Profugus Documentation

Release 1.0

Thomas Evans **Steven Hamilton** **Stuart Slattery**

Nov 07, 2017

Contents

CHAPTER 1

Description

Profugus is an open-source mini-application (mini-app) for radiation transport and reactor applications. It contains the fundamental computational kernels used in the Exnihilo code suite from Oak Ridge National Laboratory. However, Exnihilo is production code with a substantial user base. Furthermore, Exnihilo is export controlled. This makes collaboration with computer scientists and computer engineers difficult. Profugus is designed to bridge that gap. By encapsulating the core numerical algorithms in an abbreviated code base that is open-source, computer scientists can analyze the algorithms and easily make code-architectural changes to test performance without compromising the production code values of Exnihilo.

Profugus is **not** meant to be production software with respect to problem analysis. The computational kernels in Profugus are designed to analyze *performance*, not *correctness*. Nonetheless, users of Profugus can setup and run problems with enough *real-world* features to be useful as proof-of-concept for actual production work.

Profugus is also used as a test-bed mini-app for the ASCR projects XPRESS and MCREX.

1.1 Profugus Development Team

Profugus contains the computational kernels from the Exnihilo code base. The core Exnihilo development team consists of the following scientists (listed alphabetically)

- Kevin Clarno <clarnokt@ornl.gov>
- Greg Davidson <davidsongg@ornl.gov>
- Tom Evans <evanstm@ornl.gov>
- Steven Hamilton <hamiltonsr@ornl.gov>
- Seth Johnson <johnsonsr@ornl.gov>
- Tara Pandya <pandyatm@ornl.gov>
- Stuart Slattery <slatterysr@ornl.gov>
- Rachel Slaybaugh <slaybaugh@berkeley.edu>

Profugus is developed and maintained by the following team:

- Tom Evans <evanstm@ornl.gov>
- Steven Hamilton <hamiltonsr@ornl.gov>
- Stuart Slattery <slatterysr@ornl.gov>

1.2 Profugus Packages

Profugus constains the following code mini-apps:

SPn Mini-app of Simplified Spherical Harmonics (`SPn`) computational kernel.

MC Mini-app of Monte Carlo computational kernel.

Documentation on the mini-apps (methods and execution) are given in *Running The Mini-Applications*. There are several support packages that are used by the mini-apps. These are:

Utils Utilities and testing framework used by other packages.

Matprop Data structures for storing material properties (cross-sections).

Profugus is designed to build and run with a minimum of dependencies. However, there are some requirements. The third-party software (TPLs) necessary to build Profugus is all open-source and freely available. The TPLs for Profugus are listed in the following table:

TPL	Required	Comments
TriBITS	Yes	TriBITS is the Trilinos build system
Trilinos	Yes	Trilinos is an open-source solvers library.
BLAS/LAPACK	Yes	Use vendor-specific implementation when possible. The default implementations can be found at netlib.
MPI	No	OpenMPI and MPICH are suggested options. Vendor-provided options will also work.
HDF5	No	Parallel HDF5 will allow output of solution fiilds. It is not needed for general problem execution.

CHAPTER 2

Building The Code

The most straightforward method for building Profugus is to use the scripts in `Profugus/install`. Profugus uses the `TriBITS build system`. This system is a set of package-based extensions to standard cmake. So, first you need to obtain *Trilinos* and *TriBITS* and put them in your top-level Profugus directory:

```
> cd Profugus
> git clone https://github.com/TriBITSPub/TriBITS.git
> ln -s $PATH_TO_TRILINOS .
```

The preferred mechanism for using the build scripts is to make a *target* directory where the build is to be performed:

```
> pwd
  /home/me
> mkdir debug
> cd debug
> mkdir target
> cd target
> pwd
  /home/me/debug/target
```

