Nuclear Engineering and Technology 54 (2022) 3803–3810

Contents lists available at ScienceDirect

Nuclear Engineering and Technology

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

Original Article

Implementation and benchmarking of the local weight window generation function for OpenMC

Yuan Hu [a, b, *], Sha Yan [a], Yuefeng Qiu [b]

[a] State Key Laboratory of Nuclear Physics and Technology, School of Physics, Peking University, Beijing, 100871, China
[b] Institute for Neutron Physics and Reactor Technology, Karlsruhe Institute of Technology, Eggenstein-Leopoldshafen, 76344, Germany

ARTICLE INFO

Article history:
Received 18 January 2022
Received in revised form
28 March 2022
Accepted 25 April 2022
Available online 28 April 2022

Keywords:
Monte Carlo
OpenMC
Neutronics
Local variance reduction
Weight window generator

ABSTRACT

OpenMC is a community-driven open-source Monte Carlo neutron and photon transport simulation code. The Weight Window Mesh (WWM) function and an automatic Global Variance Reduction (GVR) method was recently developed and implemented in a developmental branch of OpenMC. This WWM function and GVR method broaden OpenMC's usage in general purposes deep penetration shielding calculations. However, the Local Variance Reduction (LVR) method, which suits the source-detector problem, is still missing in OpenMC. In this work, the Weight Window Generator (WWG) function has been developed and benchmarked for the same branch. This WWG function allows OpenMC to generate the WWM for the source-detector problem on its own. Single-material cases with varying shielding and sources were used to benchmark the WWG function and investigate how to set up the particle histories utilized in WWG-run and WWM-run. Results show that there is a maximum improvement of WWM generated by WWG. Based on the above results, instructions on determining the particle histories utilized in WWG-run and WWM-run for optimal computation efficiency are given and tested with a few multi-material cases. These benchmarks demonstrate the ability of the OpenMC WWG function and the above instructions for the source-detector problem. This developmental branch will be released and merged into the main distribution in the future.

1. Introduction

OpenMC is a community-driven open-source Monte Carlo (MC) neutron and photon transport simulation code [1]. It can perform fixed source, k-eigenvalue, and subcritical multiplication calculations on geometry models built using either a constructive solid geometry or CAD representation [2]. The Weight Window Mesh (WWM) function was recently developed and implemented in a developmental branch of OpenMC to expand its usage in general deep penetration shielding calculations [3]. An automatic Global Variance Reduction (GVR) method was also successfully developed and implemented in the same branch. This GVR method focuses on the global domain and generates the WWM automatically [4]. It allows OpenMC for usage on global simulations without any third-party codes, which improves the shielding simulation capabilities of OpenMC. However, this GVR method will waste the computing resource in the non-concerned area for the source-detector problem or the source-region problem (which only concerns a minor region). The Local Variance Reduction (LVR) method, which suits such problems, is still a missing part of OpenMC. Thus, in this work, the Weight Window Generator (WWG) function, which is provided as an LVR method in the MC code MCNP [5], is implemented in the above branch of OpenMC (v0.12.0) code [6]. This developmental branch will be released and merged into the main distribution in the future.

When utilizing the WWG function for shielding calculation, the user must first run one simulation with the WWG function to generate the WWM (WWG-run for short), and then run another one with the generated WWM to obtain the final results (WWM-run for short). However, little literature discussed determining the particle histories for WWG-run and WWM-run. Thus, in this work, single-material test cases with varying shielding and sources were studied to investigate the relationship between the generated WWM, the final results, and the corresponding particle histories. Based on the above results, instructions on determining the particle histories for WWG-run and WWM-run to achieve optimal

* Corresponding author. State Key Laboratory of Nuclear Physics and Technology, School of Physics, Peking University, Beijing 100871, China.
E-mail address: yuanhu@pku.edu.cn (Y. Hu).

https://doi.org/10.1016/j.net.2022.04.021

computation efficiency are proposed and tested with two multi-material cases.

2. Methodology

2.1. WWG implementation on OpenMC

The WWG function is a method that generates WWM for subsequent calculation automatically. To generate WWM, the WWG needs to estimate the importance function for specifying space (mesh bin) and energy (energy bin) based on an analog run. The importance of a mesh bin defines as the expected score generated by a unit weight particle after entering the cell, as indicated in equation (1):

$$\text{Importance(expected score)} = \frac{\text{Total score (because of particles and their progeny entering the mesh bin)}}{\text{Total weight (entering the mesh bin)}} \quad (1)$$

The WWM's lower Weight Window (WW) bound will be calculated inversely proportional to the importance function above [5,7]. And its upper WW bound is empirically decided as five times the lower WW bound [8].

Fig. 1 shows the workflow of the WWG function in OpenMC. As indicated in the dark blue box, we define two vectors to account for the "Total weight" and "Total score" in equation (1) for each mesh bin in each energy bin. Particle transport is the same as analog run when employing the WWG function, as shown in the light blue box. For each step in particle transporting, the WWG function will accumulate the "Total weight" and "Total score" (for importance calculation), as illustrated in the green box. The WWG function records the crossed mesh bins in every single step. And the particle weight will be counted in the "Total weight" vector for all crossed mesh bins in each step. In addition, if the particle has arrived at the target region in this step, its weight will also be counted as a score in the "Total score" vector for all crossed mesh bins in this particle history previously. After simulation, the WWG function will calculate the importance function based on equation (1) and the two vectors. The lower WW bound is then assigned by the WWG function, which is inversely proportional to the calculated importance function. Finally, WWG outputs the calculated lower WW bound for each mesh bin (WWM) in a file that can be used as a part of the input file in a subsequent calculation.

