International Journal of
Science and Engineering Investigations vol. 4, issue 40, May 2015
ISSN: 2251-8843

Approximate Solutions of Partial Differential Equations using Cellular Neural Networks

Mohamed A. Ramadan[1], Talaat S. El-Danaf[2], Mahmoud A. Eissa[3]
[1,2,3]Department of Mathematics, Faculty of Science, Menoufia University, Egypt
([3]mahmoud.eisa@science.menofia.edu.eg)

Abstract- This paper is concerned with an analog computing based on Cellular Neural Network (CNN) systems to develop an approximate solution of Burgers' equation. The Reaction-diffusion CNN (RD-CNN) model is explained, which is an important class of partial differential equations (PDEs). The accuracy of the proposed method is demonstrated by three test problems. The results are presented to show the efficiency of the method and compared with the exact solution to test the accuracy. The numerical results are presented graphically.

Keywords- *Cellular Neural Network, PDEs, Numerical Solution, Nonlinear Burgers' Equation, accuracy.*

I. INTRODUCTION

Consider the initial boundary value problem of Burgers' equation of the form

$$u_t + vuu_x - \varepsilon u_{xx} = 0\,,\ a \le x \le b \tag{1}$$

Which is the one dimensional quasilinear parabolic partial differential equation where ε, v are positive parameters and coefficient of the kinematics viscosity of the fluid and the respectively with initial and boundary conditions:

$$\begin{aligned} &u(x,0) = f(x) \\ &u(a,t) = g_1,\ u(b,t) = g_2 \ \ \forall t > 0, \end{aligned} \tag{2}$$

The study of Burgers' equation is important since it arises in the approximate theory of flow through a shock wave propagating in a viscous fluid and in the modeling of turbulence [1]. The exact solutions of Burgers' equation have been surveyed by Benton and Platzman [2]. In many cases these solutions involve infinite series which may converge very slowly or for small values of the viscosity coefficients. Several studies in the literature have been considered to compute numerical solutions of Burgers' equation [3-7].

There are many researchers have introduced the methods for solving PDEs using some of models of CNN such as, Chedjou1 et al [11] presented solving ordinary differential equations (ODEs) and PDEs using State controlled CNN (SC-CNN). A learning method based on CNN is introduced by Aein and Talebi [12]. Hadad and Piroozmand [13] explained an implementation of the CNN paradigm on very large scale integrated circuit (VLSI) to solve PDEs. Corinto et al [14] considered the discrete CNN paradigm to solve PDEs with applications for image multi-scale analysis. N. Zoltan et al [15] Presented an emulation of the CNN paradigm on digital platforms to solve PDEs.

The idea of fixed-point is introduced which is being exploited to decrease the computing precision and increasing computing speed Fausto Sargeni [16]. They focused on the CNN-based analog computing paradigm to solve PDEs. The paradigm is shown to be flexible in setting boundary conditions and selecting discretization methodologies as well.

The state of the art presents the CNN paradigm as being an attractive alternative solution to conventional numerical computation method [9, 10, 12-17]. It has been intensively show CNN is an analog computing paradigm which performs ultra-fast calculations and provides accurate results [9, 10]. Interestingly, a speed-up of the analog computing process is possible by an implementation on reprogrammable computing [11].

In this paper, we consider the concept of RD-CNN model which an important class of partial differential equations PDEs to develop a numerical method for obtaining approximate solution for (1). We explain the possibility of deriving appropriate RD-CNN model templates to solve (1). Using our approach, this equation is mapped to RD-CNN model array in order to facilitate templates calculation. On the other hand (1) is transformed into ODEs having array structures. This transformation is achieved by applying the method of finite difference method.

The paper is organized as follows: In section 2, the concept of CNN Paradigm and the Reaction-diffusion CNN model are explained. In section 3, an approximate solution is obtained by applying the Reaction-diffusion CNN model using initial and boundary conditions of (1). Finally, three test problems are presented to demonstrate the accuracy and efficiency of the method.

II. THE CNN PARADIGM AND (RD-CNN) MODEL

A. Cellular Neural Network

CNN was first introduced as an acronym for Cellular Neural Network by Chua and Yang [9, 10]. It is an information or signal processing system composed of a large number of simple analog processing elements, called cells which are locally interconnected and perform parallel processing in order to solve a given computational task. The key concept distinguishes a CNN from other neural networks is that, the interconnections among cells are local. This is indeed a great advantage which makes CNN models tailor-made for monolithic implementation in currently available planar technologies [8].

Definition (1): A CNN is any spatial arrangement of locally-coupled cells, where each cell is a dynamical system which has an input, an output and a state evolving according to some prescribed dynamical laws [8].

The resulting network is defined mathematically by four specifications:

1. Cell dynamics,
2. Synaptic law,
3. Boundary conditions,
4. Initial conditions.

A one-dimensional (1-D) and a two-dimensional (2-D) CNN architecture spatial arrangement of locally-coupled of a row of *N* cells and an *N x M* array of cells, respectively are show in Fig. 1(a) and Fig. 1(b). For a continuous time CNN, cells usually consist of time-invariant circuit elements similar as Fig. 2. The cells can be any dynamical system. Although Def. 1 includes discrete-time CNNs. A 3-D CNN can be built up by cascading 2-D CNN layers.

