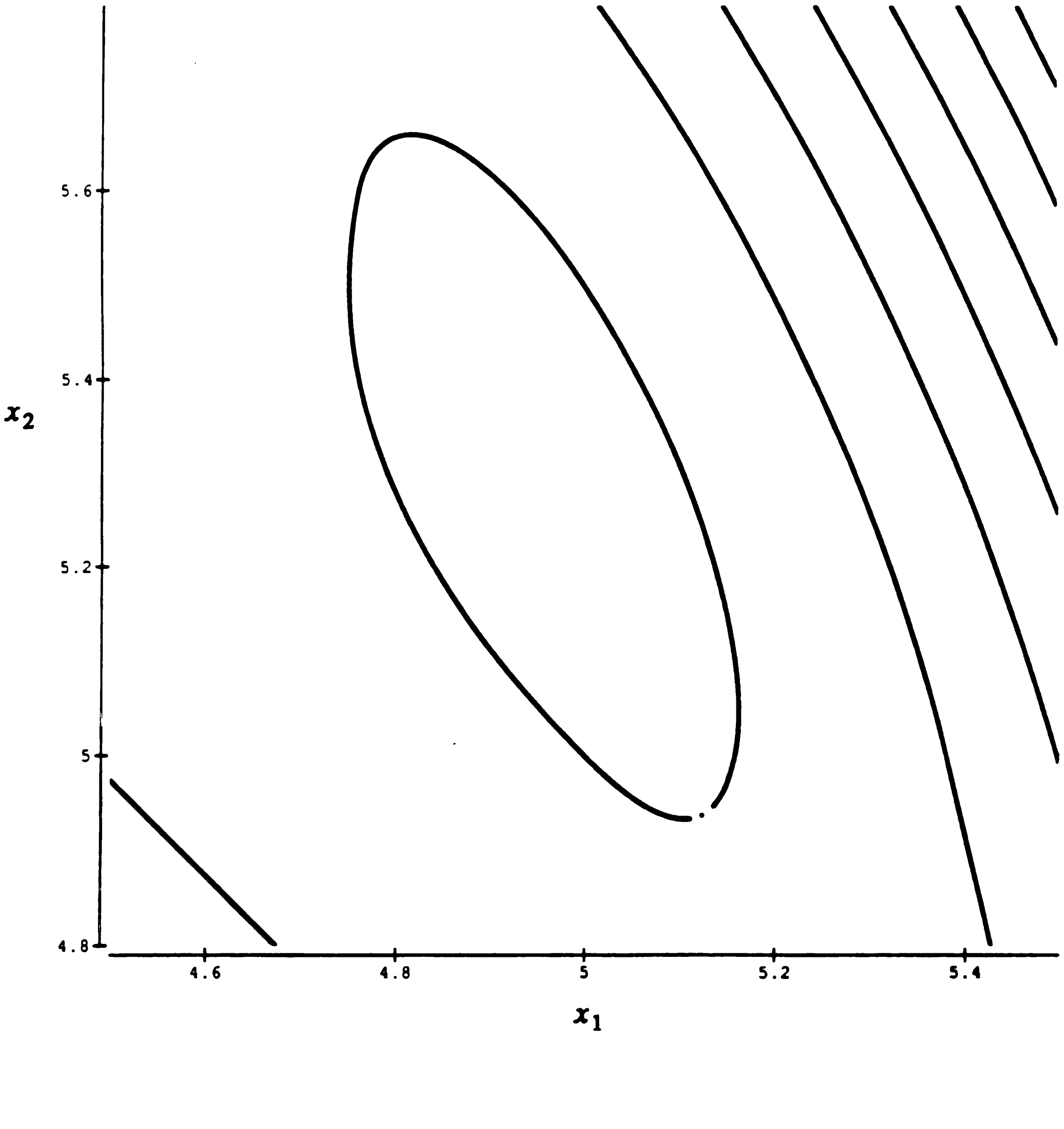


Figure 5.2. (cont'd.) (d) Detailed plot of (c) around $\bar{x}$=(5, 5).

Table 5.1. The equilibria of the network for (LP_1) for different values of s.

Parameter	Equilibrium $\tilde{x}$	$\tilde{x}-\bar{x}$	$sg_j^+(\tilde{x})$
$s=0.2$	(4.600, 8.000)	(0.400, 3.000)	(0.0, 0.4, 0.0, 0.6)
$s=1$	(4.920, 5.600)	(0.080, 0.600)	(0.0, 0.4, 0.0, 0.6)
$s=2$	(4.960, 5.300)	(0.040, 0.300)	(0.0, 0.4, 0.0, 0.6)
$s=10$	(4.992, 5.060)	(0.008, 0.060)	(0.0, 0.4, 0.0, 0.6)

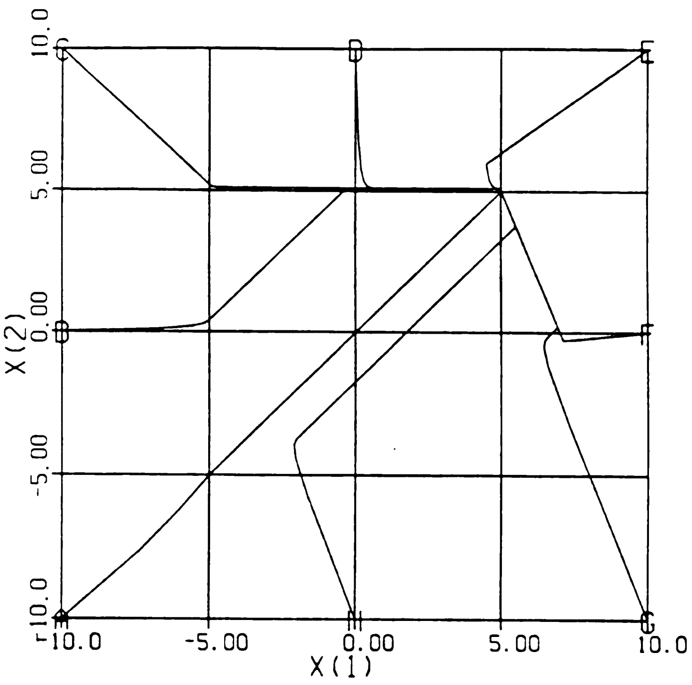
different values of s. It is evident that $\|\tilde{x}-\bar{x}\|$ decreases with respect to the increase of s. It is in fact linear.

To demonstrate the dynamics of equation 4.1 for this problem, trajectories with various initial conditions have been plotted on Figure 5.3 for $s=10$. Figure 5.3(a) illustrates the trajectories converging to the equilibrium $\tilde{x}$=(4.992, 5.060) whereas Figure 5.3(b) shows the trajectories around $\tilde{x}$. The sliding effect of each trajectory along the active constraint can be seen clearly in these figures.

Next, the 2-phase network formulation is applied to this problem. In phase 1, the trajectories are identical to those in Figure 5.3. In phase 2, the trajectories are slowly moving from $\tilde{x}$ to $\bar{x}$ as shown in Figure 5.4(a) with $s=10$ and $\varepsilon=0.2$. The corresponding trajectories of the objective function and $E_2(x)$ with respect to time are given in Figure 5.4(b). It is clear how they approach asymptotically the optimum value -10.0. In Figure 5.4(b) the line atop is the trajectory of $E_2(x)$ which is slightly large than $f(x)$ as long as there are constraint violations.

As mentioned in Section 4.4 if the initial point is in the feasible region, the network structure in phase 2 is sufficient to converge to a minimizer. To see this effect, a simulation using only the phase-2 network structure is done for $s=10$, $\varepsilon=0.2$, and $x_o=[4.8, 4.8]^T$. The simulation results show that the trajectory of x is nearly identical

(a)



(b)

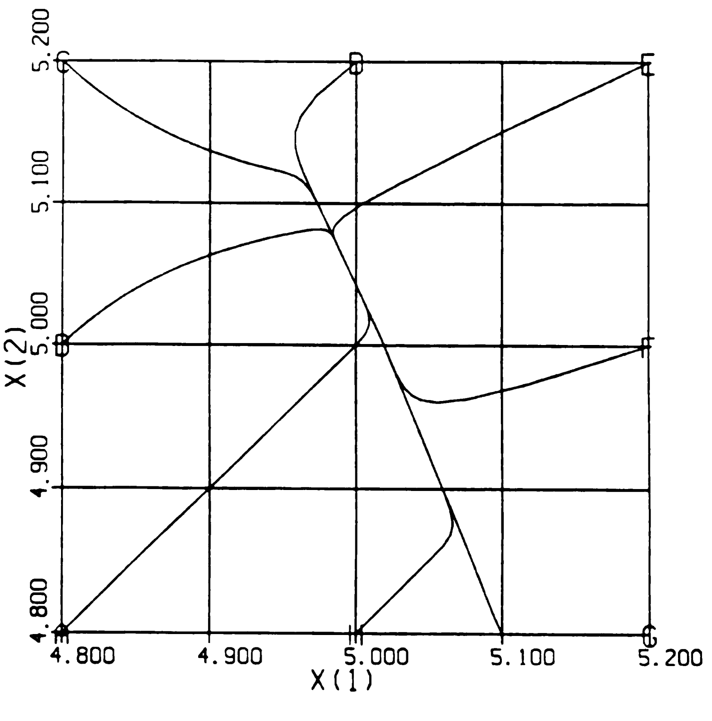
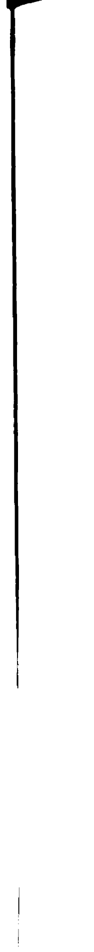
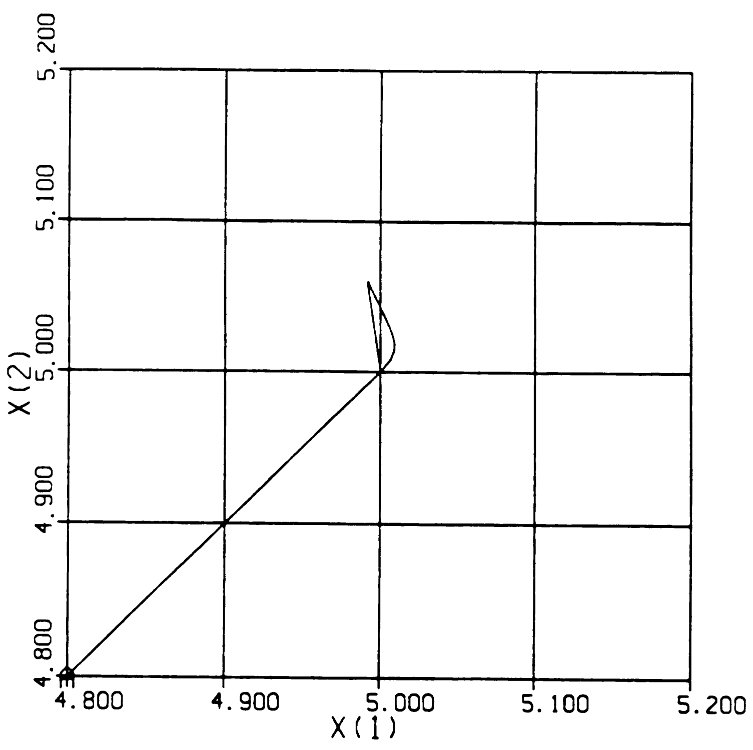


Figure 5.3. Trajectories of equation 4.1 for (LP_1) with $s=10$. (a) A broad view. (b) A view around $\bar{x}$ = (4.992, 5.060).



F
je

(a)



(b)

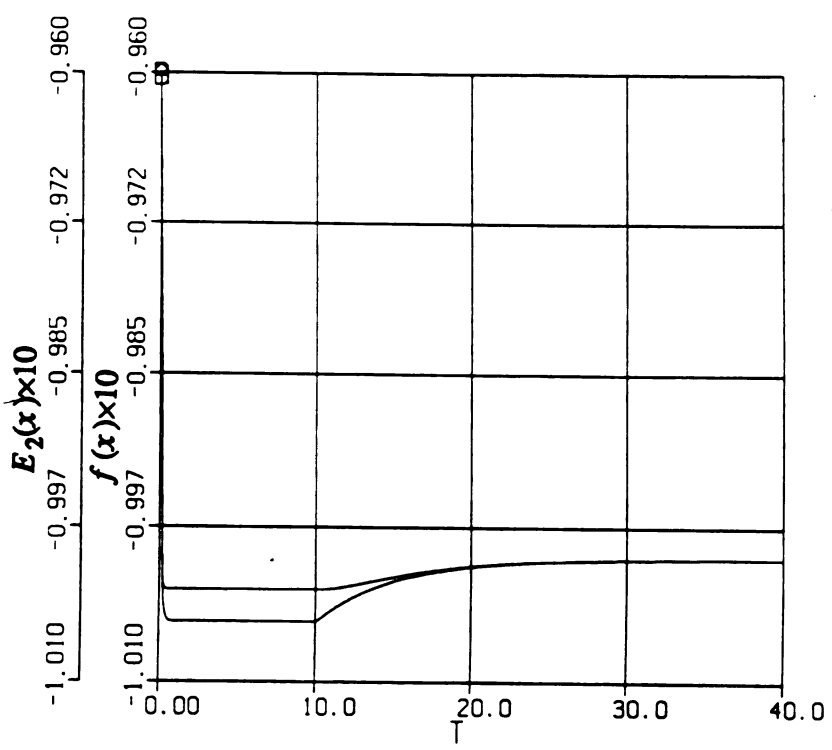


Figure 5.4. Trajectories of the 2-phase network for (LP_1) with $s=10$, $\varepsilon=0.2$. (a) Trajectory of x. (b) Trajectories of $E_2(x)$ and $f(x)$ with respect to time.

to the one in Figure 5.4(a). The corresponding trajectories of $E_2(x)$ and $f(x)$ are given in Figures 5.5(a) and (b), respectively. The lines atop in Figure 5.5 are for the case using only the phase-2 network structure. It is clear that they begin to exhibit the asymptotical behavior only after few generic time steps.

If the initial condition is not in the feasible region, using only the phase-2 network structure may not lead to convergence. This can be seen by that fact that equation 4.40 only increases the value of λ. For fixed s and ε, choosing any initial point from the infeasible region will result in a positive value of λ_i for some i. If the initial point is far enough, λ_i will become larger than the correct Lagrange multiplier before the state trajectory approaches the boundary of the feasible region. Once this happens, there is no way to bring the value of the over-estimated λ_i back down. And, the trajectory remains in the infeasible region since it can not enter into the feasible region except through the minimizer with correct Lagrange multiplier. Eventually, the system diverges.

However, if the initial condition is restricted in a bounded region, it is possible to find a small enough ε such that the phase-2 network structure is stable over this region. But the drawback is that the smaller the value of ε, the slower the rate to equilibrium.

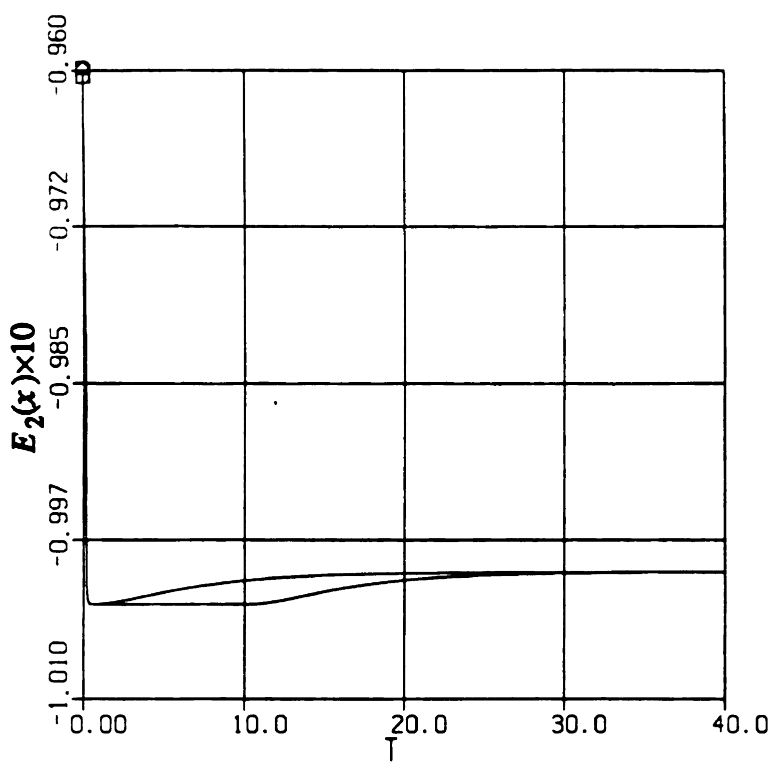
To illustrate the phenomenon described by Fact 4.1, consider the program (LP_2):

$$\text{Minimize } f(x) = -x_2$$

subject to the same constraints as (LP_1). Notice from Figure 5.1 that the minimizers of (LP_2) lie on the line segment $x_2=5$ within the feasible region. If s is chosen to be 2 in equation 4.1, then its equilibrium points are the line segment $x_2=5.5$ within

$$g_2(x) = \frac{5}{2}x_1 + x_2 - \frac{35}{2} \le 0$$

(a)



(b)

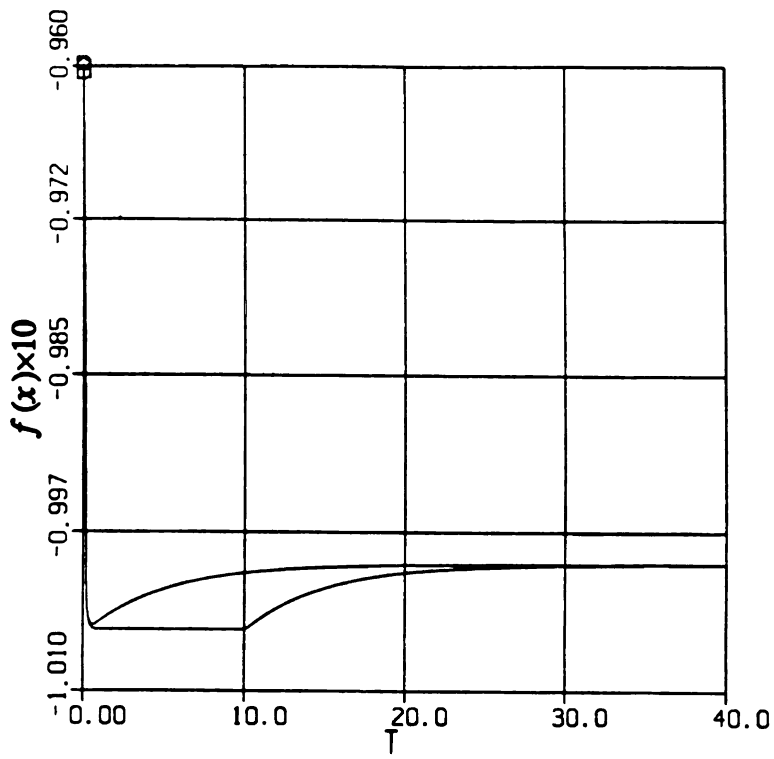


Figure 5.5. Trajectories of $E_2(x)$ and $f(x)$ for phase 2 with s=10, ε=0.2, and x_o=[4.8, 4.8]^T. (a) Trajectory of $E_2(x)$. (b) Trajectory of $f(x)$.

and

$$g_3(x) = -x_1 - 5 \leq 0.$$

In this case, the size of the equilibrium set is smaller than the size of the minimizer set. Trajectories of equation 4.1 near (5.0, 5.5) are shown in Figure 5.6(a). If we raise s from 2 to 10 after the system settles on the line segment x_2=5.5, then all the trajectories will move in parallel to x_2=5.1 as shown in Figure 5.6(b). This is exactly what have been described in Fact 4.1. Note that the trajectories of s=2→10 are different from that of s=10. The latter have been shown in Figure 5.6(c) as contrast to the trajectories in Figure 5.6(b).

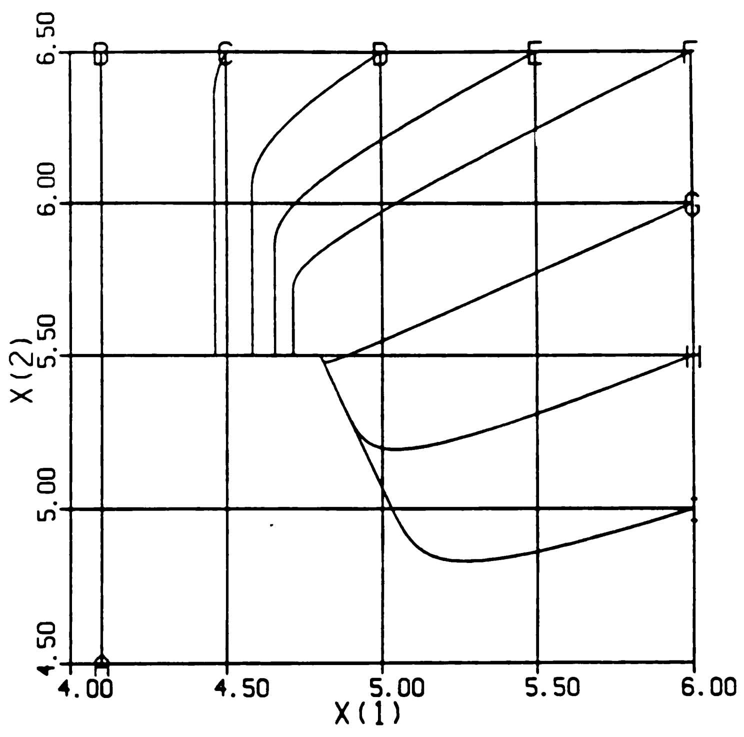
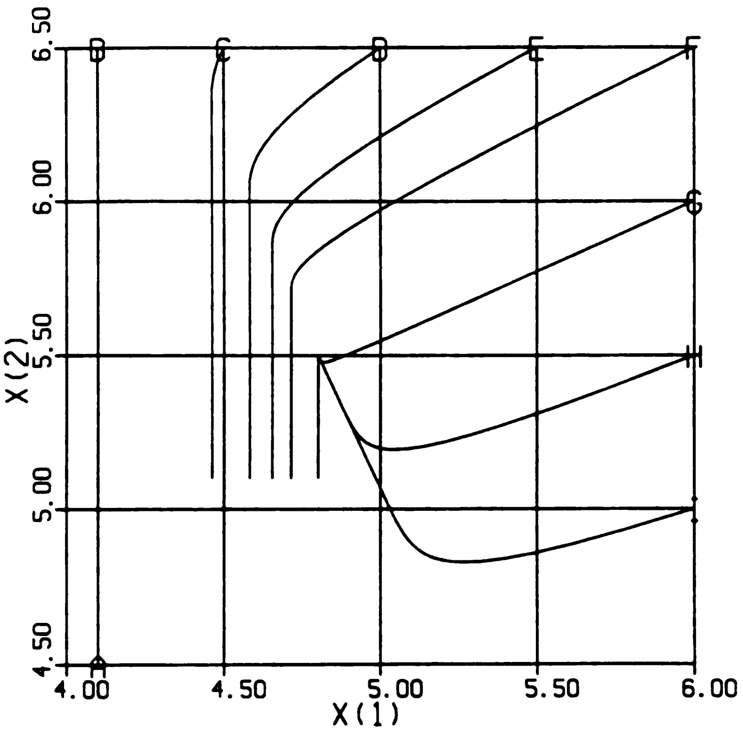


Figure 5.6. Trajectories of equation 4.1 for (LP_2) near (5.0, 5.5). (a) s = 2.

(b)



(c)

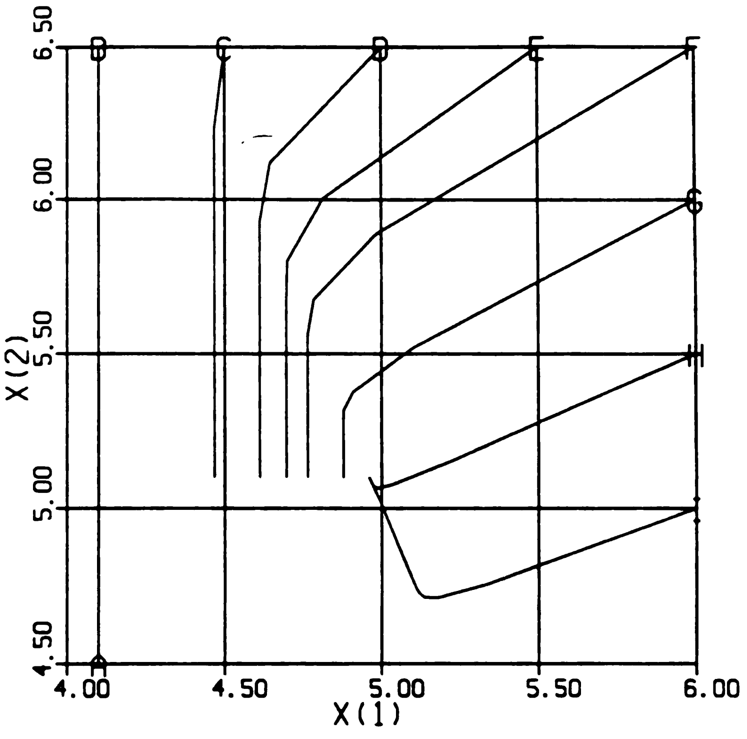


Figure 5.6. (cont'd.) (b) $s = 2 \rightarrow 10$. (c) $s = 10$.

