

PROTEUS-SN User Manual

Revision 2.0

Nuclear Engineering Division

About Argonne National Laboratory

Argonne is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC under contract DE-AC02-06CH11357. The Laboratory's main facility is outside Chicago, at 9700 South Cass Avenue, Argonne, Illinois 60439. For information about Argonne and its pioneering science and technology programs, see www.anl.gov.

DOCUMENT AVAILABILITY

Online Access: U.S. Department of Energy (DOE) reports produced after 1991 and a growing number of pre-1991 documents are available free via DOE's SciTech Connect (<http://www.osti.gov/scitech/>)

Reports not in digital format may be purchased by the public from the National Technical Information Service (NTIS):

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

www.ntis.gov

Phone: (800) 553-NTIS (6847) or (703) 605-6000

Fax: (703) 605-6900

Email: orders@ntis.gov

Reports not in digital format are available to DOE and DOE contractors from the Office of Scientific and Technical Information (OSTI):

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831-0062

www.osti.gov

Phone: (865) 576-8401

Fax: (865) 576-5728

Disclaimer

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor UChicago Argonne, LLC, nor any of their employees or officers, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of document authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof, Argonne National Laboratory, or UChicago Argonne, LLC.

PROTEUS-SN User Manual

Revision 2.0

prepared by
Emily R. Shemon, Micheal A. Smith and Changho Lee
Nuclear Engineering Division, Argonne National Laboratory

July 31, 2015

ABSTRACT

PROTEUS-SN is a three-dimensional, highly scalable, high-fidelity neutron transport code developed at Argonne National Laboratory. The code is applicable to all spectrum reactor transport calculations, particularly those in which a high degree of fidelity is needed either to represent spatial detail or to resolve solution gradients. PROTEUS-SN solves the second order formulation of the transport equation using the continuous Galerkin finite element method in space, the discrete ordinates approximation in angle, and the multigroup approximation in energy. PROTEUS-SN's parallel methodology permits the efficient decomposition of the problem by both space and angle, permitting large problems to run efficiently on hundreds of thousands of cores. PROTEUS-SN can also be used in serial or on smaller compute clusters (10's to 100's of cores) for smaller homogenized problems, although it is generally more computationally expensive than traditional homogenized methodology codes. PROTEUS-SN has been used to model partially homogenized systems, where regions of interest are represented explicitly and other regions are homogenized to reduce the problem size and required computational resources.

PROTEUS-SN solves forward and adjoint eigenvalue problems and permits both neutron upscattering and downscattering. An adiabatic kinetics option has recently been included for performing simple time-dependent calculations in addition to standard steady state calculations. PROTEUS-SN handles void and reflective boundary conditions. Multigroup cross sections can be generated externally using the MC²-3 fast reactor multigroup cross section generation code or internally using the cross section application programming interface (API) which can treat the subgroup or resonance table libraries.

PROTEUS-SN is written in Fortran 90 and also includes C preprocessor definitions. The code links against the PETSc, METIS, HDF5, and MPICH libraries. It optionally links against the MOAB library and is a part of the SHARP multi-physics suite for coupled multi-physics analysis of nuclear reactors.

This user manual describes how to set up a neutron transport simulation with the PROTEUS-SN code. A companion methodology manual describes the theory and algorithms within PROTEUS-SN.

TABLE OF CONTENTS

Abstract	i
Table of Contents	iii
List of Figures	v
List of Tables.....	v
1. Introduction	1
1.1 Code Summary.....	1
1.2 Input File Summary	2
1.3 Output File Summary.....	2
1.4 Disclaimer	3
2. Acquiring and Installing the Code	4
2.1 Acquiring the Code	4
2.2 External Library Dependencies.....	4
2.2.1 MPICH.....	4
2.2.2 METIS	4
2.2.3 PETSc	5
2.2.4 HDF5	5
2.2.5 Cross Section API.....	5
2.2.6 MOAB	5
2.3 Compiling the Code	6
2.3.1 Customization of Makefile.arch	6
2.3.2 Customization of PROTEUS_Preprocess.h	7
2.3.3 Building the Targets	7
2.3.4 Verification Checks	8
2.3.5 Recommended Compilers and Architectures	8
3. Code Execution Syntax	10
3.1.1 Serial Jobs.....	10
3.1.2 Parallel Jobs.....	10
3.1.3 Kinetics Jobs.....	10
3.1.4 Converting PMO to HDF5 Output	11
3.1.5 Other Jobs.....	11
4. Input File Descriptions.....	12
4.1 Driver Input File (*.inp).....	12
4.1.1 Required Input	12
4.1.2 Optional Input.....	14
4.1.3 Assigning Boundary Conditions.....	19
4.1.4 Parallel Partitioning Control.....	20
4.1.5 Processing Cross Sections with the Cross Section API.....	21
4.2 Cross Section File (*.ISOTXS, *.anlxs)	21
4.3 Mesh File (*.nemesh, *.ascii, *.pntmesh, *.bgpmesh)	22
4.3.1 Mesh File Types	22
4.3.2 Mesh Generation.....	22
4.3.3 Element Types	23
4.3.4 Element Blocks and Sidesets.....	23

4.3.5 Issues to Consider While Meshing	24
4.4 Material Assignment File (*.assignment)	25
4.4.1 Defining Materials	25
4.4.2 Assigning Materials to Blocks.....	26
4.4.3 Assigning Block Properties	26
4.5 Mesh Partitioning File (*.partitioning)	28
4.6 Fixed Source File (*.fixeds).....	28
4.7 Kinetics Input File (kinetics.inp).....	29
5. Output.....	35
5.1 Main Output File	35
5.2 Full Solution.....	35
5.3 VTK Files for Mesh and Cubature.....	37
5.4 Region Edit Integrals.....	37
6. Example Problem.....	38
7. Summary	44
References	45
Appendix A. Angular Cubatures in PROTEUS-SN	46
Appendix B. Example PROTEUS-SN Driver Input File: “sn2nd.inp”	54
Appendix C. Cross Section File Format for PROTEUS-SN: “anlxs”	55
Appendix D. Mesh File Format Specification for PROTEUS-SN: “nemesh”.....	58
Appendix E. Mesh File Format Specification for PROTEUS-SN: “ascii”	63
Appendix F. Example PROTEUS-SN Text Output File.....	66
Appendix G. Finite Elements Used in PROTEUS-SN Along with Vertex and Surface Numbering	71

LIST OF FIGURES

Figure 1. Recommended Customizations in Makefile.arch.	6
Figure 2. Assigning Boundary Conditions at Runtime in Driver Input File.	19
Figure 3. Example Mesh Showing Blocks and Sidesets for 1/8-th Pin Cell.	23
Figure 4. Non-conformal Mesh (left) and Conformal Mesh (right).	24
Figure 5. Material Definition in the Assignment File (Atom Fractions).	25
Figure 6. Material Definition in the Assignment File (Weight Fractions).	25
Figure 7. Recursive Material Definition in the Assignment File.	26
Figure 8. Assigning a Material to a Block in the Assignment File.	26
Figure 9. Assigning Total Atom Density to a Block in the Assignment File.	27
Figure 10. Assigning Total Mass Density to a Block in the Assignment File.	27
Figure 11. Example PROTEUS-SN Assignment File.	27
Figure 12. Example Kinetics Input File with No Fixed Source.	33
Figure 13. Example Kinetics Input File with Fixed Source.	34
Figure 14. Visualization of Geometry and Mesh Used to Analyze the Advanced Test Reactor.	36
Figure 15. ATR fast (left) and Thermal (right) Flux Calculated by PROTEUS-SN.	36
Figure 16. Visualization of 2D Fast (Left) and Thermal (Right) Fluxes for ATR Calculation.	37
Figure 17. Driver Input File for Example Case.	39
Figure 18. Cross Section File for Example Case.	40
Figure 19. Mesh File for Example Case.	41
Figure 20. VTK Mesh (left) and VTK Mesh with Block Coloring (right) for Example Case.	42
Figure 21. Material Assignment File for Example Case.	42
Figure 22. Nu*fission Rate in Test Case Plotted in VisIt.	43

LIST OF TABLES

Table 1. Possible Customizations in PROTEUS_Preprocess.h	7
Table 2. List of PROTEUS-SN Makefile Targets (Executables).	7
Table 3. Required Input for Driver Input File.	12
Table 4. Auxiliary Input Controls in Driver Input File.	14
Table 5. Fixed Source File Format.	29
Table 6. Kinetics Input Options.	29

1. Introduction

PROTEUS-SN is a highly scalable, discrete ordinates-based 3D neutron transport code targeted for high-fidelity nuclear reactor applications. This document provides a comprehensive user guide for PROTEUS-SN. The purpose of this user guide is to familiarize new users with the basics of carrying out simulations using the PROTEUS-SN code.

1.1 Code Summary

PROTEUS-SN is a three-dimensional deterministic neutron transport code which solves the second order formulation of the neutron transport equation. The second order “even-parity” form of the transport equation is discretized using the continuous Galerkin finite element method in space, the discrete ordinates approximation in angle, and the multigroup approximation in energy. A finite element mesh, discrete ordinates cubature, and multigroup cross section data set must therefore be provided.

PROTEUS-SN solves both forward and adjoint eigenvalue problems with Tchebychev acceleration and a fixed Wielandt shift. Fixed source problems are also treated and the code handles neutron upscattering and downscattering. An adiabatic kinetics option has recently been included in the code for performing simple time-dependent calculations in addition to standard steady state calculations. PROTEUS-SN handles void and reflective boundary conditions. PROTEUS-SN does not perform gamma radiation transport but accounts for gamma power production assuming that the gammas are absorbed at the location where they are emitted.

The overall solution technique in PROTEUS-SN is discretization of the even-parity transport equation into a linear system of equations for the even-parity component of the angular flux. The within-group system of equations is solved using the SSOR-preconditioned conjugate gradient method in the PETSc library, where scattering iterations are performed using Richardson iteration with diffusion synthetic acceleration. The multigroup system of equations is solved using Gauss-Seidel iteration. PROTEUS-SN permits parallelization of neutron transport problems in both space and angle which significantly reduces in the per-processor memory load for a given problem. The efficient parallel techniques in PROTEUS-SN permit the solution of very large problems. PROTEUS has been demonstrated to solve 500 billion - 1 trillion degrees of freedom on leadership computing resources.

PROTEUS-SN is written in Fortran 90 source code and also includes C preprocessor definitions. The code links against various open source libraries including a parallel linear algebra solver (PETSc), a mesh decomposition library (METIS) and a portable output library (HDF5).

1.2 *Input File Summary*

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

- a. Driver input file
- b. Multigroup cross section file or cross section library file
- c. Mesh file
- d. Material assignment file
- e. (Parallel execution mode only and optional) Mesh partitioning file
- f. (Kinetics only) Kinetics input file
- g. (Fixed source only) Fixed source input file

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 multigroup 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 latter is the native file format for Argonne's MC²-3 fast reactor multigroup cross section generation code [1] and is also the compatible format for Argonne's well-established neutronics code, DIF3D [2].

The mesh file is a formatted file containing information about the finite element mesh. PROTEUS-SN handles unstructured meshes with the triangular (2D), quadrilateral (2D), triangular prism (3D), tetrahedral (3D) or hexahedral (3D) elements. A combination of elements of the same dimensionality can be used in the same mesh provided they are in different blocks. The basis functions are polynomial, and linear or quadratic order is typically used.

The material assignment file is a keyword-based free format text file that specifies the material assignment to regions in the mesh, as well as defining properties such as density.

The mesh partitioning file is relevant only for parallel simulations. It is automatically generated during a simulation and can be re-used in future simulations using the same mesh and mesh decomposition.

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

The fixed source input file is relevant only for fixed source simulations. It can be used in either steady state or kinetics mode.

1.3 *Output File Summary*

PROTEUS-SN produces a main text output file containing an echo of the inputs, eigenvalue iteration history, and MPI timing summaries. Optionally, the full solution is exported to HDF5

file format and can be visualized using VisIt [3] with the “UNIC” plugin. Additionally, block-integrated quantities (power, fission, absorption, and group flux) can be exported. Finally, the cubature and mesh can also be individually exported to a VTK [4] format.

1.4 Disclaimer

As of the time of this document’s release, PROTEUS-SN has not had sufficient testing and development to be considered as a production level tool. The reliability of PROTEUS is therefore not guaranteed. Additionally, the accuracy of PROTEUS has not been assessed on a wide range of reactor physics problems. In many cases, performance and efficiency have been sacrificed in favor of providing a working capability which prevents one from making a realistic computational effort comparison to other available tools. In many cases, reliable algorithms have been chosen to compensate for memory shortages on high performance computing machines which degrade the wall clock performance of the existing code.

2. Acquiring and Installing the Code

This chapter briefly describes how to obtain and install PROTEUS-SN. It lists the various 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-SN 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.

PROTEUS-SN is dependent on various open source libraries which are freely available and therefore not included in the distribution. Specific versions of the libraries may be required to ensure compatibility with PROTEUS-SN, given that the various packages change function interfaces periodically. These external package dependencies as well as compiler recommendations are described in the next section.

2.2 External Library Dependencies

PROTEUS-SN achieves distributed memory parallelization using the Message Passing Interface (MPI) protocol and uses the MPI2 standard typically through the MPICH2 [5] library implementation. PROTEUS-SN also has essential dependencies on two external packages, the mesh partitioning package METIS [6] and the parallel linear algebra solver PETSc [8]. Additionally, PROTEUS-SN optionally interfaces with data format library HDF5 [9] and the mesh frameworks tool MOAB [10]. We briefly discuss these packages here. For help installing any of the packages for use with PROTEUS-SN, refer to the documentation included with the distribution, or contact nera-software@anl.gov.

2.2.1 MPICH

PROTEUS-SN supports the MPI2 standard with most of the testing work done with MPICH2 version 1.4 or later. The MPICH library provides the ability to perform parallel communication between processors in distributed memory mode. Various versions of MPICH2 are available at <http://www.mcs.anl.gov/research/projects/mpich2staging/goodell/downloads/tarballs/>. Refer to the installation documentation in the distribution for guidance on which version to use.

2.2.2 METIS

PROTEUS-SN uses the METIS package [6,7] 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-SN is currently configured to use METIS 4.0. Later versions of METIS incorporate different function interfaces which are not guaranteed to work with the PROTEUS-SN implementation.

2.2.3 PETSc

The PETSc [8] package is used by PROTEUS-SN to solve an assembled parallel matrix-vector preconditioning system, and at this time PROTEUS-SN is configured to run with PETSc version 3.1 or PETSc version 3.4 due to major changes in the PETSc interfaces in later versions. Work is being done to update PROTEUS-SN to work with later versions of PETSc. We note that the current PROTEUS distribution assumes the user is linking against PETSc 3.1. To instead link to PETSc 3.4, the user must configure PROTEUS at compile time by defining the `PROTEUS_Use_PETSC3_4` pre-processing variable in `PROTEUS_Preprocess.h`.

While advanced users can set up more complicated preconditioners, PROTEUS-SN by default uses the Symmetric Successive Over-Relaxation (SSOR) preconditioned Conjugate Gradient (CG) solver of PETSc. The primary benefit of this solver is its well-demonstrated parallel scalability and ease of use. The primary weaknesses include exceedingly high memory requirements and failure to efficiently utilize the full power of the platform which it is deployed upon (5% of peak performance). While we have researched ways into overcoming these limitations by replacing this preconditioner, time constraints have prevented us from making alternate preconditioner options available at this time.

2.2.4 HDF5

The HDF5 [9] software is used by PROTEUS-SN to store the flux solution for follow-up visualization purposes. Recent versions (1.10.0+) of VisIt [12] include a plugin called “UNIC” which can be used to directly read this data (files with .h5 extensions). The HDF5 software can be problematic for most novice users to install, and we strongly advise consulting nera-software@anl.gov if difficulties arise. PROTEUS-SN has successfully run with HDF5-1.8.10 and later versions should also work provided the function interfaces do not change.

2.2.5 Cross Section API

The Cross Section API is an external library developed by Argonne National Laboratory which processes multigroup cross sections using the subgroup or resonance table method. In order to use the Cross Section API, a transport code should provide a one-group fixed source transport solver to it. The Cross Section API can be requested with PROTEUS-SN starting in June 2014. PROTEUS-SN includes an interface to the Cross Section API such that multigroup cross sections can be calculated on-the-fly for the exact heterogeneous geometry of interest rather than providing pre-processed multigroup data. This implementation is new and considered a “beta” version, so use it with caution.

2.2.6 MOAB

The MOAB [10] package can be used by PROTEUS-SN to load meshes directly from a variety of mesh generator formats and perform dynamics coupling. Unfortunately, the interface to MOAB is not stable and could not be released at this time.

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-SN 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. Figure 1 shows an excerpt from *Makefile.arch* for a hypothetical machine called “MyMachineName”.

```

ifeq (${UNAME}, MyMachineName)
# Set the compilers to your MPICH installation
FC = /software/mpich2-1.2/O-intel-11.1/bin/mpif90
F77 = /software/mpich2-1.2/O-intel-11.1/bin/mpif90
CC = /software/mpich2-1.2/O-intel-11.1/bin/mpicc
LD = /software/mpich2-1.2/O-intel-11.1/bin/mpif90
# Set compiler options
CPPDEFS =
FFLAGS = -O3
CFLAGS =
LDFLAGS = -lstdc++
# Set the PETSc installation locations
LPETSC_LOC=/software/petsc-3.1-p2
PETSC_ARCH=O-intel-11.1
PETSC_DIR=${LPETSC_LOC}/${PETSC_ARCH}
include ${PETSC_DIR}/conf/variables

# Set other external libraries not included in the PETSc installation
METIS_LIB=/software/metis/O-intel-11.1/libmetis.a
HDF5_DIR=/software/hdf5/O-intel-11.1
HDF5_LIB=-L${HDF5_DIR}/lib -lhdf5hl_fortran -lhdf5_hl -lhdf5_fortran -lhdf5 -lz -lm
MOAB_DIR=/software/MOAB/g-intel-11.1
IMESH_DIR=${MOAB_DIR}/lib
# Subgroup API (optional)
SGAPI_DIR=./API_SubGroup
SGAPI_LIB=${SGAPI_DIR}/libsubgroup.a
# Meshtools (optional)
MESHTOOLS_DIR=./MeshTools
endif

```

Figure 1. Recommended Customizations in Makefile.arch.

We recommend that you 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 section shown above, you should assign the MPICH compiler locations, PETSc installation root location, PETSc architecture (such that the PETSC_DIR variable is the full install path), and METIS, HDF5, MOAB, and Cross Section API

installation locations. Note that the F77 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* shown in Table 1 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 (MOAB, HDF5, and Cross Section API).

Table 1. Possible Customizations in *PROTEUS_Preprocess.h*

Preprocessing Directive Variable	Meaning (if defined)
PROTEUS_Debug	Enable debug printing
PROTEUS_CallTree	Enable call tree printing
PROTEUS_InterfacesWithMOAB	Enable MOAB (reading .h5m mesh, multiphysics) (keep disabled unless you have the MOAB version)
PROTEUS_InterfacesWithHDF5	Enable HDF5 for output (output .h5 type)
PROTEUS_InterfacesWithSGAPI	Enable Cross Section API
PROTEUS_Use_PETSC3_4	Link against PETSc 3.4 instead of PETSc 3.1
PROTEUS_SQS_Precursors_By_Block	Enable storage by block (keep disabled)

Note that the HDF5 and Cross Section library locations must also be specified in *Makefile.arch* if they are uncommented in *PROTEUS_Preprocess.h*. Since the MOAB interface is not stable for release at this time, the `#define PROTEUS_InterfacesWithMOAB` line should be commented out.

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 2. Alternatively, type “**make <target>**” at the command line to create only a specific target.

Table 2. List of PROTEUS-SN Makefile Targets (Executables).

Target	Application
sn2nd.x	Steady state version of PROTEUS-SN
sn2nd_kinetics.x	Kinetics version of PROTEUS-SN

sn2nd_sg.x	Steady state version of PROTEUS-SN using Cross Section API for cross section generation
regression_gauss.x	Testing purposes (regression tests)
regression_libs.x	Testing purposes (regression tests)
PMO_AssembleHDF5.x	Converts set of .pmo files to a single .h5 file

We note that parallel *make* execution is possible on many platforms using “make -j4 <target>”.

2.3.4 Verification Checks

Both unit tests and benchmark problems are available to check whether installation was successful. The benchmark problems are provided in the /benchmark directory. The directory “SN2ND” contains 7 steady state benchmark problems. The directory “SN2ND_Kinetics” contains 4 kinetics benchmark problems. Reference solutions are provided for a variety of parallel decompositions (typically using up to 12 processors).

To perform the full suite of verification checks including unit tests and benchmark problems, use the command “**make verify**”. To run only the steady state benchmark problems, use the command “**make bench**”. To run only the kinetics benchmark problems, use the command “**make bench_kinetics**”. The benchmark problems will be performed by a script and compared to the reference solutions.

