24th Nuclear Simulation Symposium Paper 000
Ottawa, Ontario, Canada, Oct. 14-16, 2012

MODERNIZATION OF THE NESTLE-CANDU REACTOR SIMULATOR AND COUPLING TO SCALE-PROCESSED CROSS SECTIONS

S. Hart and G.I. Maldonado
The University of Tennessee, Knoxville, Tennessee, USA

Abstract

The original version of the NESTLE computer code for CANDU applications, herein referred as the NESTLE-CANDU or NESTLE-C program, was developed under sponsorship by the CNSC as a "stand-alone" program [1]. In fact, NESTLE-C emerged from the original version of NESTLE [2], applicable to light water reactors, which was written in FORTRAN 77 to solve the few-group neutron diffusion equation utilizing the Nodal Expansion Method (NEM). Accordingly, NESTLE-C can solve the eigenvalue (criticality); eigenvalue adjoint; external fixed-source or eigenvalue initiated transient problems for CANDU reactor fuel arrangements and geometries. This article reports a recent conversion of the NESTLE-C code to the Fortran 90 standard, in addition, we highlight other code updates carried out to modularize and modernize NESTLE-C in a manner consistent with the latest updates performed with the parent NESTLE code for light water reactor (LWR) applications [3]. Also reported herein, is a simulation of a CANDU reactor employing 37-element fuel bundles, which was carried out to highlight the SCALE to NESTLE-C coupling developed for two-group collapsed and bundle homogenized cross-section generation. The results presented are consistent with corresponding simulations that employed HELIOS generated cross-sections.

1. Introduction

The original NESTLE-C (or NESTLE-CANDU) computer code [1] was written using FORTRAN 77 and constituted a CANDU-oriented adaptation of the parent NESTLE [2] code developed to model PWRs. In more recent times, the parent NESTLE code was upgraded to Fortran 90 by a collaboration between Oak Ridge National Laboratory (ORNL) and the University of Tennessee [3], and has grown to include options to model BWRs [4], and to employ ORIGEN-based isotopic depletion [5], as a few of the latest enhancements. The latest work on NESTLE-C involves updating the code to Fortran 90, and streamlining ease of use for both future developers and users. Maintaining backwards compatibility with older NESTLE-C input files is also of paramount importance.

Using NESTLE-C, a benchmark CANDU core case was modelled using SCALE [6] to calculate the lattice cross-sections. These results could then be compared to those obtained using HELIOS and NESTLE-C. This task was divided into three steps:

1. The lattice calculations are done using the TRITON/NEWT [7] module of SCALE. This results in the collapsed two-group, homogenized, cross-sections.

2. Convert the cross-sections to a format recognized by NESTLE-C using T2N (TRITON-to-NESTLE).

3. Run the simulation in NESTLE-C.

These steps are discussed in detail below. The results shown are discussed for an equilibrium CANDU core. Some work on modelling a fresh core and basic attempts at modelling the refuelling process were also done.

2. Modernization of NESTLE-C

There were several steps carried out to bring NESTLE-C up to more recent Fortran coding standards. These steps are outlined in the sections below.

2.1 Changes to Code Structure

The original NESTLE-C code was written in a non-modular form, delivered with all source files in a single directory and there was no obvious differentiation between the roles of individual files. The first step for modularizing the code was to categorize the source files by areas of application and to group the files into directories representing those areas. The NESTLE work carried out at ORNL served as a template for this work [3].

After evaluating the role of each file, the directory structure shown in Figure 1 was created. This structure closely mimics that of the parent NESTLE code maintained at ORNL, which will allow for easier collaboration and potential integration in the future.

As the original NESTLE-C (and NESTLE) were both written in FORTRAN 77, there was no concept of modules that were introduced in Fortran 90/95. All data that was to be shared between routines had to be defined in a common block. In addition, since the original NESTLE made heavy use of implicit declaration (an ability of Fortran to determine if a number is an integer or a real depending on what letter the variable name starts with) it was tedious to debug since a simple typing error can create a new, undefined, variable. To this end, the vast majority of Fortran common blocks in NESTLE-C have been removed, and replaced with explicitly declared variables in Fortran modules. The new style is both cleaner and more explicit.

