ORNL/TM-2020/1619

VERA User's Guide for Ex-Core Applications

Eva Davidson

January 2022

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY

ORNL/TM-2020/1619

Reactor and Nuclear Systems Division

VERA User's Guide for Ex-core Applications

Eva Davidson

Date Published: January 2022

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-Battelle, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF INPUTS, OUTPUTS AND SCRIPTS

ACRONYMS

CADIS	Consistent Adjoint Driven Importance Sampling
EFPD	effective full power day
HDF5	hierarchical data formats version 5
GG	General Geometry
ORNL	Oak Ridge National Laboratory
SMR	small modular reactor
VERA	Virtual Environment for Reactor Applications
VR	variance reduction

ACKNOWLEDGEMENTS

The author would like to thank Andrew Godfrey for his guidance and for providing feedback on the scripts required for streamlining VERA ex-core calculations.

The author would also like to acknowledge the entire Exnihilo and MPACT development teams for their patience and user support while the author ran numerous ex-core calculations. The author also appreciates the assistance received from Tara Pandya and Katherine Royston of the Exnihilo development team in setting up the appropriate scripts to plot and display various pieces of information from the output.

ABSTRACT

The Virtual Environment for Reactor Applications, or VERA, allows users to set up models to calculate time-dependent and fully coupled solutions for ex-core quantities of interest such as vessel and coupon fluence, and detector responses for multiple statepoints and cycles. MPACT and COBRA-TF together perform in-core transport calculations with temperature feedback while Shift performs the fluence and detector response calculations in the ex-core region. The in-core region is modeled using VERA's native input format and the ex-core region is defined using Shift's general geometry package, also known as Omnibus General Geometry. Fixed source ex-core calculations with Shift can be run in forward mode without advanced variance reduction (VR) methods, or with Consistent Adjoint Driven Importance Sampling (CADIS), which is an automated VR method.

This document serves as a guide for setting up inputs, running ex-core calculations and post-processing the results.

1. INTRODUCTION

Advanced simulation methods in VERA (Turner et al. [2016]) allow users to set up models to calculate time-dependent and fully coupled solutions for coupon fluence, pressure vessel fluence, and ex-core detector responses at multiple statepoints and for multiple cycles. The in-core transport with temperature feedback is performed by coupled MPACT (Collins et al. [2016]) and COBRA-TF (Avramova [2009]) calculations, and the fluence and detector response calculations in the ex-core region are performed by Shift (Pandya et al. [2016]). The ex-core region is defined using Shift's general geometry package, also known as Omnibus General Geometry (GG) (Johnson et al. [2020]), while the in-core geometry is defined using the standard VERA input (Palmtag and Godfrey [2015]). Fixed source ex-core calculations in Shift can be run in forward mode without advanced variance reduction (VR) methods, or with Consistent Adjoint Driven Importance Sampling (CADIS) (Wagner and Haghighat [1998]), which is an automated VR method. Space- and energy-dependent weight windows for Shift are generated using Denovo, a discrete ordinates solver in the Oak Ridge National Laboratory (ORNL) Exnihilo code suite (Johnson et al. [2020]).

The VERA input set up to perform in-core calculations with MPACT can be used to run ex-core calculations with Shift. Once the VERA input has been set up, no additional user effort is required to model the core or fuel in great detail in order to perform ex-core calculations. Subsequently, vessel fluence calculations can be performed with very little user input, but more complicated ex-core geometry must be modeled using the GG input format.

This document serves as a user's guide for setting up and running ex-core calculations with VERA as well as post-processing the results.

2. RUNNING VERA USING VERARUN

VERA ex-core calculations can be invoked with `verarun`. The user can gain more information regarding all the input arguments for `verarun` by typing the following into the terminal window:

```
verarun
```

This command will show the user a list of the arguments (Listing 1) and a description of what each argument would accomplish.

Listing 1. `verarun` arguments.

```
usage: verarun [-x] [-e email_addr] [-h] [-c config_file] [-N job_name] [-l]
               [-n nprocs] [-O] [--ppn cpus_per_node] [-m mem_per_process]
               [-p project] [-s subdir] [-d vera_install_dir]
               [-v vera_version] [--verbose] [-W] [-w job_id] [-t walltime]
               [--chain] [--debug] [--hostname host]
               [-r {overwrite,readwrite}]
               [--vera-installs-dir vera_installs_dir]
               [input_path [input_path ...]]

Creates and optionally submits machine-specific VERA jobs.

positional arguments:
  input_path            path to VERA input (.inp) or XML (.xml) files

optional arguments:
  -x, --dry-run         dry run only, create but don't execute the PBS script
  -e email_addr, --email-addrs email_addr
                        comma-delimited list of email addresses to notify of
                        job completion, defaults to ${USER}@$(hostname)
  -h, --help            print detailed help
  -c config_file, --host-config-file config_file
                        override host configuration file, supercedes
                        --hostname and --vera-installs-dir
  -N job_name, --job-name job_name
                        name for the PBS job
  -l, --list-vera-versions
                        list available VERA versions
  -n nprocs, --np nprocs, --nprocs nprocs, --num-procs nprocs
                        total number of processors need for the job (mpiexec
                        -np param), defaults to value computed from input
  -O, --output-job-name
                        print the job id to stdout
  --ppn cpus_per_node, --pnode cpus_per_node
                        specify processors per node, by default this is
                        calculated
  -m mem_per_process, --pmem mem_per_process, --proc-memory mem_per_process
                        specify memory required per processor in GB, defaults
                        to undefined
  -p project, --project project
                        optional project or account to specify for the job,
                        overriding any default, where a value of "none" omits
                        a project
  -s subdir, --subdir subdir
```

```
                        create subdir, a value of "." specifies automatically
                        generated subdir name
  -d vera_install_dir, --vera-dir vera_install_dir
                        path to VERA installation, superseding --vera-
                        installs-dir, --vera-version, and the host
                        configuration
  -v vera_version, --vera-version vera_version
                        name of VERA version to use
  --verbose             turn on verbose messaging
  -W                    wait on job last submitted via verarun, overrides -w
  -w job_id, --wait-job-id job_id
                        ID of job which must complete before starting this job
  -t walltime, --wall-time walltime
                        wallclock execution time in floating point hours,
                        defaults to 24.0

advanced arguments:
  --chain, --chain-jobs
                        each job depends on its predecessor
  --debug               debug mode
  --hostname host       force the hostname
  -r {overwrite,readwrite}, --restart {overwrite,readwrite}
                        optional restart mode
  --vera-installs-dir vera_installs_dir
                        path to vera_installs directory containing VERA
                        versions

Version 1.4
```

When running ex-core problems with a supplemental Omnibus ex-core file, ensure that both the VERA input and ex-core files are in the same folder. To run Shift calculations in `cadis` mode, type in:

```
verarun vera_input_filename.inp
```

Note that if there is an Omnibus ex-core file, do not forget to insert `excore-filename.excore.xml` within the `[SHIFT]` block in the VERA input using the `excore_filename` parameter [Pandya et al. [2020]].

2.1 SETTING UP MPACT AND SHIFT PROCESSORS

When running ex-core calculations with VERA, the problem should be set up so that MPACT and Shift run on different nodes. Otherwise, it is possible to run out of memory on a node if both MPACT and Shift are sharing that node. For example, if a node has 32 processors, it is good practice to ensure that MPACT is running on a multiple of 32 processors.

The `num_space` parameter in the `[MPACT]` block refers to the number of processors MPACT requires for the problem, and it can be toggled to ensure that MPACT runs on its own independent nodes. The `num_blocks_i` and `num_blocks_j` parameters in the `[SHIFT]` block provide information on the number of Shift processors required. Note that these Shift parameters are associated with CADIS, and CADIS is the default method when invoking Shift for ex-core calculations. When using `verarun` to run a VERA ex-core calculation, the default number of Shift processors is `10 num_blocks_i` multiplied by `10 num_blocks_j`, which is equal to `100` processors for Shift. However, the user is free to run with any combination of `num_blocks_i` and `num_blocks_j` to utilize all the processors available. It must be noted that for improved performance and efficiency, it is preferable to keep the `num_blocks_i` equal to `num_blocks_j`. The user must also bear in mind that if a cluster with 32 processors per node is being used, and the default 100 processors is used for the Shift calculation when using `verarun`, then there will be 28 processors that will not be used in the calculation.

Shift ex-core calculations can also be performed without advanced variance reduction. In this case, the user must specify that the problem has to be run in `forward` mode in the `[SHIFT]` block. The user can state the total number of processors required for the VERA simulation by typing the following command into the terminal window:

```
verarun -n nprocs
```

The `nprocs` value is equal to the total number of MPACT and Shift processors. `verarun` reads the number of MPACT processors required for the problem from the `num_space` input parameter within the `[MPACT]` block; the remainder of the processors will be assigned to Shift.

3. SHIFT DEFAULT PARAMETERS

Most hybrid parameters that allow ex-core problems to run efficiently have been set as defaults in the `SHIFT.ini` file in the build directory, `$HOME/bin/Init`. The parameters in the `SHIFT.ini` file have been set to the optimum values for most problems except for a few parameters that must be specified by the user for their specific problems. Expert users of hybrid methods can toggle some of the default parameters for their specific models. More information regarding these parameters and other parameters for the [SHIFT] block in the VERA input can be found in the manual by Pandya et al. [2020].

Table 1 lists the default parameters within the `SHIFT.ini` file (Pandya et al. [2020]).

Table 1. Shift default parameters

Parameter	Default Value	Value Options	Description
`adjoint`	`true`	-	Set to true if running in `cadis` mode
`create_unique_pins`	`false`	-	Currently set to false for all problems due to memory constraints. If set to true, each pincell would have unique compositions
`dimension`	`3`	-	Spatial dimension of the problem
`do_transport`	`true`	true/false	Perform Monte Carlo transport if set to true; otherwise, exit the calculation after performing a raytrace of the geometry
`eq_set`	`sc`	-	Solution method or spatial discretization for hybrid calculation (sc = step characteristics) for ex-core calculations
`extend_axial_mesh_size`	`5.0`	positive value	Adjoint axial mesh size in ex-core region
`global_log`	`info`	debug/diagnostics/ status/info/warning/ error/critical	Level of global log information
`max_delta_z`	`5.0`	User can decrease mesh size for hybrid calculation	Maximum mesh size in z direction used for hybrid calculations
`mesh`	`1`	-	Number of mesh cells per pincell
`mode`	`n`	n/np	Type of particle to transport (neutrons/ neutrons and photons)

Parameter	Default Value	Value Options	Description
`Np`	`1000000`	Positive integer	Number of particles to transport
`new_grp_bounds`	`6.0653E+06` `3.6788E+06` `2.3457E+06` `1.6530E+06` `8.2085E+05` `2.4176E+04` `1.0130E+02` `1.0000E-05`	User can specify energy group boundaries for hybrid calculation	Collapsed group boundaries for weight windows. Note these values are lower energy bounds and these default values were obtained from BUGLE-96 neutron group bounds
`num_blocks_i`	`10`	Positive integer	Number of partitions (processors) in x
`num_blocks_j`	`10`	Positive integer	Number of partitions (processors) in y
`num_groups`	`8`	User can specify number of energy groups for hybrid calculation	Number of energy groups
`output_adjoint`	`false`	true/false	Output adjoint flux to Shift HDF5 file and adjoint source to a separate HDF5 file
`output_external_source`	`false`	-	Placeholder for future capability to add external source
`output_fission_source`	`true`	true/false	Output the initial fission source for each calculation
`output_geometry`	`true`	true/false	Output HDF5 files of raytraced geometry and compositions for each state
`Pn_order`	`0`	Higher orders allowed and limited by memory	Order of moments: going to higher orders will require more memory
`problem_mode`	`cadis`	forward/cadis	Run with (cadis) or without (forward) hybrid capability
`refl_mesh_size`	`5.0`	Positive value	Radial mesh size in ex-core region
`src_disc_l2_error`	`1e-3`	Advanced user can toggle this option	Maximum L2 error for point-sampling discretization
`src_disc_max_samples`	`1000000000`	Advanced user can toggle this option	Maximum number of discretization samples
`src_disc_samples_per_batch`	`10000000`	Advanced user can toggle this option	Maximum number of discretization samples
`store_fulcrum_string`	`false`	-	Save fulcrum string as file

Parameter	Default Value	Value Options	Description
`thermal_energy_cutoff`	`10.0`	-	Cutoff for treatment of thermal neutrons
`transfer`	`fiss_src`	-	Information transferred from MPACT to Shift (currently the option for ex-core calculations)
`transport`	`ce`	-	Continuous energy (only option for ex-core calculations)
`upscatter_subspace_size`	`20`	User can specify a higher value	Upscatter solver subspace size
`upscatter_verbosity`	`high`	-	Log verbosity for up-scatter solver
`use_external_source`	`false`	-	Placeholder for future capability that indicates an external source is being used
`use_fission_source`	`true`	-	Flag that indicates a fission source is being used
`use_mpact_spectrum`	`false`	true/false	Flag for whether to use the MPACT fission source energy spectrum to sample for neutron source energies: if false, a nuclide-watt spectrum is used; the neutron source energies are assumed to have ^{235}U, ^{238}U, ^{239}Pu, ^{241}Pu Watt spectra based on the fission source strength of these isotopes
`verbosity`	`low`	none/low/medium/high	Specify how often to print out particles being transported: *low* means less output, and *high* means more output
`xs_library`	`v7-56`	-	Name of SCALE multigroup data library file

4. VESSEL FLUENCE CALCULATIONS

Running vessel fluence simulations with VERA is the simplest application for performing ex-core calculations based on the minimal effort required by the user to set up an input. Consider a VERA input that was previously set up or run with other physics options in VERA. A user can perform a vessel fluence calculation on the same input if an additional line is added to the first state block in the VERA input.

```
[STATE]
  excore_transport on
```

Next, a user must provide more information regarding the number of meshes and the number of particles they want to run for the vessel fluence calculation using the `[SHIFT]` block in the VERA input. This input block can be added at the end of the VERA input file. Note that these vessel fluence calculations are run without the bioshield.