2.3.5 Recommended Compilers and Architectures

PROTEUS-SN has been regularly compiled with the Intel 10.1.15 compiler on Intel hardware. We do not suggest using Intel compiler versions 10.1.20 through all of version 11 (except 11.1.080) as they were unreliable with the various dependencies and PROTEUS-SN source code. Intel versions 12.1+ are required for use with the Cross Section API as it contains FORTRAN 2003 constructs and requires a FORTRAN 2003-compliant compiler.

PROTEUS-SN has also been compiled with the latest IBM XL Fortran compilers for the BlueGene P and Q systems (Intrepid and Mira at ALCF). Additionally, PROTEUS-SN has been compiled on a Cray XT5 system (Jaguar at ORNL). The code should work on most Linux workstations, clusters, and laptops as well as Macs.

The current version of the code does not utilize shared memory parallelism and thus does not utilize threading or GPUs.

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. Code Execution Syntax

3.1.1 Serial Jobs

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

```
$> sn2nd.x -input /path/to/mydriver.inp
```

The `-input` flag argument specifies that PROTEUS-SN 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-SN looks for the driver input file in the current working directory with the default name “`sn2nd.inp`”. The output can be redirected from standard output to a file using the `-output` flag.

3.1.2 Parallel Jobs

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

```
$> mpiexec -n 8 sn2nd.x -input /path/to/mydriver.inp
```

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 value of `SEGMENT_ANGLE` in the driver input. The problem is first partitioned by angle into `SEGMENT_ANGLE` processors. The case of `SEGMENT_ANGLE=0` is a special case discussed in Section 4.1.4. For `SEGMENT_ANGLE>0`, the remaining factor of processors (in this case, $8/\text{SEGMENT_ANGLE}$) 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.

3.1.3 Kinetics Jobs

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

```
$> mpiexec -n 4 sn2nd_kinetics.x -input /path/to/mydriver.inp
```

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 sn2nd_kinetics.x -input /path/to/mydriver.inp -  
kinetics MyKinetics.inp
```

3.1.4 Converting PMO to HDF5 Output

The `PMO_AssembleHDF5.x` executable converts a set of `.pmo` files to a single HDF5 file. It runs in serial and requires three arguments:

```
$> PMO_AssembleHDF5.x mymesh.nemesh color.inp myfluxfile.h5
```

The first argument to `PMO_AssembleHDF5.x` is the mesh file (`.nemesh`, `.ascii`, `.pntmesh`, or `.bgpmesh`) which was used in the problem. The second input is an optional color map. The color map is a formatted text file containing on each line the 16-character region name (`REGION_XXXXXXXX`), followed by a space and then the color number (an integer). Executing the program with no arguments will list the color number options. If you do not wish to provide a color map, keep the “color.inp” argument but remove any files with the same name in the directory to avoid runtime errors about incorrect file formatting.

The last argument is the name of the desired HDF5 output file. It must have the same prefix as the prefix of the PMO files since the executable will automatically search the directory for PMO files with the same prefix. For example, to convert several files named “MyOutput0000001.pmo”, “MyOutput0000002.pmo”, etc. to an HDF5 file, the last argument in the command should be “MyOutput.h5”.

3.1.5 Other Jobs

Job execution on leadership computing platforms and other queued platforms should follow the syntax required by the relevant job submission system.

4. Input File Descriptions

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

- **Driver input file**
- **Cross section file**
- **Mesh file**
- **Material assignment file**

Additionally, three other input files are used for parallel runs, kinetics jobs, and fixed source problems:

- (Parallel mode, optional) **Mesh partitioning file**
- (Kinetics jobs only) **Kinetics input file**
- (Fixed source jobs only) **Fixed source input file**

An example driver input file is provided in Appendix B.

4.1 Driver Input File (*.inp)

Upon execution, PROTEUS looks for the driver input file, “sn2nd.inp”, in the current working directory. This input file is a plain text (ASCII) file that drives the PROTEUS 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 “sn2nd_5g_mesh1.inp”, add the command line option “-input sn2nd_5g_mesh1.inp” at execution to alert PROTEUS of the driver input file name.

4.1.1 Required Input

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

Table 3. Required Input for Driver Input File

Keyword	Input Data	Default Value	Description
SN_TYPE	<i>1D Options:</i> LEGENDRE 1-D DOUBLE LEGENDRE <i>2D/3D Options:</i> CARLSON_EM CARLSON_LM	-	Specifies the type of SN cubature to use.

	CARLSON_EQUALW LEG-TCHEBY X_DIR_LEG-TCHEBY DOUB_LEG-TCHEBY TRI_LEG-TCHEBY LEBEDEV-LAIKOV TEG_CORNER_EQWT TEG_CORNER_LSQ TEG_CENTROID_EQWT TEG_CENTROID_LSQ THURGOOD_TN THURGOOD_LEASTSQ COBE_EVEN_EQWT COBE_EVEN_LSQ COBE_ODD_EQWT COBE_ODD_LSQ PYRAMID_CARLSON COBE_C_EVEN_EQWT COBE_C_EVEN_LSQ COBE_C_ODD_EQWT COBE_C_ODD_LSQ		
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.
SOLVE_TYPE	FORWARD ADJOINT BOTH	BOTH	Only applicable for steady state jobs. Specifies which calculations to perform (forward, adjoint, or both).
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
--	--	--	---------------------------------

The SN_TYPE input describes the type of cubature to be used in the calculation. Appendix A contains additional information about the usage of these cubatures, including spherical harmonics integration order and number of discrete directions (ordinates). The number of directions that will be used in PROTEUS-SN 3D calculations is given in the 2π column of each table. From Table 3, there are two controls to select the resolution settings of a cubature, THETA_RESOLUTION and PHI_RESOLUTION, noting that the latter is only important for product cubatures (see Appendix A).

The SOLVE_TYPE input in Table 3 is only appropriate for steady state eigenvalue problems and allows the independent selection of the forward or adjoint flux solutions, or consecutive solves of both. In the case of both, the resulting flux file will only contain the forward flux solution. For kinetics and dynamics simulations, auxiliary drivers (executables) are provided which eliminate the need for this input option.

The remaining inputs in Table 3 specify the paths to the auxiliary input files (cross sections, mesh, and material assignments). Relative file paths are permitted and are assumed to be relative to the driver input file. For the mesh, an external file is always required using the SOURCEFILE_MESH input which has four native file formats: nemesh, ascii, pntmesh, and bgpmesh. These file formats are discussed later in this chapter. The SOURCEFILE_XS and SOURCEFILE_MATERIAL specify the cross section file and material assignment file, respectively.

4.1.2 Optional Input

Other optional driver-level input options are listed in Table 4. They can be classified into the following groups: debugging options, parallelization options, iterative solver options, cross section options, boundary condition options, multi-physics coupling options, and output file options. Brief explanations of each option as well as the default values are provided.

Table 4. Auxiliary Input Controls in Driver Input File

Keyword	Input Data	Default value	Description
<i>Auxiliary Input File Options</i>			
SOURCEFILE_FIXEDSOURCE	<Max 128 Characters>	-	Specified path to fixed source input file for a fixed source problem
<i>Debugging Options</i>			

Keyword	Input Data	Default value	Description
SKIPSOLVE	YES NO	NO	Specifies whether to set up the problem only (YES) or to set up the problem and solve it (NO).
DEBUG_PRINT_LEVEL	<Integer greater than or equal to 0>	0	Specify level of debug printing for entire routine.
DEBUG_PRINT_SETUP	<Integer greater than or equal to 0>	0	Specify level of debug printing of setup related operations.
DEBUG_PRINT_FORMATION	<Integer in range [0..10]>	0	Controls debug printing during the tracking/matrix formation (max 10).
DEBUG_PRINT_OUTER	<Integer in range [0..10]>	0	Controls the debug printing during the outer iterations (max 10).
<i>Parallelization Options</i>			
SEGMENT_ANGLE	<Integer greater than or equal to 0>	0	The number of segments to attempt in the angular approximation. 0 = Code decides max parallelization 1=Serial 2 and up = # Processors (segments) in angle
SOURCEFILE_MESHPART	<Max 128 Characters>	-	For space-parallel simulations, specifies the UNIX file path to a pre-computed partitioning index file for the mesh.
<i>Iterative Solver Options</i>			
EIGENVALUE_GUESS	<Real Value>	1.0	Guess for the initial eigenvalue.
USE_TCHEBYCHEV_ACCEL	YES NO	YES	Controls Tchebychev acceleration of the fission source.
ITERATIONS_FISSION	<Integer greater than or equal to 0>	100	Maximum number of outer (fission source) iterations.

Keyword	Input Data	Default value	Description
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
ITERATIVE_IMPROVEMENT	<Real Value>	0.1	Upper limit targeted iterative improvement within each call. Value will be adjusted down as needed by the code.
ITERATIONS_MAXUPSCATTER	<Integer greater than or equal to 0>	5	Maximum Gauss-Seidel "upscatter" iterations per outer (not a fixed iteration scheme).
ITERATIONS_MINUPSCATTER	<Integer greater than or equal to 0>	5	Minimum Gauss-Seidel "upscatter" iterations per outer (not a fixed iteration scheme).
ITERATIONS_SCATTERING	<Integer greater than or equal to 0>	3	Maximum number of source iterations per within group iteration (not a fixed iteration scheme).
ITERATIONS_SA_CG	<Integer greater than or equal to 0>	1000	Maximum number of within-group CG iterations to perform.
ITERATIONS_PETSC	<Integer greater than or equal to 0>	1000	Maximum number of CG iterations to use in all levels of the PETSc preconditioner tree.
USE_PRECONDITIONER	SSOR ICC	SSOR	Select the preconditioner in PETSc. SSOR and ICC are the only two allowed at this time.
USE_DSA	YES NO	YES	Applies synthetic diffusion acceleration to the source iteration.
RELAX_SPATIAL	YES NO	YES	Relaxes spatial matrix integration (minimize effort).

Keyword	Input Data	Default value	Description
USE_WEILANDT_SHIFT	<Real value greater than eigenvalue>	-	Apply a Weilandt shift to the system to accelerate convergence. Value must be greater than largest eigenvalue in system to maintain positive definite system.
BASIC_BWO	YES NO	NO	Applies a basic bandwidth optimization to mesh (reorder mesh for minimum bandwidth with ICC preconditioner). <i>This option is seldom used to due recommended SSOR preconditioner.</i>
<i>Cross Section Options</i>			
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.
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 greater than or equal to 0>	0	Legendre expansion order of the scattering kernel.

Keyword	Input Data	Default value	Description
SCATTERING_CHANGETOTAL	YES NO	no	For some cross section sets, the total cross section is expanded into spherical harmonics. YES will auto-correct the within group scattering components.
SCATTERING_SETUPTYPE	0 1 2	0	Indicates whether the 2L+1 factor is not included (0), multiplied (1), or divided (2).
<i>Boundary Condition Options</i>			
BC_ALIAS	side_set_<#> <BCTYPE>		Assigns boundary condition at runtime to side set #. Valid BCTYPE names are VOID or REFLECTIVE.
<i>Multi-physics Coupling Options (SHARP version only)</i>			
SN2ND_COMPUTES_DENSITY	YES NO	NO	For coupling simulations, tells PROTEUS-SN to compute the densities based on temperature feedback.
<i>Output File Options</i>			
MESHSIMULATE	<Integer ≥ 0>	0	If this is non-zero, then a pntmesh will be exported using this value as the number of processors.
EXPORT_CUBATURE	YES NO	NO	Used to export the SN cubature to VTK file for visualization purposes.
EXPORT_XS_PRINT	YES NO	NO	Controls printing of XS data on rank 0.
EXPORT_MESH	YES NO	NO	Export the local NTmesh as "mesh.ascii" formatted file.
EXPORT_MESH_VTK	YES NO	NO	Export the local NTmesh to a VTK file for visualization.

Keyword	Input Data	Default value	Description
EXPORT_FLUX	YES NO	NO	Indicates whether full solution (scalar flux, reaction rates) are to be exported to file.
EXPORT_FILE	<Max 128 Characters ending in .pmo or .h5>	default.h5	Specifies the flux file name designation of a .h5 (requires HDF5) or .pmo file to export the solution.
EXPORT_EDITS	<Max 128 Characters>	(none)	Specifies that you would like to export the region edit integrals to an output file (i.e. region number, volume, flux, etc)
THERMAL_POWER	<Real>	1.0	Specifies the power normalization in Watts for the output file.

4.1.3 Assigning Boundary Conditions

The keyword BC_ALIAS in the driver input file is used to assign boundary conditions at runtime to the sidesets in the mesh. When converting an Exodus file to a PROTEUS-SN mesh format, the converted mesh will not contain the necessary boundary conditions. The boundary conditions should therefore be assigned using this input option. Figure 2 shows an example of how to assign boundary conditions to sidesets in the driver input file.

```
! Example usage of BC_ALIAS

! Assign b.c.'s of sidesets to VOID or REFLECTIVE
BC_ALIAS      side_set_0000001 REFLECTIVE      ! Sideset 1
BC_ALIAS      side_set_0000002 VOID           ! Sideset 2
```

Figure 2. Assigning Boundary Conditions at Runtime in Driver Input File.

Note the convention for referring to sidesets in PROTEUS-SN. If the sideset number in the mesh is 1, then the corresponding name in the driver input file should be “side_set_”, then that number prepended by 0’s to create 7 digits: side_set_0000001.

Possible boundary conditions are “VOID” or REFLECTIVE”. If your .ascii or .nemesh file format already contains the boundary condition assignment, this option can be used to override it.

4.1.4 Parallel Partitioning Control

PROTEUS-SN can partition the degrees of freedom with respect to both angle and space. It performs this partitioning on-the-fly after determining the number of available processors at execution time.

PROTEUS-SN always attempts to partition in angle first using the keyword `SEGMENT_ANGLE` to select the number of processors into which the angular work (number of angles) is divided. The number of angles simulated in PROTEUS-SN is only half the angles in the cubature due to symmetry of the even-parity angular flux. If unsure of the number of angles in the cubature, you can run the problem in serial with “`SKIPSOLVE YES`”, which will echo the number of angles in the output.

Thus, `SEGMENT_ANGLE` can be any positive integer which is a common factor of the total number of processors and the number of angles used to discretize the angular space. If this rule is not satisfied, PROTEUS aborts with a message stating that some of the processes were not used. The default setting (0) tells PROTEUS-SN to maximize the parallelization in angle, such that PROTEUS-SN assigns the minimum number of angles to each process, given the constraint that each process has the same number of angles. Thus the only rule in that case is that the number of processes used is a multiple of the number of angles.

Once the number of angle partitions has been determined, PROTEUS-SN computes the number of spatial processors per angle partition such that it uses the full number of processes. In this context, the mesh is decomposed into the same exact partition for all angle groups. PROTEUS-SN typically performs the spatial decomposition of the mesh at runtime; however, this step can be memory intense. An input partitioning file can be provided using `SOURCEFILE_MESHPART` to skip this runtime expense. See Section 4.5 regarding mesh partitioning for more details.

For best scaling performance, each processor’s angular partition should be assigned at least 2 angles. The optimal value of `SEGMENT_ANGLE` therefore depends on both the number of available processors and the number of angles in the cubature. Additionally, for best performance, each processor’s spatial partition should have at least 1500 vertices. This number has been shown experimentally to provide a good balance between work and communication. These two partition sizes suggest a soft upper limit on how many processors should be used to solve a given problem ($\text{max processors} = \text{\#vertices}/1500 * \text{\#angles}/2$).

Let us imagine that we have 16 processors available to solve a problem with 8 angles and 10,000 vertices. Possible decompositions include 1x16 (8 angles and 625 vertices per process), 2x8 (4 angles and 1,250 vertices per process) or 4x4 (2 angles and 2,500 vertices per process), or 8x2 (1 angles and 5,000 vertices per process). We note that the best performance will probably be obtained at the 4x4 decomposition since it uses 2 angles per process and more than 1,500 vertices per process. However, the other decompositions are also valid. Additionally, fewer than 16 processors could be used to solve the problem.

We note that the number of locally owned vertices (vertices per process) will likely vary from processor to processor in a real problem, whereas the number of locally owned angles will be constant across processors.

4.1.5 Processing Cross Sections with the Cross Section API

As noted earlier, PROTEUS-SN can be compiled with the cross section API (developed by Argonne National Laboratory) to process multi-group cross section data on-the-fly for the exact geometry using the subgroup or resonance table method. This capability is still being verified fully vetted and we recommend using with caution.

To process cross sections with the Cross Section API rather than providing an input cross section file, the line “USE_CSAPI YES” should appear in the driver input file. This option is only valid with the `sn2nd_sg.x` (PROTEUS-SN with cross section dependency) executable. The isotope names in the material assignment file must match those in the cross section library. If you are interested in using this option, please contact nera-software@anl.gov for more up-to-date information on reliability.

4.2 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-SN. The MC²-3 code (Argonne National Laboratory) can be used to process multigroup cross sections in this format. Additionally, the DRAGON code (Ecole Polytechnique de Montreal) [11] has a capability to generate ISOTXS file.

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. A detailed description and an example of the anlxs file format are given in Appendix C.

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.

As an alternative to providing pre-processed multigroup cross sections data, PROTEUS-SN can process cross section data on-the-fly using the exact heterogeneous geometry via the Cross Section Application Programming Interface (API) (see 4.1.5).

4.3 Mesh File (*.nemesh, *.ascii, *.pntmesh, *.bgpmesh, *.ufmesh)

4.3.1 Mesh File Types

PROTEUS-SN is an unstructured finite-element based method and requires a finite element mesh as input. One of the following finite element mesh formats is expected:

- a. *.nemesh
- b. *.ascii
- c. *.pntmesh (parallel execution on Linux)
- d. *.bgpmesh (parallel execution for special lower memory machines)
- e. *.ufmesh

The *nemesh* file format is a simple finite element mesh input file format which is detailed in Appendix D. The *ascii* file format is less user friendly, and relies upon structures intrinsic to the PROTEUS-SN code but is detailed in Appendix E.

The *pntmesh* file format is the base file that PROTEUS-SN will run with when executing (it converts the *nemesh* or *ascii* files into *pntmesh* formats at runtime). The *bgpmesh* file format is a pre-decomposed *ascii* file format for use with the memory-limited BlueGene/Q supercomputer at Argonne.

The *ufmesh* file format is a user-friendly mesh format which uses keywords to describe the geometry and mesh for a regular 2D Cartesian or hexagonal assembly or pin cell. Different *ufmesh* files can be merged together to create 3D cores quickly and conveniently using PROTEUS's mesh utility tools. A companion manual [13] is available describing the use of *ufmesh* and associated mesh utility tools. Note that *ufmesh* permits the naming of regions by character string rather than by number for a more intuitive input file and analysis procedure.

4.3.2 Mesh Generation

PROTEUS-SN uses the mesh file formats described above. There is no finite element meshing software that directly produces these types of files, so it is recommended to use your favorite meshing software and build your own converter to the *nemesh* or *ascii* formats. Note that the GAMBIT neutral file format is an ASCII format comparable (but not identical) to the *nemesh* file format.

We routinely use the CUBIT software [14] to build an EXODUS II (.e or .exo) mesh. The PROTEUS package includes a separate EXODUS II converter to transfer EXODUS II meshes into the *ascii* file format required by PROTEUS. Note that this converter is built on NetCDF [15] which is downloaded separately and is not part of the distribution. The EXODUS II converter is compatible with CUBIT versions 12 and 13, and may also work with CUBIT 14 (not tested).

Alternatively, the open source tool MeshKit [16] can be used to create *h5m* type meshes, which can be converted to Exodus format via MOAB's conversion tool [17].

As a final note, we have some internal tools to help with mesh creation, for example, merging a collection of assemblies into a core, or extruding a 2D mesh into 3D. These are currently not being released with PROTEUS-SN, but if interested please contact nera-software@anl.gov.

4.3.3 Element Types

PROTEUS-SN permits bar (1D), triangular (2D), quadrilateral (2D), tetrahedral (3D), prism (3D), and hexahedral (3D) element shapes. Additionally, PROTEUS accepts elements with linear, quadratic, and higher orders. For fine mesh geometries (fuel pin-cells), it is strongly advised to utilize linear finite elements in PROTEUS-SN because of the severe condition number problems associated with the even-parity method. For coarse homogeneous problems, quadratic order finite elements are typically best. Note that PROTEUS-SN can handle a mixture of element types in any given mesh (i.e. combination of triangular and quadrilateral) provided each block contains elements of a single type.

More information on the PROTEUS-SN finite element zoo is available in Appendix G. Note that the reference surface numbering in the *nemesh* file format refers to the element-wise reference surface numbering in Appendix G.

4.3.4 Element Blocks and Sidesets

An element block is a user-defined set of elements (not necessarily contiguous) which have the same element type and same material composition. A sideset is a user-defined set of surfaces (not necessarily contiguous) which have the same boundary conditions (see Figure 3).

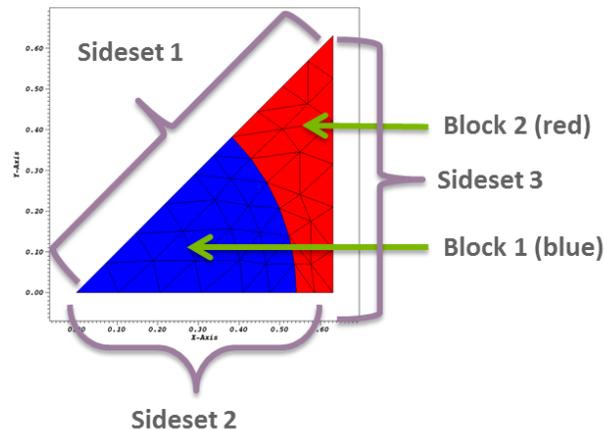


Figure 3. Example Mesh Showing Blocks and Sidesets for 1/8-th Pin Cell.

In most mesh generators, one assigns names and block numbers to physical volumes in the domain. For example, a block can be created out of a set of elements which all have “fuel” composition and are linear triangles. The elements in those volumes are typically written together in the mesh file. However, the names are rarely passed through to the interface file,

which is the case for CUBIT-created Exodus files, despite the fact that names would be more convenient for most neutronics work. Instead, only the block numbers are passed through.

In PROTEUS-SN, the preferred approach is to utilize names and thus the block numbers are translated into names. Focusing on the *nemesh* file format of Appendix B as an example, each element is assigned a “region” number. This region number, say “1”, is translated to the name “REGION_000000001” in the mesh conversion process. This block name is then used in the assignment file to assign cross section data (defined via compositions) to regions of the domain. Similarly, sidesets of similar surfaces can be defined in the Exodus mesh and assigned boundary conditions as described in Section 4.1.3. Extraneous (unused) sidesets should not be defined in the mesh as they can cause issues.

4.3.5 Issues to Consider While Meshing

PROTEUS-SN requires a conformal mesh, i.e. a mesh where a node of one element is also a node of the neighboring element (unless it lies on the boundary). Figure 4 shows an example of each type.

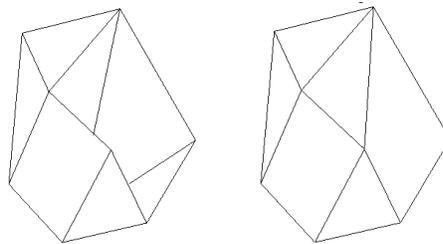


Figure 4. Non-conformal Mesh (left) and Conformal Mesh (right).

In addition to being conformal, element quality is an important consideration in the mesh. Long, thin elements can cause trouble. The ideal elements in 2D are an equilateral triangle or square.

As noted above, for fine mesh geometries (fuel pin-cells), it is strongly advised to utilize linear finite elements in PROTEUS-SN because of the severe condition number problems associated with the even-parity method. For coarse homogeneous problems, quadratic order finite elements are typically best.

Finally, note that linear elements do not exactly preserve curvilinear geometry. Therefore, in a mesh refinement study with linear elements and curvilinear geometry, one should be careful to preserve the important masses (fuel, absorber) manually by tweaking the densities. The CUBIT program has convenient functions to query the volume of the elements and blocks.

Note that the computational effort required in PROTEUS-SN increases roughly linearly with increasing number of vertices (maintaining the dimensionality). Going from 2D to 3D results in a significant increase in computational effort due to the increased number of elements and increased connectivity.

4.4 Material Assignment File (*.assignment)

The material assignment (*.assignment) file performs three main functions: (1) define materials or mixtures based on the isotopes in the cross section file, (2) assign these materials to blocks in the mesh, and (3) assign properties (e.g. density) to blocks in the mesh.

The material assignment file uses simple keyword-based input in free format. It must be created by hand by the user, although scripting procedures can often be developed to speed the process. Comments start with either “!” or “#”.

4.4.1 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 5 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-SN. 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 5. Material Definition in the Assignment File (Atom Fractions).

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

Figure 7 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.