A circuit cell for the Chua-Yang model is explained in Fig. 2. The node voltages $\upsilon_{xi,j}, \upsilon_{yi,j}$ and $\upsilon_{ui,j}$ are called the state, output and input of the cell respectively. C is a linear capacitor, R is a linear resistor, $E_{i,j}$ is an independent voltage source, I_z is an independent current source, $f(\upsilon_{xi,j})$ is a nonlinear voltage controlled voltage source, I_y, I_u and $I^s_{i,j}$ are linear voltage controlled current sources with characteristics $I_y = a_{0,0}\upsilon_{yij}, I_u = b_{0,0}\upsilon_{Uij}$.

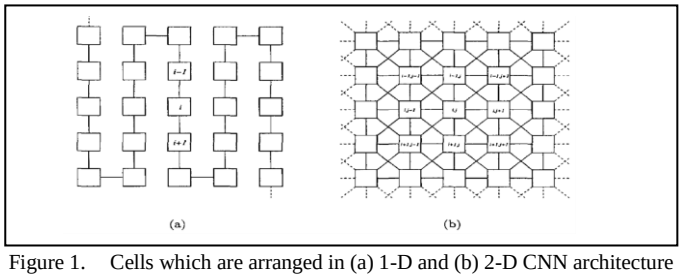

Figure 1. Cells which are arranged in (a) 1-D and (b) 2-D CNN architecture

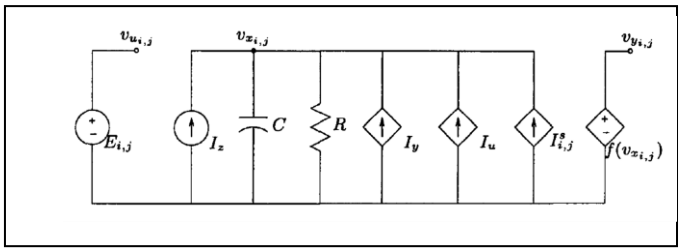

Figure 2. CNN cell defined as a nonlinear first order circuit

There are a lot of models of state equations of CNN which appear as follows:

1) Chua-Yang CNN model
2) SC-CNN model
3) Full-range CNN model
4) RD-CNN model
5) Generalized CNN models
   a. A generalized CNN model: nonlinear and delay CNNs
   b. A generalized CNN based on Chua's circuit

We will explain in details RD-CNN model which is an important class of PDEs

B. RD-CNN model

RD-CNN is an important class of PDEs is reaction diffusion CNN model which consider PDEs as follows:

$$\frac{\partial u(x,y,t)}{\partial t} = h(u(x,y,t)) + \nabla^2 u(x,y,t) \tag{3}$$

where $u = [u_1, u_2, ..., u_m]^T \in \Re^m, h: \Re^m \to \Re^m$ and

$$\nabla^2 u_i = \frac{\partial^2 u_i}{\partial x^2} + \frac{\partial^2 u_i}{\partial y^2} \qquad i = 1,2,...,m \tag{4}$$

In order to find an approximate solution to the PDEs a spatial discretization can be applied. The PDEs are transformed into a system of ODES resulting into the state equations of a CNN with an appropriate synaptic law.

The discretization in space is typically made in equidistant steps h in both $(\Delta x = \Delta y = h)$ on the $N \times M$ grid. So $u(x,y,t)$ is mapped into a CNN layer such that state variable $x_{ij}(t)$ of a CNN cell C_{ij} is associated with $u(ih, jh, t)$ where $i \in \{1,2,...,N\}$ and $j \in \{1,2,...,M\}$ .Based on the Taylor-series expansion of $u(x,y,t)$ the Laplacian operator is approximated by

$$\nabla^2 u \approx \frac{1}{h^2}[U(x+h,y) + U(x-h,y) + U(x,y+h) + U(x,y-h) - 4U(x,y)] \tag{5}$$

Hence a cell of a reaction-diffusion CNN is governed by the following **state equation**

$$\dot{x}_{ij} = h(x_{ij}) + \hat{A}_{0,0} x_{ij} + I_{ij}^{s} \tag{6}$$

where $x_{ij} \in \Re^m$, $I_{ij}^{s} \in \Re^m$ and $\hat{A}_{0,0} = -\frac{4}{h^2} I_{m\times m}$.

The synaptic law of the model is given by

$$I_{ij}^{s} = \sum_{k,l \in S_{i,j}} \hat{A}_{k-i,l-j} x_{k,l} \tag{7}$$

The model has been used as a paradigm for generating auto waves, spiral waves, scroll waves and spatial-temporal chaos.

III. SOLUTION OF NONLINEAR PDES USING RD-CNN MODEL

CNN can be used to solve PDEs. Four main variables (discrete or continuous) are considered when solving PDEs which time, value of the state variable, interaction of parameters and space. The overall approach is based on transform PDEs into ODEs and arranges these ODEs into a form which can be identified with the CNN models and calculates the templates.

We explain the basic steps for solving one dimensional non-linear Burgers' equation with boundary condition in (1) using CNN.

1. Determine the dimension and model of CNN systems to develop approximation to (1). Using 1-D CNN with RD-CNN model with ***state template*** $\hat{A}$ and ***nonlinear feedback template*** $A^n(x_i, x_k)$ and ***the threshold*** z with boundary condition to solving (1).

