RESEARCH PAPER | Computer Science | Volume : 3 | Issue : 6 | June 2013 | ISSN - 2249-555X

Optimizing Single Layer Cellular Neural Network Simulator using Simulated Annealing Technique with Neural Networks

KEYWORDS: single-layer Cellular neural networks, numerical integration algorithms, simulated annealing, neural networks

O. H. Abdelwahed | M. El-Sayed Wahed

Department of Computer Science, Faculty of Computers and Information, Suez Canal University, EGYPT

ABSTRACT *An efficient numerical integration algorithm using simulated annealing with neural networks for single layer Raster Cellular Neural Networks (CNN) simulator is presented in this work. The simulator is capable of performing CNN simulations for any size of input image, thus a powerful tool for researchers investigating potential applications of CNN. The goal is to provide optimal control with reduced calculus effort by comparing the solutions of the MRDE obtained from the well known traditional Runge Kutta(RK)method and nontraditional neural network method. Accuracy of the neural solution to the problem is qualitatively better.*

1 Introduction

Neural networks or simply neural nets are computing systems, which can be trained to learn a complex relationship between two or many variables or data sets. Having the structures similar to their biological counterparts, neural networks are representational and computational models processing information in a parallel distributed fashion composed of interconnecting simple processing nodes [26]. Neural net techniques have been successfully applied in various fields such as function approximation, signal processing and adaptive (or) learning control for nonlinear systems. Using neural networks, a variety of off-line learning control algorithms have been developed for nonlinear systems [17, 22]. A variety of numerical algorithms have been developed for solving the algebraic Riccati equation. In recent years, neural network problems have attracted considerable attention of many researchers for numerical aspects for algebraic Riccati equations see [9, 13, 14, 27]. Singular systems contain a mixture of algebraic and differential equations. In that sense, the algebraic equations represent the constraints to the solution of the differential part. These systems are also known as degenerate, descriptor or semi-state and generalized state-space systems. The complex nature of singular system causes many difficulties in the analytical and numerical treatment of such systems, particularly when there is a need for their control. The system arises naturally as a linear approximation of system models or linear system models in many applications such as electrical networks, aircraft dynamics, neutral delay systems, chemical, thermal and diffusion processes, large-scale systems, robotics, biology, etc., see [10, 11, 18]. Most of the research on nonlinear singular systems has focused primarily on issues related to solvability and numerical solutions for such systems. The literature on feedback control of nonlinear singular systems is sparse. The feedback stabilization problem for nonlinear singular systems is addressed by McClamroch [21]. In this paper, we make use of a result that generalizes the LQ theory to nonlinear systems to provide a nonlinear design method [20]. This nonlinear quadratic (NLQ) method applies to systems having a broad class of nonlinear dynamics with state dependent weighting matrices. In brief, it turns out that the infinite time horizon LQ regulator problem, when solved afresh at every point on the state trajectory, leads to an asymptotically optimal control policy. The LQ regulator problem converges to the optimal control close to the origin.

CNN is a hybrid of Cellular Automata and Neural Networks (hence the name Cellular Neural Networks), and it shares the best features of both worlds. Like Neural Networks, its continuous time feature allows real-time signal processing, and like Cellular Automata, its local interconnection feature makes VLSI realization feasible. The basic circuit unit of CNN is called a [2]. It contains linear and nonlinear circuit elements. Any cell, C(i,j), is connected only to its neighbor cells, i.e. adjacent cells interact directly with each other. This intuitive concept is called neighborhood and is denoted as N(i,j). Cells not in the immediate neighborhood have indirect effect because of the propagation effects of the dynamics of the network. Each cell has a state x input U, and output y. The state of each cell is bounded for all time t > U and, after the transient has settled down, a cellular neural network always approaches one of its stable equilibrium points. This last fact is relevant because it implies that the circuit will not oscillate. The dynamics of a CNN has both output feedback (A) and input control (B) mechanisms. The first order nonlinear differential equation defining the dynamics of a cellular neural network cell can be written as follows:

$$C\frac{d\,x_{ij}(t)}{dt} = -\frac{1}{R}x_{ij}(t) + \sum_{C(k,l)\in N(i,j)} A(i,j;k,l)\,y_{kl}(t) + \sum_{C(k,l)\in N(i,j)} B(i,j;k,l)\,u_{kl}$$

$$y_{ij}(t) = \frac{1}{2}\left(|x_{ij}(t)+1| - |x_{ij}(t)-1|\right) \qquad (1)$$

Where x_{ij} is the state of cell $C(i,j)$, $x_{ij}(0)$ is the initial condition of the cell, C is a linear capacitor, R is a linear resistor, I is an independent current source, $A(i,j;k,l)y_{kl}$ and $B\{i,j;k,l)u_{kl}$ are voltage controlled current Sources for all cells C(k,l) in the neighborhood N(ij) of cell C(ij), and y_{ij} represents the output equation.

