Web-Based Parameter Space Explorer for Non-Invertible 3D Maps

Vladimír Roth*

VRVis Research Center
Vienna / Austria
&
Faculty of Mathematics, Physics and Informatics
Comenius University, Bratislava / Slovakia

Abstract

This paper describes the process of interactive dynamical system investigation. We are working with multi-dimensional econometric dynamical systems. Various types of visualization are presented, namely the 3D and 2D visualization of the parameter space and also the 3D and 2D visualization of the phase space (see Figure 1). Efficient linking of these visualizations is described. Other related topics are also covered, like automatic matching of the objects in the phase space and distributed computation of the difference equations which describe the dynamical system. Our solutions are implemented in the application written in Java. The application is also providing a bifurcation diagram, visualized as background coloring, based on the investigation of the parameter space.

Keywords: Dynamical system visualization, Parameter space, Phase space, Attractor, Basin, Bifurcation, InfoVis

1 Introduction

With the development of computers a new topic came to attention, namely interactive investigation and visualization of dynamical systems (also via the network). Visualization helps us to see the relations and the properties of the data better and can be crucial while understanding the relations.

A solution for interactive dynamical system investigation is described in this paper. Firstly we introduce some terms and definitions which will be used in the paper. Most of the work was spent on the visualization which is described in the Chapters 4 and 5. In chapter 5 we introduce Star glyph which is used to visualize positions and characteristics of phase spaces in the parameter space. The phase space visualization is described in the Chapter 4. In the Chapter 3 we aim at the dynamical system exploration and the basic concepts like the topology of the system and previous work. Additional parameters of the system like linking and modularity are described in Chapter 6. At the end in Chapter 7 we provide concrete example of econometric dynamical system investigation.

Our system is implemented in Java using RMI (remote method invocation) for network communication, Java3D for 3D visualization and a Java library called RTVR [11] for volumetric data visualization. The system is fully interactive and asynchronous so many tasks can run together. It is also platform independent thanks to the use of Java.

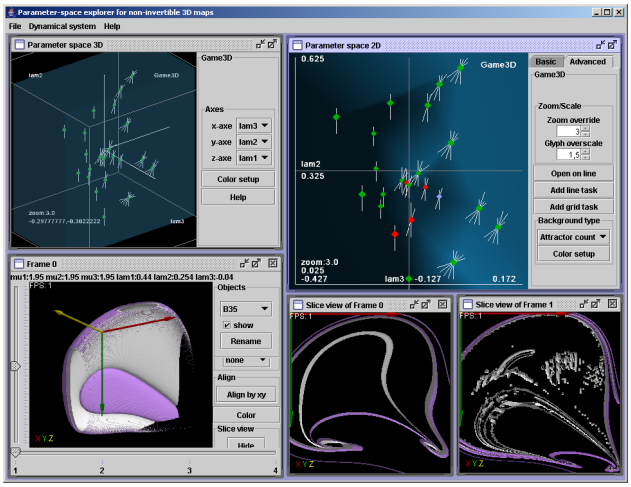

Figure 1: Main application overview with all types of visualization shown. Namely the 3D and 2D parameter space visualization and the 3D and 2D phase space visualization.

2 Dynamical systems

Dynamical systems describe the evolution of some entities within a common space. One example of dynamical system is a food chain, describing the who-eats-whom relation between several species, living in the same place. A 2D predator-prey model by Lotka and Volttera is an example of such a food chain. This model describes the evolution of dynamical system consisting of two species, predator and prey. It consists of two values representing

*vlado@roth.sk, www.roth.sk

the amount of predators and preys present in the system at a certain time, and a description of the change of these values due to the given setting of the system [8].

A *dynamical system* is specified by equations and set of parameters [2]. There are two types of dynamical systems, *continuous* and *discrete* systems. Continuous dynamical systems (also called *flows*) are defined by differential equations and the discrete ones (also called *maps*) are defined by difference equations [15]. Difference equations (e.g., $\dot{\mathbf{x}} = A \cdot \mathbf{x}$) are the discrete analog of differential equations (e.g., $\Delta \mathbf{x}_n = (I + A) \cdot \mathbf{x}_n$).

