

10-1-1991

ON SOLVING CONSTRAINED OPTIMIZATION PROBLEMS WITH NEURAL NETWORKS : A PENALTY FUNCTION METHOD APPROACH

Walter E. Lillo

Purdue University School of Electrical Engineering

Mei Heng Loh

Purdue University School of Electrical Engineering

Stefen Hui

San Diego State University, Department of Mathematical Sciences

Stanislaw H. Zak

Purdue University School of Electrical Engineering

Follow this and additional works at: <http://docs.lib.purdue.edu/ecetr>

Lillo, Walter E.; Loh, Mei Heng; Hui, Stefen; and Zak, Stanislaw H., "ON SOLVING CONSTRAINED OPTIMIZATION PROBLEMS WITH NEURAL NETWORKS : A PENALTY FUNCTION METHOD APPROACH" (1991). *ECE Technical Reports*. Paper 321.

<http://docs.lib.purdue.edu/ecetr/321>



On Solving Constrained Optimization Problems with Neural Networks: A Penalty Function Method Approach

Walter E. Lillo
Mei Heng Loh
Stefen Hui
Stanislaw H. Żak

TR-EE 91-43
October 1991

ON SOLVING CONSTRAINED OPTIMIZATION
PROBLEMS WITH NEURAL NETWORKS :
A PENALTY FUNCTION METHOD APPROACH

Walter E. Lillo
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

Mei Heng Loh
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

Stefen Hui
Department of Mathematical Sciences
San Diego State University
San Diego, CA 92182

Stanislaw H. Żak
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

Abstract

This paper is concerned with utilizing analog circuits to solve various linear and nonlinear programming problems. The dynamics of these circuits are analyzed. Then, the previously proposed circuit implementations for solving optimization problems are examined. A new nonlinear programming network and its circuit implementation is then introduced which utilizes the nonlinearities to eliminate the problems encountered in previous circuit implementations.

Key Words:

Nonlinear programming, Stability, Optimization, Circuit, implementation.

1. Introduction

Recently there has been much interest in constructing analog circuits to simulate mathematical programming problems. This concept was first proposed by Dennis ([1]), and later studied by Stern ([2]). More recently Chua and Lin ([3]) presented a general form of an analog circuit to solve nonlinear programming problems. They referred to their circuit as the canonical nonlinear programming circuit. Other canonical circuits were later introduced by Huertas, Rueda, Rodriguez-Vazquez and Chua ([4]). In [5], Chua and Lin presented a circuit implementation which is capable of solving various programming problems without numerical computation. Later, Wilson ([6]) stressed the importance of negative components in quadratic programming and proposed an implementation which utilized floating negative resistors. Tank and Hopfield ([7]) introduced a linear programming neural network which had both

inverting and noninverting outputs for each node and thus eliminating the need for the negative resistors. In addition, they introduced capacitors to allow for the modeling of the circuit dynamically. Then they showed that the state of the network evolved in such a way that the energy function of the network is monotonically nonincreasing with time. Kennedy and Chua ([8]) showed that the Tank and Hopfield linear programming network is a specific implementation of the canonical nonlinear programming circuit of Chua and Lin ([5]) with a capacitor added to account for the dynamic behavior of the circuit. Kennedy and Chua ([9]) examined the stability of the modified canonical nonlinear programming circuit and proposed a neural network circuit implementation which could be utilized to solve a class of nonlinear programming problems. The above discussed methods for solving constrained optimization problems with neural networks implicitly utilize the penalty method. It is known (Luenberger [10, Chapter 12]) that if the weight assigned to the penalty function approaches infinity then the global minimizer of the transformed problem tends to the global minimizer of the original problem. However, when it comes to a practical circuit implementation, the value which can be assigned to the weight of the penalty function is limited by physical constraints. The effect of these physical constraints may result in the convergence of the circuit trajectory to a nonfeasible state, as shown in this paper. To remedy the above mentioned difficulty, a new neural network implementation for solving constrained optimization problems is proposed. This new architecture uses the penalty function to force the circuit trajectory into the feasible region before optimizing the objective function.

This paper is divided into six sections. In the next section some background results are presented. The third section contains the stability analysis of the dynamical canonical circuit using the second method of Lyapunov. The fourth section discusses some of the implementation issues and introduces a new nonlinear programming network which is implemented utilizing common circuit elements. The fifth section gives a circuit implementation of the

proposed neural network for the case of quadratic programming problems, and presents results of several test problems. Conclusions are found in section six.

2. Problem Statement and Background Results

In this paper we are concerned with optimization problems of the following form

$$\begin{aligned} & \text{minimize } f(\mathbf{x}) \\ & \text{subject to } \mathbf{g}(\mathbf{x}) \geq \mathbf{0} \end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^n$, $f: \mathbb{R}^n \rightarrow \mathbb{R}$, and $\mathbf{g} = [\mathbf{g}_1, \dots, \mathbf{g}_p]^T: \mathbb{R}^n \rightarrow \mathbb{R}^p$ is a p -dimensional vector valued function of n variables. It is also assumed that f and \mathbf{g} are continuously differentiable functions. Utilizing the penalty method, a constrained optimization problem of the above form can be approximated by an unconstrained optimization problem. The resulting unconstrained problem has the form:

$$\text{minimize } E(\mathbf{x}) = \text{minimize } (f(\mathbf{x}) + cP(\mathbf{x})),$$

where c is a positive constant, sometimes referred to as a weight, and $P(\mathbf{x})$ is called a penalty function. A penalty function is a continuous nonnegative function which is zero at a point if and only if all the constraints are satisfied at that point.

One of the main results connecting the minimizers of the constrained and unconstrained problems is:

Theorem ([10], p. 368).

Let $\{\mathbf{x}_k\}$ be a sequence of minimizers generated by the penalty method for an increasing sequence of c_k 's tending to infinity. Then any limit of $\{\mathbf{x}_k\}$ is a minimizer of the constrained problem.

In this paper we consider penalty functions of the following form:

$$P_q(\mathbf{x}) = \frac{1}{q} \sum_{j=1}^p [g_j^-(\mathbf{x})]^q$$

where $g_j^-(\mathbf{x}) = -\min(\mathbf{0}, g_j(\mathbf{x}))$ and $q > 0$. The penalty function $P_1(\mathbf{x})$ is often referred to as an exact penalty function ([10]). The exact penalty function has the drawback that it is nondifferentiable on the border of the feasible region. This difficulty can be overcome by using a penalty function with $q > 1$ (see [10, pp. 372,373] for more details).

To simulate the constrained optimization problem using the penalty method with $P_2(\mathbf{x})$ one can employ the so called dynamical canonical nonlinear programming circuit proposed by Kennedy and Chua ([9]) - see Fig. 1. Kennedy and Chua ([9]) utilized their canonical circuit to simulate the Kuhn-Tucker conditions. In this paper we show that one can use their canonical circuit in the context of the penalty method.

The functions ϕ_j ($j=1, \dots, p$) in Fig. 1 relate the voltages across nonlinear resistors to the currents through them.

In a manner similar to [4], we can introduce equivalent alternative circuits. The conditions represented by the circuits on the left side of Fig. 1 can also be modeled using dependent current sources representing the constraint functions and nonlinear resistors, where the voltages across the nonlinear resistors correspond to the variables μ_j ($j = 1, \dots, p$). Likewise, the condition corresponding to the circuits on the right side of Fig. 1 can also be simulated by two dependent voltage sources in series with an inductor.

