Constrained Hopfield Neural Network for Real-Time Predictive Control

J.M. Quero, C.L. Janer, and L.G. Franquelo, Member, IEEE
Dpto de Ingeniería de Sistemas y Automática, Univ. de Sevilla, Avda Reina Mercedes s/n, 41012 Sevilla - Spain

phone: +34-5-4556873 ; fax: +34-5-4556849 ; e-mail: quero@gtex02.us.es

Abstract—**The hardware implementation of an optimization network with restrictions to perform real-time Generalized Predictive Control (GPC) is described. The use of space-efficient stochastic architecture allows a a realization on a programmable logic device. As a result a programmable neural chip coprocessor that solves optimization problems subject to restrictions has been developped. Expressions for network parameters are provided to implement GPC. An adaptive controller is achieved using RAM memories to store the network parameters. Experimental results from a simple implementation of the controller are included.**

I. Introduction.

Robust, real-time control is essential to most processes in industry. Modern control methods have proved to be very efficient [1]. However they are high-time consuming specially when considering constraint satisfaction, therefore more powerful processors are needed. The GPC algorithm consists of optimizing a cost function in order to achieve the best expected control sequence [2]. New approaches [3] consider constraints in control sequences leading to constrained optimization problems.

We suggest the use of Hopfield networks to override the computational effort to implement such controller. Applications of this class of neural networks can be found elsewhere [4]. Artificial neural networks consist of a set of very simple computing elements highly interconnected that perform an overall duty. Due to the simplicity of the computational elements it is worth trying to make specific electronic implementations rather than using a general purpose digital computer to simulate its behaviour.

Electronic realization of neural networks can be faced in different ways. On one hand analog aproaches are very simple in terms of circuitry and have fast convergence times, specially when they are compared with digital implementations, but on the other hand their programming flexibility is very low. Digital implementations perform high programming flexibility and easy interface with general purpose computers but their efficiency in terms of consumed silicon area is very low, as a floating point multiplier is needed in every neuron to calculate the presynaptic activity. One way to circumvent this problem is employing stochasticism to evaluate presynaptic products [5],[6].

Our aim is to design a neural coprocessor which releases the control computer in the factory from this algorithmic computational burden. The simplicity of stochastic neural networks architectures [7] minimizes the amount of hardware needed, therefore allowing the use of programmable logic devices for its implementation.

This paper is organized as follows. In Section II it is discussed how Hopfield's Neural Network can be utilized to solve constrained linear quadratic optimization problems and how such problems can be posed in terms of adimentional quantities, yielding a formulation that can be implemented in stochastic architecture. In Section III it is shown that generalized predictive controllers apply a control sequence to the system that is being regulated in order to minimize a quadratic cost function. Due to the fact that all actuators have a limited range of actuations and slew rates, linear restrictions must be imposed to this optimization problem. In Section IV the stochastic architecture that has been used to implement the Hopfield neural network and why this architecture is suitable for GPC is described. Section V is devoted to the actual hardware that has been used to realize the neural network and why a programmable logic device has been used. In Section VI we consider specific examples to demonstrate the neural net coprocessor performance. Finally, conclusions are drawn in section VII.

II. Constrained Hopfield Neural Network.

The way in which neural networks could be applied to solve linear programming networks was first described by Tank and Hopfield [4]. They suggested an optimization

0-7803-1328-3/94$03.00

network, that is able to minimize a cost function

$$\pi = \vec{A} \cdot \vec{V} \tag{1}$$

where $\vec{A}$ is an N-dimensional vector of coefficients for the N variables which are the components of $\vec{V}$. This minimization is accomplished subject to a set of M linear constraints among the variables:

$$\vec{D}_j \cdot \vec{V}_j \geq B_j \qquad j = 1..M \tag{2}$$

$$\vec{D}_j = [\vec{D}_{j1}, \vec{D}_{j2}, \ldots, \vec{D}_{jn}]^T \tag{3}$$

where the $\vec{D}_j$, for each j, contain the N variable coefficients in a constraint equation and the B_j are the bounds.

