Radiation Shielding Calculation using the Capabilities of Large-scale Mesh Generation and Efficient Parallel Computing in JSNT on Tens of Thousands of Processors

Tangpei Cheng, Zeyao Mo, Guangchun Zhang, Jie Yan, Quan Xu, Yuanguang Fu, Li Deng

Institute of Applied Physics and Computational Mathematics, Beijing, 100088, P.R. China
CAEP Software Center for High Performance Numerical Simulation, Beijing, 100088, P.R. China
cheng_tangpei@iapcm.ac.cn
zeyao_mo@iapcm.ac.cn
zhang.guangchun@qq.com
yan_jie@iapcm.ac.cn
jluxuquan@126.com
fu_yuanguang@iapcm.ac.cn
deng_li@iapcm.ac.cn

Abstract – *To address the challenge in radiation shielding calculation using discrete ordinates method, this paper presents our effort on developing the large-scale mesh generation capability and the optimization of parallel computing on tens of thousands of processors. To make tradeoff between precision and efficiency, multiple strategies are presented in the determination of material and source during the mesh generation. On top of the domain partition algorithm and the DAG-based data-driven algorithm as well as the patch-based abstraction, several optimizations such as exploit more parallelism in moment-to-discrete operation and angular directions in sweeping, develop optimal priority strategy and reduce DAG overhead by vertex clustering and coarse graph constructing have been presented. The correctness is verified using the VENUS-3 benchmark and the parallel performance is measured using a reactor cavity leakage radiation shielding model of a commercial PWR in China, which shows good scaling behavior from 768 to 10080 processors. More importantly, the relative deviation between computational and experimental results in thermal flux has been reduced to less than 20%, which improves the precision at least 2 times as compared to the traditional industry method.*

I. INTRODUCTION

Radiation shielding calculation is a branch of important application in nuclear science and engineering. For example, an accurate radiation shielding calculation of the neutron fluence and fluence rate at several locations is essential for the analysis of integral dosimetry measurements and for predicting irradiation damage values in the reactor pressure vessel (RPV), which is of paramount importance to the possibility of plant life extensions [1]. Among the radiation shielding methods, the discrete ordinates (S_N) is a popular method over the past several decades, due to its advantage in providing high efficiency, good precision and large amount of detailed information. However, as the prevalence of three-dimensional calculation and the requirement on detailed modeling to achieve higher precision, they confront nuclear engineers with a formidable set of challenges. The first is the difficulty in large-scale mesh generation. Traditional S_N programs such as TORT provide the "body-region-zone" method, which is time-consuming and error prone, especially for the model with tens or hundreds of millions of grids and tens or hundreds of thousands of bodies. The second is the efficiency and memory bottlenecks in SN transport calculation. Typically, radiation shielding calculations require millions or millions of grids, tens or hundreds of energy group and hundreds of angular directions, which result in a total of tens to hundreds of billions of DOFs (degree-of-freedoms) and requires hours to days to simulate.

The address the challenges, this paper presents our effort on developing the large-scale mesh generation capabilitiy combined with the optimization of parallel computing in the S_N radiation transport code JSNT [2, 3] based on JASMIN framework [4] as well as its applications. Our aim is to develop a massive parallel S_N code with good scalability, usability for simulating real-world problems, such as Denovo [5], UNIC [6] etc. To ease user's burden on establishing models, the mesh generation is performed according to the CAD model and the determination of material property and source are based on different strategies, such as the multi-points method, the conservation volume method etc. To overcome the limitations in efficiency and memory capacity, the domain partition algorithm and the DAG (Directed Acyclic Graph) based data driven algorithm are implemented using the patch-based method and several optimizations such as exploit more parallelism, reduce parallel overhead are developed.

II. MESH, MATERIAL AND SOURCE GENERATION

As depicted in Fig. 1, the first step is to establish the CAD model using tools such as JLAMT [7]. Then the model information is converted into GDML, which can be used both by JSNT for mesh generation and Monte Carlo codes such as JMCT [8] and Geant4. As shown in Fig. 2, the CAD model can be a large, full-core one while the S_N computing model can be a small, typically quarter-core or eighth-core model with reflective boundary conditions, following the principal "go as you meshing". This is a very useful feature since the full-core CAD model can be built in a more convenient way, such as using the "pin-assembly-core" mode in three steps as well as its variations in JLAMT. Especially, a single large CAD model can act as one fits all, such as computing models with different domain size and mesh discretization. Moreover, keeping data consistence is much easier for S_N/Monte Carlo coupled calculation using the same CAD model.

