Research Journal of Information Technology

ISSN 1815-7432

Academic Journals Inc.

www.academicjournals.com

Research Journal of Information Technology 1 (1): 1-16, 2009
ISSN 1815-7432

Time-Multiplexing CNN Simulation Using Limiting Formulas of RK(7,8)

R. Ponalagusamy and S. Senthilkumar
Department of Mathematics, National Institute of Technology,
Tiruchirappalli-620 015, Tamilnadu, India

Abstract: In this research, a versatile algorithm for simulating CNN arrays and time multiplexing is implemented using numerical integration algorithms. The approach, time-multiplexing simulation, plays a pivotal role in the area of simulating hardware models and testing hardware implementations of CNN. Owing to hardware limitations in practical sense, it is not possible to have a one-one mapping between the CNN hardware processors and all the pixels of the image. The simulator provides a solution by processing the input image block by block, with the number of pixels in a block being the same as the number of CNN processors in the hardware. Simulation results and comparison have also been presented to show the efficiency of the Numerical Integration Algorithms. In this research, RK-eight stage seventh order limiting formulas are implemented and it is found that the RK (7,8) algorithm outperforms well in comparison with the Explicit Euler, RK-Gill, RK-fifth order and RK-sixth order algorithms. A more quantitative analysis has been carried out to clearly visualize the goodness and robustness of the numerical algorithms.

Key words: Time-multiplexing, cellular neural network, numerical integration techniques, edge detection, RK-eight stage seventh order limiting formulas

INTRODUCTION

The uniqueness of Cellular Neural Networks (CNNs) are analog, time-continuous, non-linear dynamical systems and formally belong to the class of recurrent neural networks. CNNs have been proposed by Chua and Yang (1988a) and they have found that CNN has many important applications in signal and real-time image processing (Gonzalez *et al*., 2005). As Roska *et al*. (1994) have presented the first widely used simulation system which allows the simulation of a large class of CNN and is especially suited for image processing applications. It also includes signal processing, pattern recognition and solving ordinary and partial differential equations etc. Oliveira (1999) introduced the popular RK-Gill algorithm for evaluation of effectiveness factor of immobilized enzymes. 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 equivalent to five (fifth order). Ponalagusamy and Senthilkumar (2007a) have discussed RK-sixth order algorithm for raster CNN simulation. Lee and de Gyvez (1994) introduced Euler, Improved Euler, Predictor-Corrector and RK-fourth-order (quartic) algorithms in time-multiplexing CNN simulation. It is well known that RK-eight stage seventh order explicit limiting formulas are of order at most six. However, by taking the limit as the first abscissa approaches zero, the formulas can achieve seventh order. Such formulas are called limiting formulas which require the evaluations of the second derivatives of the solution. The possible order of s-stage explicit Runge-Kutta methods is s-1 for s = 5, 6, 7 but, they can achieve sth order in the limiting case where distance between some pairs of abscissas

Corresponding Author: R. Ponalagusamy, Department of Mathematics, National Institute of Technology, Tiruchirappalli-620015, Tamilnadu, India

approaches zero. Such formulas are known as s-stage sth order limiting formulas. Ono (1989) discussed five and six stage RK-type limiting formulas of orders numerically five and six. They are obtained by replacing the second derivatives involved in the limiting formulas with simplest numerical differentiation. The reason to perform is that the second derivatives in the limiting formulas does not need full significant figures carried in computation and the user can choose free parameters so as to minimize the error caused by numerical differentiation. In this research, the time multiplexing CNN simulation problem is solved with different approach using the algorithms such as Explicit Euler, RK-Gill, RK-Fifth order, RK-Sixth order and the RK-Eight stage seventh order limiting formulas (Mitsui and Shinohara, 1995) to yield higher accuracy with less error.

STRUCTURE AND FUNCTIONS OF CELLULAR NEURAL NETWORK

The general CNN architecture consists of M*N cells placed in a rectangular array. The basic circuit unit of CNN is called a cell and the array structure and Block diagram is shown in Fig. 1. It has linear and nonlinear circuit elements. Any cell, C(i,j), is connected only to its neighbor cells (adjacent cells interact directly with each other). This intuitive concept is known as neighborhood and is denoted by 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. For all time t > 0, the state of each cell is said to be bounded and after the transient has settled down, a cellular neural network always approaches one of its stable equilibrium points. It implies that the circuit will not oscillate. The dynamics of a CNN has both output feedback (A) and input control (B) mechanisms. The dynamics of a CNN network cell is governed by the first order nonlinear differential equation given:

$$c\frac{dx_{ij}(t)}{dt}=\frac{-1}{R}x_{ij}(t)+\sum_{c(k,l)\in N(i,j)}A(i,j;k,l)y_{kl}(t)+\sum_{c(k,l)\in N(i,j)}B(i,j;k,l)u_{kl}(t)+I,\ 1\le i\le M;1\le j\le N. \tag{1}$$

and the output equation is given by,

$$y_{ij}(t)=\frac{1}{2}\left[\left|x_{ij}(t)+1\right|-\left|x_{ij}(t)-1\right|\right],\quad 1\le i\le M;1\le j\le N.$$