For the case of a linear quadratic optimization problem, it can be considered an energy function of the form

$$E = \frac{1}{2}\vec{V}^T T \vec{V} + \vec{A} \cdot \vec{V} + \sum_{j=0}^{M-1} F(\vec{D}_j \cdot \vec{V} - B_j) \tag{4}$$

where $F(z)$ is a primitive of $f(z)$ given by

$$f(z) = \begin{cases} kz & \text{for } z > 0 \\ 0 & \text{for } z <= 0 \end{cases} \tag{5}$$

Consider equation 4. We shall denote the typical values of G_{ji}, I_j and V_j by G_0, I_0 and V_0. These numbers are such that the adimentional quantities χ_{ji}, μ_j and η_j defined in 6 take absolute values ranging from 0 to 1.

$$\nu = \frac{E}{V_0 I_0}; \quad \chi_{ji} = \frac{T_{ji}}{t_0}; \quad \mu_j = \frac{I_j}{I_0}; \quad \eta_j = \frac{V_j}{V_0} \tag{6}$$

Substituiting in 4

$$\nu = \frac{1}{2}\frac{t_0 V_0}{I_0}\vec{\eta}^T \chi \vec{\eta} + \vec{A} \cdot \vec{\eta} + \sum_{j=0}^{M-1} F(\vec{D}_j \cdot \vec{\eta} - \frac{B_j}{V_0}) \tag{7}$$

$$\chi_{ij} = \chi_{ji} \tag{8}$$

If V_0 is chosen so that $\frac{G_0 V_0}{I_0} = 1$ we obtain the same linear-quadratic optimization problem in terms of adimentional quantities. This formulation is suitable to be implemented in a stochastic architecture as it will be seen later.

III Application to constrained general predictive control.

The General Predictive Control algorithm consists of applying a control sequence that minimizes a multistage cost function of the form

$$\begin{aligned} J(N_1, N_2) &= E\{\sum_{j=N_1}^{N_2} [y(t+j \mid t) - w(t+j)]^2 \\ &+ \sum_{j=1}^{N_2-d} \lambda[\Delta u(t+j-1)]^2\} \end{aligned} \tag{9}$$

where E {.} is the expectation operator and $y(t+j \mid t)$ is an optimum j-step ahead prediction of the system output on data up to time t. u(t) and y(t) are the control and output sequence of the plant. N_1 and N_2 are the minimum and maximum costing horizons. λ is weighting coefficient and w(t+j) is a future set-point or reference sequence. The objective of predictive control is to compute the future control sequence u(t), u(t+1),... in such a way that the future plant output y(t+j) is driven close to w(t+j). This is accomplished by minimizing $J(N_1, N_2)$. However, it can be computationally prohibitive for real time applications.

In practice, the normal way of using GPC is to compute $u(t)$ and apply it to the process. If $u(t)$ violates the constraint it is saturated to the bounds, either by the control program or by the actuator. The case of $u(t+1), \cdots, u(t+N)$ violating the constraints is not even considered as in most cases the signals are not even computed. This way of operating restricts the optimallity of the GPC when constraints are violated. Furthermore, the main purpose of the GPC, which is to optimize equation (9), is no longer valid and the *best* expected control is not achieved.

Most processes in industry can be described by the following transfer function:

$$y(t+1) = ay(t) + b_0 u(t-d) + b_1 u(t-d-1) + \varepsilon(t+1) \tag{10}$$

If an optimal predictor is used, as shown in [8], the dead time can be ignored for designing purposes.

If we compare the energy function (4) of a Hopfield net to the cost function (9) of the GPC, considering that the output of the neural net V_i corresponds to the control sequence u_i, we obtain the expressions for the parameters of the network as funtions of the system parameters, control parameters and the system input and output sequences for a given control horizon [9]. As an example, results of this procedure are shown in Appendix I for systems described in (10).

It is remarkable that equations which define the conductances T_{ij} only depend on the systems parameters, whilst equations defining the net inputs also depend on the system input and output sequences and the reference. The values of C_i and R_i are chosen arbitrarily. We can define the constraint subnet identifying the terms of $\vec{D}_j \cdot \vec{V}_j \geq B_j$ for each restriction.

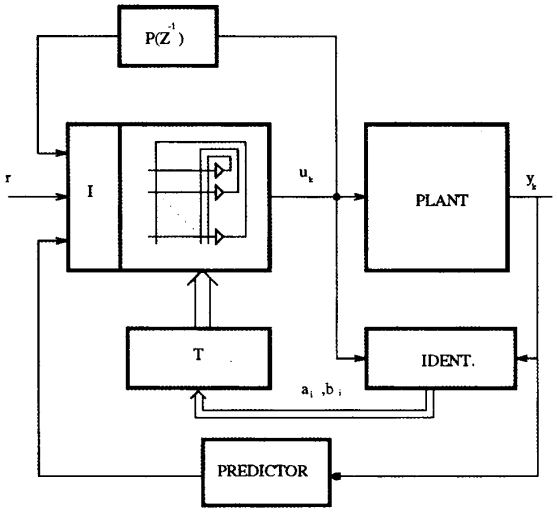

Figure 1: Control Scheme

The resulting control scheme is shown in Fig. 1. The neural controller provides the input signal sequence to the plant. The identifier estimates the plant model parameters from its input and output signals.These values are fed into block T that represents the calculus of the polynomials to obtain the conductances of the neural controller. In block I intensity dependent sources are calculated according to the current reference. As it will be explained later, block I computations are also performed by the neural network while identification and T_{ij} evaluations are carried out by the control computer.

