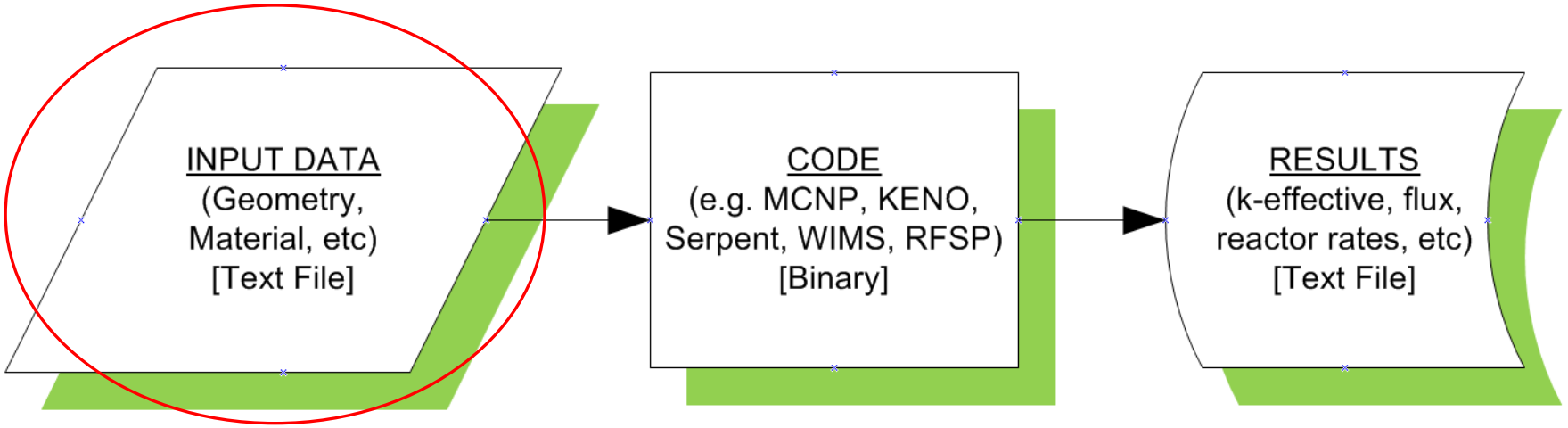
Modeling the ZED-2 Reactor w/ Serpent/MCNP/KENO

The 4th Annual Serpent Users' Group Meeting,
17–19 September 2014
Jesus College, Cambridge, UK

J. C. Chow

AECL EACL

We are stuck!



Survey

- How do you author input data files for simulations?

1. Use a text editor.
2. Use scripting languages such as Python, Perl, etc.

Motivation

- Assignment: Validation of MCNP against ZED-2 data
- Requirement: Generate KENO models equivalent to *existing* ZED-2 MCNP models for analyses with TSUNAMI
- Task: use a text editor (and a pair of aging eyeballs) to author/edit ZED-2 models (thousands of lines / model)

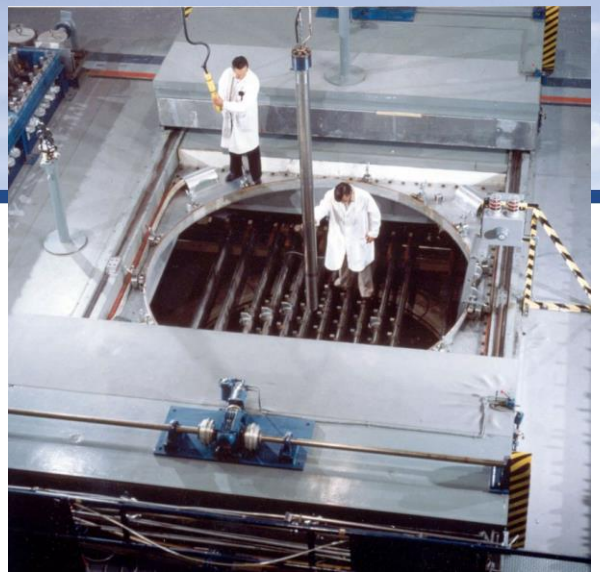
- How can I guarantee consistency between MCNP and KENO models?
- Solve the problem ***programatically…*** develop applications that share the same data source.

Conventional Approaches for Preparing Input Data

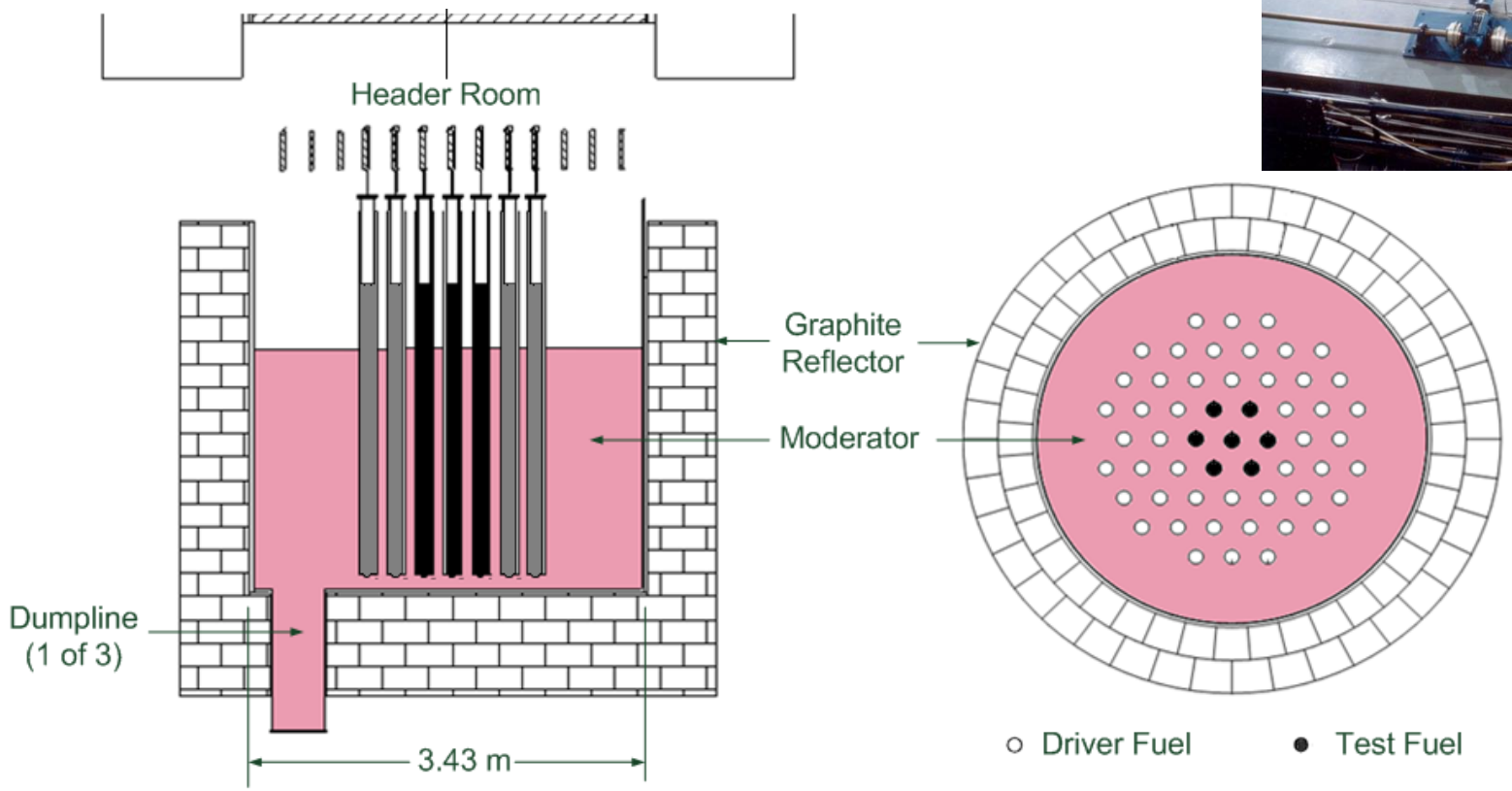
1. Direct typing using generic text editors: Notepad, Wordpad, Notepad++, vi, GEdit, Emacs, XEdit, …
2. Copy/paste pieces of data from existing models (text files) and spreadsheets.
3. Modify existing models to obtain different configurations: (change in physical properties and/or geometry).
4. Use extensive Excel® macros and/or other languages (e.g., Perl, Python, Fortran) to generate models.