```
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 7. Recursive Material Definition in the Assignment File.

It is not permitted to mix weight fractions and atom fractions in the definition of a material. Also, if a material is defined recursively, all of its precursors must use the same unit for fraction. Taking the example above, you cannot use weight fraction for Water and atom fraction for Salt, whatever your choice of fraction unit for Saline is.

4.4.2 Assigning Materials to Blocks

In the previous section, we defined a material called FUEL. We can assign this material (or any other composition appearing in the material assignment file or the cross section file) to a block in the mesh. Say block 111 (block numbering was defined by the user during mesh creation) is supposed to be a fuel pin. We can associate the material FUEL to block 111 using the REGION_ALIAS keyword shown in Figure 8.

```
REGION_ALIAS REGION_000000111 FUEL
```

Figure 8. Assigning a Material to a Block in the Assignment File.

Note the numbering convention used in PROTEUS-SN to refer to blocks. Block 111 is addressed as REGION_000000111 in the PROTEUS-SN assignment file. Block numbers in the mesh must therefore be no more than 9 digits in order to avoid possible duplicate character strings (limited to 9 digits).

4.4.3 Assigning Block Properties

We have not yet accounted for the actual material density of the block, or any other properties. The previous step simply linked cross section data to a specific region. If macroscopic cross section data was provided, no further action is necessary since the number density is incorporated into the data. More typically, the cross section data file contains microscopic cross section data and the density must be assigned. Figure 9 demonstrates the assignment of block

111's total atom density to 5.6129E-02 atoms/barn-cm using the keywords REGION_PROPERTY and ATOM_DENSITY.

```
REGION_PROPERTY REGION_000000111 ATOM_DENSITY 5.65129E-02
```

Figure 9. Assigning Total Atom Density to a Block in the Assignment File.

Alternatively, the total mass density in g/cm³ can be assigned to a block as shown in Figure 10. When using the Density (g/cc) property, PROTEUS uses the mass specified in the ISOTXS file for each basic component to convert to number density.

```
REGION_PROPERTY REGION_000000111 Density(g/cc) 8.595
```

Figure 10. Assigning Total Mass Density to a Block in the Assignment File.

The keyword REGION_PROPERTY is therefore used to assign various properties to the mesh such as ATOM_DENSITY, Density(g/cc), TEMPERATURE(K), and MATERIAL_MODEL 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.

An example of a complete assignment file for a fictitious problem is shown in Figure 11. This assignment file assigns materials to two blocks (block 1 and block 7).

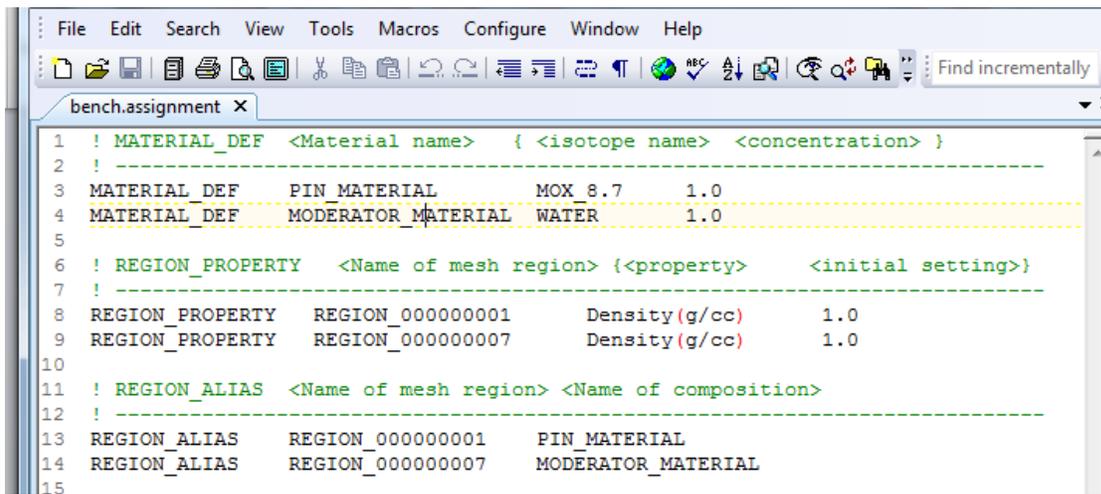


Figure 11. Example PROTEUS-SN Assignment File.

4.5 Mesh Partitioning File (*.partitioning)

As mentioned, PROTEUS performs parallel decomposition in angle and/or space. Spatial domain decomposition for finite element methods consists of breaking the mesh into contiguous smaller pieces of mesh, each piece having approximately the same number of degrees of freedom. Each small piece of the mesh is assigned to a given processor as its work (part of the global vector space). The mesh partitioning procedure is performed in PROTEUS-SN via METIS. Mesh partitioning can be done (1) dynamically at execution (note that the version of METIS we are connected to limits this work to serial) or (2) in advance by providing an additional input partitioning file. Note that PROTEUS-SN always outputs a partitioning file with the extension “.partitioning” in case (1).

For machines with considerable memory resources, users typically find it easier to provide a standard mesh file and just have PROTEUS-SN compute the partitioning at runtime. However, in cases where the user will be re-using the partitioned mesh, or where the entire mesh cannot actually fit into memory (i.e. on low memory machines such as BlueGene/P hardware), the mesh partitioning has to be done in advance. The pre-partitioned mesh file (*.partitioning) is the same as what PROTEUS-SN would partition during normal execution. Note that a valid SOURCEFILE_MESH must be provided that is consistent with the partitioning file. The path to the partitioning file is specified in the driver input file using the keyword SOURCEFILE_MESHPART as seen earlier in Table 4.

If no partitioning file is given or if the partitioning file is not valid (it does not match the spatial decomposition), PROTEUS will create a new partitioning file. To create a partitioning file of your own choosing, you can run a calculation without a partitioning file and then keep the created partitioning file for other calculations with the same mesh and spatial decomposition (only the angular or the group structures vary). It is also possible to obtain a mesh partitioning file without solving the whole problem by using the keyword MESHSIMULATE. If MESHSIMULATE is set to a non-zero positive value, a serial PROTEUS job will load the mesh and generate the partitioning file for a spatial decomposition corresponding to the number given in MESHSIMULATE. The file will be exported with the name “LegacyConversion.pntmesh.partitioning.”

The BGPmesh partitioned mesh file format was specifically constructed to minimize the memory overhead to accommodate the BlueGene/P machine. It is not supported in this release, and we suggest contacting nera-software@anl.gov for any questions or concerns with partitioning on low-memory machines.

4.6 Fixed Source File (*.fixeds)

To describe a problem with a fixed source contribution, the input option SOURCEFILE_FIXEDSOURCE can be used to specify the path to the fixed source definition file. Three types of fixed sources are supported, all of which are isotropic in angle: vertex-based, mesh region-based (flat in space), and coordinate point based (assuming there is a vertex at the coordinate). In all cases, one should define fixed source information in the fixed source file for

each and every energy group using Table 5 as a reference. Note that any combination of options in the table can be used simultaneously in the file in an additive sense

Table 5. Fixed Source File Format

Keyword	Input Data	Description
PICK_VERTEX	<VertexID> <Group1> <Group2> ... <GroupG>	Selects the vertex and the magnitude of the isotropic source to apply for each energy group.
PICK_REGION	<NAME> <Group1> <Group2> ... <GroupG>	Selects a region name, i.e. "REGION_0000001" specified in the assignment file and the magnitude of the source to apply for each energy group. Source is isotropic in angle and flat in space.
PICK_COORDINATE	<X> <Y> <Z> <Group1> <Group2> ... <GroupG>	Selects a coordinate in space (a mesh vertex must lie at this position) and the magnitude of the fixed source to apply for each energy group.

4.7 Kinetics Input File (*kinetics.inp*)

The PROTEUS-SN distribution includes a basic time-dependent capability using the adiabatic approximation (simplification to the quasi-static approximation). In addition to the four standard input files, kinetics calculations require an additional kinetics input file placed in the working directory. Additionally, the kinetics capability allows different assignment files to be used at each timestep.

PROTEUS-SN automatically searches for a kinetics input file called "kinetics.inp". To use a different kinetics input file name, PROTEUS-SN must be executed using the command line option "-kinetics <filename>".

Table 6 lists all of the input options used in the kinetics input file.

Table 6. Kinetics Input Options.

Keyword	Input Data	Default value	Description
---------	------------	---------------	-------------

Keyword	Input Data	Default value	Description
<i>Kinetics Solver Options</i>			
UPDATE_LAMBDA BETA	0,1,2	0	0 = Update Λ and β at each time step using time-dependent adjoint flux 1 = Update Λ and β at each time step using the initial condition adjoint flux 2 = Do not update Λ and β at each time step, instead use initial condition values (Note: Older versions of the code use YES or NO for this variable to correspond to option 0 and 2, respectively)
USE_RADAU	YES or NO	YES	YES = Use the Radau solver (default, more accurate) NO = Use the traditional finite difference solver
USE_FIXEDSOURCE_ADJOINT	YES or NO	NO	YES = Use the fixed source adjoint NO = Use the homogeneous adjoint
<i>Output File Options</i>			
EXPORT_PRECURSORS	<Path to file>	-	Output file for precursor and power data at each coarse time step.
INITIAL_EXPORT_FILE	<Path to file>	initial.pmo	Output file for the full flux solution at the initial timestep.
INITIAL_REGIONEDITS_FILE	<Path to file>	initial.out	Output file for the block-wise region edits at the initial timestep.
<i>Input File Options</i>			

Keyword	Input Data	Default value	Description
INITIAL_MATERIAL_FILE	<Path to file>	initial.assignment	Input assignment file for the initial timestep. Future timesteps will use the assignment files listed in TIME_STEP input.
INITIAL_FIXEDSOURCE_FILE	<Path to file> or NULL	initial.fixeds	Input fixed source file for the initial time step. Use NULL if no fixed source input. Future timesteps will use the fixed source files listed in TIME_STEP input.
TIME_STEP	<time> <intervals> <*.assignment> <*.fixeds> <*.h5> <*.out>	-	<p>Specifies time step information and inputs. Repeat this card input for each timestep.</p> <p><i>Required:</i> <time> = real time at end of the timestep <interval> = number of intervals to use (report solution at) for this timestep <*.assignment> = assignment file to use for this timestep</p> <p><i>Optional (can appear in any order):</i> <*.fixeds> = fixed source file to use or NULL <*.h5 or *.pmo> = flux solution output <*.out> = region edits output</p>
DELAY_XS_FILE	<Path to file>	default.dlayxs	Input delay cross section data file (in DLAYXS format).

Keyword	Input Data	Default value	Description
DELAY_ALIAS	<ISOTXS_XS_NAME> <DELAY_XS_NAME>	-	Maps the isotope name in ISOTXS to the delay xs isotope name in DLAYXS, in case they differ. Repeat this card input for each delay alias.

The adiabatic approximation to the quasi-static kinetics method essentially computes solutions to the point-kinetics equations at every time step. However, the kinetics parameters (Λ and β) are a function of the time-dependent forward and adjoint flux. In this approximation, the basic “steady state” forward and adjoint flux are calculated at every time step for updated material properties and used to update Λ and β if UPDATE_LAMBDABETA 0 is requested. If UPDATE_LAMBDABETA 1 is requested, the parameters are updated using the updated forward flux but only the initial condition adjoint flux. If UPDATE_LAMBDABETA 2 is requested, the parameters are calculated once at the initial condition and never updated.

The keyword USE_RADAU indicates whether the Radau solver should be used to solve the point kinetics equations. The default value is yes. The Radau solver is based on the Runge-Kutta method and is more accurate than the implicit differencing scheme that would otherwise be used. To disable the Radau solver and instead use the finite differencing scheme, USE_RADAU NO should be specified.

For fixed source problems, the keyword USE_FIXEDSOURCE_ADJOINT specifies whether the fixed source adjoint flux should be used in updating Λ and β (YES) or whether the homogeneous adjoint flux solution should be used (NO).

The delay cross section data (decay constants, emission spectra and delayed precursor yields) must be provided in the DELAY_XS_FILE input in DLAYXS file format (available from MC²-3). Material assignment files for each timestep must be provided in the INITIAL_MATERIAL_FILE and TIMESTEP inputs. The DELAY_ALIAS input is used to map the isotope name in the cross section file to delay cross section data in the DLAYXS file, in case the files contain different user-specified names for the same isotopes.

The TIMESTEP card specifies information about each timestep. This card is used to specify the real time at the end of the timestep, how many intervals to perform, and the material assignment file. Optionally, a fixed source input file (*.fixeds) and flux (*.h5 or *.pmo) and region edit (*.out) output files for the timestep may also be specified (in any order) using filenames with the appropriate suffixes. An interval is like a sub-timestep within a timestep. At the end of an interval, only the point kinetics equations are solved for power, etc using the

previous interval values for Λ , β and the eigenvalue. At the end of a timestep, the transport equation (forward and adjoint) are solved to obtain a new Λ , β and eigenvalue. These are then used as updated parameters in the point kinetics equations. Flux files are provided only at the end of a timestep.

The precursor densities and power data at each coarse time step can be output to a file specified by the EXPORT_PRECURSORS keyword. The block-wise region edits (group scalar flux, power, absorption, volume) can be exported to the file specified by INITIAL_REGIONEDITS_FILE (for the initial condition) and to separate files for each coarse timestep specified in the TIMESTEP input. Similarly, the full flux solution can be exported to INITIAL_EXPORT_FILE (for the initial condition) and to separate files for each coarse timestep specified in the TIMESTEP input.

Figure 12 shows an example kinetics.inp file with 3 timesteps beyond the initial configuration. The kinetics parameters will be updated at every time step. The alias for isotope P239H (ISOTXS) is PU239 (DLAYXS), meaning that the PU239 delay cross section data will be mapped to isotope P239H.

```
! Kinetics input file - no fixed source
DELAY_XS_FILE  bench_3g.DLAYXS.ascii ! Use this delay xs file

UPDATE_LAMBDA BETA 0 ! Update L,B,Adj every time step
USE_RADAU YES ! Use Radau rather than F.D. scheme

INITIAL_EXPORT_FILE solution.h5 ! Store IC forward soln
INITIAL_REGIONEDITS_FILE edits00.out ! Store IC region edit
INITIAL_MATERIAL_FILE initial.assignment ! IC assignment file

! TIME_STEP <end time> <intervals> <assignment> <fluxstorage> <edits>
TIME_STEP 0.001 1 step01.assignment step01.h5 edits01.out
TIME_STEP 0.002 2 step02.assignment step02.h5 edits02.out
TIME_STEP 0.005 2 step03.assignment step03.h5 edits03.out

DELAY_ALIAS P239H PU239 ! P239H in ISOTXS file maps to PU239 in DLAYXS
```

Figure 12. Example Kinetics Input File with No Fixed Source.

Figure 13 is an example of a kinetics input file with fixed source. Note that for fixed source problems, a fixed source file should be specified for all timesteps including initial condition. The value NULL can be supplied in place for timesteps without a fixed source.

```
! Kinetics.inp
DELAY_XS_FILE  bench_3g.DLAYXS.ascii ! Use this delay xs file

UPDATE_LAMBDA_BETA 1          ! Update L,B every time step
USE_RADAU YES              ! Use Radau rather than F.D. scheme

INITIAL_EXPORT_FILE      solution.h5 ! Store IC forward soln
INITIAL_REGIONEDITS_FILE edits00.out ! Store IC region edit
INITIAL_MATERIAL_FILE    initial.assignment ! IC assignment file
INITIAL_FIXEDSOURCE_FILE initial.fixedsource ! IC fixed source file

! TIME_STEP <end time> <intervals> <assignment> <fluxstorage> <edits>
<fixed source file>
TIME_STEP 0.001 1 step01.assignment step01.h5 edits01.out data1.fixedsource
TIME_STEP 0.002 2 step02.assignment step02.h5 edits01.out data2.fixedsource
TIME_STEP 0.005 2 step03.assignment step03.h5 edits01.out data3.fixedsource

DELAY_ALIAS  P239H  PU239 ! P239H in ISOTXS file maps to PU239 in DLAYXS
```

Figure 13. Example Kinetics Input File with Fixed Source.

5. Output

PROTEUS-SN has four basic types of output: (1) main output text file, (2) full solution, (3) VTK files for mesh and cubature, and (4) volume-integrated region edits. The main output text file is always produced upon execution. The other three types of output are optionally triggered by keywords `EXPORT_FLUX`, `EXPORT_MESH`, and `EXPORT_CUBATURE`, and `EXPORT_EDITS` in the driver input file.

5.1 Main Output File

The main text-based output is printed to standard output or to a file if the `-output` option is used. The output contains confirmation that the input was imported successfully, parallel timing summaries, and eigenvalue iteration history results.

Appendix F provides an example output file. At the top of the file, the various input options are echoed and errors are produced if the input files were not successfully imported. This section is followed by the outer iteration history which reports the eigenvalue (and other quantities) as the calculation proceeds. At the end of the outer iteration history, the final eigenvalue is reported along with an error estimate. At the end of the output file, an MPI timing history provides an overview of where time was spent in the calculation.