The `install` directory contains several example build scripts. General options for all platforms (which can be overridden at configure time) are specified in the `install/base.cmake`:

```
##---------------------------------------------------------------------------##
## CMAKE BASE FILE
##---------------------------------------------------------------------------##

# Default build all packages
SET(Profugus_ENABLE_Utils       ON CACHE BOOL "")
SET(Profugus_ENABLE_CudaUtils   ON CACHE BOOL "")
SET(Profugus_ENABLE_Matprop     ON CACHE BOOL "")
SET(Profugus_ENABLE_SPn         ON CACHE BOOL "")
SET(Profugus_ENABLE_MC          ON CACHE BOOL "")

# Turn on tests
SET(Profugus_ENABLE_TESTS ON CACHE BOOL "")
```

```
SET(Profugus_TEST_CATEGORIES "BASIC" CACHE STRING "")

# Turn on SS code and optional packages by default
SET(Profugus_ENABLE_ALL_FORWARD_DEP_PACKAGES OFF CACHE BOOL "")
SET(Profugus_ENABLE_ALL_OPTIONAL_PACKAGES    ON  CACHE BOOL "")
SET(Profugus_ENABLE_SECONDARY_STABLE_CODE    ON  CACHE BOOL "")

# Set explicit instantiation options
SET(Profugus_ENABLE_EXPLICIT_INSTANTIATION ON CACHE BOOL "")
SET(Teuchos_ENABLE_FLOAT OFF  CACHE BOOL "")
SET(Teuchos_ENABLE_COMPLEX OFF CACHE BOOL "")
SET(Tpetra_INST_FLOAT OFF CACHE BOOL "")
SET(Tpetra_INST_COMPLEX_FLOAT OFF CACHE BOOL "")
SET(Tpetra_INST_COMPLEX_DOUBLE OFF CACHE BOOL "")
SET(Thyra_ENABLE_EXPLICIT_INSTANTIATION OFF CACHE BOOL "")
SET(Stratimikos_ENABLE_EXPLICIT_INSTANTIATION OFF CACHE BOOL "")

# Up the max num procs
SET(MPI_EXEC_MAX_NUMPROCS 8 CACHE STRING "")

# Turn off binutils
SET(Teuchos_ENABLE_BinUtils OFF CACHE BOOL "")

# Turn off Zoltan2
SET(Profugus_ENABLE_Zoltan2 OFF CACHE BOOL "")

# Compiler options
SET(BUILD_SHARED_LIBS ON CACHE BOOL "")
SET(CMAKE_CXX_FLAGS "-std=c++11 -Wno-deprecated-declarations" CACHE STRING "")
SET(Profugus_ENABLE_CXX11 ON CACHE BOOL "")

# TriBITS stuff
SET(Profugus_ENABLE_INSTALL_CMAKE_CONFIG_FILES OFF CACHE BOOL "")
SET(Profugus_DEPS_XML_OUTPUT_FILE "" CACHE FILEPATH "")
```

By default, all of the packages inside of Profugus are turned on. Furthermore, *C++-11* is **required**. The default options specify the appropriate compiler flags for gcc. The tests are also turned on by default; to disable tests in any upstream package simply do not explicitly *ENABLE* that package. For example, to build the *SPn* package and all of its tests but only include required *source* from upstream packages, the user would specify:

```
SET(Profugus_ENABLE_SPn ON CACHE BOOL "")
```

In this case, only the pieces of *Utils* needed to build *SPn* are compiled. All tests can be turned off by setting **`Profugus_ENABLE_TESTS`** to **`OFF`**.

The `install` directory contains several build scripts that are all suffixed by the platform name. For example, to build on a Linux *x86_64* system the `install/cmake_x86_64.sh` script can be used:

```
#!/bin/sh
##---------------------------------------------------------------------------##
## CMAKE FOR X86_64
##---------------------------------------------------------------------------##

# CLEANUP
rm -rf CMakeCache.txt
rm -rf CMakeFiles

# SOURCE AND INSTALL
```

```
SOURCE=<SET_SOURCE_DIR>
INSTALL=<SET_INSTALL_DIR>

# BUILD OPTIONS
BUILD="DEBUG"
MPI="ON"

# TPL PATHS
HDF5_PATH="/vendors/hdf5_parallel"
MPI_PATH="/opt/openmpi/gcc/current"

##---------------------------------------------------------------------------##

cmake \
-DCMAKE_BUILD_TYPE:STRING="$BUILD" \
-DTPL_ENABLE_MPI:BOOL=$MPI \
-DCMAKE_INSTALL_PREFIX:PATH=$INSTALL \
\
-DMPI_BASE_DIR:PATH=$MPI_PATH \
\
-DTPL_ENABLE_HDF5:BOOL=ON \
-DHDF5_INCLUDE_DIRS:PATH=$HDF5_PATH/include \
-DHDF5_LIBRARY_DIRS:PATH=$HDF5_PATH/lib \
\
-DBLAS_LIBRARY_DIRS:PATH=/vendors/gcc/atlas/lib \
-DLAPACK_LIBRARY_DIRS:PATH=/vendors/gcc/atlas/lib \
-DBLAS_LIBRARY_NAMES:STRING="f77blas;cblas;atlas" \
-DLAPACK_LIBRARY_NAMES:STRING="lapack" \
\
-DProfugus_CONFIGURE_OPTIONS_FILE:FILEPATH="${SOURCE}/install/base.cmake" \
-DProfugus_ASSERT_MISSING_PACKAGES:BOOL=OFF \
\
${SOURCE}

##---------------------------------------------------------------------------##
## end of cmake_x86_64.sh
##---------------------------------------------------------------------------##
```

The source and install locations must be set. Also, to enable a optimized build set **BUILD** to **RELEASE**. Adjust the paths and libraries for LAPACK to fit your platform. The example assumes that the ATLAS LAPACK is available. Any standard LAPACK distribution will work. HDF5 is **not** required, to build/run/test the applications; however, problem output will be severely curtailed if a parallel HDF5 option is not provided. If HDF5 is not available, setting:

```
-DTPL_ENABLE_HDF5:BOOL=OFF \
```

will disable HDF5.

To complete the configuration, execute this script inside the *target* directory and then make/test/install:

```
> pwd
  /home/me/debug/target
> sh /home/me/Profugus/install/cmake_x86_64.sh
> make -j 8
> ctest -j 8
> make -j 8 install
```

CHAPTER 3

Running The Mini-Applications

3.1 Running the SPn Mini-Application

The `SPn` mini-app is compiled into the executable **`xspn`** that has the following options:

`-i`
: Load the subsequent xml input file.

