Annals of Nuclear Energy 55 (2013) 248–257

Contents lists available at SciVerse ScienceDirect

Annals of Nuclear Energy

journal homepage: www.elsevier.com/locate/anucene

Visual numerical steering in 3D AGENT code system for advanced nuclear reactor modeling and design

Hermilo Hernandez [a,*], Jovana Knezevic [b], Thomas Fogal [c], Todd Sherman [a], Tatjana Jevremovic [a]

[a] The University of Utah, Nuclear Engineering Program, 50 S. Central Campus Dr., Salt Lake City, UT 84112, USA
[b] Technische Universität München, Computation in Engineering, Arcisstr. 21, 80290 Munich, Germany
[c] The University of Utah, Scientific Computing and Imaging Institute, 72 S. Central Campus Dr., Salt Lake City, UT 84112, USA

ARTICLE INFO

Article history:
Received 22 November 2012
Accepted 7 December 2012
Available online 24 January 2013

Keywords:
Numerical steering
AGENT code
Deterministic neutron transport codes
Method of Characteristics
R-functions
Numerical visualizations

ABSTRACT

The AGENT simulation system is used for detailed three-dimensional modeling of neutron transport and corresponding properties of nuclear reactors of any design. Numerical solution to the neutron transport equation in the AGENT system is based on the Method of Characteristics (MOCs) and the theory of *R*-functions. The latter of which is used for accurately describing current and future heterogeneous lattices of reactor core configurations. The AGENT code has been extensively verified to assure a high degree of accuracy for predicting neutron three-dimensional point-wise flux spatial distributions, power peaking factors, reaction rates, and eigenvalues. In this paper, a new AGENT code feature, a computational steering, is presented. This new feature provides a novel way for using deterministic codes for fast evaluation of reactor core parameters, at no loss to accuracy. The computational steering framework as developed at the Technische Universität München is smoothly integrated into the AGENT solver. This framework allows for an arbitrary interruption of AGENT simulation, allowing the solver to restart with updated parameters. One possible use of this is to accelerate the convergence of the final values resulting in significantly reduced simulation times. Using this computational steering in the AGENT system, coarse MOC resolution parameters can initially be selected and later update them – while the simulation is actively running – into fine resolution parameters. The utility of the steering framework is demonstrated using the geometry of a research reactor at the University of Utah: this new approach provides a savings in CPU time on the order of 50%.

Published by Elsevier Ltd.

1. Introduction

1.1. The AGENT (Arbitrary Geometry Neutron Transport) methodology

The AGENT (Arbitrary Geometry Neutron Transport) methodology (Fogal et al., 2010; Hursin et al., 2006; Hursin and Jevremovic, 2005; Jevremovic et al., 2010, 2009, 2006, 2002; Xiao and Jevremovic, 2010a, 2010b; Xue and Jevremovic, 2008; Yang and Jevremovic, 2010a,b) provides a solution to the Boltzmann neutron transport equation using the Method Of Characteristics (MOCs) that is merged with the theory of *R*-functions (Rvachev, 1967; Shapiro, 1991) in capturing all details of geometrical and material heterogeneities of present and future reactor designs. The *R*-functions modeler as used in AGENT provides a representation of complex domains through a combination of simple primitive objects. Using *R*-functions, a Boolean combination of primitive objects (described by their corresponding simple domain functions) are combined into a single analytical equation representing that complex domain. This allows for great flexibility in hierarchical organization of any type of reactor geometry. The modeler is equally general as a Monte Carlo combinatorial geometry based approach, but is incomparably simpler, more intuitive and most importantly faster. In addition, the AGENT modeler permits automatic submesh generation, i.e. small-sized flat-flux zones as required for refined MOC accuracy. Visualization of simulation results from AGENT runs is available for a wide range of platforms including mobile devices (Jevremovic et al., 2011).

The MOC requires as flexible as possible selection of a number of azimuthal and polar directions along with a refined netting of neutron tracks (either specified by track number or by separation in between them). An additional parameter of great importance for achieving acceptable MOC accuracy is the possibility to adequately submesh the geometry. The theory of *R*-functions in the AGENT methodology allows for a flexible yet refined selection of a number of azimuthal and polar directions of neutron motion along straight rays (tracks) defined by the distance among them, and a geometry-independent method of creating submeshes of the entire reactor core. A user-defined number of rays is generated for each azimuthal and polar direction, and intersections of

* Corresponding author.
E-mail address: hermilohdez@gmail.com (H. Hernandez).

0306-4549/$ - see front matter Published by Elsevier Ltd.
http://dx.doi.org/10.1016/j.anucene.2012.12.008

neutron tracks with the reactor geometry are found for each surface within the geometrical domain. The three-dimensional (3D) AGENT methodology is realized via a combination of the two-dimensional (2D) radial MOC solution along the reactor assembly plane and the one-dimensional (1D) MOC solution, taking special care to conserve the axial neutron leakage. The whole reactor core 3D solution methodology is based on the so-called "flux tunneling" in between the assembly faces through conserved angular flux values per assembly face segments (edges) along the angular directions. All these main aspects of the AGENT methodology are briefly illustrated in Fig. 1.

1.2. Improving accuracy and speed of AGENT with steering function

In general, the MOC methodology requires a detailed survey of the most optimal combination of resolution parameters to achieve the highest accuracy yet with as reasonably as possible short CPU time and computational memory occupation. The AGENT methodology allows for a two step optimization for finding the most accurate MOC solution: in step 1, a coarse MOC resolution is selected (small number of azimuthal and polar angles and wide ray separation but with a fine submesh). This quickly provides initial estimates for the scalar and angular flux values as input to step 2, during which the fine resolution parameters are inputed. These two steps may not always produce the best result, and the survey may continue before the best estimate is found. In collaboration with Technische Universität München, we have adopted a computational steering framework into the AGENT code system allowing for the fast search of the best estimate. In general, computational steering is a powerful concept that allows scientists to interactively control a computational process during its execution in order to gain insight on how certain parameters affect the final computational result, the algorithmic behavior, as well as identify avenues for further optimizations and improvements of numerical convergence toward an accurate solution.

2. Simulation steering framework in the AGENT code system

Computational steering allows scientists to interactively control a computational process during its execution. In many cases, this is done to guide the simulation towards the solution to an 'interesting' problem, as opposed to one which is expected or lacks certain desired features. In this work, we utilize a computational steering solution to *speed-up* the simulation by limiting resolution until a particular simulation state is reached (Mulder et al., 1999). The steering function ("component") is introduced into the AGENT code system with a user-interface that provides an *instant* (interactive) communication platform for selecting desired changes to the simulation program at runtime. There are two primary state-of-the-art approaches for doing this: *checkpointing* and process *interruption*.