2.2. OpenMC WWG input

Since the main aim of the WWG function is generating the WWM file, it is convenient to set up the WWG input format based on the previous WWM input [3] (in the same branch of OpenMC v0.12.0 in [3]). The input for the WWG function in the simulation setup file (<settings.xml>) has three parts, as shown in Fig. 2.

1. The first part in red is the same as the WWM input [3], while an addition option "3" is used to call the WWG function. Currently, the only supported mesh is rectangular ones.
2. The second part in blue is the space boundary and energy group parameters for the generated WWM. These parameters are identical to those in the WWM input [3].
3. The third part in orange is the target region for the WWG, given by <lower_left> and <upper_right>.

After generating WWM, the user could change the weight window option from type "3" to type "0" and replace the <target> section (orange part) with the generated WWM for the subsequent calculation [3].

3. Benchmarks and results

A zero "Total score" will result in zero importance based on equation (1). Thus, the WWG function could not estimate the importance and give the lower WW bound with fewer particle histories used in WWG-run (NPS_{WWG} for short). And one could also deduce from the principle of the WWG function that with more NPS_{WWG}, WWG will generate a better importance map and WWM. However, the simulation time of the WWG-run will also increase with the increase of NPS_{WWG}, making the setup of NPS_{WWG} quite ambiguous. In addition, with the generated WWM above, the

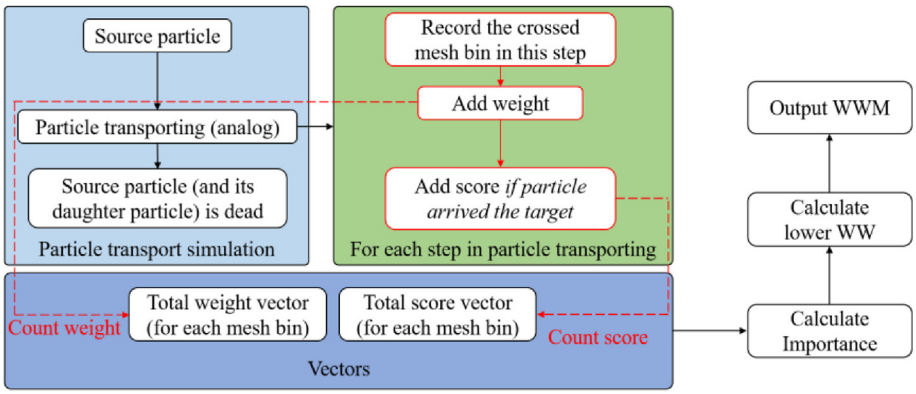

Fig. 1. The workflow of WWG function in OpenMC.

```
<weightwindow>
  <type>3</type>                    Weight Window Generator option
  <origin>0 0 0</origin>
  <xmesh>10 20</xmesh>
  <xints>10 10</xints>              Mesh cell boundaries
  <ymesh>10 20</ymesh>
  <yints>10 10</yints>
  <zmesh>10 20</zmesh>
  <zints>10 10</zints>
  <energy>
    <neutron>1e6 2e6 10e6</neutron>  Energy groups
    <photon>1e6 5e6</photon>
  </energy>
  <target>
    <lower_left>-2 -2 20</lower_left>
    <upper_right>2 2 25</upper_right>  Target region
  </target>
</weightwindow>
```

Fig. 2. The WWG input adapted into the OpenMC input file < settings.xml>.

particle histories used in the subsequent WWM-run (NPS_{WWM} for short) are also uncertain. WWM-run could produce better statistics with more NPS_{WWM}. But how much NPS_{WWM} is needed for results with wanted statistical error? These issues have received little attention in the previous literature. Thus, this work will focus on the relationship between the generated WWM, the final results, and the two particle histories (NPS_{WWG} and NPS_{WWM}).

The source-detector problem was simplified to a one-dimensional shielding problem in this work. Problems with different point sources and shielding material were used for benchmarking the OpenMC WWG function. In addition, the results of WWM-run based on different WWM (generated based on different NPS_{WWG}) and NPS_{WWM} were compared to the results of the analog run. For OpenMC neutron transport, the FENDL-3.1d [9] nuclear cross-section data library was used, converting from ACE format to HDF5 format [10]. All simulation is done under multi-thread mode with OpenMP, and the WWG function works very well in parallel.

3.1. Single-material benchmarks

Four single-material cases with various shielding and sources were tested in this benchmark. Table 1 describes these four cases, and Fig. 3 illustrates the structures of models with different source angular distributions. A point neutron source and a detector are located at the left and right center of shielding, respectively. Only particles with momentum toward the shielding are simulated for the isotropic source, as other particles have no contribution to the detector. The average volumetric neutron fluxes in the detector behind the shielding are tallied in the VITAMIN-J (175) energy group structure [11].

The results of WWM-run based on different WWM (generated based on different NPS_{WWG}) and NPS_{WWM} were compared to the analog run for all cases. In contrast to NPS_{WWM}, the particle histories used in the analog run are referred to as NPS. The statistical error of the average volumetric neutron fluxes was quantified by calculating the average relative error of each energy bin:

$$\sigma_{average} = \frac{\sum_i \sigma_i}{M} \tag{2}$$

where M is the number of energy groups, and σ_i is the relative error of the fluxes in the i-th energy group. Fig. 4 compares the $\sigma_{average}$ of four cases based on different WWM (represented by $WWG\text{-}NPS$) and NPS_{WWM} with analog runs (with NPS 100 times NPS_{WWM}). As can be seen, the WWM generated from the newly-developed OpenMC-WWG function does improve the results compared with analog runs in all cases. In addition, using more NPS_{WWG} in the WWG-run will result in a better-improving WWM (therefore, smaller $\sigma_{average}$). However, as the NPS_{WWG} increases, the improvement of WWM approaches a limit (minimum $\sigma_{average}$), implying that the NPS_{WWG} should not be as much as possible. Increasing the NPS_{WWG} after the minimum $\sigma_{average}$ has been obtained is just a waste of time and resources. Thus, a minimum $\sigma_{average}$ value for WWM-run exists with a given NPS_{WWM}. In other words, there is also a minimum NPS_{WWM} that must be used to achieve a satisfied $\sigma_{average}$ in WWM-run.

Table 2 shows the simulation time in WWM-runs of different cases. The difference in the simulation time per 10^5 source particles under different WWM (represented by $WWG\text{-}NPS$) is compared because the simulation time scales linearly with NPS_{WWM}. As can be seen, the simulation time per 10^5 source particles decreases with the NPS_{WWG} increasing generally. In addition, there is also a minimum simulation time for WWM-run with a particular NPS_{WWM}, which is more visible in Case 3 and Case 4.

The Figure of Merit for LVR (FOM_L) defines as a measurement of the computation efficiency, analogical to the FOM_G for GVR [4]:

$$FOM_L = \left(\sum_i \frac{\sigma_i^2}{M} T \right)^{-1} \tag{3}$$

where M is the number of energy groups, σ_i is the relative error of the fluxes in the i-th energy group, and T denotes the computational time of the simulation in CPU $\times$ min. The simulation takes less computer time to reach a given relative error or produces better results in the same amount of computer time with a higher FOM_L value. Thus, a higher FOM_L value indicates that the simulation is more efficient. Fig. 5 compares the calculated FOM_L of WWM-run based on different WWM (represented by $WWG\text{-}NPS$) and NPS_{WWM} with analog runs. For analog runs, the FOM_L value drops sharply as the NPS rises at first and then remains approximately constant. The sharp decrease is caused by some seldom-sampled particle paths that significantly affected the tally result and relative error estimate. With NPS increasing, these seldom-sampled particle paths

Table 1
The neutron source and shielding of 4 cases in the single-material benchmarks.

Cases	Source energy distribution	Source angular distribution	Shielding
1	14 MeV monoenergetic	monodirectional	100 cm concrete @ 2.3 g/cm^3
2	0–14 MeV uniform distribution	monodirectional	100 cm concrete @ 2.3 g/cm^3
3	14 MeV monoenergetic	isotropic	100 cm concrete @ 2.3 g/cm^3
4	14 MeV monoenergetic	monodirectional	70 cm iron @ 7.87 g/cm^3

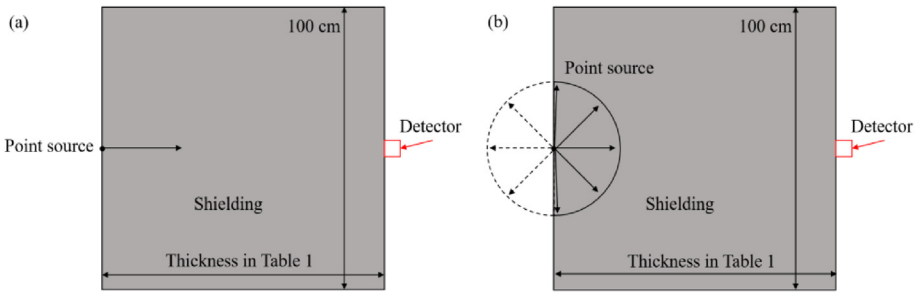

Fig. 3. The geometry model of single-material benchmarks with (a) monodirectional source and (b) isotropic source.

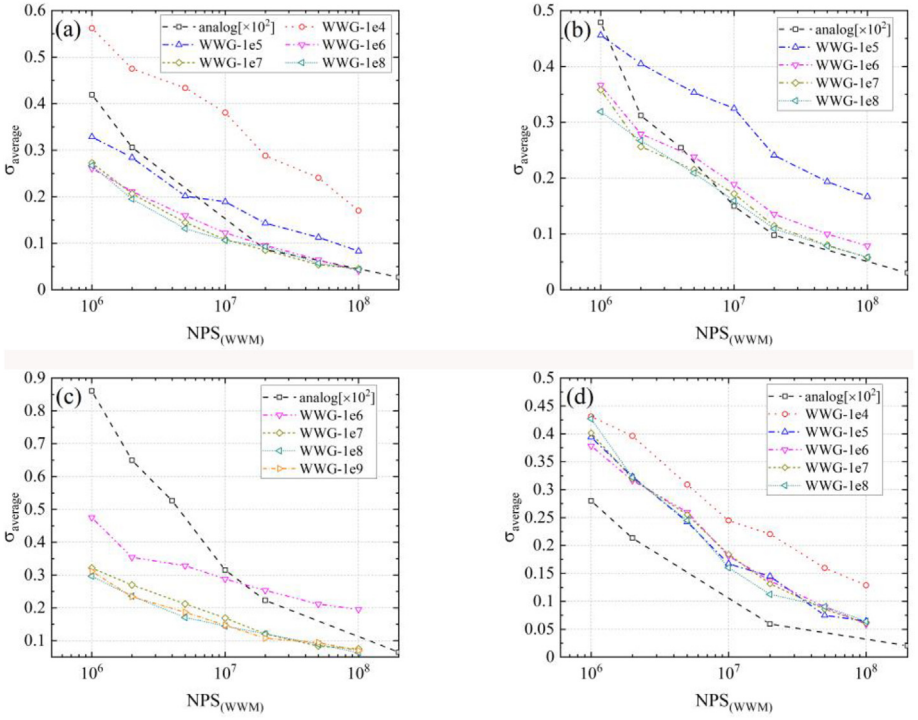

Fig. 4. The $\sigma_{average}$ of (a) case 1, (b) case 2, (c) case 3 and (d) case 4 based on different WWM (*WWG-NPS*) and NPS_{WWM}, compared with results of analog runs (with *NPS* 100 times NPS_{WWM}).

Table 2
The simulation time in WWM-runs of four cases based on different WWM (*WWG-NPS*).

WWM *(WWG-NPS)*	Simulation Time (s/10^5 source particles)			
	Case 1	Case 2	Case 3	Case 4
WWG-10^4	6.1312	–	–	5.0343
WWG-2 × 10^4	6.8769	–	–	3.6610
WWG-5 × 10^4	7.2832	–	–	2.4052
WWG-10^5	10.427	9.2764	–	1.7041
WWG-2 × 10^5	8.7350	7.6747	–	1.3753
WWG-5 × 10^5	6.5888	5.9365	13.599	1.2023
WWG-10^6	4.9972	4.8774	11.588	1.1496
WWG-2 × 10^6	3.9065	3.9555	9.7901	1.0788
WWG-5 × 10^6	2.8619	3.0844	6.1667	1.0665
WWG-10^7	2.4147	2.5760	5.0721	1.0787
WWG-2 × 10^7	2.2725	2.2882	4.2772	1.0653
WWG-5 × 10^7	2.1875	2.0770	4.0129	1.0694
WWG-10^8	2.1352	1.9719	3.8147	1.0689
WWG-2 × 10^8	–	–	3.7236	–
WWG-5 × 10^8	–	–	3.607	–
WWG-10^9	–	–	3.6041	–

are sampled more frequently, and the FOM_L value tends to be stable [5]. For WWM-runs, the FOM_L value also decrease with the increase of NPS_{WWM}. In addition, the more NPS_{WWG} is used to generate the WWM, the better the WWM will improve, and the higher the FOM_L value will be. The decrease in the FOM_L will also be more gradual with a better WWM. However, as demonstrated in Fig. 5, there is also a maximum FOM_L value under a given NPS_{WWM}.

However, the FOM_L given in Fig. 5 does not account for the time spent on WWG-run. Therefore, the $FOM_{L\text{-}adjusted}$, which considers the time of WWG-run, defines as:

$$FOM_{L-adjusted} = \left(\sum_i \frac{\sigma_i^2}{M} (T_{WWG} + T_{WWM}) \right)^{-1} \tag{4}$$

where T_{WWG} is the simulation time for WWG-run, and T_{WWM} is the simulation time for WWM-run. Fig. 6 illustrates the calculated $FOM_{L\text{-}adjusted}$ of all cases, compared with the FOM_L of analog runs. When smaller NPS_{WWG} is used, the T_{WWG} is much less than the T_{WWM}. Thus, the $FOM_{L\text{-}adjusted}$ exhibits a similar tendency as FOM_L

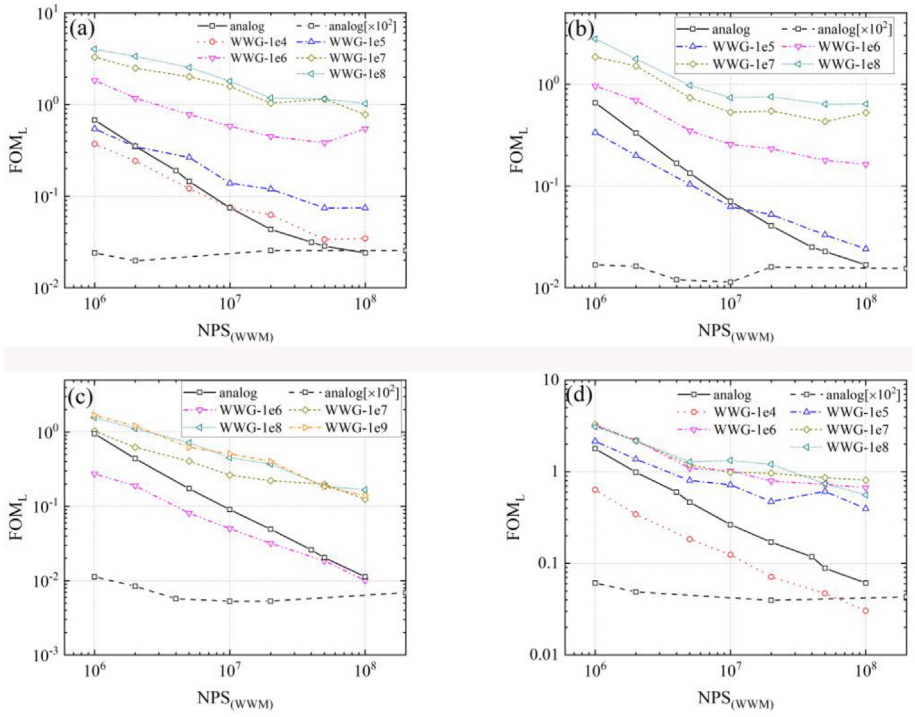

Fig. 5. The FOM_L of (a) case 1, (b) case 2, (c) case 3 and (d) case 4 based on different WWM (*WWG-NPS*) and NPS_{WWM}, compared with results of analog runs (with different *NPS*).

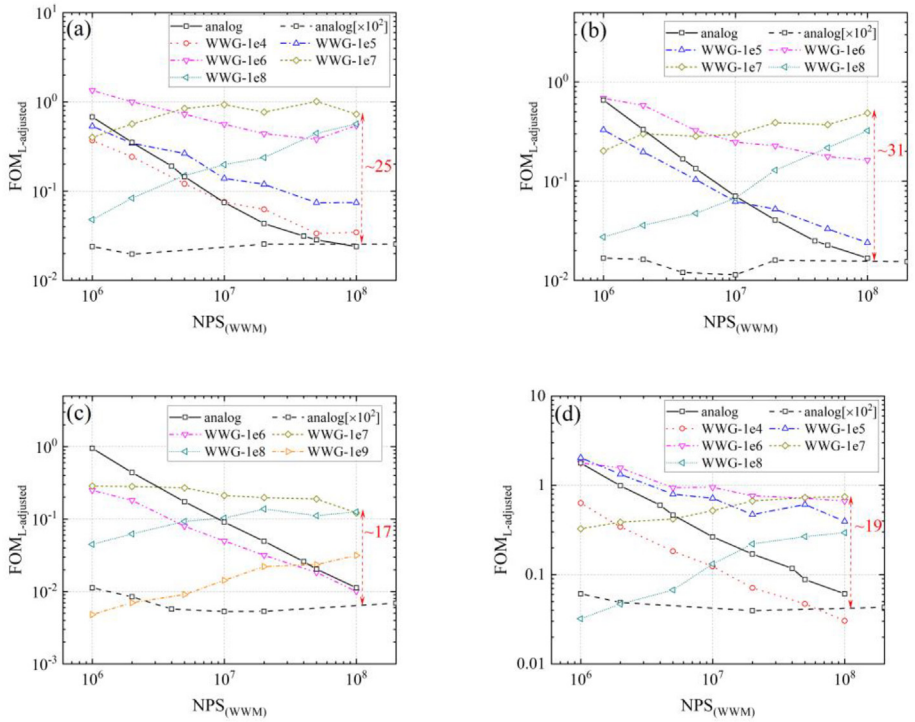

Fig. 6. The $FOM_{L\text{-}adjusted}$ of (a) case 1, (b) case 2, (c) case 3 and (d) case 4 based on different WWM (*WWG-NPS*) and NPS_{WWM}, compared with results of analog runs. The maximum speedup factor under $NPS_{WWM} = 10^8$ is also shown.

with NPS_{WWM} increasing, based on equation (5):

$$\left(\sum_i \frac{\sigma_i^2}{M}(T_{WWG}+T_{WWM})\right)^{-1} \approx \left(\sum_i \frac{\sigma_i^2}{M}T_{WWM}\right)^{-1} = FOM_L \quad (5)$$

With NPS_{WWG} increasing, the T_{WWG} rises as well, and the T_{WWM} falls (due to better WWM, as shown in Table 2). When the increase in T_{WWG} exceeds the reduction in T_{WWM}, the T_{WWG} is dominant and leading to:

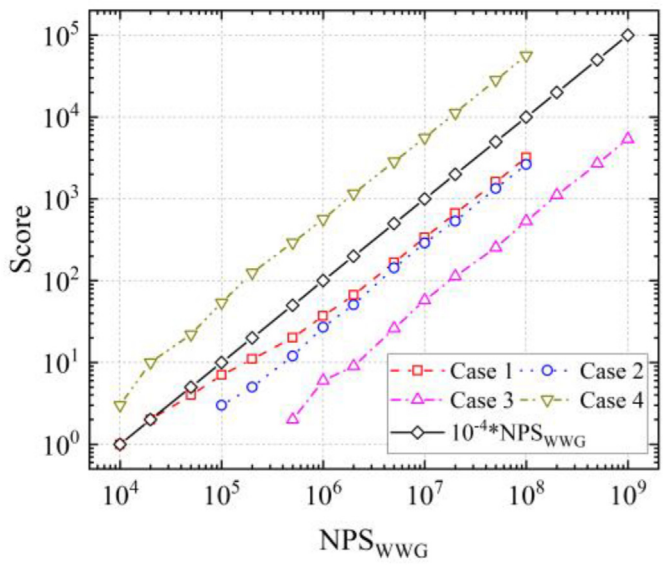

Fig. 7. The relationship between the *Score* and the NPS_{WWG}.

$$\left(\sum_i \frac{\sigma_i^2}{M}(T_{WWG}+T_{WWM})\right)^{-1} = \left(\sum_i \frac{\sigma_i^2}{M} T_{WWM}\left(1+\frac{T_{WWG}}{T_{WWM}}\right)\right)^{-1} = FOM_L\left(1+\frac{T_{WWG}}{T_{WWM}}\right)^{-1} \quad (6)$$

As a result, the $FOM_{L\text{-}adjusted}$ will grow with T_{WWM} (equivalent, NPS_{WWM}) increasing and reduce with T_{WWG} (equivalent, NPS_{WWG}) increasing as shown in Fig. 6. Since a maximum value of the FOM_L exists under a given NPS_{WWM} (as shown in Fig. 5), there will also be a maximum $FOM_{L\text{-}adjusted}$. Once the maximum FOM_L is reached, raising NPS_{WWG} will increase T_{WWG} and decrease the $FOM_{L\text{-}adjusted}$, as shown in Fig. 6. Fig. 6 also shows the maximum speedup factor, which varies depending on the cases.

Based on the above discussion, there are two questions on the setup of NPS_{WWG} and NPS_{WWM}: 1). How many NPS_{WWG} can generate WWM with the maximum improvement (the maximum $FOM_{L\text{-}adjusted}$)? 2). How many NPS_{WWM} are required to produce results with satisfied σ_{wanted}? Let's discuss these two questions one by one.

The NPS_{WWG} was used to generate WWM with the highest $FOM_{L\text{-}adjusted}$ various between cases, as illustrated in the above results. Thus, instead of particle histories (NPS_{WWG}), another value should be used to compare between cases. Because the importance (equivalent, the lower WW bound) is related to the "Total score" in each mesh bin, the "Total score" in the target region (*Score*, for short) is chosen for comparison. First, Fig. 7 depicts the relationship between the *Score* and the NPS_{WWG} in all cases. A line with $Score = 10^{-4} \times NPS_{WWG}$ is also given in Fig. 7 for comparison. In the log-log plot, the slope of *Score* with NPS_{WWG} is the same as the reference line in all cases, indicating that *Score* is generally proportional to the NPS_{WWG}. Thus, the *Score* value can be used to compare between different cases, since it is not only equivalent to the NPS_{WWG} but also related to the generated WWM.

The *Score* is used in the following discussion. Fig. 8(a) displays the $\sigma_{average}$ value of different WWM-runs with varying WWM (*Score*) under the same NPS_{WWM} (10^8). And the ratio of simulation time under various WWM (*Score*) and the minimal simulation time (which is the simulation time under the highest NPS_{WWG} in Table 2 for each case), is given in Fig. 8(b). As can be seen, the tendency of the $\sigma_{average}$ and time ratio altered with *Score* is similar for all cases. The $\sigma_{average}$ value tends to be stable after the *Score* exceeds 100, with some statistical fluctuations, while the critical *Score* value for time ratio is around 1000. Thus, both the $\sigma_{average}$ and simulation time are improved with the WWM with *Score* below 100. And the WWM could only enhance the simulation speed with *Score* between 100 and 1000. After the *Score* reaches 1000, both the $\sigma_{average}$ and simulation time reach their limit, and no further improvement could achieve. So, the answer to question 1) is: NPS_{WWG} with *Score* about 1000 can generate WWM with the maximum improvement.

Returning to question 2), the minimum $\sigma_{average}$ value (σ_{limit}) is defined as the average of the $\sigma_{average}$ value in the stable range (*Score* > 100). Fig. 9 shows the relationship between the σ_{limit} value and NPS_{WWM} for all cases. The fitting curves reveal that their relationship can be represented as a negative exponential function. The power exponent varies depending on the cases but within −0.3 ~ −0.4 for these benchmark cases. Therefore, a negative exponential function with a power exponent of −0.35 might be used to estimate the minimal NPS_{WWM} required to a desire σ_{wanted}:

$$\sigma_{limit} \propto NPS_{WWM}^{-0.35} \quad (7)$$

The statistical error of results is generally proportional to $NPS^{-0.5}$ in the analog run. The principal consideration under this relation (σ is proportional to $NPS^{-0.5}$) is the assumption that all particle histories in the simulation are independent, which is correct in the analog run. When using WWM, however, the particle will split into many particles. For these divided particles, they share the same father particle (and the same history before split). As a result, this assumption (that all particle histories in the simulation are independent) is no longer valid, leading to the varied power exponent in Fig. 9.

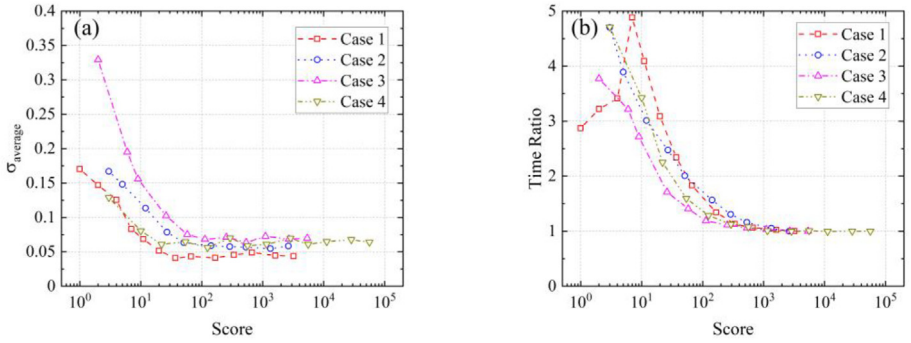

Fig. 8. The (a) $\sigma_{average}$ value and (b) ratio of the simulation time and the minimum simulation time (which is the simulation time under the highest *WWG-NPS* in Table 2 for each case), under the same NPS_{WWM} (10^8) and different WWM (*Score*).

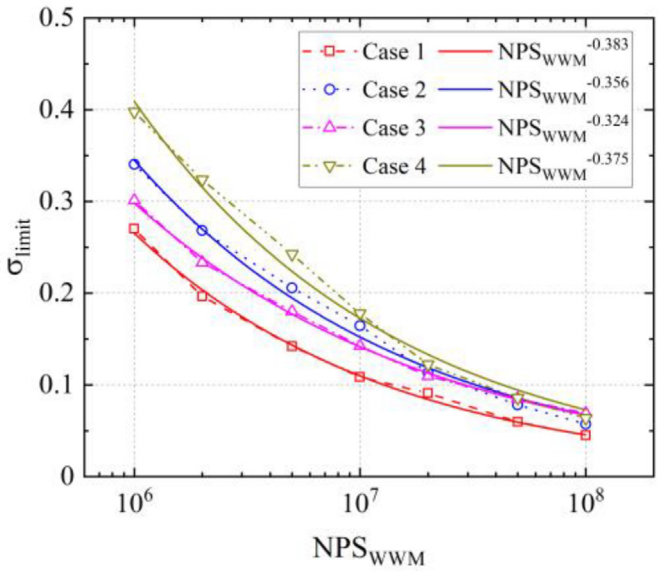

Fig. 9. The σ_{limit} value changes with the NPS_{WWM} for different cases.

