

SCONE: Overview and status

Presented: Mikolaj Adam Kowalski
Developed: Nuclear Energy Research Group

Department of Engineering, University of Cambridge

What is SCONE

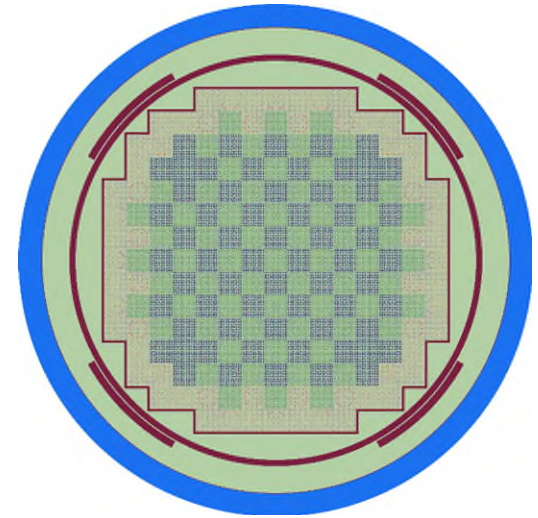
Stochastic Calculator Of Neutron Transport Equation



- Particle transport Monte Carlo code for nuclear engineering applications
- Target audience → research students
- Designed for modification: Object-Oriented, well-defined abstractions
- Use: Teaching, Prototyping of New Algorithms
- Prioritise modifiability over performance

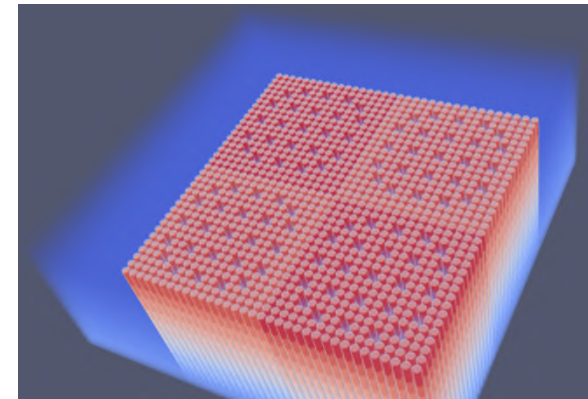
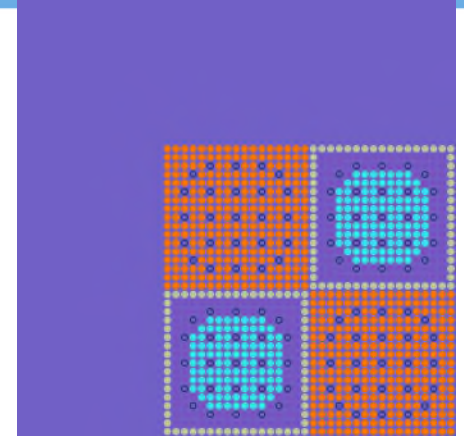
History and features

- Written in Fortran 2008:
 - Easy to learn & read without sacrificing performance
 - Informative compiler errors, easy-to-read standard
 - Reasonably well supported
- Automated testing:
 - Unit and integration tests with pfUnit framework
- Strict style guide
- Open-source: the only open-source reactor physics code in the UK
- Accessible at:
github.com/CambridgeNuclear/SCONE



Current features

- Standard Monte Carlo capabilities:
 - Transport with continuous energy and multi-group data
 - OpenMP parallelism
 - K-eigenvalue and fixed source
 - Full neutron physics (unresolved resonance and $S(\alpha, \beta)$)
 - Standard CSG representation – also mesh geometries
 - Standard and home-grown algorithmic acceleration techniques
- *Most* of photon transport
 - Photoelectric, pair production, Rayleigh + Compton
 - Final fixes on electron handling underway



Master projects

Experience with SCONE Masters projects

- Successful completion in short time (3 to 6m)
- Meaningful contribution to the development
- Positive feedback from the students 😊
- 'Hook' to fish people to join reactor physics community

Lessons learned:

- Students tend to stay quiet: can spend a lot of time struggling with problems easy to correct if they ask for help
- Necessary to enforce good style

Previous projects:

- Photon transport
- Unstructured meshes
- Alpha eigenvalue
- Photon-neutron coupling
- Implicit Monte Carlo
- Low population systems
- DBRC + OTF Doppler

Ongoing projects:

- CMFD acceleration
- Dynamic Monte Carlo

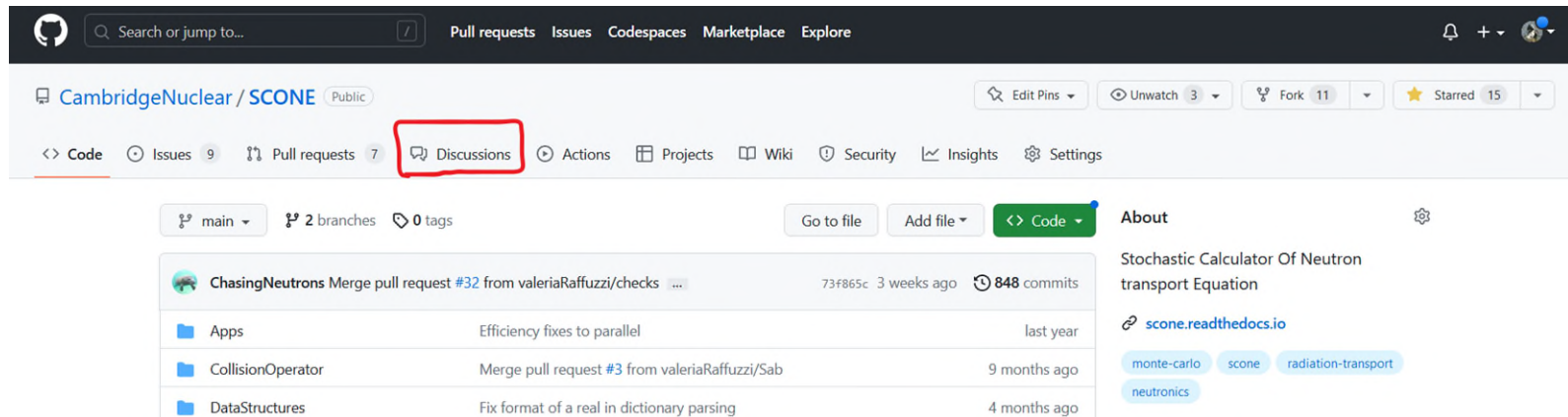
Proposed projects:

- HFR geometry modelling
- Power deposition models

'Not so greats': New user experience

Small (and) Cambridge-based user base

- Guide for compilation may be confusing, examples are sparse
- Within lab a minor problem
- **Externalise Q&A process. Github discussions!**



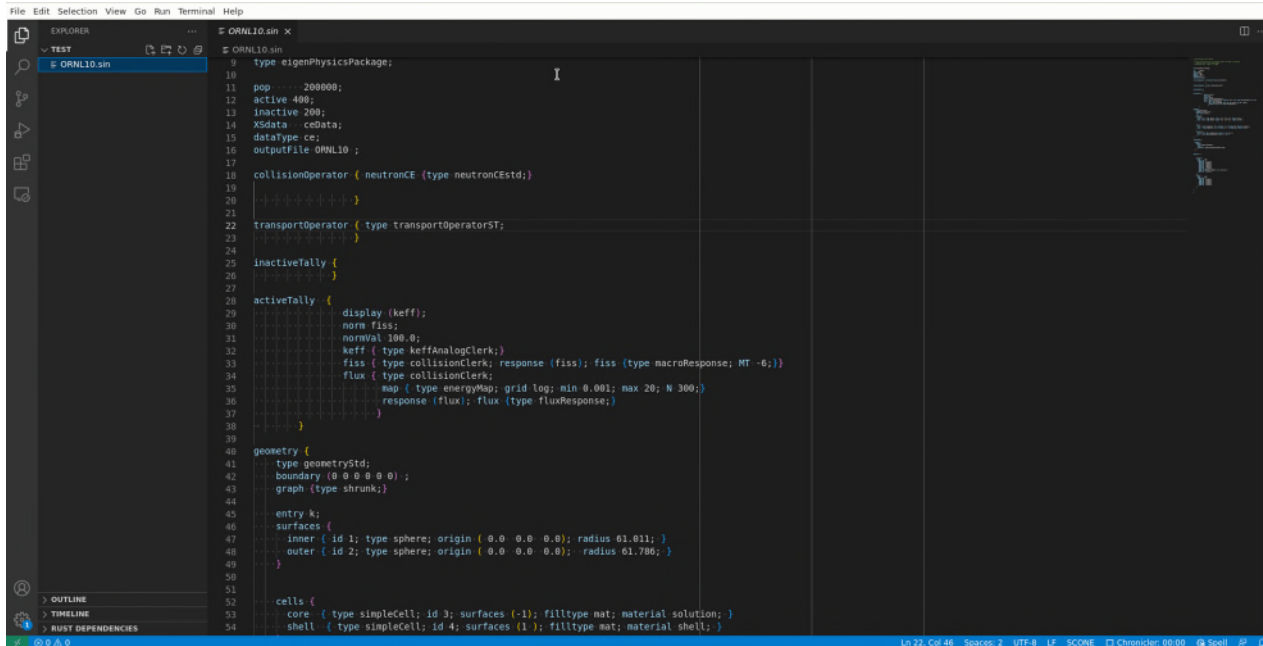
The screenshot shows the GitHub interface for the repository CambridgeNuclear/SCONE. The 'Discussions' tab is highlighted with a red box. The repository has 9 issues, 7 pull requests, and 0 discussions. The 'About' section describes it as a 'Stochastic Calculator Of Neutron transport Equation' with tags for monte-carlo, scone, radiation-transport, and neutronics.

