PHYSOR 2004 -The Physics of Fuel Cycles and Advanced Nuclear Systems: Global Developments
Chicago, Illinois, April 25-29, 2004, on CD-ROM, American Nuclear Society, Lagrange Park, IL. (2004)

AGENT Code: Open-Architecture Analysis and Configuration of Research Reactors – *Graphic Tools*

Dimitrios Andritsos[1] and Tatjana Jevremovic[2]
[1]*School of Industrial Engineering, Purdue University, West Lafayette, IN 47907 – 2023, USA*
[2]*School of Nuclear Engineering, Purdue University, West Lafayette, IN 47907 – 1290, USA*

Two papers presented in this Conference focuses on our recent development of the advanced computational environment for the analysis and configuration of University Research and Training Reactors (URTRs). This computational environment is intending to assist researchers and educators with tools for an open-architecture neutronic analysis and configuration of the URTRs (to optimize experiments, fuel locations for flux shaping, detector selection and configuration). Such computational environment does not currently exist. The method of characteristics based computer code, AGENT will revolutionize the way in which URTR research is planned, deployed and analyzed through the so called "*virtual reactor environment*." This paper provides an overview of the graphic tools that were designed to support the AGENT code programming architecture. An analysis of the visualization techniques, graphics operations that are supported and the computational interaction that is allowed is described in details.

KEYWORDS: R – Functions, research reactors, computer graphics

1. Introduction

Currently under development, the "*virtual reactor environment*" is planed to be a fully open modular code system for access and use by the entire URTR and associated reactor communities. Main applications target the basis for a variety of experimental and component design activities, and a useful aid for a number of out-reach activities at all educational levels. The platform is supposed to be accessible by K through 12 schools via the internet, so that realistic URTR reactor performance cases can be run and viewed on-line from their classrooms. As we expect this may provide a major new resource to the nation for the widespread development of educational and research activities. One of major issues related to a successful usage and application of the complex virtual computer simulation tools is a graphic support.

Once developed, the suite of codes and user interfaces will comprise a "*virtual reactor*" where users can literally configure core fuel arrangements, reactor experiments, and advanced reactor configurations. Every step of the virtual experiment will be possible to view on the screen using a variety of graphic modes available as an integral part of the "*virtual reactor environment.*" This will permit the community of URTR operators and experimenters to run a *virtual reactor* and monitor reactor behavior and performance in a unique and very new way. The *virtual reactor* system will also support distance educational and outreach programs where remote access to the code system will allow educators to perform virtual reactor demonstrations.

The current state of development of codes that address the spectrum of goals specific to core physics of the research reactors is extremely limited. We are developing a core-following calculation system for URTRs. The main part of this system is a deterministic code AGENT based on the method of characteristics that can model the actual geometry and thus simulate near-real conditions in research reactor cores. It will give users direct insight into neutronics, neutron transport modeling, the physics behind the numerical models and the physics behind experiments. The AGENT will have an extensive graphical user interface (GUI). This paper describes our current graphics capabilities of the AGENT *virtual reactor design environment* and potentials for their further development and improvements.

2. Geometry Visualization Procedure

2.1 Geometric Modeling

The purpose of AGENT code's graphic tools are to assist in the design and experimentation with different reactor configurations. Geometry visualization is a very important step between the input geometry design and the obtainment of output results. The graphic tools use the same input file as the main modules of the AGENT code and this allows detection of design errors at an early stage. Only when the input geometry is verified, the main part of the code will proceed. The numerical output like spatial and energy distribution of neutron flux and reaction rates can be displayed as well. An overview of a *virtual reactor* computation cycle is shown in "Figure 1."

