Tensor approach to software implementation of cellular automata model of diffusion

A K Matolygin, N A Shalyapina, M L Gromov and S N Torgaev
Tomsk State University, Tomsk, Russia
amatolygin@mail.ru

Abstract. The article provides a theoretical description of the tensor approach to the software implementation of the cellular automaton model of diffusion. A block-synchronous cellular automaton with a neighborhood of Margolus was chosen as a model. Within the framework of the proposed approach, the rotation of the automaton block clockwise or counterclockwise can be carried out using a special operation on tensors, namely convolution. The approach discussed in this paper allows you to use various frameworks aimed at parallelizing computations to organize scientific calculations on modern powerful graphics adapters.

1. Introduction
The role of mathematical modeling in the study of various dynamical systems is important. If we are dealing with a spatially distributed dynamical system that includes processes such as diffusion, traditionally the modeling procedure is reduced to solving a system of partial differential equations (SPDE). However, the scale and complexity of the tasks to be solved lead to certain difficulties and make the use of SPDE almost impossible. The essence of this problem is as follows: the solution of stationary problems of mathematical physics is usually performed by implicit methods that do not impose restrictions on computational stability, which makes them faster than explicit ones. However, the parallel implementation of these methods requires additional efforts of both mathematicians and programmers, and is always associated with the degradation of the parallelization efficiency with an increase in the number of processors. When solving nonstationary problems, preference is given to explicit methods that can be easily and efficiently parallelized, but are limited by the conditions of computational stability (require a small time step). [1]

In addition to the traditional approach to modeling spatially distributed systems, numerical methods and models based on them have been actively developed. One of such models, acting as an alternative to differential equations [2], is the cellular automaton (CA). The main advantages of the CA over other numerical models are as follows:

1. The parallel structure of the model allows the use of high-performance parallel computing devices, which can significantly increase the computing performance.
2. The simplicity of setting the boundary conditions makes it possible to use the CA where the solution cannot be obtained with the aid of the SPDE.

These advantages prompted many researchers to use CA for solving numerous problems, for example, modeling chemical reactions [3], processes occurring in conductors [4], studying the crystallization process of various substances [5], plasma-laser processes [6], and some social processes [7].
Modeling with the aid of the CA consists in finding the states of an ensemble of cells of the automaton after a given number of cycles of the automaton operation. The state of the process or object under study can be characterized from the obtained set of states. It follows from this that two subproblems arise in the field of cellular automata modeling. First, it is necessary to correctly select the parameters of the cellular automaton, which include: the shape of the cells, the size of the automaton field, the type of neighborhood, etc. The second problem facing the researcher is the need for a software implementation of a cellular automaton with a certain behavior. Despite the fact that the solution of the first problem is a creative and interesting task, there are already many works on the use of CA in the modeling of certain processes. Accordingly, when building your model, you can rely on these works. The construction of a cellular automaton implementation is the main task of our work.

Clusters are usually used for parallel computing. This approach has a number of significant disadvantages. First, the researcher must be able to interact with the cluster. Because of their high cost, supercomputers are acquiring a community center. Parallel computing programs are quite demanding in their implementation. This imposes the requirement that the program code must be debugged on the cluster prior to production use of this code.

In [8], we proposed and considered an alternative approach to CA the software implementation. This approach is based on the idea of using graphics adapters. Advantages of graphics adapters:

- They contain several computing cores, which allows operations to be performed in parallel.
- Affordable price, which allows you to have a computing device for personal use.
- There are special working environments - frameworks. These frameworks help to efficiently and quickly create software implementations of their approaches without interrupting the study of possible options for implementing parallel programs.

The positive result obtained in [8] inspired us to apply the proposed approach to the problem of modeling the diffusion process.

Thus, the article consists of the following parts: in the first section, the theoretical foundations necessary for describing the CA are given, in the second section, the existing CA-models of diffusion and the principles of their functioning are briefly described, in the third section we demonstrate a tensor approach to the software implementation chosen by the CA diffusion models.

2. Preliminaries

We will assume that the cellular automaton is a set of Moore machines [9]. The set $A = \{S, \hat{s}, I, O, \varphi, \psi\}$, where $S$ is a set of states of an automaton, $\hat{s} \in S$ is an initial state, $I$ is a finite set of input actions, $O$ is a finite set of output reactions, $\varphi: S \times I \rightarrow S$ is a transition function, $\psi: S \rightarrow O$ is function of the outputs, is a Moore machine. If at some moment in time the Moore machine $(S, \hat{s}, I, O, \varphi, \psi)$ is in a certain state $s \in S$ and a signal arrives at the input of the automaton $i \in I$, then the automaton goes into the next state $s' = \varphi(s, i)$, and a signal appears at its output $o = \psi(s')$.

