ANL/NSE-18/10

Argonne NATIONAL LABORATORY

PROTEUS-MOC User Manual

Revision 0

Nuclear Science and Engineering Division

ANL/NSE-18/10

PROTEUS-MOC User Manual

Revision 0

prepared by

Yeon Sang Jung, Changho Lee, and Micheal A. Smith
Nuclear and Science Engineering Division, Argonne National Laboratory

September 30, 2018

ABSTRACT

The PROTEUS-MOC code is a three-dimensional (3D) neutron transport code based on the finite element mesh and the method of characteristics (MOC) which combines a 2D MOC with the discontinuous Galerkin finite element method for the axial direction. Thus, for PROTEUS-MOC, a 3D geometry is represented with by the implicit extrusion of a single 2D planar geometry with different material assignments allowable on each plane. The code requires four input files for a steady-state calculation without the thermal feedback: a driver input, a mesh input, a cross section input, and a material assignment input. With thermal feedback, a T/H input file is needed. For kinetics, a kinetics driver input file is required. The PROTEUS-MOC software produces a text output file as well as a data file which includes integral and average quantities such on fluxes, powers, temperatures, densities, etc. The detailed output file can be converted to the VTK format using the post processing code so that the outputs can be visualized using a visualization software such as VisIt.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

PROTEUS-MOC [1] is the one of the 3D transport solver option available in PROTEUS. PROTEUS-MOC can provide a faithful 3D transport solution for 3D heterogeneous configuration that can be represented using an extruded geometry model. This document describes the required input files for a PROTEUS-MOC calculation, which helps a user perform a neutronics simulation with proper computational options. In order to perform a MOC calculation, a user needs to prepare the following four input files:

- *Driver input file*
- *Cross section file (*.ISOTXS, *.anlxs)*
- *Material assignment file (*.assignment)*
- *T/H input file (*.th)*
- *Mesh file (*. ascii)*
- *T/H assignment file (*.THassignment)*

PROTEUS-MOC adopts the cross section and material file formats of PROTEUS-SN. The detailed description of these input files can be found in the PROTEUS-SN manual [2]. This document provides only brief descriptions for first four input files.

1.1 Input File Summary

Four input files are required to perform PROTEUS calculations (four other files are relevant only for certain simulations).

a. Driver input file (*.inp)
b. Multi-group cross section file (ISOTXS, ANLXS)
c. Mesh file (ASCII, PNTMESH)
d. Material assignment file (ASSIGNMENT)
e. (Optional) T/H assignment file (*.THassignment)
f. (Optional) T/H property input (*.th)
g. (Optional) CMFD mesh input (*.ascii)
h. (Kinetics only) kinetics input file (*.inp)

The driver input file is a keyword-based free format text file that contains the simulation parameters such as angular cubature, parallelization, convergence criteria, and iteration limits.

The multi-group cross section file is a formatted file containing the multigroup cross sections for all the isotopes or materials of interest. Multigroup cross sections are provided in either ANLXS or ISOTXS formatted files.

The mesh file contains the geometry and boundary condition details of the problem to be executed and is readable by the mesh generation toolkit [3]. The results of PROTEUS-MOC can thus be visualized using the VTK export option.

To define compositions and assign them to the geometrical regions we utilize the material assignment format that is already part of the PROTEUS framework.

The PROTEUS-MOC has a CMFD acceleration scheme and requires a user-generated coarse mesh file which geometrically should be consistent with the fine mesh as illustrated in Figure 1. It can be generated using the mesh tools such as CUBIT [4] or the PROTEUS mesh toolkit [3].

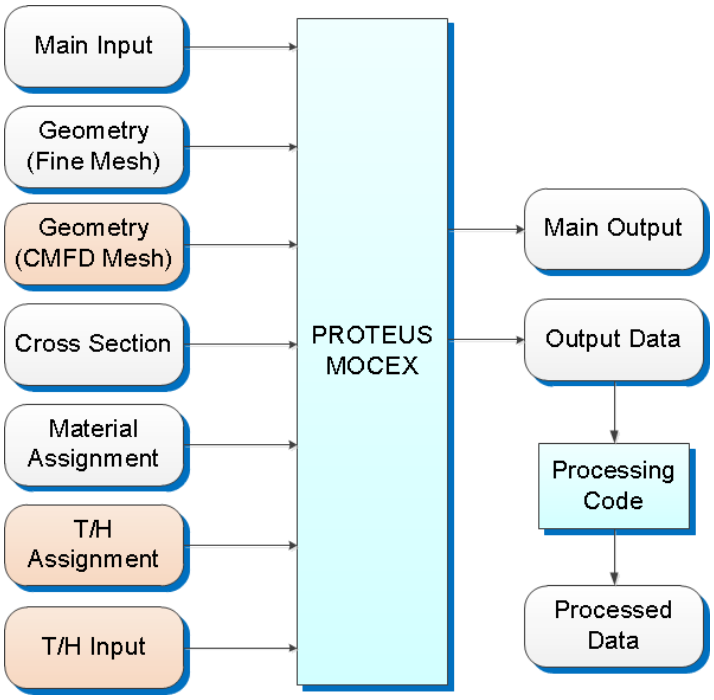

Figure 1. Input and Output Files of PROTEUS-MOC

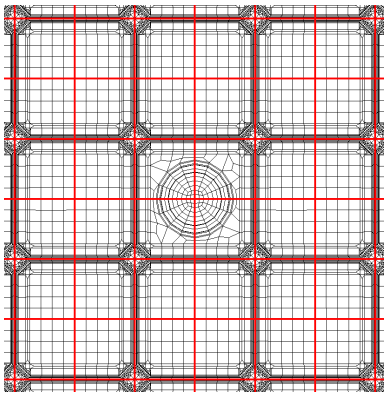

Figure 2. Illustration of Coarse Mesh Structure for CMFD Acceleration

The T/H assignment file is a keyword-based free format text file that provides the region mapping for the T/H condition updates such as fuel temperatures. This file is needed only when the T/H feedback option is turned on.

The kinetics input file is relevant only for kinetics simulations. It describes time steps to be taken, including material assignments for each time step, delay cross section data and fixed source data to be used, if applicable.

1.2 Output File Summary

PROTEUS-MOC produces a text output file and MOCEX_SOLUTION.bin. The text output file contains an echo of the inputs, eigenvalue iteration history, and timing summaries. The MOCEX_SOLUTION.bin file includes a mesh-to-composition map and mesh-averaged quantities such as power densities, neutron fluxes, etc... Using the output post-processing tool, this solution file can be processed to plot the solutions. It can also extract pin- or assembly-wise quantities from the PROTEUS-MOC output. Examples of these capabilities are shown in Figure 3 and Figure 4.

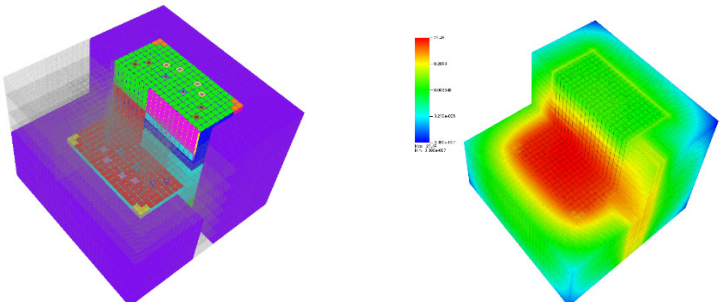

Figure 3. Example of Visualization Using Post-processing Tool

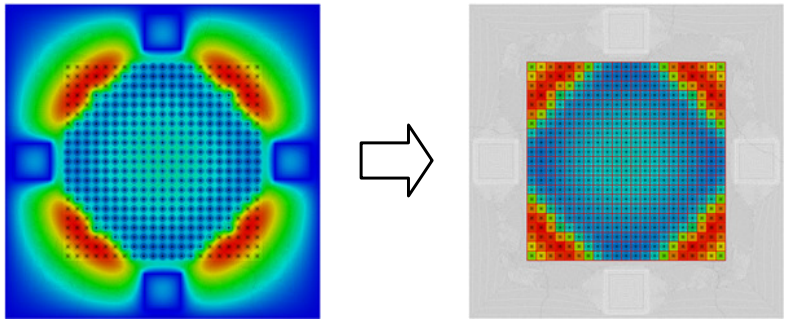

Figure 4. Example of Pin-wise Flux Editing Using Post-Processing Tool.

2. Acquiring and Installing the Code

This chapter briefly describes how to obtain and install PROTEUS-MOC. It lists the external library dependencies and also recommended compilers. For more specific (and up-to-date) compilation information including configuration options of the various packages, one should consult the installation documentation included with the distribution.

2.1 Acquiring the Code

The PROTEUS-MOC source code is export-controlled and currently obtained through Argonne National Laboratory. Contact nera-software@anl.gov for distribution information. The distribution package includes source code, build instructions, benchmark examples and documentation.

