A Hybrid Approach to Reaction-Diffusion Processes Simulation

Olga Bandman
Supercomputer Software Department
ICMMG, Siberian Branch
Russian Academy of Science
Pr. Lavrentieva, 6, Novosibirsk, 630090, Russia
E-mail: bandman@ssd.sscc.ru

Abstract. A hybrid approach for simulating reaction-diffusion processes is proposed. It combines into a single iterative procedure Boolean operations of Cellular Automata Diffusion with real number computation of nonlinear reaction function. The kernel of the proposed approach is in constructing methods for transforming reals into spatial distribution of Boolean values. Two algorithms are proposed and illustrated by the simulation of some well studied typical reaction-diffusion phenomena. Computational features of the methods are discussed and problems for future research are outlined.

1 Introduction

There is a number of well known Cellular Automata diffusion and Gas-Lattice models [1,2,3], as well as some trials to find cellular automata simulating kinetic and chemical processes. Following [4], all these models should be considered as "alternatives rather than approximations of Partial Differential Equations (PDE) solutions". These discrete models have a number of computational advantages, the most important being the absolute stability of computation and the absence of rounding off errors. These properties attract the mathematicians, while the specialists in chemistry, biology and physics are interested in creating models of phenomena, which have no mathematical description at all. Such Cellular Automata (CA) are constructed on the basis of kinetic or chemical microscopic dynamics. Boolean cell states simulate the existence or the absence of an abstract particle (molecule, velocity component, concentration, etc.) at certain points of time and space. Cell operations are represented as Boolean functions of states in the cell neighborhood. To obtain physical interpretation of Boolean results, a sum of state values over an area around each cell is calculated. Two prominent examples are a deterministic chemical CA, proposed in [5], and a "Stochastic Cellular Automaton" from [6], which are intended for simulation chemical processes in active media. In [7] a reaction-diffusion CA is presented, based on a neurolilke model, whose elementary automaton executes a threshold function and has a refractory period after the active state. In [8,9] many very interesting industrial application of Cellular-Automata models are presented.
An important problem not yet completely solved in the above approaches is to prove the correspondence of the cellular array evolution to the modeled phenomenon, as well as the way of accounting physical parameters (density, viscosity, diffusion coefficient, pressure, etc) in the array function parameters. The most correct approach to solve these problems might be a natural experiment which, however, is impractical. But such experiments are sometimes impractical. Certain particular results have been obtained theoretically for the CA-diffusion with Margolus neighborhood [2] and for Gas-Lattice FHP-model [10]. In both cases the proofs of the CA evolution correspondence to the modeled phenomenon are done by reducing the CA to the PDE of the modeled phenomenon.

There are many problems also in studying reaction-diffusion processes by PDE analysis. They are investigated literally by the piece (equations of Gordon, Fitz-Nagumo, Belousov-Zhabotinsky, etc.), and with much difficulty, because analytical solutions are impossible due to the nonlinearity, and numerical methods are limited by stability and accuracy problems [12,13].

Unfortunately up to now no method is known for determining a CA-model of process when its PDE description is known. The latter is a system of first order PDEs, having in their right sides two additive terms: 1) a Laplacian to represent the diffusion, and 2) a nonlinear function to represent the reaction (in chemistry) or the advective process (in hydrodynamics), phase conversion (in crystallization), population evolution (in ecology). The first is perfectly modeled by CA, and the second is easy to count without the danger to make the computation unstable.

From the above it follows, that it makes sense to find methods which combine CA-diffusion with calculation of reaction function in reals. We propose to state the problem as follows: given a reaction-diffusion PDE, a discrete cellular algorithm is to be constructed whose evolution approximate that of finite-difference PDE. Obviously, it should be an iterative algorithm, at each step performing the operation of transforming spatially distributed Boolean values into the averaged and reals and the inverse operation referred to as allocation procedure. The latter is precisely the most crucial point of the algorithm. Thus, we propose to exploit well studied CA-models of a diffusion [3] combining it with the integer approximation of reaction function.

The motivation for such an approach contains two arguments. The first is based on the wish to use the great experience of nonlinear phenomena study by PDE solving. The second reason is to obtain rather simple discrete models to replace PDEs, the solution of which is sometimes impractical. We do not know attempts to use such an approach, so we shall try to fill the gap.

To give a mathematical background of the proposed methods the formalism of Parallel Substitution Algorithm (PSA) [14] is used, which allows to combine real number and Boolean computation in a unique iterative process.

Apart from Introduction and Conclusion the paper contains four sections. In the second section main concepts and formalisms used in the paper are presented. The general scheme and two algorithms of transforming PDE into a discrete cellular automaton are presented in the third section. In the forth section the
computer simulation results are given. In the short fifth section the properties of proposed methods are discussed and problems for future investigation are outlined.