Based on the CAD model, the second step is to perform mesh generation and material discretization using cylinrical geometry or Cartesian geometry. The CAD model is first discretized to an initial coarse unstructured mesh using the process of Delaunay triangulation and the advancing-front method, which is based on the geometry proprieties and predefined precision. Then the initial mesh can be further discretized using both structured and unstructured mesh. For structured mesh, by sweeping the mesh boundary lines which can both be generated semi-automatically and specified manually, the material property of each grid will be determined based on different strategies, making tradeoff between precision and efficiency. For the majority of domain, the lightweight multiple points method shown in Fig. 3 acts as the default method, which has better precision than the single-point method. For any important area specified by users such as the detector and absorber regions, the volume conservation method will be applied. To overcome both storage and performance limitations in resolving large scale problem, the patch-based domain partition algorithm is introduced in mesh generation, which will be discussed in the next section.

According to the pin-by-pin power distribution and mesh boundaries, the discretization of extraneous source is generated using a weighted method, which can support both cylindrical and Cartesian geometry. In the weighted method, each pin cell is divided into several small square working grids with the weights related to the pin power. Then the discretized source on each computing grid can be calculated by counting the total number of working grids owned and summing up their weights.

After mesh generation, a quality check is performed to see whether the differences in both material volume and total source are within user defined values. If so, the parallel S_N transport calculation is performed and the simulation results are output and analyzed in the postprocessing. Otherwise, the users can adjust the mesh boundaries and perform the above processes until the quality check is passed. All the processes in the mesh generation can be performed without performing the transport calculation by setting the option "use preview". Typically, the RPV model with millions of grids and thousands of bodies only requires tens of seconds on a PC.

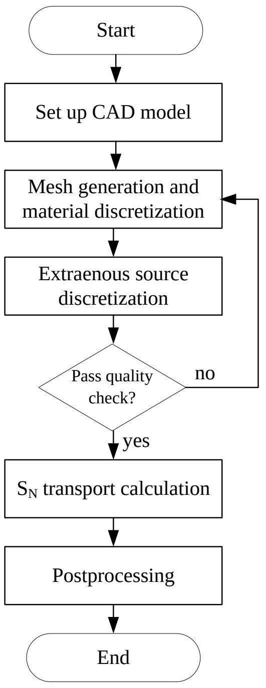
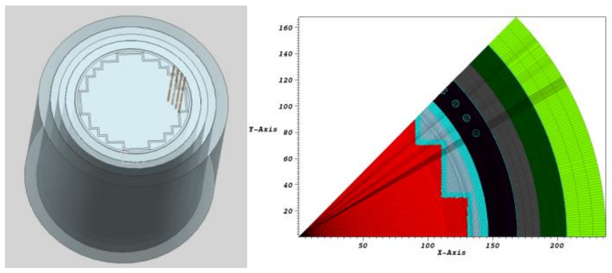
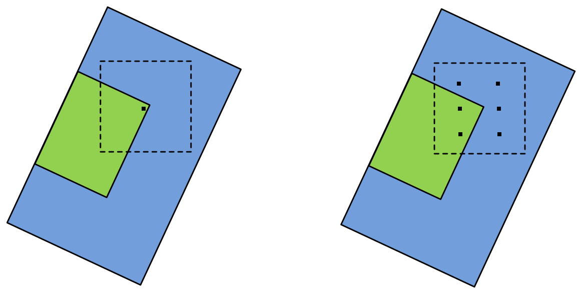

Start
Set up CAD model
Mesh generation and material discretization
Extraenous source discretization
Pass quality check?
no
yes
S_N transport calculation
Postprocessing
End

Fig. 1. program flowchart of JSNT using CAD modeling and mesh generation



(a) (b)

Fig. 2. a reactor pressure vessel radiation shielding model (a) full-core CAD model (b) eighth-core computing model after cylindrical mesh generation.



(a) (b)

Fig. 3. determination of material property during mesh generation using (a) the center point method and (b) the multi-points method. The colored regions denote different material zones and the dashed region refers to the mesh.

