# Computational Aspects of Modeling Excitable Media Using Cellular Automata

William Emmanuel S. Yu
Ateneo High Performance Computing Group
Physics Department
Ateneo de Manila University
Quezon City, Philippines
william.s.yu@ieee.org

Rafael P. Saldaña, Ph.D.
Ateneo High Performance Computing Group
Mathematics Department
Ateneo de Manila University
Quezon City, Philippines
raf@mathsci.math.admu.edu.ph

## ABSTRACT

The study of excitable media is important in understanding the behavior of complex systems including mold growth, star formation, and cardiac tissue contraction. In this paper we discuss some computational aspects of modeling excitable media using multi-state cellular automata. In particular, we discuss implementation issues, algorithm development and visualization on the AGILA High Performance Computing System using LAM-MPI.

## Keywords

Cellular Automata, Complex Systems, Excitable Media, Parallel Computing, Computational Science

## 1. INTRODUCTION

The problem of excitable media encompasses the study of several complex systems including a number of reactions such as mold growth, star formation and cardiac tissue contraction. All these reactions have one thing in common: they have various spatially distributed patterns such as concentric and spiral wave patterns. Materials that exhibit this behavior are commonly known as excitable media. Some of the wave patterns formed in these reactions are similar to those displayed by a variety of chemical systems such as expanding rings for the chemical waves found in the well-known Belousov-Zhabotinsky Reaction[1].

### 1.1 Wave Propagation and Excitable Media

Excitable media are extended non-equilibrium systems having uniform states that are linearly stable but susceptible to finite perturbations[2]. This class of systems refer to an ensemble of coupled excitable elements, where each one exhibits the dynamics of a stable fixed point when it is perturbed up to a threshold. The closed trajectory in the phase space is translated to a traveling wave in the real time-space.

Wave propagation in spatially extended media has been the subject of many studies in the last years[3]. In particular, wave propagation in excitable chemical media has been studied extensively, because it constitutes a good example for excitation waves in other contexts such as waves of excitation in neural or cardiac tissues.

These systems are composed of a lattice of cells which have three basic states: quiescent, excited and recovering. The response to a perturbation is the defining characteristic of an excitable element. Above a certain threshold, a perturbation will excite a quiescent element that then decays back to quiescence after a characteristic time insensitive to the magnitude of the perturbation. In physical and chemical systems, these waves provide striking examples of spontaneous symmetry breaking and spatio-temportal organization[4].

Excitable media has also provided models for population dynamics, chemical reactions, earthquake dynamics and others[5][6].

### 1.2 Cellular Automata

A common method for modeling excitable media is the use of discrete cellular automaton. A Cellular Automaton (or CA) consists of a discrete system of lattice sites having various initial values. These sites evolve in discrete time steps as each site assumes a new value based on the values of some local neighborhood of sites and a finite number of previous time steps[2]. The field of cellular automata is best known for the Game of Life invented in the 1960's by the mathematician John Horton Conway of the University of Cambridge. Stephen Wolfram, the inventor of *Mathematica*, also conducted several studies on cellular automata[7].

CA systems can be used to model certain reactions. The CA determines the outcome of the reaction by doing a number of iterations and during each iteration the state of a cell in the system changes based on the states of its neighborhood. The reader is referred to [2] for a more detail discussion on cellular automata.

## 2. EXPERIMENTAL BACKGROUND

Chemical Reactions normally go to completion and normally cease. These chemical reactions appear to reverse and oscillate. Thermodynamic principles, however, appear to be violated by not completing, but reverting back and repeating the process. This is the case of a oscillating chemical reaction. The Belousov-Zhabotinsky Reaction is a chemical reaction named after B. P. Belousov who discovered the reaction and A. M. Zhabotinsky who continued Belousov's work[8].

The BZR occurs when a one-electron redox catalyst, an organic substrate that can be easily brominated and oxidized, and a bromate ion in the form of $NaBrO_3$ or $KBrO_3$ are

all dissolved in sulfuric or nitric acid[8]. Table 1 provides a list of chemicals needed for a BZR. These reactants are mixed and poured into a Petri dish. The patterns formed by the BZR resemble the patterns formed by excitable media. With this in mind, it is possible to study the behavior of excitable media by studying the BZR. Figure 1 shows images of the BZR. Other work that attempt to model BZ reactions include a Chemsim[9] and some other differential methods[10].