AGENT code supports the description of complicated reactor geometries through the use of constructive solid geometry (CSG) operations [2, 3]. Simple shapes such as boxes, spheres and cylinders can be combined into more complicated ones through their union, intersection or difference. The simple shapes that are supported by AGENT are stored as a library of implicit functions. These functions are then combined through the theory of R – functions [1, 4-6]. The theory of R – functions as used in the AGENT code represents the building block of the CSG operations. The use of R – functions allows the representation of complicated shapes by single real valued functions that take positive values in the interior of the object, negative values outside the object and zero values on the boundary of the object, [1]. Internally, the functions are constructed during the parsing of the input file and are saved internally as trees. This facilitates their evaluation at the subsequent stage of the polygonization.

2.2 Polygonization of Implicit Functions

The polygonization of the implicit functions that define the various geometric bodies is the most important step towards their rendering to the screen. This step involves the construction of triangular meshes that will approximate the surfaces of the geometric bodies. During the polygonization phase, the implicit functions are evaluated at the points of three – dimensional grids in order to compute the locations of the triangles' vertices and determine the way that these should be connected. In order to achieve an accurate approximation of the surfaces there are two basic requirements. The three dimensional

grid should fully enclose the geometric body and it should be dense enough in order to capture the surface details.

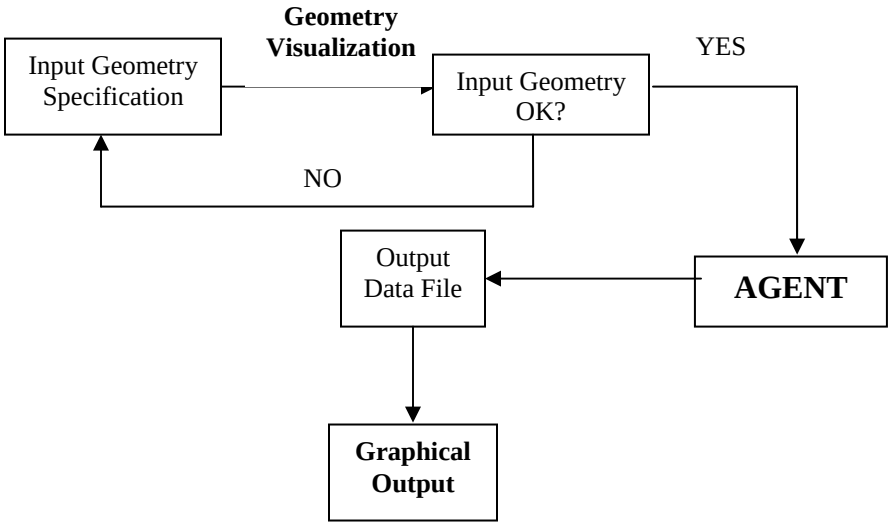

Fig.1: Overview of the design cycle

The specific algorithm that we are using in the AGENT code's graphic tools for performing implicit surfaces polygonization is the popular "marching cubes" algorithm [7]. This algorithm examines all cubical cells that construct the three dimensional grid over which the implicit function is evaluated. Each cell is considered individually and for each one the polarity of the implicit function at the corners of the cell is examined. If all corners are of the same sign then the cell is considered as empty since it will be either outside or inside the surface (depending on the polarity of the function). If there are corners of differing signs, then along the edges that connect such corners there should be an intersection with the surface. The algorithm stores internally the polygonal approximation for 15 possible cell corners' polarities combinations and by using these, it determines how should the implicit surface be approximated for each cell that it examines.

As it was mentioned above, there are two important decisions that have to be made for applying the marching cubes algorithm. The first one regards the choice of how big the three dimensional volume over which the function will be evaluated should be. This decision involves the choice of a bounding box of an appropriate size. A very small bounding box might lead to some parts of the surface being missed whereas a very big bounding box will lead to unnecessary function evaluations. In order to deal with this issue, we are employing interval analysis [8-10]. By using intervals it is easy to determine a bounding box (which can be thought of as three separate intervals – one for each of the principal directions) that will fully enclose the surface without being excessively big. To determine the bounding boxes, we begin with the bounding boxes of the simple basic shapes and then as we traverse the CSG trees, we apply to them the operations from which the geometric bodies are constructed. The second decision concerns the choice of the size of the cubical cell. A big cubical cell might result in a grid that is not dense enough whereas a very small cubical cell will lead to a grid of very high resolution which will result in an excessive number of both triangles and implicit function evaluations.

