

11-1-1991

NEURAL NETWORKS FOR CONSTRAINED OPTIMIZATION PROBLEMS

Walter E. Lillo

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.; Hui, Stefen; and Zak, Stanislaw H., "NEURAL NETWORKS FOR CONSTRAINED OPTIMIZATION PROBLEMS" (1991). *ECE Technical Reports*. Paper 322.

<http://docs.lib.purdue.edu/ecetr/322>

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.



Neural Networks for Constrained Optimization Problems

Walter E. Lillo
Stefen Hui
Stanislaw H. Żak

TR-EE 91-45
November 1991

NEURAL NETWORKS FOR CONSTRAINED OPTIMIZATION PROBLEMS

Walter E. Lillo

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 neural networks and analog circuits to solve constrained optimization problems. A novel neural network architecture is proposed for solving a class of nonlinear programming problems. The proposed neural network, or more precisely a physically realizable approximation, is then used to solve minimum norm problems subject to linear constraints. Minimum norm problems have many applications in various areas, but we focus on their applications to the control of discrete dynamic processes. The applicability of the proposed neural network is demonstrated on numerical examples.

Key Words:

Constrained optimization, Minimum norm problems, Analog circuits

1. Introduction

The idea of using analog circuits to solve mathematical programming problems seems to have been first proposed by Dennis (1959). Since then, various types of "neural" networks have been proposed to obtain solutions to constrained optimization problems. In particular Chua and Lin (1984) developed the canonical nonlinear programming circuit, using the Kuhn-Tucker conditions from mathematical programming theory, for simulating general nonlinear

programming problems. Later Tank and Hopfield (1986) developed an optimization network for solving linear programming problems using general principles resulting from the basic collective computational properties of a class of analog-processor networks. Practical design aspects of the Tank and Hopfield network along with its stability properties were discussed by Smith and Portmann (1989). An extension of the results of Tank and Hopfield to more general nonlinear programming problems was presented by Kennedy and Chua (1988). In addition, they noted that the network introduced by Tank and Hopfield could be considered to be a special case of the canonical nonlinear programming network proposed by Chua and Lin (1984), with capacitors added to account for the dynamic behavior of the circuit. Lillo et. al. (1991) have shown that the above discussed approach implicitly utilizes the penalty function method. The idea behind the penalty method is to approximate a constrained optimization problem by an unconstrained optimization problem - see Luenberger (1984, Chp. 12) for a discussion of this approach.

In this paper we use the penalty function method approach to synthesize a new neural optimization network capable of solving a general class of constrained optimization problems. The proposed programming network is discussed in section 2 along with its circuit implementation. We show that the penalty function approach allows one to better control the effects of the physical constraints of the network's building blocks than the previously proposed approaches. Our proposed architecture can be viewed as a continuous nonlinear neural network model. For a historical account of nonlinear neural networks, the reader may consult Grossberg (1988). In section 3 we discuss applications of the proposed neural optimization network to solving minimum norm problems of the form:

minimize $\|\mathbf{x}\|_p$

subject to $\mathbf{Ax} \geq \mathbf{b}$

where $p = 1, 2,$ or ∞ . The minimum norm problems are important, for example, in the context of the control of discrete processes (see Cadzow (1971) or LaSalle (1986, Chp. 17) for more information related to the issue). The behavior of the proposed networks are then tested on a numerical example and computer simulations are given in section 4. Conclusions are found in section 5.

2. Networks for Constrained Optimization

In this paper we are concerned with finding minimizers of constrained optimization problems. We consider the following general form of a constrained optimization problem

minimize $f(\mathbf{x})$

subject to

$$\mathbf{g}(\mathbf{x}) \geq \mathbf{0}$$

$$h(\mathbf{x}) = 0,$$

where $\mathbf{x} \in \mathbf{R}^n$, $f: \mathbf{R}^n \rightarrow \mathbf{R}$, $\mathbf{g} = [g_1, g_2, \dots, g_q]^T : \mathbf{R}^n \rightarrow \mathbf{R}^q$ and

$h = [h_1, h_2, \dots, h_m]^T : \mathbb{R}^n \rightarrow \mathbb{R}^m$ are vector valued functions of n variables with dimensions q and m respectively. Since we are dealing with physical devices it is reasonable to restrict the functions f, g , and h to be continuously differentiable.

Chua and Lin (1984), and later Kennedy and Chua (1988), proposed canonical nonlinear programming circuits for simulating the constrained optimization problems of the above type (see Fig. 1). They analyzed the case

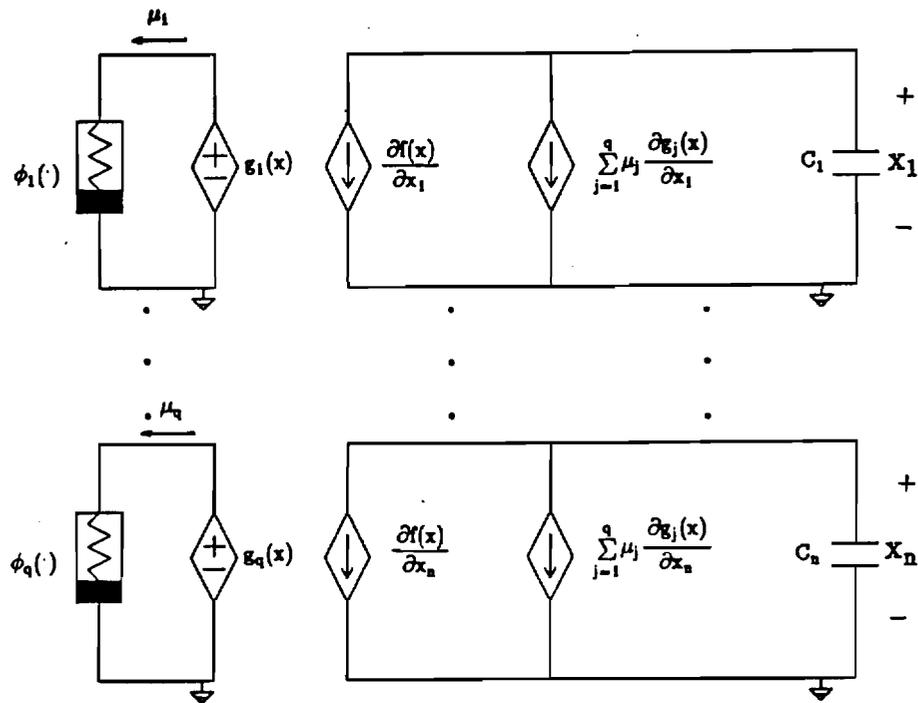


Figure 1. Dynamical canonical nonlinear programming circuit of Kennedy and Chua (1988).

when only the inequality constraints are present. Their development was based on the Kuhn-Tucker conditions from mathematical programming theory (see for example Luenberger (1984) for more information on the Kuhn-Tucker conditions). The functions ϕ_j , $j = 1, \dots, q$, on the left side of Fig. 1 are defined

by:

$$\mathbf{V} = \phi_j(\mathbf{I}) = \begin{cases} -c\mathbf{I} & \text{if } \mathbf{I} \geq \mathbf{0} \\ \mathbf{0} & \text{if } \mathbf{I} < \mathbf{0} \end{cases} .$$

Thus the μ_j terms have the form:

$$\mu_j = \phi_j(-g_j(\mathbf{x})) = \begin{cases} g_j(\mathbf{x})c & \text{if } g_j(\mathbf{x}) \leq 0 \\ 0 & \text{if } g_j(\mathbf{x}) > 0 \end{cases} \quad j = 1, \dots, q.$$

Now applying Kirchhoff's current law (see for example Halliday and Resnick (1978, p. 702)) to the circuit on the right side of Fig. 1 we obtain

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

Solving for $\frac{dx_k}{dt}$ we obtain

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

Note that if $c \rightarrow \infty$ then, in the steady state, the Kuhn-Tucker conditions are satisfied.