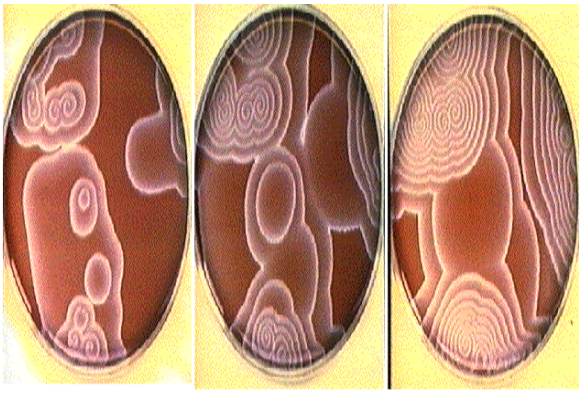

**Figure 1: Laboratory BZ Reactions(source: Ruoff, 1982)**

| | |
|---|---|
| 1.50 cc | $H_2O$ |
| 1.25 cc | $H_2SO_4$(1M) |
| 0.60 cc | Malonic Acid (1M) |
| 1.50 cc | Ferroin Solution (25mM) |
| 2.25 cc | $NaBrO_2$ (1M) |
| 7.10 cc | |

**Table 1: Chemistry of the BZ Reaction(source: Ruoff, 1982)**

# 3. CA MODELS

Several cellular automata models have been developed over the years. Below are three of the more common algorithms being used for modeling excitable media. These algorithms are characterized by their excited-refractory-receptive cycle[2].

## 3.1 Greenburg-Hasting Algorithm

A common form of excitable media is the flow of electrical activity in a biological system, such as the activity occurring in the contraction of cardiac tissues. Neuron excitation is characterized by neuron sites that are at rest or excited. A neuron can also be in any of the intermediary states. This is similar to the excited-refractory-receptive cycle mentioned above. The Greenburg-Hasting algorithm is used to model neuron excitation and recovery in a network of neurons.

### *3.1.1 Rules*

The following are the CA rules for the Greenburg-Hastings Algorithm:

- A rested neuron with at least on excited nearest neighbor neuron becomes excited.
- A rested neuron with no excited nearest neighbor neurons remains rested.
- A neuron that is in recovery state goes to the next recovery state.

### *3.1.2 States and Neighborhood*

The number of states($n = 0 \ldots r - 1$) for this model can vary. $r - 1$ represents the rest state, 0 the excited state, and intermediate numbers as the refractory states. By updating lattice sites, the CA evolves a certain number of times based on the von Neumann neighborhood[2].

## 3.2 Cyclic Space Algorithm

Another common form of excitable media is that of phase transitions. These are usually modeled using an algorithm called Cyclic Space CA. The main difference between the Greenburg-Hasting CA and a Cyclic Space CA is that the recovery of a site proceeds independently of the state of the neighboring sites. The Cyclic Space CA, however, allows the recovery process to proceed for a site only when a neighboring site is slightly more relaxed. In a Greenburg-Hasting CA, the evolution of a intermediary lattice site is inevitable; however, in the Cyclic Space CA the intermediary lattice sites could be reset.

### *3.2.1 Rules*

The following are the CA rules for the Cyclic Space Algorithm:

- A cell is state $n-1$ goes to state 0 if any of its nearest neighboring cells are in the state 0.
- A cell in state $c$, $0 \leq c < n-1$, goes to state $c+1$ if any of its nearest neighboring cells is in state $c+1$.
- A cell in state $c$, $0 \leq c < n-1$, remains in state $c$ if it has no nearest neighboring cells in state $c+1$.

### *3.2.2 States and Neighborhood*

The range $0 \ldots n-1$ represents the values of a lattice site in a periodic boundary condition. By updating lattice sites, the CA evolves a certain number of times based on the von Neumann neighborhood.

## 3.3 Hodgepodge Machine Algorithm

Hodgepodge systems are different from the above two methods because they deal with chemical systems. The hodgepodge machine was first conceived by Alan Turing in his interest to study biological morphogenesis[8]. The most common reaction modeled by this CA is the BZR.

