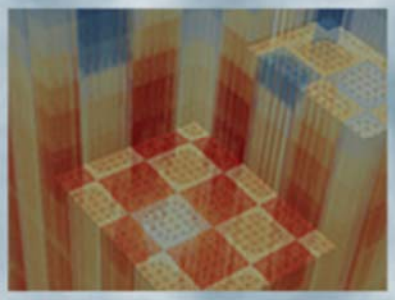
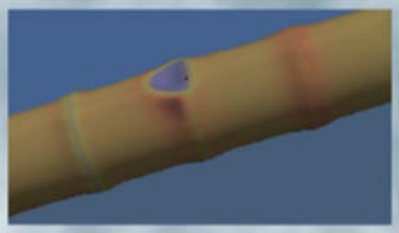
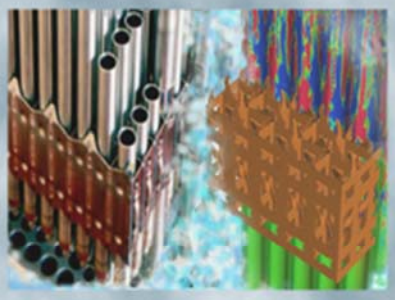
CASL-U-2016-1109-000







MPACT VERA Input User's Manual Version 2.2.0

June 9, 2016

REVISION LOG

Revision	Date	Affected Pages	Revision Description
0		All	Initial Version

Document pages that are:

Export Controlled _____None_____

IP/Proprietary/NDA Controlled___None_____

Sensitive Controlled_____None_____

Requested Distribution:

To: Unlimited distribution

Copy:

MPACT VERA Input User's Manual

Version 2.2.0

June 9, 2016

Contributors (in alphabetical order)

- Dr. Benjamin Collins (ORNL)
- Prof. Thomas J. Downar (UM)
- Andrew Fitzgerald (UM)
- Dr. Jess Gehin (ORNL)
- Andrew Godfrey (ORNL)
- Aaron Graham (UM)
- Daniel Jabaay (UM)
- Dr. Blake Kelley (formerly UM)
- Dr. Kang Seog Kim (ORNL)
- Dr. Brendan Kochunas (UM)
- Joel Kulesza (UM)
- Prof. Edward Larsen (UM)
- Dr. Yuxuan Liu (UM)
- Dr. Zhouyu Liu (formerly UM)
- Prof. William R. Martin (UM)
- Dr. Adam G. Nelson (formerly UM)
- Dr. Scott Palmtag (ORNL)
- Michael Rose (UM)
- Dr. Thomas Saller (formerly UM)
- Dr. Shane Stimpson (ORNL)
- Dr. Travis Trahan (formerly UM)
- Jipu Wang (UM)
- Dr. Will Wieselquist (ORNL)
- Mitchell T.H. Young (UM)
- Ang Zhu (UM)

Contents

Chapter 1

Introduction

The MPACT (**M**ichigan **P**Arallel **C**haractistics based **T**ransport) code is designed to perform high-fidelity light water reactor (LWR) analysis using whole-core pin-resolved neutron transport calculations on modern parallel-computing hardware. The code consists of several libraries which provide the functionality necessary to solve steady-state eigenvalue problems. Several transport capabilities are available within MPACT including both 2-D and 3-D Method of Characteristics (MOC). A three-dimensional whole core solution based on the 2D-1D solution method provides the capability for full core depletion calculations.

Specific features available in the current release of MPACT are:

- Support for Microsoft Windows Operating Systems (32-bit and 64-bit)
- Support for Linux-based Operating Systems (32-bit and 64-bit)
- OpenMP parallelism for MOC sweeps
- Support for MPACT and AMPX working cross section library formats
- Steady-state eigenvalue calculations using power iteration
- 2-D and 3-D MOC transport solvers
- 2D-1D full core solution
- Depletion capability
- Generalized pressurized water reactor (PWR) geometry
- Export of computational and results mesh to VTK files
- Visualization via VisIt

The purpose of this document is to provide users with sufficient background to be able to utilize MPACT for PWR design and analysis applications. For a more detailed description of the methods or sofware design, the reader is directed to the theory and programmer's documentation.

This user document is divided into several chapters. After this introduction, an overview is provided regarding code execution capabilities and limitations in both serial and parallel environments. Finally, a detailed description of the user input is provided.

For specific questions about the use of MPACT, the licensing of the code, or to report bugs users are encouraged to send an email to support@casl.gov. When reporting bugs, users are requested to attach the problematic input to the email and to provide information in the body of the email about the code version, machine, runtime environment and any other relevant details to permit debugging.

Explanation of Notation

In several of the code examples that follow in this document a specific syntax is used which can be described as:

- Words appearing in typewriter font within the normal text, such as `this` indicate the word is a reference to a something that is used in an example. Paragraphs or lines of text in this formatting are examples of usage.
- Words bracketed with '<' and '>' in examples such as <token> indicate a single token for which a value is expected. This value is typically some intrinsic data type such as an integer, string, real, or logical.
- Tokens that are bracketed by '[' and ']' in examples such as [<token>] indicate an optional value. Optional values may become nested such as [<token1> [<token2>]]
- A vertical bar '|' in an example indicates only one out of the set should be used. The set will be defined by one of the above sets of brackets. For example, [<token1> | <token2>] means that only one of <token1> or <token2> should be entered and that this entry is optional.
- Tokens appended by a '(:)' indicate an array of values, where the number of ':' indicates the number of dimensions of the array. For example, <matrix(:,:)> is a token that is a 2-D array.

Additionally, the input cards presented in Chapter 4 are described in the following tabular format:

`The name of the input card`	Argument data type(s) (e.g. Integer)	Required / Optional
Units: The default units associated with this card, if any. (default), Other units that can be associated with this card, if any.		
Applicable Value(s): The default values if this card is optional. (default), A list of allowable values for this card, and any explanation of those values.		
Limitation(s): This section explains the limitations of this card, specifically in regards to its interactions with other inputs.		
Description: This section offers a more verbose description of the card. It also describes interactions this card may have with other cards.		
Notes: The notes section is a catch-all for information that is not elsewhere in the table.		

Chapter 2

Executing MPACT

Depending on how MPACT was configured, compiled, and installed, it may be executed in serial and/or parallel. The following sections outline the procedures for running the code in either serial or parallel. When run as a standalone analysis tool, MPACT is executed from the command line.

2.1 Standalone Serial Execution

The syntax for MPACT is shown below:

```
$> <path_to_MPACT>/mpact.exe [<input_file> [<output_file> [<log_file>]] | -
     help]
```

All command line arguments are optional. The meaning of each is described as:

- `-help` - Displays the help message. This message describes the command line arguments and their usage.
- <`input_file`> - The name of the input file to process. If no input file is listed, then MPACT tries to process the file `mpact.inp` in the present working directory. The <`input_file`> may include an absolute or relative path to the file. This file must exist and be readable prior to execution.
- <`output_file`> - The file to use for writing the default output. If no file is listed then a file with <casename>.out will be created in the present working directory. In general, if the output file does not exist it will be created, and if it already exists it will be replaced **without** warning. <`output_file`> may include an absolute or relative path. The directory pointed to by the path must exist prior to execution.
- <`log_file`> - The file to use for writing the execution log information. If no file is listed then a file with <casename>.log will be created in the present working directory. In general, if the log file does not exist, it will be created, and if it already exists it will be replaced **without** warning. <`log_file`> may include an absolute or relative path. The directory pointed to by the path must exist prior to execution.

2.2 Standalone Parallel Execution

MPACT may only be executed in parallel if a parallel build has been installed. MPACT uses two kinds of parallel models. The first is the shared memory model which is based on the OpenMP standard (http://www.openmp.org) and the other is a distributed memory model which is based on the MPI Standard. If the MPACT executable is built with MPI then it is executed differently than in serial, but otherwise the serial description in the previous section is correct. When executing MPACT with MPI the command has the following syntax:

```
$> <mpirun_cmd> [<mpi_options>] <mpact_exe> [<mpact_options>]
```

- <mpirun_cmd> - This is the command used to launch MPI executables. This command can vary because different machines may have different implementations of the MPI library installed. Therefore, it is suggested the user consult the documentation for their cluster or workstation for this command and its arguments. For most implementations of MPI (such as OpenMPI, http://www.open-mpi.org/) the <mpirun_cmd> command is mpirun.
- <mpirun_options> - These are command line arguments for <mpirun_cmd>. Again, the user should consult their machine's documentation for usage.
- <mpact_exe> - The name of the MPACT executable that is installed. Typically, one should include the full path to the executable since the parallel execution environment may not have the same PATH setting as the run time environment in which the <mpirun_cmd> was invoked.
- <mpact_options> - The command line arguments for MPACT. See the Serial Execution section of this chapter for a complete description.