IV Architecture.

A constrained Hopfield neural network will be considered in this article as a set of saturating linear integrators whose inputs are linear combinations of other integrators' values and time varying terms. The dynamics of these terms are supposed to be much slower than net dynamics so that they may be regarded as constants during integration. Analog versions of this controller can be found elsewhere [10], but they lack of programmability.

Stochastic systems use stochastic signals whose values randomly take the value 0 or 1. The average of these

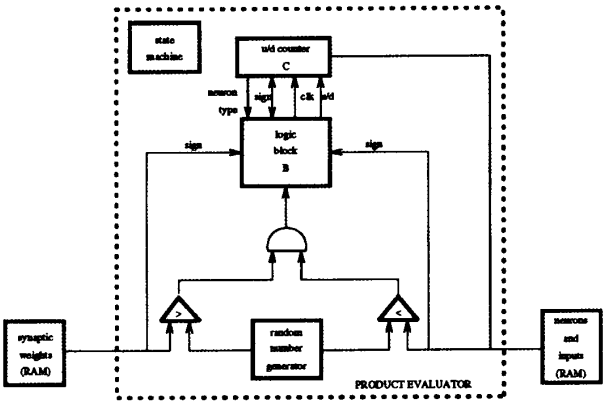

Figure 2: Stochastic Neural Net Processor

values can be viewed as an analog value in the range [-1,1]. Stochastic signals can be multiplied using only a simple AND gate.

Therefore, product terms in the newral networks can be caculated by a stochastic multiplier. The optimization problem should be posed in terms of adimentional quatities as constants and variables are to be translated into stochastic streams of pulses. This has been done in section II.

The high-level architecture is shown in Fig. 2. It consists of a stochastic neural processor, which is basically composed of a stochastic product evaluator and a state machine controller, and two random access read and write memories. Notice that neuron activity values evolve due to either other neurons' activity values or net inputs. At any rate these terms may be considered as products that can be computed by the stochastic product evaluator.

Constrained Hopfield neural network's interconnection weights and net inputs' constant terms are stored in one of the memories and neuron activities and variable terms of the net inputs are stored in the remaining one. All these magnitudes may either take positive or negative values.

Two kinds of neurons are involved in the network's evolution: *system neurons* and *restriction neurons*. System neurons are saturating linear integrators that can either take positive or negative values. Restriction neurons are saturating nonlinear integrators. They take negative values when the problem restrictions are violated and force the system neurons to evolve in such a way that constrictions are fulfilled.

Neurons' information (activity value, sign and type of neuron) are serially adressed by the controller and loaded in the up/down counter N and latches S and T. The

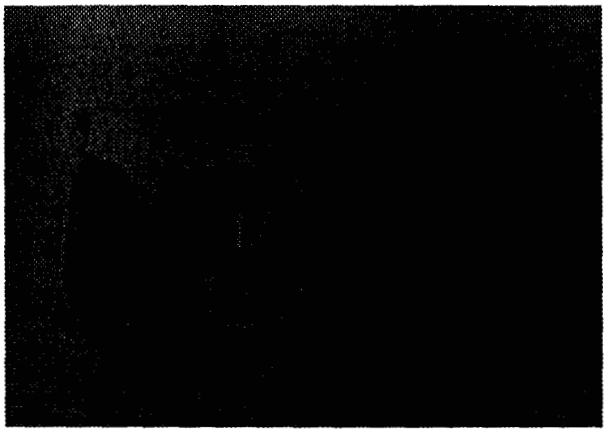

Figure 3: Test board photograph.

