Discrete neural network for solving general quadratic programming problems

VALERI MLADENOV
[1]Department of Theoretical Electrical Engineering, Faculty of Automatics, Technical University of Sofia
8, Kliment Ohridski St, Sofia-1000, BULGARIA

Abstract: - Quadratic programming problems are widespread class of nonlinear programming problems with many practical applications. The case of inequality constraints have been considered in a previous author's paper. Later on an extension of these results for the case of inequality and equality constraints has been proposed. Based on equivalent formulation of Kuhn-Tucker conditions, a new neural network for solving the general quadratic programming problems, for the case of both inequality and equality constraints has been presented. In this contribution a discrete version of this network is proposed. Two theorems for global stability and convergence of this network are given as well. The presented network has lower complexity and it is suitable for FPGA implementations. Simulation results based on SIMULINK® models are given and compared.

Key-Words: - Quadratic programming problems, Kuhn-Tucker conditions, Neural networks, Cellular neural networks

1 Introduction

The most important advantages of using NN for solving constrained optimization problems over the traditional methods are:

– The structure of a neural network can be implemented efficiently using VLSI or optical technologies;
– Neural networks can solve many optimization problems with much faster convergence rate and can overcome singular problems.

Thus the neural networks approach to optimization problems have been received considerable attention in recent years [1]. In 1984 Chua and Lin [2] developed the canonical non-linear programming circuit, using the Kuhn-Tucker conditions from mathematical programming theory [7]. Some important neural networks for solving general non-linear programming problems are presented in [3], [4]. The authors implicitly utilize the penalty function method [1], [4], [7] whereby a constrained optimization problem is approximated by an unconstrained optimization problem. The neural network models contain penalty parameter, and hence the true solution is obtained when the penalty parameter is infinite. Because of this the network generates an approximate solution only. The so-called primal-dual neural network [5], [6] has no penalty parameter and is capable of finding the exact solution. However, the size of the network is very high and it has two-layer structure.

The computing time required for a solution of constrained optimization problems and in particular quadratic programming problem is greatly dependent on the dimension of the problem. The situation is worse in the case of singularities.

Quadratic programming problems are widespread class of nonlinear programming problems with many practical applications. The case of inequality constraints have been considered in [8]. An extension of these results for the case of inequality and equality constraints is given in [9]. Based on equivalent formulation of Kuhn-Tucker conditions, a new neural network for solving the general quadratic programming problems, for the case of both inequality and equality constraints, is presented. In this paper a discrete version of this network is proposed. Two theorems for global stability and convergence of this network are given as well. The presented network has lower complexity and it is suitable for FPGA implementations.

The paper is outlined as follows. In the next section we consider the continuous neural network for solving quadratic programming problems with presence of both inequality and equality constraints. In section 3 we propose a discrete version of the continuous neural network from section 2. In Section 4 we present simulation results for both neural networks and we end up with some conclusions in Section 5.

2 Continuous neural network for solving general quadratic programming problem

We consider the following general quadratic programming problem

$$\min f(x) = \frac{1}{2} x^T P x + q^T x + r$$
$$Gx \geq h \tag{1}$$
$$Ax = b$$

where

$P \in R^{nxn}$ is positive definite matrix, $G \in R^{mxn}$, $A \in R^{pxn}$, $q \in R^n$, $r \in R$, $x \in R^n$

From Kuhn-Tucker conditions [1], [7] it follows that x* is a solution of the above problem if there exists $\lambda \in R^m$ and $\nu \in R^p$ such that

$$Px + q - G^T\lambda - A^T\nu = 0 \tag{2}$$

and

$$(Gx)_i = h_i \text{ if } \lambda_i > 0 \tag{3a}$$
$$(Gx)_i \geq h_i \text{ if } \lambda_i = 0 \tag{3b}$$
$$Ax = b \tag{3c}$$

where *(.)i* is the *i*-th element of the corresponding vector.