In this paper we examine the case where we have equality constraints as well as inequality constraints. An equality constraint $h_j(\mathbf{x}) = 0$ can be represented in terms of inequality constraint(s) in one of the following ways:

$$g_j(\mathbf{x}) = -(h_j(\mathbf{x}))^2 \geq 0,$$

or

$$g_{j1}(\mathbf{x}) = h_j(\mathbf{x}) \geq 0, \quad g_{j2}(\mathbf{x}) = -h_j(\mathbf{x}) \geq 0 .$$

However, to implement equality constraints in terms of inequality constraints would be inefficient as will be seen later. In this paper we propose an alternative circuit which has a more efficient implementation of equality constraints and a general form which more readily lends itself to implementation. This alternative approach utilizes the penalty method. Utilizing the penalty method, a constrained optimization problem considered in this paper can be approximated by an unconstrained problem of the form:

$$\text{minimize } \left[f(\mathbf{x}) + cP(\mathbf{x}) \right],$$

where $c > 0$ is a constant, often referred to as a weight, and $P(\mathbf{x})$ is a penalty function. A penalty function is a continuous non-negative function which is zero at a point if and only if all constraints are satisfied at that point. In this paper we consider penalty functions of the form:

$$P(\mathbf{x}) = \sum_{j=1}^q g_j^-(\mathbf{x}) + \sum_{j=1}^m |h_j(\mathbf{x})| ,$$

where $g_j^-(\mathbf{x}) = -\min(\mathbf{0}, g_j(\mathbf{x}))$. If we consider an equality constraint as two inequality constraints, then the penalty function can be rewritten as:

$$P(\mathbf{x}) = \sum_{j=1}^q g_j^-(\mathbf{x}) + \sum_{j=1}^m \left(g_{j1}^-(\mathbf{x}) + g_{j2}^-(\mathbf{x}) \right) ,$$

where

$$g_{j1}(\mathbf{x}) = h_j(\mathbf{x}) \text{ and } g_{j2}(\mathbf{x}) = -h_j(\mathbf{x}) .$$

The above penalty function $P(\mathbf{x})$ is often referred to as an exact penalty function because for a sufficiently large finite value of c the penalty method approximation, with the above $P(\mathbf{x})$, yields the same global minimizers as the constrained problem. The exact penalty functions have the drawback that they are not usually differentiable. Having reviewed the penalty method we now introduce the proposed network. The functions $\tilde{S}_{\alpha, \beta}$ and \tilde{S}_{γ} in Fig. 2 are smooth versions of the saturation functions $S_{\alpha, \beta}$ defined by:

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

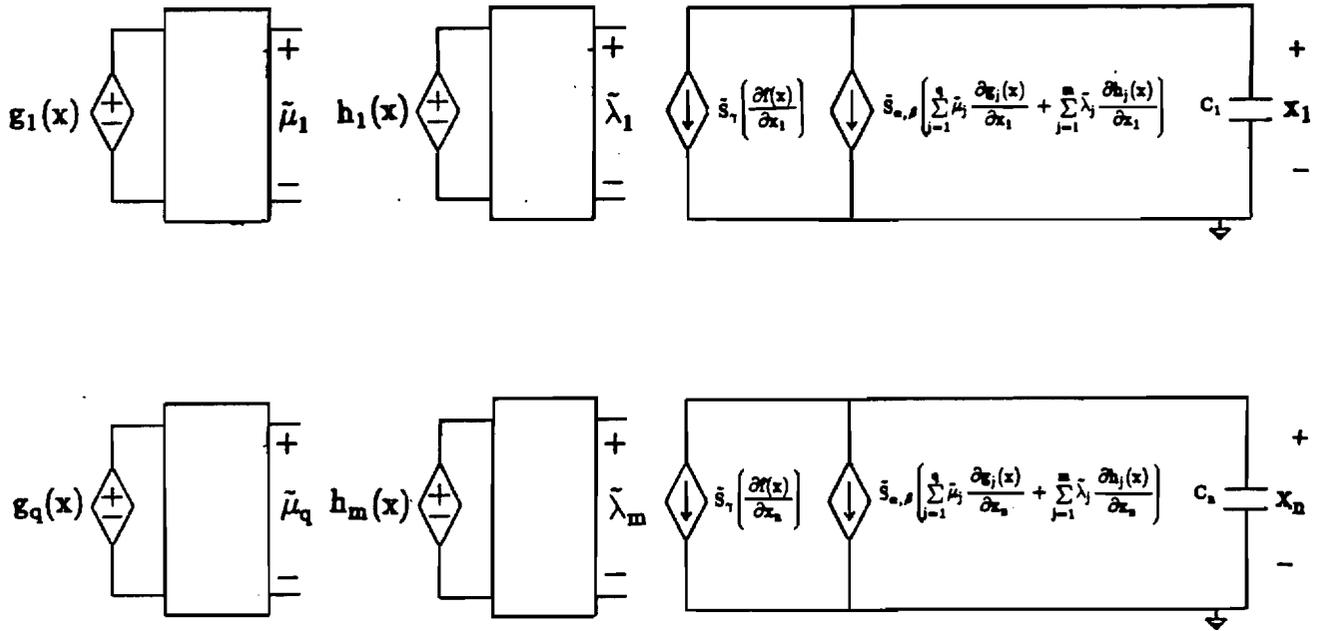


Figure 2. The proposed network for constrained optimization.