Chapter 3

Input File Structure

3.1 VERA Common Input

MPACT has two input processors: one for processing the common input that is used with CASL's VERA code suite and another for processing MPACT's native input format.

The VERA common input is an ASCII input file that is processed to create an XML-like (eXtensible Markup - Language) input file. The file describes the input hierarchically. While there are several blocks of inputs created during the file processing, the block described in this manual applies only to the MPACT block of inputs.

Since this file format augments MPACT's base functionality, it requires additional packages for MPACT to be able to read it. These packages are TeuchosWrappersExt and Trilinos, and they expose functions and subroutines that facilitate the reading of data from the XML-like format into MPACT. While the user should not have to work with these details directly, it is important to be aware of the additional requirements in order to be able to efficiently diagnose issues.

One feature of the core simulator is that a single input is used to drive all of the multiphysics codes. The benefits of this approach are that users only need to understand and be proficient with one input, and it also ensures that all codes are working from a single, common geometry description of the problem to reduce errors.

This section describes using the cards that can be specified in the VERA common input to control functionality in MPACT.

For more information about the native MPACT input, please consult the MPACT Users Manual.

Chapter 4

Using MPACT with the VERA Common Input

4.1 MPACT Block

The `MPACT` block in the VERA Common input is used to define all of the MPACT specific parameters needed. The inputs are broken up into various sections and groupings that apply to the same part of the code. The list of these sections are:

- Base Inputs
- 2D/1D Inputs
- CMFD Inputs
- Iteration Control Inputs
- Meshing Inputs
- Quadrature Set Inputs
- XS Library Inputs
- Parallel Environment Inputs
- Depletion Inputs
- Thermal-Hydraulic Inputs

All cards are alphabetized within each of their subsections.

4.1.1 Base Inputs

This section lists the inputs that are placed directly within the MPACT block in the VERA Common Input and do not correspond to a particular subsection. The rest of the subsections contain input cards pertaining to specified subsection.

4.1.1.1 checkpoint_mode

```
checkpoint_mode  [<T|F|R|W|RW|I>]
```

checkpoint_mode	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): I (default), T, F, R, W, RW which correspond to: • I — specifies that a checkpoint file may be written through a user interrupt. • T — specifies that the case will be started from a checkpoint file. • F — disables initialization of the checkpoint file. • R — same as T. • W — specifies that a checkpoint file is to be written. • RW — same as T and R but after the checkpoint file is read it can be overwritten during the calculation.		
Limitation(s): File system permissions must be configured such that MPACT can interact with files, as needed.		
Description: This card is used to control whether the calculation is restarted from a checkpoint file. The user can send the interrupt signal to MPACT after execution has begun by creating a file named "MPACT_CHECKPOINT_FILE" in the simulation's working directory. The existence of this file causes a checkpoint file to be written after every outer iteration. Likewise, the removal of "MPACT_CHECKPOINT_FILE" disables the writing of a checkpoint file. See the checkpoint_file card regarding checkpoint file naming.		
Notes: None		

4.1.1.2 checkpoint_file

```
checkpoint_file  [<filename>]
```

checkpoint_file	Free-form Character String	Optional
Units: N/A		
Applicable Value(s): <CASEID>.mcp (default)		
Limitation(s): The filename must be specified with characters valid for use on the computer system being executed on. As a general practice, one should avoid the use of "special" characters. Similarly, one must not specify a name that conflicts with other files that are (or will be) created within the MPACT directory.		
Description: If a checkpoint file will be used, then the user can optionally specify the name of this file of his or her choosing.		
Notes: There is no strict limit on how many characters can be used to to specify the filename; however, good judgment should be used to keep the filename a reasonable length.		

4.1.1.3 jagged

```
jagged  [<true|false>]
```

jagged	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): true (default), false		
Limitation(s): See Notes regarding potential inefficiencies when running a parallel-processing simulation.		
Description: This card is used to specify whether the reflector region will be modeled using a jagged (stair-step) representation or by filling the full square extent of the modeling domain with moderator material.		
Notes: When a jagged core is used, care should be taken if the user elects to perform manual parallel domain decomposition to ensure proper load balancing. Additional information is available regarding this is provided with the par_file.		

4.1.1.4 ray_spacing

```
ray_spacing  [<spacing>]
```

ray_spacing	Floating-Point Real Number	Optional
Units: cm (default)		
Applicable Value(s): 0.05 (default), Positive floating-point real numbers		
Limitation(s): None		
Description: This card is used to specify the characteristic ray spacing for the rays used in the MOC calculation. A finer spacing will permit a more-detailed calculation (with finer spatial features) at the cost of computing time. However, the decomposition of rays across multiple threads parallelizes very efficiently. Finally, one should be cognizant of minimum feature size (i.e., minimum flat-source region size) to ensure that there are an adequate number of rays traversing each region to have an accurate solution in that region. More information regarding the MOC methodology and implications of ray_spacing on the overall calculation is available in the MPACT Theory Manual.		
Notes: None		

4.1.1.5 shield_ray_spacing

```
shield_ray_spacing  [<spacing>]
```

ray_spacing	Floating-Point Real Number	Optional
Units: cm (default)		
Applicable Value(s): 0.05 (default), Positive floating-point real numbers		
Limitation(s): None		
Description: This card is used to specify the characteristic ray spacing for the rays used in the MOC sheidling calculation. A finer spacing will permit a more-detailed calculation (with finer spatial features) at the cost of computing time. However, the decomposition of rays across multiple threads parallelizes very efficiently. Finally, one should be cognizant of minimum feature size (i.e., minimum flat-source region size) to ensure that there are an adequate number of rays traversing each region to have an accurate solution in that region. More information regarding the MOC methodology and implications of ray_spacing on the overall calculation is available in the MPACT Theory Manual.		
Notes: None		

4.1.1.6 moc_kernel

```
moc_kernel  [<1G|MG>]
```

moc_kernel	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): MG (default)		
Limitation(s): None		
Description: This card is used to specify whether one-group or multi-group MOC kernels are used.		
Notes: None		

4.1.1.7 shield_moc_kernel

```
shield_moc_kernel  [<1G|MG>]
```

shield_moc_kernel	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): same value as moc_kernel (default)		
Limitation(s): None		
Description: This card is used to specify whether one-group or multi-group MOC kernels are used for the shielding sweeper.		
Notes: None		

4.1.1.8 volume_corr

```
volume_corr  [<NONE|ANGLEDEP|INTEGRAL>]
```

volume_corr	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): INTEGRAL (default)		
Limitation(s): None		
Description: This card is used to specify the volume correction being applied to the MOC segments.		
Notes: None		

4.1.1.9 modular_rays

```
modular_rays  [<DECART|TWO|THREE|CACTUS|RATFRAC>]
```

modular_rays	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): TWO (default)		
Limitation(s): None		
Description: This card is used to specify the volume correction being applied to the MOC segments.		
Notes: None		

4.1.1.10 shield_nbatch

```
shield_nbatch  [<num_batch>]
```

shield_nbatch	Integer	Optional
Units: N/A		
Applicable Value(s): 5 (default)		
Limitation(s): None		
Description: This card is used to specify the number of batches used to divide the pseudogroups of the MG shielding sweeper.		
Notes: None		

4.1.1.11 rod_treatment

```
rod_treatment  [<true|false>]
```

rod_treatment	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): false (default), true		
Limitation(s): None		
Description: This card toggles the use of volume-weighting for control rods in order to minimize the effect of control rod cusping on the calculated results. Rod cusping is a calculational effect that occurs when a control rod is partially inserted into a calculational plane. This causes an artificial reduction in the local flux which in turn causes an error in the calculated eigenvalue and global power distribution. Enabling this rod treatment card reduces the volume fraction of the control rod to correct for this artificial reduction.		
Notes: This card only has an effect when used in a 3D calculation (i.e., a calculation with axial planes).		

