SANDIA REPORT
SAND20XX-XXXX
Printed Click to enter a date

Sandia National Laboratories

SCEPTRE 2.3 Quick Start Guide

Donald E. Bruss, Clifton R. Drumm, Wesley C. Fan, and Shawn D. Pautz

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico
87185 and Livermore,
California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods/

ABSTRACT

This report provides a summary of notes for building and running the Sandia Computational Engine for Particle Transport for Radiation Effects (SCEPTRE) code. SCEPTRE is a general-purpose C++ code for solving the linear Boltzmann transport equation in serial or parallel using unstructured spatial finite elements, multigroup energy treatment, and a variety of angular treatments including discrete ordinates (Sn) and spherical harmonics (Pn). Either the first-order form of the Boltzmann equation or one of the second-order forms may be solved. SCEPTRE requires a small number of open-source Third Party Libraries (TPL) to be available, and example scripts for building these TPL are provided. The TPL needed by SCEPTRE are Trilinos, Boost, and Netcdf. SCEPTRE uses an autotools build system, and a sample configure script is provided. Running the SCEPTRE code requires that the user provide a spatial finite-elements mesh in Exodus format and a cross section library in a format that will be described. SCEPTRE uses an xml-based input, and several examples will be provided.

CONTENTS

LIST OF FIGURES

This page left blank

ACRONYMS AND DEFINITIONS

Abbreviation	Definition
SCEPTRE	Sandia computational engine for particle transport for radiation effects
TPL	Third-party libraries
DFE	Discontinuous finite elements
CFE	Continuous finite elements
SPD	Symmetric positive definite
CG	Conjugate gradients
CSD	Continuous slowing down
Sn	Discrete ordinates
Pn	Spherical harmonics
MPI	Message passing interface
QoI	Quantities of interest

1. INTRODUCTION

The Sandia Computational Engine for Particle Transport for Radiation Effects (SCEPTRE) (Pautz, Bohnhoff, Drumm, & Fan, 2009) (Pautz, Drumm, Bohnhoff, & Fan, 2009) is a general-purpose C++ code for solving the linear Boltzmann transport equation in serial or parallel using unstructured spatial finite elements, multigroup energy treatment, and a variety of angular treatments including discrete ordinates and spherical harmonics (Drumm, 2015). SCEPTRE also contains some capability for phase-space finite elements (angle and energy) (Drumm, Fan and Pautz, 2013), which should be considered experimental in this release. This capability will be further productized in future releases. The SCEPTRE code remains under active development, containing some well-tested production capability and some newer, more experimental capability. SCEPTRE has several unique features, partially motivated by the application space for which the code was developed, providing for the transport of both neutral and charged particles (photon/electron/positron).

SCEPTRE includes capability for solving the Boltzmann equation using many different numerical and iterative methods and allows for a different transport solver to be used for each energy group, enabling the user to apply the most appropriate methods for accuracy and efficiency for each energy group/particle type in the problem. Either the first-order form of the Boltzmann equation or one of the second-order forms of the Boltzmann equation may be solved (Lewis & Miller, 1984) (Duderstadt & Martin, 1979) (Bell & Glasstone, 1970):

1. SCEPTRE provides a wave front sweeping algorithm for the first-order form of the transport equation using Discontinuous Finite Elements (DFE) (Wareing, McGhee, Morel, & Pautz, 2001). In the wave front sweeps-based solver, the entire source term including the self-scatter source is on the right-hand side of the equation, the solution for each particle direction is determined independently, and the scattering source term is updated until convergence.

2. In addition to the sweeps-based solver, SCEPTRE has a class of algorithms based on an entirely different solution approach with very different iterative convergence properties (Drumm & Lorenz, 1999). With the alternative algorithm, the self-scatter source term is included with the streaming and removal operators on the left-hand side of the equation and the spatial and angular dependences of the solution are solved simultaneously for each energy group. The main drawback of this method is that the memory requirement may be large, but if enough memory is available, this method is a useful alternative to the sweeps solver that generally converges more efficiently for charged particles. In this approach, the linear system is constructed and handed off to Trilinos (Heroux & Willenbring, 2003) for solution using one of the Krylov iterative methods available in the Trilinos package.

SCEPTRE has the option of using either Discontinuous Finite Elements (DFE) or Continuous Finite Elements (CFE). Use of DFE tends to be more accurate for certain transport problems but is also more expensive. Under certain conditions the use of CFE results in a Symmetric Positive Definite (SPD) linear system that may be solved using the highly efficient Conjugate Gradients (CG) algorithm.

SCEPTRE requires a small number of open-source Third Party Libraries (TPL) to be available, and example scripts for building these TPL are provided. The TPL needed by SCEPTRE are Trilinos, Boost, and Netcdf. SCEPTRE uses an autotools build system, and a sample configure script is provided. Running the SCEPTRE code requires that the user provide a spatial finite-elements mesh in Exodus format and a cross section library in a format that will be described. SCEPTRE uses an xml-based input, and several examples will be provided.

1.1. New features in Version 2.3

TPL versions have been updated to Boost 1.76.0, Netcdf 4.8.0 and Trilinos 13.0.1.

A Boltzmann-CSD (Continuous Slowing Down) solver has been implemented that will enable more-accurate electron transport calculations to be performed. An adjoint Boltzmann-CSD capability has also been implemented, which is needed for performing sensitivity analyses. This capability requires compatible stopping powers and cross sections computed with the CEPXS code, using the csdld keyword.

Several components have been modified for use with full space/angle/energy finite elements solvers. The field metrics have been upgraded for use with angular finite elements fields, discrete ordinates fields, and angular moments fields with a single interface. An angular Jacobian matrix capability has been developed for use with angular finite elements, discrete ordinates, and angular moments, replacing the previous temporary use of the spatial Jacobian for angle space. The angular flux field containers (containers for the primary unknown in the radiation transport solution) have been generalized for use with either Sn or angular FE. The capability for mapping between the angular scattering moments and angular space has been generalized for use with either Sn or angular FE. The FE matrices for space, angle and energy are being redesigned for consistent use with the sweeps-based solver and the Trilinos-based solvers. These developments will enable code reuse for discrete ordinates and angular FE modeling and for reuse of components in the various solvers available in SCEPTRE. An angular FE sweeps-based solver test has been added.

We have made usability improvements in radlib and radTools. For example, the SCEPTRE input file can be decomposed into multiple files for enhanced reuse of material specifications and spectra. Other parameters can be automated versus requiring user specification, such as the number of energy groups.

Dependencies on Epetra/AztecOO have been removed and replaced with Tpetra/Belos in anticipation of eventual removal of Epetra/AztecOO support from Trilinos.

A number of updates to the RAPTURE package are included in Release 2.3. The format in which the spectral Quantities of Interest (QoI) are reported has been updated to improve the usefulness and readability of the output file. The algorithm that checks the convergence of reaction-rate QoI has been improved, so that moments of the QoI are only checked in regions of interest rather than across all regions. Several improvements have been made to the RAPTURE output for improved clarity, including modifications of the naming of the directories generated by RAPTURE. Additional checks of the user input have been included to prevent non-physical material densities.

2. OBTAINING AND INSTALLING TPL

All of the TPL needed to build and run SCEPTRE are freely available and may be installed without modification. The purpose of this document is not to provide detailed installation instructions for the TPL, but to provide minimal instructions for building each of them, referring the reader to the specific TPL support for help.

Consistency between the TPL and SCEPTRE is essential. Once a compiler and MPI implementation have been selected, ensure that these are specified consistently in all the TPL and SCEPTRE configuration scripts. Several additional files (such as the LAPACK and BLAS libraries) must also be specified; these must also be specified consistently. It may be helpful to modify the configure scripts to first purge all loaded modules and then only load modules which will be used in the compilation process.

Although some TPL include example configuration scripts, the scripts provided in this guide should be used to compile these libraries to ensure compatibility with SCEPTRE.

2.1. Boost

The Boost software may be obtained from boost.org. SCEPTRE uses only header files from Boost, so the Boost tarball merely needs to be unzipped in an appropriate location. SCEPTRE 2.3 has been successfully built with Version 1.76.0 of Boost.

2.2. Netcdf

The Netcdf software may be obtained from unidata.ucar.edu/software/netcdf. Netcdf may be used without modification for versions later than 4.5.0. These are some brief notes on building and installing Netcdf.

First download and untar Netcdf into some directory, which we will call NETCDFDIR.

To build Netcdf first create a target directory (NETCDF_TARGET_DIR) and an installation directory (NETCDF_INSTALL_DIR). Then do the following:

```
> cd $NETCDF_TARGET_DIR
> $NETCDFDIR/configure --prefix=$NETCDF_INSTALL_DIR CC=$CCOMPILER
CXX=$CXXCOMPILER FC= F90= --disable-netcdf-4 --disable-shared --disable-dap
```

Note that you need to specify the compilers explicitly, or the Netcdf build system will try to use whatever compilers it thinks are "vendor" ones - even on Linux it doesn't pick up gcc.

Next build, test if desired, and install:

```
> make
> make check
> make install
```

The above has been successfully tested on several systems for Netcdf version 4.7.4 with gcc and Intel.