5.2 Full Solution

The full solution in the entire domain can optionally be exported to an ASCII file (.pmo) or to HDF5 file format (.h5). We recommend using the .h5 format which requires that PROTEUS-SN be compiled with HDF5. The creation of the HDF5 or PMO files is triggered by the `EXPORT_FLUX` and `EXPORT_FILE` options in the input driver file, for example:

```
EXPORT_FLUX           YES
EXPORT_FILE           abtr_L5T7.h5
```

These two lines will ensure that a file named "abtr_L5T7.h5" is created, in HDF5 file format, containing the full flux solution. The .h5 output file includes the full 3D mesh, composition, scalar flux by group, nu*fission rate, absorption rate, and power. Temperature and density are not recorded. Also, the eigenvalue is not recorded in this file.

The .h5 file can be opened and viewed in VisIt as a "UNIC" type file in the drop-down list. Unfortunately, PROTEUS does not output integrated quantities such as pin power or assembly-total power at this time. Various techniques are available in VisIt to post-process the output, such as summing the solution in a certain region, or exporting values along a line using the "Line-out" analysis option. To use the Line-out analysis tool in VisIt, the user must click on visit's "Analysis" menu, choose Line-out, specify two points on the line, and export the database as a Curve file.

Figure 14 through Figure 16 show PROTEUS-SN results visualized with VISIT using the PROTEUS plugin. Quantities such as the mesh, power distribution, and flux distribution for each energy group can be visualized.

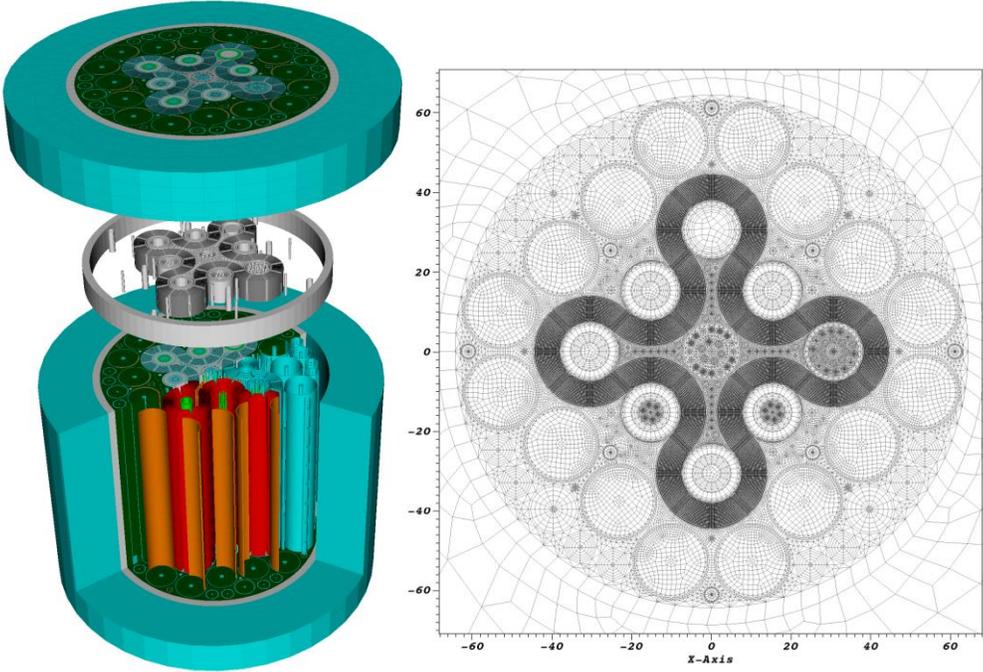


Figure 14. Visualization of Geometry and Mesh Used to Analyze the Advanced Test Reactor.

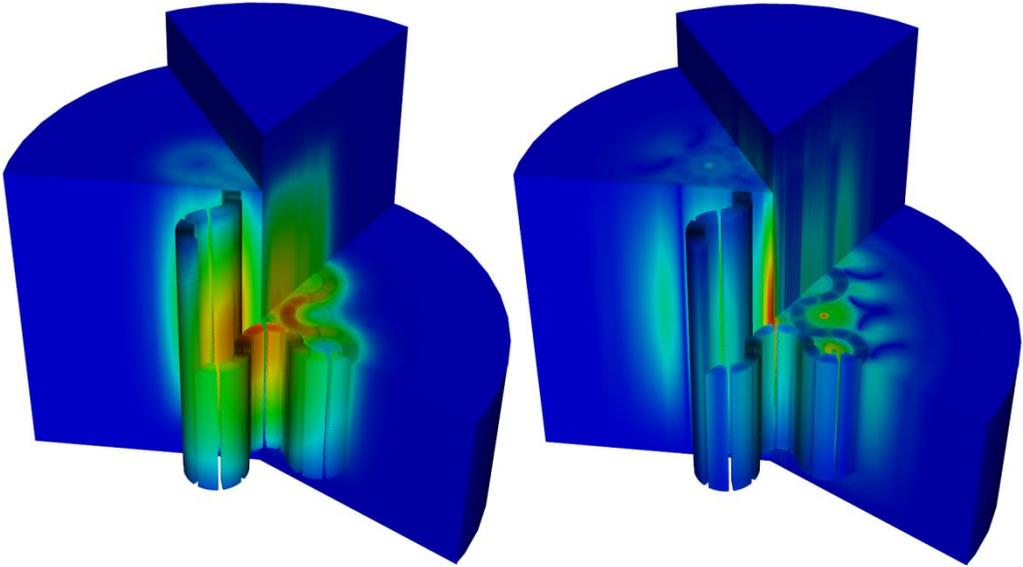


Figure 15. ATR fast (left) and Thermal (right) Flux Calculated by PROTEUS-SN.

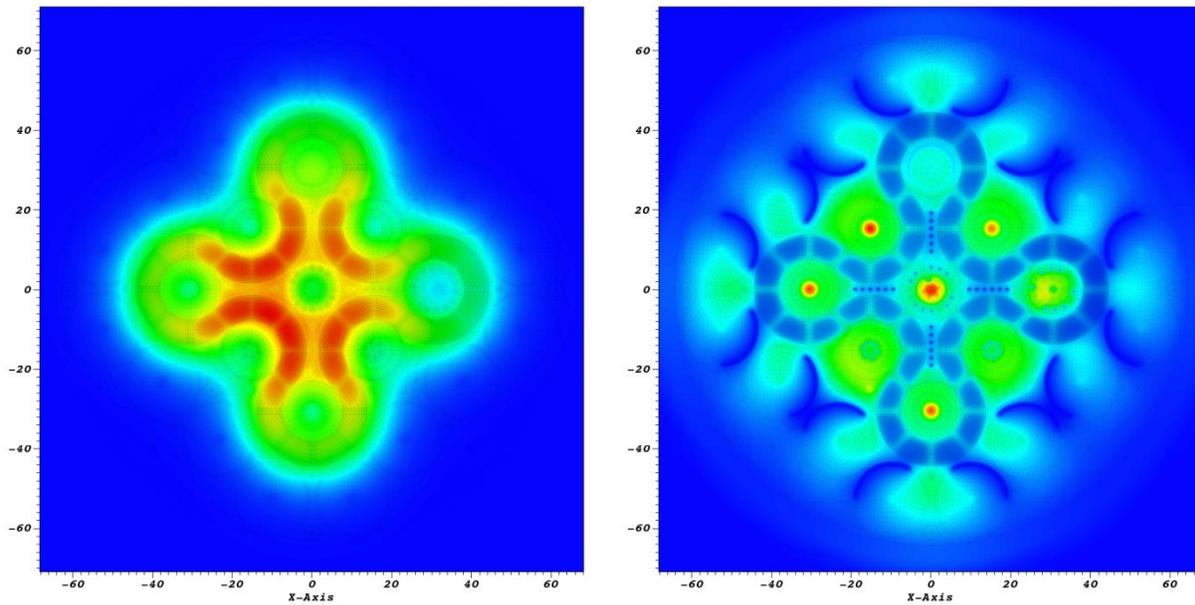


Figure 16. Visualization of 2D Fast (Left) and Thermal (Right) Fluxes for ATR Calculation.

If the output was requested in .pmo format, (.pmo recommended for supercomputers/ extremely large problems), several .pmo files will be produced in the output directory. These must first be converted to .h5 format using the PMO_AssemblyHDF5.x converter provided with PROTEUS-SN (see Section 3.1.4 Converting PMO to HDF5 Output).

5.3 VTK Files for Mesh and Cubature

The mesh can also be visualized using the EXPORT_MESH option to produce a VTK file viewable in VisIt or another VTK-friendly visualization program. This option can be invoked to visually inspect the mesh, for example. Note that the mesh is also included in the full solution output.

The angular cubature can be visualized using the EXPORT_CUBATURE option to produce a VTK file. It can be helpful to visualize the placement of angles, and associated weights for each ordinate, on the unit sphere.

5.4 Region Edit Integrals

The region edit integrals can also be exported to a simple text file for quick analysis using the EXPORT_EDITS option and specifying the name of the desired output file. The resulting file will contain a list of quantities by block number (region number) specified in the mesh: volume, power, fission, absorption, and flux by group. Keep in mind that the total power will be normalized to the user-specified value of THERMAL_POWER (excludes fixed source case).

6. Example Problem

Here we discuss a simple example taken from the PROTEUS-SN benchmark 5 test case. This test case is a steady-state forward and adjoint calculation for a 2D 1/8th pin cell with reflected boundaries. The materials in the problem are MOX fuel and water. As with any deterministic calculation, a space-angle-convergence study should be performed to assess convergence (not shown here). The following input files are given for this problem:

- bench05.inp (driver input)
- bench.nemesh (mesh)
- bench.anlxs (cross sections)
- bench.assignment (material assignment file)

Examining the driver input file in Figure 17, we see that the paths to the mesh, cross section data, and material assignment file are clearly defined by the SOURCEFILE_MESH, SOURCEFILE_XS, and SOURCEFILE_MATERIAL keywords. The Solve_Type keyword indicates that both the forward and adjoint flux are to be solved (this is the default value of Solve_Type, so this value is not required). A Legendre-Tchebychev cubature of order L3T9 will be used, for a total of $\frac{1}{2}(L+1)(T+1)$ or 20 angles in 2D PROTEUS-SN.

The code will attempt to perform maximum parallelization in angle (SEGMENT ANGLE assigned to 0). If we only perform the problem in serial, no parallelization will occur. The power will renormalized to 1 Watt (default). The SSOR preconditioner (default) will be used and Tchebychev acceleration of the fission source will be applied (default). The relative error tolerances have been somewhat tweaked. Scattering will be isotropic. The mesh and full solution will be exported to VTK and HDF5 files, respectively.

```

bench05.inp
SN_TYPE          LEG-TCHEBY          ! Cubature type
Theta_Resolution 3                    ! L order
Phi_Resolution   9                    ! T order
SEGMENT_ANGLE    0                    ! Parallelization in angle

DEBUG_PRINT_LEVEL      0
DEBUG_PRINT_SETUP      0
DEBUG_PRINT_FORMATION  0
DEBUG_PRINT_OUTER     0
THERMAL_POWER         1.0

USE_PRECONDITIONER    SSOR            ! SSOR, SOR, ILU, ICC
USE_TCHEBYCHEV_ACCEL  YES

Scattering_Order      0                ! 0,1,2,3,4,5,...
EIGENVALUE_GUESS     1.17             ! The guess for the initial eigenvalue
Tolerance_eigenvalue  1.0e-7          ! The maximum relative error to allow on the eigenvalue
Tolerance_Fission    1.0e-8          ! The maximum relative error to allow on the outer iteration
Tolerance_Flux       1.0e-7          ! The maximum relative error to allow on the inner iteration
ITERATIVE_IMPROVEMENT 0.1

Iterations_Fission    100             ! The maximum number of outer iterations
Iterations_UpScatter  5               ! The maximum number of upscatter iterations
Iterations_scattering 5
Iterations_SA_CG      100
Iterations_PETSc     1000

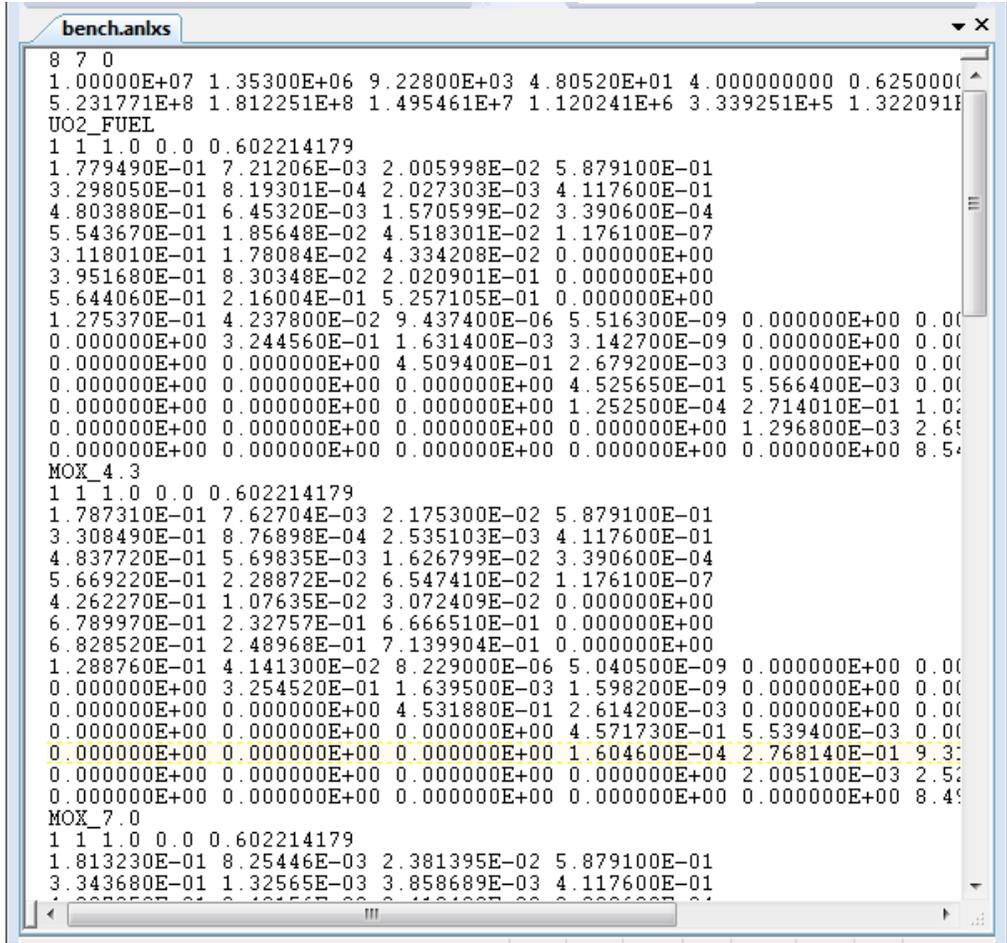
Solve_Type both

SOURCEFILE_MESH      bench.nemesh     ! Path to mesh file
SOURCEFILE_XS        bench.anlxs      ! Path to cross section file
SOURCEFILE_MATERIAL  bench.assignment ! Path to assignment file
BASIC_BWO            no

EXPORT_MESH YES
EXPORT_FLUX YES
EXPORT_FILE bench05.h5
    
```

Figure 17. Driver Input File for Example Case.

We continue on to the cross section file, specified as bench.anlxs in the driver input. Reviewing the .anlxs file format in Appendix C, we determine that the first line contains the number of compositions, number of energy groups, and another unimportant integer. This file contains 8 compositions with a 7 energy group structure which is listed on the following lines. The composition names are read as UO2_FUEL, MOX_4.3, MOX_7.0, and others (entire file not shown for brevity). Therefore, we should expect to see these composition names in the material assignment file. The multigroup cross section data is listed for each isotope as shown in Figure 18.



```
8 7 0
1.00000E+07 1.35300E+06 9.22800E+03 4.80520E+01 4.000000000 0.6250000
5.231771E+8 1.812251E+8 1.495461E+7 1.120241E+6 3.339251E+5 1.322091E
UO2_FUEL
1 1 1.0 0.0 0.602214179
1.779490E-01 7.21206E-03 2.005998E-02 5.879100E-01
3.298050E-01 8.19301E-04 2.027303E-03 4.117600E-01
4.803880E-01 6.45320E-03 1.570599E-02 3.390600E-04
5.543670E-01 1.85648E-02 4.518301E-02 1.176100E-07
3.118010E-01 1.78084E-02 4.334208E-02 0.000000E+00
3.951680E-01 8.30348E-02 2.020901E-01 0.000000E+00
5.644060E-01 2.16004E-01 5.257105E-01 0.000000E+00
1.275370E-01 4.237800E-02 9.437400E-06 5.516300E-09 0.000000E+00 0.00
0.000000E+00 3.244560E-01 1.631400E-03 3.142700E-09 0.000000E+00 0.00
0.000000E+00 0.000000E+00 4.509400E-01 2.679200E-03 0.000000E+00 0.00
0.000000E+00 0.000000E+00 0.000000E+00 4.525650E-01 5.566400E-03 0.00
0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 1.252500E-04 2.714010E-01 1.02
0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 1.296800E-03 2.65
0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 8.54
MOX_4.3
1 1 1.0 0.0 0.602214179
1.787310E-01 7.62704E-03 2.175300E-02 5.879100E-01
3.308490E-01 8.76898E-04 2.535103E-03 4.117600E-01
4.837720E-01 5.69835E-03 1.626799E-02 3.390600E-04
5.669220E-01 2.28872E-02 6.547410E-02 1.176100E-07
4.262270E-01 1.07635E-02 3.072409E-02 0.000000E+00
6.789970E-01 2.32757E-01 6.666510E-01 0.000000E+00
6.828520E-01 2.48968E-01 7.139904E-01 0.000000E+00
1.288760E-01 4.141300E-02 8.229000E-06 5.040500E-09 0.000000E+00 0.00
0.000000E+00 3.254520E-01 1.639500E-03 1.598200E-09 0.000000E+00 0.00
0.000000E+00 0.000000E+00 4.531880E-01 2.614200E-03 0.000000E+00 0.00
0.000000E+00 0.000000E+00 0.000000E+00 4.571730E-01 5.539400E-03 0.00
0.000000E+00 0.000000E+00 0.000000E+00 1.604600E-04 2.768140E-01 9.33
0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 2.005100E-03 2.52
0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00 8.49
MOX_7.0
1 1 1.0 0.0 0.602214179
1.813230E-01 8.25446E-03 2.381395E-02 5.879100E-01
3.343680E-01 1.32565E-03 3.858689E-03 4.117600E-01
```

Figure 18. Cross Section File for Example Case.

We also take a quick look at the mesh, provided in bench.nemesh (.nemesh file format described in Appendix D). Card Type 2 lists the number of elements (16), number of vertices (45), and number of boundary surfaces (12).

The first line in card type 3 can be interpreted as “Element 1 has 6 nodes and belongs to block 1”. Subsequent lines describe the other elements. Element connectivity follows, as well as a listing of vertex positions and boundary surfaces (not shown for brevity). This is in fact a quadratic-order triangular mesh.

```

bench.nemesh x
! ANL FINITE ELEMENT INPUT FILE DESCRIPTION 9 HEADER LINES ALWAYS
! CARD TYPE 1:      (Input Style: 0-indexed 1-not indexed) (Debug Printing: 1-10)
! CARD TYPE 2:      (# Elements) (# Nodes) (# Edit Regions) (# boundary element surfac
! CARD TYPE 3:      [Optional Index] (ElementType) (Material)
! CARD TYPE 4:      [Optional Index] (Element Connectivity)
! CARD TYPE 5:      [Optional Index] (X) [Y] [Z]
! CARD TYPE 6:      [Optional Index] (Element #) (Ref. Surf.) (bound. cond.)
! CARD TYPE 7:      [Optional Index] (Reaction rate) (# elements)
! CARD TYPE 8:      [Optional Index] (Edit Region Elements)
0 00
16 45 0 12
1 6 1
2 6 1
3 6 1
4 6 1
5 6 1
6 6 1
7 6 1
8 6 1
9 6 1
10 6 7
11 6 7
12 6 7
13 6 7
14 6 7
15 6 7
16 6 7
1 1 2 3 11 17 10
2 3 4 5 13 19 12
3 17 11 3 12 19 18
4 17 18 19 24 28 23
5 5 6 7 15 21 14
6 5 14 21 20 19 13
7 19 20 21 26 30 25
8 28 24 19 25 30 29
9 28 29 30 34 37 33
10 7 8 9 16 21 15
11 9 22 32 27 21 16
12 21 27 32 31 30 26
13 32 36 39 35 30 31
14 30 35 39 38 37 34
15 37 38 39 41 42 40
16 39 43 45 44 42 41
1 0 0
2 0.10237500000000001 0
3 0.20475 0
4 0.307125 0
5 0.4115000000000000 0
    
```

Figure 19. Mesh File for Example Case.