4.1.1.12 uniform_crud

```
uniform_crud  [<thickness>] [<crud_mass>] [<boron_mass>]
```

uniform_crud	Floating-Point Real Numbers	Optional
Units: microns, mg/cm^2, mg/cm^2 (default)		
Applicable Value(s): 0.0, 0.0, 0.0 (default)		
Limitation(s): None		
Description: This card is used to define a uniform layer of CRUD on all fuel pins. The thickness is the CRUD thickness in microns, the crud_mass is the surface mass density of Ni Fe_2 O_4 in mg/cm^2, and the boron_mass is the surface mass density of Li B_4 O_7 in mg/cm^2.		
Notes: None		

4.1.1.13 vis_edits

```
vis_edits  [<none|core|fsr>]
```

vis_edits	Fixed Character String	Optional
Units: N/A		

continued on next page...

`vis_edits`, continued...

Applicable Value(s): core (default), none, fsr. These are described as: 1. `core` — will print pin level edits of power for the full core 2. `none` — will not print any visualization files 3. `fsr` — will print all available edits in the code on a flat source region-basis which includes material boundaries, mesh identification indices, and group-wise scalar flux
Limitation(s): None
Description: This card is used to specify the type of visualization a outputs (edits). The visualization outputs are created in the form of the VTK legacy file format which is suitable for use with VisIt (https://wci.llnl.gov/simulation/computer-codes/visit/), or other suitable programs capable of reading the format.
Notes: The FSR edits will be very large and may consume considerable time to generate the visualization files.

4.1.2 2-D/1-D Inputs

This section lists the inputs that pertain to the 2-D/1-D section of the code, more specifically the axial solvers and solution methodology.

4.1.2.1 nodal_method

```
nodal_method  [<sanm|nem|nem-mg|sn-0|sn-1|sn-2|sn-3|sp1|sp3|sp5|hysp3|none>]
```

`nodal_method`	Fixed Character String	Optional

Units: N/A

Applicable Value(s): sp3 (default), sanm, nem-mg, sn-0, sn-1, sn-2, sn-3, sp1, sp3, sp5, hysp3 none.

Described as:

Input Option	Full Name
SANM	Semi-Analytic Nodal Method
NEM	Nodal Expansion Method
NEM-MG	Multi-Group Nodal Expansion Method
SN-0	Discrete Ordinates with 0th Azimuthal Moment
SN-1	Discrete Ordinates with 1st Azimuthal Moment
SN-2	Discrete Ordinates with 2nd Azimuthal Moment
SN-3	Discrete Ordinates with 3rd Azimuthal Moment
SP1	Simplified Pn 1st Order
SP3	Simplified Pn 3rd Order
SP5	Simplified Pn 5th Order
HYSP3	Hybrid-Simplified Pn 3rd Order with NEM
NONE	Finite-Difference Method

Limitation(s): Only applies to 3-D models run with 2-D/1-D.

Description: This card is used to specify the type of nodal axial solver that will be used to solve the 1-D portion of the 2-D/1-D solution.

Notes: The Sn methods are the most computationally intensive. SP3 is recommended as the best balance of accuracy and speed. If convergence/stability issues are encountered with SP3, then try running with NEM.

4.1.2.2 split_TL

```
split_TL  [<true|false>]
```

split_TL	Logical	Optional
Units: N/A		
Applicable Value(s): true (default), false		
Limitation(s): Only applies to 3-D models run with 2-D/1-D.		
Description: This card is used to specify whether transverse leakage splitting will be enabled for a calculation using a 2-D/1-D method. In the 2-D/1-D method the axial transverse leakage is subtracted from the total fission and scattering sources, thus in regions with relatively large axial streaming sources, the total source may become negative. To avoid negative total sources the transverse leakage is split between the right hand side and left hand side of the 2-D transport equation, thus ensuring positivity of the total source and neutron balance.		
Notes: None		

4.1.2.3 TL_treatment

```
TL_treatment [<flat|lflat>]
```

TL_treatment	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): lflat (default), flat. These are described as: • `lflat` — checks the total / transport cross section. If the value is below the threshold, leakage will not be put into that region. This process is usually to avoid leakage in the fuel- clad gap. It will then redistribute the leakage to the other regions in that pin. • `flat` — does not perform leakage threshold checks.		
Limitation(s): None		
Description: This card is used to specify the type of spatial shape of the axial transverse leakage applied to the 2-D problem. Flat means it is constant over a pin cell. This is primarily used to ensure stability of the iteration.		
Notes: None		

4.1.3 CMFD Inputs

This section lists the inputs that pertain to CMFD section of the code. In particular, it has inputs for changing the type of CMFD solver, and iteration specifications.

4.1.3.1 cmfd_angle_decomp

```
cmfd_angle_decomp  [<true|false>]
```

cmfd_angle_decomp	Logical	Optional
Units: N/A		
Applicable Value(s): true (default), false		
Limitation(s): If angle decomposition or CMFD is not used this card has no effect.		

continued on next page...

`cmfd_angle_decomp`, continued...

Description: This card is used to specify whether or not the angular decomposition processors for MOC are to be used during the CMFD setup/solve. The default for this treatment is `true`, and is recommended for better parallel efficiency.
Notes: None

4.1.3.2 cmfd

```
cmfd  [<cmfd|ycmfd|adcmfd|none>]
```

`cmfd`	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): cmfd (default), ycmfd, none described as: • `cmfd` — standard CMFD method. • `ycmfd` — modified CMFD method that enhances the stability of nonlinear iteration schemes. • `adcmfd` — the new artificially diffusive CMFD method. This method should perform about as well as CMFD, but be unconditionally stable and have improved convergence for a wider range of problems. • `none` — disables CMFD and can only be used in 2-D problems.		
Limitation(s): None		
Description: This card is used to specify which CMFD method will be used.		
Notes: CMFD must be present for every 3-D problem because it is the basis for the solution transfer between 2-D and 1-D.		

4.1.3.3 k_shift

```
k_shift  [<k_shift>]
```

`k_shift`	Floating-Point Real Number	Optional
Units: N/A		
Applicable Value(s): 1.5 (default)		
Limitation(s): Can only be used with the mgnode CMFD solver. This card is irrelevant unless the `constant` option is used for the `cmfd_shift_method` card.		
Description: This card is used to specify a shifted eigenvalue problem for the CMFD power iterations.		
Notes: `k_shift` should be larger than the eigenvalue of the system. Even a value of 2 would provide some enhanced convergence properties over not using `k_shift`.		

4.1.3.4 cmfd_shift_method

```
cmfd_shift_method [<none|constant|adaptive|sdws-ips>]
```

`cmfd_shift_method`	Fixed Character String	Optional
Units: N/A		

continued on next page...

`cmfd_shift_method`, continued...

Applicable Value(s): constant (default), none, adaptive, sdws-ips described as:

- `none` — does not apply a shift to the CMFD system.
- `constant` — applies a constant, iteration-independent shift to the CMFD system. The constant is given by the reciprocal of the input to the `cmfd` card.
- `adaptive` — uses a traditional Wielandt shift method. The shift parameter is an iteration-dependent, spatially-constant quantity defined by:

$$\lambda_{adaptive}^{(n)} = \max\left\{\lambda^{(n)} - c_1\left|\lambda^{(n)} - \lambda^{(n-1)}\right| - c_0\,, \lambda_{min}\right\}.$$

  c_1, c_0, and λ_{min} have been hard-coded to 10, 0.02, and 0.3, respectively. Future implementations of the method may allow the user to specify these parameters.

- `sdws-ips` — uses a space- and iteration-dependent Wielandt shift based on the local infinite-medium eigenvalues, $\lambda_{adaptive}$, and the current guess of the eigenvalue:

$$\lambda_{IPS}^{(n)}(\mathbf{x}) = \max\left\{\lambda_{adaptive}^{(n)}, \min\left\{\lambda_{\infty}(\mathbf{x}), \lambda^{(n)} - 0.01\right\}\right\}$$

Limitation(s): These methods can only be used with the mgnode CMFD solver or the MGRBSOR solver.

Description: This card is used to specify which Wielandt shift method will be used to accelerate the power iterations on the CMFD problem.

Notes: None

4.1.3.5 cmfd_eigen_solver

```
cmfd_eigen_solver  [<power|JD|GD|Arnoldi|SLEPc_power>]
```

`cmfd_eigen_solver`	Fixed Character String	Optional

Units: N/A

Applicable Value(s): power (default), JD, GD, Arnoldi SLEPc_power described as:

- `power` — Standard power iteration.
- `JD` — SLEPc Jacobi-Davidson Solver.
- `GD` — SLEPc Generalized Davidson Solver.
- `Arnoldi` — SLEPc Arnoldi Solver.
- `SLEPc_power` — SLEPc power iteration for comparison.