2.3. Trilinos

The Trilinos software may be obtained from https://github.com/trilinos/trilinos.github.io. Trilinos uses a cmake build system, and a sample script is shown here. The user needs to modify this script to provide the locations of MPI, LAPACK, and BLAS libraries, and the locations of the Boost and Netcdf installations. SCEPTRE 2.3 has been successfully built and tested with Trilinos version 13.0.1. A sample cmake script for building Trilinos with options needed by SCEPTRE is shown here:

```
EXTRA_ARGS=$@

module purge
module load gnu/8.2.1
module load openmpi-gnu/4.0
module load mkl/19.0
module load cmake/3.12.2

export CODE_PATH=/projects/sceptre/CTS1
export COMPILER_VERS=gcc-8.2.1
export TRILINOS_PATH=$CODE_PATH/trilinos/13.0.1/$COMPILER_VERS-openmpi_install
export INTEL_LIB=/opt/intel/19.0/compiler/lib/intel64

cmake \
  -D Trilinos_VERBOSE_CONFIGURE:BOOL=ON \
  -D CMAKE_VERBOSE_MAKEFILE:BOOL=TRUE \
  -D CMAKE_BUILD_TYPE:STRING=RELEASE \
  -D TPL_ENABLE_BinUtils:BOOL=OFF \
  -D TPL_ENABLE_MPI:BOOL=ON \
  -D MPI_BASE_DIR:PATH=$MPIHOME \
  -D TPL_ENABLE_BLAS:BOOL=ON \
  -D TPL_ENABLE_LAPACK:BOOL=ON \
  -D LAPACK_LIBRARY_NAMES:STRING="mkl_core;mkl_sequential;mkl_intel_lp64" \
  -D LAPACK_LIBRARY_DIRS:PATH=$MKL_LIBS \
  -D BLAS_LIBRARY_NAMES:STRING="mkl_core;mkl_intel_lp64;mkl_intel_thread;iomp5" \
  -D BLAS_LIBRARY_DIRS:PATH="$MKL_LIBS;$INTEL_LIB" \
  -D TPL_ENABLE_Boost:BOOL=ON \
  -D Boost_INCLUDE_DIRS:PATH=$CODE_PATH/boost/1.76.0/boost_1_76_0 \
  -D TPL_ENABLE_Netcdf:BOOL=ON \
  -D Netcdf_LIBRARY_DIRS:PATH=$CODE_PATH/netcdf/4.8.0/$COMPILER_VERS\_install/lib \
  -D Netcdf_INCLUDE_DIRS:PATH=$CODE_PATH/netcdf/4.8.0/$COMPILER_VERS\_install/include \
```

```
-D TPL_ENABLE_Matio:BOOL=OFF \
-D TPL_ENABLE_X11:BOOL=OFF \
-D Trilinos_ENABLE_ALL_OPTIONAL_PACKAGES:BOOL=OFF \
-D Trilinos_ENABLE_Amesos:BOOL=ON \
-D Trilinos_ENABLE_Amesos2:BOOL=ON \
-D Trilinos_ENABLE_AztecOO:BOOL=ON \
-D Trilinos_ENABLE_Belos:BOOL=ON \
-D Trilinos_ENABLE_Epetra:BOOL=ON \
-D Trilinos_ENABLE_EpetraExt:BOOL=ON \
-D Trilinos_ENABLE_Tpetra:BOOL=ON \
-D Trilinos_ENABLE_Xpetra:BOOL=ON \
-D Trilinos_ENABLE_Thyra:BOOL=ON \
-D Trilinos_ENABLE_Kokkos:BOOL=ON \
-D Trilinos_ENABLE_Ifpack:BOOL=ON \
-D Trilinos_ENABLE_Ifpack2:BOOL=ON \
-D Trilinos_ENABLE_ML:BOOL=ON \
-D Trilinos_ENABLE_MueLu:BOOL=ON \
-D Trilinos_ENABLE_MueLu_ENABLE_Amesos2::BOOL=ON \
-D Trilinos_ENABLE_ROL:BOOL=ON \
-D Trilinos_ENABLE_SEACAS:BOOL=ON \
-D SEACASExodus_ENABLE_MPI:BOOL=OFF \
-D Trilinos_ENABLE_Teko:Bool=ON \
-D Trilinos_ENABLE_Teuchos:BOOL=ON \
-D Teuchos_ENABLE_COMPLEX:BOOL=OFF \
-D Trilinos_ENABLE_Triutils:BOOL=ON \
-D Trilinos_ENABLE_Zoltan:BOOL=ON \
-D Trilinos_ENABLE_Zoltan2:BOOL=ON \
-D Trilinos_ENABLE_TESTS:BOOL=OFF \
-D Trilinos_ENABLE_DEBUG:BOOL=OFF \
-D Trilinos_ENABLE_EXPLICIT_INSTANTIATION:BOOL=ON \
-D Trilinos_ENABLE_STKTransfer:Bool=ON \
-D Trilinos_ENABLE_STKSearch:Bool=ON \
-D Trilinos_ENABLE_STKUtil:Bool=ON \
-D CMAKE_INSTALL_PREFIX:PATH=$TRILINOS_PATH \
$EXTRA_ARGS \
../Trilinos-trilinos-release-13-0-1
```

3. INSTALLING AND TESTING SCEPTRE

Release 2.3 consists of a single tarball, including all 10 packages in the SCEPTRE family. The packages included are:

1. radlib (foundation SCEPTRE radiation transport functionality)
2. radTools (pre- and post-processing tools, including source- and simple mesh-generation tools)
3. radlibTests (regression tests and testing tools for SCEPTRE)
4. radSens (sensitivity analysis tools)
5. radlibEM (experimental code for modeling charged particle transport in the presence of electric and magnetic fields)
6. radEMTools (post-processing tools specific for cable and box SGEMP applications)
7. radNuGET (experimental code for handoff of SCEPTRE angular flux results to the NuGET code)
8. radStochastic (research code for stochastic media)
9. radThermal (specialized executables for coupling with thermal codes)
10. rapture (a user-friendly interface to 1-D SCEPTRE)

The building and installation of radTools will be covered in the next section, and the building and installation of radlib will be covered in this section.

Radlib may be compiled with either partial or full unit testing and various levels of optimization. The compilation of optimized code is significantly slower than unoptimized code.

A successful build and installation of radlib will create three executables, 1) sceptre, 2) sceptre-1gFO, which is test code for simple one energy group first order solves, and 3) tds which is experimental code for time-dependent transport calculations.

radlib uses an autotools build system, and a sample configuration script is provided here:

```
../configure \
--prefix=/apps/radlib-2.3/$COMPILER_VERS-openmpi_install \
--disable-debug \
--with-unit-testing=full \
--with-opt=2 \
--with-cxx=mpicxx \
--with-cxxflags=-std=c++14 \
--with-cc=mpicc \
--with-mpi-type=openMPI \
--with-mpi-basedir=$MPI_HOME \
--with-mpirun-command=mpiexec \
--with-mpi-procs-flag=-np \
--with-mpi-tmpdir=/tmp \
--with-boost-incdir=$TPL_PATH/boost/1.76.0/boost_1_76_0 \
--with-exodus-basedir=$TRILINOS_PATH \
--with-nemesis-basedir=$TRILINOS_PATH \
--with-netcdf-basedir=$TPL_PATH/netcdf/4.8.0/$COMPILER_VERS\_install \
--with-trilinos-basedir=$TRILINOS_PATH \
--with-lapack-basedir=$MKL_LIBS \
--with-lapack-libs=mkl_sequential \
--with-blas-basedir=$MKL_LIBS:$INTEL_LIB \
--with-blas-libs=mkl_core:mkl_intel_lp64:mkl_intel_thread:iomp5:dl \
--with-shared-library-path=/usr/lib64:$INTEL_LIB
```

This configuration script should be modified to provide paths to the MPI installation, TPL locations, LAPACK and BLAS libraries, and compiler parameters. The following options may be modified as described below:

```
--prefix=/apps/radlib-2.3/$COMPILER_VERS-openmpi_install! location where SCEPTRE is to be
installed
--disable-debug                                        ! either enable-debug or disable-debug
--with-unit-testing=partial                            ! either partial or full testing suite
--with-opt=2                                           ! optimization level
--with-mpi-type=openMPI                                ! either mpich, mpich2 or openMPI
```

After successfully executing a configure command, radlib is built and tested using gmake (for a 12-core compilation):

```
gmake -j12 install
gmake -j12 check
```

In order to fully test the installation, `gmake check` should be used with a debug version of the code, since the test code contains many `assert` statements that are checked with a debug version.

4. INSTALLING AND TESTING RADTOOLS

The radTools package contains several useful utilities for pre- and post-processing, and for cross section reformatting. The building and installation of radTools is very similar to that for radlib. The

configure line to configure radTools is almost the same as that for radlib, with the following differences. The location of the installation directory is radTools rather than radlib

```
--prefix=/apps/radTools-2.3/$COMPILER_VERS-openmpi_install \
```

and an additional line included, which contains the location of the radlib installation directory.

```
--with-radlib-basedir=/apps/radlib-2.3/$COMPILER_VERS-openmpi_install \
```

radTools is then built and tested by executing:

```
gmake -j12 install
gmake -j12 check
```

5. RUNNING SCEPTRE

Running SCEPTRE requires three input items: 1) a mesh file in Exodus format (Schoof & Yarberry, 1994), 2) a cross section file, and 3) an xml input file containing parameters needed to run SCEPTRE. The Exodus mesh file may be generated either by using (CUBIT, 2019) or a commercial Ansys product (ANSYS ICEM CFD). It is not the purpose of this document to provide detailed guidance in generating a mesh, but some information on preparing an Exodus mesh for a parallel run will be included in this document. Also included will be some information on generating a cross section file for SCEPTRE.

The driver/test folder contains a number of test problems, some of which will be described in detail here. The examples are included merely to illustrate how to set up and run SCEPTRE and are not intended to contain parameters realistic for a particular application. In general, more energy groups, a higher angular-quadrature order, and more-refined spatial mesh will be needed to improve the accuracy of results, but the examples included are intended to be fast-running and illustrate most of the available features. These files may be used as a starting point and modified for a user's particular application. Included in the driver/test folder are:

1. rg402.xml is a test problem of a simple coaxial cable, demonstrating many SCEPTRE options and features
2. several examples of serial and parallel runs
3. forward and adjoint example problems
4. tests illustrating the use of Exodus format and binary format angular flux storage
5. several first-collision source (FCS) tests

In order to run one of the sample input files, the run command is as follows, for a 4-processor parallel run:

```
mpirun -np 4 $SCEPTRE_BIN/sceptre rg402.xml
```

Where `$SCEPTRE_BIN` is the file containing the radlib binaries and rg402.xml is the input file.