Figure 20 shows the VTK plot of the mesh along with compositions. (The mesh was exported to a VTK file by PROTEUS-SN using EXPORT_MESH.) The block coloring indicates the elements belonging to block 1 (blue) and block 7 (red).

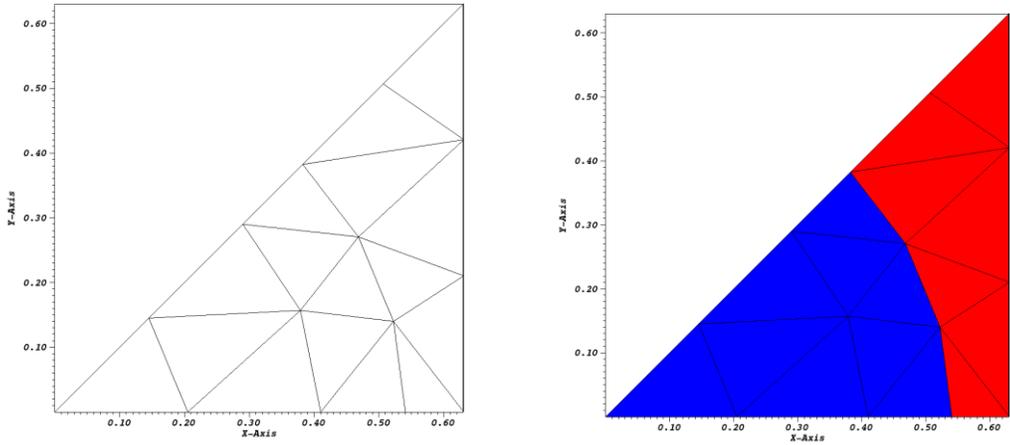


Figure 20. VTK Mesh (left) and VTK Mesh with Block Coloring (right) for Example Case.

A snapshot of the assignment file is shown in Figure 21. The first non-commented line in the file defines a material called PIN_MATERIAL made of 100% MOX_8.7. The second line similarly defines a material called MODERATOR_MATERIAL made of 100% WATER. Note that these are actually just aliases to MOX_8.7 and WATER, respectively.

Skipping to the last section, PIN_MATERIAL is assigned to block 1 (blue), and MODERATOR_MATERIAL is assigned to block 7 (red). In the middle section, both blocks are assigned a mass density of 1.0 g/cc. This mass density will be converted to atom density in the code as shown in Figure 21.

```
bench.assignment
! MATERIAL_DEF <Material name> { <isotope name> <concentration> }
! -----
MATERIAL_DEF PIN_MATERIAL MOX_8.7 1.0
MATERIAL_DEF MODERATOR_MATERIAL WATER 1.0

! REGION_PROPERTY <Name of mesh region> {<property> <initial setting>}
! -----
REGION_PROPERTY REGION_000000001 Density(g/cc) 1.0
REGION_PROPERTY REGION_000000007 Density(g/cc) 1.0

! REGION_ALIAS <Name of mesh region> <Name of composition>
! -----
REGION_ALIAS REGION_000000001 PIN_MATERIAL
REGION_ALIAS REGION_000000007 MODERATOR_MATERIAL
```

Figure 21. Material Assignment File for Example Case.

The following command executes the problem in parallel with 4 processors:

```
mpiexec -n 4 sn2nd.x -input bench05.inp > bench05.out
```

We have assumed that the executable and input files all reside in the working directory. After execution, the following files are produced:

- bench05.out
- bench05.h5
- Converted_ANLXSto.ISOTXS
- LegacyConversion.pntmesh
- LegacyConversion.pntmesh.partitioning

The first file, bench05.out, is the main text-based output file containing the eigenvalue. This file is included in Appendix F for reference. The second file, bench05.h5, is the HDF5-formatted output file containing the full solution. The remaining files are the ISOTXS and pntmesh conversions of the .anlxs and .nemesh file, respectively. PROTEUS-SN always converts into these base formats. These files can be saved and re-used in the future instead of using the .anlxs and .nemesh files.

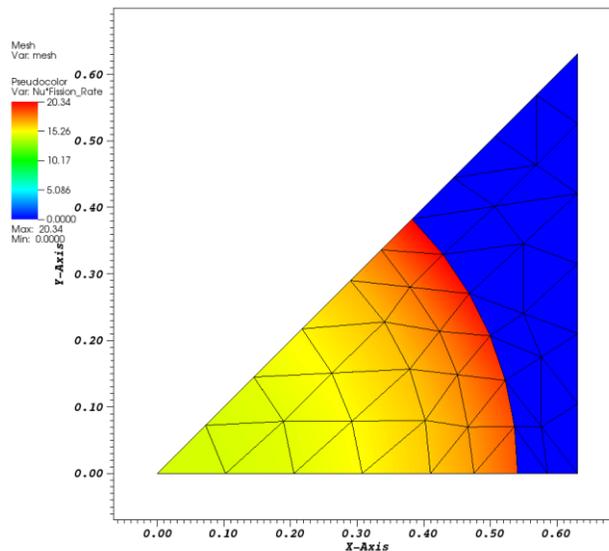


Figure 22. Nu*fission Rate in Test Case Plotted in VisIt.

Importing the .h5 output into visit and plotting the nu*fission rate and the mesh produces a figure like that in Figure 22. Note that VisIt tessellates the quadratic triangular mesh into linear triangles. This solution is also interpolated linearly on the finer mesh rather than quadratically on the original mesh. This limitation is something to consider when using VisIt to plot and obtain higher order solutions.

7. Summary

PROTEUS-SN is a highly scalable, discrete ordinates-based unstructured grid code which solves the even-parity transport equation for nuclear reactor applications. The code is targeted for reactor analysis which requires heterogeneous or unstructured grid geometry representation. The code has been demonstrated to scale to over 200,000 processors and solve problems with nearly 1 trillion space-angle-energy degrees of freedom.

This user manual documents the basics of installing and using PROTEUS-SN. It describes the external library dependencies (MPICH, PETSc, METIS, and HDF5), compilation process, and usage commands. The required input files were also described in detail including examples and file formats. To summarize, PROTEUS-SN requires a finite element mesh, a multigroup cross section data file or a cross section library file, a material assignment file, and a driver input file. For kinetics calculations, an additional kinetics input is required.

Specific tools were recommended for cross section generation, mesh generation, and visualization. For questions or more information on PROTEUS-SN, please contact nera-software@anl.gov.

References

1. C.H. Lee and W.S. Yang, “MC²-3: Multigroup Cross Section Generation Code for Fast Reactor Analysis,” ANL/NE-11-41 (Rev. 1), January, 2012.
2. K.L. Derstine, et al, “DIF3D: A Code to Solve One-, Two-, and Three-Dimensional Finite-Difference Diffusion Theory Problems,” ANL-82-64, Argonne National Laboratory, 1984.
3. VisIt User’s Manual, Version 1.5, UCRL-SM-220449, October 2005.
4. VTK Web page, www.vtk.org.
5. MPICH Web page, <http://www.mpich.org>.
6. George Karypis and Vipin Kumar, “A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs”, SIAM Journal on Scientific Computing, Vol. 20, No. 1, pp. 359—392, 1999.
7. METIS Web page, <http://glaros.dtc.umn.edu/gkhome/metis/metis/overview>
8. S. Balay, J. Brown, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. Curfman McInnes, B. F. Smith, H. Zhang, PETSc Web page, <http://www.mcs.anl.gov/petsc> (2013)
9. The HDF Group. Hierarchical Data Format, version 5, 1997-2014. <http://www.hdfgroup.org/HDF5/>.
10. T.J. Tautges, et al, “MOAB: A Mesh Oriented Database”, Report SAND2004-1592, Sandia National Laboratories, April 2004.
11. 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.
12. H. Childs, et al, “A Contract-Based System for Large Data Visualization”, Proceedings of IEEE Visualization 2005, Minneapolis, MN, pp 190-198, 2005.
13. M. A. Smith and E. R. Shemon, “User Manual for the PROTEUS Mesh Tools”, ANL-NE-15/17 (Rev. 1), Argonne National Laboratory, June, 2015.
14. CUBIT Web page, www.cubit.sandia.gov.
15. NetCDF Web page, <http://www.unidata.ucar.edu/software/netcdf/>.
16. R. Jain, T.J. Tautges, “NEAMS MeshKit: Nuclear Reactor Mesh Generation Solutions”, *Proceedings of International Congress on the Advances in Nuclear Power Plants (ICAPP)*, American Nuclear Society, Charlotte, NC, April 6-9, 2014.
17. T.J. Tautges, R. Meyers, K. Merkley, C. Stimpson, C. Ernst, *MOAB: A Mesh-Oriented Database*, Sandia National Laboratories report SAND2004-1592, April 2004.

APPENDIX A. ANGULAR CUBATURES IN PROTEUS-SN

The following tables describe the resolutions of each angular cubature included with PROTEUS-SN, such as ‘CARLSON_EM’ which is the conventional level symmetric cubature used in most discrete ordinates codes. In each table, the number of points that appears in 2π and 4π is given noting that PROTEUS-SN will report having the 2π number for 3D domains (related to symmetry in the even-parity formulation). The “Library Order” specifies what the theoretical spherical harmonics integration capability of each resolution, while the “Order when Error <10E-6” column indicates the actual spherical harmonics integration capability. For example, one should not use ‘CARLSON_LM’ for a problem with P_5 anisotropic scattering, since the actual order is stated to be only P3.

The LEG_TCHEBY cubatures are product cubatures and thus require inputs for both PHI_RESOLUTION and THETA_RESOLUTION in the code. PHI_RESOLUTION describes the azimuthal resolution, i.e. resolution in the x-y plane. THETA_RESOLUTION describes the polar resolution of the cubature. The resolutions in the table are stated for Resolution=PHI_RESOLUTION=THETA_RESOLUTION, i.e. L1T1, L3T3, etc.

Resolution	CARLSON_EM				CARLSON_LM			
	Points on 2π	Points on 4π	Library Order	Order when Error < 10E-6	Points on 2π	Points on 4π	Library Order	Order when Error < 10E-6
1	4	8	2	3	4	8	2	3
2	12	24	4	5	12	24	4	3
3	24	48	6	7	24	48	6	3
4	40	80	8	9	40	80	8	3
5	60	120	10	11	84	168	10	3
6	84	168	12	11	84	168	12	3
7	112	224	14	11	144	288	14	3
8	144	288	16	11	144	288	16	3

Resolution	CARLSON_EQUALW				PYRAMID_CARLSON			
	Points on 2π	Points on 4π	Library Order	Order when Error < 10E-6	Points on 2π	Points on 4π	Library Order	Order when Error < 10E-6
1	4	8	2	3	4	8	1	3
2	12	24	4	5	12	24	3	3
3	24	48	6	5	24	48	5	5
4	40	80	8	9	40	80	7	7
5	60	120	10	11	60	120	9	9
6	84	168	12	13	84	168	11	11
7	112	224	14	11	112	224	13	13
8	144	288	16	17	144	288	15	15
9					180	360	17	17
10					220	440	19	19

Resolution	LEG-TCHEBY				X_DIR_LEG-TCHEBY			
	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$
1	4	8	2	3	4	8	2	3
2	9	18	4	5	9	18	4	5
3	16	32	6	7	16	32	6	7
4	25	50	8	9	25	50	8	9
5	36	72	10	11	36	72	10	11
6	49	98	12	13	49	98	12	13
7	64	128	14	15	64	128	14	15
8	81	162	16	17	81	162	16	17
9	100	200	18	19	100	200	18	19
10	121	242	20	21	121	242	20	21
11	144	288	22	23	144	288	22	23
12	169	338	24	25	169	338	24	25
13	196	392	26	27	196	392	26	27
14	225	450	28	29	225	450	28	29
15	256	512	30	31	256	512	30	31
16	289	578	32	33	289	578	32	33

Resolution	DOUB_LEG-TCHEBY				TRI_LEG-TCHEBY			
	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$
1	8	16	2	3	4	8	2	3
2	18	36	4	5	12	24	4	3
3	32	64	6	7	24	48	6	3
4	50	100	8	9	40	80	8	3
5	72	144	10	11	60	120	10	3
6	98	196	12	13	84	168	12	3
7	128	256	14	15	112	224	14	3
8	162	324	16	17	144	288	16	3
9	200	400	18	19	180	360	18	3
10	242	484	20	21	220	440	20	3
11	288	576	22	23				
12	338	676	24	25				
13	392	784	26	27				
14	450	900	28	29				
15	512	1024	30	31				
16	578	1156	32	33				

Res.	LEBEDEV-LAIKOV				LEBEDEV-LAIKOV				
	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$	Res.	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$
1	3	6	3	3	11	97	194	23	23
2	7	14	5	5	12	115	230	25	25
3	13	26	7	7	13	133	266	27	27
4	19	38	9	9	14	151	302	29	29
5	25	50	11	11	15	175	350	31	31
6	37	74	13	13	16	217	434	35	35
7	43	86	15	15	17	295	590	41	41
8	55	110	17	17	18	385	770	47	47
9	73	146	19	19	19	487	974	53	53
10	85	170	21	21	20	601	1202	59	59

Resolution	THURGOOD_TN				THURGOOD_LEASTSQ			
	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$
1	4	8	3	3	4	8	3	3
2	16	32	3	3	16	32	3	3
3	36	72	3	3	36	72	3	3
4	64	128	5	3	64	128	5	5
5	100	200	7	3	100	200	7	7
6	144	288	9	3	144	288	9	9
7	196	392	11	3	196	392	11	11
8	256	512	13	3	256	512	13	13
9	324	648	15	3	324	648	15	15
10	400	800	17	3	400	800	17	17

Resolution	TEG_CORNER_EQWT				TEG_CORNER_LSQ			
	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$
1	6	12	3	5	6	12	3	5
2	21	42	5	5	21	42	5	9
3	46	92	9	5	46	92	9	11
4	81	162	11	5	81	162	11	15
5	126	252	15	5	126	252	15	17
6	181	362	17	5	181	362	17	21
7	246	492	21	5	246	492	21	23
8	321	642	23	5	321	642	23	27
9	406	812	27	5	406	812	27	29
10	501	1002	31	5	501	1002	31	33
11	606	1212	33	5	606	1212	33	9
12	721	1442	37	5	721	1442	37	9
13	846	1692	39	5	846	1692	39	5
14	981	1962	43	5	981	1962	43	9
15	1126	2252	45	5	1126	2252	45	49

Resolution	TEG_CENTROID_EQW				TEG_CENTROID_LSQ			
	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$
1	10	20	3	5	10	20	3	5
2	40	80	7	5	40	80	7	9
3	90	180	11	5	90	180	11	5
4	160	320	17	5	160	320	17	5
5	250	500	21	5	250	500	21	5
6	360	720	25	5	360	720	25	5
7	490	980	29	5	490	980	29	5
8	640	1280	35	5	640	1280	35	5
9	810	1620	39	5	810	1620	39	5
10	1000	2000	43	5	1000	2000	43	5
11	1210	2420	47	5	1210	2420	47	5
12	1440	2880	53	5	1440	2880	53	5
13	1690	3380	57	5	1690	3380	57	5
14	1960	3920	61	5	1960	3920	61	21
15	2250	4500	65	5	2250	4500	65	11

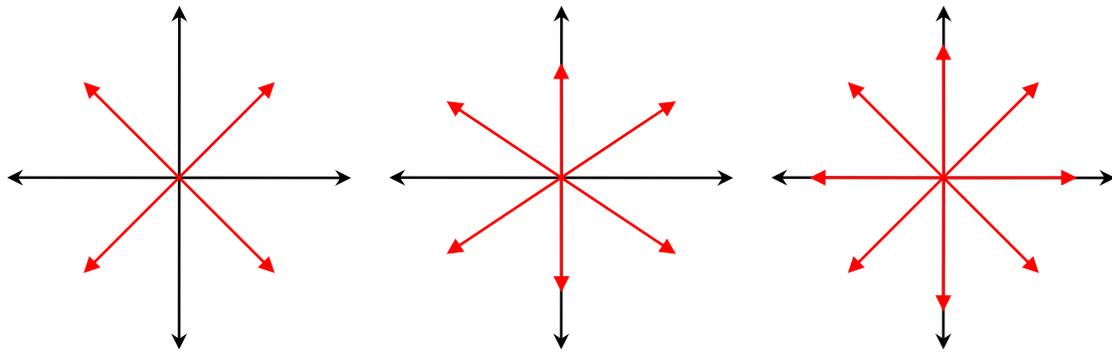
Resolution	COBE_EVEN_EQWT				COBE_EVEN_LSQ			
	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$
1	12	24	1	3	12	24	1	3
2	48	96	7	3	48	96	7	3
3	108	216	11	3	108	216	11	3
4	192	384	17	3	192	384	17	3
5	300	600	21	3	300	600	21	3
6	432	864	27	3	432	864	27	3
7	588	1176	31	3	588	1176	31	3
8	768	1536	37	3	768	1536	37	3
9	972	1944	41	3				
10	1200	2400	45	3				

Resolution	COBE_ODD_EQWT				COBE_ODD_LSQ			
	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$
1	3	6	1	3	3	6	1	3
2	27	54	7	3	27	54	7	3
3	75	150	11	3	75	150	11	3
4	147	294	17	3	147	294	17	3
5	243	486	21	3	243	486	21	3
6	363	726	27	3	363	726	27	3
7	507	1014	31	3	507	1014	31	3
8	675	1350	37	3	675	1350	37	3

Resolution	COBE_C_EVEN_EQWT				COBE_C_EVEN_LSQ			
	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$
1	4	8	2	3	4	8	2	3
2	28	56	5	3	28	56	5	7
3	76	152	12	3	76	152	12	3
4	148	296	16	3	148	296	16	17
5	244	488	22	3	244	488	22	3
6	364	728	26	3	364	728	26	27
7	508	1016	32	3	508	1016	32	3
8	676	1352	36	3	676	1352	36	3
9	868	1736	40	3	868	1736	40	41
10	1084	2168	44	3	1084	2168	44	45

Resolution	COBE_C_ODD_EQWT				COBE_C_ODD_LSQ			
	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$	Points on 2π	Points on 4π	Library Order	Order when Error < $10E-6$
1	13	26	5	3	13	26	5	7
2	49	98	10	3	49	98	10	11
3	109	218	16	3	109	218	16	17
4	193	386	24	3	193	386	24	3
5	301	602	24	3	301	602	24	25
6	433	866	30	3	433	866	30	31
7	589	1178	36	3	589	1178	36	3
8	769	1538	40	3	769	1538	40	41
9	973	1946	46	3	973	1946	46	3
10	1201	2402	52	3	1201	2402	52	3

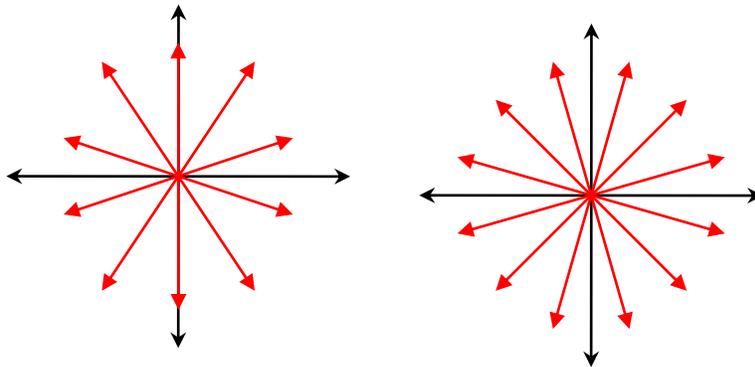
The number of directions associated with the THETA_RESOLUTION is found to be $\theta + 1$ directions on 4π and $\frac{1}{2}(\theta + 1)$ directions on 2π . The number of directions associated with PHI_RESOLUTION is indicated in the figure in the next page noting that for 2π , only one of the directions that travels in the Y direction will become part of the cubature for one when there is one direction traveling in the x-y plane associated with THETA_RESOLUTION. This usually leads to prime numbers for the section of angles and thus we strongly suggest you stick to using odd resolution numbers. Also note the x-y symmetries typically imposed for reactor geometries can be exactly handled by selecting specific resolutions of the PHI_RESOLUTION as indicated below.