2.1 Discrete systems

A *discrete dynamical system* is given by difference equations which form a map

$$\mathbf{f'_p} : \mathbb{R}^n \rightarrow \mathbb{R}^n \tag{1}$$

which transfers a point (state of the system) $\mathbf{x} \subseteq \mathbb{R}^n$ into another point $\mathbf{x'} \subseteq \mathbb{R}^n$ which is uniquely defined by the equation

$$\mathbf{x'} = \mathbf{f'_p}(\mathbf{x}) \tag{2}$$

together with a set of parameters $\mathbf{p}$. The state $\mathbf{x} = (x_1, x_2, ..., x_n)^\top \in \mathbb{R}^n$ can be seen as a point of an n-dimensional Euclidean space. The components $x_1, x_2, ..., x_n$ of the state $\mathbf{x}$ are called *state variables* of the dynamical system. One state variable therefore is an actual value of one of the differential equations which specify the dynamical system.

2.2 Phase space and Parameter space

A *phase space* is defined by associating each of the n state variables to one axis of an n-dimensional Cartesian coordinate system.

The parameters $\mathbf{p}$ of the dynamical system build up the *parameter space*. Different sets of values for parameters define different phase spaces (See Figure 2).

2.3 Attractor and basin

An *attractor* A is a kind of steady state in a dynamical system. It can be described as a set of states (points in the phase space) of the dynamical system, which do not change trough the process of iteration of the map $\mathbf{f_p}$. An attractor is defined as the smallest unit which cannot be itself decomposed into two or more attractors with distinct basins of attraction. This restriction is necessary since a dynamical system may have multiple attractors, each with its own basin of attraction.

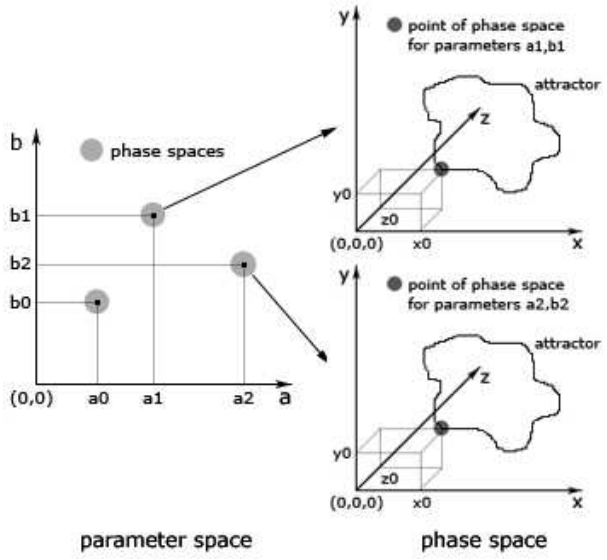

Figure 2: Example of a parameter and a phase space of a dynamical system with map $\mathbf{f_p} : (x, y, z) \rightarrow (a \cdot x + y, b + x^2, a \cdot z)$.

The *basin of attraction* (or simply $basin$) of an attractor A is the set of all point that generate trajectories converging to A

$$B(A) = \{\mathbf{x} \mid \mathbf{f}^n_{\mathbf{p}}(\mathbf{x}) \rightarrow A \Leftrightarrow n \rightarrow +\infty\} \tag{3}$$

Basin boundaries of dynamical systems can be either smooth or fractal. It appears that fractal basin boundaries are common in typical dynamical systems [10].

3 Dynamical system exploration

To explore a dynamical system we need to know the difference or differential equations which describe the system. Hence we have the equations we are just at the beginning of the exploration process. Then we can do computations with different sets of parameters to get data. These data contain objects (attractors and basins) which needs to be classified and named. We may need to establish some naming to navigate between the objects. The naming and the data must be stored effectively because they are huge ($\sim$GBs). After we have the data, we need to investigate it and try to find some relations between the computed objects. We also need to visualize the computed data and also the parameters and relations between them. The goal is to make conclusions about the system according to the computed data.

3.1 Complex systems

We are dealing with econometric dynamical systems provided by Gian-Italo Bischi[1]. These systems may have many parameters (up to 10 or 15), so we are working with multi-dimensional parameter space. Therefore it is essential to have a clear visualization to not get lost in the parameters, but still try to show as much information as possible. Also when investigating such systems we need to iterate the equations for many sets of parameters. This gives us a lot of data which needs to be managed and visualized. There is also need to have some linking between the parameter space and the according phase spaces, because to each set of the parameters in the parameter space belongs one phase space. Linking between the different visualizations (3D, 2D) of the parameter and phase spaces is also needed because the 2D visualizations are sections through the 3D spaces.