- Using text editors (1-3) is always error-prone.
- Using Excel® macros and/or other languages, etc, are preferable, but:
  - often tailored for personalized use
  - implemented in piecemeal fashions
  - little/no documentations
  - difficult to share with other colleagues.
- Substantial effort is required for verification.

The ZED-2 Reactor



Picture taken in the'60s

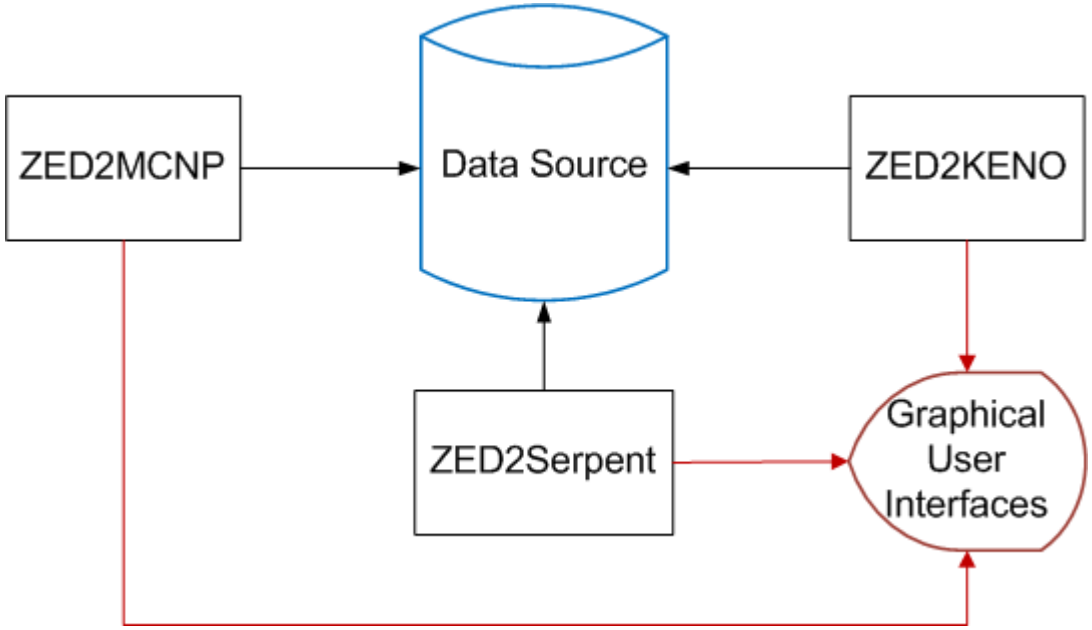


A Typical ZED-2 MCNP Input File

- ~500 surfaces
- ~1,000 cells
- ~60 materials
- ~70 universes
- ~4,000 lines

- In order to modify the model for a specific configuration, we need to edit a few 10s to 100s lines of data.
- Many (personalized) tools have been developed, mostly in EXCEL, e.g., to define the heavy water composition or to convert material cards from ENDF-VI to VII.
- Ctrl+C (from EXCEL) and Ctrl+V (to Notepad/Wordpad).

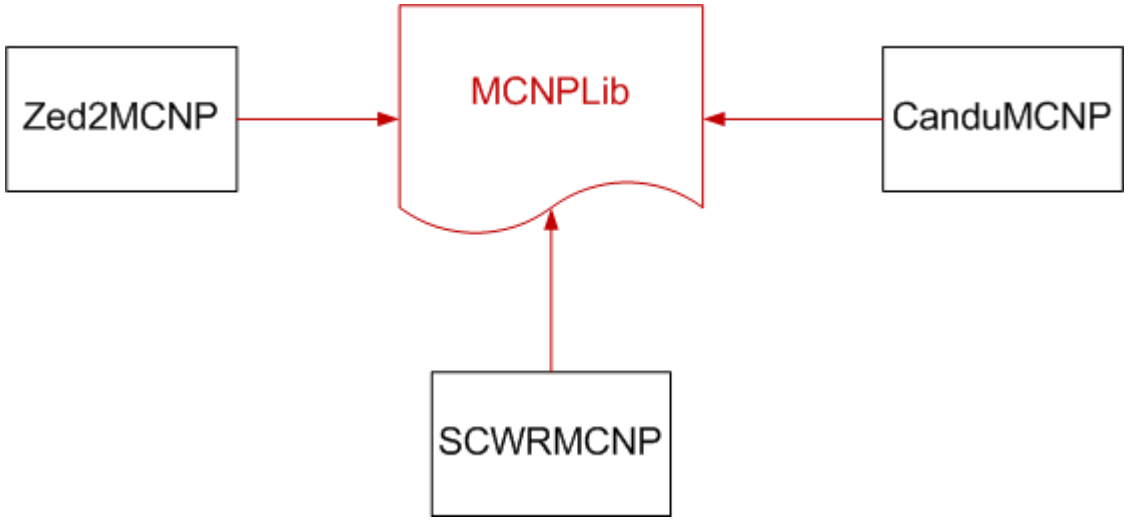
ZED-2 Model Generators Development (1)



1. Zed2MCNP (2009) J.C. Chow
2. Zed2KENO (2009) J.C. Chow
3. Zed2Serpent (2013) G. Aversano (UOIT)

They all share the same Data Source and GUI's.

ZED-2 Model Generators Development (2)



- Three applications for generating MCNP models for three physical systems: ZED-2, CANDU, and SCWR
- They all share the same MCNP class library (MCNPLib)
- Do we need a similar class library for Serpent?