Where:

c	=	A linear capacitor
x_{ij}	=	Denotes the state of cell C(i,j)
$x_{ij}(0)$	=	The initial condition of the cell
R	=	A linear resistor
I	=	An independent current source
$A(i,j;k,l)y_{kl}$ and $B(i,j;k,l)u_{kl}$	=	Voltage controlled current sources for all cells C(k,l) in the neighborhood N(i,j) of cell C(i,j)
y_{ij}	=	The output equation

For simulation purposes, a discretized form of Eq. 1 is solved within each cell to simulate its state dynamics. One common way of processing a large complex image is using a raster approach (Chua and Yang, 1988a). This approach implies that each pixel of the image is mapped onto a CNN processor. That is, it has an image processing function in the spatial domain that is expressed as:

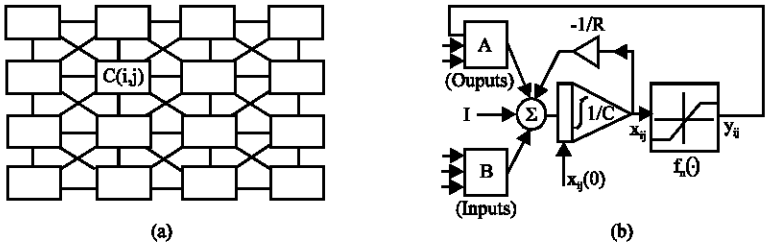


Fig. 1: Cellular neural networks: (a) Array structure and (b) Block diagram

$$g(x,y) = T(f(x,y)) \tag{2}$$

Where:

g(.) = The processed image
f(.) = The input image
T = An operator on f(.) defined over the neighborhood of (x,y)

It is an exhaustive process from the view of hardware implementation. For practical applications, in the order of 250,000 pixels, the hardware would require a large amount of processors which would make its implementation unfeasible. An alternative option to this scenario is multiplex the image processing operator.

TIME MULTIPLEXING SIMULATION APPROACH

In this procedure it is possible to define a block of CNN processors which will process a subimage whose number of pixels is equal to the number of CNN processors in the block. The processing within this subimage follows the raster approach adapted in Chua and Yang (1988b). Once convergence is achieved, a new subimage is processed. The same approach is being carried out until the whole image has been scanned. It is clear that with this approach the hardware implementation becomes feasible since the number of CNN processors is finite. Also, the entire image is scanned only once since each block is allowed to fully converge. An important point is to be noticed that the processed border pixels in each subimage may have incorrect values since they are processed without neighboring information only local interactions are important for the latency of CNNs. To overcome the aforementioned problem two sufficient conditions must be considered while performing time-multiplexing simulation. Alternatively, to ensure that each border cell properly interacts with its neighbors it is necessary to have the following. (1) To have a belt of pixels from the original image around the subimage and (2) to have pixel overlaps between adjacent subimages.

It is possible to quantize the processing error of any border cell C_{ij} with neighborhood radius of 1. By computing independently the error owing to the feed forward operator and interaction among cells for the two horizontally adjacent processing blocks, the absoulte processing error owed only to the effect of the B template is obtained by subtracting the erroneous state value from the error free states using Eq. 1. This gives,

$$\varepsilon_{ij}^{B} = \sum_{i=1}^{i=3} b_{i,j+1}\ \mathrm{sign}(u_{i,j+1}) \tag{3}$$

Where:

$b_{i,j+1}$ = The missing entries from the B template due to the absence of input signals $u_{i,j+1}$

sign(.) = The sign function

The latter function is used to represent the status of a pixel, e.g. black = 1 and white = -1. It is seen that the error is both image and template dependent. Alternatively, the steady state of a border cell may converge to an incorrect value due to the absence of its neighbors weighted input. Given the local interconnectivity properties of CNN, one can conclude that the minimum width of the input belt of pixels is equal to the neighborhood radius of the CNN.

Interaction Between Cells

In view of interation between cells, it is possible to compute the absolute error in a similar way. In absence of the B template for a moment, the error is expressed as:

$$\varepsilon_{ij}^{A} = \sum_{i=1}^{i=3} b_{i,j+1}\, y_{i,j+1}(t) \tag{4}$$

Where:

$a_{i,j+1}$ = The missing entries from the A template due to the absence of weighted output signals $y_{i,j+1}(t)$

In this case output signals depend on the state of their corresponding cells. The technique of an overlap of pixels between two adjacent blocks is proposed in order to minimize the error. The minimum overlap width must be proportional to two times of the neighborhood's radius of the CNN. The time-multiplexing procedure deals with iterating each block (subimage) until all CNN cells within the block converge. The block with converged cells will have state variables x which are the values used for the final output image shown in Fig. 2, the converged values from $Block_i$ are taken by the left side of the overlapped cells and the right side from $Block_{i+1}$. Further, the initial conditions for the border cells of $Block_{i+1}$ are the state values obtained while processing $Block_{i-1}$. In the simulator used here, the number of overlapping columns or rows between the adjacent blocks is defined by the user. Suppose if higher number of overlapping is obtained in columns or rows then it indicates the more accurate simulation of neighboring effects on the border cells. In case of practical applications the correct final