2 Continuous and Discrete Forms of Spatial Dynamics Representation

2.1 Reaction-Diffusion Partial-Differential Equations

Let us consider reaction-diffusion process as a function of concentration of a certain substance of time and space. The traditional representation of the most simple one-dimensional reaction-diffusion process has the form of the following PDE:

\[ \frac{du}{dt} = d \frac{\partial^2 u}{\partial x^2} + F(u) \]  

(1)

where \( u \) is a variable with the normalized domain from 0 to 1, \( t, x \) are continuous time and space, \( d \) is a diffusion coefficient, \( F(u) \) a differentiable nonlinear function, satisfying certain conditions, which in [11] are given as follows.

\[
F(0) = F(1) = 0; \quad F(u) > 0 \quad \text{if} \quad 0 < u < 1;
F'(0) = \alpha; \quad \alpha > 0; \quad F'(u) < \alpha; \quad \text{if} \quad 0 < u < 1;
\]  

(2)

The conditions (2) are met by a second order polinome (Fig.1a) of the form

\[ F(u) = \alpha u (1 - u); \]  

(3)

![Fig. 1. The nonlinear functions used in typical reaction-diffusion equation](image)

Equation (3) describes also the propagating front of the autocalititic reaction (Field-Noyes model [15]). The equation (1) with \( F(u) \) like (2) is studied in details [11,14]. It is known, that with the initial conditions

\[
u(x, 0) = \begin{cases} 
1 & \text{if} \ x < 0, \\
0 & \text{if} \ x \geq 0.
\end{cases}
\]  

(4)
it generates an autowave of the type \textit{propagating front}, which moves (at \( t \to \infty \)) with the velocity

\[ V = 2\sqrt{d\alpha}, \]  

(5)

In ecological research functions satisfying (2) are classified as \textit{logistic} ones and considered to be basic, although some others are also studied, for example, those represented by third order polinomes (Fig.1b), such as

\[ F(u) = \alpha u(1 - u)(u - k), \quad 0 < k < 1, \]  

(6)

which meet the following conditions:

\[
\begin{align*}
F(0) &= F(k) = F(1) = 0, \quad 0 < k < 1; \\
F'(0) &< 0, \quad F'(k) > 0, \quad F'(1) < 0,
\end{align*}
\]

(7)

With \( F(u) \) of the form (6) the propagating front velocity is

\[ V = \sqrt{\alpha/2(1 - 2k)} \]  

(8)

Moreover, when the initial condition have the form

\[ u(x, 0) = \begin{cases} 
  u_0 & \text{if } |x| \leq l, \ k < u_0 \leq 1, \\
  0 & \text{if } |x| > l,
\end{cases} \]

(9)

referred to as a "flash", then the wave may attenuate, if \( F(u)_{\text{max}} \) is not sufficiently large.

The above analytical characteristics of some simple and well studied reaction-diffusion phenomena are further used for comparing them with the similar ones obtained by simulation of CAs. Obviously, their correspondence would confirm the correctness of the proposed method.

2.2 Parallel Substitution Algorithm for Discrete Cellular Simulation

Parallel Substitution Algorithm (PSA) [14], is a convenient formalism for representing spatially distributed processes. It suits well to be used for our purpose, due to the fact that it allows to deal both with Boolean and real data. The following properties of PSA make it powerful for this purpose.

- PSA processes \textit{cellular arrays}, which are sets of cells given as pairs \( C(A, M) = \{(a, m)\} \), where \( a \in A \) is a cell \textit{state}, and \( m \in M \) is a cell \textit{name}. \( A \) - is an alphabet (in our case it is Boolean or real). \( M \) is a \textit{naming set} (in general case a countable one). On the set \( M \) naming functions \( \phi_i : M \to M \) are defined. The naming set is the set of discrete Cartesian coordinates, given as \( m = (i, j, k) \). In our case only shift naming functions are used. A set of naming functions form determines the names of any cell \textit{neighborhood}.