About the ZED2- MCNP/KENO/Serpent Applications

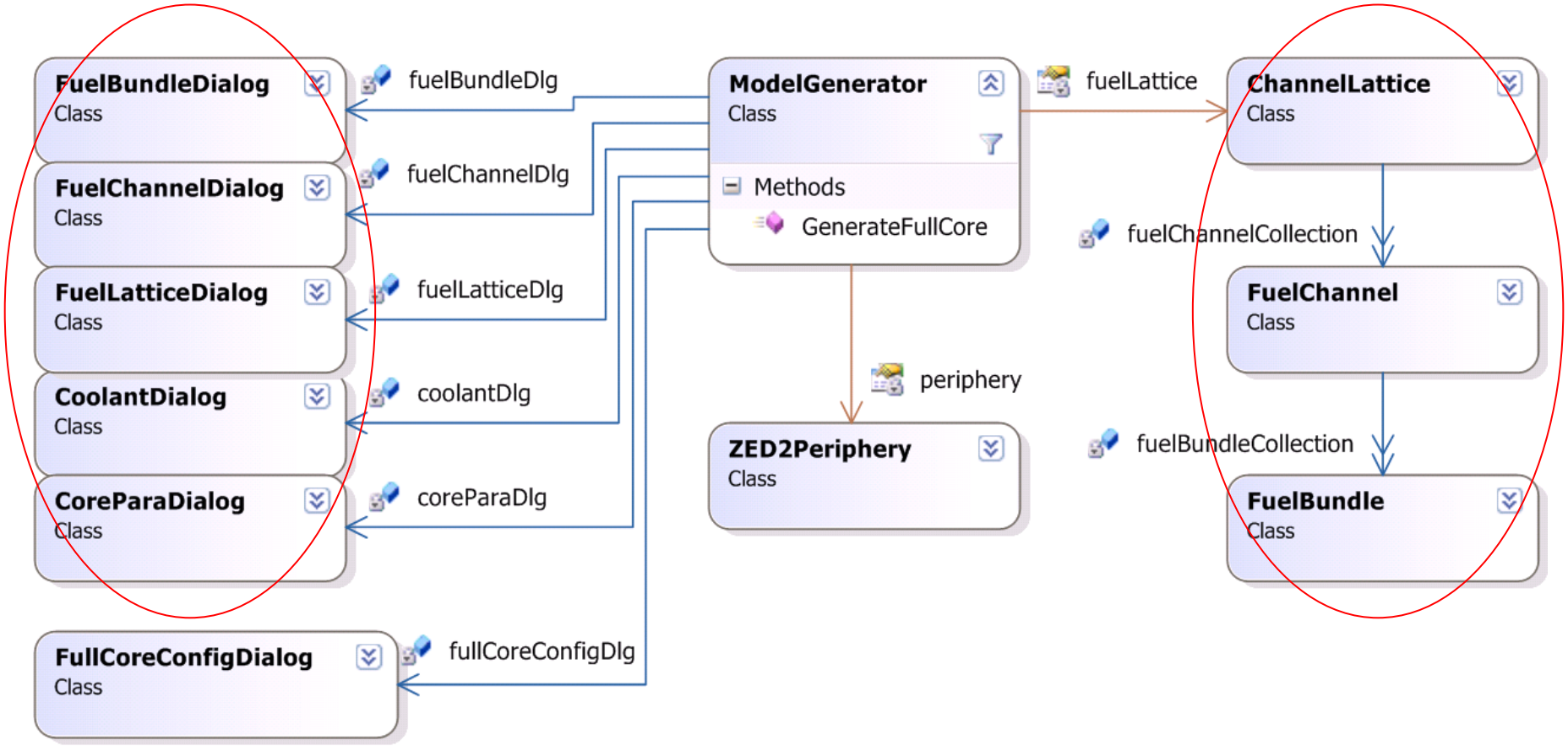
- Written in C# 3.0 (syntactically very similar to C++)
- Integrated Development Environment (IDE) Visual Studio 2008 IDE (Microsoft).
- Takes advantage of the rich Windows GUI features provided by .Net (dot-net) Framework Library 3.5 (Microsoft).
- Truly object-oriented (OOP) implementation.
- Requires a static set of input files (prerequisites):
  - Shifting the focus of managing a single (huge) data file to a set of files, each of which serves a specific purpose.
  - Analogous to the OOP; *a problem is decomposed into a number of objects (methods and data are built around the objects to solve the problem).*
- Once the set of files is ready, a full-core model can be generated with a few clicks of the mouse.

Model Prerequisites

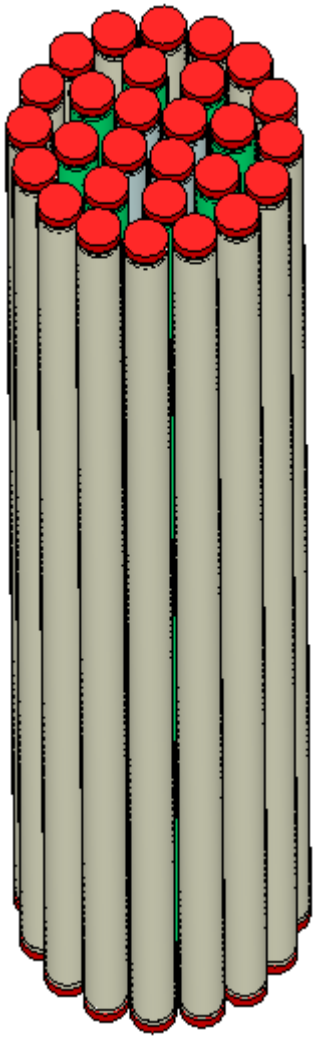
1. Fuel Bundles: 28-el NU, 37-el MOX, CANFLEX-LEU...
2. Fuel Channels: fuel bundles contained in pressure/calandria tube
3. Coolants: D_2O, H_2O, Air, CO_2…
4. Fuel Lattice: fuel channels contained in core configuration (square or hexagonal)
5. Core Parameters: critical height, moderator purity, lattice pitch, etc

- Each prerequisite is associated with a GUI (Graphical User Interface)
- Data entered for each prerequisite are saved in an XML file.
- All the (interdependent) prerequisites for a particular experimental run are assembled into a single configuration.