III. PARALLEL ALGORITHMS AND PATCH-BASED IMPLEMENTATIONS

1. Parallel Algorithms

JSNT mainly employs two parallel algorithms. The first is the domain partition algorithm, which is the fundamental algorithm for parallelizing the majority procedures such as mesh generation and acceleration methods [3]. The domain partition algorithm is based on the widely used BSP (Bulk Synchronize Parallel) model, which consists of a series of super steps and the computation on different subdomains can be done in parallel within each super step. Between two consecutive super steps, communication is performed if necessary and then followed by the synchronization. For the mesh generation, the discretized mesh domain is partitioned according to the domain partition algorithm and the work of material property determination as well as its storage requirement for the entire domain is shared among multiple processors.

The second is the DAG based data-driven algorithm for parallel sweeping [9-11], which was originally proposed for non-conforming structured and unstructured mesh because of the irregular data dependencies. As illustrated in Fig. 4, the parallel sweeping can be converted to a topological traversal on the directed graph. The DAG based algorithm can also be used for structured mesh, allowing for dynamic scheduling thus better load balancing, which shows great potential to utilize computing power of today's heterogeneous systems and that in the new era in a more efficient way [12, 13].

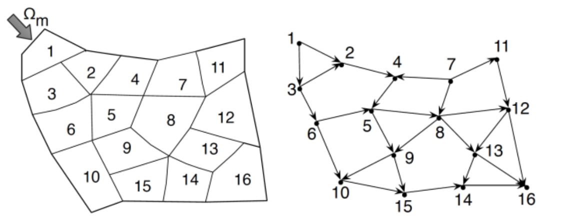


Fig. 4. the DAG based data-driven algorithm for parallel sweeping. For any given directions, the data dependency of grids can be depicted by DAG.

2. Patch-based Implementations

Both the implementations of the domain partition algorithm and the DAG based data-driven algorithm take advantage of a patch-centric data-driven abstraction. By using the abstraction, spatial subdomains are further divide into multiple logically rectangular regions named patch. To facilitate data communications, each patch box is extended to a ghost box within a specific width, which is filled with data transferred from adjacent patch boxes and physical boundary. For the domain partition algorithm, once the ghost boxes are filled, the data dependency between patches are removed and each patch is required to scheduled once to finish the computation. For the data-driven algorithm, each patch often requires to be scheduled to run arbitrary times, since each scheduling and computation on a patch is partial because the data dependency of grids is angular direction dependent.

The patch-based implementation method has several advantages: Firstly from a performance perspective, it allows for the improved cache hit ratio by adjusting patch size and more straightforward load balancing by redistributing patches among processors. Secondly, from the standpoint of software engineering, it follows the principle separation-of-concerns. To be specific, through the "Strategy" design pattern that can be implemented in any object-oriented programming languages, the patch-based implementations separates the user-specified computation on patch from mapping the computation to the particular parallel architecture. Then it allows collaboration between different develop teams, the complexity of computer architecture can be hidden and the productivity of software development can be substantially enhanced [2].

3. Optimizations

To do optimization on the parallel solution of the monogroup transport equation, there are three important factors are considered in our current implementation. The first one is to exploit more parallelism in the D2M (discrete-to-moment) computation. Traditionally while the M2D (moment-to-discrete) is a separate operation, the D2M is embedded in the sweeping to avoid the storage of directional flux. However, the parallelism in D2M is also limited by the sweeping even there is no data dependency both in spatial grids and angular directions. Thus, we separate the sweeping and the D2M, and then the two level parallelisms in D2M are fully exploited. The second one is to exploit parallelism in angular directions. For example, if the boundary condition is not reflective or periodic, the sweeping along all angular directions can be performed in fully parallel for Cartesian geometry and even for the cylindrical geometry there are still parallelism among angles at different η levels. The parallelism in directions is exploited by defining a pair of grid and angle instead of a grid, which acts a vertex in DAG scheduling algorithm. The third one is to develop optimal priority strategy for sweeping. We adopted the following strategy [14]: to advance as quickly as possible within each ordinate (avoiding jumps from one ordinate to another), and send the outgoing data to the processors that are waiting for. This is done by setting higher priority for angular directions rather than spatial grids. The forth one is to reduce the DAG overhead by vertex clustering and coarse graph constructing. For structured meshes, the DAG overhead induced for construction and scheduling can be large as compared to the traditional KBA algorithms and often exceeds numerical computation. Therefore, the DAG-based sweeping and the dynamical vertex clustering are only performed in the first iteration and the trace of vertex clustering is stored as a coarse graph, which can be reused in the later iterations.

IV. CODE VERIFICATION AND VALIDATION

The verification and validation of JSNT code without mesh generation has been performed using analytic solution, method of manufactured solutions, benchmark solutions and experimental data [15]. To demonstrate the correctness of JSNT code with mesh generation capability, the VENUS-3 neutron shielding benchmark [1] is used. As shown in Fig. 5-7, the CAD model was established and the spatial domain was discretized into 115×123×71 mesh using Cartesian geometry. Besides, a symmetrical quadrature set of 96 angular directions, a scattering expansion order of 3, 26 fast neutron groups out of 67 energy groups and a pointwise flux convergence criterion of 1.0E-4 were employed. For comparison, the model was run separately by using mesh generated from GGTM and JSNT. The mesh boundaries of JSNT are determined manually according to that mentioned in [1].

It was found that at 268 key flux positions the 26 groups scalar flux is in good agreement because the maximum relative difference is 8.40E-3. The equivalent fission flux deviations for ^{115}In(n,n’) dosimeters at all positions were also illustrated in Fig. 8, the results of which were included within 10% and the deviation of 5% was reached at approximately 90% of the positions for both run.

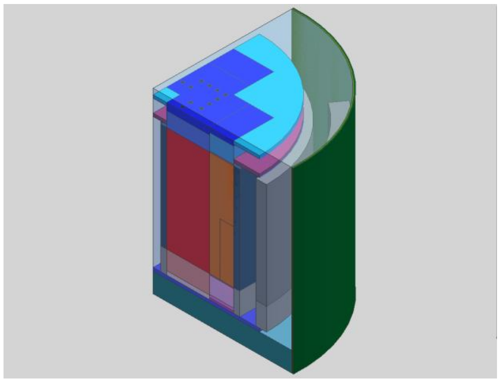


Fig. 5. the CAD model of VENUS-3 benchmark

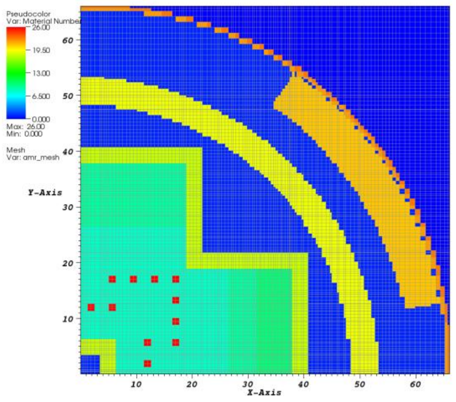


Fig. 6. discretized mesh and material of VENUS-3 benchmark in the (X, Y) plane, section at Z=106.50cm

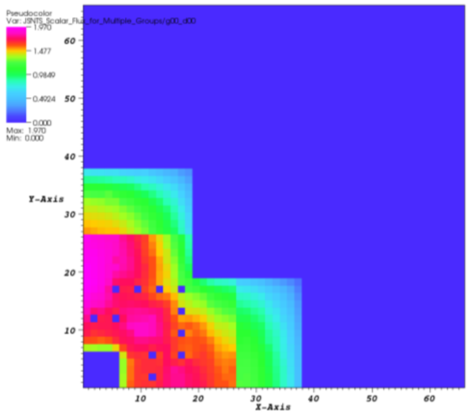


Fig. 7. the source distribution of energy group 1 of VENUS-3 benchmark in the (X, Y) plane, section at Z=131.50cm

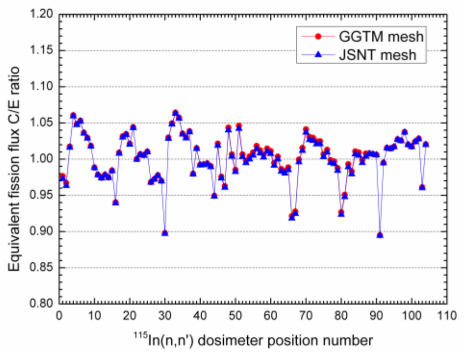


Fig. 8. the deviation of the simulation and experimental equivalent fission flux results at 104 ^{115}In(n,n’) dosimeters positions using Cartesian structured mesh generated from GGTM and JSNT. The C/E ration represents the ratio between the computational and experimental results.