The following subsection describes how to generate a coupled photon/electron cross section file for use in SCEPTRE. The subsequent subsection then contains information for creating a parallel mesh file for use in SCEPTRE. It is assumed that the user has created an Exodus format mesh file using one of the available meshing tools. The subsection following, then, contains details of the many options and features available in running SCEPTRE from the xml input file. A complete sample xml input files for performing a SCEPTRE calculation is included in Appendix I and in the test folder under the driver folder in the SCEPTRE source distribution. The individual sections of the input file will then be described in detail.

5.1. Generating a cross section library

Cross sections for SCEPTRE are typically generated by the CEPXS code, which is distributed with the ITS code package (ITS/CEPXS RSICC), but SCEPTRE may be run with other multi-group Legendre cross section sets, provided the cross sections are reformatted into a form that SCEPTRE can read.

The first step in generating a cross section library for SCEPTRE is to run the CEPXS code, with the following example lines of code copied to the `cepinp` file. A * at the beginning of a line in the input file signifies a comment line, and the sample input file includes several options for handling the CSD term (Franke, 2008). E.g. for using the Boltzmann-CSD solver, the `csdld` keyword should be uncommented and the `second-order` keyword should be commented out. The `second-order` keyword is appropriate for most typical applications.

```
title
 5 photon 5 electron group test xsec
*first-order
*csda
*bfp
*csdld
second-order
energy 0.1
cutoff 0.001
legendre 8
no-lines
photon-source
 partial-coupling
photons
 linear 5
electrons
 linear 5
material fe
material cu
material ag
material c 0.2402  f 0.7598
  density 2.2
print
 leg 0
print-all
```

Running CEPXS generates a bxslib binary-format multi-group Legendre cross section library. This bxslib file is converted to Netcdf format for use by SCEPTRE by using the convertBxslibToNetcdf utility, which is located in the radTools installation /bin directory.

```
convertBxslibToNetcdf bxslib
```

This creates a netcdf file named bxslib.ncd, which can be renamed to something more descriptive

```
mv bxslib.ncd 5photon5electronGroups.xslib
```

which is then ready for SCEPTRE. The file may be converted to a readable format by using the ncdump utility, which is contained in the Netcdf distribution.

```
ncdump 5photon5electronGroups.xslib > 5photon5electronGroups.asci
```

5.2. Creating a parallel mesh file

Running the CUBIT or ANSYS/ICEM CFD software to create a mesh file is outside of the scope of this document, but creating a parallel mesh file from a serial mesh file is briefly described here. An Exodus file is partitioned by executing the `nem_slice` and `nem_spread` executables, which are included with the Trilinos distribution and documented here SEACAS documentation

The following assumes that a subdirectory named "1" is included in the directory where the mesh files are located. After creation of a mesh file called, for example, `testMeshTri3.gen`, the first step is to execute nem_slice

```
nem_slice -e -l multikl -o testMeshTri3.nem -m mesh=4 testMeshTri3.gen
```

where the `mesh` keyword specifies the number of files to split the mesh file into.

Then the following lines of code are copied into a file named `nem_spread.inp`

```
Input FEM file                  = testMeshTri3.gen
LB file                         = testMeshTri3.nem
Parallel Disk Info= number=1
Parallel file location = root=./, subdir=.
```

And then the `nem_spread` executable is run

```
nem_spread
```

This will write the partitioned mesh files into the "1" subdirectory.

5.3. Simple parallel test problem

Running a test problem in parallel is straightforward. The name of the mesh file in the xml file is modified to be the prefix of the parallel files existing in the radlib distribution.

```
  <Mesh_File>/apps/radlib-2.3/src/driver/test/testMeshTri3.par</Mesh_File>
```

A four-processor run expects to have four mesh files available:

```
/apps/radlib-2.3/src/driver/test/testMeshTri3.par.4.0
/apps/radlib-2.3/src/driver/test/testMeshTri3.par.4.1
/apps/radlib-2.3/src/driver/test/testMeshTri3.par.4.2
/apps/radlib-2.3/src/driver/test/testMeshTri3.par.4.3
```

Then the run is executed with four processors by

```
mpirun -np 4 $SCEPTRE_BIN/sceptre simpleParallelTest.xml
```

5.4. Detailed description of the SCEPTRE input

A complete xml input file for running a SCEPTRE calculation is included in Appendix I and in the test section under the driver folder in the source code. Individual input options for running SCEPTRE are described in detail in this section. The mesh and cross section files are contained in the radlib distribution, and the `Mesh_File` and `XS_File` lines should be edited to point to the location of the radlib installation. Some of the inputs have default values that are set in the SCEPTRE driver if the keyword is not found in the xml input. If an unknown or mistyped keyword is encountered in the input, the xml parser generally skips the unknown input and uses the default, without throwing an error. More error checking of the input will be added to future releases.

Fig. 1 shows one of the test mesh files included in the SCEPTRE distribution in the driver/test/mesh subdirectory. The mesh is a very coarse triangular mesh that includes five element blocks and five side sets for the interfaces between the blocks and the external boundary.

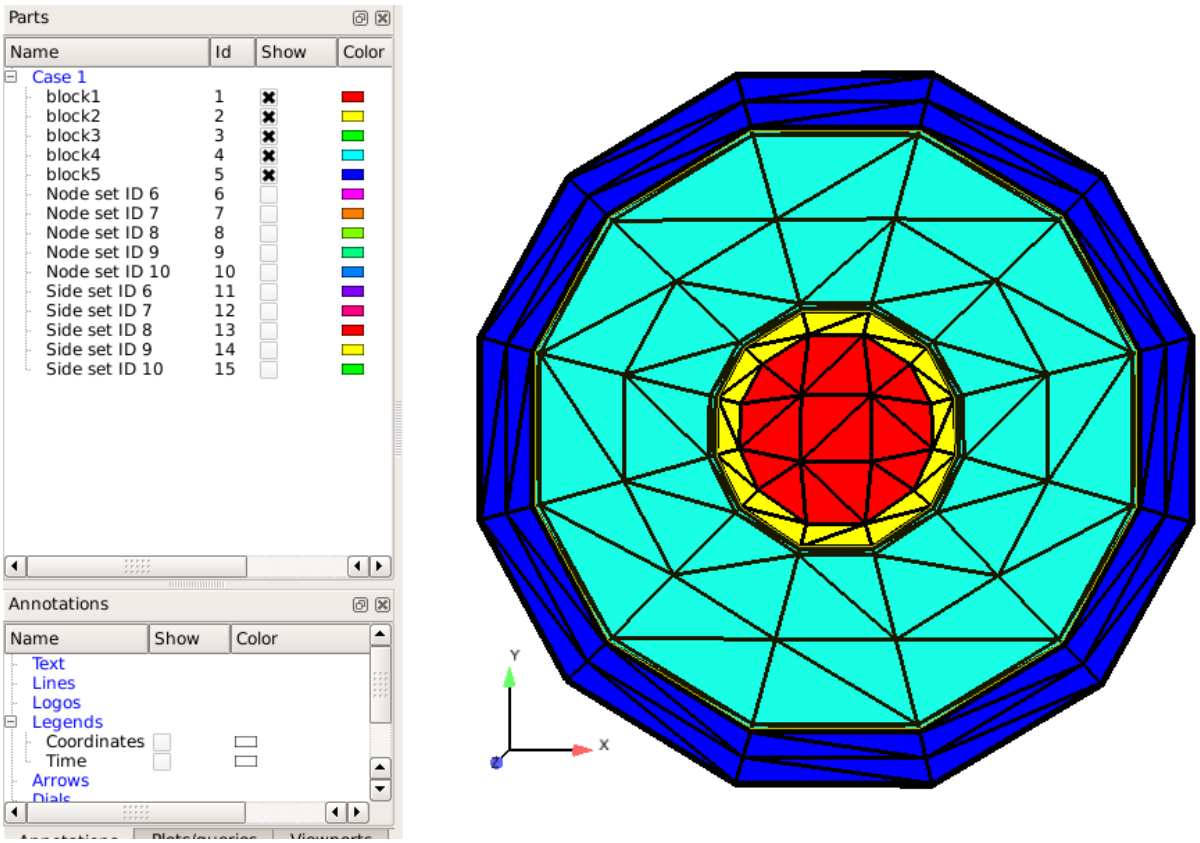

Figure 1 Mesh of a coaxial-cable cross section for the rg402 SCEPTRE example problem

The following sections describe in detail the various parameters that drive a SCEPTRE calculation.

5.4.1. Output options

The `Verbosity` tag dictates the level of output desired, and can be either `High`, `Medium` or `None`. The default is `Medium`. The `Write_Restart` tag indicates whether or not a restart file is written for performing a restart calculation from a partially-completed run. The default for `Write_Restart` is true.

```
<Output_Options>
  <Write_Restart>false</Write_Restart>
  <Verbosity>high</Verbosity>
</Output_Options>
```

5.4.2. Running in forward or adjoint mode

The `Transport_Mode` tag can be specified as either `Forward` or `Adjoint`. `Forward` is the default.

```
<Transport_Mode>Forward</Transport_Mode>
```

The same cross section library is used for either a forward or an adjoint run. The code handles energy group and angular index reversal internally, so that the user specifies fixed and boundary source terms in normal group and angle ordering. Unlike a forward run, for an adjoint run the boundary source terms are specified for outgoing directions rather than for incoming directions. The result of an adjoint calculation is the adjoint angular flux in normal energy group and angular index order, which may then be postprocessed as needed for the particular application being considered.