For the case where the size of the equilibrium set is larger than the size of the minimizer set, consider the program (LP_3):

$$\text{Minimize } f(x) = x_1$$

subject to the same constraints as (LP_1). From Figure 5.1 it may be seen that the minimizers of (LP_2) lie on the line segment x_1=–5 within the feasible region. Choosing s=2 in equation 4.1, the equilibrium points are the line segment x_2=–5.5 within

$$g_1(x) = \frac{5}{12}x_1 - x_2 - \frac{35}{12} \le 0$$

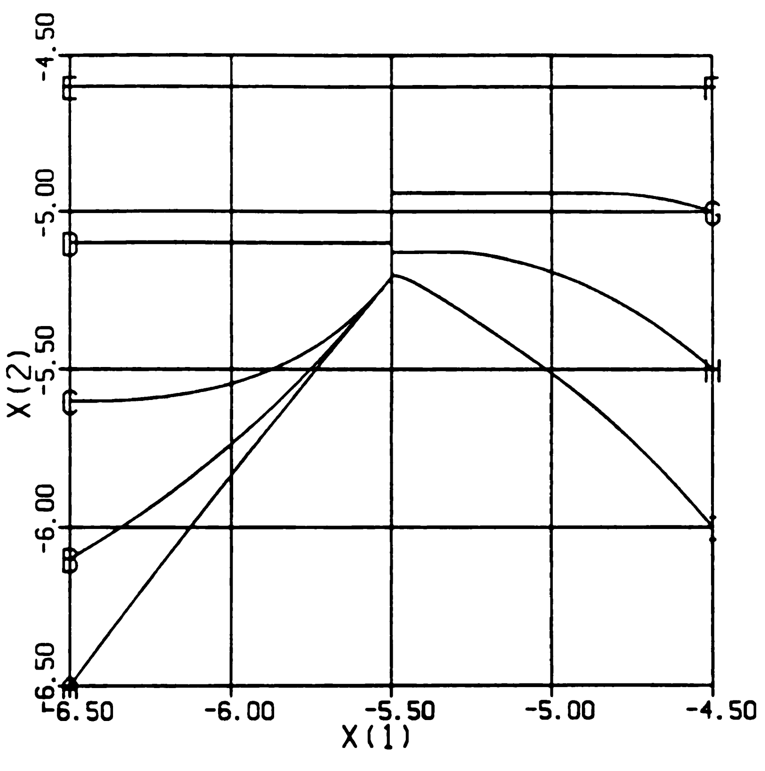
and

$$g_4(x) = x_2 - 5 \le 0.$$

Trajectories of equation 4.1 near (-5.5, -5.5) are shown in Figure 5.7(a). If we raise s from 2 to 10 after the system settles on the line segment x_1=–5.5, then all trajectories will move in parallel to x_1=–5.1 as shown in Figure 5.7(b).

The trajectory curving toward the upper-right in the middle of Figure 5.7(b) is due to the fact that the size of the equilibrium set is reduced as s increases. (In fact, the equilibrium set would eventually be identical to the set of minimizers when s becomes infinity.) This trajectory would have been a straight line along $g_1(x)$=0 if s had been changed continuously. Since $s=2 \rightarrow 10$ abruptly, the early stage of this trajectory tends to move strictly to the right and thus results in new constraint violation as it crosses to the other side of $g_1(x)$=0. Now the asymptotic nature of the system takes place to remedy such a violation and drives the trajectory toward the correct end point of the equilibrium set. The trajectories of equation 4.1 with s=10 are given in Figure 5.7(c) for comparison to the trajectories in Figure 5.7(b).

(a)



(b)

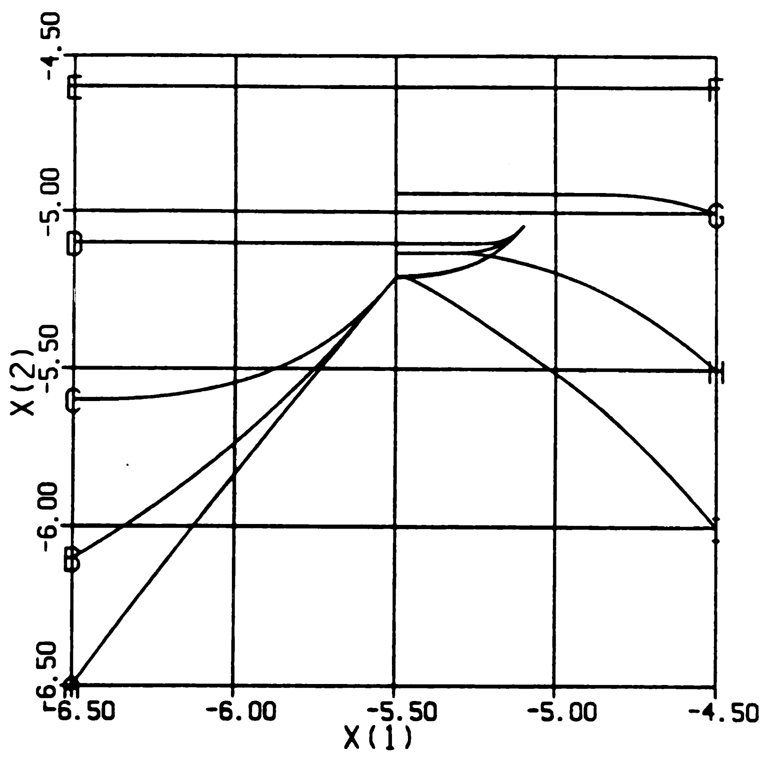


Figure 5.7. Trajectories of equation 4.1 for (LP_3) near the point (-5.5, -5.5).
(a) $s = 2$. (b) $s = 2 \rightarrow 10$.

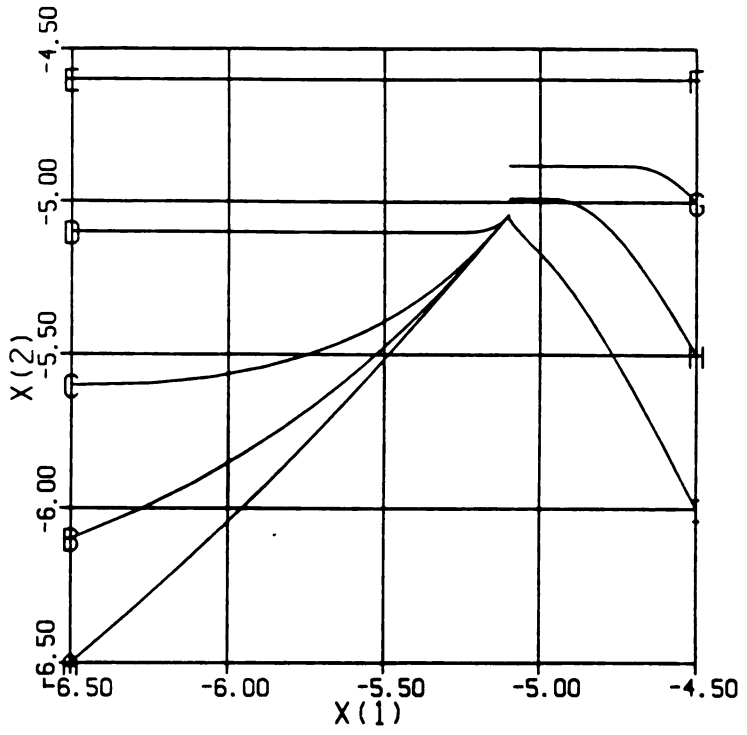


Figure 5.7. (cont'd.) (c) $s = 10$.

5.2 Quadratic Programming

For quadratic programming, consider a program (QP_1):

$$\text{minimize } f(x) = x_1^2 + x_2^2 + x_1x_2 + 3x_1 + 3x_2$$

subject to the same constraints as (LP_1). The minimum of $f(x)$ occurs at $x_1 = x_2 = -1$. Since the unique minimizer lies in the interior of the feasible region, it follows from the theorems derived in Section 4.2 that the unique equilibrium of the (QP) network is exactly the minimizer. The simulation results of the trajectories of x for the correspondingly formed (QP) network are shown in Figure 5.8. The equilibrium $\bar{x} = [-1, -1]^T$ clearly exhibits asymptotic stability. There is no need for using a 2-phase network structure.

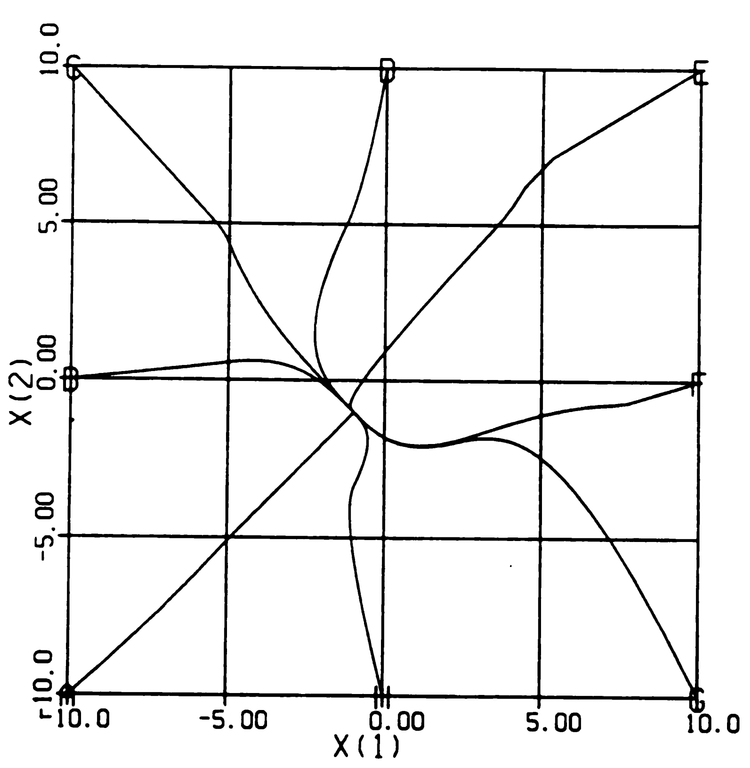


Figure 5.8. Trajectories of x of the (QP) network for (QP_1).

Suppose now the objective function in (QP_1) is replaced by

$$f(x) = x_1^2 + x_2^2 + x_1 x_2 - 30x_1 - 30x_2,$$

and this new program is denoted by (QP_2). The unconstrained minimizer of $f(x)$ of (QP_2) is [10, 10]. The simulation results using equation 4.1 are given in Figure 5.9. Wherever the initial points of x are, the trajectories approach to the equilibrium $\tilde{x} = [4.97778, 5.1745]^T$.

It can be shown that the minimizer of (QP_2) is $\bar{x} = [5.0, 5.0]^T$ at which $I(\bar{x}) = \{2,4\}$. Solving the equation

$$\nabla f(\bar{x}) + \sum_{i \in I(\bar{x})} \lambda_i \nabla g_i(\bar{x}) = 0,$$

we get the corresponding Lagrange multipliers, λ_1=0.0, λ_2=6.0, λ_3=0.0, and λ_4=9.0.

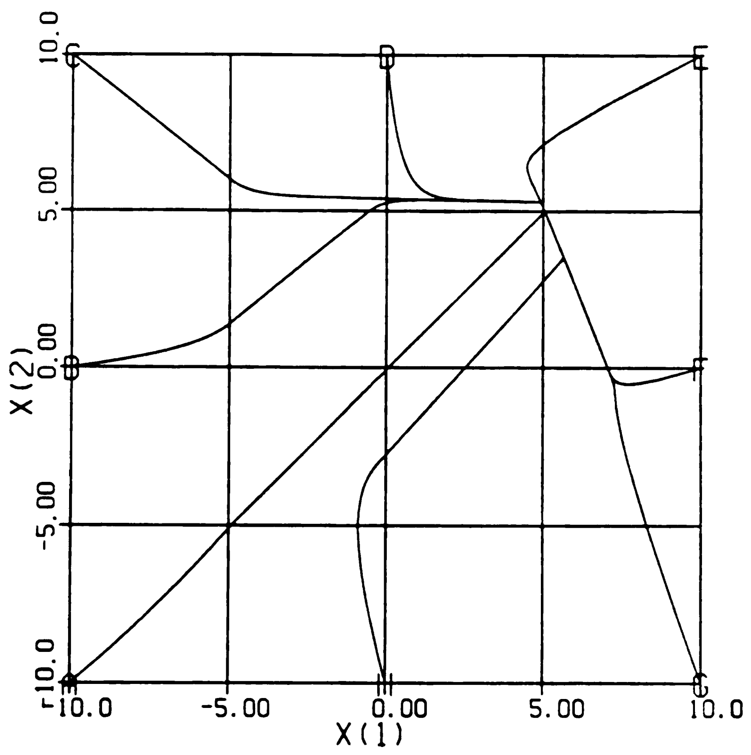


Figure 5.9. Trajectories of x of equation 4.1 for (QP_2) with s=50.

Using the 2-phase network formulation for (QP_2), a simulation is run with $s=50$, $\varepsilon=0.2$, and initial condition $x_o=[4.8, 4.8]^T$. The trajectories of x and λ are shown in Figure 5.10(a) and (b), respectively. The network is switched from phase 1 to phase 2 at T=2.0. Figure 5.10 clearly shows the dynamics of the network during phase 2 in which trajectories moves asymptotically toward the equilibrium $\bar{x} = [5.0, 5.0]^T$ and $\lambda = [0.0, 6.0, 0.0, 9.0]^T$.

To make the example more representative, add one equality constraint $x_1 = 3$ to (QP_1), and call this new program (QP_3). In the network formulation this equality constraint is replaced by

$$g_5(x) = x_1 - 3 \leq 0$$

$$g_6(x) = -x_1 + 3 \leq 0$$

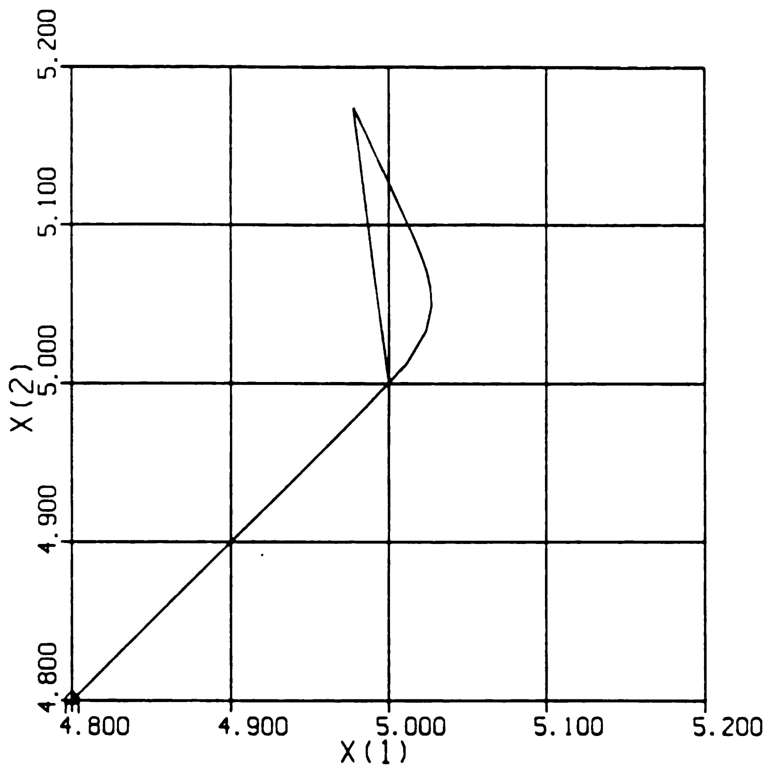
The minimizer of (QP_3) is $\bar{x} = [3, -\frac{5}{3}]^T$ at which $I(\bar{x}) = \{1,6\}$. The theoretical values of Lagrange multipliers are

$$\lambda = [\frac{8}{3}, 0, 0, 0, 0, \frac{76}{9}]^T .$$

Simulation results for the network of equation 4.1 for (QP_3) are shown in Figure 5.11(a) for $s=50$ and $\varepsilon=0.2$. All trajectories lead toward the equilibrium point $\tilde{x} = [2.84279, -1.77791]^T$. For the correspondingly formulated 2-phase network, a simulation is performed with the initial condition $x_o = [2.5, -1]^T$ and the same s and ε. The resulting trajectory of x is given in Figure 5.11(b). Again, the 2-phase network demonstrates its capability to tune the state variable x to the exact minimizer. Though not shown in figure, the final values of $\lambda(t)$ for the 2-phase network are the exact Lagrange multiplers described above.

As mentioned in Section 4.2 one of the applications of the quadratic programming optimization network is for solving the least squares problem $Bx=b$. For

(a)



(b)

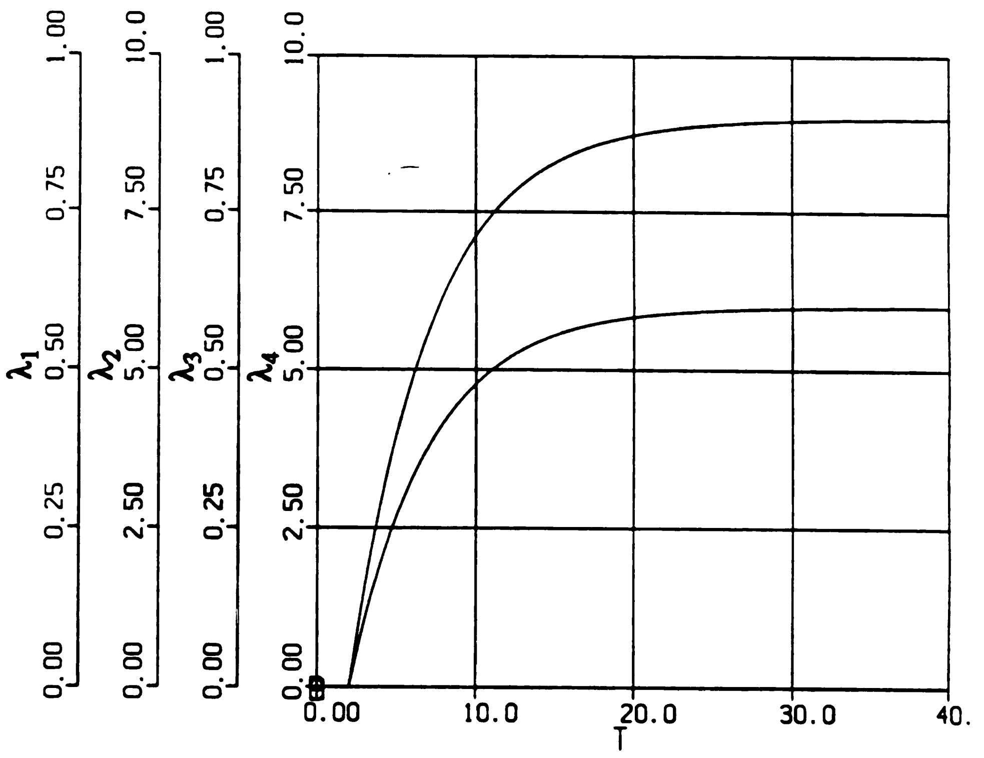
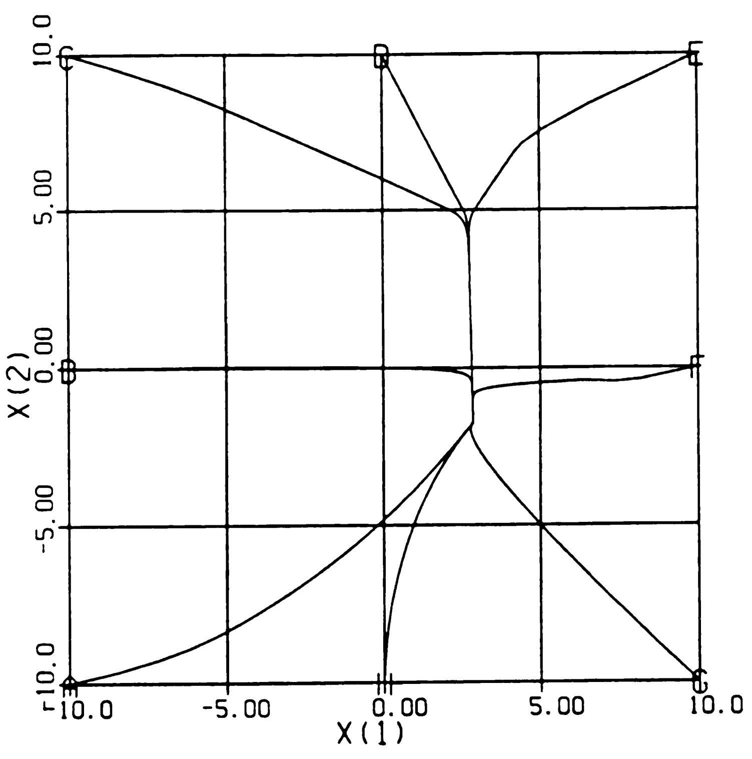


Figure 5.10. Simulation results of the 2-phase network for (QP_2) with s=50 and ε=0.2. (a) The trajectory of x. (b) The trajectories of λ_i.

(a)



(b)

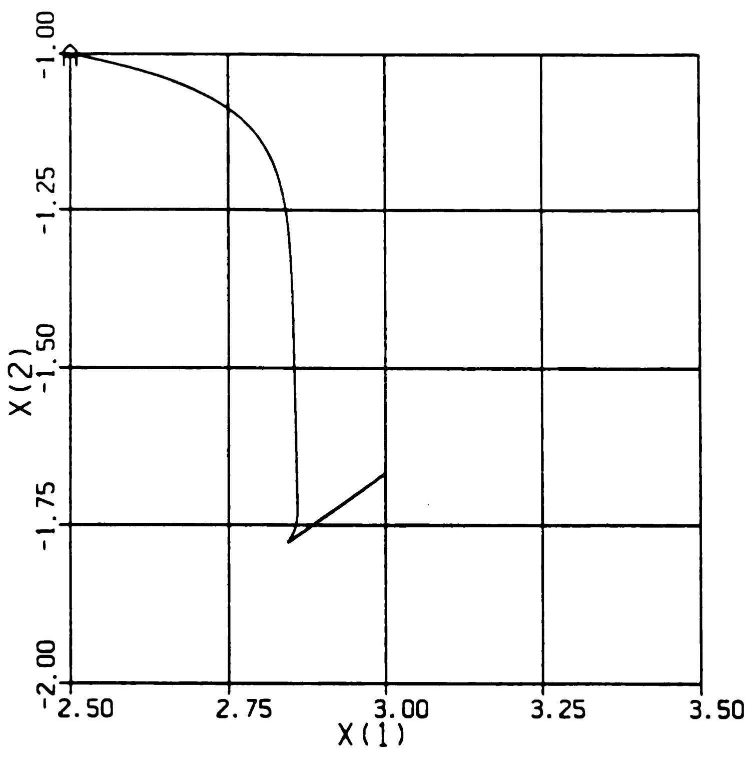


Figure 5.11. Trajectories for (QP_3) with s=50 and ε=0.2. (a) Using the network of equation 4.1. (b) Using the 2-phase network.

illustration sake, consider the following least squares problem (*LS*) for which

$$B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & -1 \end{bmatrix},$$

and

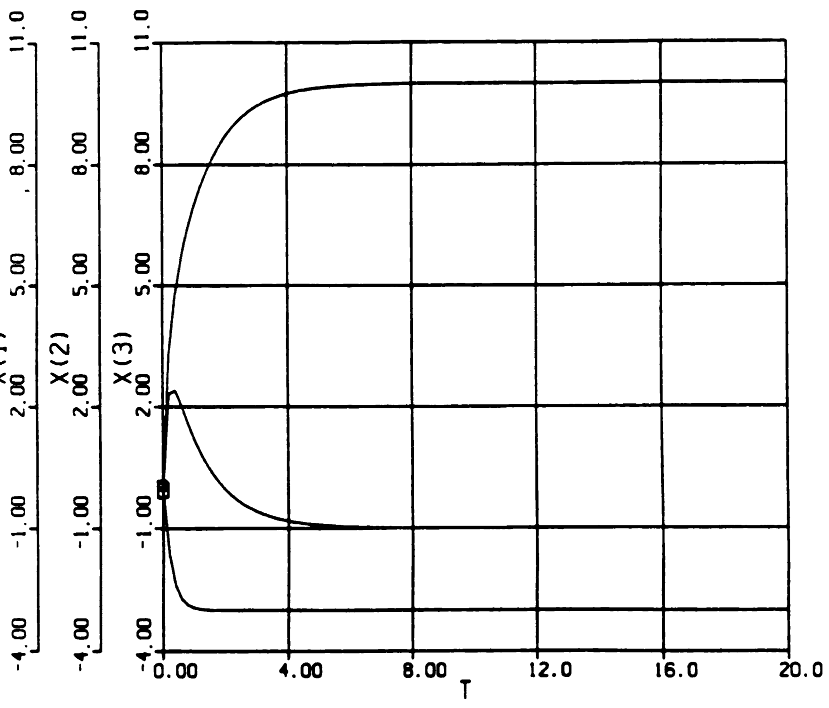
$$b = \begin{bmatrix} 6 \\ 14 \\ 7 \\ -3 \\ 1 \end{bmatrix}.$$

The least squares estimator calculated by the *normal equation* is

$$\tilde{x} = [B^T B]^{-1} B^T b = \begin{bmatrix} -1 \\ 10 \\ -3 \end{bmatrix}.$$

Since there is no constraint in this problem, $E_2(x) = f(x) = \frac{1}{2}\|Bx-b\|^2$. Also since B is of full rank, $E_2(x)$ is strictly convex and the unique equilibrium $\tilde{x}=[-1, 10, -3]^T$ of equation 4.32 is globally asymptotically stable. A simulation with initial condition $x=[0, 0, 0]^T$ has been performed and the trajectories of x and of $E_2(x)$ are plotted in Figure 5.12. Note that the x_j, $1 \le j \le n$, does not approach $\tilde{x}_j$ in a monotonic (increasing or decreasing) manner as seen in Figure 5.12(a). But the network does converge monotonically in the sense of $E_2(x)$ (see Figure 5.12(b)). More importantly, even though the network has not yet converged to its equilibrium, $E_2(x)$ becomes very close to its final value in only few generic time steps. Eight more simulations with various initial conditions were done to vindicate the stability of $\tilde{x}$ and the results are shown in Figures 5.13(a)-(h).