- Operations over a cellular array are specified by a set \( \Phi = \{\Theta_i\}, i = 1, \ldots, n \), of parallel substitutions of the form

\[ \Theta_i : C_i(m) \ast S_i(m) \to S'_i(m). \]  

(10)
where
\[ C_i(m) = \{ (y_{ik}, \phi_{ik}(m)) : k = 0, \ldots, q_y \}, \]
\[ S_i(m) = \{ (x_{ij}, \phi_{ij}(m)) : j = 0, \ldots, q_x \}, \]
\[ S'_i(m) = \{ (f_{ij}(X,Y), \phi_{ij}(m)) : j = 0, \ldots, q_x \}, \]

In (10,11) \( C_i(m), S_i(m) \) and \( S'_i(m) \) are local configurations, * meaning their union for any \( m \in M \). Further only stationary parallel substitutions are used, in which the neighborhoods of \( S_i(m) \) and \( S'_i(m) \) are formed by identical sets of naming functions, which contain an identical naming function \( \phi(m) = m \) referred to as a central cell of the substitution. A parallel substitution should meet the following conditions:

1) no pair of naming functions values in (11) are equal,
2) \( x_{ij} \in X, y_{ik} \in Y \) are state variables or constants and \( f_{ij}(X,Y) \) are cellular functions with the domain from \( A \).

- A substitution is applicable to \( C(A,M) \), if there is at least one cell named \( m \in M \) such that \( C_i(m) \cup S_i(m) \subseteq C(A,M) \). Application of a substitution at a cell \((a,m) \in C(A,M)\) yields changing cell states in \( S_i(m) \) called the base by the corresponding ones from \( S'_i(m) \), the set of cells \( C_i(m) \) (called a context) remaining unchanged.
- There are three modes of parallel substitutions application.

1) Synchronous mode, when at each step all substitutions are applied at all cells at once. At this case in order to provide determinism of the computation, one should be careful not to allow the substitutions be contradictory when \( |S'_i(m)| > 1 \) [14].
2) Asynchronous mode, when any substitution is applied at any cell, one application being allowed at a time. There is no danger of contradictoriness in this case, but a generator of random numbers should be used to determine a next cell to which the substitutions are to be applied each time.
3) 2-step synchronous mode, when cellular array under processing is to be partitioned into two parts, and at each time-step the substitutions act at one of them only.

- A Parallel Substitution Algorithm (PSA) is a set of substitutions together with indication of the mode of application. Implementation of a PSA over a cellular array \( C \) is an iterative procedure, where at each step the substitution set is executed at a set of cells, according to the given mode. The algorithm stops when no substitution is applicable to the array.
- A PSA may process not only one but a number of interacting arrays \( C = \{ C_1, \ldots, C_n \} \) as well. In the latter case each substitution \( \Theta_i \) is allowed to be applied to only one array. It means that its base \( S_i(m) \) is located in only one \( C_l \in C \), i.e. \( m \in M_l \). As for the context \( C_i(m) \), it may be located at any array, moreover, it may be composed of a number of local configurations, located in different arrays, i.e.

\[ C_i(m) = C_1(m_1) \ast \ldots \ast C_k(m_k), \quad k \leq n; m_j \in M_j. \] (12)

PSA is further used to represent reaction-diffusion processes by discrete fine-grained parallel algorithms.
3 Combining CA-diffusion with Finite-Difference Reaction

3.1 General Scheme of the Computational Process

Without loss of generality let's consider the two-dimensional case. After time and space are transformed to the discrete form resulting in \( x = hi, y = hj, t = n\tau \), where \( i, j, n \) are integers, \( h = 1 \), the equation (1) looks as follows.

\[
u_{ij}'(t + 1) = u_{ij}' + \tau dL(u_{ij}') + \tau F(u_{ij}'), \quad (13)\]

where \( L(u_{ij}') \) is a Laplacian, \( u_{ij}' = u_{ij}(t) \) are variable values in real numbers.

Let us now focus on the most discrete form of process representation, when concentration values are given in Boolean form, i.e. \( u \in \{0, 1\} \). Using the PSA notation we consider coordinates \( i, j \) as cell names \( (i, j) \in M \), and concentration values as cell states \( a \in A \), the process to be simulated being given by a set of parallel substitutions acting on the cellular array \( C \in A \times M \). The correspondence between continuous and discrete forms of representation is that \( u_{ij}'(t + 1) \) and \( u_{ij}'(t) \) are averaged values over a certain area \( Av(ij) \) around a cell named \( (i, j) \), referred to further as averaging area,

\[
u_{ij}' = \sum_{Av(ij)} u_{ij}. \quad (14)\]

When \( F(u) = 0 \), then (1) and (13) describe a "pure diffusion" which has some simple and well studied CA-models. The most known of them called Block-Rotation CA-model is theoretically proved \([2]\) to be equivalent to Laplace equation with \( d = 3/2 \) (in 2D case). Moreover, in \([3]\) it is shown how to use the model with any diffusion coefficient.

The above approves the possibility to decompose each step of the iterative simulation procedure into three operations: application of a CA-diffusion rule, computation of the reaction function and combining the result.