Limitation(s): None

Description: This card is used to specify which eigenvalue solver will be used.

Notes: CMFD must be present for every 3-D problem because it is the basis for the solution transfer between 2-D and 1-D.

4.1.3.6 cmfd_num_outers

```
cmfd_num_outers  [<n_outers>]
```

cmfd_num_outers	Integer	Optional
Units: N/A		
Applicable Value(s): 50 (default), Any positive value.		
Limitation(s): None		
Description: This card is used to specify the number of outer eigenvalue power iterations to perform during a CMFD acceleration calculation.		
Notes: The maximum value is always 200 in the first outer eigenvalue iteration.		

4.1.3.7 cmfd_solver

```
cmfd_solver  [<1gsweep|mgnode|mggroup|1grbsor|mgrbsor>]
```

cmfd_solver	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): mgnode (default), mgnode, mggroup described as: • 1gsweep — sweeps through all of the energy groups one by one using Gauss-Seidel iteration in energy. • mgnode — sets up a full multigroup CMFD matrix in node-major ordering (e.g. each node is a group-by-group block). • mggroup — sets up a full multigroup CMFD matrix in group-major ordering. • 1grbsor — sweeps through all of the energy groups one by one using Red-Black Successive Over-Relaxtion iteration. • mgrbsor — sets up a full multigroup CMFD matrix in node-major ordering (e.g. each node is a group-by-group block) and uses Red-Black Successive Over-Relaxtion iteration.		
Limitation(s): None		
Description: This card is used to specify how the CMFD linear system is setup and solved.		
Notes: 1gsweep requires less memory than the others, but is generally slower to converge than mgnode.		

4.1.3.8 cmfd_up_scatter

```
cmfd_up_scatter  [<n_upscat>]
```

cmfd_up_scatter	Integer	Optional
Units: N/A		
Applicable Value(s): 2 (default), Any positive integer.		
Limitation(s): Only applies to 1gsweep CMFD solver.		
Description: This card is used to specify the number of upscatter iterations when doing 1gsweep CMFD. This can help to converge the scattering source in thermal energy groups before updating the fission source. In general, this can be used to help optimize run time for a given problem.		
Notes: None		

4.1.3.9 subplane_target

```
subplane_target <subplane_target>
```

subplane_target	Floating-Point Real Number	Optional
Units: cm (default)		
Applicable Value(s): N/A (default), Positive real numbers		
Limitation(s): None		
Description: This card is used to designate the target thickness of axial meshes in the CMFD system.		
Notes: Under Development, do not use!		

4.1.3.10 subplane_max

```
subplane_max <subplane_max>
```

subplane_max	Floating-Point Real Number	Optional
Units: cm (default)		
Applicable Value(s): N/A (default), Positive real numbers		
Limitation(s): None		
Description: This card is used to designate the maximum thickness of axial meshes in the CMFD system. All MOC planes with thicknesses greater than this will be sub-divided in the CMFD system using the subplane_target value.		
Notes: Under Development, do not use!		

4.1.4 Iteration Control Inputs

This section lists the inputs that pertain to the {iteration} section of the code. These options control the iteration of the transport solution.

When building models, it advised to use loose tolerances at the beginning (e.g. {flux} and {k} of 1.0E-3). Once the model is free of defects, the tolerances should be set to their default values.

4.1.4.1 flux_tolerance

```
flux_tolerance  [<flux_tol>]
```

flux_tolerance	Floating-Point Real Number	Optional
Units: N/A		
Applicable Value(s): 1.0E-4 (default), >0		
Limitation(s): None		
Description: This card is used to specify the tolerance on the convergence of the 2-norm of the flux.		
Notes: None		

4.1.4.2 k_tolerance

```
k_tolerance  [<k_tol>]
```

k_tol	Floating-Point Real Number	Optional
Units: N/A		
Applicable Value(s): 1.0E-5 (default), >0		
Limitation(s): None		
Description: This card is used to specify the global tolerance on convergence of the eigenvalue.		

continued on next page...

k_tol, continued...

Notes: None

4.1.4.3 num_inners

```
num_inners  [<n_inner>]
```

num_inners	Integer	Optional
Units: N/A		
Applicable Value(s): 1 (default), $\geq$1		
Limitation(s): None		
Description: This card is used to specify the number of inner 1-group transport sweeps done during group sweeping every outer iteration.		
Notes: For 2-D/1-D problems, it is usually optimal for ninner to be set to 1. However, numerical instability is frequently an issue. The instability presents as an inability to converge to the desired tolerance. The solution will stagnate to within some tolerance and oscillate around that value until the maximum number of outers are reached. In this case, it is advised to use additional inner sweeps for stabilization. If so,num_inners=2 or 3 (with up_scatter=1) is a typical value.		

4.1.4.4 num_outers

```
num_outers  [<n_outer>]
```

num_outers	Integer	Optional
Units: N/A		
Applicable Value(s): 500 (default), $\geq$1		
Limitation(s): None		
Description: This card is used to specify the maximum number of outer eigenvalue iterations. If the case is not converged to within the specified tolerances, this input value is compared to the current outer iteration value. If the current outer iteration value is equal to it, the program execution will exit saying that the maximum number of iterations has been reached.		
Notes: None		

4.1.4.5 scattering

```
scattering  [<P0|TCP0|LTCP0|P1|P2|P3|P4|P5>]
```

scattering	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): TCP0 (default), P0, LTCP0, P1, P2, P3, P4, P5		
Limitation(s): None		

continued on next page...

scattering, continued...

Description: This card is used to specify the scattering order to use in the MOC calculation. The Pn notation implies the number of Legendre scattering moments to use. The higher the value of n the more moments get used and the more accurate the solution. Using more moments can dramatically increase run times. TCP0 (transport corrected P0 approximation) is the recommended option as it provides improved accuracy with the fastest run times. Not all cross section libraries include higher order scattering data, so the simulation will use the lowest of the specified scattering order and order of the data. Both the TCP0 and LTCP0 (limited transport corrected P0 approximation) options currently provide transport-corrected P0 solutions; however, LTCP0 will truncate any cross-section values encountered above 1 MeV.
Notes: None

4.1.4.6 up_scatter

```
up_scatter  [<upscatter>]
```

up_scatter	Integer	Optional
Units: N/A		
Applicable Value(s): 1 (default), $\geq$1		
Limitation(s): None		
Description: This card is used to specify the number of upscattering iterations that occur during group sweeping, i.e. between fission source iterations.		
Notes: Increasing up_scatter is one way to potentially remedy issues with numerical instability.		

4.1.5 Meshing Inputs

This section lists the inputs that pertain to the mesh section of the code. The inputs here specify the axial mesh and the pin meshes for the problem.

4.1.5.1 mesh

```
mesh [fuel | gtube]  <num_rad(:)> / <num_azi(:)>
```

mesh	Fixed Character String Followed by Two Arrays of Integers Separated by a '/'	Required
Units: N/A		
Applicable Value(s): num_rad = 3, 1 and num_azi = 1, 8, 8, 8, 12 (default), For num_rad, positive integers greater than zero. For num_azi, 1, 4, 8, 12, or 16. The length for num_rad is the number of geometric radii, and the length for num_azi is the sum of the sub-divided radii.		
Limitation(s): None		
Description: This card is used to specify the radial and azimuthal mesh for each cell. Currently two cell types are used: fuel and gtube. Cells containing fuel materials are flagged to use the fuel mesh and all other cells use the gtube meshing. For the inputs, num_rad is the number of radial subdivisions in each ring specified in the cell and num_azi is the number of azimuthal regions in each sub-divided radial ring. The last azimuthal value applies to the region outside the pin.		

continued on next page...

mesh, continued...

Notes: Currently insert, control, and detector rods have predefined mesh that cannot be overwritten. In both cases, the last entry will be used for any remaining unspecified regions. For example, if a given fuel pin has 3 radial and material regions, and the fuel mesh had a num_rad of 3,1 and num_azi of 1,4,8, then the third ring in the fuel pin would have 1 radial sub-division, and the fourth subdivided radius to the end of the pin cell would have 8 azimuthal sub-divisions, including the region outside the pincell. If the mesh is specified too finely, or rather, finer than the value for ray spacing, instabilities may occur where a ray is NOT traced through a flat source region and no flux is calculated for that region. The code will automatically adjust the azimuthal discretization if the given ray spacing value is too coarse (or because the azimuthal mesh is too fine). Another way to cause the above instability would be to specify a very large number of radial subdivisions for the first `num_rad value`. That large number being the area of the first radius divided by the first `num_rad` value would have to yield a radius that is smaller than the ray spacing. For a typical PWR fuel pin radius, the first `num_rad value` needs to be well over 100 for this problem to arise, and this number is impractical given the memory it will consume.

4.1.5.2 automesh_bounds

```
automesh_bounds  [<min> <max>]
```

automesh_bounds	Array of Floating-Point Real Numbers, Length = 2	Optional
Units: cm (default)		
Applicable Value(s): 2.0 10.0, when automeshing is enabled. (default), Positive real numbers greater than zero. The maximum value must be at least 1.0 greater than the minimum value.		
Limitation(s): None		
Description: This card specifies the minimum and maximum desired axial mesh for the auto axial meshing. Any geometry or mesh region larger than the specified value will be broken up into smaller mesh regions that have a height between the maximum and minimum values. Any geometry or mesh region smaller than the specified value will be homogenized and added to a neighboring mesh region until the value is above the minimum and below the maximum.		
Notes: The region where these values are applied is specified by the `meshing_method` card. This card is ignored when the `useraxialmesh` and `matbound` method is specified. It should also be noted that specifying min and max values that are close together will most likely result in more axial homogenization than may be desired by the user. It would mean that most of the material interfaces will be homogenzied to some degree. Also, this routine in no way optimizes the axial meshing for a given problem. It is primarily designed to reduce user burden from specifying a typically troublesome input parameter. It is best suited for problems with a large number of planes that vary in thickness. It is also useful for setting a problem up, if the user is unsure about the axial discretization. Using this card will save time spent on recalculating values whenever the axial mesh needs to be adjusted.		

4.1.5.3 meshing_method

```
meshing_method  [<useraxialmesh|matbound|nonfuel|all>]
```

meshing_method	Fixed Character String	Optional
Units: N/A		

continued on next page...

meshing_method, continued...

Applicable Value(s): useraxialmesh (axial_mesh card present) *or* matbound (axial_mesh card not present) (default), nonfuel, all • useraxialmesh — Requires the use of the axial mesh card and no auto meshing is performed in this instance. This option will not use the values specified by the automesh_bounds since it does not do any automeshing. • matbound — Calculates the axial mesh just at the axial material boundaries of the problem and uses the axial_edit_bounds as the mesh within the fuel regions. No further meshing is performed. This option will not use the values specified by the automesh_bounds since it does not do any automeshing. • nonfuel — Will take the material boundaries and automesh the regions below and above the fuel. The minimum and maximum bounds (or default values) specified by the automesh_bounds will be used to determine the sizing. • all — Will take the material boundaries and automesh all regions. The minimum and maximum bounds (or default values) specified by the automesh_bounds will be used to determine the sizing. When using the all option, fuel regions will not be homogenized with non-fuel regions. Homogenization will only occur within those regions.
Limitation(s): Must be sed in conjunction with the axial_edit_bounds card in the EDIT block of the VERA input when the option is not useraxialmesh. This data is required to set up the axial mesh for every input option except the useraxialmesh where it is separately specified.
Description: This card specifies the type of axial meshing to be used. If this card is not present, the method will default to useraxialmesh if the axial_mesh card is present or it will default to matbound if the axial_mesh card is not present.
Notes: When using the useraxialmesh option, it is possible to specify a mesh that does not conform or align with the problem's geometry. Warnings will be printed to the log file stating that the mesh does not match the geometry boundaries and those regions will be homogenzied.

4.1.5.4 axial_mesh

```
axial_mesh  [<plane_thickness(:)>]
```

axial_mesh	Array of Floating-Point Real Numbers, Length = User Specified	Optional
Units: cm (default)		
Applicable Value(s): N/A (default), Array of positive real numbers.		
Limitation(s): The sum of the values specified within this card must be equal to the total geometric height of the problem.		
Description: This card is used to specify the axial mesh used in the 2-D/1-D simulation. The input is the thickness of each axial section the user wishes to model. This card is optional if the meshing_method card specifies an option other than useraxialmesh. If the meshing_method is useraxialmesh, then it is required.		
Notes: If the array of axial meshes sums to less than the problem height, the geometry at the top will be truncated. If it sums to more than the problem height, the top geometry will be extended all the way to the upper mesh height. Therefore, it is very important to make sure the axial mesh is specified in accordance with the geometry.		

4.1.5.5 crud_mesh

```
crud_mesh <max_rad> <num_rad>
```

crud_mesh	One Floating-Point Real and One Integer	Optional
Units: microns (default)		
Applicable Value(s): N/A (default), Positive real numbers for max_rad and integers greater than 0 for num_rad. The max_rad is the maximum thickness of the outermost CRUD region in microns and num_rad is the number of radial subdivisions in the CRUD region.		
Limitation(s): None		
Description: This card is used to specify the radial mesh that is added for each cell to account for CRUD build-up on the surface of the fuel pins.		
Notes: None		

4.1.5.6 grid_treatment

```
grid_treatment [<homogenize|equal_mass|equal_thickness>]
```

grid_treatment	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): homogenize (default), equal_mass, equal_thickness • homogenize — will take the mass specified in the grid card, calculate the moderator volume of the lattice where the grid is located, and use the two values to compute the density of the material. This option applies the grid material uniformly throughout the lattice. • equal_thickness — uses the grid mass and the corresponding grid material density to compute the total grid volume for that lattice. The volume is then used to determine the thickness the grid would be within each pincell, and is modeled as an additional rectangular mesh around the perimeter of each pin cell in the lattice. • equal_mass — similar to the equal_thickness option, except that the thickness of the grid in each pin cell is changed throughout the lattice so that every pin cell has the same grid material mass in it.		
Limitation(s): For grids with a large mass that fall in a narrow (axially) lattice, there is a possibility that the grid will intersect one or more pins for the equal_thickness and equal_mass options. If this event occurs, MPACT will raise an error, and the user will need to change the axial meshing options, change the geometry of the lattice, or simply use the homogenize option for the grid_treatment card.		
Description: This card is used to indicate the method of applying the grid structure in a lattice on the mesh.		
Notes: None		

4.1.6 Quadrature Set Inputs

This section lists the inputs that pertain to the quad_set section of the input file. The inputs in this section control the characteristic "ray" discretization and number of angles for the transport solution.

4.1.6.1 quad_type

```
quad_type  [<quad_type>]
```

quad_type	Fixed Character String	Required
Units: N/A		

continued on next page...

quad_type, continued...

Applicable Value(s): None (default), listed below

Quadrature Name	Type	Order	Order Θ
CHEBYSHEV-CHEBYSHEV	Product	integers > 0	integers > 0
CHEBYSHEV-GAUSS	Product	integers > 0	integers > 0
CHEBYSHEV-BICKLEY	Product	integers > 0	1, 2, 3, or 4
CHEBYSHEV-YAMAMOTO	Product	integers > 0	1, 2, or 3
LEVEL-SYMMETRIC	General	even integers in [2,16]	N/A
QUADRUPLE-RANGE	Product	integers in [1,37]	integers in [1,18]

Limitation(s): None

Description: This card is used to specify the name of the angular quadrature to use for determining the angles at which the rays are traced throughout the problem.

Notes: None

4.1.6.2 quad_type

```
shield_quad_type  [<quad_type>]
```

quad_type	Fixed Character String	Required

Units: N/A

Applicable Value(s): None (default), listed below

Quadrature Name	Type	Order	Order Θ
CHEBYSHEV-CHEBYSHEV	Product	integers > 0	integers > 0
CHEBYSHEV-GAUSS	Product	integers > 0	integers > 0
CHEBYSHEV-BICKLEY	Product	integers > 0	1, 2, 3, or 4
CHEBYSHEV-YAMAMOTO	Product	integers > 0	1, 2, or 3
LEVEL-SYMMETRIC	General	even integers in [2,16]	N/A
QUADRUPLE-RANGE	Product	integers in [1,37]	integers in [1,18]

Limitation(s): None

Description: This card is used to specify the name of the angular quadrature to use for determining the angles at which the rays are traced throughout the problem for the shielding calculation.

Notes: None

4.1.6.3 azimuthals_octant

```
azimuthals_octant  [<num_azi>]
```

azimuthals_octant	Integer	Required

Units: N/A

Applicable Value(s): None (default), Column Order in the above table

Limitation(s): None

Description: This card is used to specify the number of azimuthal angles per octant and corresponds to the Order column in the table in quad_type card.