2.2 Changes to Code Source Files

FORTRAN 77 and Fortran 90/95 have slightly different syntax in their source code files. This largely arises due to the fact that Fortran 90/95 supports free-form coding whereas FORTRAN 77 has certain rules regarding the placement of code structures. When updating the source files, the old source code was updated to agree with the latest Fortran 90/95 standards.

This work mostly involved changing the style of comments and continuation markers. All source files were also updated to use the new modules that were created instead of including the common blocks.

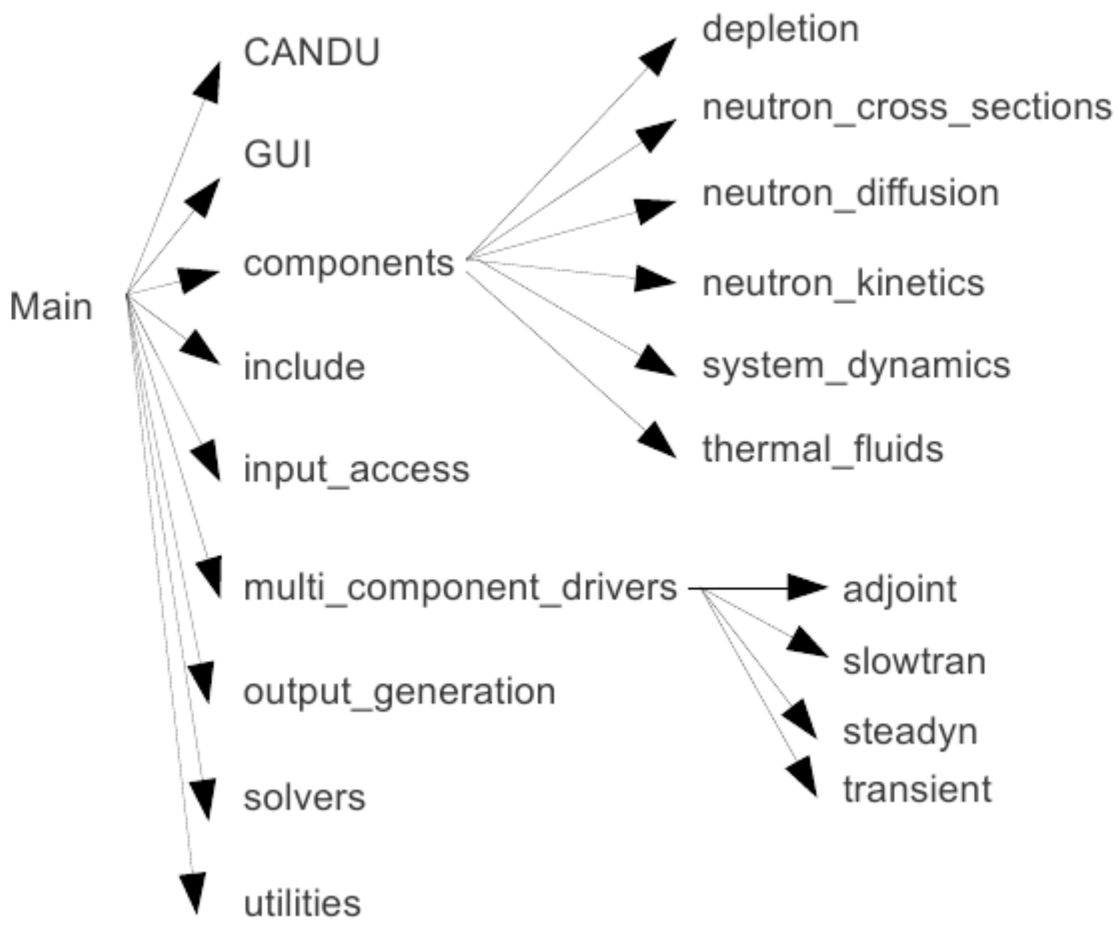


Figure 1: NESTLE-C Directory Structure

2.3 Creation of Build Scripts/Build System

2.3.1 GNU Make

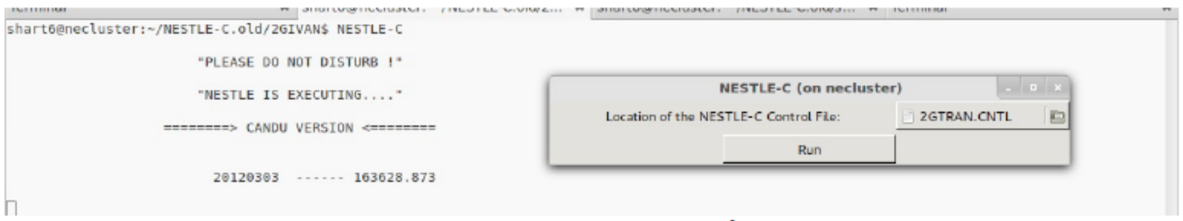
In order to ease compilation, "Make files" were created that are compatible with the GNU Make system. This vastly eases the creation of the NESTLE-C binary, and subsequent compilation speeds will benefit from conditional compilation where only source files that have changed need to be recompiled.

2.3.2 CMake

CMake is a cross-platform tool that creates build scripts for a variety of systems. By using CMake, one can create GNU Make files for Linux/Unix and Visual Studio project files for Windows. CMake is used to enable some of the more advanced features put into NESTLE-C (at this time, a Graphical User Interface).

2.4 Creation of Basic User Interface

If CMake was used to compile NESTLE-C, there is the option to use the GTK+2 libraries to create a graphical user interface to NESTLE-C. Currently, only basic functionality is implemented. For example, in Figure 2, one can choose their NESTLE control file using a file chooser dialogue.



```
shart6@necluster:~/NESTLE-C.old/2GIVAN$ NESTLE-C

                    "PLEASE DO NOT DISTURB !"

                    "NESTLE IS EXECUTING...."

               ========> CANDU VERSION <========


                  20120303  ------ 163628.873
```

Figure 2: NESTLE-C Input Chooser

There are two potential areas of expansion for the graphical user interface:

1. Input creation tools: A GUI could be used to create and/or modify input files. There are currently tools within SCALE that could be used as inspiration for this task.

2. Output visualisation tools: Plots and other data could be obtained and visualised using the GUI toolkit.

3. Simulation of CANDU Core using NESTLE-C and SCALE

As mentioned above, modelling a CANDU reactor using NESTLE-C and SCALE requires three unique steps: Creating the lattice in SCALE, converting the cross-sections to NESTLE form, and running the case in NESTLE. These steps will be discussed in the sections below, along with the challenges faced and the tools used to overcome them.

3.1 SCALE Simulation of CANDU Lattice

3.1.1 Dancoff Factors

The creation of a CANDU lattice in SCALE presents several unique challenges. Specifically, the irregular lattice of the fuel pins within the CANDU fuel bundle requires a special treatment when using the CENTRM cross-section processing modules. When determining cross-sections for each fuel region, NEWT (a 2-D discrete ordinates transport code in SCALE) uses the cross-section processing module CENTRM. In order to correctly create the problem dependent working cross-section library, CENTRM requires that the pitch structure of the fuel pins be defined. Examples of possible layouts include triangular and square geometries, however since the fuel pins in a CANDU bundle are not in any of the pre-specified shapes, an alternate treatment is required which pre-calculates the Dancoff factors and provides them directly to the NEWT code as inputs.

The CANDU bundle model, as generated by NEWT, is shown in Figure 3. By using the

MCDANCOFF module of SCALE [8], the Dancoff factors (and the associated equivalent or modified pitches) were calculated for each fuel pin as shown in Table I. Pin 1 is the center pin, and the numbering continues outward radially for each ring and, within a ring, in a clock-wise direction (as shown in the figure to the right of Table I). Note that the manner in which the adjustments are handled internally within SCALE, a "modified pitch" is evaluated for each pin to effectively apply the impact of each corresponding Dancoff factor.

