1993

Application of artificial neural networks to optimization problems in electrical power operation

Jayant Kumar
Iowa State University

Application of artificial neural networks to optimization problems in electrical power operation

by

Jayant Kumar

A Thesis Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE

Department: Electrical Engineering and Computer Engineering
Major: Electrical Engineering

Iowa State University
Ames, Iowa

1993

This work is dedicated
to the immortal memories of my parents,
to my brother Jeewan Kumar, and to my other family members.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 The Overall Problem

The basis of power system operation and economics in the United States is undergoing major structural changes. A nation's per capita power consumption is, in effect, an index of it's gross national product (GNP). Hence, the economists of the country are examining alternative ways to manage electric power system operation to improve economic efficiency. A major focus of this search is 'competition' as a replacement of 'regulation'. Economists believe that the electric pricing must be governed by free market forces rather by regulated environment. This should reduce customers' expenses and, thus, subsequently boost the nation's economy.

However, if competition replaces regulation as the norm of electric power generation and bulk power supply, a number of changes would be required. The coordination arrangements presently existing among the different players of the electric market would change from the operational, planning and organization standpoints [1]. The structure of the new framework is still being proposed in pieces and is yet not clear.

Competition is being introduced in all aspects of power system operation. Emission regulations are being designed not only to reduce emission, but to achieve the emission reduction with an overall minimum cost. The attempts are being made to decouple the generation and transmission. The idea is to move from central control to decentralized control to promote competition. Need for regulation in generation would no longer be justified. Power transmission is being considered as a transportation business to move the greatest amount of energy at the minimum marginal cost. Short term contracts are being encouraged to reduce the economic constraints imposed by long term contracts. Motivation has been given to provide utility service pricing based on the marginal cost of delivery. In the new paradigm, electricity has been treated as a commodity and, thus, can be bought and sold in the commodities market.

The attempt to deregulate the power industry will significantly affect the way power system operation is optimized today [2]. In the new environment, unit operators will play the spot market and plant owners will play the futures market for expansion capability. Hence, the operators will require more efficient and faster algorithms for real-time implementation. On the other hand, unit owners would need algorithms more suitable for combinatorial optimization. The idea of free market electricity is expected to encourage more market players, increasing the dimensions of power system optimization problems. Classical optimization techniques are inadequate to handle the problems of such high dimension. Considering the complexity of the new framework, one needs to realize the necessity of new tools and techniques to augment the capability of optimization algorithms.

1.2 Scope of This Work

This research has developed a new technique for performing linear programming (LP) and quadratic programming(QP) using an artificial neural network (ANN). A majority of the power system optimization problems can be modeled as LP or QP problems. Thus, the same neural architecture can be applied to model and solve such problems.

ANNs are nonlinear dynamic systems from a system theory point of view. They process feedback in a collective parallel analog mode. Different techniques may be used to solve the neural dynamic system. This research work has developed a novel method of simulation called the clamped state variable (CSV) method. The proposed CSV approach is a very simple approach from problem formulation point of view and it takes less cpu time for the algorithm to converge than the complete circuit simulation. The structure of the system matrix as formulated is sparse for most real-world optimization problems. Thus, sparsity-based algorithms can be applied for matrix manipulations to improve the time complexity as well as the space complexity of the software.

This research has performed an ANN simulation for generic LP and QP problems. In the area of power system optimization, the simulation has been

made using the economic dispatch problem (EDC). EDC is the basic function of power system economics, and it often involves different types of generation-cost-curve modeling. In this work, EDC has been modeled as a quadratic programming problem as well as a separable programming problem with piece-wise-linear cost curves. The algorithm used for separable programming is based on the classical method of merit order loading and linear programming rules.

As mentioned earlier, in the new electric market framework, most of the optimization problems would be difficult to solve because of the curse of dimensionality. Neural networks have highly parallel architecture. Hence, the difficulty of slow computation because of large dimensionality can be reduced to a considerable extent. The neural network that is used for simulating the optimization problems consists of operational amplifiers and, thus, can be easily integrated into a chip form. Hardware implementation could be used to perform fast calculations for real time implementation in the new environment.

1.3 Contents of This Thesis

Chapter 2 of this thesis furnishes a brief glimpse of current issues in power system economics. This chapter summarizes recent developments in regulatory policies in the United States. Principle impacts of the proposed changes on power system optimization problems are also discussed.

Chapter 3 presents an overview of classical optimization theory. The structure and mathematical foundation of linear programming (LP) and nonlinear programming (NLP) are presented. The basic concepts of nonlinear programming are discussed elaborately to provide theoretical basis to the canonical NLP neural network used in this thesis.

Chapter 4 reviews the literature of the neural networks applied to optimization problems. An analysis of the different optimization neural networks is presented from systems theory point of view. The problems associated with the neural network implementation are also highlighted. The

chapter concludes with general remarks and guidelines that might be helpful in selecting a neural network architecture for a given optimization problem.

Chapter 5 provides theoretical development in nonlinear programming ANN design and algorithm design, provided by this research. The solution convergence of network is discussed and, consequently, a modified transfer function is introduced to improve the solution speed. A novel method of simulation called the clamped state variable (CSV) method is introduced for software simulation of the neural network. The method handles linear programming and quadratic programming problems using the same input file format. A hybrid algorithm for linear programming with a piecewise linear objective function is introduced.

Chapter 6 contains the research results. Comparisons between results obtained from neural network simulation and classical optimization methods have been made to check the accuracy of solutions. Chapter 7 gives conclusions and recommendations for future work.

2. CURRENT ISSUES IN POWER SYSTEM OPTIMIZATION

2.1 Background

For nearly a century, the US electric power industry has operated under a regulated environment. The assignment of service areas and electric pricing for a utility company has been governed by regulatory bodies. There was no radical change in regulatory arrangements until late 1960s.

The 1970s was a decade of unprecedented changes for US utility industry. The 1973-74 Arab oil embargo adversely affected fuel prices. The construction cost of new power plants rose dramatically because of a combination of factors, such as high interest rate and increased environmental requirements as directed by Clean Air Act (CAA), 1970. As a consequence, there was an increase in electricity prices, which forced customers to use less electricity. Large industrial customers incurred financial loss because of high production costs. Many industrial customers felt that the option of purchasing competitive generation could be more economical. Many similar policies resulted into uncertain electric demand growth. Such uncertainties caused planning activity problems.

Rising electrical costs and fear of loosing customers became major concerns for the utilities. Power system companies decided to revise their planning process. Issues included a number of topics, such as, whether to allow units to be included in the rate base until they were operational, if at all. Study of health effects from electric and magnetic field was considered to be another concern. On the other hand, regulators tried to analyze how to promote energy conservation while keeping electrical costs to minimum. The distribution of environmental costs among the customers and utilities was an another issue to resolve. In summary, there was a need to perform thorough analysis of the existing coordinating arrangements in the utility industry.

The above problems challenged the regulatory environment of power system companies. The trend towards large, capital-intensive power plants was delayed because of the demand growth uncertainty. Policy-makers

decided to restructure the power industry to keep pace with the economic conditions by promoting competition.

2.2 Developments in the Policy Context

The Public Utility Regulatory Policy Act of 1978 (PURPA) [3] was one of the significant early developments to restructure the electric industry. This act was intended to encourage construction of non utility generators (NUGs) by requiring utilities to purchase power produced by such facilities. The idea was that the power from NUGs must be purchased by the local utility at a price that the utility would have incurred to generate the same power. The limits of such avoided costs had been set administratively by state utility commissions. Some states set prices even higher than actual utility costs in an effort to increase NUGs. Many cogenerations and alternative energy facilities appearing since 1978 were built as a result of the favorable conditions due to PURPA. The act also inspired a growing interest in independently owned, largely non utility power plants (IPPs), which would sell their generation to either utilities or customers. The desire to stimulate competition has lead to the allowance of any qualifying fascilities (QFs) to supply energy.

There has been considerable controversy over the calculation of avoided costs set by state utility commissions. Discussions were held to argue whether avoided costs had been set at levels too high, encouraging too many fascilities at consumers' expense, or too low, discouraging innovative development. In 1988, the Federal Energy Regulatory Committee (FERC) proposed changes to regulations [4] to promote competition in bidding and independent power production. The proposal was also projected as a means of determining avoided costs under state regulatory programs.

FERC's proposal led increased pressure for wheeling and for the emergence of NUGs. This requires redefinition of transmission network access and of future power transmission. As a result, FERC proposed a new framework of power transmission through the 'open transmission access' [5]. In this new paradigm, the utility owns and operates the network as a seperate

transmission company, and, utility provides conditions for pricing its service independent of generation or distribution.

In a parallel development, the 1990 CAA amendments (CAAA) [6,7] were signed into law on November 15, 1990. This act attempts to control overall emissions by costing compliance [8]. The compliance is economically regulated through units of emission allowances (EA). The units of EA are issued by the Environmental Protection Agency (EPA). There is provision for purchase and open-market trading of EAs. This means that utilities can choose to comply by cutting emissions or by buying extra allowances. The each utility may economically decide on which option is more cost effective. Individual utilities could choose their own solution for meeting the CAAA. In addition to the use of EAs, utilities could, for example, switch fuels or install scrubbers. In 1993, the Chicago Board of Trade started trading the Emission allowances.

To facilitate the growth of free market electricity, US Senate passed a comprehensive National Energy Policy Act (NEPA) [9] in 1992. The act defines exempt wholesale generators (EWGs) as any company owning or operating all or part of an eligible facility and selling electricity at wholesale cost. FERC has been given discretion to decide whether an EWG should be exempt from the Public Utility Holding Company ACT (PUHCA). Utilities are permitted to purchase from an affiliated EWG under the jurisdiction of a state commission. FERC may issue a transmission order if such an order meets certain requirements and would be in the public interest. A utility has 60 days to respond to a transmission request before an applicant can file for a wheeling order with FERC.

While defining the ground rules for bid evaluations, policy makers felt that the regulatory bodies must participate in the bid evaluation process. A need for a central coordinating body, who could make unbiased decisions for economic operation while ensuring the bulk system's reliability and security, was identified. In July, 1993, FERC issued a notice of proposed regulation (NOPR) [10] encouraging the development of regional transmission groups (RTG). The commission expects RTGs to be a means to enable a free market for electric power to operate in a more competitive and efficient way. The commission believes that RTGs can provide a means of coordinating regional

planning of the transmission system while assuring that system capabilities are always adequate to meet system demands.

2.3 Impacts on Optimization

The new approach to restructure the utility industry has direct impacts on the power system optimization problems. Many of the power transactions would take place in the spot market. Transaction evaluation and selection in the spot market price environment requires faster and efficient algorithms. Operating uncertainties will increase as generation and load control disassociate from the centralized network utility. This will require more robust statistical optimization techniques. The approach in this new environment is termed integrated least cost resource planning. Such an approach requires efficient tools based on combinatorial optimization.

The idea of free market electricity will encourage more market players, adding new dimensions to power system optimization. Classical optimization techniques are inadequate to handle the problems of such high dimension.

3. OVERVIEW OF CLASSICAL OPTIMIZATION THEORY

3.1 Introduction

This chapter is a summary of the theory for constrained optimization problems. Constrained optimization problems can be broadly classified into two classes - linear programming and non-linear programming. The chapter begins with a general discussion about linear programming. Next, linear programming with piece-wise linear objective function is discussed. The theoretical foundation of non-linear programming is explained in detail as it plays an important role in neural network formulation to be discussed in Chapter 5. Emphasis has been given on convex nonlinear programming. Quadratic programming is discussed as a special class of non-linear programming. The chapter concludes by summarizing a list of classical optimization algorithms used for the above mentioned problems.

3.2 Linear Programming (LP)

Within the area of optimization, the most widely known and implemented technique for modeling and solution is, by far, the methodology denoted as linear programming. A linear program is a mathematical program in which the objective function is linear in the unknowns and the constraints consist of linear equalities and linear inequalities. A linear program can always be transformed in the following form.

Minimize

$$\phi(x) = A^T x$$

Subject to constraints :

$$f(x) = Bx - e = 0$$

and,

$$x_i \geq 0, \qquad \forall\ i = 1,2,\ldots,n \tag{3.1}$$

Here, A is an n-dimensional column vector representing the cost per unit element of column vector x. B is an m*n matrix, and e is an m-dimensional column vector. This type of problem formulation with linear costs and linear constraints occurs frequently in resource allocation and transportation problems where some minimum cost is desired while meeting certain constraints. In most cases, the initial problem formulation will include some inequality constraints of the form

$$f(x) = Bx - e \geq or \leq 0 \tag{3.2}$$

To deal with this in classical optimization method, dummy variables known as *surplus* and *slack variables* are introduced for '≥' constraints and '≤' constraints respectively. This modification makes the form of equation (3.2) to be consistent with the form of equation (3.1). Two of the most common inequality constraints for LP problems require elements of x to remain below some maximum value

$$x_i \leq u_i, \qquad ¥\ i = 1,2,...,k \tag{3.3}$$

or above some minimum value

$$x_i \geq l_i, \qquad ¥\ i = 1,2,...,k \tag{3.4}$$

A variable without upper and lower bound is known as a *free variable.* Several methods are presently available to implement free variables in linear programs, most of which require additional variables and, hence, additional memory and computation time.

Any vector x_f that satisfies all the constraints imposed by the problem formulation including the upper and lower bounds and non-negativity constraints is known as a *feasible solution.* A vector x_n that fails to satisfy any constraint is called an *infeasible solution.* The region formed in the n-

dimensional hyperplane by the set of all feasible solutions is known as the *feasible region*. A feasible solution x^* that minimizes the objective function is called the *optimal solution*.

The simplex method [11] is the most commonly used technique to solve LP problem. The simplex method has been applied to linear programming in single as well as multiple objective [12] optimization problems. The method constructs a set of basic and nonbasic variables by choosing the linearly independent columns of sensitivity matrix D. A *basic feasible solution* is achieved by letting the nonbasic variables equal to zero. The *basic feasible solution* is an *optimal solution*, if the gradient of objective function with respect to each nonbasic variable is positive. The algorithm proceeds by exchanging the nonbasic variable having most negative gradient vector with the *most restricting basic variable* until the *optimal solution* is achieved.

A newer algorithm for LP travels through the interior of the feasible region to find the solution. The interior point algorithm was introduced by Narendra Karmakar [13] at Bell labs in 1984. Karmarkar's algorithm differs from the Simplex method in that it starts with an interior feasible solution rather than a basic feasible solution. It is called interior point algorithm because it does not operate on the boundary of feasible region as in Simplex method. Karmarkar's algorithm rescales the variables to place the solution in the center of a scaled feasible solution, allowing it to take a large step toward the optimal solution. An affine scaling algorithm is a variant of Karmarkar's algorithm with much simpler centering scheme and a gradient technique to decide the direction to move the solution.

3.3 LP with Piecewise Linear (PWL) Cost Functions

Many of the real-world problems are governed by nonlinear cost functions. It is possible to include those functions in the linear program by using piecewise-linear (PWL) approximations. The approximation is developed by choosing the intervals of linearity on the nonlinear curve. Extreme points of the interval of linearity are termed *break points*. A piecewise-linear approximation is shown in Figure 3.1.

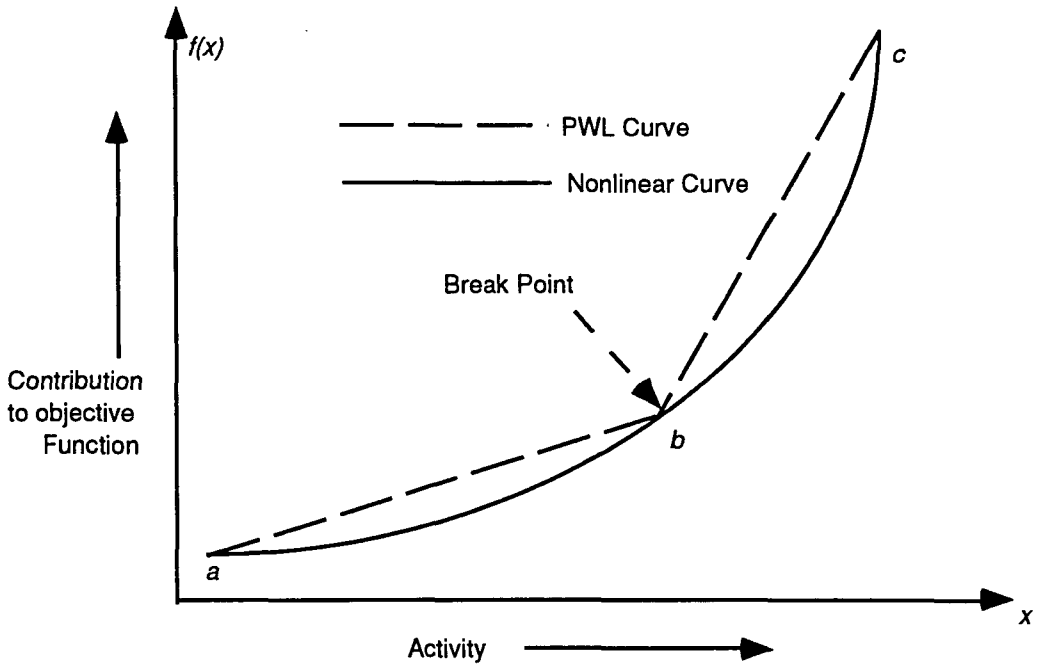

Figure 3.1 Piecewise linear approximation of cost curve

This PWL approximation contains two line segments, the first of which approximates the curve over the range [a,b] and the second of which is valid over [b,c]. Point b is called break point. Different linear programs use the PWL cost function in variety of ways.