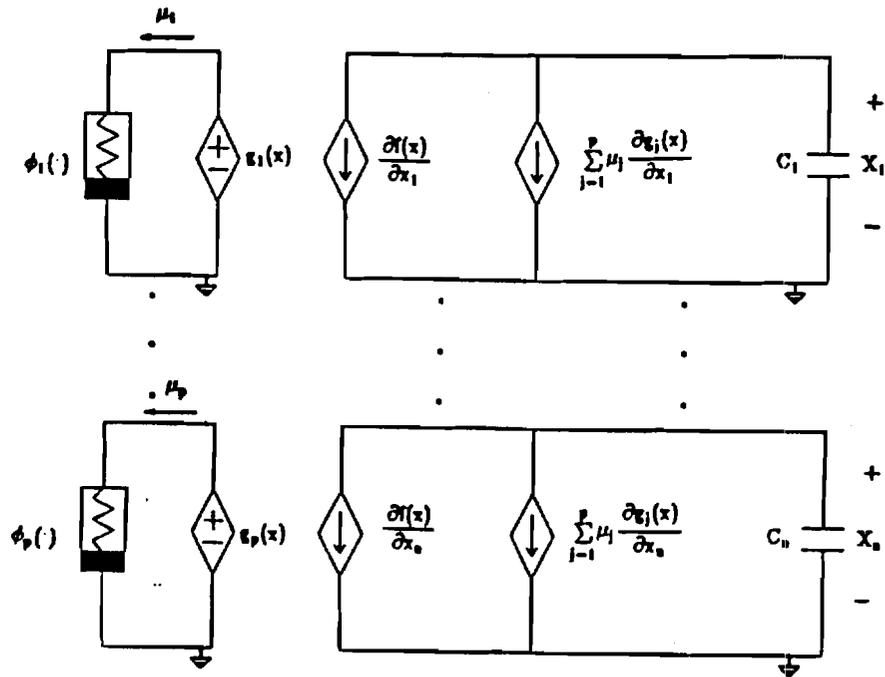


Figure 1. Dynamical canonical nonlinear programming circuit of Kennedy and Chua ([9]).

For the remainder of this section, we only consider the dynamical canonical nonlinear programming circuit shown in Fig. 2. The analysis of other canonical circuits is analogous.

The above circuit (Fig. 2) can be split into two basic parts.. The circuits on the left are used to generate the penalty function for the given constraints. At steady state, the circuits on the right correspond to the first order necessary conditions for a local minimizer of the unconstrained objective function

$$E(\mathbf{x}) = f(\mathbf{x}) + cP_2(\mathbf{x}) = f(\mathbf{x}) + \frac{c}{2} \sum_{j=1}^p (g_j^-(\mathbf{x}))^2 .$$

Observe that even though $g_j^-(\mathbf{x})$ is not differentiable at the boundary points, $[g_j^-(\mathbf{x})]^2$ is differentiable there. The components of the gradient of E have the form

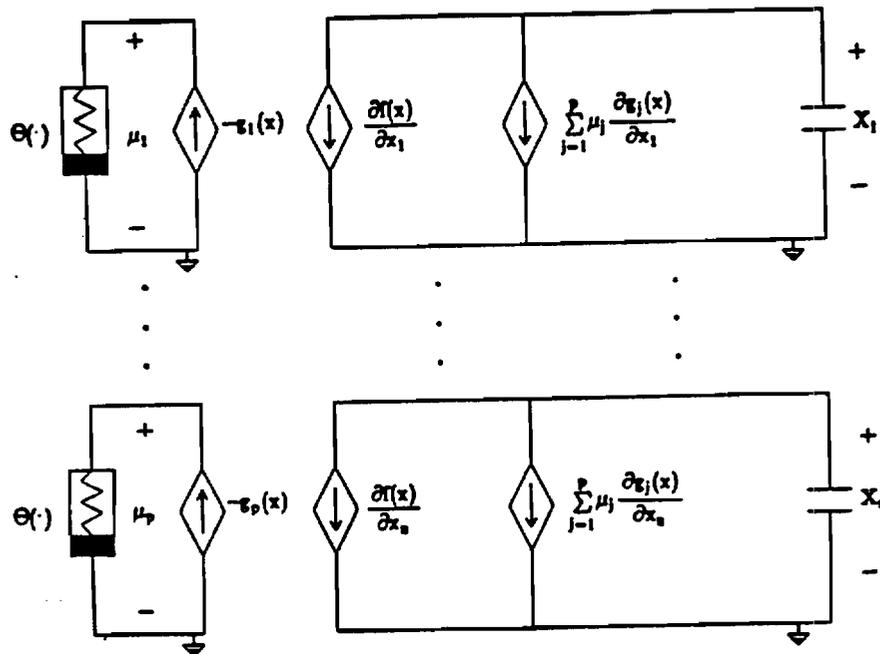


Figure 2. An alternative version of the dynamical canonical nonlinear programming circuit.

$$\frac{\partial E(x)}{\partial x_k} = \frac{\partial f(x)}{\partial x_k} + c \frac{\partial P_2(x)}{\partial x_k} = \frac{\partial f(x)}{\partial x_k} - c \sum_{j=1}^p (g_j^-(x)) \frac{\partial g_j(x)}{\partial x_k}, \quad k=1, \dots, n.$$

We will now see how the circuits on the left of Fig. 2 are used to generate the components of the gradient of the penalty term needed for the circuits on the right. The function Θ , in Fig. 2, relates the current through a nonlinear resistor to the voltage across it, and is defined by

$$V = \Theta(I) = \begin{cases} -cI & \text{if } I > 0 \\ 0 & \text{if } I \leq 0 \end{cases}$$

where $c > 0$ is the weight of the penalty function. Thus from the circuit depicted in Fig. 2, we have

$$\mu_j = \Theta(-g_j(\mathbf{x})) = \begin{cases} cg_j(\mathbf{x}) & \text{if } g_j(\mathbf{x}) < 0 \\ 0 & \text{if } g_j(\mathbf{x}) \geq 0 \end{cases}, \quad j=1, \dots, p.$$

Clearly

$$\mu_j g_j(\mathbf{x}) = \Theta(-g_j(\mathbf{x}))g_j(\mathbf{x}) = \begin{cases} cg_j^2(\mathbf{x}) & \text{if } g_j(\mathbf{x}) < 0 \\ 0 & \text{if } g_j(\mathbf{x}) \geq 0 \end{cases}, \quad j=1, \dots, p.$$

Thus

$$\frac{1}{2} \sum_{j=1}^p \mu_j g_j(\mathbf{x}) = \frac{c}{2} \sum_{j=1}^p (g_j^-(\mathbf{x}))^2 = cP_2(\mathbf{x})$$

and for $k = 1, \dots, n$,

$$\frac{\partial}{\partial \mathbf{x}_k} \left(\frac{1}{2} \sum_{j=1}^p \mu_j g_j(\mathbf{x}) \right) = \frac{\partial}{\partial \mathbf{x}_k} \left(\frac{c}{2} \sum_{j=1}^p g_j^2(\mathbf{x}) \right) = -c \sum_{j=1}^p (g_j^-(\mathbf{x})) \frac{\partial g_j(\mathbf{x})}{\partial \mathbf{x}_k} = c \frac{\partial P_2(\mathbf{x})}{\partial \mathbf{x}_k} = \sum_{j=1}^p \mu_j \frac{\partial g_j(\mathbf{x})}{\partial \mathbf{x}_k}.$$

Kirchhoff's current law applied to the right side of Fig. 2 yields

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_k} + \sum_{j=1}^p \mu_j \frac{\partial g_j(\mathbf{x})}{\partial \mathbf{x}_k} + C_k \frac{d\mathbf{x}_k}{dt} = 0, \quad k = 1, \dots, n.$$

The above equations can be rewritten as

$$\frac{d\mathbf{x}_k}{dt} = -\frac{1}{C_k} \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_k} + \sum_{j=1}^p \mu_j \frac{\partial g_j(\mathbf{x})}{\partial \mathbf{x}_k} \right),$$

or

$$\frac{dx_k}{dt} = -\frac{1}{C_k} \frac{\partial E(\mathbf{x})}{\partial x_k}, \text{ for } k=1, \dots, n.$$

Thus any equilibrium point of the circuit equations will correspond to a critical point of the function $E(\mathbf{x})$.

Having noted all this, there remains a number of questions which must be answered:

1. Will the circuit trajectory converge to a stable equilibrium?
2. Will an equilibrium point correspond to a maximum, minimum, or a saddle point?
3. How well do the solutions of the unconstrained problem approximate the solutions of the constrained problem?

