The Intrinsic Source Constructor (ISC) and MCNPTools Libraries

Clell J. (CJ) Solomon  Cameron R. Bates

Los Alamos National Laboratory
PO Box 1663
Los Alamos, NM 87545

28 Sept. 2016

LA-UR-16-27265
Outline

Intrinsic Source Constructor
  Overview
  Using MISC to Generate Source Terms for MCNP
  ISC Library Usage

MCNPTools
  Overview
  Widgets
  MCNPTools Accessor Classes
    Mctal Class
    Meshtal (mesh-tally type B) Class
    Ptrac Class
  Interactive MCNPTools Demo (time permitting)
ISC Overview

- ISC is a library for generating intrinsic radiation sources for inputs into radiation transport codes
- ISC is written in C++ and bound to Python (via SWIG)
- ISC uses a CMake build system to build the C++ library and utilities and a Python setuptools setup.py to build the Python extensions
- ISC has with it a single binary utility MISC—the MCNP Intrinsic Source Constructor—that will generate SDEF distributions
ISC Data Files

- ISC has three classes of data files
  1. natural abundance files—contains naturally occurring abundances of isotopes
  2. radioactive decay data files—contains decay mechanisms, branching ratios, and daughter isotopes
  3. particle emission data files—contains the particle emissions from the decay of an isotope

- The data files are C++ Boost Serialized versions of ISC data storage classes

- The data files are typically stored in XML formats that can be interrogated in an editor (if one knows what they are looking for)
To SZAs ... and Beyond

- To adequately capture decay channels, emissions, etc., ZA identifiers are insufficient.
- ISC uses SZA identifiers where S is the long-lived isomeric state number (not necessarily the isomer’s level number)

\[
SZA = S \times 1000000 + Z \times 1000 + A
\]

- For example, Pa-234m1 would be 1091234
Natural Abundance Files

- intended to facilitate usability
- allows a user to specify a natural ZA (i.e., A=000) and automatically expand it to isotopic ZAs
- Currently, the only available data set is the NIST natural abundances
Radioactive Decay Data Files

- This file contains information about decay mechanisms, branching ratios, and decay daughters of SZAs
- Currently, these files do not include any information about spontaneous fission products
- Available data sets include data from ENDF/VI (not recommended) and ENDF/VII.1 radioactive decay sublibrary
Particle Emission Data Files

- For each SZA, a file exists that has the emissions by particle type
- The emission files do not contain any information about coincident emissions
- Beta-particle spectra are reconstructed based on Fermi theory from end-point energies
- Watt spectrum parameters exist for many/most spontaneously fissioning isotopes (taken from the SOURCES 4C) data set
- The currently available set of particle emission data are from the ENDF/VI (not recommended) and ENDF/VII.1 radioactive decay sublibraries (RDSL)
Why ENDF7.1 and not ENDF6

- vast improvements were made to the ENDF RDSL from ENDF/VI to ENDF/VII.0 and important corrections made in ENDF/VII.1
- below is a comparison of dose measurements to simulation from a DU sphere
Aging Material

- ISC has built into it a Bateman solver
- Materials compositions can be specified, aged, reset, and aged again if desired
- Emission spectra can be generated for either or both the original material composition and the aged composition
MISC Overview

1. MISC is a standalone binary utility that generates SDEF source distributions for MCNP
2. MISC reads and input file with entries generally of the form

\[
\text{\langle keyword\rangle} = \text{\langle value\rangle}
\]

3. MISC has a User’s Manual, but an example input is given herein
4. MISC generates source distributions for a single particle type at a time, but each distribution may have discrete and continuous components
5. MISC outputs two files and output file summarizing the run and a source file that contains the MCNP SDEF distributions
Example MISC Input File (Natural U)

# lines beginning with '#' are comments
#
# specify output file
output=test001.out
#
# specify source distribution file
srcout=test001.src
#
# specify the natural abundance file
abundfile=nist.na.xml
#
# specify the decay data file
decayfile=endf7.dk.xml
#
# specify the default particle emission library
pelib=endf7
#
# specify the material isotopic description with
# SZAs (+/- = atom/mass fracs)
matspec= 92000 -1
#
# specify the material density
density=-18.0
#
# specify the aging of the material in seconds
age = 3.14e7
#
# specify the particle type for the source
particle=p
#
# specify vertical (v) or horizontal (h) formatted
# SDEF distributions
format=v
#
# specify that electron sources should be converted
# to photon sources via a thick-target bremsstrahlung
# approximation
estobrems=y
Example MISC Output File