The above formulation is equivalent to the following equation:

$$Gx = F(Gx - \lambda),\ \lambda \geq 0 \tag{4}$$

where

$$F(y) = \left(F(y_1), ..., F(y_m)\right)^T \tag{5}$$

and

$$F(y_i) = \begin{cases} h_i, & y_i < h_i \\ y_i, & y_i \geq h_i \end{cases} \tag{6}$$

Thus, x* is a solution of the original problem if and only if there exists $\lambda \in R^m$ such that

$$Px + q - G^T\lambda - A^T\nu = 0 \tag{7a}$$
$$Gx = F(Gx - \lambda) \tag{7b}$$
$$Ax = b \tag{7c}$$

Because P is nonsingular from (7a)

$$x = P^{-1}\left(G^T\lambda + A^T\nu - q\right) \tag{8a}$$

and thus

$$Gx = F(Gx - \lambda) \tag{8b}$$
$$AP^{-1}\left(G^T\lambda + A^T\nu - q\right) = b \tag{8c}$$

From (8c)

$$AP^{-1}G^T\lambda + AP^{-1}A^T\nu - AP^{-1}q = b \tag{9a}$$
$$AP^{-1}A^T\nu = b - AP^{-1}G^T\lambda + AP^{-1}q \tag{9b}$$

Therefore

$$\nu = \left(AP^{-1}A^T\right)^{-1}\left(b - AP^{-1}G^T\lambda + AP^{-1}q\right) \tag{9c}$$

Substituting (9c) in (7a) we get

$$\begin{aligned} &Px + q - G^T\lambda - A^T\left(AP^{-1}A^T\right)^{-1}b + \\ &A^T\left(AP^{-1}A^T\right)^{-1}AP^{-1}G^T\lambda - A^T\left(AP^{-1}A^T\right)^{-1}AP^{-1}q = 0 \end{aligned} \tag{10}$$

Let

$$Q = -\left(A^T\left(AP^{-1}A^T\right)^{-1}AP^{-1}G^T - G^T\right)^T \tag{11a}$$
$$R = \left(q - A^T\left(AP^{-1}A^T\right)^{-1}AP^{-1}q - A^T\left(AP^{-1}A^T\right)^{-1}b\right) \tag{11b}$$

Then x* is a solution of the original problem if and only if there exists $\lambda \in R^m$ such that

$$Px - Q^T\lambda + R = 0 \tag{12a}$$
$$Gx = F(Gx - \lambda) \tag{12b}$$

Hence

$$x = P^{-1}\left(Q^T\lambda - R\right) \tag{13a}$$
$$GP^{-1}\left(Q^T\lambda - R\right) = F\left(GP^{-1}\left(Q^T\lambda - R\right) - \lambda\right) \tag{13b}$$
$$GP^{-1}Q^T\lambda - GP^{-1}R = F\left(GP^{-1}Q^T\lambda - GP^{-1}R - \lambda\right) \tag{13c}$$

Let $W = GP^{-1}Q^T$ and $V = GP^{-1}R$, then x^* is a solution of the original problem if and only if there exists $\lambda \in R^m$ such that

$$W\lambda - V = F(W\lambda - V - \lambda) \tag{14a}$$
$$W\lambda - F(W\lambda - V - \lambda) = V \tag{14b}$$

where the vector function F is given by (6).

Based on the above we propose a continuous Neural Network for solving the original problem. Its state variables are defined by the following dynamic system:

$$\tau\frac{d\lambda}{dt} = F(W\lambda - V - \lambda) - W\lambda + V \tag{15}$$

where $W = GP^{-1}Q^T$, $V = GP^{-1}R$, $\lambda \in R^m$ and $\tau = 1/\mu$ is a diagonal matrix with time constants.

This system can be easily realized by a recurrent neural network with a single layer structure (Figure 1). In fact this is a continuous time neural network and its circuit realization consists of *m* integrators and m^2+2m summers. To get the final solution with respect to *x* we use the system architecture given in Figure 1, which includes the proposed neural network.

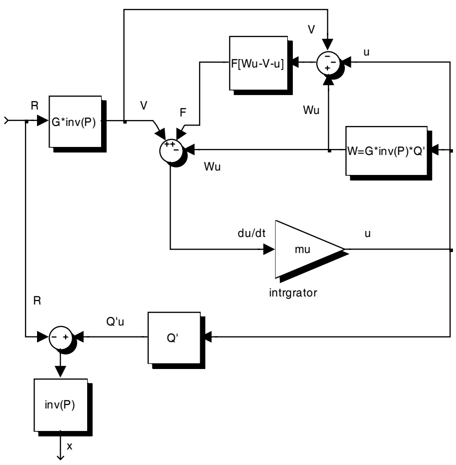

Fig.1: The architecture of the continuous time neural network presented in [9]

The proposed neural network for the case *m=2* is given in Figure 2. To get the solution with respect to *x* one has to use liner block that realizes 13(a).