The simplex algorithm would represent PWL function, shown in Figure 3.1 by two variables, x_1 and x_2. The variables that represent PWL segments are known as segment variables. Then, we have the constraint that

$$x = x_1 + x_2 \tag{3.5}$$

While the function value is determined by

$$f(x) = S_1x_1 + S_2x_2 \tag{3.6}$$

where the S_i is the slope of segment i. Also, note that x_1 and x_2 can not be in the basis at the same time, otherwise the function defined above would be

meaningless. The simplex algorithm incorporates extra logic into the routine to implement this *restricted basis entry* constraint. The technique described for transforming a nonlinear function into a PWL approximation is taken from Reference [11]. A more detailed analysis and description of the method is available in Reference [14].

Another method of solving LP with a PWL objective function includes guessing the active segment and performing the linear programming for the selected segment successively until all the constraints are satisfied. The identification of the active segment employs heuristics specific to the problem under consideration. This method allows the application of apriori knowledge to a deterministic algorithm, which is expected to be more efficient.

3.4 Nonlinear Programming (NLP)

The NLP problem arises in a myriad of forms in engineering economics. As the name suggests, NLP problem consists of optimizing a nonlinear cost function subject to a set of nonlinear constraints. The theory of nonlinear programming is based on the advancements in the field of calculus, linear algebra, and convex analysis. This section states necessary notations and definitions needed to analyze a NLP problem followed by discussion of some important developments.

3.4.1 Notations and definitions [15]

$x \subset R^n$ is said to be convex iff for any $a,b \in x$ implies $[a,b] \subset x$, where $[a,b]$ is defined as follows:

$$[a,b] = \{ x \in R^n \mid x = \lambda a + (1-\lambda)b , \qquad 0 \le \lambda \le 1 \}$$

Let $x \subset R^n$ be a non empty convex set, then the function $f : x \to R$ is said to be convex iff

$$f(\lambda x + (1-\lambda) y) \le \lambda f(x) + (1-\lambda) f(y) \text{ for any } x,y \in X \text{ and } 0 \le \lambda \le 1$$

The function $f : x \rightarrow R$ is said to be concave if -f is convex. An affine function $f : x \rightarrow R$ is a function that is convex and concave. H is a *hyperplane* in R^n, if there exists

$$a \in R^n, \quad a \neq 0, \quad \alpha \in R$$

such that

$$H = \{ x \in R^d \mid \langle a, x \rangle = \alpha \}$$

Where, $< , >$ is the *euclidean inner product* on R^n and a is a *normal vector* of the hyperplane H.

A set of n-dimensional inequality $g_j(x) \leq 0$ is said to be binding for a point x* if

$$g_j(x^*) = 0 \qquad for\ all\ j$$

A point x* satisfying a set of hyperplane $h_i(x) = 0$ and $g_j(x) = 0$ is said to be a regular point of the set if the gradient

$$\nabla h_i(x^*) = 0 \qquad \nabla g_j(x^*) = 0 \qquad for\ all\ i,j \in hyperplanes.$$

satisfy linear independence in their corresponding vector space.

3.4.2 Lagrangian method for NLP and optimality conditions

A nonlinear program P is of the form given below:

Minimize

$$f(x)$$

Subject to:

$$g_j(x^*) \leq 0 \qquad for\ all\ j \in \{1,2, \ldots, r\}$$
$$h_i(x^*) = 0 \qquad for\ all\ i \in \{1,2, \ldots, m\} \tag{3.7}$$

A vector x is called a *feasible solution* of NLP iff x satisfies the r+m constraints of NLP. The condition of optimality for a nonlinear program [16] is given by following conditions.

3.4.2.1 First order necessary condition (FONC) This condition is also known as Kuhn-Tucker's (KKT) conditions. Let $\bar{x}$ be a feasible solution to (3.7). Suppose, each g_j is differentiable at $\bar{x}$. Further, assume that $\bar{x}$ is a regular point. Then, $\bar{x}$ is a relative minimum point in the solution space of (3.7) if and only if there exists

$$\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_r]^T \qquad \lambda = [\mu_1, \mu_2, \ldots, \mu_m]^T$$

such that

$$\begin{aligned}
&(i)\ \lambda_i \geq 0 && g_i(\bar{x}) \leq 0 \\
&(ii)\ \lambda_i\, g_i(\bar{x}) = 0 && i = 1, \ldots, r \\
&(iii)\ \mu_j\, h_h(\bar{x}) = 0 && j = 1, \ldots, m \\
&(iv)\ \nabla f(\bar{x}) + \sum_{i=1}^{r} \lambda_i\, g_i(\bar{x}) + \sum_{i=1}^{r} \mu_j\, h_j(\bar{x}) = 0
\end{aligned} \tag{3.8}$$

The variables λ_i and μ_j are known as Lagrange multipliers.

3.4.2.2 Second order necessary condition (SONC) The vector $\bar{x}$ is a relative minimum point in the solution space of (3.7), if and only if Kuhn Tucker's conditions are satisfied such that

$$L(\bar{x}) = F(\bar{x}) + \lambda^T H(\bar{x}) + \mu^T G(\bar{x}) \tag{3.9}$$

is a positive semidefinite matrix on the tangent subspace of the active constraint at $\bar{x}$. The matrices F, G, and H are the corresponding Hessian matrices of the functions f, g, and h respectively. Matrix L is called the *Hessian of Lagrangian.*

3.4.2.3 Second order sufficient condition (SOSC) The vector $\bar{x}$ is a *strict minimum point* , i.e., the optimum point in the solution space of (3.7), if and only if FONC and SONC are satisfied and the Hessian of the Lagrangian is positive definite on the subspace

$$M' = \{ y : \nabla h(\bar{x})y = 0, \quad \nabla g_j(\bar{x})y = 0 \} \qquad for\ all\ j \in J$$

where,

$$J' = \{ j : g_j(\bar{x})y = 0, \quad \lambda_j > 0 \} \tag{3.10}$$

The above equation $J = 0$ together with $h(x^*) = 0$ comprises the set known as binding set. The set of remaining constraints is termed as non-binding constraints. It is the binding set of constraints that govern the optimum solution for a given problem.

A generalized NLP is convex program if f and g_j are finite convex functions on R^n and h_j are affine functions. In case of convex program, the Hessian matrices are always positive semidefinite. Thus, the requirements of second order conditions are always satisfied. Thus, the optimality condition of convex program is reduced to satisfying Kuhn-Tucker's conditions only.

3.4.3. Reduced NLP and optimality conditions

Any nonlinear program can be reduced to minimizing a nonlinear objective function subject to a set of inequality constraints only. This is possible, because an equality, in mathematical sense, is a coupled set of inequalities. The reduced representation can be achieved by defining some new Lagrange multiplier as follows:

Define

$$\mu_{j+} = max\,[\mu_j, 0] \qquad \mu_{j-} = min\,[\mu_j, 0]$$

Then

$$\mu_j \equiv \mu_{j+} + \mu_{j-} \qquad for \quad 1 \le j \le m$$

The corresponding term in the condition (iv) of Kuhn-Tucker's condition can be written as :

$$\mu_j \nabla h_j = (\mu_{j+}+\mu_{j-})\nabla h_j = \mu_{j+}\nabla h_j + (-\mu_{j-})(-\nabla h_j)$$

Since only one of μ_{j+} or μ_{j-} is non zero, and $h_j = 0$, condition (iv) can be extended by using conditions stipulating mutually exclusive terms $h_j \le 0$ and $h_j \ge 0$ as follows:

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

The Lagrange multipliers are forced to be all non-negative. Let us define the extended Lagrange multipliers as :

$$g_{r+2j-1} = h_j, \quad g_{r+2j-1} = - h_j, \quad \lambda_{r+2j-1} = \mu_{j+}, \quad \lambda_{r+2j} = - \mu_{j-}$$

Then, the reduced NLP can be expressed as follows:

Minimize

$$f(x)$$

Subject to:

$$g_i(x^*) \le 0 \qquad 1 \le i \le r+2m \tag{3.11}$$

Thus, the optimality conditions for the above reduced representation is given as follows:

(*i*) $\lambda_i \ge 0$ $\qquad g_i(\overline{x}) \le 0$

(*ii*) $\lambda_i g_i(\overline{x}) = 0$ $\qquad i = 1, ..., r$

$$(iii)\ g_i\ (\bar{x}) = 0 \qquad i = r+1, \ldots, r+2m$$
$$(iv)\ \nabla f(\bar{x}) + \sum_{i \in I(\bar{x})} \lambda_i \nabla g_i\ (\bar{x}) + \sum_{i \in I'(\bar{x})} \lambda_i\ \nabla\ g_i\ (\bar{x}) = 0 \tag{3.12}$$

Where, I denotes the binding subset of inequality constraints. Many algorithms in classical optimization work on the principles of the reduced NLP. The choice of binding set of inequalities is usually made heuristically and updated as the solution progresses.

3.4.4 Penalty function method for NLP

The penalty function method approximates the constrained optimization problem into an unconstrained optimization problem by adding a penalty term to objective function, which assigns a high cost to the constraint violation. The assignment of cost is made using a suitable value for the penalty parameter. Thus, the objective function of the reduced NLP can be written as

$$L(s_k, x) = f(x) + \frac{s_k}{2} \sum_{i=1}^{r+2m} (\ g_i^+\ (x)\ )^2 \tag{3.13}$$

Where, $g_i^+\ (x)$ is the constraint violation and $s_k\ \}_1^\infty$ is a nonnegative, strictly increasing sequence tending to infinity. The parameter s is known as the penalty factor. The optimal condition for the augmented objective function is given as:

Let the minimizer of $L\ (s_k, x)$ *be* x_k. *Then, any limit point in the sequence* x_k *is an optimal solution to equation (3.13).*

Furthermore, if $x_k \rightarrow \bar{x}$, and $\bar{x}$ is a regular point, then $s_k\ g_i^+(x_k) \rightarrow \lambda_i$, which is the corresponding Lagrange multiplier. Thus, the penalty function

approach and Lagrangian method have strong correspondence. A detailed analysis of penalty approach was first introduced by Zanghwil [17].

3.4.5 Quadratic programming

Quadratic programming arises in many applications and it forms a basis for general nonlinear programming [18]. The general form of quadratic programming is given as:

Minimize

$$\phi(x) = A^T x + 1/2\, x^T G x$$

Subject to

$$f(x) = Bx - e \geq 0 \qquad (3.14)$$

Matrix G is the quadratic cost coefficient matrix. Different methods have been applied to solve quadratic programming. Quadratic Programming becomes a convex programming, if matrix G is positive semidefinite.

4. REVIEW OF LITERATURE

4.1 Artificial Neural Network

4.1.1 Introduction

Artificial neural networks take their name from the networks of nerve cells in the human brain. The human brain contains about a hundred billion nerve cells called neurons, each with as many as 10,000 interconnections with other neurons. The interconnections comprise of chemical-filled gaps known as synapses. The input area of the neurons is called dendrites and the output area of neurons is known as axons. An impulse can be triggered by the cell, and sent along the axons to dendrites of the other neurons through synapses. When a series of impulses is received at the dendritic areas of a neuron, the result is usually an increased probability that the target neuron will fire an impulse down its axons. On the way from one neuron to other neuron, pulses are modulated by the synaptic strength of the interconnection to add local information. Thus, the human brain is a parallel, distributed information processing structure consisting of processing elements, which can carry out localized information processing operations. The sheer number of neurons and the high density of interconnections account for a substantial part of the brain's computation power.

Artificial neural networks are motivated by models of human brains and nerve cells [19]. They are characterized by parallel architecture comprising of neurons with high degree of interconnection and large degree of feedback. An artificial neuron is designed to mimic the characteristic of the biological neuron. The input data, which may come from the outside world (neural network) or from the other neurons in the same network, get multiplied by a corresponding weight (w1, w2, ..., wn) analogous to the synaptic strength in the biological neural structure. The weighted inputs are summed up to produce the quantity called *net*. The neuron acts upon this *net* and uses a nonlinear transfer function *f* to compute its output. Sometimes a bias term is added to the sum before it is passed through the transfer function

of the neuron to yield the output. The calculated result is sent along the output connections to the target cells down the line. The same output value is sent along all the output connections. Figure 4.1 shows a detailed model for an artificial neuron used in an artificial neural network.

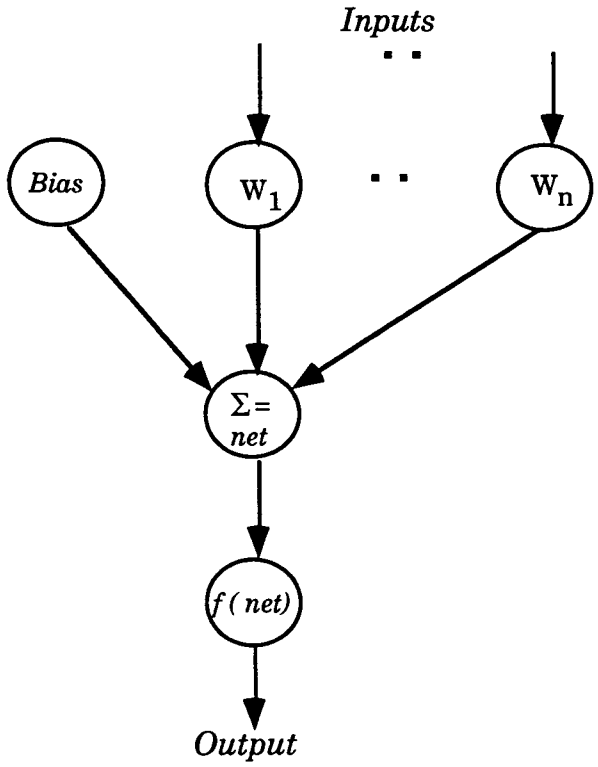

Figure 4.1 Model of an artificial neuron

Computation with an artificial neural network is usually referred to as neurocomputing [20]. Neurocomputing is considered to be the first alternative to programmed computing. Solving a problem using programmed computing involves devising an algorithm and a set of rules and then correctly coding these in software. The exhaustive design, testing and iterative improvement that software development demands make it a lengthy and expensive process. Furthermore, if the required algorithmic procedure and set of rules are not exactly known, the development process may require some additional insight.

In real life, such situations are not uncommon. To deal with these situations, neurocomputing employs neural network models to explore many competing hypotheses simultaneously. The hypotheses are based on complex interactions of several variables, where the exact functional relationship between the variables is not known.

One needs to design a neural network architecture for a desired computation. Neural network architecture is different from those of digital computers in some aspects. State-of-the-art parallel processing architecture of digital computer typically has a smaller ratio of interconnection to processing units. The processors (neurons) in a neural network are massively interconnected. The neural networks contain numerous, but simpler processing units. Parallel analog computations in a network of neurons provide high computational power and speed.

4.1.2 Neural network training

The simulation of a neural network is termed the execution phase for a given problem simulation. The execution phase of the neural network is preceded by a training phase, which involves the determination of the network parameters such as the weights of the interconnections and the thresholds for corresponding neurons. The goal of the training is to assign values to weights and thresholds by minimizing some kind of objective function. Examples of algorithms of some important neural networks are given in Reference [21].

There are different ways to determine the weights and thresholds. One way is to generate the training data, which may consist of a set of input vectors and output vectors, where each input pattern is associated with a single output pattern. Then, the error vector is minimized such that the actual output of the network is close in some sense to the associated desired output when a given input is applied. Thus, the weights and thresholds are the optimal solutions of the error minimizing problem for a given data set. Such a trained network is called a *supervisory* neural network. Single layer and multi-layer perceptron models [19] are the examples of the supervisory type. A good

summary of the recent theoretical result concerning the capabilities and limitations of the supervisory models is given in Reference [22].

In *self-organizing* neural networks, weights are assigned to satisfy some kind of mathematical criteria. For example, the Kohonen net learns a continuous topological mapping [19]. The range of the mapping is a rectangular subset of n-dimensional Euclidean space. The domain of the mapping is a bounded subset of m-dimensional Euclidean space. The mapping is governed by 'y' examples defining probability density function $\rho(y)$. The determination of the weights and thresholds is made by self-organization on 'y' examples with respect to the probability density function.

4.1.3 Neural network applications [19]

Artificial neural networks have a wide range of applications. The first highly developed application was the handwritten character identification. Other impressive applications study involved NETtalk, a neural network that learns to produce phonetic strings, which specify pronunciation for written text. Signal analysis has also been attempted with the neural network. Numerous applications have been examined in the field of pattern recognition and signal processing.

Neural networks were also configured to implement associative memory systems. They were applied to a variety of financial analysis problems, such as credit assessment. One of the common applications for an ANN is the forecasting future trends. Neural networks are presently being used to forecast stock market prices and the risk associated with different loan applications. In the area of power system operation, the neural networks have been applied to the short-term load forecasting problem. Neural network studies have also been made for adaptive control applications, such as the broom-balancing experiment.

Neural networks and optimization theory have been used to supplement each other's needs. Optimization theory has been applied for training the neural nets [23]. Various techniques of optimizing error functions to train the neural net classifiers have been investigated. On the other hand, the results

demonstrate the possibility of applying neural networks to solve optimization problems [24.25]. Large-scale neural networks are capable of yielding high quality solutions to complex combinatorial problems [26].

Neural networks are expected to complement rather than replace other technologies. For example, expert systems and rule-based knowledge-processing techniques are adequate for some applications, although neural networks have ability to learn rules more flexibly. In some cases, more sophisticated systems may be built from a combination of fuzzy logic and neural networks [27].

4.2 Optimization Neural Network

4.2.1 Introduction

The application of artificial neural networks to optimization problems has been an active area of research since the early eighties. Research work has shown that artificial neural networks are nonlinear dynamic systems from system theory point of view. A neural network with the following properties in the state space of interest can perform the task of system optimization:

- Every network trajectory always converges to a stable equilibrium point.