When $a = \beta$, we write $\tilde{S}_{\alpha,\alpha}$ as \tilde{S} . We assume that $a > \gamma$. The $\tilde{\mu}_j$ and $\tilde{\lambda}_j$ terms are defined as:

$$\tilde{\mu}_j = -\tilde{S}_{c,\zeta}^-(g_j(\mathbf{x})) = \min(0, \tilde{S}_{c,\zeta}(g_j(\mathbf{x}))) ,$$

$$\tilde{\lambda}_j = \tilde{S}_{c,\zeta}(h_j(\mathbf{x})) .$$

When ζ is small, the $\tilde{\mu}_j$ and $\tilde{\lambda}_j$ terms can be approximated as:

$$\tilde{\mu}_j \cong \frac{c}{2} \text{sgn}(g_j(\mathbf{x})) - \frac{c}{2} ,$$

$$\tilde{\lambda}_j \cong \text{csgn}(h_j(x)) .$$

Remark

The $\tilde{\mu}_j$ terms differ from the μ_j terms in the the canonical dynamical circuit of Kennedy and Chua (Fig. 1) in that their values are bounded. This modification was made in order to accommodate the saturation limits of the op-amps used in implementing the functions. As a result of replacing the μ_j terms, it is necessary to replace the linear current sources of the dynamical canonical circuit with nonlinear current sources in order to effectively enforce the constraints.

Applying Kirchoff's current law to the circuits on the right hand side of Fig. 2 yields:

$$\frac{d\mathbf{x}_k}{dt} = \left(\frac{-1}{C_k} \right) \left[\tilde{S}_{\alpha, \beta} \left(\sum_{j=1}^q \tilde{\mu}_j \frac{\partial g_j(\mathbf{x})}{\partial \mathbf{x}_k} + \sum_{j=1}^m \tilde{\lambda}_j \frac{\partial h_j(\mathbf{x})}{\partial \mathbf{x}_k} \right) + \tilde{S}_\gamma \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_k} \right) \right] , \quad k = 1, \dots, n.$$

Substituting for $\tilde{\mu}_j$ and $\tilde{\lambda}_j$, we have

$$\frac{dx_k}{dt} = \left(\frac{-1}{C_k} \right) \left[\tilde{S}_{\alpha, \beta} \left(-c \sum_{j \in J} \frac{\partial g_j(\mathbf{x})}{\partial x_k} + c \sum_{j=1}^m \text{sgn}(h_j(\mathbf{x})) \frac{\partial h_j(\mathbf{x})}{\partial x_k} \right) + \tilde{S}_\gamma \left(\frac{\partial f(\mathbf{x})}{\partial x_k} \right) \right],$$

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

where J is the index set of violated inequality constraints. In the region where the gradient of $P(\mathbf{x})$ is defined, this equation can be rewritten as

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

Note that if

$$\left| c \frac{\partial P(\mathbf{x})}{\partial x_k} \right| > \beta, \quad ,$$

then the term $\tilde{S}_{\alpha, \beta}$ saturates. If we assume the trajectory is in a region where

$\left| c \frac{\partial P(\mathbf{x})}{\partial x_k} \right| > \beta$, then by the design assumption that $a > \gamma$, we obtain:

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

and

$$\left| \frac{dx_k}{dt} \right| \geq \frac{\alpha - \gamma}{C_k} > 0.$$

In addition, since $C_k > 0$, we conclude that if $\left| c \frac{\partial P(\mathbf{x})}{\partial x_k} \right| > \beta$, then $\frac{dx_k}{dt}$ and $c \frac{\partial P(\mathbf{x})}{\partial x_k}$ have opposite signs. Hence, if $\left| c \frac{\partial P(\mathbf{x})}{\partial x_k} \right| > \beta$, then

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

Thus

$$\frac{dP(\mathbf{x})}{dt} = \sum_{k=1}^n \frac{\partial P(\mathbf{x})}{\partial x_k} \frac{dx_k}{dt} < 0.$$

This implies that whenever $\tilde{S}_{\alpha, \beta}$ saturates and the trajectory is in the region where $P(\mathbf{x})$ is differentiable, then $P(\mathbf{x})$ is decreasing along that trajectory. Note that the set of points where $P(\mathbf{x})$ is not differentiable has an n -dimensional Lebesgue measure zero and that the circuits are designed so that β is small and thus $\tilde{S}_{\alpha, \beta}$ will be saturated at almost all points outside the feasible region. Thus, one would expect that the penalty function $P(\mathbf{x})$ would decrease along the trajectories outside the feasible region. Note that if $\tilde{S}_{\alpha, \beta}$ operates in the saturated mode, then the bound for the rate of decrease of the penalty function $P(\mathbf{x})$ is independent of the form of the objective function.

It should be noted that if the initial condition is such that the system trajectory reaches the feasible region, then the circuit dynamics are governed by the equations

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

Having examined the dynamical behavior of the circuit in Fig. 2, we will now consider its implementation. For the case of quadratic programming problems subject to linear equality and inequality constraints the circuit shown in Fig. 2 could be implemented using a neural network with the structure depicted in Fig. 3. The implementation of the μ -node is the same as was proposed by Kennedy and Chua (1988) and is shown in Fig. 4. The implementations for the h and x nodes are depicted in Fig. 5 and 6. It should be clear from the implementation of the various nodes that to represent an equality constraint in terms of inequality constraints would be rather inefficient since an inequality constraint node requires more hardware than an equality constraint node. We would like to note that one may also use switched-capacitor circuits to implement neural optimization networks (Cichocki and Unbehauen (1991)).

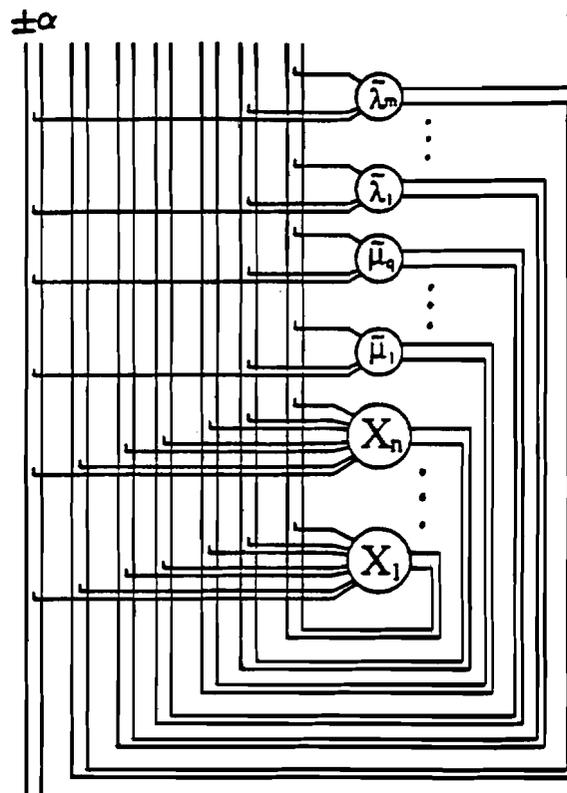


Figure 3. Neural network for solving quadratic programming problems subject to linear constraints.

Having given an implementation corresponding to the general case of quadratic programming we will now examine how a network of this basic structure can be used to solve some minimum norm problems of interest.