Let be $Z$ a set of integers and $(i, j) \in Z \times Z$ is a set of all possible pairs of integers. For each pair $(i, j)$ there is a finite set of pairs $N_{i,j} \subseteq Z \times Z$ of integers. A set $N_{i,j}$ called the neighborhood of the pair $(i, j)$. Each Moore machine of this set is a cell. The set of states of all cells of the cellular automaton at a given moment in time is called the global state of the cellular automaton at this moment in time.

One of the ways to represent cellular automata is graphical on a plane. In this case, the shape of the cells can be square, triangular, hexagonal, etc. In our study, we use cellular automata with square cells (Figure 1).
3. CA-model of diffusion

Diffusion is a process of random wandering of particles, which leads to equalization of the concentration of matter in area [10].

There are several CA models of diffusion. One of the CA models of diffusion is naive diffusion [10]. This is the most primitive model of diffusion, which directly reflects the idea of the process as a wandering of particles in an effort to equalize the concentration of matter in area.

![Diagram of naive diffusion CA](image1.png)

**Figure 1.** A method for illustrating a CA on a plane by equal squares.

The mode of operation of the CA is asynchronous, which fully corresponds to the nature of the process. The neighborhood of a cell is its closest four neighbors. The rule of operation is such that at each iteration a random cell is selected, which changes its value with equal probability with one of its neighbors. With this rule, the fulfillment of the law of conservation of masses is seen, and a random choice of one of the neighbors corresponds to a random walk of particles in accordance with the definition of the diffusion process [10].

The second known cellular automaton model of diffusion is a cellular automaton with a neighborhood of Margolus [11]. In this model, the cell array is divided into two subsets, each of these subsets consists of blocks containing four cells. The CA operates in a two-stroke synchronous mode.

Each iteration is divided into two ticks. On even clock cycles, the transition rules are applied to even blocks, on odd ones - to odd ones. The transition rules are such that the shift of states in the cells of the block is performed equally likely clockwise or counterclockwise [10].

![Diagram of Margolus neighborhood](image2.png)

**Figure 2.** Method for illustrating CA with naive diffusion.
In our work, the construction of a software implementation and adaptation of the tensor approach will be carried out specifically for this model, since there is a rigorous mathematical proof that the KA-model of diffusion with the Margolus neighborhood approximates the Laplace operator [12].

4. Tensor approach
We use the TensorFlow framework [13] as a special working environment that allows us to effectively build software implementations. The TensorFlow framework was chosen for the following reasons:

- open source code of this framework,
- native support for GPUs with CUDA cores by this framework.

Open source means that the framework is distributed under a free software license (GPL compatible). The purpose of using such a license is to provide the user with the rights to copy, modify, and distribute (including on a commercial basis) the programs, and to ensure that users of all derivative programs will also receive the above rights. It follows that for the development of free software, each user can make changes. Also, for any of your specific tasks, you can completely safely make changes. This is due to the fact that most free software is distributed as source code.

The TensorFlow framework provides the ability to parallelize data for computation. As a multi-threaded processor, the graphics adapter is the best choice. Among the presented video processors on the market, video adapters from NVIDIA with RT and CUDA cores are preferable due to their orientation towards use in projects aimed at parallel computing. TensorFlow has native graphics support.

The tensor approach to the software implementation of the CA proposed in [8] is as follows. The main data structure of the TensorFlow framework is a tensor, that is, a multidimensional matrix. Consequently, the set of states of an ensemble of cells of an automaton is described using a tensor, and the process of evolution of an automaton is described using operations on tensors.

To adapt the tensor approach to the software implementation of the CA model of diffusion, it is necessary to implement the block rotation operation using operations on tensors. Rotation of a block can be represented as rows’ rearranging and transposition.

Rearranging of rows in TensorFlow is simple. For example, each block of automaton is a tensor of rank two:

\[
A = \begin{bmatrix}
a_{11} & a_{12} \\
a_{21} & a_{22}
\end{bmatrix}.
\]

To rearrange the rows of this tensor, one should multiply tensor \(A\) (as a matrix) by the following tensor \(B\) of rank 2:
\[
B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},
\]
\[
B \times A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} a_{21} & a_{22} \\ a_{11} & a_{12} \end{bmatrix} = A'.
\]

(1)

To implement the transposition, we suggest using convolution:
\[
C \cdot A' = C \cdot \begin{bmatrix} a_{21} & a_{22} \\ a_{11} & a_{12} \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix},
\]
\[
(2)
\]

where \(d_{11} = a_{21}, d_{12} = a_{11}, d_{21} = a_{22}, d_{22} = a_{12} \).

As the result of this convolution, a tensor of the same rank as at the input should be obtained. This imposes the requirement to use the convolution kernel \(C\) of rank 4. To determine the coefficients of the kernel \(C\), a system of equations of the following form is drawn up:
\[
\begin{align*}
d_{11} &= c_{11}^{11} \cdot a_{11} + c_{11}^{12} \cdot a_{12} + c_{11}^{21} \cdot a_{21} + c_{11}^{22} \cdot a_{22} \\
d_{12} &= c_{12}^{11} \cdot a_{11} + c_{12}^{12} \cdot a_{12} + c_{12}^{21} \cdot a_{21} + c_{12}^{22} \cdot a_{22} \\
d_{21} &= c_{21}^{11} \cdot a_{11} + c_{21}^{12} \cdot a_{12} + c_{21}^{21} \cdot a_{21} + c_{21}^{22} \cdot a_{22} \\
d_{22} &= c_{22}^{11} \cdot a_{11} + c_{22}^{12} \cdot a_{12} + c_{22}^{21} \cdot a_{21} + c_{22}^{22} \cdot a_{22}
\end{align*}
\]
\[
(3)
\]

One of the possible solutions of this system can be obtained when in each equation one component is equal to 1, and the rest – 0. According to (2) the convolution kernel \(C\) has the following form:
\[
\begin{align*}
c_{11}^{11} &= 0 & c_{11}^{12} &= 0 & c_{11}^{21} &= 1 & c_{11}^{22} &= 0 \\
c_{12}^{11} &= 1 & c_{12}^{12} &= 0 & c_{12}^{21} &= 0 & c_{12}^{22} &= 0 \\
c_{21}^{11} &= 0 & c_{21}^{12} &= 0 & c_{21}^{21} &= 0 & c_{21}^{22} &= 1 \\
c_{22}^{11} &= 0 & c_{22}^{12} &= 1 & c_{22}^{21} &= 0 & c_{22}^{22} &= 0.
\end{align*}
\]

It should be noted that since all the above operations are linear and act one after another, they could be combined into one.

Thus, the block is rotated clockwise. Counterclockwise rotation is defined similarly.

5. Conclusion

In this paper, a tensor approach to the software implementation of the cellular automaton model of diffusion is considered. The approach is focused on running programs on multi-core video cards. In the future, it is planned to programmatically implement the proposed approach and conduct computer experiments to assess the adequacy of the approach.

Acknowledgments

The reported study was funded by RFBR according to the research project № 20-31-90070.

References

[1] Bandman O L 2006 Kletochno-avtomatnye modeli prostranstvennoj dinamiki Systems informatics vol 10 pp 57-113

[2] Toffoli T 1984 Cellular automata as an alternative to (rather than approximation) to differential equations in modeling physics Physica D p 117

[3] Bandman O L and Kireeva A E 2015 Stochastic Cellular Automata Simulation of Oscillations
and Autowaves in Reaction-Diffusion Systems *Numerical Analysis and Applications* vol 8 No 3 pp 208–222

[4] Kireeva A E and Sabelfeld K K 2015 Cellular Automata Model of Electrons and Holes Annihilation in an Inhomogeneous Semiconductor *Parallel Computing Technologies, 13th International Conference, PaCT* vol 9251 pp 191-200

[5] Ochoa Bique, A O Goryunov and A G Manenti F 2015 A comparison of simulation techniques for uranium crystallization process *Chemical engineering transactions* vol 43 pp 793-798

[6] Jiménez-Morales F, Guisado J L and Guerr J M 2018 Simulating Laser Dynamics with Cellular Automata *Understanding Complex Systems* Part F3 pp 405-22

[7] Shimura K, Vizzari G, Bandini S and Nishinari K 2013 Analysis and application of the pedestrian permeation through the crowd via cellular automata *PaCT* vol 7979 pp 369–380

[8] Shalyapina N A and Gromov M L 2019 «Life» in Tensor: Implementing Cellular Automata on Graphics Adapters *Trudy ISP RAN/Proc. ISP RAS* vol 31 issue 3 pp 217-228

[9] E.F. Moore 1956 Gedanken-experiments on sequential machines *Automata studies* No 34 pp 129–153

[10] Evseev A A and Nechaeva O I 2009 Kletochno-avtomatnoe modelirovanie diffuzionnyh processov na trianguljacionnyh setkah *Prikladnaya Diskretnaya Matematika* No 4 pp 72-83

[11] Toffoli T and Margolis N 1987 *Cellular automata machines: a new environment for modeling* (MIT Press, Cambridge, Massachusetts)

[12] Malinetsky G G and Stepanstov M E 1998 The Diffusion Process Simulation By Cellular Automata With The Margolus Vicinity *Computational Mathematics and Mathematical Physics* vol 36 No 6 pp 1017-1021

[13] *TensorFlow* Retrieved from: tensorflow.org (accessed: 23/07/2020)