# SCONE: AN OPEN-SOURCE MONTE CARLO NEUTRON TRANSPORT CODE FOR RESEARCH AND TEACHING

M.A. Kowalski1, P.M. Cosgrove1, V. Raffuzzi1, N. Ravoisin1, E. Shwageraus1

1Cambridge University Engineering Department, Cambridge, United Kingdom

Email contact of corresponding author: mikolaj.kowalski@cantab.net

**1. INTRODUCTION**

**S**tochastic **C**alculator **O**f the **N**eutron Transport **E**quation (SCONE) is an open-source Monte Carlo particle transport code available under the MIT licence. Its development began in late 2017 in the Nuclear Energy Group at the Cambridge University Engineering Department. The reason for the project was an observation that Monte Carlo codes used in nuclear engineering fall broadly into two groups. One is production-level codes designed for maximum computational efficiency and thoroughly validated against experimental benchmarks. The other is relatively simple, toy codes used to experiment with new methods and algorithms, which often rely on a simplified, few energy group description of nuclear data and scattering kinematics. This leaves space for a third category of Monte Carlo codes, whose aim would be to allow for easy prototyping of new algorithms and modification, while retaining a realistic description of nuclear interactions and problem geometries. Furthermore, such codes could be used for Master and PhD level research, thus increasing the availability of specialists in Monte Carlo neutron transport, which is desirable due to the growing importance of stochastic methods in reactor physics calculations. This niche has already been identified by codes such as MCATK [1] and PATMOS [2]. The aim of SCONE is to provide a free and open-source alternative in that niche.   
  
The general description of SCONE is available in [3]. To increase accessibility for engineering students with limited programming experience, SCONE is written in Fortran 2008 which, by limiting the number of features, manages to combine reasonable performance with a shallow learning curve. To enable easier modifications, SCONE relies on object-orientation and polymorphism via a plug-in architecture, where major components, such as nuclear data, are accessed via well-defined interfaces implemented as abstract classes (or derived types in Fortran terminology).

The purpose of the presentation at the ONCORE meeting is three-fold. The first goal is to share with the open-source nuclear community the experience of setting up a modern Fortran project, which tries to follow best modern development practices related to automated testing, continuous integration, and documentation. In addition, the SCONE codebase contains many general-purpose components and procedures that might be utilised by other Fortran projects. The second purpose is to demonstrate how SCONE was successfully used to prototype and investigate novel algorithms. Lastly, it is to seek feedback and advice in order to improve the code in the future.

**2. SOFTWARE ENGINEERING**

**2.1 Automated Testing and Continuous Integration**

The source code of SCONE is hosted on Bitbucket [4] and the continuous integration (CI) service is provided by the Bitbucket Pipelines [5] functionality, which offers a 500 build-time minutes per month for academic users. This particular CI service was mainly chosen for convenience of integration with the repository. However, it is likely to be changed in the near future. First of all, since the free computational time available is limited, currently the CI pipeline is configured to be triggered manually to conserve resources. Secondly, the Bitbucket Pipelines service is rather niche and is not supported by third party services such as e.g. coveralls.io (which provides public test coverage reports).

The automated testing in SCONE is performed with the pFUnit 3 test framework [6]. Although version 4 is available, it is not used in SCONE to retain support for versions 6 & 7 of gfortran’s compiler, which are not supported by pFUnit 4. Since pFUnit must be compiled with the same compiler as the Fortran code it is testing, custom Docker images are used for CI, which are hosted on dockerhub.com. Currently build and testing is performed for gfortran 6, 7 & 8; version 9.3 is not supported due to a compiler bug (segmentation error) when using Fortran’s “finalisation” features.

The tests are currently split into two suites. The “unit tests suite” contains fast-running tests, whose primary purpose is to enforce specification of different code components (derived types & procedures). In a departure from the usual unit test practice, the unit tests are not orthogonal (a single unit test may check for multiple faults). This decision was made to simplify writing tests and thus encourage novice users (students) to employ them. Naturally, the drawback is that identifying a test’s cause(s) of failure becomes more involved. The other suite of “integration tests” is also implemented with pFUnit and contains tests that involve reading/writing files.

**2.2 Documentation Style**

Most of the documentation of SCONE is contained in the source files in the form of documentation comments, which are placed before the definition of each derived type or procedure to explain its purpose, parameters, and, in particular, behaviour for edge cases. Example of the documentation comment is shown in Fig. 1. The style was inspired by the Google style for Python docstrings. More details are available in [3] and [7].