- Every state equilibrium point corresponds to an optimal solution of the problem

The first property guarantees that given any initial point to the network, the ensuing network trajectory leads to a stable steady state. The second property ensures that every steady state of the network is a solution of the underlying optimization problem. A sufficient condition for a neural network to possess the first property is the existence of the energy function associated with the network. The second property can be relaxed such that the state of

every stable equilibrium point is close to an optimal solution point of the problem. Parallel analog computations in a network of parallel interconnected neurons provide high computational power and speed. The ability of processing feedback in a collective parallel analog mode enables a neural network to simulate the dynamics that represent the optimization of an objective function subjected to its constraints for a given optimization model.

Articles have been published with the development of neural networks applicable to general nonlinear, constrained optimization problems including linear programming and nonlinear programming, and combinatorial optimization problems. Different neural networks have been analyzed from both the view point of optimization theory and dynamical system theory. Most of the optimization neural networks are based on the principle of minimization of energy function associated with the circuit. Mathematical analysis of some networks verify that the equilibrium point of the network correspond to Kuhn-Tucker condition of optimality for nonlinear optimization as discussed in the Section 3.4.2. This section presents a brief history of optimization neural networks followed by the analysis of several neural networks currently available in the literature, which are useful for system optimization. The section concludes with the guidelines and general remarks that might be helpful in selecting a neural network architecture for a given optimization problem.

4.2.2 A brief history of the optimization ANNs

4.2.2.1 Unconstrained optimization networks Recent interest in using an ANN as an alternative for solving optimization problem was intrigued, among others, by the works of Hopfield [28,29]. In 1985, Hopfield and Tank introduced a continuous model for unconstrained optimization problems based on the energy function approach [30]. The behavior of a neuron in the model was characterized by its activation level u_i, which was governed by the differential equation

$$\frac{du_i}{dt} = \sum_j T_{ij} g_j(u_j) - \frac{1}{R_i} u_i + I_i \tag{4.1}$$

$$\frac{1}{R_i} = \frac{1}{\rho_i} + \sum_j \frac{1}{R_{ij}} \tag{4.2}$$

Where, u = Internal state of neuron
T = Weight of synapse strength of coupling between neurons
$g_i(u_j)$ = Activation function for neuron j.
I= Threshold inherent to neuron

For a circuit of n neuron amplifiers, a state space model could be written by selecting the output of amplifiers as the state variables as follows:

$$\dot{x_i} = \frac{dg_i}{du_i}(u_i) * \frac{1}{C_i} * (\sum_j T_{ij} x_j - \frac{1}{R_i} u_i + uI_i) \tag{4.3}$$

When $T_{ij} = T_{ji}$, Hopfield has indicated that the following energy function constitutes Liapunov's function in dynamics or that of the following function reduces spontaneously:

$$E = -\frac{1}{2} \sum_i \sum_j T_{ij} g_i(u_i) g_j(u_j) - \sum_i I_i g_i(u_i) \tag{4.4}$$

$$\Delta E = [\sum_j T_{ij} g_j(u_j) + I_i \ ] \Delta g_i(u_i) \tag{4.5}$$

Because a network of neurons will seek to minimize its energy function, one may design a neural network for function minimization by associating variables in the energy function.

The Hopfield neural network has been applied to find solutions to difficult optimization problems. Hopfield and Tank illustrated the use of the energy function approach to configure the continuous model to solve a number of unconstrained optimization problems [30,31] including the TSP (traveling salesman problem), typical problems of NP-complete class, A/D conversion, and the decomposition of the given signal in the given set of basis signals. Inspired by their results, a number of researchers have applied feedback networks to such diverse problems as object recognition, graph recognition, graph coloring, non threatening queen placement, detecting graph isomorphism and concentrator assignment. The basic problem in using the Hopfield network arises because of following reasons:

- The network solution is always a local minima depending on the initial conditions, which are affected by the individual choice and the orientation of random noise introduced while starting the simulation.

- The rate of convergence for the solution is usually very low. Good convergence needs a thorough analysis of the data-set.

- The robustness of the algorithm is usually not guaranteed, which results in a bad solution quality.

- Simulation parameters are chosen in adhoc manner.

4.2.2.2 Constrained optimization networks In 1984, Chua and Lin developed a canonical nonlinear programming circuit [32]. However, the circuit description was not based on the neural architecture. After two years (1986), Tank and Hopfield presented an ANN-based linear programming circuit [31], which was a modified version of the analog model discussed in the previous section. The objective function and the individual constraints were expressed in the form of an energy function. The combined energy function

corresponded to the equivalent unconstrained problem. Later, Kennedy and Chua [33] claimed that the Tank and Hopfield linear programming ANN circuit obeyed the same unifying stationary co-content theorem as the canonical nonlinear programming circuit described in Reference [32]. It was established that the Tank and Hopfield linear programming ANN circuit was a special case of the Chua and Lin nonlinear programming circuit. In 1988, Kennedy and Chua [34] made further modifications to the canonical circuit that was suitable for neural network implementation. The function of the developed network was based on penalty function approach in the classical optimization theory. In 1992, Zhang and Constatinidies proposed a methodology [35], which was based on the Lagrange multiplier theory. However, the penalty factor and the Lagrange multiplier do have strong correspondence as discussed in section 3.4.4. Hence, the structure of network model proposed by Zhang and Constatinidies turned out to be quite similar to other constrained optimization networks.

The basic idea behind this approach is to embed the optimization model into the ANN by designing the energy function of the network to correspond with the objective function of the optimization problem. The constraints are mapped into the network by using feedback from the variable neurons (computing the optimization variables) to the constraint neurons (computing the constraint functions). The constraint violations are looped back to adjust the state of the variable neurons. The dynamics of such a designed network satisfy the Kuhn-Tucker's conditions for optimality given by Equation (3.8). Eventually, the network dynamics stabilize at the state that represents the optimum point in the solution space of the optimization problem.

From implementation point of view, the different nonlinear programming neural networks have unique drawbacks:

- Kennedy, Chua, and Lin ANN
    - The choice of penalty function, which depends on the knowledge of solution surface is often difficult to make.

- The Kennedy, Chua and Lin ANN is primarily designed for inequality constraints. Optimization with equality constraints requires selecting and tuning of penalty function.

- Lagrange Programming ANN

  - In this model, the number of states in the network is increased. This leads to a problem of large dimensions. This is more true in case of problems with inequality constraints that are quite frequently encountered in real world.

On the other hand, the following features make nonlinear programming neural networks fairly easy to use:

- Kennedy, Chua, and Lin ANN

  - The networks' structural parameters are in correspondence with the coefficient vectors and the coefficient matrix of the optimization problem.
  - The network can be modeled as a nonlinear control system, which can be analyzed by a number of the theoretical tools already developed.

- Lagrange Programming ANN

  - The optimization problems can be convexified by the augmented Lagrangian function and the problems can be solved by successive approximation.

  - The approach can handle problems with equality constraints quite efficiently.

4.2.3 Optimization ANN models

4.2.3.1 The Hopfield LP neural network As discussed in an earlier chapter, a LP problem can be considered as minimizing the objective function

$$\phi(x) = A^T x$$

Subject to constraints:

$$f(x) = Bx - e \geq 0 \tag{4.6}$$

where A and x are q-vectors, and f and e are p vectors, and B is a (p*q) matrix. Hopfield and Tank proposed an architecture of their continuous model, specially designed for linear programming problems, as shown in Figure 4.2. The state equation of the LP circuit and the associated energy function are given as follows:

$$C_i \frac{du_i}{dt} = \sum_j B_{ij} g_j(B_j^T x - e_j) - \frac{1}{R_i} u_i - A_i \tag{4.7}$$

and,

$$E = A^T x + \sum_j F(B_j^T x - e_j) + \frac{1}{R_i} \int_0^x g_i^{-1}(x) dx \tag{4.8}$$

in which

$$F(z) = \int_0^z f(t) dt \tag{4.9}$$

where, f and g are the nonlinearity of the sets of constraint neurons and variable neurons respectively. The energy function consists of three terms: (1) the objective function ϕ, (2) a penalty term F that penalizes solutions that violate the constraints, and (3) a conductance term for the model to be implementable by an analog circuit. Thus, the minimization of energy function corresponds to the simulation of the optimization problem.

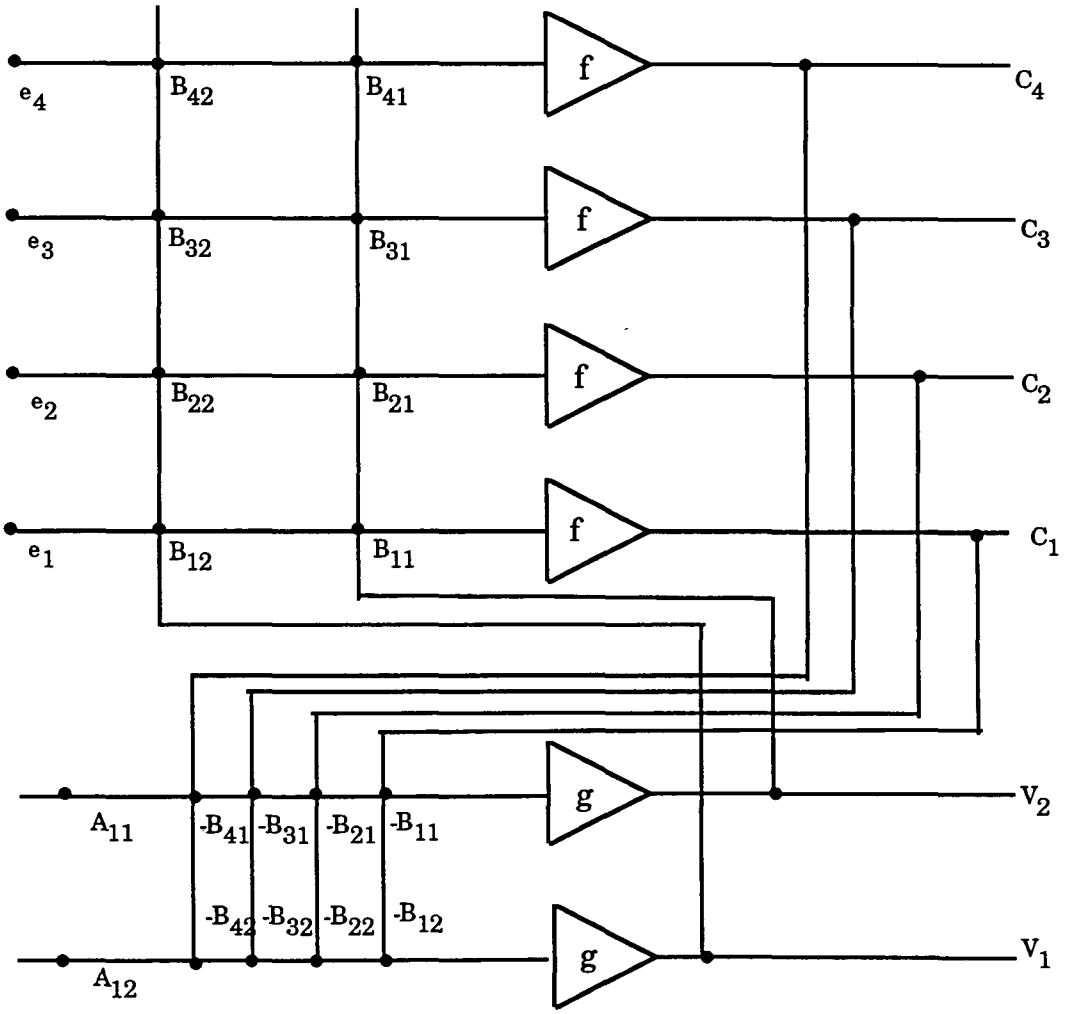

Figure 4.2 The Hopfield neural network

The time evolution of the energy function always decreases and, hence, eventually reaches the stable equilibrium point as shown below:

$$\frac{dE}{dt} = \frac{dx_i}{dt}[A_i + \sum_j (B_j{}^T x - e_j) + \frac{u_i}{R}$$

$$= \sum_i C_i \frac{dx_i}{dt}\frac{du_i}{dt} = -\sum_i C_i (g_i^{-1}(x_i))^t (\frac{dx_i}{dt})^2$$

(4.10)

4.2.3.2 The Kennedy, Chua and Lin canonical NLP circuit A nonlinear programming may be considered as a quadratic programming problem if the objective function is a quadratic polynomial and the constraints are nonlinear. A quadratic programming problem is represented by the following form:

Minimize

$$\phi(x) = A^T x + 1/2\, x^T G x$$

Subject to

$$f(x) = Bx - e \geq 0$$

(4.11)

where A and x are q-vectors, and f and e are p vectors, and B is a (p*q) symmetric positive definite matrix. Linear programming (LP) is a special case of quadratic programming (QP), where the matrix G reduces to zero. The Kennedy, Chua and Lin ANN circuit is shown in the Figure 4.3.

The lower half of the circuit consists of variable neurons that are integrator cells. The upper half of the circuit consists of constraint neurons. The transfer function of the constraint neurons is the derivative of the penalty function. The rectangular boxes shown in the circuit represent the resistance obtained from the state space model of the optimization problem. Vectors A and e are the linear cost coefficients and the resource capacity respectively,

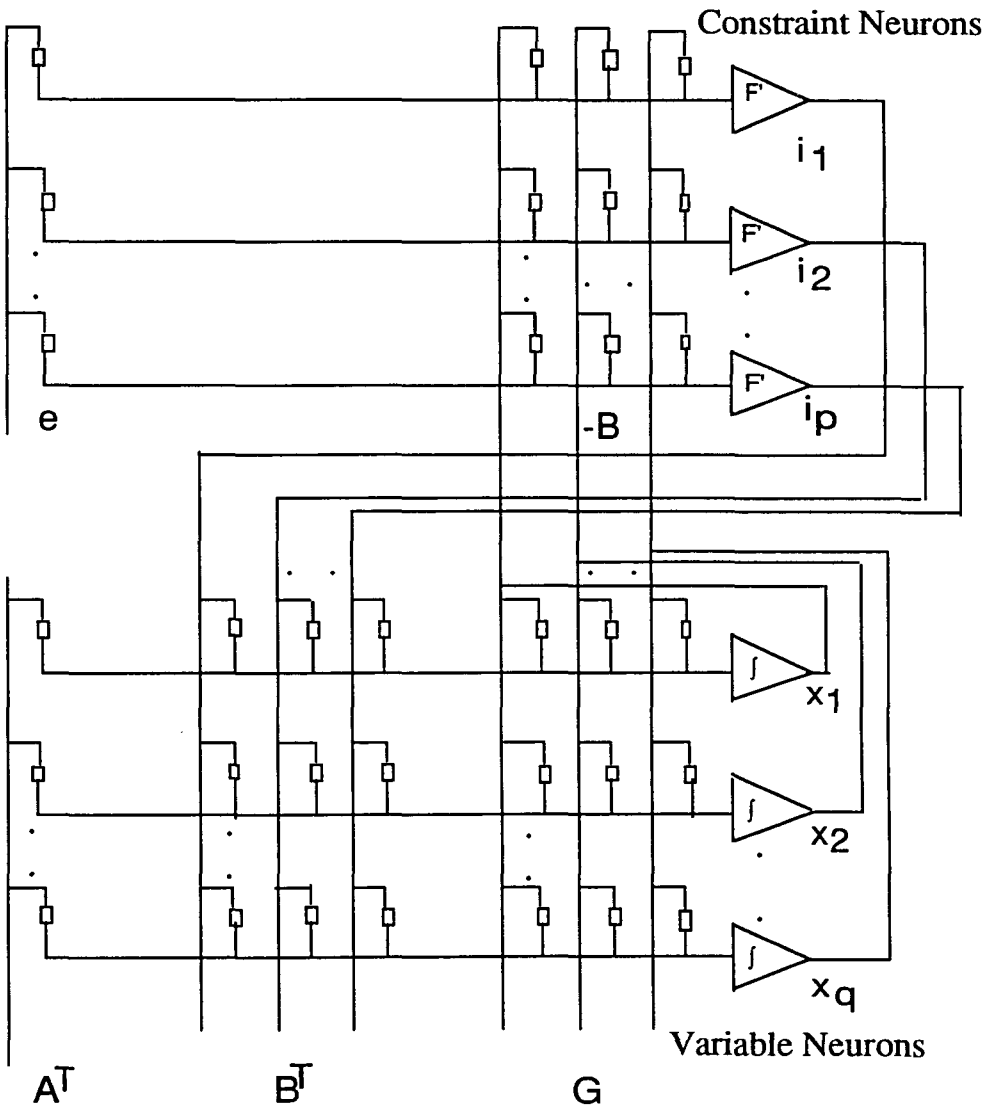

Figure 4.3 The Kennedy, Chua, and Lin neural network

and the matrices B and G are the sensitivity matrix and the quadratic cost coefficient matrix respectively for quadratic programming.

The circuit description of the ANN circuit is given by following state-space equation:

$$C_i \frac{dx_i}{dt} = -\frac{\partial \phi}{\partial x_i} - i_j \frac{\partial f_i}{\partial x_i} \tag{4.12}$$

with

$$i_j = g_j(f_j(x)) \tag{4.13}$$

where g is the nonlinearity of the constraint neurons. The energy function of the circuit and its time evolution are given as follows:

$$E = \phi(x) + \sum_j \int_0^{f_j(x)} g_j(t)\, dt \tag{4.14}$$

$$\frac{dE}{dt} = \sum_i \frac{\partial \phi_i}{\partial x_i}\frac{dx_i}{dt} + \sum_i \sum_j (g_i(f_j(x_i))\frac{\partial f_i}{\partial x_i}(\frac{dx_i}{dt})$$

$$= -\sum_i (C_i \frac{dx_i}{dt})\frac{dx_i}{dt} = -\sum_i C_i (\frac{dx_i}{dt})^2 \tag{4.15}$$

Since C_i and the square terms are always positive, the energy function of the circuit always decrease in value.

