International Journal of Computers, Communications & Control
Vol. II (2007), No. 4, pp. 367-374

An Efficient Numerical Integration Algorithm for Cellular Neural Network Based Hole-Filler Template Design

V. Murugesh, Krishnan Batri

Abstract: This paper presents, a design method for the template of the hole-filler used to improve the performance of the character recognition using Numerical integration algorithms. This is done by analyzing the features of the hole-filler template and the dynamic process of CNN and by using popular numerical algorithms to obtain a set of inequalities satisfying its output characteristics as well as the parameter range of the hole-filler template. Some simulation results and comparisons are also presented.
Keywords: Cellular Neural Networks; Euler Algorithm; RK-Gill Algorithm; RK-Butcher Algorithm; Ordinary differential equations, Hole-filler.

1 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 in 1988 (by Chua and Yang [1, 2], it has been the subject 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 a computational technique as well as a subject of research, which are discussed by Butcher [3, 4]. 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 these are the equivalent of approximating the exact solutions by matching 'n' terms of the Taylor series expansion.

Butcher [3] derived the best RK pair along with an error estimate and by all statistical measures it appeared as the RK-Butcher algorithm. This RK-Butcher algorithm is nominally considered as sixth order, since it requires six function evaluations, but in actual practice the "working order" is close to five (fifth order).

Bader [4, 5] 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 arise in theoretical chemistry problems. Recently Devarajan et al [7] used the RK-Butcher algorithm for finding the numerical solution of an industrial robot arm control problem. Oliveria [8] introduced the popular RK-Gill algorithm for the evaluation of 'effectiveness factor' of immobilized enzymes.

In this paper, we describe the dynamic behavior of CNN in section 2, Hole-filler template design ideas in Section 3, Numerical integration algorithms and its description is shown in Section 4, and simulation results in Section 5.

2 Dynamic Analysis of CNN

The dynamic equation of cell $C(i,j)$ in an M x N cellular neural network is given by Chua and Yang [1, 2].

$$C\frac{dx_{ij}(t)}{dt} = -\frac{1}{R_x}x_{ij}(t) + \sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)Y_{kl}(t) + \sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l)U_{kl} + I \tag{1}$$

$$Y_{ij}(t) = \left[\left|x_{ij}(t)+1\right| - \left|x_{ij}(t)\right| - 1\right], 1 \le i \le M, 1 \le j \le n \tag{2}$$

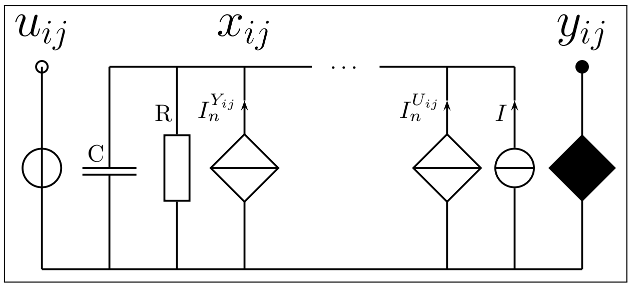

Figure 1: CNN–Cell

wherex_{ij}, y_{ij} and u_{ij} are the state voltage, output voltage and input voltage respectively and they are functions of time t. R_x is a linear resistance, C is a linear capacitor, and $A(i,j;k,l)$ and $B(i,j;k,l)$ are the transconductances of the output and input voltages of $C(k,l)$ with respect to $C(i,j)$ called the cloning templates of CNN. $N_r(i,j)$ denotes the r^{th}- neighbor of $C(i,j)$ and I is an independent current source. From equation (2) one can see that the output voltage is nonlinear. We can rewrite the cell equation (1) as follows:

$$C\frac{dx_{ij}(t)}{dt} = -f\left[x_{ij}(t)\right] + g(t) \tag{3}$$

Where

$$f\left[x_{ij}(t)\right] = \frac{1}{R_x}x_{ij}(t) \tag{4}$$

$$g(t) = \sum_{\substack{C(k,l)\in N_r(i,j)\\ C(k,l)\neq C(i,j)}} A(i,j;k,l)Y_{kl}(t) + \sum_{C(k,l)} B(i,j;k,l)U_{kl} + I \tag{5}$$

3 Hole-filler Template Design

The Hole-Filler is a cellular neural network discussed by Yin et al [9], which fills up all the holes and remains unaltered outside the holes in a bipolar image. Let $R_x = 1$, $C = 1$ and let +1 stand for the black pixel and -1 for the white one. We shall discuss the images having holes enclosed by the black pixels, when the bipolar image is input with $U = \{u_{ij}\}$ into CNN. The initial state values are set as $X_{ij}(0) = 1$. From the equation (2) the output values are $Y_{ij}(0) = 1$, $1 \le i \le M$, $1 \le j \le N$.