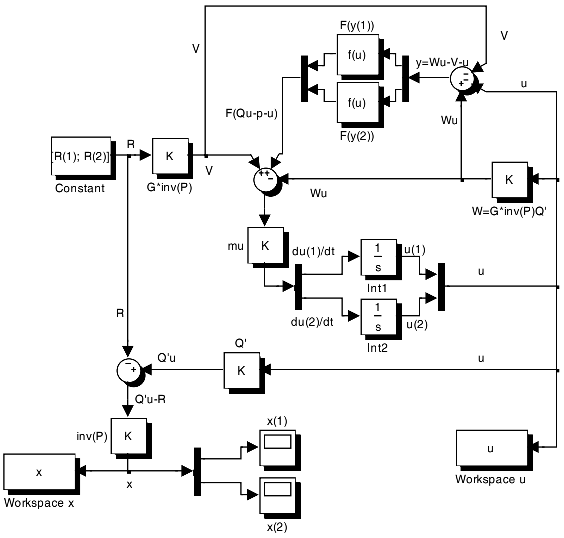

Fig.2: The continuous time neural network from [9] for the case m=2

3 Discrete neural network for solving general quadratic programming problem

The discrete time neural networks are extension of their continuous time counterparts because of the availability of design tools and the compatibility with computers and other digital devices [1]. A discrete version for solving equation (14) is given by

$$\lambda^{(k+1)} = \lambda^{(k)} + T\left[F(W\lambda^{(k)} - V - \lambda^{(k)}) - Wu^{(k)} + V\right]dt \quad (16)$$

where

$W = GP^{-1}Q^T$, $V = GP^{-1}R$, $\lambda^{(k+1)}, \lambda^{(k)} \in R^m$, $T = I + W$

and dt is the step size.

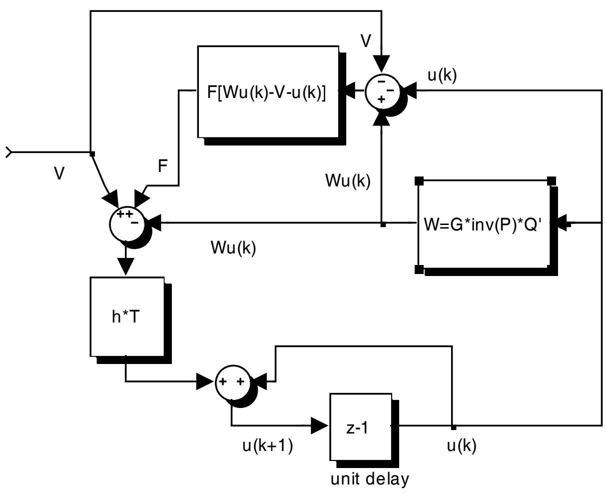

Fig.3: The architecture of the proposed discrete time neural network

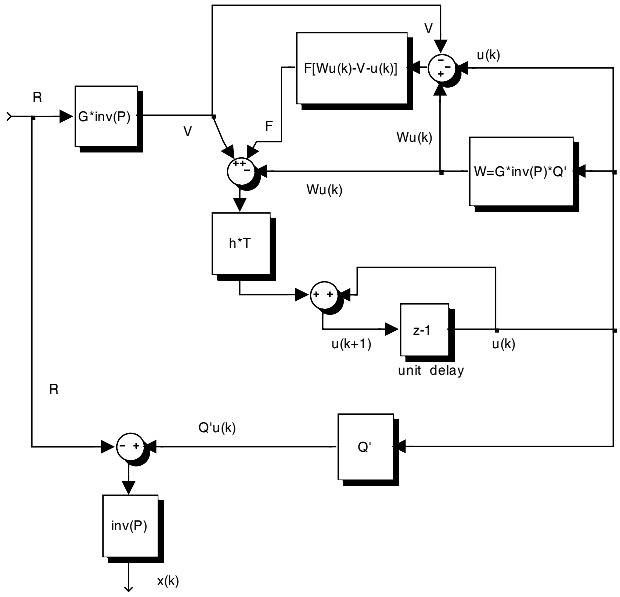

Fig.4: The discrete system for solving the general quadratic programming problem (1)

This system could be realized with a discrete time neural network given in Figure 3. This architecture is similar to the architecture of the continuous neural network given in Figure 1, but instead of integrators here we have *m* time delays. The system architecture for obtaining the solution of the original problem (1) is given in Figure 4.