Hence, a cell of RD-CNN is governed by the following ***state equation***

$$\dot{x}_i = h(x_i) + \hat{A}_{i,j} x_i + \left[\sum_{j \neq i}^{M} \hat{A}_{i,j} x_j + A_{i,j}^{n}(x_i, x_j) \right] + z_i$$

$$i = 1,2,..., N-1,\ j = 1,2,..., M \tag{8}$$

Fixed (Dirichlet) Boundary Condition

$$x_0 = E_1, x_N = E_2 \quad \text{where } E_1, E_2 \in \Re$$

2. The spatial discretization using finite difference method (FDM) is performed in order to transform (1) into sets of ODEs and arrange them into a suitable form of CNN paradigm using (Explicit Scheme of FDM)

$$u_{xx} = \frac{u_{i+1}^{j} - 2u_i^j + u_{i-1}^j}{h^2} + o(h^2),\quad u_x = \frac{u_{i+1}^j - u_{i-1}^j}{2h} + o(h), \tag{9}$$

where $h = (b-a)/N$

Equation (1) can be written in the following form

$$\dot{u}_i = \varepsilon \left[\frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{h^2} \right] - v u_i \left[\frac{u_{i+1}^j - u_{i-1}^j}{2h} \right] \tag{10}$$

$$\dot{u}_i = \frac{\varepsilon}{h^2} \left[u_{i+1}^j - 2u_i^j + u_{i-1}^j \right] - \frac{v}{2h} u_i \left[u_{i+1}^j - u_{i-1}^j \right] \tag{11}$$

Let $l = \frac{\varepsilon}{h^2}$, $r = \frac{v}{2h}$ then

$$\dot{u}_i = l\left[u_{i+1}^j - 2u_i^j + u_{i-1}^j \right] - r u_i \left[u_{i+1}^j - u_{i-1}^j \right] \tag{12}$$

$$\dot{u}_i = l u_{i+1}^j - 2l u_i^j + l u_{i-1}^j - r u_i \left[u_{i+1}^j - u_{i-1}^j \right] \tag{13}$$

The spatial domain is built from a number of grid-points localized by position x_i, the index i being an integer. Therefore (13) clearly shows that the analog computing of PDEs is possible by transforming them into ODEs which are expressed in the form of (13). This form is a set of coupled first order ODEs, the number of equations being fixed by the index i. Equation (13) in the form of i first ODEs which are further identified with RD-CNN model in (8).

3. The number of differential equations defines the maximum index of the CNN processor array.

4. The differential equations are arranged in the similar form as RD-CNN model *state equations (8) which take the form.*

$$\begin{aligned}
\dot{u}_1 &= -u_1^j + (1-2l)u_1^j + l u_2^j + l u_0^j - r\left[u_1 \left[u_2^j - u_0^j\right]\right] \\
\dot{u}_i &= -u_i^j + (1-2l)u_i^j + l u_{i+1}^j + l u_{i-1}^j - r\left[u_i \left[u_{i+1}^j - u_{i-1}^j\right]\right], i = 2,..., N-2 \\
\dot{u}_{N-1} &= -u_{N-1}^j + (1-2l)u_{N-1}^j + l u_N^j + l u_{n-2}^j - r\left[u_{N-1} \left[u_N^j - u_{N-2}^j\right]\right]
\end{aligned} \tag{14}$$

With the boundary condition

$$u_0^j = u(a,t) = g_1 = E_1, u_N^j = u(b,t) = g_2 = E_2$$

5. Compare the coefficients of variables at differential equations (14) with RD-CNN model *state equations (8)* to obtain ***the template values*** for each CNN processor.

$$\hat{A}=\begin{bmatrix} 1-2l & l & 0 & 0 & 0 & \cdots & 0 \\ l & 1-2l & l & 0 & 0 & \cdots & 0 \\ 0 & l & 1-2l & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & l & 1-2l & l \\ 0 & 0 & 0 & 0 & 0 & l & 1-2l \end{bmatrix}, z_i=\begin{bmatrix} lE_1 \\ 0 \\ \vdots \\ 0 \\ lE_2 \end{bmatrix}, \quad (15)$$

$$A^n=\begin{bmatrix} r & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & r & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & r & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & r & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & r \end{bmatrix}$$

We can set the RD-CNN model *state equations (8)* which use to solve (1) in the matrix form

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \vdots \\ \dot{x}_{N-2} \\ \dot{x}_{N-1} \end{bmatrix}=-\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix}+\begin{bmatrix} 1-2l & l & 0 & 0 & 0 & \cdots & 0 \\ l & 1-2l & l & 0 & 0 & \cdots & 0 \\ 0 & l & 1-2l & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & l & 1-2l & l \\ 0 & 0 & 0 & 0 & 0 & l & 1-2l \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix}+\begin{bmatrix} lE_1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ lE_2 \end{bmatrix} \quad (16)$$

$$-\begin{bmatrix} r & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & r & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & r & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & r & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & r \end{bmatrix}\begin{bmatrix} x_1(x_2-E_1) \\ x_2(x_3-x_1) \\ x_3(x_4-x_2) \\ \vdots \\ x_{N-2}(x_{N-1}-x_{N-3}) \\ x_{N-1}(E_2-x_{N-2}) \end{bmatrix}$$