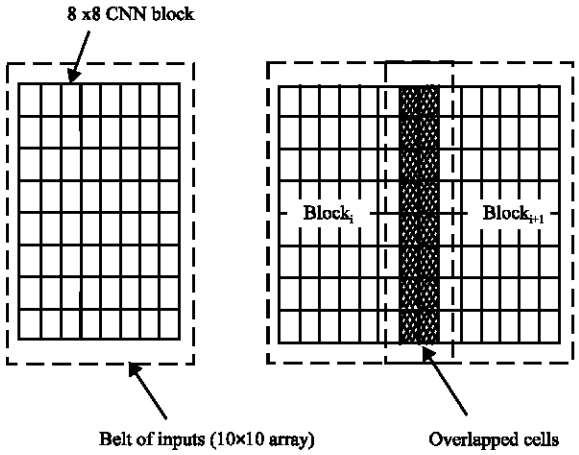


Fig. 2: CNN multiplexing with overlapped cells

state is of more importance than the transient states, an overlap of two is usually sufficient. An even number of overlapping cells is recommended, because the converged cells in the overlapped region can be evenly divided by the two adjacent blocks. Having the added overlapping feature, better neighboring interactions are achieved, but at the same time, an increase in computation time is unavoidable. On the other hand, by taking advantage of the fact that the original input image is been divided into small CNN subimages, the chance of a subimage having all its pixels black or white is high. This is another feature that can be added to the time multiplexing simulation to improve computation times. The savings in simulation time come from avoiding repetitive simulations of all-black and all-white subimages. The notion behind this timesaving scheme is that when the very first all-black/all-white block is encountered, after processing that block, the final states of the block are stored separately from the whole image. When subsequent all-black/all-white blocks are found, there is no need to simulate these blocks since the converged states are readily available in memory, which in turn leads to avoiding the most time consuming part of the simulation which is the numerical integration. The overall idea of this simulation approach is given below in the form of program fragment.

Program Fragment for Time-Multiplexing CNN Simulation

To understand the overall concept of overlap and belt approaches and raster simulation, the simplified version of algorithm is given:

Step 1: /* Defining the Variables */

$B = \{C_{ij} / i = 1,..., block\_x \wedge j = 1, ..., block\_y\}$
$P \subset B$ = set of border cells (lower left corner)
overlap = number of cell overlaps;
belt = width of the input belt
M = number of rows of the image
N = number of columns of the image
for (i = 0; i < M; i += block_x - overlap)
for (i = 0; j < N; j += block_y - overlap)
{

Step 2: /* Load the initial conditions for the cells in the block except for those in borders */

for (p = -belt; p < block_x + belt; p++)
for (q = -belt; q < block_y + belt; q++)
{

$$x_{i+p,j+j}(t_n) = \begin{cases} u_{ij} \\ 1 \quad \forall C_{i+p,j+j} \in B \\ -1 \end{cases}$$

Step 3: /* if the block contains all white or black do not process it */
if $(x_{i+p,j+q} = -1 \vee x_{i+p,j+q} = 1 \forall C_{i+p,j+q} \in B)$

{obtain the final states from memory;
continue;
}

}
do
{

Step 4: /* Perform the normal raster simulation */

for (p = 0; p < block_x; p++)
{
for (q = 0; q < block_y; q++)
{

Step 5: /* Compute of the next state excluding the belt of inputs */

$$x_{i+p,j+q}(t_{n+1}) = x_{i+p,j+q}(t_n) + \int_{t_n}^{t_{n+1}} f'(x_{i+p,j+q}(t_n))dt \forall C_{i+p,j+q} \subset B$$

Step 6: /* Check the convergence criteria */

$$\text{If } (\frac{dx_{i+p,j+q}(t_n)}{dt} = 0 \text{ and } y_{kl} = \pm 1 \forall c(k,l) \in N_r(i+p,j+q))$$

{
converged-cells++;

Step 7: /*update the state values of the entire image */
$x_{i+p,j+q}(t_n) = x_{i+p,j+q}(t_{n+1}) \forall C_{i+p,j+q} \in B$;} /* end for */
}
while (converged-cells < (block-x *block-y));

Step 8: /* store new state values excluding the ones corresponding to the border cells */

$A \leftarrow x_{ij} \forall C_{ij} \in B \setminus P$
}/* end for*/

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 behavioral simulation. For computational purposes, a normalized time differential equation describing CNN is used by Nossek *et al*. (1992).

$$f'(x(n\tau)) = \frac{dx_{ij}(n\tau)}{dt} = -x_{ij}(n\tau) + \sum_{c(k,l)\in N_r(i,j)} A(i,j;k,l)y_{kl}(n\tau) +$$
$$\sum_{c(k,l)\in N_r(i,j)} B(i,j;k,l)u_{kl}(n\tau) + I, 1 \le i \le M; 1 \le j \le N; \quad (5)$$
$$y_{ij}(n\tau) = \frac{1}{2}\left[|x_{ij}(n\tau)+1| - |x_{ij}(n\tau)-1|\right], \ 1 \le i \le M; 1 \le j \le N;$$