To summarize, the following steps are suggested for determining the NPS_{WWG} and NPS_{WWM}:

1. Make a WWG-run with a small NPS_{WWG} (for example, 10^6) to determine the relationship between the *Score* and the NPS_{WWG};
2. Estimate the NPS_{WWG} with *Score* about 1000 based on the relationship in step 1, and generate the best WWM with estimated NPS_{WWG};
3. Make a WWM-run based on the generated WWM above with a small NPS_{WWM} (for example, 10^6) to get the σ_{limit} value;
4. Estimate the minimum NPS_{WWM} to achieve a desired σ_{wanted} based on equation (7) and the NPS_{WWM}-σ_{limit} pair in step 3, make a final WWM-run with the estimated NPS_{WWM} above to obtain the results.

The final WWM-run could achieve the maximum $FOM_{L\text{-}adjusted}$ with estimated NPS_{WWG} and NPS_{WWM} in this instruction. Excessive NPS_{WWG} will waste computing resources and lower computation efficiency, while insufficient NPS_{WWG} could not generate the WWM with maximum improvement. In addition, WWM-run with insufficient NPS_{WWM} could not obtain results with the desired statistics. It is important to notice that the relationship between the σ_{limit} value and NPS_{WWM} does not follow equation (7) strictly. Thus, one could use a higher power exponent (−0.3, for example) for a conservatively estimated NPS_{WWM} in step 4.

3.2. Multi-material benchmarks

In this multi-material benchmark, two cases with different shielding and sources have been conducted, which take a similar

Table 3
The source and shielding of two cases in the multi-material benchmarks.

Cases	Source energy distribution	Source angular distribution	Shielding
1	14 MeV monoenergetic	monodirectional	40 cm concrete @ 2.3 g/cm^3; 30 cm iron @ 7.87 g/cm^3; 30 cm water @ 1.0 g/cm^3
2	0–14 MeV uniform distribution	monodirectional	20 cm SS316 @ 7.93 g/cm^3; 20 cm water @ 1.0 g/cm^3; 20 cm iron @ 7.87 g/cm^3; 20 cm concrete @ 2.3 g/cm^3

Table 4
The steps on determining the NPS_{WWG} and NPS_{WWM}, and the speedup factor for two multi-material benchmark cases.

	Case 1	Case 2
Step 1	With $NPS_{WWG} = 10^6$, *Score* = 66;	With $NPS_{WWG} = 10^6$, *Score* = 47;
Step 2	*Score* = 1000, $NPS_{WWG} \approx 1.5 \times 10^7$; Set $NPS_{WWG} = 2 \times 10^7$, *Score* = 1181;	*Score* = 1000, $NPS_{WWG} \approx 2.1 \times 10^7$; Set NPS_{WWG} as 2×10^7, *Score* = 1080;
Step 3	With $NPS_{WWM} = 10^6$, $\sigma_{limit} \approx 0.397$;	With $NPS_{WWM} = 10^6$, $\sigma_{limit} \approx 0.306$;
Step 4	Calculate based on equation (7): $\sigma_{wanted} = 0.1$, $NPS_{WWM} \approx 5.12 \times 10^7$; Set $NPS_{WWM} = 5 \times 10^7$, results: $\sigma_{limit} \approx 0.067$;	Calculate based on equation (7): $\sigma_{wanted} = 0.1$, $NPS_{WWM} \approx 2.44 \times 10^7$; Set $NPS_{WWM} = 3 \times 10^7$, results: $\sigma_{limit} \approx 0.092$;
Speedup	~22	~12

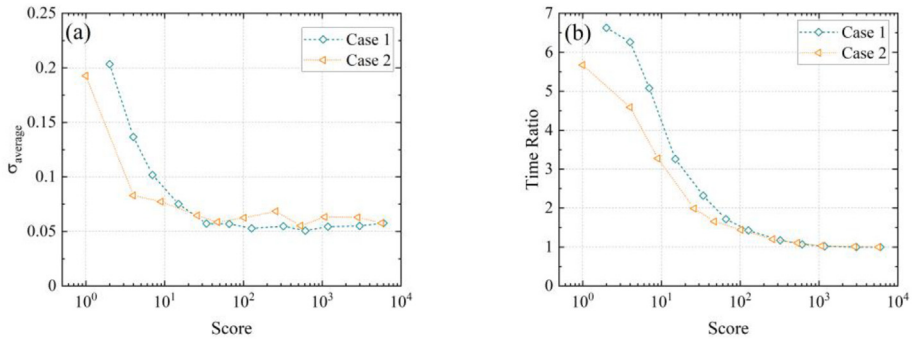

Fig. 10. The (a) $\sigma_{average}$ value and (b) ratio of the simulation time and the minimum simulation time (which is the simulation time under the highest *Score* for each case), under the same NPS_{WWM} (10^8) and different WWM (*Score*).

