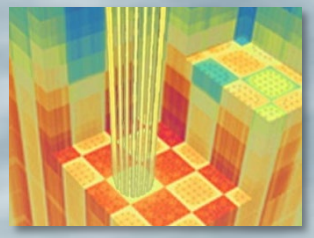
VERA Training:
Full Core Examples

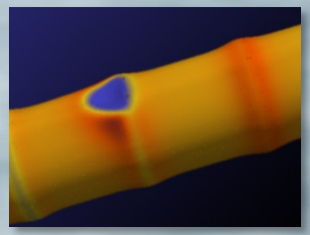
VERA Training – Core Simulator

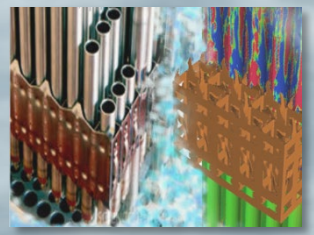
February 13, 2019

VERA Users Group Meeting

Oak Ridge National Laboratory







Training Objectives

In this module you will learn to:

- Set up full-core models
- Specify parallel decomposition
- Perform core shuffle
- Model Jump-In Cycles
- Specify Fuel Temperature Tables
- Perform Thermal Expansion

Full-Core Models

- Full-core models use the same assembly input used for the problem 3D assembly cases
- You just need to define the core geometry and place the assemblies in the right locations

Define Core Shape

```
[CORE]
  size 15
  rated 3411 128.8048    ! MW, Mlbs/hr
  apitch 21.504          ! cm
  bc_sym rot

  xlabel    R  P  N  M  L  K  J  H  G  F  E  D  C  B  A
  ylabel   01 02 03 04 05 06 07 08 09 10 11 12 13 14 15

  core_shape
    0  0  0  0  1  1  1  1  1  1  1  0  0  0  0
    0  0  1  1  1  1  1  1  1  1  1  1  1  0  0
    0  1  1  1  1  1  1  1  1  1  1  1  1  1  0
    0  1  1  1  1  1  1  1  1  1  1  1  1  1  0
    1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
    1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
    1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
    1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
    1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
    1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
    1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
    0  1  1  1  1  1  1  1  1  1  1  1  1  1  0
    0  1  1  1  1  1  1  1  1  1  1  1  1  1  0
    0  0  1  1  1  1  1  1  1  1  1  1  1  0  0
    0  0  0  0  1  1  1  1  1  1  1  0  0  0  0
```

- Need one “full” map to define the layout of the core
- Additional maps only show assembly locations
- Most maps can be octant, qtr, or full-symmetry

```
cycle 1

assm_map
  1
  2  1
  1  2  1
  2  1  2  1
  1  2  1  2  2
  2  1  2  1  2  3
  1  3  1  3  3  3
  3  3  3  3
```

Map labels correspond to labels in [ASSEMBLY] axial cards

Add Control Rod and Insert Locations

```
crd_map
   1
   -  -
   1  -  1
   -  -  -  -
   1  -  -  -  1
   -  1  -  1  -  -
   1  -  1  -  1  -
   -  -  -  -
```

Map labels correspond to labels in [CONTROL] axial cards

```
crd_bank
   D  -  A  - SE  -  C  -
   -  -  -  -  - SB  -  -
   A  -  C  -  -  -  B  -
   -  -  -  -  - SC  -  -
  SE  -  -  -  D  - SA
   - SB  - SD  -  -  -
   C  -  B  - SA  -
   -  -  -  -
```

Bank labels used to position rods in [STATE] blocks

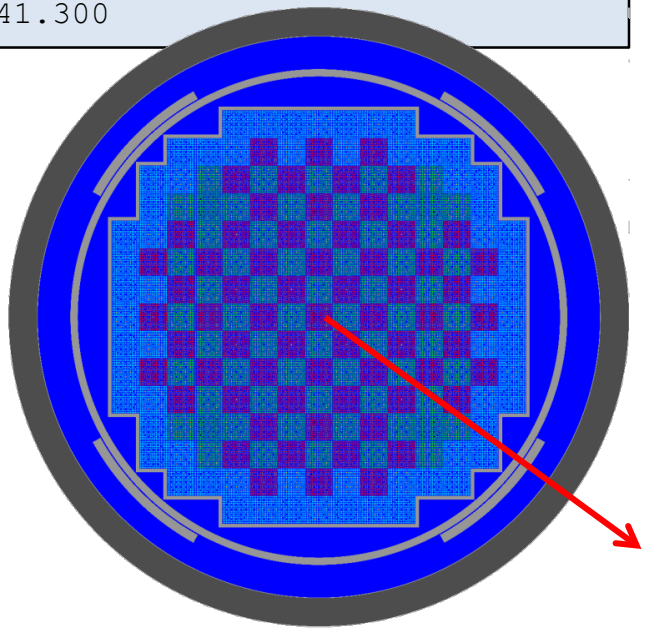
- Example maps show numbers, but they are actually user-defined strings
- Dashes "-" signify empty locations (no control rod or insert)

```
insert_map
   -  16   -  12a  -  16   -  6E
  16  -   12a  -  12a  -  20   -
   -  12a  -  12a  -  16   -  6E
 12a  -  12a  -  16   -  16   -
   -  12a  -  16   -  16   -
  16   -  16   -  16  15   -
   -  20   -  16   -   -
   -  6S   -
```

Map labels correspond to labels in [INSERT] axial cards

Add Core Geometry Data

```
height      401.03       ! cm

upper_plate ss     6.3 0.56
lower_plate ss     7.5 0.42
baffle      ss 0.1627 2.22250

pad    ss 194.84 201.63 32 45 135 225 315

vessel  mod 187.960
         ss 193.675
        mod 219.150
         ss 219.710
         cs 241.300
```