4.2.3.3 The Lagrange programming neural network (LPNN) The Lagrange programming neural network differs from Kennedy, Chua, and Lin ANN from an operational point of view. Instead of following a direct descent approach of the penalty function, the network looks for a point satisfying the

first order necessary condition (FONC). There are two classes of neurons in the network, variable neurons and Lagrangian neurons. Variable neurons seek a minimum point of the cost function and provide the solution at the equilibrium point, while the Lagrangian neurons lead the dynamic trajectory into the feasible region. Thus, the two sets of neurons are quite similar to those in the Kennedy, Chua and Lin ANN.

However, the main difference in the functioning of the Lagrangian neurons in Lagrange programming ANN and the constraint neurons in the Kennedy, Chua and Lin ANN is that the Lagrangian neurons are treated on equal basis with the variable neurons. The dynamic process is carried out simultaneously on both. Thus, the constraints can be violated during the dynamic process, but they are finally satisfied at the stable equilibrium point. The approach is especially useful in dealing with problems with equality constraints.

A nonlinear programming problem with an equality constraint can be expressed in the following form:

Minimize

$$f(x)$$

Subject to

$$h(x) = 0 \tag{4.16}$$

As discussed in the Section 3.4.2, the first order necessary condition of optimality can be expressed as a stationary point (x^*,λ^*) of $L(x,\lambda)$ over x and λ. That is,

$$\begin{aligned} \frac{\partial f(x^*)}{\partial x_i} + \sum_j \lambda_j^* \frac{\partial h_j(x^*)}{\partial x_i} &= 0 \\ h_j(x^*) &= 0 \end{aligned} \tag{4.17}$$

The Lagrange programming ANN can be defined for the above problem as follows:

$$\frac{dx}{dt} = -\nabla_x L(x,\lambda) \qquad \frac{d\lambda}{dt} = -\nabla_\lambda L(x,\lambda) \tag{4.18}$$

The state equations for the ANN can be given as :

$$\frac{d(x_i)}{dt} = -\frac{\partial f}{\partial x_i} - \sum_j \lambda_j^* \frac{\partial h_j}{\partial x_i} \qquad \frac{d(h_j)}{dt} = h_j \tag{4.19}$$

Thus, the Lagrangian function decreases as follows:

$$\frac{dL(x,\lambda)}{dt}\Big|_{\lambda = constant} = \sum_i \lambda_j^* \frac{\partial L(x,\lambda)}{\partial x_i} \frac{d(x_i)}{dt} = -\sum_i \left(\frac{d(x_i)}{dt}\right)^2 \leq 0 \tag{4.20}$$

The above equation guarantees the existence of a stable equilibrium point in the asymptotic sense.

4.2.4 Application to power system optimization

In the field of power system operation, the Hopfield ANN and Kennedy, Chua, and Lin ANN have been applied to solve optimization problems. M.H. Sendaula and S.K. Biswas applied a combination of Hopfield-Tank type and Kennedy, Chua and Lin type ANNs to solve the unit commitment problem [36,37]. The unique feature of the proposed network was to solve the nonlinear programming problem and the combinatorial optimization problem simultaneously by one network. H.Sasaki and M. Watanabe examined the possibility of applying the Hopfield network to a combinatorial optimization problem in power systems, in particular to unit commitment [38]. They

developed a two-step solution method. First, the generators to start up at each period were determined by the network and then their outputs were generated by a conventional algorithm. J.H. Park and Y.S. Kim solved the economic dispatch problem [39] for piecewise quadratic cost curves using Hopfield network. Y. Fukuyama and Y. Ueki formulated dynamic economic load dispatching [40] using an artificial neural network as opposed to formulation in which a solution had to be obtained by nonlinear programming. The method used a probabilistic artificial neural network and effectively handles the associated constraints by heuristics. In an another development, S.Matuda and Y.Akimito solved the economic dispatch problem based on the technique of representating large numbers in neural network [41].

4.2.5 General remarks

The discussions in the previous subsections present an analysis of the different optimization ANN models from an implementation as well as a theoretical view point. The analysis indicates that the selection of a particular ANN model depends on the various parameters involved in the optimization problem. The salient features, which should be considered for an appropriate selection can be summarized as follows:

- The Tank and Hopfield ANN model follows a direct descent approach. Thus, it is more easy to implement for the problems without any constraints.

- Problems with a larger number of inequality constraints can be mapped into the Kennedy, Chua, and Lin ANN model with a smaller number of state variables.

- Problems with a larger number of equality constraints can be mapped into the Lagrange programming ANN model with a smaller number of state variables.

- The Kennedy, Chua, and Lin network's structural parameters corresponds with the coefficient vectors and the coefficient matrix of the optimization problem. Thus, the circuit formulation of the optimization problem becomes quite simple.

The proper selection of the ANN model for optimization needs adequate consideration of the problem form with regard to the above general remarks.

5. THEORETICAL DEVELOPMENT

5.1 LP and QP Neural Networks

Linear programming and nonlinear programming are frequently used as efficient tools for solving optimization problems. The technique for performing linear programming and quadratic programming uses the neural network architecture proposed by Kennedy, Chua and Lin. Transfer function of neurons in the circuit proposed by Kennedy, Chua and Lin followed the conventional quadratic penalty function. This choice resulted into inherent degenerating accuracy.

By analyzing the behavior of the conventional penalty function, Maa and Shanblat [42,43] discovered the reason for the inherent degenerating accuracy. Based on this, they proposed a new combination penalty function that ensured that the equilibrium point would be acceptably close to the optimal point. They modified the neural network with the new transfer function using the combination penalty function. Furthermore, they designed the corresponding circuit scheme for hardware realization.

An analysis of the Kennedy, Chua and Lin neural network from the system theory point of view was presented in Section 4.2.3.2. This section validates the circuit from the optimization theory view point. The validation indicates that the circuit is primarily aimed at solving nonlinear programming problems with inequality constraints. Although, an equality constraint can always be mathematically represented by a coupled set of inequality constraints, the simulation of the neural network is likely to encounter solution problems. This is generally true, because a circuit with two inverse diodes can not work properly.

To circumvent the above mentioned problem, the combination penalty function proposed by Maa and Shanblatt is modified by including time dependent terms. The proposed penalty function is designed to cause sufficient penalty to satisfy the equality constraints. The time dependence has been incorporated by modifying the transfer function of neurons with the introduction of terms dependent on iteration number in the software

simulation. From the system theory view point, the time dependent terms work as integral control to reduce the steady state error.

5.1.1 Validation of LP and NLP neural networks

The Kennedy, Chua and Lin network was shown in Figure 4.3. The architecture of the neural network can also be represented by a nonlinear circuit model, shown in Figure 5.1. Figure 5.1 follows from Figure 4.3 by merely writing the circuit equations for the neural network. This circuit model can be shown as a simulation of the optimality conditions for the general nonlinear programming method discussed in the Section 3.4.

Let us consider the general nonlinear programming problem given as follows:

Minimize

$$\Phi (x_1, x_2, \ldots x_q)$$

Subject to the constraints:

$$\begin{aligned} f_1(x_1, x_2, \ldots x_q) &\geq 0 \\ f_2(x_1, x_2, \ldots x_q) &\geq 0 \\ &\vdots \\ f_p(x_1, x_2, \ldots x_q) &\geq 0 \end{aligned} \tag{5.1}$$

Using the Lagrangian method for nonlinear programming problems as discussed in Section 3.4.2, the Lagrange function can be defined as follows:

$$L(x,\lambda) = \phi(x) + \sum_{j=1}^{p} \lambda_j f_j(x) \tag{5.2}$$

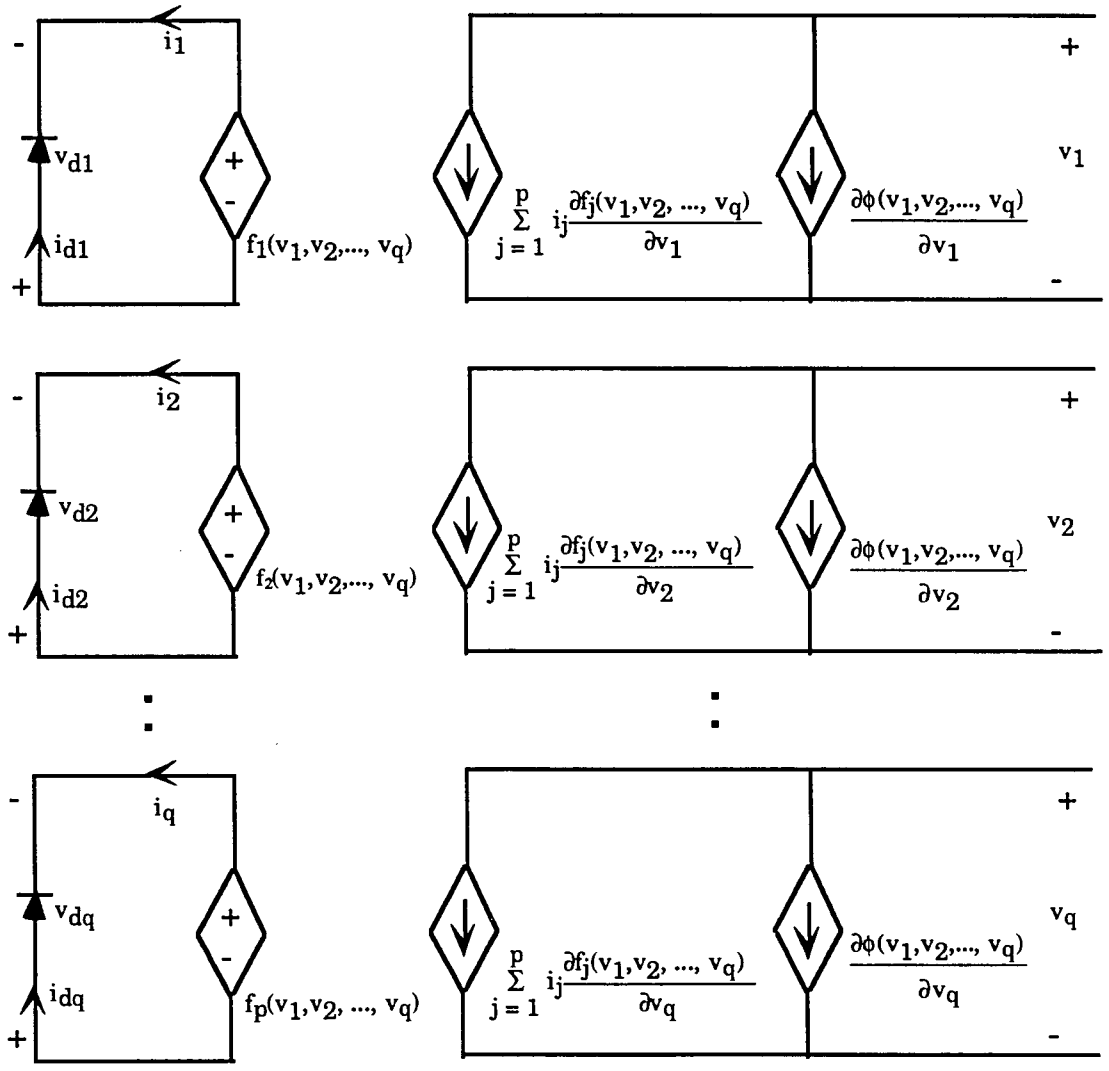

Figure 5.1 The canonical nonlinear programming circuit

Where the real constants λ_j are called the Lagrange multipliers. If the program has an optimal solution x^* , then according to the optimality conditions (Kuhn Tucker's conditions) stated in the Section 3.4.2, the following holds.

$$Min\ \phi(x) = \phi(x^*) \tag{5.3}$$

and

$$f_j(x^*) \geq 0, \qquad j = 1, 2, \ldots, p \tag{5.4.1}$$

$$\lambda_j^* \leq 0, \qquad j = 1, 2, \ldots, p \tag{5.4.2}$$

$$\lambda_j^* f_j(x^*) = 0, \qquad j = 1, 2, \ldots, p \tag{5.4.3}$$

$$\frac{\partial \phi}{\partial x_l}(x^*) + \sum_{j=1}^{p} \lambda_j^* \frac{\partial f}{\partial x_l}(x^*) = 0, \qquad j = 1, 2, \ldots, p \tag{5.4.4}$$

where, ϕ and f_j are assumed to be differentiable at x^*. Also, the set of inequality constraints is assumed to satisfy the regularity conditions stated in the Section 3.4.1.

Each diamond shape symbol enclosing a plus-minus sign in Figure 5.1 denotes a nonlinear controlled voltage source, whose terminal voltage depends on the node voltage v1, v2 , ... ,vq according to the prescribed nonlinear function f(v1, v2, ..., vq). Each diamond shape symbol in the middle column enclosing an arrowhead is a controlled current source whose terminal current depends on both the reversed diode currents i_{d1}, i_{d2}, ..., i_{dp}, and the node voltages v1, v2 , ... ,vq in accordance with the prescribed nonlinear function

$$\sum_{j=1}^{p} i_j \frac{\partial f_j(v_1, v_2, \ldots, v_q)}{\partial v_l} \tag{5.5}$$

Each diamond shape symbol on the right enclosing an arrowhead denotes a controlled current source whose terminal current depends on the node voltages v1, v2 , ... ,vq in accordance with the prescribed nonlinear function

$$\frac{\partial \phi (v_1, v_2, ..., v_q)}{\partial v_l} \tag{5.6}$$

The diodes in the figure denote ideal diodes and hence they satisfy following conditions:

$$i_{d_j} = 0, \quad v_{d_j} \leq 0 \tag{5.7}$$

$$i_{d_j} \geq 0, \quad v_{d_j} = 0 \tag{5.8}$$

and

$$i_{d_j} v_{d_j} = 0 \tag{5.9}$$

To prove that Figure 5.1 simulates the optimality conditions for nonlinear programming, note that condition 5.4.4 is obtained by applying KCL at each node l, l= 1,2, ..., q. Next, the constraint

$$f_j (v_1, v_2, ..., v_q) = - v_{d_j} \geq 0 \tag{5.10}$$

is embeded by including ideal diode in the circuit. This simulates the condition (5.4.1). When a constraint is non-binding, the corresponding controlled voltage source f_j has positive value. This keeps the diode reverse biased and, henece, there is no current in the diode circuit. In case of a binding constraint, the corresponding controlled voltage source f_j has negative value. This turns the diode on and, consequently, a current i_j flows in the circuit. The current i_j is fedback to the controlled current source in the

circuit. The parameter i_j represents Lagrange multiplier λ_j in terms of optimization theory. From (5.8) and (5.9), We note

$$\lambda_j = i_{d_j} \leq 0 \tag{5.11}$$

$$\lambda_j f_j (v_1, v_2, ..., v_q) = i_{d_j} v_{d_j} = 0 \tag{5.12}$$

The above set of equations satisfies conditions (5.4.2) and (5.4.3). Thus, the Kennedy Chua and Lin neural network satisfies optimality conditions for the nonlinear programming problem.

5.1.2 Modification of the transfer function of the neurons

Kennedy, Chua and Lin used the conventional penalty function as a transfer function of constraint neurons. Reference [43] presents a good tutorial on the analysis of the conventional penalty function. A typical conventional penalty function is shown in the Figure 5.2. Mathematically, the function can be given as follows:

$$P_i(x) = \begin{cases} 0 & \text{if } f_i(x) \geq 0 \\ |f_i(x)|^2 & \text{if } f_i(x) < 0 \end{cases} \tag{5.13}$$

In this case, as the circuit approaches the optimal point in the solution space, the outputs from constraint neurons become too week to affect the values of the variable neurons. The coefficient for the conventional penalty function discussed in Section 3.4.4 is usually taken as a constant real number. Theoretical analysis [43] shows that only when the coefficient reaches infinity, can the solution to the approximated optimization problem be the same as the solution to the original optimization problem.