Important information that is unsuitable to be stored in the source files is contained on the SCONE “Read The Docs “ website [7]. At the present time it is largely incomplete, but is intended to explain the intention behind various software design decisions and attempt to provide a more in-depth look into practical implementation details than the standard references on the neutron transport Monte Carlo codes e.g. [8].

|  |
| --- |
|  |
| *FIG 1: Example of a documentation comment of a deferred type-bound procedure.* |

**2.3 General Purpose Components**

In order to allow users to easily visualise the defined geometry, SCONE was equipped with a capability to produce images in an uncompressed 24-bit per pixel BMP format. An example of the image is shown in Fig. 2. This functionality was implemented in pure Fortran 2008 without any external dependencies or compiler extensions. Thus, it can be easily transferred to other Fortran-based software projects.

|  |
| --- |
|  |
| *FIG 2: Material composition of a MOX assembly generated with SCONE’s BMP output.* |

To handle user input, SCONE uses a hierarchical dictionary data structure that associates string keywords with an entry that can be: a real number, an integer, a character string, a 1D array composed of one of the previous types, or another, nested dictionary. This setup allows for an easy way to configure objects that build the calculation (e.g. individual tallies). Each larger object (e.g. geometry) passes appropriate sub-dictionaries to the initialisation procedures of the lower level components (e.g. universes, surfaces). Thus, to add a new option, all the programmer needs to do is to add a call for an appropriate keyword. The ASCII input file for the dictionary is based on the one used in OpenFOAM [9].

For the programmer's convenience, SCONE contains hashmap data structures that can associate int-to-int and char-to-int. These are implemented using a Knuth hash and open addressing. To speed-up comparisons, the character string keywords are hashed with a 32bit FNV-1 function. Furthermore, the hashmaps are equipped with an easy interface to iterate over all key-value pairs.

**3. PRACTICAL USE CASES**

**3.1 Particle Injection**

SCONE’s flexibility has been used on several occasions to rapidly construct different calculation schemes, e.g., creating a simple time-dependent solver. One of the less traditional computational routes that was investigated is an alternative k-eigenvalue solver where the standard generational algorithm is modified to introduce neutrons during the simulation which are not descended from neutrons in previous generations. This modification is known as ‘particle injection’, and was proposed in an attempt to mitigate correlation and clustering phenomena from which the standard generational algorithm might suffer [10].

Such a scheme may prove difficult to implement in other Monte Carlo codes in full generality, as it required sampling new neutrons from some given distribution and treating them as ‘inactive’ (preventing them from contributing to tallies) during active cycles. However, applying different numerical treatments to different portions of the particle population is relatively straightforward in SCONE: analysing problems which simultaneously handle multi-group and continuous energy neutrons was part of SCONE’s original impetus [3]. SCONE also enabled easy construction of additional tallies necessary for such a study, such as the non-standard neutron centre-of-mass estimator.

**3.2 Negative weight tracking**

SCONE has recently been used to test the computational performance of several particle tracking methods [11,12]. Some of these are weighted delta tracking [13] and tracking with negative weights [14], both being variations of the standard Woodcock delta tracking. These algorithms had shown great promise in the original papers that proposed them. However, both had only been tested on ‘toy’ Monte Carlo codes, which could only handle over-simplified models.

The investigation of tracking with negative weights with SCONE was particularly interesting. The method consists of sampling the neutron mean free path with an arbitrary macroscopic cross-section, and applying acceptance-rejection sampling with an arbitrary acceptance probability. Then, to maintain an unbiased simulation, the particle weight is corrected, which could result in negative particle weights. In the original paper, this method was tested on a simplified photon transmission problem, yielding a substantial speed-up. Later on, it was applied to a one-speed problem [15], showing an increased variance compared to delta tracking but still retaining good performance. When implemented in SCONE, however, and applied to a criticality calculation in a realistic reactor geometry, the method proved completely impractical: the weight correction generated extremely high weights, which consequently overloaded the computer memory through the amount of fission neutrons generated. This happened regardless of the parameters chosen and applied population control methods. When fixing one of the two parameters to the physical value, the weights’ oscillations were partially reduced, but still produced extremely high variance results.

While these results could not be observed within simplified toy codes and models, SCONE allowed testing on realistic reactor models, finding otherwise unexpected challenges.

**3.3 Mesh support**

Particle tracking on mesh-based representations of geometry is one of SCONE’s most recent developments, which has been contributed via a Master’s project. OpenFOAM [9] was chosen to generate mesh geometries due to its open-source nature and the numerous existing tools to create or convert meshes to the OpenFOAM format. Mesh geometries are imported into SCONE during the overall model geometry build process: SCONE reads the OpenFOAM files corresponding to the mesh(es) to be included in the simulation and creates internal mesh model(s), which are then incorporated into host legacy constructive solid geometry (CSG) universes. This approach allows one to marry CSG- and mesh-based representations of geometry without the need to completely rewrite the model geometry build process. This, along with SCONE’s object-oriented nature, enables the easy integration of multiple mesh environments within a single simulation and/or pre-existing geometrical structures such as lattice universes.

In SCONE, meshes are represented as derived-types of a polymorphic, abstract class of objects. This allows users to experiment with different tracking algorithms and easily integrate mesh types not supported by OpenFOAM (e.g. Abaqus) into SCONE. Currently, both delta- and surface-tracking are supported for OpenFOAM meshes: both algorithms rely on the “particle-in-cell” test to locate particles within a mesh geometry. Because this test uses surface normals to determine the cell occupied by a particle, SCONE internally decomposes meshes into tetrahedral elements during the importation process. Moreover, due to the large number of tests performed per simulation cycle, the two tracking algorithms employ a highly-efficient k-dimensional (kd) tree structure to accelerate particle location routines.