The `SPn` mini-app solves the SP_N form of the neutron transport equation. The equations it solves are described in `in this technical note`.

Inputs are xml-driven. To run the application do the following:

```
> mpirun -np 1 ./xspn -i inf_med.xml
```

3.1.1 Infinite-Medium Problem Example

In this case `inf_med.xml` is the xml input file. We can walk through this case; it is one of the example inputs in `packages/SPn/examples`:

```
<?xml version='1.0' encoding='ASCII'?>
<ParameterList name="profugus SPN input">
  <ParameterList name="CORE">
    <Parameter name="axial list" type="Array(string)" value="{core}"/>
    <Parameter name="axial height" type="Array(double)" value="{ 1.00000e+01}"/>
    <Parameter name="core" type="TwoDArray(int)" value="1x1:{0}"/>
  </ParameterList>
  <ParameterList name="ASSEMBLIES">
    <Parameter name="pin pitch" type="double" value="10.0"/>
    <Parameter name="assembly list" type="Array(string)" value="{assembly}"/>
    <Parameter name="assembly" type="TwoDArray(int)" value="1x1:{0}"/>
  </ParameterList>
  <ParameterList name="SOURCE">
    <Parameter name="source list" type="Array(string)" value="{uniform}"/>
```

```
    <Parameter name="source map" type="TwoDArray(int)" value="1x1:{0}"/>
    <Parameter name="axial source" type="Array(double)" value="{ 1.00000e+00}"/>
    <ParameterList name="uniform">
      <Parameter name="strength" type="TwoDArray(double)" value="1x1:{ 1.00000e+00}"/>
      <Parameter name="shape" type="Array(double)" value="{ 1.00000e+00}"/>
    </ParameterList>
  </ParameterList>
  <ParameterList name="MATERIAL">
    <Parameter name="xs library" type="string" value="xs_1G.xml"/>
    <Parameter name="mat list" type="Array(string)" value="{scatterer}"/>
  </ParameterList>
  <ParameterList name="PROBLEM">
    <Parameter name="radial mesh" type="int" value="10"/>
    <Parameter name="axial mesh" type="Array(int)" value="{10}"/>
    <Parameter name="symmetry" type="string" value="full"/>
    <Parameter name="Pn_order" type="int" value="0"/>
    <Parameter name="SPn_order" type="int" value="1"/>
    <Parameter name="problem_name" type="string" value="inf_med"/>
  </ParameterList>
</ParameterList>
```

This is an input of a 1-group infinite-medium problem that has an analytical solution. Integrating the Boltzmann transport equation over energy and angle gives the following balance equation:

$$\nabla \cdot \mathbf{J} + (\sigma_\mathrm{t} - \sigma_\mathrm{s})\phi = Q \tag{3.1}$$

In an infinite medium (all reflecting boundaries with a uniform source and material everywhere), the divergence of the current vanishes and the solution for ϕ is simply

$$\phi = \frac{Q}{(\sigma_\mathrm{t} - \sigma_\mathrm{s})} \tag{3.2}$$

The geometric model for `SPn` is a simplified input for nuclear reactor problems (although it can be used to describe other types of problems). In this application area radial dimensions are *xy** and axial dimensions are z. A core is generally defined radially by 2D maps of assemblies and pin-cells and axially by 1D dimensions. In this model a **CORE** is the outermost object. A **CORE** contains **ASSEMBLIES**. The **MATERIALS** are referenced by the **ASSEMBLIES**. For fixed-source problems a **SOURCE** may be defined at the **CORE** level. Finally, a set of problem parameters that dictates how the code is run is defined in **PROBLEM**. These are summarized below:

CORE Describes the outer geometric configuration of the problem. A core defines the axial (z-axis) definitions and a 2D assembly map.

ASSEMBLIES Individual assembly types are defined in this block. Each assembly is an $N \times M$ array of *pin-cells*. Each pin cell contains a single material, but can be descretized by a finer computational mesh grid.

SOURCE Describes fixed sources. When this option is present, `SPn` is run in fixed-source mode; otherwise, an eigenvalue problem is run.

MATERIAL Specifies the xml-file containing the material cross sections and the list of materials referenced in the **ASSEMBLIES** block.

PROBLEM Determines all problem solution parameters including computational mesh discretization, SPN-order of the problem (1, 3, 5, 7), Pn-order of the scattering expansion, boundary conditions, and solver parameters. Note that solver parameters can be passed to directly to Trilinos solvers by describing the appropriate parameters as shown below.

Details on how the problem parameters are used to set up a problem can be ascertained by perusing the well-documented source code in `the Problem_Builder class` (this is a mini-app after all, so we do not consider it unreasonable to look at code!).

Note: All of the 2D arrays in the xml-input are COLUMN-MAJOR (FORTRAN) ordered. In other words, the *i*-indices of an array cycle fastest. Thus, 2D arrays are stored internally as arrays with dimension `[N_j][N_i]`, reflecting the fact that C arrays are stored ROW-MAJOR.