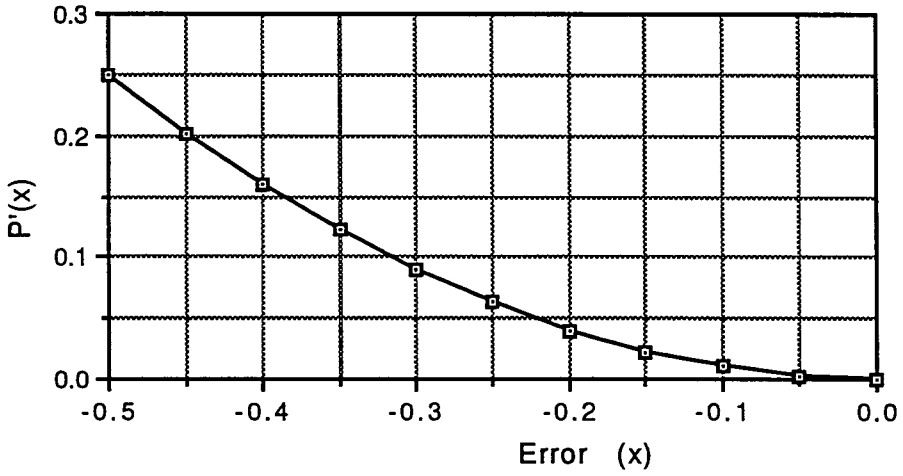

Figure 5.2 Conventional penalty function

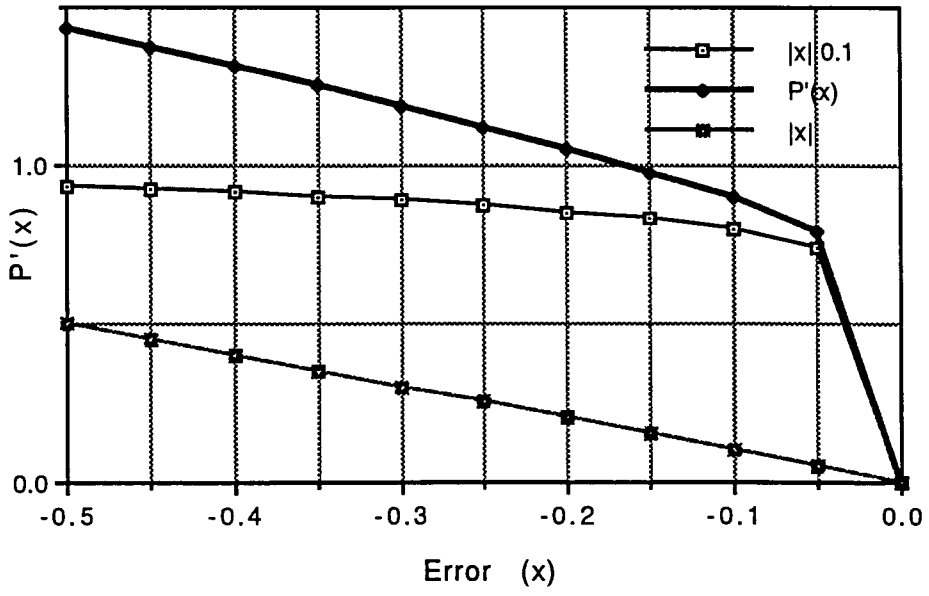

Figure 5.3 Combination penalty function

Chen, Maa and Shanblatt [43] proposed a new combination penalty function that is the sum of two penalty terms and is given as follows:

$$P_i(x) = \begin{cases} 0 & \text{if } f_i(x) \geq 0 \\ k_{1i}|f_i(x)|^2 + k_2|f_i(x)|^{\frac{p+1}{p}} & \text{if } f_i(x) < 0 \end{cases} \tag{5.14}$$

where, p is the parameter that controls the derivative of the penalty function in the neighborhood of the solution point when $f_i(x)$ is equal to zero. The first penalty term is the modified version of the conventional penalty term used in the reference [34], which is dominant when the solution point is far from the original solution point. The derivative of the second term is the polynomial in the absolute value of the constraint violation with its exponent less than unity, which provides significant feedback to adjust the state of the variable neurons when the solution point is near optimal point. Such a combination penalty function is shown in the Figure 5.3

As is evident from the discussion in Section 5.1.1, the Kennedy, Chua and Lin neural network is primarily aimed at solving nonlinear programming problems with inequality constraints. Although an equality constraint can be expressed by a coupled set of inequality constraints, the network formulation will involve the diodes connected back to back. This may result in either steady state error or oscillation in the time domain. To circumvent this problem, the following modification for the penalty function is introduced.

$$P_{newi}(x) = P_i(x) * q(t) \tag{5.15}$$

where, q(t) is a function containing the time dependent terms to provide enough penalty to satisfy the equality constraints. To facilitate the implementation, q(t) is suggested to have the terms containing the iteration number in the software simulation. This approach has not been found in the literature.

The effect of including q(t) has been shown in the Figure 5.4. The function q(t) should be as smooth as possible for maintaining the numerical stability of the algorithm. The measure of smoothness can be obtained by taking the higher derivatives of the function. When the solution point is far from the optimal point, the error function is usually high. Hence, the value of the function q(t) should be quite low in that region. Otherwise, the trajectory of simulation may turn unstable. The function q(t) used for the neural network simulation in the present work has been chosen of the following form

$$q(t) = \begin{cases} 1 & \text{if } t < t_0 \\ \frac{t}{t_0} & \text{if } t \geq t_0 \end{cases} \tag{5.16}$$

where, t is the iteration number in the software simulation and t_0 is a fixed number, which is chosen heuristically. This modification generates a set of scaled penalty functions as shown in Figure 5.4.

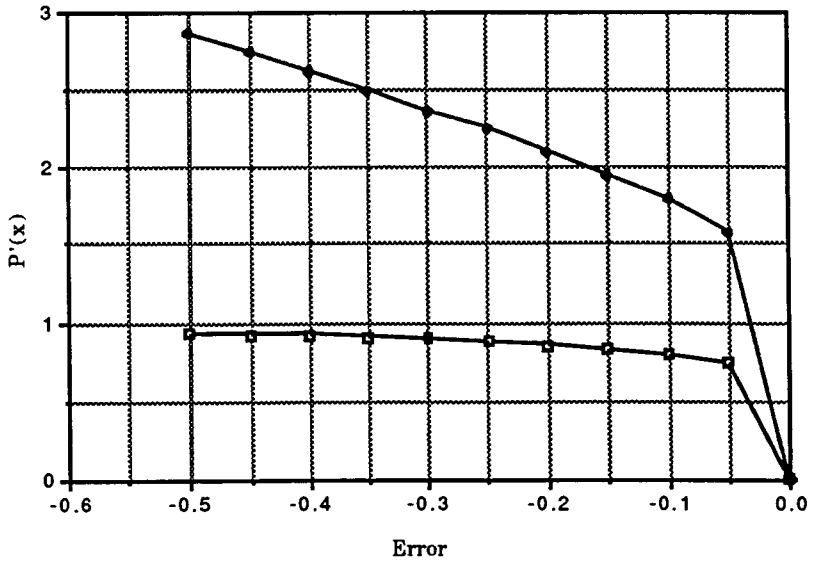

Figure 5.4 Effect of the time function on the penalty function

Inclusion of the new term $q(t)$ has the effect of integral control in the time simulation of the network. Thus, the steady state error because of the coupled constraints is expected to reduce. The results obtained with the modified transfer function has produced encouraging results.

5.2 Proposed Simulation Algorithm

The Kennedy, Chua and Lin neural network is a nonlinear dynamic system from a system theory view point as discussed in the Section 4.2.3.2. Different techniques may be used to solve the nonlinear neural dynamic system. The algorithm proposed in this work is a very simple approach from problem formulation view point. Design of the algorithm is based on the knowledge of modern control theory. The proposed methodology approximates the nonlinear structure of the artificial neural network to a sampled data control system by clamping the nonlinear states for efficient simulation. The simulation shows that the clamping is justified as long as the sampling period of the clamped-state variables is too small to violate the nonlinearity of the transfer function of neurons.

The proposed algorithm begins with formulating the control system matrices for the neural network. Development of system matrices proceeds in two phases. First, the state space representation of the neural network is formulated. The functional relationship of the nonlinear variables with the state space variables is represented in a matrix form. For realization of a linear state space system, the nonlinear variables are treated as new state variables known as the clamped-state variables of the system. Since, the clamped-state variables are time invariant variables, their first order partial time derivatives with respect to themselves are assumed to be zero. However, total time derivative of the clamped-state variables may not be zero, as they may be a function of the state variables in the system. The sampling time of the clamped-state variables is assumed to be equal to time simulation step size. Finally, the state model of the system is converted to homogeneous equivalents for the state models [44]. Thus, the nonlinear architecture of the Kennedy,

Chua and Lin artificial neural network is modeled as a linear homogeneous state space system.

The proposed solution of the linear homogeneous state space system is based on the state transition matrix approach. The classical matrix exponential technique for calculating state transition matrix of a linear causal relaxed and time-invariant system [44,45] is used as the basis of the simulation algorithm. The necessary steps are designed to include the behavior of nonlinear variables in the control system. Finally, a flowchart of the overall solution approach is presented for the neural network simulation.

5.2.1 The state model

The state model is a mathematical description of the dynamics of a system. This dynamical model is in the form of a set of first-order, ordinary differential equations called the state equations and a set of algebraic equations called the output equations. The state equations and output equations are usually represented in matrix form.

5.2.1.1 The general form of state model If the dynamics of a system are modeled by ordinary differential equations, the most general form of the state model is as follows:

$$\dot{\bar{x}} = \bar{f}(\bar{x}, \bar{u}, \bar{t}) \tag{5.17}$$

$$\bar{y} = \bar{g}(\bar{x}, \bar{u}, \bar{t}) \tag{5.18}$$

where,

- $\bar{x}$ is an n-vector function of time referred to as the system states.
- $\bar{u}$ is an m-vector of inputs or controls,
- $\bar{y}$ is an r-vector of system outputs
- $\dot{\bar{x}}$ is an n-vector of the time derivatives of the system state variables $\bar{x}$.

In Equations (5.17) and (5.18), $\bar{f}$ (.) and $\bar{g}$ (.) are n-dimensional vector functions of any type. Note that Equation (5.17) is a set of first order, first -derivative explicit differential equations relating the system dynamic variables, (the states) to the system inputs. Further, Equation (5.18) is a set of algebraic equations or measurement functions relating the system outputs to the system states and inputs. The number of state variables, n, is referred to as the *dynamic order* of the system.

5.2.1.2 Linear state model In the case of a system describable by a linear differential equation, $\bar{f}$ (.) and $\bar{g}$ (.) are linear functions and the state model can be written in following matrix form:

$$\dot{\bar{x}} = \bar{A}\,\bar{x} + \bar{B}\,\bar{u} \tag{5.19}$$

$$\bar{y} = \bar{C}\,\bar{x} + \bar{E}\,\bar{u} \tag{5.20}$$

where, $\bar{x}$, $\bar{u}$, $\bar{y}$, and $\dot{\bar{x}}$ are as previously defined and the system matrices $\bar{A}$, $\bar{B}$, $\bar{C}$, and $\bar{E}$ are coefficient matrices that could, in general, be functions of time. In the event that some of the coefficients are dependent on time, the model is appropriately called a time-varying state model. Otherwise, it is referred to as a constant or stationary model. Note that the linear state model given by Equations (5.19) and (5.20) represents a causal system as causality is inherent. A causal and linear state model is said to be relaxed at time t_0 if and only if the output $y[t_0,\infty]$ is solely and uniquely excited by $u[t_0,\infty]$. Furthermore, if the characteristics of a system do not change with time, then the system is said to be time invariant. A precise definition of a time invariant , causal and relaxed linear system is mentioned in reference [46].

5.2.1.3 Homogeneous equivalent state model It is convenient for simulation purposes to convert the state model of Equations (5.19) and (5.20) to an equivalent form that is homogeneous in nature. In this case, the model becomes

$$\dot{\bar{v}} = \bar{A}\,\bar{v} \tag{5.21}$$

$$\bar{y} = \bar{C}\,\bar{v} \tag{5.22}$$

where, $\bar{v}$ is a new vector containing both the components of $\bar{x}$ and $\bar{u}$. The conversion of a linear state model to the homogeneous equivalent of a state model can be performed by defining new set of vectors.

Let us suppose that we can find some p-dimensional vector $\bar{z}$ such that we have

$$\bar{u} = \bar{e}\,\bar{z} \tag{5.23}$$

where, $\bar{e}$ is an m*p matrix of constants. Further, suppose we know that $\bar{z}$ satisfies a homogeneous state model of the form

$$\dot{\bar{z}} = \bar{f}\,\bar{z} \tag{5.24}$$

Next, suppose we form a new vector as follows:

$$\bar{v} = \begin{bmatrix} \bar{x} \\ \bar{z} \end{bmatrix} \tag{5.25}$$

Then, we see that the following holds

$$\dot{\bar{v}} = \begin{bmatrix} \dot{\bar{x}} \\ \dot{\bar{z}} \end{bmatrix} = \begin{bmatrix} \bar{A}\,\bar{x} + \bar{B}\,\bar{u} \\ \bar{f}\,\bar{z} \end{bmatrix} = \begin{bmatrix} \bar{A}\,\bar{x} + \bar{B}\,\bar{e}\,\bar{z} \\ \bar{f}\,\bar{z} \end{bmatrix} \tag{5.26}$$

$$\bar{y} = \bar{c}\,\bar{x} \quad \bar{E}\,\bar{e}\,\bar{z} \tag{5.27}$$

Finally, Equations (5.26) and (5.27) can be rewritten as follows:

$$\begin{bmatrix} \dot{\bar{x}} \\ \dot{\bar{z}} \end{bmatrix} = \begin{bmatrix} \bar{A} & \bar{B}\,\bar{e} \\ \bar{0} & \bar{f} \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{z} \end{bmatrix} \tag{5.28}$$

$$\bar{y} = \begin{bmatrix} \bar{C} & \bar{E}\,\bar{e} \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{z} \end{bmatrix} \tag{5.29}$$

Equations (5.28) and (5.29) are the desired equations and can be written in a more compact notation as follows:

$$\dot{\bar{v}} = \overline{A_0}\,\bar{v} \tag{5.30}$$

$$\bar{y} = \overline{C_0}\,\bar{v} \tag{5.31}$$

where, we have defined the matrices A_0 and C_0 as

$$\overline{A_0} = \begin{bmatrix} \overline{A} & \overline{B}\,\bar{e} \\ \overline{0} & \overline{f} \end{bmatrix} \tag{5.32}$$

$$\overline{C_0} = \begin{bmatrix} \overline{C} & \overline{E}\,\bar{e} \end{bmatrix} \tag{5.33}$$

5.2.1.4 Solution of homogeneous equivalent state model The solution of homogeneous equivalent state model is given as

$$\bar{v}(t) = \overline{e^{A_0(t\,-\,t_0)}}\,\bar{v}(t_0) \tag{5.34}$$

where,

$$\overline{e^{A_0(t\,-\,t_0)}} \cong \sum_{k=0}^{\infty} \frac{(t\,-\,t_0)^k}{k!}\overline{A_0}^k \tag{5.35}$$

is a matrix power series that converges to a single matrix function of time usually referred to as the state transition matrix, and $\bar{v}(t_0)$ is the initial condition given at $t = t_0$. The state transition matrix is often symbolized by the expression $\bar{\phi}(t - t_0)$ and, hence, Equation (5.34) can be written as

$$\bar{v}(t) = \bar{\phi}(t - t_0)\,\bar{v}(t_0) \tag{5.36}$$

Equation (5.34) together with output equation (5.31) is used as the basis for the numerical algorithm developed for the simulation.

5.2.2 The state space representation of the Kennedy, Chua and Lin network

5.2.2.1 Formulation of clamped-state variable The circuit representation of Kennedy, Chua and Lin network is shown in Figure 5.5. In the circuit, the variable neurons are linear state variables and the constraint neurons are nonlinear elements. Hence, it is possible to develop the state space equation when the behavior of the nonlinear state variables can be realized with a linear approximation.

To compute the state space realization, the constraint neurons are modeled as clamped-state variables. This implies that the characteristics of the constraint neurons are assumed to be linear during the interval of time simulation step. Thus, the validity of assumption depends on the choice of time simulation step size. A generalized analytical derivation of the time simulation step size is complicated. Therefore, the choice of time simulation step size is left to the knowledge of nonlinearity of constraint neurons.

As discussed in Section 5.1.2, the nonlinearity of the constraint neuron is given by the penalty function expressed by Equations (5.15) and (5.16). The convergence criteria for the network simulation imposes the restriction of smoothness in the penalty function. This forces the nonlinearity of the constraint neurons to be a nice smooth function. Hence, from simulation view point, the choice of the time simulation step size is not restrictive.