Suppose that the template A and B and the independent current source I are given as

$$A = \begin{bmatrix} 0 & a & 0 \\ a & b & a \\ 0 & a & 0 \end{bmatrix}, \quad a > 0, \quad b > 0, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad I = -1 \tag{6}$$

Where the template parameters a and b are to be determined. In order to make the outer edge cells become the inner ones, normally auxiliary cells are added along the outer boundary of the image, and their state values are set to zeros by circuit realization, resulting in the zero output values. The state equation (1) can be rewritten as

$$\frac{dx_{ij}(t)}{dt} = -x_{ij}(t) + \sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)Y_{ij}(t) + 4u_{ij}(t) - I \tag{7}$$

For the cell $C(i,j)$, we call the cells $C(i+1,j)$, $C(i-1,j)$, $C(i,j+1)$ and $C(i,j-1)$ to be the non-diagonal cells. Here, several cases are to be considered.

Case 1: The input value $u_{ij} = +1$ for cell $C(i, j)$, signaling the black pixel. Because the initial state value of the cell $C(i, j)$ has been set to 1, $x_{ij}(0) = 1$, and from equation (2) its initial output value is also $y_{ij}(0) = 1$. According to the hole-filler demands, its eventual output should be $y_{ij}(\infty) = 1$. To obtain this result we set

$$\frac{dx_{ij}(t)}{dt} \geq 0 \tag{8}$$

Substituting this input $u_{ij} = 1$ and equation (6) into equation (7), we obtain

$$\frac{dx_{ij}(t)}{dt} = -x_{ij}(t) + a\left[y_{(i-1)j}(t) + y_{(i+1)j}(t) + y_{i(j-1)}(t) + y_{i(j+1)}(t)\right] + by_{ij}(t) + 3 \tag{9}$$

Combining equations (8) and (9) and considering the minimum value of $x_{ij}(t) = 1$ this case yields

$$a\left[y_{(i-1)j}(t) + y_{(i+1)j}(t) + y_{i(j-1)}(t) + y_{i(j+1)}(t)\right] + by_{ij}(t) + 2 \geq 0 \tag{10}$$

To facilitate our discussion, two sub cases are distinguished.

Sub Case 1: The cell $C(i, j)$ is inside the holes. Since $x_{ij}(0) = 1$, from equation (2) its initial output value $y_{ij}(0) = 1$. Considering equations (8) and (2), $y_{ij}(t) \geq 1$. According to the hole-filler demands, its initial output of non-diagonal black pixels should not be changed inside the holes. The weights of a and b are equal to +4 and +1, respectively.

Since $A(i, j; k, l) > \frac{1}{R_x}$ the parameter b is found to be $b > 1$, or

$$4a + b + 2 \geq 0, \quad b > 1 \tag{11a}$$

Sub Case 2: The cell $C(i, j)$ is outside the holes. To satisfy equation (10), we need to check only the minimum value on the left-hand side of equation (10). This is true when there are four non-diagonal white pixels around the cell $C(i, j)$, where the weight of a in equation (10) is -4. Since $y_{ij}(t) \geq 1$, the weight of b is equal to 1. Combining this with $b > 1$ gives

$$-4a + b + 2 \geq 0, \quad b > 1 \tag{11b}$$

Case 2: The input value of cell $C(i, j)$ is $u_{ij} = 1$, signaling the white pixel. Substituting this input value in equation (7) gives

$$\frac{dx_{ij}(t)}{dt} = -x_{ij}(t) + a\left[y_{(i-1)j}(t) + y_{(i+1)j}(t) + y_{i(j-1)}(t) + y_{i(j+1)}(t)\right] + by_{ij}(t) - 5 \tag{12}$$