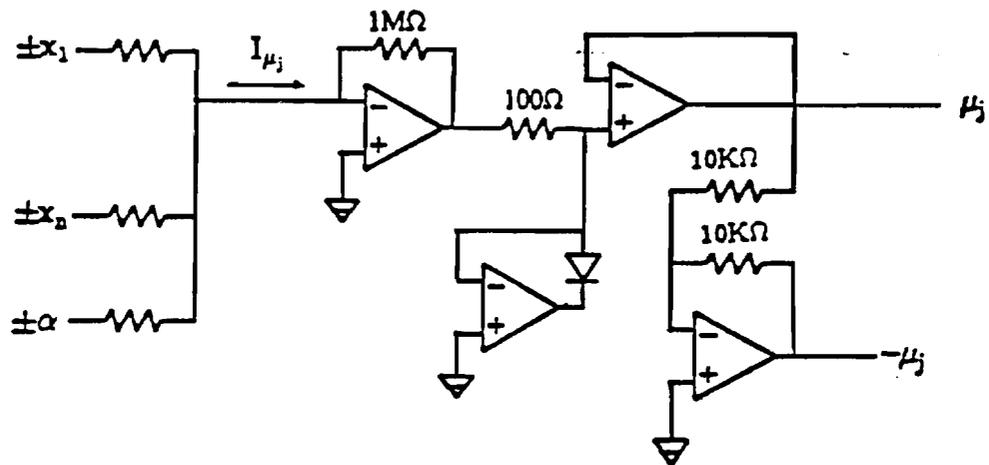


Figure 4. Circuit implementation for an inequality constraint node. The unlabeled resistances are chosen in such a way that $I_{\mu_j} = -g_j(\mathbf{x})$ mA.

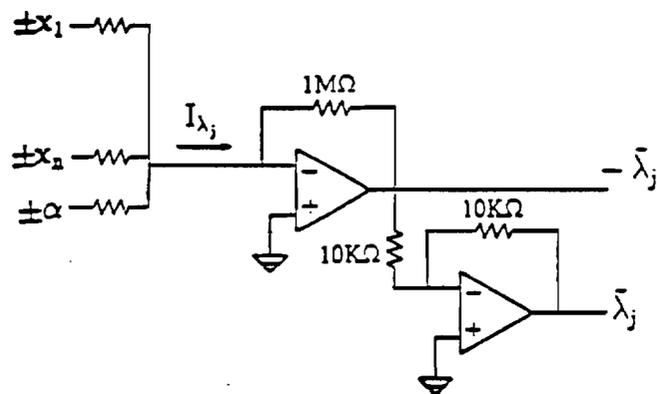


Figure 5. Circuit implementation for an equality constraint. The values of the unlabeled resistors are chosen so that $I_{\lambda_j} = -h_j(\mathbf{x})$ mA.

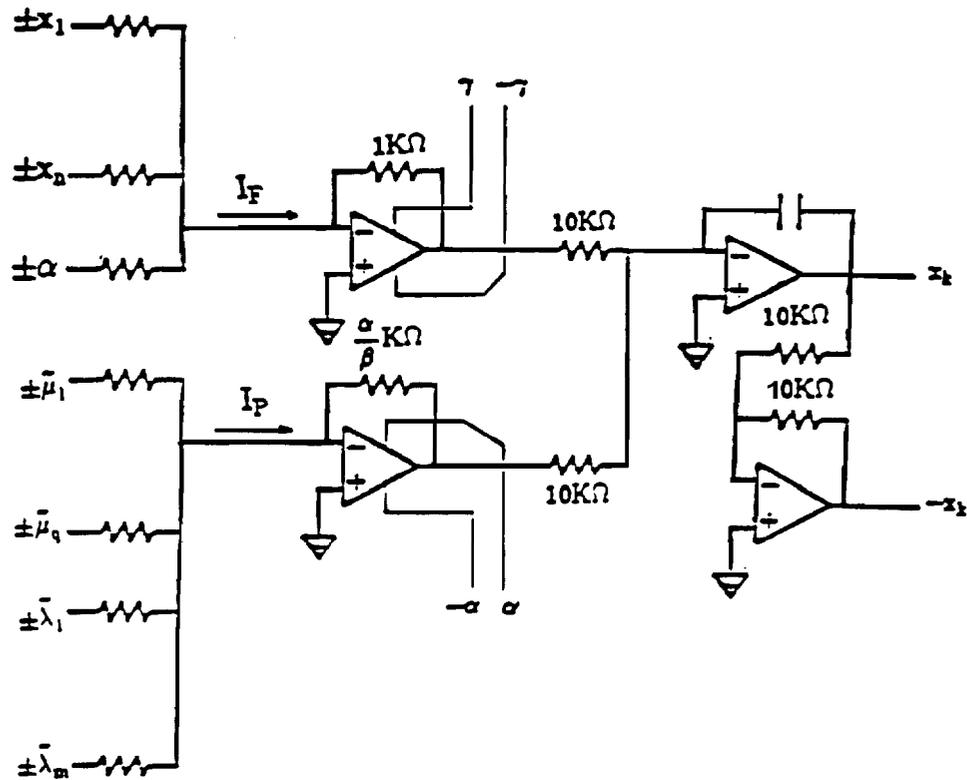


Figure 6. Circuit implementation for an x node. The values of the un-
 abled resistors are chosen so that $I_F = \frac{-\partial f(x)}{\partial x_k}$ mA and

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

3. Networks for Solving Minimum Norm Problems

In this section we show how the previously proposed neural network archi-
 tecture can be applied to control discrete dynamic systems modeled by the
 equation

$$\xi_{k+1} = F\xi_k + Gu_k \quad ,$$

where $\xi_k \in \mathbb{R}^m, u_k \in \mathbb{R}^1$, for $k = 1, 2, \dots$, and F, G are constant matrices with appropriate dimensions. If we iteratively apply the previous equation we obtain

$$\xi_N = F^N \xi_0 + F^{N-1} Gu_0 + \dots + FG u_{N-2} + Gu_{N-1} \quad .$$

We assume that our system is completely controllable (Kailath (1980)). This implies that we can drive the system to an arbitrary desired state, ξ_d , from an arbitrary initial state, ξ_0 . Thus for sufficiently large N , ($N \geq m$) we can find a sequence of inputs (u_0, u_1, \dots, u_{N-1}) such that

$$\xi_d = \xi_N = F^N \xi_0 + F^{N-1} Gu_0 + \dots + FG u_{N-2} + Gu_{N-1} \quad .$$

In the case where $N > m$ there are an infinite number of input sequences which would drive the system to the desired state. This can be seen more clearly if we rewrite the previous equation using the following definitions:

$$\mathbf{A} = [G, FG, \dots, F^{N-2}G, F^{N-1}G] \quad , \quad \mathbf{x}^T = [u_{N-1}^T, u_{N-2}^T, \dots, u_0^T] .$$

With these definitions, we have

$$\xi_d = F^N \xi_0 + Ax \quad ,$$

or

$$\xi_d - F^N \xi_0 = Ax \quad .$$

If we let $b = \xi_d - F^N \xi_0$ then we have

$$Ax = b \quad .$$

If we define $n = lN$ then A is $m \times n$, b is $m \times 1$ and x is $n \times 1$. Since the system is completely controllable, $N > m$ the rank of A is m and the null space of A has dimension $n - m > 0$. From this it should be clear that the system of equations $Ax = b$ is underdetermined (i.e. there is an infinite number of possible solutions). Since there are many possible solutions, secondary criteria are often used to determine which of the input sequences satisfying the constraints should be used. Often it is desirable to find the solution which in some sense minimizes the input x . This is the reason we consider the following constrained optimization problem

$$\text{minimize } \|x\|_p$$

$$\text{subject to } Ax = b \quad ,$$

where $p = 1, 2$, or ∞ . The solutions corresponding to these problems are referred to as the minimum fuel, minimum energy, and minimum amplitude

solutions respectively. Because of the importance of these problems they have been studied fairly extensively (see for example Cadzow (1971,1973), Koley (1975), or LaSalle (1986)). For the case of $p=2$, there are algorithms based on linear algebra which solve this problem. When $p = 1$ or $p = \infty$, the problems are somewhat more complex. There are algorithms based on results from functional analysis which have been proposed to solve these problems (Cadzow (1971,1973)). In applications such as real time control the speed at which a solution can be obtained is of the utmost importance. It is for this reason that we propose the use of analog circuits, or neural networks, which are capable of obtaining solutions on the order of a few time constants.

We will now examine how the quadratic programming implementation given in the previous section can be applied to solving the problems of interest. The first thing we notice with all these problems is that the constraints are linear. Thus in the case where $p = 2$, since the objective function of the equivalent problem can be expressed as a quadratic, the network given in the previous section can be used to solve the problem.

For the case of $p=1$, the objective function cannot be expressed as a quadratic. However, as shown below, the components of the gradient of the objective function are still simple functions of the variables x_1, \dots, x_n :

$$\frac{\partial \|\mathbf{x}\|_1}{\partial x_k} = \text{sgn}(x_k) .$$

This being the case, a component of the gradient of $\|\mathbf{x}\|_1$ can be approximated by the circuit depicted in Fig. 7.

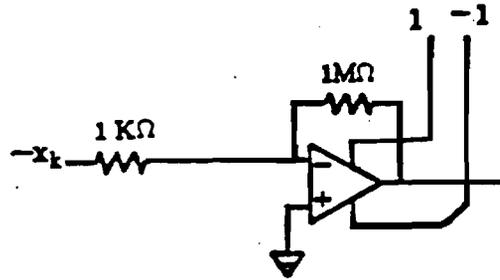


Figure 7. Implementation for approximating a component of the gradient of the objective function for the case where $p=1$.

The x-nodes would then be modified as shown in Fig. 8.

For the case $p = \infty$ the objective function cannot be expressed as a quadratic. In addition, we can see from the equation below that the components of the gradient of the objective function cannot be expressed in a simple manner as was the case when $p = 1$. They have the form

$$\frac{\partial(\|x\|_{\infty})}{\partial x_k} = \begin{cases} \text{sgn}(x_k) & \text{if } |x_k| = \max_i(|x_i|) \\ 0 & \text{otherwise} \end{cases} \quad k = 1, \dots, n \quad .$$

Rather than try to implement this problem directly by building a circuit to approximate the components of the gradient of the objective function given above, we transform the problem into an equivalent one which can be

simulated by a network of the form given in section 2. To understand how this is done, consider the level surface $\|\mathbf{x}\|_\infty = \mathbf{a}$, where $\mathbf{a} > \mathbf{0}$. This level surface corresponds to the boundary of the closed hypercube:

$$H_\sigma = \{ \mathbf{x} \in \mathbb{R}^n \mid -\sigma \leq x_j \leq \sigma, \quad j = 1, \dots, n \} \quad .$$

Thus the problem can be viewed as finding the smallest value of $\mathbf{a} > \mathbf{0}$ such that the constraint $\mathbf{Ax} = \mathbf{b}$ is satisfied and \mathbf{x} is an element of the set H . If we let $x_{n+1} = \mathbf{a}$ and $\mathbf{x}^* = [x_1, x_2, \dots, x_n]^T$ then the problem can be written as:

minimize x_{n+1}

subject to

$$h(\mathbf{x}) = \mathbf{Ax}^* - \mathbf{b} = \mathbf{0}$$

$$g_1(\mathbf{x}) = x_{n+1} \geq \mathbf{0}$$

$$g_{11}(\mathbf{x}) = x_{n+1} - x_1 \geq \mathbf{0}$$

$$g_{12}(\mathbf{x}) = x_{n+1} + x_1 \geq \mathbf{0}$$

$$g_{21}(\mathbf{x}) = x_{n+1} - x_2 \geq \mathbf{0}$$

$$g_{22}(\mathbf{x}) = x_{n+1} + x_2 \geq \mathbf{0}$$

. . .

$$g_{n1}(\mathbf{x}) = x_{n+1} - x_n \geq \mathbf{0}$$

$$g_{n2}(\mathbf{x}) = x_{n+1} + x_n \geq \mathbf{0}.$$

We have transformed the original problem into a linear **programming** problem and the quadratic programming network introduced in the previous section can be used to solve this problem.

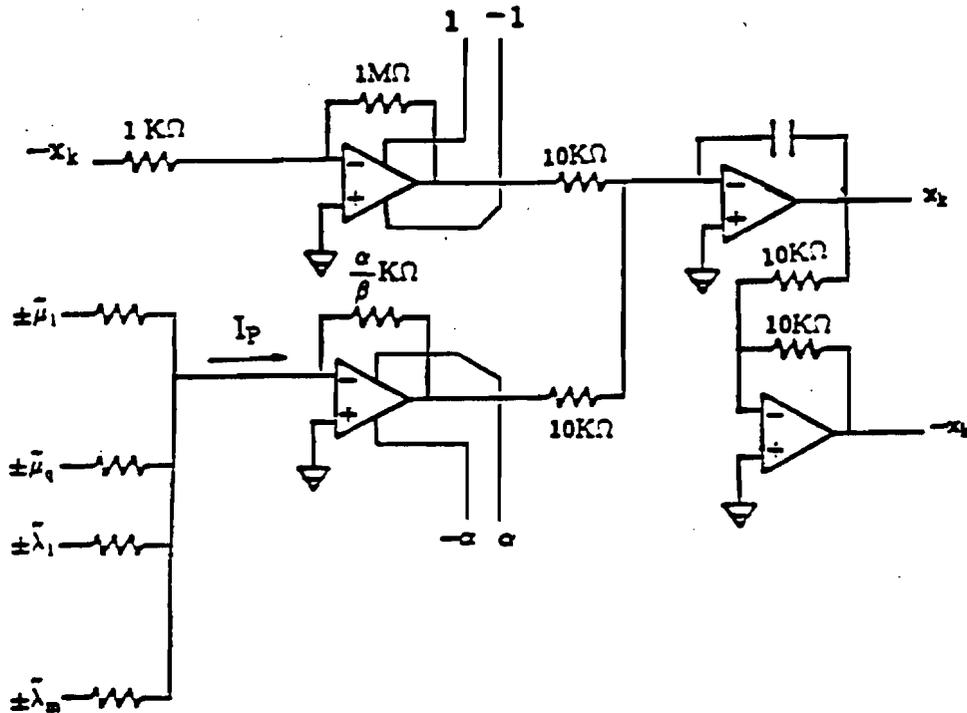


Figure 8. Implementation for an x node for the case where $p=1$. The unlabeled resistances are chosen so that the current $I_p = -\frac{\partial P(\mathbf{x})}{\partial x_k}$ mA.