5.2.2.2 State space realization The circuit equation of the Kennedy, Chua and Lin network can be written as follows:

$$C_i \frac{dx_i}{dt} = -\frac{\partial \phi}{\partial x_i} - \sum_{j=1}^{p} i_j \frac{\partial f}{\partial x_i}, \qquad i = 1,2, ..., q \tag{5.37}$$

$$i_j = P(f_j(x)), \qquad j = 1,2, ..., p \tag{5.38}$$

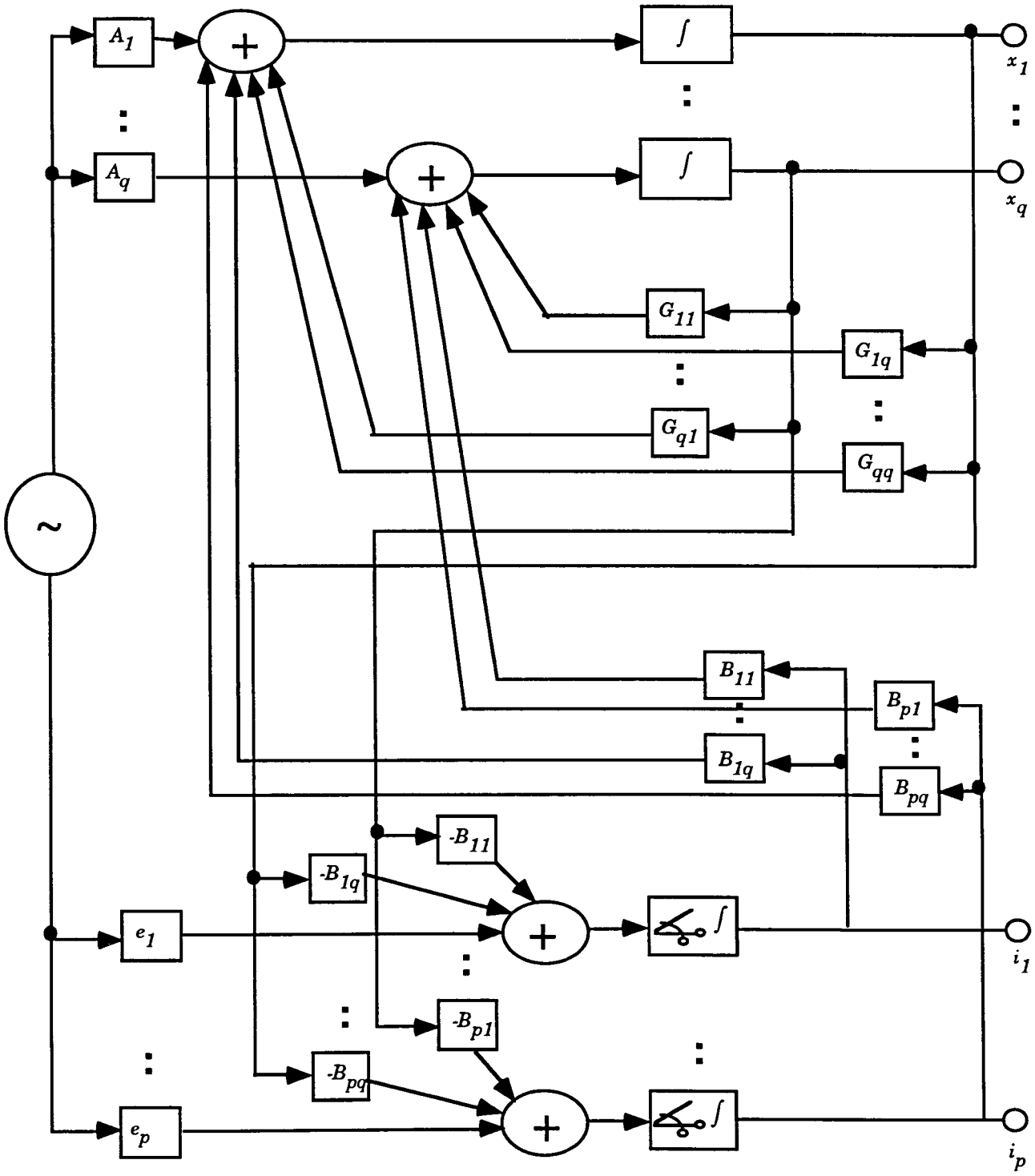

Figure 5.5 Block diagram for Kennedy, Chua and Lin neural network

Considering the nonlinear variables i_j as the clamped-state variables, Equation (5.37) can be written in linear state space form as follows:

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_q \\ \dot{i}_1 \\ \vdots \\ \dot{i}_p \end{bmatrix} = \begin{bmatrix} 0 & B^T \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \dot{x}_q \\ i_1 \\ \vdots \\ i_j \end{bmatrix} + [A^T][\mathrm{I}] \tag{5.39}$$

where, the functional relationship of the clamped-state variables i_j with the ordinary state variables is given by the matrix E as follows:

$$\begin{bmatrix} i_1 \\ \vdots \\ i_j \end{bmatrix} = [B \quad 0 \quad -e] \begin{bmatrix} x_1 \\ \vdots \\ x_q \end{bmatrix} \tag{5.40}$$

Equation (5.39) is not in the form of a homogeneous linear state space representation. As discussed in the Section 5.2.1.3, Equation (5.39) can be converted to a homogeneous equivalent form by introducing a new state variable u . Then, the homogeneous state space representation of the network can be given by following equation:

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_q \\ \dot{i}_1 \\ \vdots \\ \dot{i}_p \\ \dot{u} \end{bmatrix} = \begin{bmatrix} 0 & B^T & A^T \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \dot{x}_q \\ i_1 \\ \vdots \\ i_j \\ u \end{bmatrix} \tag{5.41}$$

Comparing Equations (5.21) and (5.41) , the matrix A_0 is found to be:

$$A_0 = \begin{bmatrix} 0 & B^T & A^T \\ 0 & 0 & 0 \end{bmatrix} \tag{5.42}$$

Using Equations (5.35) and (5.42), the state transition matrix of the formulated the Kennedy, Chua and Lin network can be computed.

5.2.3 Overall solution approach

5.2.3.1 The simulation algorithm A flowchart of the proposed solution approach is shown in Figure 5.6. The overall solution proceeds as follows:

- Develop the linear or quadratic programming formulation for a given optimization problem. Reduce the formulation to conform with the Equation (5.41)

- Using Equation (5.41), develop the homogeneous equivalent state model of the neural network for the given optimization problem.

- Using Equation (5.40), develop the functional relation mapping matrix that represents the feedback from the variable neurons to the constraint neurons.

- Derive the state matrix A_0 from the developed homogeneous equivalent state space model.

- Generate a record to keep track of the clamped-state variables in the homogeneous equivalent state model.

- Using Equation (5.35), calculate the state transition matrix for the homogeneous equivalent state model.

- Perform simulation for a heuristically chosen time simulation step size until the solution converges. At the end of every time simulation step size, update the clamped-state variable for its nonlinearity using the functional relation mapping matrix E.

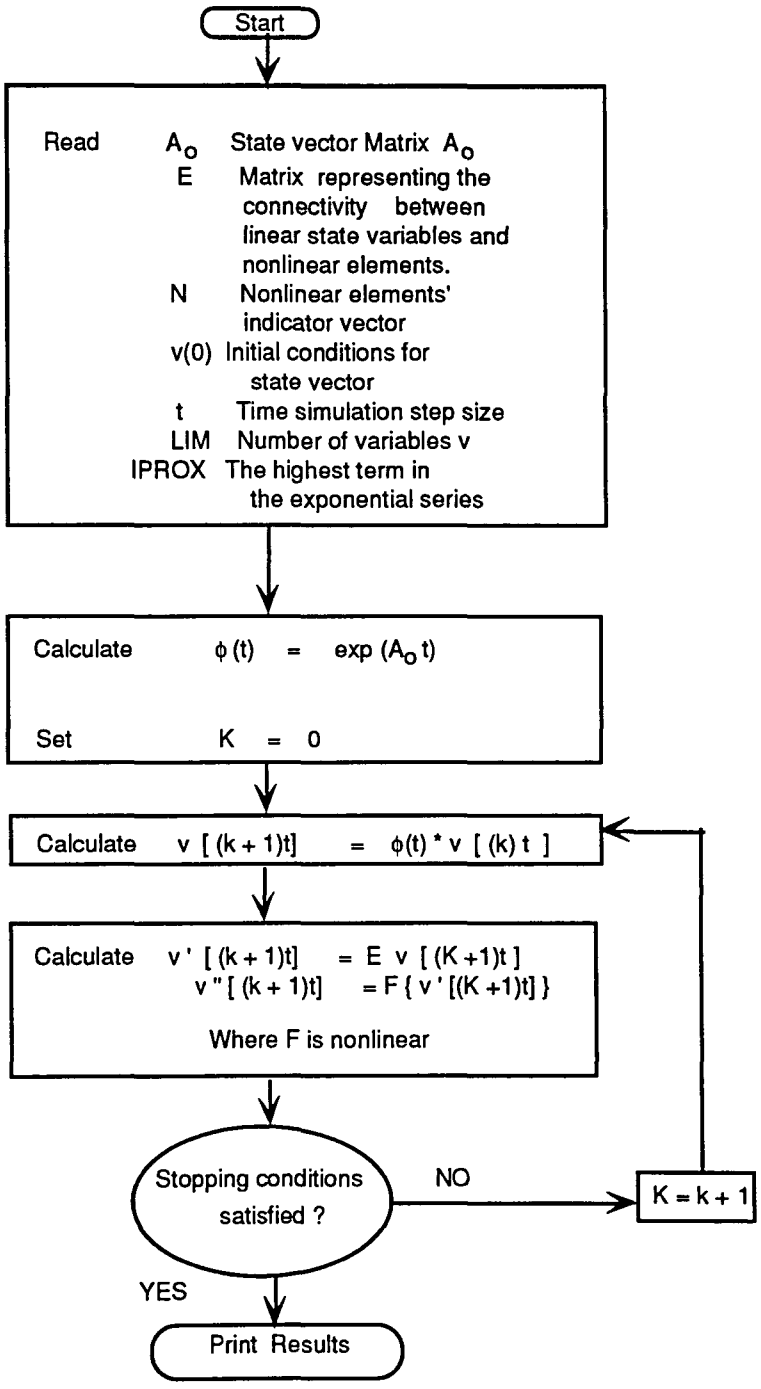

Figure 5.6 The clamped-state variable approach

5.2.3.2 Scaling and normalization The proposed algorithm employs numerous multiplications of the state transition matrix. It is possible to develop numerical instability, when the multiplication is of high order [47]. Hence, the system equations should be properly scaled and normalized before the simulation.

Since the state transition matrix predominantly consists of the sensitivity coefficient matrix B, the constraint equations should be scaled to improve the condition number of the state transition matrix. The voltage values should also be normalized to improve the numerical stability of the algorithm. The choices for the parameters involved in scaling and normalization are left to engineering judgment.

5.2.3.3 Sparsity-based formulation Note, that although the number of neurons become larger with the larger dimension of optimization problems, matrices A and E remain extremely sparse. Sparsity comes from the inherent structure of the system matrices as given below:

$$A_0 = \begin{bmatrix} 0 & A_{12} \\ 0 & 0 \end{bmatrix} \tag{5.43}$$

$$E = [E_{11} \quad 0 \quad E_{1n}] \tag{5.44}$$

where, A_{12} is (q) * (q+p+1) matrix and A_0 is (q+p+1) * (q+p+1) matrix. This indicates that the percentage of sparsity can be given by q / (q+p+1). Since ninety percent of the computer time taken by the algorithm is dedicated to matrix manipulation, it is evident that a sparsity-based algorithm will result into saving of approximately [0.9 * q / (q+p+1)] percent of the computer time. Hence, sparsity-based techniques can be used for matrix operations in solving the larger optimization problems to save computation time.

5.3 Economic Dispatch Problem

Generation scheduling is the basic function for power system economics and operation. The economic power dispatch is used for generation scheduling. The goal of the dispatch is to minimize total cost of power generation subject to the constraint that the sum of power generation of individual units be within their respective operating limits [48]. The dispatch problem involves the modeling of generating units and that of the loss representation in the electrical power network. The dispatch program is used for operation as well as planning activities in electrical power systems.

5.3.1 Modeling of generator units

A fossil generating unit's hourly fuel input is expressed as a function of its power output, $F_i(P_i)$. This expression is referred to as the incremental heat rate curve or input-output (I/O) characteristic. Several mathematical functions have been used for F_i, such as exponential, polynomial etc. The most commonly used are the piecewise linear representation, the quadratic representation, and the reduced cubic representation. Equations (5.45) and (5.46) represent the quadratic representation and the reduced cubic representation respectively.

$$F_i(P_i) = a_i + b_i P_i + c_i P_i^2 \tag{5.45}$$

$$F_i(P_i) = a_i + b_i P_i + d_i P_i^3 \tag{5.46}$$

where,

$F_i(P_i)$ = fuel input (MBTU/hr)

P_i = power output (MW)

i = generator index

a_i, b_i, c_i, d_i = input-output characteristic coefficient.

The input-output equation coefficients may be obtained from the curve-fitting measured data or typical design data, using the appropriate polynomials. Since, the modeling is determined using a statistical curve fitting technique, an inadvertent inaccuracy is associated with the model. The value of inaccuracy may be as much as ten percent

A piecewise linear representation consists of break points with linearized segments. The curve is generated by selecting the break points at different power P_is, which determine the bounds of linear segments. Linear segments are fit using the measured data within the corresponding bounds. Choice of the number of break points is an another optimization problem, which is a tradeoff between the speed of the solution algorithm and accuracy in the results. In practice, this number depends on how nonlinear the generation characteristics are.

Multiplying the I/O equation by the fuel price, gives an equation, $C_i(P_i)$, which provides the hourly fuel cost as a function of the power output. Since the fuel prices include prorated maintenance and operation costs, $C_i(P_i)$ actually relates the hourly unit operating costs to the net generated power.

5.3.2 Loss representation

Power system losses are often incorporated for solving the economic dispatch problem. In power system operation, different methods are used to represent losses for economic dispatch. Some of the most common methods are described below.

5.3.2.1 B matrix loss representation In this case, the losses are given by the following matrix equation:

$$P_{losses} = P^T B P + P^T B_0 + B_{00} \tag{5.47}$$

or,

$$P_{losses} = \sum_i P_i B_{ij} P_j + \sum_i B_{i0} P_i + B_{00} \tag{5.48}$$

where,

P = vector of all outputs
B = square matrix of the same leading dimension as P
B_{i0} = Vector of same length as P
B_{00} = Constant

This can be shown [48] that the B-matrix formulation introduces a penalty factor given by

$$Pf_i = \frac{1}{1 - 2\sum_j B_{ij} P_j + B_{00}}$$

The B matrix coefficients are derived using the equivalent total load center approach. The B matrix loss representation was developed mainly by L. Kirchmayer [49, 50] and G. Kron [51]. The penalty factors penalize those units that are farther from the load center. The B matrix is the most complex representation as it is based on the detailed modeling of the actual transmission network. But, the loss matrix is always an approximation, because assumptions made in the calculation are based on the conformity of load and power that are never seen on a real system. Hence, the modern practice is to use the real time load condition and the loss penalty factor as described briefly in the following subsections.

5.3.2.2 Constant penalty factor representation In this case, penalty factors are assumed to be constant independent of the unit power outputs. Theses values are derived somewhat heuristically using actual data and past experience. Losses are either considered to be included in the load demand values or a function of the load demand as calculated using a polynomial.

5.3.2.3 Loss calculation using state estimators Modern practice in power system operation is to calculate the losses using real time load conditions and the loss penalty factor obtained by on-line computation using state estimators. State estimators provide the real time power flow state in the network. An interface program is usually employed to calculate the penalty factors for the current power flow state. This approach is commonly used as it is the most accurate representation of the losses.

5.3.3 Mathematical formulation of economic dispatch

The goal of economic dispatch is to minimize the total cost of generation subject to constraints that the sum of power generation must equal the received load and the network losses and the power generation of individual units be within their respective operating limits. Mathematically, the problem may be formulated as follows

Minimize

$$F_T = \sum_{i=1}^{n} F_i(P_i)$$

Subject to :

$$P_R - \sum_{i=1}^{n} P_i + P_L = 0$$

$$P_{imin} \leq P_i \leq P_{imax} \qquad (5.48)$$

where

F_T = Total cost of power generation
$F_i(P_i)$ = Cost of generation of i-th unit
P_R = Total load to be served
P_L = Power system transmission losses
P_i = Power generation of i-th unit
P_{imin} = Minimum power generation limit for i-th unit
P_{imax} = Maximum power generation limit for i-th unit

In general, the optimal allocation of system generation among the individual units of a power system is based on a marginal cost basis. The Lagrange multiplier of the coordination Equation [48] formulated for equation (5.48) works as marginal cost. The cost function of generation and power system transmission losses are modeled as described in the Section 5.3.2.

Equation (5.48) represents the basic structure of economic dispatch problem. However, in real life, there may be some practical requirements such as reserve margin constraints [52] and emission constraints [53] that may not allow the operation of system at the optimal point of problem (5.48). A detailed review of recent advances in economic dispatch is reported in reference [54].

5.3.4 Economic dispatch simulation using an artificial neural network

The developed simulation scheme assumes the quadratic representation or piecewise linear representation of the generator model. First, the simulation algorithm is developed for the lossless economic dispatch case. Next, transmission losses are incorporated by the constant penalty representation. Finally, a hybrid neural network algorithm for economic dispatch with piecewise linear cost curves is developed.

5.3.4.1 Economic dispatch without losses In this case, the quadratic representation of cost curve for generation is used. Thus, the economic dispatch problem becomes a quadratic programming problem. Furthermore, the quadratic cost coefficient matrix for the overall problem turns out to be strictly positive definite. Thus, the Kennedy, Chua, and Lin network can be applied to solve the problem using the scheme given in Figure 5.6.

5.3.4.2 Real time economic dispatch with losses In this case, again the quadratic representation of cost curve for generation is used. Loss representation is made using either constant penalty function (Section 5.3.2.2) or using state estimators (Section 5.3.2.3) and real time load conditions.

If the loss representation is made using constant penalty factors, then the constraint equations remain linear. Thus, the economic dispatch problem remains a quadratic programming problem. However, the values in functional relationship mapping matrix E is changed, since a new feedback term is introduced to the constraint neuron computing the power balance constraints. The neural network algorithm for real-time economic dispatch is summarized in Figure 5.7.

