7th International Conference on Modelling and Simulation in Nuclear Science and Engineering (7ICMSNSE)
Ottawa Marriott Hotel, Ottawa, Ontario, Canada, October 18-21, 2015

7ICMSNSE-16

Development of Parallel 3D Discrete Ordinates Transport Program on JASMIN Framework

T. Cheng[1,2], J. Wei[1], H. Shen[1], B. Zhong[1] and L. Deng[1]
[1] Institute of Applied Physics and Computational Mathematics, Beijing, China
[2] CAEP Software Center for High Performance Numerical Simulation, Beijing, China
cheng_tangpei@iapcm.ac.cn, wei_junxia@iapcm.ac.cn, shen_huayun@iapcm.ac.cn, zhong_bin@iapcm.ac.cn, deng_li@iapcm.ac.cn

Abstract

A parallel 3D discrete ordinates radiation transport code JSNT-S is developed, aiming at simulating real-world radiation shielding and reactor physics applications in a reasonable time. Through the patch-based domain partition algorithm, the memory requirement is shared among processors and a space-angle parallel sweeping algorithm is developed based on data-driven algorithm. Acceleration methods such as partial current rebalance are implemented. The correctness is proved through the VENUS-3 and other benchmark models. In the radiation shielding calculation of the Qinshan-II reactor pressure vessel model with 24.3 billion DoF, only 88 seconds is required and the overall parallel efficiency of 44% is achieved on 1536 CPU cores.

Keywords: Discrete Ordinates, Sweeping, Parallel Computing, Partial Current Rebalance, Radiation Shielding Calculation.

1. Introduction

The discrete ordinates (also known as S_N) codes have been succeed in nuclear reactor simulation over the past several decades. However, to solve real-world problems using three-dimensional models remains a challenge, owing to the tremendous computational efforts and storage requirement for solving the Boltzmann transport equation with six-dimensional phase-space. For instance, it generally requires solving the equation with $10^{10\text{-}12}$ DOF (degree-of-freedom) for the majority of today's engineering problems and $10^{17\text{-}21}$ DOF for high-fidelity simulations in the future [1]. For such cases, employing a massive parallel algorithm for S_N calculation is a necessity.

Early efforts on parallelizing S_N calculation can data back to the 1990s. Inspired by the ASCI program, many parallel sweeping algorithms (KBA algorithm [2] on Cartesian meshes and graph-based direct algorithm [3] on unstructured meshes) and parallel codes (PARTISN [4], PENTRAN [5], ATTILA [6], SWEEP3D [7] etc.) have been developed. Based on their sequential versions, these S_N codes are often parallelized through the use of traditional low-level MPI on high-performance computers. To get optimal performance, application developers must understand some basic properties of underlying architecture, parallel programming model in addition to physics models. This could be a daunting task and the situation becomes even worse in the context of today's ultra-scalable architectures and multidisciplinary and possibly distributed development teams. The challenge mainly arises from the following two increasing complexities. The first is the complexity in the evolving computer architecture. To fully exploit the potential of underlying architecture, application developers should pay excessive attentions to issues such as data-locality and memory access patterns in multilevel memory hierarchy. The second is the complexity in the evolving applications. To allow newly built algorithms/codes (for example, the linear algebraic solver) to coexist or incorporate the legacy S_N codes into a multiphysics environment, application developers should pay tremendous efforts in the development of "coupler tools" due to the diversity in implementations (such as data structures, interfaces etc).

For this reason, research on developing S_N codes on parallel-computing framework has received significant attentions in recent years, which is evidenced by the newly developed parallel programs such as Denovo [1], SN2ND [8] etc. Parallel-computing framework is built on top of traditional parallel programming model (such as MPI, OpenMP, OpenCL) with high-level abstraction. To hide parallel-computing details, framework encapsulates high performance implementation of data structures, data transfers, load balancing strategies and performance optimizations and provides flexible user interfaces. Through data structures and modules reuse to allow high performance computing, modularization and interface standardization to allow collaboration between different develop teams, the complexity of computer architecture can be hidden and the productivity of software development can be substantially enhanced.