### *3.3.1 Rules*

The following are the CA rules for the Hodgepodge Machine Algorithm:

- If a cell is ill then it becomes healthy.
- If a cell is healthy, it becomes infected and its value changes to an integer value determined by the formula described in Equation 1. The $Min[]$ function is used

to ensure that the value will not exceed $r$. Let $k_1$ and $k_2$ be infection constants, $infected$ be the number of infected neighboring cells and $ill$ be the number of ill neighboring cells.

$$Min[r, Floor[\frac{infected}{k_1}] + \frac{ill}{k_2}] \quad (1)$$

- If a cell is infected, it becomes more infected and its value is determined by Equation 2. Let $g$ be a constant that refers to the speed of infection and $sum$ be the sum of all the values of its neighboring cells.

$$Min[r, g + Floor[\frac{sum}{infected}]] \quad (2)$$

#### 3.3.2 States and Neighborhood

The Hodgepodge System like any typical CA system consists of a square lattice with periodic boundary conditions. The Lattice sites will have values ranging from 0 to $r$. The lattice sites with values of 0 are said to be healthy; lattices sites having values of $r$ are said to be ill. The other lattice sites with values between 0 and $r$ are said to be infected. The CA is updated based on the von Neumann neighborhood.

## 4. COMPUTATIONAL TOOLS

This section discusses the computational tools used in the study. They include the AGILA High Performance Computing System, LAM-MPI, MPE, and the Pseudo-Random Number Generator.

### 4.1 AGILA HPCS

The high performance computing system used for the computation is the AGILA HPCS–a Beowulf-class supercomputer housed in the Advanced Computing Laboratory of the Mathematics Department, Ateneo de Manila University. The system was built by the Ateneo High Performance Computing Group using commodity hardware and free or open source software(e.g. Linux). Message Passing Libraries (e.g. Parallel Virtual Machine (PVM) and Message Passing Interface (MPI)) are utilized to simplify the task of creating applications on the AGILA HPCS[11][12]. The AGILA HPCS has a peak performance of 3.16 GFlops based on the High Performance LINPACK benchmarks[13].

The Ateneo High Performance Computing Group developed the AGILA HPCS due to the shortage of high performance computing systems in the Philippines[19][12][13].

### 4.2 LAM-MPI

To generate parallel codes used for the cluster, a message passing library was utilized. Message Passing Interface(MPI) is a standard library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementors, and users[14]. The message passing library used in the CA computation is LAM-MPI.

Local Area Multicomputer(LAM) is an MPI programming environment and development system for heterogeneous computers on a network[15]. With LAM, a dedicated cluster or an existing network computing infrastructure can act as one parallel computer solving one problem. LAM features extensive debugging support in the application development cycle and peak performance for production applications. LAM features a full implementation of the MPI communication standard.

### 4.3 MPE

For visualization, the Message Passing Environment(MPE) was used. MPE was installed on the cluster and compiled to utilize LAM-MPI as its default message passing interface.

MPE extensions provide a number of useful facilities for MPI programmers. These include several profiling libraries to collect information on MPI programs, including log files for *post-mortem* visualization and real-time animation. Also included are routines to provide simple X windows system graphics to parallel programs. MPE may be used with any implementation of MPI[16]. MPE enables the researchers to call a set of functions and routines in order to perform the visualization tasks that are needed. This visualization is critical since it enables the researchers to perform the necessary analysis of the model. Using MPE is a better alternative than writing custom built visualization routines or simply using numerical plain text outputs.

### 4.4 Pseudo-Random Number Generator

To obtain a substantially random initial state for the CA model, an external pseudo-random number generator was used. The generator was developed by the pLab team at the University of Salzburg that came out with a library of pseudo-random number generators called PRNG, which is available for download at http://random.mat.sbg.ac.at/.

The Explicit Inversive Congruential Generator(EICG) algorithm in PRNG was used to generate the random numbers for the initial states of the CA. The EICG was selected due to its excellent splitting properties. This is because ICG methods such as EICG differ strongly from other linear methods in its geometrical structure. According to Marsaglia[17] and Eichenauer-Herrmann[18] the ICG methods are less prone to biases compared to other methods.

## 5. METHODOLOGY

Modeling wave propagation in excitable media is very compute intensive. Solving large CA matrices on single processor machines will take a substantial amount of computing time. For more efficient computation a supercomputer would be required.