2.1. Checkpoints implementation

A set of *checkpoints* are inserted at fixed places in the code, and are used to test if an input has changed on the user's side. If something has changed, the instrumentation reconfigures internal data structures as appropriated, and backs up the simulation to be consistent with the modified inputs. Such an approach has several disadvantages. Firstly, it involves major code modifications, making the implementation of such an approach tedious for the end-user (researcher) writing the simulation code. As a consequence, despite the advantages that computational steering may offer, researchers become reluctant to apply the concept in their applications. The second disadvantage is the difficulty in making the right decision as to *where* to insert these checkpoints, since the answer

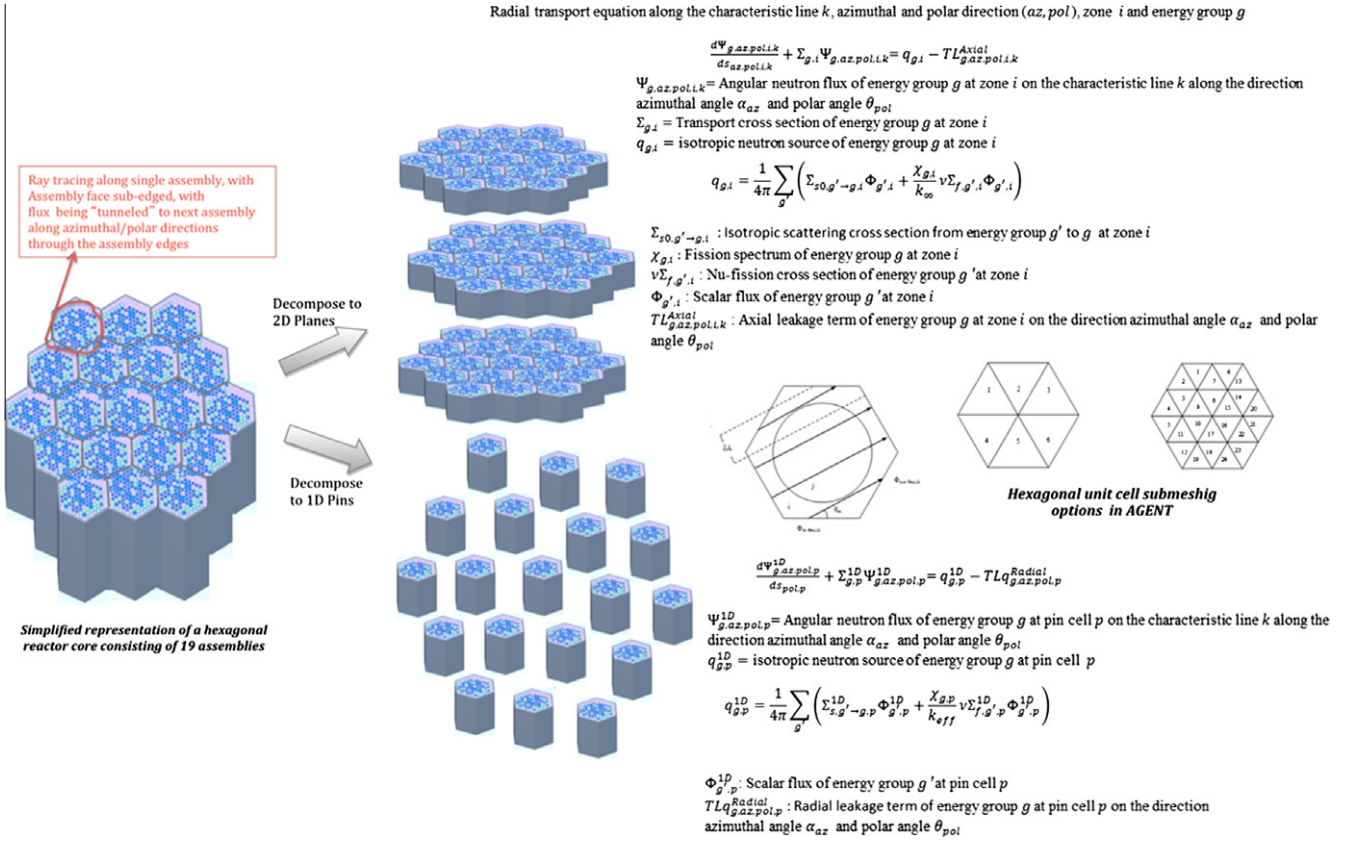

Fig. 1. The AGENT methodology for hexagonal reactor core geometry.

to that question is entirely simulation dependent, and, moreover, problem-size dependent. In order to guarantee the *immediate* response of the computational model to user modifications, one would have to estimate this in advance, based on the experience gained by previous program runs. Another disadvantage is the inevitable trade-off between frequent polling (at considerable computational cost) and responsiveness to new user input. In addition to the insertion of checkpoints, a common approach is in assigning one thread per simulation process *exclusively* to check for updates from the user, but this presents logistical issues in coordinating access to shared data, as illustrated in Fig. 2.

2.2. Interruption implementation

We chose to use an elegant and straightforward integration framework, previously proven to give excellent results in other engineering applications (Fogal and Krüger, 2010; Knežević et al., 2011). This framework provides instant responses for user interaction with only minimal code modifications. The main idea of the framework is to exploit user-generated interrupts, i.e. *signals*, to interfere with the regular operation of the simulation. Signals may be raised by an operating system to notify a program about an error which occurred (e.g., floating point exception), or by a user by a keyboard event (in Unix-based systems CTRL-C by default for program termination e.g. but there are also other keyboard-generated signals possible). The default action for such a signal is to terminate the process; however this computational steering framework utilizes a signal handler to override that operation and provide the user with the opportunity to modify the state of the running simulation.

The user may send the signal at any point during program execution. However, this means that the program could be executing any arbitrary piece of code at that point in time; if the simulation was interrupted after acquiring a resource, we do not want to alter the program state such that the resource is never released. To workaround this problem, instead of directly modifying important state, the signal handler simply rewrites loop indices so that inner loops are considered 'done'. When the signal handler returns, the control is given back to the function which was executed once the signal has occurred, and it may continue from exactly that (previously saved) state. This ensures any required cleanup occurs whilst avoiding the heavy cost of finishing the current iteration at the old data values. Finally, the simulation code is instrumented at its outermost loop to check and see if a user interruption has occurred, and if so it updates the relevant simulation variables.

In AGENT, these updates may refer to changing the convergence criteria parameters, the MOC resolution parameters (such as a number of polar and azimuthal angles and ray separation), maximum number of iterations, etc. Depending on the nature of the modification, appropriate re-initialization steps for the data have to be taken at the beginning of the new iteration, to ensure the correct execution of the AGENT simulation.