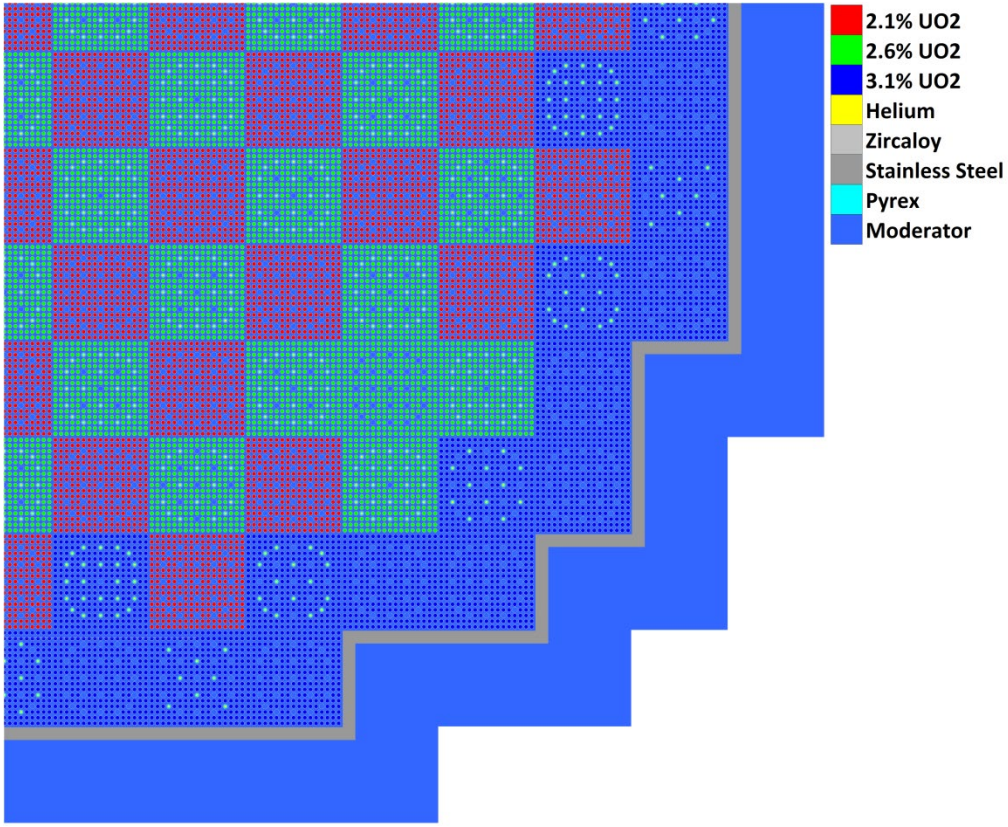
- All axial measurements are relative to the top of the lower core plate
- Height is the distance from the top of the lower core plate to the bottom of the top core plate
- Baffle material, gap size, and thickness

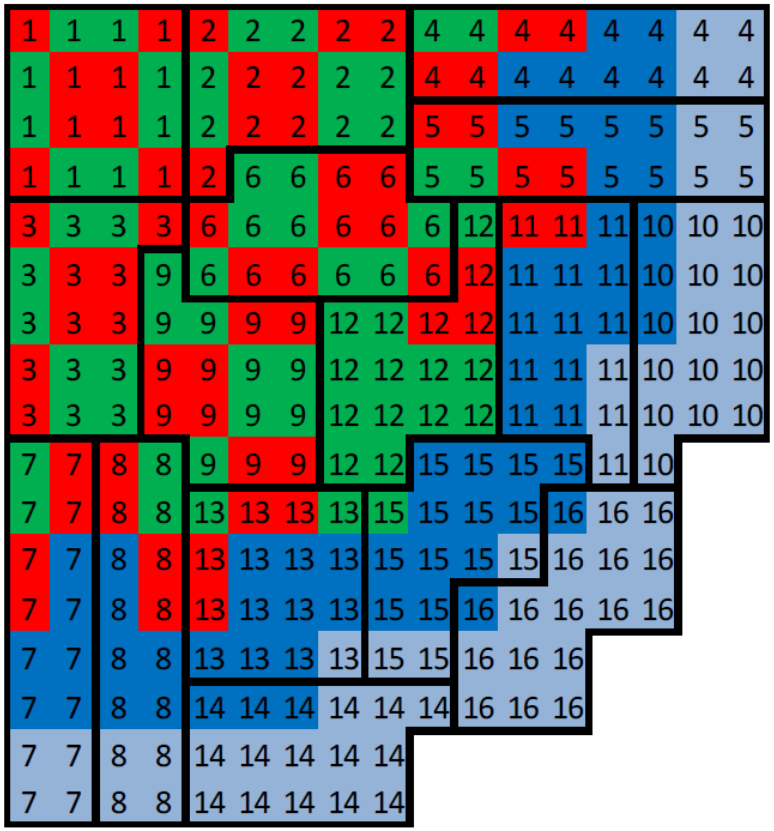
- Pad and vessel are modeled by specifying radii for each region and angles

Radial Parallel Partitioning Scheme

- Axial partitioning was covered in previous presentation
- Radial partitioning is performed on a “module” basis
  - Modules in MPACT are quarter-assemblies
  - Flexible assignment of modules to partitions
- Optimal load balance is achieved by assigning equal number of quarter-assemblies to each partition
- Partitions should be as “square” as possible
  - Recommend that you use examples provided
- Total number of cores equals number of radial partitions by number of axial partitions

Full-Core 16 Partition Example





- Assign quarter-core modules to partitions (include reflector modules)
- Good load balance with equal sized partitions

Partition Input – MPACT Block

```
[MPACT]
  num_space    8     ! Multiply by number of axial planes

  par_method   EXPLICITRADIAL
  par_map
   1  1  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2
   1  1  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2
   1  1  1  1  1  1  2  2  2  2  2  2  2  2  2  2  5
   1  1  1  1  1  1  4  4  4  4  5  5  5  5  5  5  5
   1  1  1  1  4  4  4  4  4  5  5  5  5  5  5  5  5
   1  1  1  1  4  4  4  4  4  5  5  5  5  5  5  5  5
   3  3  3  4  4  4  4  4  4  5  5  5  5  5  5  5  5
   3  3  3  4  4  4  4  4  4  8  8  8  8  8  8  8  8
   3  3  3  4  4  4  4  4  7  8  8  8  8  8  8  8  8
   3  3  3  4  6  6  6  7  7  8  8  8  8  8  8  0  0
   3  3  3  6  6  6  6  7  7  7  8  8  8  8  8  0  0
   3  3  3  6  6  6  6  7  7  7  7  7  8  8  8  0  0
   3  3  3  6  6  6  6  7  7  7  7  7  7  8  8  0  0
   3  3  3  6  6  6  6  7  7  7  7  7  7  0  0  0  0
   3  3  3  6  6  6  6  7  7  7  7  7  7  0  0  0  0
   3  3  3  6  6  6  6  7  7  0  0  0  0  0  0  0  0
   3  3  6  6  6  6  6  7  7  0  0  0  0  0  0  0  0
```