UML Class Diagram of a Model Generator

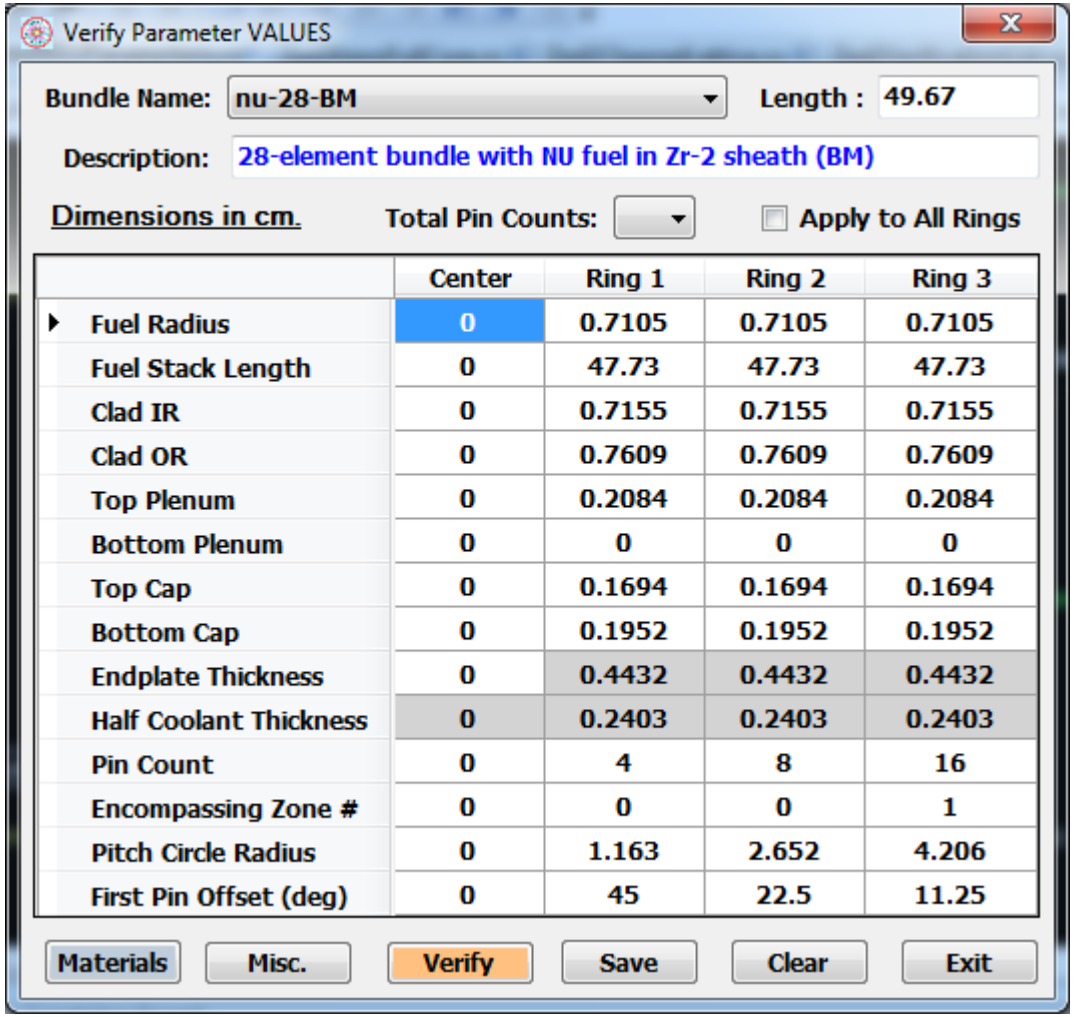


Fuel Bundle Geometry



Model

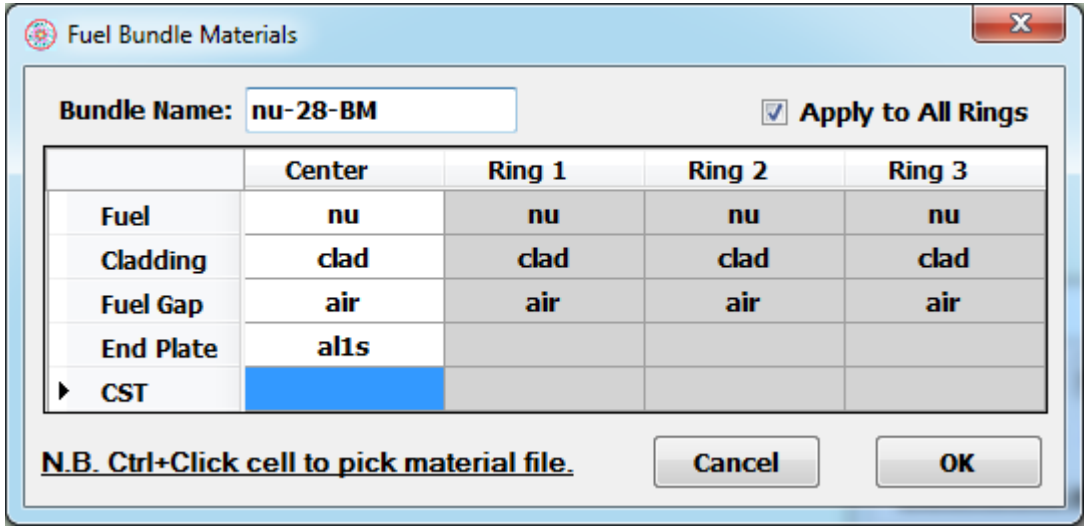
Fuel Bundle Configuration



	Center	Ring 1	Ring 2	Ring 3
Fuel Radius	0	0.7105	0.7105	0.7105
Fuel Stack Length	0	47.73	47.73	47.73
Clad IR	0	0.7155	0.7155	0.7155
Clad OR	0	0.7609	0.7609	0.7609
Top Plenum	0	0.2084	0.2084	0.2084
Bottom Plenum	0	0	0	0
Top Cap	0	0.1694	0.1694	0.1694
Bottom Cap	0	0.1952	0.1952	0.1952
Endplate Thickness	0	0.4432	0.4432	0.4432
Half Coolant Thickness	0	0.2403	0.2403	0.2403
Pin Count	0	4	8	16
Encompassing Zone #	0	0	0	1
Pitch Circle Radius	0	1.163	2.652	4.206
First Pin Offset (deg)	0	45	22.5	11.25