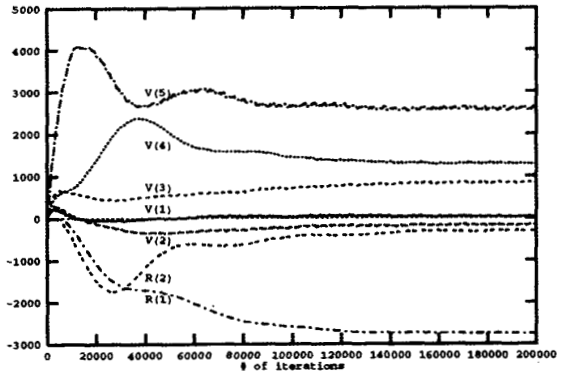

Figure 4: Controller's chip transient response.

stochastic product evaluator calculates the inhibitory or excitatory influence that neurons (v_j, $j = 1..n$) (and input time varying terms) have over the one which has been loaded (v_i). To achieve this goal these values and their related constants are serially adressed by the controller and read. Random numbers are then generated and compared with them, producing two stochastically independent streams of pulses that are ANDed and fed to logic block B. This logic block either increments or decrements the up/dowm counter according to the signs of the three numbers involved and the kind of neuron that is being integrated. It also changes the sign bit (S latch) whenever a zero crossing takes place. When the process has finished the new computed value of of v_i is written in the external memory and another neuron is loaded.

V Physical realization.

A neural optimization problem accelerator has been developped (Fig. 3) and tested following the described structure. The stochastic neural processor has been implemented in an Erasable Programable Logic Device. EPLDs can be designed and programed in-house, eliminating the long engineering lead times and high tooling efforts and costs of full custom or semicustom devices. They are also very appropriate for short series fabrication where an Application-Specific Integrated Circuit would be uneconomical. An Altera EPM5130 device has been chosen to develop this application [10].

The number of system restriction neurons and network input signals are fully programmable. Two external RAMs have been used for data storage, one for neurons' values and other for synaptic weight values. Net inputs' time-varying terms are sampled by the host computer and written within the neuron RAM. Their associated constants are plant model parameters which were stored by the host computer in the weight RAM when the net was configured. Net configuration is accomplished by simply loading the net's interconnection weights and net inputs' constant terms in one of the external memories and by loading the initial neurons' activity values and net inputs' time varying initial values in the other one.

The system's clock may be either external, in case monitoring the net's evolution is desired, or internal, for normal operation. Most industrial processes have slow dynamics. For this reason high speed memories have not been used, yielding a system's clock speed of 5MHz.

VI Application.

As an example the system to be controlled has a pure delay of one sampling period and is described by the following polynomials:

$$\begin{aligned} A(z^{-1}) &= 1 - 0.4z^{-1} - 0.32z^{-2} \\ B(z^{-1}) &= 0.1 \end{aligned}$$

For the first set of tests the future reference was considered to be a previously known square wave. The weighing factor λ was made equal to 0.0001. The maximun control signals were constrained by ± 10, the slew rates were considered to be limited by ± 5 and a control horizon of 5 has been used.

The transient response of the neural network is shown in Fig. 4 when calculating the first control signal sequence. When the stable state is reached, V(5) represents the control signal to be applied to the system. R(1) and R(2) are the outputs for the restricctions $V(5) < 10$ and

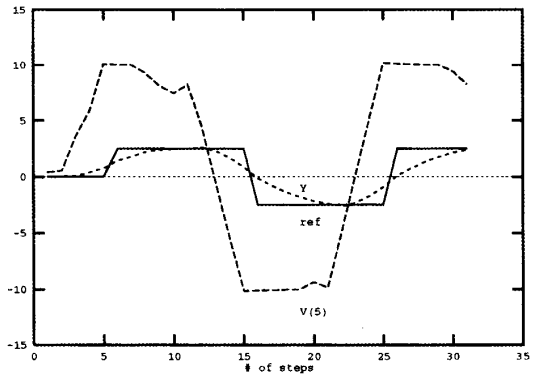

Figure 5: System's closed loop behaviour.

$V(5) - V(4) < 5$ respectively. This stable state matches theoretical result and its computation time has been 0.3 seconds.

In order to check the performance of the neural controller, the response of the closed loop system has been studied when changing the reference. In Fig. 5 'ref' represents the applied refernce signal,'V(5)' the steady state of the first system neuron in the controller and 'Y' the output of the plant as functions of the number of the controller's actuations.

VII Conclusions.

Digital stochastic architectures are a very efficient way to realize neural networks as they significantly reduce the needs of silicon area. We have implemented a general purpose version of a constrained Hopfield network on a programmable logic device. This coprocessor solves any constrained quadratic optimization problem. It has been used to implement the General Predictive Control problem in order to release the control computer from the heavy computational burden of this algorithm. Expressions of net parameters are provided for a common plant model which permit an adaptive control. The chip has been incorporated into a personal computer. Results obtained with this experimental chip have been reported.