For some other interesting applications of neural networks for quadratic minimization the reader may consult Sudharsanan and Sundareshan (1991).

4. Case Study

In order to test the ideas presented in this paper, simulations of the proposed implementations were performed on a digital computer. The simulations are based on the following differential equations (see section 2):

$$\frac{dx_k}{dt} = \left(\frac{-1}{C_k} \right) \left[S_{\alpha, \beta} \left(\sum_{j=1}^q \tilde{\mu}_j \frac{\partial g_j(x)}{\partial x_k} + \sum_{j=1}^m \tilde{\lambda}_j \frac{\partial h_j(x)}{\partial x_k} \right) + S_{\gamma} \left(\frac{\partial f(x)}{\partial x_k} \right) \right], \quad k = 1, \dots, n,$$

where $S_{\alpha, \beta}$ and S_{γ} are as defined in Section 2 with $\alpha = 12$, $\beta = 0.5$, and $\gamma = 6$. we use $c = 1000$ in the definitions of the variables $\tilde{\mu}_j, j = 1, \dots, n$ and $\tilde{\lambda}_j, j = 1, \dots, q$. We approximate the signum function $\text{sgn}(x)$ by

$$S_{1, .001} = \begin{cases} 1 & \beta > 0.001 \\ 1000\beta & 0.001 > \beta > -0.001 \\ -1 & \beta < -0.001 \end{cases} .$$

The problem which we choose to simulate is taken from Cadzow (1973) and has the form:

minimize $\|\mathbf{x}\|_p$

subject to $A\mathbf{x} = \mathbf{b}$

where $p = 1, 2$, or ∞ , and

$$A = \begin{bmatrix} 2 & -1 & 4 & 0 & 3 & 1 \\ 5 & 1 & -3 & 1 & 2 & 0 \\ 1 & -2 & 1 & -5 & -1 & 4 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ 1 \\ -4 \end{bmatrix}.$$

The variables x_j , $j = 1, \dots, n$, are constrained to be in the interval $[-12, 12]$. The results of the simulations for $p = 1, 2$ and ∞ are given below.

For $p = 1$, as shown in Fig. 9, the trajectories converged to the point

$$\mathbf{x} = [0.000, 0.000, 0.192, 0.756, 0.410, 0.001]^T,$$

which gives $\|\mathbf{x}\|_1 = 1.36$.

For $p = 2$, as shown in Fig. 10, the trajectories converged to the point

$$\mathbf{x} = [0.088, 0.112, 0.273, 0.503, 0.383, -0.310]^T,$$

which gives $\|\mathbf{x}\|_2 = 0.769$.

For $p = \infty$, as shown in Fig. 11 and 12, the trajectories converged to the point

$$\mathbf{x} = [0.113, 0.372, 0.351, 0.372, 0.372, -0.372]^T,$$

which gives $\|\mathbf{x}\|_\infty = 0.372$.

The analytical solutions to the three problems are

$$\begin{aligned} & (0.000, 0.000, 0.192, 0.756, 0.410, 0.000), \\ & (0.088, 0.108, 0.273, 0.505, 0.383, -0.310), \\ & (0.113, 0.372, 0.351, 0.372, 0.372, -0.372). \end{aligned}$$

Thus the results of the simulations closely correspond to the analytical solution.

Another important consideration is the speed with which the network converges to the correct solution. This depends on the value of the time constants and the initial condition of the network. In the above simulations we assumed there is no initial charge on the capacitors in the networks. This corresponds to the condition $\mathbf{x}_j = \mathbf{0}$, $j = 1, \dots, n$. From the following plots of the trajectories of the variables for the three problems we can see that the network converged to the solution within a few time constants.