Following the idea of developing applications on parallel-computing framework, this paper present our effort on developing JSNT-S on JASMIN (J Adaptive Structured Meshes applications Infrastructure) framework [9], aiming at modelling real-world radiation shielding and reactor physics applications in a reasonable time. JSNT-S is a 3D S_N multigroup radiation transport code under active development at CAEP-SCNS (Software Center for High Performance Numerical Simulation, China Academy of Engineering Physics). We parallelized the JSNT-S code largely through domain partition algorithm. We also implemented the space-angle parallelization scheme based on data-driven algorithm (a pipelining wavefront algorithm with privilege strategies in essential) for flux sweeping and developed acceleration schemes including PCR (Partial Current Rebalance) method with AMG (Algebraic Multigrid) solver [10] and several other methods. We demonstrated the accuracy and performance of JSNT-S through modelling three cases: the small LWR core benchmark *k-eigenvalue* problem [11], the VENUS-3 radiation shielding problem [12] and the shielding calculation of the RPV (Reactor Pressure Vessel) of QS-II (Qinshan-II) nuclear reactor.

2. Description of discrete ordinates transport

The basis for neutron transport simulation is the time-independent, multi-group, inhomogeneous Boltzmann transport equation, which is formulated as

$$\begin{aligned}&\nabla\cdot\boldsymbol{\Omega}\psi(\boldsymbol{r},E,\boldsymbol{\Omega})+\Sigma_{\mathrm{T}}(\boldsymbol{r},E,\boldsymbol{\Omega})\psi(\boldsymbol{r},E,\boldsymbol{\Omega})=\iint\Sigma_{\mathrm{S}}(\boldsymbol{r},E,E',\boldsymbol{\Omega},\boldsymbol{\Omega}')\psi(\boldsymbol{r},E',\boldsymbol{\Omega}')\mathrm{d}\boldsymbol{\Omega}'\mathrm{d}E'+\\&\tfrac{1}{4\pi}\chi(\boldsymbol{r},E)\iint\nu(\boldsymbol{r},E')\Sigma_{\mathrm{F}}(\boldsymbol{r},E')\psi(\boldsymbol{r},E',\boldsymbol{\Omega}')\mathrm{d}\boldsymbol{\Omega}'\mathrm{d}E'+Q(\boldsymbol{r},E,\boldsymbol{\Omega})\end{aligned} \tag{1}$$

Here, ψ is the directional flux at the spatial location $\boldsymbol{r}$, with energy E, traveling in direction $\boldsymbol{\Omega}$. Σ_{T}, Σ_{S} and Σ_{F} are the macroscopic cross sections for total interaction, scattering and fission. ν and χ are the total fission yield of secondary particles and the corresponding energy distribution. Q is the extraneous source.

Firstly, we use the discrete ordinates method to discretize the directional flux ψ into a specific set of quadrature points. Then we use the theta-weighted diamond difference approximation for spatial discretization and a multigroup formation to treat the energy dependence. To remove the implicitness caused by fission and scattering between groups, an iterative procedure termed "outer iteration" is applied. Within each outer iteration loop, a "flux iteration" or "inner iteration" is introduced to resolve the implicitness brought by scattering between directions within a single energy group. These are standard treatments which can be found in TORT and many other well-known discrete ordinates codes.

3. Parallel methodology and implementations

3.1 Program flow chart

The compute flow of JSNT-S program can be schematically outlined as in Figure 1, which is mainly composed of three phases and can be further divided into several procedures. The first phase is the data initialization, which reads and handles various data from user input, such as the problem control parameters, the geometry information, the cross section and the source specification, etc. For example, in the problem control parameters, both fixed source and *k-eigenvalue* search calculation, forward and adjoint modes are supported. In the geometry parameters, both Cartesian (*XYZ*) and cylindrical (*RΘZ*) geometry with non-uniform regular grids are supported, as well as several 2D subsets.

The primary computational phase is the S_N solve, which is divided into outer iterations over energy using the traditional Gauss-Seidel method and flux iterations over space-angle using the Richardson method (also known as the source iteration method). In the outer iterations, the extraneous source is input and the fission source is obtained from the flux moments at the beginning, and then followed by the outer iteration acceleration. After that, the group-to-group scattering source expanded in spherical harmonics is formed for each energy group. In the flux iterations, the self-scattering source within the same energy group is added to the total source. Once the total source is formed, flux sweeping calculation based on the data driven algorithm is performed. In the flux sweeping, multiple spatial discretization methods such as the theta-weighted diamond and the weighted diamond with flux-fixup can be used. Based on the flux obtained, the combination of multiple flux iteration acceleration algorithms are then performed, such as groupwise rebalance, damped partial current rebalance etc. Finally, the convergence of flux iteration is calculated by the maximum relative change in scalar flux; and the convergence of outer iteration is determined by the maximum relative change in both fission density and *k-eigenvalue*.

In the postprocessing phase, the overall performance is monitored by the JASMIN time manager and memory utilities, and then the finalization of parallel-computing environment and the result output. In the output data, *k-eigenvalue* and convergence information are organized in text format while the cellwise data such as material, scalar flux and response are organized in HDF5 format for data visualization and analysis.