We can set (16) in the system form

$$\begin{aligned} \dot{x}_1 &= -x_1+\hat{A}_{1,1}x_1+\hat{A}_{1,2}x_2+lE_1-A^n_{1,1}[x_1(x_2-E_1)] \\ \dot{x}_i &= -x_i+\hat{A}_{i,i-1}x_{i-1}+\hat{A}_{i,i}x_i+\hat{A}_{i,i+1}x_{i+1}-A^n_{i,i}[x_i(x_{i+1}-x_{i-1})]\ i=2,...,N-2 \\ \dot{x}_{N-1} &= -x_{N-1}+\hat{A}_{N-1,N-2}x_{N-2}+\hat{A}_{N-1,N-1}x_{N-1}+lE_2-A^n_{N-1,N-1}[x_{N-1}(E_2-x_{N-2})] \end{aligned} \quad (17)$$

1. Use the template values to implement the RD-CNN processor in Matlab using (programming /Simulink).

We must determine ***the solver function*** in Matlab which appropriate with the CNN system and give the high accuracy to the approximate solution of (1).

There is a lot of ***solver function*** in Matlab which used to solve the initial and boundary ordinary differential equation problems but we used to solve (1) ***ode45*** as a basic solver.

IV. Application and Numerical Result

We now obtain numerical solutions of Burgers' equation for the 1st and 2nd test problems. The versatility and the accuracy of the proposed method is measured using the L_2 and L_∞ error norms for the 1st and 2nd test problems to make comparison with exact solution and some published methods [18].

$$L_\infty=\left|u-u_N\right|_\infty=\max_j\left|u_j-(u_N)^n_j\right|,\ j=1,2,...,N-1 \quad (18)$$

$$L_2=\left|u-u_N\right|^2=h\sum_{j=0}^{N}\left|(u_j-(u_N)^n_j)^2\right|, \quad (19)$$

Where u_j is the exact solution and $(u_N)^n_j$ is the approximate solution at step j.

A. Test Problem (1)

Shock-like solution of the Burgers equation (1) is studied through the analytical solution [19]

$$U(x,t)=\frac{x/t}{1+\sqrt{t/t_0}\exp(x^2/4vt)},\quad t\geq 1, 0\leq x\leq 1, \quad (20)$$

where $t_0=\exp(\frac{1}{8v})$. This solution represents the propagation of the shocks which becomes slightly smoother as time progresses. Initial condition which is taken when $t=1$ in (20) is used. We have run the method for boundary conditions and choose the result with selection of $U(a,t)=0$ and $U(b,t)=0$. With Parameters $h=0.005$, $\Delta t=0.01, v=0.005$ and computation is done up to time $t=3.1$ are selected to comparison the result with [18]. Use various values of v and computation is done up to time $t=7$ over the problem domain [0, 1].

The results of the algorithms together with exact solutions are documented in Table I, II. RD-CNN model produces almost the same results in terms of the L_2 and L_∞ error norms. The results of the present simulation exhibited the same results with QBGM , CBGM [18] and quadratic Galerkin method [20] as compared in Table I. We observe that, RD-CNN model produced a little higher error than alternative approach and good behavior of the result up to $t=7$. The numerical solution is visualized at various times in Fig.3, 4. Error between the analytical and numerical solutions is graphed in Fig. 5.

TABLE I. Comparison of result at different times for $v=0.005, h=0.005, \Delta t=0.01$

t	L_2			
	QBGM	***CBGM***	***[29]***	***RD – CNN***
1.7	3.51E-04	3.51E-04	8.57E-04	1.49E-08
2.4	2.45E-04	2.44E-04	4.23E-04	7.50E-09
3.1	6.33E-04	6.33E-04	2.35E-04	3.28E-07
3.5	-	-	-	8.76E-06
4	-	-	-	1.18E-04
4.5	-	-	-	3.61E-04
5	-	-	-	5.10E-04
6	-	-	-	5.41E-04
7	-	-	-	4.91E-04

t	L_∞			
	QBGM	***CBGM***	***[29]***	***RD – CNN***
1.7	1.21E-03	1.21E-03	2.58E-03	4.72E-04
2.4	8.02E-04	8.02E-04	6.88E-04	3.02E-04
3.1	4.79E-03	4.79E-03	1.24E-03	4.11E-03
3.5	-	-	-	2.07E-02
4	-	-	-	7.50E-02
4.5	-	-	-	1.31E-01
5	-	-	-	1.53E-01
6	-	-	-	1.47E-01
7	-	-	-	1.31E-01

TABLE II. COMPARISON OF RESULT AT DIFFERENT TIMES FOR $\nu, h = 0.005, \Delta t = 0.01$

t	$\nu = 0.025$		$\nu = 0.05$		$\nu = 0.1$	
	L_2	L_∞	L_2	L_∞	L_2	L_∞
1.7	9.35E-06	1.42E-02	4.48E-04	7.12E-02	3.21E-03	1.44E-01
2.4	1.24E-04	4.38E-02	1.01E-03	8.99E-02	3.50E-03	1.24E-01
3.1	3.72E-04	6.78E-02	1.29E-03	9.04E-02	3.11E-03	1.03E-01
3.5	5.19E-04	7.61E-02	1.36E-03	8.74E-02	2.80E-03	9.34E-02
4	6.70E-04	8.18E-02	1.38E-03	8.27E-02	2.40E-03	8.30E-02
4.5	8.47E-04	8.36E-02	1.30E-03	7.25E-02	1.71E-03	6.71E-02
5	9.04E-04	7.97E-02	1.16E-03	6.34E-02	1.21E-03	5.56E-02
6	8.98E-04	7.42E-02	9.99E-04	5.58E-02	8.72E-04	4.71E-02
7	9.35E-06	1.42E-02	4.48E-04	7.12E-02	3.21E-03	1.44E-01

