Information Technology Journal 6 (2): 202-206, 2007
ISSN 1812-5638

An Efficient Numerical Integration Technique for Multi-Layer Raster CNN simulator

[1]V. Murugesh and [2]N. Rengarajan
[1]Department of Computer Science and Engineering,
National Institute of Technology, Tiruchirappalli-620 015, Tamil Nadu, India
[2]Department of Electronics and Communication Engineering,
K.S. Rangasamy College of Technology, Tiruchencode-637 209, Tamil Nadu, India

Abstract: An efficient simulator for Cellular Neural Networks (CNNs) is presented in this study. This simulator is capable of performing Multi-Layer Raster Simulation for any size of input image, thus a powerful tool for researchers investigating potential applications of CNN. This study reports an efficient algorithm exploiting the latency properties of Cellular Neural Networks along with popular numerical integration techniques; simulation results and comparison are also presented.

Key words: Multi-layer raster simulation, cellular neural networks, numerical integration algorithms, RK-gill algorithm, RK-butcher algorithm

INTRODUCTION

Cellular Neural Networks (CNNs) are analog, time-continuous, nonlinear dynamical systems and formally belong to the class of recurrent neural networks. Since their introduction by Chua and Yang (1988a, b) they have been the subjects of intense research. Initial applications include image processing, signal processing, pattern recognition and solving partial differential equations, etc.

Runge-Kutta (RK) methods have become very popular, both as computational techniques as well as subject for research, which were discussed by Butcher (1987, 2003). This method was derived by Runge around the year 1894 and extended by Kutta a few years later. They developed algorithms to solve differential equations efficiently and yet are the equivalent of approximating the exact solutions by matching "n" terms of the Taylor series expansion.

Butcher (1987) derived the best RK pair along with an error estimate and by all statistical measures it appeared as the RK-Butcher algorithms. This RK-Butcher algorithm is nominally considered sixth order since it requires six functions evaluation, but in actual practice the working order is closer to five (fifth order).

Bader (1987,1988) introduced the RK-Butcher algorithm for finding the truncation error estimates and intrinsic accuracies and the early detection of stiffness in coupled differential equations that arises in theoretical chemistry problems. Recently Murugesh and Murugesan (2006) used the RK-Butcher algorithm for time-multiplexing scheme of Cellular Neural Networks. Oliveria (1999) introduced the popular RK-Gill algorithm for evaluation of effectiveness factor of immobilized enzymes.

Lee and Pineda de Gyvez (1994) introduced Euler, Improved Euler Predictor-Corrector and Fourth-Order Runge-Kutta algorithms in Raster CNN simulation. In this article, we consider the same problem (discussed by Lee and Pineda de Gyvez (1994)) but presenting a different approach using the algorithms such as Euler, RK-Gill and RK-Butcher with more accuracy.

CELLULAR NEURAL NETWORKS

The basic circuit unit of CNN is called a cell. 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 $t>0$ and, after the transient has settled down, a cellular neural network always approaches one of its stable equilibrium points. This 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.

Corresponding Author: V. Murugesh, Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli-620 015, Tamil Nadu, India Tel: +91-9865409292

The first order nonlinear differential equation defining the dynamics of a cellular neural network cell can be written as follows

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

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

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(i,j) of cell C(i,j) 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 I and the cloning templates determine the transient behavior of the cellular nonlinear network.

MULTI-LAYER RASTER CNN SIMULATION

Multi-Layer Raster CNN simulation is an image scanning procedure for solving the system of difference equations of CNN. In this approach the templates A and B are applied to a square subimage area centred at (x,y), whose size is the same as that of the templates. The centre of the templates are then moved left to right pixel by pixel from the top left corner to the bottom right corner applying the A and B templates at each location (x,y) to solve the system of difference equations. This full scanning of the image is repeated for each time-step which is defined as iteration. The processing is stopped when the states of all CNN cells have converged to the steady-state values.

A simplified algorithm is presented below for this approach. The part where the integration is involved is explained in the Numerical Integration Techniques section.

Algorithm: (Multi-layer raster CNN simulation)

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

```
/* M,N = # of rows/columns of the image */
while (converged_cells<total # of cells) {
   for(layer = 0; layer<3; layer ++) {
     for (i=1; i<=M; i++)
        for (j=1; j<=N; j++) {
          if (convergence_flag [i] [j])
          continue;/* currnet cell already
          converged */
          /* calculation of the next state */
            x_ij(t_n+1) = x_ij(t_n) + ∫_{t_n}^{t_n+H} f(x(t_n))dt
          /* convergence criteria */
  if ( dx_ij(t_n)/dt = 0 ) and y_kl = ±1, ∀C(k, l)∈N_r(i, j)
  {
          convergence_flag[i][j]=1;
          converged_cells++;
  }/* end for */
  /* update the state values of the whole image */
  for (i=1; i<= M; i++)
        for (j=1; j<= N; j++)
              {
                 if (convergence_flag [i][j]) continue;
                 X_layer,ij(t_n) = x_layer,ij(t_n+1);
              }
  #_of_iteration++;
}/* end while */
```