Accordingly, the array \( C \) under simulation is partitioned into three parts: diffusion array \( C_D \) with Boolean cell states, reaction array \( C_R \) and resulting array \( C' \) both with real cell states, the naming sets of the parts being in one-to-one correspondence.

CA-diffusion rules are applied to the diffusion array resulting in \( C_D(t + 1) \). As for the reaction function computation, it may be accomplished only in reals, resulting in \( C_R(t + 1) \). At last, to obtain the next state both results should be combined in such a way that the result of the \( i \)-th iteration satisfies the following conditions:

\[
C(t + 1) = (C_D(t + 1)) \odot C_R(t + 1), \quad C'(t + 1) = Av(C(t + 1)), \quad (15)
\]

where "\( \odot \)" means cell-wise states summing, \( Av(C_D(t+1)) \) has states obtained by (14) applied to corresponding states of \( C_D(t+1) \). Both cellular arrays \( C(t+1) \) and
Fig. 2. General scheme of one iteration of the iterative hybrid method of simulation the reaction-diffusion process given by an equation of the form (13)

$C'(t+1)$, representing $t$-th iteration result in Boolean and real form respectively, are the initial arrays for the next iteration (Fig. 2).

The main problem to be solved for constructing the computational algorithms according to the general scheme given in Fig. 2 is to find the procedure which is inverse to averaging. It comprises the distribution of "ones" over the cellular array in a way, that guarantees given averaged values and is referred to further as allocation procedure. Allocation is just the problem that constitutes the kernel of the proposed approach. Two allocation procedures determine two algorithms for combining Boolean and real computations into a single iterative procedure. The first is called a multilayer method. It requires one dimension to be added to the diffusion cellular space. So, the array is treated as a multilayer one. An iteration of CA-diffusion algorithm is executed in all diffusion layers independently, and the averaging and the allocation is performed over the subarray, containing corresponding cells names, i.e. differing only by the layer number. The second method is referred to as a three-layer method. In it the CA-diffusion is performed in only one layer of the array where the averaging and allocation is performed over the averaging area which contains cells whose spatial coordinates differ in no more than a constant $\rho$, referred to as a radius of averaging. The allocation is done by inverting the cell state with a probability, depending of the number of bits to be allocated and the neighborhood size.

In the following subsections the above methods are given formally in terms of PSA. CA-diffusion algorithms used in the methods are not discussed there, they are given in brief in the examples of section 5.
3.2 A Multilayer Reaction-Diffusion Method

To simulate a reaction-diffusion process in an n-dimensional space an (n+1)-
dimensional cellular array should be taken, which is further considered as a
multilayer n-dimensional one. So, the naming set of the array in 2D case is
\( M = \{ (i, j, k) \} \), where \( i, j = 0, 1, 2, \ldots, \) are coordinates of 2D infinite space,
further referred to as spatial coordinates, and \( k \in \{ 0, 1, \ldots, L \} \) is a layer number.
The naming set is partitioned into two subsets: \( M = M_D \cup M_R \), \( M_R = \{ (i, j, 0) \} \)
contains the names of the zero layer cells, \( M_D \) – the cell names of all other layers.
\( M_D \) in its turn is partitioned into subsets of names, differing only by the layer
numbers: \( M_D = \cup_{k=1}^{L} M_{ij} \), where \( M_{ij} = \{ (i, j, k) : k = 1, 2, \ldots, L \} \) is the average
area forming the averaging subarray \( C_{ij} \) of the cell, with spatial coordinates
\((i, j)\).

Cells with names from \( M_D \) and \( M_R \) have the state alphabets \( A_D = \{ 0, 1 \} \)
and \( A_R = \mathbb{R} \), respectively.

The subarray \( C_R = \{ (v, (i, j, 0)) : v \in A_R, (i, j, 0) \in M_R \} \) plays a twofold role:
it is destined for computing the nonlinear function and storing the averaged
result. In each k-th layer from the diffusion part \( C_D = \{ (u, (i, j, k)) : u \in A_D, (i, j, k) \in M_D \} \) one of the chosen 2D CA-diffusion algorithm is realized.
The scheme of the multilayer algorithm is as follows.

Given an initial cellular array \( C(0) = C_D(0) \cup C_R(0) \) of finite size \( G \times H \times L \) where
\( M = \{ (i, j, k) : i = 0, 1, \ldots, G - 1, j = 0, 1, \ldots, H - 1, k = 0, \ldots, L \} \), and cells of diffusion layers \( k = 1, \ldots, L \) have Boolean states, zeros and ones
being distributed over the layer in such a way that the number of ones in the
averaging subarray of the cell is equal to the initial concentration of the substance
in the space under simulation. These concentration values \( v_{ij} = u_{ij0} \) given in
real numbers are cell states of the reaction layer \((k = 0)\).