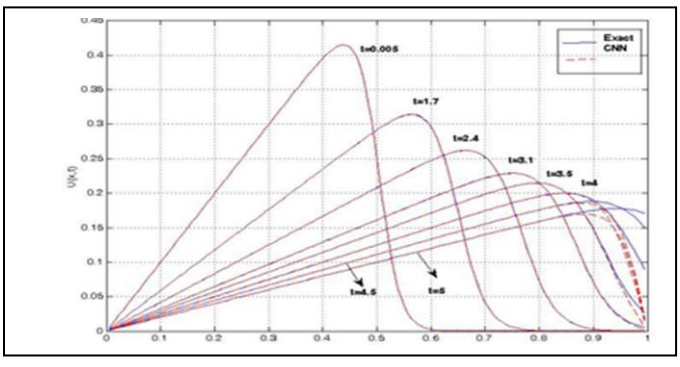

Figure 3. Comparison of result at different times for $\nu = 0.005, h = 0.005, \Delta t = 0.01$

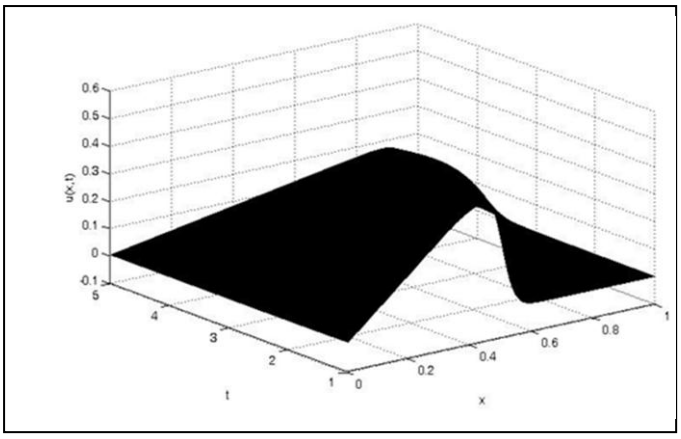

Figure 4. The numerical solution with. $\nu = 0.005, h = 0.005, \Delta t = 0.01$

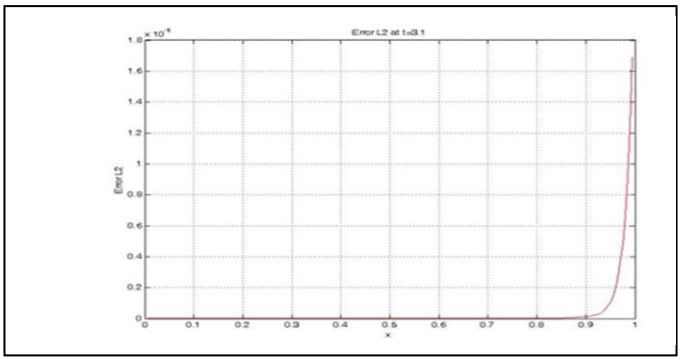

Figure 5. Error at time 3.1 with $\nu = 0.005, h = 0.005, \Delta t = 0.01$.

B. Test Problem (2)

For our second test example consider the particular solution of the Burgers equation (1) [21, 22]

$$U(x,t) = \frac{\alpha + \mu + (\mu - \alpha)\exp(\eta)}{1 + \exp(\eta)}, 0 \le x \le 1, t \ge 0 \qquad (21)$$

where

$$\eta = \frac{\alpha(x - \mu t - \gamma)}{\nu},$$

and α, μ and γ are constants and this is a wave which moves to the right with speed μ. The constants are chosen to have values $\alpha = 0.4$, $\mu = 0.6$, $\gamma = 0.125$ with boundary conditions

$$U(a,t) = 1 \text{ and } U(b,t) = 0.2, \qquad t \ge 0$$

and initial condition is obtained when $t = 0$ is taken in (21). We take space step $h = 1/36$ and time step $\Delta t = 0.001$ and velocity constant $\nu = 0.01$ and computation is done up to time $t = 0.5$ are selected to comparison the result with [18]. Use

various values of v and computation is done up to time $t = 3$ over the problem domain [0, 1].

The results of the algorithms together with exact solutions are documented in Tables III, IV. RD-CNN model produces almost the same results in terms of the L_2 and L_∞ error norms. Numerical solution of RD-CNN model with corresponding exact solutions are documented at some values over the domain of the problem at time $t = 0.5$ to comparison with [18] to time $t = 3$ in Table III. The L_2 and L_∞ error norms for this experiment are recorded almost the same at the time $t = 0.5$. It is seen from Table 3.3 that good agreement with both numerical values and exact values is evident. Numerical result at various time are graphed in Fig.6, 7. Errors between the analytical and numerical solutions are graphed in Fig.8.

