Engineering Journal of the University of Qatar, Vol. 13, 2000, pp.

CURRENT-MODE NEURAL NETWORKS FOR SOLVING NONLINEAR PROGRAMMING PROBLEMS

Fawzi Al-Naima and Samir M. Hameed
Department of Electronic and Communications Engineering
College of Engineering/ Saddam University
P.O. Box 64040
Baghdad-Iraq

ABSTRACT

In this paper, neural networks for online solution of linear and nonlinear programming problems are presented. Current mode circuits are used in the design of networks. Transimpedance techniques based current conveyors are used for implementing the neurons. Two types of neurons are used in these networks: one being an integrator and the other being used as constraint. Many methods are suggested in this paper to implement networks weights by using current mode circuits such as Operational Transconductance Amplifiers. Network parameters are explicitly computed based upon problem specifications, to cause the network to converge to an equilibrium that represents a solution. Simulation results using Electronic workbench EDA version 5.0a-1996 shows the optimization solutions are obtained almost in real time, acceptable if they are compared with theoretical values, and are stable.

INTRODUCTION

Artificial neural networks techniques have been applied to many areas, where traditional serial processing is difficult to handle, such as data analysis, pattern and image recognition, control function, content-addressable memory, optimization and numerical analysis [1-5]. Neural networks can provide a computing model capable of exploiting the parallelism to solve a rich class of optimization problems [5]. The advantages of using a neural electronic system in applications of optimization is to provide a real time system and satisfying on-line solutions to the problems [5-7]. Moreover, many applications could be obtained from it in control, robotics, real-life problems, and others. Several methods have been proposed to implement neural networks in hardware. They can be classified as digital, analogue, hybrid, and optical [3,4]. In this work, we present a neural network that is capable of solving a large class of constrained and unconstrained optimization problems (both maximization and minimization), where the current mode circuits are used in the design of the proposed neurons. The overall system is based on formulating a dynamic gradient system. In section two the mathematical representation for unconstrained and constrained optimization will be introduced. Optimization network analysis is given in section three. The details for current mode implementation are presented in section four. Simulation results of the proposed procedure are given in section five.

MATHEMATICAL REPRESENTATION

Unconstrained Optimization

Consider the general problem of minimizing a scalar cost function $\Phi(x_1,x_2,\dots,x_n)$ without any constraints. We seek minimization methods that lead to stationary points of $\Phi(x)$, that is $\nabla\Phi(x^*)=0$. The approach for automatic minimization is to use gradient information in seeking for optimum points. Optimization in classical analogue computer was accomplished by implementing the following companion dynamic gradient system [6]:

$$c\frac{dx_i}{dt} = -\frac{\partial\Phi}{\partial x_i} \qquad i=1,2,\dots,n \tag{1}$$

Equation (1) ensures automatic minimization. The time derivative of $\Phi(x)$ can be expressed as:

$$\frac{d\Phi}{dt} = \sum_{i=1}^{n}\frac{\partial\Phi}{\partial x_i}\bullet\frac{dx_i}{dt} \tag{2}$$

It follows that:

$$\frac{d\Phi}{dt} = -\sum_{i=1}^{n}c\left(\frac{dx_i}{dt}\right)^2 \tag{3}$$

Each c is strictly positive and $(dx_i/dt)^2 \geq 0$, therefore, $d\Phi/dt < 0$.This implies that $d\Phi/dt$ is less than zero for all x in R^n except at equilibrium points.At such points $dx_i/dt = 0$, where it vanishes. Consequently, $\Phi(x)$ is a *Lyapunov* function for the systems [6,7]. This ensures that the system is completely stable.

Constrained Optimization

The consideration of the constrained optimization problem is as follows:
Minimize: $\Phi(x_1, x_2, \ldots, x_n)$

Subject to inequality constraints:

$$g_j(x_1, x_2, \ldots, x_n) \geq 0$$

Where, j=1,2,....,m, and (m and n) are two independent integers.

In this section we consider only two basic methods for solving constrained optimization: Kuhn-Tucker [8-10] theorem and the penalty multiplier theorem [6,10].

Kuhn-tucker optimality theorem

This section develops the Kuhn-Tucker necessary conditions for identifying stationary points of a nonlinear constrained problem subject to inequality constraints. The well-known Kuhn-Tucker conditions for solving a nonlinear programming problem are [9]:

$$\begin{aligned} \frac{\partial \Phi(x^*)}{\partial x_i} + \sum_{j=1}^{m} \lambda_j^* \frac{\partial g_j(x^*)}{\partial x_i} &= 0 \\ g_j(x^*) &\geq 0 \\ \lambda_j^* &\leq 0 \\ \lambda_j^* \cdot g_j(x^*) &= 0 \end{aligned} \qquad (4)$$

Where x^* is the unique solution to the problem (optimum) and λ^* is optimal Lagrange multiplier. A necessary condition for optimality is that λ_j be non-positive (nonnegative) for minimization (maximization) problems.

Penalty multiplier method

To solve a constrained problem by a steepest descent scheme, we convert it into an equivalent problem. The way to do this is to define a *pseudocost* function $\Psi(x)$ as follows:

$$\Psi(x) = \Phi(x) + \mu P(x) \tag{5}$$

Where P(x) is the penalty function, and μ is the penalty multiplier as follows:

$$P(x) = \frac{1}{2}\sum_{j=1}^{m} U(-g_j)(g_j)^2 \tag{6}$$

U(.) represents the unit step function.

Now, applying a dynamic gradient approach to equation (5), we can obtain the following set of differential equations:

$$c\frac{dx_i}{dt} = -\nabla\Phi(x) - \mu\sum_{j=1}^{m} U(-g_j)g_j(x)\nabla g_j(x) \tag{7}$$

Where c is strictly positive.