With 50 axial nodes, the total number of cores would be ~50 x 8 = 400 cores

- Simple to input (2D configuration layout in par_map)
- EXPLICITRADIAL par_method
- Same radial partitioning is applied for each plane

More Examples

17 assemblies across (with reflectors)
4-Loop Plant

1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3
1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3
1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3
1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3
1	1	1	1	1	1	2	2	2	2	2	2	3	3	3	3	3
1	1	1	1	1	5	5	5	5	5	2	2	3	3	3	3	3
4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	6
4	4	4	4	4	5	5	5	5	5	5	6	6	6	6	6	6
4	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6	6
4	4	4	4	4	4	5	5	5	5	5	6	6	6	6	0	0
4	4	4	4	4	4	5	5	5	5	5	6	6	6	6	0	0
7	7	4	4	4	4	8	8	8	8	8	8	6	6	6	0	0
7	7	7	7	7	7	8	8	8	8	8	8	6	6	6	0	0
7	7	7	7	7	7	8	8	8	8	8	8	8	0	0	0	0
7	7	7	7	7	7	8	8	8	8	8	8	8	0	0	0	0
7	7	7	7	7	7	8	8	8	0	0	0	0	0	0	0	0
7	7	7	7	7	7	8	8	8	0	0	0	0	0	0	0	0

8 radial partitions

1	1	1	1	2	2	2	2	4	4	4	4	7	7	7	7	7
1	1	1	1	2	2	2	2	4	4	4	4	7	7	7	7	7
1	1	1	1	2	2	2	2	4	4	4	4	7	7	7	7	7
1	1	1	1	2	2	2	2	4	4	4	4	11	11	11	11	7
3	3	3	3	5	5	5	5	8	8	8	8	11	11	11	11	11
3	3	3	3	5	5	5	5	8	8	8	8	11	11	11	11	11
3	3	3	3	5	5	5	5	8	8	8	8	12	12	12	11	11
3	3	3	3	5	5	5	5	8	8	8	8	12	12	12	12	12
6	6	6	6	9	9	9	9	13	13	13	13	12	12	12	12	12
6	6	6	6	9	9	9	9	13	13	13	13	12	12	12	0	0
6	6	6	6	9	9	9	9	13	13	13	13	16	16	16	0	0
6	6	6	6	9	9	9	9	13	13	13	13	16	16	16	0	0
10	10	10	14	14	14	15	15	15	15	16	16	16	16	16	0	0
10	10	10	14	14	14	15	15	15	15	16	16	16	0	0	0	0
10	10	10	14	14	14	15	15	15	15	16	16	16	0	0	0	0
10	10	10	14	14	14	14	15	15	0	0	0	0	0	0	0	0
10	10	10	10	14	14	14	15	15	0	0	0	0	0	0	0	0

16 radial partitions

Remember to include reflector row!

More Examples

- The following layouts have already been created and are included in repository:
  - bw_16r_full.txt
  - bw_16r_qtr.txt
  - bw_24r_full.txt
  - bw_32r_full.txt
  - w2loop_32r_full.txt
  - w2loop_8r_qtr.txt
  - w3loop_16r_qtr.txt
  - w3loop_8r_qtr.txt
  - w4loop_16r_qtr.txt
  - w4loop_32r_full.txt
  - w4loop_32r_qtr.txt
  - w4loop_48r_full.txt
  - etc.

You can use these partitions by adding in an include file:

```
[MPACT]
   include w4loop_16r.txt
```

No path required, the examples are in the search path

Actual include files are located in $VERAHOME/bin/Init

New GRAPH Partition

- Very flexible
- Use values proportional to the number of axial levels

```
[MPACT]
  par_method GRAPH
  graph_part_method REB
  num_space [number of planes]*[number radial]
```

Setting Number of Cores for CTF

- CTF runs on same cores as MPACT, but parallel domains are flow channels
- CTF is limited and can only run 1/4/9 cores per assembly
  - Recommend using 4 cores per assembly
- CTF will almost always use less cores than MPACT
  - This is handled automatically inside code, user does not have to do anything
- The following options are default for problems with more than one assembly:

```
[COBRATF]
  parallel       1
  proc_per_assem 4
```

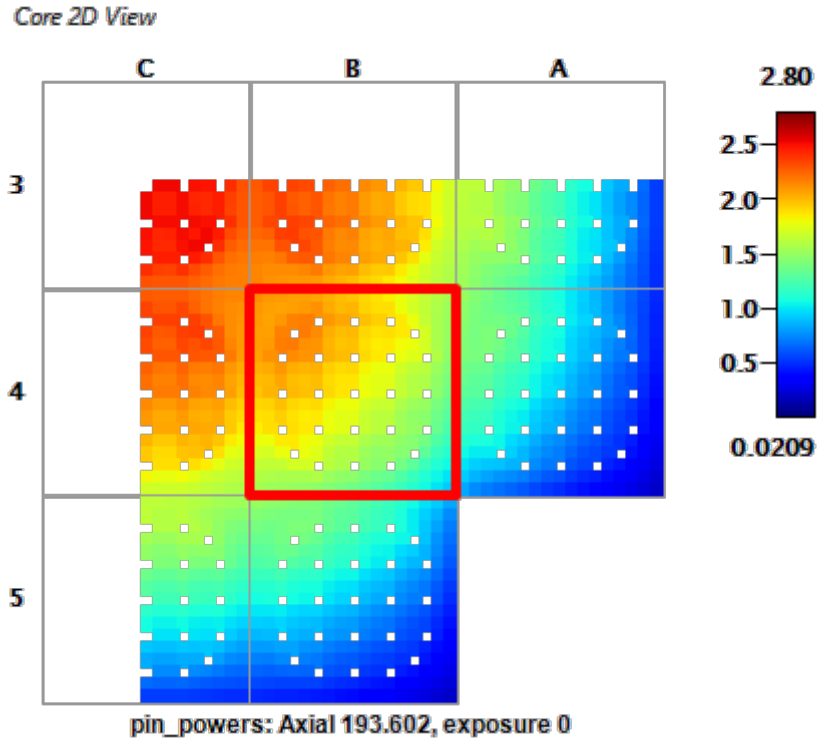
Typical Run-Times

- Problem 9 – Watts Bar Unit 1
- 16 radial x 58 axial = 928 cores
- 32 statepoints
- 24.5 hours

- approximately 46 min/statepoint

Class Exercise: Set up Mini-Core

- Start with “p6” input deck
- Convert to “minicore” input
  - Geometry is 21 assemblies
  - Modify CORE block – 6 input cards