Sub Case 1: The cell $C(i, j)$ is inside the holes. Since $x_{ij}(0) = 1$, from equation (2) its initial output value is $y_{ij}(0) = 1$. According to the hole-filler demands, the holes should be filled by the black pixels, whereas its initial black pixels remain unaltered:

$$\frac{dx_{ij}(t)}{dt} \geq 0 \tag{13}$$

Combining equations (12) and (13) and considering $x_{ij}(t) \geq 1$ yields

$$a\lfloor y_{(i-1)j}(t) + y_{(i+1)j}(t) + y_{i(j-1)}(t) + y_{i(j+1)}(t)\rfloor + by_{ij}(t) - 6 \geq 0 \tag{14}$$

where we use the minimum value of $x_{ij}(t)$ in equation (12). Since the cell is inside the holes, its initial output of non-diagonal black pixels remain unchanged. The weight of a and b are equal to +4 and +1, respectively. Combining this with $b > 1$ gives

$$4a + b - 6 \geq 0, \quad b > 1 \tag{15}$$

Sub Case 2: The cell $C(i,j)$ is outside the holes. Since $x_{ij}(0)=1$, from equation (2) its initial output value is $y_{ij}(0)=1$. According to the hole-filler demands, the final output of this cell should be white, and in this case $y_{ij}(\infty)\leq -1$.

$$\frac{dx_{ij}(t)}{dt}<0 \tag{16}$$

Combining equations (12) and (16) and considering $x_{ij}(t)\leq 1$. we get

$$a\lfloor y_{(i-1)j}(t)+y_{(i+1)j}(t)+y_{i(j-1)}(t)+y_{i(j+1)}(t)\rfloor+by_{ij}(t)-6<0 \tag{17}$$

where we use the maximum value of $x_{ij}(t)$ in equation (12).

Initially $y_{ij}(0)=1$. How can the output of cell $C(i,j)$ be changed to -1?. Where does this change begin?. First we consider the situation where the change begins from the inside of the bipolar image. If the maximum value on the left-hand side in equation (17) is less than zero, equation (17) holds. Inside the image and outside the holes, the maximum weights of a and b are +4 and +1, respectively. This case was described by equation (15). In fact, the change of the output of the cell $C(i,j)$ is like a wave propagating from the edges to the inside of the image and it is verified from the simulated result. Therefore, we should first consider the edge cell $C(i,j), i=1$ or $M, j=1$ or N. For this the maximum weight of a in equation (17) is +3, which is also the maximum weight of a outside the holes. The maximum weight of b is +1, occurring at the initial time:

$$3a+b-6<0,\quad b>1 \tag{18}$$

Combining Cases 1 and 2, we obtain

$$\begin{array}{rcl} 3a+b-6 & < & 0,\\ 4a+b-6 & \geq & 0,\\ -4a+b+2 & \geq & 0. \end{array} \tag{19}$$

4 Numerical Integration Algorithms

The CNN dynamics on a digital architecture requires discretization in time and suitable numerical integration algorithms. Three of the most widely used Numerical Integration Algorithms are used in Raster CNN Simulation described here. They are the Euler's Algorithm, RK-Gill Algorithm discussed by Oliveria [8] and the RK-Butcher Algorithm discussed by Badder [5, 6] and Murugesh and Murugesan [10, 11, 12].

4.1 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.

$$x_{ij}((n+1)\tau)=x_{ij}(\pi\tau)+\tau f'(x(\pi\tau)) \tag{20}$$

4.2 RK-Gill Algorithm

The RK-Gill algorithm discussed by Oliveria [8] 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_1^{ij}. This result is used in the next iteration for evaluating k_2^{ij}. The same must be done for k_3^{ij} and k_4^{ij}.

$$\begin{aligned} k_1^{ij} &= f'(x_{ij}(\tau\pi)) \\ k_2^{ij} &= f'\left(x_{ij}(\tau\pi) + \frac{1}{2}k_1^{ij}\right) \\ k_3^{ij} &= f'\left(x_{ij}(\tau\pi) + \left(\frac{1}{\sqrt{2}} - \frac{1}{2}\right)k_1^{ij} + \left(1 - \frac{1}{\sqrt{2}}\right)k_2^{ij}\right) \\ k_4^{ij} &= f'\left(x_{ij}(\tau\pi) - \frac{1}{\sqrt{2}}k_2^{ij} + \left(1 + \frac{1}{\sqrt{2}}\right)k_3^{ij}\right) \end{aligned} \tag{21}$$

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

$$x_{ij}((n+1)\tau) = x_{ij} + \frac{1}{6}\left[kij_1 + \left(2-\sqrt{2}\right)k_2^{ij} + \left(2+\sqrt{2}\right)k_3^{ij} + k_4^{ij}\right] \tag{22}$$