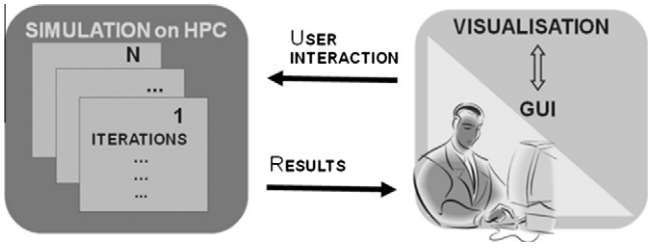

Fig. 2. Typical user-simulation interactive steps with *checkpoints* steering approach.

2.3. Illustration

The whole computation is intended to be restarted by manipulating the iteration vector $i = (idx_1, idx_2, \ldots, idx_n)$, (i.e. the loop indices idx_i of all loops) by setting each loop index to be some value out of its actual range, as shown in the following pseudo code:

```
for (t ← T0 to TN) do          //iterations over time
  reinitialize_data()          //(re)initialize MAX1,MAX2 and other
  necessary data
  for (idx1 ← 1 to MAX1)       //in the case of interrupt, MAX1 and
  MAX2 are both set to −1
    for (idx2 ← 1 to MAX2)
      process(data[idx1][idx2])  //can be interrupted at any
  point
```

2.4. Visualizing the effect of numerical steering framework in AGENT

To make an informed decision on the new AGENT parameter set, a user must be able to understand the efficacy of the in-progress solution. Thus the simulation periodically outputs the current state to disk. Visualization software converts that data into a representable form, and the results are displayed for the user's evaluation.

A common mode of operation for simulations is to run the simulation code on a remote computing resource, such as a supercomputer. This presents a problem for visualization software, which typically runs on the user's desktop and therefore cannot directly access the data. To solve this problem, we have implemented a client/server solution: a simple daemon process runs on the server and brokers access to the data to clients. Client applications are presented with a list of available iterations of the currently running simulation, which is dynamically updated as the simulation progresses. Requesting a data set sends it over to the client machine and automatically loads up the *ImageVis3D* volume rendering tool to visualize the data (Knezevic et al., 2011). Fig. 3 shows the use of *ImageVis3D* client to visualize the main AGENT simulation parameters after multiple iterations have been completed.

3. Effect of steering framework in enhancing AGENT's speed and accuracy

3.1. Parameters of interest to test the effect of steering framework

In addition to all MOC parameters as required to be selected in producing the best estimate result, the AGENT code utilizes (as many other similar methodologies in solving neutron transport equation) in using an iterative process; two criteria are applied to validate the convergence of iterative solution: the first criterion is related to relative difference of the multiplication factors between any two iterations, defined as:

$$k_{diff} = \left| \frac{k_{new} - k}{k} \right| \tag{1}$$

The second criterion is related to the maximum relative difference of zone flux for all zones (a *zone* being defined as the smallest-sized flat-flux area in the core geometry), defined as:

$$\phi_{diff,\max} = \max \left(\left| \frac{\phi_{i,new} - \phi_i}{\phi_i} \right| \text{ for all zone } i \right) \tag{2}$$

When $k_{diff} < \varepsilon_1$ and $\phi_{diff,\max} < \varepsilon_2$, the calculation is considered converged; otherwise, the iterations will proceed until convergence is complete, or until reaching a maximum (user defined) number of iterations.

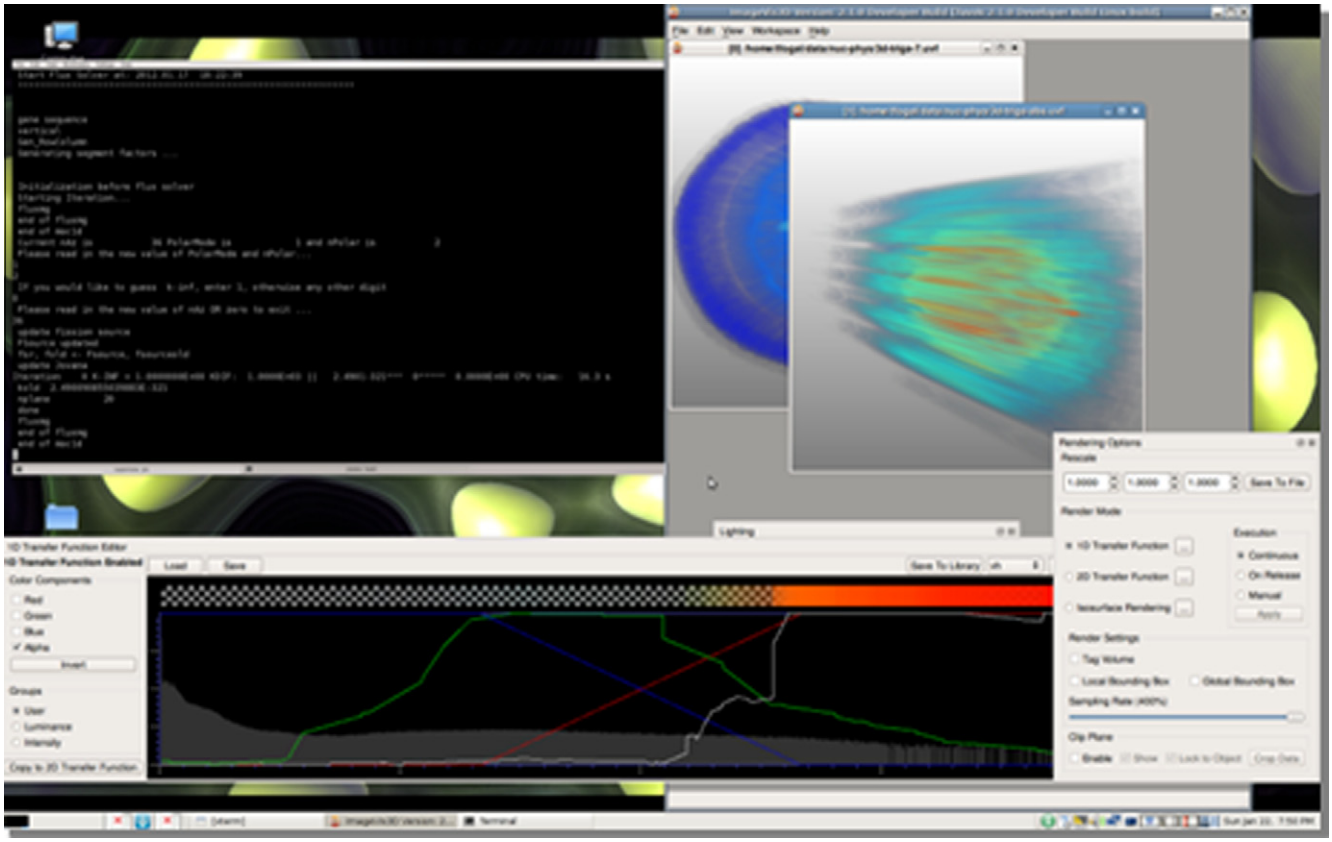

Fig. 3. Instrumented steering framework in AGENT – the user runs AGENT on a remote server (terminal, top left); as iterations complete, visualizations become available appearing in locally-running *ImageVis3D* instance.