The computation is an iterative procedure, each t-\( \text{th} \) iteration being composed
of the following steps.

**Step 1.** In all diffusion layers an iteration of CA-diffusion transition
rule is performed. It results in changing the cell states of \( C_D(t) \), i.e.
\[
C_D(t + 1) = \text{Diff}(C_D(t)).
\]
It should be noted, that at this step there is no interactions between the layers.

**Step 2.** In each cell \( (v, (i, j, 0)) \in C_R \) the nonlinear function \( F(v) \) is
computed and the nearest integer to the result becomes the cell state.
\[
\Theta_1 : \{ (v, (i, j, 0)) \} \rightarrow \{ (y, (i, j, 0)) \}, \text{ where } y = \text{Int}(F(v)).
\]

**Step 3.** Allocation operation is performed as follow. In each subset
\( C_{ij} \in C_D \) the amount of cells equal to the state of the cell \( (y, (i, j, 0)) \) is inverted
according to its sign. If \( y > 0 \), then cell states \( u = 1 \) are inverted, else, if \( y < 0 \),
the same is done with cell states \( u = 0 \).
\[
\Theta_2 : \{ (|y| > 1, (i, j, 0)) \} \rightarrow \{ (u, (i, j, k)) \}
\Theta_3 : \{ (|y| > 1, (i, j, 0)) \} \rightarrow \{ (|y| - 1, (i, j, 0)) \}.
\]
$k$ ranging from 1 to L.

Allocation operation results in the subarray $C_D(t + 1)$.

- **Step 4.** Averaging operation over all $M_{ij}$ is performed according to (14).

$$\Theta_4: \{(u_1, (i, j, 1)), \ldots, (u_L, (i, j, L))\} \ast \{(v, (i, j, 0))\} \rightarrow \{(u', (i, j, 0))\}$$ (19)

where

$$u' = \sum_{k=1}^{L} u_{ijk},$$ (20)

Averaging operation results in the subarray $C_R(t + 1)$.

- **Step 5.** If the whole computation is not completed, i.e. $t + 1 < T$, then the subarrays obtained in step 3 and 4 are taken as the initial ones for the next iteration, else the computation is considered to be completed and $C(t + 1) = C_D(t + 1) \cup C_R(t + 1)$ is its result.

In section 4.1 this algorithm is illustrated by simulation results of two types of autowaves: 1D and 2D propagating fronts.

### 3.3 Three-Layer Reaction-Diffusion Method

Three-layer hybrid method provides for a three-layer array, whose spatial coordinates together with the layer number ($k = 0, 1, 2$) form the naming set. For definiteness 2D-case is further considered. Let the layer with $k = 0$ be the reaction subarray $C_R = \{(y, (i, j, 0))\}$, the layer with $k = 1$ the diffusion layer $C_D = \{(u, (i, j, 1))\}$ and the last one with $k = 2$ the layer storing the averaged result it contains and counts the averaged diffusion results $C' = \{(v, (i, j, 2))\}$.

Respectively, $u \in A_D, y \in A_R$. Averaging is performed over the neighborhood $Q = \{i + h, j + l, 1: h, l = -r, \ldots, r\}$. Initially $C_0$ and $C'(0)$ contain the Boolean distribution and averaged values of concentration at $t = 0$, $C_R(0)$ has zero-states in all cells.

The simulation procedure is an iterative one with the $t$-th iteration consisting of the following steps.

- **Step 1.** In the diffusion layer an iteration of a CA-diffusion algorithm is executed resulting in

$$C_D(t + 1) = \{(u, (i, j, 1)) : i = 0, \ldots, M - 1; j = 0, \ldots, N - 1\}.$$ (9a)

**Step 2.** In the cells of reaction subarray $C_R$ the nearest integer to the function $F(v)$ value is computed according to the substitution

$$\Theta_5: \{(v, (i, j, 2))\} \ast \{(0, (i, j, 0))\} \rightarrow \{(y, (i, j, 0))\}, \text{ where } y = \text{Int}(F(v)).$$ (21)

- **Step 3.** Allocation operation is performed by inverting cell states in the diffusion layer according to the following probabilities.

$$p = \frac{y}{Q}$$ if $x = 0$ & $y > 0$

$$p' = \frac{y}{v}$$ if $y < 0.$ (22)
Two parallel substitutions executing this operation are as follows.