4.3 RK-Butcher Algorithm

The RK-Butcher algorithm discussed by Badder [5, 6] and Murugesh and Murugesan [10, 11, 12], 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_1^{ij}. This result is used in the next iteration for evaluating k_2^{ij} . The same must be done for k_3^{ij}, k_4^{ij}, k_5^{ij} and k_6^{ij}.

$$\begin{aligned} k_1^{ij} &= \tau f'(x_{ij}(\pi\tau)) \\ k_2^{ij} &= \tau f'\left(x_{ij}(\pi\tau) + \frac{1}{4}k_1^{ij}\right) \\ k_3^{ij} &= \tau f'\left(x_{ij}(\pi\tau) + \frac{1}{8}k_1^{ij} + \frac{1}{8}k_2^{ij}\right) \\ k_4^{ij} &= \tau f'\left(x_{ij}(\pi\tau) - \frac{1}{2}k_2^{ij} + k_3^{ij}\right) \\ k_5^{ij} &= \tau f'\left(x_{ij}(\pi\tau) + \frac{3}{16}k_1^{ij} + \frac{9}{16}k_4^{ij}\right) \\ k_6^{ij} &= \Delta t f\left(x_{ij}(\pi\tau) - \frac{3}{7}k_1^{ij} + \frac{2}{7}k_2^{ij} + \frac{12}{7}k_3^{ij} - \frac{12}{7}k_4^{ij} + \frac{8}{7}k_5^{ij}\right) \end{aligned} \tag{23}$$

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

$$x_{ij}((n+1)\tau) = \frac{1}{90}\left(7k_1^{ij} + 32k_3^{ij} + 12k_4^{ij} + 32k_5^{ij} + 7k_6^{ij}\right) \tag{24}$$

5 Simulated Results

This Hole-filler template has been simulated using Pentium IV Machine with 3.0 Ghz. speed using different Numerical integration algorithms. The Settling time T and integration time T_s is obtained with various step sizes is to be displayed below in the Table-1. The settling time T_s describes the time from start of integration until the last cell leaves the interval [-1.0, 1.0] which is based on certain limit (e.g.,$\left|\frac{dx}{dt} < 0.01\right|$). The simulation shows the desired output for every cell. We use +1 and -1 to indicate the black and white pixels, respectively.

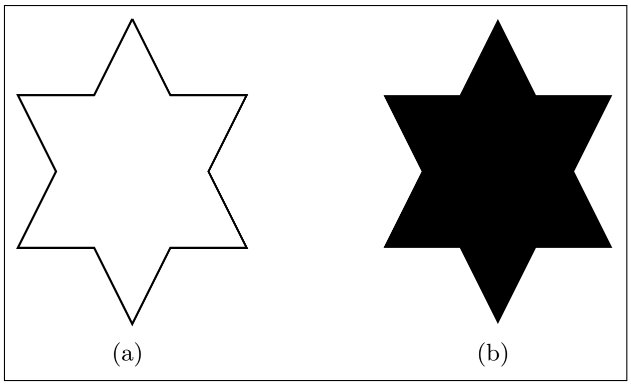

Figure 2: Image Before and After Hole Filling

Step Size	Euler Algorithm		RK-Gill Algorithm		RK-Butcher Algorithm	
	Settling Time(T)	Integration Time(T_s)	Settling Time(T)	Integration Time(T_s)	Settling Time(T)	Integration Time(T_s)
0.5	6.5	2.5	6.8	2.4	5.8	2.4
0.6	15.5	12.7	16.4	13.7	11.4	12.5
0.7	32.5	28.3	32.0	27.4	30.0	27.2
0.8	35.0	30.7	34.6	30.0	32.4	29.6
0.9	36.8	32.6	36.6	32.0	34.2	31.6
1.0	37.9	33.6	37.6	33.0	36.0	32.8
1.5	44.8	36.8	45.7	36.9	41.1	36.0
2.0	47.4	43.2	48.2	43.6	46.2	42.8
2.5	50.6	45.6	52.6	44.5	48.3	44.7
3.0	53.5	49.3	54.8	50.2	52.3	49.2

Table 1: Simulated Results of Hole-Filler Template Design

Example
The templates A, B and I are given as follows:

$$A = \begin{bmatrix} 0 & 1.0 & 0 \\ 1.0 & 3.0 & 1.0 \\ 0 & 1.0 & 0.0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad I = -1.0$$

Using the simulation program developed in C++, the input image is shown in Figure-2(a) and the output image in Figure-2(b). The obtained result is represented in Table-1. From the table-1, we find that RK-Butcher algorithm yields less settling time and integration time compared to Euler and RK-Gill algorithms.