The proposed discrete neural network for the case *m=2* is given in Figure 5. It is a single layer structure that contains *m=2* unit delay elments.

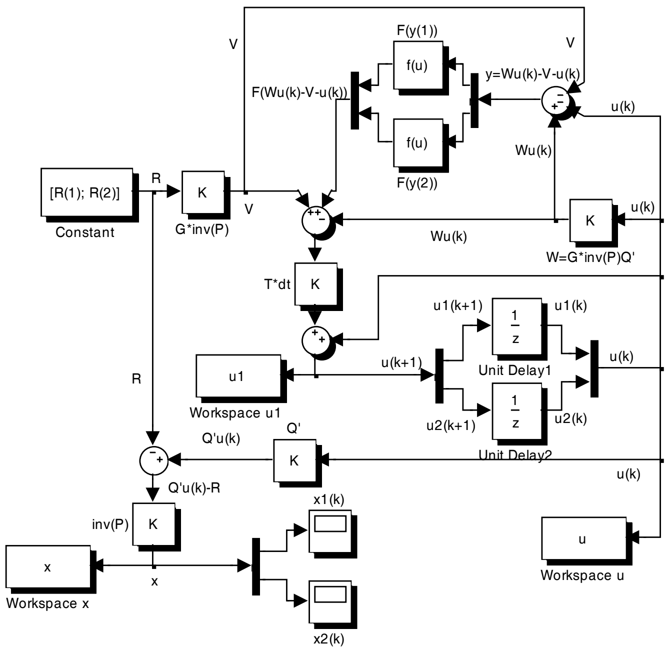

Fig.5: The proposed discrete time neural network for the case m=2

The proposed structure can be viewed as a common model of discrete cellular neural network for solving general quadratic programming problem.
The following theorems give conditions for global stability and convergence of the proposed neural network model.
<u>Theorem 1:</u> Given any initial point, there exists a unique solution of (13). The equilibrium point of (13) correspond to a solution of (1) and it is unique when *rank (A)=m* and P is positive definite.
Proof: See [8], [9].

<u>Theorem 2:</u> If $h < \dfrac{2}{\|T\|^2}$ the discrete sequence $\{\lambda^{(k)}\}$ generated by the discrete time neural network (13) is globally convergent to a solution of (1).
Proof: See [8], [9].

4 Simulation results

In this section we present simulation results for the Neural Network architectures considered. We consider the following simple example:

$$\min_{x_1,x_2} f(x) = 2x_1^2 + x_2^2 \tag{17a}$$

subject to:

$$\left|\begin{array}{l} g_1(x) = -x_1 + x_2 - 2 \ge 0 \\ g_2(x) = x_1 + x_2 - 2 \ge 0 \\ a(x) = x_1 + x_2 - 3 = 0 \end{array}\right. \tag{17b}$$

For this example we have $m=2$, $n=2$ and $p=1$. The goal function at levels $1,2,\ldots,10$ and the constraints of problem (18) are given in Figure 6. The solution of this problem is the point $x^* = (x_1^*, x_2^*) = (0.5, 2.5)$ as can be easily verified analytically.