Res. 1
 $\delta = \frac{\pi}{2} = 90^\circ$

Res. 2
 $\delta = \frac{\pi}{3} = 60^\circ$

Res. 3
 $\delta = \frac{\pi}{4} = 45^\circ$



Res. 4
 $\delta = \frac{\pi}{5} = 36^\circ$

Res. 5
 $\delta = \frac{\pi}{6} = 30^\circ$

General Tchebychev Relation
 $\delta = \frac{\pi}{(R+1)}$ Points = $2(R+1)$ $\phi = \frac{\delta}{2} + (i-1)\delta$ $i \in 1, \text{Points}$
 Resolutions with 45 degree symmetry = 1,3,5,7,9,...odd
 Resolutions with 30 or 60 degree symmetry = 2,5,8,11,14,17,20,23

Appendix B. Example PROTEUS-SN Driver Input File: “sn2nd.inp”

The following is an example of the driver input file for PROTEUS-SN, taken from bench05. This problem uses Legendre-Tchebyshev cubature of order L3T9. It uses Tchebychev acceleration and the SSOR preconditioner. The maximum number of outer iterations is set to 100 and the upscatter iterations are limited to 5. Note that this driver input files points to the other required input files using the SOURCEFILE_MESH, SOURCEFILE_MATERIAL, SOURCEFILE_XS keywords.

```

SN_TYPE                LEG-TCHEBY
Theta_Resolution       3
Phi_Resolution         9
SEGMENT_ANGLE         0

DEBUG_PRINT_LEVEL      0
DEBUG_PRINT_SETUP      0
DEBUG_PRINT_FORMATION  0
DEBUG_PRINT_OUTER     0
THERMAL_POWER         1.0

USE_PRECONDITIONER     SSOR      ! SSOR,SOR,ILU,ICC
USE_TCHEBYCHEV_ACCEL  YES

Scattering_Order       0          ! 0,1,2,3,4,5,...
EIGENVALUE_GUESS      1.17      ! The guess for the initial eigenvalue
Tolerance_eigenvalue   1.0e-7    ! The maximum relative error to allow on the eigenvalue
Tolerance_Fission      1.0e-8    ! The maximum relative error to allow on the outer iterations for the
flux
Tolerance_Flux         1.0e-7    ! The maximum relative error to allow on the inner iterations for the
flux
ITERATIVE_IMPROVEMENT 0.1

Iterations_Fission     100       ! The maximum number of outer iterations
Iterations_UpScatter   5         ! The maximum number of upscatter iterations
Iterations_scattering  5
Iterations_SA_CG       100
Iterations_PETSc       1000

Solve_Type both

SOURCEFILE_MESH        bench.nemesh
SOURCEFILE_XS          bench.anlxs
SOURCEFILE_MATERIAL    bench.assignment
BASIC_BWO              no
    
```

Appendix C. Cross Section File Format for PROTEUS-SN: “anlxs”

This section specifies the PROTEUS-SN “anlxs” cross section file format. An example input file is also provided.

C.1 File format description for “anlxs” cross section file

```
*****
*
*      FILE FORMAT DESCRIPTION      *
*
*      <<<<  ANLXS  >>>>          *
*
*      Last updated: 06/18/2013     *
*      nera-software@anl.gov        *
*
*****

***  CARD TYPE DIRECTORY  ***

====  =====
CARD  CONTENTS
====  =====
00  BASIC DATA
01  GROUP BOUNDARIES
02  GROUP VELOCITIES
** Repeat Cards 03-06 for each isotope **
03  ISOTOPE/COMPOSITION NAME
04  BASIC PARAMETERS
** Repeat Card 05 for each group **
05  GROUP CROSS SECTIONS
** Repeat Card 06 for each group, and for each Legendre moment **
** Outer loop: 1,LegendreMoments, Inner Loop:1,Groups **
06  LEGENDRE GROUP SCATTERING MATRIX

! Card types 03-06 are repeated for each isotope as follows
! DO I=1,NumIsotopes
!   READ CARD 03
!   READ CARD 04
!   DO G=1,NumGroups
!     READ CARD 05
!     ENDDO
!     DO L=1,NumLegendre
!       DO G=1,NumGroups
!         READ CARD 06
!       ENDDO
!     ENDDO
!   ENDDO

***  CARD TYPE DESCRIPTIONS  ***

====  =====
CARD TYPE 00: BASIC DATA
====  =====
LINE NUMBER: 1

FORMAT: <NumIsotopes> <NumGroups> <DebugPrint>

DESCRIPTION:
<NumIsotopes> : Integer...Number of isotopes
<NumGroups>   : Integer...Number of energy groups
<DebugPrint>  : 0-10.....Debug print flag to apply within the "HomogenizedXS" module
                (0=no debug, 10=full debug)
```

```
=====
CARD TYPE 01: GROUP BOUNDARIES
=====
LINE NUMBER: 2
```

FORMAT: <EnergyGroupBdry(G)>, G=1,NumGroups+1

DESCRIPTION:

<EnergyGroupBdry> : Real array of energy boundaries that define the group structure (listed highest to lowest). The last value is the lower energy threshold.

```
=====
CARD TYPE 02: GROUP VELOCITIES
=====
LINE NUMBER: 3
```

FORMAT: <Velocity(G)>,G=1,NumGroups

DESCRIPTION:

<Velocity> : Real array of average neutron velocity in each group (listed highest to lowest)

```
=====
CARD TYPE 03: ISOTOPE/COMPOSITION NAME
=====
LINE NUMBER: 1 in local isotope block
```

FORMAT: <IsotopeName>

DESCRIPTION:

<IsotopeName> : Character... Name of the isotope or composition being described in this block

```
=====
CARD TYPE 04: BASIC PARAMETERS
=====
LINE NUMBER: 2 in local isotope block
```

FORMAT: <NumLegendre> <DataPresent> <Energy_Fission> <Energy_Capture> <Gram_Atom_Weight> [Optional Temp]

DESCRIPTION:

<NumLegendre> : Integer....The number of Legendre terms in the scattering data for each isotope

<DataPresent> : 0..... Total cross sections provided
1.....Total,Fission,Nu,Chi cross sections provided
2.....Total,Alpha,Proton,N2N,Deuteron,Tritium,Fission,Nu*Fission,Chi cross sections provided

<Energy_Fission> : Real.....Thermal energy released per fission (W-seconds/fission)

<Energy_Capture> : Real.....Thermal energy released per capture (W-seconds/capture)

<Gram_Atom_Weight>: Real.....Gram-atom weight of the isotope (use Avogadro's 0.602214179 for macroscopic xs)

[Optional Temp] : Real.....If present, the data temperature in Kelvin.
If not present, default assumes data is given at 300 K.

```
=====
CARD TYPE 05: GROUP CROSS SECTIONS
=====
LINE NUMBER: 3 or greater in local isotope block
```

FORMAT: Depends on <DataSet> from CARD TYPE 04 in local isotope block

DataPresent=0 : <Total(Group)>

DataPresent=1 : <Total(Group)> <Fission(Group)> <Nu*Fission(Group)> <Chi(Group)>

DataPresent=2 : <Total(Group)> <Alpha(Group)> <Proton(Group)> <N2N(Group)> <Deuteron(Group)> &
<Tritium(Group)> <Fission(Group)> <NuFission(Group)> <Chi(Group)>

DESCRIPTION:

The multigroup cross section data for the given isotope and group are given according to the above formats.

The types of cross sections are defined here:

<Total> = Total cross section
<Alpha> = n,alpha cross section
<Proton> = n,proton cross section
<N2N> = n,2n reaction based cross section (i.e. 1/2 of scattering matrix sum)
<Deuteron> = n,deuteron cross section
<Tritium> = n,tritium cross section
<Fission> = Fission cross section
<NuFission> = Nu*Fission cross section
<Chi> = Chi emission spectrum
<Scattering> = Scattering cross section assuming Gprime -> Group for Legendre moment Leg

=====
CARD TYPE 06: LEGENDRE GROUP SCATTERING MATRIX
=====
LINE NUMBER: 4 or greater in local isotope block

FORMAT: <Scattering(Group,Gprime,Leg)>, Gprime=1,NumGroups

DESCRIPTION:

The group-to-group scattering matrix is printed for each energy group and each Legendre moment.

<Scattering(Group,Gprime,Leg)> = Scattering cross section from Gprime -> Group for Legendre moment Leg

* END OF FILE FORMAT DESCRIPTION *
* <<< ANLXS >>> *

C.2 Example "anlxs" cross section file (taken from bench02 directory)

```
3 2 0
1.0000E+07 6.82560E-01 1.00000E-05
1.00000E+05 1.00000E-01
Fuel
1 1 1.0 0.0 0.602214179
0.22 0.003 0.006 1.0
0.8 0.05 0.1 0.0
0.193 0.017
0.0 0.73
Clad
1 0 1.0 0.0 0.602214179
0.53
0.94
0.526 0.001
0.0 0.83
Moderato
1 0 1.0 0.0 0.602214179
0.701
2.00
0.65 0.05
0.0 1.97
```

Appendix D. Mesh File Format Specification for PROTEUS-SN: “nemesh”

This section specifies the PROTEUS-SN “nemesh” mesh file format. An example “nemesh” input file is also provided.

D.1 File Format Description for “nemesh” mesh file

```
*****
*                               *
*   FILE FORMAT DESCRIPTION   *
*                               *
*   <<<< NEMESH >>>>         *
*                               *
*   Last updated: 06/18/2013   *
*   nera-software@anl.gov     *
*                               *
*****

*** CARD TYPE DIRECTORY ***

====
CARD CONTENTS
====
00 HEADER LINES
01 CONTROL FLAGS
02 BASIC MESH INFO
03 ELEMENT LIST
04 ELEMENT CONNECTIVITY
05 VERTEX POSITION DATA
06 BOUNDARY SURFACE LIST
07 BLANK LINE

*** CARD TYPE DESCRIPTIONS ***

=====
CARD TYPE 00: HEADER LINES
=====
LINE NUMBERS: 1-9

FORMAT: N/A

DESCRIPTION:
The first nine lines of the input file must be blank or commented.
Do not place any other card data in lines 1-9 of the input file.

=====
CARD TYPE 01: CONTROL FLAGS
=====
LINE NUMBER: 10

FORMAT: <Input Style Flag> <Debug Print Flag>

DESCRIPTION:
<Input Style Flag> : 0...Indexed Input (numbered elements and nodes)
                   1...Not Indexed

<Debug Print Flag> : Integer...0-10 (0=no printing, 10=full debug printing)

=====
CARD TYPE 02: BASIC MESH INFO
=====
LINE NUMBER: 11

FORMAT: <NumElements> <NumNodes> <unused integer> <NumBoundarySurfaces>
```

DESCRIPTION:
<NumElements> : Integer...Number of elements in the mesh

<NumNodes> : Integer...Number of nodes (vertices) in the mesh

<unused integer> : 0.....(reserved for future use)

<NumBoundarySurfaces> : Integer...Number of boundary surfaces in the mesh

=====
CARD TYPE 03: ELEMENT LIST
=====
LINE NUMBERS: 12+, with a total of NumElements lines

FORMAT: { [Optional Index] <ElementType> <MaterialBlock> };
repeat for each element specified in <NumElements>

DESCRIPTION:
[Optional Index] : Integer... Optional index of the element

<ElementType> : Integer from the following list

1...1-D Bar	Linear
2...1-D Bar	Quadratic
5...2-D Triangular	Linear
6...2-D Triangular	Quadratic
10...2-D Quadrilateral	Linear
11...2-D Quadrilateral	Quadratic
15...3-D Tetrahedron	Linear
16...3-D Tetrahedron	Quadratic
20...3-D Tri. Prismatic	Linear
21...3-D Tri. Prismatic	Quadratic
25...3-D Hexadron	Linear
26...3-D Hexadron	Quadratic
31-39...1-D Bar	Lagrangian (31 = linear, 39 = 9th order)
41-49...2-D Triangular	Lagrangian (41 = linear, 49 = 9th order)
51-59...2-D Quadrilateral	Lagrangian (51 = linear, 59 = 9th order)
61-69...3-D Tetrahedron	Lagrangian (61 = linear, 69 = 9th order)
71-79...3-D Tri. Prismatic	Lagrangian (71 = linear, 79 = 9th order)
81-89...3-D Hexadron	Lagrangian (81 = linear, 89 = 9th order)
91-99...2-D Quadrilateral	NonConforming (Lagrangian setup)
101-109...3-D Hexadron	NonConforming (Lagrangian setup)

<MaterialBlock> : Integer... Material block for this element

=====
CARD TYPE 04: ELEMENT CONNECTIVITY
=====
LINE NUMBERS: Immediately following last CARD TYPE 03, with a total of NumElements lines

FORMAT: { [Optional Index] <ConnectivityArray> };
repeat for each element specified in <NumElements>

DESCRIPTION:
[Optional Index] : Integer... Optional index of the element

<ConnectivityArray> : Array of integers read as CONNECTIVITY(J),J=1,ELEMENTVERTICES
This array lists the indexed vertices which define the element.
ELEMENTVERTICES is determined automatically from ElementType.
For example ElementType 1 (1-D Bar Linear) has 2 vertices.

=====
CARD TYPE 05: VERTEX POSITION DATA
=====
LINE NUMBERS: Immediately following last CARD TYPE 04, with a total of NumNodes lines

FORMAT: { [Optional Index] <X> [Y] [Z] };
repeat for each node specified in <NumNodes>

DESCRIPTION:

[Optional Index] : Integer.... Optional index of the node

<X> : Real....X-value of this node (always present)

[Y] : Real....Y-value of this node (present only for 2D and 3D elements)

[Z] : Real....Z-value of this node (present only for 3D elements)

=====
CARD TYPE 06: BOUNDARY SURFACE LIST
=====

LINE NUMBERS: Immediately following last CARD TYPE 05, with a total of NumBoundarySurfaces lines

FORMAT: { [Optional Index] <ElementIndex> <ReferenceSurface> <BoundaryConditionFlag> };
repeat for each node specified in <NumNodes>

DESCRIPTION:

[Optional Index] : Integer....Optional index of the boundary surface

<ElementIndex> : Integer....Index of the element with the boundary surface

<ReferenceSurface> : Integer....Reference element surface number corresponding to the boundary surface

<BoundaryConditionFlag> : 1..... Reflected
Other....Void

=====
CARD TYPE 07: BLANK LINE
=====

LINE NUMBER: Last line in the file

FORMAT: blank line

DESCRIPTION:

A blank line must be placed at the end of the line in order to avoid "end of file" error upon reading.

* END OF FILE FORMAT DESCRIPTION *
* <<<< NEMESH >>> *

D.2 Example “nemesh” file (taken from bench05)

! ANL FINITE ELEMENT INPUT – NEMESH FILE DESCRIPTION - 9 HEADER LINES ALWAYS

! This example was taken from bench05 of the benchmarks directory

! Header line 3

! Header line 4

! Header line 5

! Header line 6

! Header line 7

! Header line 8

! Header line 9

0 0

16 45 0 12 ! CARD TYPE 1 control info - indexed input with no debug printing

! CARD TYPE 2 mesh info - 16 elements, 45 nodes, 12 boundary surfs

! CARD TYPE 3 element info - first element is type 6 with material 1

1 6 1

2 6 1

3 6 1

4 6 1

5 6 1

6 6 1

7 6 1

8 6 1

```

9      6      1
10     6      7
11     6      7
12     6      7
13     6      7
14     6      7
15     6      7
16     6      7

1      1      2      3      11      17      10 ! CARD TYPE 4 connectivity: element 1 is connected to
2      3      4      5      13      19      12 !
3      17     11     3      12      19      18
4      17     18     19     24      28      23
5      5      6      7      15      21      14
6      5      14     21     20      19      13
7      19     20     21     26      30      25
8      28     24     19     25      30      29
9      28     29     30     34      37      33
10     7      8      9      16      21      15
11     9      22     32     27      21      16
12     21     27     32     31      30      26
13     32     36     39     35      30      31
14     30     35     39     38      37      34
15     37     38     39     41      42      40
16     39     43     45     44      42      41

1      0 ! CARD TYPE 5 node info: node 1 at (0.0,0.0)
2      0.1023750000000001 0
3      0.20475 0
4      0.307125 0
5      0.4094999999999999 0
6      0.4747513185137461 0
7      0.5400026370274923 0
8      0.5850013185137463 0
9      0.63 0
10     0.07239005672397281 0.07239005672397281
11     0.1891643342816859 0.07835443277675214
12     0.291539334281686 0.07835443277675214
13     0.4016315723251228 0.07988948686560449
14     0.4655512466845283 0.06988148343414935
15     0.5353828396092143 0.07048448799998519
16     0.5758012466845285 0.06988148343414935
17     0.1447801134479456 0.1447801134479455
18     0.2615543910056587 0.1507444895007249
19     0.3783286685633719 0.1567088655535044
20     0.4499655809662141 0.1482359162109015
21     0.5216024933690565 0.1397629668682987
22     0.63 0.1050000000000001
23     0.2171701701719185 0.2171701701719183
24     0.3404868062378923 0.2275060104215271
25     0.4229923351698838 0.2133550920336252
26     0.4988973838518217 0.2066500626238805
27     0.5758012466845285 0.1748814834341492
28     0.2895602268958912 0.2895602268958912
29     0.3786081143361436 0.2797807727048186
30     0.4676560017763959 0.2700013185137462
31     0.548828008881977 0.2400006592568731
32     0.63 0.2100000000000001
33     0.3356998766983245 0.3356998766983244
34     0.4284128958518366 0.3287327769853338
35     0.548828008881977 0.3450006592568731
36     0.63 0.3150000000000001
37     0.3818395265007578 0.3818395265007578
38     0.5059197632503788 0.400919763250379
39     0.63 0.4199999999999998
40     0.4438796448755683 0.4438796448755683
41     0.5679598816251895 0.4629598816251895
42     0.5059197632503788 0.5059197632503788
43     0.63 0.5249999999999998
44     0.5679598816251895 0.5679598816251895
45     0.63 0.63
    
```

```
1  1  1  1  ! CARD TYPE 6 boundary data: the first boundary surface is on ele 1, ref. surf 1, refl bc
2  1  3  1
3  2  1  1
4  4  3  1
5  5  1  1
6  9  3  1
7 10  1  1
8 11  1  1
9 13  1  1
10 15  3  1
11 16  1  1
12 16  2  1
```

! MAKE CERTAIN THERE IS AN EXTRA BLANK LINE AFTER ALL OF THE INPUT TO PREVENT EOF ERROR

Appendix E. Mesh File Format Specification for PROTEUS-SN: “ascii”

This section specifies the PROTEUS-SN “ascii” mesh file format. An example input file is also provided.