6 Conclusion

It is shown that the cellular neural network based hole-filler template could be designed from its dynamic behavior using different numerical algorithms, and also the template for other cellular neural network can similarly be designed. The hole is filled and the outside image remains the same. The templates of the cellular neural network are not unique and this is important in its implementation.

Bibliography

[1] L. O. Chua, L. Yang, Cellular Neural Networks: Theory, *IEEE Transactions on Circuits and Systems*, Vol. 35, pp. 1257 - 1272, 1988.

[2] L. O. Chua, L. Yang, Cellular Neural Networks: Applications, *IEEE Transactions on Circuits and Systems*, Vol. 35, pp. 1273 - 1290, 1988.

[3] J. C. Butcher, *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*, Chichester: John Wiley, 1987.

[4] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, Chichester: John Wiley, 2003.

[5] M. Bader, *A comparative study of new truncation error estimates and intrinsic accuracies of some higher order Runge-Kutta algorithms*, Computers & Chemistry, Vol. 11 pp. 121-124, 1987.

[6] M. Bader, *A new technique for the early detection of stiffness in coupled differential equations and application to standard Runge-Kutta algorithms*, Theoretical Chemistry Accounts, Vol. 99, pp. 215-219, 1988.

[7] G. Devarajan, V. Murugesh, K. Murugesan, *Numerical Solution of Second-order Robot Arm Control Problem using Runge-Kutta Butcher Algorithm*, International Journal of Computer Mathematics, Vol. 83 pp. 345-356, 2006.

[8] S. C. Oliveira, *Evaluation of effectiveness factor of immobilized enzymes using Runge-Kutta-Gill method: how to solve mathematical undetermination at particle center point?*, Bio Process Engineering, Vol. 20, pp. 85-187, 1999.

[9] C. L. Yin, J. L. Wan, H. Lin, W. K. Chen, *The cloning template design of a cellular neural network*, Journal of the Franklin Institute, Vol. 336, pp. 903-909, 1999.

[10] V. Murugesh, K. Murugesan, *Comparison of Numerical Integration Algorithms in Raster CNN Simulation*, Lecture Notes in Computer Science, Vol. 3285 pp. 115-122, 2004.

[11] V. Murugesh, K. Murugesan, *Simulation of Cellular Neural Networks using the RK-Butcher algorithm*, International Journal of Management and Systems, Vol. 21, pp. 65-78, 2005.

[12] V. Murugesh, K. Murugesan, *Simulation of Time-Multiplexing Cellular Neural Networks with Numerical Integration Algorithms*, Lecture Notes in Computer Science, Vol. 3991, pp. 115-122, 2005.

V. Murugesh
Department of Information and Communication Engineering
Hannam University
133 Ojung-dong Daeduk-gu, Daejeon 306-791, Republic of Korea
E-mail: murugesh72@gmail.com

K. Batri
Department of Computer Science and Engineering
Muthyammal Engineering College
Rasipuram 637 408, India
E-mail: krishnan.batri@gmail.com

Received: July 25, 2007

Dr. V. Murugesh obtained his Bachelor of Science in Computer Science and Master of Computer Applications degree from Bharathiar University, Coimbatore, India during 1992 and 1995 respectively. Completed his PhD in Computer Science from Bharathidasan University, Tiruchirappalli, India during 2006. He has held various positions at National Institute of Technology, Tiruchirappalli, India and Sona College of Technology, Salem, India. Currently, he is working as Assistant Professor in the Department of Information and Communication Engineering at Hannam University, Daejeon, Republic of Korea. His fields of interest are in Neural Network based Image Processing and Scientific computing. He has published more than 30 technical papers in International, National journals and conferences.

Krishnan Batri received the M.E. from Madurai Kamarj University in 2003. He is a Research Scholar with the Department of Computer Science and Engineering in the National Institute of Technology Tiruchirapalli, Tamil Nadu, India. Currently he is working as a Assistant Professor with the department of Computer Science and Engineering at Muthayammal Engineering College, Rasipuram, Tamilnadu, India. His research interests include Information Retrieval, Data fusion and Genetic algorithms.