V.PARALLEL PERFORMANCE AND SCALING

The parallel performance test was conducted using a reactor cavity leakage radiation shielding model of a commercial PWR in China (Fig. 9). The accurate modeling and simulation is of crucial importance for the lifetime evaluation of the equipments in the main reactor building. The test was run on the Tianhe-II supercomputer in China with the following configuration: each node is equipped with two Intel Xeon E5-2692v2 CPU and 64GB memory and nodes are interconnected using 40GB/s bandwidth Tianhe-Express-II network. The operating system is Kylin Linux and the code was compiled using the Intel 14.0 compiler and the customized MPI 3.0.

As illustrated in Fig. 10 and 11, the computational model was established using the eighth-core cylindrical geometry, $432\times137\times144$ spatial grids and a symmetrical quadrature set of 320 angular directions, which result in a total of 3.1 billion DOFs per group. The problem contains 172 energy groups and the order of scattering expansion was set to 3. The pointwise flux convergence criterion was set to 5.0E-3. As compared to the experimental data obtained from the three detectors located in the reactor tube (point E, F and G in Fig. 10), we found the computational results in thermal flux has been reduced to less than 20%, which improves the precision at least 2 times as compared to the traditional industry method.

Fig. 12 shows the parallel scaling curve obtained from the original code and the optimized one on 768 cores to 10080 cores with the vertex clustering size of 1000. From the figure, we found that the performance was dramatically improved by using the optimization mentioned in section III, which outperforms the original one by 2.3 to 3.0 times depending on the number of cores. Fig. 13 compares the ideal and observed parallel scaling curve and demonstrates the curve of its major components, such as the sweeping, the numerical computation (the D2M and M2D) and the initialization. We found that the numerical computation shows super linear scaling behavior from 768 to 6144 cores. The explanation of the phenomenon is that a better cache hit ratio was obtained in numerical computation (more data can be fitted into each processor's cache and thus better cache hit ratio) and the contention within intersocket memory bandwidth was mitigated (less data communicated among cores within single node) as the processor number increases. It was also found that the curve of the observed result is close to the ideal one at the beginning but the deviation becomes larger. There are three major limiting factors which are responsible for the performance degradation: Firstly, as the cell number assigned to each processor becomes smaller, consequently the communication to computation ratio becomes high, especially in the sweeping procedure. Secondly, the load imbalance becomes more serious as the processor number increases. For example, the load balance of the numerical computation has been dropped to 47.74%. Thirdly, though the mesh generation and material discretization can be fully in parallel in the initialization procedure, currently the initialization of extraneous source is still largely in sequential (read by the master processor and then scattered to multiple processors). Thus, as shown in Fig. 13, there is no performance improvement in the initialization as the processor number increases. Nevertheless, the overall parallel efficiency on 6144 and 10080 cores are 52.0% and 31.3% respectively.

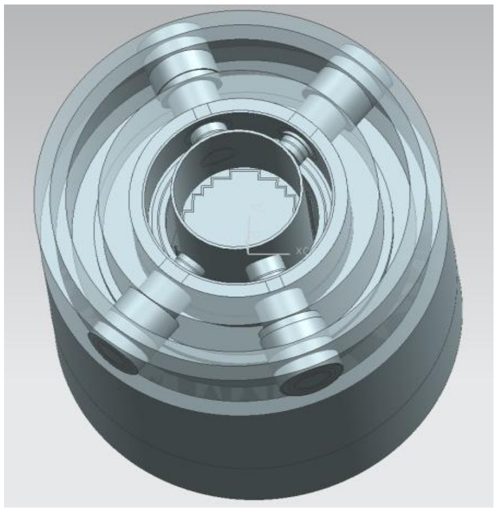


Fig. 9. the CAD model of a commercial PWR in China for reactor cavity leakage radiation shielding calculation.