As an example, let us review the following input:

Listing 2. Simple vessel fluence VERA input.

```
[SHIFT]
  Np               50e6
  !
  ! tally_information
  n_bounds     2.0e7 1.0e6
  num_theta    128
  num_axial    10
  radial_mesh 0 61.99999 63.0 64.0 65.0 66.0 67.0 68.0 69.0 70.0 71.0 72.00001
```

This input shows that this particular case will be run with 50 million particles, and the energy bounds on the tally will range from 2.0×10^7 eV to 1.0×10^6 eV. The energy bounds must be supplied in descending order. There are 128 meshes along theta and 10 axial meshes. The bounds of the radial mesh are provided from 0 cm to the outer extent of the vessel. The radial mesh must always start with 0 cm, which is the origin. The second value is then the inner radius of the vessel liner or vessel, whichever the user prefers.

The user can add additional inputs that were provided in the previous Shift default parameters section to their `[SHIFT]` block. In the following input, the `output_adjoint` parameter is set to `true` so that the user can use the python post-processors available with VERA to visualize the adjoint. The default number of processors that a Shift calculation runs on is 100 processors. This can be changed to 400 processors or any other number, as the user prefers. In the input below, the Shift calculation will be performed on 400 processors (20×20).

Listing 3. VERA vessel fluence input with additional parameters.

```
[SHIFT]
  Np               50e6    ! # of particles (Default: 1,000,000 particles)
  output_adjoint true      ! Produces HDF5 file with the adjoint
  num_blocks_i     20      ! Denovo decomposition in x direction
  num_blocks_j     20      ! Denovo decomposition in y direction
  !
  ! tally_information
  n_bounds     2.0e7 1.0e6
  num_theta    128
  num_axial    10
```

```
radial_mesh 0 61.99999 63.0 64.0 65.0 66.0 67.0 68.0 69.0 70.0 71.0 72.00001
```

After the Shift parameters are added, the input is ready to be run using `verarun`. Once the run is complete, the output can be viewed in VERAView by clicking on `Load Single New File` and selecting the HDF5 output file that ends in your `input_filename.h5`. Note that there are several HDF5 output files in the run folder; however, the output read by VERAView is the `input_filename.h5`.

Once the file is loaded, the following screen will be displayed.

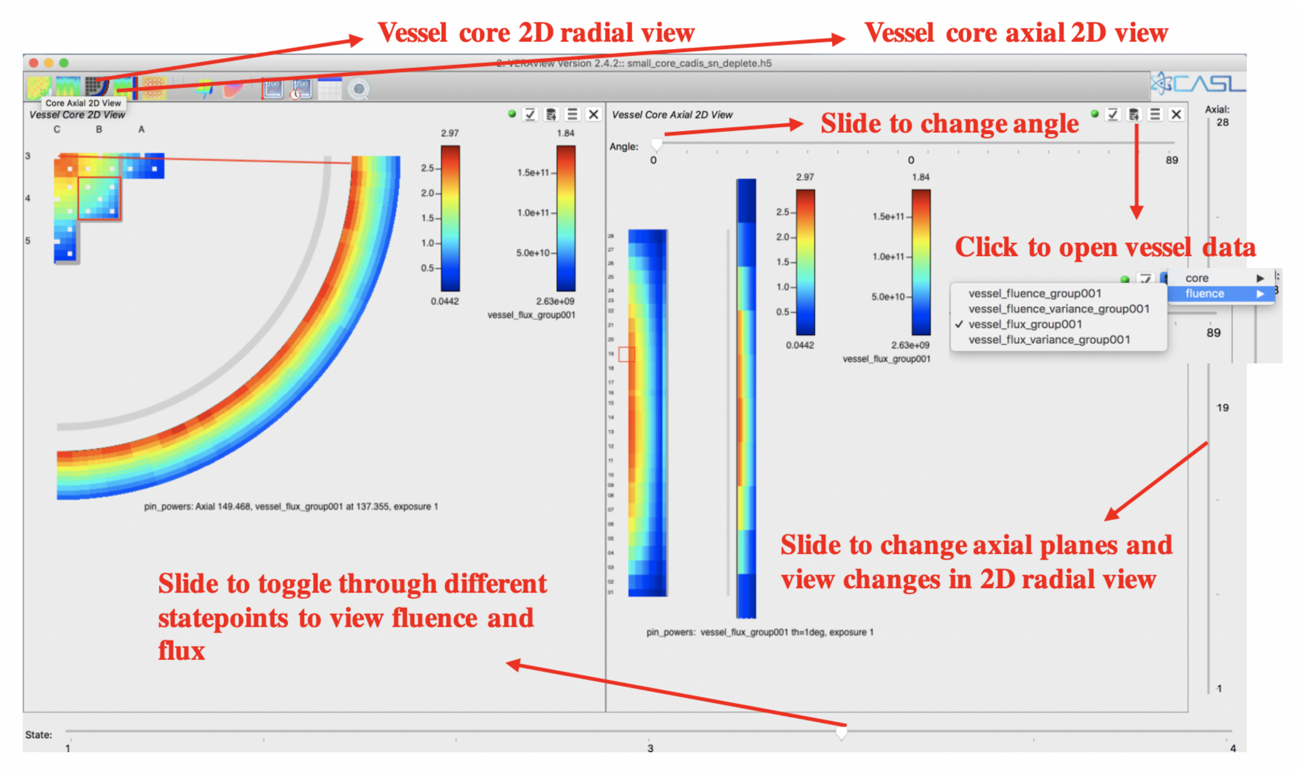
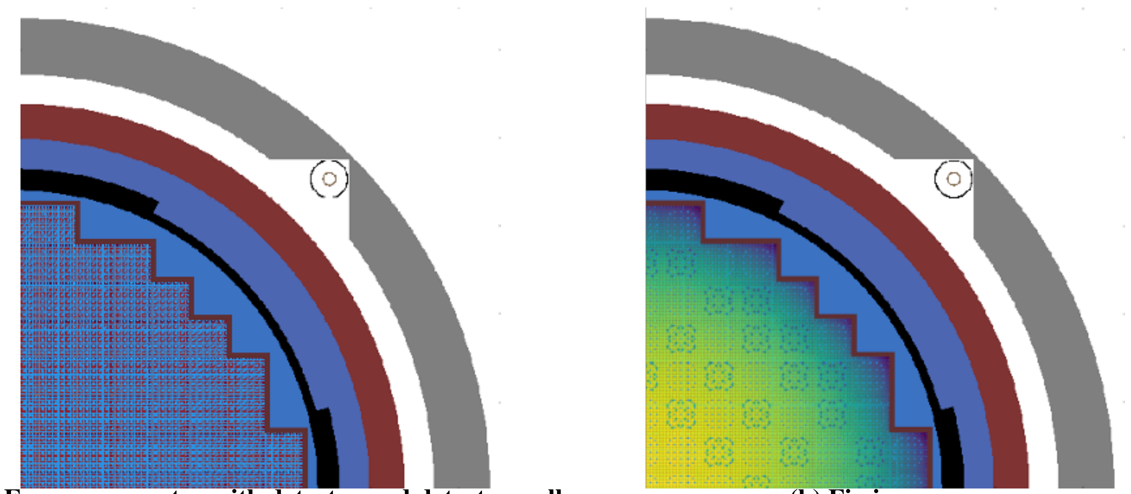

Vessel core 2D radial view
Vessel core axial 2D view
Slide to change angle
Click to open vessel data
Slide to change axial planes and view changes in 2D radial view
Slide to toggle through different statepoints to view fluence and flux
VERAView Version 2.4.2:: small_core_cadis_sn_deplete.h5
Core Axial 2D View
Vessel Core 2D View
Vessel Core Axial 2D View
Angle:
Axial:
State:
core
fluence
vessel_fluence_group001
vessel_fluence_variance_group001
✓ vessel_flux_group001
vessel_flux_variance_group001
pin_powers: Axial 149.468, vessel_flux_group001 at 137.355, exposure 1
pin_powers: vessel_flux_group001 th=1deg, exposure 1

Figure 1. VERAView screen with vessel fluence output.

In this figure, the plot on the left shows the radial view of the pin powers and the vessel flux, while the plot on the right shows an axial view of the pin powers and vessel flux with two different color maps shown for pin powers and the vessel flux. Sliding the bar at the bottom of the screen allows the user to toggle between different statepoints, and sliding the bar at the top of the screen can be used to change the angle. The bar on the right-hand side can be used to toggle between different axial planes. The user can view the vessel fluence or flux by clicking on the "hamburger" icon on the top right corner of the screen and selecting the data to plot. Note that in the figure generated by VERAView, the vessel is not presented at scale and looks like it is really far away from the reactor. VERAView also does not plot the core pads.

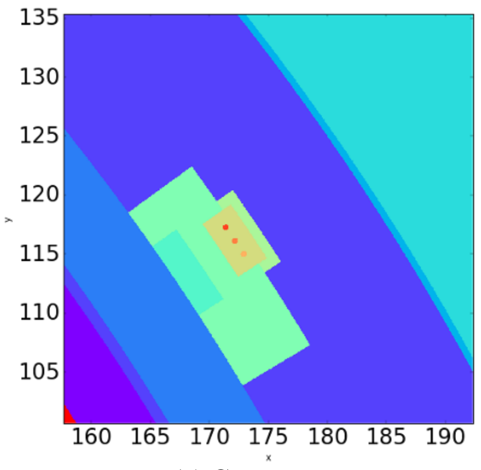
5. OTHER EX-CORE CALCULATIONS

To perform ex-core calculations such as detector response beyond the pressure vessel or coupon fluence calculations, the user must set up a separate input to define the ex-core geometry using Omnibus input format. This process enables the user to set up a detailed core model using VERA and to integrate this core model with the Omnibus geometry to define the ex-core problem. Setting up an Omnibus input can be as simple or as complex as the geometry for the region the user is interested in studying. The Omnibus geometry package is called General Geometry (GG), and it is based on the concept of geometric primitives combined with unions and intersections. If the user is familiar with SCALE's KENO or Monte Carlo N-Particle (MCNP) code, then GG-based input should not present a steep learning curve. Figure 2 below shows an ex-core detector response problem set up for Shearon Harris [Smith et al. [2018] and Davidson et al. [2020]]. Figure 3 shows coupons in single surveillance capsules modeled for Watts Bar Unit 1 [Davidson et al. [2018]. Figure 4 shows a full core Watts Bar model [Davidson et al. [2018] with all the ex-core detectors (power range and source range) and surveillance capsules (single and double) off the pads.