Conversely, the scattering cross sections in the cross section xml files are ROW-MAJOR ordered. Thus, the 2D arrays have dimensions `[g][g']`.

This will become clean in the examples.

Let's us now step through the input step-by-step. The **CORE** block for this problem defines the following parameters:

```
<Parameter name="axial list" type="Array(string)" value="{core}"/>
<Parameter name="axial height" type="Array(double)" value="{ 1.00000e+01}"/>
<Parameter name="core" type="TwoDArray(int)" value="1x1:{0}"/>
```

This first parameter defines a list of axial core maps. The second parameter defines the height of each axial level and should have the same number of entries as the `axial list` parameter. The final set of parameters are the core maps listed in hte `axial list` array. In this case there is a single axial core map named `core` (the names can be unique). The `core` map specifies a core containing a single assembly. This indices in the core map are ordered $[0, \ldots, N)$ and refer to assemblies defined in the **ASSEMBLIES** block `assembly list` parameter. This brings us to the parameters defined in the **ASSEMBLIES** block:

```
<Parameter name="pin pitch" type="double" value="10.0"/>
<Parameter name="assembly list" type="Array(string)" value="{assembly}"/>
<Parameter name="assembly" type="TwoDArray(int)" value="1x1:{0}"/>
```

The `pin pitch` parameter gives the width of each (square) pin-cell. The `assembly list` parameter gives the list of assembly types. As in the **CORE** block, the final set of parameters are the 2D assembly maps for each assembly defined in the `assembly list`. Every assembly must have the same dimensions, but they can be composed of different materials. Each assembly map refers to a material defined in the **MATERIAL** block `mat list` parameter and is ordered $[0, \ldots, N)$. So, this block defines a single assembly type that is 10×10 cm in radial width that contains a single pin-cell with material 0. Examining the 2 parameters in the **MATERIAL** block:

```
<Parameter name="xs library" type="string" value="xs_1G.xml"/>
<Parameter name="mat list" type="Array(string)" value="{scatterer}"/>
```

We see from the `mat list` parameter that material 0 corresponds to the material `scatterer`. This name refers to a material defined in the cross-section file `xs_1G.xml` that is loaded based on the value of the `xs library` parameter. The cross section file, `xs_1G.xml`, containst the following data:

```
<?xml version='1.0' encoding='ASCII'?>
<ParameterList name="cross sections">
  <Parameter name="num groups" type="int" value="1"/>
  <Parameter name="pn order" type="int" value="0"/>
  <Parameter name="group v" type="Array(double)" value="{ 4.373937e+09}"/>
  <ParameterList name="void">
    <Parameter name="sigma_t" type="Array(double)" value="{ 0.00000e+00}"/>
    <Parameter name="sigma_s0" type="TwoDArray(double)" value="1x1:{ 0.00000e+00}"/>
  </ParameterList>
  <ParameterList name="absorber">
    <Parameter name="sigma_t" type="Array(double)" value="{ 1.00000e+00}"/>
    <Parameter name="sigma_s0" type="TwoDArray(double)" value="1x1:{ 0.00000e+00}"/>
  </ParameterList>
  <ParameterList name="scatterer">
    <Parameter name="sigma_t" type="Array(double)" value="{ 1.00000e+00}"/>
    <Parameter name="sigma_s0" type="TwoDArray(double)" value="1x1:{ 9.00000e-01}"/>
```

```
  </ParameterList>
  <ParameterList name="fissionable">
    <Parameter name="sigma_t" type="Array(double)" value="{ 1.00000e+00}"/>
    <Parameter name="nu_sigma_f" type="Array(double)" value="{ 1.20000e+00}"/>
    <Parameter name="sigma_f" type="Array(double)" value="{ 5.00000e-01}"/>
    <Parameter name="chi" type="Array(double)" value="{ 1.00000e+00}"/>
    <Parameter name="sigma_s0" type="TwoDArray(double)" value="1x1:{ 2.00000e-01}"/>
  </ParameterList>
</ParameterList>
```

Thus, the geometric description of this problem is a $10 \times 10 \times 10$ cm box containing the material `scatterer` that has a total cross section of 1.0 and a scattering cross section of 0.9 (resulting in an aborption/removal cross section of 0.1).

To make this a fixed-source problem, the **SOURCE** block must be present:

```
<Parameter name="source list" type="Array(string)" value="{uniform}"/>
<Parameter name="source map" type="TwoDArray(int)" value="1x1:{0}"/>
<Parameter name="axial source" type="Array(double)" value="{ 1.00000e+00}"/>
<ParameterList name="uniform">
  <Parameter name="strength" type="TwoDArray(double)" value="1x1:{ 1.00000e+00}"/>
  <Parameter name="shape" type="Array(double)" value="{ 1.00000e+00}"/>
</ParameterList>
```