5.4.3. Specifying the input mesh and output result files

The `Mesh_File` tag specifies the path of the mesh file for either a serial or parallel run. For serial run, the path for the Genesis mesh file is specified, e.g. `testMeshTri3.gen`. For a parallel run, the prefix of the partitioned Genesis file is specified, e.g. `testMeshTri3.par`, and SCEPTRE then appends a partition number to complete the name of the partitioned Genesis file, e.g. `testMeshTri3.par.4.0` ... `testMeshTri3.par.4.3` for a 4-processor run.

```
<Mesh_File>mesh/testMeshTri3.par</Mesh_File>
```

The `Output_Format` tag specifies the format for writing the resulting angular flux files, which may be quite large, including the full spatial, angular and energy angular flux. Three format options are available, `Exodus`, `Netcdf`, or `Binary`, with `Exodus` being the default. `Exodus` directs SCEPTRE to write the angular-flux results in an Exodus file, which may be quite time-consuming for large problems. `Binary` directs SCEPTRE to write the angular flux results in a SCEPTRE binary format that is much faster than writing to Exodus format. `Netcdf` directs SCEPTRE to write the angular fluxes to a SCEPTRE Netcdf-format file (not Exodus format), that is intermediate between Exodus and Binary formats for read/write times

```
<Output_Format>Exodus</Output_Format>
```

The Output_Prefix tag specifies the prefix of the Exodus output, which may or may not include the angular-flux results from the transport calculation. For Binary or Netcdf output format, the Exodus output file does not include angular flux results, while for Exodus output format, the Exodus output file does include angular flux results.

```
<Output_Prefix>testMeshTri3.e</Output_Prefix>
```

For Binary or Netcdf output format, an additional tag Dump_File_Prefix specifies the file prefix for writing the SCEPTRE angular flux results.

```
<Dump_File_Prefix>testMeshTri3.bin</Dump_File_Prefix>
```

5.4.4. Specifying the cross-section file

The name of the Netcdf cross section file is specified by the XS_File tag. The creation of a Netcdf-format cross section file from a CEPXS run is described in Sec. 5.1 of this document.

```
<XS_File>xsec/10g.xslib</XS_File>
```

5.4.5. Angular quadrature and scattering options

The angular quadrature parameters are specified in the Sn_Options section. The Sn_Order tag specifies the S_N order for the transport calculation, and the Angular_Quadrature_Type tag specifies which angular quadrature is desired. For one-dimensional calculations, acceptable values for the Angular_Quadrature_Type are Gauss_Legendre or Gauss_Lobatto. For multi-dimensional calculations, acceptable values are Level_Symmetric, Lebedev (Lebedev & Laikov, 1999) or Chebyshev. Acceptable values for the Sn_Order for specific Angular_Quadrature_Type are as follows. For the one-dimensional quadratures, any even order is allowed. For Level_Symmetric quadrature, any even order up to and including order 20 is allowed. Arbitrary-order Chebyshev quadratures are allowed. Levedev quadratures of orders: 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 34 40 46 52 58 64 70 76 82 88 94 100 106 112 118 124 130 130 are allowed. Quadrature weights and direction cosines for the various quadratures are included in a separate document (Bruss. et. al, 2021)

```
<Sn_Options>
   <Sn_Order>4</Sn_Order>
   <Angular_Quadrature_Type>Level_Symmetric</Angular_Quadrature_Type>
 </Sn_Options>
```

Options for handling the particle scattering are included in the Scattering_Options section. The Scattering_Order tag specifies the order of the Legendre expansion of the angular treatment of the particle scattering. The user is responsible for ensuring that the Sn_Order is sufficiently large to accurately integrate the angular dependence of the scattering; typically, an Sn_Order of at least two times the Scattering_Order is sufficient.

```
<Scattering_Options>
 <Scattering_Order>1</Scattering_Order>
 <Delta_Function_Scattering_Correction>true</Delta_Function_Scattering_Correction>
 <Scattering_Moments_Method>STANDARD</Scattering_Moments_Method>
</Scattering_Options>
```

SCEPTRE has two methods for handling extremely forward-peaked scattering, that are needed for charged-particle transport. In one method, a δ-function down scattering portion of the scattering is treated explicitly, and the remaining less forward-peaked part of the scattering is handled with a standard multigroup-Legendre approach. The `Delta_Function_Scattering_Correction` tag specifies whether a δ-function (extended-transport) correction is applied to the angular approximation (Drumm, 2007). This option greatly increases accuracy for particles with highly-forward-peaked scattering, e.g., electron transport. The `Delta_Function_Scattering_Correction` has very little effect on x-ray transport but can be used for any particle type. The user is responsible for ensuring that the cross-section library was generated with a Legendre order of at least one greater than the `Scattering_Order` desired for the SCEPTRE calculation (or with a Legendre order at least as great as the `Scattering_Order` without the `Delta_Function_Scattering_Correction`).

Another method for handling the extreme forward-peaked electron scattering is to use a Galerkin scattering treatment (Morel J. , 1989), which is invoked by setting the `Scattering_Moments_Method` to `Galerkin`. For `Galerkin` `Scattering_Moments_Method`, the scattering moments are determined from the S_N order and angular quadrature type specified, rather than from the `Scattering_Order`. Using a `Galerkin` treatment (Morel, J. 1989), there is an exact mapping from the angular moments to angular quadrature space, and the number of scattering moments is equal to the number of directions in the angular quadrature. `Standard` is the default `Angular_Moments_Method`. This is the approach that is used in the one-dimensional CEPXS/ONELD and ADEPT codes.

In the Galerkin approach, a square moment-to-discrete matrix is constructed that is invertible, so that the conversion from discrete space to moment space and back again is exact. For one-dimensional geometry, the use of Gauss-Legendre quadrature with the same number of Legendre moments as discrete directions is equivalent to Galerkin. For multi-dimensional geometries, the Galerkin scattering treatment is not unique, but the procedure is the same. A square, invertible moment-to-discrete matrix is constructed, which can exactly handle the δ-function down scatter. SCEPTRE contains algorithms for computing Galerkin moment-to-discrete matrices for all available quadrature types: level-symmetric, Lebedev, Gauss-Legendre, and Lobatto.

For methods resulting in an SPD linear system, the δ-function down scatter method should be used rather than the Galerkin method, since use of the Galerkin scattering treatment results in a non-symmetric matrix.

5.4.6. Controlling the outer iterations

For problems involving upscattering or full-particle scattering coupling, several outer iterations may be required. The `Outer_Iteration_Options` include two parameters, the `Maximum_Number_Iterations` and `Convergence_Tolerance`. There are no default values for these parameters.

```
<Outer_Iteration_Options>
  <Maximum_Number_Iterations>2</Maximum_Number_Iterations>
  <Convergence_Tolerance>1.e-3</Convergence_Tolerance>
</Outer_Iteration_Options>
```

5.4.7. Material assignment and mixing options

In the Materials section of the xml input, the materials from the cross-section library are sequentially named and assigned a material density.

```
<Materials enable="true">
  <Material name="iron">
    <Density>7.874</Density>
  </Material>
  <Material name="copper">
    <Density>8.96</Density>
  </Material>
  <Material name="silver">
    <Density>10.5</Density>
  </Material>
  <Material name="teflon">
    <Density>2.2</Density>
  </Material>
</Materials>
```

In addition to Density, there are several optional parameters that can be assigned to each material: `Conductor` (true or false), `DielectricConstant` and `ElectricalConductivity`.

In the Mixtures section, material mixtures are named and assembled from individual materials from the cross-section library

```
<Mixtures enable="true">
  <Mixture name="Void">
    <Composition>
       <Material_Fraction>copper 0.0</Material_Fraction>
    </Composition>
  </Mixture>
  <Mixture name="FeCu">
    <Composition>
       <Material_Fraction>iron 0.5</Material_Fraction>
       <Material_Fraction>copper 0.5</Material_Fraction>
    </Composition>
  </Mixture>
</Mixtures>
```

In the Material_Assignment section, materials are assigned to element blocks in the Exodus mesh. The materials are assigned by element block name.

```
<Material_Assignment enable="true">
  <ElementBlock name="block1">copper</ElementBlock>
  <ElementBlock name="block2">iron</ElementBlock>
  <ElementBlock name="block3">silver</ElementBlock>
  <ElementBlock name="block4">teflon</ElementBlock>
  <ElementBlock name="block5">copper</ElementBlock>
</Material_Assignment>
```

5.4.8. Assigning boundary conditions

Boundary conditions are assigned in the `Boundary_Condition_Assignment` section. Boundary conditions are assigned to side sets in the Exodus mesh, and can be either `Vacuum`, `Reflective` or `Periodic`. For a periodic boundary condition, two side sets are specified, which must have nodes that match up with each other. As an example, for a mesh with four side sets, `TOP`, `BOTTOM`, `LEFT` and `RIGHT`, boundary conditions could be assigned as follows.

```
<Boundary_Condition_Assignment explicit="true">
  <Boundary_Condition>Vacuum TOP</Boundary_Condition>
  <Boundary_Condition>Reflective BOTTOM</Boundary_Condition>
  <Boundary_Condition>Periodic LEFT RIGHT</Boundary_Condition>
</Boundary_Condition_Assignment>
```

5.4.9. Boundary source options

SCEPTRE has very powerful source-assignment capability, which will be described in this section. Boundary sources are assigned by energy group, angle index, and side set. The boundary source is specified in the `Boundary_Source_Options` section. Individual boundary sources are assigned in sections with the Source tag, and the individual sources are then summed together. The angular distribution is specified with the `Angle_Expansion` keyword, the energy distribution is specified with the `Energy_Expansion` keyword, and the side set expansion is specified with the `SideSet_Expansion` keyword.

Boundary sources are specified for all incoming directions on the external boundary of the geometry. If an `Energy_Expansion` is omitted, the default is uniform in each energy group. If an `Angle_Expansion` is omitted, the default is isotropic in direction. If the `SideSet_Expansion` keyword is omitted, the default is uniform over all external boundary sides. A `Scale_Factor` may also be specified to multiply the boundary source by a single scaling factor.

The simplest source specification is

```
<Source>
</Source>
```

which specifies a unit source by energy group, isotropic in angle and for all incoming directions on the external boundary of the geometry.

If a Scale_Factor is included, as here

```
<Source>
  <Scale_Factor>3.</Scale_Factor>
</Source>
```

the source is the same as the previous example, except multiplied by a factor of 3.