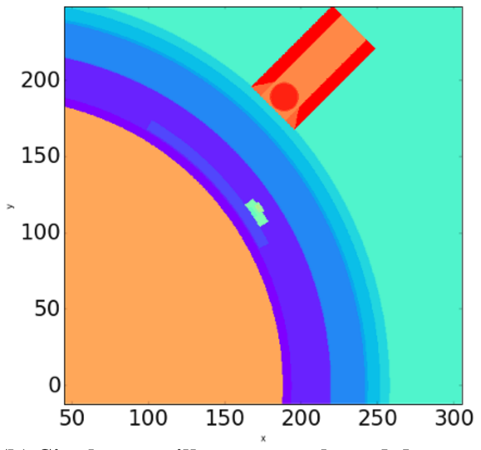


(a) Ex-core geometry with detector and detector well **(b) Fission source**

Figure 2. Example of (a) ex-core detector within a detector well in the bioshield and (b) the spatial fission source.



(a) Coupons



(b) Single surveillance capsule and detector

Figure 3. Example of a geometry set up with a single surveillance capsule and a power range detector.

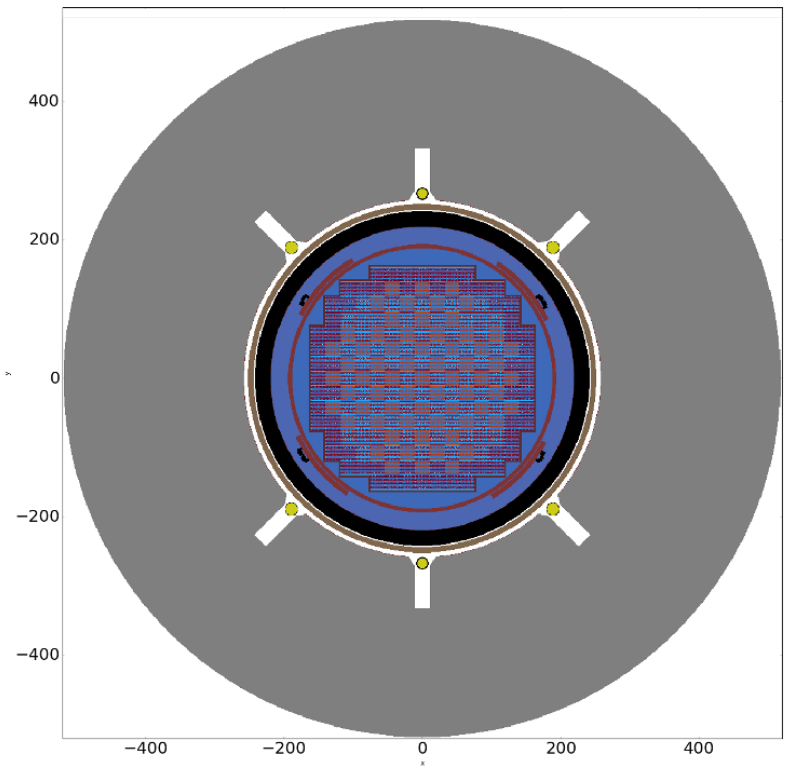
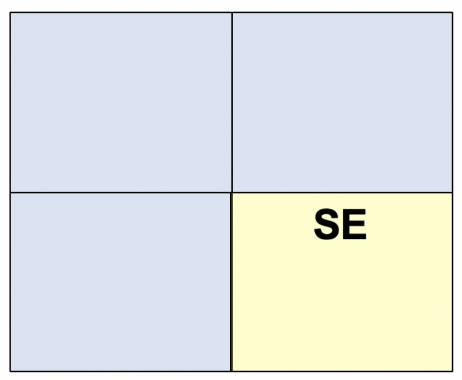


Figure 4. Full core model.

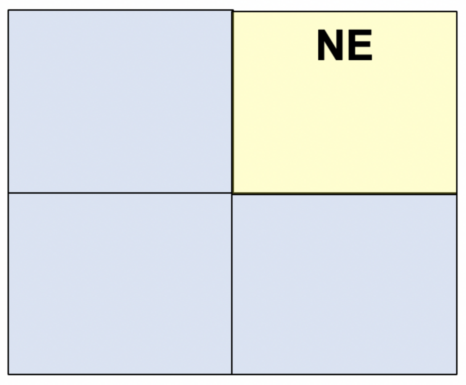
5.1 SETTING UP THE OMNIBUS EX-CORE GEOMETRY

As mentioned earlier, the effort required to set up the ex-core input depends on the complexity of the ex-core region's geometry that the user wants to model. In this section, the steps required to set up the ex-core geometry and any potential user pitfalls are discussed.

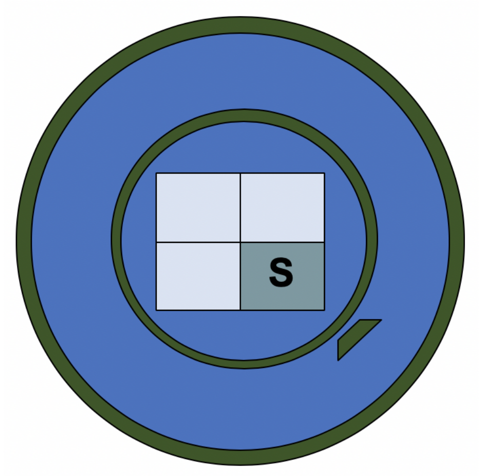
Shift does not support rotational symmetry through VERA: it only supports mirror symmetry. Therefore, any VERA input with rotational symmetry must be unfolded manually by the user in the input before running VERA for ex-core calculations. Another quirk to note is that MPACT and Shift model different quadrants when running quarter core problems. This can be a major source of user input error if the user adds features to the ex-core geometry when running a quarter core model with VERA, especially if the ex-core features are not symmetrical. The difference in these quadrants being modeled is shown in Figure 5.



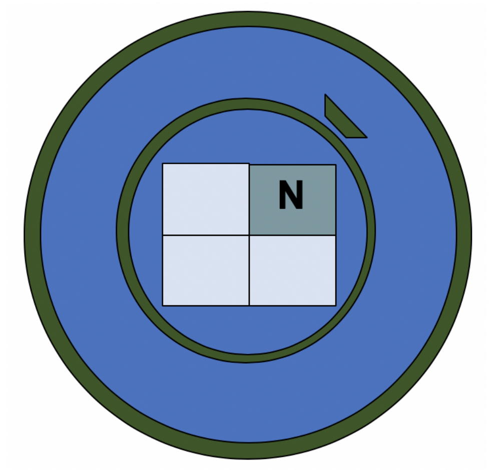
(a) MPACT quadrant



(b) Shift quadrant



(c) Location of ex-core feature in MPACT quadrant



(d) Location of ex-core feature in Shift quadrant

Figure 5. Quarter core quadrants modeled with MPACT and Shift for quarter core problems.

An Omnibus input file requires several blocks of input (Listing 4). Detailed information regarding the set up of Omnibus inputs can be found in the Exnihilo manual (Johnson et al. [2020]). However, the process to set up an Omnibus GG input is revisited here for the purposes of setting up an input for VERA ex-core calculations.

The input always starts with the [GEOMETRY] block, which tells Omnibus that the input following this block of text will be geometry related. The global parameter is followed by a short string describing the type of geometry being set up in the input file. Next, the user must provide one standard line of input, [UNIVERSE=core reactor], that incorporates the VERA core geometry within the Omnibus input.

Once the VERA core geometry has been incorporated, the boundaries are defined for the entire ex-core geometry. A boundary box is sufficient in most cases, and this bounding box is defined using the cuboid shape in Omnibus. This boundary box must be a cuboid and cannot be another shape because Shift in VERA cannot apply a boundary condition to any other shapes.

Once the boundary is defined, the shapes within the excore universe are defined: coupons, detectors etc. A complete list of shapes and their definitions are inluded in the Exnihilo manual (Johnson et al. [2020]). Table 2 summarizes the shapes available through the Omnibus input format.

Table 2. OMNIBUS shapes

[SHAPE=TYPE]	DESCRIPTION
CUBOID/BOX	Box
SPHERE	Sphere
CYL	Cylinder
CYLSEGMENT/PAD	Cylindrical segment
RING/CYLSHELL	Cylindrical shell
PRISM	Regular prism
SLAB	Infinite slab
PLANE	Infinite half-space
WEDGE	Wedge
CONE	Cone
ELLIPSOID	Ellipsoid
HOPPER	Hopper
RIGHTHET	Right tetrahedron
TRIPRISM	Triangular prism
ECYLINDER	Elliptical dodecahedron
RHOMBDOD	Rhombic dodecahedron
PPIPED	Parallelepiped
QUADRIC	General quadric

After the shapes are defined, holes must be defined to incorporate the core universe from the VERA input or to include any detectors that might have been set up as an independent universe. These holes contain independent universes defined within the global geometry.

The cell definitions follow the hole definitions. All cells can be defined using combinatorial geometry. More examples of this are presented in subsequent sections. The user must bear in mind that there are two "cutoff" points in the radial direction for importing the geometry from the VERA input. The user has the choice of importing the geometry up to (1) the outer radius of the barrel, or (2) the outer radius of the pressure vessel, and then modeling everything beyond that point in the Omnibus ex-core file. For example, the user might consider importing up to the outer radius of the barrel from the VERA input and then modeling everything beyond it in the Omnibus ex-core file to model the coupons and surveillance capsules. In another scenario, a user might want to import everything up to the outer radius of the pressure vessel from the VERA input and then add features beyond the vessel such as a bioshield or ex-core detectors in the Omnibus ex-core file. In the axial direction, the user must ensure that the extents in the z-direction for the boundary box extend at least from the bottom of the lower core plate to the top of the upper core plate. Note that all the regions within the boundary box must be defined. A simple Omnibus ex-core input is presented in Section 5.2, but the main pieces of an Omnibus ex-core input are shown below. The [COMP], [RESPONSE] and [TALLY] blocks will be discussed in more detail in the following subsections.

Listing 4. Blocks of input in Omnibus ex-core file.

```
! Every Omnibus GG input begins with the [GEOMETRY] block

[GEOMETRY]
global "ex-core file for an awesome reactor"

! Incorporate core from VERA input - this line never changes
[UNIVERSE=core reactor]

! ----------------------------------------------------------------------------
! Define the ex-core universe with boundary - in this case a bounding box

[UNIVERSE=general excore]
boundary "box"

! Boundary box
[UNIVERSE][SHAPE=cuboid box]
faces -190 190 -190 190 -6 250

! ***************************************************************************
! Define other shapes within this bounding box

! Outer radius of the barrel or vessel
[UNIVERSE][SHAPE=cyl outer_radius_barrel_or_vessel]
radius  220.0
extents -6.0 240.0
origin   0 0 0
axis    z

! ----------------------------------------------------------------------------
! After all shapes are defined, a hole is defined to insert the VERA "reactor"
! core defined earlier

[UNIVERSE][HOLE thecore]
fill reactor
translate -outer_radius_barrel_or_vessel -outer_radius_barrel_or_vessel 0.0

! ----------------------------------------------------------------------------
! Define the cells

[UNIVERSE][CELL void]
comp void
shapes +outer_radius_vessel -shield_inner_diameter

! ----------------------------------------------------------------------------
! Define the compositions

[COMP]

[COMP][MATERIAL al]
```

```
matid 0
temperature 293
zaid 13027
nd    6.0E-02

! -----------------------------------------------------------------------------
! Set up a response block to look at a specific detector response

[RESPONSE=xs]
name          boron
mt            101
nuclide       5010
physics       ce
particle_type n
tmp           293

! -----------------------------------------------------------------------------
! Set up tallies
[TALLY]

[TALLY][CELL flux_tally]
cells void
reactions flux
```

5.2 A SIMPLE EX-CORE INPUT

An example of a simple input for an SMR ex-core region as shown in Figure 6 is defined in this section. In this example (Listing 5), everything up to the outer radius of the vessel is pulled in from the VERA input. Only a biological shield is defined outside the vessel using the ex-core input shown below.