AECL EACL

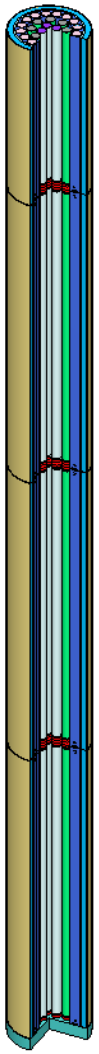
Fuel Bundle Material

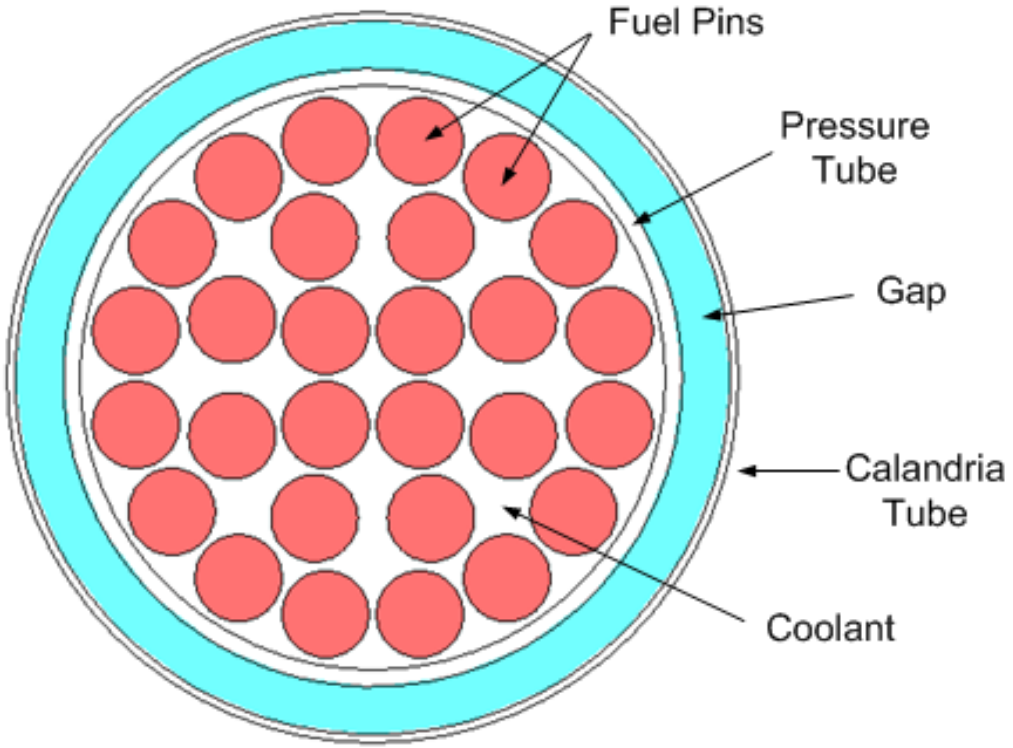


Bundle Data Saved as XML

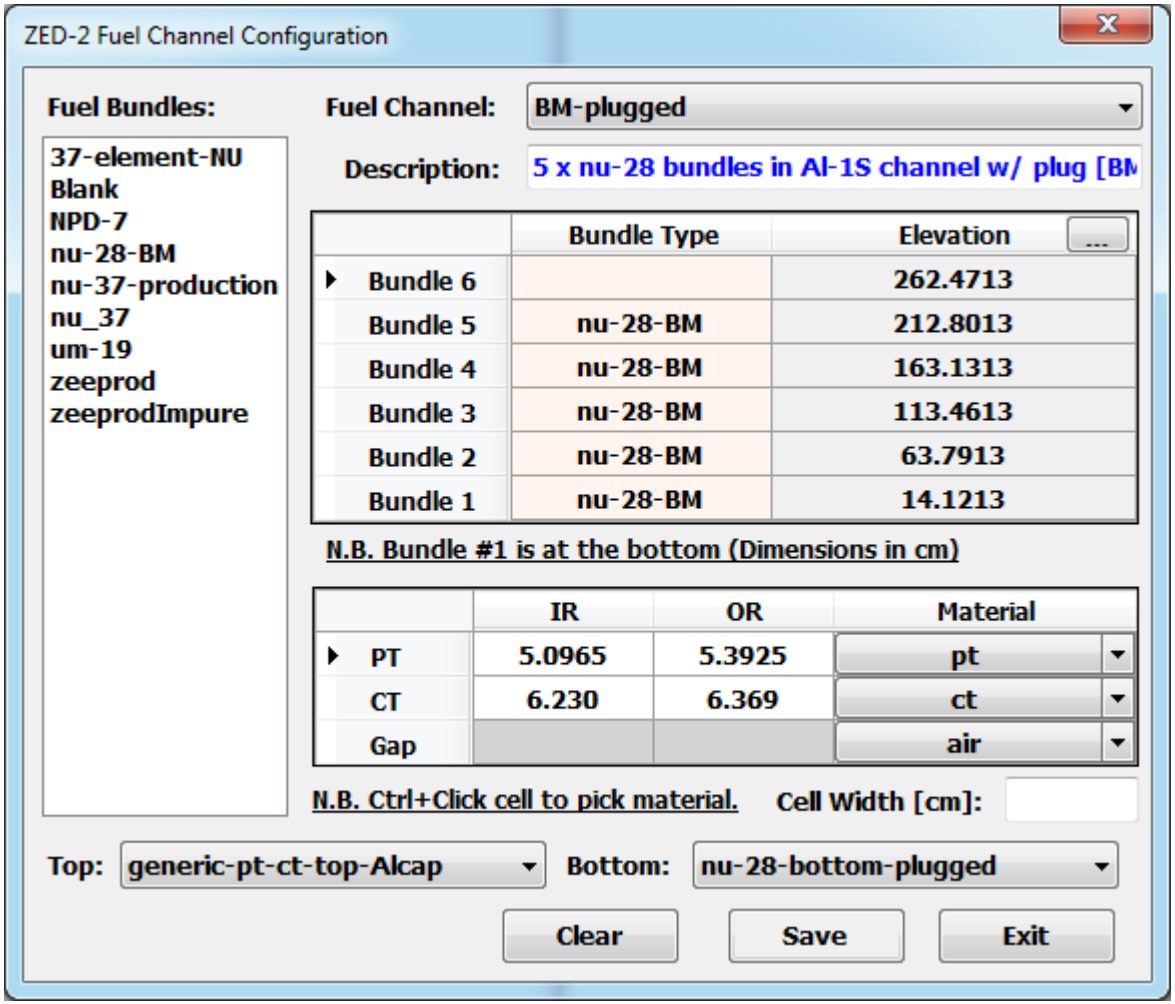
```
<?xml version="1.0" encoding="UTF-8"?>
<BundleParameter Description="28-element bundle with NU fuel in Zr-2 sheath (BM)">
  <BundleName>nu-28-BM</BundleName>
      <!--Logical length that defines a bundle-->
  <bundleLength>49.67</bundleLength>
  <fuel>
    <fuelRadius>0, 0.7105, 0.7105, 0.7105</fuelRadius>
    <fuelStackLength>0, 47.73, 47.73, 47.73</fuelStackLength>
    <material>nu, nu, nu, nu</material>
  </fuel>
  <clad>
    <cladIR>0, 0.7155, 0.7155, 0.7155</cladIR>
    <cladOR>0, 0.7609, 0.7609, 0.7609</cladOR>
      <!--endcap thicknesses-->
    <topCap>0, 0.1694, 0.1694, 0.1694</topCap>
    <bottomCap>0, 0.1952, 0.1952, 0.1952</bottomCap>
    <material>clad, clad, clad, clad</material>
  </clad>
  <plenum>
      <!--void gap between fuel and endcap-->
    <topPlenum>0, 0.2084, 0.2084, 0.2084</topPlenum>
    <bottomPlenum>0, 0, 0, 0</bottomPlenum>
    <material>air, air, air, air</material>
  </plenum>
  <endplate>
```