2.2 External Library Dependencies

PROTEUS-MOC achieves distributed memory parallelization using the Message Passing Interface (MPI) protocol and uses the MPI2 standard typically through the MPICH2 [5] library implementation. PROTEUS-MOC also has essential dependencies on the mesh partitioning package METIS [6]. Additionally, PROTEUS-MOC optionally interfaces with data format library HDF5 [7]. For help installing any of the packages for use with PROTEUS-MOC, refer to the documentation included with the distribution, or contact nera-software@anl.gov.

PROTEUS-MOC uses the METIS package to determine the mesh partitioning scheme for spatial domain decomposition. To download METIS, visit the webpage http://glaros.dtc.umn.edu/gkhome/fsroot/sw/metis/OLD. PROTEUS-MOC is currently configured to use METIS 4.0 or METIS 5.0. To use METIS 4.0, no action is necessary. To use METIS 5.0, the pre-processing variable PROTEUS_Use_METIS_VERSION_5 must be defined in PROTEUS_Preprocess.h.

2.3 Compiling the Code

Detailed compilation instructions are provided with the distribution, including suggested configuration options for many of the external dependencies. Once the external library dependencies have been downloaded and built, PROTEUS-MOC is compiled using the provided makefile in the *source* directory of the distribution. The makefile includes a second file, *Makefile.arch*, also located in the same directory.

2.3.1 Customization of Makefile.arch

Makefile.arch is a text file in the /source directory which contains typical compiler options and flags. We recommend you modify *Makefile.arch* to add an architecture-specific compilation section for your machine.

It is recommended to add a section to *Makefile.arch* for your machine similar to above where the machine name should be recovered earlier in the script using the "uname" or "dnsdomainname" UNIX functions. In the file, the locations of MPICH compiler, METIS, and

HDF5 should be assigned. Note that the F90 compiler must be specified individually as it is used to compile one of the kinetics routines.

2.3.2 Customization of PROTEUS_Preprocess.h

The header file *PROTEUS_Preprocess.h* [2, 8] controls various data assignments via pre-processor directives. Most of the contents of this file should remain unchanged for routine use. However, the top few definitions can be commented or uncommented to enable debug printing, and to enable compilation with optional dependencies.

2.3.3 Building the Targets

Once you have customized *Makefile.arch* and *PROTEUS_Preprocess.h*, invoke the "make all" command from Linux command prompt inside the /source directory. This command creates all of the targets (executables) in Table 1. Alternatively, type "make <target>" at the command line to create only a specific target.

Table 1. List of PROTEUS-MOC Makefile Targets (Executables)

Target	Application
mocex.x	Steady state version of PROTEUS-MOC
mocex_kinetics.x	Kinetics version of PROTEUS-MOC
bench	Install Verification

2.3.4 Recommended Compilers and Architectures

PROTEUS-MOC has been regularly compiled with the Intel compiler version 13.1 or older on Intel hardware. Compilation with GNU compilers should be relatively straightforward using the flags "-ffree-line-length-0 -ffixed-line-length-0" on the FFLAGS line in Makefile.arch. However, the code is not routinely tested with the GNU compilers so minor changes may be required for successful compilation.

3. Methodologies

3.1 Method of Characteristics

The PROTEUS-MOC solver of PROTEUS is a 3D MOC formulation [9] that combines the 2D method of characteristics (MOC) with a discontinuous Galerkin finite element method for the axial variable. In this formulation, the axial variation of angular flux and neutron source in individual computing meshes is represented with the linear basis functions as:

$$\varphi_i(x,y,z,\Omega) \approx \sum_{j=1}^{2} \varphi_i^j(x,y,\Omega) u_i^j(z), \tag{1}$$

$$q_i(x,y,z,\Omega) \approx \sum_{j=1}^{2} q_i^j(x,y,\Omega) u_i^j(z), \tag{2}$$

where i is the index for 3D mesh and $u_i^j(z)$ is the j-th basis function. This introduces an approximation at the axial element interfaces. In the PROTEUS-MOC, the following linear basis functions are used:

$$u_i^1(z) = 1 \ , \tag{3}$$

$$u_i^2(z) = \frac{2}{\Delta_i}\left(z - \frac{(z_i^- + z_i^+)}{2}\right), \tag{4}$$

where Δ_i is the axial height of the i-th mesh. By inserting these angular flux and neutron source into the 3D Boltzmann transport equation and by applying the discontinuous weighted residual technique, the following equation, which is the analogy of the conventional 2-D MOC equation, can be yielded for a polar direction in the upper hemisphere of angular domain ($\Omega_z > 0$):

$$\left(\Omega_x \frac{\partial}{\partial x} + \Omega_y \frac{\partial}{\partial y}\right)\Psi_i(x,y,\Omega) + \boldsymbol{\Sigma}_i \Psi_i(\mathrm{x},\mathrm{y},\Omega) = S_i \ , \tag{5}$$

where Ψ_i is the angular flux vector, which consists of the φ_i^1 and φ_i^2. The cross section matrix, $\boldsymbol{\Sigma}_i$, and the source vector, S_i, can be obtained as:

$$\boldsymbol{\Sigma}_i = \begin{bmatrix} \Sigma_i + \Omega_z/\Delta_i & \Omega_z/\Delta_i \\ -3\Omega_z/\Delta_i & \Sigma_i + 3\Omega_z/\Delta_i \end{bmatrix}, \tag{6}$$

$$S_i = \begin{bmatrix} q_i^1(x,y,\Omega_z) + \Omega_z \varphi_i^-(x,y,\Omega_z)/\Delta_i \\ q_i^2(x,y,\Omega_z) - 3\Omega_z \varphi_i^-(x,y,\Omega_z)/\Delta_i \end{bmatrix}, \tag{7}$$

where φ_i^- is the incoming angular flux at the bottom of the mesh and q_i^j is the j-th basis function contribution to the fission and scattering source. The detailed derivation and the equation for opposite angular direction ($\Omega_z < 0$) can be found in Ref. 9.

3.2 Group-Sweeping Solution Scheme

The original PROTEUS-MOC code solved the discretized multi-group transport equation using the GMRES method, which is one of the Krylov subspace methods. To explain, we assemble the space-angle-energy terms into the flux vector Ψ and source Q (fission and/or fixed) such that we can write:

$$\begin{aligned}(\mathbf{L}-\mathbf{W})\Psi &= Q,\\ (\mathbf{I}-\mathbf{L}^{-1}\mathbf{W})\Psi &= \mathbf{L}^{-1}Q, \ .\\ \mathbf{A}\Psi &= b.\end{aligned} \tag{8}$$

In this system, the loss and collision operator $\mathbf{L}$ is inverted as part of the $\mathbf{A}$ matrix definition which is done because it is block diagonal with respect to energy and minimizes the vector size of Ψ. This leaves the remaining inversion of the scattering operator, $\mathbf{W}$, to be iteratively converged. The original intention of the PROTEUS-MOC implementation was to develop an appropriate preconditioner using conventional Gauss-Seidel iteration in energy where a simplified within group diffusion representation of the scattering operator would be constructed. The above approach is advantageous for parallel computation since energy group decomposition can be made scalable simultaneously with space and angle decomposition.

With an un-preconditioned GMRES method, the number of back vectors required to solve the system can become quite high. The set of back vectors are defined using a Gram-Schmidt orthogonalization procedure and require a certain span (number of them) in order to prevent oscillation about local minimum or divergence. In the implementation, the GMRES method is applied to a multi-group equation and the basis vectors span the entire space, angle and energy domains which makes each back vector considerably large. When solving neutronics problems with PROTEUS-MOC, we have observed the need to use between 30 and 100 back vectors where we had hoped the preconditioned GMRES approach would only require 5-10. Without preconditioning, excessive memory is required to accommodate the large basis vector sets, which strongly limits the applicability of the code to real world transport calculations.

In PROTEUS-MOC, energy decomposition is not currently scalable and requires additional research to understand the fundamental implementation problem. In a classical iterative approach in neutronics, the multigroup system is solved using a Gauss-Seidel scheme in energy. In that approach, the error norm is typically focused on the eigenvalue and fission source convergence because of using the power method. In that situation, the individual energy group equations contribute to the error norms in disproportional amounts. As an example, in fast spectrum systems, the lowest energy groups typically have no impact on the fission source or eigenvalue and can be loosely converged relative to the energy groups that make up the bulk of the neutrons in the system. Further, the condition number of each diagonal system, i.e. the within group transport equation, is dependent upon the scattering ratio and optical properties (total cross section) of the energy group. Additionally, because of parallel decomposition, the condition number of each within group equation is modified by the optical thickness of each processors

assigned subdomain and the total number of subdomains. In this situation, the amount of computational work required to solve each within group system can be considerably different from that observed in serial (i.e. because of the parallelism) and because of the physics properties.