One of the main goals of the AGENT code's graphic tools is to allow the user to interact with the geometry input data and its graphic display. Thus, operations such as rotation or zooming are of great importance. Polygonal approximations that consist of a very big number of triangles might make such operations difficult to perform. Taking this into account, we choose a grid resolution that is sufficient for capturing the surface topology and will not cause the problems as just described.

A characteristic of the bodies that we are modeling is the presence of sharp features. This, combined with the fact that the grid resolution used is not very fine, leads to an initial triangular mesh that will not be an approximation to the implicit surface completely free of faceting or other visual imperfections. As a result, in order to deal with these issues, after performing the initial polygonization step, we apply a mesh optimization post processing step as described in [12]. At this step, the vertices of the triangular mesh and the mesh normals are modified in order to obtain a more accurate polygonization of the implicit surface. Three different operators are applied to the initially constructed mesh. The first acts on the positions of the mesh vertices and moves them towards the implicit surface. The second operator modifies mesh vertices in order to reduce the deviation of mesh normals from their respective surface normals. Finally, the third operator improves the regularity of the triangular mesh. This process is repeated in an iterative manner and is controlled by two error metrics that measure the deviation of the mesh vertices from the surface and the deviation of the triangle normals from the surface normals.

2.3 Arbitrary vs. Lattice Input Mode

In AGENT there are two different input modes, the arbitrary and the lattice input modes, [11]. In the arbitrary mode each geometric body has to be individually defined. The input in this case is similar to standard Monte Carlo codes. For this type of input, an implicit function has to be polygonized for each geometric body. In contrast, in the lattice mode, unit cells consisting of several geometric bodies are defined and then these are combined in order to generate the lattice area. Therefore, each geometrically different body has to be polygonized only once. Afterwards, it is sufficient to translate it and assign a color to it according to the input data related to material (cross section) specification. This fact, allows geometrically equivalent but materially different geometric bodies to be rendered to the screen efficiently. An overview of the visualization procedure for these two input modes is provided in "Figure 2."

3. Geometry Visualization Examples

In this section we provide examples that illustrate the functionality of the AGENT code's graphic tools. Operations, such as rotation and zooming, that facilitate viewing of the geometry as a whole are presented in addition to the option of using transparency for the display of a geometric body. Rotation and zooming can be performed by the user once the geometry is rendered to the screen with the use of the mouse. Transparency has to be specified for individual geometric bodies before rendering takes place.