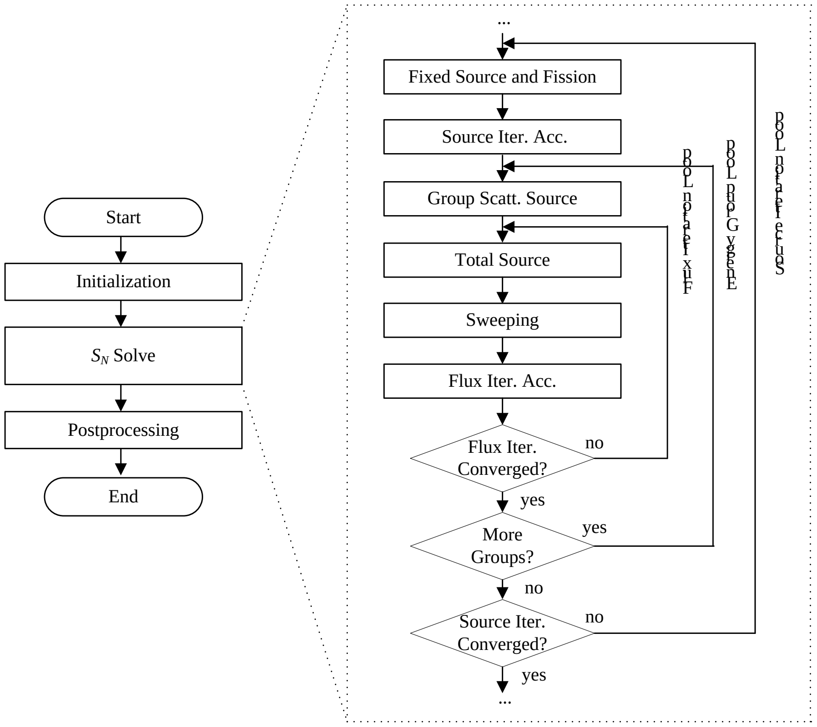


Figure 1 Overall program structure of JSNT-S

3.2 Domain partition algorithm

Developing a domain partition algorithm is a first, critical step for a successful parallel scheme. Essentially, domain partition algorithm is a divide-and-conquer method which decompose the spatial grid domain into several subdomains, each of which is handled by a separate processor. In our implementation, we further divide the spatial subdomains into multiple nonoverlapping, logically rectangular regions named patch (Figure 2), which is the fundamental element in each computation. For a given cache memory size, the size of patches can be adjusted to improve the cache hit ratio. In addition, the redistribution of patches can be used to achieve dynamic load balancing. To deal with data communication among processors, each patch box is extended to a ghost box (overlapped regions with other patches, illustrated in Figure 2) within a specific width, which is filled with data transferred from adjacent patch boxes and physical boundary.

As aforementioned, since the traditional Gauss-Seidel multigroup solver is used, only one group of data is stored in memory and solved at one time. Therefore, only a two-level decomposition on space-angle is used for the flux sweeping procedure (discussed in the section 3.3) and one-level decomposition on space for the other procedures, which may limits the parallel scalability using massive number of processors (see section 4.2). Though the Krylov multigroup solver (such as GMRES) is much more efficient at solving the upscatter groups and allows parallelism over energy in addition to space-angle [1], it has to spend extra memory (often more than one order), which could limit its application in the radiation shielding calculation when using a comparatively small number of processors. And the feasibility to develop multilevel decomposition on group-space-angle will be fully investigated in our extended work.

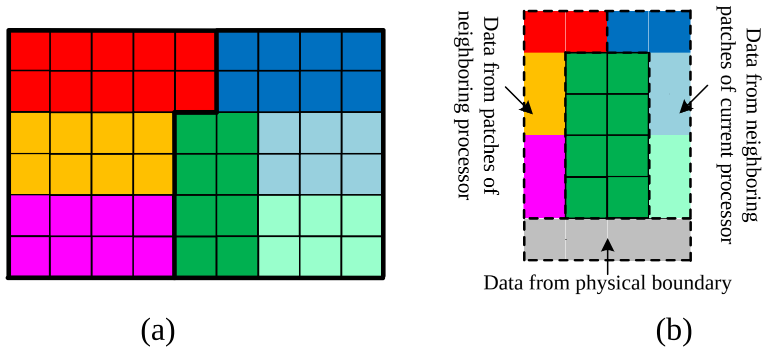

Data from patches of neighboring processor
Data from neighboring patches of current processor
Data from physical boundary
(a)
(b)

Figure 2 (a) a 2D domain with 9×6 grids is partitioned to 7 patches (represent by different colors), which are assigned to 2 processors (denote with bold dark boundaries); (b) ghost box (denotes with dashed boundaries) with one cell width for the green patch.