In a conventional, non-parallel implementation, a relative error criteria applied to the within group equation in the Gauss-Seidel scheme that is weighted with respect to the fission source contribution can be shown to lead to the minimal computational effort in an un-accelerated power method. In the un-preconditioned GMRES solver of PROTEUS-MOC, there is no similar physical control that can be applied to the space of $\mathbf{A}\Psi$ that is consistent with the physically interpretable space of Ψ. Thus without a preconditioner that can account for the relative importance of the components of Ψ, an un-preconditioned GMRES approach is woefully inefficient compared with a Gauss-Seidel approach to solving the system of equations with or without parallelism.

Because of the time constraints on the development of PROTEUS-MOC, we have implemented the conventional Gauss-Seidel iterative approach in energy for the time being. This simultaneously reduces the memory requirements as it eliminates the GMRES solver applied to the entire space and improves the performance as Gauss-Seidel is almost always the optimal scheme for the multigroup system when no parallelism is being employed. PROTEUS-MOC was already equipped with a within-group solver as it is required for the fixed source problems in the sub-group method. In this case, the within group equation has a form identical to Eq. (8) except for the understanding that the $\mathbf{L}$ matrix is the group-wise loss and collision operator, $\mathbf{W}$ is the within-group scattering operator, Q is the group source, and Ψ is the mesh-averaged angular fluxes for a given energy group. In this implementation, the GMRES solver is applied to the group-wise form of Eq. (8).

In the parallel computation with domain decomposition, Eq. (8) is modified to account for the communication between the neighboring domains explicitly in the formulation as:

$$(\mathbf{I} - \mathbf{L}_i^{-1}\mathbf{W}_i\mathbf{K}_i)\Gamma_i = \mathbf{L}_i^{-1}Q_i \quad , \tag{9}$$

where i is the index for the processor, Γ_i is the new solution in the parallel computation that additionally includes the boundary angular flux information, and $\mathbf{K}_i$ is the communication operator.

In order to employ the Gauss-Seidel scheme using the existing within-group solver, the group source construction including fission and scattering reactions was added to the PROTEUS-MOC code. Note that the within-group scattering is excluded in the group source computation since it is internally updated in the within-group solver as shown in Eq. (9). In the conventional calculation procedure of PROTEUS-MOC, the Gauss-Seidel scheme was added yielding three layers of nested iterations: the fission source iteration (power method), Gauss-Seidel iteration over energy, and the within-group Krylov subspace iteration (each diagonal inversion of Gauss-Seidel).

The updated version of PROTEUS-MOC code is equipped with a multi-group Krylov (MGK) solver and a Gauss-Seidel (GS-WGK) in energy scheme with a within-group Krylov (WGK) solver. In the PROTEUS-MOC calculations, the default flux solver is the MGK solver. Alternatively, the GS-WGK solver can be invoked in the flux calculation when this solver is specified in an input file.

3.3 CMFD Formulation

The Coarse Mesh Finite Difference (CMFD) scheme satisfies the neutron balance equation in conjunction with the corrective current relation that preserves the leakage of the original higher order solution. Therefore, in order to use the CMFD acceleration scheme, the spatial discretization employed in the transport solver should strictly satisfy the global and local neutron balances. Because of the non-conservative discretization property of the Galerkin method applied to the axial variable in the PROTEUS-MOC method, a non-conservative discretization results and the CMFD prerequisite is not satisfied. A blind application of the CMFD scheme results in inconsistency of the CMFD solution with the PROTEUS-MOC solution. To enforce neutron balance in each coarse mesh and consistency between CMFD and higher order equations, a consistent CMFD formulation for PROTEUS-MOC calculation was devised by introducing a fictitious cross section, referred to as a pseudo absorption cross section, in each CMFD neutron balance equation. [10]

In order to formulate a CMFD problem for accelerating the 3D transport calculation, the following quantities are needed: the homogenized group constants for coarse meshes and the coupling coefficients that specify interface current relations between the two adjacent coarse meshes. From the transport solution, homogenized group constants can be generated by flux-volume weighting over the fine meshes belonging to the coarse mesh. The coupling coefficients can be determined using the following CMFD relation:

$$J_{i\to k} = -\tilde{D}_{i\to k}(\bar{\phi}_k - \bar{\phi}_i) - \hat{D}_{i\to k}(\bar{\phi}_k + \bar{\phi}_i) \ , \tag{10}$$

where i and j are indices for the coarse meshes that are connected through a common interface. $\bar{\phi}_i$ and $\bar{\phi}_j$ is the averaged fluxes of the coarse meshes i and j, respectively, and $J_{i\to k}$ is the surface current from the coarse mesh i to the coarse mesh j. The average flux and surface current can be readily obtained from the PROTEUS-MOC transport solution. $\tilde{D}$ is the coupling coefficient obtained from the conventional finite difference diffusion formulation, and $\hat{D}$ are the current correction factor to preserve the interface current obtained from the MOC solution. The CMFD relation in Eq. (10) can be solved for the current correction factor as:

$$\hat{D}_{i\to k} = -\frac{J_{i\to k} + \tilde{D}_{i\to k}(\bar{\phi}_k - \bar{\phi}_i)}{(\bar{\phi}_k + \bar{\phi}_i)} \ . \tag{11}$$

Using the surface currents obtained from the PROTEUS-MOC solution, the current correction factors can be determined. Under the current continuity condition, the current correction factors satisfy the following relation:

$$\hat{D}_{i\to k} = -\hat{D}_{k\to i} \ . \tag{12}$$

As a remedy to enforce the neutron balance and the consistency of the CMFD equation, a fictitious cross section, referred as a pseudo absorption cross section, is introduced in the CMFD neutron balance equation as:

$$\sum_k J_{i\to k} + (\Sigma_r^i + \Sigma_p^i)\bar{\phi}_i V_i = \bar{q}_i V_i \ , \tag{13}$$

where Σ_p is the pseudo absorption cross section. The pseudo absorption cross section of each coarse mesh can be determined from the PROTEUS-MOC solution as:

$$\Sigma_p^i = \frac{\bar{q}_i V_i - \sum_k J_{i\to k}}{\bar{\phi}_i V_i} - \Sigma_r^i \ . \tag{14}$$

The homogenized group constants, current correction factors and pseudo absorption cross sections are iteratively updated during the iterative solution process because those parameters are a function of the transport solution.

In the PROTEUS-MOC method, the angular flux and current can be discontinuous at the mesh interfaces. In order to secure the consistent CMFD current relation, the discontinuity of current is inevitably allowed and consequently, Eq. (12) does not hold for the current correction factors. Therefore, when the correction factor for the current from a coarse mesh *i* to a coarse mesh *k*, $\hat{D}_{i\to k}$, is determined, the current value obtained from the angular flux distribution in the coarse mesh *i* should be used. Alternatively, a single current at the coarse mesh interface can be defined by making use of the outgoing angular fluxes of two neighboring coarse meshes at the common interface and this current is used in the determination of the current correction factor in Eq. (11) and the PAXS in Eq. (14). The Overall calculation flow is illustrated in Figure 5.