(a)



(b)

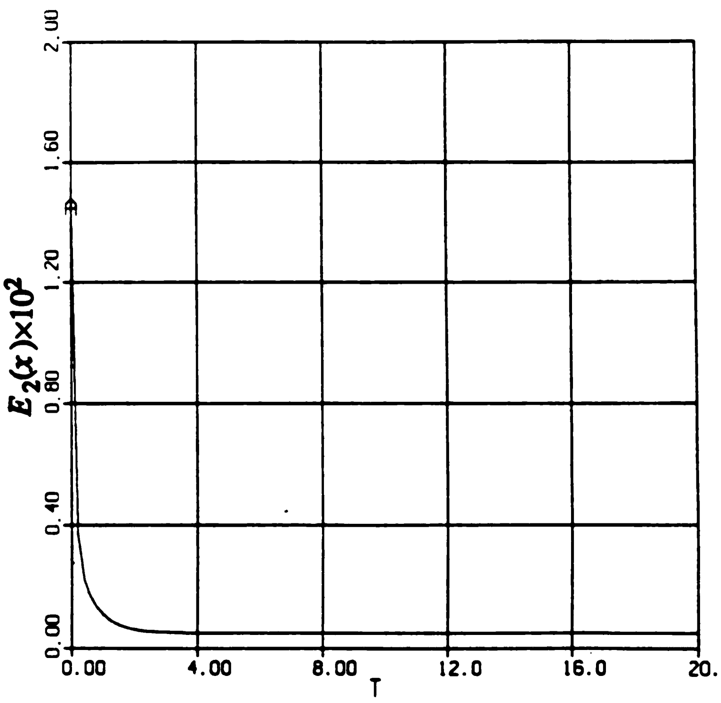
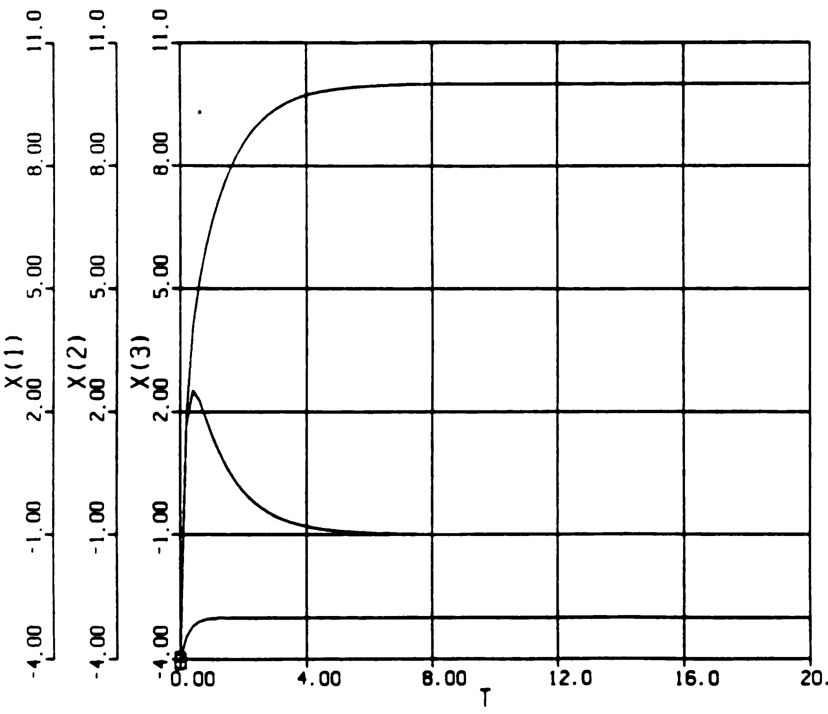


Figure 5.12. The simulation results of the (LS) problem with initial condition $x_o=[0, 0, 0]^T$. (a) The trajectories of the state variables. (b) The trajectory of $E_2(x)$.

(a)



(b)

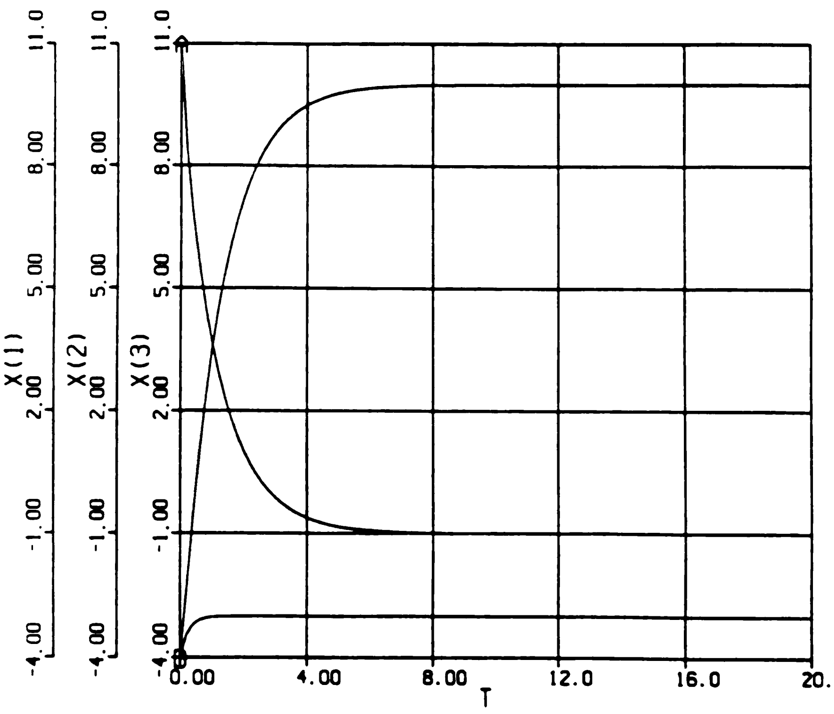
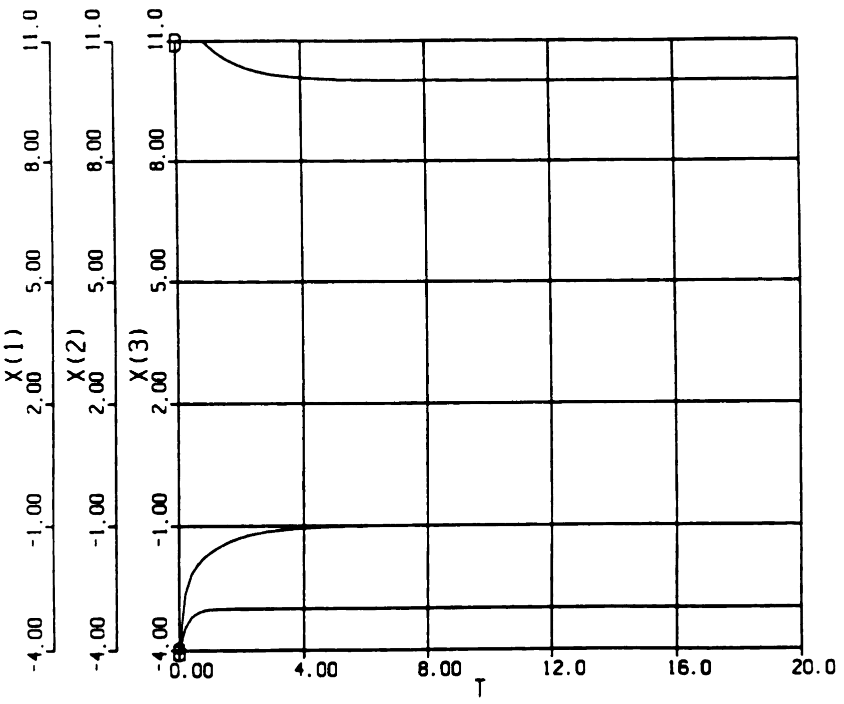


Figure 5.13. The trajectories of x for the (LS) problem with different initial conditions. (a) x_o=[-4, -4, -4]^T. (b) x_o=[11, -4, -4]^T.

(c)



(d)

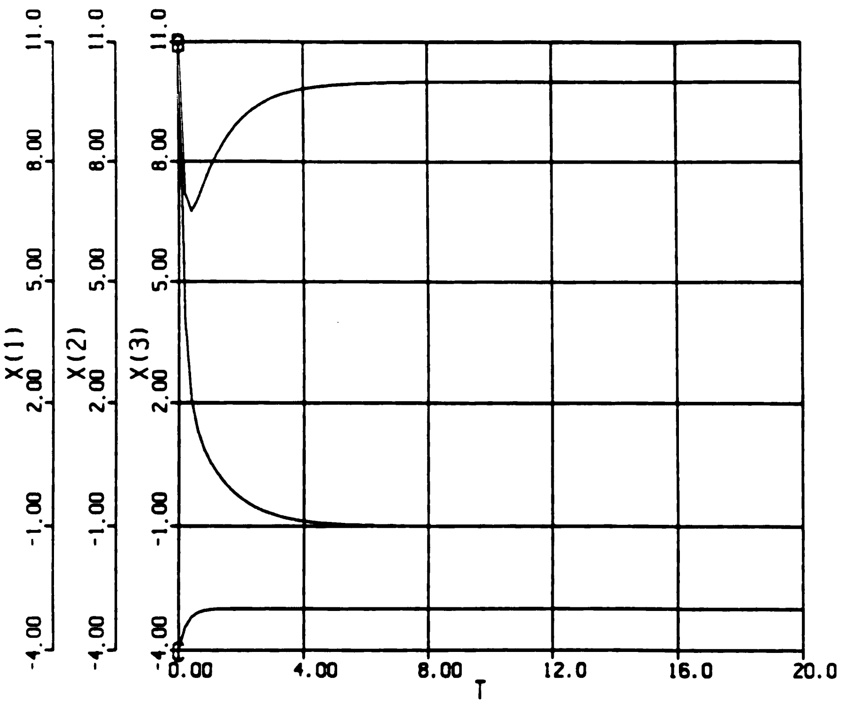
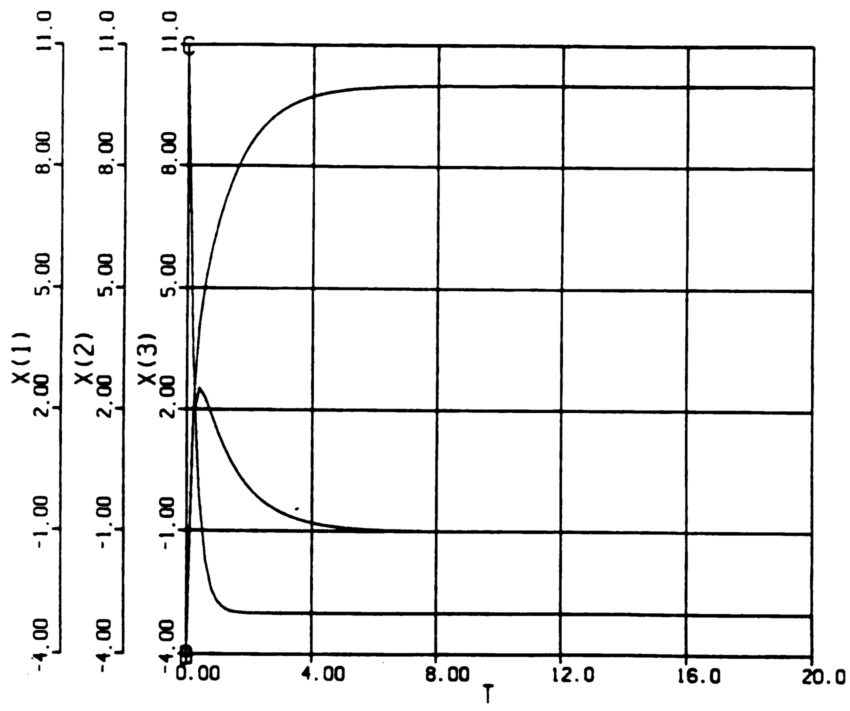


Figure 5.13. (cont'd.). (c) $x_o=[-4, 11, -4]^T$. (d) $x_o=[11, 11, -4]^T$.

(e)



(f)

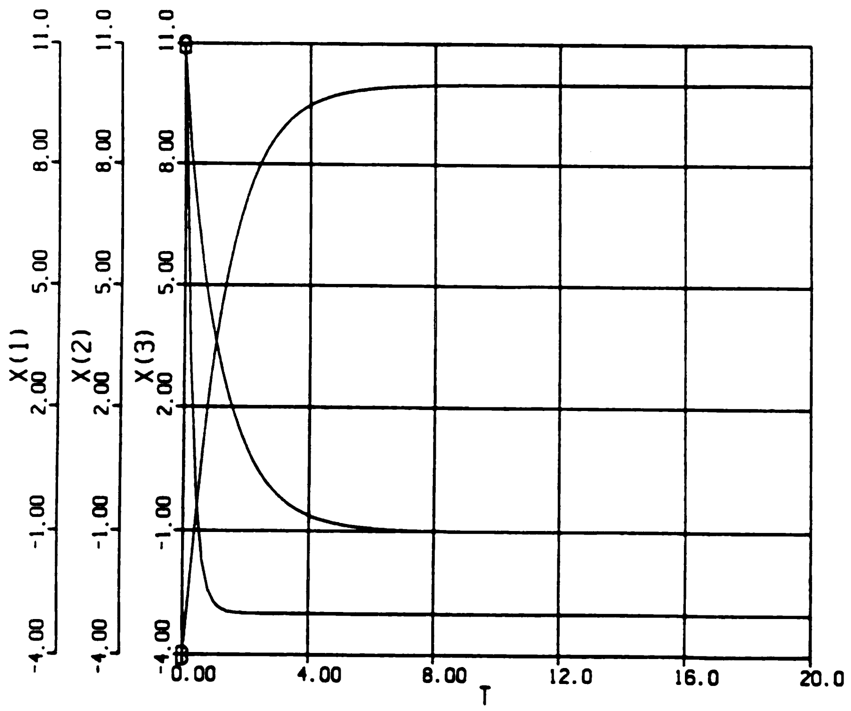
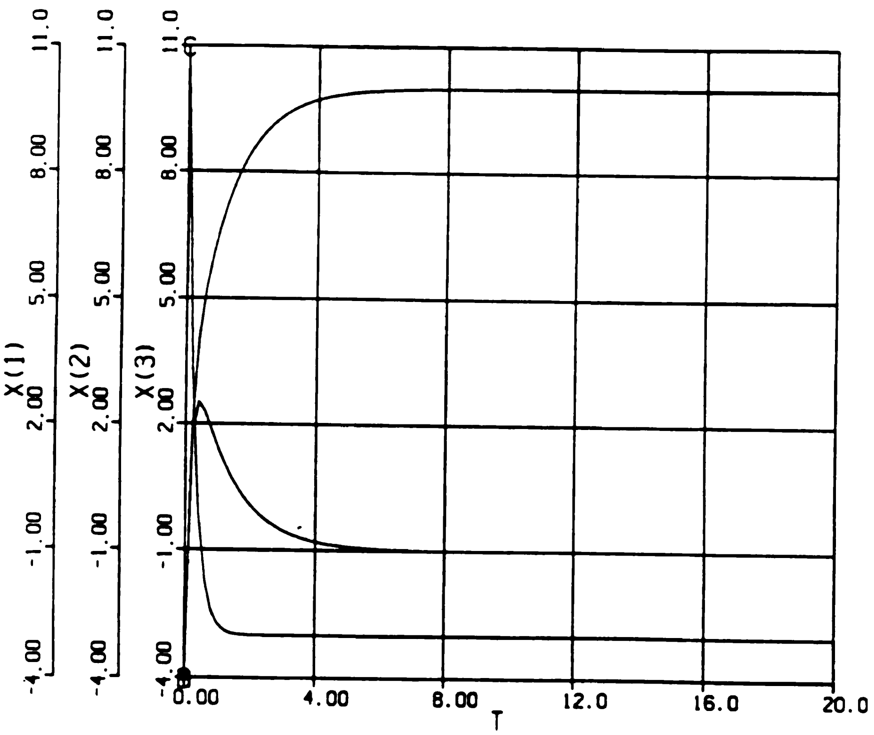


Figure 5.13. (cont'd.). (e) $x_o=[-4, -4, 11]^T$. (f) $x_o=[11, -4, 11]^T$.

(e)



(f)

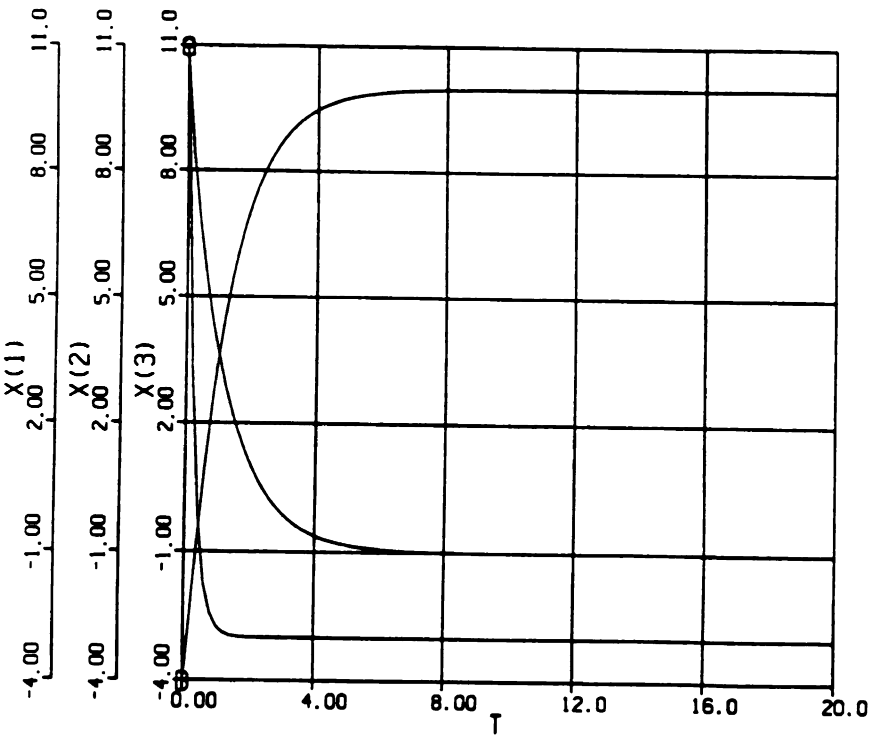
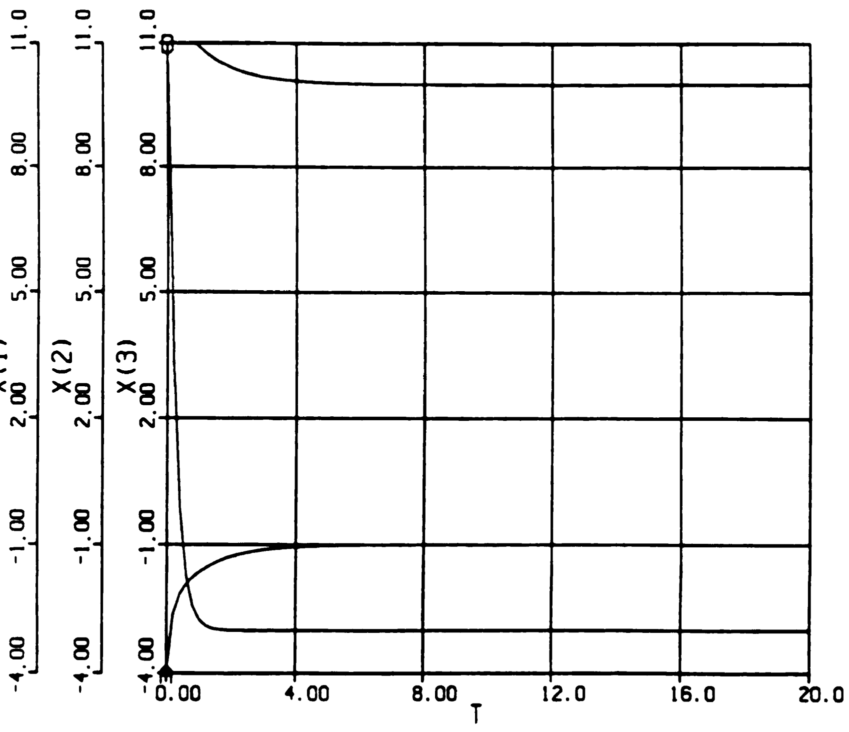


Figure 5.13. (cont'd.). (e) $x_o=[-4, -4, 11]^T$. (f) $x_o=[11, -4, 11]^T$.

(g)



(h)

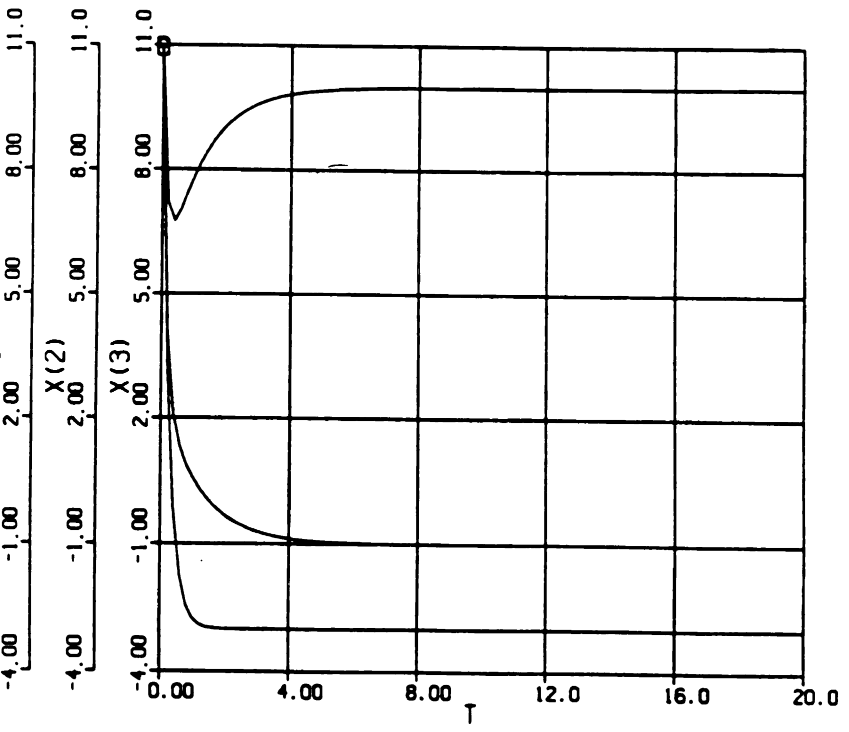


Figure 5.13. (cont'd.). (g) x_o=[-4, 11, 11]^T. (h) x_o=[11, 11, 11]^T.

5.3 Nonlinear Programming

Consider the following program (NP_1):

$$\text{Minimize } f(x) = x_1^2 + (x_2 - 1)^2$$

$$\text{subject to } g(x) = x_2 - x_1^2 = 0.$$

Note that $f(x)$ is strictly convex on R^2 as shown by the following derivation. For $\lambda \in (0, 1)$,

$$f(\lambda x + (1-\lambda)y)$$

$$= (\lambda x_1 + (1-\lambda)y_1)^2 + (\lambda x_2 + (1-\lambda)y_2 - 1)^2$$

$$= (\lambda x_1 + (1-\lambda)y_1)^2 + [\lambda(x_2-1) + (1-\lambda)(y_2-1)]^2$$

$$= \lambda^2 x_1^2 + 2\lambda(1-\lambda)x_1y_1 + (1-\lambda)^2 y_1^2 + \lambda^2(x_2-1)^2 + 2\lambda(1-\lambda)(x_2-1)(y_2-1)$$

$$= \lambda^2[x_1^2 + (x_2-1)^2] + (1-\lambda)^2[y_1^2 + (y_2-1)^2] + 2\lambda(1-\lambda)[x_1y_1 + (x_2-1)(y_2-1)]$$

$$= \lambda[x_1^2 + (x_2-1)^2] + (\lambda^2-\lambda)[x_1^2 + (x_2-1)^2] + (1-\lambda)[y_1^2 + (y_2-1)^2]$$

$$= [(1-\lambda)^2 - (1-\lambda)][y_1^2 + (y_2-1)^2] + 2\lambda(1-\lambda)[x_1y_1 + (x_2-1)(y_2-1)]$$

$$= \lambda f(x) + (1-\lambda)f(y) + \lambda(\lambda-1)\left[x_1^2+(x_2-1)^2+y_1^2+(y_2-1)^2-2x_1y_1-2(x_2-1)(y_2-1)\right]$$

$$= \lambda f(x) + (1-\lambda)f(y) + \lambda(\lambda-1)\left[(x_1 - y_1)^2 + ((x_2-1) - (y_2-1))^2\right]$$

$$< \lambda f(x) + (1-\lambda)f(y),$$

since $\lambda(\lambda-1) < 0$ and $(x_1 - y_1)^2 + ((x_2-1) - (y_2-1))^2 > 0$. To verify the convexity of $g(x)$, by definition it follows that

$$g(\lambda x + (1-\lambda)y)$$

$$= (\lambda x_2 + (1-\lambda)y_2) - (\lambda x_1 + (1-\lambda)y_1)^2$$

$$= \lambda x_2 + (1-\lambda)y_2 - [\lambda^2 x_1^2 + 2\lambda(1-\lambda)x_1y_1 + (1-\lambda)^2 y_1^2]$$

$$= \lambda(x_2 - x_1^2) + (1-\lambda)(y_2 - (1-\lambda)y_1^2) - 2\lambda(1-\lambda)x_1y_1$$

$$= \lambda g(x) + (1-\lambda)g(x) - 2\lambda(1-\lambda)x_1y_1$$

$$\leq \lambda g(x) + (1-\lambda)g(x),$$

if $x_1y_1 \geq 0$. Thus $g(x)$ is convex on the closed half spaces $\{x \in R^2 | x_1 \geq 0\}$ and $\{x \in R^2 | x_1 \leq 0\}$, but not on R^2.