The next example specifies individual source values of 1.0, 0.4, and 0.2 for groups 1, 2, and 3, respectively, and a value of 0.6 for groups 4 through 6.

```
<Source>
  <Scale_Factor>2.</Scale_Factor>
```

```
    <Energy_Expansion enable="true">
      <Group index="1">0.5</Group>
      <Group index="2">0.2</Group>
      <Group index="3">0.1</Group>
      <Group_Range>0.3 4 6</Group_Range>
    </Energy_Expansion>
  </Source>
```

The next example specifies a source for energy groups 1 and 6, with all the remaining groups defaulting to zero. Angular values are specified for angles with indices of 1, 2, and 3, with the angular distribution for the remaining angle indices defaulting to zero. Boundary sources are scaled by a factor of 5 for all sides included in the side set named "sideset5", with default values of zero for all other sides in the external boundary. Finally, the source is scaled by a factor of 8.

```
  <Source>
    <Scale_Factor>8.</Scale_Factor>
    <Energy_Expansion enable="true">
      <Group index="1">2.5</Group>
      <Group index="6">3.5</Group>
    </Energy_Expansion>
    <Angle_Expansion enable="true">
      <Angle index="1">3.2</Angle>
      <Angle_Range>3.3 2 3</Angle_Range>
    </Angle_Expansion>
    <SideSet_Expansion enable="true">
      <SideSet name="sideset5">5.</SideSet>
    </SideSet_Expansion>
  </Source>
```

As noted previously, multiple source distributions may be specified, which are added together to generate a complete boundary source for the SCEPTRE transport calculation, allowing for very complicated boundary source specifications. In this manner, it is possible to specify a full energy/angular distribution, by either specifying an energy distribution for each angle, or by specifying an angular distribution for each energy group.

```
<Boundary_Source_Options enable="true">
  <Source>
  </Source>
  <Source>
    <Scale_Factor>3.</Scale_Factor>
  </Source>
  <Source>
    <Scale_Factor>2.</Scale_Factor>
    <Energy_Expansion enable="true">
      <Group index="1">0.5</Group>
      <Group index="2">0.2</Group>
      <Group index="3">0.1</Group>
      <Group_Range>0.3 4 6</Group_Range>
    </Energy_Expansion>
  </Source>
  <Source>
    <Scale_Factor>8.</Scale_Factor>
    <Energy_Expansion enable="true">
      <Group index="1">2.5</Group>
```

```
      <Group index="6">3.5</Group>
    </Energy_Expansion>
    <Angle_Expansion enable="true">
      <Angle index="1">3.2</Angle>
      <Angle_Range>3.3 2 3</Angle_Range>
    </Angle_Expansion>
    <SideSet_Expansion enable="true">
      <SideSet name="sideset5">5.</SideSet>
    </SideSet_Expansion>
  </Source>
</Boundary_Source_Options>
```

5.4.10. Fixed source options

The fixed-source capability in SCEPTRE is similar to that described in the previous section for the boundary-source options, with the notable exception that the fixed source can be specified by element block rather than by side set. The fixed source is specified by setting the `Fixed_Source_Options` tag to "true". Angular and energy dependencies and scaling are the same for both source types. In order to specify a fixed source by element block, the `Block_Expansion` tag is set to "true", and the values are set by block name with the `Block` tag.

```
    <Block_Expansion enable="true">
      <Block name="block4">5.</Block>
    </Block_Expansion>
```

It is also possible to read in a fixed source from a file, either in Exodus format or in binary format. In order to read in a fixed source in Exodus format, the fixed source is simply added to the `Mesh_File`, as shown here for an adjoint source example. Use of this method requires that the fixed source has been previously written to the Exodus mesh file. Tools for adding a fixed source are available in the preprocessing tools included in the SCEPTRE distribution, and users requiring this capability are encouraged to contact the SCEPTRE support team for assistance with their particular application.

```
    <Mesh_File>source/adjointSource.e</Mesh_File>
```

In order to read in a fixed source from a binary format file requires inclusion of the `Binary_Fixed_Source_Input` tag, which is given a `File_Prefix` that provides the binary adjoint source file or files.

```
    <Binary_Fixed_Source_Input enable="true">
       <File_Prefix>source/adjointSource.bin</File_Prefix>
    </Binary_Fixed_Source_Input>
```

A complete example fixed source specified in the xml input file is shown as follows.

```
<Fixed_Source_Options enable="true">
  <Source>
  </Source>
  <Source>
    <Scale_Factor>3.</Scale_Factor>
  </Source>
  <Source>
```

```
    <Scale_Factor>2.</Scale_Factor>
    <Energy_Expansion enable="true">
      <Group index="1">0.5</Group>
      <Group index="2">0.2</Group>
      <Group index="3">0.1</Group>
      <Group_Range>0.3 4 6</Group_Range>
    </Energy_Expansion>
  </Source>
  <Source>
    <Scale_Factor>8.</Scale_Factor>
    <Energy_Expansion enable="true">
      <Group index="1">2.5</Group>
      <Group index="6">3.5</Group>
    </Energy_Expansion>
    <Angle_Expansion enable="true">
      <Angle index="1">3.2</Angle>
      <Angle_Range>3.3 2 3</Angle_Range>
    </Angle_Expansion>
    <Block_Expansion enable="true">
      <Block name="block4">5.</Block>
    </Block_Expansion>
  </Source>
</Fixed_Source_Options>
```

5.4.11. Setting the initial solution

The default initial solution is set to zero, but if an estimate of the final solution is available, use of the estimate as an initial guess can greatly speed up convergence of the result. The initial estimate of the angular flux solution is specified by setting the `Internal_Angular_Flux_Initialization_Options` tag to "true". The options available for setting the initial solution estimate are the same as those for setting the fixed source. An example for setting the initial guess is shown here.

```
<Internal_Angular_Flux_Initialization_Options enable="true">
  <Scale_Factor>1</Scale_Factor>
  <Energy_Expansion enable="true">
    <Group index="1">1</Group>
    <Group index="2">1</Group>
    <Group index="3">1</Group>
    <Group index="4">1</Group>
    <Group index="5">1</Group>
  </Energy_Expansion>
  <Angle_Expansion enable="false">
    <Angle index="1">1</Angle>
  </Angle_Expansion>
  <Block_Expansion enable="true">
    <Block index="1">1</Block>
  </Block_Expansion>

</Internal_Angular_Flux_Initialization_Options>
```

5.4.12. Source input from a separate file

Alternatively, source information can be included from a separate xml file

```
<Fixed_Source_Options enable="true">
    <Input_From_Separate_File enable="true">sourceInput.xml</Input_From_Separate_File>
</Fixed_Source_Options>

<Boundary_Source_Options enable="true">
    <Input_From_Separate_File enable="true">sourceInput.xml</Input_From_Separate_File>
</Boundary_Source_Options>
```

with the fixed source and/or boundary source input included in a separate xml file named sourceInput.xml

5.5. Solver options

SCEPTRE includes capability for solving the Boltzmann equation using many different numerical and iterative methods and allows for a different transport solver to be used for each energy group, enabling the user to apply the most appropriate methods for accuracy and efficiency for each energy group/particle type in the problem. For example, the sweeps-based solver is very efficient for neutral-particle transport, e.g., photon transport, while one of the Krylov iterative solvers may be more efficient for solving charged-particle transport. The user can define as many solvers as desired, giving each solver a unique name.

```
<Solver name="Sn-LS">
  …
</Solver>
```

Then the individual solvers are assigned by energy group by using the `Solver_Assignment` tag, as described in the previous section.

By setting the `Enable_User_Defined_Solvers` tag to "true", the user can define any number of different solvers, each solver within a `Solvers` section.

```
<Enable_User_Defined_Solvers>true</Enable_User_Defined_Solvers>
  <Solvers>
    <Solver name="1stOrder">
      <Solver_Form>First_Order</Solver_Form>
    </Solver>
  </Solvers>
```

The `Solver_Form` keyword can be either `First_Order` to define a wave front sweeping solver, or `Krylov` to define a solver using Trilinos to perform a simultaneous space/angle solve. Options for the wave-front sweeping solvers will be described in the next section, and options for the Trilinos solvers will be described in the following section.

5.5.1. Wave front sweeping algorithm keywords

If `First_Order` is specified as the solver form, the following options may be specified within that `Solver` section:

The Element_Set_Size option is used to control how the solver aggregates spatial elements during the solution process. The default is 1 if this option is not explicitly specified. This is an experimental option; it may lead to increased number of iterations in order to obtain solution convergence if a value greater than 1 is used.

The Coarse_Sn_Order option is used to control how the solver aggregates angular directions during the solution process. The default is the value specified in the Sn_Order option described earlier. Values less than that may reduce the time required per iteration, but they also may increase the number of iterations required for solution convergence; the trade-off will be problem- and machine-dependent.

The Preconditioner option is used to control how and whether the solver attempts to speed up the iterative process. Valid options are "none" (the default), "dsa" or "tsa". If "dsa" is specified, then additional options may be specified with a Preconditioner_Options keyword. Within that option are the following suboptions:

- Solver (valid options are "gmres" (the default) or "cg")
- Maximum_Number_Iterations (default is 1000)
- Convergence_Tolerance (default is 1.e-10)

For "tsa" preconditioner options, refer to Sec. 5.7.3.