\[ \Theta_6 : \{(v, (i, j, 2)), (y, (i, j, 0))\} \ast \{(u, (i, j, 1))\} \rightarrow \{(\phi(u, v, y), (i, j, 1))\}. \]

\[ \Theta_7 : \{(v, (i, j, 2)), (y, (i, j, 0))\} \rightarrow \{(0, (i, j, 2)), (0, (i, j, 0))\}, \]

where

\[ \phi(u, v, y) = \begin{cases} 1, & \text{if } (u = 0) \& (y > 0) \& (\text{rand}(1) < p); \\ 0, & \text{if } (u = 1) \& (y < 0) \& (\text{rand}(1) < p'); \end{cases} \]

- Step 4. Averaging operation is performed in all cells of the averaging subarray \( C_D' \) according to the substitution

\[ \Theta_8 : \{(u_{i-r,j-r}(i-r, j-r, 1)), \ldots, (u_{i+r,j+r}(i+r, j+r, 1))\} \ast \{(0, (i, j, 2))\} \rightarrow \{(v, (i, j, 2))\}, \]

where

\[ v = \sum_{h=-r}^{h=r} \sum_{l=-r}^{l=r} u_{i+h,j+l} \]

- Step 5. If the whole computation is not completed, i.e. \( t + 1 < T \), then the subarrays obtained in steps 3 and 4 are taken as the initial ones for the next iteration, else the computation is considered to be completed and \( C(t + 1) = C_D(t + 1) \cup C' \) is its result.

The above method of allocation is approximate. It is absolutely accurate only in case of uniform probability distribution over the cell neighborhood. In case of \( p_{ij} \) variation, the expectation \( \mathcal{M} \) of the event \( Y \) meaning that the number of inverted cell-states in the neighborhood is \( y_{ij} \) is equal to

\[ \mathcal{M}(Y) = \sum_{Q} p_{ij} \simeq y_{ij}. \]

Moreover the approximation error approaches to zero when the deviations are of different signs. At any case some corrective coefficients may be provided to reduce the error to any small value.

4 Computer Simulation Results

4.1 Simulating 1D Propagating Front by Multilayer Method

As it was mentioned above, the use of hybrid simulation methods suggests to choose appropriate CA diffusion model to be included in the algorithms.

Comparative analysis of CA diffusion models presented in [3] allows to make the following conclusion: the model, called a naive CA-diffusion should be used for 1D-case, and the Block-Rotation method (BR-method) is the best for the 2D one. The diffusion coefficient of naive CA-diffusion is not known. So, it has been obtained by simulation by comparing the results with those obtained solving
PDE, the result being $d = 1.1$. So, the hybrid method of one-dimensional propagating front combines 1D naive CA-diffusion [3] with the nonlinear function of the form (3).

Naive CA-diffusion is the most simple model of equalizing the concentration by local stirring along one direction. Let it be the direction along the axis $j$ of the 1D $(L+1)$-layer cellular array $C = \{(u_{jk}, (j, k)) : j = 0, 1, \ldots, G; k = 0, \ldots, L\}$, $C = C_D \cup C_R$. The diffusion subarray $C_D$, which contains the layers $(k = 1, \ldots, L)$ uses the alphabet $A = \{0, 1\}$, the variables being specified by $u_{jk}$. Naive CA-diffusion dictates to each cell to exchange states with one of its randomly chosen neighbor. To avoid contradictoriness the asynchronous mode of execution is used. It means, that at each time only one (randomly chosen) pair of cells exchanges states. So, one iteration which corresponds to Step 1 of the general scheme comprises $G \times L$ times, each time the following operations should be executed.

1) Two random numbers $j, k$, $0 \leq j < G$, $1 \leq k < L$, are obtained. They indicate the cell, to which the algorithm is applied.

2) A context cell $(\alpha, m_0)$ is introduced to indicate the neighbor with whom the cell should interact. The neighbor is determined according to the probability $p = 1/2$. So, if a random number $r \leq 1/2, (0 < r < 1)$, then $\alpha = 1$, which means, that the neighbor to interact with is at the right side of the cell $(i, j)$. If $r > 1/2$ then $\alpha = 0$ and the left neighbor is chosen.

3) The following parallel substitutions are applied to a chosen cell of $C_D$.

\[
\begin{align*}
\Theta_0 : & (1, m_0) \ast \{(u, (j, k)), (u', (j + 1, k))\} \rightarrow \{(u', (j, k)), (u, (j + 1, k))\}; \\
\Theta_0 : & (0, m) \ast \{(u, (j, k)), (u', (j - 1, k))\} \rightarrow \{(u', (j, k)), (u, (j - 1, k))\};
\end{align*}
\]

(27)

The other steps are executed in complete accordance with the general scheme (section 3.2). The difference is only in the absence of coordinate $i$ in the names.