The program (NP_1) is equivalent to finding points on the parabola $x_1^2 = x_2$ closest to the point (0, 1). To solve the problem precisely, substitute $x_1^2 = x_2$ into $f(x)$ and solve

$$x_2 + (x_2 - 1)^2 = c.$$

This gives

$$x_2 = \frac{1 \pm \sqrt{4c - 3}}{2}.$$

It can be seen geometrically that there is only one solution of x_2. This implies $c_{\min} = \frac{3}{4}$ and $x_2 = \frac{1}{2}$. Correspondingly, $x_1 = \pm\frac{1}{\sqrt{2}}$. Thus the minimizer of (NP_1) is

$$x = (\pm\frac{1}{\sqrt{2}}, \frac{1}{2})$$

at which $f(x) = \frac{3}{4}$.

$E_2(x)$ for the above program (NP_1) is

$$E_2(x) = x_1^2 + (x_2 - 1)^2 + \frac{s}{2}(x_2 - x_1^2)^2.$$

The contours of $E_2(x)$ for different values of s are shown in Figure 5.14. When s is small, the shape of the contour of $E_2(x)$ is dominated by $f(x)$. As s increases, it tends toward the shape of the parabola $g(x)$ with two minimizers

$$(\pm\sqrt{\frac{s-2}{2s}}, \frac{1}{2}),$$

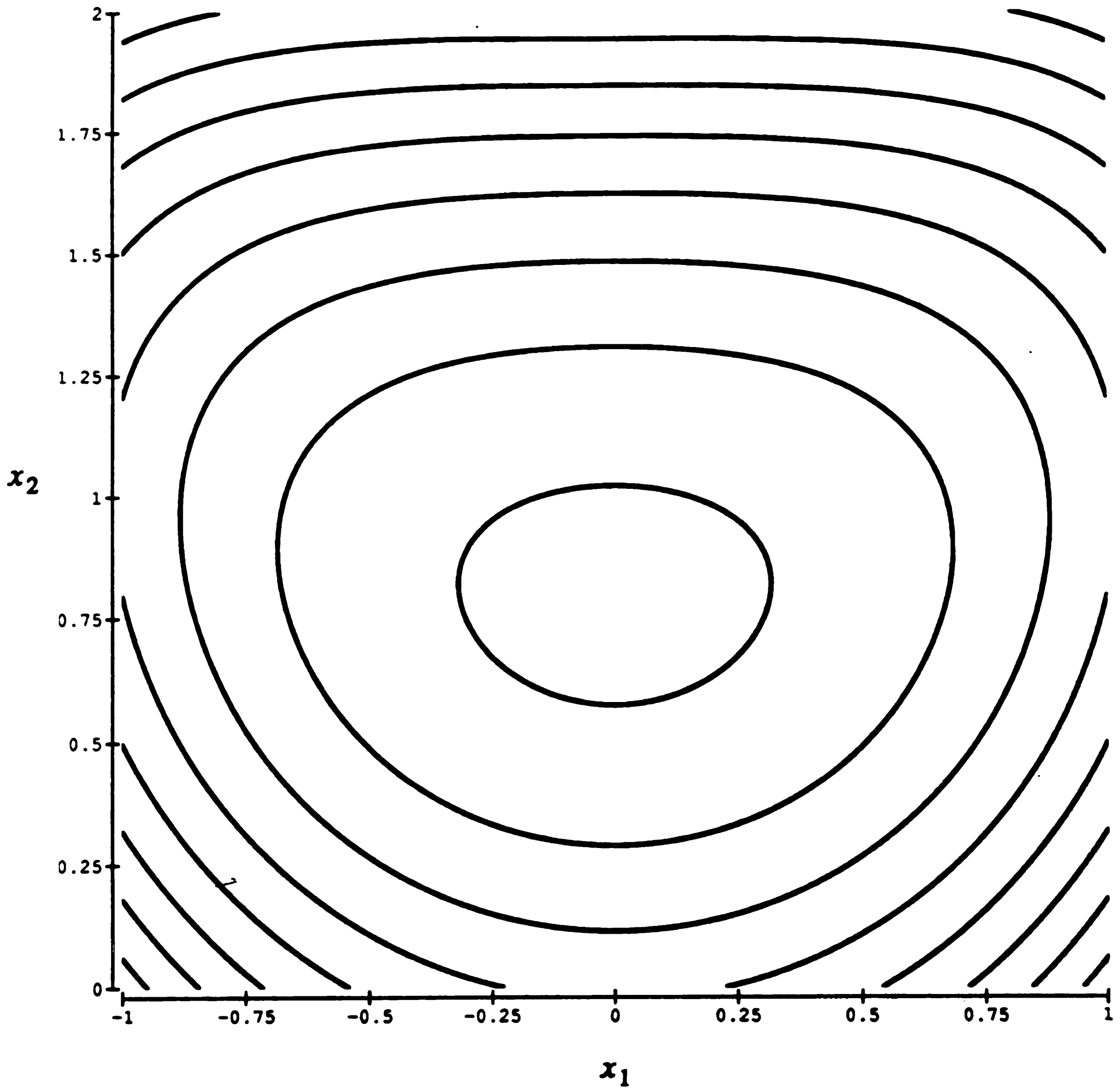


Figure 5.14. Contours of $E_2(x)$ for (NP_1). (a) The contour for s=0.5.

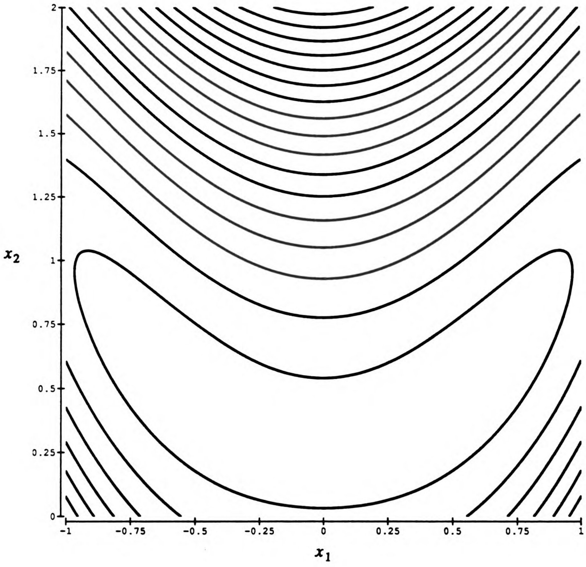


Figure 5.14. (cont'd.) (b) The contour for $s=5$.

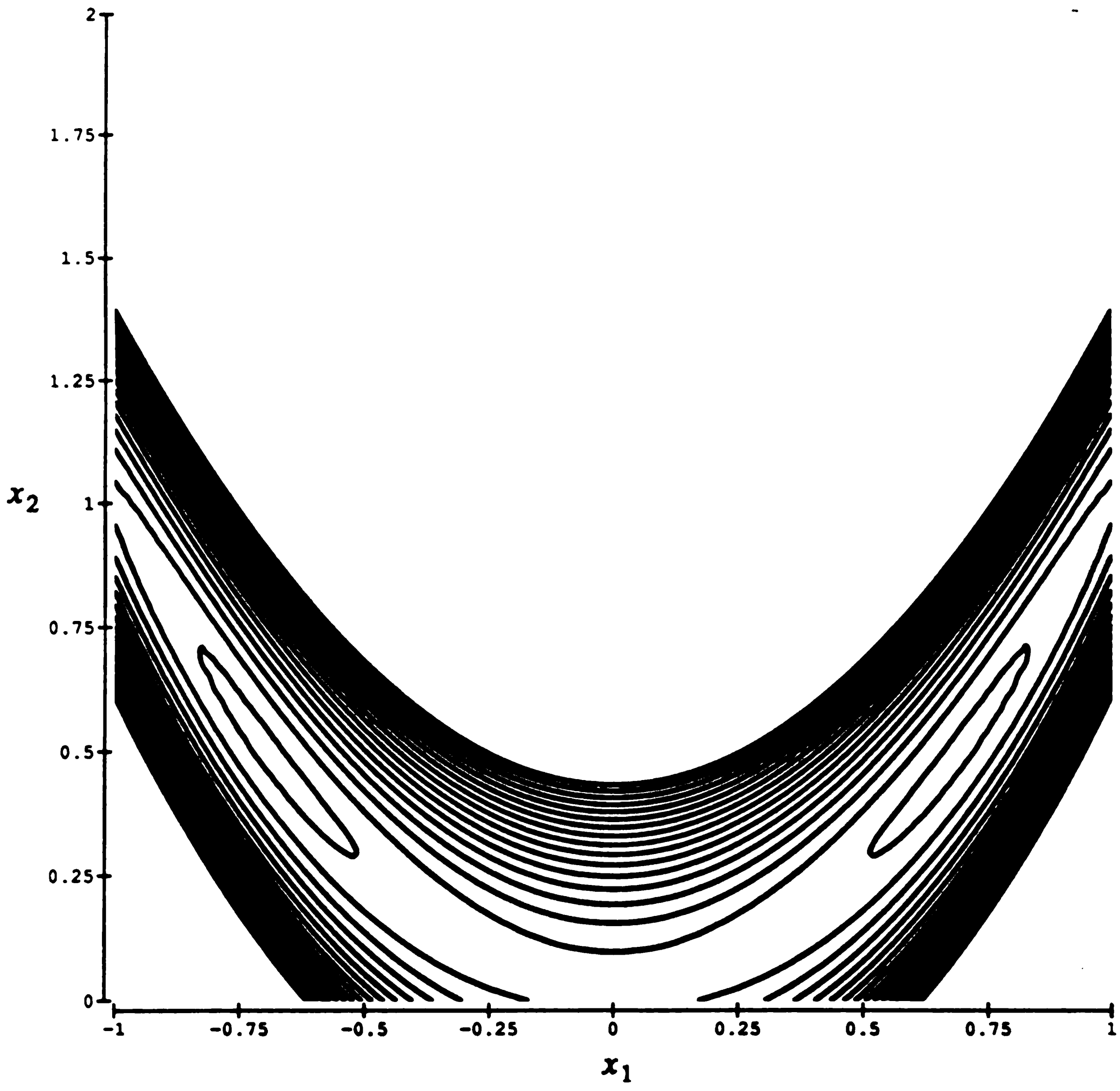


Figure 5.14. (cont'd.) (c) The contour for s=50.

derived in the following.

By direct calculation, it follows that

$$\nabla E_2(x) = \begin{bmatrix} 2x_1(sx_1^2 - sx_2 + 1) \\ -sx_1^2 + (s+2)x_2 - 2 \end{bmatrix}$$

and

$$\nabla^2 E_2(x) = \begin{bmatrix} 2+s\,(6x_1^2 - 2x_2) & -2sx_1 \\ -2sx_1 & s+2 \end{bmatrix}.$$

By setting $\nabla E_2(x) = 0$ the critical points of $E_2(x)$ are found to be

$$x_{e1} = (0, \frac{2}{s+2}) \text{ and } x_{e2} = (\pm\sqrt{\frac{s-2}{2s}}, \frac{1}{2}).$$

For $x = x_{e1}$, the eigenvalues of $\nabla^2 E_2(x)$ are

$$\frac{4-2s}{s+2} \text{ and } (s+2)$$

and they are positive for $s<2$. Thus $x = x_{e1}$ is the minimizer for $E_2(x)$ when $s<2$. Similarly, the eigenvalues of $\nabla^2 E_2(x)$ at $x = x_{e2}$ are

$$\frac{(3s+2) \pm\sqrt{(3s+2)^2 - 4(4s-8)}}{2} > 0$$

for $s>2$. Hence, x_{e2} is the minimizer for $E_2(x)$ when $s>2$. As $s \to \infty$, the minimizer x_{e2} approaches $(\pm\frac{1}{\sqrt{2}}, \frac{1}{2})$, which are the exact minimizers of (NP_1).

Converting the program (NP_1) into the form of equation 4.1, the simulation results of the trajectories of x with s=50 and various initial points are given in Figure 5.15. The trajectories of x, except for initial points with x_1=0, converge to either

$$\tilde{x}_1 = (0.69282, 0.5) \text{ or } \tilde{x}_2 = (-0.69282, 0.5)$$

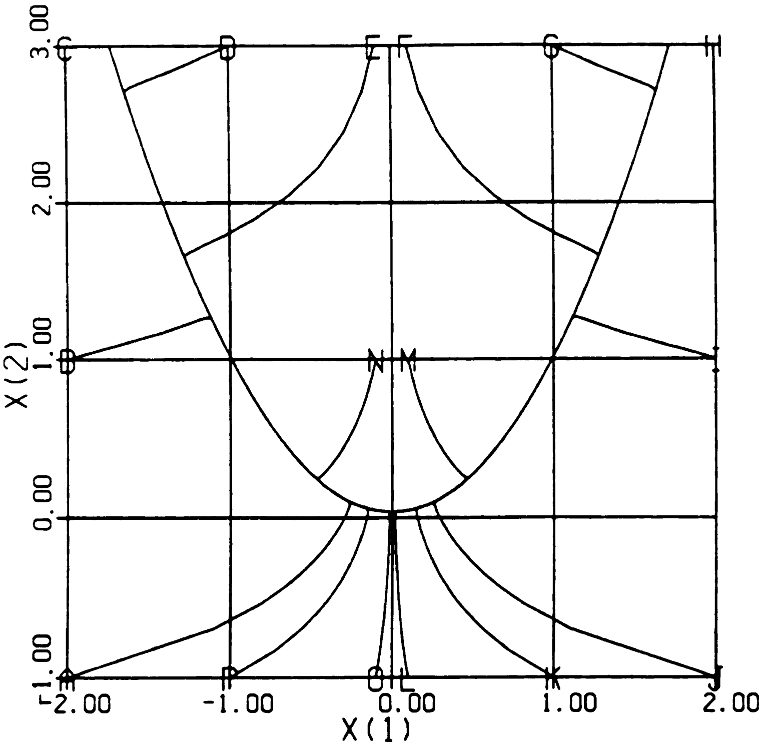


Figure 5.15. Trajectories of x of equation 4.1 for (NP_1) with s=50.

depending on whether x_1 of the initial point is greater or smaller the zero. In fact, it can be shown that $\bar{x}_1$ is asymptotically stable on $\{x \in R^2 | x_1 > 0\}$ whereas $\bar{x}_2$ is asymptotically stable on $\{x \in R^2 | x_1 < 0\}$. Though it was mentioned in Section 4.3 that for nonlinear programming the asymptotic stability of the equilibrium points of equation 4.1 are held locally for sufficiently large s, this example shows for (NP_1) that the basins of attraction of the two equilibrium points nearly cover R^2 except for the line x_1=0. For s=50 if a trajectory starts out with x_1=0, it stays on x_1=0 and moves toward

$$x = (0.0, 0.03846),$$

which is a saddle point originating from x_{e1} as $s>2$. Since in hardware implementation one can not select a point which is precisely zero, the effect of this saddle point is actually unobservable. Thus the network may be thought of almost completely stable in practice.

A 2-phase network has been used to shift the equilibrium points

$$(\pm\sqrt{\frac{s-2}{2s}}, \frac{1}{2}) \text{ to } (\pm\frac{1}{\sqrt{2}}, \frac{1}{2}).$$

The simulation results are shown in Figure 5.16. The little hook near the end of the trajectory is due to the effect of phase-2 dynamics. By the choice of s and ε for the 2-phase network in solving (NP_1), the time for phase-2 to converge is roughly triple of the time for phase-1. Generally speaking, the smaller ε, the longer the time for phase-2 to converge.

Consider now the following nonlinear program (NP_2) quoted from [19]:

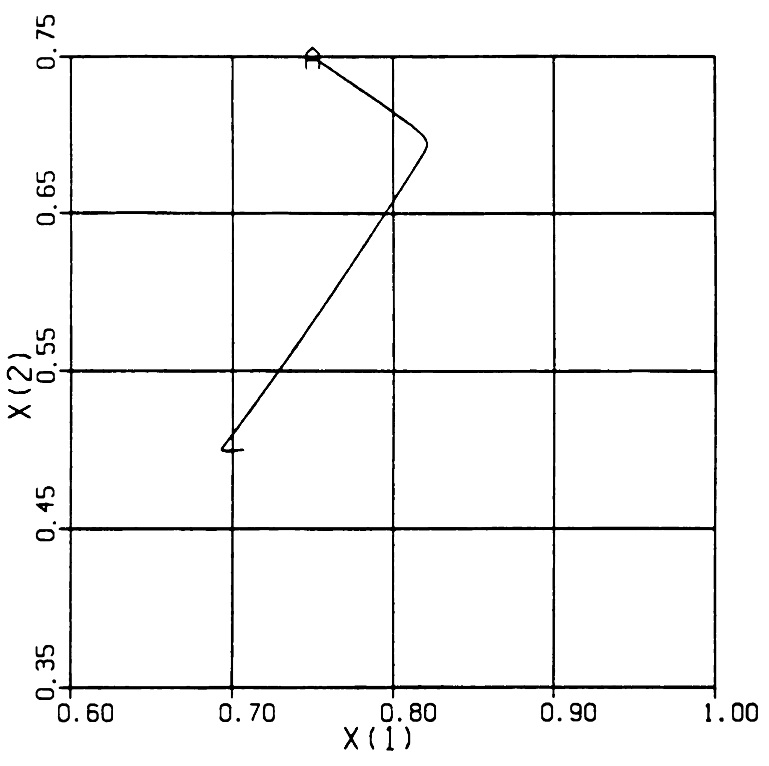
$$\text{Minimize } f(x) = x_1^2 + x_2^2 - x_1x_2 + 0.4x_2 + \frac{x_1^3}{30}$$

$$\begin{aligned}\text{subject to } & g_1(x) = x_1 + 0.5x_2 \geq 0.4, \\ & g_2(x) = 0.5x_1 + x_2 \geq 0.5, \\ & g_3(x) = x_1 \geq 0, \\ & g_4(x) = x_2 \geq 0.\end{aligned}$$

This is in fact a convex program because $f(x)$ is a convex function and the feasibility set is convex. The convexity of $f(x)$ may be seen by deriving the Hessian matrix of $f(x)$,

$$\nabla^2 f(x) = \begin{bmatrix} 0.2x_1+2 & -1 \\ -1 & 2 \end{bmatrix},$$

(a)



(b)

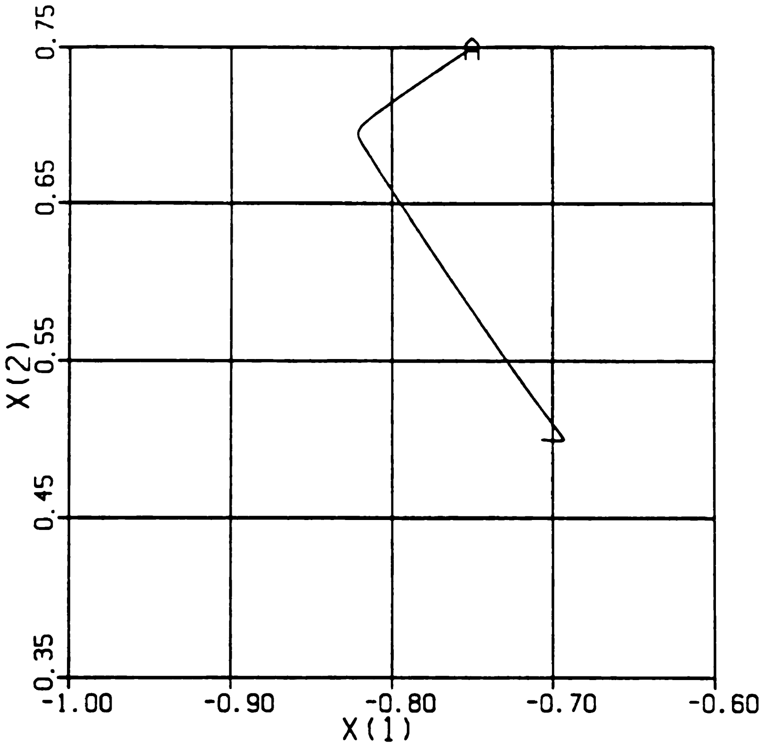


Figure 5.16. Trajectories of x for (NP_1) using the 2-phase network with s=10 and ε=0.2. (a) x_o = (0.75, 0.75). (b) x_o = (−0.75, 0.75).

and showing that its eigenvalues

$$(0.1x_1+2)\pm\sqrt{0.01x_1^2+1}$$

are positive for $x_1 \geq 0$.

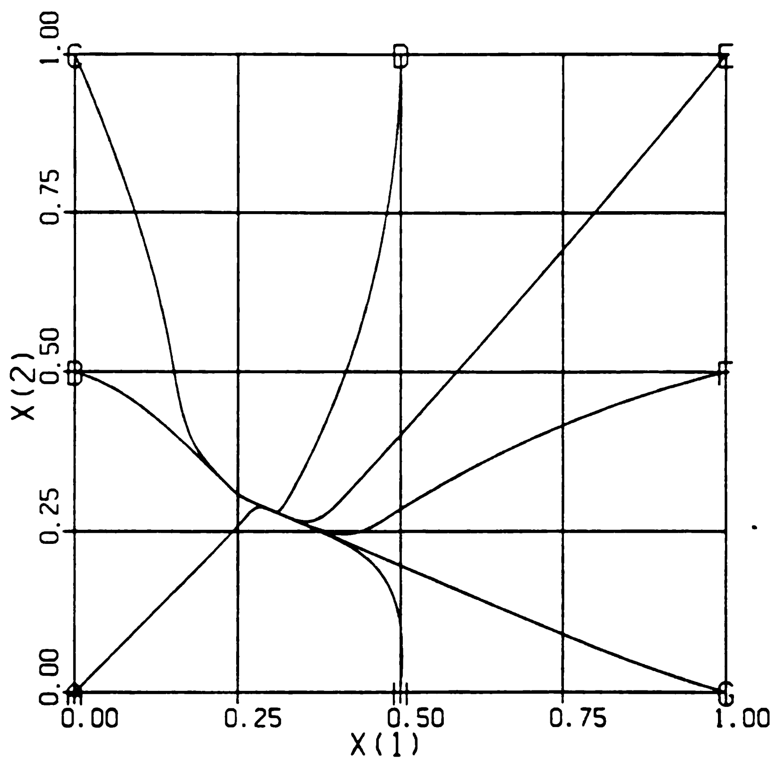
Using equation 4.1 to solve (NP_2), the trajectories of x of are shown in Figure 5.17(a). The simulations are done with $s=10$. The equilibrium of the network is given in the first entry of Table 5.2 and contrasted to the equilibrium obtained by a 2-phase network with $s=10$ and $\varepsilon=0.2$. As seen from the table, the solution obtained by the 2-phase network is very accurate. The trajectories of the 2-phase network with initial points (0.25, 0.25) and (0.45, 0.45) are plotted in Figure 5.17(b). The line segment near the middle of the figure is caused by the phase-2 dynamics.

Table 5.2. The equilibrium points for different networks for (NP_2).

Network formulation	x_1	x_2
Equation 4.1 with $s=10$	0.3023760	0.2825410
2-phase net with $s=10$, $\varepsilon=0.2$	0.3395630	0.3302180
Exact solution	0.3395628	0.3302186

As shown in Section 4.3 the network formulation of equation 4.1 may be used find minimizers of a pseudo-convex function subject to some quasi-convex and quasi-concave constraints. If an equilibrium $\tilde{x}$ of equation 4.1 occurs at the interior of the feasibility set, $\nabla f(\tilde{x})=0$ must hold. By the properties of a pseudo-convex function it follows that $\tilde{x}$ is a global minimizer of f. Suppose now that the objective function is $f: X \rightarrow R$ and $f \in C^1$, where X is an open interval in R. Let (P_1) be the problem to minimize f over S, where S is a subinterval of X. Mapping (P_1) to the network of equation 4.1, the network dynamics drive the trajectory continuously along the descending direction on the surface of $f(x)$. The trajectory ends at either the boundary of S or a point in S for which $\nabla f(x)=0$. If the former happens, then f is strictly

(a)



(b)

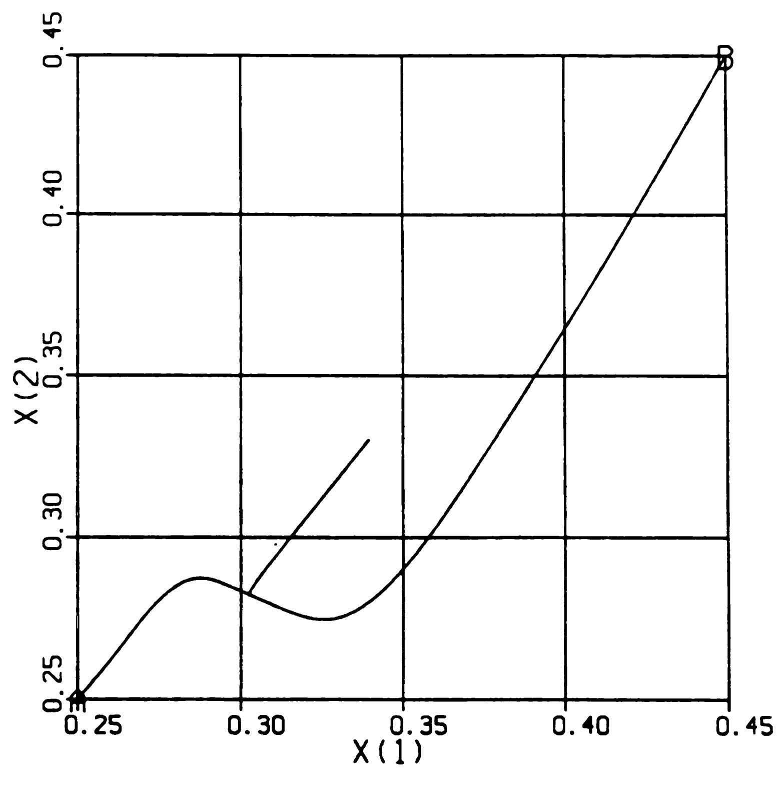


Figure 5.17. Trajectories of x for different networks for (NP_2). (a) Equation 4.1 with s=10. (b) 2-phase network with s=10 and ε=0.2.

monotonic (either increasing or decreasing) on S. Both cases lead to an interesting application, namely, solving $p(x)=0$ over an interval.