USER INPUT MATERIAL SPECIFICATION:

<table>
<thead>
<tr>
<th>SZA</th>
<th>Isotope</th>
<th>Mass Frac</th>
</tr>
</thead>
<tbody>
<tr>
<td>92000</td>
<td>U-Nat</td>
<td>1.00000e+00</td>
</tr>
</tbody>
</table>

Material density: 1.80000e+01 g/cm³

The material was aged for 3.14000e+07 s using data from endf7.dk.xml

RESULTING MATERIAL SPECIFICATION:

<table>
<thead>
<tr>
<th>SZA</th>
<th>Isotope</th>
<th>Atom Frac</th>
<th>Mass Frac</th>
</tr>
</thead>
<tbody>
<tr>
<td>85215</td>
<td>At-215</td>
<td>1.91435e-35</td>
<td>1.72913e-35</td>
</tr>
<tr>
<td>85218</td>
<td>At-218</td>
<td>3.99948e-30</td>
<td>3.66309e-30</td>
</tr>
<tr>
<td>85219</td>
<td>At-219</td>
<td>4.92028e-30</td>
<td>4.52716e-30</td>
</tr>
<tr>
<td>86218</td>
<td>Rn-218</td>
<td>9.33212e-35</td>
<td>8.54709e-35</td>
</tr>
<tr>
<td>86219</td>
<td>Rn-219</td>
<td>3.29601e-25</td>
<td>3.03264e-25</td>
</tr>
<tr>
<td>86222</td>
<td>Rn-222</td>
<td>4.40017e-21</td>
<td>4.10792e-21</td>
</tr>
<tr>
<td>87223</td>
<td>Fr-223</td>
<td>1.93298e-24</td>
<td>1.81109e-24</td>
</tr>
<tr>
<td>88223</td>
<td>Ra-223</td>
<td>8.21964e-20</td>
<td>7.70130e-20</td>
</tr>
<tr>
<td>88226</td>
<td>Ra-226</td>
<td>6.93897e-16</td>
<td>6.58905e-16</td>
</tr>
<tr>
<td>89227</td>
<td>Ac-227</td>
<td>7.29169e-17</td>
<td>6.95469e-17</td>
</tr>
<tr>
<td>90227</td>
<td>Th-227</td>
<td>1.45707e-19</td>
<td>1.38973e-19</td>
</tr>
<tr>
<td>90230</td>
<td>Th-230</td>
<td>1.51702e-10</td>
<td>1.46606e-10</td>
</tr>
<tr>
<td>90231</td>
<td>Th-231</td>
<td>2.97954e-14</td>
<td>2.89201e-14</td>
</tr>
<tr>
<td>90234</td>
<td>Th-234</td>
<td>1.44078e-11</td>
<td>1.41666e-11</td>
</tr>
<tr>
<td>91231</td>
<td>Pa-231</td>
<td>7.02967e-12</td>
<td>6.82315e-12</td>
</tr>
<tr>
<td>91234</td>
<td>Pa-234</td>
<td>2.66978e-16</td>
<td>2.62508e-16</td>
</tr>
<tr>
<td>92234</td>
<td>U-234</td>
<td>5.40000e-05</td>
<td>5.30953e-05</td>
</tr>
<tr>
<td>92235</td>
<td>U-235</td>
<td>7.20400e-03</td>
<td>7.11366e-03</td>
</tr>
<tr>
<td>92238</td>
<td>U-238</td>
<td>9.92742e-01</td>
<td>9.92833e-01</td>
</tr>
<tr>
<td>1091234</td>
<td>Pa-234m1</td>
<td>4.81174e-16</td>
<td>4.73117e-16</td>
</tr>
</tbody>
</table>