The Multi-Layer 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:

$$g(x,y) = T(f(x,y)) \quad (2)$$

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

NUMERICAL INTEGRATION TECHNIQUES

The CNN is described by a system of nonlinear differential equations. Therefore, it is necessary to discretize the differential equation for performing simulations. For computational purpose, a normalized time differential equations describing CNN is used by Nossek *et al.* (1992).

$$f'(x(\pi\tau)) := \frac{dx_{ij}(\pi\tau)}{dt} = -x_{ij}(\pi\tau) + \sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)$$

$$y_{kl}(\pi\tau) + \sum_{C(k,l)\in N_r(i,j)} B\begin{pmatrix} i,j; \\ k,l \end{pmatrix} u_{kl} + I$$

$$y_{ij}(t) = \frac{1}{2}\left(|x_{ij}(t)+1| - |x_{ij}(t)-1|\right) \tag{3}$$

Where τ is the normalized time. For the purpose of solving the initial-value problem, well established numerical integration techniques are used. These methods can be derived using the definition of the definite integral

$$x_{ij}((n+1)\tau) - x_{ij}(\pi\tau) = \int_{\tau_n}^{\tau_{n+1}} f'(x(\pi\tau))d(\pi\tau) \tag{4}$$

Three of the most widely used Numerical Integration Algorithms are used in CNN Raster Simulation described here. They are the Euler's Algorithm, RK-Gill Algorithm discussed by Oliveria (1999) and the RK-Butcher Algorithm discussed by Badder (1987, 1988) and Murugesh and Murugesan (2006).

Euler algorithm: Euler's method is the simplest of all algorithms for solving ODEs. It is explicit formula which uses the Taylor-series expansion to calculate the approximation.

$$x_{ij}((n+1)\tau) = x_{ij}(\pi\tau) + \tau f'(x(\pi\tau)) \tag{5}$$

RK-Gill algorithm: The RK-Gill algorithm discussed by Oliveria (1999) is an explicit method requiring the computation of four derivatives per time step. The increase of the state variable x^{ij} is stored in the constant k^{ij}_1. This result is used in the next iteration for evaluating k^{ij}_2. The same must be done for k^{ij}_3 and k^{ij}_4.

$$\begin{aligned}
k^{ij}_1 &= f'(x_{ij}(\pi\tau)) \\
k^{ij}_2 &= f'\left(x_{ij}(\pi\tau) + \frac{1}{2}k^{ij}_1\right) \\
k^{ij}_3 &= f'\left(x_{ij}(\pi\tau) + \left(\frac{1}{\sqrt{2}} - \frac{1}{2}\right)k^{ij}_1 + \left(1 - \frac{1}{\sqrt{2}}\right)k^{ij}_2\right) \\
k^{ij}_4 &= f'\left(x_{ij}(\pi\tau) - \frac{1}{\sqrt{2}}k^{ij}_2 + \left(1 + \frac{1}{\sqrt{2}}\right)k^{ij}_3\right)
\end{aligned} \tag{6}$$

The final integration is a weighted sum of the four calculated derivatives:

$$x_{ij}((n+1)\tau) = x_{ij} + \frac{1}{6}\begin{bmatrix} k^{ij}_1 + \left(2-\sqrt{2}\right)k^{ij}_2 + \\ \left(2+\sqrt{2}\right)k^{ij}_3 + k^{ij}_4 \end{bmatrix} \tag{7}$$

RK-butcher algorithm: The RK-Butcher algorithm discussed by Badder (1987, 1988) and Murugesh and Murugesan (2006), is an explicit method. It starts with a simple Euler step. The increase of the state variable x^{ij} is stored in the constant k^{ij}_1. This result is used in the next iteration for evaluating k^{ij}_2. The same must be done for k^{ij}_3, k^{ij}_4, k^{ij}_5 and k^{ij}_6.

$$\begin{aligned}
k^{ij}_1 &= \tau f'(x_{ij}(\pi\tau)) \\
k^{ij}_2 &= \tau f'\left(x_{ij}(\pi\tau) + \frac{1}{4}k^{ij}_1\right) \\
k^{ij}_3 &= \tau f'\left(x_{ij}(\pi\tau) + \frac{1}{8}k^{ij}_1 + \frac{1}{8}k^{ij}_2\right) \\
k^{ij}_4 &= \tau f'\left(x_{ij}(\pi\tau) - \frac{1}{2}k^{ij}_2 + k^{ij}_3\right) \\
k^{ij}_5 &= \tau f'\left(x_{ij}(\pi\tau) + \frac{3}{16}k^{ij}_1 + \frac{9}{16}k^{ij}_4\right) \\
k^{ij}_6 &= \Delta t f\begin{pmatrix} x_{ij}(\pi\tau) - \frac{3}{7}k^{ij}_1 + \frac{2}{7}k^{ij}_2 + \\ \frac{12}{7}k^{ij}_3 - \frac{12}{7}k^{ij}_4 + \frac{8}{7}k^{ij}_5 \end{pmatrix}
\end{aligned} \tag{8}$$

The final integration is a weighted sum of the five calculated derivatives:

$$x_{ij}((n+1)\tau) = \frac{1}{90}\begin{pmatrix} 7k^{ij}_1 + 32k^{ij}_3 + 12k^{ij}_4 \\ +32k^{ij}_5 + 7k^{ij}_6 \end{pmatrix} \tag{9}$$

where f(.) is computed according to (1). There are many methods available to us for this purpose. Among all the methods, RK-Butcher algorithm is a very efficient for solving this problem.