Where:

τ = The normalized time

For the purpose of solving the initial-value problem, well established Single Step methods of numerical integration techniques are used in (Lai and Leong, 1995). These methods can be derived using the definition of the definite integral:

$$x_{ij}((n+1)\tau)-x_{ij}(n\tau)) = \int_{\tau_n}^{\tau_{n+1}} f'(x(n\tau))d(n\tau) \tag{6}$$

Five types of numerical integration algorithms are used in time-multiplexing simulations which are described in the present study. They are Explicit Euler's Algorithm and the Fourth-Order Runge-Kutta Algorithm discussed respectively by Bader (1987, 1988). RK-Gill Algorithm was discussed by Oliveira (1999) and Ponalagusamy and Senthilkumar (2007) discussed in detail about the RK-Fifth and RK-Sixth order algorithms for raster CNN simulation.

Explicit Euler's Algorithm

Euler's method is the simplest of all algorithms for solving ordinary differential equations. It is an explicit formula which uses the Taylor-series expansion to calculate the approximation.

$$x_{ij}((n+1)\tau) = x_{ij}(n\tau)+\tau f'(x(n\tau)) \tag{7}$$

RK-Gill Algorithm

The RK-Gill algorithm was discussed by Oliveira (1999) is an explicit method which requires the computation of four derivatives per time step. The increase of the state variable x_{ij} is stored in the k_1^{ij} constant This result is used in the next iteration for evaluating k_2^{ij} and repeat the same process to obtain the values of k_3^{ij} and k_4^{ij}.

$$\begin{aligned}
k_1^{ij} &= f'(x_{ij}(n\tau)),\\
k_2^{ij} &= f'(x_{ij}(n\tau))+\frac{1}{2}k_1^{ij},\\
k_3^{ij} &= f'(x_{ij}(n\tau))+(\frac{1}{\sqrt{2}}-\frac{1}{2})k_1^{ij})+(1-\frac{1}{\sqrt{2}})k_2^{ij},\\
k_4^{ij} &= f'(x_{ij}(n\tau))-\frac{1}{\sqrt{2}}k_2^{ij}+(1+\frac{1}{\sqrt{2}})k_3^{ij},
\end{aligned} \tag{8}$$

Therefore, the final integration is a weighted sum of the four calculated derivates is given:

$$x_{ij}((n+1)\tau)= x_{ij}(n\tau)+\frac{1}{6}[k_1^{ij}+(2-\sqrt{2})k_2^{ij}+(2+\sqrt{2})k_3^{ij}+k_4^{ij}] \tag{9}$$

RK-Fifth Order Algorithm

The Fifth Order Runge-Kutta algorithm is an explicit method discussed by Badder (1987, 1988). It starts with a simple Euler method. 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 procedure must be repeated to compute the values of k_3^{ij}, k_4^{ij}, k_5^{ij} and k_6^{ij}.

$$
\begin{aligned}
k_1^{ij} &= \tau f'(x_{ij}(n\tau)),\\
k_2^{ij} &= \tau f'(x_{ij}(n\tau)) + \frac{1}{4}k_1^{ij},\\
k_3^{ij} &= \tau f'(x_{ij}(n\tau)) + (\frac{1}{8})k_1^{ij}) + (\frac{1}{8})k_2^{ij},\\
k_4^{ij} &= \tau f'(x_{ij}(n\tau)) - \frac{1}{2}k_2^{ij} + k_3^{ij},\\
k_5^{ij} &= \tau f'(x_{ij}(n\tau)) + \frac{3}{16}k_1^{ij} + \frac{9}{16}k_4^{ij},\\
k_6^{ij} &= \tau f'(x_{ij}(n\tau)) - \frac{3}{27}k_1^{ij} + \frac{2}{7}k_2^{ij} + \frac{12}{7}k_3^{ij} - \frac{12}{7}k_4^{ij} + \frac{8}{7}k_5^{ij}
\end{aligned}
\tag{10}
$$

Therefore, the final integration is a weighted sum of the five calculated derivatives which is given:

$$
x_{ij}((n+1)\tau) = x_{ij}(n\tau) + \frac{1}{90}[7k_1^{ij} + 32k_3^{ij} + 12k_4^{ij} + 32k_5^{ij} + 7k_6^{ij}] \tag{11}
$$

Where, f(.) is computed according to (1).