Fuel Channel





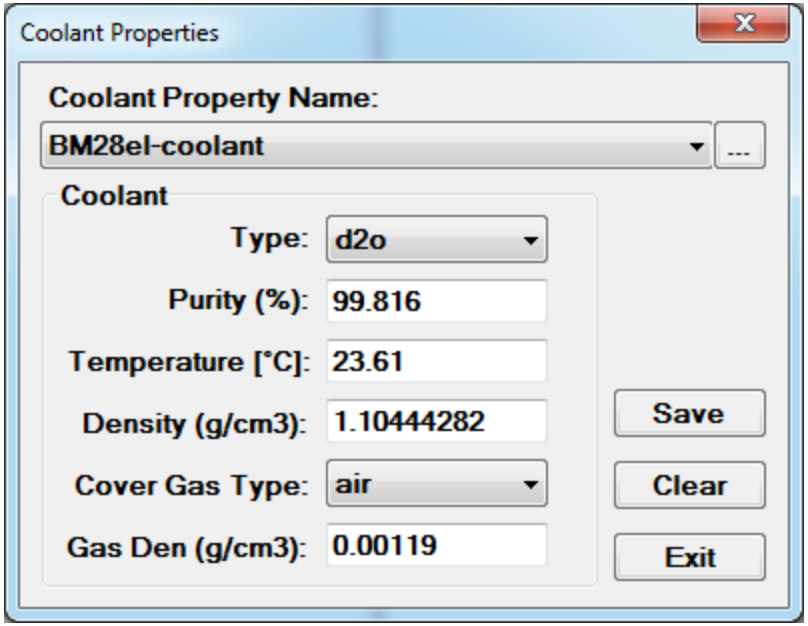
Fuel Channel Configuration



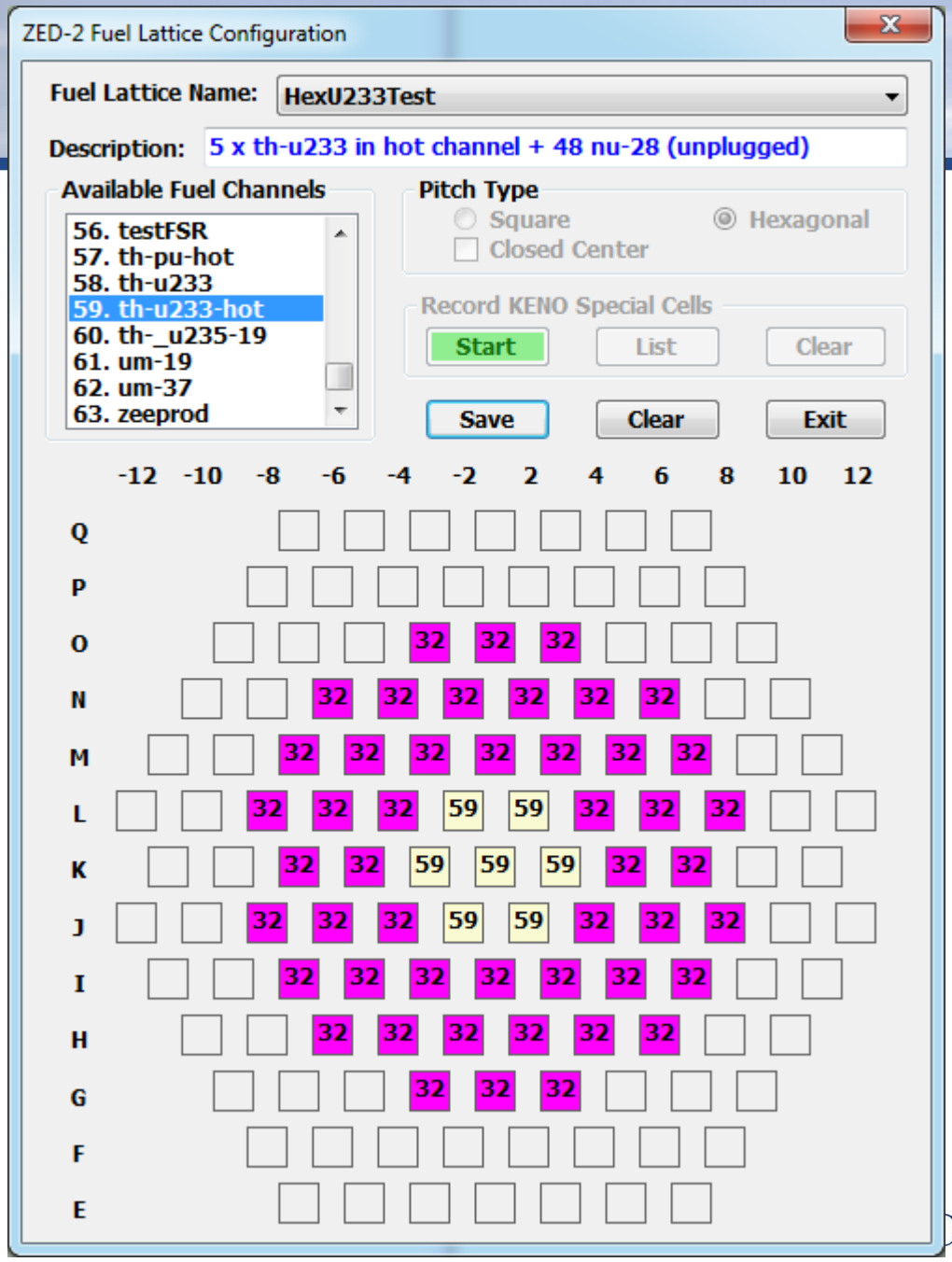
AECL EACL

UNRESTRICTED

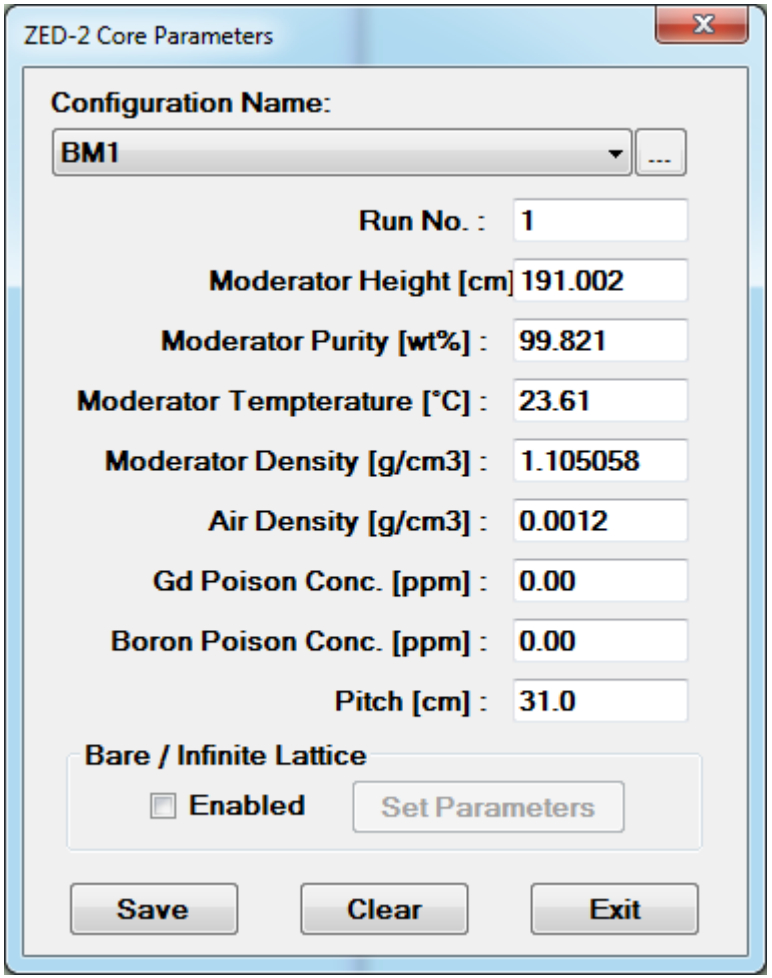
Coolant Property