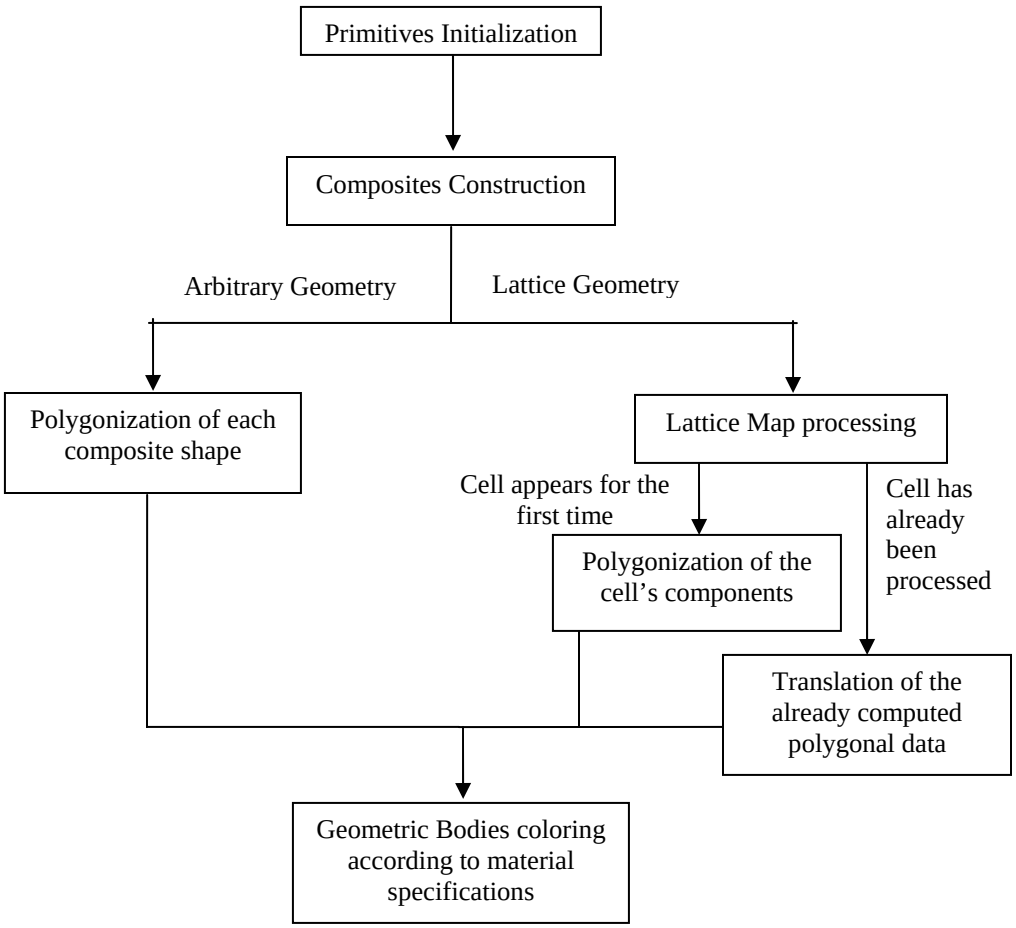

Fig.2: Outline of the visualization procedure

3.1 Power Reactor Unit Assembly (7 x 7 BWR Assembly)

In this first example, 7x7 BWR assembly geometry with heterogeneous pin cells, modeled with AGENT code's graphic tools is shown in "Figure 3" and "Figure 4." "Figure 3" depicts an image vertical to the z axis and then "Figure 4" provides an image of the geometry after it has been rotated showing an arbitrary selected assembly height.

This assembly, as an example of lattice area, consists of only two geometrically different unit cells and six materially different cells. This is also shown in figures through the choice of different colors.

3.2 Research Reactor Geometry

In this second example, we show the complex geometry of a research reactor core. As it was the case in the first example, this configuration has also been modeled by using the lattice geometry input mode. In "Figure 5" we provide a two dimensional image whereas in "Figure 6" we provide a zoomed section that display details in more clear manner. It can be seen from this example that the complicated assemblies and all

fine details of the configuration can be captured using AGENT code's geometric modeling capabilities and verified through the use of the graphic tools.

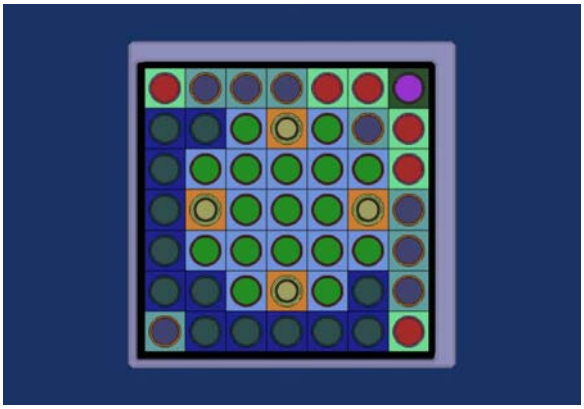

Fig. 3: A two dimensional image of a 7x7 BWR geometry with heterogeneous pin cells