RK-Sixth Order Algorithm

The RK-Sixth order algorithm is an explicit method discussed by Ponalagusamy and Senthilkumar (2007b). It starts with a simple Euler method. 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 procedure must be repeated to compute the values of k_3^{ij}, k_4^{ij}, k_5^{ij} and k_6^{ij}.

$$
\begin{aligned}
k_1^{ij} &= \tau f'(x_{ij}(n\tau)),\\
k_2^{ij} &= \tau f'(x_{ij}(n\tau)) + \frac{1}{2}k^{ij}{}_1,\\
k_3^{ij} &= \tau f'(x_{ij}(n\tau)) + (\frac{2}{9})k_1^{ij}) + (\frac{4}{9})k_2^{ij},\\
k_4^{ij} &= \tau f'(x_{ij}(n\tau)) + \frac{7}{36}k_1^{ij} + \frac{2}{9}k_2^{ij} - \frac{1}{12}k_3^{ij},\\
k_5^{ij} &= \tau f'(x_{ij}(n\tau)) - \frac{35}{144}k_1^{ij} - \frac{55}{36}k_2^{ij} + \frac{35}{48}k_3^{ij} + \frac{15}{8}k_4^{ij},\\
k_6^{ij} &= \tau f'(x_{ij}(n\tau)) - \frac{1}{360}k_1^{ij} - \frac{11}{36}k_2^{ij} - \frac{1}{8}k_3^{ij} + \frac{1}{2}k_4^{ij} + \frac{1}{10}k_5^{ij},\\
k_7^{ij} &= \tau f'(x_{ij}(n\tau)) - \frac{41}{260}k_1^{ij} + \frac{22}{13}k_2^{ij} + \frac{43}{156}k_3^{ij} - \frac{118}{39}k_4^{ij} + \frac{32}{195}k_5^{ij} + \frac{80}{39}k_6^{ij}
\end{aligned}
\tag{12}
$$

Therefore, the final integration is a weighted sum of the seven calculated derivatives which is given:

$$
x_{ij}((n+1)\tau) = x_{ij}(n\tau) + [\frac{13}{200}k_1^{ij} + \frac{11}{40}k_3^{ij} + \frac{11}{40}k_4^{ij} + \frac{4}{25}k_5^{ij} + \frac{4}{25}k_6^{ij} + \frac{13}{200}k_7^{ij}] \tag{13}
$$

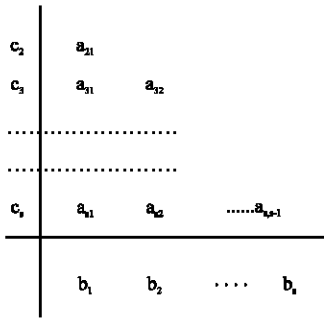
Where, f(.) is computed according to (1).

RK-Eight Stage Seventh Order Limiting Formulas

Let us consider an initial value problem

$$\frac{dx}{dt} = f(t,x), x(t_0) = x_0$$

Where, f and x are vectors and f is assumed to be differentiable sufficiently often for the definition to be meaningful. The parameters of an s-stage explicit Runge-Kutta method are represented in the following Butcher array (Butcher, 1987).



and x_i is used to denote the x ordinate at the abscissa c_i namely,

$$x_i = x_n + h\sum_{i=1}^{i-1} a_{ij} f_j$$

Where:
$f_1 = (t_n, x_n)$, $f_i = f(t_n + c_i h, x_i)$ $(i = 2,3\ldots,s)$
Using them, the method can be written as

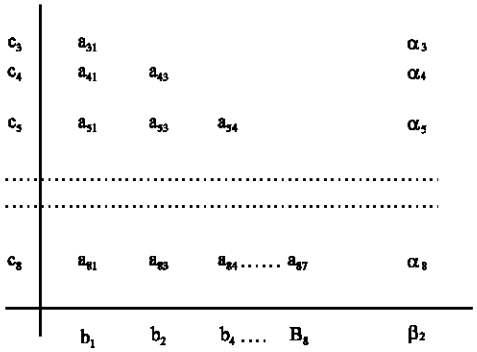
$$x_{n+1} = x_n + h\sum_{i=1}^{s} b_i f_i$$

Many RK-eight stage sixth order formulas that uses are known (Butcher, 1987; Tanaka *et al*., 1993) and their properties are precisely reported (Butcher, 1987).

An eight stage limiting formula that uses the values of the second derivatives at the point (t_n, x_n) is of the form,

$$\begin{aligned}
f_1 &= f(t_n, x_n),\\
F_2 &= D(f(t_n, x_n)).u(f_1),\\
x_3 &= x_n + h(a_{31} f_1 + h\alpha_3 F_2),\\
f_3 &= f(t_n + c_3 h, x_3),\\
x_i &= x_n + h(a_{i1} f_1 + \sum_{j=3}^{i-1} a_{ij} f_j + h\alpha_i F_2),\\
f_i &= f(t_n + c_i h, x_i) \qquad (i = 4,5,\ldots,8)\\
x_{n+1} &= x_n + h(b_1 f_1 + \sum_{i=3}^{8} b_i f_i + h\beta_2 F_2
\end{aligned} \tag{14}$$

Where, $D(f(t_n,x_n))$ and $u(f_1)$ denote the Jacobian matrix of f at the point (t_n,x_n) and the vector $(1, f_1^1, f_1^2,.....,f_1^n)^T$, respectively (the superscripts denote the component numbers). The parameters of this limiting formula can be written in the following array analogous to Butcher array.