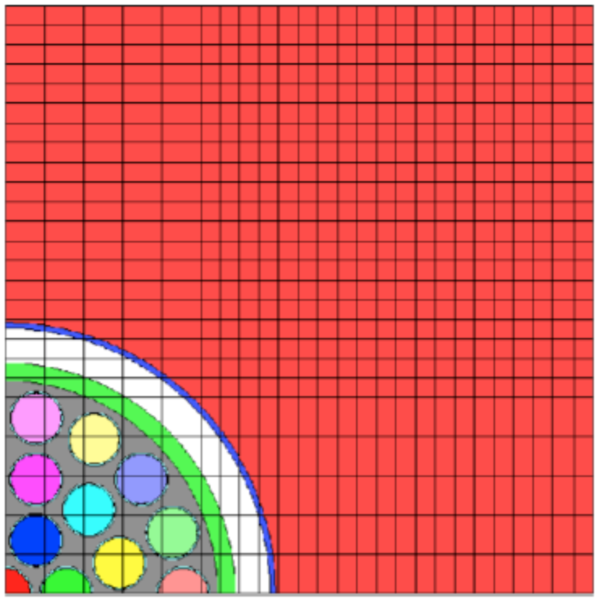


Figure 3: NEWT model of CANDU Bundle (1/4 symmetry)

Pin Number	Dancoff Factor	Modified Pitch (cm)
1	0.781569	1.5610
2	0.779812	1.5659
3	0.779786	1.5670
4	0.745004	1.6421
5	0.745584	1.6357
6	0.745134	1.6346
7	0.455967	2.3854
8	0.457870	2.3696
9	0.458498	2.3659
10	0.456444	2.3735
11	0.459030	2.3719

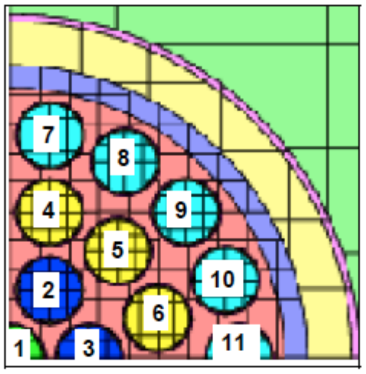


Table 1: Dancoff Factors and Modified Pitches for CANDU Bundle (Assignments on RHS)

3.1.2 TRITON/NEWT and TRITON/KENO Models

In order to obtain cross-sections at a variety of bundle burnups, the TRITON module of SCALE was used. TRITON calls ORIGEN in between transport calculations in order to perform isotopic depletion for the compositions in the model. TRITON can be used with either the NEWT transport code or with a Monte Carlo based code named KENO, both part of SCALE. Results from both codes are discussed here for completion, but it should be noted that as of presently, only NEWT can generate the collapsed and homogenized lattice cross-sections required by NESTLE. In order to compare lattice results obtained with TRITON/NEWT and TRITON/KENO with HELIOS, the TRITON model is designed to include as many depletion steps as possible from the HELIOS model. Not every depletion step present in the HELIOS model was replicated in the SCALE model, but the majority of steps were.

The results for the depletion run are shown in Table II and a graph of this data is shown in Figure 4. The large 3% difference at the beginning of the burnup history arises from the fact that HELIOS assumes xenon and samarium to be at equilibrium at 0.0 MWd/MT, whereas TRITON mimics the actual problem, which assumes no xenon and samarium at the beginning of cycle, so it takes a couple of depletion points (40 to 50 hours, or a couple of days) for the xenon and samarium to build up to equilibrium in the TRITON calculation.