3.2 Distributed computation

When working with dynamical systems with multi-dimensional parameter spaces the difference equations used can be complex. We are working with volumetric data of the dimension 256^3. For every point of this 256^3 cube we need to iterate the difference equations several times ($\sim$1000 times). As they are complex this cannot be done in real-time, in some cases even not in a single day. Therefore we need distributed computation.

In our project we used Java RMI (remote method invocation) to implement the distributed computation part. The computational system consist of one master which is also the main data storage and computational clients called slaves (see Figure 3). Master is distributing the computational tasks to the slaves and collets the results. Than the naming is done and data are stored to files. The system is also robust and can fix corrupted tasks or assign them again.

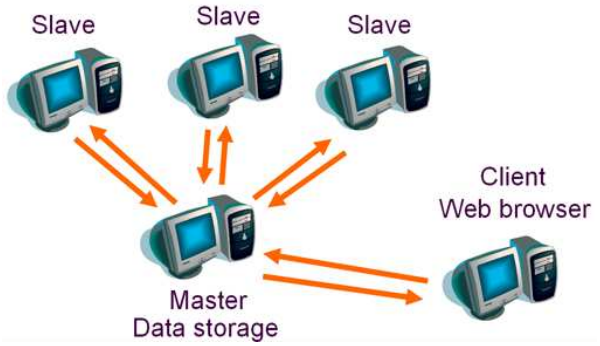

Figure 3: Topology of the computational system.

[1]Istituto di Scienze Economiche, Department of Economics University of Urbino, Italy

3.3 Previous work

Although a variety of software which can be used for dynamical system investigation is available, few of them are primarily focused on general dynamical system visualization. Our solution provides a system for the visualization of high-dimensional dynamical systems as described by difference equations, which is modular and does not depend on one concrete system. We are using it mainly for investigation of econometric dynamical systems as mentioned before.

There were programs to visualize dynamical systems in the past. One of the oldest programs is AUTO from 1986 [5]. It can do a limited bifurcation analysis of algebraic systems and of systems of ordinary differential equations subject to initial conditions, boundary conditions, and integral constraints. Later came AVS (The application visualization system) [16] and VTK (The visualization toolkit) [13] which were not primarily intended to be used for dynamical system visualization, but for general visualization. Another program named DsTool is focusing on the interactive dynamical systems investigation using Unix and the X window system. It has interactive graphical interface [3]. There has been little development of the programs since 1997. Therefore the graphical interface is outdated and does not support techniques like opacity, two-level volume rendering, etc. One of the youngest environments is DynSys3D from 1998 [9]. It is based on AVS and allows to easily combine different visualization techniques corresponding to the needs of the user.

Even though there were solutions and environments to investigate dynamical systems, none of them was providing both a parameter and a phase space visualization next to each other and these systems also were not general enough to handle sundry dynamical systems.

4 Phase space visualization

To visualize the phase space we need a slightly different approach than for the visualization of the parameter space because we have only a three-dimensional space because we are using dynamical systems defined by three difference equations (which forms 3D phase space). We decided also to use an additional 2D visualization of the 3D phase space. The 2D visualization is in fact a slice view on a plane through the 3D phase space. With this 3D projection and 2D section has the user full knowledge about the objects in the phase space. Each of them is described below. Every phase space has accurate position in the parameter space of the dynamical system which can be denoted by a point. This point is called node, so each node contains 3D phase space.

4.1 Naming

To distinguish between the objects in the different nodes (phase spaces) we need to build up some naming scheme. We have found that it is better to do the naming on the server side so the client is not loaded with too many tasks and can run at interactive frame-rates.

Naming is done immediately after a new node in the parameter space is computed. As mentioned before this is done on the server also because the server has all the data which is needed to compare the objects in the nodes. The server takes newly computed objects and compares it to the adjacent objects to decide whether they match. If they fulfill the criteria, then the same name is assigned to them. The criteria depend on the type of the objects. The objects are compared by the size, box dimension and position in phase space.