TABLE III. COMPARISON OF RESULT AT DIFFERENT TIME FOR $v = 0.01, h = 1/36, \Delta t = 0.01$

t	L_2		
	QBGM	***CBGM***	***RD – CNN***
0.5	1.93E-03	1.73E-03	1.22E-04
1	-	-	1.32E-04
1.5	-	-	1.65E-04
2	-	-	1.68E-03
2.5	-	-	1.68E-03
3	-	-	1.68E-03

t	L_∞		
	QBGM	***CBGM***	***RD – CNN***
0.5		6.35E-03	5.49E-03	3.82E-02
1		-	-	4.22E-02
1.5		-	-	6.89E-02
2		-	-	2.44E-01
2.5		-	-	2.44E-01
3		-	-	2.43E-01

TABLE IV. COMPARISON OF RESULT AT DIFFERENT TIME FOR $v, h = 1/36, \Delta t = 0.01$

t	$v = 0.025$		$v = 0.05$		$v = 0.1$	
	L_2	L_∞	L_2	L_∞	L_2	L_∞
0.5	2.38E-04	3.65E-02	3.27E-03	1.03E-01	1.40E-02	1.77E-01
1	2.23E-04	3.45E-02	2.73E-03	9.45E-02	1.03E-02	1.46E-01
1.5	1.26E-03	2.03E-01	3.39E-03	2.85E-01	7.89E-03	3.31E-01
2	3.14E-03	3.19E-01	1.16E-02	5.07E-01	1.91E-02	5.18E-01
3	3.17E-03	3.20E-01	1.37E-02	5.47E-01	3.35E-02	6.53E-01

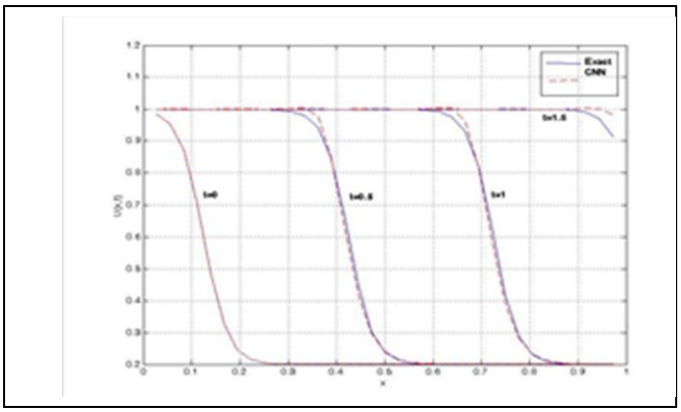

Figure 6. Comparison of result at different time for $v = 0.01, h = 1/36, \Delta t = 0.01$

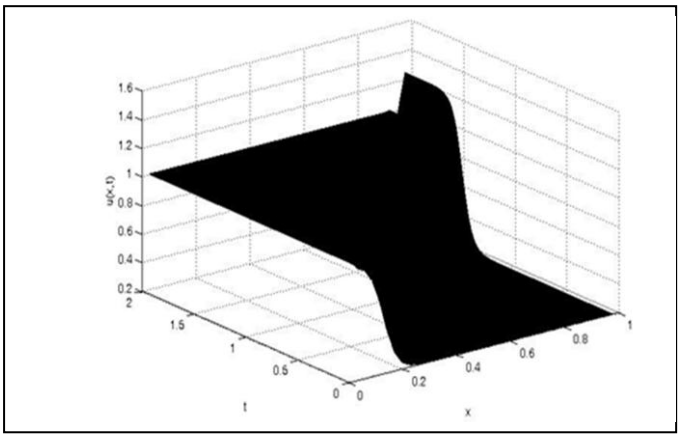

Figure 7. The numerical solution with $v = 0.01, h = 1/36, \Delta t = 0.01$

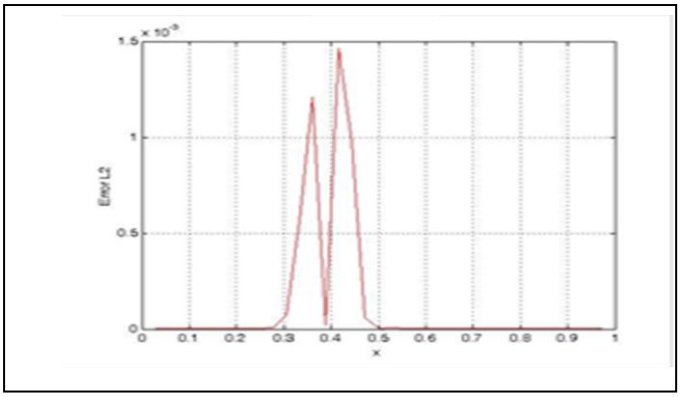

Figure 8. Error at time 0.5 with. $v = 0.01, h = 1/36, \Delta t = 0.01$.

C. Test Problem (3)

The Burgers' equation (1) has the analytic solution of the form [3]

$$u(x,t)=\frac{v}{1+vt}(x+\tan[\frac{x}{2+2vt}]), \qquad t\geq 0 \tag{22}$$

and both boundary conditions with $a=0.5, b=1.5$ are given as:

$$u(0.5,t)=\frac{v}{1+vt}(0.5+\tan[\frac{1}{4+4vt}]),$$
$$u(1.5,t)=\frac{v}{1+vt}(1.5+\tan[\frac{3}{4+4vt}]), \tag{23}$$

and initial condition is obtained when $t=0$ is taken in (22). We take space step $h=1/40$, time step $\Delta t=0.01$, velocity constant $v=1/500$ and computation is done up to time $t=2.1$ are selected to comparison the result with [3]. Use various values of v and computation is done up to time $t=10$ over the problem domain [0.5, 1.5]. We observe that, the RD-CNN model produces almost the same results in terms of the absolute error L_2 and L_∞ error norms which is compute as the form.