Notes: None

4.1.6.4 polars_octant

```
polars_octant  [<num_pol>]
```

polars_octant	Integer	Required
Units: N/A		
Applicable Value(s): None (default), Column Order Θ in the above table		
Limitation(s): None		
Description: This card is used to specify the number of polar angles per octant and corresponds to the Order Θ column in the quadrature table specified in quad_type card. Note the number of polar angles may be limited by the quadrature type used. Also, any non-product quadrature types will not use this input card (i.e., in the only applicable case LEVEL-SYMMETRIC).		
Notes: None		

4.1.6.5 shield_polars_octant

```
shield_polars_octant  [<num_pol>]
```

polars_octant	Integer	Required
Units: N/A		
Applicable Value(s): None (default), Column Order Θ in the above table		
Limitation(s): None		
Description: This card is used to specify the number of polar angles per octant for the shielding calculation and corresponds to the Order Θ column in the quadrature table specified in quad_type card. Note the number of polar angles may be limited by the quadrature type used. Also, any non-product quadrature types will not use this input card (i.e., in the only applicable case LEVEL-SYMMETRIC).		
Notes: None		

4.1.6.6 shield_azimuthals_octant

```
shield_azimuthals_octant  [<num_azi>]
```

shield_azimuthals_octant	Integer	Required
Units: N/A		
Applicable Value(s): None (default), Column Order in the above table		
Limitation(s): None		
Description: This card is used to specify the number of azimuthal angles per octant for the shielding sweeper and corresponds to the Order column in the table in quad_type card.		
Notes: None		

4.1.7 XS Library Inputs

This section lists the inputs that pertain to the xs_library section of code. The inputs here specify the name of the cross section library and its format, as well as the resonance shielding parameters.

4.1.7.1 mats_file

```
mats_file  [<filename>]
```

mats_file	Free-Form Character String, Max. Length = 200	Optional
Units: N/A		
Applicable Value(s): No default value (default), filename of a HDF5 material database file		
Limitation(s): None		
Description: This card is used to specify the name of the HDF5 material database file. This file is used to overwrite the isotopic and weight fraction values for default VERA material.		
Notes: Marked for deprecation, do not use!		

4.1.7.2 mod_mat

```
mod_mat  <name>
```

mod_mat	Free-Form Character String, Max. Length = 200	Optional
Units: N/A		
Applicable Value(s): mod (default), any user-defined name of the moderator material.		
Limitation(s): None		
Description: This card is used to rename the moderator material.		
Notes: None		

4.1.7.3 subgroup_set

```
subgroup_set  [<set>]
```

subgroup_set	Integer	Optional
Units: N/A		
Applicable Value(s): 4 (default), integers 1 through 9		
Limitation(s): The shield_method must be set to subgroup. ESSM ignores the subgroup_set option.		
Description: This card is used to specify the subgroup set.		
Notes: In most cases, 4 (the default) should be used. This option finds a good balance on accuracy and computing time. In general, the numbering is from 1 to 9, with 1 being the simplest set (fast), and 9 being the most explicit set (slow).		

4.1.7.4 cat_onegroup

```
cat_onegroup  [<1g_categories(:)>]
```

cat_onegroup	Array of Integers, Length = User Specified	Optional
Units: N/A		
Applicable Value(s): 3(if subgroup_set = 4) (default), any integer number		
Limitation(s): The shield_method must be set to subgroup. ESSM ignores the cat_onegroup option.		
Description: This card is used to specify the categories that use one-group subgroup.		
Notes: The user can specify the categories that will use one-group subgroup treatment, which means a fast and approximate subgroup calculation in that category. If subgroup_set = 4 (default), the default value of this option is 3 (clad category), otherwise no default category will be assigned to one-group subgroup unless user speficies. User can also specify zero or a negative integer number to use MG-subgroup for all categories.		

4.1.7.5 xs_filename

```
xs_filename  [<filename>]
```

xs_filename	Free-Form Character String, Max. Length = 200	Required
Units: N/A		
Applicable Value(s): No default value (default), filename of a supported cross section library		
Limitation(s): None		
Description: This card is used to specify the name of the cross-section file to use.		
Notes: None		

4.1.7.6 xs_shielder

```
xs_shielder  [<true|false|t|f>]
```

xs_shielder	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): true (default), false,t,f		
Limitation(s): None		
Description: This card is used to specify whether to shield the cross sections or not: ture–enabled, false–disabled		
Notes: If shielder is disabled, the infinite-dilute cross sections for the resonance energy groups are used.		

4.1.7.7 shield_method

```
shield_method  [<subgroup|essm>]
```

shield_method	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): subgroup (default), essm		
Limitation(s): The xs_shielder card must be enabled (default) in order to enable this card, otherwise unshielded cross section (infinite-dilute) will be used.		
Description: This card is used to specify the method used to shield the cross sections.		
Notes: In general, subgroup is slower than essm (by a factor of 2 to 5 depending on the subgroup_set option). However, subgroup method has a few advantages over ESSM, such as a better representation of distributed self-shielding within the fuel and the resonance category treatment (resonance isotopes are grouped into categories). Therefore, subgroup method is an option with better accuracy in the current version.		

4.1.7.8 xs_type

```
xs_type [<NONE|ORNL>]
```

xs_type	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): NONE (default), ORNL		
Limitation(s): None		
Description: This card is used to specify the type of cross-section file to use.		
Notes: None		

4.1.7.9 quasi_1D

```
quasi_1D  [<t|f|true|false>]
```

quasi_1D	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): false (default), true,t,f		
Limitation(s): None		
Description: This card is used to specify whether to perform the quasi-1D slowing-down correction for self-shielding calculation. Currently, this option can only be toggled on with essm.		
Notes: None		

4.1.7.10 ce_filename

```
ce_filename  [<filename>]
```

ce_filename	Free-Form Character String, Max. Length = 200	Optional
Units: N/A		
Applicable Value(s): No default value (default), filename of an indexing file for CE library		
Limitation(s): None		
Description: This card is used to specify the name of the indexing file of continuous-energy cross-section library to be used when quasi_1D is toggled on.		
Notes: None		

4.1.8 Parallel Environment Inputs

This section lists the inputs that pertain to the parallelization of the problem to be solved. Parallelization means that multiple processes are used to calculate the solution faster than with one process. The input options below allow the user to choose the methods of parallelization, and to quantitatively control how the problem is parallelized.

4.1.8.1 par_method

```
par_method  [<DEFAULT|ASSEMBLY|EXPLICITFILE>]
```

par_method	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): DEFAULT (default), EXPLICITFILE		
Limitation(s): The EXPLICITFILE option may be used only if the user has created a partition file. For a description of the partition file, see the input option par_file.		

continued on next page...

par_method, continued...

Description: This card is used to specify the method of parallel decomposition. • DEFAULT — The parallelization is specified by three input options listed below: num_angle, num_space, num_threads. The meanings of these input options are explained below. DEFAULT is the simpler method for parallelizing the problem, and is recommended for most users. • ASSEMBLY — The parallelization scheme for decomposing a problem spatially. The problem will be decomposed radially first, and if there are more processors, will then attempt to parallelize the problem axially. This process is done automatically, and only requires the user to specify the number of spatial processors available in the num_space card described below. It is the recommended method for large problems. • EXPLICITFILE — For more advanced users who are running large problems, using the EXPLICITFILE option may enable the user to parallelize the problem more effectively. For a description of the EXPLICITFILE method, see the input option par_file.
Notes: None

4.1.8.2 num_angle

```
num_angle  [<n_angle>]
```

num_angle	Integer	Optional
Units: N/A		
Applicable Value(s): 1 (default), Integer greater than 0 and less than the number of CPU cores.		
Limitation(s): Specifying a value greater than 2*azimuthals_octant will cause an exception error.		
Description: This input options specifies the number of parallel partitions used to decompose the problem based on the azimuthal angle (i.e. ray directions in the x-y plane). To get the 2D MOC solution for a single x-y plane, rays are traced through the domain in multiple azimuthal directions, as specified by the user in the option azimuthals_octant (the user should note that the terms octant and quadrant are interchangable in the context of azimuthal angles). The azimuthal angles are divided into num_angle groups, and each groups is assigned to a parallel partition (i.e. process). If spatial decomposition is used in the same problem, then each spatial decomposition region is copied to num_angle partitions. Therefore, the total number of parallel partitions is num_angle*num_space.		
Notes: The user is cautioned against using too many processes to decompose the problem. Due to the increase in inter-process communication with increased parallel decomposition, excessive parallelization will not yield speedup of the solution. The proper amount of paralleization will have to be determined on a case-by-case basis.		