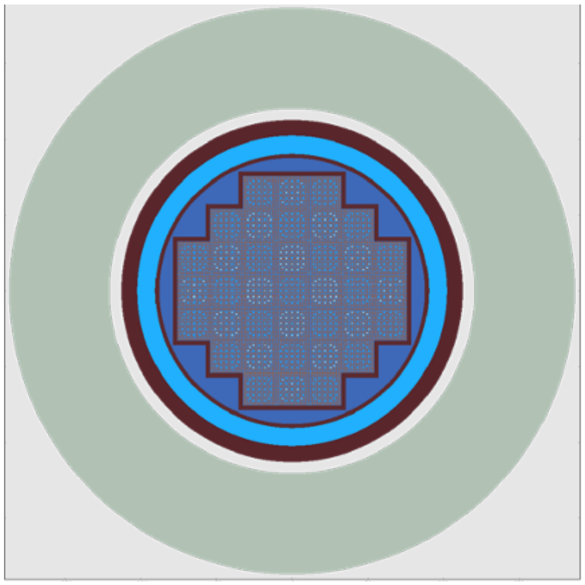


Figure 6. Simple ex-core region for an SMR.

In this input, the `global "excore"` geometry is defined. Next, the general universe is defined with a boundary box that encompasses the entire geometry. Following this definition, the shapes are defined for the bioshield. Note that all geometry up to the outer radius of the vessel is being pulled from the VERA input. Therefore, no additional geometry definitions within the vessel region need to be defined in the Omnibus input file. The `HOLE` is defined next to integrate the VERA core geometry with the `excore` universe defined here. Since the VERA and Omnibus origins are not the same, they must be aligned and this alignment is achieved with a `translate` parameter. This is based on the outer radius of the vessel in this problem and is defined as follows:

```
translate -outer_radius_of_the_vessel -outer_radius_of_the_vessel 0.0
```

The holes must be defined after the shape definitions, and this order matters in Omnibus GG. The `[CELL]` block is defined next using the shapes defined earlier. Any region lying within the surface or has a negative sense to a plane is defined by a negative (-) sign and any region lying outside a surface or has a positive sense to a plane is defined by a positive (+) sign. Next, the compositions and tallies are defined using the `[COMP]` and `[TALLY]` blocks. An example for a cell flux tally in the bioshield region is shown. The user can define as many neutron energy bins as desired if they want to tally binned flux; otherwise, the user can omit this line to tally the total flux in the cell.

Listing 5. Simple ex-core file.

```
! ============================================================================
!                          SIMPLE SMALL MODULAR REACTOR EX-CORE REGION
! ============================================================================

[GEOMETRY]
global "excore"

! Reactor core from VERA input inserted into this universe
[UNIVERSE=core reactor]

! -------------------------- UNIVERSE: EX-CORE -------------------------------

! ****************************************************************************
! Define the problem boundary
[UNIVERSE=general excore]
boundary "box"

! boundary box
[UNIVERSE][SHAPE=cuboid box]
faces -190 190 -190 190 -6 250

! ****************************************************************************
! Define the shapes (or surfaces in MCNP/KENO terminology)
! Vessel radius is 112.79 and all the surfaces outside the vessel is defined

! Shield
[UNIVERSE][SHAPE=cyl shield_id]
radius  120.4
extents  -6.0 240.0
origin  0 0 0
axis    z

[UNIVERSE][SHAPE=cyl shield_od]
radius  186.67
```

```
extents  -6.0  240.0
origin  0 0 0
axis    z

! ****************************************************************************
! Create a hole and fill in universes defined outside of the
! [UNIVERSE=general excore]. In this example, there is only one universe,
! which is the VERA core model. Everything up to the outer radius of the
! vessel, which is defined in the VERA geometry, is dropped in this "hole."
! ****************************************************************************

[UNIVERSE][HOLE thecore]
fill reactor
translate -112.79 -112.79 0.0

! Define cells

[UNIVERSE][CELL void_reg1]
comp void
shapes +thecore -shield_id

[UNIVERSE][CELL bioshield]
comp concrete
shapes +thecore +shield_id -shield_od


[UNIVERSE][CELL void_reg2]
comp void
shapes +thecore +shield_id +shield_od -box

! ------------------------------ COMPOSITIONS -------------------------------

[COMP]

[COMP][MATERIAL void]
matid 0
temperature 293
zaid --
nd   --

! Concrete, Oak Ridge (ORNL) 2.3 g/cm3 material data from PNNL 15870 Rev 1
! The density specified above is estimated to be accurate to 2 significant digits.
! Uncertainties are not addressed. The following data were calculated from the
! input weight and converted to number densities.
[COMP][MATERIAL concrete]
matid 1
temperature 293
zaid 1001  1002  6012  6013  8016  8017  11023 12024 12025 12026 13027 14028
     14029 14030 19039 19040 19041 20040 20042 20043 20044 20048 26054 26056
     26057 26058
nd   8.49996E-03 1.70019E-06 1.99790E-02 2.24257E-04 3.54247E-02 8.52237E-05
     1.63354E-05 1.46966E-03 1.86056E-04 2.04848E-04 5.55961E-04 1.56827E-03
     7.95785E-05 5.25422E-05 3.76029E-05 4.03217E-09 2.71356E-06 1.07627E-02
     7.21657E-05 1.55434E-05 2.32041E-04 2.10946E-05 1.12950E-05 1.77148E-04
     4.09321E-06 5.40617E-07
```

```
! -------------------------------- TALLIES --------------------------------
[TALLY]

! Tally the cell flux in the bioshield
[TALLY][CELL flux_tally]
reactions flux
cells bioshield
nbins 20e6 1e5 1 1e-5
```

5.3 SMR EX-CORE INPUT WITH A DETECTOR

An example of a slightly more complex ex-core input for a small modular reactor (SMR) shown in Figure 7 is defined in this section. A detector well is modeled in the ex-core region in this model.

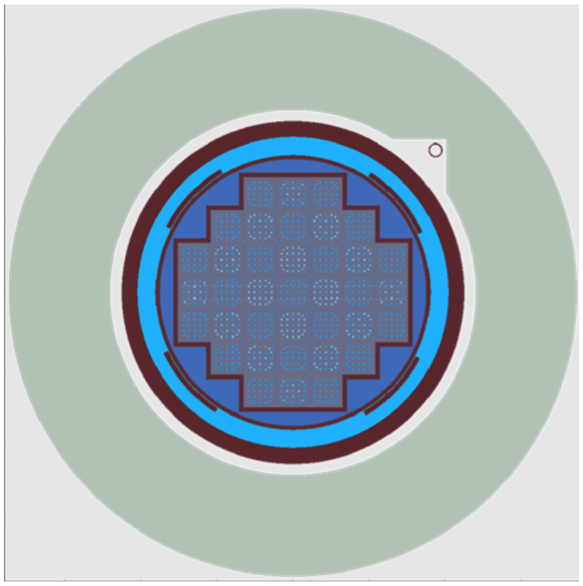


Figure 7. An SMR with an ex-core detector.

See Listing 6 for the Omnibus input. The first step in setting up the ex-core geometry is to define the `[GEOMETRY]` block and to also integrate the VERA core geometry with the ex-core geometry using the `[UNIVERSE=core reactor]` block. Next, a separate universe that defines the detector region in the northeast quadrant is set up. This universe is called `det_cutout` and has a boundary shape called `shield_id`. Under the `SHAPE` (or surface) definitions, it can be seen that `shield_id` is a ring shape, and the detector lies within this ring. The ring's inner and outer radii are the inner and outer radii of the bioshield. The detector well region is defined by the `cuboid` shape, which is cut by the ring, giving it the detector well shape. The detector is split into top and bottom halves, and a pipe surrounds the detector that extends the entire length of the top and bottom halves. After the shapes are defined in the `det_cutout` universe, the cells are defined. Finally, the `hole_boundary` for the `det_cutout` universe must be defined. The universe lies within the `shield_id` boundary.

After the `det_cutout` universe has been defined, the `excore` universe is outlined. This universe contains a boundary box shape that encompasses the entire geometry. In this example, the geometry up to the outer radius of the barrel from the VERA input is integrated into this ex-core file. All regions beyond the barrel are defined in the `excore` universe. This means that the pads, reactor pressure vessel liner, and vessel in the ex-core file must be redefined in the ex-core file. This gives the user the flexibility to add surveillance capsules off the pads or within the vessel itself, or it allows the user to expand the ex-core region beyond the vessel to add the bioshield and to add a detector, as in this case.

After the boundary for the problem has been defined, the next step is to define all the shapes to be represented in the ex-core geometry. After the shape definitions, the `HOLES` are defined to integrate the universes defined earlier within the `excore` universe. In this example, the reactor universe represent the VERA geometry and the `detector_cutout` universe representing the detector well and the detector are inserted within the `excore` universe. The reactor universe is translated within `thecore` hole because the origin of the VERA geometry and the Omnibus geometry must align. This translation is based on the outer radius of the barrel for this problem and is defined as follows:

```
translate -outer_radius_of_the_barrel -outer_radius_of_the_barrel 0.0
```

The holes must be defined after the shape definitions, and this order matters in Omnibus GG. Next, the cells are defined. All regions inside a surface is defined by the minus (-) symbol, and all regions outside a surface are defined by the plus (+) symbol. For example, the moderator region is defined by all of the region outside the VERA geometry that is defined by `+thecore`, inside the reactor vessel liner (`-rpv_liner`), and outside the surfaces defining the pads (`+pad_ne +pad_nw +pad_sw +pad_se`). The user must keep in mind that all the regions extending up to the problem boundary, which are defined as `box` in this case, must be defined using cells.

After all the shapes have been defined, the next block is the `[COMP]` blocks in which the compositions are defined. Note that these compositions are referenced in each cell that was defined earlier to specify the material in that cell.

The `[RESPONSE]` blocks are defined next, and here, any specific types of responses related to dose or reaction rates are specified. The example here sets up a response for fission reactions in ^{235}U. The user can also specify the associated MT values from the ENDF files here. This response can then be used within the `[TALLY]` block that is set up next to calculate the detector response. Different types of `[RESPONSE]` blocks can be found in the Johnson et al. [2020].