E.1 File Format Description for “ascii” mesh file

```
*****
*                               *
*       FILE FORMAT DESCRIPTION   *
*                               *
*       <<<< ASCII >>>>         *
*                               *
*       Last updated: 06/18/2013  *
*       nera-software@anl.gov     *
*                               *
*****

*** CARD TYPE DIRECTORY ***

====
CARD CONTENTS
====
01 BASIC MESH INFO
02 VERTEX POSITION DATA
03 BLOCK DATA

*** CARD TYPE DESCRIPTIONS ***

=====
CARD TYPE 01: BASIC MESH INFO
=====
LINE NUMBER: 1

FORMAT(4I10): <NumVertices> <NumDimensions> <NumBlocks> <unused integer>

DESCRIPTION:
<NumVertices> : (I10)...Number of nodes (vertices) in the mesh
<NumDimensions> : (I10)...Number of dimension (1,2,3)
<NumBlocks> : (I10)...Number of blocks (block=group of elements with common type and material)
<unused integer> : (I10)...0

=====
CARD TYPE 02: VERTEX POSITION DATA
=====
LINE NUMBERS: 2+, number of lines = ceil{NumVertices*NumDimensions/8}

FORMAT(8(1PE15.8)): <X_Vert_Array> <Y_Vert_Array> <Z_Vert_Array>
                These arrays will break over multiple lines in groups of 8.

DESCRIPTION:
<X_Vert_Array> : Reals.... X value of each vertex in the mesh
<Y_Vert_Array> : Reals.... Y value of each vertex in the mesh
                  (present only for 2D and 3D problems)
<Z_Vert_Array> : Reals.... Z value of each vertex in the mesh
                  (present only for 3D problems)

=====
CARD TYPE 03: BLOCK DATA
=====
LINE NUMBERS: immediately following last card type 02 data

DESCRIPTION:
*** The following lines are repeated for each block in NumBlocks:***
CARD SUBTYPE 3A, FORMAT( 6I10): <NumElementsInBlock> <NumBoundarySurfacesInBlock> <NodesPerElement>
```

```

                <unused integer> <Order_Interp> <Order_Equat>
CARD SUBTYPE 3B, FORMAT( 2A16): <ElementType> <RegionName>
CARD SUBTYPE 3C, FORMAT(12I10): { <ConnectivityArray> }; repeat for each element in block
CARD SUBTYPE 3D, FORMAT( 7A16): { <SurfaceBCTag> }; repeat for each surface element in block
CARD SUBTYPE 3E, FORMAT(12I10): { <ElementNumber> <ReferenceSurface> } repeat for each boundary surface in
block

```

```

<NumElementsInBlock>      : Integer....Number of elements in this block

<NumBoundarySurfInBlock>  : Integer....Number of boundary surfaces in this block

<NodesPerElement>        : Integer....Number of nodes on an element
                           (determined by element type and order)

<unused integer>          : 0 (reserved for future use)

<Order_Interp>            : Integer....Order of the spatial interpolation for
                           elements in this block (1=linear, 2=quadratic...)

<Order_Equat>             : Integer....Order of the spatial equations

<ElementType>            : Character....from the following list:
                           BAR.....1-D element
                           QUADRILATERAL...2-D 4-sided element
                           TRIANGLE.....2-D 3-sided element
                           HEXAGON.....2-D 6-sided element
                           SPHERICAL.....2-D spherical element
                           CYLINDRICAL....2-D circular element
                           BRICK.....3-D brick
                           TETRAHEDRON....3-D tetrahedon
                           HEXPRISM.....3-D hexahedral prism
                           PRISMATIC.....3-D prismatic

<RegionName>              : Character... The assigned region name for the current
                           block of elements, ex. REGION_00000001

<ConnectivityArray>      : CONNECTIVITY(J),J=1,NodesPerElement lists the global indices of nodes on the
element.
                           For example a quadratic triangular element will list 6 nodes.
                           This array is then repeated for each element in the block.

<SurfaceBCTag>           : Character.... (REFLECTIVE or VOID)

<ElementNumber>          : Integer.... Element corresponding to this boundary surface

<ReferenceSurface>       : Integer.... Reference element surface number corresponding to this boundary
surface

```

```

*****
*   END OF FILE FORMAT DESCRIPTION   *
*   <<<< ASCII >>>>                 *
*****

```

E.2 Example “ascii” mesh file (converted from bench05 nemes file)

```

    45      2      2      0
0.0000000E+00 1.02375000E-01 2.04750000E-01 3.07125000E-01 4.09500000E-01 4.74751319E-01 5.40002637E-01 5.85001319E-01
6.30000000E-01 7.23900567E-02 1.89164334E-01 2.91539334E-01 4.01631572E-01 4.65551247E-01 5.35382840E-01 5.75801247E-01
1.44780113E-01 2.61554391E-01 3.78328669E-01 4.49965581E-01 5.21602493E-01 6.30000000E-01 2.17170170E-01 3.40486806E-01
4.22992335E-01 4.98897384E-01 5.75801247E-01 2.89560227E-01 3.78608114E-01 4.67656002E-01 5.48828001E-01 6.30000000E-01
3.35699877E-01 4.28412896E-01 5.48828001E-01 6.30000000E-01 3.81839527E-01 5.05919763E-01 6.30000000E-01 4.43879645E-01
5.67959882E-01 5.05919763E-01 6.30000000E-01 5.67959882E-01 6.30000000E-01 0.00000000E+00 0.00000000E+00 0.00000000E+00
0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00 7.23900567E-02 7.83544328E-02
7.83544328E-02 7.98894869E-02 6.98814834E-02 7.04844880E-02 6.98814834E-02 1.44780113E-01 1.50744490E-01 1.56708866E-01
1.48235916E-01 1.39762967E-01 1.05000000E-01 2.17170170E-01 2.27506010E-01 2.13355092E-01 2.06650063E-01 1.74881483E-01
2.89560227E-01 2.79780773E-01 2.70001319E-01 2.40000659E-01 2.10000000E-01 3.35699877E-01 3.28732777E-01 3.45000659E-01
3.15000000E-01 3.81839527E-01 4.00919763E-01 4.20000000E-01 4.43879645E-01 4.62959882E-01 5.05919763E-01 5.25000000E-01
5.67959882E-01 6.30000000E-01
    9      6      6      0      2      2
TRIANGLE REGION_00000001
    1      2      3      11      17      10      3      4      5      13      19      12
    17      11      3      12      19      18      17      18      19      24      28      23
    5      6      7      15      21      14      5      14      21      20      19      13
    19      20      21      26      30      25      28      24      19      25      30      29
    28      29      30      34      37      33
REFLECTIVE REFLECTIVE REFLECTIVE REFLECTIVE REFLECTIVE REFLECTIVE
    1      1      1      3      2      1      4      3      5      1      9      3
    7      6      6      0      2      2
TRIANGLE REGION_00000007
    7      8      9      16      21      15      9      22      32      27      21      16
    21      27      32      31      30      26      32      36      39      35      30      31
    30      35      39      38      37      34      37      38      39      41      42      40
    39      43      45      44      42      41
REFLECTIVE REFLECTIVE REFLECTIVE REFLECTIVE REFLECTIVE REFLECTIVE
    1      1      2      1      4      1      6      3      7      1      7      2
    
```



```

[SN2ND]...Debug print setting for iteration history..... 0
[SN2ND]...Global number of energy groups..... 7
[SN2ND]...Global number of spatial mesh points..... 45
[SN2ND]... Corresponding number of finite elements..... 16
[SN2ND]... Number of spatial dimensions..... 2
[SN2ND]...Cubature to use for the angular domain.....LEG-TCHEBY
[SN2ND]...Global angular expansion order..... 6
[SN2ND]... Number of even-parity directions..... 20
[SN2ND]... Expansion order of the scattering kernel..... 0
[SN2ND]... Even-parity spherical harmonics..... 1
[SN2ND]... Odd -parity spherical harmonics..... 0
[SN2ND]...Global space-angle degrees of freedom..... 900
[SN2ND]...Source type information.....
[SN2ND]... Is a fission source present..... Yes
[SN2ND]... Is a fixed source present..... No
[SN2ND]... Is a up/down scattering present..... Yes
[SN2ND]...Outer iteration control variables.....
[SN2ND]... Iterative improvement scaling factor..... 1.000E-01
[SN2ND]... Use Tchebychev acceleration..... Yes
[SN2ND]... Targeted error on the eigenvalue..... 1.000E-07
[SN2ND]... Targeted error on the fission source..... 1.000E-08
[SN2ND]... Targeted error on the MGS flux solution..... 1.000E-07
[SN2ND]... Maximum fission source iterations..... 100
[SN2ND]... Maximum upscatter iterations..... 5
[SN2ND]...Scattering iteration control variables.....
[SN2ND]... Use synthetic diffusion acceleration..... Yes
[SN2ND]... Maximum number of scattering iterations..... 5
[SN2ND]... Maximum space-angle CG iterations..... 100
[SN2ND]... Maximum PETSc preconditioner iterations..... 1000
[SN2ND]...Break down of total job memory usage (MB)..... PEAK TOTAL
[SN2ND]... Memory usage for spatial mesh..... 0.001900..... 0.007599
[SN2ND]... Memory usage for spatial matrices..... 0.022003..... 0.088013
[SN2ND]... Vector memory usage..... 0.001572..... 0.006287
[SN2ND]... Memory usage for A matrices..... 0.076134..... 0.264355
[SN2ND]... Memory usage for PC matrices..... 0.014420..... 0.050468
[SN2ND]... Memory usage for scattering source..... 0.005127..... 0.020508
[SN2ND]... Memory usage for fission source..... 0.008652..... 0.034607
[SN2ND]... Scratch memory for power iteration..... 0.041199..... 0.164795
[SN2ND]... Total tracked memory usage.(overhead?)..... 0.171005..... 0.636631
[SN2ND]... Total reported memory usage..... 0.041199..... 0.164795
[SN2ND]...Solving the preceding Sn system of equations.....
=SN2ND Fission Source Iteration
=SN2ND Seconds|Itr|Up| Eigenvalue | Error | Fis Err| Flux Err| Dom | Slope| Error |Active| Dom |Scat| Within Group Flux | Preconditioner CG | Synth |
=SN2ND 0.1|001|01| 1.19515E+00| 2.1E-02| 5.5E-02|T 3.9E-02| 0.000| 0.000| 3.5E-01| T 0| 0.055| 7| 381| 101| 1| 1.097E+00| 22823| 116| 1| 9| 75|
=SN2ND 0.1|002|01| 1.16719E+00| 2.3E-02| 2.3E-02|T 1.4E-03| 0.009| 0.000| 1.2E-02| T 0| 0.420| 7| 249| 35| 2| 7.444E-02| 18232| 117| 1| 9| 79|
=SN2ND 0.0|003|01| 1.16773E+00| 4.6E-04| 6.9E-04|T 1.5E-04| 0.322| 0.000| 6.3E-03| T 1| 0.029| 7| 103| 12| 2| 4.363E-02| 8671| 112| 1| 9| 78|
=SN2ND 0.0|004|01| 1.16651E+00| 1.0E-03| 1.0E-03|T 6.4E-05| 0.041| 0.000| 4.0E-03| T 0| 1.500| 7| 46| 5| 7| 1.306E-02| 3309| 112| 1| 8| 82|
=SN2ND 0.3|005|03| 1.17301E+00| 5.6E-03| 3.1E-03|T 1.6E-04|36.314| 0.405| 1.0E-06| T 1| 2.997| 35| 1212| 65| 7| 1.179E-05| 82842| 116| 1| 9| 387|
=SN2ND 0.1|006|01| 1.17009E+00| 2.5E-03| 2.5E-03|T 1.2E-07| 0.000| 0.752| 9.6E-07| T 0| 0.818| 8| 268| 45| 2| 6.777E-06| 22599| 122| 1| 9| 93|
=SN2ND 0.0|007|01| 1.17009E+00| 1.2E-07| 1.5E-07|T 3.6E-08| 0.376| 0.752| 7.4E-07| T 1| 0.000| 7| 147| 83| 1| 4.879E-06| 12920| 122| 1| 9| 81|
=SN2ND 0.2|008|02| 1.17009E+00| 3.8E-07| 4.0E-07|T 1.1E-08| 0.150| 0.752| 8.9E-09| T 0| 2.703| 17| 548| 6| 7| 4.708E-08| 39760| 116| 1| 9| 189|
=SN2ND 0.0|009|01| 1.17009E+00| 4.4E-09| 2.3E-09|F 0.0E+00| 0.013|-1.963| 8.4E-09| T 1| 0.006| 7| 79| 4| 7| 4.150E-08| 7345| 122| 1| 9| 80|
[SN2ND]...!!!SUCCESS!!! All convergence criteria satisfied at iteration..... 9
[SN2ND]...Final eigenvalue..... 1.17008921
[SN2ND]...Final error on the eigenvalue..... 0.00000000
[SN2ND]...Final iterative error on the flux solution..... 0.00000000
[SN2ND]...Total number of fission source iterations..... 9
[SN2ND]...Total number of upscatter iterations..... 12
[SN2ND]...Total number of scattering source iterations..... 102
[SN2ND]...Within Grp Flx auto adjustment ranges: 1.000E+00 to 1.000E+00 and iteration count of..... 3033
[SN2ND]...Preconditioner auto adjustment ranges: 1.000E-01 to 1.000E+00 and iteration count of..... 218501
[SN2ND]...Total number of synthetic diffusion iterations..... 1144
[SN2ND]...Total computational time required for real solve (seconds)..... 0.865868
[SN2ND]...Finished linking the materials with the mesh.....
[SN2ND]...Obtaining the preconditioner matrices for the second order discrete ordinates method.....

```

```

[SN2ND].....
[SN2ND]...Starting the ADJOINT transport equation solver.....
[SN2ND].....
[SN2ND]...Primary control variables used for execution job size..... 4
[SN2ND]...Debug print setting for setup..... 0
[SN2ND]...Debug print setting for A formation..... 0
[SN2ND]...Debug print setting for iteration history..... 0
[SN2ND]...Global number of energy groups..... 7
[SN2ND]...Global number of spatial mesh points..... 45
[SN2ND]... Corresponding number of finite elements..... 16
[SN2ND]... Number of spatial dimensions..... 2
[SN2ND]...Cubature to use for the angular domain..... LEG-TCHEBY
[SN2ND]...Global angular expansion order..... 6
[SN2ND]... Number of even-parity directions..... 20
[SN2ND]... Expansion order of the scattering kernel..... 0
[SN2ND]... Even-parity spherical harmonics..... 1
[SN2ND]... Odd-parity spherical harmonics..... 0
[SN2ND]...Global space-angle degrees of freedom..... 900
[SN2ND]...Source type information.....
[SN2ND]... Is a fission source present..... Yes
[SN2ND]... Is a fixed source present..... No
[SN2ND]... Is a up/down scattering present..... Yes
[SN2ND]...Outer iteration control variables.....
[SN2ND]... Iterative improvement scaling factor..... 1.000E-01
[SN2ND]... Use Tchebychev acceleration..... Yes
[SN2ND]... Targeted error on the eigenvalue..... 1.000E-07
[SN2ND]... Targeted error on the fission source..... 1.000E-08
[SN2ND]... Targeted error on the MGS flux solution..... 1.000E-07
[SN2ND]... Maximum fission source iterations..... 100
[SN2ND]... Maximum upscatter iterations..... 5
[SN2ND]...Scattering iteration control variables.....
[SN2ND]... Use synthetic diffusion acceleration..... Yes
[SN2ND]... Maximum number of scattering iterations..... 5
[SN2ND]... Maximum space-angle CG iterations..... 10000
[SN2ND]... Maximum PETSc preconditioner iterations..... 1000
[SN2ND]...Break down of total job memory usage (MB)..... PEAK TOTAL
[SN2ND]... Memory usage for spatial mesh..... 0.001900..... 0.007599
[SN2ND]... Memory usage for spatial matrices..... 0.022003..... 0.088013
[SN2ND]... Vector memory usage..... 0.001572..... 0.006287
[SN2ND]... Memory usage for A matrices..... 0.076134..... 0.264355
[SN2ND]... Memory usage for PC matrices..... 0.014420..... 0.050468
[SN2ND]... Memory usage for scattering source..... 0.005127..... 0.020508
[SN2ND]... Memory usage for fission source..... 0.008652..... 0.034607
[SN2ND]... Scratch memory for power iteration..... 0.041199..... 0.164795
[SN2ND]... Total tracked memory usage. (overhead?)..... 0.171005..... 0.636631
[SN2ND]... Total reported memory usage..... 0.041199..... 0.164795
[SN2ND]...Solving the preceding Sn system of equations.....
=#SN2ND Fission Source Iteration | b-Ax | Tchebychev | Within Group Flux | Preconditioner CG | Synth |
=#SN2ND Seconds|Itr|Up| Eigenvalue | Error | Fis Err| Flux Err| Dom | Slope| Error |Active| Dom |Scat| CG |Max/G|Group|Assoc.Error| Total | Max/G |Group| Max/A | Total |
=#SN2ND 0.1|001|01| 1.22808E+00| 5.0E-02| 5.0E-02|T 8.5E-03| 0.000| 0.000| 4.5E-02| T 0| 0.050| 7| 208| 101| 7| 1.166E-01| 13140| 112| 7| 9| 77|
=#SN2ND 0.1|002|01| 1.16402E+00| 5.2E-02| 5.2E-02|T 2.4E-03| 0.220| 0.000| 1.3E-02| T 0| 1.100| 7| 249| 101| 7| 3.663E-02| 18039| 123| 7| 9| 78|
=#SN2ND 0.0|003|01| 1.16468E+00| 5.6E-04| 3.0E-04|T 2.8E-04| 0.127| 0.000| 6.1E-03| T 1| 0.005| 7| 100| 23| 7| 1.704E-02| 6334| 117| 7| 8| 78|
=#SN2ND 0.0|004|01| 1.16761E+00| 2.5E-03| 2.5E-03|T 1.5E-04| 0.497| 0.000| 2.1E-03| T 0| 8.603| 7| 151| 64| 7| 6.384E-03| 10962| 115| 7| 8| 78|
=#SN2ND 0.3|005|02| 1.17211E+00| 3.8E-03| 2.1E-03|T 3.6E-05| 0.440| 2.152| 5.1E-06| T 1| 0.834| 25| 1133| 97| 7| 5.313E-06| 78232| 122| 7| 9| 281|
=#SN2ND 0.0|006|01| 1.16899E+00| 2.7E-03| 1.7E-03|T 9.6E-08| 0.002| 0.984| 1.3E-06| T 0| 0.894| 7| 117| 17| 7| 2.640E-06| 6886| 122| 7| 9| 80|
=#SN2ND 0.1|007|01| 1.17097E+00| 1.7E-03| 9.4E-04|T 1.0E-07| 0.832| 0.984| 5.6E-07| T 1| 0.542| 7| 248| 94| 7| 1.242E-06| 17922| 121| 7| 9| 79|
=#SN2ND 0.0|008|01| 1.16982E+00| 9.9E-04| 7.6E-04|T 1.7E-08| 0.228| 0.984| 2.3E-07| T 0| 0.860| 7| 138| 48| 7| 6.265E-07| 10130| 124| 7| 9| 74|
=#SN2ND 0.0|009|01| 1.17029E+00| 4.1E-04| 2.3E-04|T 1.1E-08| 0.647| -0.303| 1.0E-07| T 1| 0.307| 7| 154| 43| 7| 2.232E-07| 10712| 122| 7| 9| 80|
=#SN2ND 0.1|010|01| 1.17006E+00| 2.0E-04| 1.8E-04|T 3.6E-09| 0.367| -0.303| 3.5E-08| T 0| 0.792| 7| 177| 70| 7| 1.146E-07| 13603| 120| 7| 9| 78|
=#SN2ND 0.1|011|01| 1.17011E+00| 4.7E-05| 2.9E-05|T 1.7E-09| 0.550| -0.303| 2.0E-08| T 1| 0.163| 7| 177| 71| 7| 6.317E-08| 13619| 123| 7| 9| 80|
=#SN2ND 0.1|012|01| 1.17009E+00| 2.0E-05| 1.9E-05|F 0.0E+00| 0.321| -0.873| 8.2E-09| T 0| 0.683| 7| 218| 97| 7| 1.940E-08| 17052| 116| 7| 9| 78|
=#SN2ND 0.0|013|01| 1.17009E+00| 2.4E-06| 1.6E-06|F 0.0E+00| 0.263| -0.873| 6.5E-09| T 1| 0.083| 7| 25| 1| 6| 3.961E-08| 1461| 118| 7| 9| 80|
=#SN2ND 0.0|014|01| 1.17009E+00| 8.0E-07| 8.0E-07|F 0.0E+00| 0.039| -0.873| 5.5E-09| T 0| 0.519| 7| 8| 2| 7| 3.792E-08| 548| 108| 7| 8| 83|
=#SN2ND 0.0|015|01| 1.17009E+00| 2.3E-12| 5.8E-12|F 0.0E+00| 0.273| -1.347| 4.3E-09| T 1| 0.000| 7| 8| 2| 7| 2.052E-08| 545| 108| 7| 8| 81|
[SN2ND]...!!!SUCCESS!!! All convergence criteria satisfied at iteration..... 15
    
```

```

[SN2ND]...Final eigenvalue..... 1.17008921
[SN2ND]...Final error on the eigenvalue..... 0.00000000
[SN2ND]...Final iterative error on the flux solution..... 0.00000000
[SN2ND]...Total number of fission source iterations..... 15
[SN2ND]...Total number of upscatter iterations..... 16
[SN2ND]...Total number of scattering source iterations..... 123
[SN2ND]...Within Grp Flx auto adjustment ranges: 1.000E+00 to 1.000E+00 and iteration count of..... 3111
[SN2ND]...Preconditioner auto adjustment ranges: 1.000E-01 to 1.000E+00 and iteration count of..... 219185
[SN2ND]...Total number of synthetic diffusion iterations..... 1385
[SN2ND]...Total computational time required for real solve (seconds)..... 0.894863
[SN2ND].....
[SN2ND]...Timing history information for total job.....
[SN2ND]...Total cumulative program time..... 1.80
[SN2ND]...Wall clock time.....Date (06/17/2013)...Time (15:22:33)....
[SN2ND]...Wall clock time.....Date (06/17/2013)...Time (15:22:35)....
[SN2ND]...Total cumulative wall clock time..... 2.00
[SN2ND].....
[SN2ND].....Rank 0 snapshot of timing events.....
[SN2ND].....
[SN2ND]...| Id | Event Name | Measurements | Time (seconds) |
[SN2ND]...| | | | Ignore | Total | Avg/Call | Total | Percent |
[SN2ND]...| 1| Reading and Transmitting the Input | 0.00 | 3 | 0.0023 | 0.0070 | 0.39 |
[SN2ND]...| 2| Group-Angle-Space Segmentation | 0.00 | 0 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 3| METIS decomposition of the mesh | 0.00 | 0 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 4| Cross Section Homogenization | 0.00 | 0 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 5| Cross Section Communication | 0.00 | 16 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 6| UNIC native HDF5 export option | 0.00 | 2 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 7| SN2ND input checking and PETSc setup | 0.00 | 1 | 0.0010 | 0.0010 | 0.06 |
[SN2ND]...| 8| SN2ND stenciling of the A matrix | 0.00 | 1 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 9| SN2ND obtaining the spatial matrices | 0.00 | 1 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 10| SN2ND treatment of the boundary conditions | 0.00 | 1 | 0.0200 | 0.0200 | 1.11 |
[SN2ND]...| 11| SN2ND computation of the A matrices | 0.00 | 2 | 0.0060 | 0.0120 | 0.66 |
[SN2ND]...| 12| SN2ND computation of the PC matrices | 0.00 | 2 | 0.0010 | 0.0020 | 0.11 |
[SN2ND]...| 13| SN2ND flux initialization | 0.00 | 0 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 14| SN2ND solve of the forward transport equation | 0.00 | 2 | 0.8804 | 1.7607 | 97.56 |
[SN2ND]...| 15| SN2ND fission source iteration | 0.00 | 26 | 0.0677 | 1.7597 | 97.51 |
[SN2ND]...| 16| SN2ND scattering source operations | 0.00 | 374 | 0.0000 | 0.0020 | 0.11 |
[SN2ND]...| 17| SN2ND fission source operations | 0.00 | 213 | 0.0000 | 0.0030 | 0.17 |
[SN2ND]...| 18| SN2ND within-group flux solution | 0.00 | 187 | 0.0094 | 1.7537 | 97.17 |
[SN2ND]...| 19| SN2ND within group scattering operations | 0.00 | 412 | 0.0000 | 0.0010 | 0.06 |
[SN2ND]...| 20| SN2ND within group odd flux updates | 0.00 | 225 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 21| SN2ND top level CG solve operation | 0.00 | 225 | 0.0076 | 1.7067 | 94.57 |
[SN2ND]...| 22| SN2ND A*x operations | 0.00 | 6594 | 0.0001 | 0.4799 | 26.59 |
[SN2ND]...| 23| SN2ND diagonal block CG solve operation (PETSc) | 0.00 | 6203 | 0.0002 | 1.0948 | 60.66 |
[SN2ND]...| 24| SN2ND P-multigrid prolongation | 0.00 | 0 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 25| SN2ND P-multigrid restriction | 0.00 | 0 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 26| SN2ND P-multigrid smoothing | 0.00 | 0 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 27| SN2ND synthetic diffusion solve (PETSc) | 0.00 | 225 | 0.0001 | 0.0170 | 0.94 |
[SN2ND]...| 28| SN2ND DSA P-multigrid prolongation | 0.00 | 0 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 29| SN2ND DSA P-multigrid restriction | 0.00 | 0 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 30| SN2ND DSA P-multigrid smoothing | 0.00 | 0 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 31| SN2ND Communicate the angular flux | 0.00 | 6390 | 0.0000 | 0.0510 | 2.83 |
[SN2ND]...| 32| SN2ND global reduce of the PN flux | 0.00 | 608 | 0.0000 | 0.0020 | 0.11 |
[SN2ND]...| 33| SN2ND SN to PN flux conversion | 0.00 | 1432 | 0.0000 | 0.0000 | 0.00 |
[SN2ND].....
[SN2ND].....Parallel performance timing events.....
[SN2ND].....
[SN2ND]...| Id | Event Name | Measurements | Time (seconds) |
[SN2ND]...| | | | Max/Min | Avg/Proc | Max/Min | Avg/Proc | Percent |
[SN2ND]...| 1| Reading and Transmitting the Input | 1.00 | 3 | 11.9990 | 0.0060 | 0.33 |
[SN2ND]...| 2| Group-Angle-Space Segmentation | 0.00 | 0 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 3| METIS decomposition of the mesh | 0.00 | 0 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 4| Cross Section Homogenization | 0.00 | 0 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 5| Cross Section Communication | 0.00 | 4 | 0.0000 | 0.0000 | 0.00 |
[SN2ND]...| 6| UNIC native HDF5 export option | 1.00 | 2 | 0.0000 | 0.0000 | 0.00 |