In order to answer these questions we must first analyze the stability properties of the circuit in Fig. 2.

Remark

The unconstrained optimization function $E(\mathbf{x})$ can be rewritten as follows

$$E(\mathbf{x}) = f(\mathbf{x}) + \sum_{j=1}^p \int_0^{g_j(\mathbf{x})} \Theta(w) dw .$$

The above expression corresponds to the energy function of the circuits in Fig. 1 and 2. This energy function was analyzed by Kennedy and Chua ([9]) in connection with the circuit depicted in Fig. 1.

3. Stability Analysis of Programming Circuits Via the Lyapunov Second Method

In [7],[9], and [13] it was shown that the function $E = E(x(t))$ is a monotonically decreasing function of time. It was this condition combined with the assumption that E was bounded from below that led the authors of [7],[9], and [13] to conclude that the circuit trajectory would reach a steady state. While their conclusion was correct, it is worth mentioning that $\lim_{t \rightarrow \infty} \frac{dE}{dt}$ may not exist. As an example of the above scenario consider the following function:

$$E(t) = \sum_{n=1}^{\infty} \frac{1}{n^2} \arctan(n^3(n-t)) .$$

Notice that this function is continuous and bounded from below. However, $\lim_{t \rightarrow +\infty} \frac{dE}{dt}$ does not exist.

Having pointed this out we will now examine the stability of the circuits in Fig. 1 and 2.

From the discussion before Remark 1, it follows that there is a one-to-one correspondence between the critical points of the function E and the equilibrium points of the circuit. What remains to be examined is whether any of the critical points are stable, and if so, which ones. Before we proceed with the stability analysis we will first review a few basic definitions and theorems pertaining to the stability of dynamical systems. For a more detailed discussion of stability analysis one may consult [11].

The models for our circuits can be represented in the following general form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) , \quad \mathbf{x}_0 = \mathbf{x}(t_0) ,$$

where $\mathbf{x} = \mathbf{x}(t) \in \mathbb{R}^n$ and \mathbf{f} is a vector valued function having components

$$f_i(x_1, x_2, \dots, x_n) , \quad i = 1, \dots, n.$$

It is assumed that each f_i is continuous and has continuous first partial derivatives. For further discussion we need the following definitions.

Definition 1 (Stability in the sense of Lyapunov)

Let $\mathbf{x}(t)$ be a solution of $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$. An isolated equilibrium point \mathbf{x}^* , that is $\mathbf{f}(\mathbf{x}^*) = 0$, is said to be stable in the sense of Lyapunov if for any $\mathbf{x}_0 = \mathbf{x}(t_0)$ and any scalar $\epsilon > 0$ there exists a $\delta > 0$ so that

$$\| \mathbf{x}(t_0) - \mathbf{x}^* \| < \delta \text{ implies } \| \mathbf{x}(t) - \mathbf{x}^* \| < \epsilon \quad \forall t \geq t_0 ,$$

where $\| \mathbf{x} \|$ is a norm of the vector \mathbf{x} . In our discussion we use the Euclidean norm.

Definition 2 (Asymptotic stability)

An isolated equilibrium point \mathbf{x}^* is said to be asymptotically stable if in addition to being stable in the sense of Lyapunov it has the property that $\mathbf{x}(t) \rightarrow \mathbf{x}^*$ as $t \rightarrow \infty$, if $\| \mathbf{x}(t_0) - \mathbf{x}^* \| < \delta$.

Definition 3 (Positive definite function)

Let $\Omega \subseteq \mathbf{R}^n$ be an open neighborhood of $\mathbf{x} = \mathbf{0}$. A continuously differentiable function $V: \mathbf{R}^n \rightarrow \mathbf{R}$ is said to be positive definite over the set $\Omega \subset \mathbf{R}^n$ if $V(\mathbf{x}) > 0 \quad \forall \mathbf{x} \in \Omega, \mathbf{x} \neq \mathbf{0}$, and $V(\mathbf{0}) = 0$.

In the following theorems we assume, without loss of generality, that the equilibrium point of interest has been translated to the origin of \mathbf{R}^n .

Theorem (Lyapunov Stability Theorem)

An isolated equilibrium point $\mathbf{x}^* = \mathbf{0}$ is stable in the sense of Lyapunov if for some neighborhood Ω about \mathbf{x}^* , there exists a positive definite function $V(\mathbf{x})$ such that the function $\dot{V}(\mathbf{x}) = \langle \nabla V(\mathbf{x}), \dot{\mathbf{x}} \rangle$ satisfies the inequality

$$\dot{V}(\mathbf{x}) \leq 0 \quad \forall \mathbf{x} \in \Omega .$$

Here $\langle \mathbf{u}, \mathbf{v} \rangle$ denotes the dot product of the vectors $\mathbf{u}, \mathbf{v} \in \mathbf{R}^n$:

$$\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{j=1}^n u_j v_j .$$

Theorem (Asymptotic Stability Theorem)

An isolated equilibrium point $\mathbf{x}^* = \mathbf{0}$ is asymptotically stable if for some neighborhood Ω about \mathbf{x}^* , there exists a positive definite function $V(\mathbf{x})$ so that the function $\dot{V}(\mathbf{x}) = \langle \nabla V(\mathbf{x}), \dot{\mathbf{x}} \rangle$ satisfies

$$\dot{V}(\mathbf{x}) < 0 \quad \forall \mathbf{x} \in \Omega, \mathbf{x} \neq \mathbf{0}$$

and

$$\dot{V}(\mathbf{0}) = \mathbf{0} .$$

Theorem (Instability Theorem)

An isolated equilibrium point $x^* = \mathbf{0}$ is unstable if there exists a function $V(\mathbf{x})$ with continuous partials such that for some neighborhood Ω about x^* , the following conditions hold:

- a) $V(\mathbf{0}) = 0$.
- b) In each neighborhood $N \subset \Omega$ about the point $x^* = \mathbf{0}$, there exists a point \mathbf{z} such that $V(\mathbf{z}) > 0$.
- c) $\dot{V}(\mathbf{x}) > 0 \quad \forall \mathbf{x} \in \Omega \setminus \{\mathbf{0}\}$.

Having briefly reviewed the stability of dynamical systems, we now examine the stability of the nonlinear programming circuits. We had previously shown that the dynamics of the circuits are described by the equations:

$$\frac{dx_k}{dt} = -\frac{1}{C_k} \left(\frac{\partial f(\mathbf{x})}{\partial x_k} + \sum_{j=1}^p \Theta(-g_j(\mathbf{x})) \frac{\partial g_j(\mathbf{x})}{\partial x_k} \right) , \quad k = 1, \dots, n.$$

Rewriting the above equations using vector notation we have

$$\frac{d\mathbf{x}}{dt} = -\mathbf{C}\nabla E(\mathbf{x}) ,$$

where

$$C = \begin{bmatrix} \frac{1}{C_1} & 0 & & & 0 \\ 0 & \frac{1}{C_2} & & & \\ \cdot & 0 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ 0 & 0 & & 0 & \frac{1}{C_n} & \end{bmatrix}$$

Note that $C = C^T > 0$.

Systems of this form are referred to as gradient systems ([11] and [12]). These systems have been shown to have some nice properties, some of which we use in our analysis.

Theorem 1

An isolated equilibrium point of the circuit equations is asymptotically stable if and only if it is a strict local minimizer of the function E .

Proof

Let \mathbf{x}^* be a strict local minimizer of E . Without loss of generality we assume that the equilibrium point is $\mathbf{x}^* = \mathbf{0}$. Let us define the Lyapunov function candidate $V(\mathbf{x}) = E(\mathbf{x}) - E(\mathbf{0})$. Note that since \mathbf{x}^* is a strict local minimizer of the function $E(\mathbf{x})$, and $V(\mathbf{0}) = 0$, $V(\mathbf{x})$ is clearly positive definite in some neighborhood Ω about the point \mathbf{x}^* . In addition, $V(\mathbf{x})$ has continuous first partial derivatives. We have

$$\dot{V}(\mathbf{x}) = \langle \nabla V(\mathbf{x}), \dot{\mathbf{x}} \rangle = \langle \nabla E(\mathbf{x}), -C \nabla E(\mathbf{x}) \rangle = -\nabla E(\mathbf{x})^T C \nabla E(\mathbf{x}) < 0$$

since $C = C^T > 0$. The result follows from the Asymptotic Stability Theorem.