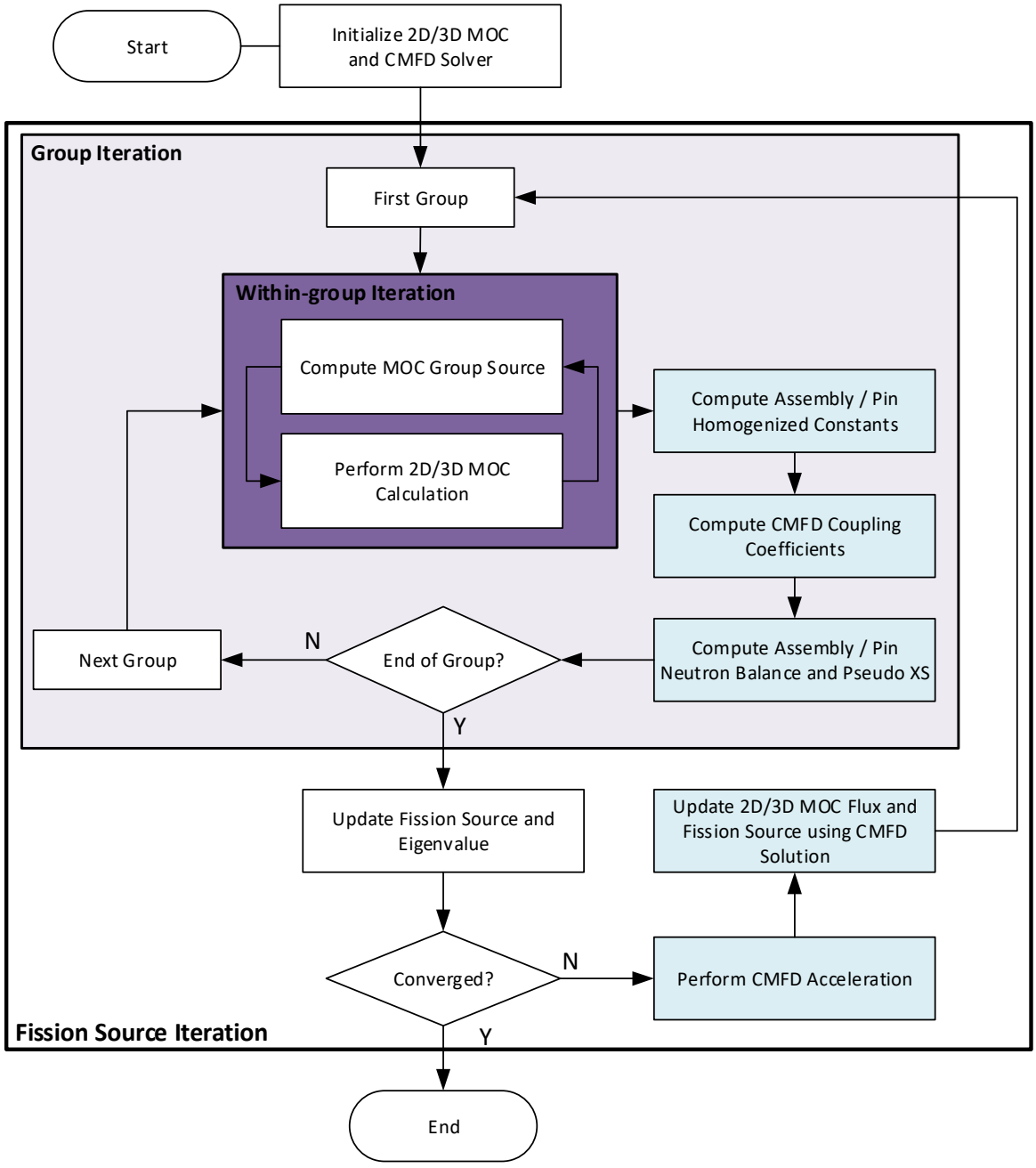

Figure 5. Calculation Flow of PROTEUS-MOC with CMFD Acceleration

.

4. Code Execution Syntax

4.1 Serial Jobs

PROTEUS-MOC is executed via command line on Linux platforms. To perform a serial job, type the following at the command prompt (assuming the executable is in the current directory or in the user's path):

```
$ mocex.x –input mydriver.inp
```

```
$ mocex.x –input mydriver.inp  -output  mydriver.out
```

The "-input" flag argument specifies that PROTEUS-MOC should look for a specific driver input file in the location specified by the following argument. If the "-input" flag and argument is omitted, PROTEUS-MOC looks for the driver input file in the current working directory with the default name "mocex.inp". The output can be redirected from standard output to a file using the -output flag.

4.2 Parallel Jobs

To execute the steady state solver in parallel mode, type the following:

```
$ mpiexec –n 8 mocex.x –input mydriver.inp
```

```
$ mpiexec –n 8 mocex.x –input mydriver.inp  -output mydriver.out
```

The *mpiexec* command is standard with MPICH and specifies that 8 processors should be used in this example. The parallel decomposition of the problem is fully specified by the number of processors and the values of SEGMENT_ANGLE and SEGMENT_PLANE in the driver input. The problem is first partitioned by angle and plane into SEGMENT_ANGLE and SEGMENT_PLANE processors, respectively. The remaining factor of processors is automatically dedicated to spatial decomposition. The spatial mesh is decomposed at runtime based on these numbers. Parallelism is discussed further in the next chapter.

4.3 Kinetics Jobs

The kinetics solver is run identically to the above except using a different executable:

```
$ mpiexec –n 4 mocex_kinetics.x –input mydriver.inp
```

```
$ mpiexec –n 4 mocex_kinetics.x –input mydriver.inp  -output mydriver.out
```

As in the steady state solver case, only the driver input file name is specified. The kinetics input file is also required and assumed to reside in the working directory and have the default name "kinetics.inp". To specify an alternative kinetics input file name, use the following syntax:

```
$ mpiexec –n 4 mocex_kinetics.x –input mydriver.inp –kinetics mykinetics.inp
```

```
$ mpiexec –n 4 mocex_kinetics.x –input mydriver.inp –kinetics mykinetics.inp  -output mydriver.out
```

5. Input File Descriptions

This section describes the required input files for a PROTEUS-MOC calculation:

- Driver input file
- Cross section file
- Mesh file
- Material assignment file
- T/H assignment file

5.1 Driver Input File (.inp)*

Upon execution, PROTEUS-MOC searches for the driver input file, "mocex.inp" by default, in the current working directory. This input file is a plain text (ASCII) file that drives the PROTEUS-MOC calculation by specifying solver tolerances, the angular discretization, parallelization options, and other input options. Additionally, the UNIX file paths to the other input files (cross sections, mesh, and material assignment file) are specified. Input options are specified in the file by special keywords which can appear in any order.

To use a different file name for the driver input such as "mocex_7g_L5T5.inp", the command line option "–input mocex_7g_L5T5.inp" should be added at the command line as below. The output file name can be added with "-output" as well.

```
$ mocex.x  -input  mocex_7g_L5T5.inp  -output mocex_7g_L5T5.out
```

5.1.1 Required Input

The following table shows the essential driver level input required in every PROTEUS-MOC calculation. The keywords and values are not case sensitive. Default values are listed when applicable.

Table 2. Required Keywords for Driver Input File

Keyword	Input Data	Default Value	Description
SN_TYPE	*See SN manual*	-	Specifies the type of SN cubature to use.
THETA_RESOLUTION	[Integer > 0[]	-	Specifies the polar angle resolution of the SN cubature.
PHI_RESOLUTION	[Integer > 0]	-	Only applicable for product cubatures such as Leg-Tcheby. Specifies the azimuthal angle (x,y) resolution of the SN cubature.
SOURCEFILE_MESH	[Max 128 Characters]	-	Specifies the UNIX file path to a spatial

			geometry mesh file.
SOURCEFILE_XS	[Max 128 Characters]	-	Specifies the Unix file path to a cross section data file.
SOURCEFILE_MATERIAL	[Max 128 Characters]	-	Specifies the Unix file path to a material mapping file.

5.1.2 Optional Input

Other optional driver-level input options are listed in Table 3. They can be classified into the following groups: parallelization options, 2D ray-tracing options, iterative solver options, cross section options, boundary condition options, and CMFD options. Brief explanations of each option as well as the default values are provided.

Table 3. Auxiliary Keywords for Driver Input File

Keyword	Input Data	Default Value	Description
Parallelization Options			
SEGMENT_ANGLE	[Integer ≥ 0]	0	The number of segments to attempt in the angular directions. 0 = Code decides max parallelization 1=Serial 2 and up = # Processors (segments) in angle
SEGMENT_PLANE	[Integer ≥ 0]	0	The number of segments to attempt in the axial planes. 0 = Code decides max parallelization 1=Serial 2 and up = # Processors (segments) in axial planes
2D Ray-Tracing Options			
TRAJECTORY_AREA	[Real Value]	1.0D0	The maximum trajectory area to use.
BACK_PROJECTION	DOMAIN LOCAL VOLUME FULL	DOMAIN	Back-projection technique to employ.

Keyword	Input Data	Default Value	Description
EQUAL_AREA_SPACING	YES NO	NO	Use equal area cubature on the domain boundary is used for defining trajectories.
Iterative Solver Options			
EIGENVALUE_GUESS	[Real Value]	1.0	Guess for the initial eigenvalue.
ITERATIONS_FISSION	[Integer > 0]	100	Maximum number of outer (fission source) iterations.
TOLERANCE_EIGENVALUE	[Real Value]	1.0E-6	Targeted relative error on the eigenvalue.
TOLERANCE_FISSION	[Real Value]	5.0E-6	Targeted relative error on the fission source.
TOLERANCE_FLUX	[Real Value]	1.0E-7	Targeted relative error on the flux solution.
BASIC_BWO	YES NO	NO	Applies a basic bandwidth optimization to mesh (reorder mesh for minimum bandwidth with ICC preconditioner).
TRANSPORT_ ITERATION_TYPE	WGS_KRYLOV MGS_KRYLOV WGS_LEGACY	WGS_KRYLOV	Specify transport iteration type used in the eigenvalue/transient calculation
USE_WGS_KRYLOV	YES NO	NO	Specify whether WGS Krylov scheme is used in the eigenvalue calculation (recommended). (NO) MGS Krylov scheme is used
(For MGS Krylov) USE_MGS_FGMRES *(For WGS Krylov)* USE_WGS_FGMRES	YES NO	YES	USE the FGMRES algorithm in MGS/WGS Krylov Solver.
(For MGS Krylov) ITERATIONS_MGS_KRYLOV *(For WGS Krylov)* ITERATIONS_WGS_KRYLOV	[Integer > 0]	1000	The maximum number of iterations in MGS/WGS Krylov solver.