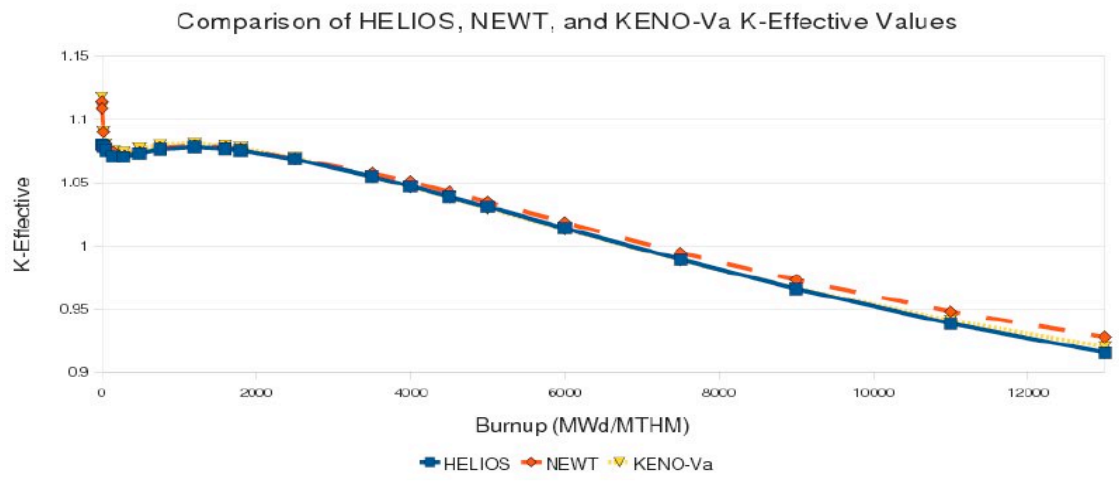


Figure 2: Comparison of HELIOS, NEWT, and KENO-Va K-Effective Values

The KENO-Va calculation was included to help verify the Dancoff factors were being applied correctly (since continuous energy KENO does not require input of Dancoff factors). The KENO run consisted of 5000 active generations of 10000 particles each for a total of 50 million histories per depletion step. The results of the KENO run are contrasted against those of the NEWT calculation, and also against the HELIOS reference calculation. At a quick glance, the results from KENO obtained are deemed to be consistent with the NEWT results obtained with Dancoff factors implemented, and appear to be reasonable relative to the HELIOS reference used. The NEWT and KENO differences increase versus HELIOS at the end of the depletion steps provided, but these differences occur at a high burnup that is not characteristic of standard CANDU fuel. Furthermore, this comparison versus HELIOS is not intended to be a validation check, because a different (older) cross-section library was employed relative to the SCALE tools. Thus, this comparison is herein provided to note that the results obtained by SCALE tools fall within a reasonable range relative to other tools employed to model a CANDU bundle.

Time (d)	Burnup (MWd/MT)	HELIOS (K-EFF)	NEWT (K-EFF)	NEWT-HELIOS (% Diff)	KENO (K-EFF)	KENO-HELIOS (% Diff)
0.0	0	1.08014	1.11357	3.1	1.11726	3.4
0.1	4	1.07978	1.10866	2.7	1.10909	2.7
0.6	19	1.07831	1.08991	1.1	1.08982	1.1
1.6	52	1.07524	1.08029	0.5	1.08006	0.5
4.7	152	1.07030	1.07366	0.3	1.07412	0.4
8.8	287	1.07002	1.07086	0.1	1.07344	0.3
15.0	488	1.07291	1.07374	0.1	1.07667	0.4
23.3	757	1.07616	1.07711	0.1	1.07938	0.3
36.9	1200	1.07778	1.07862	0.1	1.08077	0.3
49.2	1600	1.07654	1.07732	0.1	1.07838	0.2
55.3	1800	1.07524	1.07580	0.1	1.07702	0.2
76.9	2500	1.06836	1.06866	0.0	1.06940	0.1
107.6	3500	1.05459	1.05713	0.2	1.05479	0.0
123.0	4000	1.04685	1.05070	0.4	1.04628	-0.1
138.3	4500	1.03875	1.04223	0.3	1.03850	0.0
153.7	5000	1.03046	1.03449	0.4	1.02927	-0.1
184.5	6000	1.01366	1.01809	0.4	1.01292	-0.1
230.6	7500	0.98890	0.99414	0.5	0.98856	0.0
276.7	9000	0.96568	0.97279	0.7	0.96613	0.0
338.2	11000	0.93831	0.94776	1.0	0.94067	0.3
399.6	13000	0.91521	0.92759	1.4	0.91995	0.5