### 5.1 Nature of the Problem

The current CA model can be considered data-parallel. Hence, it is possible to divide the CA matrix and assign each portion to a different processing element. Boundary problems will arise since in CA it is necessary to get the values of the closest neighbors that maybe in another compute node. Therefore, the boundary information would have to be propagated.

### 5.2 Implementation

The implementation of CA models above was done on the AGILA HPCS. LAM MPI was used as the message passing library. All the codes were written in the C programming language. Visualization was done using MPE. The following steps were used to implement the three models:

1. Obtain the rules and parameters of the algorithm
2. Write the pseudo-code for the algorithm in serial
3. Parallelize codes using a data-parallel paradigm
4. Code the implementation in C using LAM-MPI libraries
5. Use MPE for visualization
6. Capture results using XVidcap and Xwd capturing utilities for X windows

## 5.3 Parallelization Issues

The CA model can be parallelized on N number of processes. Multiple processes can run on a single compute node. The approach taken for this problem was to use data-parallelization. This entails dividing the CA matrix into smaller matrices and distributing it across the different running processing elements which is to be defined during cluster boot-up[20].

To determine the portion of the CA matrix that belongs to a single process, the algorithm is defined by Equation 3 where $m$ is the width of the matrix that belongs to that particular process, $M_w$ is the width of the entire CA matrix, $N$ is the number of processes spawned for the current job and $R$ is the rank of the current process. The offset from the left most portion of the entire CA matrix will be defined by Equation 4 where $o$ is the offset of the individual matrix from the entire CA matrix.

$$m = \frac{M_w}{N + \alpha} \tag{3}$$

where
$\alpha = 1$ if $R < M_w mod N$
$\alpha = 0$ if $R \geq M_w mod N$

$$o = R * \frac{M_w}{N} \tag{4}$$

The matrix strip defined by Equation 3 and Equation 4 is the matrix of the process with rank $R$ in a pool of $N$ number of processes. Each process operates only on this assigned matrix strip. Each process will then allocate memory for a matrix $M^P$ and $M^T$ that is of the size $m$. $M^P$ is to be used as the main processing matrix. $M^T$ is a temporary storage matrix for miscellaneous operations. Both $M^P$ and $M^T$ will contain the data subset defined by Equation 3 and Equation 4 per process. Per timestep, data from $M^P$ is processed and the new values are stored in $M^T$. The values of $M^T$ are then swapped with the values of $M^P$ after the entire matrix strip is processed. This method prevents overlapping that may take place if only one matrix were used.

Another issue that was considered was that of the boundary information that needed to be propagated. This was done by following the procedure below:

1. Transfer the contents of $M^P[1]$ to the process defined by $R - 1$
2. Receive the contents of $M^P[0]$ from the process defined by $R - 1$
3. Transfer the contents of $M^P[m]$ to the process defined by $R + 1$
4. Receive the contents of $M^P[m + 1]$ from the process defined by $R + 1$

Two boundary vectors on both sides of the current compute node's $M^P$ are propagated to its neighboring compute nodes.

# 6. VISUALIZATION RESULTS

Below are representative simulations results obtained using the three CA algorithms. Figure 2 represents an image of the initial CA matrix that was generated by the pseudo-random number generator. This initial CA matrix served as the initial states for all three algorithms. The reason for using an external pseudo-random generator software is because the standard rand64 posix system function will produce random states only up to a certain extent and will begin to either show clear biases or would begin to recur.

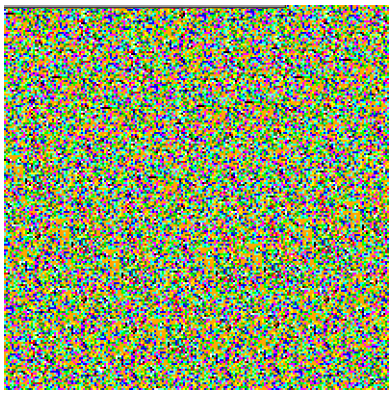

**Figure 2: Starting Random State**

## 6.1 Greenburg-Hastings CA

The Greenburg-Hasting CA uses a 300x300 CA matrix. The simulation has 14 intermediary states excluding the quiescent and excited states. The total of 16 states was chosen instead of a regular three state transition(quiescent intermediary excited), to make single state transition cycle take longer. This entails a process that would have to undergo more states before becoming rested again.