Listing 6. Ex-core file with bioshield and detector.

```
! ==========================================================================
!                          MORE COMPLEX SMR EX-CORE REGION
! ==========================================================================
[GEOMETRY]
global "excore"

! Reactor core from VERA input inserted into this universe
[UNIVERSE=core reactor]

! ============================================================
!   UNIVERSE: DETECTOR - NE QUADRANT
! ============================================================

[UNIVERSE=general det_cutout]
boundary shield_id
```

```
! *******  SHAPE DEFINITIONS FOR DETECTOR IN THE NE QUADRANT  *******

[UNIVERSE][SHAPE=ring shield_id]
ri 120.4
ro 186.67
extents -6 240.0
origin 0 0 0

! Detector well
[UNIVERSE][SHAPE=cuboid void_reg]
xmin 60.0
xmax 101.0
ymin 60.0
ymax 101.0
zmin 20.0
zmax 220.0

[UNIVERSE][SHAPE=cyl det_tube_id_bot]
radius  3.73
extents 20.0  120.0
origin  93.62 93.62 0
axis z

[UNIVERSE][SHAPE=cyl det_tube_id_top]
radius  3.73
extents 120.0  220.0
origin  93.62 93.62 0
axis z

! Detector Pipe
[UNIVERSE][SHAPE=cyl det_pipe_od]
radius  4.67
extents 20.0  220.0
origin  93.62 93.62 0
axis z

! ********  CELL DEFINITIONS FOR DETECTOR IN THE NE QUADRANT  *****

[UNIVERSE][CELL detector_top]
comp void
shapes -det_tube_id_top

[UNIVERSE][CELL detector_bot]
comp void
shapes -det_tube_id_bot

[UNIVERSE][CELL detector_housing]
comp ss
shapes +det_tube_id_top +det_tube_id_bot -det_pipe_od

[UNIVERSE][CELL void]
comp void
shapes -shield_id -void_reg +det_pipe_od

[UNIVERSE][CELL shield]
comp concrete
```

```
shapes -shield_id +void_reg

[UNIVERSE][CELL hole_boundary]
comp concrete
shapes -shield_id

! ============================================================
!                          UNIVERSE: EX-CORE
! ============================================================

! *****************  DEFINE PROBLEM BOUNDARY  ****************

! Define the problem boundary
[UNIVERSE=general excore]
boundary "box"

! boundary box
[UNIVERSE][SHAPE=cuboid box]
faces -190 190 -190 190 -6 250

! *********************  DEFINE SHAPES  **********************

! Pads
[UNIVERSE][SHAPE=cylsegment pad_ne]
angle 0.06944444
inner_radius 90.832
outer_radius 94.094
arc 0.08888889
extents -6.0 240.0
origin 0 0 0

[UNIVERSE][SHAPE=cylsegment pad_nw]
angle 0.33472222
inner_radius 90.832
outer_radius 94.094
arc 0.08888889
extents -6.0 240.0
origin 0 0 0

! Pads
[UNIVERSE][SHAPE=cylsegment pad_ne]
angle 0.06944444
inner_radius 90.832
outer_radius 94.094
arc 0.08888889
extents -6.0 240.0
origin 0 0 0

[UNIVERSE][SHAPE=cylsegment pad_nw]
angle 0.33472222
inner_radius 90.832
outer_radius 94.094
arc 0.08888889
extents -6.0 240.0
origin 0 0 0
```

```
! RPV
[UNIVERSE][SHAPE=cyl inner_vessel]
radius 102.53
extents -6.0 240.0
origin 0 0 0
axis z

[UNIVERSE][SHAPE=cyl outer_vessel]
radius 112.79
extents -6.0 240.0
origin 0 0 0
axis z

! Shield
[UNIVERSE][SHAPE=cyl shield_id]
radius 120.4
extents -6.0 240.0
origin 0 0 0
axis z

[UNIVERSE][SHAPE=cyl shield_od]
radius 186.67
extents -6.0 240.0
origin 0 0 0
axis z

! *****************  CREATE HOLES AND FILL WITH UNIVERSES  *******************

! The translate card corresponds to the outer radius of the barrel.
! The translation must be performed so that origins of the VERA geometry and
! the Shift excore geometry are aligned.

[UNIVERSE][HOLE thecore]
fill reactor
translate -90.38 -90.38  0.0

[UNIVERSE][HOLE detector_cutout]
fill det_cutout

! *****************************  DEFINE CELLS  *******************************

[UNIVERSE][CELL moderator]
comp mod
shapes +thecore  -rpv_liner  +pad_ne  +pad_nw  +pad_sw  +pad_se

[UNIVERSE][CELL pad_ne]
comp ss
shapes -pad_ne

[UNIVERSE][CELL pad_nw]
comp ss
shapes -pad_nw

[UNIVERSE][CELL pad_sw]
comp ss
shapes -pad_sw
```

```
[UNIVERSE][CELL pad_se]
comp ss
shapes -pad_se

[UNIVERSE][CELL rpv_liner]
comp ss
shapes +rpv_liner -inner_vessel

[UNIVERSE][CELL rpv]
comp cs
shapes +inner_vessel -outer_vessel

[UNIVERSE][CELL void_reg1]
comp void
shapes +outer_vessel -shield_id

[UNIVERSE][CELL bioshield]
comp concrete
shapes +shield_id -shield_od +detector_cutout

[UNIVERSE][CELL void_reg2]
comp void
shapes +thecore   +outer_barrel +rpv_liner +inner_vessel +outer_vessel
       +shield_id +shield_od    -box

! ===========================================================
!                   DEFINE THE COMPOSITIONS
! ===========================================================
[COMP]

[COMP][MATERIAL void]
matid 0
temperature 293
zaid --
nd   --

! from page 114 of CASL progression problems (CASL-U-2012-0131-004)
[COMP][MATERIAL ss]
matid 1
temperature 293
zaid  6000 14028 14029 14030 15031 24050 24052 24053 24054 25055
     26054 26056 26057 26058 28058 28060 28061 28062 28064
nd   3.20895E-04 1.58197E-03 8.03653E-05 5.30394E-05 6.99938E-05
     7.64915E-04 1.47506E-02 1.67260E-03 4.16346E-04 1.75387E-03
     3.44776E-03 5.41225E-02 1.24992E-03 1.66342E-04 5.30854E-03
     2.04484E-03 8.88879E-05 2.83413E-04 7.21770E-05

! from page 114 of CASL progression problems (CASL-U-2012-0131-004)
[COMP][MATERIAL mod]
matid 2
temperature 293
zaid 1001 8016 5010 5011
nd   6.667279121238251E-02 3.333639560619125E-02 3.325516211867062E-05
     1.338562053118350E-04
```

```
! from page 114 of CASL progression problems (CASL-U-2012-0131-004)
[COMP][MATERIAL cs]
matid 3
temperature 293
zaid 6000 26054 26056 26057 26058
nd   3.93598E-03 4.89841E-03 7.68945E-02 1.77583E-03 2.36330E-04

! Concrete, Oak Ridge (ORNL), 2.3 g/cm3 material data from PNNL 15870 Rev 1
! The above density is estimated to be accurate to 2 significant digits.
! Uncertainties are not addressed. The following data were calculated from the
! input weight and converted to number densities.

[COMP][MATERIAL concrete]
matid 4
temperature 293
zaid  1001  1002  6012  6013  8016  8017  11023 12024 12025 12026
      13027 14028 14029 14030 19039 19040 19041 20040 20042 20043
      20044 20048 26054 26056 26057 26058
nd    8.49996E-03 1.70019E-06 1.99790E-02 2.24257E-04 3.54247E-02
      8.52237E-05 1.63354E-05 1.46966E-03 1.86056E-04 2.04848E-04
      5.55961E-04 1.56827E-03 7.95785E-05 5.25422E-05 3.76029E-05
      4.03217E-09 2.71356E-06 1.07627E-02 7.21657E-05 1.55434E-05
      2.32041E-04 2.10946E-05 1.12950E-05 1.77148E-04 4.09321E-06
      5.40617E-07

!############################## RESPONSES ##################################

[RESPONSE=xs]
name          U235resp
mt            N_FISSION
nuclide       92235
physics       ce
particle_type n
tmp           293

! ============================================================
!                      DEFINE THE TALLIES
! ============================================================
[TALLY]

[TALLY][CELL flux_tally]
reactions flux
cells detector_top:detector_bot

[TALLY][CELL detTop_tally]
cells detector_top
responses U235resp

[TALLY][CELL detBot_tally]
cells detector_bot
responses U235resp
```

5.4 AUTOMATED GENERATION OF THE BIOSHIELD AND EX-CORE DETECTORS

VERA can generate an automated detector input, which is in an Omnibus ex-core file format (`*.omn`), so that the user does not have to manually create an ex-core input file from scratch. This is a good way to start learning how to generate an ex-core file for a user unfamiliar with how to set up an Omnibus GG input.

The `[CORE]` block in the VERA input needs some additional user input regarding the bioshield and the detector (see Listing 7). Let us assume in this example that the height of the active core region (from the top of the bottom core plate to the bottom of the upper core plate) is 240 cm. Therefore, the mid-plane of the active core region is at z=120 cm. A bioshield is first set up, and it extends radially from 120.0 cm to 250 cm. Following this definition, there are four types of detectors that are set up in six different locations.

Listing 7. VERA input file for automated detector generation.

```
vessel   mod   87.71    ! barrel IR (cm)
          ss   90.38    ! barrel OR (cm)
         mod  102.27    ! vessel liner IR (cm)
          ss  102.53    ! vessel liner OR / vessel IR (cm)
          cs  112.79    ! vessel OR (cm)

bioshield void  120.0   ! void region between vessel and inner radius of bioshield
      concrete  250.0   ! outer radius of bioshield

!    ID   type       radii     / mats   / heights     / response_type [well_type]
! ---------------------------------------------------------------------------------
  det   PWR1  power    2.0 3.0 / void ss/ 70.0 70.0 / b10 wedge
  det   PWR2  power    2.0 3.0 / void ss/ 70.0 70.0 / b10 none
  det   SRC1  source   1.5 3.0 / void ss/ 50.0      / u235 wedge
  det   SRC2  source   1.5 3.0 / void ss/ 70.0      / u235 none

!             ID,  radius, degree, elevation
! ---------------------------------------------
  det_locations  PWR1   123.0   315  50.0
                 PWR2   116.39   45  50.0
                 PWR2   116.39    0  50.0
                 SRC1   123.0   135  95.0
                 SRC1   123.0   270  95.0
                 SRC2   123.0   225  85.0
```

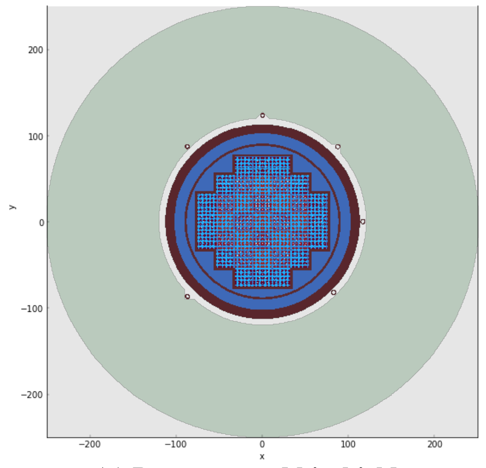
The four types of detectors that are set up are addressed first. The user must first up a detector using the parameter `det`, and then the user must provide the detector ID, such as `PWR1`. The detector ID can be any string that the user would like to specify. The detector type is defined after the detector ID. The two types of detectors are `power` for power range detectors and `source` for source range detectors. Next, the user has to enter the radius of the detector followed by the radius of the metal casing around the detector. Note that the user can enter as many concentric rings within the detector as needed. For example, the user can specify an inner void region followed by an aluminum casing, and then another void region and an outer steel casing. Once the radii have been defined, the user can specify the materials in each region. In this case, the inner region is `void`, and the outer region is `ss`. The default compositions available for the automated detector set up are provided in the ex-core manual Pandya et al. [2020], and is also provided in Listing 8 for completeness.

Listing 8. Default materials in automatically generated ex-core file.

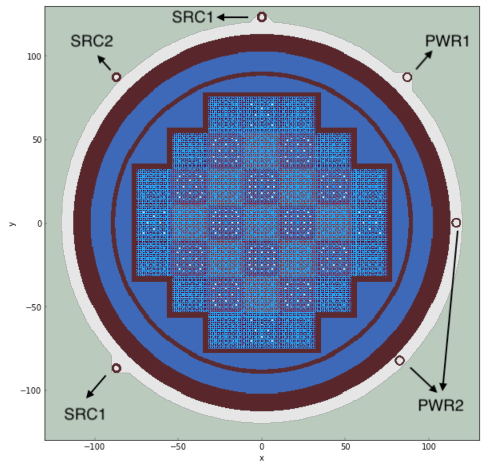
```
[COMP]

[COMP][MATERIAL void]
matid 0
temperature 293
zaid --
nd   --

! from page 114 of CASL progression problems (CASL-U-2012-0131-004)
[COMP][MATERIAL ss]
matid 1
temperature 293
zaid  6000 14028 14029 14030 15031 24050 24052 24053 24054 25055
     26054 26056 26057 26058 28058 28060 28061 28062 28064
nd   3.20895E-04 1.58197E-03 8.03653E-05 5.30394E-05 6.99938E-05
     7.64915E-04 1.47506E-02 1.67260E-03 4.16346E-04 1.75387E-03
     3.44776E-03 5.41225E-02 1.24992E-03 1.66342E-04 5.30854E-03
     2.04484E-03 8.88879E-05 2.83413E-04 7.21770E-05

! from page 114 of CASL progression problems (CASL-U-2012-0131-004)
[COMP][MATERIAL mod]
matid 2
temperature 293
zaid 1001 8016 5010 5011
nd   6.667279121238251E-02  3.333639560619125E-02 3.325516211867062E-05
     1.338562053118350E-04

! from page 114 of CASL progression problems (CASL-U-2012-0131-004)
[COMP][MATERIAL cs]
matid 3
temperature 293
zaid 6000 26054 26056 26057 26058
nd   3.93598E-03 4.89841E-03 7.68945E-02 1.77583E-03 2.36330E-04

! Concrete, Oak Ridge (ORNL), 2.3 g/cm3 material data from PNNL 15870 Rev 1
! The density specified above is estimated to be accurate to 2 significant digits.
! Uncertainties are not addressed. The following data were calculated
! from the input weight and converted to number densities.
[COMP][MATERIAL concrete]
matid 4
temperature 293
zaid  1001  1002  6012  6013  8016  8017  11023 12024 12025 12026
      13027 14028 14029 14030 19039 19040 19041 20040 20042 20043
      20044 20048 26054 26056 26057 26058
nd    8.49996E-03 1.70019E-06 1.99790E-02 2.24257E-04 3.54247E-02
      8.52237E-05 1.63354E-05 1.46966E-03 1.86056E-04 2.04848E-04
      5.55961E-04 1.56827E-03 7.95785E-05 5.25422E-05 3.76029E-05
      4.03217E-09 2.71356E-06 1.07627E-02 7.21657E-05 1.55434E-05
      2.32041E-04 2.10946E-05 1.12950E-05 1.77148E-04  4.09321E-06
      5.40617E-07

! Aluminum
[COMP][MATERIAL al]
matid 5
temperature 293
zaid 13027
nd   6.02626E-02
```

After the material has been defined, the user must specify the height of the detector. The height can be divided into two axial regions with top and bottom halves, as in the case of PWR1 and PWR2, or the detector can have one axial region, as in the case of SRC1 and SRC2. After the detector height is specified, the response type must be specified. The detector can be set up with absorption reactions in ^{10}B or fission reactions in ^{235}U. Finally, the detector well type is specified. A wedge-shaped well can be specified, or the detector can be defined without a well.