Therefore, the effect of the numerical steering functionality introduced into AGENT is then analyzed during the full core simulation of the University of Utah TRIGA research reactor (UUTR). The UUTR has a hexagonal core structure with different types of fuel elements including the reflector elements and moderators, and as such represents a very heterogeneous system. By setting $\square_1 = 0.00001$ and $\square_2 = 0.0001$ for this work there a number of other AGENT resolution parameters available as follows (refer to Fig. 1):

a. Number of polar angles (held fixed for this work).
b. Number of azimuthal angles (n_θ).
c. Ray separation (δA).
d. Number of boundary edges per reactor core face (n_b).

3.2. AGENT search for best estimate

In order to obtain the best estimate for a MOC-based solution to neutron transport in reactor geometries, an optimization between resolution parameters (as described previously) is required. The process is often time consuming. As an example, we describe the best estimate for a 3D AGENT model of the UUTR core as it was determined in the past using the standard optimization process (Yang et al., 2010); the parameters are: $n_\theta = 36$, $n_b = 44$ and $\delta A = 0.1$ cm. The simulation reached the convergence criteria at the 155th iteration with total CPU time of approximately 5 h. The initial flux values were set to zero at the beginning of the simulation. For this configuration, seven energy groups were used and the reactor core was divided into 20 axial planes, with 9414 zones per each plane, thus totaling over 180,000 zones. Since the scalar flux is calculated for each zone, the convergence for scalar neutron flux is slow. Fig. 4 shows the scalar neutron flux values normalized to unit absorption; the highest fast neutron flux values are found at the center of the core; the water and heavy water rods located at the lower right corner and upper left corner of the core elliptically shape the neutron flux at the intermediate energy levels; thermal neutron flux peaks in water regions.

3.3. Monitoring the AGENT live simulation

In this section we show now how the AGENT live simulation of the UUTR as an example, may be monitored; if a user starts the AGENT simulation with a coarse resolution (say, $n_\theta = 8$, $n_b = 44$ and $\delta A = 1.2$ cm) in order to use the coarse solution as input for the refined model, the user needs to verify that under this set of parameters the simulation converges. The solution should converge faster for k but the convergence for neutron flux toward $\square_2$ is slower. After the execution, the convergence can be monitored, and as soon as the k value is greater than 1.0 the calculation can be interrupted. For the aforementioned coarse example, the k value became greater than 1.0 after 80 s of AGENT simulation at the 56th iteration step. After interruption, the MOC resolution parameters are refined (changing δA from 1.2 cm to 0.9 cm for example) allowing the solution to converge quickly. Fig. 5 illustrates the value of k_{diff} (between the two iteration values) for interrupted and uninterrupted simulation cases, as a function of the number of iterations. Since interruption took place at the iteration step 56, we observe a peak at the iteration number 57 in k_{diff}. On the other side, Fig. 6 shows the values for $\phi_{diff,\max}$ (max relative difference of zone flux). For the interrupted scheme, the solution converges after 167

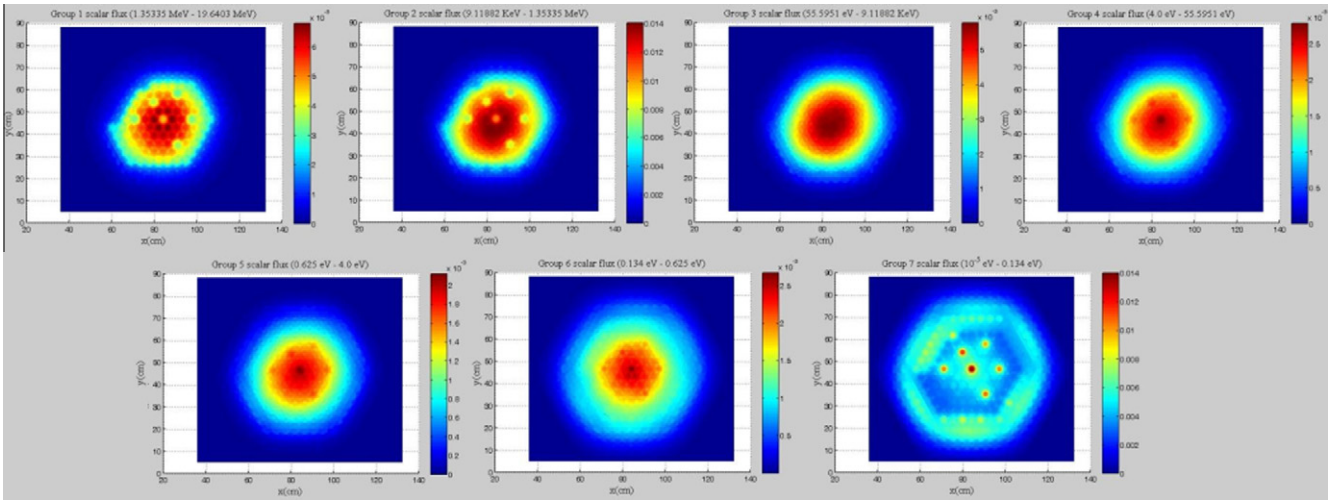

Fig. 4. AGENT radial neutron flux distribution of the UUTR per each of seven energy groups (the first two energy groups are in the fast region, the following two energy groups are in the resonance region and the last three energy groups are thermal).

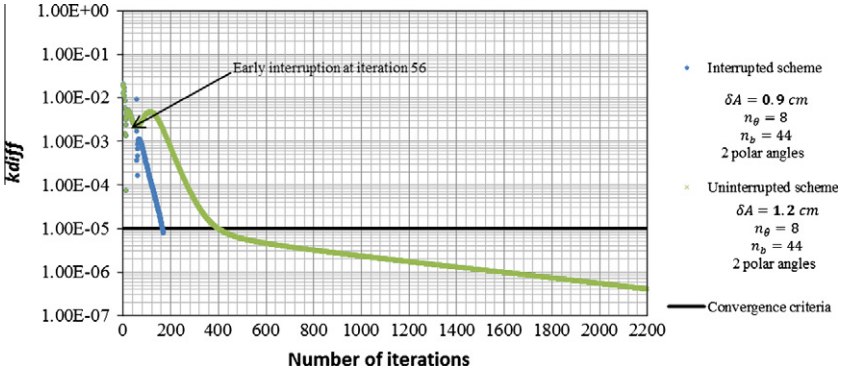

Fig. 5. AGENT k_{diff} for the UUTR as a function of number of iterations when refining the MOC resolution parameters with steering framework.