Keyword	Input Data	Default Value	Description
(For MGS Krylov) BACKVECTORS_MGS_KRYLOV *(For WGS Krylov)* BACKVECTORS_WGS_KRYLOV	[Integer > 0]	30	The number of back vectors to employ in MGS/WGS Krylov solver.
(For MGS Krylov) TOLERANCE_MGS *(For WGS Krylov)* TOLERANCE_WGS	[Real Value]	0.1	Relative error target in MGS/WGS Krylov solver.
ITERATIONS_MGS_GS	[Integer > 0>]	1	The number of upscattering iterations in WGS Krylov solver (WGS Krylov solver only). 1= No upscattering iteration
Cross Section Options			
USE_CSAPI, USE_XSAPI	NO SUBGROUP (or SG) RESONANCETABLE (or RT)	NO	Indicates whether the subgroup or resonance table cross section library is to be used to generate cross sections. The cross section libraries are expected for heterogeneous geometry problems.
XSAPI_TOLERANCE_FSS	[Real Value]	1.0E-4	Targeted relative error on the fixed source problems for cross section generations.
USE_TRANSPORT_XS	YES NO	NO	Indicates that the transport corrected cross section is to be used as the “total” cross section. All anisotropic scattering data is ignored.
SCATTERING_ORDER	[Integer ≥ 0]	0	Legendre expansion order of the scattering kernel.

Keyword	Input Data	Default Value	Description
CHECK_XS_BALANCE	YES NO	NO	Check the cross section balance and correct total cross section if imbalance of cross section is detected.
Boundary Condition Options			
BC_ALIAS	*(For radial surface)* [Side set ID] [BCTYPE] *(For axial surface)* TOP <BCTYPE> Bottom <BCTYPE>	-	Assigns boundary condition at runtime to side set ID, top and bottom. Valid BCTYPE names are VOID or REFLECTIVE.

5.1.3 CMFD Input Keywords

PROTEUS-MOC includes the coarse mesh finite difference (CMFD) acceleration capability which can effectively accelerate an eigenvalue calculation. Table 4 presents the input keywords associated with the use of the CMFD acceleration. Note that the prerequisite for facilitating the CMFD acceleration is the coarse mesh structure. Thus, the PROTEUS-MOC run with the CMFD acceleration requires the user-generated coarse mesh file that should be consistent with the fine meshes. It can be generated using the mesh tools such as CUBIT and the PROTEUS mesh toolkit. PROTEUS-MOC is able to import the user-generated coarse mesh structure using the same format of the fine mesh file.

Table 4. CMFD Keywords for Driver Input File

Keyword	Input Data	Default Value	Description
Required Input Options			
USE_CMFD	*YES* *NO*	NO	Specify whether the CMFD acceleration is to be used in the eigenvalue calculations.
SOURCEFILE_COARSEMESH	*[Max 128 Characters]*	-	Specifies the UNIX file path to a CMFD mesh file.
CMFD_FORM			CMFD PCMFD
CMFD_SOLVER_TYPE			WGS_CMFD MGS_CMFD
CMFD Outer Iteration Options			

Keyword	Input Data	Default Value	Description
CMFD_ITERATION_FISSION	[Integer > 0]	100	The maximum number of outer iterations in each CMFD eigenvalue calculation.
CMFD_REDUCTION_EIGENVALUE	[Real Value]	0.1	Target error reduction of eigenvalue in CMFD calculation (partial convergence criterion).
CMFD_REDUCTION_FISSION	[Real Value]	0.1	Target error reduction of fission in CMFD calculation (partial convergence criterion).
CMFD_REDUCTION_FLUX	[Real Value]	0.1	Target error reduction of flux in CMFD calculation (partial convergence).
CMFD_TOLERANCE_EIGENVALUE	[Real Value]	1.0E-10	Relative target error of eigenvalue in CMFD calculation (Absolute convergence criterion).
CMFD_TOLERANCE_FISSION	[Real Value]	1.0E-10	Relative target error of fission in CMFD calculation (Absolute convergence criterion).
CMFD_TOLERANCE_FLUX	[Real Value]	1.0E-10	Relative target error of flux in CMFD calculation (Absolute convergence criterion).
CMFD_UNDERRELAXATION	[Real Value]	0.5	The under-relaxation parameter in CMFD prolongation process. 1.0 = No under-relaxation.
CMFD Krylov Solver (Inner Iteration) Options			

Keyword	Input Data	Default Value	Description
CMFD_BACKVECTORS_WGS_K	[Integer > 0]	30	The number of back vectors to employ in Krylov solver for WGS CMFD calculation.
CMFD_WGS_K_ITER	[Integer > 0]	100	The maximum number of inner Krylov iterations in each WGS CMFD calculation.
CMFD_TOLERANCE_WGS	[Real Value]	0.1	Relative error target in Krylov solver for WGS CMFD calculation.
Void (Low Density) Region Treatment Options			
CMFD_VOID_EXIST	YES NO	NO	Specify whether voided regions exist in CMFD meshes
CMFD_VOID_CRITERION	[Real Value]	1.0E-3	Tolerance of total macroscopic cross section for identifying voided regions.

5.1.4 Feedback Input Keywords

For typical PWR applications, PROTEUS-MOC includes a simple T/H model based upon the radial and axial heat convection in closed flow channels [11]. A configuration of flow channel, in which the flow mixing between adjacent channels is not considered, is specified through a coarse mesh input file. Thus, the coarse mesh input file should be provided using SOURCEFILE_COARSEMESH keyword when the feedback option is turned on. Along with the closed channel input, a mapping between 3D blocks and T/H region types should be provided using a T/H assignment file, which facilitates the update of T/H properties to each cross section region. The input keywords for T/H feedback capability is explained in Table 5.

In the current release of PROTEUS-MOC, note that the feedback is not fully functioned. It only computes the temperature distribution for given cross sections without the T/H feedback. The cross section update accounting for T/H properties is being implemented.

Table 5. Feedback Keywords for Driver Input File

Keyword	Input Data	Default Value	Description
SOURCEFILE_ISOPAR	*[Max 128 Characters]*	-	Specifies the Unix file path to a ISOPAR file which is a set of ISOTXS files tabulated with state parameters (Note that this function is not complete yet at the moment)
SOURCEFILE_THREGION	*[Max 128 Characters]*	-	Specifies the Unix file path to a T/H region mapping file (named T/H assignment)
SOURCEFILE_SIMPLETH	*[Max 128 Characters]*	-	Specifies the Unix file path to a simple T/H input file name
USE_TH_FEEDBACK	YES NO	NO	Turn on the simple T/H calculation
USE_TH_FEEDBACK_DENSITY	YES NO	NO	Turn on the simple T/H calculation for density change (Note that this function is not completed yet at the moment)

5.2 Cross Section File (.ISOTXS, *.anlxs)*

The cross section file consists of the multigroup cross sections for all isotopes and/or compositions in the problem. Multigroup cross sections must be provided in one of the following formats:

a. *.ISOTXS (binary file)
b. *.anlxs (ASCII file)

The ISOTXS format is the preferred file format for cross section data used by PROTEUS-MOC. The MC^2-3 code [14] can be used to process multigroup cross sections in this format. Additionally, the DRAGON code (Ecole Polytechnique de Montreal) [15] has a capability to generate ISOTXS file. The Serpent [16] and OpenMC [17] Monte Carlo codes can be used to generate cross sections, for which their cross section outputs should be converted to an ISOTXS format file using the GenISOTXS code [0]. Note that Serpent generates macroscopic cross sections only.

The anlxs file format is a simple ASCII interpretation of the data provided in ISOTXS. Any anlxs file format that is provided is converted into the appropriate ISOTXS file at runtime.

The cross section file does not need to be located in the working directory. The UNIX file path to the cross section file must always be specified in the driver input file using the SOURCEFILE_XS keyword.

5.3 Material Assignment Files (.assignment)*

The main role of material assignment file is to provide a mapping of material to 3D blocks defined in the mesh file. In PROTEUS-MOC, the 3D heterogeneous domain is *internally* constructed by combining the 2D unstructured mesh file and the geometry extrusion information. Thus, the material assignment file contains the additional geometry extrusion information. Upon execution of PROTEUS-MOC, the material assignment file (*.assignment) file performs four functions:

- *Define materials or mixtures based on the isotopes in the cross section files*
- *Create 3D blocks by axially extruding 2D blocks in the mesh file*
- *Assign the materials to 3D blocks in the mesh*
- *Assign properties (e.g. density) to 3D blocks in the mesh*

The material assignment file uses simple keyword-based inputs in free format. It must be created by hand by the user, although scripting procedures can often be developed to speed the process. A python script is available to help a user create an assignment file in a more user-friendly manner. A comment starts with either "!" or "#".