Then the detector locations for each of the detector types are provided with radius and degrees. Note that when defining the degrees, the user must bear in mind that 0 to 90° refers to the southeast quadrant, 90 to 180° is the southwest quadrant, 180 to 270° is the northwest quadrant, and 270 to 360° refers to the northeast quadrant. The elevation of the detector refers to the distance from z=0 cm, which is the top of the bottom core plate to the bottom of the detector if there is only one axial region in the detector. Or, if there are two axial regions, then it refers to the distance from the top of the bottom core plate to the bottom of the detector located in the lowest axial region. In this example, the mid-plane of the core is at z=120 cm, so for the PWR1 and PWR2 detectors, the elevation is 50 cm (120 minus 70 cm). The elevation for SRC1 is 95 cm (120 minus 25 cm), and the elevation for SRC2 is 85 cm (120 minus 35 cm), which makes them axisymmetric about the core mid-plane. A plot of the detectors is shown in Figure 8.



(a) Detectors and bioshield



(b) Close-up view of the detectors

Figure 8. Automatically generated detectors and bioshield.

The next step is to ensure that in the [SHIFT] block of the VERA input file, the parameter vera_pressure_vessel is set to true, and the core_translate parameter corresponds to the outer radius of the reactor pressure vessel and is set to the following:

```
    vera_pressure_vessel true
    core_translate       -outer_radius_of_vessel  -outer_radius_of_vessel  0.0
```

More details regarding the set up of the [SHIFT] block are provided in Section 5.5. Once these parameters have been added to the Shift block, if the user does not intend to change the default materials or their temperatures, then VERA can be run using verarun. However, if the user would like to update the materials and temperatures, then there are two options: (1) submit a job using verarun vera_input_file.inp command and then immediately kill the job that was just submitted using the command qdel jobid, or (2) run the react2xml executable by typing the following into the terminal:

```
    /PATH/TO/VERA/INSTALL/bin/react2xml  vera_input_filename.inp
```

An ex-core `vera_input_filename.omn` file will be output to the working directory. The user can then scroll down to the `[COMP]` block to make any changes to the material, number densities, or the temperatures for the appropriate detectors. The detector ID is provided in the comments within the input to make it is easier for the user to identify the cells associated with a detector. By scrolling down to the `[TALLY]` block, the user will see a list of tallies associated with the detectors set up in the VERA input. For this example, see Listing 9 for the tallies automatically generated in the ex-core input file.

Listing 9. `[TALLY]` block automatically generated in the ex-core file.

```
[TALLY]

[TALLY][CELL detector_0_0]
cells detector_0_0
responses b10

[TALLY][CELL detector_0_2]
cells detector_0_2
responses b10

[TALLY][CELL detector_1_0]
cells detector_1_0
responses b10

[TALLY][CELL detector_1_2]
cells detector_1_2
responses b10

[TALLY][CELL detector_2_0]
cells detector_2_0
responses b10

[TALLY][CELL detector_2_2]
cells detector_2_2
responses b10

[TALLY][CELL detector_3_0]
cells detector_3_0
responses u235

[TALLY][CELL detector_4_0]
cells detector_4_0
responses u235

[TALLY][CELL detector_5_0]
cells detector_5_0
responses u235
```

`detector_0_0` and `detector_0_2` are associated with the lower and upper detector regions of `PWR1`. `detector_1_0` and `detector_1_2` are associated with the lower and upper detector regions of `PWR2` located at a radius of 116.39 cm and 45°. `detector_1_0` and `detector_1_2` are associated with the lower and upper detector regions of `PWR2` located at a radius of 116.39 cm and 0°. Next, `detector_3_0` and detector_4_0 are associated with SRC1 detectors located at a 123 cm radius and at 135°and 270°, respectively. Finally, detector_5_0 is associated with the SRC2 detector located at a 123 cm radius and at 225 °.

The next step can be prone to error, so it is highly recommended that this modified *.omn file be saved in a separate folder so that it is not accidentally overwritten. Once a backup of this ex-core file has been created, the next step is to run verarun again by typing in the input file name but this time with the *.xml extension.

```
verarun vera_input_filename.xml
```

This will bypass the react2xml execution, and a new *.omn file with the default materials will not be produced and override the modified *.omn file with the updated materials. After the execution of the *.xml file, open the *.omn file and make sure that the density and temperature changes that were made show up accurately and were not accidentally overwritten. The user can also check *.excore.xml to ensure that the correct compositions and temperatures were indeed used in the calculation.

5.5 RUNNING VERA WITH AN EX-CORE FILE

Once the ex-core file has been set up correctly, the [SHIFT] block in the VERA input must be set up correctly to account for the additional ex-core file. See Listing 10 for an example.

Listing 10. [SHIFT] block in VERA input for running ex-core calculations with an ex-core file.

```
[SHIFT]
  Np                        50e6
  hybrid_tally_names        flux_tally
  hybrid_multiplier_names flux
  output_adjoint            true
  num_blocks_i              10
  num_blocks_j              10
  !
  ! Name of the file produced from running excore2xml
  excore_filename           smr-detwell.excore.xml
  !
  vera_pressure_vessel    true
  !
  ! Same translation of the core from your ex-core file
  core_translate          -112.79 -112.79 0.0
  bc_bnd_mesh             reflect vacuum reflect vacuum vacuum vacuum
  x_bnd_mesh              0.0 190
  y_bnd_mesh              0.0 190
  z_bnd_mesh              -6.0 250
  raytrace_levels         200.0
```

Np represents the total number of particles to be used in the Shift fixed source calculation. hybrid_tally_names represents the name of the tally in the ex-core file that the user would like to optimize in the hybrid calculation. If the user is interested in optimizing for the flux_tally shown in Section 5.3, then that needs to be specified using the hybrid_tally_names. Note that the user can optimize for multiple cell tallies. The hybrid_mulitplier_names should be set to flux since this is the only option currently available to the user because Shift integrated in VERA cannot currently optimize for other responses such as dose with CADIS. It can only optimize for flux.

If the user would like to plot the adjoint, then the `output_adjoint` option should be set to `true`. The user can also change the default setting of `num_blocks_i` and `num_blocks_j` to lower or higher numbers so that the Shift transport calculation runs on a lower or higher number of processors. Note that `num_blocks_i` multiplied by `num_blocks_j` is the total number of processors used for the Shift calculation. Next, the user must specify the name of the ex-core file using the `excore_filename` parameter and append it with the `*.excore.xml` extension. This definition is necessary because VERA will run a script that converts the ex-core file to an `xml` format which is then read by Shift to perform transport on the ex-core geometry.

If the user is integrating all the geometry up to the reactor pressure vessel's outer radius from the VERA input, with the remainder of the geometry outside the pressure vessel defined in the ex-core file, then the user must set the `vera_pressure_vessel` parameter to `true`. This is also a requirement for cases run with the automated generation of the detector and bioshield. However, if the user is only integrating the geometry up to the outer radius of the barrel from the VERA input and then defining all the geometry outside the barrel using the ex-core file, then `vera_pressure_vessel` must be set to `false`.

The user must also align the origin in the VERA input with the origin in the ex-core file. Note that this was done previously in the ex-core file using the `translate` card when integrating the VERA geometry using the `HOLE`. It must be defined again here in the `[SHIFT]` block, and the translation remains the same. The translation in the VERA input is provided using the `core_translate` parameter. This is another potential user pitfall.

Next, the boundary conditions must be specified. The `bc_bnd_mesh` parameter allows the user to specify the boundary conditions along the `-x +x -y +y -z +z` boundaries, in that order. In this example, the boundary conditions are provided for a quarter core problem, so the boundary conditions are `reflect vacuum reflect vacuum vacuum vacuum`. Next, the problem boundaries along x, y and z are provided using the `x_bnd_mesh`, `y_bnd_mesh`, and `z_bnd_mesh` parameters. The user may provide the same boundaries that were used for boundary `box` in the ex-core file here. Note that the user must define the full geometry in the ex-core file even if they are running a quarter core geometry in VERA. Shift performs the fixed-source transport on the full geometry. Finally, the user can specify a z-plane along which a raytrace will be performed for plotting purposes using the `raytrace_levels` parameter. This will allow the user to produce an image of the VERA geometry integrated with the ex-core geometry.

6. PRE- AND POST-PROCESSING INPUTS AND OUTPUTS

When setting up a geometry, the user should check to ensure that there are no geometry errors. A python script has been set up for this purpose so that Omnibus modules associated with the geometry raytracing can be loaded and then used to produce plots to check for errors.

6.1 CHECKING FOR GEOMETRY ERRORS DURING EX-CORE INPUT SETUP

It is important for analysts to view the geometry and check for any geometry errors while setting up the ex-core geometry. Unfortunately, the only way to do this currently is to run a python script that generates plots at different planes.

To run the script, a few changes must be made to the ex-core `*.omn` file from that discussed in previous sections. Listing 11 provides an example showing how to set up an ex-core file for geometry visualization and error checking. First, the `[TALLY]` and `[COMP]` blocks at the bottom of the input files must be commented out or removed, and then the user must specify a `comp:matid` list at the beginning of the ex-core file for each composition specified in the ex-core file (see an example below). Lines that are to be commented out, modified, or added are highlighted in yellow. The first example shown of a simple SMR ex-core file in Section 5.2 is modified here so that plots of the geometry can be obtained to check for any errors before running it with VERA.

Next, the `[UNIVERSE=core reactor]` block must be commented out, and in its place, a new `[UNIVERSE=general reactor]` must be defined. This step is necessary because the user is must generate the geometry using the python script before running VERA. Therefore, the code does not have the information required to plot the VERA geometry. In its place, a "fake core" is modeled which models a cylinder up to the outer radius of the barrel or the outer radius of the reactor pressure vessel, depending on the boundary up to which the VERA geometry is integrated. Finally, the user must remove or comment out only the translate card for the `[HOLE thecore]` since this translate card is only aligning the VERA geometry origin with the Shift ex-core origin.