File	Description	Last Commit
Apps	Efficiency fixes to parallel	last year
CollisionOperator	Merge pull request #3 from valeriaRaffuzzi/Sab	9 months ago
DataStructures	Fix format of a real in dictionary parsing	4 months ago

Work on generating user manual from in-source documentation comments is ongoing.

- Parsing and object-documentation association is working (using FORD)
- Next week: work on Sphinx domain and manual-generation will begin

'Not so greats': New user experience



```
9 type eigenPhysicsPackage;
10
11 pop 200000;
12 active 400;
13 inactive 200;
14 XSdata ceData;
15 dataType ce;
16 outputFile ORNL10 ;
17
18 collisionOperator ( neutronCE (type neutronCEstd);
19
20 }
21
22 transportOperator ( type transportOperatorST;
23
24 }
25
26 inactiveTally {
27
28 }
29
30 activeTally {
31   display (keff);
32   norm fiss;
33   normval 100.0;
34   keff ( type keffAnalogClerk;
35     fiss ( type collisionClerk; response (fiss); fiss (type macroResponse; MT -6;))
36     flux ( type collisionClerk;
37       map ( type energyMap; grid log; min 0.001; max 20; N 300;
38         response (flux); flux (type fluxResponse;
39
40       )
41     )
42   }
43
44 geometry {
45   type geometryStd;
46   boundary (0 0 0 0 0 0);
47   graph (type shrunk);
48
49   entry k;
50   surfaces {
51     inner ( id 1; type sphere; origin ( 0.0 0.0 0.0); radius 61.011; )
52     outer ( id 2; type sphere; origin ( 0.0 0.0 0.0); radius 61.786; )
53   }
54
55   cells {
56     core ( type simpleCell; id 3; surfaces (-1); filltype mat; material solution; )
57     shell ( type simpleCell; id 4; surfaces (1 ); filltype mat; material shell; )
58   }
59 }
60
61 }
```

<https://github.com/CambridgeNuclear/vscode-scone>

Some low hanging fruits:

- VSCode integration: Syntax highlighting (including **invalid** syntax), folding etc.
- We will use VSCode Language Server Protocol for context-sensitive help

'Not so greats': Dependency Management

Manual setup of dependencies can be time-consuming. Also deviates from modern day standards:

- Make LAPACK & BLAS dependency optional
- Use FetchContent (or CPM) to get rest of dependencies automatically with CMake:
 - Downloads dependencies on configure step. Compiles on build.
 - Will increase first compilation time (no affect on recompilation)
 - Requires internet access

Work ongoing:

- Switching to pFUnit 4 (supports FetchContent build)
- 'Modernisation' of Cmake configuration

Current Focus

The Random Ray Method



Development for shielding problems

- Very low memory usage (no angular fluxes) [only **39 Mb** on 3D Dogleg!]
- Good angular coverage (random rays! MC element)
- Good parallel efficiency

Ask Paul Cosgrove for more details!

Variance Reduction

Integration of all Valeria's PhD improvements:

- Multi-group generation
- Uniform Fission Source
- Weight windows

Multiphysics

- Thermal Motion sampling and DBRC (Jamie Edwards presentation)
- Code-to-code coupling

Technical Debt Reduction

Going through all TODO's left over the years:

- CI restoration
- Better error and warning handling
- Efficiency improvements (input parsing, data loading)
- Refactoring

Thank you for your attention

<*)*))><