This simple naming scheme works fine in real use and can be tuned up by different comparing criterions. Also for different object types different sets of comparing criteria can be used.

4.2 3D projection

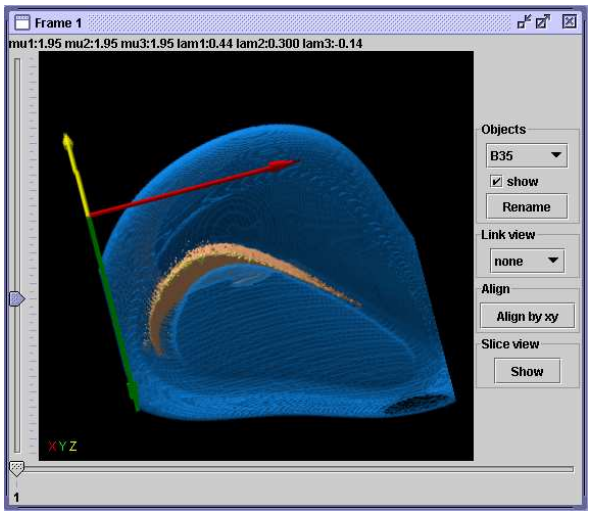

Figure 4: 3D phase space visualization. There are three basins in the image (blue, orange, and yellow). On the left is a slider to move 2D section. At the bottom is another slider to move between nodes opened in the actual window.

The main purpose of the 3D phase space visualization is to show the shapes of the objects (attractors, basins) in the node and their relations (naming). The user needs the possibility to rotate and zoom the view at interactive frame rates and a suited data representation should be used. We decided to use a voxel data representation rather than polygons. The choice was simply because voxel data representation can better handle points and in many cases the objects in the phase space consists of many disjoint points (like chaotic attractors [10]). With the use of a voxel data representation the voxel data visualization problem arises. For this purpose we have chosen two-level volume rendering [6] as a solution. This solution is fusing many techniques, maximum-intensity projection (MIP) [12] and direct volume rendering (DVR) [7], and also others like non-photorealistic rendering, phong shading, etc. It provides better results than the techniques used only alone.

In the visualization of phase space, the naming can be also used to bring order into the chaos. The possibility to open more nodes in one window and to navigate through them with a slider is highly suggested. This adds another dimension to the 3D visualization.

Our implementation meets all the requirements mentioned before. For the visualization of the voxel data RTVR [11] is used. RTVR is a flexible Java library for interactive volume rendering. It offers real-time rendering of volume data. The RTVR library uses a fast shear-warp projection achieving interactive frame-rates also on low-end hardware. It also uses the two-level volume rendering as described above.

The rotation and zooming is done with the mouse while pressing the right button or right button and Shift key. Actual parameters as used for the computation of the node are shown at the top of the window (see Figure 4). On the right side there are controls to select the objects and a checkbox to show or hide them. The user can also interactively change the color and transparency of the objects by holding the control keys and moving the mouse.

If the user opens more than one node in the same window he (or she) can navigate through the opened nodes by the slider which is situated at the bottom of the window. Only one node is visible in the window at the moment so the other ones are temporary invisible. Also when rotating the actual 3D view the other invisible nodes which are opened in the actual window are rotated so the system is consistent. When the objects in different nodes are the same (this is decided by the naming) and the user can change the color in the actual view the color changes also in the temporary invisible nodes in the actual window.

Another possibility is to open each node in a single window and link these windows together. So the transformations (rotation, zooming) and the color are changing in all linked windows simultaneously. User can link just some windows, when needed.

There is another slider on the left side of the window which controls the 2D phase space visualization (slice view). The user has the possibility to move the 2D projection plane by adjusting the slider. The 2D section view is described in the next section.

4.3 2D projection

The 2D phase space visualization is actually a slice view through the objects in the 3D phase space. It is important to show the alignment of the view as compared to the

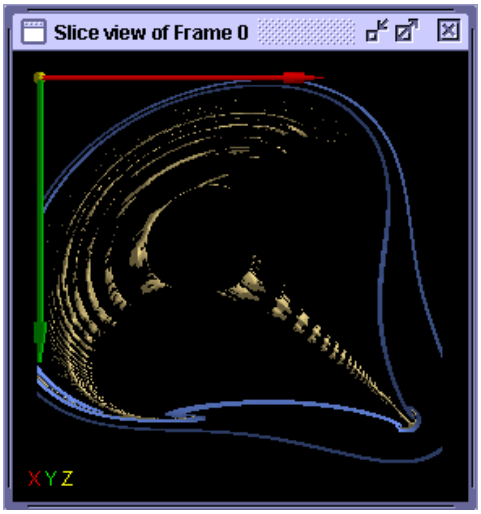

Figure 5: 2D phase space visualization. The section through two basins (blue and yellow) can be seen in the image. The caption of the window shows the correspondence to the 3D phase space window.