Appendix I
Network Parameters

For this system, the net parameters are given by the following equations:

$$T_{i,i} = \begin{cases} -2(\lambda + b_0^2) & i = n \\ -2(ab_0 + b_1)^2 - 2(2\lambda + b_0^2) & i = n-1 \\ -2(a^{n-i}b_0 + a^{n-i-1}b_1)^2 + T_{i+1,i+1} & i < n-1 \end{cases} \quad (11)$$

$$T_{i-1,i} = \begin{cases} -2[(ab_0 + b_1)b_0 - \lambda] & i = n \\ -2a^{2(n-i)-1}(ab_0 + b_1)^2 + T_{i,i+1} & i < n \end{cases} \quad (12)$$

$$T_{i-2,i} = \begin{cases} -2ab_0(ab_0 + b_1) & i = n \\ -2a^{2(n-i)}(ab_0 + b_1)^2 + T_{i-1,i+1} & i < n \end{cases} \quad (13)$$

$$T_{k,i} = a^{i-k-2}T_{i-2,i} \qquad i - k > 2 \quad (14)$$

$$T_{i,j} = T_{j,i} \quad (15)$$

$$\begin{aligned} I_1 &= -2\{(ay_t + b_1u_{t-1})(\sum_{i=j}^{2n-j} a^{i-1}b_0^{odd(i)}b_1^{even(i)}) \\ &- [(b_0 + b_1)(\sum_{i=1}^{n-j+1} a^{i-1}) - a^{n-j}b_1]r\} \\ &+ 2\lambda u_{t-1} \end{aligned} \quad (16)$$

if j *odd*

$$\begin{aligned} I_j &= -2\{(ay_t + b_1u_{t-1})(\sum_{i=j}^{2n-j} a^{i-1}b_0^{odd(i)}b_1^{even(i)}) \\ &- [(b_0 + b_1)(\sum_{i=1}^{n-j+1} a^{i-1}) - a^{n-j}b_1]r\} \end{aligned} \quad (17)$$

if j *even*

$$\begin{aligned} I_j &= -2\{(ay_t + b_1u_{t-1})(\sum_{i=j}^{2n-j} a^{i-1}b_0^{even(i)}b_1^{odd(i)}) \\ &- [(b_0 + b_1)(\sum_{i=1}^{n-j+1} a^{i-1}) - a^{n-j}b_1]r\} \end{aligned} \quad (18)$$

where $odd(i) = remainder(i/2)$ and $even(i) = 1 - odd(i)$.

References

[1] M. Gopal. *Model Control System Theory.* Wiley Eastern Limited 1985.

[2] D. W. Clarke C. Mohtadi and P.S. Tuffs. Generalized Predictive Control. Part I The Basic Algorithm. *Automatica* vol.23, pp. 137-148. 1988

[3] E. F. Camacho. Constrained Generalized Predictive Control *IEEE Trans. on Automatic Control*, vol 38, pp. 327-332, 1993.

[4] 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. on Circuit and Systems*, vol.35, pp. 554-562, 1988

[5] D.E. Van den Bout and T.K. Miller III A Digital Architecture Employing Stochaticism for the Simulation of Hopfield Neural Nets. *IEEE Trans. on Circuit and Systems*, vol.36, pp. 732-738, 1989

[6] M. S. Melton, T. Phan, D. S. Reeves and D. E. Van den Bout The TInMANN VLSI Chip *IEEE Trans. on Neural Networks*, vol.3, pp.375-384, 1992.

[7] Y. Kondo and Y. Sawada Functional Abilities of a Stochastic Logic Neural Networks *IEEE Trans. on Neural Networks*, vol.3, pp.434-443, 1992.

[8] E. F. Camacho and J.M. Quero. Precomputation of Generalized Predictive Controllers. *IEEE Trans. on Automatic Control*, vol. AC-36, pp. 852-859, 1991.

[9] J. M. Quero and E. F. Camacho. Neural Generalized Predictive Self-tunning Controllers. *Proc. IEEE Congress on Systems Control*, pp 160-163, 1990

[10] J.M. Quero, L.G. Franquelo and E.F. Camacho. "Networks for constrained Predictive Control". *IEEE Trans. on Circuits and Systems*, vol. 40, 621-626, 1993.

[11] Altera Corp. *Altera Data Book*, 1990.