Hindawi
Science and Technology of Nuclear Installations
Volume 2021, Article ID 5549602, 16 pages
https://doi.org/10.1155/2021/5549602

Research Article

An Efficient Scheme for Coupling OpenMC and FLUENT with Adaptive Load Balancing

Qingyang Zhang,[1,2] Tianji Peng,[3,4] Guangchun Zhang,[1,2] Jie Liu,[1,2] Xiaowei Guo,[1,2] Chunye Gong,[1,2] Bo Yang,[1,2] and Xukai Fan[3]

[1]*National University of Defense Technology, Science and Technology on Parallel and Distributed Processing Laboratory, Changsha 410073, China*
[2]*National University of Defense Technology, Laboratory of Software Engineering for Complex Systems, Changsha 410073, China*
[3]*Chinese Academy of Sciences, Institute of Modern Physics, Lanzhou 730030, China*
[4]*University of Chinese Academy of Sciences, School of Nuclear Science and Technology, Beijing 100049, China*

Correspondence should be addressed to Jie Liu; liujie@nudt.edu.cn

Received 18 February 2021; Revised 6 August 2021; Accepted 31 August 2021; Published 24 September 2021

Academic Editor: Guglielmo Lomonaco

This paper develops a multi-physics interface code MC-FLUENT to couple the Monte Carlo code OpenMC with the commercial computational fluid dynamics code ANSYS FLUENT. The implementations and parallel performances of block Gauss–Seidel-type and block Jacobi-type Picard iterative algorithms have been investigated. In addition, this paper introduces two adaptive load-balancing algorithms into the neutronics and thermal-hydraulics coupled simulation to reduce the time cost of computation. Considering that the different scalability of OpenMC and FLUENT limits the performance of block Gauss–Seidel algorithm, an adaptive load-balancing algorithm that can increase the number of nodes dynamically is proposed to improve its efficiency. Moreover, with the natural parallelism of block Jacobi algorithm, another adaptive load-balancing algorithm is proposed to improve its performance. A 3 x 3 PWR fuel pin model and a 1000 MWt ABR metallic benchmark core were used to compare the performances of the two algorithms and verify the effectiveness of the two adaptive load-balancing algorithms. The results show that the adaptive load-balancing algorithms proposed in this paper can greatly improve the computing efficiency of block Jacobi algorithm and improve the performance of block Gauss–Seidel algorithm when the number of nodes is large. In addition, the adaptive load-balancing algorithms are especially effective when a case demands different computational power of OpenMC and FLUENT.

1. Introduction

Although nuclear energy has been contributing to sustainable development goals for decades, its economic feasibility and safety remain the major issues widely concerned [1, 2]. Meanwhile, computer science (CS) and information and communications technologies (ICTs) have also demonstrated their key roles in environmental sustainability [3, 4]. Combining CS and ICTs with state-of-the-art physics codes is a cutting edge research in the field of engineering and scientific simulations [5–7]. Existing studies show that the use of high-performance computers to perform high accuracy multi-physics simulations can reduce design costs and thereby enhance the safety and economic feasibility of nuclear energy [7, 8].

Inherent feedback mechanisms in reactors necessitate a multi-physics approach, particularly between neutronics and thermal-hydraulics [9, 10]. Numerous neutronics and thermal-hydraulics coupling research studies have been done with simplified physical models, such as nodal diffusion theory and subchannel or single-channel thermal hydraulic codes [11–14]. Although such models are effective, the assumptions made in their formulation limit the accuracy of solution. Consequently, three-dimensional (3D) neutron transport codes and computational fluid dynamics (CFD) codes have received significant attention because of their detailed model and accurate solutions. Bousbia-Salah et al. discussed the main features and limitations of coupled code techniques in 2007 [15]. They pointed out that the capabilities of computers should be properly utilized and summed up two schemes that can be utilized when performing the coupling. One is the internal coupling, which provides better convergence but still faces some numerical challenges [16]. The other is the external coupling. This scheme allows neutron transport codes and CFD codes to run separately and exchange data through interface codes [17], and most importantly, it is efficient on massive parallel computer systems [18]. Because of the minimal changes to individual codes, many researchers choose to use the external coupling scheme for multi-physics coupled work [19]. Marzano discussed about the methodology and used an interface code to exchange data between the individual codes for coupling 3D neutron transport calculation and full-field CFD calculation [10]. In the coupled calculation procedures of Marzano's methodology, PENTRAN and STAR-CCM + are calculated in turn during each iteration. Hsingtzu and Rizwan-uddin verified the integrated tight coupling method and implemented it using DRAGON and OpenFOAM and adopted a similar iterative method. Since then, more and more researchers have applied this method, which is known as block Gauss–Seidel-type Picard iteration, in coupled simulation [20]. Gurecky and Schneider coupled MCNP and FLUENT with this method [21]. In 2017, Daeubler et al. developed a CFD/Monte Carlo coupling scheme with this iteration method and applied it to TRIGA reactor [14]. More related works can be found in [16, 22–26]. It seems that the current research studies on coupling 3D neutron transport codes and CFD codes rely on the use of a block Gauss–Seidel-type Picard iteration.

It is well known that coupled studies based on 3D neutron transport codes and CFD codes demand a lot of computational power, and thus high-performance computers are usually needed. Block Gauss–Seidel-type Picard iteration offers a simple path to couple different physics codes with minimal code interaction required. However, because the physics codes have hardly been modified, their parallel performances and scalability are usually different. This means that there may be load-balancing problems when applying it to high-performance computers. Moreover, Kelley studied iterative methods for linear and nonlinear equations and mentioned block Jacobi algorithm [27]. This method seems naturally parallelized since each block of the iteration can be performed independently of the others. Cervera et al. researched the computing efficiency and implementation of block-iterative algorithms for nonlinear coupled problems and pointed out that block Jacobi algorithm can be satisfactory when the problems to be solved are strongly nonlinear on their own [28]. Matthies et al. took block Jacobi method into account very naturally when mentioned about block Gauss–Seidel methods while considering the algorithms for computing the response of a coupled problem [29]. However, as far as we know, few researchers have applied the block Jacobi method on coupling 3D neutron transport codes and CFD codes.

The primary objective of the present research is to introduce computer technology into the neutronics and thermal-hydraulics coupled program to improve its performance in massively parallel computing so as to reduce the time cost of simulation. Firstly, we develop a multi-physics interface code MC-FLUENT to couple the Monte Carlo code OpenMC with the commercial computational fluid dynamics code ANSYS FLUENT. Then, we propose an adaptive load-balancing algorithm to improve the computing efficiency of block Gauss–Seidel-type Picard iterative method on high-performance computers. In addition, although some scholars believe that the block Jacobi method has a poor convergence [27, 28], its natural parallelism makes it more suitable in high-performance computers. The implementation and performance of the block Jacobi-type Picard iterative method have been studied in this paper, and another adaptive load-balancing algorithm is proposed to improve its performance.

The paper is organised as follows. Section 2 describes the codes used for coupling. Section 3 is devoted to introducing the coupling schemes, adaptive load-balancing algorithms, and data exchange method. Section 4 focuses on the models simulated in this paper. Section 5 contains numerical results and discussion. Section 6 presents the conclusion and the future working plan.

2. Code Descriptions

OpenMC is an open source Monte Carlo transport code that uses the portable and more "readable" XML file format instead of the arbitrary format ASCII file for input, which helps to increase the efficiency of data exchange between programs and interfaces more [30]. HDF5 files are used to store continuous-energy particle interaction, which can be converted from ACE files produced by NJOY. High-performance parallel algorithms in OpenMC are enabled via a MPI and OpenMP programming model [31]. These features of OpenMC provide a lot of convenience for the code to couple with others. Equally important, ANSYS FLUENT is a popular commercial CFD code which can achieve the best convergence speed and solution accuracy [32]. FLUENT can improve the performance of the solver by linking the user-defined function (UDF) dynamically. UDF allows users to write information to a case or read information from a data file and extend the initialization and postprocessing capabilities of the solver. Moreover, a journal file which contains TUI and/or scheme commands allows FLUENT to automate setup process for the tasks within the same model and control the solving behavior of a model in addition to the setup. The diverse numerical models and flexible user-defined functions of FLUENT allow it to work well with other software.

On account of the functional properties of the aforementioned two codes, an interface code named MC-FLUENT has been developed to couple OpenMC and ANSYS FLUENT in order to study the implementation and optimization method of iterative algorithms in 3D neutron transport code and CFD code coupling. Written in C++ language, the code can submit the running instructions of OpenMC and ANSYS FLUENT to Linux system and Slurm Workload Manager through shell script. A fixed-format input card containing problem information and iterative information must be filled out before running MC-FLUENT. Through this input card, the interface code can obtain the corresponding relationship between OpenMC cells and FLUENT grid domain. Then, MC-FLUENT enabled to collect the cells and materials information by parsing XML format files. After each iteration, the interface code will be able to extract the temperature and density of materials from the output file of FLUENT and modify the material parameters in XML format files. Meanwhile, it can also extract the fission rate distribution calculated by OpenMC to calculate the power density and write it into UDF files. Moreover, MC-FLUENT can adaptively balance the load through dynamic scheduling compute nodes according to the time consumed by each iteration. The specific coupling format and adaptive load-balancing algorithms will be introduced in Section 3. The code procedure with block Gauss–Seidel-type Picard algorithm and block Jacobi-type Picard algorithm is shown in Figures 1 and 2, respectively. Anyone can download the source code for this interface by connecting to https://github.com/DHMephisto/MC-FLUENT.

3. Methodology

In this section, we investigate the method for coupling OpenMC and FLUENT. The coupling schemes, adaptive load-balancing algorithms, and data exchange strategy are proposed.

3.1. Coupling Scheme. The coupling framework in this paper is designed by the external coupling method, which solves physical fields by passing data repeatedly between independent codes. Two types of Picard iterative algorithms are used to calculate the coupled problems.

3.1.1. Block Gauss–Seidel-Type Picard Scheme. The first one is known as block Gauss–Seidel-type Picard iteration method. The method helps to execute any two codes being adopted in MC/CFD coupling in a serial way. During each iteration, OpenMC first calculates a new fission rate distribution (ψ) according to the density (ρ) and temperature (T) calculated in the previous iteration. Then, MC-FLUENT will be responsible for data passing. It can calculate the new source term (S) of energy equation at each domain according to the new fission rate distribution and write the power density to the UDF files. After that, FLUENT will calculate a new density distribution and a temperature distribution according to the new power density. At last, MC-FLUENT will rewrite the XML files with new temperature and density. The simulation is stopped according to the criterion shown below:

$$\varepsilon = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\varphi_i^n - \varphi_i^{n-1}\right)^2}, \qquad (1)$$

where φ is variable of interest (in this paper, temperature and power density), i is the ith cell, N is the number of cells, and n is the nth iteration. When ε reaches a certain value, the algorithm is stopped.

The outline of block Gauss–Seidel-type Picard coupling algorithm is depicted in Algorithm 1 [21, 28, 33]. The superscript k of the letters means that this is the result of the kth iteration. With this scheme, the second physics (FLUENT in this paper) works with a more up-to-date linearization of the coupling term as compared to block Jacobi algorithm, which tends to accelerate convergence. However, it does not seem to be friendly to parallel processing. Additionally, the scalability of OpenMC and FLUENT dealing with one same coupling problem can differ from each other, which means that when the number of computing nodes increases, the computing efficiency of one code may start to decline while that of another code keeps on improving. In view of this, an adaptive load-balancing algorithm is proposed by present research to improve the efficiency of the coupled program. The procedure of MC-FLUENT with this scheme is presented in Figure 1.

3.1.2. Block Jacobi-Type Picard Scheme. The second alternative, block Jacobi-type Picard scheme, can deal with OpenMC and FLUENT in a parallel manner. During each iteration, OpenMC and FLUENT are calculating with the coupling terms updated in the last iteration at the same time. MC-FLUENT only needs to collect the results and update the coupling items after each iteration. The algorithm is also stopped according to the criterion shown in equation (1). The outline of the block Jacobi-type Picard coupling algorithm is shown in Algorithm 2 [28]. The superscript k-1 indicates that neither of the two codes needs the results calculated by each other at current iteration step. In this case, both codes would be running in parallel and fully exploiting the computational resources.

The procedure of MC-FLUENT with block Jacobi-type Picard algorithm is presented in Figure 2. With this scheme, OpenMC and FLUENT will be running in a parallel manner and each code will use half of the available compute nodes. Nevertheless, the computing efficiency of OpenMC and FLUENT dealing with the same coupling problems differs from each other. The equal number of compute nodes is not the optimal solution, which may lead to longer waiting time. Therefore, the resource allocation strategy can exert a great influence on the computational performance of block Jacobi scheme. To increase computing efficiency, an adaptive load-balancing algorithm is proposed by present research. Based on the respective calculating time of the two codes in the last iteration, this algorithm is able to make dynamic adjustments to their respective number of nodes in order to reduce waiting time and improve overall computational performance of the algorithm.

3.2. The Adaptive Load-Balancing Algorithms for MC and CFD Coupling Simulations. According to the shortcomings of the two algorithms stated in Section 3.1, two adaptive load-balancing algorithms are proposed in this paper.

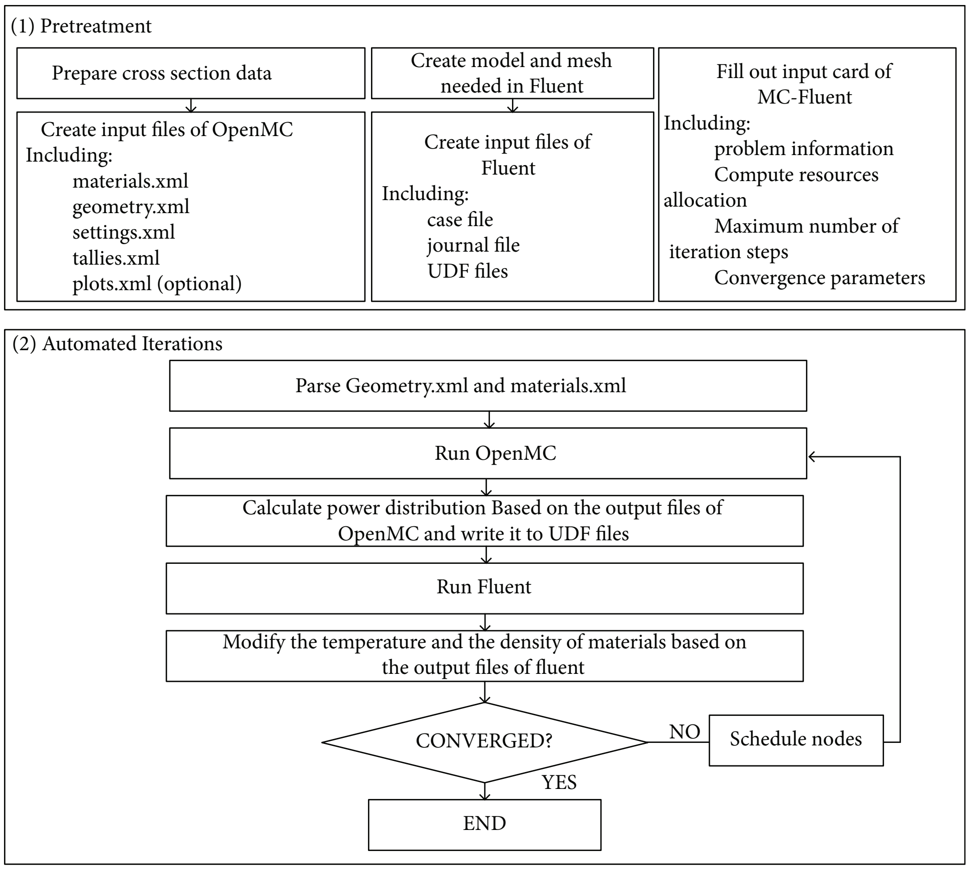

(1) Pretreatment
Prepare cross section data
Create model and mesh needed in Fluent
Fill out input card of MC-Fluent
Including:
problem information
Compute resources allocation
Maximum number of iteration steps
Convergence parameters
Create input files of OpenMC
Including:
materials.xml
geometry.xml
settings.xml
tallies.xml
plots.xml (optional)
Create input files of Fluent
Including:
case file
journal file
UDF files
(2) Automated Iterations
Parse Geometry.xml and materials.xml
Run OpenMC
Calculate power distribution Based on the output files of OpenMC and write it to UDF files
Run Fluent
Modify the temperature and the density of materials based on the output files of fluent
CONVERGED?
NO
Schedule nodes
YES
END

Figure 1: The procedure of MC-FLUENT with block Gauss–Seidel-type Picard algorithm.

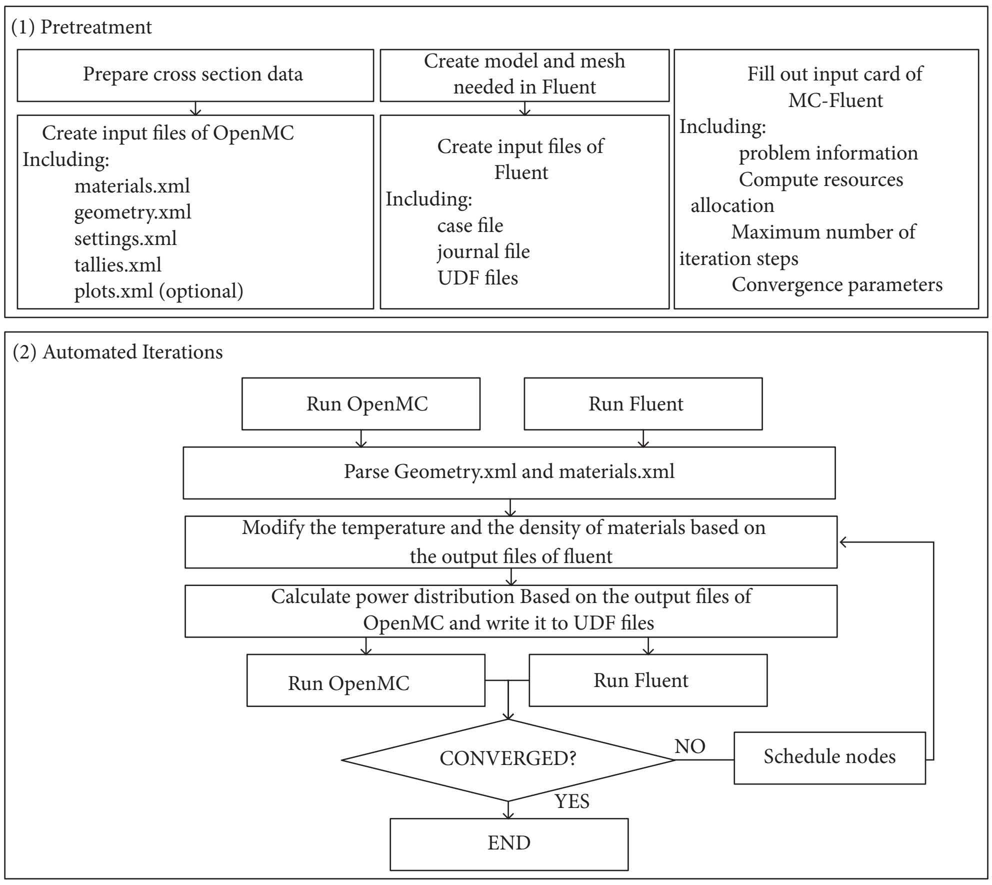

(1) Pretreatment
Prepare cross section data
Create model and mesh needed in Fluent
Fill out input card of MC-Fluent
Including:
problem information
Compute resources allocation
Maximum number of iteration steps
Convergence parameters
Create input files of OpenMC
Including:
materials.xml
geometry.xml
settings.xml
tallies.xml
plots.xml (optional)
Create input files of Fluent
Including:
case file
journal file
UDF files
(2) Automated Iterations
Run OpenMC
Run Fluent
Parse Geometry.xml and materials.xml
Modify the temperature and the density of materials based on the output files of fluent
Calculate power distribution Based on the output files of OpenMC and write it to UDF files
Run OpenMC
Run Fluent
CONVERGED?
NO
Schedule nodes
YES
END

Figure 2: The procedure of MC-FLUENT with block Jacobi-type Picard algorithm.

```
(1) **Requires:** initial guess for $\psi^0, T^0, \rho^0$, and $\mathbf{v}^0$
(2)     $k = 0$
(3)     **while** Not Converged **do:**
(4)     $k += 1$
(5)     Neutron physics code solve transport. $f_\psi(\psi^{k-1}, \rho^{k-1}, T^{k-1}) = 0$ for $\psi^k$
(6)     Obtain Source term $S^k$ of energy equation $S^k = S(\psi^k)$
(7)     Thermal hydraulic solve. $f_{TH}(\rho^k, T^k, v^k, r(S^k)) = 0$ for $\rho^k, T^k, \mathbf{v}^k$
(8)     **end while**
```

Algorithm 1: Picard: block Gauss–Seidel.

```
(1) **Requires:** initial guess for $\psi^0, T^0, \rho^0$, $v^0$ and $S^0$
(2)     $k = 0$
(3)     **while** Not Converged **do:**
(4)     $k += 1$
(5)     Neutron physics code solve transport. $f_\psi(\psi^{k-1}, \rho^{k-1}, T^{k-1}) = 0$ for $\psi^k$
(6)     Thermal hydraulic solve. $f_{TH}(\rho^k, T^k, v^k, r(S^{k-1})) = 0$ for $\rho^k, T^k, v^k$
(7)     Obtain Source term $S^k$ of energy equation. $S^k = S(\psi^k)$
(8)     **end while**
```

Algorithm 2: Picard: block Jacobi.

3.2.1. For Block Gauss–Seidel-Type Picard Scheme. For block Gauss–Seidel scheme, OpenMC and FLUENT are calculated in turn, and each code will use all the compute nodes assigned to the coupled program. Consequently, there will be no load-balancing problem. However, when the number of computing nodes increases, the computing efficiency of one code may start to decline while that of another code keeps on improving, which is not conducive to the improvement of overall efficiency. Considering that different coupling problems have different requirements for scalability, an adaptive load-balancing algorithm that can adjust the number of nodes dynamically is proposed.

With this algorithm, the total number of compute nodes (N_{total}) and the minimum number of nodes used for MC code (N_{mc}) and CFD code (N_{cfd}) should be set before running the coupled program. Then, MC-FLUENT will try to increase the number of nodes after each iteration step. In this paper, it will start with the minimum number of nodes and multiply by 2 every iteration. In addition, MC-FLUENT will count the running time of MC code (t_{mc}^k), CFD code (t_{mc}^k), and the total running time (t_{total}^k) during each iteration step. If the calculation time increases, it will stop adjusting and reuse the last number of nodes. The outline of this algorithm is illustrated in Algorithm 3 with the superscript k of letters indicating the kth iteration.

3.2.2. For Block Jacobi-Type Picard Scheme. With block Jacobi scheme, the MC code and the CFD code will be running in a parallel manner. By default, each code will use half of the compute nodes assigned to the coupled program. However, since the resources required by OpenMC and FLUENT are usually unequal, this solution can be less than the best. Furthermore, as the coupled problems met vary, the corresponding resource allocation schemes are different. Therefore, an adaptive load-balancing algorithm has been proposed for the block Jacobi scheme. With this algorithm, the MC-FLUENT will allocate resources according to the total number of nodes (which must be larger than 2). During initialization, nodes are allocated according to the ratio of 1 : 1. Then, the running time of each code during each iteration is counted, respectively. If the running time of one code is shorter than another one in the current iteration, the number of nodes used by this code will be halved during next iteration. However, if the total time of current iteration is longer than the last one, MC-FLUENT will stop adjusting and use the last allocative decision. Ideally, the adjustment will stop when the running time of the two codes is equal. The outline of the algorithm for block Jacobi scheme is depicted in Algorithm 4. The meaning of letters and superscripts is the same as Algorithm 3.

3.3. Data Exchange. One major challenge to the coupling codes is the communication of the variables, which, in this paper, refer to temperature, density, and power density. Since the number of cells in OpenMC is smaller than the number of volumes in the FLUNET model, the decision of a suitable spatial mapping is important. During coupling, OpenMC needs a discrete set of temperatures and density for each cell while FLUENT needs correct power density for each volume element. In this case, the data exchange method used in Henry et al. [34] is adopted for reference in present research: an averaging operation is necessary on the FLUNET volume elements corresponding to the OpenMC cell while transmitting the temperature and density. In addition, the cell of OpenMC model corresponds to the domains of the FLUENT model which is regarded as the coarser mesh, and interpolation can be done with FLUENT UDF.

```
(1) Requires: initial set for $\mathbf{N_{mc}}$, $\mathbf{N_{cfd}}$ and $\mathbf{N_{total}}$
(2) bool $\mathbf{update_{mc}}$, $\mathbf{update_{cfd}}$ = True
(3) Get $t^0_{mc}$ and $t^0_{cfd}$ in the 0th iteration
(4) while Iteration Not End do:
(5) Get $t^k_{mc}$ and $t^k_{cfd}$ in the $k$th iteration
(6) If $\mathbf{update_{mc}}$:
(7) Set $N_{mc} = (t^k_{mc} < t^{k-1}_{mc})?((2N_{mc} < N_{total})?(2*N_{mc}): N_{total}): (N_{mc}/2)$
(8) $\text{update}_{mc} = (t^k_{mc} < t^{k-1}_{mc})$?True: False
(9) If $\mathbf{update_{cfd}}$ :
(10) Set $\mathbf{N_{cfd}} = (\mathbf{t^k_{cfd}} < \mathbf{t^{k-1}_{cfd}})?((2\mathbf{N_{cfd}} < \mathbf{N_{total}})?(2*\mathbf{N_{cfd}}): \mathbf{N_{total}}): (\mathbf{N_{cfd}}/2)$
(11) $\mathbf{update_{cfd}} = (\mathbf{t^k_{cfd}} < \mathbf{t^{k-1}_{cfd}})$?True: False
(12) end while
```

Algorithm 3: Adaptive load-balancing algorithm for block Gauss–Seidel scheme.

```
(1) Requires: initial set for $\mathbf{N_{total}}$
(2) bool update = True
(3) Set $\mathbf{N_{mc}} = \mathbf{N_{total}}/2$, $\mathbf{N_{cfd}} = \mathbf{N_{total}} - \mathbf{N_{mc}}$
(4) Get $t^0_{mc}$, $t^0_{cfd}$ and $t^0_{total}$ in the 0th iteration
(5) while Iteration Not End && update do:
(6) Get $t^k_{mc}$, $t^k_{cfd}$ and $t^k_{total}$ in the $k$th iteration
(7) If $(\mathbf{t^k_{total}} < \mathbf{t^{k-1}_{total}}$ && $\mathbf{t^k_{mc}} < \mathbf{t^k_{cfd}})$:
(8) Set $\mathbf{N_{mc}} = (\mathbf{N_{mc}}/2 > 1)?\mathbf{N_{mc}}/2 : 1$, $\mathbf{N_{cfd}} = \mathbf{N_{total}} - \mathbf{N_{mc}}$
(9) Else if $(\mathbf{t^k_{total}} < \mathbf{t^{k-1}_{total}}$ && $\mathbf{t^k_{mc}} \geq \mathbf{t^k_{cfd}})$ :
(10) Set $\mathbf{N_{cfd}} = (\mathbf{N_{cfd}}/2 > 1)?\mathbf{N_{cfd}}/2 : 1$, $\mathbf{N_{mc}} = \mathbf{N_{total}} - \mathbf{N_{cfd}}$
(11) Else if $(\mathbf{t^k_{total}} < \mathbf{t^{k-1}_{total}}$ && $\mathbf{t^k_{mc}} = \mathbf{t^k_{cfd}})$ :
(12)     update = False
(13) Else
(14) Set $\mathbf{N_{mc}} = (\mathbf{t^k_{mc}} > \mathbf{t^k_{cfd}})?(2*\mathbf{N_{mc}}): (\mathbf{N_{total}} - 2*\mathbf{N_{cfd}})$, $\mathbf{N_{cfd}} = \mathbf{N_{total}} - \mathbf{N_{mc}}$
(15)     update = False
(16) end while
```

Algorithm 4: Adaptive load-balancing algorithms for block Jacobi scheme.

The third technique for updating the cross sections used in Seker et al. [35] is adopted for reference. With this approach, a library would be pregenerated for each nuclide with a proper temperature increment. Then, with the use of the interpolation method in OpenMC manual [30], the cross sections lying between the temperature intervals would be almost accurate [35].

4. Computational Models

A single pressurized water reactor (PWR) cell model is simulated to verify the validity of the MC-FLUENT. In addition, as is mentioned in Section 3, the computing efficiency and scalability of OpenMC and FLUENT are observed to show a tendency to disaccord, which is likely to reduce the overall efficiency of the iteration algorithms. Therefore, a 3 x 3 fuel pin example with similar requirements for the OpenMC and FLUENT and a 1000 MWt ABR metallic benchmark core example which centers on the performance of MC code are, respectively, adopted to study the computing efficiency of the two different schemes.

4.1. Single PWR Cell Model. Although MC-FLUENT aims to analyze complex models with massively parallel computing, a single PWR cell model can verify the validity of the coupled program. Cardoni and Rizwan-uddin developed a coupled simulation tool MULTINUKE and performed a steady-state PWR simulation in 2011 [7, 8]. Their work has been studied in many papers, and their results are considered to agree with PWR values typically reported in literature [36–39]. In the present study, we simulated the same PWR cell model in [7] and compared our results with theirs. As shown in Figure 3, this model consists of a fuel rod surrounded by cladding and water and has a height of 20 cm. Figure 4 shows the cell division of OpenMC and meshing for FLUENT in this paper.

4.2. 3 X 3 Fuel Pins of PWR. The second example is a 3 x 3 array of PWR pins, in which eight fuel pins were modeled around a central guide. The fuel pellet and the cladding were, respectively, made of UO2 and Zr. The gap between pellet and cladding has not been taken into consideration since this example does not emphasize the temperature distribution

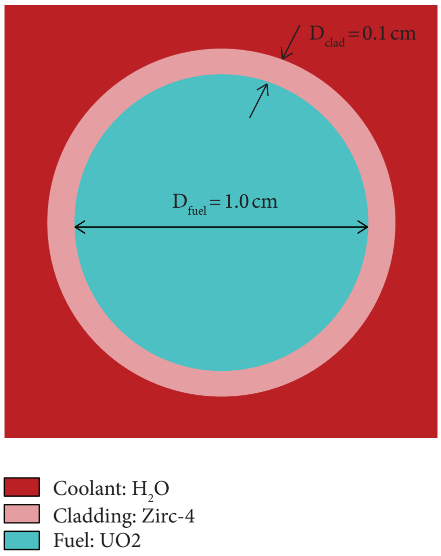


Figure 3: Schematic of the single PWR cell.

inside the cladding; however, the equivalent thermal conductivity was set for the cladding. Figure 5 shows the schematic of the model and the number of fuel pins. The design parameters of the fuel pin and central guide are provided in Table 1.

The entire OpenMC model was divided into 10 layers along the axis. For each layer, every single fuel pellet was trisected, and each of its three layers was divided into 4 cells, along with which the cladding was quartered and altogether the water was divided into 40 cells as is schematically illustrated in Figure 6(a). Reflective boundary conditions were set to the surface around the rod bundles. Considering that the model is strictly centrosymmetric in space, a 1/4 model was established for FLUENT to reduce the cost of computation. At the end of the calculation, the results were extended to the whole geometric region for data exchange using symmetry. Three varisized meshes were applied to verify the mesh sensitivity, which, respectively, contained 14000 K cells, 16000 K cells, and 18000 K cells. Each of the three meshes was employed in calculating the total pressure loss, and their errors from one another observed ranged acceptably less than 2%. Therefore, the mesh containing 16,726,000 cells was taken as the final choice for FLUENT as is shown in Figure 6(b).

4.3. 1000 MWt ABR Metallic Benchmark Core. The third example is a 1000 MWt advanced burner reactor (ABR) metallic benchmark core developed by Organisation for Economic Co-operation and Development (OECD) [40]. The nominal power is 1000 MWt, and the core is divided into the inner and the outer core zones, with 78 and 102 assemblies, respectively. There are 271 fuel pins wrapped with wire in each driver subassembly. In order to simplify the geometry, the wire has been smeared with cladding during the modeling, and thus the cladding thickness has been slightly increased. In addition, the gap between fuel pellet and cladding was filled with bond sodium. The model built for OpenMC is presented in Figure 7.

Uniform temperature values in the coolant, structural materials, and fuel have been assumed in this benchmark problem which are shown in Table 2. In practice, however, the core temperature distribution should be dependent on the power distribution. Therefore, we hope to provide a reasonable temperature distribution for OpenMC simulation by coupling with FLUENT.

In CFD calculation, due to the noticeably complex reactor structure, the computing scale corresponding to detailed geometric model is unsupportable. Considering that the example here focuses much more on the temperature and density distributions of the reactor rather than on flow details in the bundle, the porous model can be used to replace the detailed bundle structure during CFD calculation for simplicity. As shown in Figure 8(a), according to the division of regions in the benchmark, a quarter model was constructed according to the principle of volume equivalence and the reactor was divided into four parts: inner core, outer core, reflector region, and shield region. The porous media model has been used to simulate the influence of solid region on the flow. The detailed setting method of porous media model is described in FLUENT Users Guide [32]. For bundles of rods with wire wrap, Darcy friction factor can be calculated based on the empirical formula proposed by Cheng and Todreas [41, 42]. The nonthermal balance model has been used in FLUENT to evaluate the heat transfer between fluid and solid in porous regions. With this model, the CFD code is enabled to generate overlapped solid regions in the porous regions, so that the temperatures of solid and liquid can be obtained, respectively, which will be transmitted to the MC program for further computing. The heat transfer coefficient of solid material can be set based on the formula proposed by Borishanski et al. [43, 44].

After verifying the mesh sensitivity, the mesh containing 1,975,640 cells was taken as the final choice for FLUENT as shown in Figure 8(b). At the end of the calculation, the results were extended to the whole geometric region for data exchange using symmetry.

5. Results and Discussion

5.1. Single PWR Cell Model. The purpose of the single PWR cell model is to verify the validity of the coupling program. The converged results are compared with reference [7] and presented in Table 3. Figure 9 shows the converged relative axial power distribution and the axial fuel temperature distribution at different radial locations (r). The axial distribution of average coolant temperature and density is shown in Figure 10. It can be found that our results are in good agreement with those in reference [7].

5.2. 3 X 3 Fuel Pins of PWR. MC-FLUENT has been performed on a high-performance computing system with the 64-bit CentOS Linux operating system. Each compute node of the cluster consists of one Intel Xeon E5-2620 CPU with 12 cores and a frequency of 2 GHz. In order to make a full test on the parallel performances of the two schemes, 1/2/4/8/16/32 compute nodes were used, respectively, which are

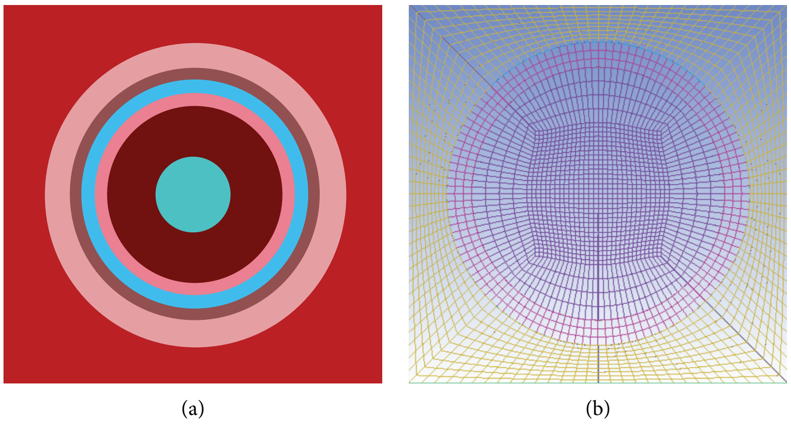


Figure 4: Cell division and meshing for PWR cell model. (a) Cell division in OpenMC. (b) Meshing for FLUENT.

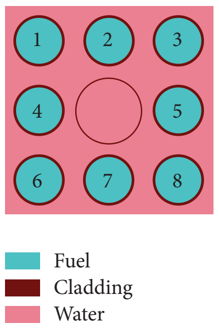


Figure 5: Schematic of the 3 x 3 fuel pins.

Table 1: Parameters for 3 x 3 fuel pin model.

	Unit	Parameters
Total power	KW	120
Fuel pellet radius	cm	0.418
Cladding outer radius	cm	0.475
Guide tube inner radius	cm	0.561
Guide tube outer radius	cm	0.602
Cell pitch	cm	1.26
Active height	cm	200

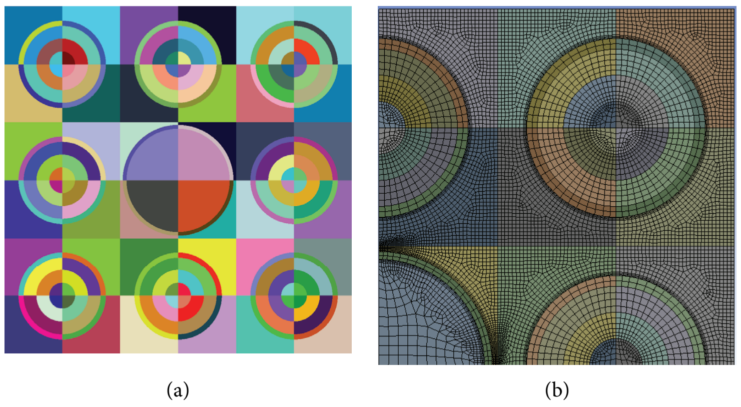


Figure 6: Cell division and meshing for 3 x 3 fuel pins. (a) Cell division in OpenMC. (b) Meshing for FLUENT.

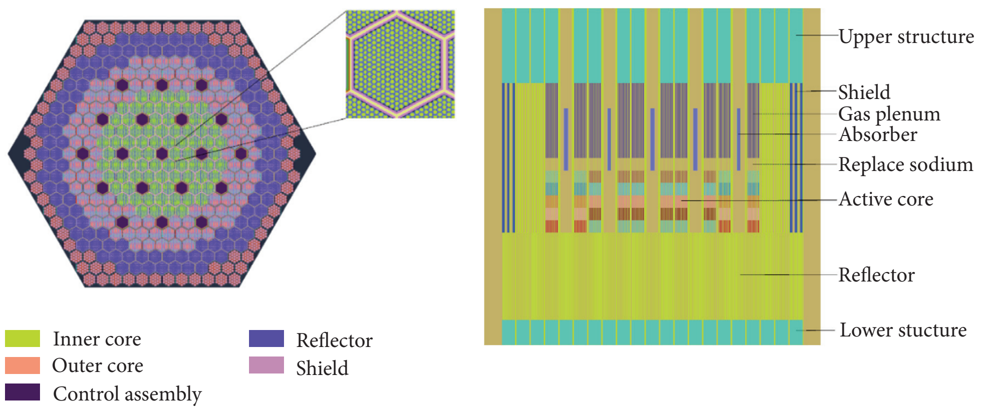


Figure 7: OpenMC model of MET-1000. (a) Top view. (b) Side view.

Table 2: Nominal operating condition.

Reactor power	1000 MWt (°C)
Coolant temperature	432.5
Average core structural temperature	432.5
Average fuel temperature	534.0

equal to 12/24/48/96/192/384 CPU cores. In the cases of this paper, different numbers of nodes are utilized to make a comparative study of computational performance. However, as data transmission remains the same way all along, changes in the number of nodes do not lead to any change in the number of iterations.

Only 9 iteration steps were required for the block Gauss–Seidel scheme to converge while 19 were required for the block Jacobi scheme in the 3 X 3 fuel pin example. Figure 11 represents the total calculation time and speedup of the two algorithm using different number of nodes. The speedup is calculated by the following formula:

$$S_p = \frac{t}{t_0}, \tag{2}$$

where S_p is the speedup, t is the total calculation time, and t_0 is the total calculation time of the block Gauss–Seidel scheme using one compute node.

As shown in Figure 11(a), the block Gauss–Seidel scheme demonstrates a higher convergence speed than block Jacobi scheme, which results in less total computation time. However, due to the poor scalability of a certain code, the speedup of block Gauss–Seidel begins to experience a downward trend after the nodes used peaks at 16. Therefore, we add the adaptive load-balancing algorithm described in Section 3.2 to block Gauss–Seidel algorithm. For this example, we set the algorithm of increasing from half of the total number of nodes. Because our adaptive load-balancing algorithm cannot make full use of all resources at the beginning of the iteration, the computing time is slightly higher than the original algorithm while number of nodes is less than 16. In fact, the decrease of efficiency can be avoided by setting a larger number of nodes at beginning. Nevertheless, the adaptive load-balancing algorithm successfully prevented the decrease of speedup when the number of nodes reaches 32. Therefore, applying our algorithm when the number of nodes is larger can improve the performance of the block Gauss–Seidel scheme.

As shown in Figure 11(b), the total running time of the block Jacobi scheme is generally longer than that of block Gauss–Seidel algorithm because of its poor convergence. However, by adding the adaptive load-balancing algorithm described in Section 3.2, resources can be allocated more reasonably, and in this way, the performance of this algorithm can be optimized. However, when there are only two nodes, there are no resources to allocate. When the number of nodes reaches 32, computing resources are sufficient for this case, so the load-balancing algorithm cannot further improve the computing efficiency. Although the demand for computational power of OpenMC and FLUENT is similar in this example, our adaptive load-balancing algorithms can still improve the efficiency of the two schemes.

The errors of effective neutron multiplication coefficients (K_{eff}) during the iterations are presented in Figure 12. The temperature of materials is 300 K for the initial calculation. 1000 batches and 100,000 particles per batch were specified to OpenMC simulation. Both schemes converge to a unified solution. The combined K_{eff} is 1.26133 ± 0.0001. The power and temperature distributions along the axis of the second fuel pin are shown in Figure 13.

5.3. 1000 MWt Metallic Fuel Core Benchmark. Before the coupling iteration begins, core multiplication factor (K_{eff}), sodium void worth ($\Delta\rho_{Na}$), Doppler constant ($\Delta\rho_{\text{Doppler}}$), control rod worth ($\Delta\rho_{CR}$), inlet and outlet temperature of coolant, and the average fuel temperature of the benchmark are calculated to verify the correctness of the model. The formula of these parameters and the related works can be found in the benchmark for neutronic analysis of sodium-cooled fast reactor cores with various fuel types and core sizes [40]. The results before coupling have been verified and are summarized in Table 4. 10^5 is reactivity difference in units of pcm. It is observed that there are no statistical differences from the eigenvalues obtained by other researchers [40].

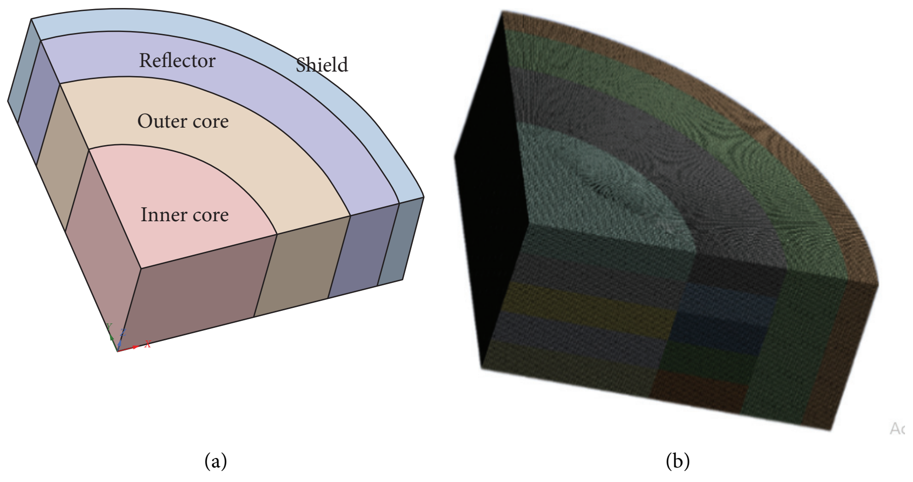


Figure 8: (a) Regional model and (b) mesh of MET-1000.

Table 3: Comparison of the single PWR cell model results.

	K_{eff}	Peak clad temperature	Temperature rise of coolant
Results in reference [7]	0.6601 ± 0.0009	720.1 K at 10.9 cm	11.0 K
Results in this paper	0.6602 ± 0.0003	720.0 K at 10.9 cm	10.9 K

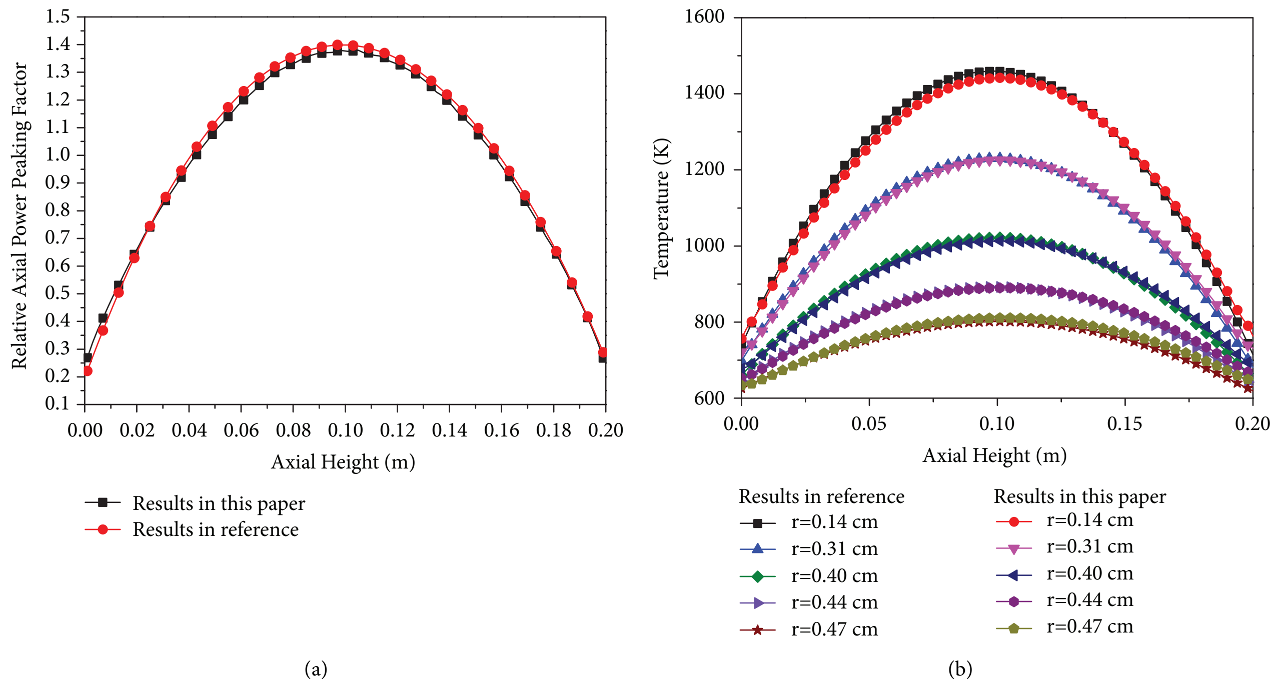
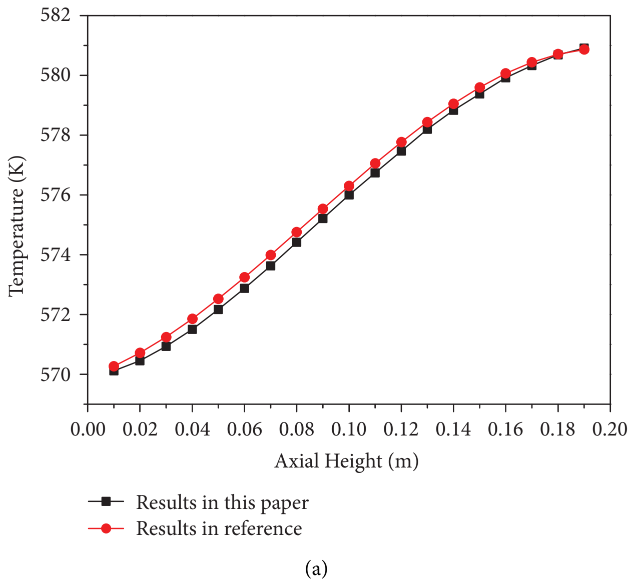


Figure 9: Comparison of (a) the relative axial power distribution and (b) the axial fuel temperature distribution at different radial locations.

For the coupling simulation, MC-FLUENT has been performed on the same high-performance computing system as the one employed in the preceding 3 x 3 fuel pin example. 1/2/4/8/16/32 compute nodes were used, respectively, to test the parallel performances of two schemes, which are equal to 12/24/48/96/192/384 CPU cores. 1000 batches and 500,000 particles per batch were specified during OpenMC simulation while RNG k-ε turbulence model and COUPLED discrete format were adopted during FLUENT calculation. In this case, the porous model is used in FLUENT while OpenMC utilizes the detailed model, so the computational resources required by OpenMC are much more than those required by FLUENT.

Four iteration steps were required for the block Gauss–Seidel scheme to converge, while 6 were required for the block Jacobi scheme. Figure 14 illustrates the calculation



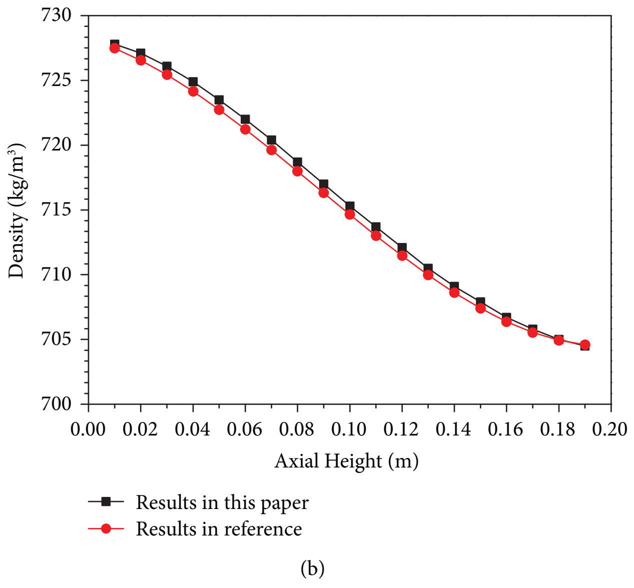
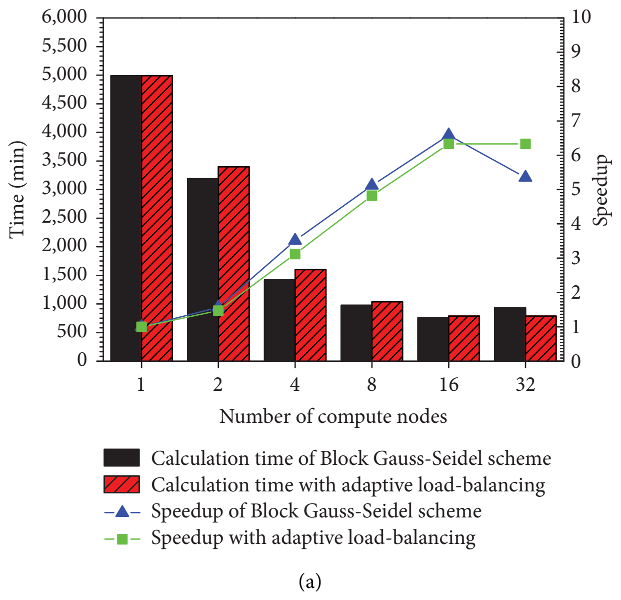


FIGURE 10: Comparison of the axial distribution of average coolant (a) temperature and (b) density.



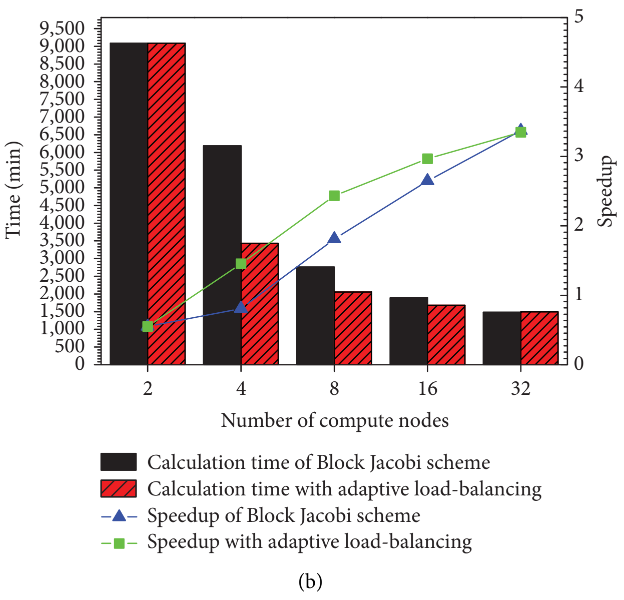


FIGURE 11: Total running time and speedup of the two schemes for the 3 x 3 fuel pin example with different number of compute nodes. (a) Block Gauss–Seidel scheme. (b) Block Jacobi scheme.

time and speedup of the two schemes using different number of nodes. For the block Gauss–Seidel scheme, since OpenMC needs a lot of resources, only the number of nodes for FLUENT is limited while applying our adaptive load-balancing algorithms. The number of nodes for FLUENT grows from 1 according to the algorithm. As shown in Figure 14(a), our algorithm hardly increases the running time while the number of nodes is small. Moreover, compared with the original block Gauss–Seidel algorithm, when there are more nodes, our algorithm obtains a higher speedup. The main reason is that the scalability of FLUENT in this case limits the speedup growth of the original algorithm. For block Jacobi algorithm, as represented in Figure 14(b), our adaptive load-balancing algorithms bring remarkable improvement in computing performance. The reasonable allocation of resources improves the computing efficiency of block Jacobi algorithm a lot. It is worth noting that when there are 32 nodes, the performance of the block Jacobi scheme while adding adaptive load-balancing algorithm is almost equal to that of the original block Gauss–Seidel scheme. It can be seen that for this example which demands different computational power of OpenMC and FLUENT, our algorithms can obviously improve the efficiency of the two schemes.

Monte Carlo results after OpenMC and FLUENT coupling are represented in Table 5. The eigenvalue calculated

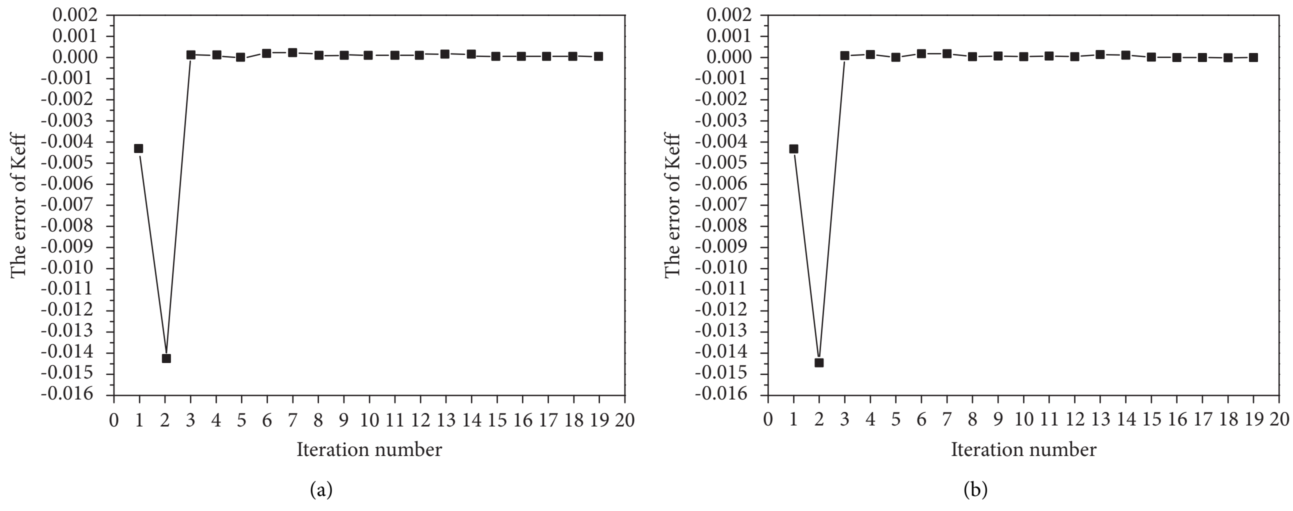


Figure 12: K_{eff} versus the iteration number. (a) Block Gauss–Seidel scheme. (b) Block Jacobi scheme.

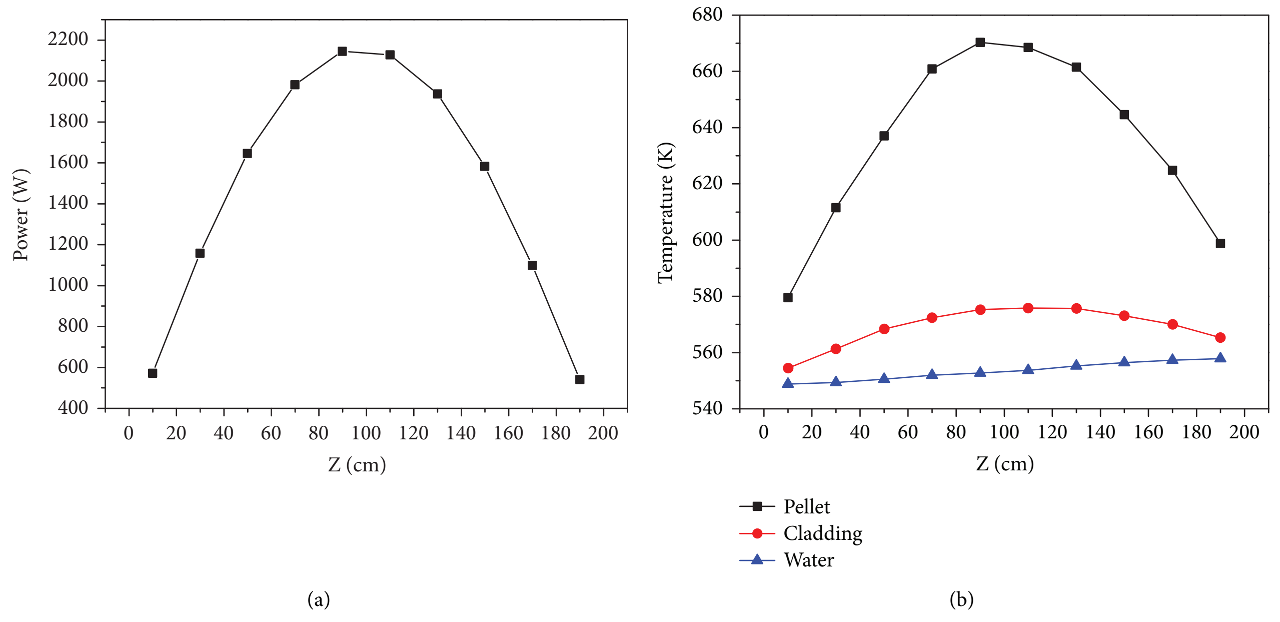


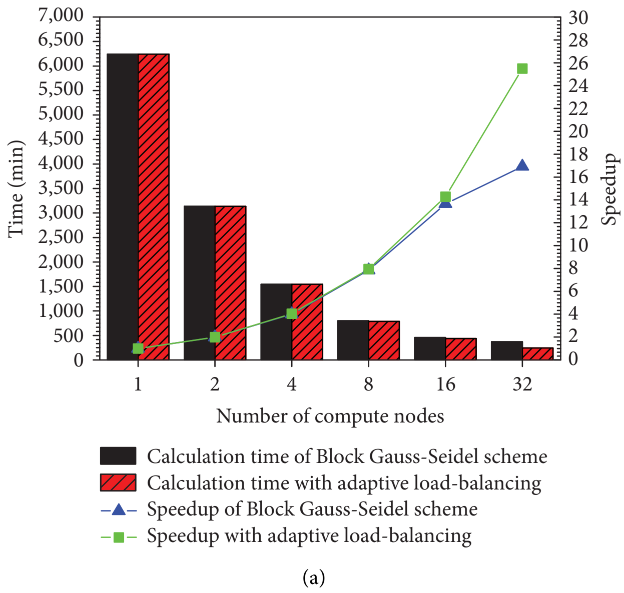
Figure 13: (a) Power distribution and (b) temperature distribution along the axis of the second fuel in the corner.

Table 4: Results of MET-1000 before coupling.

K_{eff}	1.02811 ± 0.00003
$\Delta\rho_{\mathrm{Doppler}}$	−334
$\Delta\rho_{Na}$	1903
$\Delta\rho_{CR}$	18128
Average inlet temperature (K)	628
Average outlet temperature (K)	783
Average fuel temperature (K)	809

after OpenMC and FLUENT coupling featured 54 pcm higher than the one obtained with OpenMC alone at a constant temperature. Besides, the average control rod worth slightly rose after coupling by 10 pcm. Unlike the Doppler constant, which has changed little, the average sodium void worth declined considerably due to the temperature and density distribution change of coolant.

The axial distribution of power and core fuel temperature is shown in Figure 15 while the final temperature distribution of sodium on the symmetry section and z-axis intermediate section can be seen in Figure 16. In the calculation process of FLUENT, the density is usually set to have a polynomial correlation to temperature.



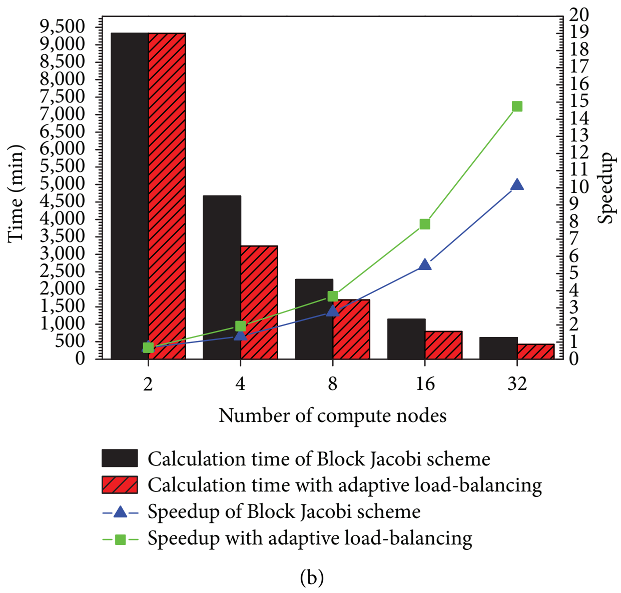
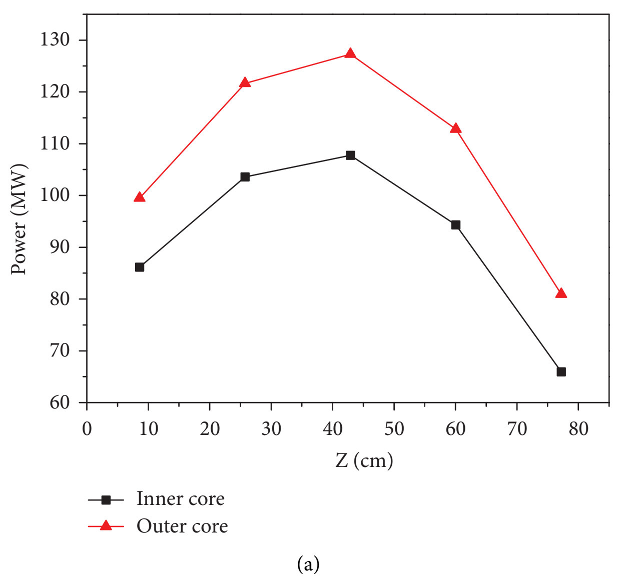


FIGURE 14: Total running time and speedup of the two schemes for 1000 MWt metallic fuel core benchmark with different number of compute nodes. (a) Block Gauss–Seidel scheme. (b) Block Jacobi scheme.

TABLE 5: Results of MET-1000 after coupling.

K_{eff}	1.02865 ± 0.00003
$\Delta\rho_{\text{Doppler}}$	−338
$\Delta\rho_{Na}$	1717
$\Delta\rho_{CR}$	18147



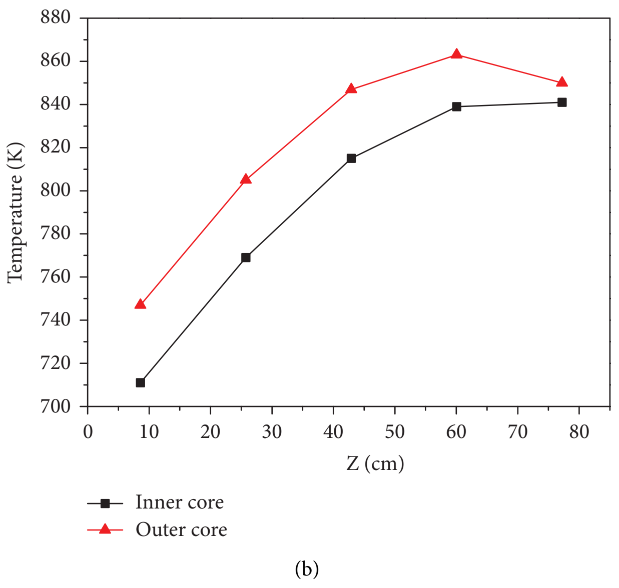


FIGURE 15: Axial distribution of power and average fuel temperature. (a) Power. (b) Average fuel temperature.

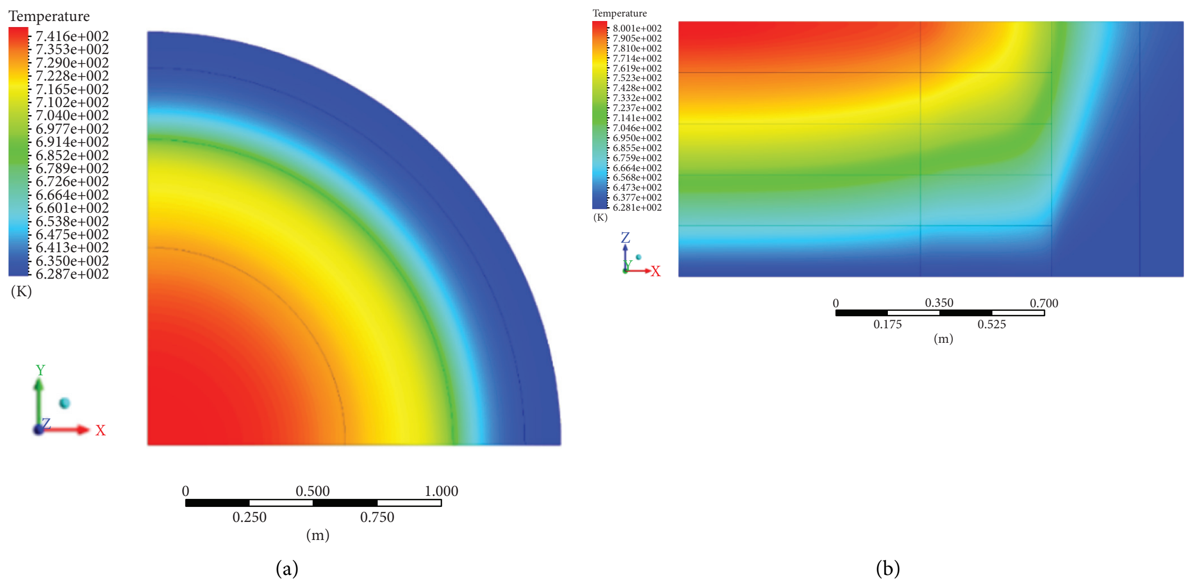


Figure 16: Final temperature distribution of sodium. (a) Section at $z = 42.91$ cm. (b) Symmetry section.

6. Conclusions

In this paper, an interface code MC-FLUENT has been developed to couple the Monte Carlo code OpenMC with the computational fluid dynamics code ANSYS FLUENT. Block Gauss–Seidel-type Picard algorithm and block Jacobi-type Picard algorithm were researched to control the calculation of OpenMC and FLUENT. Moreover, in order to improve the performance of multi-physics simulation in massive parallel computing, two adaptive load-balancing algorithms are proposed to reduce the time cost of two algorithms when coupling OpenMC and FLUENT on the high-performance computer. A single PWR cell model is simulated, and the simulation results are compared with the values in literature. A 3 x 3 fuel pin example with similar requirements for the OpenMC and FLUENT and a 1000 MWt ABR metallic benchmark core example which centers on the performance of MC code are used to compare the performance of the two algorithms and verify the effectiveness of the adaptive load-balancing algorithms.

Overall, it was found that the single PWR cell simulation results in this paper are in good agreement with the values in literature, which verifies the validity of MC-FLUENT. Compared with block Jacobi algorithm, the block Gauss–Seidel algorithm has a higher efficiency, but the growth rate of its speedup may limited by the different scalability of OpenMC and FLUENT. The adaptive load-balancing algorithm for block Gauss–Seidel algorithm proposed in this paper can improve its performance when there are many nodes. The adaptive load-balancing algorithm for block Jacobi algorithm proposed in this paper can greatly improve its performance. In addition, with the adaptive load-balancing algorithm, block Jacobi algorithm can achieve similar performance to the original block Gauss–Seidel algorithm when the resources are sufficient. The adaptive load-balancing algorithms are more effective when the case demands different computational power of OpenMC and FLUENT.

The future work will move focus from the conventional multi-core CPU platforms to the heterogeneous system architecture (HSA) which can be more cost-effective as an alternation. With the HSA, the coupling simulations may be able to expect better performances.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was funded by the National Key Research and Development Program of China (2017YFB0202104 and 2018YFB0204301) and the National Natural Science Foundation of China (11705255).

References

[1] M. Nikolaus, D. William, R. Reko et al., *The Role of Nuclear Energy in a Low-Carbon Energy Future*, JRC, Belgium, 2012.

[2] C. Karakosta, C. Pappas, V. Marinakis, and J. Psarras, "Renewable energy and nuclear power towards sustainable development: characteristics and prospects," *Renewable and Sustainable Energy Reviews*, vol. 22, pp. 187–197, 2013.

[3] J. Wu, S. Guo, H. Huang, William L., and Yong X., "Information and communications technologies for sustainable development goals: state-of-the-art, needs and perspectives," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2389–2406, 2018.

[4] R. Atat, L. Liu, J. Wu, G. Li, C. Ye, and Y. Yang, "Big data meet cyber-physical systems: a panoramic survey," *IEEE Access*, vol. 6, pp. 73603–73636, 2018.

[5] P. Hidalga, A. Abarca, R. Miró, A. Sekrhi, and G. Verdú, "A multi-scale and multi-physics simulation methodology with the state-of-the-art tools for safety analysis in light water reactors applied to a turbine trip scenario (PART I)," *Nuclear Engineering and Design*, vol. 350, pp. 195–204, 2019.

[6] M. Avramova, A. Abarca, J. Hou, and I. Kostadin, "Innovations in multi-physics methods development, validation, and uncertainty quantification," *Journal of Nuclear Engineering*, vol. 2, no. 1, pp. 44–56, 2021.

[7] J. N. Cardoni and U. Rizwan, "Nuclear reactor multi-physics simulations with coupled MCNP5 and STAR-CCM+[C]," in *Proceeding of the M&C*, Rio de Janeiro, RJ, Brazil, November 2011.

[8] J. N. Cardoni and U. Rizwan, *Nuclear Reactor Multi-Physics Simulations with Coupled MCNP5 and STAR-CCM+[D]*, University of Illinois, Chicago, IL, USA, 2011.

[9] K. Ivanov and M. Avramova, "Challenges in coupled thermal-hydraulics and neutronics simulations for LWR safety analysis," *Annals of Nuclear Energy*, vol. 34, no. 6, pp. 501–513, 2007.

[10] M. J. Marzano, *Approach to Coupling 3-D Deterministic Neutron Transport and Full Field Computational Fluid Dynamics*, University of Florida, Gainesville, FL, USA, 2011.

[11] A. L. Costa, C. Pereira, W. Ambrosini, and F. D'Auria, "Simulation of an hypothetical out-of-phase instability case in boiling water reactor by RELAP5/PARCS coupled codes," *Annals of Nuclear Energy*, vol. 35, no. 5, pp. 947–957, 2008.

[12] C. Watta, T. Schulenburg, X. J. F. Cheng, and E. Laurien, *Coupling of MCNP with a Sub-channel Code for Analysis of a HPLWR Fuel Assembly*, Institut für Kern- und Energietechnik (IKET), Berlin, Germany, 2006.

[13] D. P. Weber, T. Sofu, W. S. Yang et al., "High-fidelity light water reactor analysis with the numerical nuclear reactor," *Nuclear Science & Engineering*, vol. 155, no. 3, pp. 395–408, 2007.

[14] M. Daeubler, J. Jimenez, and V. Sanches, "Development of a high-fidelity Monte Carlo thermal-hydraulics coupled code system Serpent/SUBCHANFLOW–first results," in *Proceedings of the PHYSOR 2014 – The Role of Reactor Physics Toward a Sustainable Future*, Kyoto, Japan, October 2014.

[15] A. Bousbia-Salah, F. D'auria, and F. D'Auria, "Use of coupled code technique for Best Estimate safety analysis of nuclear power plants," *Progress in Nuclear Energy*, vol. 49, no. 1, pp. 1–13, 2007.

[16] V. S. Mahadevan, J. C. Ragusa, and V. A. Mousseau, "A verification exercise in multiphysics simulations for coupled reactor physics calculations," *Progress in Nuclear Energy*, vol. 55, pp. 12–32, 2012.

[17] M. Vazquez, H. Tsige-Tamirat, L. Ammirabile, and F. Martin-Fuertes, "Coupled neutronics thermal-hydraulics analysis using Monte Carlo and sub-channel codes," *Nuclear Engineering and Design*, vol. 250, pp. 403–411, 2012.

[18] D. Kotlyar and E. Shwageraus, "Numerically stable Monte Carlo-burnup-thermal hydraulic coupling schemes," *Annals of Nuclear Energy*, vol. 63, pp. 371–381, 2014.

[19] C. Cd, Use and Development of Coupled Computer Codes for the Analysis of Accidents at Nuclear Power Plants, 2007.

[20] H. Wu and U. Rizwan, "A tightly coupled scheme for neutronics and thermal-hydraulics using open-source software," *Annals of Nuclear Energy*, vol. 87, pp. 16–22, 2016.

[21] W. Gurecky and E. Schneider, "Development of an MCNP6-ANSYS FLUENT multiphysics coupling capability," *International Conference on Nuclear Engineering*, p. V004T10A028, 2016.

[22] R. Tuominen, *Coupling Serpent and OpenFOAM for Neutronics-CFD Multi-Physics calculations[J]*, 2015.

[23] A. Erfaninia, A. Hedayat, S. M. Mirvakili, and M. R. Nematollahi, "Neutronic-thermal hydraulic coupling analysis of the fuel channel of a new generation of the small modular pressurized water reactor including hexagonal and square fuel assemblies using MCNP and CFX," *Progress in Nuclear Energy*, vol. 98, pp. 213–227, 2017.

[24] L. Castro, J.-L. François, and C. García, "Coupled Monte Carlo-CFD analysis of heat transfer phenomena in a supercritical water reactor fuel assembly," *Annals of Nuclear Energy*, vol. 141, Article ID 107312, 2020.

[25] Z. Jiang, C. Rieck, A. Bück, and E. Tsotsas, "Modeling of inter- and intra-particle coating uniformity in a Wurster fluidized bed by a coupled CFD-DEM-Monte Carlo approach," *Chemical Engineering Science*, vol. 211, Article ID 115289, 2020.

[26] D. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grandié, "MOOSE: a parallel computational framework for coupled systems of nonlinear equations," *Nuclear Engineering and Design*, vol. 239, no. 10, pp. 1768–1778, 2009.

[27] C. T. Kelley, *Iterative Methods for Linear and Nonlinear equations*, SIAM, Chennai, India, 1995.

[28] M. Cervera, R. Codina, and M. Galindo, "On the computational efficiency and implementation of block-iterative algorithms for nonlinear coupled problems," *Engineering Computations*, vol. 13, no. 6, pp. 4–30, 1996.

[29] H. G. Matthies, R. Niekamp, and J. Steindorf, "Algorithms for strong coupling procedures[J]," *Computer Methods in Applied Mechanics Engineering*, vol. 195, no. 17-18, pp. 2028–2049, 2006.

[30] P. K. Romano, N. E. Horelik, and B. R. Herman, "OpenMC: a state-of-the-art Monte Carlo code for research and development," in *Proceedings of the Joint International Conference on Supercomputing in Nuclear Applications+ Monte Carlo*, Paris, October 2013.

[31] The OpenMC Monte Carlo Code — OpenMC Documentation, https://docs.openmc.org/en/stable/.

[32] I. J. C. Ansys, ANSYS FLUENT User's guide, 2011.

[33] R. P. Pawlowski, K. N. Belcourt, R. Hooper, S. Rod, and B. Roscoe, *VERA MultiphysicsCoupling with LIME: Code Requirements*, Sandia National Lab.(SNL-NM), Albuquerque, NM, USA, 2011.

[34] R. Henry, I. Tiselj, and L. Snoj, "CFD/Monte-Carlo neutron transport coupling scheme, application to TRIGA reactor," *Annals of Nuclear Energy*, vol. 110, pp. 36–47, 2017.

[35] V. Seker, J. W. Thomas, and T. J. Downar, "Reactor simulation with coupled Monte Carlo and computational fluid dynamics," in *Proceedings of the Joint International Topical Meeting on Mathematics and Computations and Supercomputing in Nuclear Applications*, Monterey, California, April 2007.

[36] J. Chen, C. Liang-zhi, Z. You-qi, H. Wu, and K. Wang, "Neutronics and thermal-hydraulics coupling calculation based on Monte Carlo and CFD method," *Atomic Energy Science and Technology*, vol. 50, no. 2, pp. 301–305, 2016.

[37] D. F. Gill, D. P. Griesheimer, and D. L. Aumiller, "Numerical methods in coupled Monte Carlo and thermal-hydraulic calculations," *Nuclear Science & Engineering*, vol. 185, no. 1, pp. 194–205, 2017.

[38] S. K. Chimakurthi, S. Reuss, M. Tooley, and S. Stephen, "ANSYS Workbench System Coupling: a state-of-the-art computational framework for analyzing multiphysics problems," *Engineering with Computers*, vol. 34, pp. 385–411, 2017.

[39] L. Ye, M. Wang, X. a. Wang et al., "Thermal hydraulic and neutronics coupling analysis for plate type fuel in nuclear reactor core," *Science and Technology of Nuclear Installations*, vol. 2020, pp. 1–12, Article ID 2562747, 2020.

[40] N. Stauff, T. Kim, and T. Taiwo, *Benchmark for Neutronic Analysis of Sodium-Cooled Fast Reactor Cores with Various Fuel Types and Core Sizes*, Organisation for Economic Co-Operation and Development, UK, 2016.

[41] S.-K. Cheng and N. E. Todreas, "Hydrodynamic models and correlations for bare and wire-wrapped hexagonal rod bundles - bundle friction factors, subchannel friction factors and mixing parameters," *Nuclear Engineering and Design*, vol. 92, no. 2, pp. 227–251, 1986.

[42] S. K. Chen, R. Petroski, and N. E. Todreas, "Numerical implementation of the Cheng and Todreas correlation for wire wrapped bundle friction factors-desirable improvements in the transition flow region," *Nuclear Engineering and Design*, vol. 263, pp. 406–410, 2013.

[43] V. M. Borishanskii, M. A. Gotovskii, and V. Firsova, "Heat transfer to liquid metals in longitudinally wetted bundles of rods," *Soviet Atomic Energy*, vol. 27, no. 6, pp. 1347–1350, 1969.

[44] K. Mikityuk, "Heat transfer to liquid metal: review of data and correlations for tube bundles," *Nuclear Engineering and Design*, vol. 239, no. 4, pp. 680–687, 2009.