3.3 Sweeping based on data-driven algorithm

In general, flux sweeping is the procedure which requires the majority of computational time in S_N codes. To address the performance issue, a space-angle parallelization scheme based on data-driven algorithm [13, 14] is used in JSNT-S. The parallelization scheme takes a hierarchy design strategy, which hides parallelization details from the end users. As shown in Figure 3, it is composed of three layers from the bottom to top: the supporting layer, the interface layer and the application layer. The supporting layer mainly includes some important algorithms in parallel data-driven, such as Directed acyclic graph (DAG) generation, privilege strategies, pipelining wavefront algorithm etc. The supporting layer receives directed graph generated from the interface layer and then send the information on computable nodes back. The interface layer refers to the parallel sweeping integrator component, which converts the data dependency depicted in application layer to directed graph, and also translates the computable nodes received to computable grids for specific angular direction. The application layer only includes the implementation of specific spatial difference approximation (such as the theta-weighted diamond, the weighted diamond with flux-fixup etc.) on computable grids, without any necessary to be aware of parallel computing details.

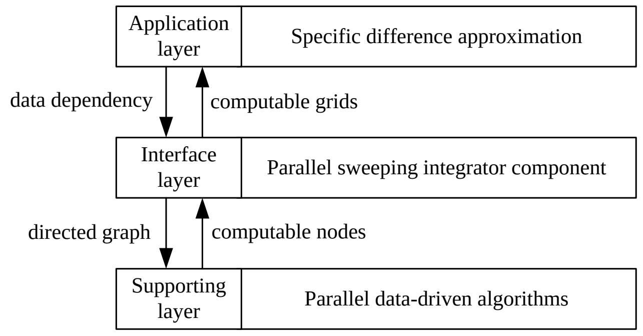


Figure 3 Hierarchy design of sweeping procedure based on data-driven algorithm

3.4 Partial-current rebalance acceleration algorithm

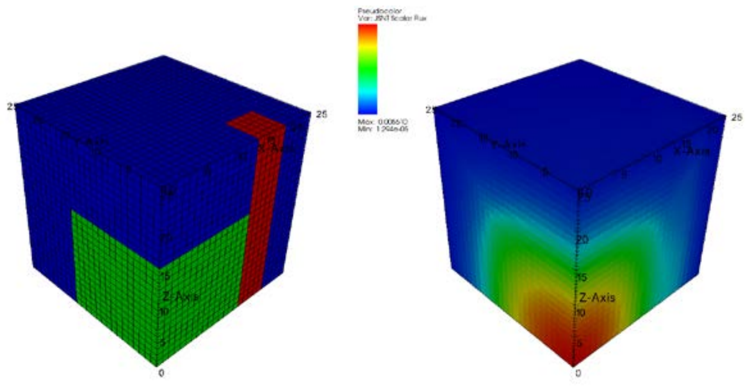
As aforementioned, we have implemented various acceleration algorithms for both source and flux iterations, among which the PCR method is one of the important methods in flux iteration acceleration. Although less sophisticated than some later procedures, it is robust and very effective, especially for the deep-penetration transport problems. In the PCR method, Eq. (1) is integrated over angular direction but not over space, which can give rise to a linear algebraic equation finally. Therefore, there are two major computational components in the PCR method: the formation and the resolve of the linear system. In formatting the system, computational steps involved are similar to that of the sequential algorithm except for some data communications, which are handled by using the ghost box method. In resolving the system, since the nature of the integrated equation is a diffusion equation, it is very suitable for multigrid method, which has better convergence rate than stationary iterative methods such as Jacobi, Gauss-Seidel, SOR (successive over-relaxation, the standard method in TORT) and exhibits great scalability in parallel computing. Therefore, a parallel AMG solver off-the-shelf in JASMIN is integrated in JSNT-S. The shortcoming of AMG method lies in its additional memory requirement. However, the problem is largely mitigated, since the acceleration equation has much reduced DOF (only one group and independent of angular direction) and the memory requirement is shared among processors when using the domain partition algorithm.

4. Verification, applications and performance evaluation

4.1 Code verification and validation

The verification and validation of JSNT-S program is carried out using both a code-to-code comparison (with TORT and Monte Carlo codes) and experimental data. Although we have performed verification tests on several benchmark models, only two of them are chosen here to demonstrate the correctness of our code for simulating different problems, namely, the *k-eigenvalue* search calculation and the fixed source calculation.