5.3.1 Defining Materials

The user-defined isotope or composition names in the *anlxs* or *ISOTXS* files are the base materials we can work with. These base materials can be directly assigned to blocks. Additionally, new materials can be defined as mixtures of the base materials.

Figure 6 demonstrates the definition of a material called FUEL from 8 compositions appearing in the cross section file, "U234A", "U235A", etc. The keyword MATERIAL_DEF is used recursively to add compositions to the material FUEL with the given atom fractions. The sum of all atom fractions is renormalized to 1 inside PROTEUS-MOC. In this way, the true atom densities can be given (in #/barn-cm) for easier recordkeeping.

```
MATERIAL_DEF FUEL U234A    2.7572E-5
MATERIAL_DEF FUEL U235A    2.5533E-3
MATERIAL_DEF FUEL U236A    9.5684E-6
MATERIAL_DEF FUEL U238A    1.5316E-4
MATERIAL_DEF FUEL B10A     1.5539E-4
MATERIAL_DEF FUEL B11A     6.2547E-4
MATERIAL_DEF FUEL C12A     1.9521E-4
MATERIAL_DEF FUEL ALA      5.2751E-2
```

Figure 6. Material Definition in the Assignment File (Atom Fractions).

Figure 7 demonstrates the definition of a material called Water from 2 compositions appearing in the cross section file, "H___1" and "O__16". Again, the MATERIAL_DEF keyword is used. The negative fraction indicates that the value is given in terms of weight fraction rather than atom fraction.

```
MATERIAL_DEF Water H___1    -0.1121
MATERIAL_DEF Water O__16    -0.8879
```

Figure 7. Material Definition in the Assignment File (Weight Fractions).

Figure 8 demonstrates the definition of a material called Water from 2 compositions appearing in the cross section file, "H1" and "O16", as well as the definition of a material called Salt from compositions "NA23" and "Chlor". The fractions are assumed to be atom fractions since they are positive. The final line then defines a mixture called Saline which is 0.9% Salt and 99.1% Water by atom fraction. This example demonstrates the recursivity and flexibility of the MATERIAL_DEF keyword. It is not permitted to mix weight fractions and atom fractions in the definition of a material.

```
MATERIAL_DEF Water H1       2.0
MATERIAL_DEF Water O16      1.0

MATERIAL_DEF Salt  NA23     1.0
MATERIAL_DEF Salt  Chlor    1.0

MATERIAL_DEF Saline 0.009 Salt 0.991 Water
```

Figure 8. Recursive Material Definition in the Assignment File.

5.3.2 Creation of 3D Blocks via Axial Extrusion

The specification of 3D block structures via the axial extrusion is a unique feature of material assignment file for PROTEUS-MOC. The assignment file creates the 3D blocks with the following three steps:

- Step 1: Define a group of 2D blocks (referring to assembly) that will be collectively extruded in *Step 2* (*Input keyword: EXTRUDE*).
- Step 2: Extrude each assembly defined in *Step 1* with axial segmentations, which results in the 3D blocks along the axial direction (*Input keyword: ASSEMBLY*).
- Step 3: Define axial mesh structures that are consistent with the axial segmentations specified in *Step 2* (*Input keyword: ZGRID*).

The resulting 3D blocks become basic entities for material and property assignments in the subsequent processing steps. Figure 9 demonstrates the construction of 3D blocks from the 2D blocks with axial segmentations. Note that the driver input file should be updated with the axial boundary conditions as below.

```
BC_ALIAS    top             reflective  ! or void
BC_ALIAS    bottom          reflective  ! or void
```

Step 1

Keyword	Name of 2D Block	Name of Assembly	-	-
EXTRUDE	MESHREGION_0011	ASSEMBLY_001		
EXTRUDE	MESHREGION_0012	ASSEMBLY_001		
EXTRUDE	MESHREGION_0013	ASSEMBLY_001		
EXTRUDE	MESHREGION_0020	ASSEMLBY_002		

Step 2

Keyword	Name of Assembly	Name of 3D Block	Lower Axial Bound, cm	Upper Axial Bound, cm
ASSEMBLY	ASSEMBLY_001	BLOCK_001_01	0.00	20.00
ASSEMBLY	ASSEMBLY_001	BLOCK_001_02	20.00	40.00
ASSEMBLY	ASSEMBLY_002	BLOCK_002_01	0.00	20.00
ASSEMBLY	ASSEMBLY_002	BlOCK_002_02	20.00	40.00

Step 3

Keyword	Lower Position, cm	Upper Position, cm	Number of Sub-divisions	-
ZGRID	0.00	20.00	4	
ZGRID	20.00	40.00	4	

Figure 9. Creation of 3D Blocks in Assignment File

5.3.3 Assigning Materials to 3D Blocks

For each 3D block, the material can be assigned using the REGION_ALIAS keyword shown in Figure 10.

Keyword	Name of 3D Block	Name of Material	-
REGION_ALIAS	BLOCK_001_01	FUEL	
REGION_ALIAS	BLOCK_001_02	FUEL	
REGION_ALIAS	BLOCK_002_01	COOLANT	
REGION_ALIAS	BLOCK_002_02	COOLANT	

Figure 10. Assigning Materials to 3D Blocks in Assignment File

5.3.4 Assigning Properties to 3D Blocks

The keyword REGION_PROPERTY is used to assign various properties to the mesh such as ATOM_DENSITY, DENSITY(G/CC) and TEMPERATURE(K) to each mesh region. This approach is similar to MCNP where the same material can be used for multiple regions but at different concentrations. For the standard PROTEUS-SN user, the only properties of interest are Density(g/cc) and ATOM_DENSITY. Figure 11 shows the example of keyword REGION_ALIAS. Note that the actual material density of 3D block should be specified through this keyword because the keyword REGION_ALIAS simply links cross section data to a specific region.

Keyword	Name of 3D Block	Name of Property	Property Value
REGION_ALIAS	BLOCK_001_01	DENSITY(G/CC)	10.3
REGION_ALIAS	BLOCK_001_01	TEMPERATURE(K)	600.0
REGION_ALIAS	BLOCK_001_02	DENSITY(G/CC)	10.3
REGION_ALIAS	BLOCK_001_02	TEMPERATURE	600.0
REGION_ALIAS	BLOCK_001_01	DENSITY(G/CC)	0.7
REGION_ALIAS	BLOCK_001_01	TEMPERATURE(K)	300.0
REGION_ALIAS	BLOCK_001_02	DENSITY(G/CC)	0.7
REGION_ALIAS	BLOCK_001_02	TEMPERATURE	300.0

Figure 11. Assigning Region Properties to 3D Blocks in Assignment File

5.4 T/H Assignment File (.assignment)*

When temperatures are determined for a region which includes many elements with different materials (fuel, gap, cladding, and coolant), these inputs allow a user to provide a way to assign the temperatures to the elements on the material basis.

Table 6. T/H Keywords for Assignment File

Keyword	Input Data	Default Value	Description
FDK_REGION_ALIAS	*[3D block name]* *[T/H region type]*	-	Mapping between 3D blocks (defined in material assignment file) and T/H regions. The followings are valid T/H region types. FUEL : fuel region GAP : gap region CLAD : cladding region COOL : coolant region AVERAGE : average temperatures in the neighboring regions NONE : no temperature

5.5 T/H Input File (.th)*

The purpose of T/H input file is to provide essential quantities and properties for the simple T/H model, which are listed in Table 7.

Table 7. T/H Keywords for T/H Input File

Keyword	Input Data	Default Value	Description
ASSEMBLY_PITCH	[Real Value]	1.0E-10	Assembly pitch [cm]
FUEL_PIN_GEOMETRY	[Integer > 0] [Real Value]	0 1.0E-10	Fuel pin geometry (number of data(=3), pellet radius, inner radius of cladding, outer radius of cladding) [cm]
FUEL_PINS_IN_ASSEMBLY	[Integer > 0]	0	Number of fuel pins in an assembly
GT_PIN_GEOMETRY	[Integer > 0] [Real Value]	0 1.0E-10	GT pin geometry (number of data(=2), inner radius of cladding, outer radius of cladding) [cm]
GT_PINS_IN_ASSEMBLY	[Integer > 0]	0	Number of GT pins in an assembly
INLET_TEMPERATURE	[Real Value]	1.0E-10	Inlet temperature [^{o}C]