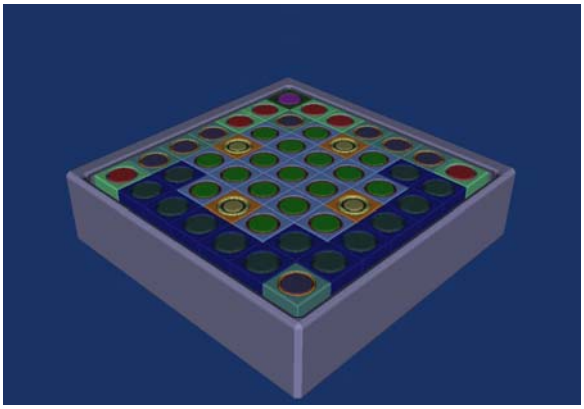

Fig. 4: A three dimensional image of the same geometry presented in Figure 3

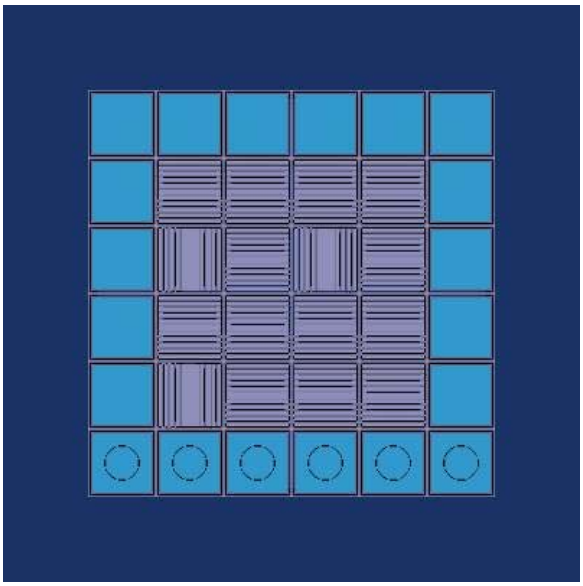

Fig. 5: Geometry display of a research reactor full core

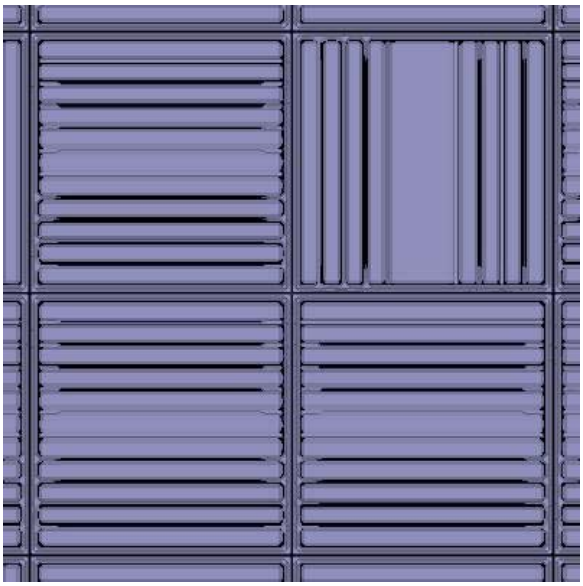

Fig. 6 A zoomed section of the full reactor core presented in Figure 5

3.3 Use of Transparency Graphics Modality

This third example illustrates the use of transparency for verifying the location of different geometric bodies relative to each other. In "Figure 7" and "Figure 8" we can see two different images of a geometrical domain consisting of a thin fuel plate located in water. The water is modeled as a rectangular body which fully encloses the fuel plate. In such a situation, transparency is the only means for ensuring that the relative positions of the geometric bodies are correct.