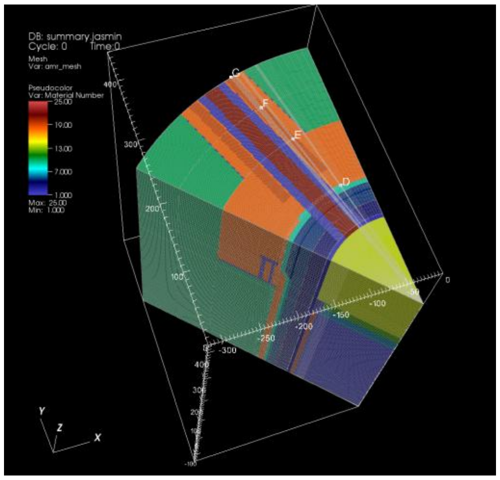


Fig. 10. the mesh and material discretization of the reactor cavity leakage radiation shielding model.

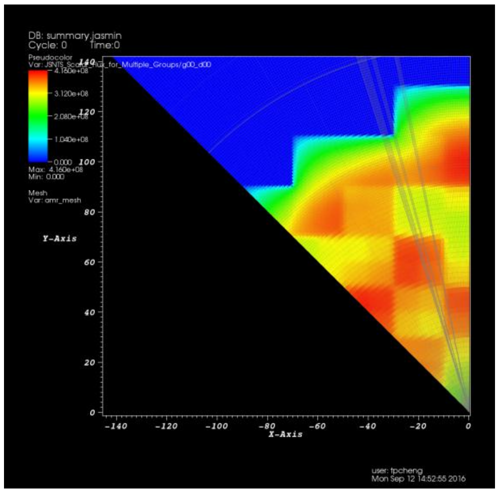


Fig. 11. the source distribution of energy group 1 of the reactor cavity leakage radiation shielding model.

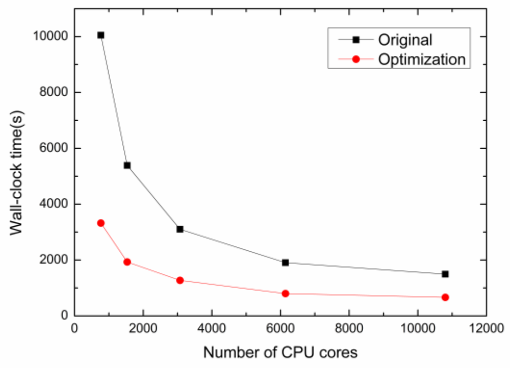


Fig. 12. the strong-scaling performance comparison between the original and optimized code. The baseline performance was obtained using 768 cores.

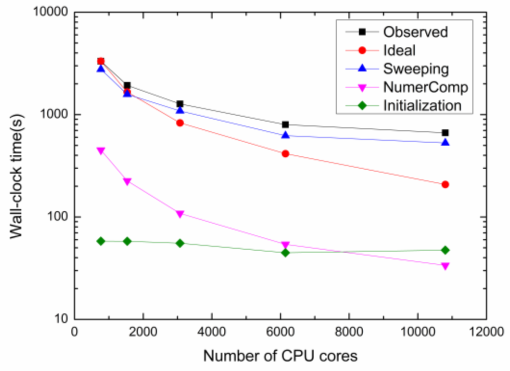


Fig. 13. the ideal and observed strong-scaling performance as well as that of major components. The NumerComp denotes the D2M and M2D computation.

VI. CONCLUSIONS

In this work, we have developed the large-scale mesh generation capability to JSNT and optimized its parallel algorithms and implementations for running on tens of thousands of processors. We have demonstrated the capability of mesh generation for complex problems, the parallel performance improvement and the benefit gained in precision by simulating a real-world reactor cavity leakage radiation shielding problem. Future work involves exploiting parallelism in energy, investigating the opportunity for vectorization and developing efficient iteration and acceleration algorithms for radiation shielding problems with strong scattering.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China under Grant No. 61602047 and 91430218, the Science Challenge Program under Grant No. JCKY2016212A502, the National Energy Administration of China under Grant No. 2015ZX06002008 and Special Program for Applied Research on Super Computation of the NSFC-Guangdong Joint Fund (the second phase). We would like to acknowledge the JASMIN development team from IAPCM for their help and constructive suggestions.