The first test case is taken from the small LWR core benchmark problem described in the NEACRP report L-330 [10]. As shown in Figure 4, the benchmark problem is a model of the Kyoto University Critical Assembly (KUCA) with 2 energy groups, 32 angles and 25×25×25 spatial grids (the reference mesh size is 1cm×1cm×1cm). There are two cases considered in this problem: the control rod position is empty (void) and the control rod is inserted. Table 1 gives the *k-eigenvalue* obtained from different codes, from which we found a good agreement between JSNT-S and other codes for both cases.



(a) (b)

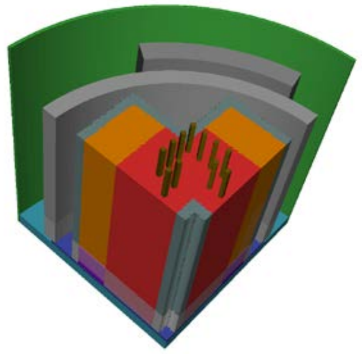
Figure 4 The small LWR core benchmark model: (a) The mesh discretization and material distribution, (b) the scalar flux distribution.

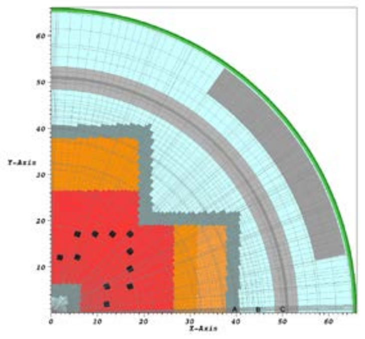
Table 1 Comparisons of *k-eigenvalue* obtained from different codes

Codes	Case 1	Case 2
Monte-Carlo [10]	0.9778±0.0005	0.9624±0.0005
TORT(S4)	0.97676	0.96206
JSNT-S(S4)	0.97676	0.96206

The second test case is the VENUS-3 benchmark model taken from [12]. The experimental configuration of this model was made to be representative of typical irradiation conditions of a modern PWR vessel. Since the model calculation was dedicated to obtain the target quantities for the fast neutron fluxes (neutron energy > 0.1 MeV) and the total iron displacement rates per atom, a fixed source calculation were performed using the first 26 fast neutron groups (down to the lower energy limit of 0.111 MeV) of the BUGLE-96 cross section library [12], the S8 order in the flux angular discretization (96 angles) and the P3 order in the expansion of scattering cross section. As depicted in Figure 5, the cylindrical geometry and a spatial discretization of 111×113×71 were employed. The theta-weighted difference approximation was selected for the flux extrapolation and the pointwise flux convergence criterion was set to 1.0E-04. The neutron source distribution, given by OECD/NEA to perform the VENUS-3 benchmark calculations, is expressed in units of fissions per pin per second, arbitrarily normalized to a core averaged power of one fission per second per active fuel pin.

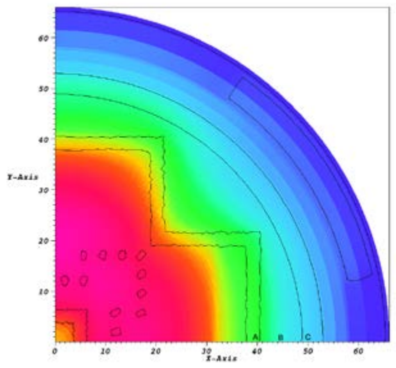
In the VENUS-3 experiment there are 386 dosimeters, which are placed at 268 different spatial locations. The first investigation is to compare the fast neutron flux above 0.1 MeV at different spatial locations. Figure 6(a) demonstrates the azimuthal distribution of the fast neutron flux obtained from JSNT-S; and Figure 6(b), (c) and (d) compare the average fast neutron flux at different dosimeter locations by using TORT and JSNT-S, from which we found a good agreement. In addition, we also investigated the deviation of the equivalent fission fluxes between the computational results and experimental data for different types of dosimeters at different regions. Figure 7 illustrates the deviation of the 115In(n,n’) dosimeters at 104 positions (the inner and outer baffle, the water gap and the barrel) obtained from JSNT-S and NEA/OECD using TORT 3.2 [12]. For both codes, we found that the equivalent fission flux deviations at all positions were included within 10% and the deviation of 5% was reached at approximately 90% of the positions.