Here, the `source list` parameter gives the list of sources. The `source map` parameter has the same dimensions as the core maps and gives the source index that resides in each core location. A value of -1 indicates that there is no source in a given location. A value in the range $[0, \ldots, N)$ refers to the source indicated by the `source list`. The `axial source` parameter has the same number of entries as the `axial list` and `axial height` parameters in the **CORE** block. The `axial source` is a multiplier that is applied to the source at each axial level. Each radial source is defined in its own sublist. In this case, there is a single source, `uniform`. Each source sublist contains a `strength` and `shape` parameter. The `strength` parameter gives a $N \times M$ 2D axial array of the source strength in each pin-cell. The `shape` is dimensioned by the number of groups in the `xs library` parameter `num groups`. It gives the energy-spectral shape of the source. Thus, in any pin-cell, the group source is given by the product of the shape and strength:

$$q^g_{\text{ext}\,i,j} = \text{shape[g]} \times \text{strength[j][i]} \tag{3.3}$$

And, as described above, the `strength` map is COLUMN-MAJOR ordered in the xml file.

Finally, the **PROBLEM** database provides entries that control the problem setup and solver. Many of these entries will be set by default, although all can be overrided. In this example, we only set a few parameters:

```
<Parameter name="radial mesh" type="int" value="10"/>
<Parameter name="axial mesh" type="Array(int)" value="{10}"/>
<Parameter name="symmetry" type="string" value="full"/>
<Parameter name="Pn_order" type="int" value="0"/>
<Parameter name="SPn_order" type="int" value="1"/>
<Parameter name="problem_name" type="string" value="inf_med"/>
```

Here the `radial mesh` indicates that each pin-cell is meshed 10×10. The `axial mesh` value is an array giving the number of computational mesh cells in each axial level. The `symmetry` parameter tells that mesh generator to use the full problem description and not apply any symmetry conditions. The `Pn_order` parameter gives the scattering order for the solver; it must be less than or equal to the `pn order` specified in the cross section library file. The `SPn_order` gives the order of the SP_N approximation. Finally, the `problem_name` gives the base name that will be used for all output files. We have not specified any specific solver options, so the defaults will be used.

The final specification of this problem is $10 \times 10 \times 10$ cm box with computational mesh cells with dimension $1 \times 1 \times 1$ cm resulting in 1000 total cells ($10 \times 10 \times 10$). There is a uniform 1 particle/cc source throughout the box. The box

has a uniform material of `scatterer` as defined in the `xs_1G.xml` file. The solver will run a `SP1` calculation with `P0` scattering.

Note: The only symmetry option currently supported is `full`; however, ``qtr`` symmetry will be added for 1/4 symmetry.