Let $p:S \rightarrow R$ and $p \in C^2$, for S an closed interval in R. Let (P_2) be the problem to solve $p(x)=0$ for $x \in S$. Assume (P_2) is feasible, i.e., there exists an $x \in S$ such that $p(x)=0$. Assume that p is strictly monotonic on S. Let $E(x)=\frac{1}{2}p^2(x)$, then

$$E'(x) = p'(x)p(x) \tag{5.1}$$

and

$$E''(x)=(p'(x))^2 + p(x)p''(x). \tag{5.2}$$

For $p(\bar{x})=0$ we have $E'(\bar{x})=0$ and $E''(\bar{x})=(p'(\bar{x}))^2>0$ by the strictly monotonic assumption of p. Thus $\bar{x}$ is a local minimizer of $E(x)$ and and a solution to $p(x)=0$. In fact, $\bar{x}$ is also the global minimizer of $E(x)$ on S, since the strictly monotonic assumption assures that $E(x)$ is pseudo-convex on S with a unique minimizer.

To relax the above argument to more general functions, assume that $p \in C^1$. Note that since $p(\bar{x})=0$, it follows

$$p'(\bar{x}) = \lim_{h \to 0} \frac{p(\bar{x}+h)}{h}.$$

Now taking the limit of the difference quotient of $E'(x)$ at $\bar{x}$, we get

$$\lim_{h \to 0} \frac{E'(\bar{x}+h) - E'(\bar{x})}{h}$$

$$= \lim_{h \to 0} \frac{p'(\bar{x}+h)p(\bar{x}+h) - p'(\bar{x})p(\bar{x})}{h}$$

$$= \lim_{h \to 0} \frac{p'(\bar{x}+h)p(\bar{x}+h)}{h}$$

$$= \lim_{h \to 0} p'(\bar{x}+h) \lim_{h \to 0} \frac{p(\bar{x}+h)}{h}$$

$$= (p'(\bar{x}))^2 > 0. \tag{5.3}$$

Thus $\bar{x}$ is again a local minimizer of $E(x)$ on S. By the strictly monotonic assumption of $p(x)$ on S, $\bar{x}$ is also a global minimizer of $E(x)$ on S.

Since $E(x)$ is pseudo-convex in the above, choose the network

$$\dot{x} = -E'(x) = -p(x)p'(x) \tag{5.4}$$

such that

$$\frac{dE(x)}{dt} = p(x)p'(x)\dot{x} \tag{5.5}$$

$$= -(p(x)p'(x))^2.$$

Since by the assumption of strict monotonicity $p'(x) \neq 0$ for $x \in S$, the unique equilibrium of equation 5.4 is $\bar{x}$ at which $E(x)$ achieves its minimum. The idea behind this technique is that since $E(x)$ is pseudo-convex with its minimum in S, then it is a valid Lyapunov function for the system of equation 5.4.

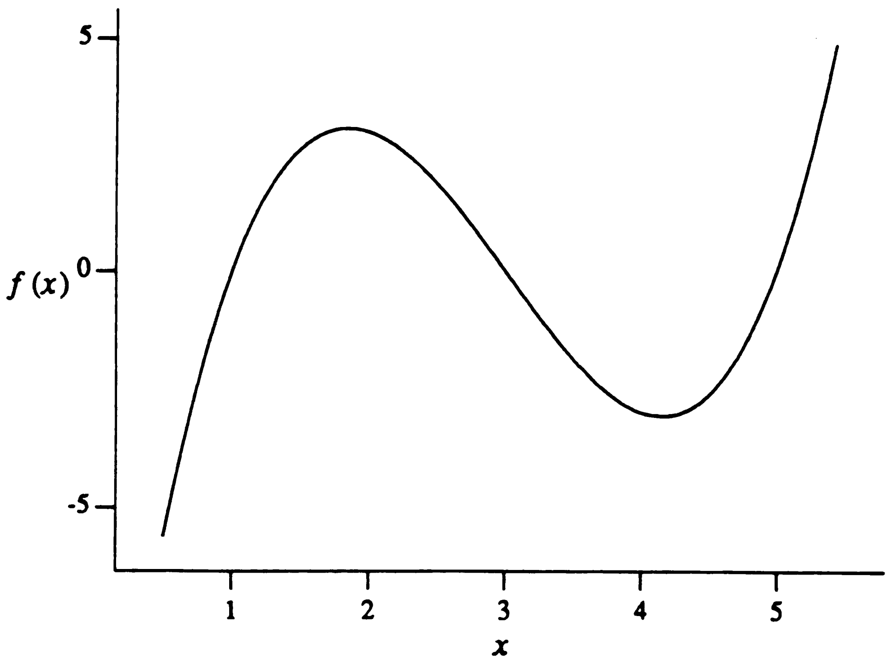
Same technique is applicable to solving any feasible problem $q(x)=0$ on a closed interval S, where q is such that q^2 is C^1 pseudo-convex on S. Thus q^2 may be a valid Lyapunov function for equation 5.4. It is possible for q^2 to have more than one minimizer. If $q \in C^1$, and if q^2 has only a local minimum equal to zero, finitely many local maximizers, and no saddle points, then this technique is also usable. Since for such a function q, except when the initial point is one of the maximizers, the system of equation 5.4 will converge to one of the minimizers $\bar{x}$ of q^2 for which $q(\bar{x})=0$. In hardware implementation, the effect of the finitely many maximizers is unobservable. Thus the system may be regarded as almost completely stable in practice.

Consider the following problem (NP_3):

$$\text{Solve } f(x) = x^3 - 9x^2 + 23x - 15 = 0.$$

The function $f(x)$ is plotted in Figure 5.18(a). It has three roots at 1, 3, and 5. The profile of $f^2(x)$ is shown in Figure 5.18(b). $f^2(x)$ has three minimizers, 1, 3, and 5,

(a)



(b)

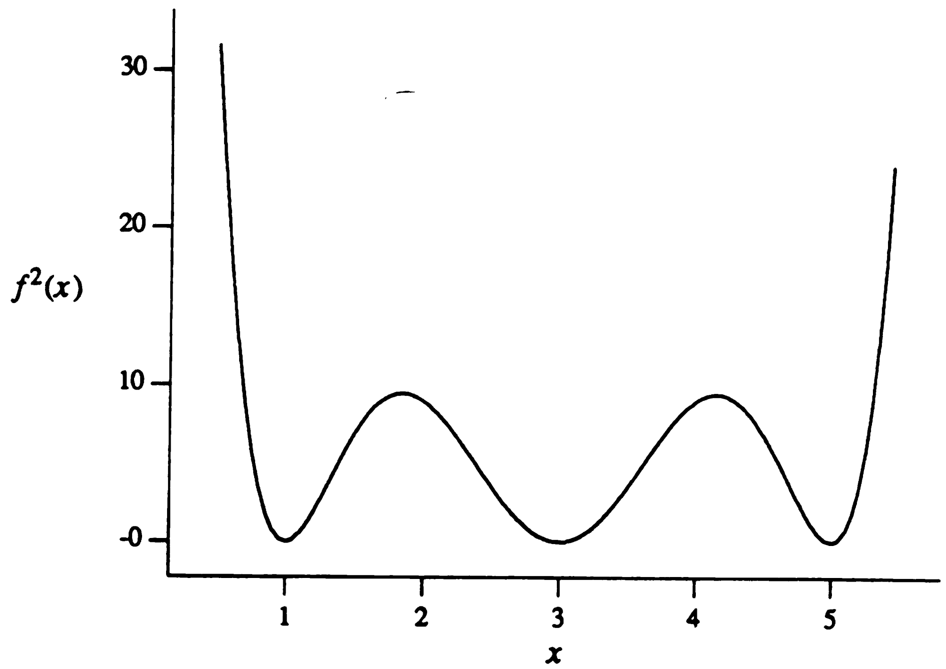


Figure 5.18. (a) $f(x)$ for (NP_3). (b) $f^2(x)$ for (NP_3).

and two local maximizers,

$$4.1547 \text{ and } 1.8453.$$

Solving this problem by equation 5.4, the trajectories of of x versus $E(x)$ are shown in Figure 5.19. The asymptotical stability of equation 5.4 on its three equilibria can be clearly seen in the figure. The regions of attraction for x=1, 3, and 5 are respectively $(-\infty, 1.8435)$, $(1.8435, 4.1547)$, and $(4.1547, \infty)$. They indeed cover almost all R except two points. It is thus clear the system is almost completely stable.

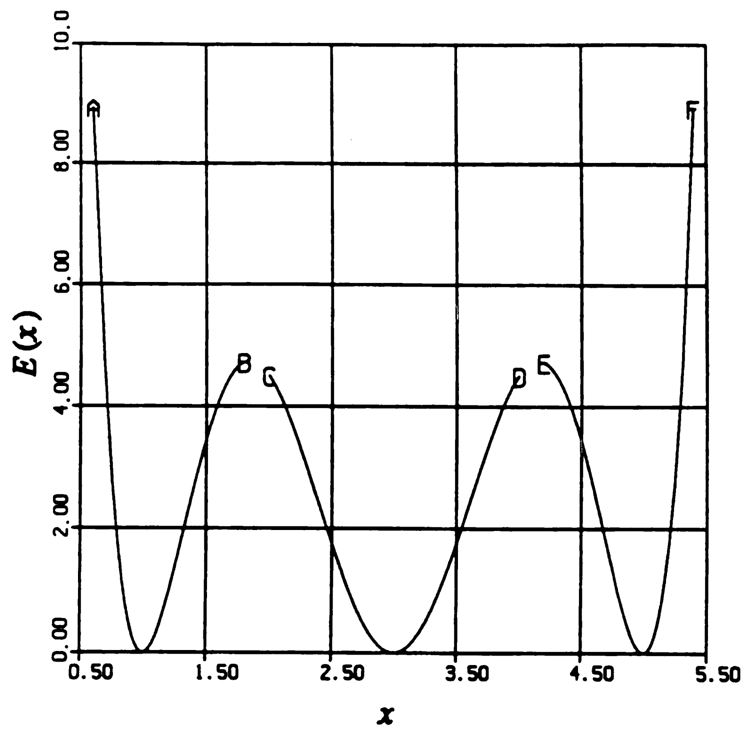


Figure 5.19. The trajectories of x of equation 5.4 versus $E(x)$ for (NP_3).

CHAPTER VI

CASE STUDIES

Two optimization problems encountered in power system engineering are solved by the developed network formulation in this chapter. They are the economic power dispatch (EPD) problem and the optimal power flow (OPF) problem. The results of this case study demonstrate the applicability of the developed network for solving the optimization problems encountered in real engineering situations.

6.1 Economic Power Dispatch

The EPD is a classical problem in power system optimization. The goal of EPD is to determine the amount of power to be produced by each generating unit in the system such that the load (demand) can be met with a minimum total generation cost. The power system model typically consists of n thermal-generating units connected to a single load R. Let x_i be the power generated by the ith unit and f_i be the generation cost rate function of that unit. f_i is generally approximated by a quadratic polynomial of the form

$$f_i(x_i) = b_i + a_i x_i + A_{ii} x_i^2 \tag{6.1}$$

where b_i, a_i, and q_{ii} are positive constants. Each x_i is restricted between $[x_{i,min}, x_{i,max}]$ as determined by the generation limits of the unit. Without considering the transmission line losses, the EPD problem (D_1) may be expressed as:

$$\text{Minimize} \quad f(x) = \sum_{i=1}^{n} f_i(x_i)$$

$$\text{subject to} \quad x_{i,min} \le x_i \le x_{i,max} \quad \text{for } i=1,\dots,n$$

$$\text{and} \quad \sum_{i=1}^{n} x_i - R = 0.$$

(D_1) is actually a quadratic program and thus the results of Section 4.2 may be applied. Traditional methods to solve this problem can be found in [153].

If the line losses are taken into account, the EPD problem changes its form to (D_2):

$$\text{Minimize} \quad f(x) = \sum_{i=1}^{n} f_i(x_i)$$

$$\text{subject to} \quad x_{i,min} \le x_i \le x_{i,max} \quad \text{for } i=1,\dots,n$$

$$\text{and} \quad \sum_{i=1}^{n} (x_i - PL) - R = 0$$

where PL represents the line losses. A general version of PL may be expressed as the quadratic function

$$PL = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i B_{ij} x_j + \sum_{i=1}^{n} B_i x_i + B_o. \tag{6.2}$$

Due to the nonlinearity of PL it is more difficult to solve (D_2) than (D_1). An iterative process is normally adopted to solve (D_2) (see Chapter 4 of [153]). The developed optimization network is superior in that with same mapping technique it is applicable no matter whether the line losses are considered or not.

Example 6.1 [153]: A power system consists three generating units. The minimum and maximum output of units 1, 2, and 3 are [150 MW, 600 MW], [100 MW, 400 MW], and [50 MW, 200 MW], respectively. A total of 850 MW is to be delivered at a minimum overall cost.

Case 1: Suppose the cost rates for the units are

$$f_1(x_1) = 561 + 7.92x_1 + 0.001562x_1^2,$$

$$f_2(x_2) = 310 + 7.85x_2 + 0.00194x_2^2,$$

and

$$f_3(x_3) = 78 + 7.97x_3 + 0.00482x_3^2.$$

Matching the corresponding terms of $f(x)$ to the objective function of a quadratic program (QP), gives

$$A = \begin{bmatrix} 0.003124 & 0 & 0 \\ 0 & 0.00388 & 0 \\ 0 & 0 & 0.00964 \end{bmatrix},$$

$$a = \begin{bmatrix} 7.92 \\ 7.85 \\ 7.97 \end{bmatrix},$$

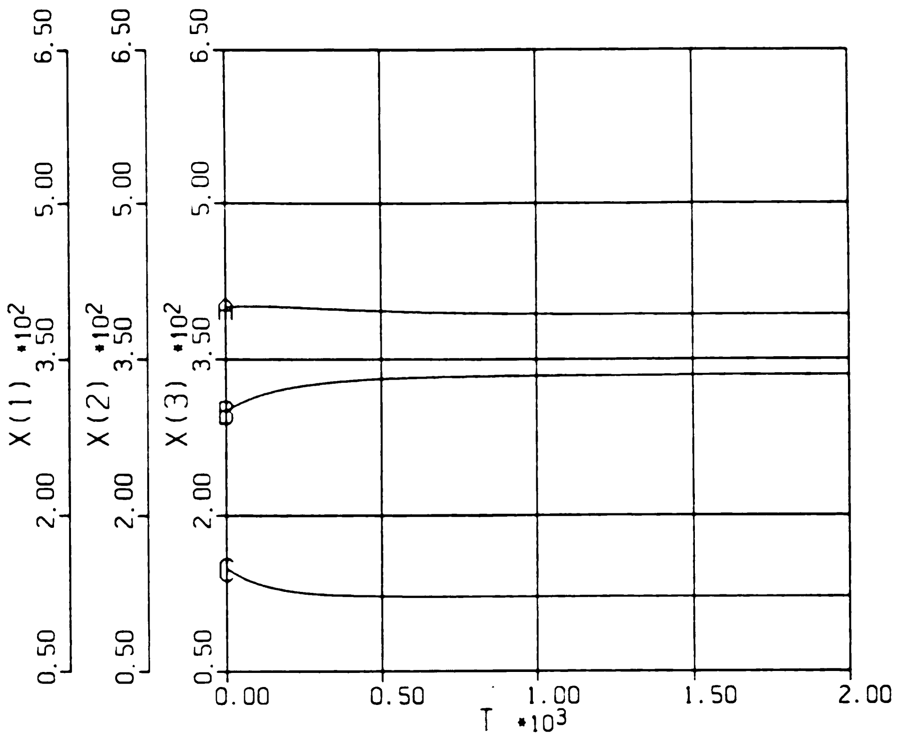
and

$$c = \begin{bmatrix} 561 \\ 310 \\ 78 \end{bmatrix}.$$

It is clear that A is positive definite. Furthermore, the feasibility set of this problem is convex and compact. Therefore, the results in Section 4.2 can be applied presuming that the minimizer set O contains only regular points.

The simulations of the network of equation 4.1 are done with s=50 and the initial condition x_o=[400, 300, 150]^T. The trajectories of the state variables are plotted in Figure 6.1(a); the trajectories of $f(x)$ and $E_2(x)$ are shown in Figure 6.1(b). The asymptotic nature of the equilibrium point is clearly seen in these figures. The final values of the state variables, $E_2(x)$, and $f(x)$ are given in Table 6.1 as is the exact solution. The error between the simulation result and the exact solution is less than 0.1% on average. The value of $E_2(x)$ is slightly larger than the objective function.

(a)



(b)

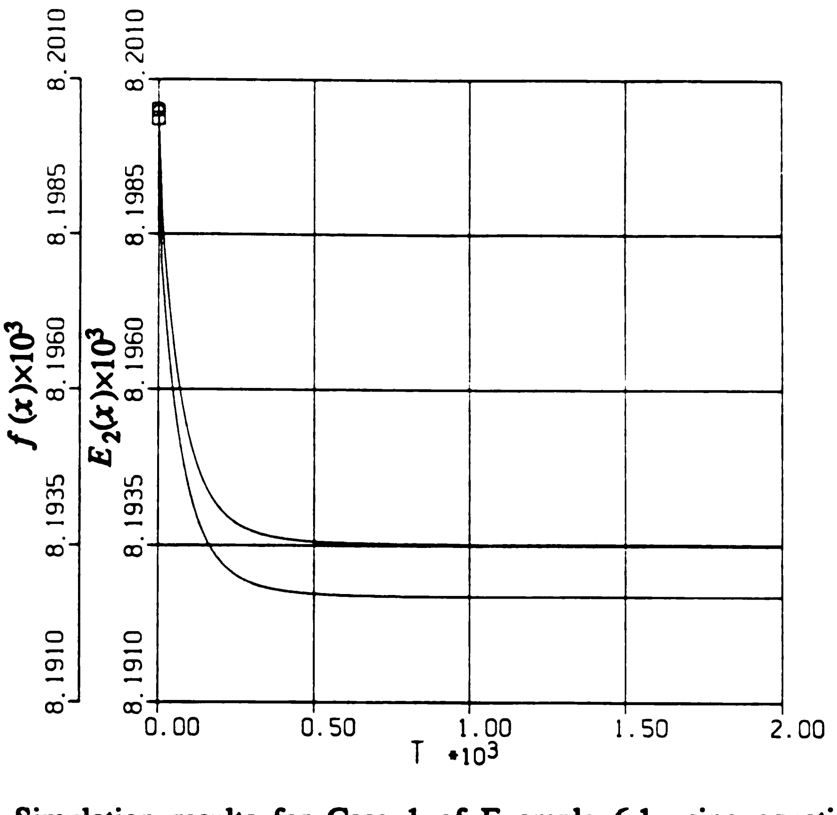


Figure 6.1. Simulation results for Case 1 of Example 6.1 using equation 4.1 with s=50 and x_o=[400, 300, 150]^T. (a) Trajectories of x. (b) Trajectories of $f(x)$ and $E_2(x)$.

This is due to the fact that the equilibrium of the network always lies in the infeasible region and thus results in a small positive value in the second term on the right hand side of equation 3.10.

The simulation results for Case 1 using the 2-phase network with s=50 and ε are shown in Figure 6.2. The system is switched to phase-2 at t=1000. Normally, phase-2 dynamics requires longer times to converge, but for this particular case the phase-2 network converges rather quickly to its equilibrium as can be seen from Figure 6.2(b). From Table 6.1 the final values of state variables of the 2-phase network are the same as the exact solution.

Table 6.1. Equilibrium points of two networks for Case 1 of Example 6.1.

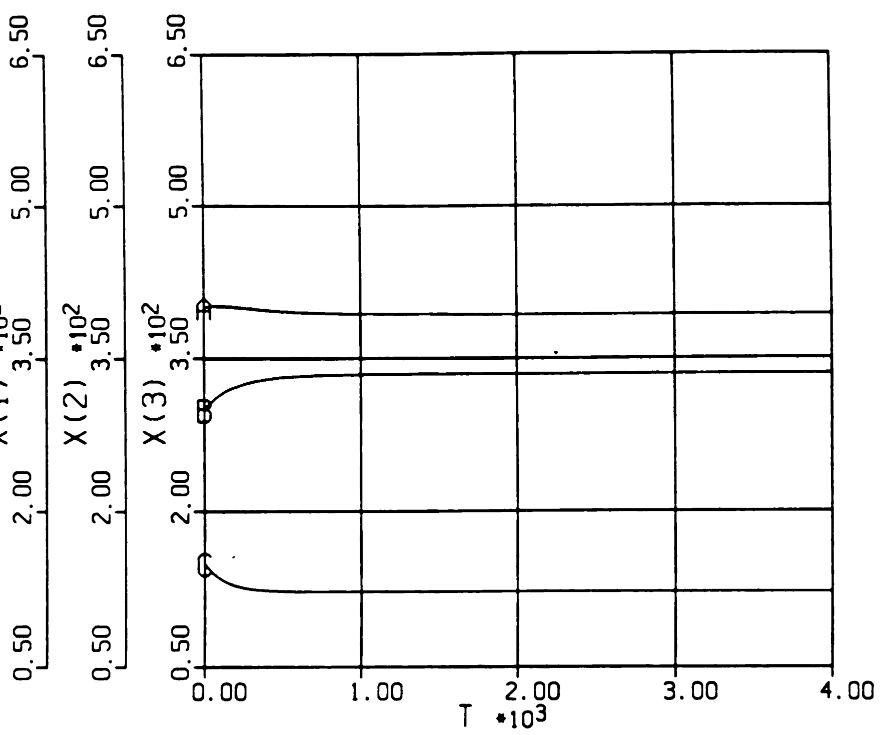
Networks	x_1	x_2	x_3	$f(x)$	$E_2(x)$
Equation 4.1, s=50	393.10	334.52	122.20	8192.68	8193.52
2-phase net, s=50, ε=0.2	393.17	334.60	122.23	8194.36	8194.36
Exact Solution	393.17	334.60	122.23	8194.36	8194.36

Case 2: Suppose now that due to fluctuation of the resource price, the cost rate of unit 1 becomes

$$f_1(x_1) = 459 + 6.48x_1 + 0.00128x_1^2 .$$

After adjusting the corresponding terms in the network, the simulation is performed with the same parameter s and initial condition. The results are listed in Table 6.2. Again, the results of using 1-phase network (equation 4.1) approximate the exact solution closely while the results of using 2-phase network match it precisely. The trajectories of the network variables for equation 4.1 are plotted in Figure 6.3. Unit 1 in this case must produce its maximum output. In [153], a different scheme is required

(a)



(b)

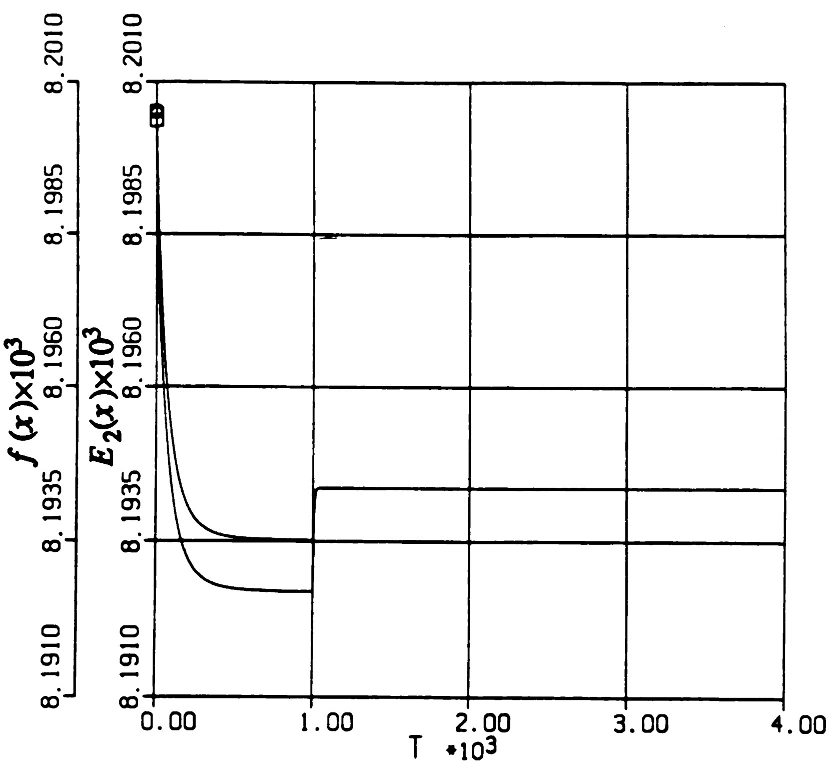


Figure 6.2. Simulation results for Case 1 of Example 6.1 using the 2-phase network with s=50, ε=0.2 and x_o=[400, 300, 150]^T. (a) Trajectories of x. (b) Trajectories of $f(x)$ and $E_2(x)$.

to handle the case when state variables are achieving their extremum. Our network formulation, however, can apply to such a case without any change.

□

Table 6.2. Equilibrium points of two networks for Case 2 of Example 6.1.

Networks	x_1	x_2	x_3	$f(x)$	$E_2(x)$
Equation 4.1, s=50	600.01	187.00	62.82	7250.63	7251.37
2-phase net, s=50, ε=0.2	600.00	187.13	62.87	7252.11	7252.11
Exact Solution	600.00	187.13	62.87	7252.11	7252.11

The above example does not take the line losses into account. To demonstrate the capability of the developed network formulation in solving the EPD problem with line losses, consider the following.