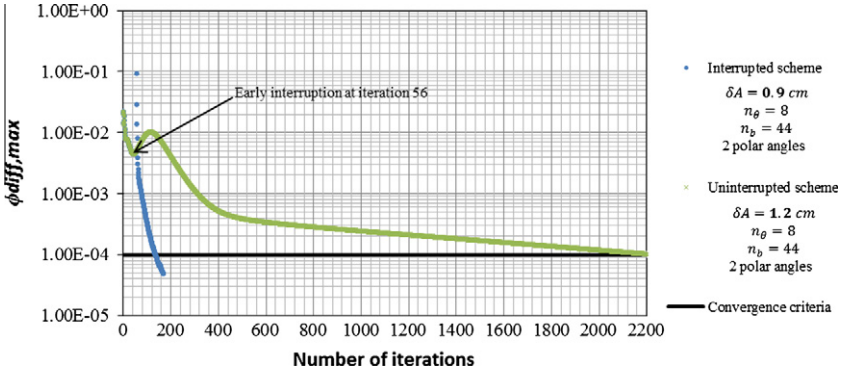

Fig. 6. AGENT $\phi_{diff,\max}$ for the UUTR as a function of number of iterations when refining the MOC resolution parameters with steering framework.

Table 1
AGENT code performance versus MOC resolution parameters for the UUTR.

UUTR core	Total number of iterations	CPU time (s) (gain in CPU time (%))	k_{eff}
Medium-level resolution (no interruption; scalar and angular flux guess according to default values)	157	1000	1.02328
Low → medium level resolution (simulation started with low resolution, interrupted at iteration step 56 and continued with medium level resolution)	159	843 (16)	1.02411
High-level resolution (no interruption; scalar and angular flux guess according to default values)	157	8511	1.02265
Low → high level resolution (simulation started with low resolution, interrupted at iteration step 56 and continued with high level resolution)	158	6634 (22)	1.02331

iteration steps, while in the case of non-interrupted simulation the convergence is very slow, reaching the criteria after 2214 iterations.

3.4. Interrupting AGENT live simulation while achieving higher accuracy and reducing the total CPU time

In this example we show the effect of the interruption early on during the AGENT simulation for the UUTR. The AGENT iterative process starts with *low-level* resolution: $n_\theta = 8$, $\delta A = 0.9$ cm and $n_b = 44$ (in accordance to previously illustrated example). The interruption is introduced at an early stage of the simulation, i.e. as soon as the k_{eff} value reaches 1.0. Two independent cases are then analyzed to compare the gain in CPU time with interruption. Table 1 shows the total number of iterations and CPU-time for fully converged AGENT simulations of the UUTR with respect to two different simulation schemes (*medium-level* resolution and *high-level* resolution) in comparison to simulations that started with a *low level* resolution scheme and then were interrupted after the 56th iteration to continue the iterative process until convergence with higher resolution parameters (*low → medium level resolution* and *low → high level resolution*):

- Medium-level resolution ($n_\theta = 20$, $\delta A = 0.45$ cm and $n_b = 44$).
- Low → medium level [($n_\theta = 8$, $\delta A = 0.9$ cm and $n_b = 44$) → ($n_\theta = 20$, $\delta A = 0.45$ cm and $n_b = 44$)].
- High-level resolution ($n_\theta = 32$, $\delta A = 0.1$ cm and $n_b = 44$).
- Low → high level resolution [($n_\theta = 8$, $\delta A = 0.9$ cm and $n_b = 44$) → ($n_\theta = 32$, $\delta A = 0.1$ cm and $n_b = 44$)].

It can be observed from Table 1 that the *low → medium* resolution interruption scheme consumes approximately 85% of the CPU time with respect to the *medium-level* resolution simulation without interruption. For the *low → high* case, better results are seen, improving simulation time by 22% as compared to the running at *high-level resolution* without interruptions.

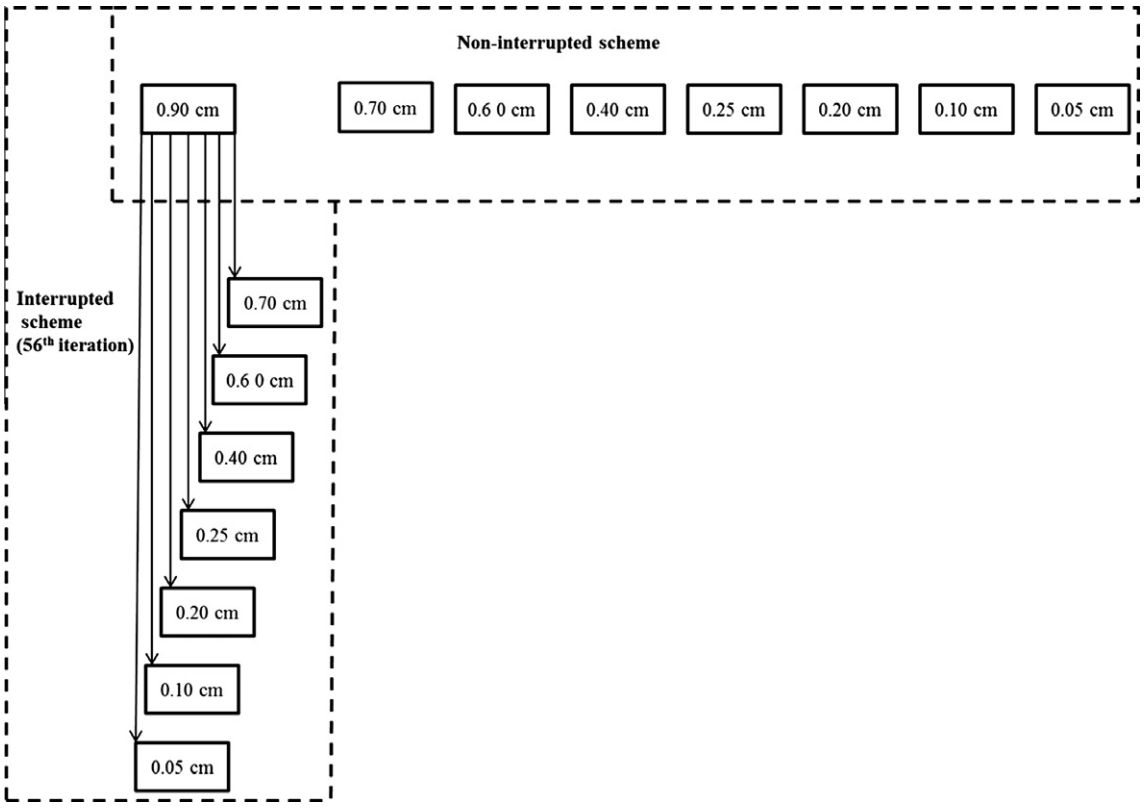

Fig. 7. AGENT best estimate as a function of ray separation survey for the UUTR full core model with and without steering framework (interruption).