The outputs for this problem are contained in the `examples` directory. Automatically, the code will output a final problem xml file so that the user can see what defaults were added. In this case, the output xml file is stored in `inf_med_db.xml`:

```
<ParameterList name="inf_med-PROBLEM">
  <Parameter docString="" id="0" isDefault="false" isUsed="true" name="radial mesh"␣
↪type="int" value="10"/>
  <Parameter docString="" id="1" isDefault="false" isUsed="true" name="axial mesh"␣
↪type="Array(int)" value="{10}"/>
  <Parameter docString="" id="2" isDefault="false" isUsed="true" name="symmetry" type=
↪"string" value="full"/>
  <Parameter docString="" id="3" isDefault="false" isUsed="true" name="Pn_order" type=
↪"int" value="0"/>
  <Parameter docString="" id="4" isDefault="false" isUsed="true" name="SPn_order"␣
↪type="int" value="1"/>
  <Parameter docString="" id="5" isDefault="false" isUsed="true" name="problem_name"␣
↪type="string" value="inf_med"/>
  <Parameter docString="" id="6" isDefault="false" isUsed="true" name="num_cells_i"␣
↪type="int" value="10"/>
  <Parameter docString="" id="7" isDefault="false" isUsed="true" name="delta_x" type=
↪"double" value="1.0000000000000000e+01"/>
  <Parameter docString="" id="8" isDefault="false" isUsed="true" name="num_cells_j"␣
↪type="int" value="10"/>
  <Parameter docString="" id="9" isDefault="false" isUsed="true" name="delta_y" type=
↪"double" value="1.0000000000000000e+01"/>
  <Parameter docString="" id="10" isDefault="false" isUsed="true" name="z_edges" type=
↪"Array(double)" value="{0.0000000000000000e+00, 1.0000000000000000e+00, 2.
↪0000000000000000e+00, 3.0000000000000000e+00, 4.0000000000000000e+00, 5.
↪0000000000000000e+00, 6.0000000000000000e+00, 7.0000000000000000e+00, 8.
↪0000000000000000e+00, 9.0000000000000000e+00, 1.0000000000000000e+01}"/>
  <Parameter docString="" id="11" isDefault="false" isUsed="true" name="num_blocks_i"␣
↪type="int" value="1"/>
  <Parameter docString="" id="12" isDefault="false" isUsed="true" name="num_blocks_j"␣
↪type="int" value="1"/>
  <Parameter docString="" id="13" isDefault="false" isUsed="true" name="num_z_blocks"␣
↪type="int" value="1"/>
  <Parameter docString="" id="14" isDefault="false" isUsed="true" name="num_sets"␣
↪type="int" value="1"/>
  <Parameter docString="" id="15" isDefault="false" isUsed="true" name="dimension"␣
↪type="int" value="3"/>
  <Parameter docString="" id="16" isDefault="true" isUsed="true" name="g_first" type=
↪"int" value="0"/>
  <Parameter docString="" id="17" isDefault="true" isUsed="true" name="g_last" type=
↪"int" value="0"/>
  <Parameter docString="" id="18" isDefault="false" isUsed="true" name="problem_type"␣
↪type="string" value="fixed"/>
  <Parameter docString="" id="19" isDefault="true" isUsed="true" name="solver_type"␣
↪type="string" value="stratimikos"/>
  <Parameter docString="" id="20" isDefault="true" isUsed="true" name="tolerance"␣
↪type="double" value="9.9999999999999955e-07"/>
  <Parameter docString="" id="21" isDefault="true" isUsed="true" name="max_itr" type=
↪"int" value="100"/>
```

```
  <ParameterList id="67" name="Stratimikos">
    <Parameter docString="" id="22" isDefault="false" isUsed="true" name=
↪"Preconditioner Type" type="string" value="None"/>
    <Parameter docString="" id="23" isDefault="false" isUsed="true" name="Linear␣
↪Solver Type" type="string" value="AztecOO"/>
    <Parameter docString="" id="24" isDefault="true" isUsed="true" name="Enable␣
↪Delayed Solver Construction" type="bool" value="false"/>
    <ParameterList id="66" name="Linear Solver Types">
      <ParameterList id="65" name="AztecOO">
        <Parameter docString="" id="25" isDefault="true" isUsed="true" name="Output␣
↪Every RHS" type="bool" value="false"/>
        <ParameterList id="43" name="Forward Solve">
          <Parameter docString="" id="26" isDefault="true" isUsed="true" name="Max␣
↪Iterations" type="int" value="400"/>
          <Parameter docString="" id="27" isDefault="true" isUsed="true" name=
↪"Tolerance" type="double" value="9.9999999999999955e-07"/>
          <ParameterList id="42" name="AztecOO Settings">
            <Parameter docString="" id="28" isDefault="true" isUsed="true" name=
↪"Aztec Preconditioner" type="string" validatorId="0" value="none"/>
            <Parameter docString="" id="29" isDefault="true" isUsed="true" name=
↪"Aztec Solver" type="string" validatorId="1" value="GMRES"/>
            <Parameter docString="" id="30" isDefault="true" isUsed="true" name=
↪"Overlap" type="int" validatorId="2" value="0"/>
            <Parameter docString="" id="31" isDefault="true" isUsed="true" name=
↪"Graph Fill" type="int" validatorId="3" value="0"/>
            <Parameter docString="" id="32" isDefault="true" isUsed="true" name="Drop␣
↪Tolerance" type="double" validatorId="4" value="0.00000000000000000e+00"/>
            <Parameter docString="" id="33" isDefault="true" isUsed="true" name="Fill␣
↪Factor" type="double" validatorId="5" value="1.00000000000000000e+00"/>
            <Parameter docString="" id="34" isDefault="true" isUsed="true" name="Steps
↪" type="int" validatorId="6" value="3"/>
            <Parameter docString="" id="35" isDefault="true" isUsed="true" name=
↪"Polynomial Order" type="int" validatorId="7" value="3"/>
            <Parameter docString="" id="36" isDefault="true" isUsed="true" name="RCM␣
↪Reordering" type="string" validatorId="8" value="Disabled"/>
            <Parameter docString="" id="37" isDefault="true" isUsed="true" name=
↪"Orthogonalization" type="string" validatorId="9" value="Classical"/>
            <Parameter docString="" id="38" isDefault="true" isUsed="true" name="Size␣
↪of Krylov Subspace" type="int" validatorId="10" value="300"/>
            <Parameter docString="" id="39" isDefault="true" isUsed="true" name=
↪"Convergence Test" type="string" validatorId="11" value="r0"/>
            <Parameter docString="" id="40" isDefault="true" isUsed="true" name="Ill-
↪Conditioning Threshold" type="double" validatorId="12" value="1.
↪00000000000000000e+11"/>
            <Parameter docString="" id="41" isDefault="true" isUsed="true" name=
↪"Output Frequency" type="int" validatorId="13" value="0"/>
          </ParameterList>
        </ParameterList>
        <ParameterList id="61" name="Adjoint Solve">
          <Parameter docString="" id="44" isDefault="true" isUsed="true" name="Max␣
↪Iterations" type="int" value="400"/>
          <Parameter docString="" id="45" isDefault="true" isUsed="true" name=
↪"Tolerance" type="double" value="9.9999999999999955e-07"/>