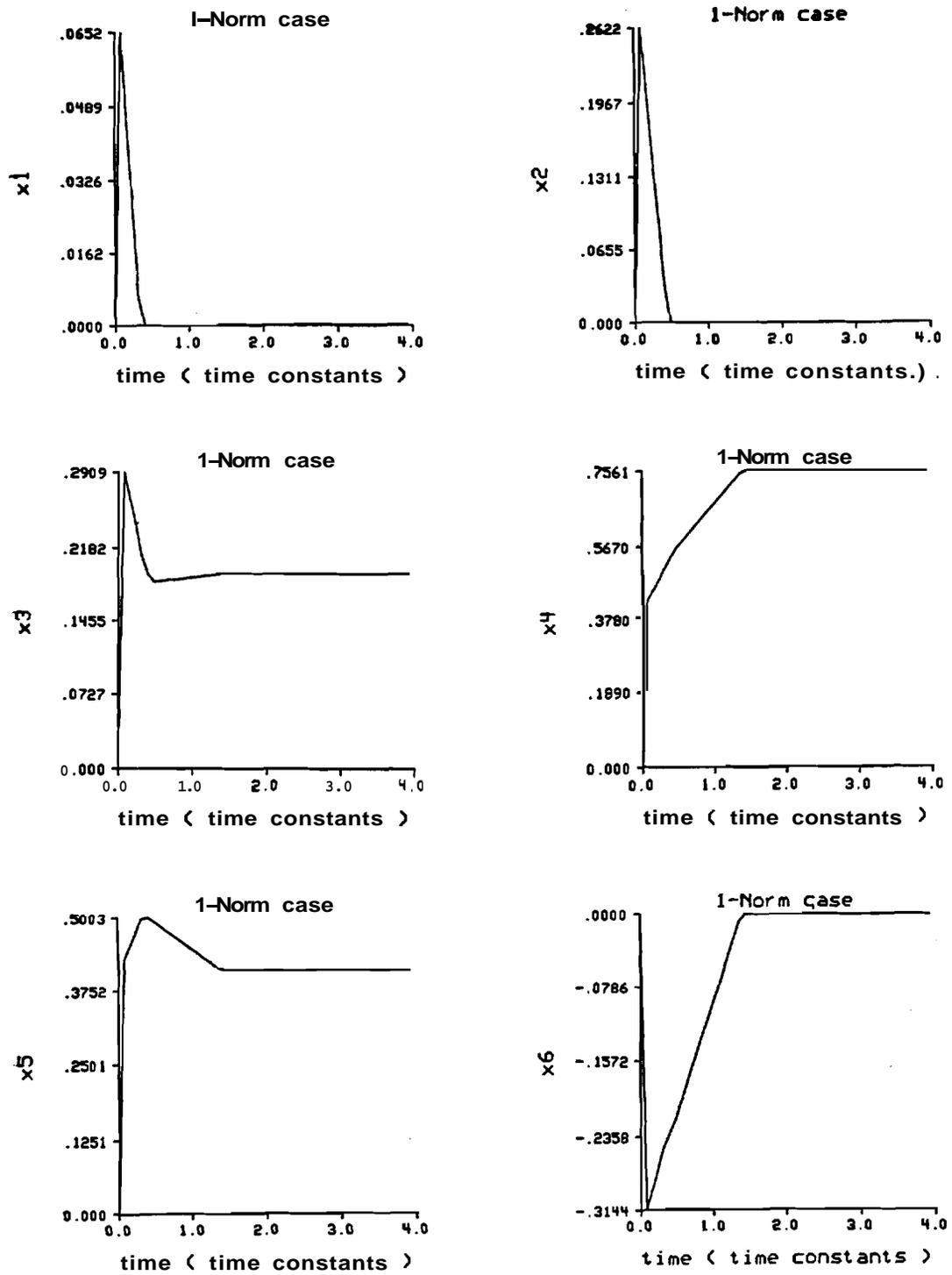


Figure 9. Trajectories corresponding to the case $p=1$.

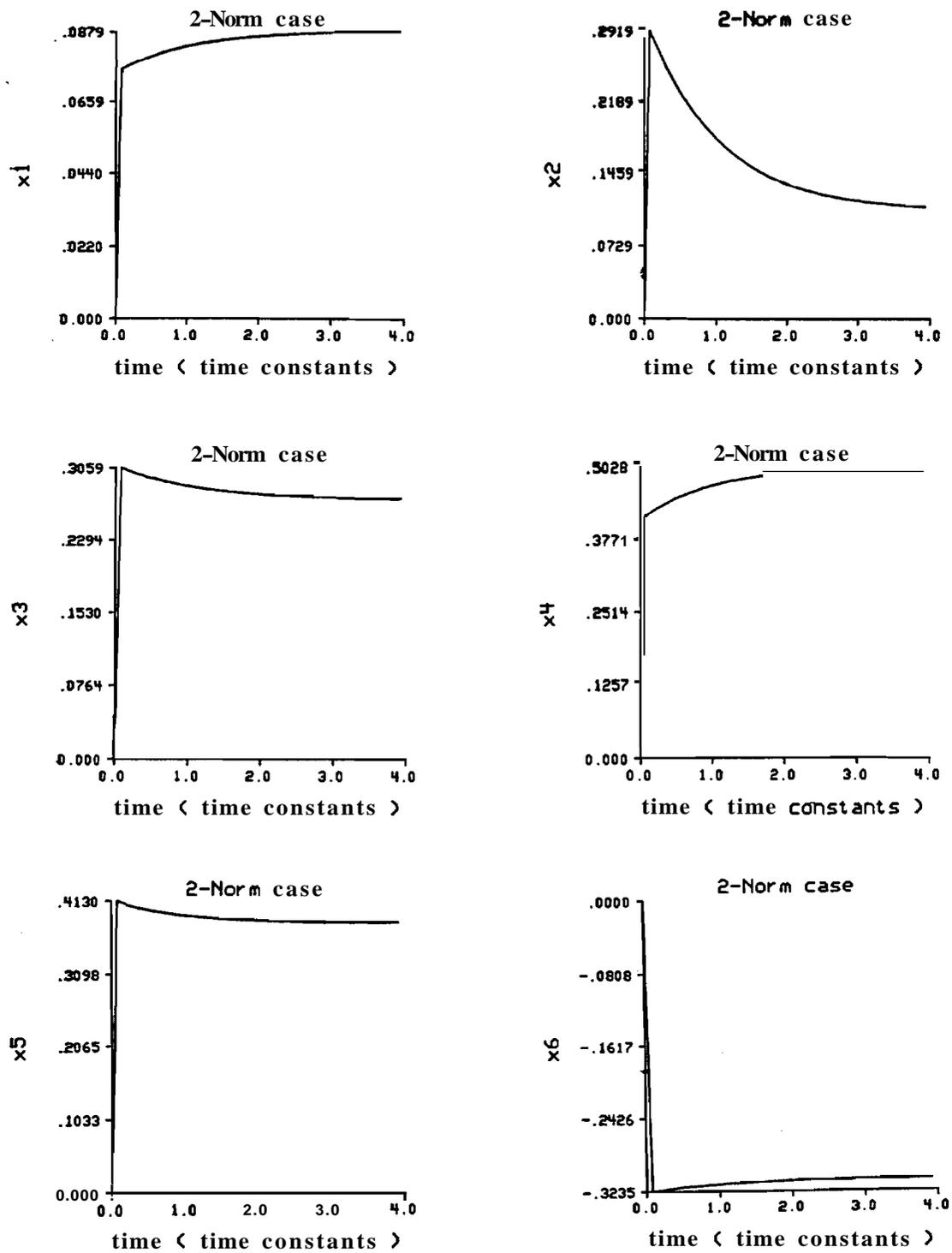


Figure 10. Trajectories corresponding to the case $p=2$.