Example 6.2: This example problem is taken from Example 4E in [153]. In this power system there are three generation units with unit dispatch limits

$$50.0 \text{ MW} \le x_1 \le 200 \text{ MW},$$

$$37.5 \text{ MW} \le x_2 \le 150 \text{ MW},$$

$$45.0 \text{ MW} \le x_3 \le 180 \text{ MW}.$$

A total of 210 MW is to be delivered at a minimal overall cost. The generation cost rates are

$$f_1(x_1) = 213.1 + 11.669x_1 + 0.00533x_1^2,$$

$$f_2(x_2) = 200.0 + 10.333x_2 + 0.00889x_2^2,$$

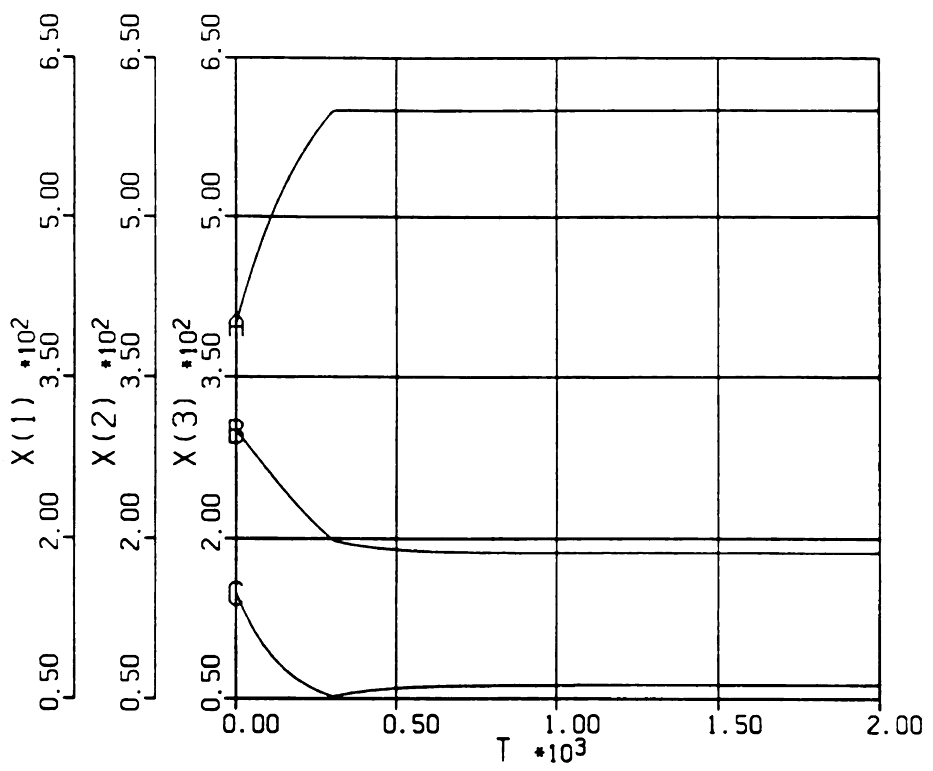
$$f_3(x_3) = 240.0 + 10.833x_3 + 0.00741x_3^2.$$

Fig

$s=$

$E_2($

(a)



(b)

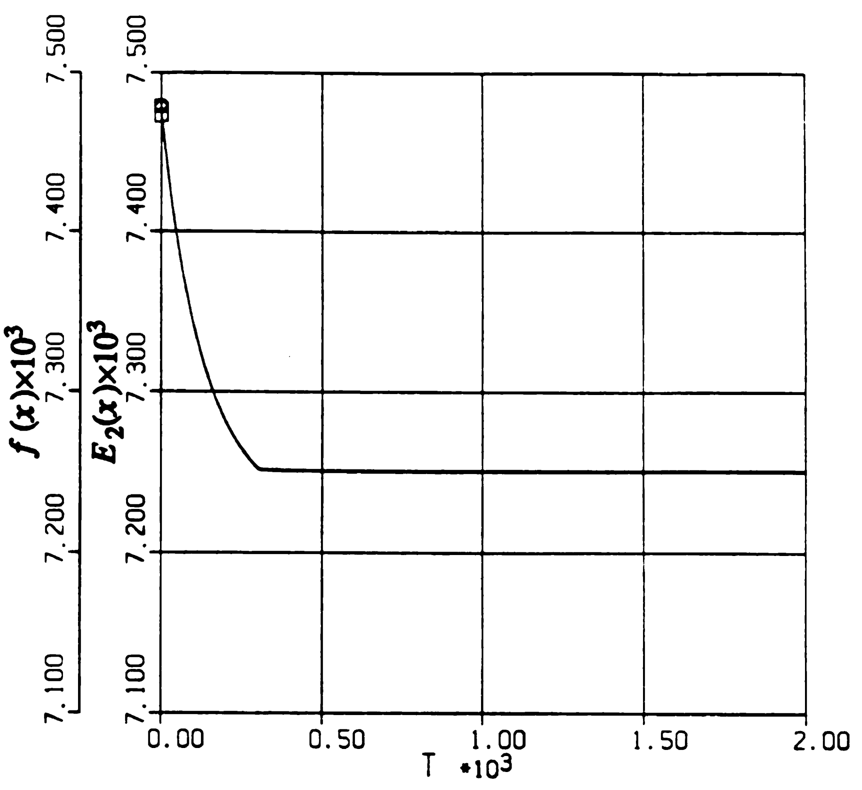
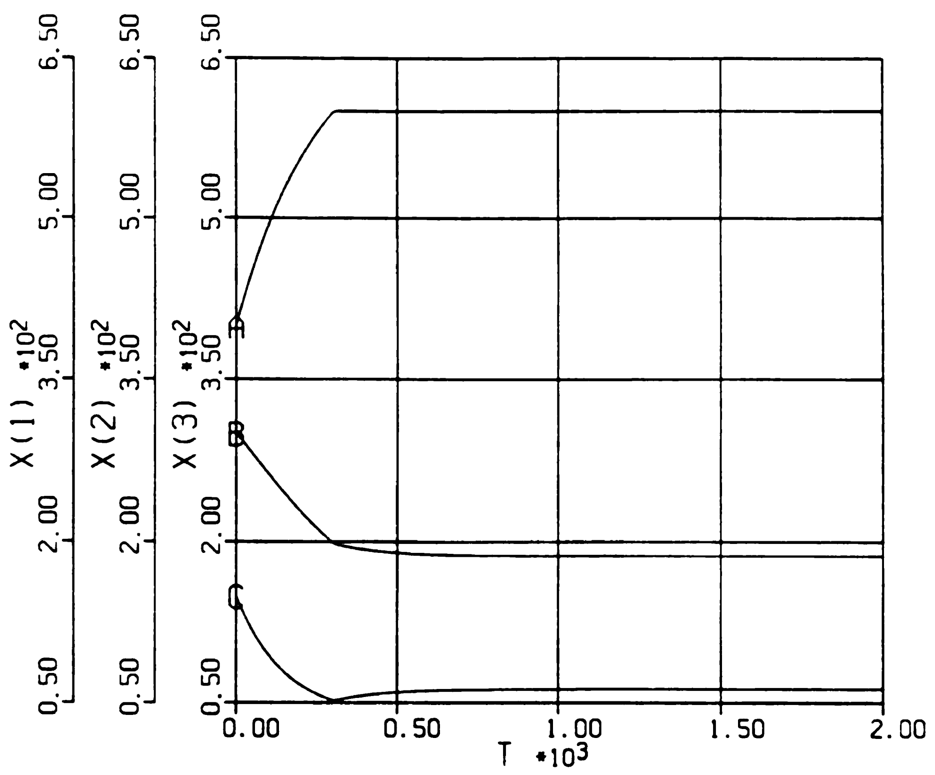


Figure 6.3. Simulation results for Case 2 of Example 6.1 using equation 4.1 with s=50 and x_o=$[400, 300, 150]^T$. (a) Trajectories of x. (b) Trajectories of $f(x)$ and $E_2(x)$.

(a)



(b)

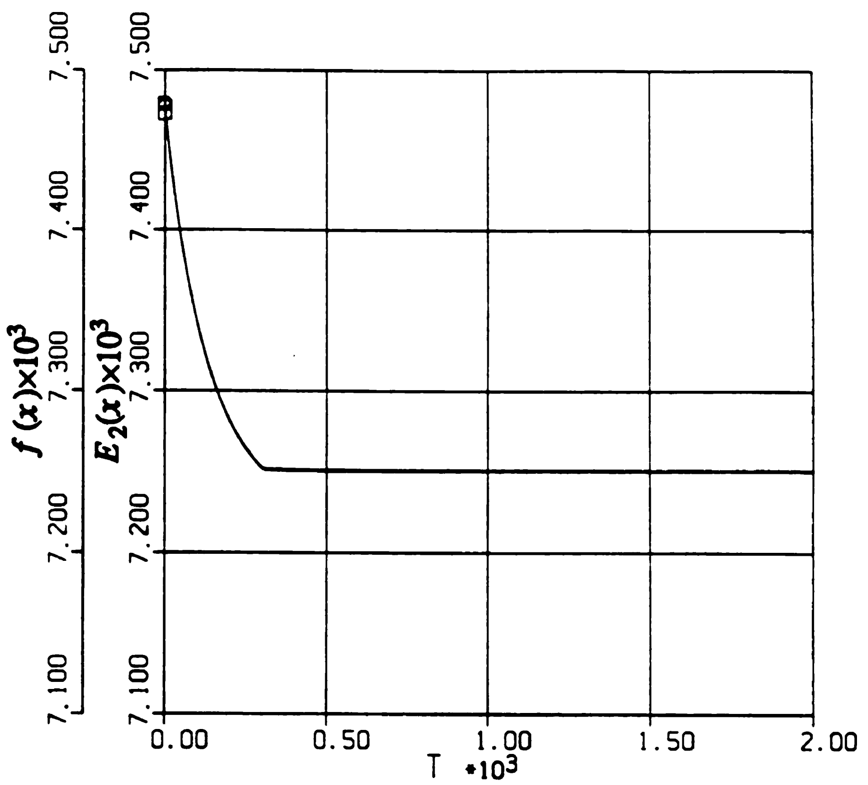


Figure 6.3. Simulation results for Case 2 of Example 6.1 using equation 4.1 with s=50 and x_o=[400, 300, 150]^T. (a) Trajectories of x. (b) Trajectories of $f(x)$ and $E_2(x)$.

It is clear that the objective function $f(x)=\sum_{i=1}^{3} f_i(x_i)$ is a convex function. The corresponding B coefficients of the line losses PL are given by

$$[B_{ij}] = \begin{bmatrix} 6.760 & 0.953 & -0.507 \\ 0.953 & 5.210 & 0.901 \\ -0.507 & 0.901 & 2.940 \end{bmatrix} \times 10^{-4},$$

$$[B_i] = \begin{bmatrix} -0.07660 \\ -0.00342 \\ 0.01890 \end{bmatrix},$$

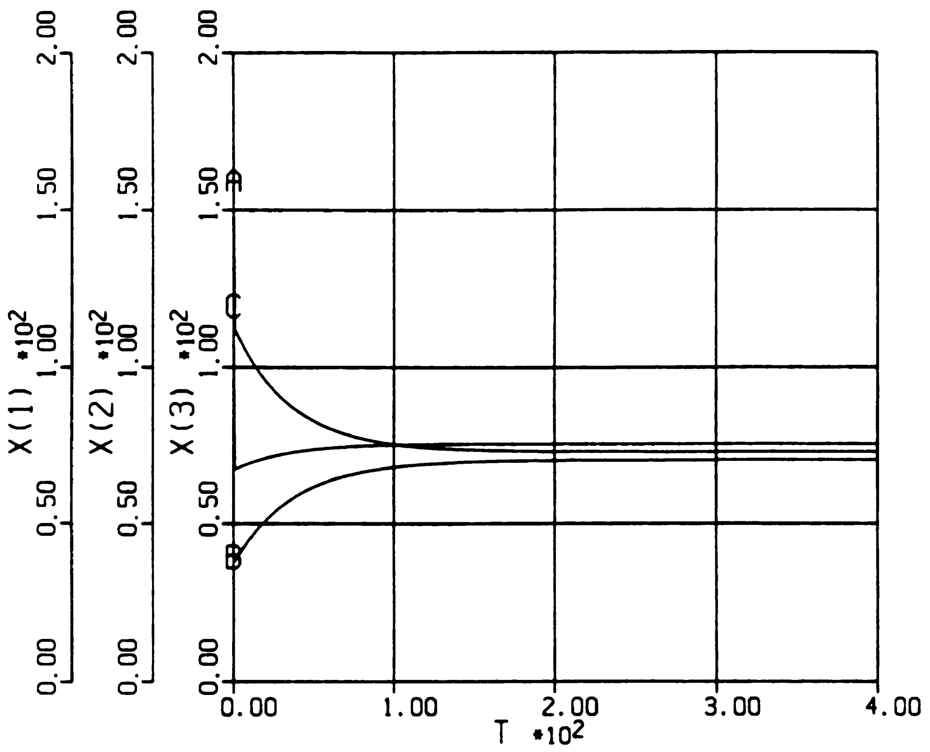
and

$$B_o = 4.0357.$$

Since $[B_{ij}]$ is positive definite, the equality constraint for this problem is a concave function. The feasible region, defined by the unit dispatch limits, is a rectangular solid which is clearly convex. Therefore, the results in Section 4.3 can be applied presuming that the set of optimal solutions O is nonempty and contains only regular points.

The problem is mapped into the network formulation of equation 4.1. Simulations of the network are done with s=50 and initial condition x_o=[160, 40, 120]^T MW. The trajectories of the state variables are plotted in Figure 6.4(a). $f(x)$ and $E_2(x)$ are plotted in Figure 6.4(b) and the line losses in Figure 6.4(c). Again, the asymptotic nature of the network equilibrium is seen in these figures. As the system evolves with time, $E_2(x)$ is almost identical to $f(x)$ due to the appropriate choice of s. The final values of all variables are shown in Table 6.3 as are the results as given in [153]. The proposed network achieves a better objective value with less line loss. More importantly, even though the network has not yet converged to its equilibrium, $E_2(x)$ becomes very close to its final value in only few generic time steps. This may be explained by the slow manifold effect stemming from the smallness of $[B_{ij}]$.

(a)



(b)

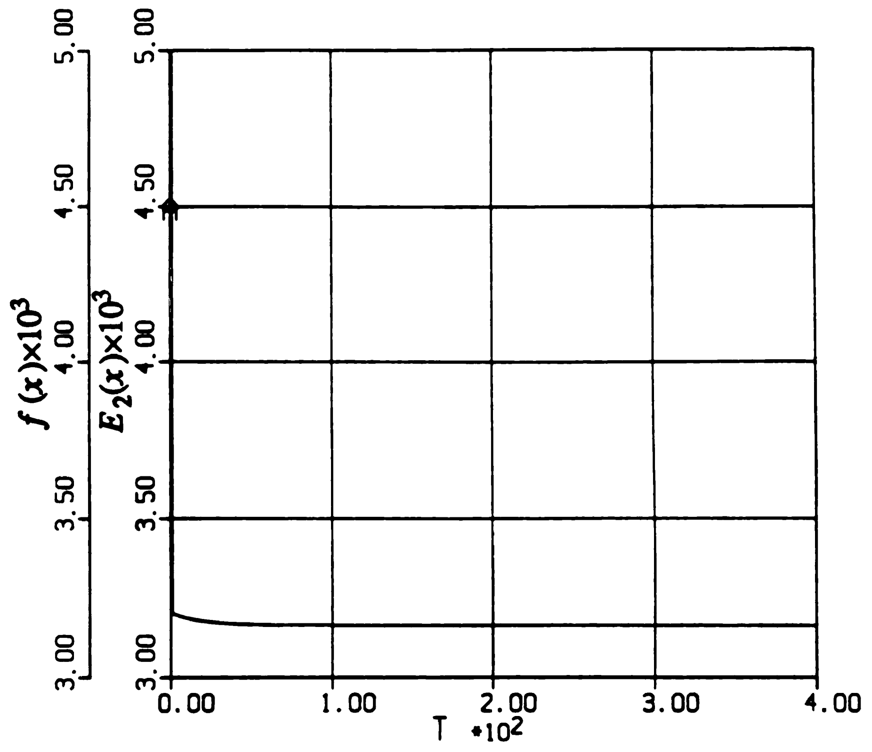


Figure 6.4. Simulation results for Example 6.2 using equation 4.1 with $s=50$ and $x_o=[160, 40, 120]^T$. (a) Trajectories of x. (b) Trajectories of $f(x)$ and $E_2(x)$.

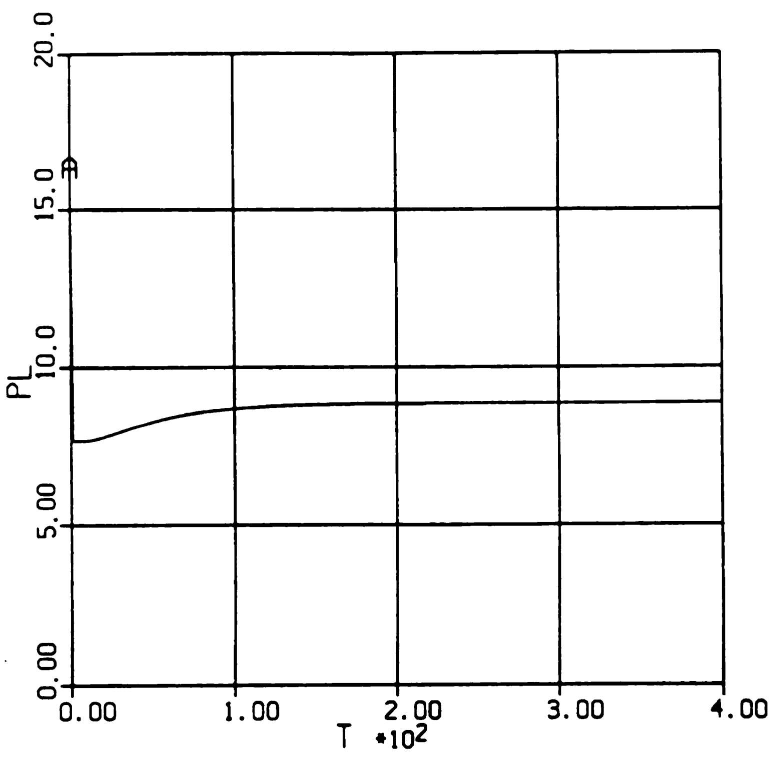


Figure 6.4. (cont'd.) (c) The trajectory of PL.

Simulations with 8 possible dispatch limits as initial condition have been done and the results confirm the complete stability of the system as well as the uniqueness of the equilibrium in this example.

This problem has also been solved by the 2-phase network for s=50, ε=0.2, and $x_o=[160, 40, 120]^T$. The trajectories of the variables of the 2-phase network are not shown since they are nearly the same as the trajectories in Figure 6.4. The final results are listed in Table 6.3. Though the exact solution of the problem is not available, the results obtained by the 2-phase network may be regarded as nearly exact. This is because at the results obtained by the 2-phase network, the constraint violations are less than 2×10^{-2}, which is considered negligible. □

Table 6.3. Comparison of the simulation results for Example 6.2 with the results in [153].

Methods	x_1	x_2	x_3	PL	$f(x)$	$E_2(x)$
Equation 4.1, s=50	72.8876	70.2605	75.4437	8.8484	3162.35	3164.00
2-phase net, s=50, ε=0.2	72.9822	70.3250	75.5580	8.8652	3165.64	3165.64
Method in [153]	60.2677	79.4462	80.1503	9.8650	3168.62	--

6.2 Optimal Power Flow

The OPF is essentially a static optimization problem of power system operations. The aim is to find an optimal generation schedule that evokes a minimal production cost and simultaneously satisfies the power flow equations and the operational constraints. The general formulation of the OPF is:

$$\begin{aligned} \text{Minimize} \quad & f(u,x) \\ \text{subject to} \quad & g(u,x) \leq 0 \\ & h(u,x) = 0 \end{aligned}$$

where u is the set of independent variables, x is the set of dependent variables, $g(u,x)$ reflects the operational constraints, and $h(u,x)$ corresponds to the load flow equations.

The cost functions $f(u,x)$ of the OPF problem are similar to those of the EPD problem. But, OPF problems are more difficult to solve because of the inclusion of the smooth but non-convex power flow equations. Also the OPF problems differ from the basic load flow problem in the presence of a performance index criterion (i.e., the minimization of a cost function) and the explicit inclusion of inequality constraints $h(u,x)$. These constraints refer to lower and upper limits on real and reactive power generation, power flows on lines and transformers, and voltage levels.

Approaches currently used to solve the OPF problems include quadratic programming, separable programming, and the Newton method [154]. These approaches, if successful, converge to one of the local minimizers of the problem depending on the starting point of the iterative process. If the developed optimization network technique is regarded as a continuous mode of the discrete, iterative optimization process, it is expected that the region of attraction of the network is similar to the region that results in convergence when using the traditional approaches. Moreover, the equilibrium points of the network should be very close to the solution obtained by other approaches.

Example 6.3: For a power system, P_{gi} (Q_{gi}) is the real (reactive) power generation at bus i, P_{di} (Q_{di}) is the real (reactive) power load at bus i, and V_i is the voltage magnitude at bus i with a phase angle δ_i. Consider now the system shown in Figure 6.5 [155]. The costs of generation for generator 1 and 2 are, respectively,

$$f_1(P_{g1}) = 1 + P_{g1} + 3(P_{g1})^2$$

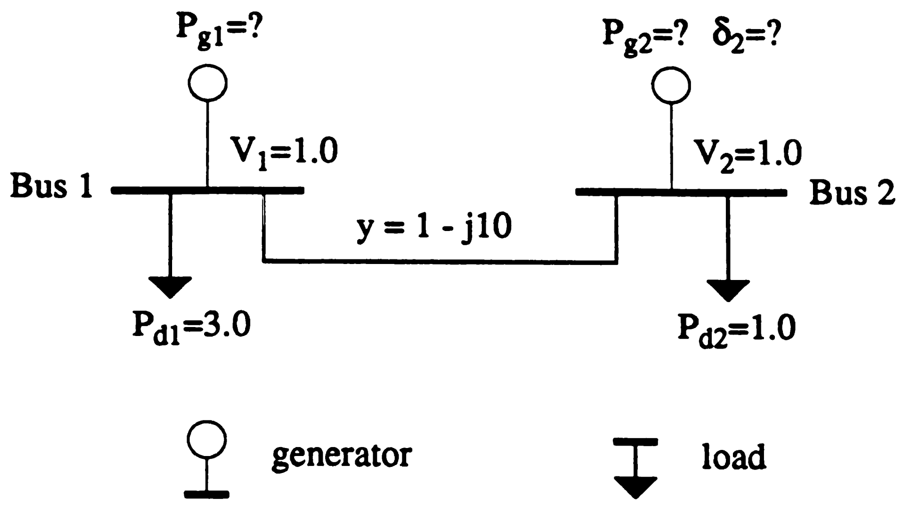


Figure 6.5. Power system network for Example 6.3.

and

$$f_2(P_{g2}) = 0.5 + 0.5P_{g2} + 0.5(P_{g2})^2.$$

Bus 1 is taken to be the swing bus (i.e., δ_1=0). By the power flow equation, we have

$$P_{g1} - 3 = 1 - \cos\delta_2 - 10\sin\delta_2$$

and

$$P_{g2} - 1 = 1 - \cos\delta_2 + 10\sin\delta_2.$$

Replacing P_{g1}, P_{g2}, and δ_2 by x_1, x_2, and x_3, respectively, produces the following nonlinear programming problem:

$$\text{Minimize} \quad f(x) = f_1(x_1) + f_2(x_2)$$

$$\text{subject to} \quad h_1(x) = \cos x_3 + 10\sin x_3 + x_1 - 4 = 0.$$

$$h_2(x) = \cos x_3 - 10\sin x_3 + x_2 - 2 = 0.$$

The non-convex nature of $h_1^2(x)$ and $h_2^2(x)$, as shown respectively in Figures 6.6 and 6.7, shows the difficulty encountered in solving such a problem. But the smoothness of these two functions suggests a high probability that the $E_2(x)$ thus formed will have a local convexity around the points for which $h_1(x)$=0 and $h_2(x)$=0. By presuming the existence of such points, this problem is mapped into the network of equation 4.1.

Since there is no operation limits in this particular example, the initial values for x_1 and x_2 are both chosen to be zero. If desired, artificial constraints $x_1 \geq 0$ and $x_2 \geq 0$ may be added to the network to ensure that the solution of these two variables stays positive. The phase angle variable (x_3 in this example) is generally assumed to be close to 0 [156], thus $x_3(0)$=0 is chosen. The simulation results are tabulated in Table 6.4 with s=100 and s=200 and are compared to the solution obtained by the 2-phase network with s=100 and ε=0.1. Though the exact solution to the problem is not

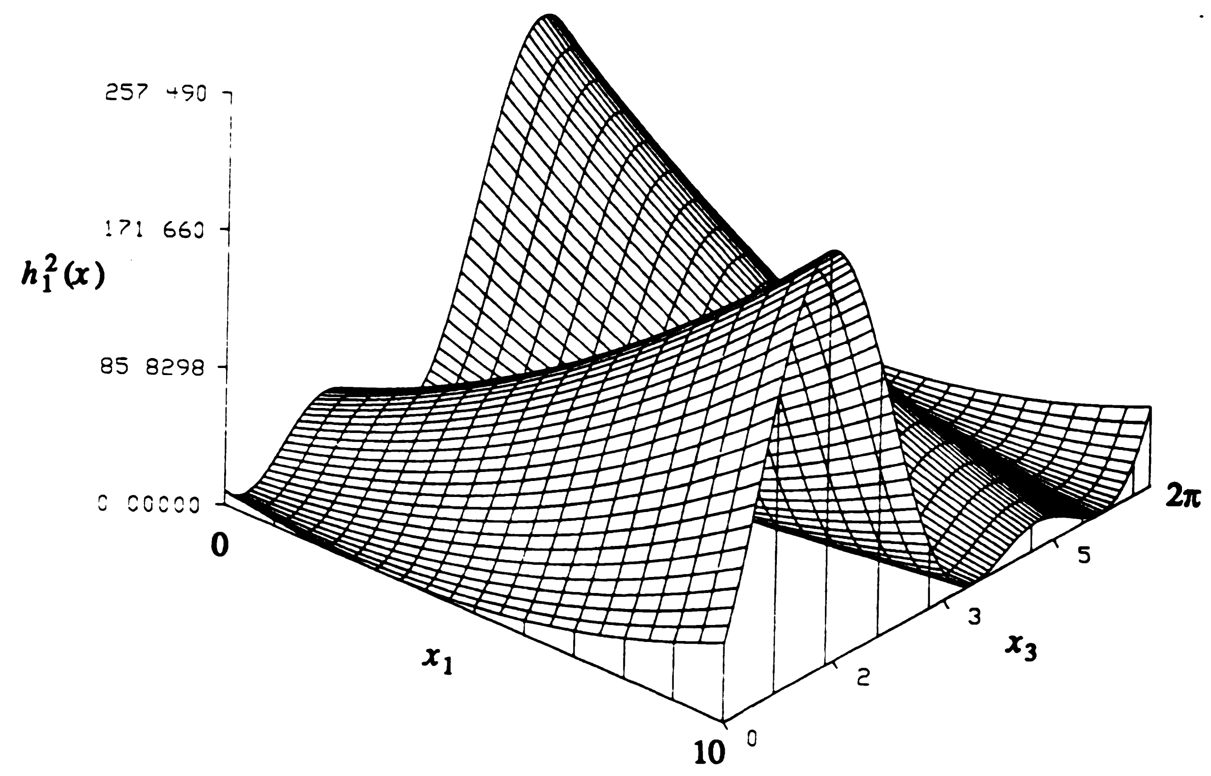


Figure 6.6. The surface of $h_1^2(x)$.