          <ParameterList id="60" name="AztecOO Settings">
            <Parameter docString="" id="46" isDefault="true" isUsed="true" name=
↪"Aztec Solver" type="string" validatorId="1" value="GMRES"/>
            <Parameter docString="" id="47" isDefault="true" isUsed="true" name=
↪"Aztec Preconditioner" type="string" validatorId="0" value="ilu"/>
```

```
            <Parameter docString="" id="48" isDefault="true" isUsed="true" name=
↪"Overlap" type="int" validatorId="2" value="0"/>
            <Parameter docString="" id="49" isDefault="true" isUsed="true" name=
↪"Graph Fill" type="int" validatorId="3" value="0"/>
            <Parameter docString="" id="50" isDefault="true" isUsed="true" name="Drop␣
↪Tolerance" type="double" validatorId="4" value="0.0000000000000000e+00"/>
            <Parameter docString="" id="51" isDefault="true" isUsed="true" name="Fill␣
↪Factor" type="double" validatorId="5" value="1.0000000000000000e+00"/>
            <Parameter docString="" id="52" isDefault="true" isUsed="true" name="Steps
↪" type="int" validatorId="6" value="3"/>
            <Parameter docString="" id="53" isDefault="true" isUsed="true" name=
↪"Polynomial Order" type="int" validatorId="7" value="3"/>
            <Parameter docString="" id="54" isDefault="true" isUsed="true" name="RCM␣
↪Reordering" type="string" validatorId="8" value="Disabled"/>
            <Parameter docString="" id="55" isDefault="true" isUsed="true" name=
↪"Orthogonalization" type="string" validatorId="9" value="Classical"/>
            <Parameter docString="" id="56" isDefault="true" isUsed="true" name="Size␣
↪of Krylov Subspace" type="int" validatorId="10" value="300"/>
            <Parameter docString="" id="57" isDefault="true" isUsed="true" name=
↪"Convergence Test" type="string" validatorId="11" value="r0"/>
            <Parameter docString="" id="58" isDefault="true" isUsed="true" name="Ill-
↪Conditioning Threshold" type="double" validatorId="12" value="1.
↪0000000000000000e+11"/>
            <Parameter docString="" id="59" isDefault="true" isUsed="true" name=
↪"Output Frequency" type="int" validatorId="13" value="0"/>
          </ParameterList>
        </ParameterList>
        <ParameterList id="64" name="VerboseObject">
          <Parameter docString="" id="62" isDefault="true" isUsed="true" name="Output␣
↪File" type="string" value="none"/>
          <Parameter docString="" id="63" isDefault="true" isUsed="true" name=
↪"Verbosity Level" type="string" value="default"/>
        </ParameterList>
      </ParameterList>
    </ParameterList>
  </ParameterList>
  <Parameter docString="" id="68" isDefault="true" isUsed="true" name="eqn_type" type=
↪"string" value="fv"/>
  <Parameter docString="" id="69" isDefault="false" isUsed="true" name="boundary"␣
↪type="string" value="reflect"/>
  <ParameterList id="71" name="boundary_db">
    <Parameter docString="" id="70" isDefault="false" isUsed="true" name="reflect"␣
↪type="Array(int)" value="{1, 1, 1, 1, 1, 1}"/>
  </ParameterList>
  <Validators>
    <Validator allowDouble="true" allowInt="true" allowString="true" prefferedType=
↪"int" type="anynumberValidator" validatorId="3"/>
    <Validator caseSensitive="true" defaultParameterName="Aztec Solver" integralValue=
↪"int" type="StringIntegralValidator(int)" validatorId="1">
      <String integralValue="0" stringValue="CG"/>
      <String integralValue="1" stringValue="GMRES"/>
      <String integralValue="2" stringValue="CGS"/>
      <String integralValue="3" stringValue="TFQMR"/>
      <String integralValue="4" stringValue="BiCGStab"/>
      <String integralValue="10" stringValue="LU"/>
      <String integralValue="7" stringValue="GMRESR"/>
      <String integralValue="8" stringValue="FixedPoint"/>
    </Validator>
```

```
    <Validator caseSensitive="true" defaultParameterName="Aztec Preconditioner"␣
↪integralValue="(anonymous namespace)::EAztecPreconditioner" type=
↪"StringIntegralValidator((anonymous namespace)::EAztecPreconditioner)" validatorId=
↪"0">
      <String integralValue="0" stringValue="none"/>
      <String integralValue="1" stringValue="ilu"/>
      <String integralValue="2" stringValue="ilut"/>
      <String integralValue="3" stringValue="Jacobi"/>
      <String integralValue="4" stringValue="Symmetric Gauss-Seidel"/>
      <String integralValue="5" stringValue="Polynomial"/>
      <String integralValue="6" stringValue="Least-squares Polynomial"/>
    </Validator>
    <Validator allowDouble="true" allowInt="true" allowString="true" prefferedType=
↪"double" type="anynumberValidator" validatorId="4"/>
    <Validator allowDouble="true" allowInt="true" allowString="true" prefferedType=
↪"int" type="anynumberValidator" validatorId="2"/>
    <Validator allowDouble="true" allowInt="true" allowString="true" prefferedType=
↪"double" type="anynumberValidator" validatorId="5"/>
    <Validator allowDouble="true" allowInt="true" allowString="true" prefferedType=
↪"int" type="anynumberValidator" validatorId="6"/>
    <Validator allowDouble="true" allowInt="true" allowString="true" prefferedType=
↪"int" type="anynumberValidator" validatorId="7"/>
    <Validator caseSensitive="true" defaultParameterName="RCM Reordering"␣
↪integralValue="int" type="StringIntegralValidator(int)" validatorId="8">
      <String integralValue="1" stringValue="Enabled"/>
      <String integralValue="0" stringValue="Disabled"/>
    </Validator>
    <Validator allowDouble="true" allowInt="true" allowString="true" prefferedType=
↪"double" type="anynumberValidator" validatorId="12"/>
    <Validator caseSensitive="true" defaultParameterName="Orthogonalization"␣
↪integralValue="int" type="StringIntegralValidator(int)" validatorId="9">
      <String integralValue="0" stringValue="Classical"/>
      <String integralValue="1" stringValue="Modified"/>
    </Validator>
    <Validator allowDouble="true" allowInt="true" allowString="true" prefferedType=
↪"int" type="anynumberValidator" validatorId="10"/>
    <Validator allowDouble="true" allowInt="true" allowString="true" prefferedType=
↪"int" type="anynumberValidator" validatorId="13"/>
    <Validator caseSensitive="true" defaultParameterName="Convergence Test"␣
↪integralValue="int" type="StringIntegralValidator(int)" validatorId="11">
      <String integralValue="0" stringValue="r0"/>
      <String integralValue="1" stringValue="rhs"/>
      <String integralValue="2" stringValue="Anorm"/>
      <String integralValue="6" stringValue="no scaling"/>
      <String integralValue="3" stringValue="sol"/>
    </Validator>
  </Validators>
</ParameterList>
```