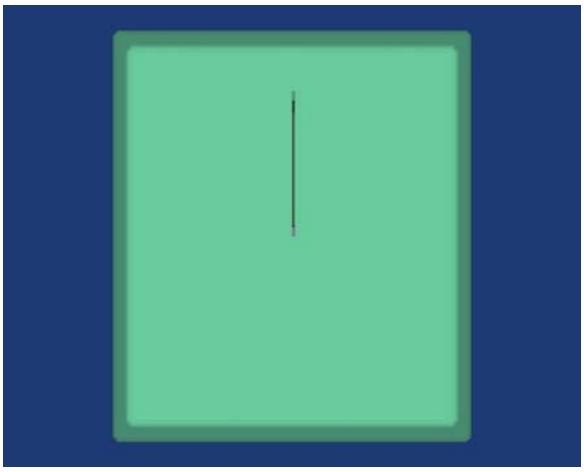

Fig. 7: A simple example of a thin fuel plate located in water, X-Y plane view

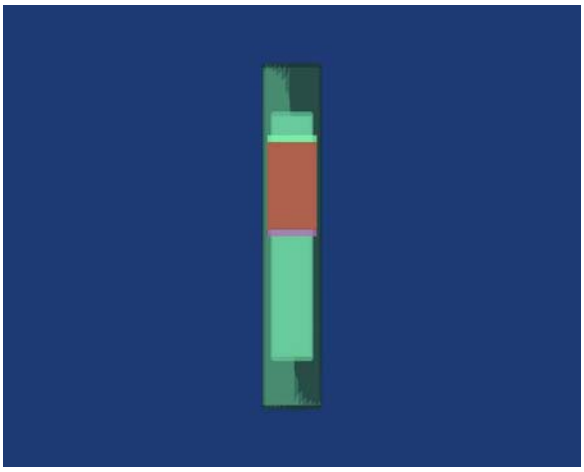

Fig. 8: A simple example of a thin fuel plate located in water, Z-Y plane view

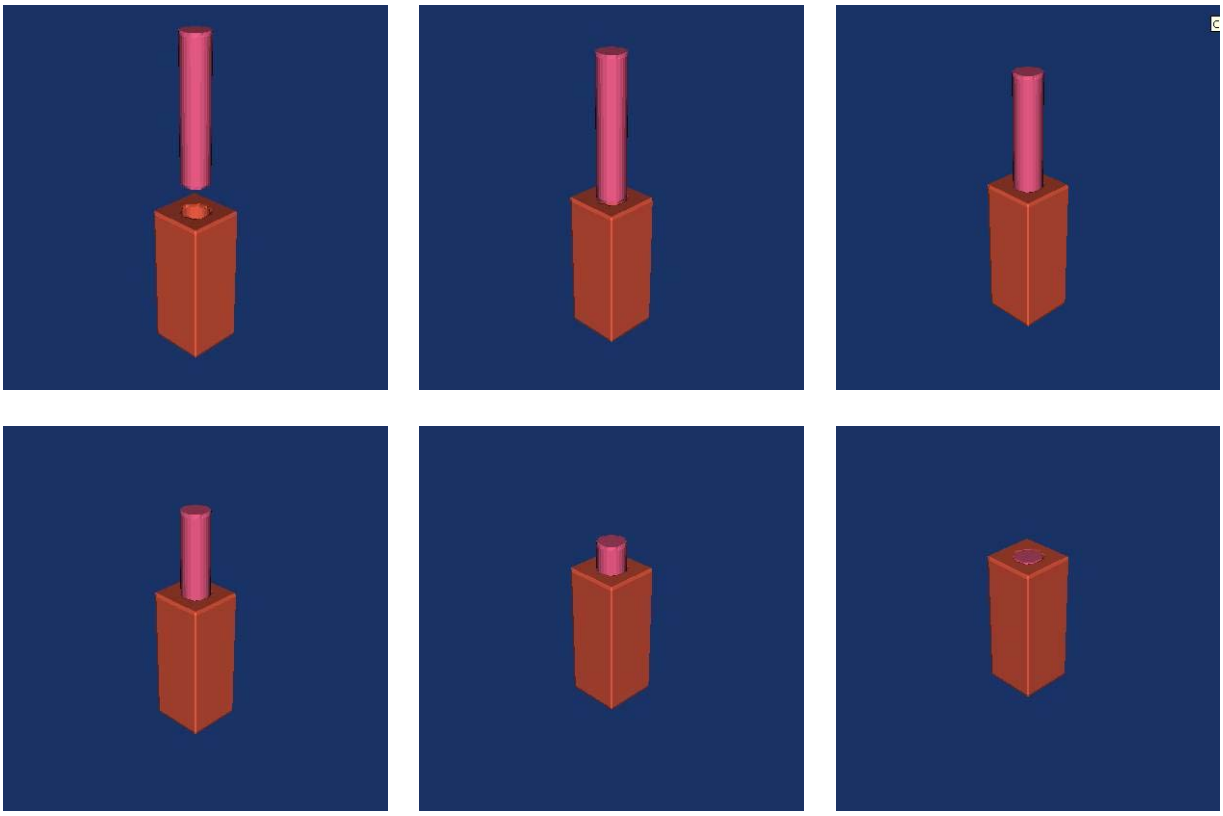

Fig.9: Example of time varying geometry modeled by AGENT graphic tools (control rod movements)

3.4 Graphics Representation of Time -Varying Geometries

AGENT graphic modules based on the CSG modeler and the theory of R – functions allow for an easy representation of complex geometries in *n*-dimensional space. That includes a possibility to represent 3D object in time. As one example we show in "Figure 9" a movement of a single control rod.

4. Conclusion

The graphical user interface of the AGENT *virtual reactor design environment* allows the user to view and verify input data, enabling an early detection of errors. The user can view also the output data such as flux and reaction rates in forms of energy and 3-D spatial maps, [1]. The input file used for the main code is also used as input for the graphics routines. Our current graphic module translates complex geometrical arrangements into colorful drawings with *rotation* and *zooming* operations to facilitate the viewing of the whole domain. In addition we are developing the time dependent geometry representation such as control rod movements. Our future work will be more focused on modeling motions and deformations of geometrical domains.

Acknowledgements

Research supported by Grant Number 2402-PU-DOE-4423 under the Innovations in Nuclear Infrastructure and Education (INIE) Program of the US Department of Energy.

References