Listing 11. Revised ex-core file for geometry visualization and error-checking.

```
! ============================================================================
!                           SIMPLE SMR EX-CORE REGION
! ============================================================================

[GEOMETRY]
global "excore"

  ! This replaces the [COMP] block that needs to be commented out or deleted
  comp : matid
     fake_core  0
     void       1
     concrete   2

  ! Reactor core from VERA input inserted into this universe needs to be
  ! commented out
  ! [UNIVERSE=core reactor]

  ! ============================================================
  !   UNIVERSE: REACTOR - FAKE CORE DEFINITION FOR RAYTRACE
  ! ============================================================

  [UNIVERSE=general reactor]
  boundary "rpv_outer "

  ! Reactor Pressure Vessel Outer Radius
  [UNIVERSE][SHAPE=cyl rpv_outer]
  radius 112.79
  extents -6.0 240.0
  origin 0 0 0
  axis z

  [UNIVERSE][CELL fake_core]
  comp fake_core
  shapes -rpv_outer

! ------------------------- UNIVERSE: EX-CORE ------------------------------

! ****************************************************************************
! Define the problem boundary
[UNIVERSE=general excore]
boundary "box"

! boundary box
[UNIVERSE][SHAPE=cuboid box]
faces -190 190 -190 190 -6 250

! ****************************************************************************
! Define the shapes (or surfaces in MCNP/KENO terminology)
! Vessel radius is 112.79, and all the surfaces outside the vessel are defined

! Shield
[UNIVERSE][SHAPE=cyl shield_id]
radius  120.4
extents  -6.0 240.0
origin  0 0 0
axis    z

[UNIVERSE][SHAPE=cyl shield_od]
radius  186.67
extents  -6.0  240.0
origin  0 0 0
axis    z

  ! ****************************************************************************
  ! This HOLE is now filled with the new universe defined for the FAKE CORE.
  ! There is no need to translate the origin since VERA is not being run
  ! while running the check for geometry errors.
  ! ****************************************************************************

[UNIVERSE][HOLE thecore]
fill reactor
  ! translate -112.79 -112.79 0.0

! Define cells
```

```
[UNIVERSE][CELL void_reg1]
comp void
shapes +thecore -shield_id

[UNIVERSE][CELL bioshield]
comp concrete
shapes +thecore +shield_id -shield_od


[UNIVERSE][CELL void_reg2]
comp void
shapes +thecore +shield_id +shield_od -box

 ! Comment out or delete [COMP] and [TALLY] blocks below
```

The next step is to run the `*.omn` excore file using the `gg2xml` executable, `/PATH/TO/VERA/INSTALL/bin/gg2xml`, located in the bin folder with the VERA build.

```
    /PATH/TO/VERA/INSTALL/bin/gg2xml excore_filename.omn
```

This will generate an `excore_filename.gg.xml` file. Note that at this step, if there are any input errors that `gg2xml` can catch, they will be printed to the terminal. The user can read the errors on the terminal and fix the input as necessary. Instead of typing in the entire path to point to the `gg2xml` executable, the user can create a soft link by typing the following command into the terminal window:

```
    ln -s /PATH/TO/VERA/INSTALL/bin/gg2xml
```

This will create a soft link to the `gg2xml` executable which can then be invoked by typing the following command into the terminal window:

```
    ./gg2xml excore_filename.omn
```

Once the user has corrected all the input errors and the `*.gg.xml` file has been successfully generated, the file can be used as an input to the python script, `plot.py`, which is used to generate the plots to view any further geometry errors (see Listing 12). Note that the user must provide the correct `omnibus_gg_xml_filename` at the top of the python script. Unfortunately, `gg2xml` will not catch all the input errors in the input parsing step. Once the python script is run by typing the command below into the terminal window, any additional geometry error messages will appear.

```
    python plot.py
```

In addition to the error messages, a plot called `plot1.png` will be output in the working directory. Users can modify this python script as they see fit.

All parts of the input that the user needs to define or for which the user needs to provide input are highlighted in yellow. The `lower` parameter requires the user to provide the lower bounds of the (x, y, z) axes for plotting purposes, and the `upper` parameter requires the user to provide the respective upper bounds. This area will be raytraced by Omnibus to check for geometry errors. The sum of all the values in the basis must add up to 1. If the user would like to generate a plot along a 45° angle, then the basis should be `(0.707107, 0.707107, 0)`.

Listing 12. Python script for geometry error-checking.

```
# plot.py

import os
import shutil
import h5py as h5
import matplotlib.pyplot as plt
plt.switch_backend('agg')

from omnibus.raytrace.h5imager import Imager as H5Imager
from omnibus.raytrace.imager import Imager
from omnibus.formats.comp import load as load_comps
from omnibus.raytrace.colors import ColorMap
from omnibus.data import plot
from omnibus.raytrace.load import load_gg
from geometria import GG_Geometry

# ---------------------------------------------------------------------------
#                            LOAD *.GG.XML FILE
# ---------------------------------------------------------------------------
# USER TO SPECIFY THE INPUT BELOW

omnibus_gg_xml_filename =  'excore_filename.gg.xml'

# ---------------------------------------------------------------------------
#                          PLOT ENTIRE EXCORE GEOMETRY
# ---------------------------------------------------------------------------

# Load the *.gg.xml file
model = load_gg(omnibus_gg_xml_filename)

# Look in the terminal window for more messages on the geometry error

imager = Imager(model.geometry,
                lower=(-400, -400, 515),
                upper=(400, 400, 515),
                basis=(1,0,0),
                max_pixels=1024)
imager.trace = 'cell' # This can be set to âĂŸmatâĂŹ if you want to see materials
imager.plot()
imager.check_errors=True
fig = imager.plot(figsize=(20,20))

plt.savefig('plot1.png', format='png', dpi=600)
```

An example of the plot output from `plot.py` is shown in Figure 9.

Finding all the geometry errors is an iterative process. Once the user is confident that the input is correct, the input that was added to the `*.omn` file to generate the `*.gg.xml` file can be commented out or deleted, and the `[COMP]` and `[TALLY]` blocks can be added so that it will run through VERA. Note that there may be outstanding geometry errors that will not be caught until the input is run through VERA with the VERA core model integrated. However, most geometry errors should be found at this step with the `plot.py` script. Note that it is necessary for the user to plot along different `basis`, `upper` and `lower` bounds so that Omnibus will raytrace along multiple planes and directions to find geometry errors.

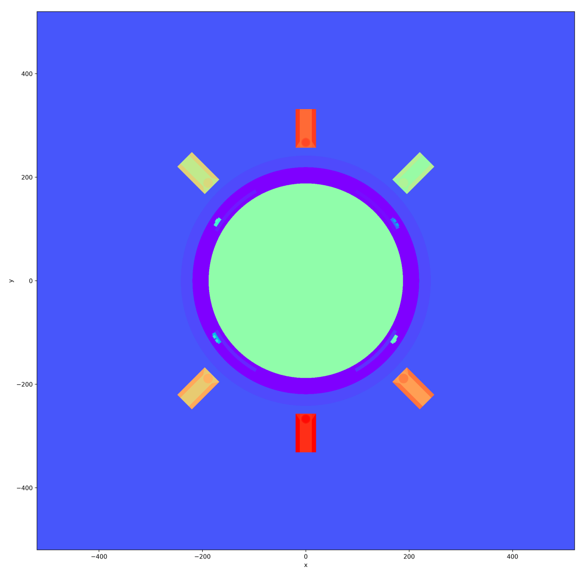


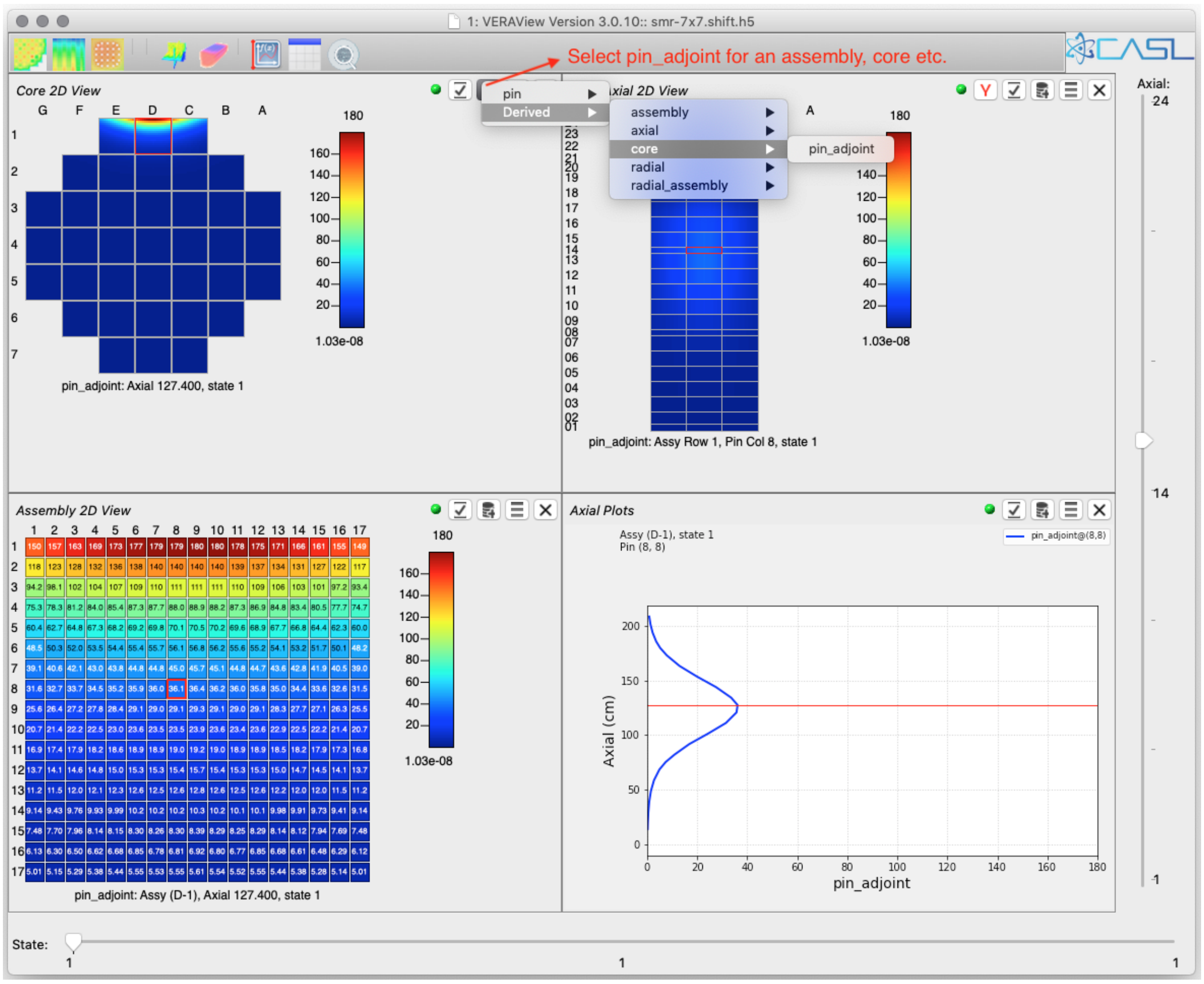
Figure 9. An example of a plot output from plot.py.

6.2 GENERATING PLOTS AFTER RUNNING VERA

There are different methods to extract plots and analyze data after running VERA. These tools are discussed further in the following subsections.

6.2.1 Using VERAView To Examine Pin Adjoints

In addition to viewing vessel fluence, VERAView can be used to view the pin adjoints. These pin adjoints indicate which pins contribute the most to a vessel or detector response. Figure 10 shows a VERAView screen shot of the pin adjoints for an ex-core detector. The first step is to load the Shift HDF5 output file, `vera_output_filename.shift.h5`. Note that the user is not opening the VERA output file, `vera_output_filename.h5`. VERAView will load a 2D core view of the pin adjoints, an axial view as well as the assembly view. The user can click on a specific assembly to view the pin adjoints in greater detail. Shift currently only performs the adjoint calculation at the first statepoint and uses the same adjoint to perform advanced variance reduction throughout the rest of the calculation for subsequent states. The user can slide the `Axial` bar on the right-hand side to slide between different axial planes to view the adjoint.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	150	157	163	169	173	177	179	179	180	180	178	175	171	166	161	155	149
2	118	123	128	132	136	138	140	140	140	140	139	137	134	131	127	122	117
3	94.2	98.1	102	104	107	109	110	111	111	111	110	109	106	103	101	97.2	93.4
4	75.3	78.3	81.2	84.0	85.4	87.3	87.7	88.0	88.9	88.2	87.3	86.9	84.8	83.4	80.5	77.7	74.7
5	60.4	62.7	64.8	67.3	68.2	69.2	69.8	70.1	70.5	70.2	69.6	68.9	67.7	66.8	64.4	62.3	60.0
6	48.5	50.3	52.0	53.5	54.4	55.4	55.7	56.1	56.8	56.2	55.6	55.2	54.1	53.2	51.7	50.1	48.2
7	39.1	40.6	42.1	43.0	43.8	44.8	44.8	45.0	45.7	45.1	44.8	44.7	43.6	42.8	41.9	40.5	39.0
8	31.6	32.7	33.7	34.5	35.2	35.9	36.0	36.1	36.4	36.2	36.0	35.8	35.0	34.4	33.6	32.6	31.5
9	25.6	26.4	27.2	27.8	28.4	29.1	29.0	29.1	29.3	29.1	29.0	29.1	28.3	27.7	27.1	26.3	25.5
10	20.7	21.4	22.2	22.5	23.0	23.6	23.5	23.5	23.9	23.6	23.4	23.6	22.9	22.5	22.2	21.4	20.7
11	16.9	17.4	17.9	18.2	18.6	18.9	18.9	19.0	19.2	19.0	18.9	18.9	18.5	18.2	17.9	17.3	16.8
12	13.7	14.1	14.6	14.8	15.0	15.3	15.3	15.4	15.7	15.4	15.3	15.3	15.0	14.7	14.5	14.1	13.7
13	11.2	11.5	12.0	12.1	12.3	12.6	12.5	12.6	12.8	12.6	12.5	12.6	12.2	12.0	12.0	11.5	11.2
14	9.14	9.43	9.76	9.93	9.99	10.2	10.2	10.2	10.3	10.2	10.1	10.1	9.98	9.91	9.73	9.41	9.14
15	7.48	7.70	7.96	8.14	8.15	8.30	8.26	8.30	8.39	8.29	8.25	8.29	8.14	8.12	7.94	7.69	7.48
16	6.13	6.30	6.50	6.62	6.68	6.85	6.78	6.81	6.92	6.80	6.77	6.85	6.68	6.61	6.48	6.29	6.12
17	5.01	5.15	5.29	5.38	5.44	5.55	5.53	5.55	5.61	5.54	5.52	5.55	5.44	5.38	5.28	5.14	5.01

pin_adjoint: Assy (D-1), Axial 127.400, state 1

Figure 10. Pin adjoint plots generated with VERAView.