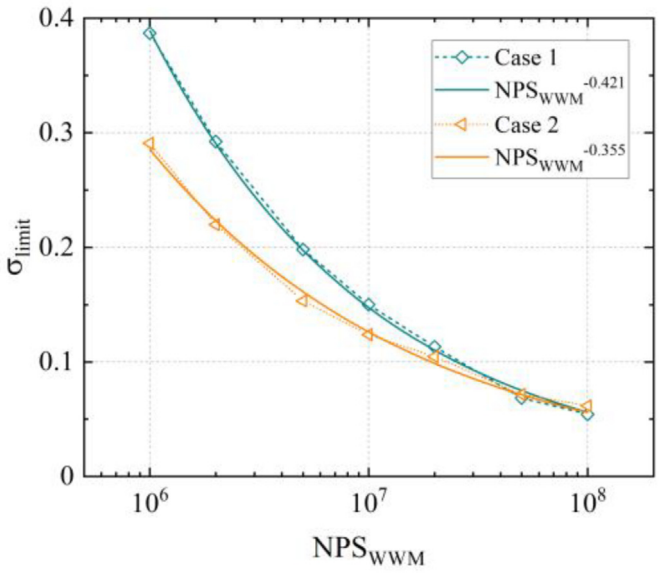

Fig. 11. The σ_{limit} value changes with the NPS_{WWM} for different cases.

setup as the single-material benchmark above. The same structures (Fig. 3) are used in these two cases, and the description of these two cases is given in Table 3. The average volumetric neutron fluxes in the detector behind the shielding are also tallied in the VITAMIN-J (175) energy group structure [11]. This benchmark also employs the previously suggested four-steps instruction for determining the NPS_{WWG} and NPS_{WWM}. The specific calculation steps and the final speedup factor are shown in Table 4 for two multi-material cases.

For test purposes, WWM-runs based on different WWM (generated by different NPS_{WWG}) and NPS_{WWM} were simulated. The $\sigma_{average}$ value and the similar simulation time ratio under different WWM (*Score*) but the same NPS_{WWM} (10^8), as given in Fig. 10, show a similar tendency as the single-material benchmark cases. Thus, the determined NPS_{WWG} in Table 4 does generate the WWM with the maximum improvement and $FOM_{L\text{-}adjusted}$. Fig. 11 shows the relationship between the σ_{limit} value and NPS_{WWM} in this benchmark. The negative exponential relationship is also shown by the fitting curves. In addition, the power exponents of two multi-material cases are lower than the power exponent (−0.35) utilized in equation (7). Thus, the obtained σ_{limit} value with determined NPS_{WWM} is better than σ_{wanted} (0.1), confirming that one can use a higher power exponent for a conservatively estimated NPS_{WWM}.

4. Conclusions

In this work, the Weight Window Generator function is developed and implemented in a developmental branch of the OpenMC (v0.12.0) Monte Carlo code. This developmental branch will be released and merged into the main distribution in the future. With the WWG function, OpenMC could generate the WWM for the subsequent calculation without any third-party codes. This function has been benchmarked with a series of single-material cases and multi-material cases. Results of the single-material benchmark show that using more NPS_{WWG} in WWG-run could generate better WWM. However, the minimum $\sigma_{average}$, minimum simulation time, and maximum FOM_L for a WWM-run with given NPS_{WWM} indicate a limit of the improvement when employing WWM. This limit also causes the maximum $FOM_{L\text{-}adjusted}$ when considering the simulation time of the WWG-run. The "Total score" in the target region (*Score*), instead of NPS_{WWG}, is chosen to compare the WWM between various cases. The *Score* is not only equivalent to the NPS_{WWG} but also related to the generated WWM, making it suitable for assessing the WWM between different cases. Comparing results show that the $\sigma_{average}$ value tends to be stable after the *Score* exceeds 100, while the critical threshold for the simulation time is around 1000. Based on this, instruction with four steps is suggested to determine the NPS_{WWG} and NPS_{WWM} and tested by two multi-material benchmark cases. These results demonstrate the ability of the OpenMC WWG function and the instruction for the source-detector problem.

However, the WWM used in this work has only considered one energy group. Further research on the generated WWM with multiple energy groups is needed. In addition, the source-detector problem is simplified to a one-dimensional problem with a point source in this work. The method of determining the NPS_{WWG} and NPS_{WWM} needs also be investigated with more complex sources and geometry in further work.

CRediT authorship contribution statement

Yuan Hu: Writing - original draft, Investigation. Sha Yan: Writing – review & editing, Supervision, Resources. Yuefeng Qiu: Writing – review & editing, Supervision, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Part of the analysis was performed on the High Performance Computing Platform of the Center for Life Science (Peking University).

References

[1] Paul K. Romano, et al., OpenMC: a state-of-the-art Monte Carlo code for research and development, Ann. Nucl. Energy 82 (2015) 90–97.
[2] https://docs.openmc.org/en/stable/.
[3] Y. Hu, et al., Development and benchmarking of the weight window mesh function for OpenMC, Fusion Eng. Des. 170 (2021), 112551.
[4] Y. Hu, et al., Implementation and benchmarking of an automatic global variance reduction method on OpenMC, Fusion Eng. Des. 173 (2021), 112829.
[5] X-5 Monte Carlo Team, MCNP – A General Monte Carlo N-Particle Transport Code, Version 5, 2003. LA-UR-03-1987.
[6] https://github.com/openmc-dev/openmc.
[7] Thomas E. Booth, et al., Importance estimation in forward Monte Carlo calculations, Nucl. Technol. Fusion 5 (1) (1984) 90–100.
[8] Thomas E. Booth, Sample Problem for Variance Reduction in MCNP, LA-10363-MS, 1985.
[9] FENDL, Fusion evaluated nuclear data library. https://www-nds.iaea.org/fendl/.
[10] convert_fendl.py in, https://github.com/openmc-dev/data.
[11] The FISPACT-II User Manual, UKAEA-R, 2016 (11)11 Issue 8.