The Error_Control_Options section is always required with the sweeping algorithm; some of its suboptions do not have default values. The following suboptions are contained in this section:

- The Maximum_Number_Iterations suboption specifies the maximum number of source iterations that this solver will use. There is no default value, so this suboption must be explicitly specified.
- The Convergence_Tolerance suboption specifies the maximum iterative error allowed; iterations will continue until either the error is smaller or until Maximum_Number_Iterations is reached. There is no default value, so this suboption must be explicitly specified.
- The Number_Aggregated_Iterations suboption specifies how many source iterations will be used in between the measurement of errors to determine iterative convergence. The default is 1. Higher numbers can help produce a better estimate of the iterative error, but can also cause more iterations to be used than strictly necessary.
- The next two suboptions require some explanation. Within the sweep solver are a variety of tests to determine if iterative convergence has been achieved. Some transport quantities of interest may converge to the desired accuracy more rapidly than others. The suboptions Error_Metrics and Boundary_Errors are provided to allow the user to specify one or more iterative tests that must all pass in order for the iterative process to complete. Each estimates the remaining iterative error in some quantity. We describe each in turn:
- The Error_Metrics suboption allows the user to specify one or more metrics that measure some quantity in the interior of the problem. The default is "default" – this creates a single interior error metric with default values for each of the various parameters described below. Alternatively, the user may desire to explicitly create some error metrics. To do so the user needs to specify one or more Error_Metric sections within the Error_Metrics suboption. Within each of the sections the following options may appear:

- `Metric_Type`: This specifies the region of interest for the error measurement. The following are valid values:
  - "whole": The entire volume of the problem will be examined. This is the default.
  - "region": Attention will be restricted to a particular region of the problem. If this value is specified, the user will also need to include a `Region_Name` option that specifies the name of the element block to be examined.
  - "surface": Fluxes at a surface will be examined. If this value is specified, the user will also need to include a `Surface_Name` option that specifies the name of the surface (Exodus side set) to be examined.
  - "leakage": This is similar to the "surface" option, except that the net leakage at a surface will be examined. The `Surface_Name` option will also need to be specified.
  - "null": This produces an error metric that does nothing. This can be useful if the user desires to have the solver perform a fixed number of iterations regardless of whether convergence has been achieved.
- `Integration_Policy`: This specifies how the various point values of the fluxes are weighted relative to each other. Valid values are:
  - "discrete": The set of fluxes is treated as a simple vector quantity; each value is equally weighted. This is the default.
  - "continuous": A true integration of the fluxes over the mesh is performed. If the volumes and/or shapes of the elements in the mesh differ from each other the continuous option will yield a different value than the discrete option; it is more accurate but more expensive.
- `Error_Norm`: This specifies whether the angular fluxes themselves will be used or if some operator will be applied first. Valid values are
  - "L": This applies the familiar L-norm to the errors in the fluxes. This is the default.
  - "H": This applies the H-norm, i.e. the L-norm of $|\nabla\psi|$, the magnitude of the gradient in the errors in the fluxes.
  - "S": This applies the streaming norm, which is the L-norm of $\Omega \cdot \nabla\psi$.
- `Error_Order`: This specifies the power of the metric. Valid values are "1", "2" (the default), or "I" (infinity). For example, an `Error_Norm` of "L" and an `Error_Order` of "2" produces the L2 norm.
- `Sign_Policy`: This determines whether the absolute value of the integrand is used (a true norm) or whether sign cancellations are allowed (as happens when calculating integral quantities such as reaction rates). Valid values are "absolute" (the default) and "signed".

- The `Boundary_Errors` suboption has been largely superseded by the `Error_Metrics` suboption. A single metric may be specified that is applied to the entire external boundary. This suboption has the same sections as the `Error_Metrics` suboption: `Metric_Type`, `Integration_Policy`, `Error_Order`, and `Sign_Policy`. The difference is that the default for `Metric_Type` is "null", and the keyword "external" is used instead of "surface".

5.5.2. Krylov solver keywords

SCEPTRE includes many options for solving the transport equation, including several different

- forms of the Boltzmann transport equation
- angular approximations
- preconditioning options
- iterative methods

In order to define one of the Trilinos-based solvers, the solver is given a unique name, and the `Solver_Form` tag is set to `Krylov`.

```
<Enable_User_Defined_Solvers>true</Enable_User_Defined_Solvers>

  <Solvers>
    <Solver name="Sn-LS">
      <Solver_Form>Krylov</Solver_Form>
        …
    </Solver
  </Solvers>
```

SCEPTRE includes three forms of the transport equation that may be solved, that are defined by the `Krylov_Transport_Method` key. Currently available methods include a `LeastSquares` method (Drumm, 2011), a self-adjoint angular flux (SAAF) method (Morel, 1999), specified by the `SelfAdjoint` key, and solving the standard first-order form of the transport equation, specified by the `FirstOrder` key. The `FirstOrder` method results in a solution identical to the use of the sweeps-based solver. Other solvers are planned to be included in SCEPTRE in the future, e.g. an even-odd parity solver, but this has not been implemented yet.

```
<Krylov_Transport_Method>LeastSquares</Angular_Transport_Method>
```

SCEPTRE includes four methods for handling the angular dependence of the radiation transport, specified by the `Angular_Method` tag, two production methods and two experimental methods that are under active development. The two production methods are discrete ordinates, specified by the `Sn` key, and spherical harmonics, specified by the `Pn` key. The two experimental methods for handling the angular dependence are `AngularCFE` for continuous angular finite elements and `AngularDFE` for discontinuous angular finite elements.

```
<Angular_Method>Sn</Angular_Method>
```

SCEPTRE includes capability for both continuous spatial finite elements, `SpatialCFE`, and discontinous spatial finite elements, `SpatialDFE`, specified by the `SpatialFE_Method` tag. Methods resulting in a symmetric positive definite (SPD) linear system (`SelfAdjoint` and `LeastSquares`) must use `SpatialCFE`, while the `FirstOrder Krylov_Transport_Method` must use `SpatialDFE`. This will be enforced in future releases, but for now the user is responsible for using the correct spatial FE method.

```
<SpatialFE_Method>SpatialCFE</SpatialFE_Method>
```

A number of different solver options are available in Trilinos, and SCEPTRE can make use of three of them, by setting the `Linear_Solver` tag. `Belos` is by far the most commonly used linear solver for most problems. For small problems, e.g. one-dimensional applications, use of a direct solver may be very efficient, used by setting the `Linear_Solver` tag to `Direct`. SCEPTRE also can make use of MueLu as a solver, by setting the `Linear_Solver` tag to `MueLu`.

```
<Linear_Solver>Belos</Linear_Solver>
```

Some additional parameters can be set when using one of the Trilinos Krylov iterative solvers, but are not needed when using a direct solver. Trilinos includes a number of Krylov iterative solvers, and SCEPTRE can make use of two of them by setting the `Iterative_Method` tag. `CG` specifies a conjugate-gradients iterative solver is to be used and `GMRES` indicates that a GeneralizedMinimumRESiduals solver is to be used. `CG` is used for the methods that result in an SPD linear system, `LeastSquares` and `SelfAdjoint`, and `GMRES` is used for the `FirstOrder` method. This will be enforced in future releases, but for now the user is responsible for using the appropriate Krylov iterative method for the particular transport method chosen for the solver.

```
<Iterative_Method>CG</Iterative_Method>
```

The `Verbosity` is set to `high`, `medium`, `low` or `none`.

```
<Verbosity>medium</Verbosity>
```

Control of the Krylov iterative solvers is specified by setting the `Maximum_Number_Iterations`, `Convergence_Tolerance`, and, for a `GMRES` solve, `Krylov_Subspace_Size`.

```
<Maximum_Number_Iterations>100</Maximum_Number_Iterations>
<Convergence_Tolerance>1.e-2</Convergence_Tolerance>
<Krylov_Subspace_Size>200</Krylov_Subspace_Size>
```

By using Trilinos to perform the iterative linear solve, it is fairly easy to include various preconditioiners in the algorithm. SCEPTRE includes options for two different preconditioners, 1) an incomplete-factorization method and 2) a multi-level method. It generally takes some experimentation to determine which combination of solvers and preconditioners is optimal for a given application and particle/energy group.

The Preconditioner can be either `None` for no preconditioner of the linear system, `IncompleteFactorization` for algebraic preconditioning using IFPACK (Sala & Heroux, 2005), or `MultiLevel` to use multigrid preconditioning using MuLue (Propenko et al., 2014).

```
<Preconditioner>IncompleteFactorization</Preconditioner>
```

5.5.3. Transport Synthetic Acceleration (TSA) keywords

A Transport Synthetic Acceleration (TSA) preconditioner (Adams and Larsen, 2003) uses a low-order transport solve to reduce the low-order error in the angular flux for each source iteration. This method may greatly speed up convergence for applications with high scattering ratios, e.g. electron/positron transport. Within a `First_Order` solver block, the keyword `TSA` is specified as a `Preconditioner`. In the `Preconditioner_Options` block, any of the Krylov transport solver options may then be specified to define the coarse solve. The only restriction at this point is the only allowed angular method is `Pn`. Future releases of SCEPTRE will include capability for an `Sn TSA` solve.

```
<Preconditioner>TSA</Preconditioner
<Preconditioner_Options>
  <SpatialFE_Method>SpatialDFE</SpatialFE_Method>
  <Angular_Method>Pn</Angular_Method>
```

```
      <Pn_Order>1</Pn_Order>
      <Krylov_Transport_Method>FirstOrder</Krylov_Transport_Method>
      <Iterative_Method>gmres</Iterative_Method>
      <Preconditioner>None</Preconditioner>
      <Maximum_Number_Iterations>1000</Maximum_Number_Iterations>
      <Convergence_Tolerance>1.e-4</Convergence_Tolerance>
      <Verbosity>low</Verbosity>
    </Preconditioner_Options>
```

The TSA preconditioner can be further preconditioned by specifying either `IncompleteFactorization` of `MultiLevel` for the `Preconditioner` tag, thus providing great flexibility in specifying a TSA preconditioner. Much experimentation with different options and parameters is often needed to optimize convergence for a particular application.

5.6 Assigning solvers by energy group

SCEPTRE includes several different methods for solving the Boltzmann transport equation, along with various acceleration methods. Solvers available in SCEPTRE include a wave front sweeping first-order transport algorithm (Wareing, McGhee, Morel, & Pautz, 2001), a Self-Adjoint Angular Flux (SAAF) (Morel & McGhee, 1999) second-order transport solver, and a Least-Squares Finite-Element (LSFE) solution method. The user may define any number of different solvers with associated preconditioners/accelerators. The solvers are then assigned to specific energy groups for the multi-group transport solve. Keywords used to define specific solvers are described in the next section.