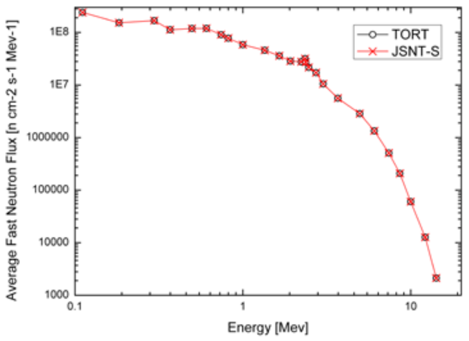

(a)



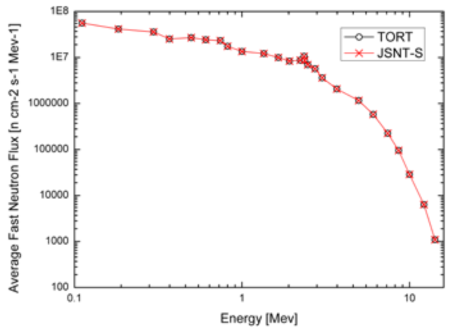
(b)

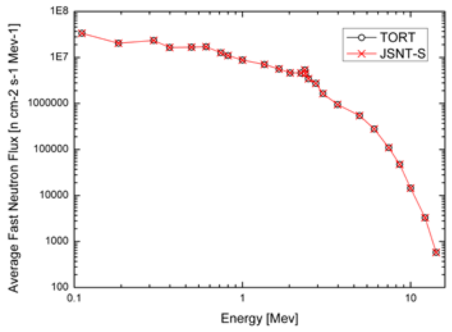
Figure 5 Geometry specification for the VENUS-3 benchmark: (a) Core region and steel zones, (b) Horizontal meshes in the (R,θ) plane, section at Z =106.50cm.





(a) (b)





(c) (d)

Figure 6 (a) Azimuthal distribution of the fast neutron flux above 0.1 MeV, section at Z =106.50cm; (b), (c) and (d): Comparison of average fast neutron flux for different energy groups at the dosimeter location A, B and C in (a). location A, B and C at the outer baffle (39.69, 0.69, 106.50), the water gap (44.73, 0.63, 106.50) and the barrel (49.77, 0.63, 106.50), respectively.

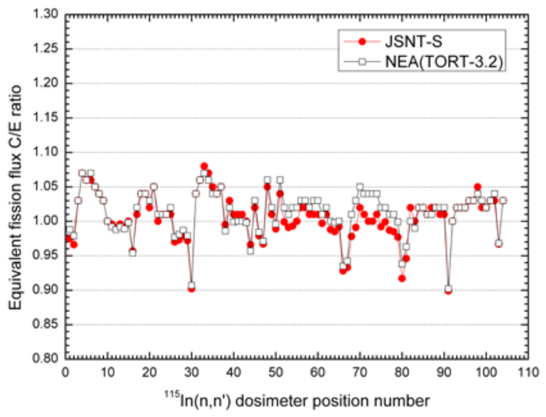
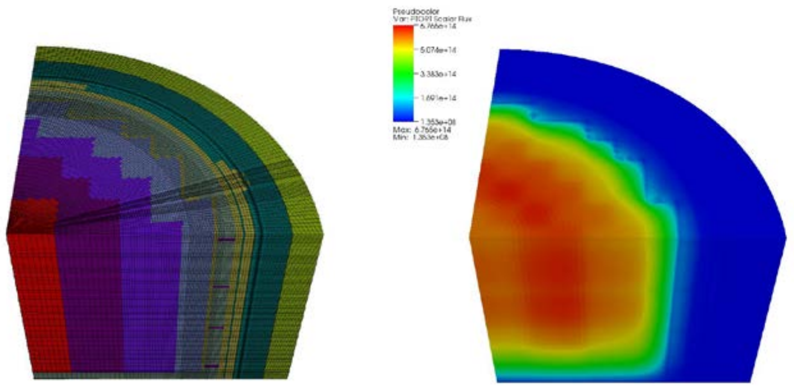


Figure 7 the deviation of the simulation and experimental equivalent fission flux results at 104 ^{115}In(n,n’) dosimeters positions. The C/E ration represents the ratio between the computational and experimental results.

4.2 QS-II shielding calculation

To investigate the performance in simulating real-world applications, we performed the shielding calculation of the RPV of QS-II nuclear reactor in China. The shielding model was spatially discretized into 139×106×181 using the cylindrical geometry and its material distribution is shown in Figure 8. The problem contains 29 energy groups, with 28 neutron groups and 1 photon group, and was executed using a quadrature set of 320 angels. The order of scattering expansion was set to 3 and the theta-weighted diamond difference spatial discretization was used, which result in a total of 24.3 billion DOF. The pointwise flux convergence criterion was set to 5.0E-3.