Modifications Needed for Mini-Core

[CORE]

- Increase size to 5 (5x5 across)
- Increase rated power and flow
  - multiply by 21
- Modify core_shape
- Modify assm_map
- Remove bc_rad
- Add baffle

[COBRATF]

- Remove parallel card
- Parallel CTF will be enabled by default

```
[CORE]
  size    5
  rated   371.07 14.3283
         ! MW, Mlbs/hr

  core_shape
     0 1 1 1 0
     1 1 1 1 1
     1 1 1 1 1
     1 1 1 1 1
     0 1 1 1 0

 assm_map
    ASSY ASSY ASSY
    ASSY ASSY ASSY
    ASSY ASSY

  baffle ss 0.1 2.22

  ! Remove bc_rad

[COBRATF]
  parallel 1
  proc_per_assem 4
```

Modifications Needed for Mini-Core

[MPACT]

- Add parallel partition

```
[MPACT]
  num_space 290    ! 5x58

  par_method       EXPLICITRADIAL
  par_map
    1  1  1  3  3  3  3
    1  1  1  3  3  3  3
    1  1  1  3  2  2  2
    5  5  5  2  2  2  2
    5  5  4  4  4  2  2
    5  5  4  4  4  0  0
    5  5  4  4  4  0  0
```

par_map is for a qtr-core, including reflector nodes

Class Exercise: Set up Mini-Core

- Start with “p6” input deck
- Convert to “minicore” input
  - Geometry is 21 assemblies
  - Modify CORE block – 8 input cards

- Submit Job

```
verarun mini
```

- Wait….
- Check queue

```
qstat
```

Modify Input and Submit

Mini-Core Output

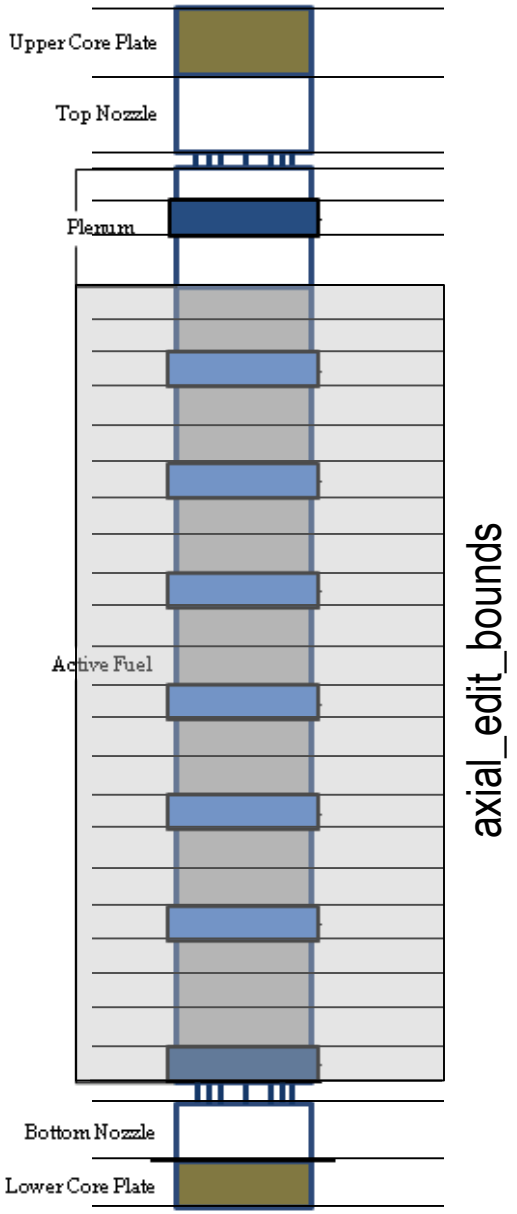
```
********************************************************************************
****************************** STATE_0001 ************************************
********************************************************************************

     State Summary
     ----------------------
     Core Exposure           0.00 MWD/MTHM
     Relative Power        100.00 %
     Thermal Power          92.77 MWt
     Relative Flow         100.00 %
     Absolute Flow         451.33 kg/s
     Inlet Temperature     291.85 C
     Boron Conc.          1300.00 ppm
     k-eff                   1.06394

                            Assembly Average Power
--------------------------------------------------------------------------------
               C          B          A
    3     1.8461     1.5480     0.7966
    4     1.5480     1.2637     0.5901
    5     0.7966     0.5901
```

Runs in about 27 minutes on 290 cores

View Results in VERAView

Axial Meshing Best Practices

- The *axial_edit_bounds* in the EDIT block defines the computational boundaries in the active fuel region
  - Start at BAF and end at TAF
  - Include all fuel boundaries (IFBA, blankets)
  - Include black absorber boundaries (WABA)
  - Include top and bottom of spacer grids
  - Subdivide remaining regions into 1.5” to 4” regions (larger may be doable)
  - Avoid thin regions (< 2 cm)
  - Manually merge boundaries that are close
    - Move items of lesser importance (grids) to match higher important boundaries (fuel, poison)
  - Avoid too many decimal places
- By default, the upper and lower reflector regions are auto-meshed



Example Axial Mesh

Users explicitly enter axial mesh for active fuel

Grid above and below fuel is automatically meshed

1	5	Lower Core Plate
2	6.053	Bottom nozzle
3	5.898	
4	3.866	Bottom of active fuel
5	8.211	
6	8.211	
7	8.211	
8	8.212	
9	8.211	
10	8.211	
11	8.211	
12	3.81	grid
13	8.065	
14	8.065	
15	8.065	
16	8.065	
17	8.065	
18	8.065	
19	3.81	grid
20	8.065	
21	8.065	
22	8.065	
23	8.065	
24	8.065	
25	8.065	
26	3.81	grid
27	8.065	
28	8.065	
29	8.065	

30	8.065	
31	8.065	
32	8.065	
33	3.81	grid
34	8.065	
35	8.065	
36	8.065	
37	8.065	
38	8.065	
39	8.065	
40	3.81	grid
41	8.065	
42	8.065	
43	8.065	
44	8.065	
45	8.065	
46	8.065	
47	3.81	grid
48	7.9212	
49	7.9212	
50	7.9212	
51	7.9212	
52	7.9212	Top of active fuel
53	8.556	
54	3.866	grid
55	3.578	
56	3.799	
57	8.827	Upper Nozzle
58	7.6	Upper Core Plate