Table 2
AGENT survey for the ray separation toward best estimate for the UUTR.

δA (cm)	Non-interrupted scheme (8 azimuthal & 2 polar angles) CPU time for the best estimate = 11,407 s				Interrupted scheme (Fig. 7) CPU time for the best estimate = 8682 s					
	k_{eff}	CPU time (s)	Sum CPU time (s)	Iterations	k_{eff}	k_{eff} error (%)	CPU time (s)	Sum CPU time (s)	CPU time gain (%)	Iterations
0.90	1.02362	598	–	159	–	–	–	–	–	–
0.70	1.02370	714	1312	158	1.02372	−0.002	677	1275	5.2	157
0.60	1.02358	793	2105	157	1.02361	−0.003	732	2007	7.7	156
0.40	1.02385	1149	3254	158	1.02387	−0.002	933	2940	18.8	155
0.25	1.02403	1788	5042	157	1.02405	−0.002	1339	4279	25.1	154
0.20	1.02395	2082	7124	157	1.02398	−0.003	1512	5791	27.4	154
0.10	1.02399	4283	11,407	157	1.02401	−0.002	2891	8682	27.4	154
0.05	1.02402	8441	19,848	157	1.02404	−0.002	5484	14,166	32.5	154

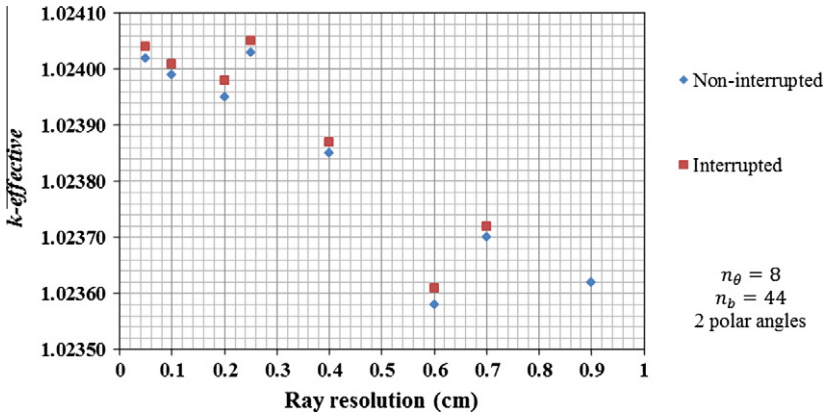

Fig. 8. AGENT k_{eff} versus ray separation for the ray separation survey with and without interruption.

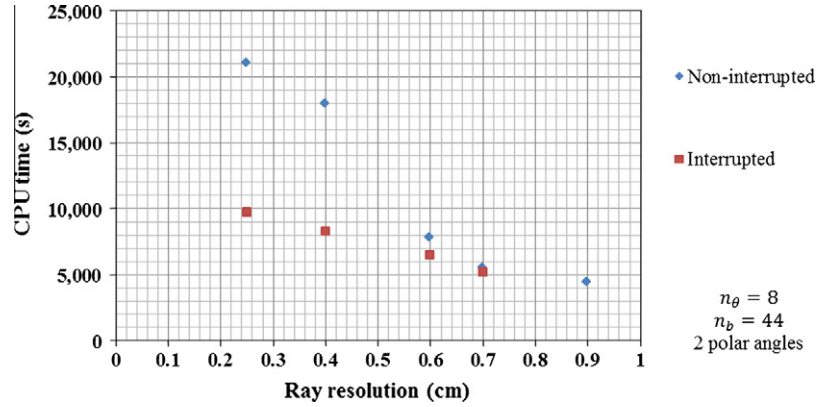

Fig. 9. AGENT CPU time versus ray separation for the ray separation survey with and without interruption.

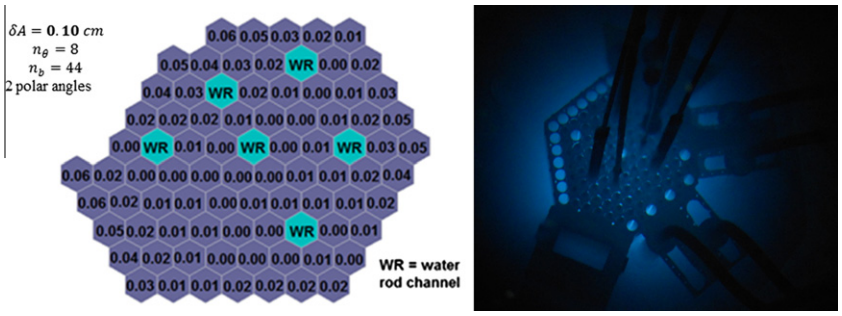

Fig. 10. AGENT fission rates absolute percent error for the optimum ray separation (comparison between interrupted and non-interrupted scheme).

Table 3
AGENT survey for the most optimal number of azimuthal angles toward best estimate.

n_θ	Non-interrupted scheme (0.1 cm ray separation & 2 polar angles) CPU time for the best estimate = 35,815 s				Interrupted scheme CPU time for the best estimate = 24,442 s					
	k_{eff}	CPU time (s)	Sum CPU time (s)	Iterations	k_{eff}	k_{eff} error (%)	CPU time (s)	Sum CPU time (s)	CPU time gain (%)	Iterations
8	1.02340	4443	–	157	–	–	–	–	–	–
16	1.02471	5584	10,027	156	1.02471	0.000	5178	9621	7.27	153
24	1.02405	7821	17,848	156	1.02410	−0.005	6507	16,128	16.80	153
36	1.02344	17,967	35,815	155	1.02348	−0.004	8314	24,442	53.73	153
40	1.02366	21,110	56,925	155	1.02373	−0.007	9715	34,157	53.98	152

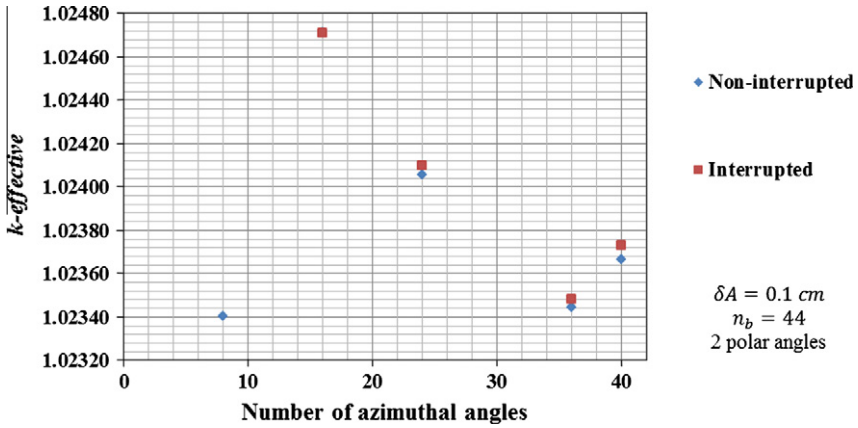

Fig. 11. AGENT k_{eff} versus number of azimuthal angles with and without interruption.