Figure 3 shows that there are small oscillation regions formed. After 300 iterations, the simulation stabilized and oscillated about these regions. The reason is because the system eventually reached a rested state since it necessary that the cycle be completed regardless of the state of its neighbors.

Physically, these regions could represent the points in which electrical excitation would occur. The center of these regions are the sources of the initial electrical discharge. It is clear

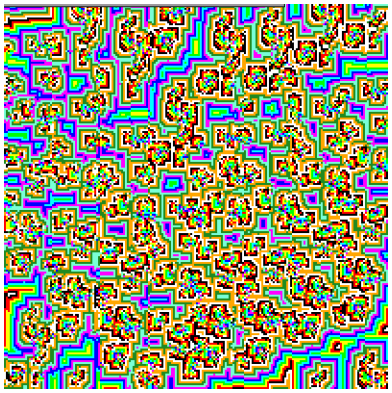

**Figure 3: After 300 iterations in a 300x300 Greenburg-Hastings Model**

in this process that a new site can be excited and it will inevitably reach a rested state regardless of the value of its neighbors. This process can model the spread of an electric discharge in neurons and other phenomenon.

## 6.2 Cyclic-Space CA

The Cyclic-Space CA uses a 300x300 CA matrix. The simulation has 9 intermediary states excluding the quiescent and excited states. This algorithm has a fewer number of states because having too much states would not allow the system to immediately oscillate. This is because it is harder for a single site to reach a rested state unless its neighbors are in a more rested state.

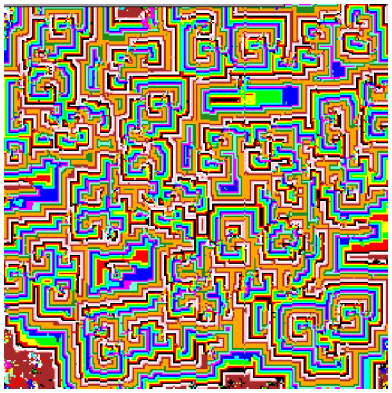

**Figure 4: After 600 iterations in a 300x300 Cyclic-Space Model**

Unlike the Greenburg-Hastings algorithm, the Cyclic-Space algorithm does not guarantee that the excited sites would reach the rested state after a certain number of iterations because this algorithm allows for the site to maintain its state if there are no sites around it that are more rested than the site. This explains why the spread of the oscillations with this algorithms is larger than that of the Greenburg-Hastings algorithm. This can be seen in Figure 4.

It can be seen from this simulation that when the wave front moves it excites the neighboring site. This differs from Greenburg-Hasting excitation since the excitation comes to an end in a short instance.

## 6.3 Hodgepodge Machine CA

The Hodgepodge Machine CA uses 300x300 CA matrix. The simulation has 14 intermediary states excluding the quiescent and excited states. This number of intermediary states was choosen for the same reason as that of the Greenburg-Hastings CA.

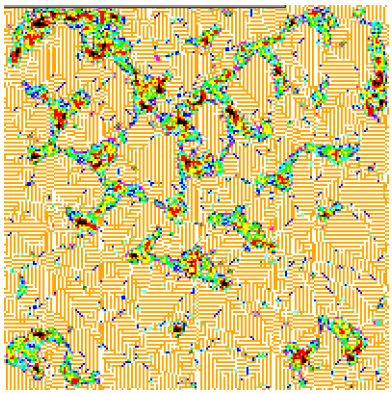

**Figure 5: After 600 iterations in a 300x300 Hodgepodge Machine Model**

This Hodgepodge Machine CA uses 0.5 as both the infection($k_1$ and $k_2$) constants and the speed of infection constant($g$). Looking at Figure 5, it can be noticed that there are more healthy or close to healthy sites. This is actually expected behavior since the Hodgepodge Machine CA is biased towards healing.

Figure 5 shows more oscillating centers as compared to Figure 1. This is because Figure 5 uses an initial random state. The reaction in Figure 1 was constructed placing the catalyst on only few points.

# 7. CONCLUSION

This study has shown that excitable media can be modeled using multi-state cellular automata on the AGILA High Performance Computing System. Some preliminary results were presented. Computational and implementation issues such as software tools selection, random number generators, visualization, boundary issues and matrix division issues were discussed.