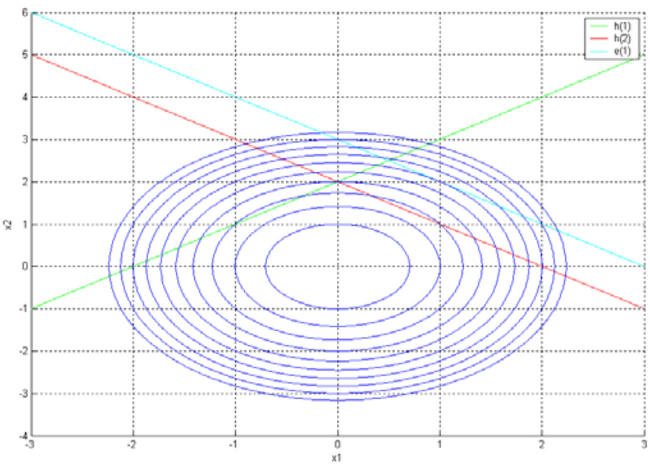

Fig.6: The goal function at levels 1,2,…,10 and the constraints of problem (18)

The same solution of problem (17) is obtained by use of standard function *fmincon* from Optimization Toolbox of MATLAB®.

We utilize the SIMULINK® model of the continuous Neural Network given in Figure 2, and have simulated its behavior with time constants $\tau_1=1$ and $\tau_2=1$.
The simulation results for state variables (x_1, x_2) and λ_1 and λ_2 for initial conditions $\lambda(0) = (\lambda_1(0), \lambda_2(0)) = (4, 5)$ are given in Figure 7 and Figure 8.

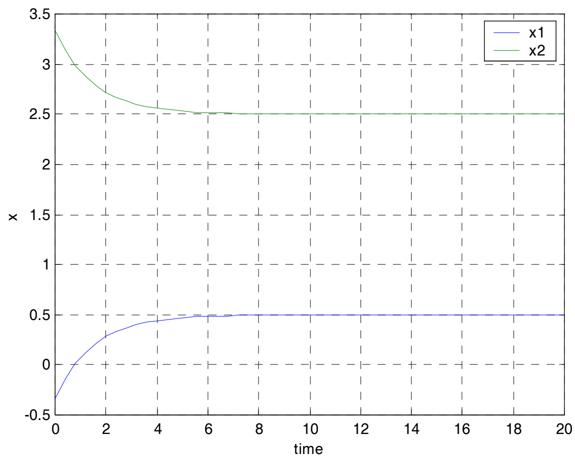

Fig.7: Simulation results for state variables x_1 and x_2 with initial conditions $\lambda(0)=(\lambda_1(0), \lambda_2(0))=(4, 5)$ for the continuous neural network model from Figure 2.

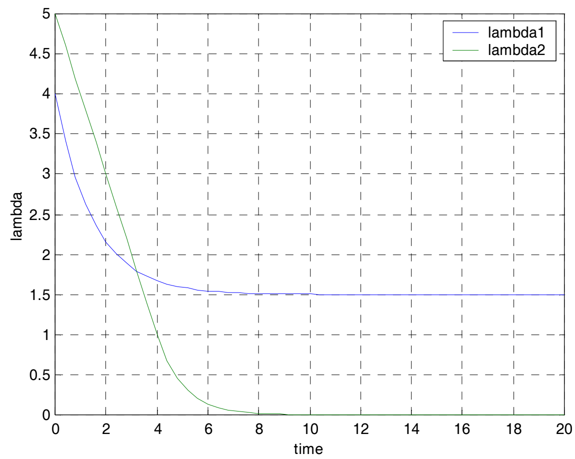

Fig.8: Simulation results for state variables λ_1 and λ_2 with initial conditions $\lambda(0)=(\lambda_1(0), \lambda_2(0))=(4, 5)$ for the continuous neural network model from Figure 2.

The speed of the transient depends on time constants. The network settled down at the equilibrium point which is the solution of the original problem (17).
We used the SIMULINK® model of the discrete Neural Network given in Figure 5, and have simulated its behavior for step size $dt=0.001$. The simulation results for state variables (x_1, x_2) and λ_1 and λ_2 for initial conditions $\lambda(0) = (\lambda_1(0), \lambda_2(0)) = (4, 5)$ are given in Figure 9 and Figure 10.
The results are similar with this obtained by the

continuous model, but the general advantage of the discrete model is that it is suitable for FPGA implementations. The low complexity of the proposed network is also advantage of the model considered.