3D view. Also coloring is useful to differ between the object on the slice. Every 2D phase space visualization is strictly connected to some 3D phase space so it needs some indication to which window it belongs.

Our 2D phase space visualization can be seen in Figure 5. At the left upper corner two axes are shown of the 3D space which are used. The coloring is used also from the 3D visualization and changes in real-time. The correspondence to the 3D phase space window (called frame) is in the caption of the window.

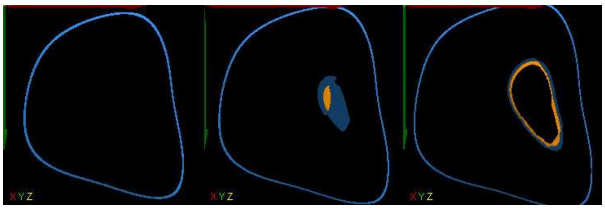

Figure 6: Tracing of objects with 2D phase space visualization. There are three states of 2D phase space visualization after adjusting the slider which controls the position of the 2D section in the 3D phase space. There are two basins on the image, blue and orange.

The main purpose of this visualization is to trace the objects which are inner structures (inside other objects). This can be successfully done as shown in Figure 6. There are two basins in the image, one blue and one orange. The orange is the inner one. The figure show 2D phase space visualizations after adjusting the slider in the 3D phase space window which controls the position of the 2D plane.

5 Parameter space visualization

As mentioned before, we are working with a multi-dimensional parameter space. We live in a three-dimensional world so we can visualize just in three dimensions. We do not know how to draw four-dimensional or even more-dimensional system. So what to do when we have ten or fifteen dimensions? We have decided to visualize the multi-dimensional parameter space as 3D projection and 2D section of the n-dimensional space. This is sufficient for our needs, because through the investigation of the dynamical system we change just few parameters, not all of them. Some parameters are more interesting than others. Also the 3D visualization provides knowledge of the position of the 2D section within the 3D projection. A description of the 3D and 2D projection is below. To represent the positions of the phase spaces for different sets of parameters in the parameter space we need suitable glyph. We have found that the star glyph design is capable to show the main characteristics of the phase space, like count and type of the objects (attractors and basins) in the node. It is described in detail below.

5.1 Star glyph design

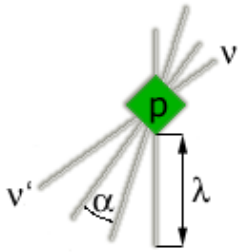

Figure 7: Star glyph design (**p** - position in the parameter space, ν - attractors, ν' - basins, λ - length of the ray, α - angle between rays).

The star glyph [14] was designed to represent the nodes (phase spaces) in the parameter space visualization. The final design is shown in Figure 7. In the middle there is a small rectangle which is situated at the accurate position of the node in the parameter space (denoted by **p**). Its color depends on the state of the node (see Figure 8): when computed and ready for opening, the node is green. When the node was assigned for computation but, still the computation did not finish, the color of the node is yellow. After opening of the node, the color of the glyph changes to red. When several nodes are opened, the actual one on which the user is looking, is colored purple. The colors change interactively so user knows exactly which node is opened and active directly just from the coloring.

The rays (denoted by ν) of the glyph represent the objects in the node. When investigating dynamical systems we encounter objects like attractors, basins, and critical

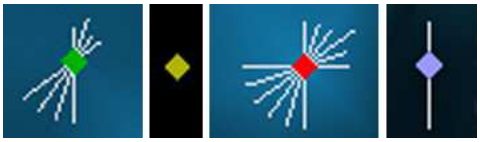

Figure 8: Star glyph implementation. There are four glyphs on the image, each with different colour and different meaning (green-computed, yellow-not yet computed, red-opened, purple-opened and shown in the actually focused window).

surfaces in the phase space of the system [2]. Each ray represents one of these objects. Rays in the upper right corner represent attractors and the ones opposite to them represent basins of attraction, related to the attractors. The length (denoted by λ) of a ray depends on the type of the object in the case of attractors and on the box dimension of the object in the case of basins. The angle (denoted by α) between the rays is fixed and based on a maximal count estimation. It can be changed to meet the needs of the actual dynamical system. For our systems the estimation is maximally six objects for each type.