Axial Reflector Regions

- The auto meshes created in the **non-fuel** regions are based on simple MPACT input
  - Regions are homogenized in regions > 2 cm and < 20 cm
  - Can result in 4-6 axial planes in EACH axial region
  - Can smear regions that may not be ideal
  - Final mesh is in output under 'Axial Mesh Data'
- For ultimate control of the entire axial mesh, use the *axial_mesh* MPACT input
  - can manually combine regions as desired
  - Input is mesh sizes, not elevations
  - Must be consistent with axial_edit_bounds
  - Recommended to use 'Axial Mesh Data' output as a starting point

```
[MPACT]
  meshing_method    nonfuel
  automesh_bounds   2 20
```

```
meshing_method useraxialmesh
axial_mesh
     10.0    ! water
     5.0     ! lower core plate
     6.0     ! bottom nozzle
     2.3     ! gap and plug
     4.4     ! fuel
     3.866   ! end grid
     7.0     ! fuel
   . . . . . . .
     7.6     ! fuel
     11.9    ! plenum
     10.3    ! grid, plenum, gap
     8.8     ! top nozzle
   ! 7.6     ! upper plate
```

Thoughts and Rules-of-Thumb

- Creating the axial mesh is the most cumbersome part of the input. It is recommend to make simplifications where appropriate and not use too many digits.
- If you are limited in resources, you can probably get away with 3 regions below the BAF and 3 regions about the TAF
  1. Bottom end plug and gap
  2. Bottom nozzle
  3. Lower Plate

  1. Upper Plenum
  2. Upper grid, plenum, plug, gap
  3. Top nozzle
- Consider the distance from the fuel (i.e. the upper core plate probably has no effect)
- To improve solution stability, limit the number of thin regions in the axial reflectors
- To get initial core count, run on 1 processor with par_method=ASSEMBLY, and kill job at 'Generating Input Edits'

Mixed Cores and Axial Mesh

- VERA requires ONE axial mesh for all fuel
- Optimal mesh may change if fuel designs change
- The axial mesh can be changed between cycles
  - The reinsert fuel from the restart file will be remeshed on the new cycle mesh using volume weighting

- Best practice: For varying IFBA, WABA, and blanket lengths, choose an axial mesh that will be most flexible for modeling many/future cycles
  - 6” and 8” blankets
  - 120”, 128”, 132” IFBA/WABA

Choose your axial mesh carefully and consider multiple cycles

Fuel Shuffling

- Core shuffling is done by reading previous cycle EOC restart file(s) and providing shuffle map
  - Depleted assemblies are shuffled from old locations to new locations by specifying old location coordinates
  - New assemblies are specified with a “+” indicator
- Shut-down cooling calculated for each assembly using shutdown date of previous cycle and startup date of new cycle
  - EOC restart file must have “op_date” specified
- Longer example shown in VERAIn Manual

Fuel Shuffle – STATE Block

```
[STATE]

restart_shuffle ../cycle1/beavrs_cy1_restart.h5 EOC1

shuffle_label
                        L-10 +F34 +F32 +F34 +F32 +F34 E-10
              G-10 +F32 +F32 L-02 P-12 N-03 B-12 E-02 +F32 +F32 J-10
        F-09 +F34 N-02 N-10 +F32 D-11 R-10 M-11 +F32 C-10 C-02 +F34 K-09
        +F32 P-03 L-08 +F32 M-09 E-15 G-08 L-15 D-09 +F32 H-05 B-03 +F32
   F-05 +F32 F-03 +F32 M-04 +F32 M-03 A-10 D-03 +F32 D-04 +F32 K-03 +F32 K-05
   +F34 P-05 +F32 G-04 +F32 N-08 R-09 G-14 A-09 H-03 +F32 J-04 +F32 B-05 +F34
   +F32 D-02 E-12 A-11 N-04 G-01 B-09 H-15 J-14 J-01 C-04 R-11 L-12 M-02 +F32
   +F34 N-13 F-15 H-07 F-01 B-07 A-08 F-14 R-08 P-09 K-15 H-09 K-01 C-03 +F34
   +F32 D-14 E-04 A-05 N-12 G-15 G-02 H-01 P-07 J-15 C-12 R-05 L-04 M-14 +F32
   +F34 P-11 +F32 G-12 +F32 H-13 R-07 J-02 A-07 C-08 +F32 J-12 +F32 B-11 +F34
   F-11 +F32 F-13 +F32 M-12 +F32 M-13 R-06 D-13 +F32 D-12 +F32 K-13 +F32 K-11
        +F32 P-13 H-11 +F32 M-07 E-01 J-08 L-01 D-07 +F32 E-08 B-13 +F32
        F-07 +F34 N-14 N-06 +F32 D-05 A-06 M-05 +F32 C-06 C-14 +F34 K-07
              G-06 +F32 +F32 L-14 P-04 C-13 B-04 E-14 +F32 +F32 J-06
                        L-06 +F34 +F32 +F34 +F32 +F34 E-06
```

Label syntax: <unit number>:<cycle number><X label>-<Y Label>

- Unit numbers not required
- Cycle number defaults to previous cycle
- + Signifies fresh assembly, label following + sign is not currently used

Fuel Shuffle – CORE Block

```
[CORE]

 cycle 2                  ! Cycle number
 op_date 04/01/1985       ! Date of cycle startup

 xlabel    R  P  N  M  L  K  J  H  G  F  E  D  C  B  A
 ylabel   01 02 03 04 05 06 07 08 09 10 11 12 13 14 15

  assm_map                ! Assembly types
    X                     ! X is burned fuel (arbitrary)
    X  X                  ! new fuel must have map
    X  X  X
    X  X  4  X
    X  X  X  4  X
    X  X  4  X  X  5
    X  X  X  4  4  X
    5  4  5  X

  insert_map              ! Add new insert locations (if any)
     -
     -  -
     -  -  -
     -  - 08  -
     -  -  - 12b -
     -  - 08  -  -  -
     -  -  - 04  -  -
     -  -  -  -
```

Shutdown Decay

- Shutdown decay will be calculated between the shutdown date on the restart and the startup date in the shuffle case
- e.g. xenon goes to zero, samarium usually peaks
- Every isotope decays using ORIGEN models

```
[STATE]
 deplete EFPD 441.0
 restart_write p9.res EOC
 op_date 1997/9/6
```