OPTIMIZATION NETWORK ANALYSIS

Hopfield first constructed the analogue circuit architectures for neural optimization network [11]. Kennedy and Chua further extended Hopfield's model for the optimum solution ensuring the stability condition [7]. Maa and Shanblatt gave a complete analysis of this model [10]. The extended equation of the Kennedy and Chua model is

$$\dot{x} = -\nabla\Phi(x) - i_j\nabla g_j(x) \tag{8}$$

$$i_j = \begin{cases} 0 & \text{if } g_j(x) > 0 \\ g_j(x)R & \text{if } g_j(x) \le 0 \end{cases} \tag{9}$$

Where R is the gain of the nonlinearity.

Equation (8) actually fulfills both the Kuhn-Tucker conditions and the penalty function method. It can be seen that equation (4) and equation (7) are the same as equation (8), since $i_j = \lambda_j = \mu U(-g_j)\, g_j$. Equation (8) is a Lyapunov function and is a completely stable ensuring the network converges to a stable equilibrium point without oscillation.

The Treatment of Equality Constraints

The problem is to minimize a cost function $\Phi(x_1,x_2,\ldots,x_n)$
Subject to:

$g_j(x_1,x_2,\ldots,x_n) \geq 0 \qquad j=1,\ldots,m$

and

$h_k(x_1,x_2,\ldots,x_n) = 0 \qquad k=1,\ldots,r$

Let the cost function be in the following quadratic form

$$\Phi(x_1,x_2,\ldots,x_n) = \frac{1}{2}\left[x^T Q\, x\right] + a^T x \tag{10}$$

and the constraints:

$$g_j(x_1,x_2,\ldots,x_n) = Bx - e \geq 0 \tag{11}$$

$$h_k(x_1,x_2,\ldots,x_n) = Dx - f = 0 \tag{12}$$

Where a and x are (n-vectors), g and e are(m-vectors), h and f are (r-vectors), B and D are (mxn), (rxn) matrices respectively and Q is (nxn) symmetric positive definite-matrix. We modify equation (12) by stipulating mutually exclusive terms for $h_k \geq 0$ and $h_k \leq 0$ [12]. That is

$$g_{m+2k-1} = h_k \quad , \quad g_{m+2k} = -h_k \tag{13}$$

This modification makes the networks to deal with equality and inequality constraints.

The Maximization Model

Consider the problem of minimizing a scalar cost function. The search may be in the gradient direction or opposite direction [8]. So equation (1) can be applied for maximization, if the negative sign is replaced by positive sign. The gradient method with a positive sign was called a steepest ascent method. Now we write the improved equation models i.e. a maximization model as follows:

$$\dot{x}=\nabla\Phi(x) \tag{14}$$

$$\dot{x}=\nabla\Phi(x)-i_j\nabla g_j(x) \tag{15}$$

Where, equation (14) solves maximization problems without any constraints and equation (15) can be applied for maximization problems with bound constraints.

The Network Formulation

Equation (8) can solve both constrained and unconstrained nonlinear programming problems. Figure 1 shows the simplified diagram of equation (8). This figure can be applied to solve constrained optimization problems with equality and inequality (minimization and maximization). There are two types of neurons used in this network: one type being an integrator, and the other type being used as constraint qualifier. Figure 1 is valid for unconstrained problems if the constraint blocks disappeared. The integrator in this figure is continuously operated until the energy function is minimized, so through the feedback terminal x still changing and the energy function decreasing. When ($dx/dt \rightarrow 0$) the energy function is minimum and x is a target. The connection patterns between neurons can be realized as resistive connection and the entire matrix corresponds to conductance value. So the weight value equals the inverse of resistance value. There are different methods suggested to implement these weights [3]. We try to implement these weights by using current mode techniques.