Atom density: 4.55401e-02 a/b²cm

Mass density: 1.80000e+01 g/cm³

TOTAL PARTICLE EMISSION RATES

<table>
<thead>
<tr>
<th>Particle</th>
<th>cm⁻³ s⁻¹</th>
<th>g⁻¹ s⁻¹</th>
<th>Frac. Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>2.42704e-01</td>
<td>1.34835e-02</td>
<td>3.80112e-07</td>
</tr>
<tr>
<td>p</td>
<td>1.86172e+05</td>
<td>1.03429e+04</td>
<td>2.91574e-01</td>
</tr>
<tr>
<td>a</td>
<td>4.52334e+05</td>
<td>2.51297e+04</td>
<td>7.08425e-01</td>
</tr>
<tr>
<td>Total</td>
<td>6.38506e+05</td>
<td>3.54726e+04</td>
<td>1.00000e+00</td>
</tr>
</tbody>
</table>
Example MISC Source File

- The source emission rate normalization is given per unit volume or mass of the source.
- The source is broken into two pieces, the discrete and continuous spectra.
- The source can also be output as a function of isotope if desired.
- The starting source distribution number can be set in the input file.

```
c wgt should equal 1.86172e+05*VOL or 1.03429e+04*MASS
si001 s 2 3
sp001 d 5.89603e-01 4.10397e-01
#    si002    sp002
  l    d
2.29873e-03 3.35218e-11
2.29880e-03 3.26936e-21
2.29892e-03 4.66760e-19
2.29965e-03 2.08984e-14
2.37789e-03 7.33374e-14
.
.
. 3.09400e+00 1.45646e-14
3.14256e+00 4.59933e-14
3.14900e+00 3.29619e-15
3.16050e+00 4.07529e-14
3.18363e+00 5.09759e-14
#    si003    sp003
  h    d
0.00000e+00 0.00000e+00
2.00000e-02 2.89830e-01
4.00000e-02 1.37373e-01
6.00000e-02 8.78325e-02
8.00000e-02 6.35485e-02
1.00000e-01 4.92536e-02
.
.
. 4.28000e+00 3.38511e-22
4.30000e+00 1.56604e-22
4.32000e+00 5.99534e-23
4.34000e+00 1.66812e-23
4.36000e+00 2.36227e-24
```

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy’s NNSA
Interactive MISC example (Co-60)
# specify output file
output=co60.out
#
# specify source distribution file
srcout=co60.src
#
# specify the natural abundance file
abundfile=nist.na.xml
#
# specify the decay data file
decayfile=endf7.dk.xml
#
# specify the default particle emission library
pelib=endf7
#
# specify the material isotopic description with
# SZAs (+/- = atom/mass fracs)

matspec= 27060 1.0
#
# specify the material density
# normalize to 1 a/b-cm
density=1.0
#
# specify the aging of the material in seconds
# 1 hl = 5.271 a = 1.663355e+8 s
age=1.663355e+8
#
# specify the particle type for the source
particle=p
#
# specify vertical (v) or horizontal (h) formatted
# SDEF distributions
#
format=v
Examples of Using the ISC Library

- MISC is useful...but the true power of ISC is that YOU can write your own scripts/programs to do whatever you want with decay data
- the ISC library supports C++ with Python bindings
- examples of using the library in Python follow
ISC Example 1a: Loading Data Files

- this example (next slide) demonstrates how to load abundance, decay, and particle emission data files
- the natural abundance and decay data files load information for all SZAs
- the particle emission data is loaded on an SZA basis (loading all the data takes time)
ISC Example 1a: Loading Data Files Continued

```python
import os
import isc # import the isc module

# set the path to the ISC data
iscdata = os.getenv("ISCDATA")

# open an abundance data file and convert it to an abundance library
abund_file = isc.AbundanceFile(os.path.join(iscdata,"nist.na.xml"))
abund_lib = isc.AbundanceLib(abund_file)

# open a decay data file and convert it to a decay data library
decay_file = isc.DecayFile(os.path.join(iscdata,"endf7.dk.xml"))
decay_lib = isc.DecayLib(decay_file)

# open an emission file index (contains relative paths to emission data files)
emission_index = isc.EmissionFileIndex(os.path.join(iscdata,"endf7.idx.xml"))
# initialize an empty emission library
emission_lib = isc.EmissionLib()

# loop over all SZAs and import the emission data
# NOTE: one need not load everything, only the things you need
for sza in emission_index.GetSZAs():
    print("loading emission data for isotope of SZA {}").format(sza))
    emission_file = isc.EmissionFile( os.path.join(iscdata,emission_index.GetPath(sza)) )
    emission_lib.SetFromEmissionFile( emission_file )
```
ISC Example 1b: Loading Binary Data Files

- for convenience and speed, sets of abundance, data, and particle emission data are bundled into binary data files
- these binary data files can be used to load sets of the data
- this is much faster than the previous example, but it loads all the emission data
import os
import isc # import the isc module

# set the path to the ISC data
iscdata = os.getenv("ISCDATA")

# initialize an empty abundance lib
abund_lib = isc.AbundanceLib()

# initialize an empty decay lib
decay_lib = isc.DecayLib()

# initialize an empty emission library
emission_lib = isc.EmissionLib()

# load the libraries from the prebuilt binary file
isc.loadLibs(abund_lib, decay_lib, emission_lib, os.path.join(iscdata,"endf7.isc.bin"))
ISC Example 2: Querying Abundance Data

- This example (next slide) demonstrates how to get mass and natural abundance data out of the abundance library.
- First, a set of all U isotopes the library knows about is generated.
- Second, a loop is constructed over the set of all U isotopes and the mass and natural abundance is obtained.
- Last, if the abundance is greater than zero, the SZA is added to the list of naturally occurring U isotopes.
import os
import isc # import the isc module

# load the data
iscdataloc = os.getenv("ISCDATA")
abund_lib = isc.AbundanceLib()
decay_lib = isc.DecayLib()
emission_lib = isc.EmissionLib()
iscdataloc = os.path.join(iscdataloc,"endf7.isc.bin")
isc.loadLibs(abund_lib, decay_lib, emission_lib, os.path.join(iscdataloc,"endf7.isc.bin"))

# get all U isotopes with data
u_isos = abund_lib.GetIsosForZ(92)
print(u_isos)

# get mass and abundance for each naturally occuring isotope
nat_u_isos = list()
for iso in u_isos:
    # get mass and abundance for iso
    mass = abund_lib.GetMass(iso)
    abundance = abund_lib.GetAbundance(iso)
    print("{:7d} {:7.3f} {:12.5e}".format(iso, mass, abundance))
    # if abundances is non–zero add it to the list of naturally occurring isos
    if( abundance > 0.0 ):
        nat_u_isos.append(iso)
print(nat_u_isos)
ISC Example 3: Querying Decay Data

- this example shows how to query the decay data to obtain information such as half life, daughters, and branching ratios
- First, all the daughters of Cs-137 are requested
- Next, a loop over the number of decay pathways is constructed
- Then, for each decay pathway, the daughter and branching ratio is obtained
import os
import isc # import the isc module

# load the data
iscdata = os.getenv("ISCDATA")
abund_lib = isc.AbundanceLib()
decay_lib = isc.DecayLib()
emission_lib = isc.EmissionLib()
is.data = os.path.join(iscdata,"endf7.isc.bin")

# get all the daughters of Cs–137
cs137_daughters = decay_lib.GetAllDaughters(55137)
print(c137_daughters)

# get the decay data for Cs–137
cs137_decay_data = decay_lib.GetDecayData(55137)
print("Cs–137 half-life = {:12.5e} s".format(cs137_decay_data.GetHalfLife()))

# loop over the number of decay pathways
for i in range(cs137_decay_data.GetNumber()):
    # get daughter SZA is branching ratio
    daughter = cs137_decay_data.GetDaughter(i)
    branching_ratio = cs137_decay_data.GetBranchingRatio(i)
    print("{:7d} => {:12.5e}".format(daughter, branching_ratio))
ISC Example 4: Querying Emission Data

- This example demonstrates how to get emission spectra of particular particle type out of the emission data.
- First, all the spectra from Co-60 are requested.
- Second, the gamma emission spectrum is obtained (an emission spectrum can have discrete and continuous pieces).
- Lastly, a loop over the discrete emissions is created and the energy and #/decay are printed.
import os
import isc  # import the isc module