Order Conditions

We restrict ourselves to the case that

$$C_8 = 1, \qquad b_3 = 0$$

and the following simplifying assumptions hold:

$$\alpha_3 = \frac{c_3^2}{2} \tag{15}$$

$$\sum_{j=3}^{i-1} a_{ij}c_j + \alpha_i = \frac{c_i^2}{2} \qquad (i=4,5,...8) \tag{16}$$

$$\sum_{j=3}^{i-1} a_{ij}c_j^2 = \frac{c_i^3}{3} \qquad (i=4,5,...8) \tag{17}$$

Comparing the Taylor series expansion for Eq. 15 with that of the true value $x(t_n+h)$ and matching the coefficients of each elementary differential, after tedious computations, one may get the following equations of conditions for seventh order accuracy:

$$a_{31} = c_3, \qquad a_{i1} + \sum_{j=3}^{i-1} a_{ij} = c_i \qquad (i=4,5,...8) \tag{18}$$

$$\sum_{i=j+1}^{8} b_i a_{ij} = b_j(1-c_j) \qquad (i=4,5,...7) \tag{19}$$

$$\sum_{i=4}^{8} b_i a_{i3} = 0 \tag{20}$$

$$\sum_{i=5}^{8} b_i \sum_{j=4}^{i-1} a_{ij}a_{j3} = 0 \tag{21}$$

$$\sum_{i=6}^{8} b_i \sum_{j=5}^{i-1} a_{ij} \sum_{k=4}^{j-1} a_{jk} a_{k3} = 0 \tag{22}$$

$$\sum_{i=5}^{8} b_i \sum_{j=4}^{i-1} a_{ij} c_j a_{j3} = 0 \tag{23}$$

$$b_1 + \sum_{i=4}^{8} b_i = 1 \tag{24}$$

$$\sum_{i=4}^{8} b_i c_i + \beta_2 = \frac{1}{2} \tag{25}$$

$$\sum_{i=4}^{8} b_i c_i^2 = \frac{1}{3} \tag{26}$$

$$\sum_{i=5}^{8} b_i \sum_{j=4}^{i-1} a_{ij} c_j^2 = \frac{1}{12} \tag{27}$$

$$\sum_{i=5}^{8} b_i \sum_{j=4}^{i-1} a_{ij} c_j^3 = \frac{1}{20} \tag{28}$$

$$\sum_{i=6}^{8} b_i \sum_{j=5}^{i-1} a_{ij} \sum_{k=4}^{j-1} a_{jk} c_k^2 = \frac{1}{60} \tag{29}$$

$$\sum_{i=5}^{8} b_i \sum_{j=4}^{i-1} a_{ij} c_j^4 = \frac{1}{30} \tag{30}$$

$$\sum_{i=6}^{8} b_i \sum_{j=5}^{i-1} a_{ij} \sum_{k=4}^{j-1} a_{jk} c_k^3 = \frac{1}{120} \tag{31}$$

$$\sum_{i=7}^{8} b_i \sum_{j=6}^{i-1} a_{ij} \sum_{k=5}^{j-1} a_{jk} \sum_{l=4}^{k-1} a_{kl} c_l^2 = \frac{1}{360} \tag{32}$$

$$\sum_{i=5}^{8} b_i \sum_{j=4}^{i-1} a_{ij} c_j^5 = \frac{1}{42} \tag{33}$$

$$\sum_{i=6}^{8} b_i \sum_{j=5}^{i-1} a_{ij} \sum_{k=4}^{j-1} a_{jk} c_k^4 = \frac{1}{210} \tag{34}$$

$$\sum_{i=7}^{8} b_i \sum_{j=6}^{i-1} a_{ij} \sum_{k=5}^{j-1} a_{jk} \sum_{l=4}^{k-1} a_{jk} c_k^3 = \frac{1}{840} \tag{35}$$

$$\sum_{i=6}^{8} b_i \sum_{j=5}^{i-1} a_{ij} c_j \sum_{k=4}^{j-1} a_{jk} c_k^3 = \frac{1}{168} \tag{36}$$

Solving Eq. (19-36), the values of the parameters are obtained. For more detail, refer the book written by Mitsui and Shinohara (1995). The RK-Eight stage Seventh order limiting algorithm is an explicit method which is given as follows (Mitsui and Shinohara, 1995).