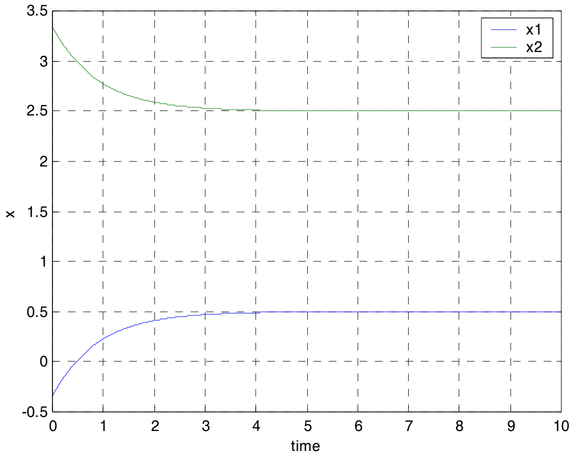

Fig.9: Simulation results for state variables x_1 and x_2 with initial conditions $\lambda(0)=(\lambda_1(0), \lambda_2(0))=(4, 5)$ for the discrete neural network model from Figure 5.

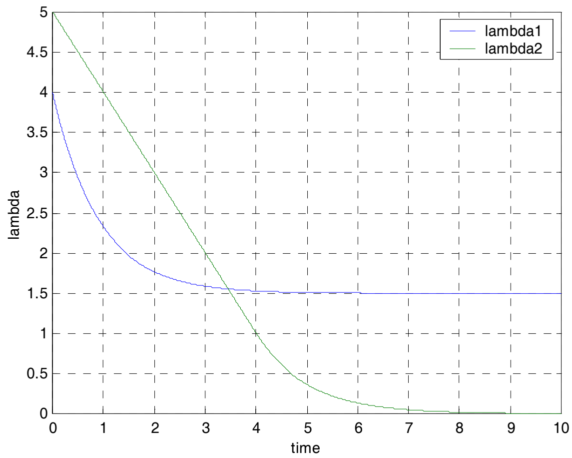

Fig.10: Simulation results for state variables λ_1 and λ_2 with initial conditions $\lambda(0)=(\lambda_1(0), \lambda_2(0))=(4, 5)$ for the discrete neural network model from Figure 5.

5 Conclusions

In this paper a discreet version of recently proposed recurrent neural network for solving general quadratic programming problems, for the case of both inequality and equality constraints, is presented. The global convergence and stability of this network is proved. The presented network has lower complexity and it is suitable for FPGA implementations. Simulation results based on SIMULINK® models are given and compared.

Acknowledgments

This work is supported by TU-Sofia project 452-8/ 2004, “Design and simulation of Neural Networks for solving Quadratic Programming Problems”.

References:

[1] A. Cichocki, R. Unbehauen, Neural Networks for Optimization and Signal Processing, John Wiley & Sons, Chichester-New York-Brisbane-Toronto-Singapore, 1993.

[2] L. O. Chua and G. N. Lin. “Non-linear programming without computation”, IEEE Trans. Circuits and Systems, CAS-31, pp.182-186, 1984.

[3] M. P. Kennedy and L. O. Chua. “Neural networks for non-linear programming”, IEEE Trans. Circuit and Systems, Vol. 35, pp.554-562, 1988.

[4] W. E. Lillo, M. H. Loh, S. Hui and S. H. Žak. “On solving constrained optimization problems with neural networks : a penalty method approach”, Technical Report TR EE 91-43, School of EE, Purdue University, West Lafayette, IN, 1991

[5] Y. Xia, “A New Neural Network for Solving Linear and Quadratic Programming Problems”, IEEE Trans. on Circuits and Systems - Part I, pp. 1544-1547, 1996.

[6] Y. Xia, J. Wang, L. Hung, “Neural Networks for Solving Inequalities and Equations, IEEE Trans. on Circuits and Systems - Part I, vol. 46, pp. 452-462, 1999.

[7] D.G.Luenberger, Linear and Nonlinear Programming, Addison-Wesley, 1984.

[8] V.M. Mladenov, F.H. Uhlmann, “Recurrent Neural Networks for Solving General Quadratic Programming Problems”, Proceedings of the 48. Internationales Wissenschaftliches Kolloquium, 22.09-25.09, 2003, TU-Ilmenau, Germany.

[9] V.M.Mladenov, „On the recurrent neural networks for solving general quadratic programming problems”, Proceedings of the 7th Seminar on Neural Network Applications in Electrical Engineering, NEUREL 2004, University of Belgrade, Serbia and Montenegro, 23-25 September, 2004, pp.5-9.