Table 11: K-effective Results from KENO and NEWT contrasted to HELIOS

3.2 T2N-C Cross Section Processing

The coupling code which converts SCALE cross-section data into a NESTLE-friendly format is called T2N, which is a short name assigned to the "TRITON-to-NESTLE" program. T2N was slightly modified from that used by the parent NESTLE code to convert the SCALE/NEWT generated cross-sections into the slightly different format required by NESTLE-C. NESTLE-C uses a polynomial fit based on perturbations to coolant density, boron density, temperatures, etc., to get the cross-sections for each node. This saves space, but requires cross-sections to be fit to this polynomial model. The polynomial fit for NESTLE-C is shown in Equation 1 below. The order of the cross-secions in NESTLE-C is slightly different from that of NESTLE, therefore T2N had to be adjusted to fit this arrangement, and is called T2N-C to differentiate it from its non-CANDU counterpart.

$$\begin{aligned}\hat{\Sigma}_{xg} &= a_{1_{xxg}} + \sum_{n=1}^{2} a_{(n+1)_{xg}} (\Delta\rho_c)^n + \sum_{n=1}^{2} a_{(n+3)_{xg}} (\Delta T_c)^n + \sum_{n=1}^{2} a_{(n+5)_{xg}} (\Delta\sqrt{T_{F_{eff}}})^n \\ &\quad + \sum_{n=1}^{2} a_{(n+7)_{xg}} (\Delta P_c)^n + \sum_{n=1}^{2} a_{(n+9)_{xg}} (\Delta P_m)^n + \sum_{n=1}^{2} a_{(n+11)_{xg}} (\Delta N_{sp})^n\end{aligned} \tag{1}$$

where;

$$\hat{\Sigma}_{xg} = \text{macroscopic cross-section for reaction type } x \text{ and energy group } g \text{ without transient fission products corrected to local conditions}$$

$$a_{j_{xg}} = \text{expansion coefficients}$$

$$\Delta\rho_c = \rho_c - \rho_c^{(0)} = \text{change in coolant density } (g/cm^3) \text{ from reference condition}$$

$$\Delta T_c = T_c - T_c^{(0)} = \text{change in coolant temperature } (^oF) \text{ from reference condition}$$

$$\Delta\sqrt{T_{F_{eff}}} = \sqrt{T_{F_{eff}}} - \sqrt{T_{F_{eff}}^{(0)}} = \text{change in square root of effective fuel temperature } (^oF) \text{ from reference condition}$$

$$\Delta P_c = P_c - P_c^{(0)} = \text{change in coolant purity from reference condition}$$

$$\Delta P_m = P_m - P_m^{(0)} = \text{change in moderator purity from reference condition}$$

$$\Delta N_{sp} = N_{sp} - N_{sp}^{(0)} = \text{change in soluble poison number density } (cm^{-3} * 10^{-24}) \text{ from reference condition.}$$

3.3 NESTLE-C Models

NESTLE-C was provided to this project with a sample CANDU core equilibrium case already packaged that employed the HELIOS cross-sections used in the comparisons above. By using T2N-C, the cross sections from SCALE/TRITON can be used with this equilibrium case and the results compared.

The equilibrium core case included bundles in various stages of burnup and control devices positioned into a critical configuration. This case, when run with the specified operational conditions and with HELIOS cross sections, yielded a k-effective of 1.0015987. The cross-section file was then modified to incorporate the cross sections produced by T2N-C from the SCALE/TRITON calculation. It should be noted that only the fuel cross sections were generated by TRITON, while the cross sections for the reflector surrounding the reactor core were not changed from the original equilibrium core.

The resulting k-effective of the equilibrium case with TRITON cross sections was 1.0087206. The higher k-effective value for the SCALE-based cross section data reflects, in part, the higher k-infinity values calculated with TRITON, relative to HELIOS-based data. Figure 5 shows the relative powers across row L for the TRITON-based NESTLE-C model, which for all practical purposes were identical between the TRITON or HELIOS generated cases.

3.3.1 Coolant Void Reactivity (CVR)

One of the frequently modelled scenarios in a CANDU core is that of a large loss of cooling accident (LOCA). This can be modelled as either a full break in both directions, or as a "checker board" where only one direction of the flow is voided. This is illustrated graphically in Figure 6.

Note that this figure is provided for illustrative purposes only, and the color of the moderator is changed in the figure even though that is not what is being voided because the coolant only runs through the pin cells of the bundle. In order to showcase which channels are actually void in each case, it was deemed easier to colour the entire block.

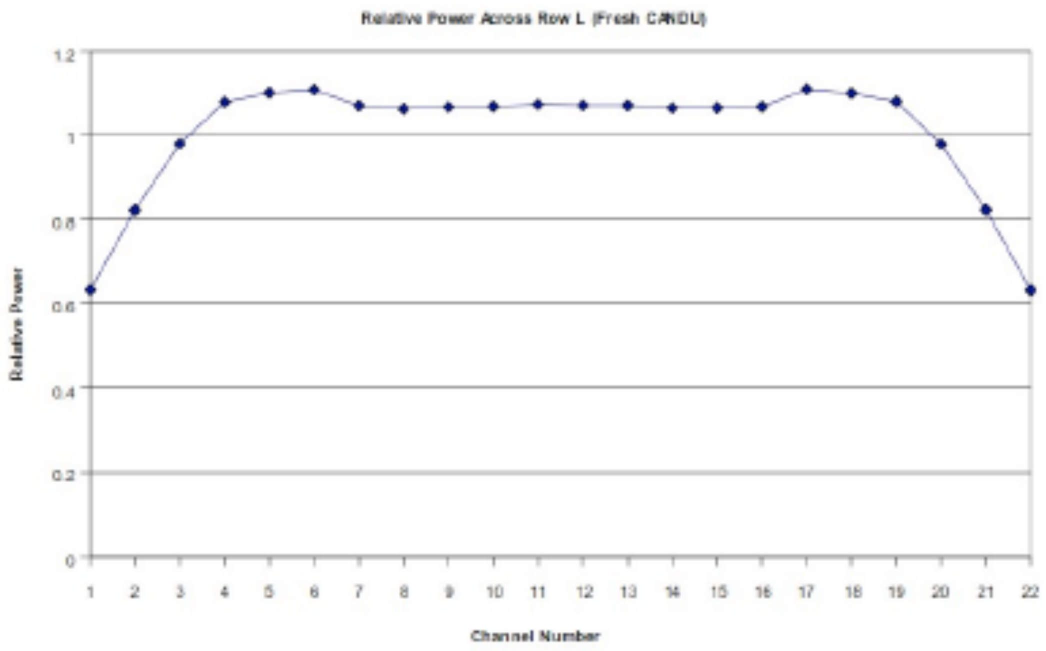


Figure 3: Relative Powers in Row L for Equilibrium CANDU core in NESTLE-C

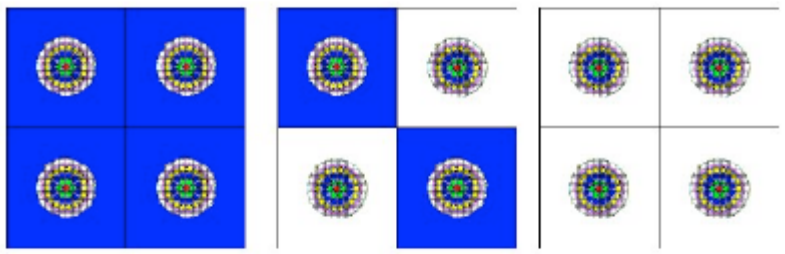


Figure 4: CVR Cases - (L) Fully Cooled, (C) Checker Board, (R) Fully Voided

The CVR calculations can be modelled in NESTLE-C by reducing the coolant density in the desired channels to 0.001 g/cc and calculating the rise in k-effective. The CVRs for the equilibrium case are shown in Table III, which correspond to values of CVR of approximately 17.2 mk (fully voided) and 7.7 mk (checkerboard voided). It should be noted that these results are for an equilibrium core example, thus probably representative of an average bundle burnup. Also, these results included the impact of adjuster rods. The CVR results herein obtained appear to be of similar order as other published results, such as those obtained by F. Talebi, et al. [9] which reports values of 16 mk for the CVR validations for the CANDU-6 using the DRAGON code. Likewise, other results using MCNP published by Rahnema, et al. [10] calculated CVR values of 16.34 mk (fully voided) and 9.33 mk (checkerboard voided). A previous study using NESTLE by H. Sarsour, et al. [11] reports fairly consistent MCNP results of 16.05 mk and 9.17 mk for the fully voided and checkerboard voided, respectively, in addition to corresponding NESTLE-based results of 16.82 mk and 8.41 mk for full-core nodal method calculations that included assembly discontinuity factors. At a glance, the SCALE-based NESTLE-C model in this study appears to slightly over-predict the fully voided CVR cases and to slightly under-predict the checkerboard cases, relative to the other reported works. However, these results fall within a similar range and order based on data from prior studies by other researchers.