Keyword	Input Data	Default Value	Description
IT_PIN_GEOMETRY	[Integer > 0] [Real Value]	0 1.0E-10	IT pin geometry (number of data(=2), inner radius of cladding, outer radius of cladding) [cm]
IT_PINS_IN_ASSEMBLY	[Integer > 0]	0	Number of IT pins in an assembly
MASS_FLOWRATE	[Real Value]	1.0E-10	Mass flow rate [g/sec]
NUMBER_OF_AXIALNODES	[Integer > 0]	0	Number of axial thermal nodes which are normally the same as that of neutronics nodes
NUMBER_OF_CHANNELS	[Integer > 0]	0	Number of channels in the core
PINS_IN_ASSEMBLY	[Integer > 0]	0	Number of pins in an assembly
PIN_PITCH	[Real Value]	1.0E-10	Pin pitch [cm]
PRESSURE	[Real Value]	1.0E-10	Core pressure [MPa]
RELATIVE_POWER	[Real Value]	1.0E-10	Relative power [0 – 1]
TOTAL_POWER	[Real Value]	1.0E-10	Total power [MW]

5.6 Kinetics Input (.inp)*

Upon execution, PROTEUS-MOC searches for the kinetics driver input file, "mocex.inp" by default, in the current working directory. To use a different file name for the kinetics input such as "kinetics_case1.inp", the command line option "-kinetics_input kinetics.inp" should be added at the command line as below. The output file name can be added with "-output" as well.

```
$ mocex_kinetics.x -input mocex.inp -kinetics_input kinetics_case1.inp
$ mocex_kinetics.x -input mocex.inp -kinetics_input kinetics_case1.inp  -output kinetics_case1.out
```

The input keywords for T/H feedback capability is explained in Table 8. One notable thing is that material perturbations and time steps are specified through keyword TIME_STEP. This keyword is repeatedly used for defining individual time step including the material perturbation such as control rod movement.

Table 8. Keywords for kinetics input file

Keyword	Input Data	Default Value	Description
INITIAL_MATERIAL_FILE	*[Max 128 Characters]*	-	Specifies the Unix file path to material assignment file for initial state of kinetics calculation
TIME_STEP	[Real Value]	-	End time of kinetics time interval
	[Integer Value>0]	-	Define number of time step in this time inveral
	[Max 128 Characters]	-	Specifies the Unix file path to material assignment file for this time interval.
TFSP_TOLERANCE_FLUX	[Real Value]	1.0E-10	Define convergence criterion for kinetics solver
TFSP_ITERATIONS	[Integer > 0]	0	Define maximum number of outer iterations in the transient fixed source problem.
DELAY_XS_FILE	[Max 128 Characters]	-	Specifies the Unix file path to a delayed neutron parameter file in DLAYXS file format
INITIAL_MATERIAL_FILE	[Integer > 0] [Real Value]	0 1.0E-10	GT pin geometry (number of data(=2), inner radius of cladding, outer radius of cladding) [cm]

6. Sample Inputs

6.1 Driver Input File

```
! Parallel processing
SEGMENT_ANGLE            1
SEGMENT_PLANE            1

! Angles
THETA_RESOLUTION         3
PHI_RESOLUTION           3
SN_TYPE                  LEG-TCHEBY

! Convergence criteria
ITERATIONS_FISSION       150
EIGENVALUE_GUESS         1.0
TOLERANCE_EIGENVALUE     1.0e-5
TOLERANCE_FISSION        1.0e-5
TOLERANCE_FLUX           1.0e-5

! Iterative solver option
USE_WGS_KRYLOV           YES
TOLERANCE_WGS            0.05d
BACKVECTORS_WGS_KRYLOV   10

! Mesh and material assignment input files
SOURCEFILE_MESH          ../00_Mesh/pin_1x1.ascii
SOURCEFILE_MATERIAL      ../00_Mesh/pin_1x1_3d.assignment

! Cross section input file
SOURCEFILE_XS            ../../../00_XSLIB/THERMAL_11G.ISOTXS
USE_TRANSPORT_XS         NO
SCATTERING_ORDER         1

! T/H inputs
USE_TH_FEEDBACK          YES
USE_TH_FEEDBACK_DENSITY  NO  ! YES
SOURCEFILE_SIMPLETH      Thermalcore.th
SOURCEFILE_THREGION      Thermalcore.assignment

! Boundary conditions
BC_ALIAS    top              reflective
BC_ALIAS    bottom           reflective
BC_ALIAS    SIDESET_00000001 reflective
BC_ALIAS    SIDESET_00000002 reflective
BC_ALIAS    SIDESET_00000003 reflective
BC_ALIAS    SIDESET_00000004 reflective
BC_ALIAS    SIDESET_00000005 reflective
BC_ALIAS    SIDESET_00000006 reflective
BC_ALIAS    SIDESET_00000007 reflective
BC_ALIAS    SIDESET_00000008 reflective
```

Figure 12. Sample Driver Input File

6.2 Assignment Input File

```
! ZGRID <Lower Position in cm> <Upper Position in cm> <subintervals to apply>

ZGRID         0.0        20.0    5
ZGRID        20.0        40.0    5

! EXTRUDE <Name of 2D mesh region> <Name of assembly>

EXTRUDE              FUEL  ASM2D001
EXTRUDE              CLAD  ASM2D002
EXTRUDE               MOD  ASM2D003

! ASSEMBLY <Assembly Name> <Region Name> <Lower Axial Bound> <Upper Axial Bound>

ASSEMBLY  ASM2D001 REG2D001_3D001        0.0       20.0
ASSEMBLY  ASM2D001 REG2D001_3D002       20.0       40.0

ASSEMBLY  ASM2D002 REG2D002_3D001        0.0       20.0
ASSEMBLY  ASM2D002 REG2D002_3D002       20.0       40.0

ASSEMBLY  ASM2D003 REG2D003_3D001        0.0       20.0
ASSEMBLY  ASM2D003 REG2D003_3D002       20.0       40.0

! REGION_PROPERTY <Name of mesh region> {<property> <initial setting>}

REGION_PROPERTY  REG2D001_3D001 ATOM_DENSITY  1.00000E+00
REGION_PROPERTY  REG2D001_3D002 ATOM_DENSITY  1.00000E+00

REGION_PROPERTY  REG2D002_3D001 ATOM_DENSITY  1.00000E+00
REGION_PROPERTY  REG2D002_3D002 ATOM_DENSITY  1.00000E+00

REGION_PROPERTY  REG2D003_3D001 ATOM_DENSITY  1.00000E+00
REGION_PROPERTY  REG2D003_3D002 ATOM_DENSITY  1.00000E+00

! REGION_ALIAS <Name of assembly region> <Name of Material>

REGION_ALIAS     REG3D001_3D001     MAT_FUEL
REGION_ALIAS     REG3D001_3D002     MAT_FUEL

REGION_ALIAS     REG3D002_3D001     MAT_CLAD
REGION_ALIAS     REG3D002_3D002     MAT_CLAD

REGION_ALIAS     REG3D003_3D001     MAT__MOD
REGION_ALIAS     REG3D003_3D002     MAT__MOD

! MATERIAL_DEF <Material name> {<isotope name> <concentration>}

MATERIAL_DEF          MAT_FUEL      XS_FUEL  1.00000E+00
MATERIAL_DEF          MAT_CLAD      XS_CLAD  1.00000E+00
MATERIAL_DEF          MAT__MOD      XS__MOD  1.00000E+00
```

Figure 13. Sample Assignment Input File for a 3D Problem with Macroscopic Cross Sections

```
! ZGRID <Lower Position in cm> <Upper Position in cm> <subintervals to apply>

ZGRID         0.0        20.0    5
ZGRID        20.0        40.0    5

! EXTRUDE <Name of 2D mesh region> <Name of assembly>

EXTRUDE              FUEL  ASM2D001
EXTRUDE              CLAD  ASM2D002
EXTRUDE               MOD  ASM2D003

! ASSEMBLY <Assembly Name> <Region Name> <Lower Axial Bound> <Upper Axial Bound>

ASSEMBLY  ASM2D001 REG2D001_3D001        0.0       20.0
ASSEMBLY  ASM2D001 REG2D001_3D002       20.0       40.0

ASSEMBLY  ASM2D002 REG2D002_3D001        0.0       20.0
ASSEMBLY  ASM2D002 REG2D002_3D002       20.0       40.0

ASSEMBLY  ASM2D003 REG2D003_3D001        0.0       20.0
ASSEMBLY  ASM2D003 REG2D003_3D002       20.0       40.0

! REGION_PROPERTY <Name of mesh region> {<property> <initial setting>}

REGION_PROPERTY  REG2D001_3D001 ATOM_DENSITY  3.74648E-02
REGION_PROPERTY  REG2D001_3D002 ATOM_DENSITY  3.74648E-02

REGION_PROPERTY  REG2D002_3D001 ATOM_DENSITY  4.27669E-03
REGION_PROPERTY  REG2D002_3D002 ATOM_DENSITY  4.27669E-03

REGION_PROPERTY  REG2D003_3D001 ATOM_DENSITY  7.44336E-02
REGION_PROPERTY  REG2D003_3D002 ATOM_DENSITY  7.44336E-02

! REGION_ALIAS <Name of assembly region> <Name of Material

REGION_ALIAS     REG3D001_3D001     MAT_FUEL
REGION_ALIAS     REG3D001_3D002     MAT_FUEL

REGION_ALIAS     REG3D002_3D001     MAT_CLAD
REGION_ALIAS     REG3D002_3D002     MAT_CLAD

REGION_ALIAS     REG3D003_3D001     MAT__MOD
REGION_ALIAS     REG3D003_3D002     MAT__MOD

! MATERIAL_DEF <Material name> {<isotope name> <concentration>}

MATERIAL_DEF          MAT_FUEL         U235  2.48293E-04
MATERIAL_DEF          MAT_FUEL         U238  7.65991E-03
MATERIAL_DEF          MAT_FUEL          O16  2.95566E-02

MATERIAL_DEF          MAT_CLAD         ZR90  4.26303E-03
MATERIAL_DEF          MAT_CLAD         FE56  1.36597E-05

MATERIAL_DEF          MAT__MOD           H1  4.96224E-02
MATERIAL_DEF          MAT__MOD          O16  2.48112E-02
```