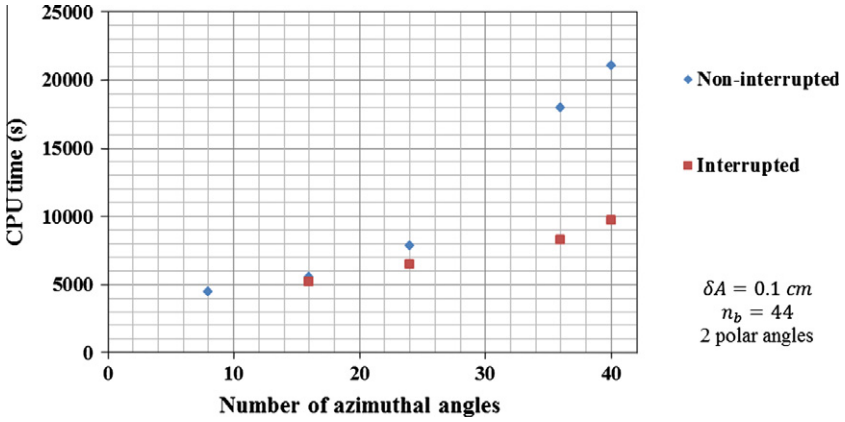

Fig. 12. AGENT CPU time versus number of azimuthal angles with and without interruption.

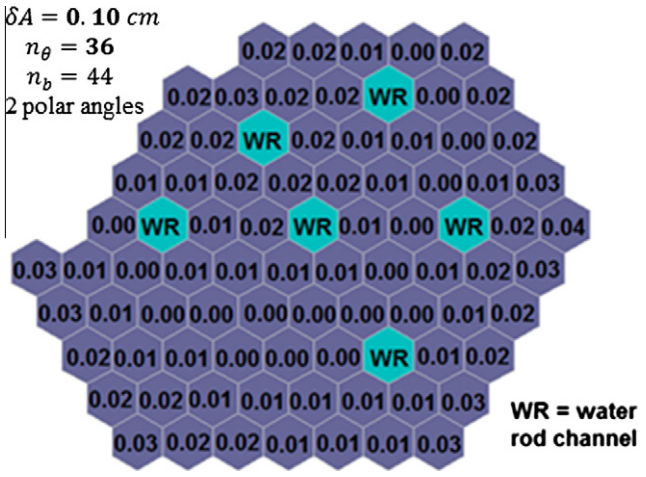

Fig. 13. AGENT fission rates absolute percent error for the optimum number of azimuthal angles (comparison between interrupted and non-interrupted scheme).

4. Optimization of the AGENT resolution parameters in the UUTR model using numerical steering framework

The main benefit of the steering framework is to interrupt complex simulations, starting with coarse resolution parameters and, at the interruption point, redefining them; the coarse MOC resolution parameters generate better initial flux values, thus reducing the number of iterations and total CPU run times, approaching the best estimate faster. In this section we show how the effect of the steering framework is beneficial in shortening AGENT CPU time while obtaining accurate best estimates. Using the UUTR core model, the coarse resolution parameters are used to generate improved initial flux values, and then the simulation is interrupted and modified to utilize fine ray separation and a larger number of azimuthal angels. The number of polar angles is kept the same; using the Leonard and McDaniel scheme (Leonard and McDaniel, 1995), as well as the number of edges (set at 44) (see Fig. 1). The steering functionality was turned ON at the early stage of the simulation, i.e. as soon as k_{eff} value equals 1.0. The following is a

Table 4
CPU time comparison for the AGENT modeling of the UUTR: Effect of numerical steering framework in comparison to non-interrupted scheme.

Comparison	Non-interrupted scheme (s)	Interrupted scheme (s)	CPU time saving (%)
AGENT survey to find optimal ray separation (Table 2)	11,407	8682	24
AGENT simulation for optimal ray separation of 0.1 cm (Table 2)	4283	2891	33
AGENT survey to find optimal number of azimuthal angles (Table 3)	35,815	24,442	32
AGENT simulation for optimal number of azimuthal angles of 36 (Table 3)	17,967	8313	54

summary of the gain in optimization for the best estimate based on the steering function in the AGENT code.

4.1. Optimum ray separation

As shown in Fig. 7, we consider two schemes: non-interrupted AGENT simulation of the UUTR full core model with eight fixed azimuthal angles ($n_\theta = 8$) and changeable ray separations (δA) as follows: 0.9, 0.7, 0.6, 0.4, 0.25, 0.20, 0.10, and 0.05 cm; the interrupted scheme starts with $\delta A = 0.9$ cm and changes it at the 56th iteration (as soon as k approaches 1.0).

The k_{eff} CPU time and number of iterations after convergence are presented in Table 2. The k_{eff} error (in %) represents the relative error between the k_{eff} obtained with the interrupted scheme with respect to non-interrupted scheme. The absolute relative error is considerably small as expected. These small differences in k_{eff} values between the interrupted and non-interrupted schemes are shown in Fig. 8. This figure also shows that the variation of k_{eff} as a function of ray separation decreases with ray separation becoming smaller; thus, the optimum ray separation is 0.1 cm. The CPU time as a function of ray resolution is shown in Fig. 9; the total gain for the optimal case is 27.4% (Table 2). The interrupted and non-interrupted simulation for this optimal case was also compared in terms of fission rates variations. Fig. 10 shows the absolute percentage error in fission rate variations between the interrupted simulation and non-interrupted simulation for the optimum 0.1 cm ray separation. The highest absolute error is as low as 0.06%.

4.2. Optimum number of azimuthal angles

After determining that the most optimum ray separation is $\delta A = 0.1$ cm, the n_θ is changed to 8, 16, 24, 36, and 40 in the non-interrupted simulations. For the interrupted procedure, all the simulations started with $n_\theta = 8$ and the interruption was introduced by changing n_θ to a higher number. The k_{eff}, CPU time and number of iterations are summarized in Table 3. For these AGENT simulations the maximum absolute relative error in k_{eff} was about 0.007% compared to the values obtained with non-interrupted scheme. The CPU time between the interrupted and non-interrupted schemes is considerably reduced. The optimum solution is obtained for the number of azimuthal angles of 36; the interrupted case consumes only 46% of the total CPU time consumed by the non-interrupted scheme simulation; thus the numerical steering provides a significant CPU time saving.