REFERENCES

1. M. Pescarini, R. Orsi, M. Borgia, T. Martinelli, “ENEA Nuclear Data Centre Neutron Transport Analysis of the VENUS-3 Shielding Benchmark Experiment”, KTSCG-00013, ENEA-Bologna (2001).
2. T. Cheng, J. Wei, H. Shen, B. Zhong, L. Deng, “Development of Parallel 3D Discrete Ordinates Transport Program on JASMIN Framework”, *Proc. 7ICMSNSE*, Ottawa, Ontario, Canada, Oct. 18–21, 2015, Canadian Nuclear Society (2015) (CD-ROM).
3. T. Cheng, Z. Mo, J. Wei, G. Zhang, H. Shen, L. Deng, “Acceleration of Discrete Ordinates Calculations using Parallel Partial Current Rebalance Algorithm and Algebraic Multigrid Solver”, *Proc. PHYSOR 2016*, Sun Valley, Idaho, May 1–5, 2016, American Nuclear Society (2016) (CD-ROM).
4. Z. Mo, A. Zhang, X. Cao, Q. Liu, X. Xu, H. An, W. Pei, S. Zhu, “JASMIN: a parallel software infrastructure for scientific computing”, *Frontiers of Computer Science in China*, 4, 4, 480-488 (2010).
5. G.G. Davidson, T.M. Evans, J.J. Jarrell, S.P. Hamilton, T.M. Pandy, R.N. Slaybaugh, “Massively parallel, three-dimensional transport solutions for the k-eigenvalue problem”, *Nuclear Science and Engineering*, 177, 2, 111-125 (2014).
6. D. Kaushik, M. Smith, A. Wollaber, B. Smith, A. Siegel, W.S. Yang, “Enabling high-fidelity neutron transport simulations on petascale architectures”, *Proc. SC*, Portland, Oregon, Nov. 14-20, 2009, IEEE (2009).

7. Y. Ma, Y. Fu, G. Qin, G. Li, L. Deng, “The Large-Scale Auto Modeling Tool for Monte Carlo Simulations”, *Proc. 7ICMSNSE*, Ottawa, Ontario, Canada, Oct. 18–21, 2015, Canadian Nuclear Society (2015) (CD-ROM).
8. L. Deng, G. Li, B. Zhang, D. Shangguan, Y. Ma, Z. Hu, Y. Fu, R. Li, X. Hu, T. Cheng, D. Shi, “JMCT Monte Carlo Simulation Analysis of Full Core PWR Pin-By-Pin and Shielding”, *Proc. 7ICMSNSE*, Ottawa, Ontario, Canada, Oct. 18–21, 2015, Canadian Nuclear Society (2015) (CD-ROM).
9. S. Plimpton, B. Hendrickson, S. Burns, W. McLendon, “Parallel algorithms for radiation transport on unstructured grids”, *Proc. SC*, Dallas, U.S., Nov., 2000, IEEE (2000).
10. S. Pautz, “An algorithm for parallel Sn sweeps on unstructured meshes”, *Nuclear Science and Engineering*, 140, 111-136 (2002).
11. Z. Mo, A. Zhang, G. Wittum, “Scalable heuristic algorithms for the parallel execution of data flow acyclic digraphs”, *SIAM Journal on Scientific Computing*, 31, 5, 3626-3642 (2009).
12. S. Moustafa, M. Faverge, L. Plagne, P. Ramet, “3D Cartesian Transport Sweep for Massively Parallel Architectures with PaRSEC”, *Proc. IPDPS*, Hyderabad, India, May 25–29, 2015, IEEE (2015).
13. G. Bosilca, A. Bouteiller, A. Danalis, M. Faverge, T. Herault, J.J. Dongarra, PaRSEC: Exploiting Heterogeneity to Enhance Scalability, *Computing in Science & Engineering*, 15, 6, 36-45 (2013).
14. G. Colomer, R. Borrell, F.X. Trias, I. Rodr ǵuez, Parallel algorithms for transport sweeps on unstructured meshes, *Journal of Computational Physics*, 232, 1, 118-135 (2013)..
15. G. Zhang, T. Cheng, G. Fu, L. Deng, “A Verification methodology and procedure of JSNT-S”, *Proc. PHYSOR 2016*, Sun Valley, Idaho, May 1–5, 2016, American Nuclear Society (2016) (CD-ROM).