Conversely, suppose that x^* is an asymptotically stable equilibrium point of the circuit equations. Then there exists $\delta > 0$ so that if $\|x^* - x(0)\| < \delta$ we have $x(t) \rightarrow x^*$ as $t \rightarrow \infty$. Suppose $\|x_0 - x^*\| < \delta$ and $x_0 \neq x^*$. Let $x(0) = x_0$. Then we have

$$\begin{aligned} E(x^*) - E(x(0)) &= \int_{x(0)}^{x^*} \frac{dE}{dt} dt \\ &= \int_0^\infty \sum_{k=1}^n \frac{\partial E}{\partial x_k} \frac{dx_k}{dt} dt \\ &= - \int_0^\infty \sum_{k=1}^n \frac{1}{C_k} \left(\frac{dx_k}{dt} \right)^2 dt \\ &< 0 \end{aligned}$$

since the trajectory moves from $x(0)$ to $x^* = x(\infty) \neq x(0)$. Therefore, x^* is a strict local minimizer of the function E on $\|x - x^*\| < \delta$.

□

Theorem 2

If an isolated equilibrium point of the circuit is stable, then it is a local minimizer of the function E .

P'roof

We use contraposition. Without loss of generality, we assume $x^* = 0$. Let us introduce the function $V(x) = E(0) - E(x)$. Note that $V(0) = 0$, and V has continuous partial derivatives. Since the equilibrium point $x^* = 0$ is not a local minimizer it is clear that in each neighborhood N about $x^* = 0$ there exists $z \in N$ so that $E(z) < E(0)$. This implies in every neighborhood of $x^* = 0$ there exists a z in the neighborhood so that $V(z) > 0$ in the neighborhood of $x^* = 0$.

Furthermore, we have

$$\dot{V}(\mathbf{x}) = \langle \nabla V(\mathbf{x}), \dot{\mathbf{x}} \rangle = \nabla E(\mathbf{x})^T C \nabla E(\mathbf{x}) > 0 \quad .$$

This is due to the fact that \mathbf{x}^* is an isolated equilibrium point. Thus there exists a neighborhood Ω of $\mathbf{0}$ such that if $\mathbf{x} \in \Omega$, $\mathbf{x} \neq \mathbf{0}$, then $\nabla E(\mathbf{x}) \neq \mathbf{0}$. The result then follows from the Instability Theorem.

□

Corollary 1

An isolated equilibrium point of the circuit corresponding to a local maximizer of the function E is unstable.

The next question to ask is whether the minimizers of E correspond to feasible, locally optimal solutions to the original constrained minimization problem. To answer this question let us examine the function E which is being minimized. Recall that the circuit equations are given by

$$\frac{dx_k}{dt} = -\frac{1}{C_k} \left(\frac{\partial f(\mathbf{x})}{\partial x_k} + \sum_{j \in J} \mu_j \frac{\partial g_j(\mathbf{x})}{\partial x_k} \right) \quad , \quad k = 1, \dots, n,$$

where J is the index set of violated constraints. If we let $c \rightarrow \infty$, then $\mu_j \rightarrow -\infty$ if $g_j(\mathbf{x}) < 0$, and $\mu_j = 0$ if $g_j(\mathbf{x}) \geq 0$. Thus it is clear that the circuit works first to find a critical point of the penalty function. This, however, does not guarantee that the circuit trajectory will converge to a feasible point. It merely states that the circuit will be primarily concerned with finding a critical point of the penalty function. Only then will it concern itself with the objective function f . In the event that the penalty function is convex we expect that the circuit will be driven to a feasible critical point of the constrained optimization problem.

4. Implementation Issues

Until now we have approached the problem in a very idealistic fashion. To this point we have not considered any circuit limitations such as op-amp saturations and the corresponding nonlinearities. The variables x_1, \dots, x_n and μ_1, \dots, μ_n correspond to the outputs of the op-amps. Therefore, they must be within the saturation limits of the op-amps. Thus the state of the circuit is constrained to be within the hypercube defined by

$$x_{j_{\min}} \leq x \leq x_{j_{\max}} \quad \text{for } j = 1, \dots, n .$$

In addition, the weight c for the penalty function cannot be arbitrarily large in the circuit implementation due to physical limitations of the circuit components. This limitation on c degrades the approximation of the constrained problem by the unconstrained problem. The difficulties which arise due to the solution space being constrained to be in a fixed hypercube can be overcome somewhat by scaling. This is accomplished by scaling the original problem as follows:

$$\text{minimize } f(\mathbf{ax})$$

subject to

$$g_j(\mathbf{ax}) \geq 0, \quad j = 1, \dots, p,$$

where the scalar $a > 0$ is sufficiently large to ensure that the solution space lies inside the hypercube constraining the problem.

We now examine the type of difficulties which arise from the variables μ_j ($j=1, \dots, p$) being constrained. Recall that in the case examined earlier, $\mu_j = 0$ if $g_j(\mathbf{x}) > 0$, and $\mu_j \rightarrow -\infty$ as $c \rightarrow \infty$ if $g_j(\mathbf{x}) < 0$. As a result the

system first seeks to minimize the penalty function, and then if it reaches the feasible region, it would then search for a local minimizer of f in the feasible region. However, when the μ_j 's are bounded we find that the situation is quite different.

As pointed out earlier we would like to have the weight c be as large as possible in order to best approximate the unconstrained problem. However, if c is large, the circuit component producing μ_j may saturate. When this happens, the outputs no longer represent the true p_j 's. If the saturation limit is $b < 0$, then the output $\tilde{\mu}_j$ is a smooth approximation of the function $b\chi_j$, where χ_j is the characteristic function of $\{g_j(x) < 0\}$ defined by

$$\chi_j(\mathbf{x}) = \begin{cases} 0 & \text{if } g_j(\mathbf{x}) \geq 0 \\ 1 & \text{if } g_j(\mathbf{x}) < 0 \end{cases} .$$

If we use $\tilde{\mu}_j$ in place of μ_j and the component is saturated, then in the region where the constraints are violated we have

$$g_j(\mathbf{x})c = \mu_j = \tilde{\mu}_j = b \ .$$

One can see that the value of c is not well defined, since it varies depending on the value of the constraint $g_j(\mathbf{x})$. Thus it would seem intractable to analyze the circuit implementation utilizing the penalty method with $P_2(\mathbf{x})$ as the penalty function. In order to see which model would be more appropriate to use we will examine the dynamics of the previously proposed network ([9]) keeping in mind the threshold nature of the variables $\tilde{\mu}_j$. Recall that,

$$\frac{dx_k}{dt} = -\frac{1}{C_k} \left(\frac{\partial f(\mathbf{x})}{\partial x_k} + \sum_{j \in J} \tilde{\mu}_j \frac{\partial g_j(\mathbf{x})}{\partial x_k} \right), \quad k=1, \dots, n.$$

When c is large, we can assume for practical purposes that $\tilde{\mu}_j = b\chi_j$. The above equation becomes

$$\frac{dx_k}{dt} = -\frac{1}{C_k} \left(\frac{\partial f(\mathbf{x})}{\partial x_k} + \sum_{j=1}^p b\chi_j \frac{\partial g_j(\mathbf{x})}{\partial x_k} \right), \quad k=1, \dots, n.$$

Note that the exact penalty function $P_1(\mathbf{x})$ is differentiable except on the union of the boundaries of the sets $\{x \mid g_j(\mathbf{x}) < 0\}$ which has n -dimensional Lebesgue measure 0. In the region where $P_1(\mathbf{x})$ is differentiable, the components of its gradient are:

$$\frac{\partial P_1}{\partial x_k} = [\nabla P_1(\mathbf{x})]_k = -\sum_{j=1}^p \chi_j \frac{\partial g_j}{\partial x_k}, \quad k=1, \dots, n.$$

Therefore, the circuit corresponds to solving the unconstrained problem using the exact penalty function $P_1(\mathbf{x})$ and $c = -b$. Since our circuit utilizes a smooth approximation of the exact penalty function, the resulting solution will closely approximate that of the original constrained problem provided that the weight c is sufficiently large. The following example illustrates what can happen if c is not sufficiently large.