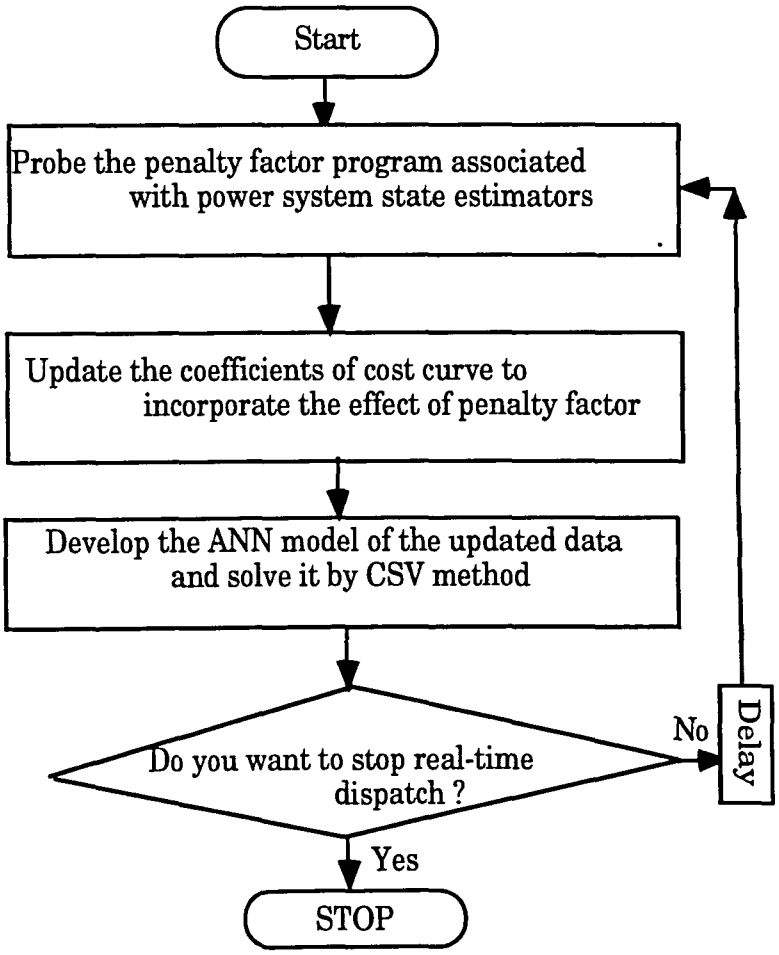

Figure 5.7 Real-time economic dispatch using a neural network

The loss representation used in this simulation is based on the following formulation:

$$\sum P_i = D + L + I \tag{5.49}$$

where,

P_i = Power generated by ith unit.
D = Load demand
L = Transmission losses
I = Interchange power

Incremental loss (ITL)can be calculated as follows:

$$ITL = \sum \frac{\partial L}{\partial P_i} \Delta P_i \tag{5.50}$$

A summary of the classical optimization algorithms for real time economic dispatch is reported in reference [55]. Unlike most classical algorithms, the proposed algorithm does not contain the double loop.

5.3.4.3 Hybrid dispatch algorithm with PWL cost curves Piecewise linear approximation of the cost function of generating units is quite common in digital computer execution of economic dispatch. This representation is supposed to be quite accurate generator modeling as it can capture the nonlinear characteristics of generation in the best possible way.

In this case, the economic dispatch becomes a mixed integer programming problem. Classical optimization theory suggests a number of methods to handle mixed integer programming problems. Most of them are quite complex from the theoretical as well as the implementation view point. Therefore, in practice, mixed integer programming problems are quite frequently solved by using apriori knowledge and heuristics.

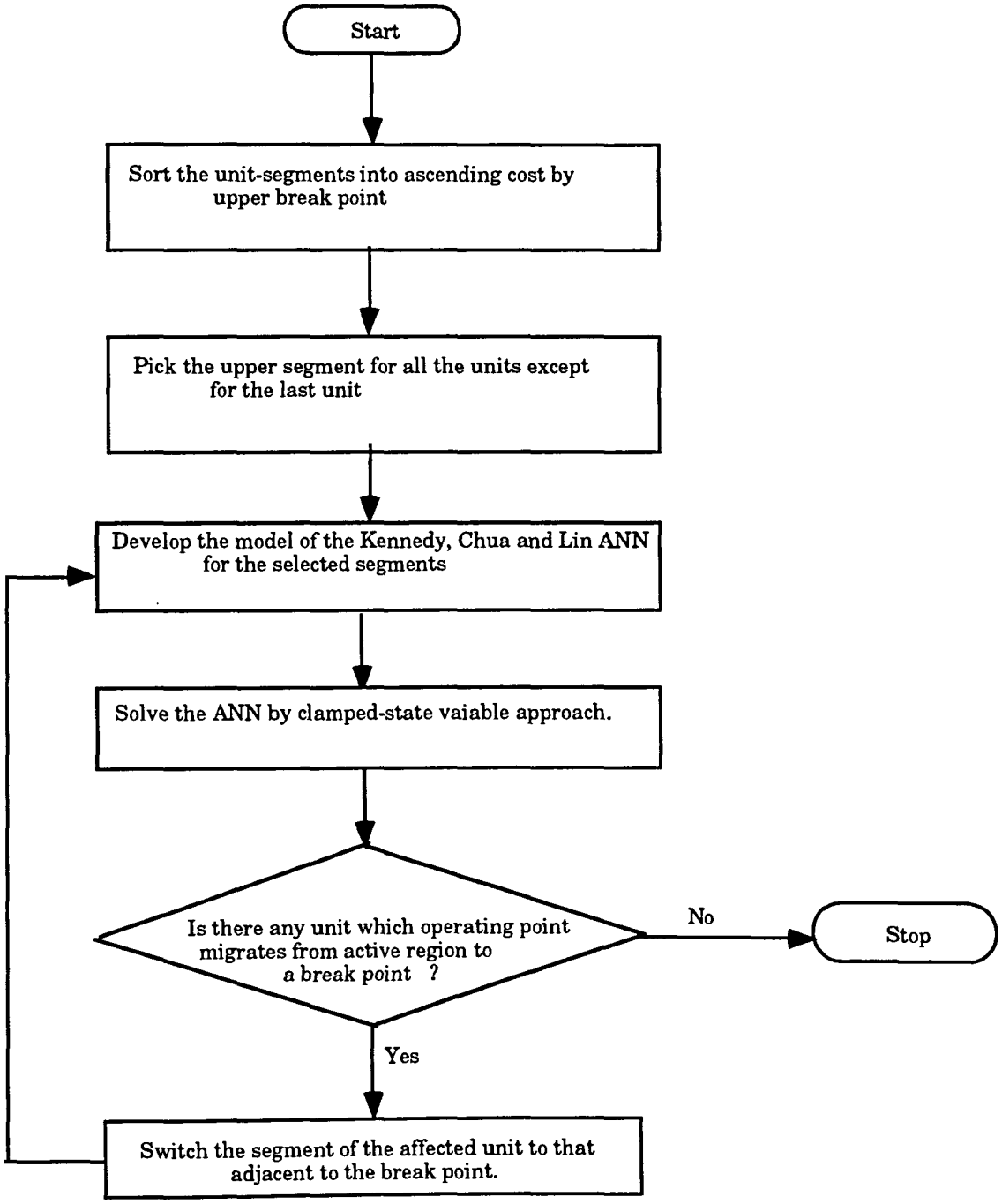

Figure 5.8 A hybrid dispatch algorithm with PWL cost curve using a neural network

One of the heuristics used for economic dispatch with piecewise-linear cost curves is merit order loading [55]. This algorithm is one of the quickest unconstrained economic dispatch algorithm. Constraints in the dispatch program are externally satisfied. The merit order loading starts from the least expensive unit-segment and incrementally loads each unit-segment. The loading is successively performed until all the energy constraints are satisfied. The proposed method captures the simplicity of merit order loading and is suitable for real time implementation. The simulation algorithm is shown in Figure 5.8.

6. NUMERICAL RESULTS

A simulation program was coded in FORTRAN-77 to implement both linear and quadratic programming with the Kennedy, Chua and Lin neural network. New penalty function as described in Section 5.1.2 was used as the nonlinearity of the constraint neurons in the Kennedy, Chua and Lin circuit. The algorithm was based on the clamped-state variable approach introduced in the last chapter. The code developed was same for the linear programming and quadratic programming algorithms with the same input format. Several cases were simulated on Sun 4/c SPARC work station. To validate the algorithm, the results obtained for different cases were compared to those obtained by classical optimization techniques. The test cases used for simulation were as follows:

(a) A generic linear programming example.
(b) A generic quadratic programming example.
(c) Economic dispatch calculation
    (i) Economic dispatch without loss modeling
    (ii) Economic dispatch with loss modeling using real time load condition and state estimators
    (iii) Economic dispatch with loss modeling using constant penalty factors
    (iv) Economic dispatch with piecewise linear cost curves.

To facilitate the above simulations, a subroutine was coded that generated the state variable matrix A and functional relationship matrix E for a given problem. MATLAB was used to solve the same problems by classical optimization methods. In all the cases, errors were calculated with respect to the results found by classical optimization methods. Computer time (cpu time) for Sun SPARC workstation was also recorded.

6.1 Example 1: A Generic Linear Programming Example

A linear program of the following form was used as the test case for the simulation.

Minimize

$$\phi(x_1,x_2) = c_1x_1 + c_2x_2$$

Subject to:

$$\frac{5}{12}x_1 + x_2 \leq \frac{35}{12}$$

$$\frac{5}{2}x_1 + x_2 \leq \frac{35}{2}$$

$$-x_1 \leq -5$$

$$x_2 \leq 5$$

Table 6.1 shows the data for the cost coefficient parameter 'c' in the linear programming.

Table 6.1. Data for a generic linear programming

Examples	c_1	c_2
Case (1a)	-1	-1
Case (1b)	-1	1
Case (1c)	1	-1
Case (1d)	1	1

The linear program in example 1 was represented by a state space model of two linear state variables (number of optimization variables) and four clamped-state variables (number of constraint equations). Coefficients of x1 and x2 in the constraint equations represented feedback from the linear state variables to the clamped state-variables.

The feedback elements constituted the matrix E and the coefficients of x1 and x2 constituted the matrix A as explained in the Section 5.2. The voltage values of the formulated circuit were normalized by a factor of 2. The first and third constraint equations were scaled by multiplying them by 3.00 and 0.75 respectively. Choices for the parameters involved in scaling and normalization were experimental. The resultant matrices A and E were as follows:

$$A = \begin{bmatrix} 0.00 & 0.00 & -12.50 & -20.00 & 10.00 & 0.00 & -10.00 \\ 0.00 & 0.00 & 30.00 & -8.00 & -10.00 & 0.00 & -10.00 \\ & & & 0 & & & \end{bmatrix}$$

$$E = \begin{bmatrix} 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 1.25 & -3.00 & 0.00 & 0.00 & 0.00 & 0.00 & -17.5 \\ 2.00 & 0.800 & 0.00 & 0.00 & 0.00 & 0.00 & 84.0 \\ -1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -10.0 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 30.0 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix}$$

The simulation was made with the initial condition given as below:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.0 \end{bmatrix}$$

Note that the initial condition for the variables is not a basic feasible solution. This indicates that the ANN simulation for linear programming can handle basic feasible as well as basic infeasible initial condition. On the other hand, the Simplex method for solving LP requires a basic feasible solution as the initial set. This feature of ANN simulation may have practical advantage. For example, one may have an estimate, which may be basic infeasible, but quite close to optimal solution. In that case, it would be difficult to use the Simplex method. Whereas, the ANN approach would be more appropriate.

Results for the simulation of the four different cases are shown in Table 6.2. Computer time (cpu seconds) indicated in the results correspond to SUN Sparc work stations. Theoretical results were computed using MATLAB.

Table 6.2. Results for the generic linear programming

Example	Parameter	Theoretical Result	ANN Result
Case 1a	x1	5.000	5.004
	x2	5.000	5.002
	Error		0.06%
	Iteration		110
	cpu time		0.4 cpus
Case 1b	x1	7.000	7.012
	x2	0.000	0.022
	Error		0.14%
	Iteration		275
	cpu time		0.7cpus
Case 1c	x1	-5.000	-5.004
	x2	-5.000	-4.979
	Error		0.17%
	Iteration		110
	cpu time		0.4cpus
Case 1d	x1	-5.000	-5.002
	x2	5.000	4.998
	Error		$0.08*10^{-6}$%
	Iteration		110
	cpu time		0.4 cpus

Note that the corresponding solutions to the different cost coefficient vectors are four vertices of the polytope generated by hyperplanes of the constraint equations. Thus, solutions for the four different cases by the proposed algorithm verify the neural network dynamics in all directions.

The trajectory of the network dynamics for case (1a) with two initial conditions are shown in Figure 6.1. When the initial condition (point (a)) is feasible, the minimization is done in the direction of steepest descent. When the trajectory hits a constraint boundary, the minimization is done along the hyperplane. In case of an infeasible initial condition (point (b)), the constraint violation drives the trajectory perpendicular to the guiding hyperplane. This indicates that the network is capable of handling any initial condition.

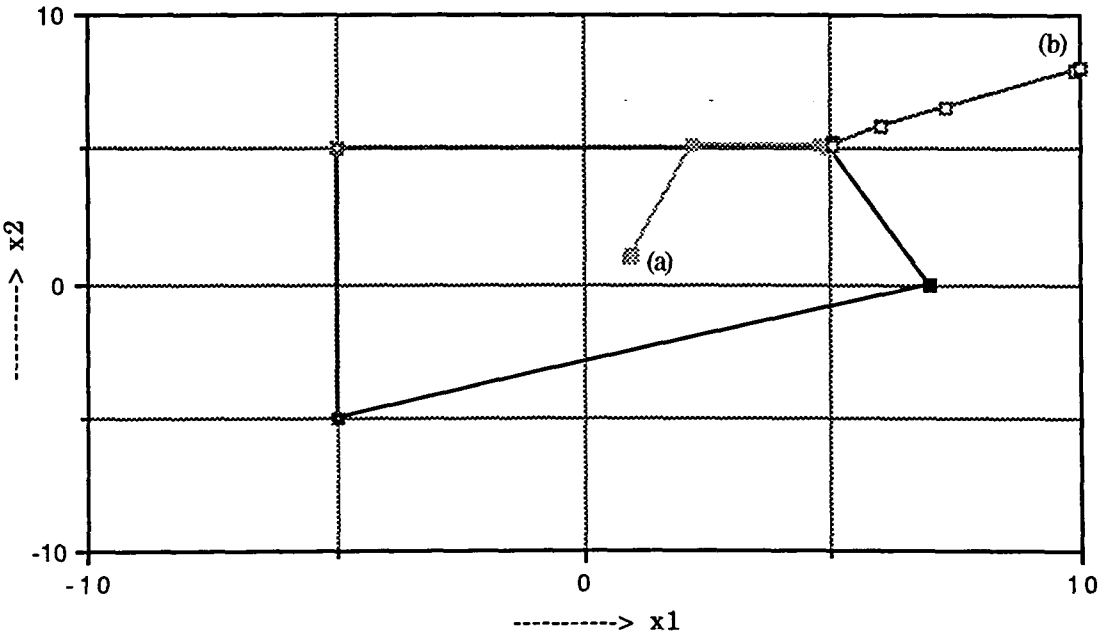

Figure 6.1 Trajectory of network dynamics

The errors represent the percentage difference between the values of respective cost function calculated by different techniques. Note that the error in results also involves the effect of the approximation involved in the simulation technique. The errors in the results of all cases are less than 1.0%. Also, the cpu time is well below 1 %.

6.2 Example 2: A Generic Quadratic Programming Example

A generic quadratic programming of the following form was considered as a test case.

Minimize

$$\phi(x_1,x_2,x_3) = 0.4x_1 + 0.5(5x_1^2 + 8x_2^2 + 4x_3^2) - 3x_1x_2 - 3x_2x_3$$

Subject to:

$$x_1 + x_2 + x_3 \geq 1$$
$$x_1, x_2, x_3 \geq 0$$

The data in matrix form for the quadratic program is shown in Table 6.3.

Table 6.3. Data for a generic quadratic programming

A^T	G	B	e
$\begin{bmatrix} 0.4 \\ 0.0 \\ 0.0 \end{bmatrix}$	$\begin{bmatrix} 2.5 & -3.0 & 0.0 \\ -3.0 & 4.0 & -3.0 \\ 0.0 & -3.0 & 2.0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} -1.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{bmatrix}$

The quadratic program was simulated using the following initial conditions.

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \end{bmatrix}$$

The results of the quadratic program simulation are shown in Table 6.4.

Table 6.4 Results for the generic quadratic programming

Parameter	Theoretical Result	ANN Result
x1	0.2520	0.2484
x2	0.3328	0.3290
x3	0.4150	0.4105
Error		2.07 %
Iteration		25
cpu time		0.2cpus

In this case, the steady state error is about 2%. Reason for this high error is left as a matter of investigation in future work.

6.3 Economic Dispatch Problem

A test case of 3 generator 6-bus system from reference [48] was selected to test the simulation of the economic dispatch algorithm. The data for generator model having quadratic cost curve is given in Table 6.5. In the case of the first example, all units are supposed to run on the active region of their cost curves. Whereas, in the second case, the first unit is constrained to operate at its maximum capacity.

Table 6.5 Generator data (quadratic cost curve)

Case A

Unit	Input-Output Curve (MBtu / h)			P_{min} (MW)	P_{max} (MW)	Fuel Price ($/MBtu)	Fuel Type
	a_i	b_i	c_i				
1	510.0	7.20	0.00142	150	600	1.1	Coal
2	310.0	7.85	0.00194	100	400	1.0	Oil
3	78.0	7.97	0.00482	50	200	1.0	Oil

Case B

Unit	Input-Output Curve (MBtu / h)			P_{min} (MW)	P_{max} (MW)	Fuel Price ($/MBtu)	Fuel Type
	a_i	b_i	c_i				
1	510.0	7.20	0.00142	150	600	0.9	Coal
2	310.0	7.85	0.00194	100	400	1.0	Oil
3	78.0	7.97	0.00482	50	200	1.0	Oil

The equality constraints for the dispatch problem were represented by a coupled set of inequalities. This approach was validated in Chapter 3. Test cases were specifically chosen to test the capability of the algorithm to handle units, either operating in active region of the cost curve or generating power at the minimum or maximum limit.

6.3.1 Economic dispatch without loss modeling

Power transmission losses in the test system was assumed to be negligible. All the three generators were dispatched to satisfy the total demand of 850 MW for both test cases. The simulation was made using the initial condition as follows:

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = \begin{bmatrix} 400 \\ 300 \\ 100 \end{bmatrix}$$

The results for the dispatch using the proposed algorithm and the classical optimization methods are compared in Table 6.6.

