Discrete neural network for solving general quadratic programming problems

VALERI MLADENOV
1Department 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.

2 Continuous neural network for solving general quadratic programming problem
We consider the following general quadratic programming problem
where

\[ P \in \mathbb{R}^{nxn} \text{is positive definite matrix, } G \in \mathbb{R}^{mxn}, A \in \mathbb{R}^{pxn}, q \in \mathbb{R}^p, r \in \mathbb{R}, x \in \mathbb{R}^n \]

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

\[ 0 = -\nu - \lambda^T A^T G q P x \tag{2} \]

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

\[ F(y) = (F(y_1),...,F(y_m))^T \tag{5} \]

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 \( \lambda \in \mathbb{R}^n \) 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)

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

and thus

\[ Gx = F(Gx - \lambda) \tag{8b} \]

\[ AP^{-1}(G^T \lambda + A^T \nu - q) = b \tag{8c} \]

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

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

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

\[ Px + q - G^T \lambda - A^T \left( AP^{-1}A^T \right)^{-1} b + A^T \left( AP^{-1}A^T \right)^{-1} AP^{-1}G^T \lambda - A^T \left( AP^{-1}A^T \right)^{-1} AP^{-1}q = 0 \tag{10} \]
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).

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

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

where $W = G P^{-1} Q^T$, $V = G P^{-1} R$, $\lambda^{(k+1)}, \lambda^{(k)} \in R^m$, $T = 1 + W$ and $dt$ is the step size.

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 elements.
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.

**Theorem 1:** Given any initial point, there exists a unique solution of (13). The equilibrium point of (13) corresponds to a solution of (1) and it is unique when rank \( A = m \) and \( P \) is positive definite.

**Proof:** See [8], [9].

**Theorem 2:** If \( h < \frac{2}{\| r \|} \) 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:

\[
\begin{align*}
g_1(x) &= -x_1 + x_2 - 2 \geq 0 \\
g_2(x) &= x_1 + x_2 - 2 \geq 0 \\
a(x) &= x_1 + x_2 - 3 = 0 
\tag{17b}
\end{align*}
\]

For this example we have \( m=2, n=2 \) and \( p=1 \). The goal function at levels 1,2,...,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.

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 \( \lambda_1 \) and \( \lambda_2 \) for initial conditions \( \lambda(0) = (\lambda_1(0), \lambda_2(0)) = (4, 5) \) are given in Figure 7 and Figure 8.

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 \( \lambda_1 \) and \( \lambda_2 \) 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.](image1)

![Fig.10: Simulation results for state variables $\lambda_1$ and $\lambda_2$ with initial conditions $\lambda(0)=(\lambda_1(0), \lambda_2(0))=(4, 5)$ for the discrete neural network model from Figure 5.](image2)

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: