Modeling a Symbolic Image of a Dynamic System

V. A. Lukianenko[1] **M. G. Kozlova.**[2]

Keywords: symbolic image, dynamical systems, trajectories of the systems, path on the graph.

MSC2010 codes: 37A05, 37B10

Introduction. In the study of dynamic systems, methods based on the construction of a symbolic image, which is a directed graph obtained with aid, are applied final phase space cover [1, 2, 7].

The trajectories of the system are encoded by paths on the graph [1, 6] and can be found using route algorithms in the graph by many agents, and closed paths by many salesmen. The structure of the graph changes, and vertices and edge arise as a result of algorithmic procedures of numerical resolution of the corresponding dynamic systems. There are a number of problems associated with numerical implementation of symbolic image algorithms on large-dimensional graphs [3-5].

1. Symbolic image of a dynamical system. Let $f : M \longrightarrow M$ be a homeomorphism of compact manifold M generating a discrete dynamical system and $p(x, y)$ by a distance on M. Let $C = \{M(1), \ldots, M(n)\}$ be a finite closed covering of manifold M. The set $M(i)$ is called cell with index i.

Definition 1 [1, 7]. Symbolic image of the dynamical system $x_{n+1} = f(x_n)$ for covering C is a directed graph G with vertices $\{i\}$ corresponding to cells $\{M(i)\}$. The vertices i and j are connected by the edge $i \longrightarrow j$ if $f(M(i)) \bigcap M(j) \neq 0$.

Symbolic image is a tool for a space discretization and graphic representation about the global structure of the system dynamics. Symbolic image depends on a covering C. In other words, an edge $i \longrightarrow j$ is the trace of the mapping $x \longrightarrow f(x)$, where $x \in M(i)$, $f(x) \in M(j)$. If there isn't an edge $i \longrightarrow j$ on G then there are not the points $x \in M(i)$ such that $f(x) \in M(j)$.

Definition 2 [1, 7]. A vertex of a symbolic image G is said to be recurrent if there is a periodic path passing through it. The set of recurrent vertices is denoted by RV. The recurrent vertices i and j are called equivalent if there exists a periodic path passing through i and j.

Thus, the set of recurrent vertices RV is split into equivalence classes $\{H_k\}$. In the graph theory such classes are called strong connectivity component.

2. Program implementation of the symbolic image construction algorithm. Consider the following algorithm for constructing a symbolic image.

Algorithm for the construction of a symbolic image

1. For the initial set M_0, in which the values of a discrete dynamical system lie, we construct a cell cover C. In this particular case, the cells have the shape of a square with a predetermined edge size d_0 (cells can have either an arbitrary shape or an arbitrary size).

2. To cover C, we construct a graph G, such that it is a symbolic image.

3. Using one of the algorithms for finding strongly connected vertices, for example, Tarjan's algorithms [3, 4] or the algorithm for finding strongly connected vertices based on the path search [5], we allocate strongly connected vertices $\{i_k\}$. If $\{i_k\} = \emptyset$, then there is no attractor in M_0 and the localizeable chain-recurrent set is empty and the process of its localization stops [2]. Otherwise, we remove the irrevocable vertices from the graph and move from the M_0 region to the new M_1 region, such that $M_1 = \{x \in M_0^{i_k} : i_k \in G\}$.

[1]V. I. Vernadsky Crimean Federal University, Taurida academy, Russian Federation, Simferopol. Email: art-inf@yandex.ru

[2]V. I. Vernadsky Crimean Federal University, Taurida academy, Russian Federation, Simferopol. Email: art-inf@mail.ru

4. We build for M_1 a coverage of C_1, so that the edge size of the new cell $d_1 = \frac{d_0}{2}$. Go to step 2 if $d_1 > \varepsilon$, where ε is a predefined limit cell size.

The problem of software implementation of the algorithm for constructing a symbolic image is based on the theory proposed by G. S. Osipenko [1, 2]. For the implementation of this project, the Python programming language was chosen, which is justified by a wide choice of libraries that are convenient for working with graphs and other mathematical structures, its stability and portability. This language is well suited for the implementation of the task. The OpenGL graphics environment is selected to display the graphics and render the images needed to render the symbolic image. OpenGL (Open Graphics Library) – a specification that defines a platform-independent programming interface for writing applications that use two-dimensional and three-dimensional computer graphics. The OpenGL GLUT library is used. OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL applications that is mainly responsible for the system level of I/o operations when working with the operating system. The following functions are used: window creation, window management, monitoring of keyboard input and mouse events. The NetworkX library created specifically for the selected Python programming language is connected. The NetworkX package is designed to create and analyze the study of structures, dynamics and functions of complex networks, and therefore directed graphs. The library includes special data structures for graphs, both directed and non-directed, a lot of standard algorithms on graphs, generators of classical and random graphs, flexible work with vertices and edges, which can be almost any object and have any parameters and open code, which allows you to freely use any algorithms from the library.

To work with arrays of data, the NumPy library was connected. This library provides ample opportunities for working with two-dimensional arrays: a lot of convenient methods for working with arrays, the ability to create large arrays in one line, and more.

Was also used by SciPy. The part relating to differential equations was imported from it. The SciPy package allows you to solve both simple systems and systems of differential equations with parameters. Anaconda was chosen as the development environment. The convenience of working with Anaconda is that most of the widely used Python libraries are already installed in distributions, which greatly simplifies their connection.

As test cases built display Hanona, Zaslavsky, attractors of Peter de Jon etc.

References

[1] G. S. Osipenko, N. B. Ampilova. Introduction to symbolic analysis of dynamical systems. Ed. St. Petersburg state University. 2005.

[2] E. I. Petrenko. Development and implementation of algorithms for constructing a symbolic image // Electronic journal differential equations and control processes. 2006. No. 3. P. 42.

[3] R. E. Tarjan. Depth-first search and linear graph algorithms // SIAM Journal on Computing. 1972. V. 1, No. 2. P. 146-160.

[4] R. Sedgwick. Algorithms on graphs. 3rd ed. Russia, St. Petersburg: "Diasoftyup", 2002.

[5] R. Sedgwick. Algorithms in Java, Part 5 – Graph Algorithms. 3rd ed. Cambridge, MA: Addison-Wesley.

[6] D. Lind, B. Marcus. An introduction to symbolic dynamics and coding. Cambridge: Cambridge University Press, 1995.

[7] G. S. Osipenko. Dynamical systems, Graphs and Algorithms. Lectures Notes in Mathematics, 1889, Berlin: Springer, 2007.