(a) (b)

Figure 8 The QS-II RPV shielding model (cut in half). (a) The mesh discretization and material distribution, (b) the scalar flux distribution.

To investigate the scalability of the code, we performed the strong scaling test (to evaluate solution time variability with the number of processors for a fixed problem size). In this test, we solved the shielding model using 24, 48, 96, 192, 384, 768 and 1536 processors respectively on a petascale supercomputer. Figure 9 shows both the observed and theoretical parallel scaling. The observed execution time decreases dramatically as the processor number increases. Especially, the observed curve is below the ideal one (relative to 24 processors) when the processor number is less equal than 192, which denotes the “supper linear speedup”. The supper linear speedup occurs because as the processor number increases, more data can be fitted into each processor’s cache and thus better cache hit ratio can be achieved in numerical computations. Besides, it is observed that there is a slight deviation between the observed and ideal scaling curve when the processor number increases from 192 to 1536. This phenomenon can be mainly attributed to three factors. Firstly, the performance of sweeping drops gradually, which dominants the overall execution time. Two reasons are responsible: one is the communication to computation ratio increases substantially and therefore somewhat decrease the efficiency; the other is the aforementioned “cache effect” becomes less obvious. Secondly, load balancing in sweeping and other numerical steps becomes much harder as the processor number increases. Thirdly, there are almost no change in execution times of initialization due to its inherently sequential nature (read by the master processor and then broadcast). As pointed out by the Amdahl’s Law, the overall performance improvement is limited to such non-parallelizable component and parallel I/O will be considered in our extended work. Though the parallel performance can be limited by the above three factors, the overall efficiency is still impressive for such real-world application, which can reach up to 84% on 768 processors and 66% on 1536 processors.

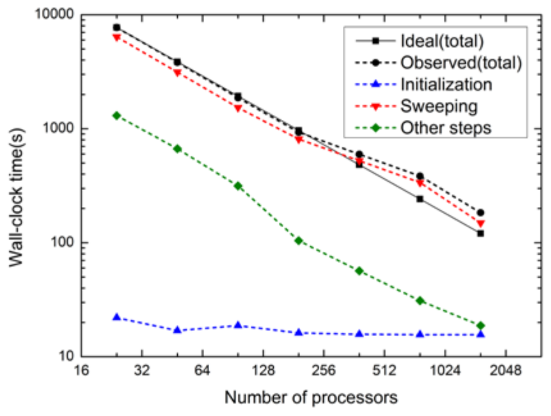


Figure 9 Parallel strong scaling observed from 24 to 1536 processors for the QS-II RPV shielding model. As compared to the performance obtained on 24 processors, the ideal parallel scaling slope denotes that the decrease in program execution time should be in proportion to the increase in processor number.

To further improve performance, we enabled the proposed parallel PCR acceleration algorithm. Figure 10 compares the overall performance with and without the acceleration algorithm when using different number of processors, and gives the execution time of the two major steps (sweeping and PCR acceleration) in the accelerated program. One can see that there is a substantial performance improvement when using the PCR acceleration algorithm. This is largely because that the iteration number drops from 717 to 201, which is independent of the processor number. However, the performance improvement becomes less obvious when the number of processors increases. For example, the ratio drops from 3.1 on 24 processors to 2.1 on 1536 processors. This can be attributed to the relatively poor parallel efficiency obtained from the accelerated program when using large number of processors. The overall parallel performance is limited by the PCR algorithm, which is evidenced in Figure 10 that there is almost no performance improvement in the PCR algorithm from 768 to 1536 processors. This phenomenon is owing to the poor efficiency in resolving the scalar balance equation in the PCR algorithm. Since the equation to be solved is resulted from integrating Equation (1) over direction, it has much reduced DOF. For example, in this case the average number of cells assigned to each processor is less than 2000 using 1536 processors, which is comparatively small. As a consequence, the communication to computation ratio becomes high and the parallel program suffers significant performance degradation. Nevertheless, the joint use of parallel computing and the PCR acceleration algorithm reduces the simulation time to 88 seconds, and the overall parallel efficiency achieved is 53% on 768 processors and 44% on 1536 processors.