The names of the solvers are arbitrary and may be defined by the user. The solvers are assigned by energy group range; in this example the unpreconditioned first-order solver is assigned to energy groups 1-4 (which are photon groups for this test problem) and first-order solvers with various preconditioners and the SAAF solver are assigned to groups 6-10 (which are electron groups). Details of the solver specifications will be described in detail in the next section for each of the transport solver methods.

```
<Enable_User_Defined_Solvers>true</Enable_User_Defined_Solvers>
<Solvers>
  <Solver name="1stOrder">
      …
  </Solver>
</Solvers>
```

Individual solvers are assigned to specific energy groups by specifying either the `Solver_By_Group_Range` tag to associate a solver with a range of energy groups, or the `Solver_By_Group` tag to associate a solver with a single energy group.

```
<Solver_Assignment explicit="true">
  <Solver_By_Group_Range>1stOrder 1 4</Solver_By_Group_Range>
  <Solver_By_Group>
    <Group index="5">Pn-Direct</Group>
    <Group index="6">1stOrder</Group>
    <Group index="7">Sn-FirstOrderKrylov</Group>
    <Group index="8">1stOrder-TSA-P0</Group>
    <Group index="9">1stOrder-TSA-P0</Group>
    <Group index="10">Sn-LS</Group>
```

```
    </Solver_By_Group>
  </Solver_Assignment>
```

A solver must be assigned to each energy group.

REFERENCES

Adams, M. L. and Larsen, E. W. (2002), "Fast Iterative Methods for Discrete-Ordinates Particle Transport Calculations," *Prog. Nucl. Energy*, **40**, 3.

Bell, G. I., & Glasstone, S. (1970). *Nuclear Reactor Theory*. New York: Van Nostrand Reinhold.

Bruss, D. & Campbell, B. (2020), *RAPTURE User's Guide*, Albuquerque: Sandia National Laboratories Report, SAND2020-3581.

Bruss, D., Drumm, C., Fan, W., and Pautz, S. (2021), *SCEPTRE 2.2 Angular Quadrature Sets*, Albuquerque: Sandia National Laboratories Report, SAND 2021-3084.

CUBIT user's manual. (2019). Retrieved from CUBIT 15.4

Drumm, C. (2007). An Analysis of the Extended-Transport Correction with Application to Electron Beam Transport. *Nuclear Science and Engineering*, **155**, 355-366.

Drumm, C. R., & Lorenz, J. (1999), Parallel FE Electron-Photon Transport Analysis on a 2-D Unstructured Mesh. *Mathematics and Computation, Reactor Physics and Environmental Analysis in Nuclear Applications* (pp. 858-868). Madrid: American Nuclear Society.

Drumm, C., Fan, W., & Pautz, S. (2013). Phase-Space Finite Elements in a Least-Squares Solution of the Transport Equation. *Proceedings of the 2013 International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering - M and C 2013* (pp. 877-892). Sun Valley: American Nuclear Society.

Drumm, C., Fan, W., Bielen, A., & Chenhall, J. (2011). Least Squares Finite Elements Algorithms in the SCEPTRE Radiation Transport Code. *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2011)*. Rio de Janeiro: American Nuclear Society.

Drumm, C, "Spherical Harmonics (P_N) Methods in the SCEPTRE Radiation Transport Code," *ANS MC2015 - Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method*, Nashville, TN (*M&C 2015*).

Duderstadt, J. J., & Martin, W. R. (1979). *Transport Theory*. New York: John Wiley & Sons.

Franke, B. et al., "ITS Version 6.7: The Integrated TIGER Series of Coupled Electron/Photon Monte Carlo Transport Codes," Sandia National Laboratories report, SAND2008-3331 (2008).

Heroux, M. A., & Willenbring, J. M. (2003). *Trilinos Users Guide*. Albuquerque: Sandia National Laboratories report, SAND2003-2952.

Mark Hoemmen, Jonathan Hu, and Chris Siefert, Ifpack2 documentation.

Lebedev, V. I., & Laikov, D. N. (1999). A Quadrature Formula for the Sphere of the 131st Algebraic Order of Accuracy. *Doklady Mathematics*, 741-745.

Lewis, E. E., & Miller, W. F. (1984). *Computational Methods of Neutron Transport*. New York: John Wiley & Sons.

Lorence, L. J., Morel, J. E., & and Valdez, G. D. (1989). *Physics Guide to CEPXS: A Multigroup Coupled Electron-Photon Cross-Section Generating Code Version 1.0*. Albuquerque, NM 87185: SAND89-1685 Sandia National Laboratories.

Morel, J. (1989). A Hybrid Collocation-Galerkin-Sn Method for Solving the Boltzmann Transport Equation. *Nuclear Science and Engineering*, 72-87.

Morel, J. E., & McGhee, J. M. (1999). A Self-Adjoint Angular Flux Equation. *Nuclear Science and Engineering*, 312-325.

Pautz, S., Bohnhoff, W., Drumm, C., & Fan, W. (2009). Parallel Discrete Ordinates Methods in the SCEPTRE Project. *International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2009).* Saratoga Springs, New York: American Nuclear Society.

Pautz, S., Drumm, C., Bohnhoff, B., & Fan, W. (2009). Software Engineering in the SCEPTRE Code. *International Conference on Mathematics, Computational Methods & Reactor Physics (M&C 2009).* Saratoga Springs: American Nuclear Society.

Prokopenko, A., Hu, J., Wiesner, T., Siefert, C., and Tuminaro, R., (2014) MueLu User's Guide for Trilinos Version 11.12, Albuquerque, NM 87185, SAND2014-18874 Sandia National Laboratories.

Schoof, L. A., & Yarberry, V. R. (1994). *EXODUS II: A finite element data model.* Albuquerque: Sandia National Laboratories report SAND92-2137.

Wareing, T. A., McGhee, J. M., Morel, J. E., & Pautz, S. D. (2001). Discontinuous Finite Element SN Methods on Three-Dimensional Unstructured Grids. *Nuclear Science and Engineering*, 256-268.