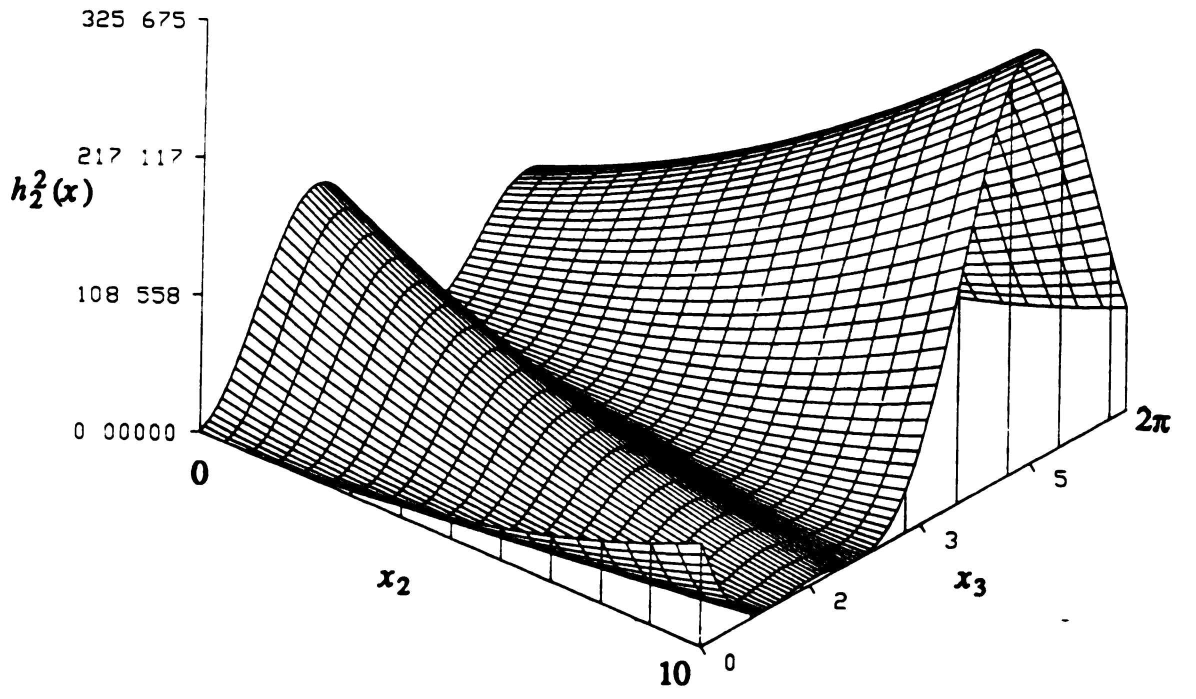


Figure 6.7. The surface of $h_2^2(x)$.

Table 6.4. Simulation results for Example 6.3.

Networks	x_1	x_2	x_3	$f(x)$	$E_2(x)$
Equation 4.1, s=100	0.52666	3.45379	0.23881	10.5500	10.7147
Equation 4.1, s=200	0.53467	3.48660	0.25015	10.7138	10.7146
2-phase net, s=100, ε=0.1	0.53938	3.52372	0.25187	10.8824	10.8824

available, the solution obtained by the 2-phase network may nonetheless be regarded as nearly exact since at this solution the constraint violations are less than 1×10^{-6}. Various initial points confirm the robustness of results obtained by these three networks. The exact Lagrange multipliers for $h_1(x)$ and $h_2(x)$ obtained by the 2-phase network are

$$\lambda_1 = -4.23629 \quad \text{and} \quad \lambda_2 = -4.02371,$$

respectively. □

It is worth noting that although a large value of s results in an equilibrium point closer to the exact solution, thus giving a better initial point for the second phase dynamics of a 2-phase network, a 2-phase network with a smaller value of s can still converge the exact solution. This phenomenon eases the burden of selecting s in the 2-phase network.

Though the operation limits are not included in above example, they may be easily incorporated into the network, since they are linear and define a hyper-rectangle (naturally convex) in R^n. Other criteria, such as pollution criterion, security criterion, and load shedding criterion, may replace or be added to the operation cost criterion to make the OPF problem more general. The same network formulation may be applied to all these different OPF problems as long as the objective functions are C^1 functions.

Furthermore, when compared to the OPF method in [155], the optimization network technique in this thesis is also superior in that first, there is no need for Jacobian matrix inversion. This is a necessity for most other OPF methods and is very time-consuming. Second, the solution for the power flow equations is obtained simultaneously since the power flow constraints are satisfied automatically when the OPF is solved. In fact, viewing the power flow equations (PFE) as a sub-problem of a generic OPF problem with a null objective function, the optimization network technique can solve the PFE as well. To illustrate such an idea, consider the follow example.

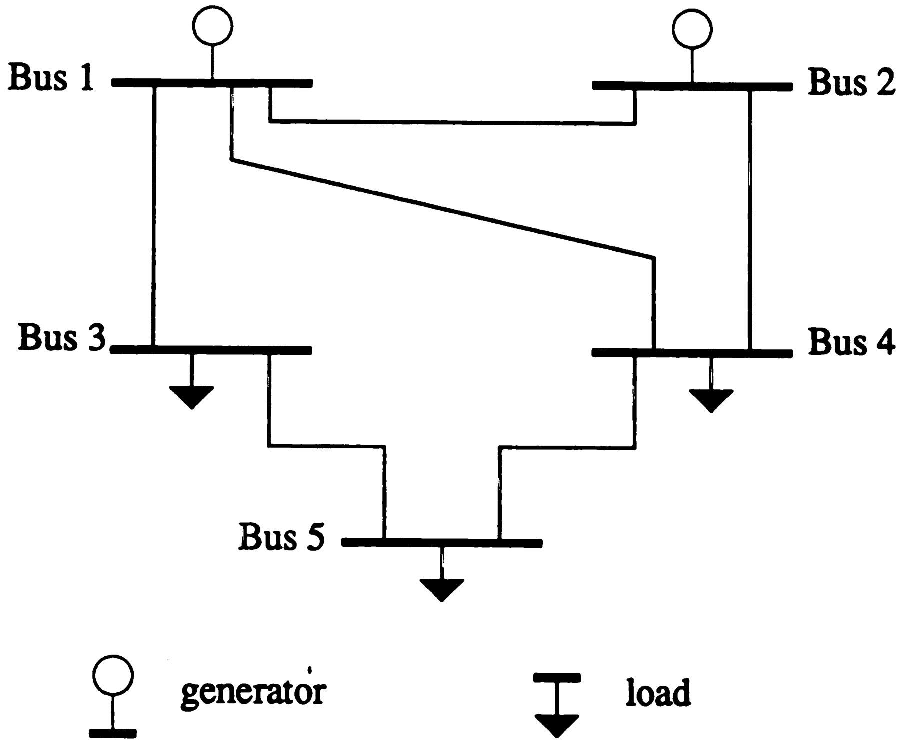


Figure 6.8. A 5-bus power system used in Example 6.4.

Example 6.4: A 5-bus power system is connected as in Figure 6.8. The bus data and the line data for this system are given in Tables 6.5 and 6.6, respectively. All the variables in Table 6.5 have been normalized. Bus 1 is the swing bus for which the voltage is constant and its phase angle is zero (δ_1=0). Bus 2 is a generation bus, i.e., a PV bus for which the voltage and the real power are constant. Busses 3-5 are strictly load busses (PQ busses) because their real power *P* and reactive power *Q* are constant.

Table 6.5. Bus data for the 5-bus system.

Bus #	Gen (pu MW)	Voltage (pu KV)	P load (pu MW)	Q load (pu MVAR)
Swing 1	0.00	1.05	0.00	0.00
2	0.80	1.07	0.00	0.00
3	0.00	1.05	0.50	-0.30
4	0.00	1.05	0.50	0.30
5	0.00	1.05	0.50	-0.20

Table 6.6. Line data for the 5-bus system.

Line #	From bus #	To bus #	R	X	G	B
1	1	2	0.10	0.20	2.00	-4.00
2	1	3	0.30	0.40	1.20	-1.60
3	1	4	0.10	0.30	1.00	-3.00
4	2	4	0.15	0.20	2.40	-3.20
5	3	5	0.10	0.20	2.00	-4.00
6	4	5	0.10	0.30	1.00	-3.00

The impedance of the transmission line from bus *i* to *j* is

$$Z_{ij} = R_{ij} + jX_{ij},$$

and the corresponding admittance is

$$Y_{ij} = \frac{1}{Z_{ij}} = G_{ij} + jB_{ij}.$$

A matrix Y_{bus} called the *bus admittance matrix* is often used when deriving the power flow equations [156]. Y_{bus} is a symmetric $n \times n$ matrix (n is the number of busses in the system) with elements

y_{ii} = sum of admittances of $\prod$-equivalent circuit elements incident to the ith bus, and

y_{ik} = -(admittance of $\prod$-equivalent circuit element bridging the ith and kth busses).

Denoting $y_{ik} = g_{ik} + jb_{ik}$, the power flow equations at bus i are

$$P_i = \sum_{k=1}^{n} |V_i||V_k|[g_{ik}\cos(\delta_i-\delta_k) + b_{ik}\sin(\delta_i-\delta_k)] \tag{6.3}$$

and

$$Q_i = \sum_{k=1}^{n} |V_i||V_k|[g_{ik}\sin(\delta_i-\delta_k) - b_{ik}\cos(\delta_i-\delta_k)], \tag{6.4}$$

where δ_i is the phase angle of the voltage V_i at bus i, and P_i (Q_i) is the real (reactive) power delivered from bus i. Also

$$P_i = P_{gi} - P_{li} \tag{6.5}$$

and

$$Q_i = Q_{gi} - Q_{li} \tag{6.6}$$

where $P_{gi}+jQ_{gi}$ is the complex power generation and $P_{li}+jQ_{li}$ is the complex power load at bus i.

By equations 6.3 and 6.4, a system with n busses has $2n$ power flow equations, but only part of them are used to solve the system variables V_i and δ_i. For the example considered here, both real and reactive power flow equations are needed for busses 3-5 since they are strictly load busses. The real power flow equation for bus 2 is also needed because it is a voltage controlled generation bus. The real and reactive generation of bus 1 and the reactive generation of bus 2 do not contribute to the solution of the problem; they are automatically obtained when the problem is solved by the seven other power flow equations. Since δ_1 and V_1 are fixed for the swing bus and V_2 is fixed for the voltage-controlled bus, there are only seven variables left in the system, δ_i for i=2 to 4 and V_j for j=3 to 5. It is thus a nonlinear system with seven equations and seven unknowns as follows.

$$h_1(V,\delta) = V_2^2 g_{22} + V_2[V_1(g_{21}\cos(\delta_2-\delta_1)+b_{21}\sin(\delta_2-\delta_1))$$

$$+ V_4(g_{24}\cos(\delta_2-\delta_4)+b_{24}\sin(\delta_2-\delta_4))] - P_{g2} = 0$$

$$h_2(V,\delta) = V_3^2 g_{33} + V_3[V_1(g_{31}\cos(\delta_3-\delta_1)+b_{31}\sin(\delta_3-\delta_1))$$

$$+ V_5(g_{35}\cos(\delta_3-\delta_5)+b_{35}\sin(\delta_3-\delta_5))] + P_{l3} = 0$$

$$h_3(V,\delta) = -V_3^2 b_{33} + V_3[V_1(g_{31}\sin(\delta_3-\delta_1)-b_{31}\cos(\delta_3-\delta_1))$$

$$+ V_5(g_{35}\sin(\delta_3-\delta_5)-b_{35}\cos(\delta_3-\delta_5))] + Q_{l3} = 0$$

$$h_4(V,\delta) = V_4^2 g_{44} + V_4[V_1(g_{41}\cos(\delta_4-\delta_1)+b_{41}\sin(\delta_4-\delta_1))$$

$$+ V_2(g_{42}\cos(\delta_4-\delta_2)+b_{42}\sin(\delta_4-\delta_2))$$

$$+ V_5(g_{45}\cos(\delta_4-\delta_5)+b_{45}\sin(\delta_4-\delta_5))] + P_{l4} = 0$$

$$h_5(V,\delta) = -V_4^2 b_{44} + V_4[V_1(g_{41}\sin(\delta_4-\delta_1)-b_{41}\cos(\delta_4-\delta_1))$$

$$+ V_2(g_{42}\sin(\delta_4-\delta_2)-b_{42}\cos(\delta_4-\delta_2))$$

$$+ V_5(g_{45}\sin(\delta_4-\delta_5)-b_{45}\cos(\delta_4-\delta_5))] + Q_{l4} = 0$$

$$h_6(V,\delta) = V_5^2 g_{55} + V_5[V_3(g_{53}\cos(\delta_5-\delta_3)+b_{53}\sin(\delta_5-\delta_3))$$

$$+ V_4(g_{54}\cos(\delta_5-\delta_4)+b_{54}\sin(\delta_5-\delta_4))] + P_{l5} = 0$$

$$h_7(V,\delta) = -V_5^2 b_{55} + V_5[V_3(g_{53}\sin(\delta_5-\delta_3)-b_{53}\cos(\delta_5-\delta_3))$$

$$+ V_4(g_{54}\sin(\delta_5-\delta_4)-b_{54}\cos(\delta_5-\delta_4))] + Q_{l5} = 0$$

To apply the optimization network, an energy function $E(V,\delta)$ is chosen as

$$E(V,\delta) = \frac{1}{2}\sum_{i=1}^{7} h_i^2(V,\delta). \tag{6.7}$$

Denoting $x = [V_3\ V_4\ V_5\ \delta_2\ \delta_3\ \delta_4\ \delta_5]^T$ and substituting in the values of V_1, V_2, and δ_1, the energy function becomes

$$E(x) = \frac{1}{2}\sum_{i=1}^{7} h_i^2(x). \tag{6.8}$$

Viewing equation 6.8 as a nonlinear programming problem with the objective function $\sum_{i=1}^{7} h_i^2(x)$ and no constraints, the one-phase Kennedy and Chua network [19] is sufficient to solve the problem. This is similar to the case where the minimizer of a convex problem lies in the relative interior of the feasible region. In the problem considered here the feasible region is simply the whole space. With such a viewpoint in mind, the network structure for solving this PFE problem is formulated as

$$\dot{x} = -(\sum_{i=1}^{7} h_i(x)\nabla h_i(x)). \tag{6.9}$$

The network of equation 6.9 is simulated by using the initial conditions given in Table 6.5. Following the common practice for a *flat start* in solving PFE, the initial values of V_3, V_4, and V_5 are chosen to be the same as the voltage magnitude of the swing bus. Though not shown in Table 6.5, the initial values of all phase angles are zero. Simulation results are given in Table 6.7. The values of P_{g1}, Q_{g1}, and Q_{g2} are obtained by substituting V_i's and δ_i's into equations 6.3 to 6.6. The results have been confirmed by a traditional method converging to the exact solution for the problem. Compared to the traditional methods for solving the PFE problem, the optimization network technique eliminates the most time-consuming computation (inversion of the Jacobian matrix) and results in the same exact solution. □

Table 6.7. Power flow bus output for the 5-bus system.

Bus #	Voltage	Phase Angle		Generation		Load	
		degrees	radians	PG	QG	PL	QL
1	1.050	0.000	0.0000	0.926	0.016	0.000	0.000
2	1.070	0.837	0.0146	0.800	0.170	0.000	0.000
3	0.964	-16.529	-0.2885	0.000	0.000	0.500	-0.300
4	0.958	-5.932	-0.1035	0.000	0.000	0.500	0.300
5	0.959	-16.530	-0.2885	0.000	0.000	0.500	-0.200

CHAPTER VII
CONCLUSION

Solving optimization problems is one of the major applications for engineering-oriented ANNs. Due to the lack of a more theoretically sound basis, however, the optimization formulations of artificial neural networks have been limited in applicability and have caused overstatement of what the artificial neural networks can do, thus abating their creditability. This shortcoming has been resolved as far as solving non-linear programming problems by ANNs is concerned because of the outcome of this work.

7.1 Summary

Based on optimization theory, it has been shown that the network by Kennedy and Chua, or equivalently the network of equation 4.1 fulfills both the Kuhn-Tucker optimality conditions and the penalty function method. From this viewpoint, the network structure of Hopfield and Tank is thus invalid and the network by Rodriguez-Vazquez, *et al.* is a special case of equation 4.1 for $s=\infty$.

The optimization network theory was derived in detail in Chapter 4. The key idea is to use the energy function or the penalty function as a Lyapunov function. The optimization network formed by equation 4.1 traverses along the surface of the energy function in a manner of steepest descent and eventually settles on a local minimum of the energy function. If the problem is a linear program, a quadratic program, or a convex program, this local minimum is also a global minimum. Proper pseudo- and quasi-convexity assumptions placed on the original problem also lead to a global minimum. For a nonlinear program, however, the minimum is ensured only locally. By the theorem of the penalty function method, the local minimum obtained by the network is an approximation to the exact solution, and these two can be moved arbitrarily close by making the penalty parameter sufficiently large.

One interesting application of the quadratic programming network is in solving the least squares problem. The network may be used to solve regular least squares problems as well as the non-negative least squares and least distance problem [157]. The regular least squares problem seeks the least squares solutions in the sense of l_2 norm. The network in fact may be used to solve the least square solutions in the sense of l_p norm, for $1<p<\infty$, with or without constraints. This is because the l_p norm for $1<p<\infty$ has been shown to be strictly convex [158].

It has also been shown in Chapter 4 that the optimization network formulation by equation 4.1 is much like the gradient projection method. But they differ in that the latter results in a solution in the feasible region while the former may end up at a point in the infeasible region whenever there are binding constraints in the exact solution. The reason for such a slight infeasibility of the solution is due the fact that the energy function used by the optimization network originates from the penalty function method.

For most nonlinear programming problems, the approximate solution obtained by the network is acceptable. But there are cases when the feasibility is very crucial to the problem. A new 2-phase network formulation has been developed in Chapter 4 to achieve such a goal. This network additionally provides the corresponding Lagrange multipliers for each constraint. This is specially useful for linear programming problems because the Lagrange multipliers solve the dual problem of the primary program. Though the 2-phase network has not been proven in a rigid argument, the simulation results nevertheless show its robustness in converging to an exact solution.

Through simulation, it has been shown in Chapter 6 that the networks (1-phase and 2-phase) are useful in solving EPD problems with or without the consideration of transmission line losses, as well as OPF problems with all kinds of constraints as long as the constraints are continuously differentiable. The network of equation 4.1 also solves exactly $h(x)=0$ for $h:R^n \rightarrow R^m \in C^1$, no matter whether h is linear or not, as exemplified by the power flow equations problem. Reduced to one dimension, this is equivalent to finding the root of $f(x)=0$ for $f:R \rightarrow R \in C^1$.

The optimization ANNs developed in this work are applicable to a very large class of (constrained or unconstrained) nonlinear programming problems with $f \in C^1$ and $g_i^2 \in C^1$ for all i. For the unconstrained problems, they may be solved exactly by equation 4.1, similarly to the constrained problems with the solution in the relative interior of the feasible region. In such cases, the network performs similarly to a gradient descent method. For a general constrained problem, the 2-phase network may be used. If a problem has only constraints, $h(x)=0$ and $g(x) \leq 0$, the objective function is formed by summing $h^2(x)$ and $(g^+(x))^2$ and the sum may then be regarded an as unconstrained problem and be solved accordingly. A good example is the solution of the power flow equations as shown in Chapter 6.

Note that in applying the optimization network, the problem must be feasible. Furthermore, the regularity assumption and the boundedness of the solution set must be met. This raises no difficulty at all, since in the practice of nonlinear programming the regularity of the solution is normally presumed for large systems, and the boundedness of the solution may be achieved by adding some artificial upper and/or lower bound constraints.

As mentioned above the optimization ANNs are similar in theory to many of the existing optimization methods. In fact, the networks may be regarded as continuous optimization systems in contrast to the discrete nature of many optimization processes. Any nonlinear programming problem which is solvable by the traditional discrete methods is also solvable by the optimization ANNs developed in this work. Traditional discrete optimization methods such as the Newton's method and Newton-Raphson method often demand the calculation of the inverse of the Jacobian matrix and then update the state variables by difference equations, which is a function of the inverse Jacobian. The calculation of the inverse Jacobian grows factorially in computational complexity with respect to the size of the Jacobian and thus restricts the applicability of these discrete optimization methods to large-scale systems. The optimization ANNs, on the other hand, require no matrix inversion and, consequently, have more potential for use with larger systems.

The above argument disagrees with the generalized networks proposed by Tsirukis, *et al.*[159] In their networks, the difference equations used in traditional discrete optimization processes are transformed into differential equations in which the inverse Jacobian is included. But since the differential equations describe a dynamic system and there is no software algorithm nor hardware structure that can calculate the inverse Jacobian instantaneously (not even for a small system), their networks are not hardware implementable. Even when implementing their networks in software, the inverse Jacobian in the differential equations actually slows down the convergence rate when compared to the software implementation of difference equations in traditional optimization methods. This make their networks less useful. So a direct copy from difference equations to differential equations does not necessarily yield any advantage.

7.2 Contributions

The following are the salient contributions of this dissertation:

(1) ANNs for optimization have been analyzed from the viewpoint of optimization theory leading to discovery of the reasons why the ANNs succeed or fail.

(2) The optimization network theory for linear programming, quadratic programming, convex programming, and nonlinear programming has been derived and a 2-phase optimization network has been developed. These results lay a mathematically sound foundation for the optimization ANNs and extend the applicability of ANNs.

(3) The quality of the solutions obtained by the optimization ANNs has been quantified through simulation. The solutions of the 1-phase networks are adjustable by tuning the penalty parameter and the solutions of the 2-phase networks are exact.

(4) The applicability of the optimization ANNs for solving real-world problems like the economic power dispatching problem and the optimal power flow problem has been demonstrated. It was shown that the mapping technique of the optimization ANNs is simple and that they are able to handle various kinds of constraint sets. It was demonstrated that the optimization ANNs attain a better objective function value.

As a whole, this work lays a solid groundwork for the optimization ANNs in both the theoretical and practical aspects. As far as solving nonlinear programming problems by ANNs is concerned, the work is completed, except perhaps a more rigid analysis of the 2-phase network. The optimization ANNs also suggest a possible structure for the next-generation analog computer. Future research should emphasize developing suitable hardware implementation for the optimization ANNs so as to fully exploit their capability.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Hopfield, J.J., "Artificial Neural Networks," *IEEE Circuits and Devices Magazine*, pp. 3-10, September 1988.

[2] Tank, D.W. and Hopfield, J.J., "Collective Computation in Neuronlike Circuits," *Scienific American*, pp. 104-114, December 1987.

[3] Hopfield, J.J. and Tank, D.W., "Computing with Neural Circuits: A Model," *Science*, vol. 233, no. 4764, pp. 625-633, August 8, 1986.

[4] Hopfield, J.J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the National Academy of Science USA*, vol. 79, pp. 2554-2558, April 1982.

[5] Hopfield, J.J., "Neurons with Graded Response Have Collective Computational Properties like those of Two-state Neurons," *Proceedings of the National Academy of Science USA*, vol. 81, pp. 3088-3092, May 1984.

[6] Tank, D.W. and Hopfield, J.J., "Simple Neural Optimization Networks: An A/D Converter, Signal Decision Network, and a Linear Programming Circuit," *IEEE Transactions on Circuits and Systems*, vol. CAS-33, no. 5, pp. 533-541, 1986.

[7] Hopfield, J.J. and Tank, D.W., "'Neural' Computation of Decisions in Optimization Problems," *Biological Cybernetics*, vol. 52, pp. 141-152, 1985.

[8] Bagherzadeh, N., *et al.*, "On Parallel Execution of the Traveling Salesman Problem on a Neural Network Model," *Proceedings of the IEEE First International Conference on Neural Networks*, San Diego, CA, vol. III, pp. 317-324, July 1987.

[9] Brandt, R.D., *et al.*, "Alternative Networks for Solving the Traveling Salesman Problem and the List-Matching Problem," *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, CA, vol. II, pp. 333-340, July 1988.

[10] Szu, H., "Fast TSP Algorithm Based on Binary Neuron Output and Analog Neuron Input Using the Zero-Diagonal Interconnect Matrix and Necessary and Sufficient Constraints of the Permutation Matrix," *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, CA, vol. II, pp. 259-266, July 1988.