Figure 14. Sample Assignment Input File for a 3D Problem with Microscopic Cross Sections

For comparison between the assignment inputs for PROTEUS-MOC and PROTEUS-SN, the sample assignment input for PROTEUS-SN is shown in Figure 15. Note that this is for a 2D problem, while the assignment input for PROTEUS-MOC is for a 3D problem. As discussed in the previous section, the input keywords ZGRID, EXTRUDE, and ASSEMBLY are for use in PROTEUS-MOC only. For a 3D problem with PROTEUS-SN, a 3D mesh file for it should be provided.

```
! REGION_PROPERTY <Name of mesh region> {<property> <initial setting>}

REGION_PROPERTY             FUEL ATOM_DENSITY  3.74648E-02
REGION_PROPERTY             CLAD ATOM_DENSITY  4.27669E-03
REGION_PROPERTY              MOD ATOM_DENSITY  7.44336E-02

! REGION_ALIAS <Name of assembly region> <Name of Material>

REGION_ALIAS                FUEL     MAT_FUEL
REGION_ALIAS                CLAD     MAT_CLAD
REGION_ALIAS                 MOD     MAT__MOD

! MATERIAL_DEF <Material name> {<isotope name> <concentration>}

MATERIAL_DEF            MAT_FUEL         U235  2.48293E-04
MATERIAL_DEF            MAT_FUEL         U238  7.65991E-03
MATERIAL_DEF            MAT_FUEL          O16  2.95566E-02

MATERIAL_DEF            MAT_CLAD         ZR90  4.26303E-03
MATERIAL_DEF            MAT_CLAD         FE56  1.36597E-05

MATERIAL_DEF            MAT__MOD           H1  4.96224E-02
MATERIAL_DEF            MAT__MOD          O16  2.48112E-02
```

Figure 15. Sample Assignment Input File for a 2D Problem with Microscopic Cross Sections for PROTEUS-SN

6.3 T/H Input Files

```
PRESSURE                15.5      ! MPa
INLET_TEMPERATURE       286.0     ! deg-C
MASS_FLOWRATE           82.12102  ! g/sec

TOTAL_POWER             17.67516  ! MW
Relative_Power          1.

Number_of_Channels      1
Number_of_AxialNodes    16

Pins_In_Assembly        289
Fuel_Pins_in_Assembly   264
GT_Pins_in_Assembly     24
IT_Pins_in_Assembly     1
Pin_Pitch               1.26      ! cm
Assembly_Pitch          21.606    ! cm
Active_Fuel_Height      367.2     ! cm

Fuel_Pin_Geometry       3   0.41195 0.47585 0.0571 ! cm
GT_Pin_Geometry         2   0.561   0.61295        ! cm
IT_Pin_Geometry         2   0.559   0.605          ! cm
```

Figure 16. Sample T/H Input File

```
! REGION_ALIAS  <Name of assembly region> <Name of Material
! ------------------------------------------------------------------------
FDK_REGION_ALIAS    REGION_FUEL                  FUEL
FDK_REGION_ALIAS    REGION_GAP                   GAP
FDK_REGION_ALIAS    REGION_CLAD                  CLAD
FDK_REGION_ALIAS    REGION_MOD1                  COOL
FDK_REGION_ALIAS    REGION_GT                    AVERAGE
FDK_REGION_ALIAS    REGION_IT                    AVERAGE
FDK_REGION_ALIAS    REGION_REFL                  NONE
```

Figure 17. Sample T/H Assignment Input File

6.4 Kinetics Input

```
DELAY_XS_FILE            c5g7_7g.DLAYXS.ascii
INITIAL_MATERIAL_FILE    step000.assignment
TFSP_TOLERANCE_FLUX      1.0E-5
TFSP_ITERATIONS       50

! TIME_STEP  <end time> <intervals> <assignment file>  <hdf5 storage file at end of time step>
TIME_STEP    0.01      1           step001.assignment
TIME_STEP    0.02      1           step002.assignment
TIME_STEP    0.03      1           step003.assignment
TIME_STEP    0.04      1           step004.assignment
TIME_STEP    0.05      1           step005.assignment     data005.bin
TIME_STEP    0.06      1           step006.assignment
TIME_STEP    0.07      1           step007.assignment
TIME_STEP    0.08      1           step008.assignment
TIME_STEP    0.09      1           step009.assignment
TIME_STEP    0.10      1           step010.assignment     data010.bin
```

References

1. G. PALMIOTTI, E.E. LEWIS, and C.B. CARRICO, "VARIANT: VARIational Anisotropic Nodal Transport for Multidimensional Cartesian and Hexagonal Geometry Calculation," ANL-95/40, Argonne National Laboratory, 1995.
2. E.R. SHEMON, M.A. SMITH, and C.H. LEE, "PROTEUS-SN User Manual," ANL/NE-14/6 Rev. 3.0, Argonne National Laboratory, September 2016.
3. M.A. SMITH and E.R. SHEMON, "User Manual for the PROTEUS Mesh Tools," ANL/NE-15/17 Rev. 2.0, Argonne National Laboratory, September 2016.
4. CUBIT Web Page, www.cubit.sandia.gov.
5. MPICH Web page, http://www.mpich.org.
6. METIS Web page, http://glaros.dtc.umn.edu/gkhome/metis/metis/overview.
7. The HDF Group. Hierarchical Data Format, version 5, 1997-2014. http://www.hdfgroup.org/HDF5.
8. E.R. SHEMON, M.A. SMITH, and C.H. LEE, "PROTEUS-SN Methodology Manual," ANL/NE-14/5, Argonne National Laboratory, June 2014.
9. A. MARIN-LAFLECHE, M. A. SMITH, and C. H. LEE, "PROTEUS-MOC: A 3D Deterministic Solver Incorporating 2D Method of Characteristics," Proc. M&C2013, Sun Valley, Idaho, May 5-9, 2013.
10. Y. S. JUNG and W. S. YANG. "A Consistent CMFD Formulation for the Acceleration of Neutron Transport Calculations Based on the Finite Element Method," Nucl. Sci. Eng., **185**, 307-324, 2018.
11. C.H. LEE et al, "FY17 Status Report on NEAMS Neutronics Activities," ANL/NE-17/28, Argonne National Laboratory, September 2017.
12. M.H. BAE, "Development of Triangle-based Polynomial Expansion Nodal SP_3 Method for Hexagonal Core Transport Calculation," Seoul National University, 2010.
13. C.H. LEE et al, "FY17 Status Report on NEAMS Neutronics Activities," ANL/NE-17/28, Argonne National Laboratory, September 2017.
14. C.H. LEE and W.S. YANG, "MC^2-3: Multigroup Cross Section Generation Code for Fast Reactor Analysis," *Nucl. Sci. Eng*., **187**, 268-290, June 2017.
15. G. MARLEAU, R. ROY, and A. HÉBERT, "DRAGON: A Collision Probability Transport Code for Cell and Supercell Calculations," Report IGE-157, Institut de génie nucléaire, École Polytechnique de Montréal, Montréal, Québec, 1994.
16. J. LEPPANEN, "Serpent – a Continuous-energy Monte Carlo Reactor Physics Burnup Calculation Code," VTT Technical Research Centre of Finland, June 18, 2015.
17. P.K. Romano et al., "OpenMC: A State-of-the-Art Monte Carlo Code for Research and Development," *Ann. Nucl. Energy*, **82**, 90, 2015.
18. N.E. STAFF, P.K. ROMANO, C.H. LEE, and T.K. KIM, "Verification of Mixed Stochastic/Deterministic Approach for Fast and Thermal Reactor Analysis," ICAPP 2017, Fukui and Kyoto, Japan, April 24-27, 2017.

Nuclear and Science Engineering Division
Argonne National Laboratory
9700 South Cass Avenue, Bldg. 208
Argonne, IL 60439-4842

www.anl.gov

Argonne National Laboratory is a U.S. Department of Energy
laboratory managed by UChicago Argonne, LLC