Absolute Error = $e = |u_j - (U_N)_j^n|$, (24)

$$L_2=\sqrt{\sum_{i=1}^{N-1}(u_j-(u_N)_j^n)^2}, \tag{25}$$

$$L_\infty=\max_j|u_j-(u_N)_j^n| \tag{26}$$

where u_j is the exact solution and $(u_N)_j^n$ is the approximate solution at step j.

Numerical solution of the reaction-diffusion CNN model with corresponding exact solutions are documented at some values over the domain of the problem at time $t=2.1$ to comparison with [16] to time $t=10$ in Table V. The absolute error, L_2 and L_∞ error norms for this experiment are recorded almost the same at the time $t=2.1$. It is seen from Table VI. Numerical result at various time are graphed in Fig.9. Error between the analytical and numerical solutions is graphed in Fig.10.

TABLE V. COMPARISON OF RESULT AT DIFFERENT TIME FOR $v=1/500, h=1/40, \Delta t=0.01$

t	Absolute Error *t=2.1*	
	ADM [22]	*RD-CNN*
0.6	5.16E-06	8.59E-10
0.8	1.91E-05	2.76E-10
1.1	3.49E-05	5.28E-10
1.4	2.09E-05	4.20E-09
L_2	-	7.97E-09
L_∞	-	4.20E-09

RD – CNN					
t	L_2	L_∞	**t**	L_2	L_∞
2.1	7.97E-09	4.20E-09	6	2.38E-06	7.79E-07
3	7.73E-07	3.72E-07	7	1.14E-06	3.47E-07
4	1.69E-06	6.75E-07	8	5.51E-06	1.63E-06
4.5	1.94E-06	6.94E-07	9	4.46E-06	1.25E-06
5	1.02E-05	3.61E-06	10	4.81E-07	1.32E-07

TABLE VI. COMPARISON OF RESULT AT DIFFERENT TIME FOR $v=1/500, h=1/40, \Delta t=0.01$

t	$v=0.025$		$v=0.05$		$v=0.1$	
	L_2	L_∞	L_2	L_∞	L_2	L_∞
2.1	2.27E-06	7.51E-07	4.61E-05	1.21E-05	4.64E-04	1.04E-04
3	1.75E-05	5.15E-06	3.64E-05	8.66E-06	5.36E-05	1.21E-05
4	1.11E-05	2.94E-06	9.83E-07	2.46E-07	8.61E-05	1.93E-05
5	4.63E-06	1.15E-06	1.18E-05	2.70E-06	4.20E-05	9.43E-06
6	5.69E-06	1.35E-06	2.24E-07	6.99E-08	1.57E-04	3.49E-05
7	4.47E-06	1.04E-06	4.35E-06	1.00E-06	3.77E-05	8.44E-06
8	1.91E-06	4.48E-07	6.81E-05	1.53E-05	9.01E-05	2.01E-05
9	3.85E-06	8.86E-07	2.13E-05	4.78E-06	1.35E-04	3.00E-05
10	1.75E-06	4.08E-07	2.86E-06	6.65E-07	3.53E-05	7.87E-06

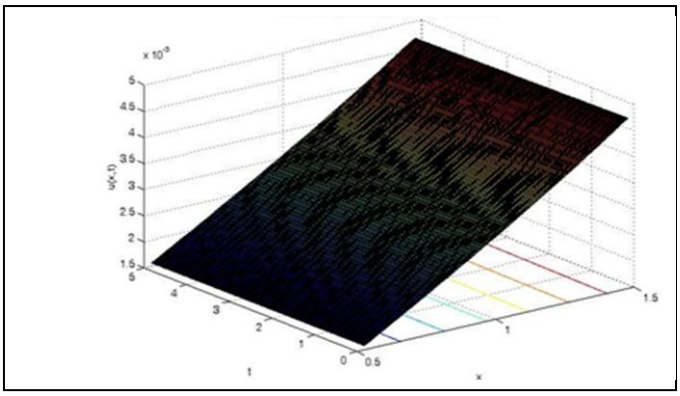

Figure 9. The numerical solution with $v=1/500, h=1/40, \Delta t=0.01$.

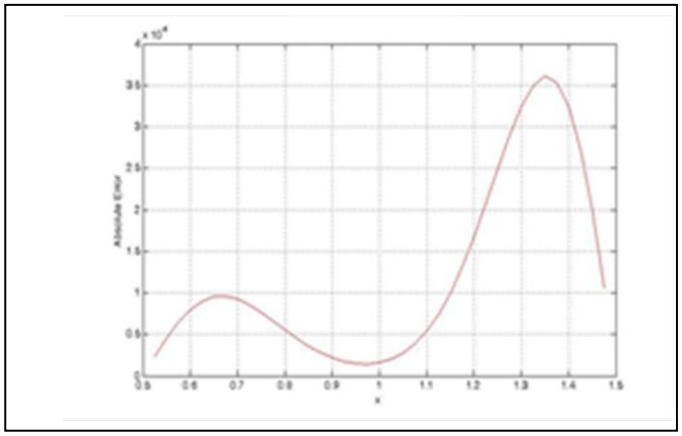

Figure 10. Error at time 2.1 with $\nu = 1/500, h = 1/40, \Delta t = 0.01$

ACKNOWLEDGMENT

I thank my professors who taught me and helped me to take out of this research; I also thank all of the supports scientific.