Table 6.6 Results for economic dispatch without loss

Example	Unit	Theoretical Result (In MW)	ANN Result (In MW)	Error (In %)
Case A	1	393.16	395.29	0.0913
	2	334.60	331.81	
	3	122.22	122.06	
Case B	1	600.00	600.00	0.0815
	2	187.16	188.08	
	3	62.88	61.26	

Note that infeasible parameters were chosen as the initial condition. This demonstrates that the proposed approach can handle infeasible starting points. The errors represent percentage difference between values of the respective cost function calculated by different techniques. The errors in results of all cases are far below 1.0%. As explained in Chapter 5, the power system data for the economic dispatch problem is known to be in the error of approximately 10.0%. Thus, the solution accuracy is better than the model data.

6.3.2 Economic dispatch with loss modeling using state estimators

For simulation purposes, the data needed from the power system state estimator was created using the B-matrix formulation of the network given in reference [48]. The data is shown in Table 6.7.

The generator data given in Table 6.5 were modified by scaling the incremental cost equation [48] to account for the individual unit's contribution to transmission losses. The load supplied by the units was taken as sum of the

demand load and transmission losses. The state space model of the Kennedy, Chua and Lin neural network was developed for the modified data.

Results for the neural network simulation are given in Table 6.8. Again, the errors represent percentage difference between values of the respective cost functions calculated by different techniques. The errors in the results of all cases are well within the input range.

Table 6.7 Loss representation data from state estimators

Unit	Penalty Factor	Transmission Loss	System Load
1	1.00813	9.85 Mw	850 Mw
2	1.00791		
3	1.00768		

Table 6.8 Results for economic dispatch with loss for the data of Table 6.7

Example	Unit	Theoretical Result (In MW)	ANN Result (In MW)	Error (In %)
Case A	1	397.37	398.91	0.0928
	2	338.50	336.24	
	3	124.01	123.89	
Case B	1	600.00	600.00	0.0785
	2	194.05	195.10	
	3	65.86	64.12	

6.3.3 Economic dispatch with loss modeling using constant penalty factors

Table 6.9 shows hypothetical data for the constant penalty factors. In this case, the transmission losses are not explicitly known. They are a function of the power generated by the individual units.

Table 6.9 Data for penalty factors

Unit	Penalty Factor	Fixed Loss	System Load
1	1.04111	3.0458 Mw	850 Mw
2	1.13175		
3	1.09363		

Table 6.10 Results for economic dispatch with losses using penalty factors

Example	Unit	Theoretical Result (In MW)	ANN Result (In MW)	Error (In %)
Case A	1	397.37	398.88	0.1005
	2	338.50	336.21	
	3	124.01	123.88	
Case B	1	600.00	600.00	0.0785
	2	194.05	195.10	
	3	65.86	64.12	

In this case, the load-power balance equation was modified to include the constant penalty factors. The fixed system loss was added to the demand. The results of the simulation are shown in Table 6.10.

6.3.4. Economic dispatch with piecewise linear cost curve

Quadratic cost curves of the generating units of Table 6.5 were approximated by piecewise linear cost curves. Data after the approximation are shown in Table 6.11.

Table 6.11 Generator data (PWL cost curve)

Example	Unit	Break-Points of Segment 1 (MW, $)	Break-Points of Segment 2 (MW, $)
Case A	1	(150,1784.15) (375, 3750.66)	(375, 3750.66) (600,5875.20)
	2	(100,1114.40) (250, 2393.75)	(250, 2393.75) (400,3760.40)
	3	(50, 488.55) (125, 1149.56)	(125, 1149.56) (200,1864.80)
Case B	1	(150, 1459.80) (375, 3069.00)	(375, 3069.00) (600,4807.20)

The proposed hybrid dispatch algorithm described in the last chapter was used to simulate the economic dispatch. Power transmission losses in the test system were assumed to be negligible. All the three generators were dispatched to satisfy total demand of 850 MW for both the test cases. The results obtained are summarized in Table 6.12

Table 6.12 Results for economic dispatch with PWL cost function

Example	Unit	Theoretical Result [8] for quadratic cost curve (a) (Power In MW)	Theoretical Result for PWL cost curve using MATLAB (b) (Power In MW)	ANN Result for PWL cost curve CSV approach (c) (Power In MW)
Case A	1	393.20	375.00	374.79
	2	334.60	350.00	348.77
	3	122.20	125.00	125.23
Case B	1	600.00	600.00	599.82
	2	187.10	200.00	198.84
	3	62.10	50.00	50.00

Table 6.13 Errors in EDC simulation with PWL cost function

Error due to PWL Approximation [(a) - (b)]/(a)* (In %)	Error in ANN Simulation [(b) - (c)]/(b)* (In %)
0.012%	0.14%
0.015%	0.16%

The errors due to Piecewise linear approximation are far below 0.1%. This verifies that the selected number of linear segments was sufficient enough to represent the nonlinearity of cost curve. Furthermore, the error in ANN simulation is far below 1.0%.

7. CONCLUSION AND SUMMARY

This research explored the possibility of applying an artificial neural network to optimization problems. Linear programming and quadratic programming problems have been simulated using the Kennedy, Chua and Lin network. The simulation is performed using a new methodology called the clamped-state variable approach. Network parameters are modified to compensate for the approximations caused by the proposed approach. The methodology was extended to solve the economic dispatch problem for different generator models and transmission loss representations. A hybrid neural network algorithm was developed to solve the economic dispatch problem with piecewise linear cost curves.

The proposed methodology and algorithms have some very attractive characteristics:

Clamped State Variable approach

- The approach can simulate both generic linear programming and quadratic programming problems.
- The results obtained by the neural network simulation are quite comparable to those obtained by classical optimization techniques.
- The system matrices in the proposed approach are extremely sparse. Hence, the software implementation can be accelerated using sparsity-based techniques.
- The simulation circuit can also be implemented in hardware for real-time applications.

- The architecture of the proposed neural network is highly parallel. Hence, the algorithms can be implemented using parallel processors. This is especially beneficial for high dimension problems.

- The simulation network can begin with feasible as well as infeasible initial conditions.

A number of neural network models can be represented by the state space formulation. Although, this work presents a simulation of a specific network by the proposed approach, the simulation scheme is valid for other similar neural network models. The approach is guaranteed to lead to correct results as long as the simulation step size is kept small to represent the nonlinearity of the system. The choice of simulation step size is a tradeoff between the speed and accuracy. The size of simulation step should be chosen by engineering judgment with due considerations to the system parameters.

Real-Time Economic Dispatch with Neural Network

- The coefficients of the formulated problem is in strict correspondence with the network parameters in the proposed neural network.

- Most of the algorithms used for economic dispatch with transmission losses have double loops. Sometimes, this leads to convergence problems. The proposed formulation does not contain nested loops.

Hybrid ANN algorithm for Economic Dispatch with PWL curve

- The proposed algorithm uses heuristics based on merit order loading. Thus, apriori knowledge is easily incorporated to enhance the speed of algorithm.

- In this work, the number of segments used to approximate the cost curve was limited to two. The choice was sufficient enough to yield the desired solution accuracy. The scheme is valid for any number of segments as long as the generating units are represented by monotonically increasing nonlinear cost curves.

This work basically focused on optimization problems with single objective function. However, the neural network architectures discussed have potential to solve problems with multiple objectives. The same principle of energy minimization can be applied to solve combinatorial optimization problems with neural networks. Many new area might be investigated further.

- Many power system problems can be formulated as LP or QP problems. Thus, they can be simulated by the proposed approach. The optimal power flow would be one example.

- Simulations can also be extended to problems of higher dimension, such as unit commitment, to solve either the EDC subproblem or the overall problem.

Neural networks are viewed as tools to augment the existing algorithms. The proposed work solves generic LP and QP problems, which form the basis of most optimization problems. A proper application of the proposed work would be to solve optimization subproblems in modules.

BIBLIOGRAPHY

1. Casazza, John A. "Free Market Electricity: Potential Impacts on Utility, Pooling and Coordination." Public Utility Fortnightly, February 18, 1988: 16-23.

2. Lamont, J.W., and G.B. Sheble. "Economic Operation and Optimization." Power Seminar, Iowa State University, Ames, Iowa, October 5, 1993.

3. U.S. Congress, Office of Technology Assessment. Electric Power Wheeling and Dealing: Technological Considerations for Increasing Competetion. Washington, DC: U.S. Government Printing Office, May 1989.

4. U.S. Congress. "Notice of Proposed Rulemaking: Docket no. RM88-4, RM88-5, RM88-6." Congressional Record, March 16, 1988.

5 "Focus on Wheeling & Access." Electric Light And Power, March 1992: 12-15.

6. U.S. Congress. House. Summary of HR. 3030 "Clean Air Act Amendments of 1990." Congressional Record, April 20, 1990.

7. U.S. Congress. Senate. Summary of S. 1630 "Clean Air Act Amendments of 1990." Congressional Record, April 20, 1990.

8. Makansi, J. "Clean Air Act Amendments: The Engineering Response." Power, June 1991: 11-56.

9. Engle, R. "New Challenges in The Electric Utility Business: Highlights of The Comprehensive National Energy Policy Act, 1992." Power Seminar, Iowa State University, Ames, Iowa, November 3, 1992.

10. U.S. Congress. "Notice of Proposed Rulemaking: Docket no. RM93-3." Congressional Record, July 30, 1993.

11. Hillier, F.S., and G.J.Liberman. Introduction to Operations Research. New York: McGraw Hill, 1990.

12 Ignizio James P. Linear Programming in Single and Multiple Objective Systems. Englewood Cliffs, New Jersey: Printice Hall Inc., 1982.

13. Karmarkar, N. "A New Polynomial-Time Algorithm for Linear Programming." Combinatorica, 4, 1984: 373-395.

14. Cooper, L, and D. Steinberg. Methods and Applications of Linear Programming. Philadelphia: W.B. Saunders Company, 1974.

15. Bazarra, H.S., Jarvis J.J., and H.D.Sherali. Linear Programs and Network flows. New York: John Wiley and Sons, 1990.

16. Luenberger, D.G. Linear and Nonlinear Programming. Reading, Massachusetts: Addison-Wesley, 1989.

17. Zangwill, W.I. "Nonlinear Programming Via Penalty Functions." Management Science, Vol.13, No. 5, 1967: 344-358.

18. Simmons, D. M. Nonlinear Programming for Operations Research. Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1975.

19. Khanna, T. Foundations of Neural Networks Reading, Massachusetts: Addison-Wesley, 1990.

20. Hecht-Nelson, R. Neurocomputing. Reading, Massachusetts: Addison-Wesley, 1991.

21. Lippmann, R.P. "An Introduction to Computing with Neural Nets." IEEE Acoustics, Speech and Signal Processing Magazine, April 1987: 4-21.

22. Hush, D.R. and B.G. Horne. "Progress in Supervised Neural Networks: What's New Since Lippmann?" IEEE Acoustics, Speech and Signal Processing Magazine, January 1993: 8-38.

23. Barnell, E. "Optimization for Training Neural Nets." IEEE Transactions on Neural Networks, Vol.3, No.2, March 1992: 232-240.

24. Tagliarini, G.A., Christ, J.F., and E.W. Page. "Optimization Using Neural Networks." IEEE Transactions on Computers, Vol.40, No.12, December 1991: 1347-1358.

25. Herault, L. and J. Naz. "Neural Networks and Combinatorial Optimization: A Study of NP-complete Graph Problems." Neural Networks: Advances and Application, B.V., North Holland: Science Publishers, 1991:165-212.

26 Kosko, B. Neural Networks and Fuzzy Systems: A dynamical Systems Approach. Englewood Cliffs, New Jersey: Simon and Schuster, 1991.

27. Rodriguez, A. "Nonlinear Switched-Capacitor Neural Networks for Optimization Problems." IEEE Transactions on Circuit and Systems, Vol.37, March 1990: 384-398.

28. Hopfield J.J. "Neural Networks and Physical Systems with Emergent Collective Computations Abilities." Proceedings of the National Academy of Science, USA, Vol.79, 1982: 2554-2558.

29. Hopfield J.J. "Neurons with Graded Response Have Collective Computational Properties Like Those of 2-State Neurons." Proceedings of the National Academy of Science, USA, Vol.81, May 1984: 3088-3092.

30. Hopfield, J.J. and D.W. Tank, "Neural Computation of Decision Optimization Problems." Biological Cybernetics, Vol.52, 1985: 141-152.

31. Hopfield, J.J. and D.W. Tank, "Simple Neural Optimization Networks: An A/D Converter, Signal Decision Network, and A Linear Programming Circuit." IEEE Transactions on Circuit and Systems, CAS-33, May 1986: 533-541.

32. Chua, L.O. and G.N. Lin. "Nonlinear Programming without Computation." IEEE Transactions on Circuit and Systems, CAS - 31, February 1984: 182-188.

33. Kennedy, M.P. and L.O. Chua. "Unifying The Tank and Hopfield Linear Programming Circuit and The Canonical Nonlinear Circuit of Chua and Lin." IEEE Transactions on Circuit and Systems, CAS - 34, February 1987: 210-214.

34. Kennedy, M.P. and L.O. Chua. "Neural Networks for Nonlinear Programming" IEEE Transactions on Circuit and Systems, CAS - 35, May 1988: 554-562.

35. Zhang, S., and A.K. Constantinidies, "Lagrange Programming Neural Network." IEEE Transactions on Circuit and Systems, CAS - 39, May 1992: 441-452.

36. Sendula, M.H., Biswas, S.K., Elton, A., Parter, C. and W. Kazibwe. "Application of Artificial Neural Networks to Unit Commitment." Proceedings of International Forum on Application of Artificial Neural Networks to Power System, Seattle, Washington, July 1991: 256-260.

37. Sendula, M.H., Biswas, S.K., Elton, A., Parter, C. and W. Kazibwe. "Simultaneous Solution of Unit Commitment and Dispatch Problems Using Artificial Neural Networks." Electrical Power and Energy Systems, Vol. 15, 1993: 193-199.

38. Sasaki, H., Watanabe M., Kubokawa J., Yorino N., and B. Yokoyama, "A Solution Method of Unit Commitment by Artificial Neural Network." presented at the 1991 IEEE Power Engineering Society Summer Meeting, San Diego.

39. Park, J.H., Kim, Y.S., Eom, I.K., and K,Y, Lee. "Economic Load Dispatch for Piecewise Quadratic Cost Function Using Hopfield Neural Network." Presented at the 1992 IEEE Power Engineering Society Summer Meeting, Seattle.

40. Fukuyama, Y. and Y.Ueki, "An Application of Artificial Neural Network to Dynamic Load Dispatching." Proceedings of International Forum on Application of Artificial Neural networks to Power System, Seattle, Washington, July 1991: 261-265.

41. Matuda, S. and Y. Akimito. "The Representation of Large Numbers in Neural Networks and Its Application to Economical Load Dispatching of Electric Power." ICNN, Vol.1, June 1989: 587-592.

42 Maa, C.Y. and M.A. Shanblatt, "Linear and Quadratic Programming Neural Network Analysis." IEEE Transactions on Neural Networks, Vol. 3, July 1992: 580-594.

43. Maa, C.Y. and M.A. Shanblatt, "Improved Neural Networks for Linear and Nonlinear Programming." Submitted to International Journal of Neural Systems.

44. Grisby, L. EE 551,Class Notes, Auburn University, Auburn, 1986.

45. Tou, J.T., Modern Control Theory. New York: McGraw Hill, 1964.

46. Chen, C. Linear System Theory and Design. New York: Holt Renihart and Winsten Inc., 1984.

47. Watkins, D.S. Fundamentals of Matrix Computations. New York: John Wiley and Sons, 1991.

48. Wood, A. J., and B. F. Wollenberg. Power Generation, Operation and Control. New York: John Wiley and Sons, Inc., 1984.

49. Kirchmayer, L. K., and G. W. Stagg. "Analysis of Total and Incremental Losses in Transmission Systems." AIEE Transactions, Vol. 70, 1951: 1197-1205.

50. Kirchmayer, L. K., Happ H. H., Stagg G. W., and J. F. Hohenstein. "Direct Calculation of Transmission Loss Formula." AIEE Transactions, Vol. 79, 1960: 962-969.

51. Kron, G. "Tensorial Analysis of Integrated Transmission Systems - Part I: The Six Basic Reference Frames." AIEE Transactions, Vol. 70, 1951: 1239-1248.

52. Waight, J.G., Bose, A., and G.B. Sheble'. "Generation Dispatch with Reserve Margin Constraints Using Linear Programming." IEEE Transaction on Power Apparatus and Systems, Vol. 100, January 1981: 252-258.

53. Brodsky, S. F., and R. W. Hahn. "Assessing the Influence of Power Pools on Emission Constrained Economic Dispatch." IEEE Transactions on Power Apparatus and Systems, Vol. 1, 1986: 57-62.

54. Chowdhury, B. H., and S. Rahman. "A Review of Recent Advances in Economic Dispatch." IEEE Transactions on Power Systems, Vol. 5, 1990: 1248-1257.

55. Sheble', G.B. "Real-Time Economic Dispatch and Reserve Allocation Using Merit Order Loading And Linear Programming Rules." IEEE Transactions on Power Systems, Vol. 4, 1990: 1414-1420.

ACKNOWLEDGMENTS

I am deeply indebted to my major professor, Dr. Gerald B. Sheble' for his guidance, support and encouragement during this research. Comments and suggestions from Dr. John Lamont, Dr. K. Jo Min and Dr. Jenifer Davidson, additional members of my committee, are also very much appreciated. I am also thankful to the director of electric power research center (EPRC), Dr. John Lamont, for providing the financial support during my graduate study at Iowa State University.