Case	K-Effective	CVR (milli-k)
Fully Cooled	1.0087204	N/A
Checkerboard Voided	1.0166116	7.70
Fully Voided	1.0264843	17.16

Table III: CVR Values for Cooled, Checkerboard Void, and Void Case

4. Conclusions and Future Work

The NESTLE-CANDU code has been updated to be consistent with the latest Fortran 90 standards. In order to test of the code, cross-sections were obtained using the Scale sequence TRITON, formatted into the NESTLE-C form, and the results compared to a equilibrium core modelled by HELIOS.

Future work involves further updates and modifications to the NESTLE-C code. Modifications will concentrate on modularity, ease of use, and further being able to integrate NESTLE with other codes. Care has been taken to ensure that any changes to NESTLE-C were comparable and compatible to changes made to the parent NESTLE code maintained by ORNL and UT.

5. Acknowledgements

The authors acknowledge the sponsorship of this project provided by the Canadian Nuclear Safety Commission (CNSC). In particular, we are thankful for the support, technical advice, and software tools provided by Dr. Dumitru Serghiuta and by our collaborators at NC State University, namely, Dr. Paul Turinsky and Dr. Hisham Sarsour.

6. References

[1] Sarsour H. N. and Turinsky P. J., "Reactor Physics Aspects of LOCA Analysis: Phase 2 [CNSC Project 87055-0273], April (2003).

[2] NESTLE: Code System to Solve the Few Group Neutron Diffusion Equation Utilizing the Nodal Expansion Method (NEM) for Eigenvalue, Adjoint, and Fixed-Source Steady-State and Transient Problems, Version 5.2.1, June 2003. RSICC CCC-641.

[3] G.I. Maldonado, J. Galloway, H. Hernandez, K.T. Clarno, E.L. Popov, M.A. Jessee, “Integration of the NESTLE Core Simulator with SCALE,” Trans. Am. Nucl. Soc., 100, 619-620, 2009.

[4] J. Galloway, H. Hernandez, G.I. Maldonado, M. Jessee, E. Popov, K. Clarno, “BWR Modeling Capability and SCALE/TRITON Lattice-to-Core Integration of the NESTLE Nodal Simulation,” CD-ROM, PHYSOR 2010, Pittsburgh, PA, May 9-14, 2010.

[5] J.D. Galloway, G.I. Maldonado, I. Gauld, M.A. Jessee, K.T. Clarno, “Generalized Isotopic Tracking Capabilities within the 3-D BWR Nodal Simulator NESTLE,” Trans. Am. Nucl. Soc., 103, 745-750, 2010.

[6] Scale: A Comprehensive Modeling and Simulation Suite for Nuclear Safety Analysis and Design, ORNL/TM-2005/39, Version 6.1, June 2011.

[7] M. D. DeHart, "High-Fidelity Lattice Physics Capabilities of the SCALE Code System Using TRITON," *Trans. Am. Nucl. Soc*., **97**, 598-600 (2007).

[8] L.M. Petrie, B.T. Readden, “MCDancoff Data Guide,” ORNL/TM-2005/39, Version 6, Vol. III, Sect. M24, January 2009.

[9] F. Talebi, G. Marleau, J. Koclas, “A Model for Coolant Void Reactivity Evaluation in Assemblies of CANDU Cells,” Annals of Nuclear Energy, 33, 975-983 (2006).

[10] F. Rahnema, S. Mosher, M. Pitts, P. Akhtar and D. Serghiuta, "MCNP Simulation of Void Reactivity in a Simplified CANDU Core Sub-Region", Proceedings of the International Conference Monte-Carlo-2000, Lisbon (2000).

[11] H. N. Sarsour, et al., “HELIOS/DRAGON/NESTLE Codes’ Simulation of Void Reactivity in a CANDU Core,” Proc. CNS (2002).