Write EOC restart file with shutdown date

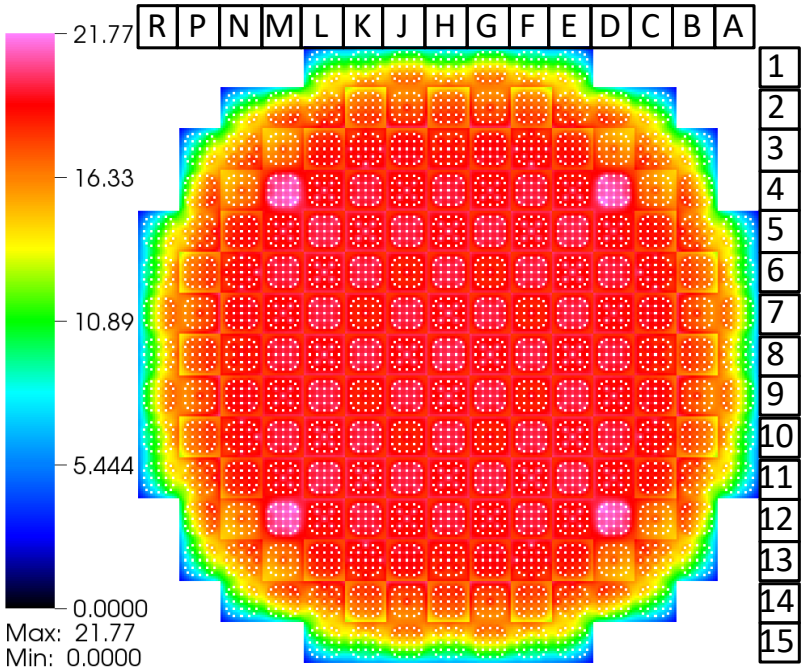
```
[CORE]

 cycle 2                ! Cycle number
 op_date 04/01/1985     ! Date of cycle startup
```

Specify startup date on BOC shuffle

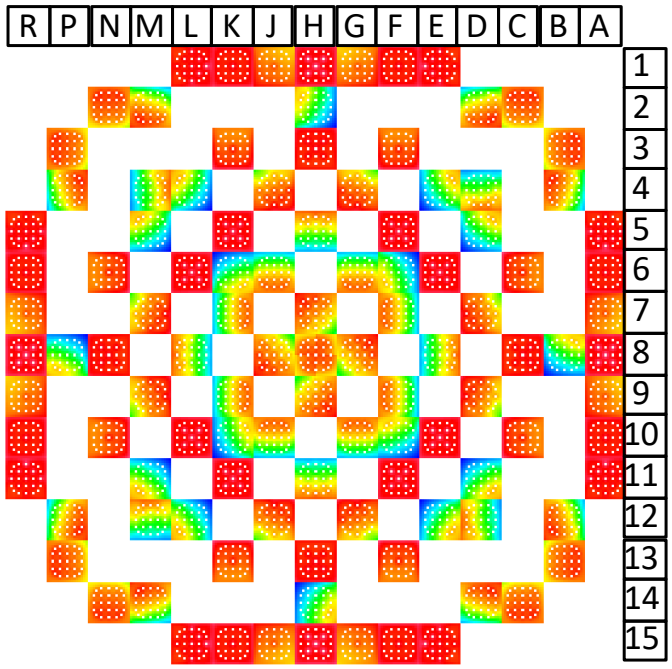
Fuel Shuffling Verification

Pin-wise Exposures Before and After Shuffle

EOC 1

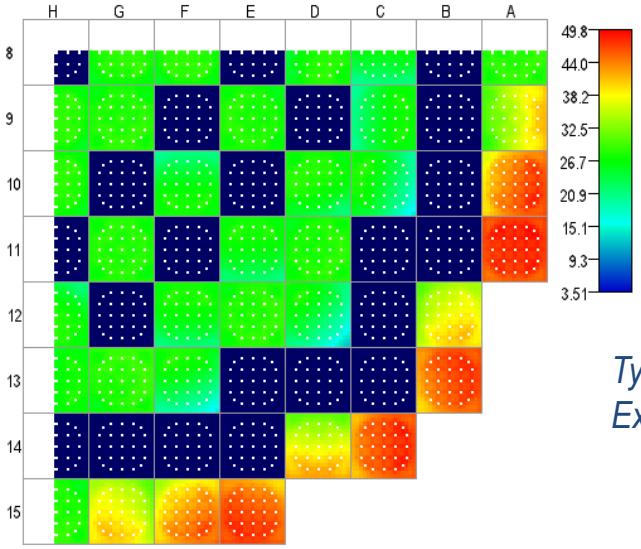


BOC 2

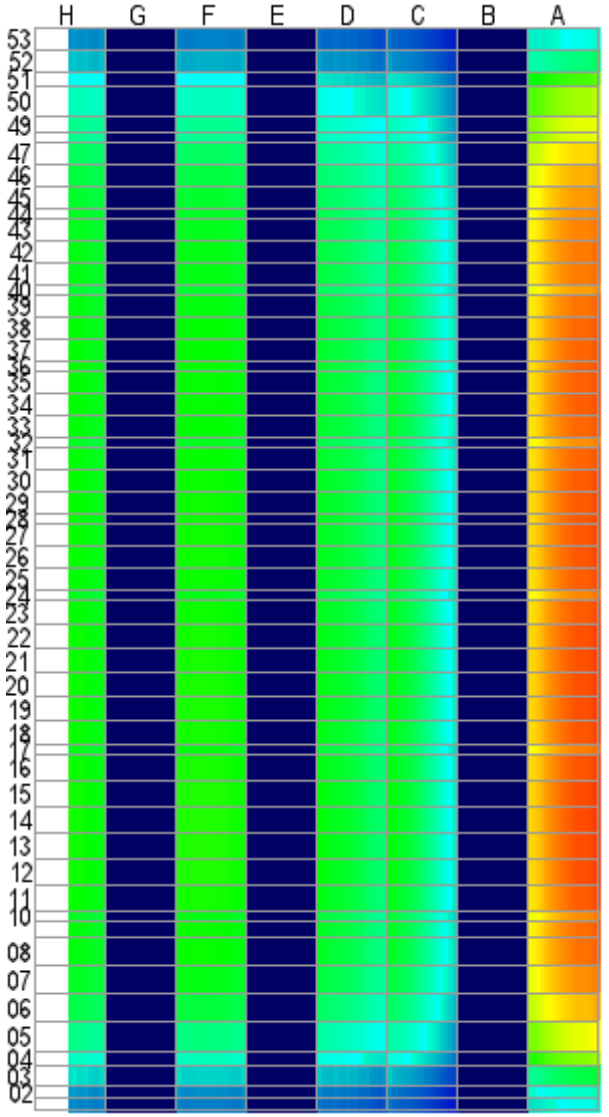


Jump-In Cycles

- Typical commercial power plants replace 1/3-1/2 of the fuel during reloads
- In order to start an analysis without going back to Cycle 1, we need a method to approximate the depletion history of reinserted fuel
- In VERA currently, this means we need to generate a reasonable 3D isotopic distribution for depleted fuel