Fig. 11 shows k_{eff} trend as a function of the number of azimuthal angles. The discrepancy between the interrupted scheme k_{eff} value with respect to the non-interrupted scheme increases as with the number of azimuthal angles. However, for these UUTR simulations these k_{eff} differences are relatively small as expected. Form Fig. 12 it can be observed that the use of numerical steering provides a considerable savings of CPU time toward higher number of azimuthal angles ($\sim$54% for 40 azimuthal angles).

Fig. 13 shows the absolute percentage error of average fission rates obtained at the end of the interrupted simulation with respect to the non-interrupted calculation for the optimum number of azimuthal angles (i.e. 36). The calculations for the interrupted simulation show excellent agreement with the non-interrupted scheme, with absolute error peaking at only 0.04%.

Table 4 shows a summary of CPU times in modeling the full 3D UUTR core without any simplifications or assumptions, in concluding the benefit of steering function. It can be seen that the CPU time savings are indeed significant, ranging from 24% to up to 54%.

5. Conclusion

We have introduced a new feature into AGENT, a deterministic neutron transport methodology, based on the concept of computational steering. The steering showed to be extremely well suited for the Method of Characteristics methodologies where survey time increases with the complexity of simulation geometry. This function is non-existent in the neutron transport codes based on deterministic methods. Our analysis proved that this new feature is highly desirable due to its ability to provide significant CPU time savings while preserving the accuracy. In the case of our AGENT code we showed the importance and advantage of providing the new functionality to the user in being able to monitor the code execution and alter the solution parameters. For example, by starting an AGENT simulation with a coarse resolution and then interrupting it early on, the user can drastically accelerate the overall time-to-convergence simulation run. Also, at runtime, the user can correct initial resolution parameters thereby increasing the accuracy of the simulation. The steering interruption function was shown to be an effective tool in increasing of up to 54% the overall efficiency of the AGENT simulations.

References

Fogal, T., Krüger, J., 2010. Tukov: an architecture for large scale volume rendering. In: Proceedings of the 15th International Workshop on Vision, Modeling and Visualization, Siegen, Germany, November 15–17, 2010.

Fogal, T., Xiao, S., Yang, X., Krueger, J., Jevremovic, T., 2010. Visualization as a bridge between chemical and nuclear engineering simulations. In: Proceedings of the AIChE Annual Meeting, Salt Lake City, UT, USA, November 7–12, 2010, p. 231.

Hursin, M., Jevremovic, T., 2005. AGENT code – neutron transport benchmark example and extension to 3D lattice geometry. Nuclear Technology & Radiation Protection 20 (2), 10–16.

Hursin, M., Xiao, S., Jevremovic, T., 2006. Synergism of the method of characteristics, *R*-functions and diffusion solution for accurate representation of 3D neutron interactions in research reactors using the AGENT code system. Annals of Nuclear Energy 33, 1116–1133.

Jevremovic, T., Itoh, T., Inaba, Y., 2002. ANEMONA: multiassembly neutron transport modeling. Annals of Nuclear Energy 29 (17), 2105–2125.

Jevremovic, T., Hursin, M., Satvat, N., Hopkins, J., Shanjie, X., Godfree, G., 2006. Performance, accuracy and efficiency evaluation of a three-dimensional whole-core neutron transport code AGENT. In: Proceedings of ICONE14 International Conference on Nuclear Engineering, Miami, Florida, USA, July 17–20, 2006, pp. 435–445.

Jevremovic, T., Xiao, S., Satvat, N., Gert, G., 2009. Neutron transport benchmark examples with web-based AGENT. Nuclear Engineering and Design 238, 1975–1986.

Jevremovic, T., Yang, X., Xiao, S., Satvat, N., 2010. Modeling reactors with AGENT: verification, validation, efficiency, analysis and 3D-visuals on iPod. In: Proceedings of the INREC10 Conference, Amman, Jordan, March 21–24, 2010.

Jevremovic, T., Fogal, T., Choe, D., Yang, H., Krüger, J., 2011. The role of virtual engineering and EMERGING visualization tools in nuclear engineering education and training at the University of Utah. In: Proceedings of the NESTet Conference, Prague, Czech Republic, May 15–18, 2011.

Knezevic, J., Frisch, J., Mundani, R.P., Rank, E., 2011. Interactive computing framework for engineering applications. Journal of Computer Sciences 7 (5), 591–599.

Knežević, J., Mundani, R., Rank, E., 2011. Interactive computing-virtual planning of hip joint surgeries with real-time structure simulations. International Journal of Modeling and Optimization 1 (4), 308–313.

Leonard, A., McDaniel, C.T., 1995. Optimal polar angles and weights. Transactions of the American Nuclear Society 73, 171.

Mulder, J.D., van Wijk, J.J., van Liere, R., 1999. A survey of computational steering environments. Future Generation Computer Systems 15 (1), 119–129.

Rvachev, V.L., 1967. Geometric Applications of Logic Algrebra. Naukova Dumka, Kiev, Ukraine.

Shapiro, V., 1991. Theory of *R*-Functions and Applications: A Primer. Cornel University, Ithaca, NY.

Xiao, S., Jevremovic, T., 2010a. High performance reconfigurable hardware acceleration applied to neutron transport computation based on the AGENT methodology. In: Proceedings of the ICONE18 Conference, vol. 2, Xi'an, China, May 17–21, 2010, pp. 205–219.

Xiao, S., Jevremovic, T., 2010b. FPGA hardware acceleration for high performance neutron transport computation based on AGENT methodology. In: Proceedings of the PHYSOR Conference, vol. 1, Pittsburg, PA, USA, May 9–14, 2010, pp. 492–505.

Xue, Y., Jevremovic, T., 2008. Full hexagonal core modeling with the AGENT code. In: Proceedings of the ANS Annual Meeting, Anaheim, CA, USA, June 8–12, 2008.

Yang, X., Jevremovic, T., 2010a. Solving the time-dependent transport equation using time-dependent method of characteristics and Rosenbrock method. In: Proceedings of the ICONE18 Conference, vol. 2, Xi'an, China, May 17–21, 2010, pp. 211–216.

Yang, X., Jevremovic, T., 2010b. Spatial time-dependent reactor kinetics methodology based on the method of characteristics. In: Proceedings of the PHYSOR Conference, Pittsburg, PA, USA, May 9–14, 2010, pp. 373–386.

Yang, X., Xiao, S., Choe, D., Jeveremovic, T., 2010. Neutronics modeling of TRIGA reactor at the University of Utah using AGENT, KENO6 and MCNP5 codes. In: Proceedings of the RRFM Conference, Marrackech, Morocco, March 21–15, 2010.