Example 1

Consider the programming problem:

minimize $f(x) = 10x$

subject to

$$g_1(x) = \frac{x+1}{4} \geq 0$$

$$g_2(x) = \frac{1-x}{4} \geq 0.$$

We consider the case where $\tilde{\mu}_j$, $j = 1, 2$, is defined by $\tilde{\mu}_j = 0$ if $g_j(x) \geq 0$, and $\tilde{\mu}_j = -15$ if $g_j(x) < 0$. We also assume that x is constrained to be in the interval $-15 \leq x \leq 15$. Let $c = 15$. The energy function becomes a smooth approximation to the function $E^*(x)$ given by:

$$E^*(x) = f(x) + cP_1(x) = \begin{cases} f(x) - 15g_1(x) & \begin{cases} 6.25x - 3.75 & -15 \leq x < -1 \\ 10x & -1 \leq x \leq 1 \\ f(x) - 15g_2(x) & 1 < x \leq 15 \end{cases} \end{cases}$$

The dynamics of the circuit which models this programming problem would then be approximately described by the following equation:

$$\frac{dx}{dt} = -\frac{1}{C} \frac{\partial E^*}{\partial x} = \begin{cases} \frac{-6.25}{C} & -15 < x < -1 \\ \frac{-10}{C} & -1 < x < 1 \\ \frac{-13.75}{C} & 1 < x < 15 \end{cases} .$$

Thus, regardless of the initial value of x , the circuit trajectory will converge to the extreme point $x = -15$, which does not even correspond to a feasible point of the original problem. The solution to the problem is $x = -1$.

There are two approaches which can be used to avoid this type of difficulty. The most obvious approach would be to scale down the cost function f so that the penalty term in the energy function would dominate the cost function. This is accomplished by multiplying the function f by a sufficiently small scalar r and then solve the problem:

$$\text{minimize } rf(\mathbf{ax})$$

subject to

$$g(\mathbf{ax}) \geq 0 .$$

There are some drawbacks to this approach. First of all, finding an appropriate value of r might involve a great deal of work. Secondly, even if one is able to find an appropriate value of r , that value is only good for that particular problem. The approach which we now propose has neither of these shortcomings. In this approach the circuit trajectory first seeks to minimize the exact penalty function approximation of the problem regardless of the value of the μ_j 's. Before introducing the proposed approach it is necessary to define the saturation function:

$$S_{\alpha, \beta}(x) = \begin{cases} -\alpha & \text{for } x < -\beta \\ \frac{a}{\beta}x & \text{for } -\beta \leq x \leq \beta \\ a & \text{for } x > \beta \end{cases}$$

Let $\tilde{S}_{\alpha, \beta}(x)$ be a smoothed version of $S_{\alpha, \beta}(x)$ with the corners at $x = -\beta$ and $x = \beta$ smoothed out. When $\alpha = \beta$, we write $\tilde{S}_{\alpha, \alpha}$ as \tilde{S} . The proposed approach is illustrated by the network shown in Fig. 3.

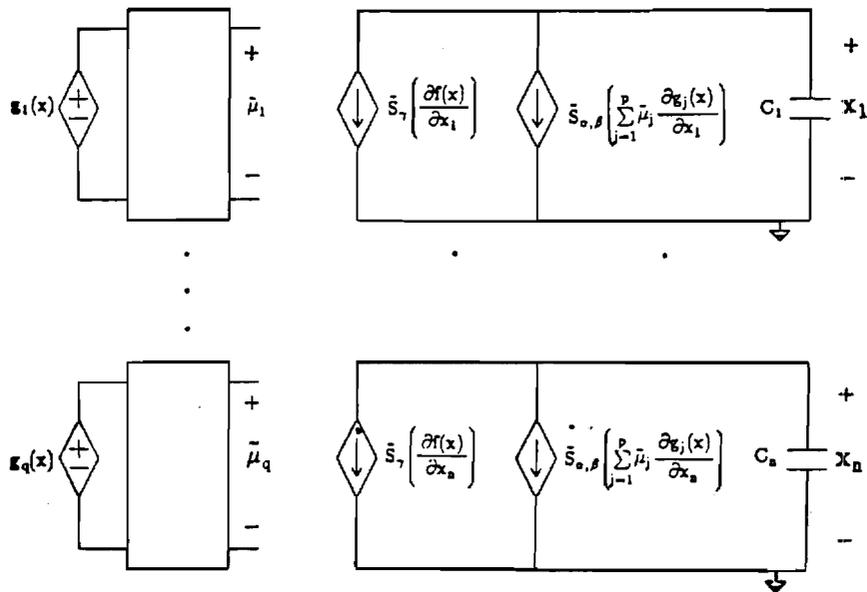


Figure 3. Proposed new network for solving programming problems.

In the above circuit, a , β , and γ are design parameters. We assume that $a > \gamma$. Applying Kirchhoff's current law to the circuits on the right side of Fig. 3, we have for $k = 1, \dots, n$:

$$\begin{aligned}\frac{d\mathbf{x}_k}{dt} &= -\frac{1}{C_k} \left[\tilde{S}_{\alpha,\beta} \left(\sum_{j=1}^p \tilde{\mu}_j \frac{\partial g_j(\mathbf{x})}{\partial \mathbf{x}_k} \right) + \tilde{S}_\gamma \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_k} \right) \right] \\ &= -\frac{1}{C_k} \left[\tilde{S}_{\alpha,\beta} \left(\sum_{j=1}^p b\chi_j \frac{\partial g_j(\mathbf{x})}{\partial \mathbf{x}_k} \right) + \tilde{S}_\gamma \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_k} \right) \right],\end{aligned}$$

and thus,

$$\frac{d\mathbf{x}_k}{dt} = -\frac{1}{C_k} \left[\tilde{S}_{\alpha,\beta} \left(\tilde{c} \frac{\partial P_1(\mathbf{x})}{\partial \mathbf{x}_k} \right) + \tilde{S}_\gamma \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_k} \right) \right],$$

if $\frac{\partial P_1(\mathbf{x})}{\partial \mathbf{x}_k}$ is defined and $\tilde{c} = -b$. Observe that if

$$\left| \tilde{c} \frac{\partial P_1(\mathbf{x})}{\partial \mathbf{x}_k} \right| > \beta,$$

then $\tilde{S}_{\alpha,\beta}$ saturates. In which case,

$$\operatorname{sgn} \left[\tilde{S}_{\alpha,\beta} \left(\tilde{c} \frac{\partial P_1(\mathbf{x})}{\partial \mathbf{x}_k} \right) + \tilde{S}_\gamma \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_k} \right) \right] = \operatorname{sgn} \left[\tilde{S}_{\alpha,\beta} \left(\tilde{c} \frac{\partial P_1(\mathbf{x})}{\partial \mathbf{x}_k} \right) \right]$$

since $a > \gamma$ by the design assumption. Since $C_k > 0$, we conclude that if $\left| \frac{\partial P_1(\mathbf{x})}{\partial \mathbf{x}_k} \right| > \beta$, then $\frac{d\mathbf{x}_k}{dt}$ and $\tilde{S}_{\alpha,\beta} \left(\tilde{c} \frac{\partial P_1(\mathbf{x})}{\partial \mathbf{x}_k} \right)$, which has the same sign as $\tilde{c} \frac{\partial P_1(\mathbf{x})}{\partial \mathbf{x}_k}$, have opposite signs. Hence, if $\left| \frac{\partial P_1(\mathbf{x})}{\partial \mathbf{x}_k} \right| > \beta$, then

$$\left| \frac{dx_k}{dt} \right| \geq \frac{\alpha - \gamma}{C_k}$$

and

$$\frac{\partial P_1(x)}{\partial x_k} \frac{dx_k}{dt} < -\frac{\beta}{\tilde{c}} \left(\frac{\alpha - \gamma}{C_k} \right) < 0.$$

Thus

$$\frac{dP_1(x)}{dt} = \sum_{k=1}^p \frac{\partial P_1(x)}{\partial x_k} \frac{dx_k}{dt} < 0.$$

This implies that whenever $\tilde{S}_{\alpha,\beta}$ saturates and the trajectory is in the region where P_1 is differentiable, then P_1 is decreasing along that trajectory. Note that generically, P_1 is differentiable in the complement of the feasible region except for a set of measure zero and that the circuits are designed so that $\tilde{S}_{\alpha,\beta}$ is almost indistinguishable from $\tilde{S}_{\alpha,0}$, which operates in the saturated mode. Thus, one would expect that the penalty function P_1 would decrease along the trajectories outside the feasible region. Note that if $\tilde{S}_{\alpha,\beta}$ operates in the saturated mode, then the value of $\tilde{c} > 0$ and the form of the objective function have no effect on the rate of decrease of P_1 along any trajectory.