```

[SN2ND]...	7 SN2ND input checking and PETSc setup		1.00	1	1.0000	0.0010	0.06
[SN2ND]...	8 SN2ND stenciling of the A matrix		1.00	1	0.0000	0.0000	0.00
[SN2ND]...	9 SN2ND obtaining the spatial matrices		1.00	1	0.0000	0.0005	0.03
[SN2ND]...	10 SN2ND treatment of the boundary conditions		1.00	1	1.0501	0.0202	1.12
[SN2ND]...	11 SN2ND computation of the A matrices		1.00	2	1.4999	0.0095	0.53
[SN2ND]...	12 SN2ND computation of the PC matrices		1.00	2	0.0000	0.0007	0.04
[SN2ND]...	13 SN2ND flux initialization		0.00	0	0.0000	0.0000	0.00
[SN2ND]...	14 SN2ND solve of the forward transport equation		1.00	2	1.0011	1.7622	97.65
[SN2ND]...	15 SN2ND fission source iteration		1.00	26	1.0017	1.7620	97.63
[SN2ND]...	16 SN2ND scattering source operations		1.00	374	1.9999	0.0013	0.07
[SN2ND]...	17 SN2ND fission source operations		1.00	213	6.9991	0.0030	0.17
[SN2ND]...	18 SN2ND within-group flux solution		1.00	187	1.0029	1.7562	97.31
[SN2ND]...	19 SN2ND within group scattering operations		1.00	412	0.0000	0.0003	0.01
[SN2ND]...	20 SN2ND within group odd flux updates		1.00	225	0.0000	0.0003	0.01
[SN2ND]...	21 SN2ND top level CG solve operation		1.00	225	1.0118	1.7027	94.35
[SN2ND]...	22 SN2ND A*x operations		1.00	6594	1.0738	0.4637	25.69
[SN2ND]...	23 SN2ND diagonal block CG solve operation (PETSc)		1.00	6203	1.0799	1.0461	57.96
[SN2ND]...	24 SN2ND P-multigrid prolongation		0.00	0	0.0000	0.0000	0.00
[SN2ND]...	25 SN2ND P-multigrid restriction		0.00	0	0.0000	0.0000	0.00
[SN2ND]...	26 SN2ND P-multigrid smoothing		0.00	0	0.0000	0.0000	0.00
[SN2ND]...	27 SN2ND synthetic diffusion solve (PETSc)		0.00	56	0.0000	0.0042	0.24
[SN2ND]...	28 SN2ND DSA P-multigrid prolongation		0.00	0	0.0000	0.0000	0.00
[SN2ND]...	29 SN2ND DSA P-multigrid restriction		0.00	0	0.0000	0.0000	0.00
[SN2ND]...	30 SN2ND DSA P-multigrid smoothing		0.00	0	0.0000	0.0000	0.00
[SN2ND]...	31 SN2ND Communicate the angular flux		1.00	6390	1.7805	0.0575	3.19
[SN2ND]...	32 SN2ND global reduce of the PN flux		1.00	608	4.0011	0.0057	0.32
[SN2ND]...	33 SN2ND SN to PN flux conversion		1.00	1432	0.0000	0.0018	0.10
[SN2ND].....							

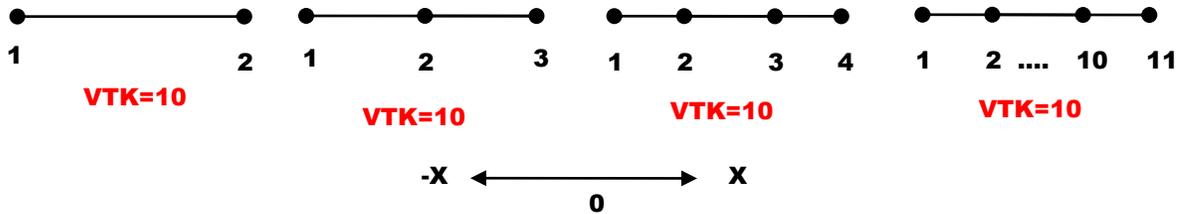
Appendix G. Finite Elements Used in PROTEUS-SN Along with Vertex and Surface Numbering

Red indicates elements that are already in PROTEUS.

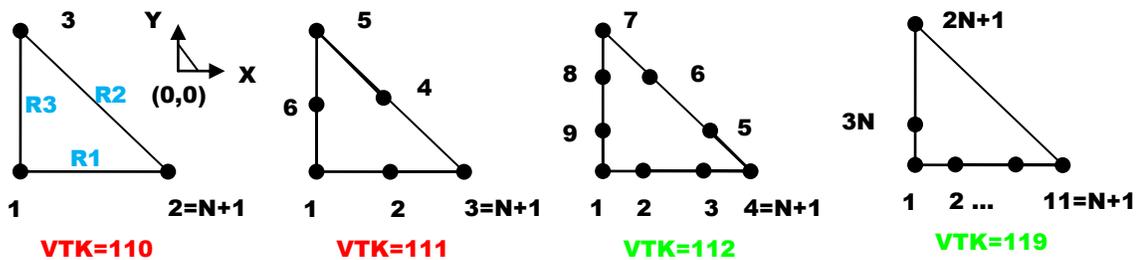
Blue indicates elements that are to be added to PROTEUS in the future.

Green indicates elements that we may implement into PROTEUS at some point.

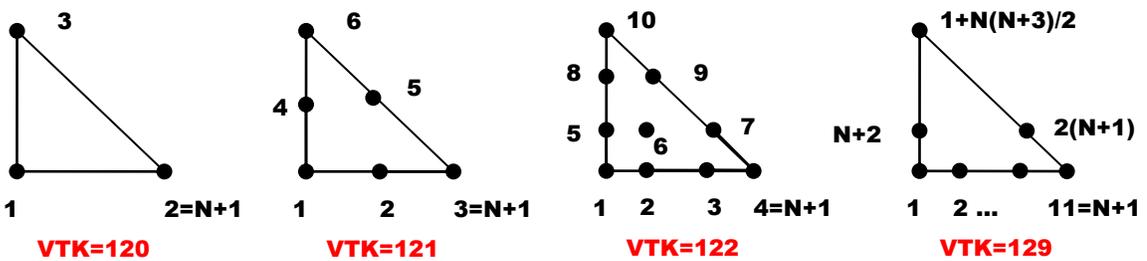
Cyan indicates the reference surface numbering for each element.



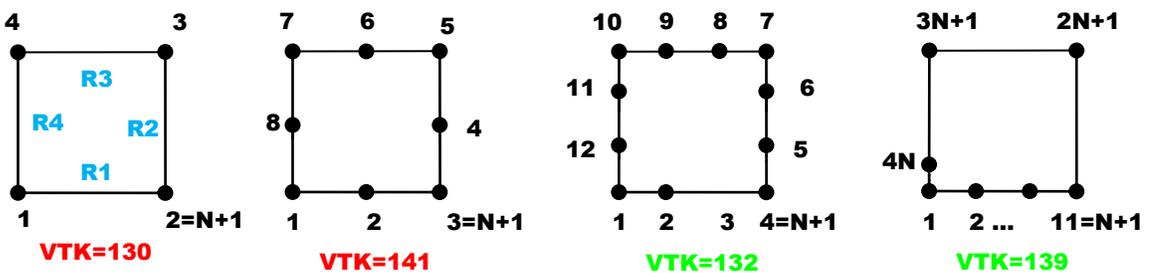
PROTEUS 1-D Finite Element Vertex Layout and VTK Numbering



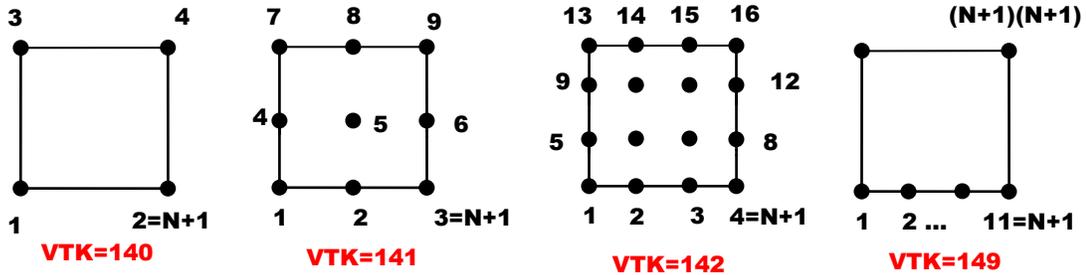
PROTEUS 2-D Triangular Serendipity Finite Element Vertex Layout and VTK Numbering



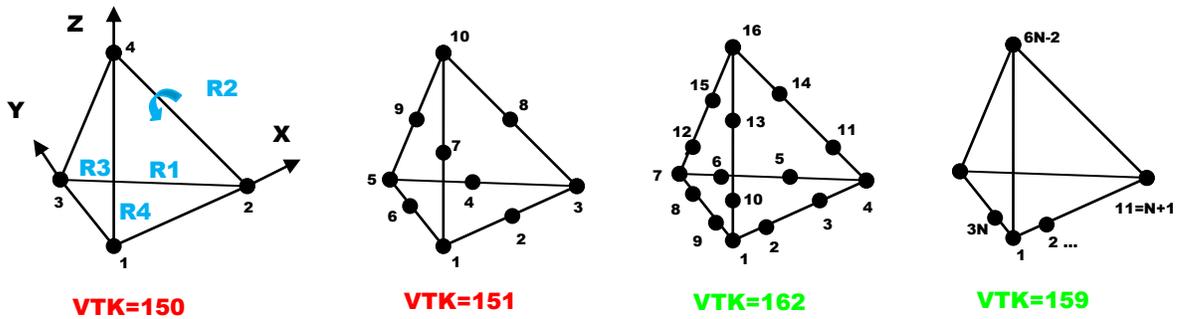
PROTEUS 2-D Triangular LaGrange Finite Element Vertex Layout and VTK Numbering



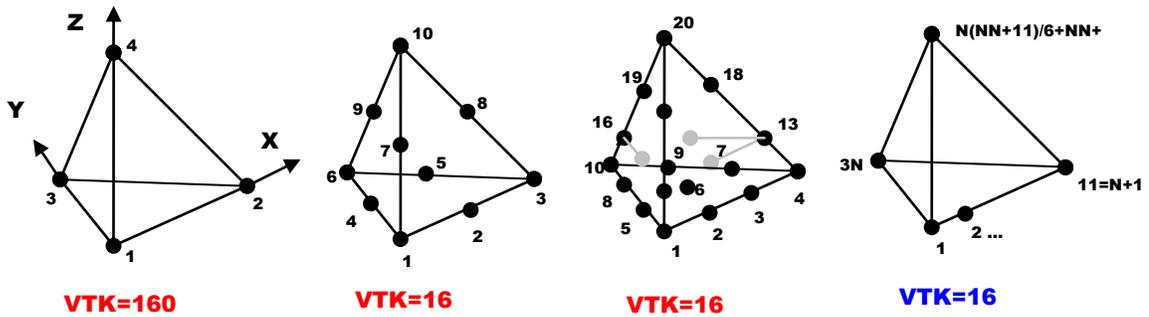
PROTEUS 2-D Quadrilateral Serendipity Finite Element Vertex Layout and VTK Numbering



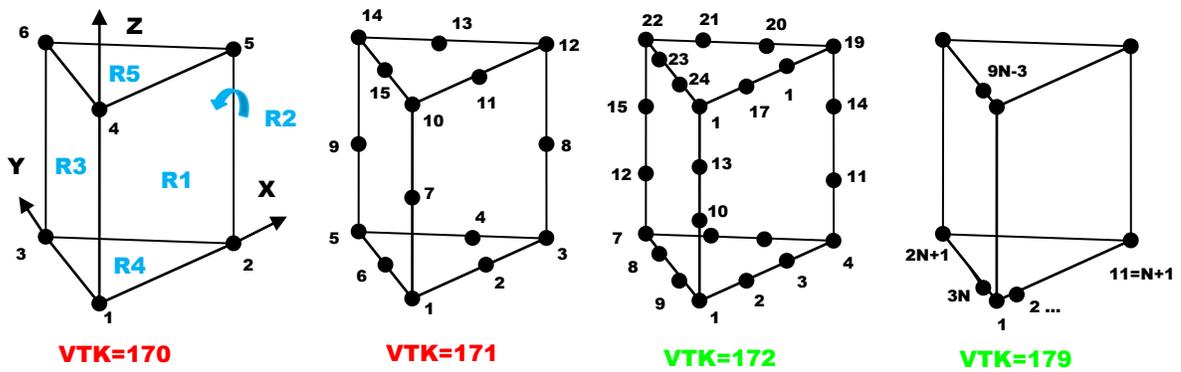
PROTEUS 2-D Quadrilateral LaGrange Finite Element Vertex Layout and VTK Numbering



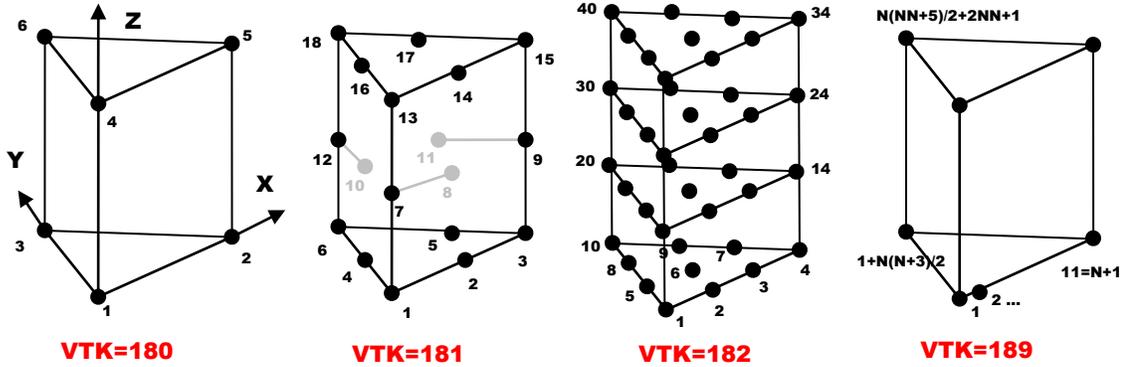
PROTEUS 3-D Tetrahedral Serendipity Finite Element Vertex Layout and VTK Numbering



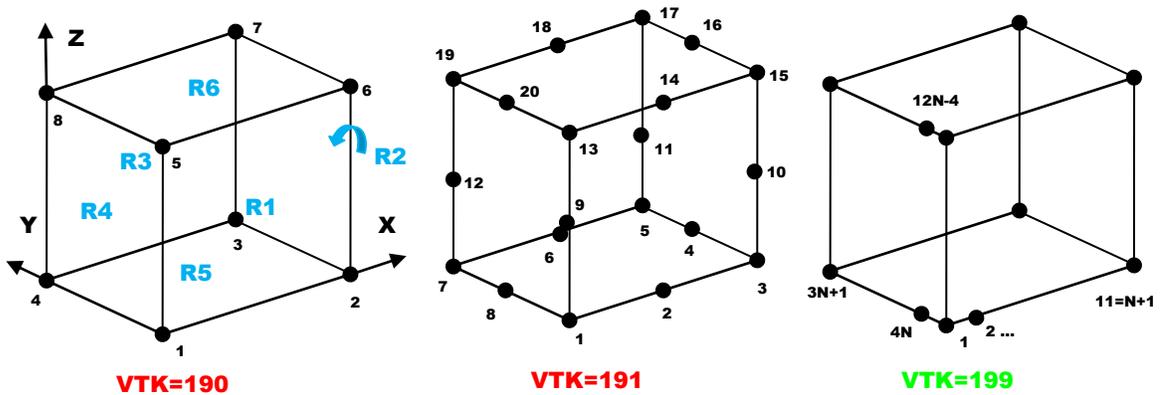
PROTEUS 3-D Tetrahedral LaGrange Finite Element Vertex Layout and VTK Numbering



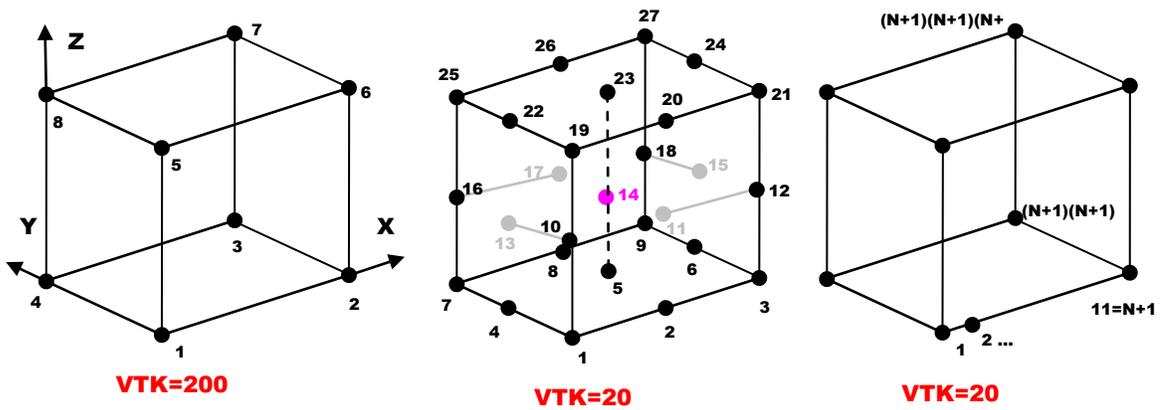
PROTEUS 3-D Wedge Serendipity Finite Element Vertex Layout and VTK Numbering



PROTEUS 3-D Wedge LaGrange Finite Element Vertex Layout and VTK Numbering



PROTEUS 3-D Hexahedral Serendipity Finite Element Vertex Layout and VTK Numbering



PROTEUS 3-D Hexahedral LaGrange Finite Element Vertex Layout and VTK Numbering



Nuclear 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