\[\text{Fig. 3. A pair of snapshots of the profiles of 1D propagating front, obtained by the multilayer method with naive asynchronous CA-diffusion and and nonlinear function of the form (3)}\]

In Fig.3a two profiles of 1D propagating front are given. They have been obtained by simulation using the multilayer algorithm with $N = 128, L = 32, F(u)$
of the form (3) with $\alpha = 1.2$ Propagation velocity has been determined by analyzing the fronts profiles, obtained from the series of similar experiments. The coincidence with that, obtained by formula (5) is in the limits of the accuracy of the experiment. For example, in accordance with the well known character of propagating front behavior \cite{11,14} its propagating velocity decreases with time, approaching (according to (5)) at $t = \infty$ to $V_0 = 2.3$ (with $\alpha = 1.2, d = 1.1$).

4.2 Simulating 2D Propagating Front by Multilayer Method

The algorithm to be presented here combines the 2D BR-diffusion with a nonlinear functions (3) and (6). BR-diffusion, which is referred to in \cite{2} as a CA with Margolus neighborhood, works in a two-step mode of execution over a cellular array $C = \{(u, (i, j, k)) : i = 0, \ldots G - 1, j = 0, \ldots, H - 1, k = 0, \ldots, L\}$. In the diffusion layers ($k = 1, \ldots, L$), two types of cells are distinguished: even cells and odd cells. Even cells have both $i$ and $j$ even, odd cells have both $i$ and $j$ odd. A cell from the even (odd) subset is considered to be a central one for each $k$-th layer block $B(i, j, k) = \{(u_1, (i, j, k)), (u_2, (i, j + 1, k)), (u_3, (i + 1, j, k)), (u_4, (i + 1, j + 1, k))\}$. Similar blocks are formed by cells belonging to the even subset. Clearly, the intersections between blocks from one the same subset are empty, from which follows the noncontradictoriness. To indicate the type of blocks an additional context cell $(\beta, m_0)$ is introduced, $\beta = 0, \beta = 1$ corresponding to even and odd cells, respectively.

Each diffusion iteration (Step 1) consists of two times: at the even time even blocks turn with probability $p = 1/2$ to $\pi/2$ either clockwise or counterclockwise. To indicate the rotation direction an additional context cell $(\gamma, m_1)$ is introduced. If $\gamma = 1$ then the rotation is clockwise, else – counterclockwise. In PSA notation it looks like this.

\[ \Theta_{11} : \{(1, m_0)(1, m_1)\} * \{(u_1, (i, j, k)), (u_2, (i, j + 1, k)), (u_3, (i + 1, j + 1, k)), (u_4, (i + 1, j, k))\} \rightarrow \{(u_4, (i, j, k)), (u_1, (i, j + 1, k)), (u_2, (i + 1, j + 1, k)), (u_3, (i + 1, j, k))\}; \]

\[ \Theta_{12} : \{(1, m_0)(0, m_1)\} * \{(u_1, (i, j, k)), (u_2, (i, j + 1, k)), (u_3, (i + 1, j + 1, k)), (u_4, (i + 1, j, k))\} \rightarrow \{(u_2, (i, j, k)), (u_3, (i, j + 1, k)), (u_4, (i + 1, j + 1, k)), (u_1, (i + 1, j, k))\} \]  

(28)

For the odd times the substitutions differ from (30) only by the context cell $(\beta, m_0)$, which is in this case $(0, m_0)$. Other steps are in accordance with the general scheme of the algorithm.

2D propagation front simulation has been done for the cellular array with $G = H = 64, L = 32$ with two different nonlinear functions, given by (3) and by (6). The initial array had the cells states $u = 1$ in the cells from $\{(u, (i, j, k)) : (G/2 - g) < i < (G/2 + g); (H/2 - g) < j < (H/2 + g); k = 1, \ldots, L\}$, $g = 6$, the rest of cells had $u = 0$. Such initial states are referred to as a flash in the ecological research. In Fig.4a two snapshots of front profile propagating from the flash are shown, the reaction function being of the form (3) with $\alpha = 1.2$. In Fig.4b two snapshots are shown, which are obtained simulating diffusion-reaction
with the function $F(u)$ of the form given by (6) having $F_{\text{max}}(u)$ not sufficiently large to support the front to propagate. In this case the flash diminishes and disappears.

![Fig. 5. The initial state and the snapshot at $T = 16$ of two propagating fronts](image)

In Fig. 5 a diffusion two propagating towards each other fronts array shown after $T = 16$ iterations (Fig.5b), being initiated by two dense spots (Fig.5a).