Remark

If the initial condition is such that the system trajectory reaches the feasible region, then the circuit dynamics are governed by the equations

$$\frac{dx_k}{dt} = \left(-\frac{1}{C_k} \right) \tilde{S}_\gamma \left(\frac{\partial f}{\partial x_k} \right), \quad k=1, \dots, n.$$

5. Circuit Implementation of Proposed Neural Network

In order to test the ideas set forth in the previous sections, a circuit was built to solve 3-dimensional quadratic programming problems subject to two constraints (see Fig. 4). The implementation proposed differs from that proposed in [7] and [9] in that op-amps are used to implement the nonlinear dependent sources $\tilde{S}_{\alpha, \beta}$, and \tilde{S}_γ as opposed to resistors being used to generate linear dependent sources as in [7] and [9]. We can view the circuit as an interconnection of subcircuits which we refer to as nodes. There are two kinds of nodes in the circuit: the ones with noninverting output corresponding to x_i 's, and the ones with noninverting output corresponding to the $\tilde{\mu}_j$'s. We shall refer to these nodes as variable nodes and constraint nodes, respectively. The circuit implementation for these nodes is given in Fig. 5 and Fig. 6. The constraint and variable nodes are connected in the manner shown in Fig. 7.

As mentioned earlier, we would like to be somewhat careful in choosing a value for β . We want β to be small enough so that the constraint terms $\tilde{S}_{\alpha, \beta}$ are not a factor when the circuit is operating in the feasible region, while at the same time we want the the constraint terms to dominate when the circuit is operating outside the feasible region. Before choosing a value of β we should first examine the behavior of the μ terms which are used to determine the constraint terms $\tilde{S}_{\alpha, \beta}$. We examine $\tilde{\mu}_j$ as a function of the input voltage. A graph

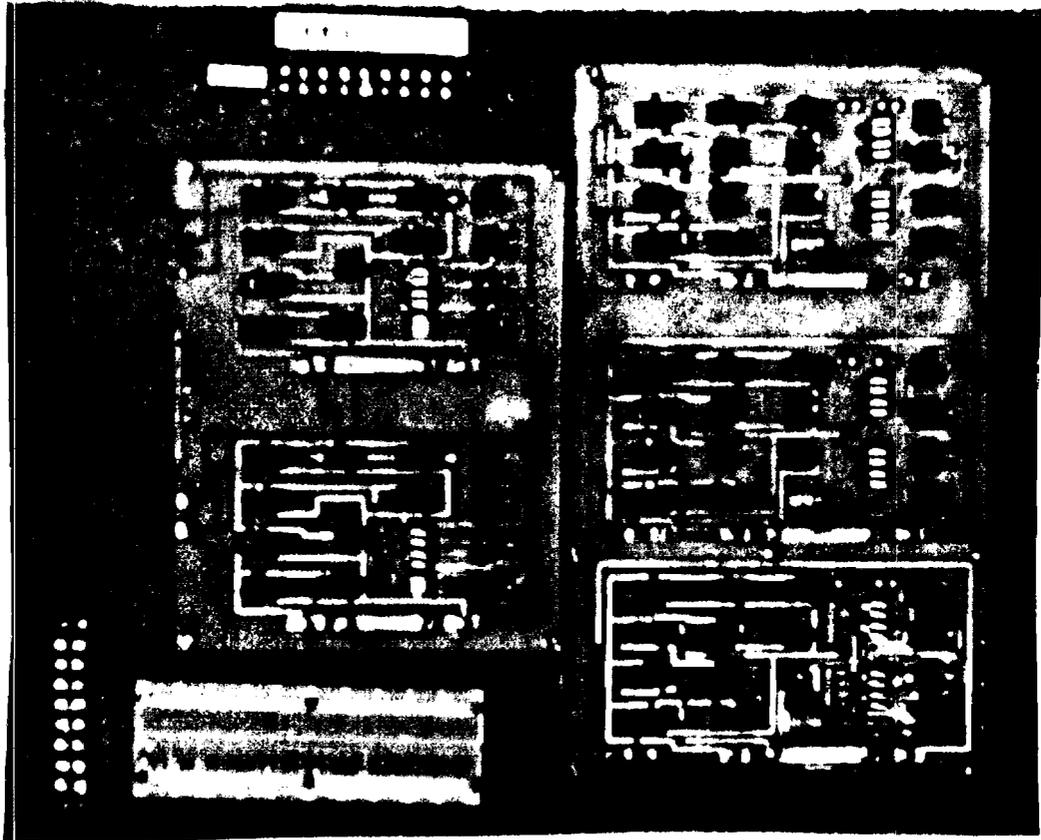


Figure 4. Circuit implementation of the proposed neural network for solving constrained optimization problems.

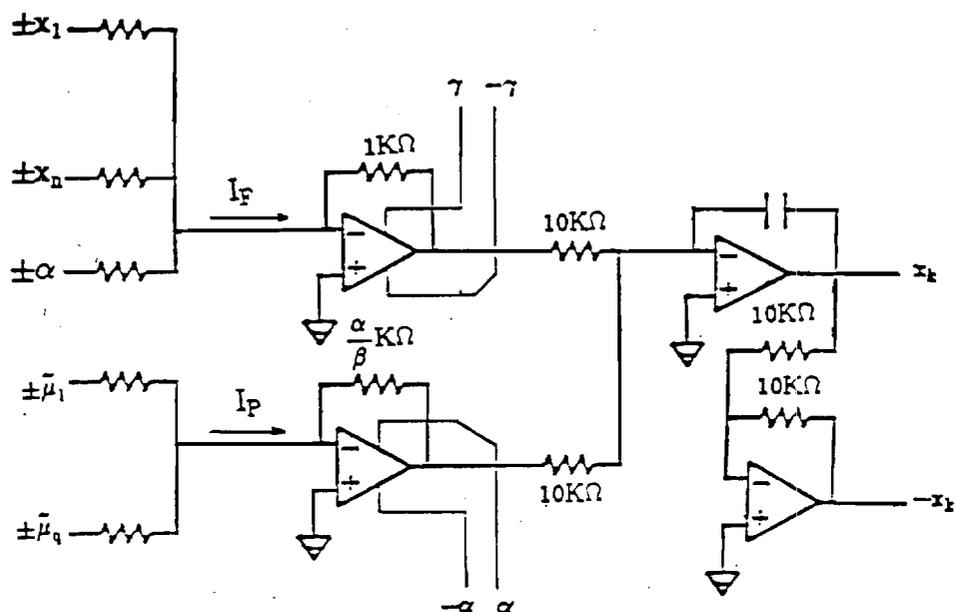


Figure 5a. Circuit implementation for an x node. The values of the unlabeled resistors are chosen such that $I_F = \frac{-\partial f(\mathbf{x})}{\partial x_k}$ mA and

$$I_P = \frac{-\partial P_1(\mathbf{x})}{\partial x_k} \text{ mA.}$$

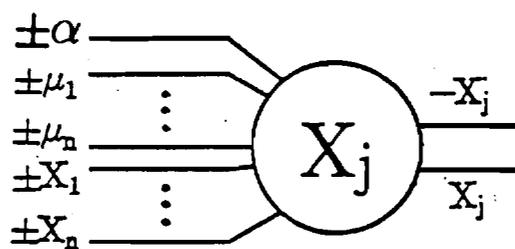


Figure 5b. Symbol for a variable node.