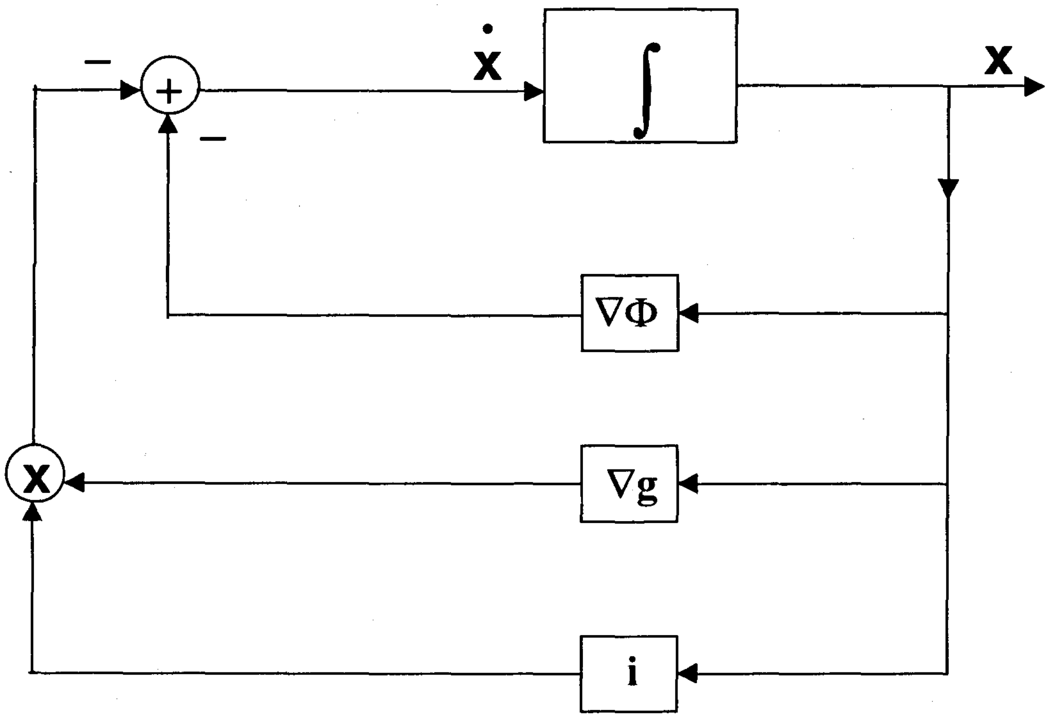


Fig. 1. A simplified diagram for a constrained optimization

CURRENT MODE IMPLEMENTATION

Current Mode circuits

In these circuits the current rather than the voltage is used as the active variable. The current mode signal processing circuits have recently demonstrated many advantages over their voltage mode counterparts including increased bandwidth, higher dynamic range, and better suitability for operation in reduced supply environments. In addition, the current mode processing often leads to simpler circuitry and lower power consumption (3,4, 13,14).

The Current Conveyor

The current conveyor is essentially a voltage/current mode hybrid circuit. It is a four terminal device which when arranged with other electronic elements in specific circuit configurations can perform many useful analogue signal processing functions [13]. Current conveyors can be classified into three types: first, second and third generation current conveyors respectively designated symbolically as CCI, CCII, and CCIII respectively [13,15]. These conveyors ideally can be described using the following matrix relation:

$$\begin{bmatrix} i_y \\ v_x \\ i_z \end{bmatrix} = \begin{bmatrix} 0 & a & 0 \\ 1 & 0 & 0 \\ 0 & b & 0 \end{bmatrix} \begin{bmatrix} v_y \\ i_x \\ v_z \end{bmatrix} \tag{16}$$

b characterizes their current transfer from x to z. For b positive, the circuit is a positive transfer conveyor. This becomes a conveyor with a negative transfer when b is a negative. For a=1, the circuit is CCI. If a=0, then the circuit is CCII. With a=-1 the circuit is CCIII. Figure 2 shows the ideal model of CCII. In 1991 commercial current conveyor (either positive current conveyor AD844 and cascade negative current conveyor AD844S) chips were manufactured by Analogue Devices, which show a good performance over a large bandwidth of frequency range [16]. The models of non-ideal second-generation current conveyors are shown in Figure 3 [16].

The Design of Networks

The problem is to design a network of Figure 1 using current conveyor components. This network consists of two types of neurons: constraint and integrator. Both neurons were designed by using a *Transimpedance* circuit based on commercial current conveyors (AD844 and AD844S). A Transimpedance circuit has a low input – low output impedance [17].

Figure 4 shows the nonlinear constraint model. These conveyors are operating as a Transresistance circuits. The input current will be fed into x_1 and then it will be transferred to z_1, during this transferring the phase of current is inverted through the current controlled–current source CCCS. The current passes through Rz_1, R and Ry_2. The voltage across (Rz_1//R//Ry_2) is Vy_2, and through voltage controlled-voltage source VCVS of the second current conveyor, the voltage passes to x_2 terminal. The controlled voltage of VCVS is Vy_2, and Vx_2 is the source. A switch is connected with the output of the second CCII to perform the constraints nonlinearity. The controlled and input voltage of a switch is the same and is Vx_2. The controlled voltage of the switch is connected with the negative terminal to pass only signals having a negative polarity to achieve

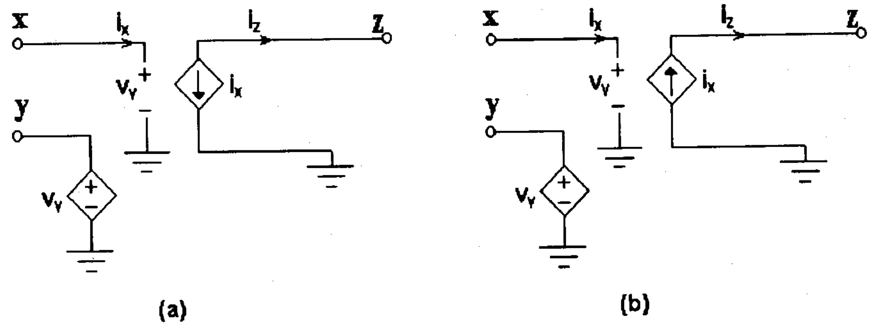
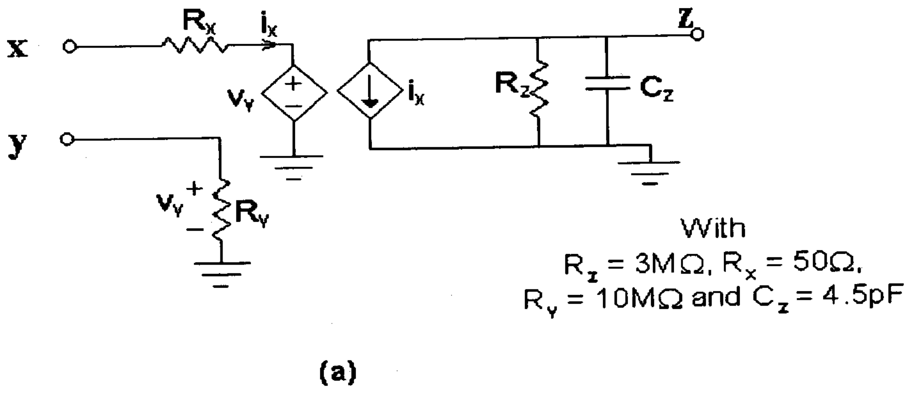