$$
\begin{aligned}
k_1^{ij} &= \tau f'(x_{ij}(n\tau)) \\
k_2^{ij} &= \tau f'(x_{ij}(n\tau)) + \frac{7}{20}k_1^{ij} \\
k_3^{ij} &= \tau f'(x_{ij}(n\tau)) + (\frac{22979}{100842})k_1^{ij}) + (\frac{33275}{201684})k_2^{ij} \\
k_4^{ij} &= \tau f'(x_{ij}(n\tau)) + \frac{25760306}{57421875}k_1^{ij} - \frac{11585024}{2296875}k_2^{ij} - \frac{2139752}{390625}k_3^{ij} \\
k_5^{ij} &= \tau f'(x_{ij}(n\tau)) + \frac{119094452}{855999375}k_1^{ij} - \frac{22528}{25725}k_2^{ij} + \frac{37929472}{25788125}k_3^{ij} + \frac{19000}{288827}k_4^{ij} \\
k_6^{ij} &= \tau f'(x_{ij}(n\tau)) - \frac{34346067405574}{580779387890625}k_1^{ij} - \frac{6059333632}{91822828125}k_2^{ij} \\
&\quad - \frac{306126104994304}{5780137717578125}k_3^{ij} - \frac{49440496}{3893987515}k_4^{ij}) + \frac{1168031718}{76431573125}k_5^{ij} \\
k_7^{ij} &= \tau f'(x_{ij}(n\tau)) - \frac{1484329913137}{501007140480}k_1^{ij} - \frac{235840}{196049}k_2^{ij} + \frac{235609507990864}{260700167892285}k_3^{ij} \\
&\quad + \frac{1702700078125}{3787716228384}k_4^{ij} + \frac{16765288525}{1576329984}k_5^{ij} + \frac{935180524328125}{256363606146816}k_6^{ij}
\end{aligned}
\tag{37}
$$

Therefore, the final integration is a weighted sum of the seven calculated derivatives which is given below.

$$
\begin{aligned}
x_{ij}((n+1)\tau) = x_{ij}(n\tau) + [&\frac{8835}{108416}k_1^{ij} + \frac{24748509184}{60419933937}k_3^{ij} \\
&- \frac{6640625}{86062944}k_4^{ij} + \frac{951125}{2363904}k_5^{ij} + \frac{57826519140625}{210293765402112}k_6^{ij})
\end{aligned}
\tag{38}
$$

Where, f(.) is computed according to (1).

SIMULATION RESULTS AND COMPARISONS

The remarkable features of the raster CNN simulator (Chua and Yang, 1988a,b) are included in the time multiplexing simulator, namely the choice of three integration methods. In time-multiplexing simulation involving the timesaving scheme, the number of all-black/all-white blocks are encountered during simulation. The number of blocks depends on the image itself and the block size chosen by the user. Figure 4 shows the benefit of this time-saving scheme and it is noticed that it is not necessary to simulate all white block more than once.

All the simulated outputs are performed using a high power workstation and the simulation time used for comparisons is the actual CPU time used. The input image format is the X windows bitmap format (xbm), which is commonly available and easily convertible from popular image formats like GIF or JPEG. Figures 3-5 show the results of the time-multiplexing simulator obtained from a complex image of (256*256) pixels and an averaging template is used for simulation comparisons. By raster CNN simulator discussed by Ponalagusamy and Senthilkumar (2007a), the simulation took 189.56 sec. Next, with the regular time-multiplexing simulator (with overlapping and input belt) the simulation

(a) (b) (c)

Fig. 3: (a) Original lena image, (b) After averaging template and (c) After averaging and edge detection templates by employing RK-fifth order algorithm

(a) (b) (c)

Fig. 4: (a) Original lena image, (b) After averaging template and (c) After averaging and edge detection templates by adapting RK-sixth order algorithm

(a) (b) (c)

Fig. 5: (a) Original lena image, (b) After averaging template and (c) After averaging and edge detection templates by adapting RK-eight stage seven order algorithm

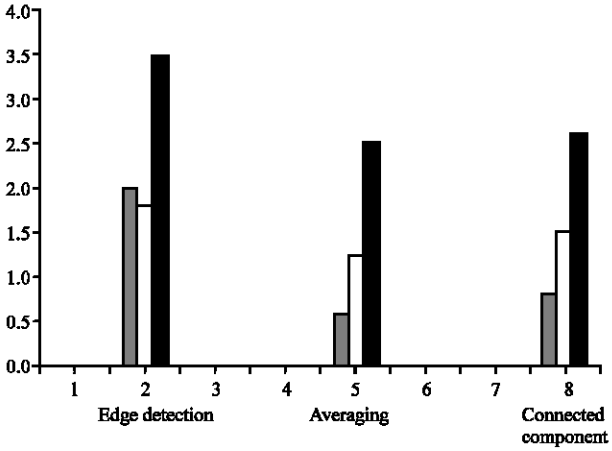


Fig. 6: Maximum step size (Δt) yields the convergence for three different templates

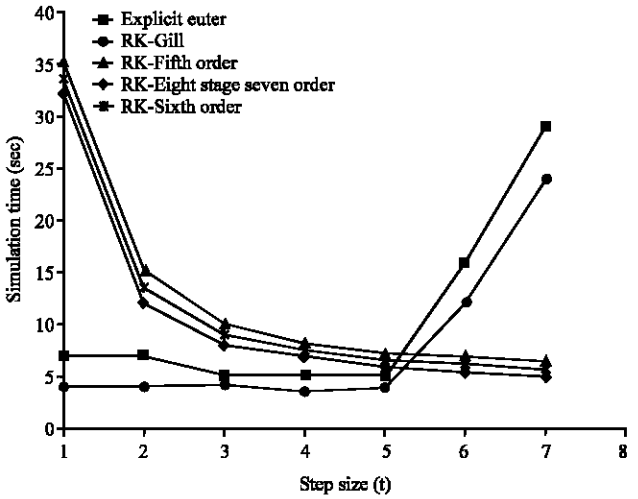


Fig. 7: Comparison of five numerical integration techniques using the averaging template