REFERENCES

[1] Rubin SG and Graves RA. "Viscous flow solutions with a cubic spline approximation. Computer and Fluids", pergaman, pp.3,1-36, 1975.

[2] Benton E and Platzman GW. "A table of solutions of the one – dimensional Burger's equations", Quart Apl. Math., Vol. 30, pp. 195-212, 1972.

[3] M. A. Ramadan and T. S.ElDanaf, "On the Analytical and Numerical Solutions of the One-Dimensional Nonlinear Burgers' Equation", The Open Applied Mathematics Journal, Vol. 1, pp. 1-8, 2007.

[4] M. A. Ramadan and T. S.ElDanaf, "Application of the Non-Polynomial Spline Approach to the Solution of the Burgers' Equation", The Open Applied Mathematics Journal, Vol. 1, pp. 15-20, 2007.

[5] I. Dag, B. Saka and A. Boz, "B-spline Galerkin methods for numerical solutions of the Burgers' equation", Applied Mathematics and Computation, Vol. 166, pp. 506–522, 2005.

[6] M. Javidi, "A numerical solution of Burger's equation based on modified extended BDF scheme", International Mathematical Forum, No. 32, pp. 1565 - 1570, 2006.

[7] K. R. Desai and V. H. Pradhan "Solution of Burger's equation and coupled Burger's equations by Homotopy perturbation method", International Journal of Engineering Research and Applications(IJERA), Vol. 2, No. 3, pp. 2033-2040, 2012.

[8] R.Wilmans, "A Cellular Neural Network The design of a Full-Range Cellular Neural Network and a method to find the Basin of Attraction in CNN's", Eindhoven: Eindhoven University of Technology, Faculty of Electrical Engineering, Systems for Electronic Signal processing,pp 8-19, 1997.

[9] V. Sai, "A Cellular Neural Network Based Analog Computing Approach for Ultra-fast Adaptive Scheduling", Thesis of Master degree,Alpen-Adria University Klagemfurt,2009

[10] Brown and Ray, "Generalizations of the Chua Equations", IEEE Transactions on Circuits and Systems,Fundamental Theory and Applications,Vol.40,No.11, pp. 878-884, November 1993.

[11] J. C. Chedjou1, K. Kyamakya, M.A. Latif, U.A.Khan, I. Moussa and Trong Tuan. "Solving Stiff Ordinary Differential Equations and Partial Differential Equations Using Analog Computing Based on Cellular Neural Networks", ISAST Transactions on Computers and Intelligent Systems, Vol. 1, No. 2, pp. 38-46, 2009.

[12] M. J. Aein and H. A. Talebi, "Introducing a training methodology for Cellular Neural Networks solving partial differential equations", International joint conference on neural networks,ISBN:978-1-4244-3548-7, 2009.

[13] K. Hadad and A. Piroozmand, "Application of Cellular Neural Network(CNN) method to the Nuclear Reactor Dynamic Equations", Elsevier, Annals of nuclear energy Vol.34, pp. 406-416, 2007.

[14] F. Corinto, M. Biey and M. Gilli, "Non-linear coupled CNN models for Multiscale Image Analysis", International journal of circuit theory and applications(special issue on CNN technology), vol. 34, issue 1, pp. 77-88, 2006.

[15] N. Zoltan, V. Zsolt and S. Peter, "Emulated digital CNNUM solution of partial differential equations", International journal of circuit theory and applications, ISSN: 0098-9886, Vol. 34, pp. 445- 470, 2006.

[16] F. Sargeni and V. Bonaiuto, "Programmable CNN Analogue chip for RD-PDE multi method simulations", Analog integrated circuits and signal processing, ISSN: 0925-1030, Vol. 44, Issue 3, pp. 283-292, September 2005.

[17] M. Gilli, T. Roska, L. O. Chua, and P.P. Civalleri, "On the relationship between CNNs and PDEs", Proceedings of the 7th IEEE workshop on Cellular Neural Networks and their applications, IEEE circuits and systems society, 2002.

[18] I. Dag, B. Saka and A. Boz, "B-spline Galerkin methods for numerical solutions of the Burgers' equation", Applied Mathematics and Computation, Vol. 166, pp. 506–522, 2005.

[19] H. Nguyen and J. Reynen, "A space time finite element approach to Burgers equation, in: E.Hinton et al. (Eds.)", Numerical Methods for Non-linear Problems, Vol. 3, pp. 718–728, 1987.

[20] A.H.A. Ali, L.R.T. Gardner and G.A. Gardner, "A Galerkin approach to the solution of Burgersequation", U.C.N.W Maths Pre-print, 90.04, 1990.

[21] I. Christie, D.F. Griffiths, A.R. Mitchell and J.M. Sanz-Serna, "Production approximation for non-linear problems in the finite element method",
J. Numer. Anal. IMA 1, pp. 253–266, 1981.

[22] B.M. Herbst, S.W. Schoombie and A.R. Mitchell, "A moving Petrov–Galerkin method for transport equations", Int. J. Numer. Meth. Eng. Vol.18, pp. 1321–1336, 1982.