Notice from the summation operators that each cell is affected by its neighbor cells. A(.) acts on the output of neighboring cells and is referred to as the feedback operator. B(.) in turn affects the input control and is referred to as the control operator. Specific entry values of matrices A(.) and B(.) are application dependent, are space invariant and are called cloning templates. A current bias Z and the cloning templates determine the transient behavior of the cellular nonlinear network.

CNNs have as input a set of analog values and its programmability is done via cloning templates.Thus, programmability is one of the most attractive properties of CNNs, but how to choose the optimal network and how to program it to perform a given task are still topics under investigation. This is the reason why there is a need for behavioral CNN simulator capable of helping investigators design and manipulate cloning templates ("programming"). Existent tools are not meant

Volume : 3 | Issue : 6 | June 2013 | ISSN - 2249-555X

to deal with a significant number of pixels typical in common image processing applications [5]. The simulator presented here not only satisfies this need, but it also can be used for testing CNN hardware implementations. El-Sayed Wahed and O.H. Abdel wahed[8] introduced an efficient numerical integration algorithm for Single-Layer Raster Cellular Neural Networks Simulator. In this paper, we consider the same problem but by using RK4 and neural networks.

2 Behavioral Simulation

Recall that equation (1) is space invariant, which means that A(i,j;k,l) = A(i-k,j-1) and B(i,j;k,l) = B(i,k;j,l) for all i,j,k,l.

Therefore, the solution of the system of difference equations can be seen as a convolution process between the image and the CNN processors. The basic approach is to imagine a square subimage area centered at (x,y), with the subimage being the same size of the templates involved in the simulation. The center of this subimage is then moved from pixel to pixel starting, say, at the top left comer and applying the A and B templates at each location (x,y) to solve the differential equation. This procedure is repeated for each time step, for all the pixels. An instance of this image scanning-processing is referred to as an "iteration". The processing stops when it is found that the states of all CNN processors have converged to steady-state values [2] and the outputs of its neighbor cells are saturated, e.g. they have a +1 value.

This whole simulating approach is referred to as raster simulation. A simplified algorithm is presented below for this approach. The part where the integration is involved (i.e. calculation of the next state) is explained in the Numerical Integration Methods section.

In the following two subsections we will discuss simulated annealing algorithm and the mathematical modeling used in simulated annealing and our proposed simulator.

2.1 The Simulated Annealing Algorithm

In the early 1980s Kirkpatrick et al. (1983) and independently Cemy (1985) introduced the concepts of annealing in combinatorial optimization. Originally these concepts were heavily inspired by an analogy between the physical annealing process of solids and the problem of solving large combinatorial optimization problems. Since this analogy is quite appealing we use it here as a background for introducing simulated annealing. In condensed matter physics, annealing is known as a thermal process for obtaining low energy states of a solid in a heat bath. The process consists of the following two steps:

- increase the temperature of the heat bath to a maximum value at which the solid melts;
- decrease carefully the temperature of the heat bath until the particles arrange themselves in the ground state of the solid.

In the liquid phase, all particles arrange themselves randomly, whereas in the ground state of the solid, the particles are arranged in a highly structured lattice, for which the corresponding energy is minimal. The ground state of the solid is obtained only if the maximum value of the temperature is sufficiently high and the cooing is performed sufficiently slowly. Otherwise, the solid will be frozen into a meta-stable state rather than into the true ground state.

Metropolis et al. [7] introduced a simple algorithm for simulating the evolution of a solid in a heat bath to thermal equilibrium. Their algorithm is based on Monte Carlo techniques and generates a sequence of states of the solid in the following way.