took 330.42 sec. Finally, the time-multiplexing with the time-saving scheme performed the same simulation in 190.35 sec, almost a 42.42% improvement from the regular time-multiplexing. Similarly by RK-eight order seven stage raster CNN simulator discussed by Ponalagusamy and Senthilkumar (2007b) the simulation took 176.34 sec. Next, with the regular time-multiplexing simulator (with overlapping and input belt) the simulation took 316.27 sec. Finally, the time-multiplexing with the time-saving scheme performed the same simulation in 175.93 sec, almost a 44.37% improvement from the regular time-multiplexing. The size of two dimensional window of 10×10, with two column overlapping is used. It may be noted that this algorithm maintains all the edges of the original one. Using RK-Eight stage seven order, RK-Fifth order and RK-Sixth order algorithms the results of the time-multiplexing simulator obtained from a complex image of 256*256 pixels are depicted, respectively in Fig. 3-5. For the present example an averaging template followed by an Edge Detection template were applied to the original image to yield the images displayed in Fig. 3a, b and c, respectively. The same procedure has been adapted for getting the results shown in Fig. 4a, b and c

and Fig. 5a, b and c. It is observed from Fig. 3-5 that the edges obtained by the RK-Eight stage Seven order algorithm is better than that obtained by the RK-Sixth order and RK-Fifth order algorithms. As speed is one of the major concerns in the simulation, determining 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. 6 still serve as good references. These results were obtained by trial and error over more than 100 simulations on a lena image. Figure 7 shows that the importance of selecting an appropriate time step size (Δt). If the step size is chosen is too small, it might take many iterations, hence longer time, to achieve convergence. But, 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 in the case of the Euler algorithm. The results of Fig. 7 were obtained by simulating a small image of size (256*256) pixels using Averaging template on a Lena image. For a larger step size (Δt), the RK-Eight stage seven order algorithm takes lesser simulation time in comparison with other numerical integration algorithms namely RK-Sixth order, RK-Fifth order, RK-Gill and Explicit Euler.

CONCLUSIONS

The time multiplexing simulator presented here process the image block by block, simulating CNN the way the hardware specifications, if the templates number of CNN processors of the hardware is smaller than the input image, which usually it is the case with practical size images. With the overlapping and external belt of inputs, the neighboring interaction between CNN blocks is ensured, but at the same time, computation costs also increased. On the other hand, with the added feature of processing the all-black and all-white blocks just once for the entire simulation, the simulation time is brought down to the levels of raster simulation, if not better, in some cases, depending on the input image and the size of block and overlap chosen. For a given step size, the convergence time of this algorithm is log linear for all larger size images and this is an additional attractive feature to deal with high resolution/ large images. It is pertinent to pin-point out here that the RK-eight stage seven order algorithm guarantees the accuracy of the detected edges and greatly reduces the impact of random noise on the detection results in comparison with the RK-fifth order algorithm and RK-sixth order algorithms. It is of interest to mention that RK-eight stage seventh order algorithm preserves the edges of the output images, proved to be feasible and effective by theoretic analysis and simulation.

ACKNOWLEDGMENTS

This research was partially supported as a part of Technical Quality Improvement Programme [TEQIP], sponsored by Government of India, National Institute of Technology, Tiruchirappalli-620015, Tamilnadu, India.

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.

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

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

Gonzalez, R.C., R.E. Woods and S.L. Eddin, 2005. Digital Image Processing using MATLAB. Pearson Education Asia.

Lai, K.K. and P.H.W. Leong, 1995. An Area Efficient Implementation of a Cellular Neural Network. IEEE, pp: 51-54.

Lee, C.C. and J.P. de Gyvez, 1994. Single-Layer CNN Simulator. Int. Symp. Circuits and Systems, 6: 217-220.

Mitsui, T. and Y. Shinohara, 1995. Numerical Analysis of Ordinary Differential Equations and its Applications. Published by World Scientific Publishers. ISBN 981-02-2229-71-15.

Nossek, J.A., G. Seiler, T. Roska and L.O. Chua, 1992. Cellular Neural Networks: Theory and Circuit Design. Int. J. Circuit Theory Applications, 20: 533-553.

Oliveira, 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.

Ono, H., 1989. J. Inform. Processing, 12: 251-160.

Ponalagusamy, R. and S. Senthilkumar, 2007a. Investigation on Raster CNN Simulation by Numerical Integration Algorithm. J. Combinatorial Mathematics and Combinatorial Computing (Accepted for Publication).

Ponalagusamy, R. and S. Senthilkumar, 2007b. Multilayer Raster CNN Simulation Using Limiting Formulas of RK (7, 8), EJUM. (Communicated and Under Review).

Roska *et al*., 1994. CNNM Users Guide. Version 5.3x, Budapest.

Tanaka, M., K. Kasuga, S. Yamashita and H. Yamazaki, 1993. Trans. Inform. Proce. Soc. Japan, 34: 62-74 (Japanese).