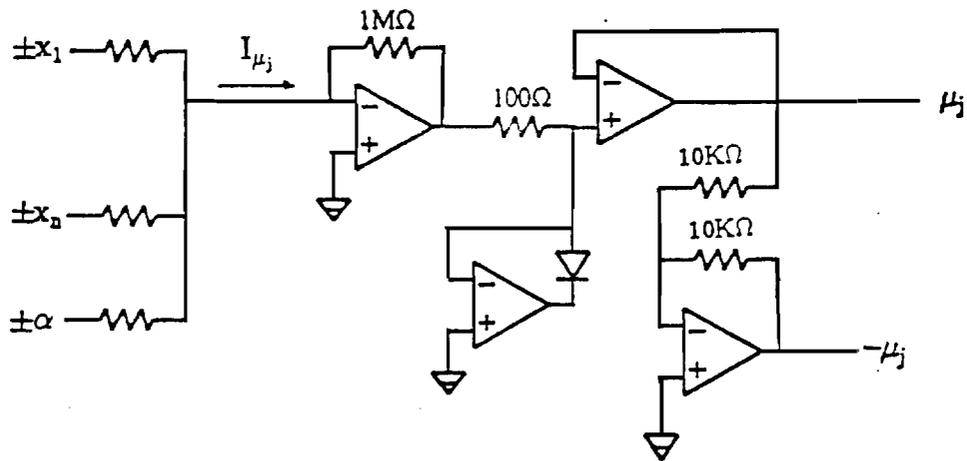


Figure 6a. Circuit implementation for an inequality constraint node. The unlabelled resistances are chosen in such a way that $I_{\mu_j} = -g_j(x)$ mA.

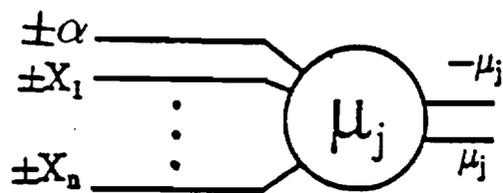


Figure 6b. Symbol for a constraint node.

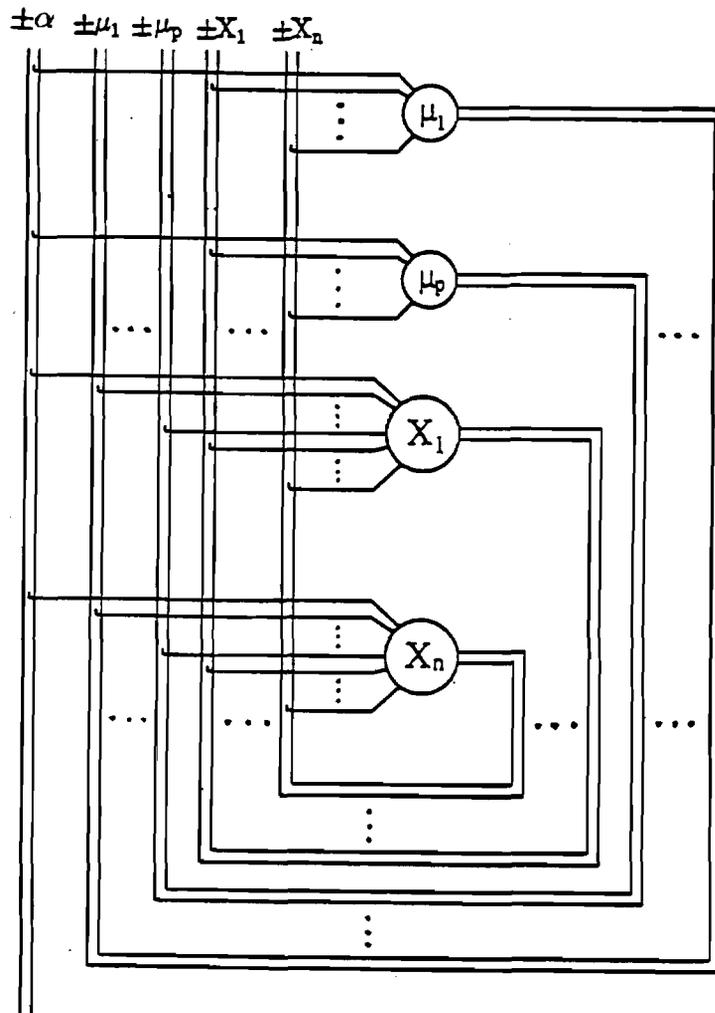


Figure 7. Schematic of the circuit implementation of the **proposed** neural network for solving constrained optimization problems.

of this function in terms of the voltage in the constraint node is depicted in Fig. 8.

For our circuit we chose $\beta = 0.5$, $\alpha = 2\gamma$. The implementation is such that $\tilde{\mu}_j$ is 0.43 mV when the constraint $g_j(x) > 0$ is satisfied and is equal to y when $g_j(x) < 0$. It follows from the equations

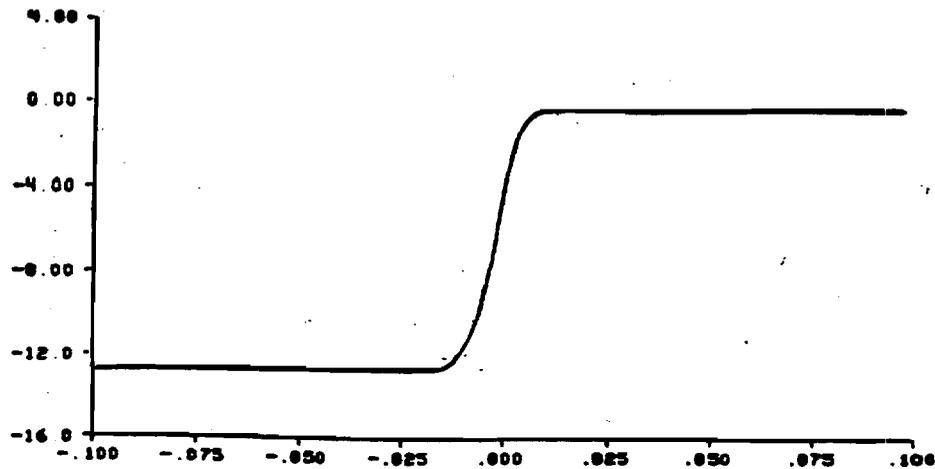


Figure 8. $\tilde{\mu}_j$ as a function of $g_j(\mathbf{x})$.

$$\frac{dx_k}{dt} = \left(\frac{-1}{C_k} \right) \left[\tilde{S}_{\alpha, \beta} \left(\tilde{c} \frac{\partial P_1(\mathbf{x})}{\partial x_k} \right) + \tilde{S}_\gamma \left(\frac{\partial f(\mathbf{x})}{\partial x_k} \right) \right], \quad k=1, \dots, n,$$

and the definition of S_γ and $S_{\alpha, \beta}$ that a sufficient condition for the constraint term to dominate is

$$\left(\frac{\alpha}{\beta} \right) \left| \sum_{j=1}^p \tilde{\mu}_j \frac{\partial g_j(\mathbf{x})}{\partial x_k} \right| > \gamma .$$

If $\alpha = 12$, the condition necessary for the $\tilde{S}_{\alpha, \beta}$ term to dominate outside the feasible region is

$$\left| \sum_{j=1}^p \lambda_j \frac{\partial g_j(\mathbf{x})}{\partial x_k} \right| > .02 .$$

Note that this condition is independent of the objective function,. It is also of interest to see what contribution a nonactive constraint could have on the dynamics of the system under the same assumptions. Since the maximum gain of the constraint node circuit is 20, the maximum contribution a nonactive constraint could have on the dynamics is given by

$$20 \tilde{\mu}_j \left(\frac{\partial g_j(\mathbf{x})}{\partial \mathbf{x}_k} \right) = 0.0092 \left(\frac{\partial g_j(\mathbf{x})}{\partial \mathbf{x}_k} \right) .$$

Using the previously specified values of β the circuit was tested on three problems.