Given a current state i of the solid with energy E_i, then a subsequent state j is generated by applying a perturbation mechanism which transforms the current state into a next state by a small distortion, for instance by displacement of a particle. The energy of the next state is E_j. If the energy difference, E_j— E_i, is less than or equal to zero, the state j is accepted as the current state. If the energy difference is greater than zero, then the state j is accepted with a probability given by

$$\exp\left(\frac{E_i - E_j}{k_B T}\right) \qquad (2)$$

where T denotes the temperature of the heat bath and KB is a physical constant called the Boltzmann constant. The acceptance rule described above is known as the Metropolis criterion and the algorithm that goes with it is known as the Metropolis algorithm. It is known that, if the lowering of the temperature is done sufficiently slowly, the solid can reach thermal equilibrium at each temperature. In the Metropohs algorithm this is achieved by generating a large number of transitions at a given value of the temperature. Thermal equilibrium is characterized by the Boltzmann distribution, which gives the probability of the solid of being in a state with energy Ei at temperature T, and which is given by

$$P_T\{X = i\} = \frac{\exp(-E_i / k_B T)}{\sum_j \exp(-E_i / k_B T)} \qquad (3)$$

where X is a random variable denoting the current state of the solid and the summation extends over all possible states. As we indicate below, the Boltzmann distribution plays an essential role in the analysis of the convergence of simulated annealing. Returning to simulated annealing, the Metropolis algorithm can be used to generate a sequence of solutions of a combinatorial optimization problem by assuming the following equivalences between a physical many-particle system and a combinatorial optimization problem:

- solutions in the combinatorial optimization problem are equivalent to states of the physical system;
- the cost of a solution is equivalent to the energy of a state.

Furthermore, we introduce a control parameter which plays the role of the temperature. Simulated annealing can thus be viewed as an iteration of Metropolis algorithms, executed at decreasing values of the control parameter.

We can now let go of the physical analogy and formulate simulated annealing in terms of a local search algorithm. To simplify the presentation, we assume, in the remainder of this chapter, that we are dealing with a minimization problem. The discussion easily translates to maximization problems. For an instance (S, f) of a combinatorial optimization problem and a neighborhood function .The meaning of the four functions in the below procedure in fig. 2. is obvious: INITIALIZE computes a start solution and initial values of the parameters c and L; GENERATE selects a solution from the neighborhood of the current solution; CALCULATE.LENGTH and CALCULATE_CONTROL compute new values for the parameters L and c, respectively.

As already mentioned, a typical feature of simulated annealing is that, besides accepting improvements in cost, it also accepts deteriorations to a limited extent. Initially, at large values of c, large deteriorations will be accepted; as c decreases, only smaller deteriorations will be accepted and, finally, as the value of c approaches 0, no deteriorations will be accepted at all.

The below is the algorithm of simulated annealing:

procedure SIMULATED ANNEALING;

begin

INITIALIZE (i_{start}, C_o, L_o)

k:=0;

Volume : 3 | Issue : 6 | June 2013 | ISSN - 2249-555X

repeat

for i := 1 to L_k do

begin

GENERATE (j from S_i);

i f f(j) $\leq$ f(i) then i := j

else

if $\exp\left(\frac{f(i)-f(j)}{c_k}\right)$ > random[0, 1) then i := j

end;

k:= k + 1;

CALCULATE_LENGTH (L_k);

CALCULATE_CONTROL(c_k);

until stop criterion

end;

2.2The Mathematical Model and the proposed simulator

Simulated annealing can be mathematically modeled by means of Markov chains. In this model, we view simulated annealing as a process in which a sequence of Markov chains is generated, one for each value of the control parameter. Each chain consists of a sequence of trials, where the outcomes of the trials correspond to solutions of the problem instance.

Let {S, f) be a problem instance, N a neighborhood function, and X(k) a stochastic variable denoting the outcome of the kth trial. Then the transition probability at the kth trial for each pair i, j € S of outcomes is defined as

$$pij(k) = p\{x(k) = j | x(k-1 = i)\} = \begin{cases} \{G_{ij}(c_k A_{ij}(c_k)) \; if \; i \neq j \\ 1 - \sum_{t \in S, j \neq t} \{G_{ij}(c_k A_{ij}(c_k)) \; if \; i = j \end{cases} \quad (4)$$

where G_{ij} (ck) denotes the generation probability, i.e. the probability of generating a solution j when being at solution i, and Gij,/ (ck) denotes the acceptance probability, i.e. the probability of accepting solution j , once it is generated from solution i. The most frequently used choice for these probabilities is the following:

G_{ij}(ck) =

$$\begin{cases} |N(i)|^{-1} & if \quad j \in S_i \\ 0 & if \quad j \notin S_i \end{cases} \quad (5)$$

And

A_j(ck) =

$$\begin{cases} 1 & if \quad f(j) \leq f(i) \\ \exp((f(i)-f(j))/c) & if \quad f(j) > f(i) \end{cases} \quad (6)$$

For fixed values of c, the probabilities do not depend on k, in which case the resulting Markov chain is time-independent or homogeneous. Using the theory of Markov chains it is fairly straightforward to show that, under the condition that the neighborhoods are strongly connected—in which case the Markov chain is irreducible and periodic—there exist a unique stationary distribution of the outcomes. This distribution is the probability distribution of the solutions after an infinite number of trials.