### 4.3 Simulating 1D Propagating Front by Three-Layer Method

Let us combine naive 2-step CA-diffusion with reaction function of the form (3). The array has three 1D layers: $C_R$, $k = 0$, $C_D$, $k = 1$ and $C'$, $k = 2$. One-dimensional 2-step naive diffusion is similar to the Block-Rotation diffusion. The array is partitioned into two subsets: a subset of even cells, which form even blocks: $\{(u_1, j, 0), (u_2, j+1, 0) : j = 0, 2, \ldots, N-2\}$, and a subset of odd ones: $\{(u_1, j, 0), (u_2, j+1, 0) : j = 1, 3, \ldots, N-1\}$. Each diffusion iteration consists of two times: at even time cells of even blocks exchange states. The same do the odd blocks at odd times. Substitution of state exchange is as follows.
\[ \Theta_{13} : \{(u_1, \langle j, 0 \rangle), (u_2, \langle j + 1, 0 \rangle)\} \rightarrow \{(u_2, \langle j, 0 \rangle), (u_1, \langle j + 1, 0 \rangle)\}. \] (29)

The averaging (step 2) is performed using \( \Theta_{8} \), neighborhood size being chosen according to the required accuracy. Reaction function (step 2 of the algorithms in section 4.3) is computed in cells of \( C_R \) according to \( \Theta_{2} \).

The allocation procedure is performed in the cells of \( C_D \) with the account of corresponding probabilities counted by (22) and (23). In Fig.6 two snapshots of a propagating front obtained by this method are shown, propagation speed being in good accordance with the theoretic value.

![Fig.6](image)

**Fig. 6.** Two snapshots of propagating front profiles, obtained by stochastic method

5 Characterisation of the Proposed Method

To make a conclusion about the proposed method of reaction-diffusion phenomena simulation, its computational characteristics (performance, accuracy, stability) should be assessed. The most correct way to do this is to compare them to the similar ones, for PDE solution. Among the scope of PDE solution methods the finite-difference one is chosen, because of the two following reasons. The first is its fine-grained parallelism, which is considered as a very important feature, due to its simplicity and capability of decomposition. The second is its simulation similarity to a CA evolution, which allows to consider the comparison to be correct.

On this stage of proposed method development only qualitative comparison is possible. The qualitative assessment may be done after a long and hard work both theoretical and experimental. Here only some considerations can be applied to the problems of the above characteristics determining. They are as follows.
Accuracy. The accuracy is determined by two types of errors. The round off errors and approximation errors. Round off errors are very small in the proposed method, because the CA-diffusion is absolutely free of them, as for averaging and reaction function calculation they might be done in integers. Approximation errors emerge in the procedures of counting the reaction function, as well as in the stochastic updating cell neighborhood. Since these errors depend on the size of the averaging space, the price for the accuracy is the size of the array.

Stability of the computation. According to the finite-difference PDE theory the stability of the iterative algorithm to compute Laplace operator is conditioned by a relationship among a diffusion coefficient, and spatial (h) and time discretization (τ) steps. To meet these conditions the steps are to be chosen sufficiently small, which yields in large time of computation. When the proposed CA-method is used, no restriction on computation stability is imposed. The choice of discretization steps is done according to required accuracy and smoothness of the resulting dependencies (absence of ”CA noise”).

Performance, which is the inverse of the time, needed to solve a given problem may be assessed by three time parameters: 1) number of iterations for reaching the result, 2) number of elementary operations in each iteration, and 3) number of cells (grid nodes in PDE) needed to provide the required properties. Each of the above time parameters needs to be of assessed with the account of many factors: kind of nonlinearity, spatial dimension, number of variables, etc. At this stage of study the following remarks may be done. The amount of iteration is expected to be less than when the corresponding PDE is solved. The number of elementary operations, the amount of bit-operations should be computed and compared to that needed for PDE solution. It is obvious, that the amount of bit operations is less for CA-diffusion in each iteration, but the averaging and updating procedures may override the difference. As for the size of the array, on one hand it can be smaller than for the PDE solution because the value of h is not restricted by stability conditions, on the other hand, it should be larger to diminish the CA-noise.

The above considerations show that a correct comparison of CA-model with finite-difference solution is a separate hard task, which may be solved basing on the considerable body of experience. This task is most likely to be formulated as follows: for certain class of reaction-diffusion PDEs the domain of parameters should be found such that the proposed approach is preferable.

6 Conclusion

Two methods for constructing CA representation of reaction-diffusion PDE are presented, both using the approach of combining known CA diffusion models with conventional computation of nonlinear function. The methods are illustrated by simulation on array of limited size. Simulation results coincide with those known from the corresponding PDE analysis. Some considerations are applied to the assessment of the approach and future investigations, among which the most important is to promote applications and accumulate experience.
References