5.2 2D projection

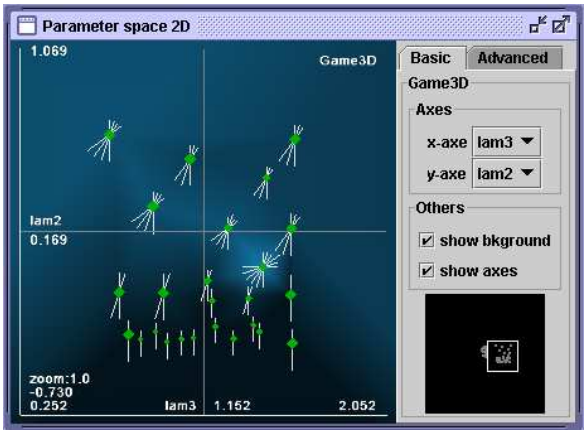

Figure 9: 2D parameter space visualization. We can see 25 nodes (phase spaces) visualized by star glyphs in the image. The background coloring interpolates between the count of objects in the nodes. Actual parameters of the axes are also shown.

The visualization as a 2D projection of the parameter space is more challenging than the 3D case. The main purpose is to show as much information about the nodes of the parameter space as possible without the need to directly open them. This can by done by a careful glyph design which represent the nodes and by additional background visualization. There is also the need to navigate through the space and zoom. This can be done as in the 3D case simply just by mouse and control keys. Users must be able to change the parameters which are shown on the axes of the 2D parameter space. An additional overview showing the actual position and size of the main view of the parameter space is highly suggested so the user knows overall density and distribution of the nodes. The background of the parameter space can be colored depending on the properties of the nodes (phase spaces). We are using the triangulation between the nodes and then interpolate color between the vertices of the triangles depending on some property of the node, like a count of the attractors.

This visualization is also very suitable for assigning new computational tasks. Computational task is a request for the master to compute (iterate) the difference equations of the system for given set of parameters. By clicking on a point in the parameter space the user can simply select the place of his (or her) interest and the data for the phase space can be computed (new computational task is assigned to the master). When we want to view the node (phase space) of the parameter space we need to open the 3D phase space visualization window. This can be solved in the same way by as assigning of new computational tasks by clicking on the nodes.

Our implementation, also using Java3D [1] like in the 3D case, can be seen in Figure 9. For background visualization the triangulation between the nodes is used. This is computed in the real-time from the actual visible nodes. For the node visualization the Star glyph design is used which was described widely in the previous section. As in the 3D case there are axes and actual parameters on the screen so the user knows the exact position of the nodes. To fulfill this task a small overview is situated on the side of the main window, showing the spread and density of the nodes and the actual position of the main viewing window. There is the possibility to change the type of the background visualization and also the color. The type depends on the contents of the nodes.

In some parts of the parameter space the distribution of the nodes can be dense, so it is hard to see the glyphs which represent the nodes. To solve this problem we divided the parameter space into parts and assigned different glyph-zoom factors for each part, depending on the density of the nodes in the part. This is done in real-time, so when the user changes his (or her) place of interest to a place with a small density of nodes, the nodes become interactively larger and vice versa.

Assigning of new nodes and opening of the existing ones is done by clicking on them with mouse. This is simple enough to not overload the user with fruitless activity. After clicking on the node there is the possibility to open it in new window or in the existing one. When clicking on the empty space a table is shown which provides the possibility to tune the parameters which will be used for new computation.