Fig. 2. models of ideal second generation current conveyor
(a) CCII (b) –CCII



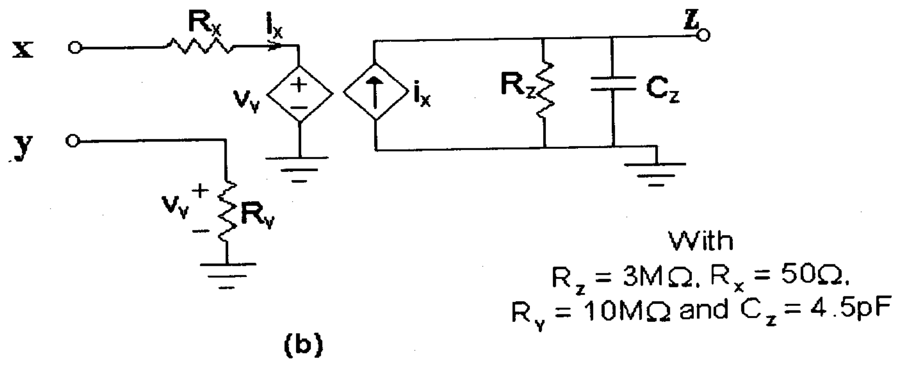
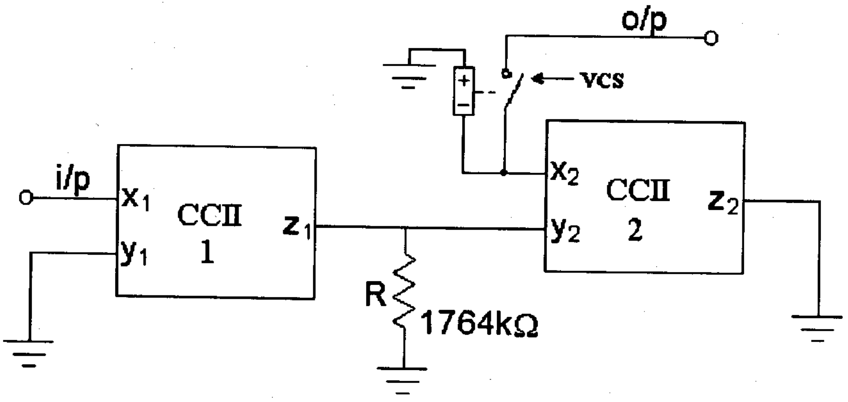


Fig. 3. Models of non-ideal current conveyors
(a) CCII (b) –CCII



`Fig.4. A constraint neuron

equation (17). The ground connector was connected with a positive terminal of controlled voltage of a switch, to make the turnoff voltage of a switch at zero volts. Thus, the function of switch is to perform a constraint nonlinearity. The output voltage of a switch is:

$$\begin{cases} -i_{x_1}(R_{x1} // R // R_{y2}) & \text{if } i_{x_1} \geq 0 \\ 0 & \text{if } i_{x_1} < 0 \end{cases} \qquad (17)$$

Let R= 1764 MΩ, ∴ $Rz_1//R//Ry_2$ = 1MΩ

The equivalent resistance is 1MΩ; which is a good value when tested in simulation and it gives good results.

Figure 4 is valid if a constraint equation is in the Bx-e>0 form. If a constraint is in the form g(x)= Bx-e<0 , the switching scheme must be modified in Figure 4. There are two ways to accomplish g (x)=-Bx+e>0:

1) By achieving the same equation and connect with output of a constraint neuron an inverter circuit.

2) By connecting –CCII instead of CCII numbered 1 and replacing the switching scheme.

Note in case of inequality constraints, we need one constraint neuron for every constraint equation. In case of equality constraint, we use two neurons as stated earlier, the first operates as g (x) >0 and the second operates as g (x)<0.

Figure 5 shows an integrator neuron. This circuit operates as a *Transcapacitance* circuit. The current feeds to x_1 and then transfers with inverting the phase to z_1, and then passing through the capacitor. The voltage from x_2 is the integration of the input current through x_1, which by neglecting the parasitic resistance (10MΩ//3MΩ) becomes

$$V = -\frac{1}{C}\int I(t)dt \tag{18}$$

The function of integrator neuron is to accomplish equation (1) i.e.

$$C\frac{dv_i}{dt} = -\sum \text{Currents} \tag{19}$$

The output voltage of integrator neuron is the variable of the optimization problem. The quantity of integrator neuron depends on the number of variables on the optimization problems. For example, if the number of variables is three, then the number of the neurons are three i.e. six current conveyors.

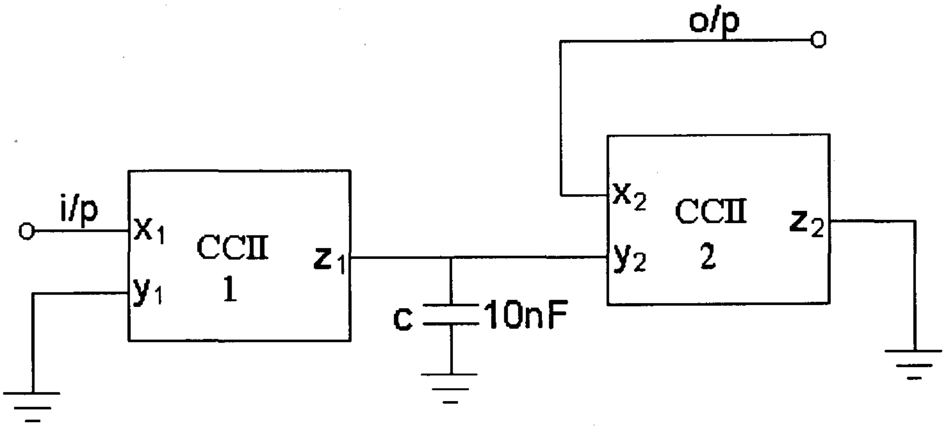


Fig. 5. An integrator neuron

The low output impedance of the second current conveyor of both neurons makes this chip work as a voltage buffer. The voltage buffer appears to be one of the most useful building blocks in analogue electronics. High performance voltage buffers are needed in applications of current conveyor to achieve the desired final parameters in the design. In addition, it is suitable for driving heavy load.