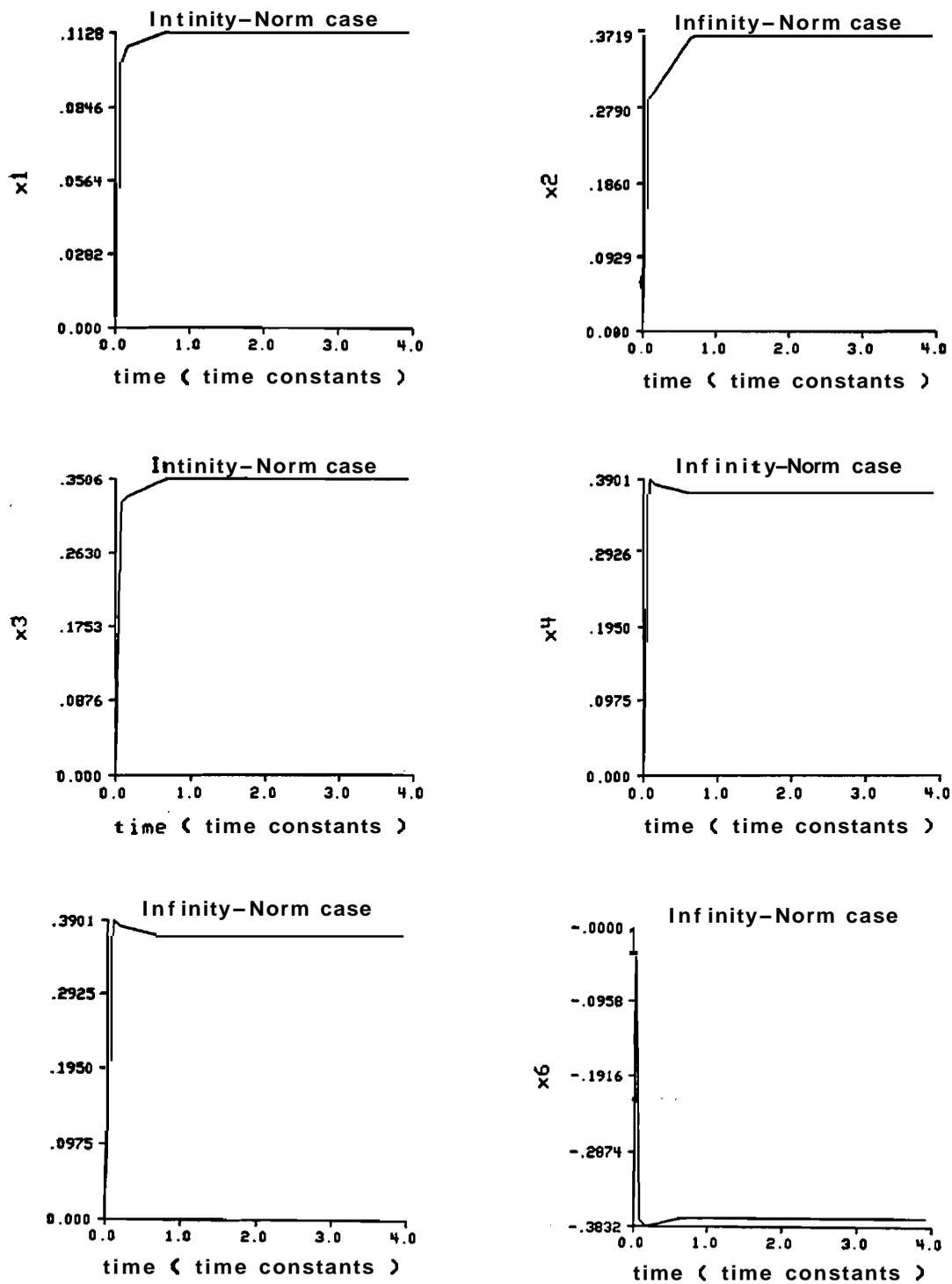


Figure 11. Trajectories corresponding to the case $p = \infty$.

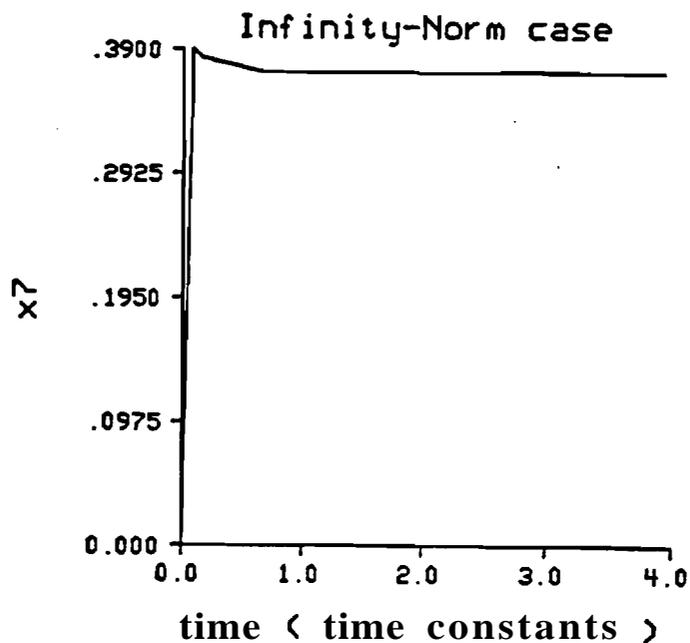


Figure 12. Trajectory of the augmented variable for the case $p=\infty$.

5. Conclusions

A general form of a network was given which can minimize a function subject to both equality and inequality constraints. An implementation was given for the case of quadratic programming with linear equality and inequality constraints. Next the minimum norm problems were introduced and it was shown how the previously introduced implementation could be used and modified to solve the various minimum norm problems of interest. The networks were then simulated on a digital computer and successfully tested on a benchmark problem.

References

Cadzow, J.A. (1971). "Algorithm for the minimum-effort problem." IEEE Trans. Automatic Control, vol. AC-16, no. 1, pp. 60-63.

Cadzow, J.A. (1973). "Functional analysis and the optimal control of linear discrete systems." Int. J. Control, vol. 17, no. 3, pp. 481-495.

Chua, L.O., and Lin, G.-N. (1984). "Nonlinear programming without computation", IEEE Trans., Circuits and Systems, vol. CAS-31, no. 2, pp. 182-188.

Cichocki, A., and Unbehauen, R. (1991). "Switched-capacitor neural networks for differential optimization," Int. J. Circuit Theory and Applications, vol. 19, no. 2, pp. 161-187.

Dennis, J.B. (1959). *Mathematical Programming and Electrical Networks* , London, England, Chapman & Hall.

Grossberg, S. (1988). "Non-linear neural networks: Principles, mechanisms, and architectures," Neural Networks, vol. 1, no. 1, pp. 17-61.

Halliday, D., and Resnick, R. (1978). *Physics*, Third Edition, J. Wiley & Sons, New York.

Kailath, T. (1980). *Linear Systems*. Englewood Cliffs, New Jersey, Prentice-Hall.

Kennedy, M.P., and Chua, L.O. (1988). "Neural networks for nonlinear programming." *IEEE Trans. Circuits and Systems*, vol. 35, no. 5, pp. 554-562.

Kolev, L. (1975). "Iterative algorithm for the minimum fuel and minimum amplitude problems for linear discrete systems." *Int. J. Control*, vol. 21, no. 5, 779-784.

LaSalle, J.P. (1986). *The Stability and Control of Discrete Processes*. New York, Springer-Verlag.

Lillo, W.E., Loh, M.H., Hui, S., and Žak, S.H. (1991). "On solving constrained optimization problems with neural networks : A penalty method approach," Technical Report TR-EE-91-43, School of EE, Purdue Univ., West Lafayette, IN.

Luenberger, D.G. (1984). *Linear and Nonlinear Programming*. Reading, Massachusetts, Addison-Wesley.

Smith, M.J.S., and Portmann, C.L. (1989). "Practical design and analysis of a simple "neural" optimization circuit", *IEEE Trans. Circuit and Systems*, vol. 36, no. 1, pp. 42-50.

Sudharsanan, S.I., and Sundareshan, M.K. (1991). "Exponential stability and a systematic synthesis of a neural network for quadratic minimization", *Neural Networks*, vol. 4, no. 5, pp. 599-613.

Tank, D.W., and Hopfield, J.J (1986). "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.