4.1.8.3 num_space

```
num_space  [<n_space>]
```

num_space	Integer	Optional
Units: N/A		
Applicable Value(s): 1 (default), Integer greater than 0 and less than the number of CPU cores.		
Limitation(s): None		

continued on next page...

num_space, continued...

Description: This card is used to specify the number of spatial decomposition regions used in a parallel execution step. this value can be: 1. a subset of the number of planes in the model, 2. the total number of planes, or 3. a product of all of the planes and any number of radial regions comprised of groups of quarter assemblies. The ability to decompose a problem by planes can be used with the DEFAULT partition method. Any partition that decomposes the problem radially requires the EXPLICITFILE partition method.
Notes: See description of card num_angle for explanation of using spatial and angular decomposition in conjunction.

4.1.8.4 num_threads

```
num_threads  [<n_threads>]
```

num_threads	Integer	Optional
Units: N/A		
Applicable Value(s): 1 (default), Integer greater than 0 and less than the number of CPU cores.		
Limitation(s): None		
Description: This card is used to specify the number of threads used in parallel execution. The number of threads specified are used only during the MOC transport sweep. For a given ray direction (i.e. angle), threads are used to sweep multiple rays in parallel.		
Notes: It is recommended that num_angle*num_space*num_threads does not exceed the total number of physical CPU cores. MPACT will still run if the user exceeds this limit, but the parallel performance will be degraded.		

4.1.8.5 par_file

```
par_file  [<filename>]
```

par_file	Free-Form Character String	Optional
Units: N/A		
Applicable Value(s): partition.txt (default)		
Limitation(s): No comments are allowed in the file.		

continued on next page...

`par_file`, continued...

Description: This card is used to specify the parallel decomposition file if `EXPLICITFILE` is used. This is an advanced feature that is not recommended for most users. The MPACT domain is broken into a regular grid of ray trace modules; the partition file allows the user to specify the spatial decomposition of the domain by listing the ray trace modules in each spatial partition via their (x,y,z) indices (this is explained more in the following paragraphs). The partition file also allows the user to decompose the MPACT domain radially, which is not possible with the `DEFAULT` partition method.
The file structure itself has two header lines followed by the specification of the radial partition regions.
The first line has 3 values, the first is the number of MPACT ray trace modules in the x direction, the second is the number of ray trace modules in the y direction, and the third is the number of axial planes in the model.
The second line also has 3 values. The first two pertain specifically to how MPACT partitions ray trace modules in space, and these values should always be 0 and 1 respectively. The third value should be the number of radial partitions being subsequently specified.
The following lines should describe all radial partition regions for the problem including any regions that will be used with a jagged core. The input for each line is 6 integers. The first pair of integers are the starting and stopping module indices in the x direction, the second pair are the starting and stopping module indices in the y direction, and the last pair is for the z direction, but they are ignored currently and all radial partitions are assumed to be the same for each axial plane. The coordinate system point of origin when specifying the starting and stopping indices is the lower left (south-west) corner of the module. When specifying the starting and stopping indices, it is important to note that these are not necessarily the assembly positions. Typically, in the case of modeling a full reactor, the ray trace modules represent a quarter of an assembly. In this case, the number of ray trace modules in a given direction will be about twice the number of assemblies in that direction.

Notes: If the core is jagged, additional attention is required to keep track of the actual number of processors being used by MPACT. Even though the non-existent assemblies are "partitioned" in the explicit file, nothing there will be run. So the user cannot simply take the third value from the second line and multiply it by the third value from the first line to get the total number of spatial partitions for this case. In the example below, the third value in the second line must have the number of "jagged" partitions subtracted from it. In this case, the actual number of processors per plane becomes 49 - 8 = 41. That number can then be multiplied by the number of planes to get 2378 processors, which should be input into the `num_space` card.
Also, it may be unclear to the user how many planes will be created in MPACT before the case is run. The output file has a summary of the axial mesh information, including the total number of planes. If the case crashes when using the partition file, check that the number of planes specified matches the value in the output file.

A sample explicit file is given below:

```
17 17 58
0 1 49
 1  3  1  2  1  1
 4  6  1  2  1  1
 7  9  1  2  1  1
 1  3  3  4  1  1
 4  6  3  4  1  1
 7  9  3  4  1  1
10 11  3  4  1  1
12 13  3  4  1  1
 1  3  5  6  1  1
 4  6  5  6  1  1
 7  9  5  6  1  1
10 11  5  6  1  1
12 13  5  6  1  1
14 15  5  6  1  1
 1  3  7  8  1  1
 4  6  7  8  1  1
 7  9  7  8  1  1
10 11  7  8  1  1
12 13  7  8  1  1
14 15  7  8  1  1
 1  3  9 11  1  1
```

```
  4  6  9 11  1  1
  7  9  9 11  1  1
 10 11  9 11  1  1
 12 13  9 11  1  1
 14 15  9 11  1  1
 16 17  9 11  1  1
  1  3 12 14  1  1
  4  6 12 14  1  1
  7  9 12 14  1  1
 10 11 12 14  1  1
 12 13 12 14  1  1
 14 15 12 14  1  1
 16 17 12 14  1  1
  1  3 15 17  1  1
  4  6 15 17  1  1
  7  9 15 17  1  1
 10 11 15 17  1  1
 12 13 15 17  1  1
 14 15 15 17  1  1
 16 17 15 17  1  1
 10 11  1  2  1  1
 12 13  1  2  1  1
 14 15  1  2  1  1
 16 17  1  2  1  1
 14 15  3  4  1  1
 16 17  3  4  1  1
 16 17  5  6  1  1
 16 17  7  8  1  1
```

The radial partition map specified by the above example will look like the following:

```
 34 34 34 35 35 35 36 36 36 37 37 38 38 39 39 40 40 !17
 34 34 34 35 35 35 36 36 36 37 37 38 38 39 39 40 40 !16
 34 34 34 35 35 35 36 36 36 37 37 38 38 39 39 40 40 !15
 27 27 27 28 28 28 29 29 29 30 30 31 31 32 32 33 33 !14
 27 27 27 28 28 28 29 29 29 30 30 31 31 32 32 33 33 !13
 27 27 27 28 28 28 29 29 29 30 30 31 31 32 32 33 33 !12
 20 20 20 21 21 21 22 22 22 23 23 24 24 25 25 26 26 !11
 20 20 20 21 21 21 22 22 22 23 23 24 24 25 25 26 26 !10
 20 20 20 21 21 21 22 22 22 23 23 24 24 25 25 26 26 !9
 14 14 14 15 15 15 16 16 16 17 17 18 18 19 19 48 48 !8
 14 14 14 15 15 15 16 16 16 17 17 18 18 19 19 48 48 !7
  8  8  8  9  9  9 10 10 10 11 11 12 12 13 13 47 47 !6
  8  8  8  9  9  9 10 10 10 11 11 12 12 13 13 47 47 !5
  3  3  3  4  4  4  5  5  5  6  6  7  7 45 45 46 46 !4
  3  3  3  4  4  4  5  5  5  6  6  7  7 45 45 46 46 !3
  0  0  0  1  1  1  2  2  2 41 41 42 42 43 43 44 44 !2
  0  0  0  1  1  1  2  2  2 41 41 42 42 43 43 44 44 !1

! 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
```

4.1.9 Depletion Inputs

This section lists the inputs that pertain to the depletion section of the code. The depletion kernel solves the time-dependent Bateman equation to track isotope concentration change during the reactor operation. The inputs below define the solvers and the methods that can be used.

4.1.9.1 dep_edit

```
dep_edit  [<true|false>]
```

dep_edit	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): `true` (default), `false`		
Limitation(s): None		
Description: This card is used to specify if the depletion `Isum` and `Pnum` files are written, which print the pin-wise averaged isotope number densities. `Isum` prints the isotope summary file with isotopes tracked in XSMesh and `Pnum` file prints the particle number density file with all isotopes in the depletion library.		
Notes: The option has excessive memory requirements and is not advised for general usage. Only use when absolutely necessary.		