The following is the Time-Multiplexing CNN simulation with Simulated Annealing:

Algorithm: (Single-Layer CNN simulation with Simulated Annealing)

Obtain the input image, initial conditions and templates from user;

/* M,N = # of rows/columns of the image */

/* APPLY SIMULATED ANNEALING */

begin

INITIALIZE (i_{start}, C_o, L_o);

k:=0;

repeat

for i := 1 to L_k do

begin

GENERATE (j from S_i);

i f f(j) $\leq$ f(i) then i := j

else

if $\exp\left(\frac{f(i)-f(j)}{c_k}\right)$ > random[0, 1) then i := j

end;

k:= k + 1;

CALCULATE_LENGTH (L_k);

CALCULATE_CONTROL(c_k);

until stopcriterion

end;

/* Use the optimized parameters from the simulated annealing */

while (converged-cells < total # of cells) (

for (i=l; i<=M; i++)

for (j=l; j<=N; j++) (

if (convergence-flag[i][j])

/* calculation of the next state*/

continue; /* current cell already converged */

$$x_{ij}(t_{n+1}) = x_{ij}(t_n) + \int_{t_n}^{t_{n+1}} f(x(n\tau))d n\tau$$

/* convergence criteria */

$$if\left(\frac{dx_{ij}}{dt} = 0 and \; y_{kl} = \pm 1 \forall C(k,l) \in N_r(i,j)\right) \{$$

{

convergence-flag[i][j] = 1;

converged-cells++ ;

Volume : 3 | Issue : 6 | June 2013 | ISSN - 2249-555X

```
}

} /* end for */

/* update the state values of the whole image*/

for (i=l; i<=M; i++)

for (j=l; j<=N; j++) (

if (convergence-flag[i] [j]) continue;

X_ij(t_n+1) = X_ij(t_n);

}

#_of_iteration++;

) /* end while */
```

The raster approach implies that each pixel is mapped onto a CNN processor. That is, we have an image processing function in the spatial domain that can be expressed as:

$$g(x,y) = T(f(x,y)) \qquad (7)$$

where f(.) is the input image, g(.) the processed image, and T is an operator on f(.) defined over the neighborhood of (x,y).

3 Numerical Integration Methods

M. El-Sayed Wahed and O.H. Abdel wahed [8] introduced An efficient numerical integration algorithm for Single-Layer Raster Cellular Neural Networks Simulator. We consider the same problem but by using three of the single-step numerical integration algorithms used in the CNN behavioral simulator described here. They are the RK4 , k11, and k12 used in our network solution.

4 Simulation Results and Comparisons

The simulation time used for comparisons is the actual CPU time used. The input image format for this simulator is a JPEG. format.

method	Mean Square Error	Peak Signal to Noise Ratio	MNormalized Cross-Correlation	Average Difference	Structural Content	Maximum Difference	Normalized Absolute Error
Runge Kutta (RK4)	1.7000e+003	11.8021	0.9449	8.8555	1.1055	240	0.0700
K11	1.6900e+003	10.4000	0.9305	6.4432	1.1010	234	0.0480
K12	1.6700e+003	9.9931	0.9998	6.2236	.90910	232	0.0370

Fig.3. Image processing (a) After Averaging Template, (b) After Averaging and Edge Detection

Fig. 3 shows results of the raster simulator obtained from a complex image of 65,536(256x256) pixels. For this example an Averaging template followed by an Edge Detection template were applied to the original image to yield the images displayed in Figs. 3a and 3b, respectively.

5 Conclusion

The solution of MRDE can be obtained by neural network approach. A neuro computing approach can yield a solution of MRDE significantly faster than standard solution techniques like RK method . As researchers are coming up with more and more CNN applications, an efficient and powerful simulator is needed. The simulator hereby presented meets the need in five ways: 1) Depending on the accuracy required for the simulation, the user can choose from three numerical methods to perform the numerical integration, 2) The input image format is JPEG, which is commonly available, 3) The input image can be of any size, allowing simulation of images available in common practices, 4) CPU time of our methods is better than those in the literature, 5), the quality measures of the pictures and the edge detection for our method is better than those in the literature.