The simulations presented in this study utilized only randomized starting points. Practical simulations would entail that the starting points of these simulations be predetermined. For a wave propagation simulation, it is possible to define the centers of the disturbances. In the case of infection problems, it is possible to determine the initial infection points. In the case of neuron excitation, the initial electric discharge can be specified.

# 8. ACKNOWLEDGMENT

This project was funded in part by the CHED Center of Excellence in Science and Mathematics Grant of the Ateneo de Manila University.

## 9. REFERENCES


[1] A. Dewdney. "Computer Recreations." Scientific American, (August 1988).

[2] R. Gaylord, P. Wellin. *Computer simulations with Mathematica: explorations in the physical, biological and Social Sciences*(New York:TELOS 1994).

[3] M.C. Cross, P.C. Hohenberg. Rev. Mod. Phys. 65, 851, (1993).

[4] J.R. Weimar, J.J. Tyson, L.T. Watson. "Diffusion and Wave propagation in Cellular Automata Models for Excitable Media." Physica D, 55, 309-327, (1992).

[5] J. Cartwright, E. Garcia, O. Piro. "Burridge-Knopoff models as elastic excitable media." Phys. Rev. Lett., 79, 527-530, (1997).

[6] V. Eguiluz, E. Garcia, O. Piro. "Dynamics of elastic excitable media." Int. J. Bifurcation and Chaos, 9, 2197-2202, (1999).

[7] S. Wolfram. "Various Articles on Cellular Automata." Available on the Internet as: http://www.stephenwolfram.com/publications/articles/ca/.

[8] P. Ruoff. "Excitability in a Closed Stirred Belousov-Zhabotinsky System." Chem. Phys. Lett. 90, 76, (1982).

[9] P. Meyer. "Computer Simulation of the Belousov-Zhabotinsky Reaction." Available on the Internet as http://serendipity.magnet.ch/hermetic/misc/chemsim/chemsim.htm. (1999).

[10] G. Peterson. "The Belousov-Zhabotinsky Reaction." Available on the Internat as http://online.redwoods.cc.ca.us/intruct/darnold/deproj/sp98/gabe/

[11] T. Sterling, D. Becker, and D. Savarese. Beowulf: A Parallel Workstation for Scientific Computation. *Proceedings of the Fourth 1995 International Conference on Parallel Processing (ICPP)*, vol. 1, pp. 111-14, (August 1995).

[12] R. Saldaña, J. Garcia, F. Muga II, W. Yu. "AGILA: the Ateneo High Performance Computing System". Proceedings of the First Philippine Computing Science Congress, Manila, Philippines, (Nov. 2000).

[13] R. Saldaña, J. Garcia, F. Muga II, W. Yu. "Development of a Beowulf-Class High Performance Computing System for Computational Science Application". Proceedings of the 18th National Physics Congress, Palawan, Philippines, (Oct. 27-29,2000).

[14] W. Gropp, E. Lusk, A. Skjellum. *USING MPI: Portable Parallel Programming with the Message-Passing Interface*(Massachusetts:MIT 1994).

[15] Ohio Supercomputing Center. "MPI Primer/Developing with LAM." *National Science Foundation Grant CCR-9510016.* (1996).

[16] W. Gropp, E. Lusk. "User's Guide for MPE: Extensions for MPI Programs". Available on the Internet as http://http://www-unix.mcs.anl.gov/mpi/mpich/docs/mpeguide/paper.htm.

[17] Marsaglia, G. Random numbers fall mainly in the planes. Proceedings of the National Academy of Sciences of the United States of America 60:25-28. (1968).

[18] Eichenauer-Hermann, J. Inversive congruential pseudorandom numbers avoid the planes. Math. Comp. 56:297-301. (1991).

[19] R. Saldaña. "A Report on the Workshop and Conference on Modeling, Simulation, and Scientific Computing (MODEL '99)." *Meeting of the Steering Committee of the High-Performance Computing in Asia-Pacific (HPC-Asia).* Maui High Performance Computing Center, Hawaii, (December 7-9, 1999).

[20] I. Foster. "Designing Parallel Programs". Available on the Internet as http://www-unix.mcs.anl.gov/dbpp/. (1995).