Suggestions of Weights Implementation

As stated before, we presented how the connection patterns were directly implemented using resistive elements depending on the value of problem parameters. The resistive elements tend to be nonlinear and occupy too much area to make practical VLSI implementations [7]. Many methods were proposed by using analogue, digital, or both techniques. In this section we present the current- mode implementation to these weights by using an Operational Transconductance Amplifier OTA. An OTA circuit is one of the current-mode circuits, it has five terminals as follows: inverting voltage V^-, non-inverting voltage V^+, output current I_o, biasing current, and ground terminals. It is a voltage controlled-current source VCCS. The OTA circuits use the MOS transistor characteristic equations to produce an output current that is proportional to the difference of two voltages. The output current can be described as follow [13,14], see Figure 6:

$$I = g_m (V^+ - V^-) \qquad (20)$$

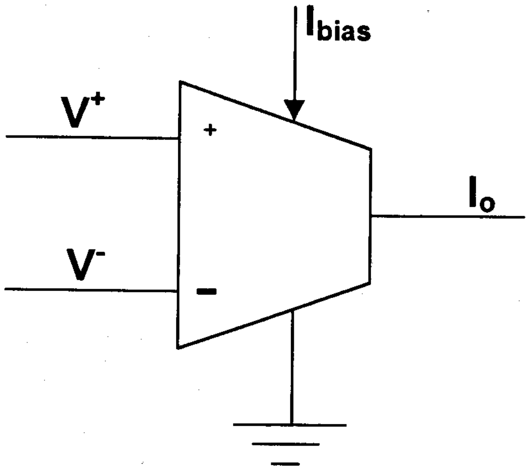


Fig. 6. An OTA symbol

Where g_m is a transconductance gain which depends on a bias current and the inner circuit parameter. The weight value is the same g_m value.

The neural network employing Transconductors offers several advantages as follows:

1) Increased dynamic range of the signals when the MOS transistors are operated from the weak inversion all the way to strong inversion.

2) Increased frequency range of operation due to the use of the low-impedance internal nodes minimizing the capacitance charging and discharging.

3) With this type of weights, we can obtain negative valued weights as we described depending on equation (20).

4) It has high input impedance.

5) It can control the value of the weights by using a floating gate transistor with the OTA circuit [18] or by using external resistance [19] or by controlling the value of bias current [3].

6) In reference [20], there is a technique, which implements multiple input OTA circuit; they modeled each weight connection with only two MOSFET transistors.

SIMULATION EXAMPLES

Example 1. [7]

Minimize

$$\Phi(x_1,x_2,x_3) = 0.4x_1 + \frac{1}{2}(5x_1^2 + 8x_2^2 + 4x_3^2) - 3x_1x_2 - 3x_2x_3$$

Subject to

$$g_1(x_1,x_2,x_3) = x_1 + x_2 + x_3 \geq 1$$

$$g_{2,3,4}(x_1,x_2,x_3) = x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$$

In this example, we expect to understand the performance of proposed neurons for quadratic programming problem, see Figure 7. This problem was described by Kennedy and Chua [7].