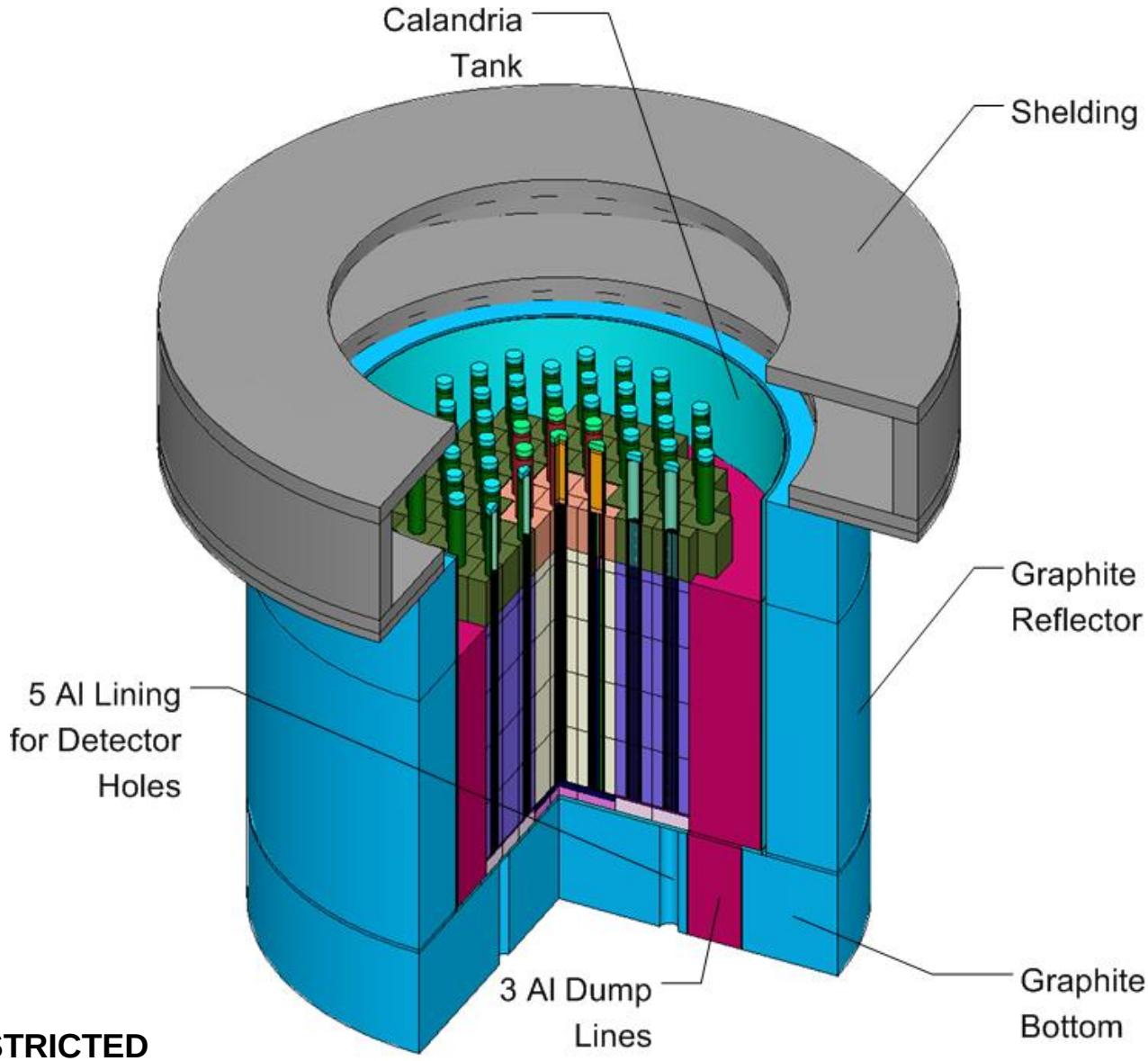
Core Layout



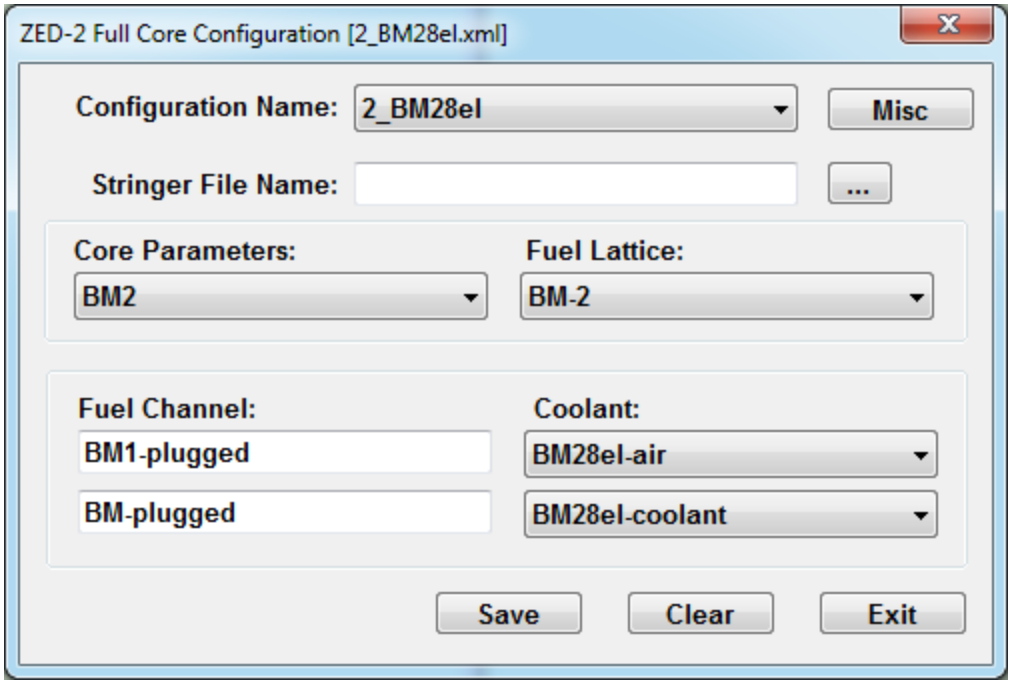
Core Parameters



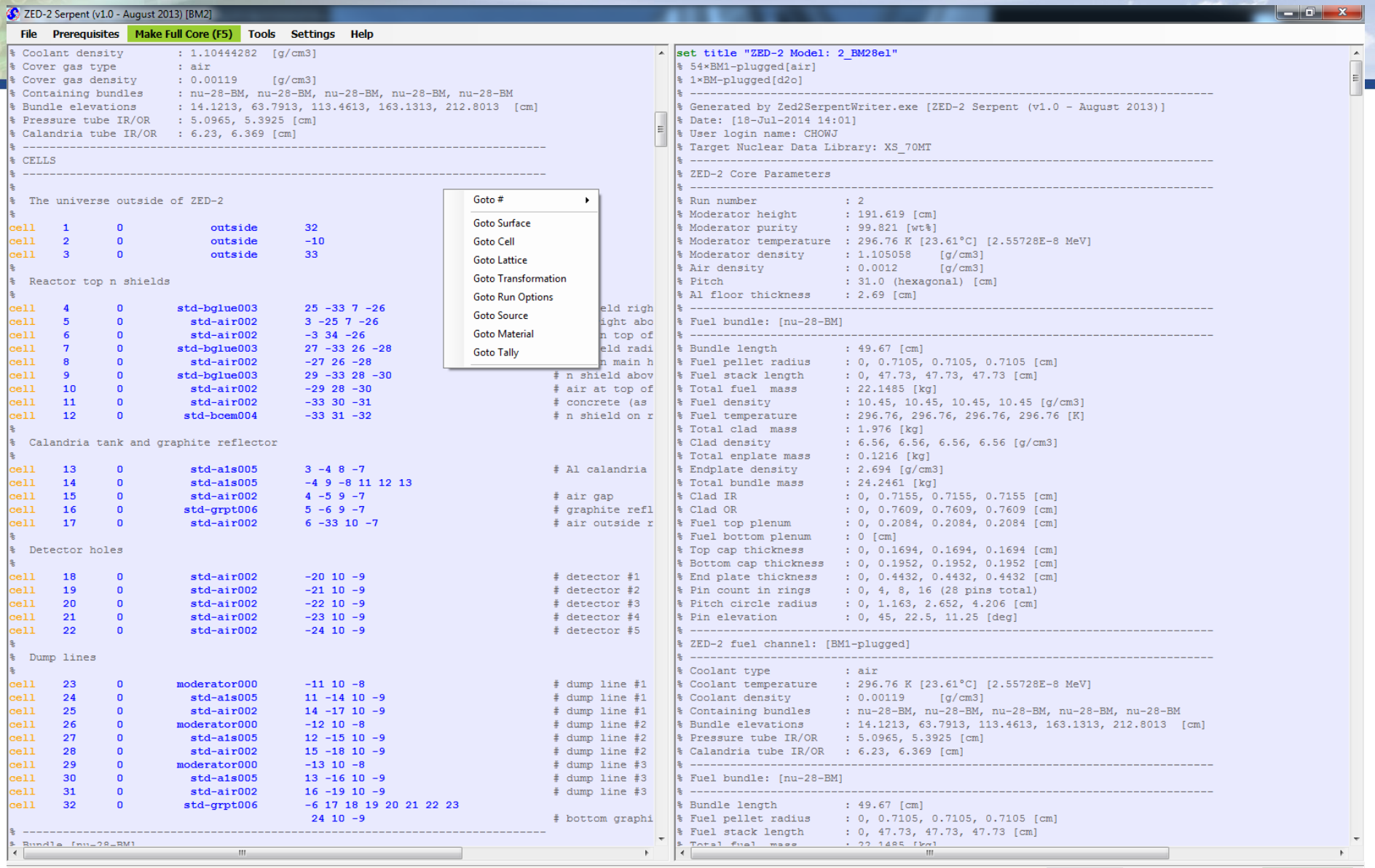
ZED-2 Core Periphery – Anything other than the Core