SIMULATION RESULTS AND COMPARISONS

All the simulation reported here are performed using a SUN BLADE 1500 workstation and the simulation time used for comparisons is the actual CPU time used. The input image format is the windows bitmap format (xbm), which is commonly available and easily convertible from popular image formats like GIF or JPEG.

Figure 1 shows results of the raster simulator obtained from a complex image of 1,25,600 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 Fig. 1b and c, respectively.

(a) (b) (c)

Fig. 1: Image processing (a) Original image (b) After averaging template (c) After averaging and edge detection templates

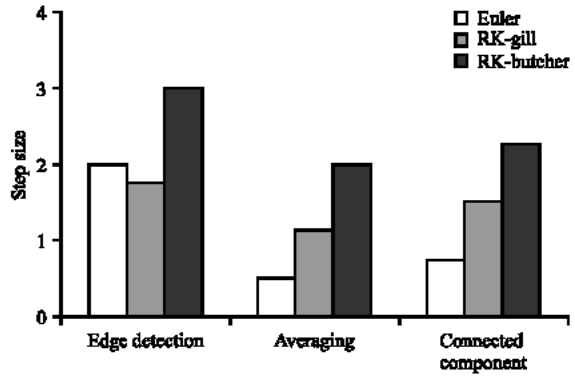

Fig. 2: Maximum step size still yields convergence for three different templates

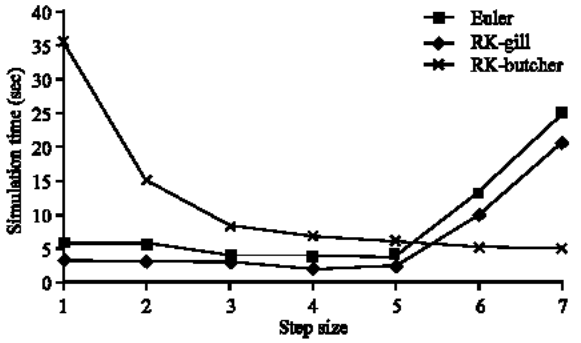

Fig. 3: Simulation time comparison of the three methods using the averaging template

Since speed is one of the main concerns in the simulation, finding the maximum step size that still yields convergence for a template can be helpful in speeding up the system. The speed-up can be achieved by selecting an appropriate Δt for that particular template. Even though the maximum step size may slightly vary from one image to another, the values in Fig. 2 still serve as good references. These results were obtained by trial and error over more than 100 simulations on a diamond figure. If the step size is chosen is too small, it might take many iterations, hence longer time, to achieve convergence. On the other hand, if the step size taken is too large, it might not converge at all or it would be converges to erroneous steady state values; the latter remark can be observed for the Euler algorithm.

The results of Fig. 3 were obtained by simulating a small image of size 16×16 (256 pixels) using Averaging template on a diamond figure.

CONCLUSIONS

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 three ways: (1) Depending on the accuracy required for the simulation, the user can choose from three numerical integration methods (2) The input image format is the X-Windows bitmap (xbm), which is commonly available and (3) The input image can be of any size, allowing simulation of images available in common practices.

REFERENCES

Bader, M., 1987. A comparative study of new truncation error estimates and intrinsic accuracies of some higher order Runge-Kutta algorithms. Comput. Chem., 11: 121-124.

Bader, M., 1988. A new technique for the early detection of stiffness in coupled differential equations and application to standard Runge-Kutta algorithms. Theor. Chem. Accounts, 99: 215-219.

Butcher, J.C., 1987. The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods. John Wiley and Sons, UK.

Butcher, J.C., 2003. The Numerical Analysis of Ordinary Differential Equations. John Wiley and Sons, UK.

Chua, L.O. and L. Yang, 1988a. Cellular Neural Networks: Theory. IEEE Trans. Circuits Sys., 35: 1257-1272.

Chua, L.O. and L. Yang, 1988b. Cellular Neural Networks: Applications. IEEE Trans. Circuits Sys., 35: 1273-1290.

Lee, C.C. and J. Pineda de Gyvez, 1994. Single-Layer CNN Simulator. Intl. Symposium Circuits Sys., 6: 217-220.

Murugesh, V. and K. Murugesan, 2006. Simulation of time-multiplexing CNN with numerical integration algorithms. Lecture Notes in Computer Science (LNCS), 3991: 457-464.

Nossek, J.A., G. Seiler, T. Roska and L.O. Chua, 1992. Cellular neural networks: Theory and circuit design. Intl. J. Circuit Theory Applic., 20: 533-553.

Oliveria, S.C., 1999. Evaluation of effectiveness factor of immobilized enzymes using Runge-Kutta-Gill method: how to solve mathematical undetermination at particle center point?. Bio Process Eng., 20: 185-187.