The delta- and surface-tracking algorithms were validated on highly irregular geometries, including the famous Stanford Bunny [16], a model virtually impossible to construct using traditional CSG methods. Validation simulations showed (relatively) rapid run times, even for meshes containing several hundred thousand elements. These results will hopefully be presented at the upcoming PHYSOR 2022 conference.

**4. CONCLUSIONS**

At its core, SCONE remains fundamentally a student project. As a result, in its current state, it does not enjoy production-level code quality and thorough validation against experimental benchmarks. Although it showed reasonable accuracy in a code-to-code comparison [3] against Serpent [17], it is clear that SCONE is presently unsuitable for practical reactor calculations. However, as demonstrated by examples in Section 3, it has been successfully utilised in research on the development of new Monte Carlo techniques. Furthermore, being a software project developed in Fortran, the experience accumulated over its development may be of interest to the nuclear engineering open-source community, where Fortran is still a relatively popular language.

One of the more substantial present difficulties (which it is hoped this paper will help alleviate) is that SCONE’s user base consists only of Cambridge staff and students. This is natural, although the aspiration of the developer team is that SCONE will also be successfully used by other institutions for prototyping their own research ideas in Monte Carlo methods. To this end, we would strongly encourage feedback on SCONE and suggestions on how best to improve its appeal to Monte Carlo researchers.

**References**

1. ADAMS T. et al., “Monte Carlo Application ToolKit (MCATAK)”, Ann. Nucl.Energy **82** (2015) 41–47.
2. BRUN E., CHAUVEAU S., MALVAGI F., “PATMOS: A prototype Monte Carlo transport code to test high performance architectures”, M&C 2017 - International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering, Jeju, Korea, April 16-20, 2017.
3. KOWALSKI M.A., COSGROVE P., BROMAN J., SHWAGERAUS E., SCONE: A Student-Oriented Modifiable Monte Carlo Particle Transport Framework, J. Nucl. Eng. **2** (2021) 57-64.
4. SCONE Repository (2021), <https://bitbucket.org/Mikolaj_Adam_Kowalski/scone>
5. Bitbucket Pipelines (2021) <https://atlassian.com/software/bitbucket/features/pipelines>
6. RILEE M., CLUNE T, “Towards Test Driven Development for Computational Science with pFUnit”, 2014 Second International Workshop on Software Engineering for High Performance Computing in Computational Science and Engineering, New Orleans, USA, Nov. 21 2014.
7. SCONE Sphinx Documentation (2021), <https://scone.readthedocs.io>
8. BRIESMEISTER J.F., MCNP - A General Monte Carlo N-Particle Transport Code, technical report, LA-13709-M, Los Alamos national Laboratory, 2000.
9. OpenFOAM (2021) <https://openfoam.org>
10. COSGROVE P., KOWALSKI M.A., SHWAGERAUS E., PARKS G.T., “Countering neutron clustering in Monte Carlo with a neutron source injection”, Proc. International Conference on Physics of Reactors: Transition to a Scalable Nuclear Future, The European Physical Journal Conferences **247**(6) (2020).
11. RAFFUZZI V., SHWAGERAUS E., MORGAN L., “Weighted delta tracking performance evaluation in the presence of localized heavy absorbers”, Proc. Mathematics & Computational Methods Applied to Nuclear Science & Engineering, Raleigh, NC, Oct. 11-17, 2021.
12. RAFFUZZI V., SHWAGERAUS E., MORGAN L., “Woodcock delta tracking with arbitrary sampling parameters in fission and fusion reactor models”, Proc. Mathematics & Computational Methods Applied to Nuclear Science & Engineering, Raleigh, NC, Oct. 11-17, 2021.
13. MORGAN L., KOTLYAR D., “Weighted-delta-tracking for Monte Carlo particle transport”, Ann. Nucl. Energy, **85** (2015) 1184-1188.
14. MOLNAR B., TOLNAI G., LEGRADY D., “Variance Reduction and Optimization Strategies in a Biased Woodcock Particle Tracking Framework”, Nucl. Sci. Eng. **190** (2018).
15. BELANGER H., MANCUSI D., ZOIA A., “Review of Monte Carlo methods for particle transport in continuously-varying media”, Eur. Phys. J. Plus (2020) **135**: 877.
16. TURK G., LEVOY M., “Zippered Polygon Meshes from Range Images”, SIGGRAPH ‘94 (Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, New York, 1994), Association for Computing Machinery, New York, NY, USA (1994).
17. LEPPÄNEN J., PUSA M., VIITANEN T., VALTAVIRTA V., KALTIAISENAHO T., “The Serpent Monte Carlo code: Status, development and applications in 2013”, Ann. Nucl.Energy **82** (2015) 142–150.