5.3 3D projection

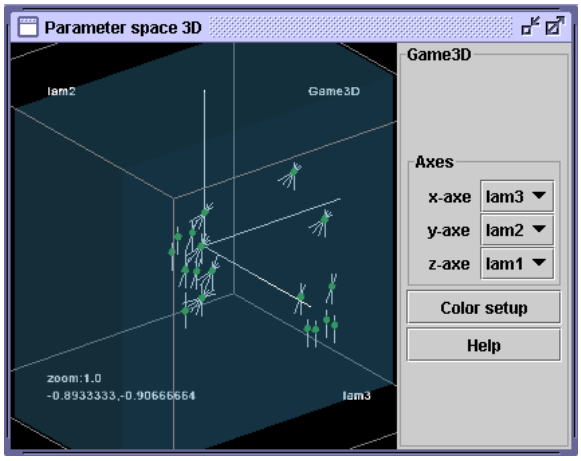

Figure 10: 3D parameter space visualization. There are Star glyph nodes in the 3D parameter space projection in the image. Actual parameters and 2D section position is shown.

The main purpose of the 3D projection is to show arrangements of the nodes (phase spaces) in the 3D space to get a better insight into the spatial distribution of the nodes. So there is need to show a 3D space and it is important to have the ability to rotate, pan, and zoom in this space. The presence of actual parameters of the nodes is also important. There must be also the possibility to select which parameters are mapped to the axes because we are working with a high-dimensional dynamical system but 3D parameter space is able to show just three of them at a time.

For implementation of this projection Java3D [1] was used. A sample result can be seen in the Figure 10. Rotation, panning and zooming can be done done by the mouse and the control keys so user can simply navigate through the space and is not confused by complex controls. The orientation of the space is represented by three axes. Actual parameters of the intersection of the axes are shown. Switching of the parameter-to-axis mapping is done by selecting the parameter for each axis in the menu (on the right in the image). We have added also a color setup so the user can select the color of the background and other elements in the view.

6 Linking and Modularity

All the visualizations of the parameter and the phase spaces are useful and satisfy the needs of interactive investigation of dynamical systems. But they are much more useful when linked together by some relations. Namely when the user changes the parameter on the axis in the 3D parameter space visualization window the parameter should also change in the 2D parameter space visualization and the same with naming and coloring of objects in the phase space visualization.

Linking is also implemented in our system and wages in real-time. So the system is even more interactive because all changes done by the user are reflected immediately in other windows if needed.

The 2D parameter space visualization is connected with the 3D phase space visualization by the relation of coloring the star glyphs, depending on the state of the node. This was described before.

Also the 3D phase space visualizations windows can be linked together and each of them can have 2D phase space visualization window linked with it.

All the parts of the system will be useless when focused only on one concrete dynamical system. Therefore there is need that the system is modular enough to easily implement new dynamical system.

Our implementation meets this condition. All classes are modular and do not use fixed numbers for count of parameters or objects. Also they do not use fixed names for dynamical systems and objects in them. Everything is dynamically loaded at the runtime depending just on one class for each dynamical system. This class contains count of parameters, names of parameters and other concrete values.

7 Concrete dynamical system

Images used in this paper were created by using real econometric dynamical systems. One of them is called triopoly game (Game3D). It is three-dimensional discrete dynamical system with six-dimensional parameter space, which simulates the time evolution of an oligopoly game with three competing firms [4]. The time evolution is obtained by the iteration of the three-dimensional map T:

$$\begin{aligned} q_1' &= (1-\lambda_1)q_1 + \lambda_1\mu_1[q_2(1-q_2) + q_3(1-q_3)] \\ q_2' &= (1-\lambda_2)q_2 + \lambda_2\mu_2[q_3(1-q_3) + q_1(1-q_1)] \\ q_3' &= (1-\lambda_3)q_3 + \lambda_3\mu_3[q_1(1-q_1) + q_2(1-q_2)] \end{aligned} \quad (4)$$

where $q_i, i = 1, 2, 3$, represents the productions at time t of the competing firms which sell the same good in a given market, and q_i' the respective productions at time $t+1$.

One of the tasks which can be solved by interactive investigation is detection of the global bifurcations that cause the creation of non-connected basins.

A part of its parameter space can be seen in Figure 9 and Figure 10. Phase space for parameters $\mu_1 = \mu_2 = \mu_3 = 1.95, \lambda_1 = 0.44, \lambda_2 = 0.3, \lambda_3 = -0.14$ is shown in Figure 4. We can see that for these parameters the system has three basins so all three firms still figurate on the market. From the parameter space visualization (Figure 10) we can see that when increasing μ_2 the count of basins decrease. Deeper investigation of this system is described elsewhere [4].