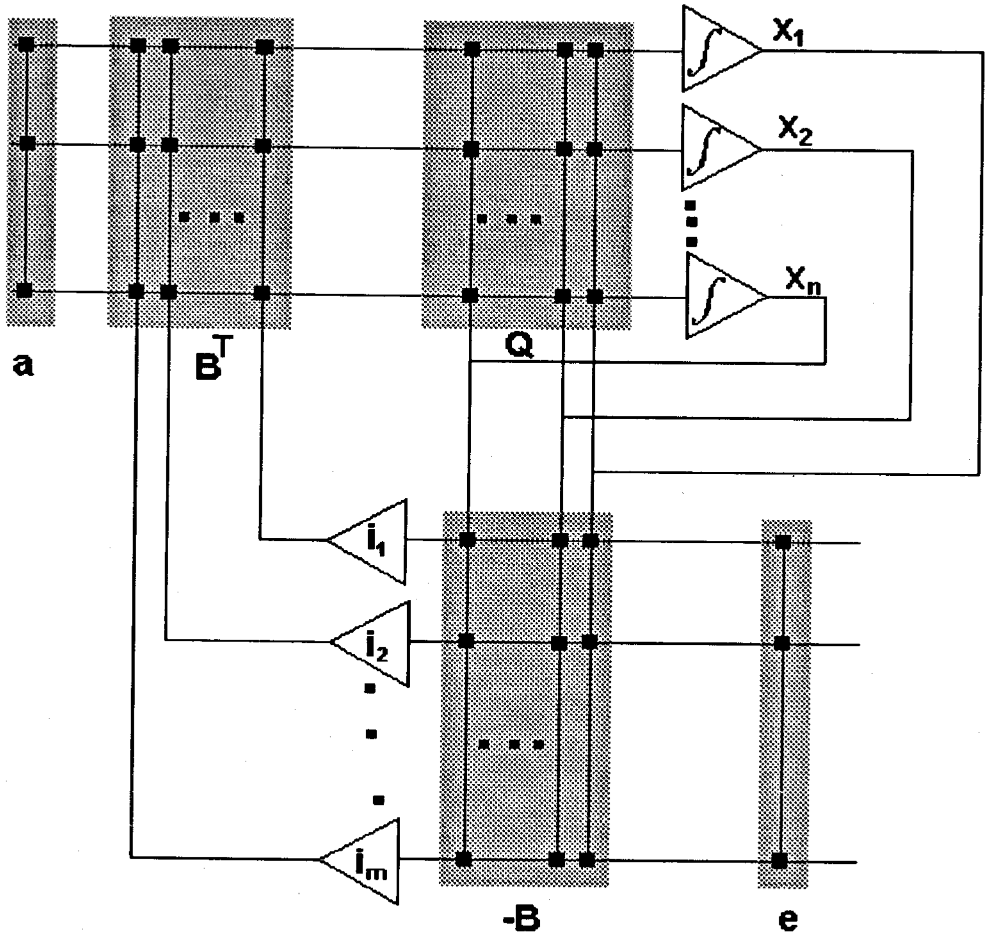


Fig. 7. Network formulation of quadratic problems

To map this circuit we find ∇Φ(x) as follows:

$$\frac{\partial\Phi(x_1,x_2,x_3)}{\partial x_1} = 0.4 + 5x_1 - 3x_2$$

$$\frac{\partial\Phi(x_1,x_2,x_3)}{\partial x_2} = 8x_2 - 3x_1 - 3x_3$$

$$\frac{\partial\Phi(x_1,x_2,x_3)}{\partial x_3} = 4x_3 - 3x_2$$

From this information we can find [Q] and [a] as follows:

$$Q = \begin{bmatrix} 5 & -3 & 0 \\ -3 & 8 & -3 \\ 0 & -3 & 4 \end{bmatrix} \qquad a = \begin{bmatrix} 0.4 \\ 0 \\ 0 \end{bmatrix}$$

We find [B] and [e] directly from constraint equation. [B^T] is the transposed of [B], but for nonlinear constraints we must evaluate ∇g.

Vector [e] and [a] are connected with fixed voltage source, which is taken as 1V. Seven neurons are used in this example, three for integrators and four for constraint amplifiers. The results of this example are shown in Table 1. See Figure 8 to further understand the transient response for different outputs in OP-AMP and Practical CC. In this figure the current conveyor reaches the optimum value (steady state) at time interval smaller than OP-AMP and the accuracy for CC is better than OP-AMP due to the properties of current-mode circuit. Figure 9 illustrates the error graph of each output. Note these errors in Figure 9 are determined as (theoretical value - simulated value).

Table 1 Results of Example 5.1

	Theoretical Values	OP-AMP (Kennedy and Chua model)	Ideal Current Conveyor	Current Conveyor (practical)
x_1	0.252	0.2615	0.2507	0.2498
x_2	0.3328	0.3309	0.3323	0.3314
x_3	0.415	0.3985	0.4133	0.4122
% Error of x_1	—	3.77	0.516	0.873
% Error of x_2	—	0.571	0.1502	0.4207
% Error of x_3	—	3.976	0.4096	0.6747
% Avg. error	—	2.772	0.3586	0.6561
Total analysis time in seconds	—	56	1.2	1.3

Example 2. [8]

Minimize

$$\Phi(x) = x_1^4 + x_2^2$$

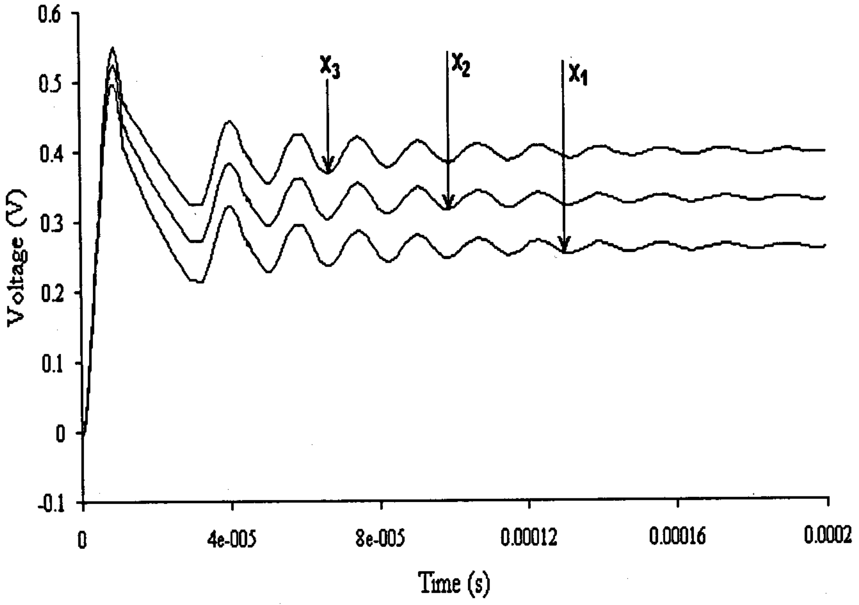
Subject to

$$g_1(x) = x_1 x_2 > 8$$

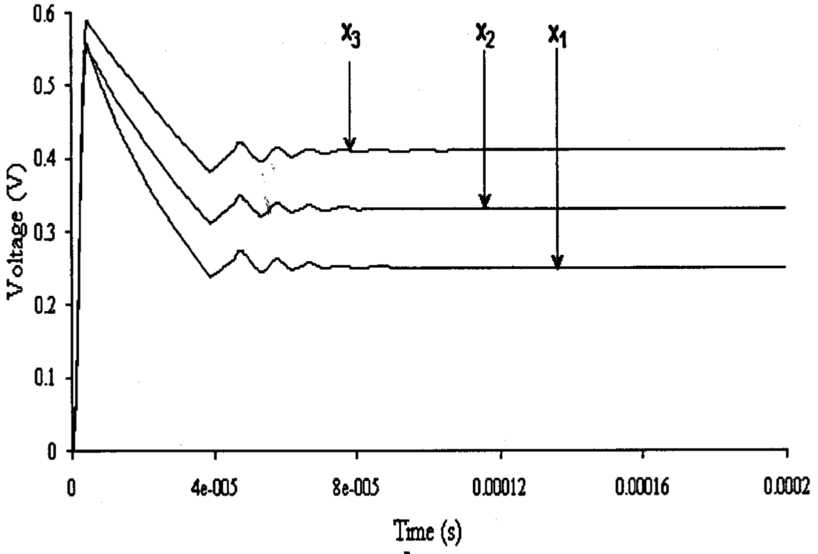
$$g_{2,3}(x) = x_1 > 0, x_2 > 0$$

To map this problem we find $\nabla\Phi$ and ∇g as follows:

$$\nabla\Phi(x) = \begin{bmatrix} 4x_1^3 \\ 2x_2 \end{bmatrix} \quad , \text{ and } \quad \nabla g(x) = \begin{bmatrix} x_2 & x_1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

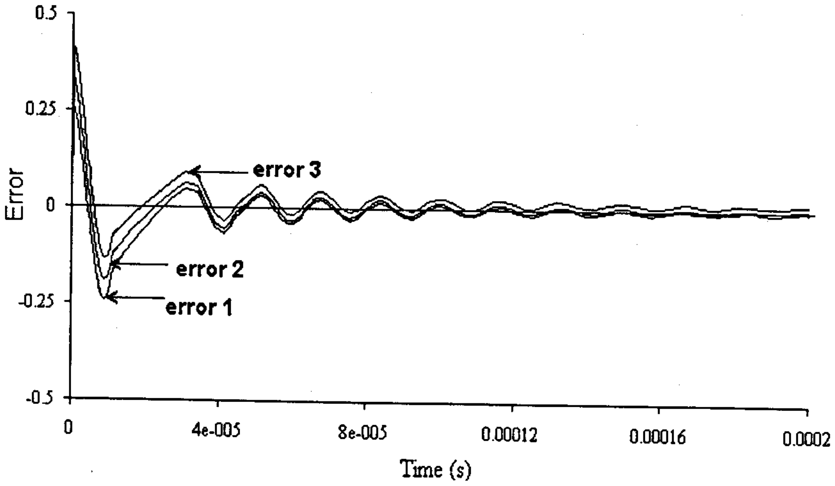


a

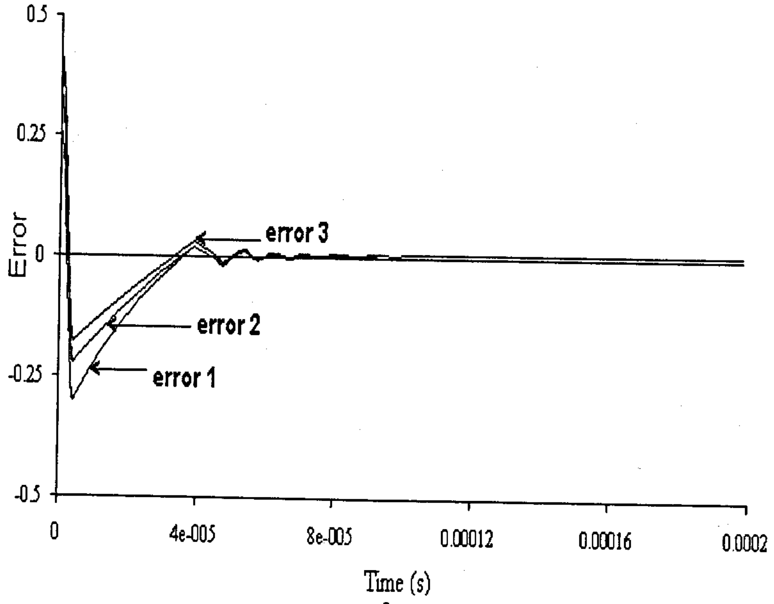


b

Fig. 8. Transient response for different outputs for example 5.1 (a) OP-AMP (b) CC



a



b

Fig .9. Error curves for examples 5.1 (a) OP-AMP (b) CC

By applying the principle that was stated in Figure 1, we connect two multipliers to obtain x^3, and two multiplier to achieve (i .∇g) block, and one multiplier to produce the first constraint (g_1). To map the constrained elements in both constraint and integrator neurons, first, we find B^T from ∇g and B from constraint equations (g_2 & g_3). Finally we connect the three multiplier (i_1x_1, i_1x_2 and $-x_1x_2$) to accomplish constraint number g_1 as shown in Figure 10. The results of this example are shown in Table 2.