1) T. Jevremovic, H.C. Lee, K. Retzke and Y. Peng "AGENT Code Open – Architecture Analysis and Configuration of Research Reactors – *Neutron Transport Modeling with Numerical Examples*", This Conference (2004)
2) A.A.G. Requicha, "Representation for rigid solids: theory, methods and systems." ACM Comp. Surv., 12 (4), 434 (1980)
3) A.A.G. Requicha and H.B. Voelcker, "Boolean operations in solid modeling: boundary evaluation and merging algorithm", Proc. of the IEEE, 73(1), 30 (1985)
4) V.L. Rvachev, "Theory of R – Functions and some applications", Naukova Dumka, Kiev (1982) (In Russian).
5) V. Shapiro, "Theory of R – Functions and applications – A Primer", Tech. Report CPA 88 – 3, Cornell Programmable Automation, Cornell University, Ithaca, New York (1988)
6) V. Shapiro, "Real functions for representation of rigid solids", Comp. Aided. Geom. Design, 11, 153 (1994)
7) W.E. Lorensen and H.E. Cline, "Marching Cubes: a high resolution 3D surface construction algorithm", Computer Graphics (Proceedings of Siggraph '87), 21(3), 163 (1987)
8) R.E. Moore, "Interval analysis", Prentice Hall, Englewood Cliffs, New Jersey, 1 (1966)
9) R.E. Moore, "Methods and Applications of Interval Analysis", SIAM, Philadelphia (1979)
10) T. Duff, "Interval Arithmetic and Recursive Subdivision for Implicit Surfaces and Constructive Solid Geometry", Siggraph '92, 26(2), 131 (1992)
11) T. Jevremovic, T. Ito, Y. Inaba, "ANEMONA : multiassembly neutron transport modeling," *Annals of Nuclear Energy*, **29**, 2105, (2002)
12) Yu. Ohtake, A.G. Belyaev and A.A. Pasko, "Dynamic meshes for accurate polygonization of implicit surfaces with sharp features." Proc of International Conference on Shape Modeling and Applications, Genova, Italy, May 2001, 74 (2001)