8 Conclusions

During the work on this project I have found that it is possible to show very much properties of the data at a small space. This can be achieved by a careful selection of the type of the visualization and also the design of the parts of the visualization like star glyph in our case. Star glyph showed to be really suitable to visualize positions of the phase spaces in the parameter space, because it provides also wide knowledge about the properties of the objects in the node (phase space) without directly visualizing it in 3D. The background coloring of the parameter space also greatly improves ability to see the distribution of some characteristic (like count of attractors) of the nodes in the parameter space.

9 Acknowledgements

This work was done as part of the basic research on information visualization at the VRVis Research Center in Vienna, which is funded by an Austrian research program called K plus. The author would like to thank Lukas Mroz[2] and Helwig Hauser[3] for supervising the whole work. Thanks goes also to Gian-Italo Bischi[4] who is the end user of the application and provides all the dynamical systems information. Special thanks go to Andrej Ferko[5] for the advices given while guiding the diploma seminar on our school.

References

[1] Java3D : The API for 3D graphics for Java. URL: http://java.sun.com/products/java-media/3D/.

[2] D. K. Arrowsmith and C. M. Place. *An Introduction to Dynamical Systems*. Cambridge University Press, 1990.

[3] A. Back, J. Guckenheimer, M. Myers, and P. Worfolk. dstool: Dynamical systems toolkit with interactive graphic interface user's manual. User's manual, Cornell University, 1992.

[4] G.-I. Bischi, L. Mroz, and H. Hauser. Studying basin bifurcations in nonlinear triopoly games by using 3D visualization. *Nonlinear Analysis*, 47/8:5325–5341, 2001.

[5] E. J. Doedel and J. P. Kernévez. AUTO: Software for continuation and bifurcation problems in ordinary differential equations. Applied mathematics report, California Institute of Technology, Pasadena CA 91125, 1986.

[6] H. Hauser, L. Mroz, G.-I. Bischi, and E. Gröller. Two-level volume rendering - fusing MIP and DVR. In *Proceedings of IEEE Visualization 2000 (Vis 2000)*, pages 211–218, October.

[7] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, May 1988.

[8] H. Löffelmann. *Visualizing Local Properties and Characteristic Structures of Dynamical Systems*. PhD thesis, Vienna University of Technology, Austria, 1998.

[9] H. Löffelmann and E. Gröller. DynSys3D: A workbench for developing advanced visualization techniques in the field of three-dimensional dynamical systems. In *Proceedings of The Fifth International Conference in Central Europe on Computer Graphics and Visualization '97 (WSCG '97)*, pages 301–310, 1997.

[10] S. W. McDonald, C. Grebogi, E. Ott, and J. A. Yorke. Fractal basin boundaries. *Physica D*, 17:125–153, 1985.

[11] L. Mroz and H. Hauser. RTVR - a flexible Java library for interactive volume rendering. In *Proceedings of IEEE Visualization 2001*, pages 279–286, 2001.

[12] G. Sakas, M. Grimm, and A. Savopoulos. Optimized maximum intensity projection (MIP). In *Rendering Techniques '95*, Eurographics, pages 51–63. Springer-Verlag Wien New York, 1995.

[13] W. J. Schroeder, K. M. Martin, and W. E. Lorensen. The design and implementation of an object-oriented toolkit for 3D graphics and visualization. In *Proceedings of IEEE Visualization '96*, pages 93–100, 1996.

[14] J. H. Siegel, J. E. Farrell, R. M. Goldwyn, and H. P. Friedman. The surgical implication of physiologic patterns in myocardial infarction shock. *Surgery*, 72:126–141, 1972.

[15] A. A. Tsonis. *Chaos - from theory to applicatons*. Plenum Press, New York and London, 1992.

[16] C. Upson, T. A. Faulhaber, D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, and R. Gurwitz. The Application Visualization System: a computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, 9(4):30–42, July 1989.

[2] Tiani Medgraph, Austria

[3] VRVis Research Center, Austria

[4] Istituto di Scienze Economiche, Department of Economics University of Urbino, Italy

[5] Faculty of Computer Graphics and Image Processing, Department of Comenius University, Slovakia