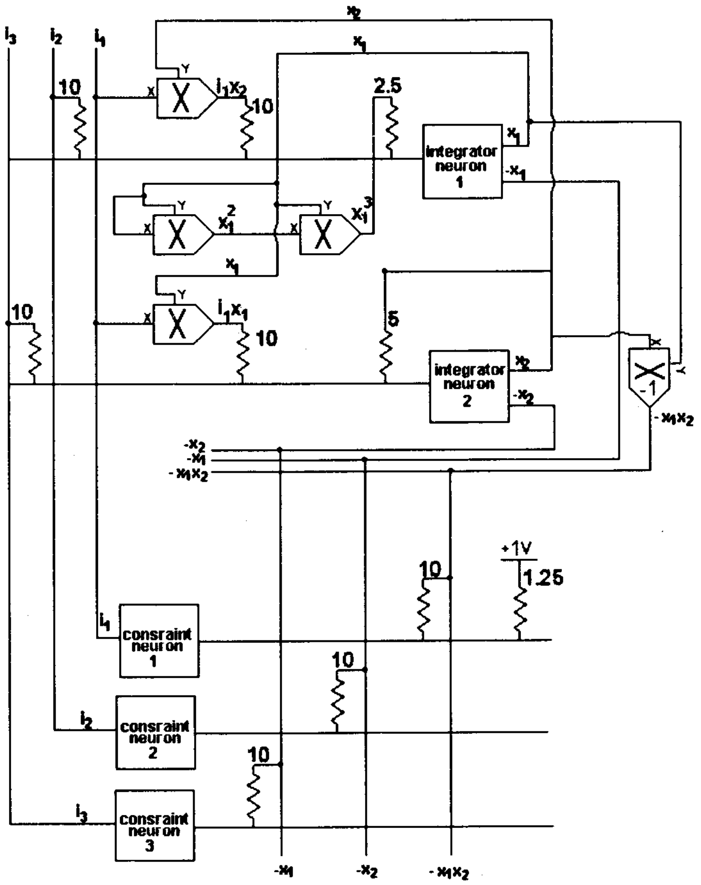


Fig. 10. The mapping of example 5.2

Table 2 Results of Example 5.2

	Theoretical Values	OP-AMP	Ideal Current Conveyor	Current Conveyor (practical)
x_1	1.7818	1.787	1.779	1.778
x_2	4.4898	4.447	4.472	4.469
% Error of x_1	—	0.2918	0.1571	0.2133
% Error of x_2	—	0.9533	0.3965	0.4633
% Avg. error	—	0.6225	0.2768	0.3383
Total analysis time in seconds	—	28.25	1	1.25

CONCLUSION

From the point of view of energy function, we can map the nonlinear programming problem onto the current mode circuits, and successfully simulate the on-line performance. The proposed neurons exhibit the modularity as well as simple analogue processor. The current conveyor component were tested and showed good performance (both ideal and non-ideal commercial current conveyor type AD844). A transimpedance technique was used in designing of the proposed neurons (Transcapacitance for integrator neuron and Transresistance technique for constrained neuron which are based on current conveyor components). With these proposition we can obtain good results and small total absolute relative error with gain in simulation time. The suggested weights have good properties such as producing negative weights, programmability, using MOS transistor techniques and others. All these properties are regarded good advantages in VLSI implementation.

The proposed networks are valid for both linear and nonlinear programming problems, constrained and unconstrained, quadratic or higher degree optimization.

It is shown that the equilibrium of the network is asymptotically stable and approximating the optimum of cost function (minimum or maximum).

REFERENCES

1. **Lippmnn R.P., 1987** " An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, pp.4-22.

2. **Fausett L., 1994** Fundamentals of Neural Networks, Prentice-Hall Inc., New Jersey, USA.

3. **Zurada J.M., 1996** Introduction to Artificial Neural Systems, Jaico Publishing House, India.

4. **El-Masry E.I., 1997** " Implementations of Artificial Neural Networks Using Current-Mode Pulse Width Modulation", IEEE Trans. Neural Networks, Vol.8, No.5, pp.532-547.

5. **Tagliarini G.A, 1991** " Optimization Neural Networks", IEEE Trans. Computers, Vol.40, No.12, pp.1347-1358.

6. **Rodriguez-Vazquez, A., 1990** " Nonlinear Switched-Capacitor 'Neural' Networks for Optimization Problems", IEEE Trans. Circuits Sys., Vol.37, No.3, pp.384-397.

7. **Kennedy, M.P.** and **Chua, L.O., 1988** " Neural Networks for Nonlinear Programming", IEEE Trans. Circuits Sys, Vol.35, No.5, pp.554-562.

8. **Bunday, B.D., 1985** Basic Optimization Methods, Edward Arnold, London, Uk.

9. **Huertas, J.L., 1987** " Canonical Nonlinear Programming Circuits", Int. J. Circuit Theory Appl., Vol.15, No.1, pp.71-77.

10. **Maa C.,** and **Shanblatt M.A., 1992** " A Two-Phase Optimization Neural Networks", IEEE Trans. Neural Networks, Vol.3, No.6, pp.1003-1009.

11. **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 Sys, Vol.33, No.5, pp.533-544.

12. **Maa, C.** and **Shanblatt M.A., 1992** " Linear and Quadratic Programming Neural Network Analysis", IEEE Trans. Neural Networks, Vol.3, No.6, pp.580-594.

13. **Toumazou C., 1993,** Analogue IC Design: the Current Mode Approach, Peter Peregrines LTD., London, UK.

14. **Sion, R.S.,1991** Analogue-Digital ASICs: Circuit Techniques, Design Tools and Applications, Peter Peregrines LTD., London, UK.

15. **Fabre, A.1995** " Third Generation Current Conveyor: a New Helpful Active Element", Electronics Letters, Vol.31, No.5.

16. **Svoboda, J.A.1994** " Comparison of RC op.amp and RC Current Conveyor Filters", Int.J.Electronics, Vol.76, No.4, pp.615-626.

17. **Chen, J.J.1992** " Operational Transresistance Amplifier Using CMOS Technology", Electronics Letters, Vol. 28, No.22.

18. **Lee, B.W. 1991**" Analogue Floating-Gate Synapses for General-Purpose VLSI Neural Computation", IEEE Trans.Circuits.Sys., Vol.38, No.6, pp.654-658

19. **Cichocki A.** and **Unbehauen, R.1992** " Neural Networks for Solving Systems of Linear Equations –Part II: Minimax and Least Absolute Value Problems", IEEE Trans. Circuits.Sys II: Analogue and Digital Signal Processing, Vol.39, No.9, pp.619-633.

20. **Reed, R.D.** and **Getger, R.L.1989** " A Multiple-Input OTA Circuit For Neural Networks ", IEEE Trans.Circuits.Sys., Vol.36, No.5, pp.767-769.