Perusing this database, we see the default solver options that were set. These can be overriden by adding a `Stratimikos` parameterlist to the input xml. Other items of note are:

```
<Parameter  name="num_blocks_i" type="int" value="1"/>
<Parameter  name="num_blocks_j" type="int" value="1"/>
<Parameter  name="g_first" type="int" value="0"/>
<Parameter  name="g_last" type="int" value="0"/>
```

The num_blocks parameters allow the user to specify a parallel decomposition. The total number of processes is:

$$N_p = N_i \times N_j$$

Note: The SPn mini-app currenly only supports *xy* decompositions. This is an historical requirement so that the SPn matches a similar decomposition in the |Exnihilo| production code for another physics model. It is not a *true* restriction.

The g_first and g_last parameters allow the user to specify a range of groups to run over. For example, if a 23-group cross section set is loaded, but the user is only interested in the solution in groups 0 and 1 set:

```
<Parameter  name="g_first" type="int" value="0"/>
<Parameter  name="g_last" type="int" value="1"/>
```

If HDF5 is available, the group-wise fluxes are output in the file inf_med_output.h5. From (3.2) the solution to this problem should be $\phi = 10.0$ everywhere. Using **h5dump** on the output file yields:

```
HDF5 "SPn_output.h5" {
GROUP "/" {
   GROUP "fluxes" {
      DATASET "group_0" {
         DATATYPE  H5T_IEEE_F64LE
         DATASPACE  SIMPLE { ( 10, 10, 10 ) / ( 10, 10, 10 ) }
         DATA {
         (0,0,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (0,1,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (0,2,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (0,3,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (0,4,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (0,5,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (0,6,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (0,7,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (0,8,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (0,9,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (1,0,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (1,1,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (1,2,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (1,3,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (1,4,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (1,5,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (1,6,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (1,7,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (1,8,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (1,9,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (2,0,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (2,1,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (2,2,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (2,3,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (2,4,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (2,5,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (2,6,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (2,7,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (2,8,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (2,9,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (3,0,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
         (3,1,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
```

```
(3,2,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(3,3,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(3,4,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(3,5,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(3,6,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(3,7,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(3,8,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(3,9,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(4,0,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(4,1,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(4,2,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(4,3,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(4,4,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(4,5,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(4,6,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(4,7,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(4,8,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(4,9,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(5,0,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(5,1,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(5,2,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(5,3,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(5,4,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(5,5,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(5,6,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(5,7,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(5,8,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(5,9,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(6,0,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(6,1,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(6,2,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(6,3,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(6,4,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(6,5,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(6,6,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(6,7,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(6,8,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(6,9,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(7,0,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(7,1,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(7,2,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(7,3,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(7,4,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(7,5,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(7,6,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(7,7,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(7,8,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(7,9,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(8,0,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(8,1,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(8,2,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(8,3,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(8,4,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(8,5,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(8,6,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(8,7,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(8,8,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
(8,9,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
```

```
          (9,0,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
          (9,1,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
          (9,2,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
          (9,3,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
          (9,4,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
          (9,5,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
          (9,6,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
          (9,7,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
          (9,8,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
          (9,9,0): 10, 10, 10, 10, 10, 10, 10, 10, 10, 10
          }
          ATTRIBUTE "data_order" {
             DATATYPE  H5T_STD_I32LE
             DATASPACE  SIMPLE { ( 1 ) / ( 1 ) }
             DATA {
             (0): 0
             }
          }
       }
    }
}
}
```

Thus, the correct solution is attained for this problem.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Index

Symbols

X