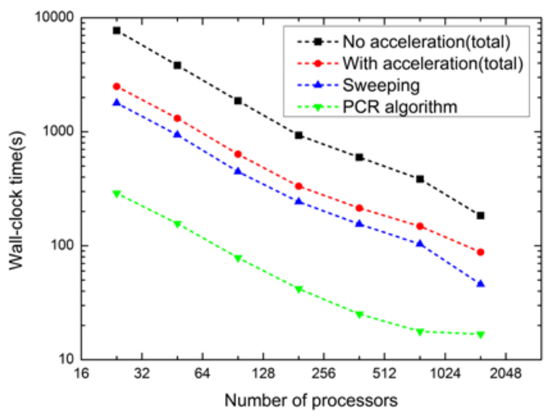


Figure 10 Overall execution time with and without the acceleration algorithm and the execution time for sweeping and PCR algorithm when using different number of processors. The execution time for the one with acceleration largely equals to the summation of time spent on sweeping, PCR algorithm and initialization.

5. Conclusions and future works

In this work, we have developed a parallel radiation transport code JSNT-S on JASMIN framework. We have demonstrated its potential to utilize large-scale parallel machines and the ability that allows simulating real-world radiation shielding and reactor physics applications in a reasonable time through the QS-II RPV radiation shielding model with 24.3 billion DOF. With the increasing availability of parallel hardware, it provides a tremendous opportunity to utilize JSNT-S in accelerating today's engineering applications and high-fidelity problems in the future. At the same time, we are still dedicating to further improve the performance and usability of current JSNT-S code. As pointed out above, we have been working on the investigation of more efficient solver for the inner iterations, developing features such as parallel I/O, visualized modelling and automatic mesh generation etc. Moreover, to further exploit parallelism and take full advantage of ultra-scale supercomputers with more 100,000 CPU cores, we plan to investigate the feasibility to add energy decomposition and develop new iteration algorithms (such as Krylov subspace method).

6. Acknowledgement

This research is supported by the National Natural Science Foundation of China under Grant No. 91430218 and 11201036, the National Energy Administration of China under Grant No. 2015ZX06002008 and the State Administration of Science, Technology and Industry for National Defense, PRC under Grant No. [2012]1523.

7. References

[1] 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*, Vol. 177, No. 2, 2014, pp. 111-125.

[2] R.S. Baker and K.R. Koch, “An Sn algorithm for the massively parallel CM-200 computer”, *Nuclear Science and Engineering,* Vol. 128, No. 3, 1998, pp. 312-320.

[3] S.D. Pautz, “An algorithm for parallel Sn sweeps on unstructured meshes”, *Nuclear Science and Engineering,* Vol. 140, No. 2, 2002, pp. 111-136.

[4] R.E. Alcouffe, R.S. Baker, J.A. Dahl, S.A. Turner, R. Ward, “PARTISN: A time-dependent, parallel neutral particle transport code system”, Los Alamos National Laboratory, LA-UR-05-3925 (May 2005), 2005.

[5] G. Sjoden, A. Haghighatm, “PENTRAN-A 3-D cartesian parallel SN code with angular, energy, and spatial decomposition”, Proceedings of the Joint International Conference on Mathematical Methods and Supercomputing in Nuclear Applications, Vol. 2, 1997, pp. 1267-1276.

[6] J.E. Morel, “3-D DETERMINISTIC TRANSPORT METHODS RESEARCH AT LANL UNDER ASCI”, Los Alamos National Lab., NM (US) , 2000.

[7] SWEEP3D: 3D Discrete Ordinates Neutron Transport Benchmark Codes, http://www.llnl.gov/asci_benchmarks/asci/limited/sweep3d/sweep3d_readme.html.

[8] D. Kaushik, M. Smith, A. Wollaber, B. Smith, A. Siegel, W.S. Yang, “Enabling high-fidelity neutron transport simulations on petascale architectures”, High Performance Computing Networking, Storage and Analysis, Proceedings of the Conference on, IEEE, 2009.

[9] 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*, Vol. 4, No. 4, 2010, pp. 480-488.

[10] Z. Mo, X. Xu, “Relaxed RS0 or CLJP coarsening strategy for parallel AMG”, *Parallel Computing*, Vol. 33, No. 3, 2007, pp. 174-185.

[11] T. Takeda, H. Ikeda, “3-D neutron transport benchmarks”, *Journal of Nuclear Science and Technology,* Vol. 28, No. 7, 1991, pp. 656-669.

[12] 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.

[13] Z. Mo, A. Zhang, G. Wittum, “Scalable heuristic algorithms for the parallel execution of data flow acyclic digraphs”, *SIAM Journal on Scientific Computing*, Vol. 31, No. 5, 2009, pp. 3626-3642.

[14] Z. Mo, A. Zhang, Z. Yang, “A new parallel algorithm for vertex priorities of data flow acyclic digraphs”, *The Journal of Supercomputing*, Vol. 68, No. 1, 2014, pp. 49-64.