[11] Wacholder, E., Han, J. , and Mann, R.C., "An Extension of the Hopfield-Tank Model for Solution of the Multiple Traveling Salesmen Problem," *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, CA, vol. II, pp. 305-324, July 1988.

[12] Gunn, J.P. and Weidlich, R.B., "A Derivative of the Hopfield-Tank Neural Network Model that Reliably Solved the Traveling Salesman Problem," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. II, pp. 588, June 1989.

[13] Angeniol, B., *et al.*, "Self-Organizing Feasture Maps and the Traveling Salesman Problem," *Neural Networks*, vol. 1, pp. 289-293, 1988.

[14] Culioli, J.-C., *et al.*, "Neural Network Models for Linear Programming," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., pp. 293-296, January 1990.

[15] Culioli, J.-C., *et al.*, "A Neural Network for Explicitly Bounded Linear Programming," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., pp. 381-384, January 1990.

[16] Foo, Y.S. and Takefuji, Y., "Integer Linear Programming Neural Networks for Job-Shop Scheduling," *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, CA, vol. II, pp. 341-348, July 1988.

[17] Yao, Y. and Yang, Q, "Programming Neural Networks: A Dynamic-Static Model," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., pp. 345-348, January 1990.

[18] Kennedy, M.P. and Chua, L.O., "Unifying the Tank and Hopfield Linear Programming Circuit and the Canonical Nonlinear Programming Circuit of Chua and Lin," *IEEE Transactions on Circuits and Systems*, vol. CAS-34, no. 2, pp. 210-214, February 1987.

[19] Kennedy, M.P. and Chua, L.O., "Neural Networks for Nonlinear Programming," *IEEE Transactions on Circuits and Systems*, vol. CAS-35, no. 5, pp. 554-562, May 1988.

[20] Tsirukis, A.G., Reklaitis, G.V. and Tenorio, M.F., "Nonlinear Optimization Using Generalized Hopfield Networks," *Neural Computation*, vol. 1, no. 4, pp. 511-521, 1989.

[21] Smith, M.J.S. and Portmann, C.L., "Practical Design and Analysis of a Simple 'Neural' Optimization Circuit," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 1, pp. 42-50, January 1989.

[22] Rodriguez-Vazquez, A., *et al.*, "Analog Integrated Neural-Like Circuits for Nonlinear Programming," *Proceedings of 32nd Midwest Symposium on Circuits and Systems*, Champaign, IL, pp. 234-237, August 1989.

[23] Rodriguez-Vazquez, A., *et al.*, "Analog Neural Networks for Constrained Optimization," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[24] Rodriguez-Vazquez, A., *et al.*, "Nonlinear Switched-Capacitor 'Neural' Networks for Optimization Problems," *IEEE Trans. on Circuits and Systems*, vol. CAS-37, no. 4, April 1990.

[25] Atlas, L. and Conner, J., "Existing and Potential ANN Techniques for Power Systems Applications," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[26] Damborg, M.J., El-Sharkawi, M.A. and Marks, R., "Potential Applications of Artificial Neural Networks to Power System Operation," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[27] Baum, E.B., "Towards Practical 'Neural' Computations for Combinatorial Optimization Problems," *Proceedings of the AIP Conference on Neural Networks for Computing*, Snowbird, UT, no. 151, pp. 53-58, 1986.

[28] Jeffrey, W. and Rosner, R., "Neural Network Processing as a Tool for Function Optimization," *Proceedings of the AIP Conference on Neural Networks for Computing*, Snowbird, UT, no. 151, pp. 241-246, 1986.

[29] Jeffrey, W. and Rosner, R., "Optimization Algorithms: Simulated Annealing and Neural Network Processing," *Astrophysical Journal*, vol. 310, pp. 473-481, November 1986.

[30] Kamgar-Parsi, B. and Kamgar-Parsi, B., "An Efficient Model of Neural Networks for Optimization," *Proceedings of the IEEE First International Conference on Neural Networks*, San Diego, CA, vol. III, pp. 785-790, July 1987.

[31] Kuczewski, R.M., "Neural Network Approaches to Multi-Target Tracking," *Proceedings of the IEEE First International Conference on Neural Networks*, San Diegon, CA, vol. IV, pp. 619-633, July 1987.

[32] Ramanujam, J. and Sadayappan, P., "Optimization by Neural Networks," *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, CA, vol. II, pp. 325-332, July 1988.

[33] Tagliarini, G.A. and Page, E.W., "Learning in Systematically Designed Networks," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. I, pp. 497-502, June 1989.

[34] Bankes, S.C., "Constrained Differential Optimization for Temporally Dynamic Problems," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. II, p. 587, June 1989.

[35] Kajiura, M., Akiyama, Y. and Anzai, Y., "Neural Networks vs. Tree Search in Puzzle Solving," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. II, p. 588, June 1989.

[36] Kitamura, S. and Qing, P., "Neural Network Application to Solve Fredholm Integral Equations of the First Kind," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. II, p. 589, June 1989.

[37] Li, T., Fang, L. and Wilson, W.H., "Parallel Approximate Solution of the 0/1 Knapsack Optimization on Competitive Neural Networks," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. II, p. 589, June 1989.

[38] Kamgar-Parsi, B. and Kamgar-Parsi, B., "On Problem Solving with Hopfield Neural Networks," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. II, p. 589, June 1989.

[39] Peng, Y., "A Connectionist Solution for Vertex Cover Problems," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. II, p. 590, June 1989.

[40] Rignot, E.J.M., "Three-Dimensional Point Pattern Tracking Using a Completely Determined Hopfield Network Independent of the Input Data," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. II, p. 590, June 1989.

[41] Thrift, P., "Function Optimization Using a Lattice of Neurons," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. II, p. 591, June 1989.

[42] Tanaka, T., *et al.*, "Optimal Task Assignment Using Neural Network," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. II, p. 591, June 1989.

[43] Graupe, D. and Liu, R., "A Neural Network Approach to Decomposing Surface EMG Signals," *Proceedings of 32nd Midwest Symposium on Circuits and Systems*, Champaign, IL, pp. 740-743, August 1989.

[44] Mjolsness, E., Garrett, C. and Miranker, W.L., "Multiscale Optimization in Neural Nets: A Preliminary Report," *International Joint Conference on Neural Networks*, San Diego, CA, June 1990.

[45] Maa, C.-Y. and Shanblatt, M.A., "Adjustment of Parameters for Signal Decision Networks," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. II, p. 589, June 1989.

[46] Zak, M., Toomarian, N. and Barhen, J., "Creative Dynamics Approach to Optimization Problems," *International Joint Conference on Neural Networks*, San Diego, CA, June 1990.

[47] Chua, L.O. and Yang, L., "Cellular Neural Networks: Theory," *IEEE Transactions on Circuits and Systems*, vol. CAS-35, no. 10, pp. 1257-1272, October 1988.

[48] Chua, L.O. and Yang, L., "Cellular Neural Networks: Applications," *IEEE Transactions on Circuits and Systems*, vol. CAS-35, no. 10, pp. 1273-1290, October 1988.

[49] Lippmann, R.P., "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, pp. 4-22, April 1987.

[50] Guez, A., Protopopsecu, V. and Barhen, J., "On the Stability, Storage Capacity, and Design of Nonlinear Continuous Neural Networks," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 1, pp. 80-87, 1988.

[51] Foo, Y.S. and Takefuji, Y., "Stochastic Neural Networks for Solving Job-Shop Scheduling: Part 1. Problem Representation," *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, CA, vol. II, pp. 275-282, July 1988.

[52] Foo, Y.S. and Takefuji, Y., "Stochastic Neural Networks for Solving Job-Shop Scheduling: Part 2. Architecture and Simulation," *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, CA, vol. II, pp. 283-290, July 1988.

[53] Levy, B.C. and Adams, M.B., "Global Optimization with Stochastic Neural Networks," *Proceedings of the IEEE First International Conference on Neural Networks*, San Diego, CA, vol. III, pp. 681-689, July 1987.

[54] Peterson, C. and Soderberg, B., "A New Method for Mapping Optimization Problems onto Neural Networks," *International Journal of Neural Systems*, vol. 1, no. 1, pp. 3-22, 1989.

[55] Bruck, J. and Goodman, J.W., "A Generalized Convergence Theorem for Neural Networks and its Applications in Combinatorial Optimization," *Proceedings of the IEEE First International Conference on Neural Networks*, San Diego, CA, vol. III, pp. 649-656, July 1987.

[56] Kennedy, M.P. and Chua, L.O., "Circuit Theoretic Solutions for Neural Networks - An Old Approach to a New Problem," *Proceedings of the IEEE First International Conference on Neural Networks*, San Diego, CA, vol. II, pp. 169-176, July 1987.

[57] Grujic, L.T. and Michel, A.N., "Qualitative Analysis of Neural Networks Under Structural Perturbations," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[58] Michel, A.N., Farrell, J.A. and Porod, W., "Qualitative Analysis of Neural Networks," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 2, pp. 229-243, 1989.

[59] Li, J.-H., Michel, A.N. and Porod, W., "Analysis and Synthesis of a Class of Neural Networks: Variable Structure Systems with Infinite Gain," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 5, pp. 713-731, 1989.

[60] Li, J.-H., Michel, A.N. and Porod, W., "Analysis and Synthesis of a Class of Neural Networks: Linear Systems Operating on a Closed Hypercube," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 11, pp. 1405-1422, 1989.

[61] Kohonen, T., *Self-Organization and Associative Memory*, Series in Information Sciences, vol. 8, Spring-Verlag, 2nd ed., 1988.

[62] Stryer, L., *Biochemistry*, W.H. Freeman and Company, 2nd ed., 1981.

[63] Minsky, M. and Papert, S., *Perceptrons*, MIT Press, MA, 1969.

[64] Kung, S.Y. and Hwang, J.N., "Parallel Architectures for Artificial Neural Nets," *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, CA, vol. II, pp. 165-172, July 1988.

[65] Anderson, J.A. and Rosenfeld, E., Eds., *Neurocomputing: Foundations of Reserach*, MIT Press, MA, 1988.

[66] Grossberg, S., "How does a Brain Build a Cognitive Code?" *Psychological Review*, vol. 87, pp. 1-51, 1980.

[67] Cohen, M.A. and Grossberg, S., "Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Network," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, no. 5, pp. 815-826, 1983.

[68] Grossberg, S., *The Adaptive Brain, II: Vision, Speech, Language, and Motor Control*, North-Holland, Amsterdam, 1987.

[69] Grossberg, S., Eds., *Neural Networks and Natural Intelligence*, MIT Press, MA, 1988.

[70] Kohonen, T., "Self-organized Formation of Topologically Correct Feature Maps," *Biological Cybernetics*, vol. 43, pp. 59-69, 1982.

[71] McClelland, J.L. and Rumelhart, D.E., "An Interactive Activation Model of Context Effects in Letter Perception: Part 1. An Account of Basic Findings," *Psychological Review*, vol. 88, pp. 375-407, 1981.

[72] McClelland, J.L. and Rumelhart, D.E., "An Interactive Activation Model of Context Effects in Letter Perception: Part 2. The Contextual Enhancement Effect and Some Tests and Extensions of the Model," *Psychological Review*, vol. 89, pp. 60-94, 1982.

[73] McClelland, J.L. and Rumelhart, D.E., Eds., *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, vol. 1, MIT Press, MA, 1986.

[74] McClelland, J.L. and Rumelhart, D.E., Eds., *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, vol. 2, MIT Press, MA, 1986.

[75] Rumelhart, D.E., Hinton, G.E., and Williams, R.J., "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, vol. 1, D.E. Rumelhart and J.L. McClelland (Eds.), MIT Press, MA, pp. 318-362, 1986.

[76] Rumelhart, D.E., Hinton, G.E., and Williams, R.J., "Learning Representations by Back-Propagating Errors," *Nature*, vol. 323, pp. 533-536, 1986.

[77] *DARPA Neural Network Study*, AFCEA International Press, VA, 1988.

[78] Wilson, G.V. and Pawley, G.S., "On the Stability of the Travelling Salesman Problem Algorithm of Hopfield and Tank," *Biological Cybernetics*, vol. 58, pp. 63-70, 1988.

[79] Kahng, A.B., "Traveling Salesman Heuristics and Embedding Dimension in the Hopfield Model," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. I, pp. 513-520, June 1989.

[80] Maa, C.-Y. and Shanblatt, M.A., "Stability of Linear Programming Neural Network for Problems with Hypercube Feasible Region," *International Joint Conference on Neural Networks*, San Diego, CA, June 1990.

[81] Maa, C.-Y. and Shanblatt, M.A., "Improved Linear Programming Neural Networks," *Proceedings of 32nd Midwest Symposium on Circuits and Systems*, Champaign, IL, pp. 740-743, August 1989.

[82] Abe, S. and Kawakami, J., "Theories on the Hopfield Neural Networks with Inequality Constraints," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., pp. 349-352, January 1990.

[83] Chiu, C., Maa, C.-Y. and Shanblatt, M.A., "An Artificial Neural Network Algorithm for Dynamic Programming," *International Journal of Neural Systems*, to appear.

[84] Rauch, H.E. and Winarske, T., "Neural Networks for Routing Communication Traffic," *IEEE Control Systems Magazine*, pp. 26-31, April 1988.

[85] Zhang, L. and Thomopoulos, S.C.A., "Neural Network Implementation of the Shortest Path Algorithm for Traffic Routing in Communication Networks," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. II, p. 591, June 1989.

[86] Zhang, C.-X. and Mylinski, D.A., "VLSI Placement with a Neural Network Model," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[87] Prasitjutrakul, S. and Kubitz, W.J., "Path-Delay Constrained Floorplanning: A Mathematical Programming Approach for Initial Placement," *Proceedings of Design Automation Conference*, Las Vegas, NV, pp. 364-369, June 1989.

[88] Herrigel, A. and W. Fichtner, W., "An Analytic Optimization Technique for Placement of Marco-Cells," *Proceedings of Design Automation Conference*, Las Vegas, NV, pp. 376-381, June 1989.

[89] Libeskind-Hadas, R. and Liu, C.L., "Solutions to the Module Orientation and Rotation Problems by Neural Computation Networks," *Proceedings of Design Automation Conference*, Las Vegas, NV, pp. 400-405, June 1989.

[90] Caviglia, C., *et al.*, "Neural Algorithms for Cell Placement in VLSI Design," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. I, pp. 573-580, June 1989.

[91] Naft, J, "Neuropt: Neurocomputing for Multiobjective Design Optimization for Printed Circuit Board Component Placement," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. I, pp. 503-506, June 1989.

[92] Melsa, P.J.W. and Kenney, J.B., "A Neural Network Solution for Routing in Three Stage Interconnection Networks," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[93] Green, A.D.P. and Noakes, P.D., "Neural Network - Their Use for the Routing of Integrated Circuits," *Proceedings of 32nd Midwest Symposium on Circuits and Systems*, Champaign, IL, pp. 501-504, August 1989.

[94] Yih, J.-S. and Mazumder, P., "A Neural Network Design for Circuit Partitioning," *Proceedings of Design Automation Conference*, Las Vegas, NV, pp. 406-411, June 1989.

[95] Yu, M.-L., "A Study of the Applicability of Hopfield Decision Neural Nets to VLSI CAD," *Proceedings of Design Automation Conference*, Las Vegas, NV, pp. 412-417, June 1989.

[96] Sculley, T.L. and Brooke, M.A., "A Neural Network Approach to High Performance Analog Circuit Design," *Proceedings of 32nd Midwest Symposium on*

Circuits and Systems, Champaign, IL, pp. 497-500, August 1989.

[97] Osowski, S., "Electrical Circuits Optimization Using Standard Network Analysers," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[98] Starzyk, J.A. and El-Gamal, M., "Artificial Neural Network for Testing Analog Circuits," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[99] Xiangming, X. and Spence, R., "A Fast Constrained Optimization Algorithm for IC Design," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[100] Thomas, R.J., *et al.*, "On-Line Security Classification Using an Artificial Neural Network," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[101] Chow, J.-C., *et al.*, "An Improved Hopfield Model for Power System Contingency Classification," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[102] Pao, Y.-H., Sobajic, D.J. and Nyo, W., "Real-Time Security Monitoring of Electric Power Systems Using Parallel Associative Memories," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[103] Mori, H. and Tsuzuki, S., "Determination of Power System Topological Observability Using the Boltzmann Machine," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[104] Karady, G. and Hubele, N.F., "Conceptual Approach to the Application of Neural Network for Short Term Load Forecasting," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[105] Maa, C.-Y. and Shanblatt, M.A., "A Constrained Optimization Neural Net Technique for Economic Power Dispatch," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[106] Maa, C.-Y. and Shanblatt, M.A., "A Neural Net Technique for Economic Power Dispatching with Consideration of Transmission Losses," submitted to, *International Journal of Neural Systems*.

[107] Matsuda, S. and Akimoto, Y. "The Representation of Large Numbers in Neural Networks and its Application to Economical Load Dispatching of Electric Power," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. I, pp. 587-592, June 1989.

[108] Aggoune, M.E., *et al.*, "Artificial Neural Networks for Power System Static Security Assessment," *Proceedings of IEEE International Symposium on Circuits and System*, Portland, OR, pp. 490-494, May 1989.

[109] Fischl, R., *et al.*, "Screening Power System Contingencies Using a Back-propagation Trained Multiperceptron," *Proceedings of IEEE International Symposium on Circuits and System*, Portland, OR, pp. 486-489, May 1989.

[110] Chan, E.H., "Alarm Pattern Recognition by Using Artificial Neural Networks," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[111] Di Zitti, E., *et al.*, "Efficient Emulation of Neural Networks on Concurrent Architectures for Optimization Problems," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[112] Park, S., "Hopfield Neural Network for AR Spectral Estimator," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[113] Gao, K., Ahmad, M.O. and Swamy, M.N.S., "A Neural Network Least-Square Estimator," *International Joint Conference on Neural Networks*, San Diego, CA, June 1990.

[114] Takefuji, Y. and Lee, K.C., "A Two Step Sorting Algorithm," *International Joint Conference on Neural Networks*, San Diego, CA, June 1990.

[115] Sudharsanan, S.I. and Sundareshan, M.K., "Equilibrium Uniqueness and Global Exponential Stability of a Neural Network for Optimization Applications," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., pp. 472-475, January 1990.

[116] Sudharsanan, S.I. and Sundareshan, M.K., "Neural Network Computational Algorithm for Least Squares Estimation Problem," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. II, pp. 590, June 1989.

[117] Szu, H.H., "Reconfigurable Neural Nets by Energy Convergence Learning Principle Based on Extended McCulloch-Pitts Neurons and Synapses," *Proceedings*

of International Joint Conference on Neural Networks, Washington, D.C., vol. I, pp. 485-496, June 1989.

[118] van Laarhoven, P.J.M. and Aarts, E.H.L., *Simulated Annealing: Theory and Applications*, Reidel, Dordrecht, 1987.

[119] Van den Bout, D.E. and Miller III, T.K., "Graph Partitioning Using Annealed Neural Networks," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. I, pp. 521-528, June 1989.

[120] Akiyama, Y., *et al.*, "Combinatorial Optimization with Gaussian Machines," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. I, pp. 533-540, June 1989.

[121] de Carvalho, L.A.V. and Barbosa, V.C., "Toward a Stochastic Neural Model for Combinatorial Optimization," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. II, pp. 587, June 1989.

[122] Wong, W.S. and Funka-Lea, C.A., "An Elastic Net Solution to Obstacle Avoidance Tour Planning," *International Joint Conference on Neural Networks*, San Diego, CA, June 1990.

[123] Davenport, M.R. and Hoffmann, G.W., "An Extension of the Hopfield Neural Network to Include Hidden Neurons," *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1990.

[124] Yue, T.W. and Fu, L.C., "Ineffectiveness in Solving Combinatorial Optimization Problems Using a Hopfield Network: A New Perspective from Aliasing Effect," *International Joint Conference on Neural Networks*, San Diego, CA, June 1990.

[125] Hellstrom, B.J. and Kanal, L.N., "The Definition of Necessary Hidden Units in Neural Networks for Combinatorial Optimization," *International Joint Conference on Neural Networks*, San Diego, CA, June 1990.

[126] Abe, S., "Theories on the Hopfield Neural Networks," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., vol. I, pp. 557-564, June 1989.

[127] Abu-Mostafa, Y.S. and St. Jacques, J.-M., "Information Capacity of the Hopfield Model," *IEEE Transactions on Information Theory*, vol. IT-31, no. 4, pp. 461-464, 1985.

[128] McEliece, R.J., *et al.*, "The Capacity of the Hopfield Associative Memory," *IEEE Transactions on Information Theory*, vol. IT-33, no. 4, pp. 461-482, 1987.

[129] Davis, G.W., "Sensitivity Analysis of Hopfield Neural Net," *Proceedings of IEEE First International Conference on Neural Networks*, San Diego, CA, vol. III, pp. 325-528, July 1987.

[130] Reibling, L.A. and Olinger, M.D., "A Neural Network Implementation of Parallel Serach for Multiple Paths," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., pp. 293-296, January 1990.

[131] Coleland, B.R., "Global Minima within the Hopfield Hypercube," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., pp. 377-380, January 1990.

[132] Hirsch, M.W., "Convergent Activation Dynamics in Continuous Time Networks," *Neural Networks*, vol. 2, pp. 331-349, 1989.

[133] Salam, F., "A Tutorial Workshop on Neural Nets and Their Engineering Implementations," *31st Midwest Symposium on Circuits and Systems*, St. Louis, August 1988.

[134] Graf, H.P., *et. al.*, "VLSI Implementation of a Neural Network Memory with Several Hundred Neurons," *Proceedings of the AIP Conference on Neural Networks for Computing*, Snowbird, UT, no. 151, pp. 182-187, 1986.

[135] Graf, H.P., Hubbard, W., Jackel, L.D., and deVegvar, P.G.N., "A CMOS Associative Memory Chip," *Proceedings of the IEEE First International Conference on Neural Networks*, San Diego, CA, vol. III, pp. 461-468, July 1987.

[136] Graf, H.P., Jackel, L.D., and Hubbard, W.E., "VLSI Implementation of a Neural Network Model," *Computer*, vol. 21, no. 3, pp. 41-49, March 1988.

[137] Sivilotti, M.A., Emerling, M.R., and Mead, C.A., "VLSI Architectures for Implementation of Neural Networks," *Proceedings of the AIP Conference on Neural Networks for Computing*, Snowbird, UT, no. 151, pp. 408-413, 1986.

[138] Sivilotti, M.A., Mahowald, M.A., and Mead, C.V, "Real-time Visual Computations Using Analog CMOS Processing Arrays," *Advanced Research in VLSI: Proceedings of the 1987 Stanford Conference*, P. Losleben (Ed.), MIT Press, MA, pp. 295-312, 1987.

[139] Mead, Carver, *Analog VLSI and Neural Systems*, Addison-Wesley, MA, 1989.

[140] Hopfield, J., "Neural Networks: Algorithm and Microhardware," *Tutorial #7, International Joint Conference on Neural Networks*, Washington, D.C., June 1989.

[141] Jackel, L., "VLSI Technology and Neural Network Chips," *Tutorial #8, International Joint Conference on Neural Networks*, Washington, D.C., June 1989.

[142] Murray, A.F. and Smith, A.V.W., "Asynchronous VLSI Neural Networks Using Pulse-Stream Arithmetic," *IEEE Journal of Solid-State Circuit*, vol. 23, no. 3, pp. 688-697, 1989.

[143] Salam, F., "A Formulation for the Design of Neural Processors," *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, CA, vol. I, pp. 173-180, July 1988.

[144] Li, J.H., Michel, A.N., and Porod, W., "Qualitative Analysis and Synthesis of a Class of Neural Networks," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 8, pp. 976-986, August 1988.

[145] Rockafellar, R.T., *Convex Analysis*, Princeton University Press, NJ, 1979.

[146] Bazaraa, M.S. and Shetty, C.M., *Nonlinear Programming*, John Wiley & Sons, NY, 1979.

[147] Luenberger, D.G., *Introduction to Linear and Nonlinear Programming*, Chapter 12, Addison-Wesley, MA, 1973.

[148] Chua, L.O. and Lin, G.-N., "Nonlinear Programming without Computation," *IEEE Transactions on Circuits and Systems*, vol. CAS-31, no. 2, pp. 182-188, Feb. 1984.

[149] Chua, L.O. and Wang N.N., "Complete Stability of Autonomous Reciprocal Nonlinear Networks," *Circuit Theory and Applications*, vol. 6, pp. 211-241, 1978.

[150] Vidyasagar, M., *Nonlinear System Analysis*, Chapter 5, Prentice-Hall, NJ, 1978.

[151] Fletcher, R., *Practical Methods of Optimization*, 2nd ed., John Wiley & Sons, NY, 1987.

[152] Bertsekas, D.P., *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, NY, 1982.

[153] Wood, A.F. and Wollenberg, B.F, *Power Generation Operation and Control*, Chapter 3, John Wiley & Sons, NY, 1984.

[154] Stott, B., Alsac, O., and Monticelli, A.J., "Security Analysis and Optimization," *Proceedings of the IEEE*, vol. 75, no. 12, pp. 1623-1644, Dec. 1987.

[155] Debs, A.S., *Modern Power Systems Control and Operation*, Kluwer Academic Publishers, MA, 1988.

[156] Bergen, A.R., *Power Systems Analysis*, Prentice-Hall, NJ, 1986.

[157] Lawson, C.L. and Hanson, R.J., *Solving Least Squares Problems*, Prentice-Hall, NJ, 1974.

[158] Sreedharan, V.P., "An Algorithm for Non-negative Norm Minimal Solutions," *Numer. Funct. Anal. and Optimiz.*, vol. 9, pp. 193-232, 1987.

[159] Tsirukis, A.G., Reklaitis, G.V., and Tenorio, M.F, "Nonlinear Optimization Using Generalized Hopfield Networkks," *Neural Computation*, vol. 1, no. 4, pp. 511-521, 1989.