APPENDIX A. COMPLETE XML INPUT FILE FOR RUNNING SCEPTRE

```
<?xml version="1.0" encoding="utf-8"?>

<SCEPTRE_Input>

  <Output_Options>
    <Verbosity>high</Verbosity>
  </Output_Options>

  <Transport_Mode>Forward</Transport_Mode>

  <Mesh_File>mesh/rg402Tri3.par</Mesh_File>

  <XS_File>xsec/rg402_10g.xslib</XS_File>

  <Output_Prefix>rg402Tri3</Output_Prefix>
  <Output_Format>Exodus</Output_Format>

  <Sn_Options>
    <Sn_Order>4</Sn_Order>
    <Angular_Quadrature_Type>Level_Symmetric</Angular_Quadrature_Type>
  </Sn_Options>

  <Scattering_Options>
    <Scattering_Order>1</Scattering_Order>
    <Delta_Function_Scattering_Correction>true</Delta_Function_Scattering_Correction>
    <Scattering_Moments_Method>STANDARD</Scattering_Moments_Method>
  </Scattering_Options>

  <Outer_Iteration_Options>
    <Maximum_Number_Iterations>2</Maximum_Number_Iterations>
    <Convergence_Tolerance>1.e-3</Convergence_Tolerance>
  </Outer_Iteration_Options>

  <Enable_User_Defined_Solvers>true</Enable_User_Defined_Solvers>
  <Solvers>

    <Solver name="1stOrder">
      <Solver_Form>First_Order</Solver_Form>
      <Element_Set_Size>1</Element_Set_Size>
      <Coarse_Sn_Order>4</Coarse_Sn_Order>
      <Preconditioner>None</Preconditioner>
      <Error_Control_Options>
        <Maximum_Number_Iterations>100</Maximum_Number_Iterations>
        <Convergence_Tolerance>1.e-4</Convergence_Tolerance>
        <Error_Metrics>
          <Error_Metric>
            <Metric_Type>whole</Metric_Type>
            <Integration_Policy>discrete</Integration_Policy>
            <Error_Norm>L</Error_Norm>
            <Error_Order>2</Error_Order>
            <Sign_Policy>absolute</Sign_Policy>
          </Error_Metric>
        </Error_Metrics>
        <Boundary_Errors>
          <Metric_Type>null</Metric_Type>
        </Boundary_Errors>
      </Error_Control_Options>
    </Solver>

    <Solver name="1stOrder-DSA">
      <Solver_Form>First_Order</Solver_Form>
      <Element_Set_Size>1</Element_Set_Size>
      <Coarse_Sn_Order>4</Coarse_Sn_Order>
      <Preconditioner>DSA</Preconditioner>
      <Preconditioner_Options>
        <Solver>CG</Solver>
        <Maximum_Number_Iterations>1000</Maximum_Number_Iterations>
        <Convergence_Tolerance>1.e-2</Convergence_Tolerance>
      </Preconditioner_Options>
```

```
  <Error_Control_Options>
    <Maximum_Number_Iterations>100</Maximum_Number_Iterations>
    <Convergence_Tolerance>1.e-4</Convergence_Tolerance>
    <Error_Metrics>
      <Error_Metric>
        <Metric_Type>whole</Metric_Type>
        <Integration_Policy>discrete</Integration_Policy>
        <Error_Norm>L</Error_Norm>
        <Error_Order>2</Error_Order>
        <Sign_Policy>absolute</Sign_Policy>
      </Error_Metric>
    </Error_Metrics>
    <Boundary_Errors>
      <Metric_Type>null</Metric_Type>
    </Boundary_Errors>
  </Error_Control_Options>
</Solver>

<Solver name="1stOrder-TSA-S2">
  <Solver_Form>First_Order</Solver_Form>
  <Element_Set_Size>1</Element_Set_Size>
  <Coarse_Sn_Order>4</Coarse_Sn_Order>

  <Preconditioner>TSA</Preconditioner>
  <Preconditioner_Options>
    <SpatialFE_Method>SpatialDFE</SpatialFE_Method>
    <Angular_Method>Sn</Angular_Method>
    <Sn_Order>2</Sn_Order>
    <Pn_Order>1</Pn_Order>
    <Krylov_Transport_Method>FirstOrder</Krylov_Transport_Method>
    <Iterative_Method>gmres</Iterative_Method>
    <Preconditioner>None</Preconditioner>
    <Maximum_Number_Iterations>1000</Maximum_Number_Iterations>
    <Convergence_Tolerance>1.e-4</Convergence_Tolerance>
    <Verbosity>medium</Verbosity>
  </Preconditioner_Options>

  <Error_Control_Options>
    <Maximum_Number_Iterations>10</Maximum_Number_Iterations>
    <Convergence_Tolerance>1.e-4</Convergence_Tolerance>
    <Error_Metrics>
      <Error_Metric>
        <Metric_Type>whole</Metric_Type>
        <Integration_Policy>discrete</Integration_Policy>
        <Error_Norm>L</Error_Norm>
        <Error_Order>2</Error_Order>
        <Sign_Policy>absolute</Sign_Policy>
      </Error_Metric>
    </Error_Metrics>
    <Boundary_Errors>
      <Metric_Type>null</Metric_Type>
    </Boundary_Errors>
  </Error_Control_Options>
</Solver>

<Solver name="1stOrder-TSA-P0">
  <Solver_Form>First_Order</Solver_Form>
  <Element_Set_Size>1</Element_Set_Size>
  <Coarse_Sn_Order>4</Coarse_Sn_Order>

  <Preconditioner>TSA</Preconditioner>
  <Preconditioner_Options>
    <SpatialFE_Method>SpatialDFE</SpatialFE_Method>
    <Angular_Method>Pn</Angular_Method>
    <Pn_Order>0</Pn_Order>
    <Krylov_Transport_Method>FirstOrder</Krylov_Transport_Method>
    <Iterative_Method>gmres</Iterative_Method>
    <Preconditioner>None</Preconditioner>
    <Maximum_Number_Iterations>1000</Maximum_Number_Iterations>
    <Convergence_Tolerance>1.e-4</Convergence_Tolerance>
    <Verbosity>medium</Verbosity>
```

```
  </Preconditioner_Options>

  <Error_Control_Options>
    <Maximum_Number_Iterations>10</Maximum_Number_Iterations>
    <Convergence_Tolerance>1.e-4</Convergence_Tolerance>
    <Error_Metrics>
      <Error_Metric>
        <Metric_Type>whole</Metric_Type>
        <Integration_Policy>discrete</Integration_Policy>
        <Error_Norm>L</Error_Norm>
        <Error_Order>2</Error_Order>
        <Sign_Policy>absolute</Sign_Policy>
      </Error_Metric>
    </Error_Metrics>
    <Boundary_Errors>
      <Metric_Type>null</Metric_Type>
    </Boundary_Errors>
  </Error_Control_Options>
</Solver>

<Solver name="Sn-LS">
  <Solver_Form>Krylov</Solver_Form>

  <!-- Sn, Pn, AngularCFE or AngularDFE -->
  <Angular_Method>Sn</Angular_Method>

  <!-- SelfAdjoint, LeastSquares, EvenOddParity, EvenParity, OddParity or FirstOrder -->
  <Krylov_Transport_Method>LeastSquares</Krylov_Transport_Method>

  <!-- SpatialCFE or SpatialDFE -->
  <SpatialFE_Method>SpatialCFE</SpatialFE_Method>

  <!-- Belos, MueLu or Direct -->
  <Linear_Solver>Belos</Linear_Solver>

  <!-- CG or GMRES -->
  <Iterative_Method>CG</Iterative_Method>

  <Maximum_Number_Iterations>100</Maximum_Number_Iterations>
  <Convergence_Tolerance>1.e-2</Convergence_Tolerance>

  <!-- high, medium, low or none -->
  <Verbosity>medium</Verbosity>

  <!-- None, MultiLevel, IncompleteFactorization -->
  <Preconditioner>IncompleteFactorization</Preconditioner>

</Solver>

<Solver name="Pn-Direct">
  <Solver_Form>Krylov</Solver_Form>

  <!-- Sn, Pn, AngularCFE or AngularDFE -->
  <Angular_Method>Pn</Angular_Method>

  <!-- SelfAdjoint, LeastSquares, EvenOddParity, EvenParity, OddParity or FirstOrder -->
  <Krylov_Transport_Method>FirstOrder</Krylov_Transport_Method>

  <!-- SpatialCFE or SpatialDFE -->
  <SpatialFE_Method>SpatialDFE</SpatialFE_Method>

  <!-- Belos, MueLu or Direct -->
  <Linear_Solver>Direct</Linear_Solver>

  <!-- high, medium, low or none -->
  <Verbosity>medium</Verbosity>

</Solver>

<Solver name="Sn-FirstOrderKrylov">
  <Solver_Form>Krylov</Solver_Form>
```

```
    <!-- Sn, Pn, AngularCFE or AngularDFE -->
    <Angular_Method>Sn</Angular_Method>

    <!-- SelfAdjoint, LeastSquares, EvenOddParity, EvenParity, OddParity or FirstOrder -->
    <Krylov_Transport_Method>FirstOrder</Krylov_Transport_Method>

    <!-- SpatialCFE or SpatialDFE -->
    <SpatialFE_Method>SpatialDFE</SpatialFE_Method>

    <!-- Belos, MueLu or Direct -->
    <Linear_Solver>Belos</Linear_Solver>

    <!-- CG or GMRES -->
    <Iterative_Method>GMRES</Iterative_Method>

    <Maximum_Number_Iterations>100</Maximum_Number_Iterations>
    <Convergence_Tolerance>1.e-2</Convergence_Tolerance>
    <Krylov_Subspace_Size>100</Krylov_Subspace_Size>

    <!-- high, medium, low or none -->
    <Verbosity>medium</Verbosity>

    <!-- None, MultiLevel, IncompleteFactorization -->
    <Preconditioner>None</Preconditioner>

  </Solver>

</Solvers>

<Solver_Assignment explicit="true">
  <Solver_By_Group_Range>1stOrder 1 4</Solver_By_Group_Range>
  <!--Solver_By_Group_Range>Sn-LS 6 10</Solver_By_Group_Range-->
  <Solver_By_Group>
    <Group index="5">Pn-Direct</Group>
    <Group index="6">1stOrder</Group>
    <Group index="7">Sn-FirstOrderKrylov</Group>
    <Group index="8">1stOrder-TSA-P0</Group>
    <Group index="9">1stOrder-TSA-P0</Group>
    <Group index="10">Sn-LS</Group>
  </Solver_By_Group>
</Solver_Assignment>

<Materials enable="true">
  <Material name="iron">
  </Material>
  <Material name="copper">
  </Material>
  <Material name="silver">
  </Material>
  <Material name="teflon">
  </Material>
</Materials>

<Material_Assignment enable="true">
  <ElementBlock name="block1">copper</ElementBlock>
  <ElementBlock name="block2">iron</ElementBlock>
  <ElementBlock name="block3">silver</ElementBlock>
  <ElementBlock name="block4">teflon</ElementBlock>
  <ElementBlock name="block5">copper</ElementBlock>
</Material_Assignment>

<Internal_Angular_Flux_Initialization_Options enable="false">
  <Scale_Factor>1</Scale_Factor>
  <Energy_Expansion enable="true">
    <Group index="1">1</Group>
    <Group index="2">1</Group>
    <Group index="3">1</Group>
    <Group index="4">1</Group>
    <Group index="5">1</Group>
  </Energy_Expansion>
```

```
  <Angle_Expansion enable="false">
    <!-- Angle index="1">1</Angle -->
  </Angle_Expansion>
  <Block_Expansion enable="true">
    <Block index="1">1</Block>
    <Block index="2">0</Block>
    <Block index="3">0</Block>
    <Block index="4">0</Block>
    <Block index="5">0</Block>
  </Block_Expansion>

</Internal_Angular_Flux_Initialization_Options>

<Fixed_Source_Options enable="true">
  <Source>
  </Source>
  <Source>
    <Scale_Factor>3.</Scale_Factor>
  </Source>
  <Source>
    <Scale_Factor>2.</Scale_Factor>
    <Energy_Expansion enable="true">
      <Group index="1">0.5</Group>
      <Group index="2">0.2</Group>
      <Group index="3">0.1</Group>
      <Group_Range>0.3 4 6</Group_Range>
    </Energy_Expansion>
  </Source>
  <Source>
    <Scale_Factor>8.</Scale_Factor>
    <Energy_Expansion enable="true">
      <Group index="1">2.5</Group>
      <Group index="6">3.5</Group>
    </Energy_Expansion>
    <Angle_Expansion enable="true">
      <Angle index="1">3.2</Angle>
      <Angle_Range>3.3 2 3</Angle_Range>
    </Angle_Expansion>
    <Block_Expansion enable="true">
      <Block name="block4">5.</Block>
    </Block_Expansion>
  </Source>
</Fixed_Source_Options>

<Boundary_Source_Options enable="true">
  <Source>
  </Source>
  <Source>
    <Scale_Factor>3.</Scale_Factor>
  </Source>
  <Source>
    <Scale_Factor>2.</Scale_Factor>
    <Energy_Expansion enable="true">
      <Group index="1">0.5</Group>
      <Group index="2">0.2</Group>
      <Group index="3">0.1</Group>
      <Group_Range>0.3 4 6</Group_Range>
    </Energy_Expansion>
  </Source>
  <Source>
    <Scale_Factor>8.</Scale_Factor>
    <Energy_Expansion enable="true">
      <Group index="1">2.5</Group>
      <Group index="6">3.5</Group>
    </Energy_Expansion>
    <Angle_Expansion enable="true">
      <Angle index="1">3.2</Angle>
      <Angle_Range>3.3 2 3</Angle_Range>
    </Angle_Expansion>
    <SideSet_Expansion enable="true">
      <SideSet name="sideset5">5.</SideSet>
```

```
      </SideSet_Expansion>
    </Source>
  </Boundary_Source_Options>

</SCEPTRE_Input>
```

DISTRIBUTION

Click here, then press delete to remove this guidance statement.
Required. Must be on an odd-numbered page. SAND Reports submitted through R&A are automatically sent to the Technical Library; however, it still needs to be included on the distribution. Ensure a blank odd-numbered page is inserted prior to the back cover.

Email—Internal

Name	Org.	Sandia Email Address
Donald Bruss	01341	dbruss@sandia.gov
Clifton Drumm	01341	crdrumm@sandia.gov
Wesley Fan	01341	wcfan@sandia.gov
Shawn Pautz	01341	sdpautz@sandia.gov
Technical Library	01977	sanddocs@sandia.gov

This page left blank

This page left blank

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.