6.2.2 Custom Python Scripts

Once VERA finishes running successfully with the ex-core file, the user can use the scripts `input.py` and `plt_adj.py` to view the adjoint flux. These scripts provide a basis to the user that can be expanded upon to include additional plotting features.

In the `input.py` script (see Listing 13), the user must provide the name of the VERA input file without any extensions. Then the user can plot and view the adjoint flux at various locations and provide the `zlevel`, group number, and the plot file name using the `plt_adj.plot_adjoint_flux function`. The `imager` and `shift_file` should be left as it is in the python script and should not be changed. The `zlevel` in the example below is set to the same level as the `raytrace_levels` parameter in the `[SHIFT]` VERA input. This allows the user to overlay the adjoint flux over the VERA geometry using the python script. Note that the user can overlay the adjoint flux over the geometry of any axial plane (`raytrace_levels`) but they need to bear in mind that the geometry being overlaid may not capture the features of the axial plane that the adjoint flux represents. To view the total adjoint flux, call the `plt_adj.plot_tot_adj_flux` function in the `input.py` script, as shown below. The `plt_adj.py` script contains the functions that process the parameters specified in the `input.py` file. The user must ensure that both the `plt_adj.py` and `input.py` python scripts are in the same working directory as the VERA output. Next, the user can run the set of scripts by typing the following in the terminal:

```
    python input.py
```

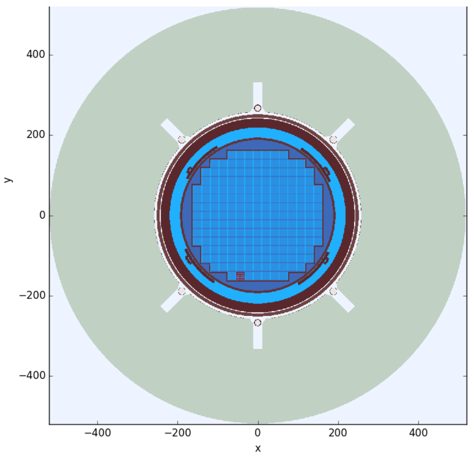
Once again, the user is encouraged to change and customize this script to suit their application. These scripts serve as an example and form a basis for any python scripts the user would like use to produce plots. The `plt_adj.py` script is not shown here, but it is available with the manual.

Listing 13. Python script for viewing the geometry and adjoint after running VERA.

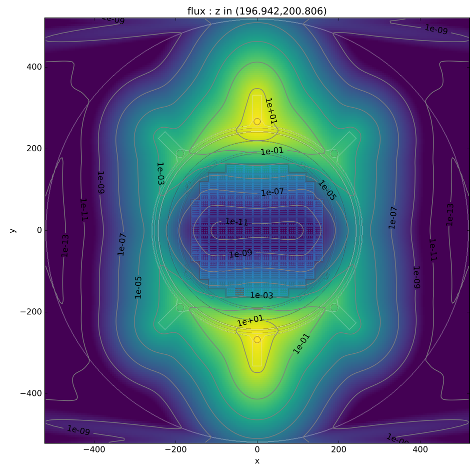
```
# input.py

import os
import h5py as h5
import matplotlib.pyplot as plt
plt.switch_backend('agg')

from omnibus.raytrace.h5imager import Imager as H5Imager
from omnibus.formats.comp import load as load_comps
from omnibus.raytrace.colors import ColorMap
from omnibus.data import plot

import plt_adj

# ----------------------------------------------------------------------------
# USER TO SPECIFY THE INPUT NAME BELOW (WITHOUT THE *.SHIFT.H5 EXTENSION)

vera_input_filename = 'boc'

# ----------------------------------------------------------------------------

input_filename      = plt_adj.input_file(vera_input_filename)
shift_file          = plt_adj.load_shift_hdf5(vera_input_filename)
imager              = plt_adj.load_imager(vera_input_filename)

# Plot adjoint flux at various levels and groups (0 to 7)
# plot_adjoint_flux(zlevel, group, filename)
plt_adj.plot_adjoint_flux(200, 0, imager, shift_file, 'adjFlux_g0')
plt_adj.plot_adjoint_flux(200, 1, imager, shift_file, 'adjFlux_g1')
plt_adj.plot_adjoint_flux(200, 2, imager, shift_file, 'adjFlux_g2')
plt_adj.plot_adjoint_flux(200, 3, imager, shift_file, 'adjFlux_g3')
plt_adj.plot_adjoint_flux(200, 4, imager, shift_file, 'adjFlux_g4')
plt_adj.plot_adjoint_flux(200, 5, imager, shift_file, 'adjFlux_g5')
plt_adj.plot_adjoint_flux(200, 6, imager, shift_file, 'adjFlux_g6')
plt_adj.plot_adjoint_flux(200, 7, imager, shift_file, 'adjFlux_g7')

# Plot total adjoint flux
plt_adj.plot_tot_adj_flux(200, imager, shift_file, 'total_adjFlux')
```

Once the script runs successfully, a folder called `Plots` is produced. If a folder called `Plots` already exists, then it must be moved to a different location, renamed, or deleted. In the `Plots` folder, the user will see three items: (1) `output_filename.png` which shows the integrated in-core and ex-core geometry, (2) a `GroupwiseAdjointFlux` folder with plots of the groupwise adjoint flux, and (3) a `TotalAdjointFlux` folder with a plot of the total adjoint flux.

Examples of the plots produced by running `input.py` are shown in Figure 11.



(a) In-core geometry integrated with ex-core geometry.



(b) Total adjoint flux.

Figure 11. Integrated in-core and ex-core geometry with the adjoint flux overlaid on the geometry.

6.3 POST-PROCESSING CELL TALLIES AFTER RUNNING VERA

A user who is experienced in navigating through HDF5 files can look through the `*.h5` VERA output or the `*.shift.h5` output to find the ex-core tallies named in the `*.omn` file. However, this can be cumbersome for users not familiar with HDF5 files, so a python script has been created to facilitate a quick method to pull the ex-core tally data into a text file. This python script is called `collate_tally.py`, and it requires the user to type the following into the terminal window.

```
    python collate_tally.py vera_output.h5
```

This will invoke the script to process the HDF5 output files, and it will print out the tally information in a text file called `tally.txt` (see Listing 14 for an example). The details of the script are not shown in the manual, but they are available for the user with access to VERA.

Listing 14. Python script for viewing the geometry and adjoint after running VERA.

```
##########################################################################################################
                         Results for detector_tally cell tally from output_filename.h5 output file
 Note: If the cell volume is 0, then the flux reported below is volume integrated. The user must divide the flux by the cell
      volume to obtain the volume-averaged flux. If the cell volume is calculated by Shift and is printed below, then the flux is
      volume averaged.
##########################################################################################################

State  %Power  Exposure (EFPD)   Tally Flux (*)   RE (%)   Detector Response  RE (%)  Total Source Strength (n/s)   Volume (cm3)
  1     0.01        0.00         1.000000e+12      0.02      1.000000e+07      0.10          4.000000e+14               0.000000
  2     5.20        0.00         8.000000e+14      0.05      3.000000e+14      0.05          4.000000e+18               0.000000
```

The output prints out the power in percent for each state point in the calculation, the associated exposure in effective full power days (EFPDs), the tally flux and its associated relative error in percent, the detector response and its associated percent relative error, the total source strength in units of neutrons per second, and the volume of the tally in cubic centimeters. If the cell volume is zero, then the flux reported under the `Tally Flux` is volume integrated. Shift is unable to automatically calculate the cell volume because the cell definition was complex, so the user must divide the `Tally Flux` by the volume of the cell (calculated by the user) to obtain a volume-averaged flux. However, if there is a volume printed out in the output file, then Shift is able to automatically calculate the volume and can divide the cell tally by the volume to provide the user with the volume-averaged tally flux.

7. CONCLUSION

To conclude, this guide serves the purpose of helping the user set up an ex-core calculation using VERA, guiding the user through an ex-core input setup, and providing the user with directions for post-processing results after a calculation. Any feedback from users on this document or the workflow of VERA ex-core calculations is much appreciated. This feedback can be provided through VERA User's Group. The python scripts provided with this user's guide are meant to serve as the basis on which the user can build for their specific applications.

8. REFERENCES

M. N. Avramova. CTF: A Thermal Hydraulic Sub-Channel Code for LWR Transient Analyses, User's Manual. Technical report, Pennsylvania State University Department of Nuclear Engineering, 2009.

B. Collins, S. Stimpson, B. Kelley, M. Young, B. Kochunas, A. Graham, E. Larsen, T. Downar, and A. Godfrey. Stability and Accuracy of 3D Neutron Transport Simulation Using the 2D/1D Method in MPACT. *Journal of Computational Physics*, 326:612–628, 2016.

E. E. Davidson, T. M. Pandya, A. T. Godfrey, and M. Asgari. Watts Bar I Ex-Core Analyses Using VERA. In *Proceedings of the 20th Topical Meeting of the Radiation Protection Shielding Division*, Santa Fe, NM, USA, 2018. American Nuclear Society.

E. E. Davidson, T. M. Pandya, K. E. Royston, T. M. Evans, A. T. Godfrey, Shane C. Henderson, Gary Wolfram, and Joel M. Risner. Effect of Fission Source Spectrum on Monte Carlo Calculation of Ex-core Quantities. In *Proceedings of PHYSOR 2020: Transition to a Scalable Nuclear Future*, Cambridge, United Kingdom, 2020. American Nuclear Society.

S. Johnson, T. Evans, G. Davidson, S. Hamilton, T. Pandya, K. Royston, and E. Biondo. Omnibus User Manual. Technical Report ORNL/TM-2018/1073, ORNL, TBD 2020.

S. Palmtag and A. T. Godfrey. VERA Common Input User Manual. Technical Report CASL-U-2014-0014-002, CASL, February 2015.

T. Pandya, T. Evans, K. Royston, K. Clarno, B. Collins, S. Stimpson, and S. Henderson. VERAShift User's Manual. Technical Report CASL-U-2019-1921-000, CASL, January 2020.

T. M. Pandya, S. R. Johnson, T. M. Evans, G. G. Davidson, S. P. Hamilton, and A. T. Godfrey. Implementation, Capabilities, and Benchmarking of Shift, a Massively Parallel Monte Carlo Radiation Transport Code. *Journal of Computational Physics*, 308:239–272, 2016.

H. P. Smith, E. E. Davidson, A. T. Godfrey, and T. M. Pandya. An Analysis of Various Solution Strategies and Perturbations on Inputs of the Reactor Shielding Problem. In *Proceedings of the 20th Topical Meeting of the Radiation Protection Shielding Division*, Santa Fe, NM, USA, 2018. American Nuclear Society.

J. A. Turner, K. Clarno, M. Sieger, R. Bartlett, B. Collins, R. Pawlowski, R. Schmidt, and R. Summers. The Virtual Environment for Reactor Applications (VERA): Design and Architecture. *Journal of Computational Physics*, 326:544–568, December 2016.

J. C. Wagner and A. Haghighat. Automated Variance Reduction of Monte Carlo Shielding Calculations Using the Discrete Ordinates Adjoint Function. *Nuclear Science and Engineering*, 128(2):186–208, 1998.