REFERENCE

[1] J. Abdul Samath and N. Selvaraju(2010). " Solution of matrix Riccati differential equation for nonlinear singular system using neural networks". International Journal of Computer Applications (0975 - 8887) Volume 1 – No. 29. | [2] L. 0. Chua and L. Yang(1988). "Cellular Neural Networks: Theory & Applications," IEEE Trans. Circuits and Systems, Vol. CAS-35, pp. 1257-1290. | [3] L.O. Chua and T. Roska(1992). "The CNN Universal Machine Part 1: The Architecture", in Int. Workshop on Cellular Neural Networks and their Applications (CNNA), pp. 1-10. | [4] J. A. Nossek, G. Seiler, T. Roska and L. 0. Chua (1992.). "Cellular Neural Networks: Theory and Circuit Design," International Journal of Circuit Theory and Applications, Vol. 20, pp. 533-553. | [5] J. Varrientos and E. Sanchez-Sinencio(1992), "CELLSIM: A cellular neural network simulator for the personal computer," in Proc. 35th Midwest Symp. Circuits Systs, pp. 1384-1387. | [6] W. H. Press, B. P. Flannery, S.A. Teukolsky, and W.T.g Vetterling(1986). "Numerical Recipes. The Art of Scientific Computing", Cambridge University Press, New York. | [7] P.J.M. van Laarhoven and E.H.L. Aarts, (1987) Simulated Annealing: Theory and Application. ISBN 90-277-2513-6 | [8] M. El-Sayed Wahed and O.H. Abdel wahed (2012). An efficient numerical integration algorithm for Single-Layer Raster Cellular Neural Networks Simulator. the international Journal of the physical sciences(IJPS), Vol. 7(47), pp. 6144-6148. | [9] P. Balasubramaniam, J. Abdul Samath and N. Kumaresan, Neuro Approach for solving Matrix Riccati Differential Equations, Neural, Parallel Sci. Comput., 2 (2007) 125–135. | [10] S. L. Campbell, Singular Systems of Differential Equations, Pitman, Marshfield, MA, 1980. | [11] S. L. Campbell, Singular Systems of Differential Equations II, Pitman, Marshfield, MA, 1982. | [12] G. Da Prato and A. Ichikawa, Quadratic control for linear periodic systems, Appl. Math. Optim.,18 (1988), 39–66. | [13] S. W. Ellacott, Aspects of the numerical analysis of neural networks, Acta Numer., 5 (1994), 145–202. | [14] F. M. Ham and E. G. Collins, A neurocomputing approach for solving the algebraic matrix Riccati equation,Proceedings IEEE International Conference on Neural networks, 1 (1996), 617 – 622. | [15] M. Jamshidi, An overview on the solutions of the algebraic matrix Riccati equation and related problems, LargeScale Systems, 1 (1980), 167–192. | [16] L. Jodar and E. Navarro, Closed analytical solution of Riccati type matrix differential equations, Indian J. Pureand Appl. Math., 2 (1992), 185–187. | [17] A. Karakasoglu, S. L. Sudharsanan and M. K. Sundareshan, Identification and decentralized adaptive control usingneural | networks with application to robatic manipulators, IEEE Trans. Neural Networks, 4 (1993), 919–930. | [18] F. L. Lewis, A Survey of Linear Singular Systems, Circ. Syst. Sig. Proc., 5(1)(1986), 3–36. | [19] N. Lovren and M. Tomic, Analytic solution of the Riccati equation for the homing missile linear quadratic control problem, J. Guidance. Cont. Dynamics , 17 (1994), 619–621. | [20] D. Mccaffrey and S. P. Banks, Lagrangian manifolds and asymptotically optimal stabilizing feedback control, Syst.Control Lett., 43 (2001), 219-224. | [21] N. H. McClamroch, Feedback stabilization of control systems described by a class of nonlinear differential algebraicequations, Systems Control Lett., 15 (1990), 53–60. | [22] K. S. Narendra and K. Parathasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural networks, 1 (1990), 4–27. | [23] M. Razzaghi, Solution of the matrix Riccati equation in optimal control, Information Sci., 16 (1978), 61–73. [32] M. Razzaghi, A computational solution for a matrix Riccati differential equation, Numerical Math., 32 (1979), 271–279. | [24] D. R. Vaughu, A negative exponential solution for the matrix Riccati equation, IEEE Trans Automat. Control, 14 (1969), 72–75. | [25] P. De. Wilde, Neural Network Models, Second ed., Springer-Verlag, London, 1997. | [26] J. Wang and G. Wu, A multilayer recurrent neural network for solving continuous-time algebraic Riccati equations, Neural Networks, 11 (1998), 939–950. | [27] K. Zhou and Khargonekar, An algebraic Riccati equation approach to H optimization, Systems Control Lett., 11 (1998), 85–91.