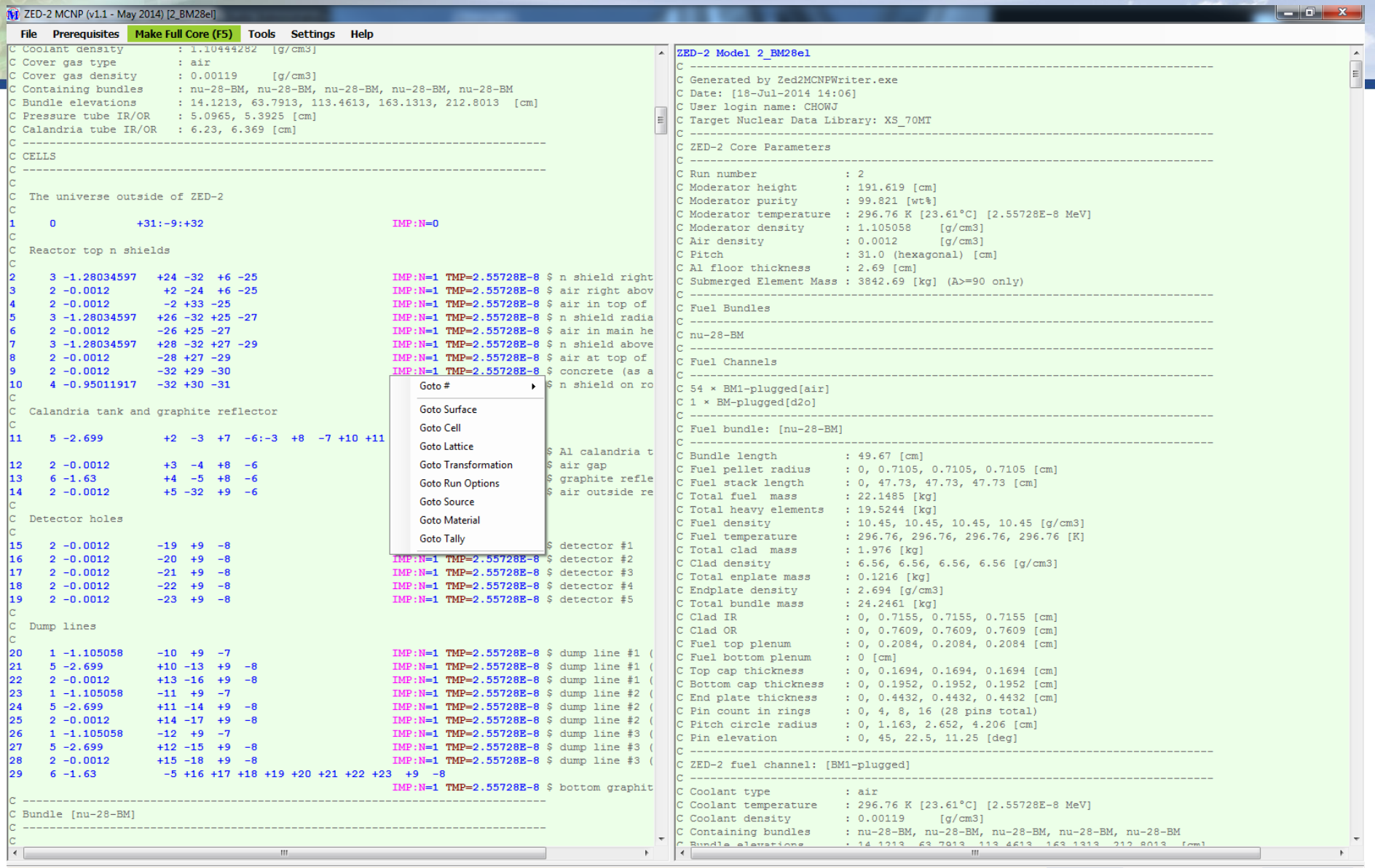
Full Core Configuration



Zed2Serpent



Zed2MCNP



AECL EACL

UNRESTRICTED

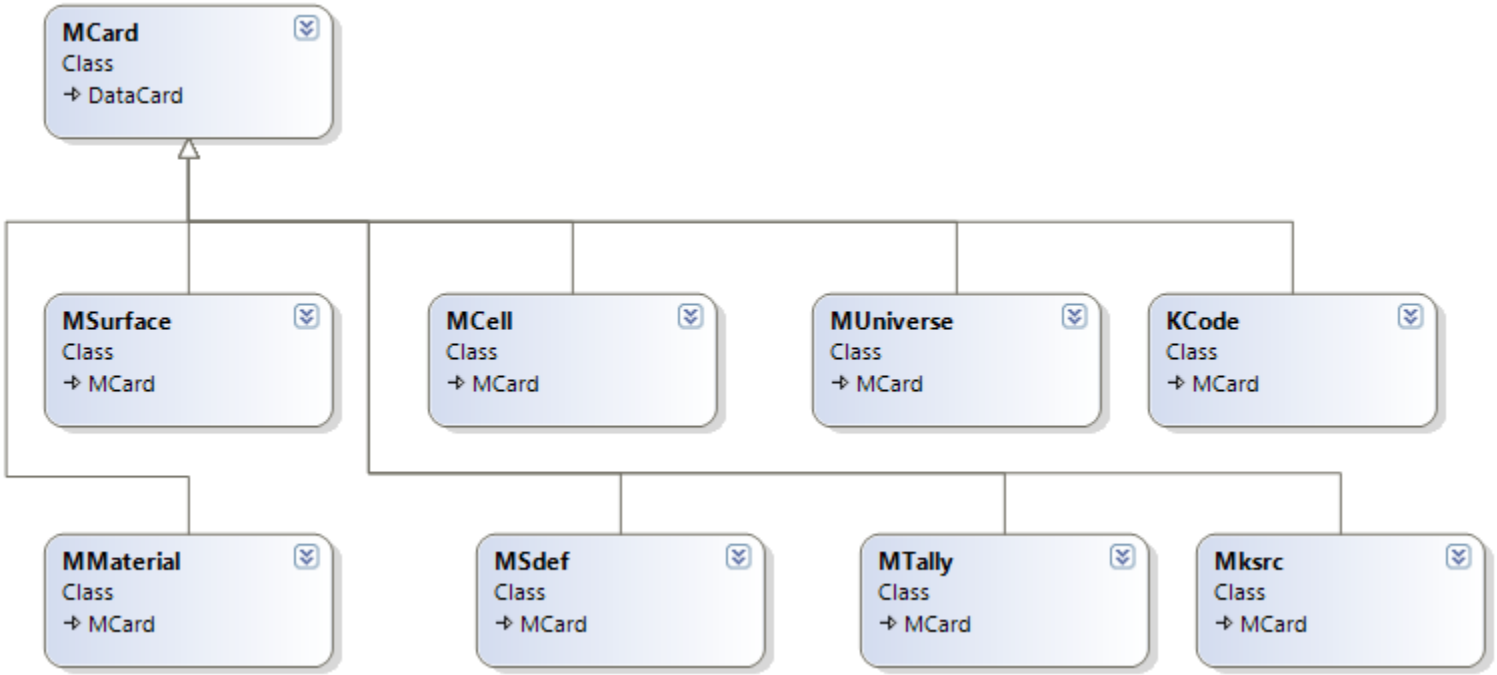
Final Remarks

My personal claim:

This approach of generating input data for simulation can be generalized to apply to any type of simulations, of any physical systems, with any complexity, and using any code.

Will a Serpent Class Library be useful?

MCNP Class Library



KENO Class Library