Test Problem 1

$$\text{Minimize } f(\mathbf{x}) = -2.5(x_1^2 + x_2^2) - 5x_3^2 + 3x_1x_2 + 5x_1 + x_2 + 7x_3$$

For this problem we choose $\alpha = 9$ and $\gamma = 4.5$. The saturation limits of the op-amps cause the circuit trajectory to be constrained to the hypercube

$$\begin{aligned} -8.0 &\leq x_1 \leq 8.7 \\ -8.0 &\leq x_2 \leq 8.8 \\ -8.0 &\leq x_3 \leq 8.8 . \end{aligned}$$

Since the objective function is concave, one would expect the circuit trajectory to be driven to the boundaries of the hypercube imposed on the circuit by the saturation limits of the variable op-amps. The circuit went to one of the following boundary points depending on the initial condition. This is illustrated by the measurements depicted in the following table.

Initial point	Final point
(-2.5,1.5,3.0)	(-8.0,8.8,8.8)
(1.5,-2.5,3.5)	(8.7,-8.0,8.8)
(4.0,4.5,-2.0)	(8.7,8.8,-8.0)
(4.0,5.0,4.0)	(8.7,8.8,8.8)
(-5.0,-5.1,0.0)	(-8.0,-8.0,-8.0)
(-5.0,-5.1,2.0)	(-8.0,-8.0,8.8)
(2.4,-3.3,-1.8)	(8.7,-8.0,-8.0)
(-2.8,2.0,0.3)	(-8.0,8.8,-8.0)

Test Problem 2

$$\text{Minimize } f(\mathbf{x}) = -2.5(x_1^2 + x_2^2) + x_3^2 + 30x_1x_2 + 5x_1 + x_2 - 7x_3 .$$

For this problem we choose $\alpha = 12$ and $\gamma = 6$. The circuit trajectory is constrained to be in the hypercube

$$-13.0 \leq x_1 \leq 13.7$$

$$-12.9 \leq x_2 \leq 14.0$$

$$-13.0 \leq x_3 \leq 14.0.$$

The objective function has local minimizers at the points **(-13.0, 14.0, 3.5)** and **(13.7, -12.9, 3.5)**. The circuit trajectory converged to one of the above two points depending on the initial point as shown in the table below.

Initial point	Final point
(3.6, -8.3, -1.7)	(13.7, -12.9, 3.4)
(-9.4, 3.1, -3.7)	(-13.0, 14.0, 3.4)
(0.8, 3.0, -2.5)	(-13.0, 14.0, 3.4)
(3.0, 0.8, -2.5)	(13.7, -12.9, 3.4)
(-4.3, -5.3, 5.3)	(13.7, -12.9, 3.4)
(4.4, -6.1, 5.3)	(13.7, -12.9, 3.4)
(2.3, -6.3, 1.4)	(13.7, -12.9, 3.4)
(-3.0, -1.3, 0.9)	(-13.0, 14.0, 3.4)
(-8.8, 2.0, -1.7)	(-13.0, 14.0, 3.4)

Test Problem 3

Minimize $10x$

subject to

$$\frac{x}{4} + \frac{1}{4} \geq 0$$

$$\frac{1}{4} - \frac{x}{4} \geq 0 .$$

For this experiment the circuit parameters a and γ are **12** and **6** respectively. This problem has one minimizer at $x = -1$. Regardless of the initial conditions the network converged to the point $x = -0.995$. Notice that this problem is identical to Example 1. We see from the above experimental result that the proposed new implementation does a satisfactory job of solving constrained optimization problems.

6. Conclusions

The subject of this paper is solving constrained optimization problems with neural networks. We first examined the canonical dynamical nonlinear programming circuit proposed in [9] and noted that their circuit, is a gradient **system** which acts to minimize the penalty function **approximation** of the constrained optimization problem of interest utilizing the penalty function $P_2(\mathbf{x})$. Next we looked at their nonlinear programming neural network; implementation and discussed the effects of the fact that the variables $\mathbf{x}_1, \dots, \mathbf{x}_n$, and the variables μ_1, \dots, μ_p are constrained by the saturation limits of the op-amps in the network. We found that in the case where the function **which** generated the μ_j terms approximated a hard limiter the behavior of the circuit trajectory was very close to that which would result if the circuit was based on the exact penalty method (i.e. using penalty function $P_1(\mathbf{x})$). It was then noted that since the weight of the penalty function was bounded there would be cases where the circuit trajectory would converge to a nonfeasible solution. To **remedy** this difficulty we came up with a neural network which utilizes **non-linearities** to overcome the constraint problem associated with the implementations proposed in [7] and [9]. It was then shown that the proposed network would act first to decrease the exact penalty function before considering the objective function. Some practical implementation issues such as the effects of circuit nonlinearities on the solutions of the optimization problems were discussed. The proposed network was implemented and tested. The circuit was shown not to have the some of the shortcomings of the previously proposed networks. Finally, we would like to mention that one may try **different technologies** while implementing neural optimization networks. Some promising results in this direction, utilizing switched-capacitor circuits, are presented in [14].

References

- [1] J. B. Dennis, *Mathematical Programming and Electrical Networks*, London, England; Chapman & Hall, 1959.
- [2] T. E. Stern, *Theory of Nonlinear Networks and Systems*, New York; Addison-Wesley, 1965.
- [3] L. O. Chua and G.-N. Lin, "Non-linear optimization with constraints: A cookbook approach," *Int. J. Circuit Theory and Applications*, Vol. 11, No. 2, pp. 141-159, April 1983.
- [4] J. L. Huertas, A. Rueda, A. Rodriguez-Vazquez, and L. O. Chua, "Canonical nonlinear programming circuits," *Int. J. Circuit Theory and Applications*, Vol. 15, No. 1, pp. 71-77, Jan. 1987.
- [5] L. O. Chua and G.-N. Lin, "Nonlinear programming without computation," *IEEE Trans. Circuit and Systems*, Vol. CAS-31, No. 2, pp. 182-188, Feb. 1984.
- [6] G. Wilson, "Quadratic programming analogs," *IEEE Trans. Circuits Systems*, Vol. CAS-33, No. 9, pp. 907-911, Sept. 1986.
- [7] D. W. Tank and J. J. Hopfield, "Simple neural optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits and Systems*, Vol. CAS-33, No. 5, pp. 533-541, May 1986.
- [8] M. P. Kennedy and L. O. Chua, "Unifying the Tank and Hopfield linear programming network and the canonical nonlinear programming network of Chua and Lin," *IEEE Trans. Circuits and Systems*, Vol. GAS-34, No. 2, pp. 210-214, Feb. 1987.
- [9] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Trans. Circuits and Systems*, Vol. 35, No. 5, pp. 554-562, May 1988.

- [10] D. G. Luenberger, *Linear and Nonlinear Programming*, Reading, Massachusetts; Addison-Wesley, 1984.
 - [11] A. N. Tikhonov, A. B. Vasil'eva, A. G. Sveshnikov, *Differential Equations*, New York; Springer-Verlag, 1985.
 - [12] M. W. Hirsh, and S. Smale, *Differential Systems and Linear Algebra*, New York; Academic Press, 1974.
 - [13] L. O. Chua, and L. Yang. "Cellular neural networks: **Theory**," IEEE Trans. Circuit and Systems, Vol. 35, No. 10, Oct. 1988.
 - [14] A. Cichocki and R. Unbehauen, "Switched-capacitor neural networks for differential optimization," Int. J. Circuit Theory and Applications, Vol. 19, No. 2, pp. 161-187, March-April 1991.
-