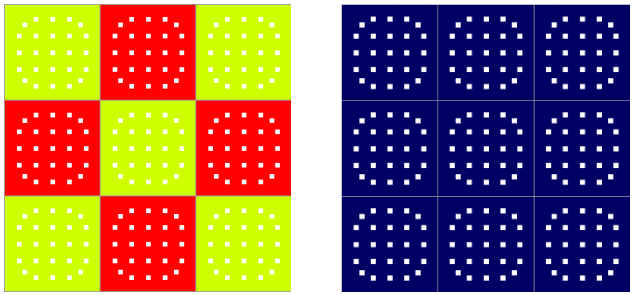
Typical Beginning-of-Cycle Exposure Distribution



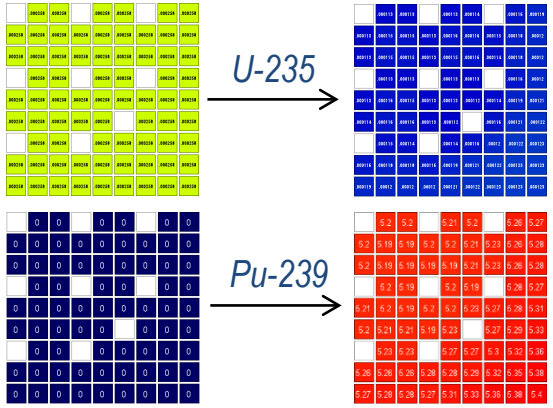
Jump-In Procedure

1. Create model for the jump-in cycle
   - Treat as “all fresh” fuel at first
2. Use model to perform single assembly depletion for burned fuel
   - Only need to modify STATE and CORE blocks
   - Deplete at HFP nominal conditions out to desired burnup and write a restart file
   - Use a unique cycle identifier and x/y labels
   - Provide a simple op_date for shutdown
3. Use standard shuffling capability to load burned fuel isotopics from single-assemblies into model from Step 1
   - Provide restart file from Step 2 on restart_shuffle
   - Set op_date in CORE block to perform decay
   - Input shuffle_label using unique cycle and core location from Step 2.

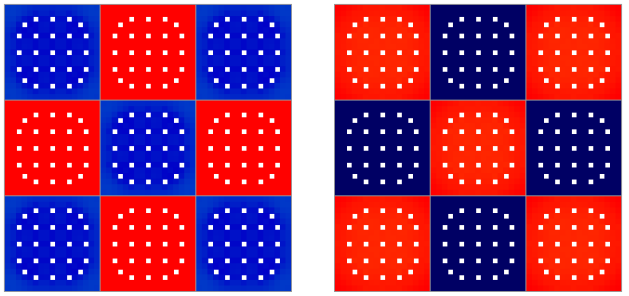
1. Initial U-235 and Pu-239 distributions



2. Single Assembly Depletion to 15 GWd/MT



3. Final U-235 and Pu-239 distributions



Jump-In Example

- Step 1: Deplete Single Assembly model to average exposure when reloaded.

```
[STATE]
  power    100.0    ! %
  pressure 2250     !psia
  sym      qtr
  feedback on
  boron    1285.2
  tinlet   565 K
  search   boron
  deplete  GWDMT 0.0 0.1 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 7.5 10.0 12.5 15.0
  restart_write batch8A.res EOL
```

- Step 2: Load data into Full Core model

```
[STATE]
  !restart file for each sub-batch at the batch-average burnup
  restart_shuffle  ./ batch8A.res EOL
                   ./ batch8B.res EOL
                   ./ batch9A.res EOL
                   ./ batch9B.res EOL
                   ./ batch9C.res EOL

  shuffle_label
     19Z-0 29Z-0 +     29Z-0 +     29Z-0 29Z-0 28Z-0
     29Z-0 +     29Z-0 +     29Z-0 39Z-0 +     18Z-0
     +     29Z-0 39Z-0 39Z-0 +     39Z-0 +     28Z-0
     29Z-0 +     39Z-0 +     29Z-0 +     +     28Z-0
     +     29Z-0 +     29Z-0 29Z-0 +     29Z-0
     29Z-0 39Z-0 39Z-0 +     +     18Z-0 28Z-0
     29Z-0 +     +     +     29Z-0 28Z-0
     28Z-0 18Z-0 28Z-0 28Z-0
```

Jump-In Comments and Challenges

- 3D single assembly depletions tend to develop unphysical axial power distributions at higher burnups
  - The assembly AO needs to be close to zero (or close to the actual reactor core history)
  - We have been able to significantly improve the accuracy of the jump-in cycle by artificially increasing the boron concentration for the depletion (increases the MTC and keeps the power more centered)
  - Because of this, the depletion can have trouble converging at high exposures
- Rather than using single assemblies, fuel can be "recycled" from future cycles or other plants
  - Axial meshes don't need to match. MPACT can "remesh" the shuffled fuel, which introduces some additional error.
- Number of depletion regions and sizes need to match
  - Otherwise, fuel mass may not be conserved
  - IFBA patterns must be consistent so number of depletion regions match

Fuel Temperatures

Currently three methods of providing fuel temperatures to MPACT for Doppler feedback

1. Fixed, uniform value via ***tfuel*** in STATE block
   - Requires ***feedback=off*** and ***modden***
2. CTF heat conduction model
   - ***NC=1*** in the COBRATF block
   - Requires user input for gap conductance
   - No burnup dependency (yet)
3. Temperature table with (T_{fuel}-T_{mod}) as a function of LHR and pin exposure
   - Only one fuel type supported (for now)
   - Applied rod-by-rod at each axial plane
   - Temperature is used uniformly within the fuel pellet

Fuel Temperature Table

- Fuel temperature input table is quadratic coefficients for power at different burnups:

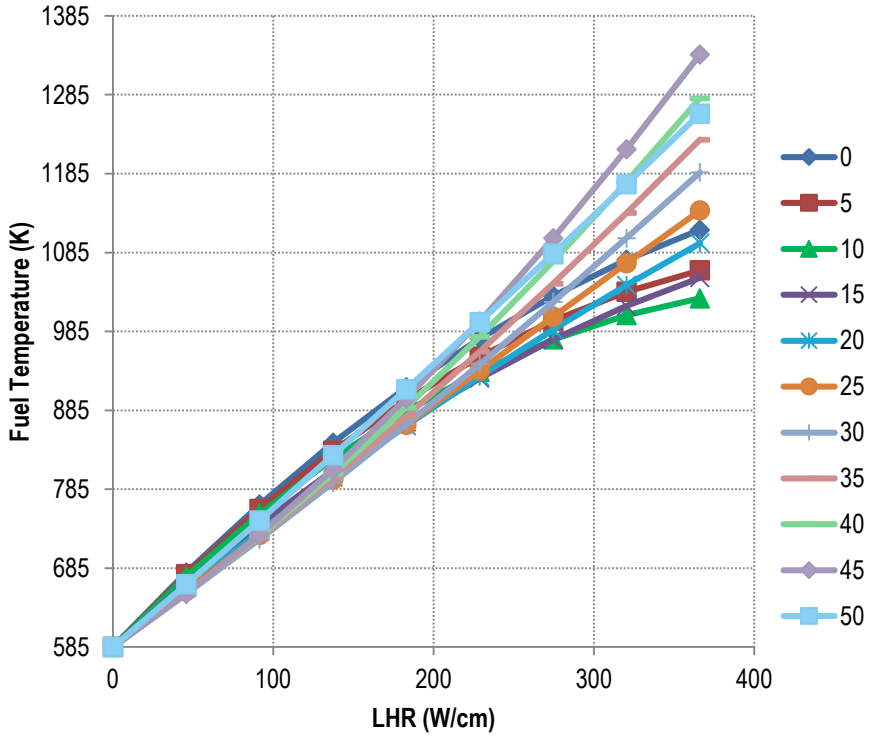
$$T_{fuel} = T_{mod} + a(exp) \times LHR + b(exp) \times LHR^2$$

where LHR is the power deposited in the fuel in W/cm

- Linear interpolation is used between burnups
- No zero power intercept is allowed (must go to zero)
- A LHR constant is available to provide flexibility in table generation
  - If table is based on LHR deposited, use 1.0
  - If table is based on LHR generated, use (1.0-dhfrac)
  - If table is based on normalized power distribution, use core average LHR times (1-dhfrac) in W/cm

Example Fuel Temperature Table

```
!      nBuPoints        LHR (W/cm)
          11               0.974

!   Bu (GWD/MT)      a (K)          b (K)
       0.00      2.14920235    -0.00192506
       5.00      2.11615487    -0.00221779
      10.00      2.05743514    -0.00232281
      15.00      1.84058120    -0.00153461
      20.00      1.66621956    -0.00073186
      25.00      1.56530475    -0.00014700
      30.00      1.44101883     0.00055258
      35.00      1.43921965     0.00086746
      40.00      1.41862382     0.00131394
      45.00      1.39113479     0.00180470
      50.00      1.72317832     0.00033787
```



$$T_{fuel} = T_{mod} + a(exp) \times LHR + b(exp) \times LHR^2$$

Data from bison_table_tavg_rev6.tab
Plot is for a fixed Tmod=585K

Fuel Temperature Table Input

```
[MPACT]
   temptable_boundary    bulk_cool    ! new BC option
   temptable_qprime      0.974
   temptable_polynomial
      0.00     2.14920235   -0.00192506
      5.00     2.11615487   -0.00221779
     10.00     2.05743514   -0.00232281
     15.00     1.84058120   -0.00153461
        ……

[COBRATF]
   nc 0           ! disable conduction model
   dhfrac 0.026   ! direct moderator heating fraction
```

- Temperature tables usually come from BISON, so users do not need to develop the table by hand
- Users may also use tables generated by other fuel performance codes

Fuel Temperature Table Input

- Example fuel temperature tables exist for most users!
- Based on Watts Bar geometry

Most Applications

```
[MPACT]
   include bison_table_tavg_rev7.tab
```

- If you need to generate your own tables, ask for assistance.
- Additional features are under active development (multiple tables, gad tables, etc.)
- Fast running fuel performance models also under development! **CTFFuel**

CTFFuel should be in final 4.0 Release!

Ask for Help!

- Following topics are “advanced” and there is still active development in these areas
  - Jump-in cycles
  - Axial mesh for non-uniform assembly types
  - Fuel temperature tables
- Detailed examples usually exist for different reactor types

Don’t be afraid to ask developers for guidance!

Class Exercise: SMR

- Run small sample SMR reactor, which has full-core details but runs fast
- Additional simplifications to run fast
- Reactor is based on public NuScale geometry with fuel from Watts Bar (i.e. not realistic)
- Copy sample input:

```
cp /projects/vera-users-grp/training/smr/smr.inp .
```

- Look at input:
  - How many assemblies?
  - How many axial mesh?
  - How many radial partitions?
  - How many depletion steps?

Class Exercise: Run SMR

- Create subdirectory and copy sample problem

```
mkdir smr
cd smr
```

- Copy sample file from training directory

```
cp /projects/vera-users-grp/training/smr/smr.inp .
```

- Run Job

```
verarun smr
```

- Wait….
- Check queue

```
qstat
```

Copy Input file and Submit

SMR Results

```
================================
       Statepoint Summary
================================
   N    exposure   exposure   eigenvalue    boron      3PIN      2PIN      3EXP      A/O(%)
   1     0.0000       0.00     1.000003    881.42    2.1577    1.4695    0.0000    -7.1589
   2     0.1000       5.82     0.999999    854.19    2.1261    1.4644    0.2144    -7.0148
   3     1.0000      58.19     1.000000    828.31    2.0391    1.4151    2.0872    -6.4080
   4     3.0000     174.57     0.999995    754.02    1.7896    1.3305    5.8890    -4.1446
   5     5.0000     290.95     0.999999    635.63    1.6853    1.3139    9.2614    -3.0477
   6     7.0000     407.33     1.000000    502.36    1.5934    1.3168   12.3037    -2.3103
   7     9.0000     523.70     1.000000    355.75    1.5165    1.3069   15.1461    -1.6116
   8    11.0000     640.08     0.999998    198.73    1.4952    1.2916   17.9903    -0.8725
   9    12.0000     698.27     0.999999    117.74    1.4765    1.2880   19.3352    -0.6380
```

Runs in about 33 min on 320 cores
(with fast run-time options)

Questions?

www.casl.gov