4.1.9.2 dep_filename

```
dep_filename  [<filename>]
```

dep_filename	Free-Form Character String, Max. Length = 200	Required
Units: N/A		
Applicable Value(s): No default value (default), filename of a supported cross section library		
Limitation(s): The format of this file should be consistent with the standard MPACT depletion library file `MPACT.dpl`.		
Description: This card is used to specify the depletion file to use, which provides all the data required in addition to the data in the transport library for depletion calculation.		
Notes: None		

4.1.9.3 dep_kernel

```
dep_kernel  [<internal|origen>]
```

dep_kernel	Fixed Character String	Required
Units: N/A		
Applicable Value(s): `internal`(MPACT's internal depletion kernel) (default), `origen`(coupled `origen` kernel)		
Limitation(s): None		
Description: This card is used to specify the depletion kernel to use. The MPACT internal depletion kernel is based on the same methodology as `origen`, but uses simplified depletion chains and runs faster than `origen`.		
Notes: None		

4.1.9.4 depl_time_method

```
depl_time_method  [<p-c|semip-c|postcorrector>]
```

depl_time_method	Fixed Character String	Required
Units: N/A		
Applicable Value(s): `p-c`(predictor-corrector) (default), `semip-c`(semi-predictor-corrector) or `postcorrector`(semi-predictor-corrector-post-corrector)		
Limitation(s): None		

continued on next page...

depl_time_method, continued...

Description: This card is used to specify the time stepping method in depletion. The p-c method computes a predicted nuclide concentration based on the steady state flux condition at the beginning of time step, which is then averaged with the corrected nuclide concentration based on the steady state flux condition at the end of time step. Two steady-state eigenvlaue calcualtions are performed for each depletion time step. The p-c method is a well demonstrated method and it can be used for large time steps. The semip-c method simplifies the p-c method by skipping the second steady-state eigenvalue calculation and thus becomes more efficient in small time step depletion calculation. The postcorrector method is identical to semip-c method with the exception that the number densities used for the beginning of times step steady-state eigenvalue calculation are "post-corrected" so that they more closely representthe averaged number densities of the full p-c method. This allows for accuracy coparable to the full p-c method while still skipping the second steady-state eigenvalue calculation
Notes: The semip-c method can result in an inconsistency when restarting. However, the differences that arise from a semip-c restart are smaller in magnitude than the differences between semip-c and p-c. The inconsistency in the semip-c restart arises from an extra flux calculation that occurs on restart, so presumably the difference results in a more accurate solution.

4.1.9.5 dep_substep

```
dep_substep  [<nsubstep>]
```

dep_substep	Integer	Optional
Units: N/A		
Applicable Value(s): 1 (default), Positive integers greater than 0		
Limitation(s): None		
Description: This card is used to specify the number of substeps used in depletion. The substep method is applied to perform multiple depletion calculations between transport calculations. Since the depletion calculation typically takes less time than the transport calculation, this will often save computational time.		
Notes: Two or three substeps per depletion step is recommended.		

4.1.10 Thermal-Hydraulic Inputs

This section lists the inputs that pertain to the TH section of code. The inputs here specify the parameters for the TH solution.

4.1.10.1 coupling_method

```
coupling_method [<simplified|ctf|ctf_external|none>]
```

coupling_method	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): simplified (if not configured with COBRA-TF) *or* ctf (if configured with COBRA-TF) (default), ctf_external, none The simplified option uses MPACT's internal TH solver. The ctf option internally couples COBRA-TF to MPACT, and ctf_external couples MPACT and COBRA-TF through the lime interface. The none option will use parameters from the STATE block: fuel temperatures will be constant and equal to tfuel, moderator temperatures will be constant and equal to tinlet, and moderator densities will be constant and equal to modden.		
Limitation(s): The feedback card in the STATE block must be set to on for any of the three TH coupling methods		
Description: This card is used to indicate which TH coupling method should be used.		

continued on next page...

coupling_method, continued...

Notes: For either the ctf or ctf_external options, MPACT must be configured with COBRA-TF. The internal option may regardless of whether MPACT was configured with COBRA-TF or not.

4.1.10.2 shielder_th

```
shielder_th [<shield_max_outers> <shield_min_dT> <shield_min_drho>]
```

shielder_th	Integer, Floating-Point Real Number, Floating-Point Real Number	Optional
Units: {unitless, K, g/cm^3} (default)		
Applicable Value(s): 4, 25.0, 0.005 (default), or any positive integer and any 2 positive real numbers The first input is the maximum number of outer iterations for which MPACT will perform cross-section shielding calculations following a TH update. The second input is the minimum change in temperature for which MPACT will perform cross-section shielding calculations following a TH update. The third and last input is the minimum change in moderator density for which MPACT will perform cross-section shielding calculations following a TH update.		
Limitation(s): If the xs_shielder card is set to f or false, this card does nothing, since cross-section shielding calculations will never be performed.		
Description: This card is used to control how many cross-section shielding calculations are performed when using TH feedback. It sets a maximum number of iterations with shielding calculations, and also sets parameters to stop the shielding calculations earlier if the TH feedback effects on temperature and moderator density are small enough.		
Notes: If multiple state points are performed in the calculation, the counter for the <shield_max_outers> input is reset for each state point. If the xs_shielder card is not set to f or false, shielding calculations will always be performed on the first iteration.		

4.1.10.3 outers_per_TH

```
outers_per_TH [<outers_per_TH>]
```

outers_per_TH	Integer	Optional
Units: N/A		
Applicable Value(s): 1 (default), or any positive integer		
Limitation(s): None		
Description: This card is used to indicate how many outer iterations MPACT should perform before performing an additional TH update.		
Notes: None		

4.1.10.4 average_ftemp

```
average_ftemp [<true|false>]
```

average_ftemp	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): true (default), false		
Limitation(s): None		

continued on next page...

average_ftemp, continued...

Description: If true, this card applies a volume-avergaed fuel temperature to each fuel pin. If false, it applies a radially dependent fuel temperature to each fuel pin.
Notes: None

4.1.10.5 ctf_basename

```
ctf_basename [<ctf_basename>]
```

ctf_basename	Free-Form Character String, Max. Length = 200	Optional
Units: N/A		
Applicable Value(s): deck (when COBRA-TF is run in serial) *or* pdeck (when COBRA-TF is run in parallel) (default), Any filename base for valid COBRA-TF input decks.		
Limitation(s): Filename must have ".inp" extension.		
Description: This card is used to indicate the "basename" of the CTF input files for CTF coupling. The "basename" is the section of the CTF input filename(s) without any extensions.		
Notes: Absolute or relative paths to the file are both acceptable.		

4.1.10.6 sth_dhfrac

```
sth_dhfrac [<sth_dhfrac>]
```

sth_dhfrac	Floating-Point Real Number	Optional
Units: N/A		
Applicable Value(s): 0.02 (default), 0.0–1.0		
Limitation(s): It is ignored if feedback is off or if coupling with COBRA-TF is being used.		
Description: This card is used to set the fraction of the power which is directly deposited in the moderator in internal TH calculations.		
Notes: None		

4.1.10.7 sth_hgap

```
sth_hgap [<sth_hgap>]
```

sth_hgap	Floating-Point Real Number	Optional
Units: $W/m^2 \cdot K$ (default)		
Applicable Value(s): 4500.0 (default), or any positive real number		
Limitation(s): It is ignored if feedback is off or if coupling with COBRA-TF is being used.		
Description: This card is used to set the gap conductance value for internal TH calculations.		
Notes: Typical values range from 1000 (very low) to 10000 (very high).		

4.1.10.8 sth_channeltype

This card is used to set the size of the region over which average moderator conditions will be applied. Acceptably values are assembly, node (quarter assembly), or pin (flow channel between four fuel pins).

```
sth_channeltype [<assembly|node|pin>]
```

4.1.10.9 UMVERA_sth_avgpin

This card is used to determine whether an average pin is used for each region or if fuel conduction calculations are done for each pin uniquely. If true, a representative pin will be used. If sth_channeltype is set to pin, this card is ignored.

```
sth_avgpin [<true|false>]
```

4.1.10.10 temptable_filename

```
temptable_filename [<temptable_filename>]
```

temptable_filename	Free-Form Character String, Max. Length = 200	Optional
Units: N/A		
Applicable Value(s): N/A (default), Filename of any valid fuel temperature table file		
Limitation(s): If the card is present, temperature tables in the named file will be used to calculate fuel temperatures instead of the internal conduction solvers or COBRA-TF. If this card is not present, then internal or COBRA-TF solvers are used.		
Description: This card is used to indicate the name of the file containing the temperature tables.		
Notes: Temperature tables contain fuel temperature values as functions of power and burnup. When depleting, the thermal properties of the fuel change significantly. Internal TH and COBRA-TF do not know how these properties change when depleting, so temperature tables can be used to more accurately perform TH calculations during depletion simulations using tabulated data rather than fuel conduction solvers.		