# load the data
iscdata = os.getenv("ISCDATA")
abund_lib = isc.AbundanceLib()
decay_lib = isc.DecayLib()
emission_lib = isc.EmissionLib()
iscc.loadLibs(abund_lib, decay_lib, emission_lib, 
   os.path.join(iscdata, "endf7.isc.bin"))

# get the emission spectra for Co–60
co60_spectra = emission_lib.GetSpectra(27060)

# get a list of isc particle types for which spectra exist
co60_particle_types = co60_spectra.GetParticleTypes()
print(co60_particle_types)

# loop over the particle types
for ptype in co60_particle_types:
    if ptype == isc.ENDF_DECAY_BETAP:  
        print("Co–60 emits beta+, ptype={:.d}".format(ptype))
    elif ptype == isc.ENDF_DECAY_BETAM: 
        print("Co–60 emits beta-, ptype={:.d}".format(ptype))
    elif ptype == isc.ENDF_DECAY_GAMMA: 
        print("Co–60 emits gammas, ptype={:.d}".format(ptype))
    elif ptype == isc.ENDF_DECAY_IT: 
        print("Co–60 has internal transition, ptype={:.d}".format(ptype))
    elif ptype == isc.ENDF_DECAY_ALPHA: 
        print("Co–60 emits alphas, ptype={:.d}".format(ptype))
    elif ptype == isc.ENDF_DECAY_ELECTRON: 
        print("Co–60 emits electrons, ptype={:.d}".format(ptype))
    elif ptype == isc.ENDF_DECAY_XRAY: 
        print("Co–60 emits xrays, ptype={:.d}".format(ptype))

# get the gamma spectrum
co60_gammas = co60_spectra.GetSpectrum(isc.ENDF_DECAY_GAMMA)
print("The number of discrete emissions per decay is {:.3f}".format(co60_gammas.GetDNorm()))

# loop over all the discrete emissions
for i in range(co60_gammas.GetDNumber()):
    energy = co60_gammas.GetDEnergy(i)
    intensity = co60_gammas.GetDIntensity(i)
    print("{:12s} {:12s}".format("energy", "/decay")
    energy = co60_gammas.GetDEnergy(i)
    intensity = co60_gammas.GetDIntensity(i)
    print("{:12.5e} {:12.5e}".format(energy, intensity))
MCNPTools History and Overview

- MCNPTools was born out of the continual need to process MCNP outputs.
- MCNPTools is a library that provides object-oriented access to MCNP outputs:
  1. MCTAL files
  2. MESHTAL B (MCNP5 style) files
  3. PTRAC files
  4. LNK3DNT files (not herein discussed)
- MCNPTools is written in C++ and bound to Python and Perl.
- MCNPTools also has some binary “widgets”.
**13dcoarsen Widget**

**USAGE:**

```
13dcoarsen [--help] [--novoid] [--ifact ifact] [--jfact jfact]
            [--kfact kfact] [--maxmats maxmats] <LNK3DNT> [OUTPUT]
```

**DESCRIPTION:**

13dcoarsen coarsens a LNK3DNT file mesh by specified factors

**OPTIONS:**

--help, -h : print usage information
--novoid, -n : Make voids material '0' rather than the assumed material '1' (not recommended)
--ifact, -i : Factor by which to coarsen in the first mesh dimension
--jfact, -j : Factor by which to coarsen in the second mesh dimension (if applicable)
--kfact, -k : Factor by which to coarsen in the third mesh dimension (if applicable)
--maxmats, -m : Maximum number of materials to keep include on the coarsened LNK3DNT file (default: same as original)

**LNK3DNT** : LNK3DNT file name to coarsen

**OUTPUT** : coarsened LNK3DNT output name (Default: lnk3dnt.coarse)

**AUTHOR:** Clell J. (CJ) Solomon [csolomon@lanl.gov]
13dinfo Widget

USAGE: 13dinfo [--full] [--help] <LNK3DNT [LNK3DNT ... ]>

DESCRIPTION:

13dinfo produces information about LNK3DNT files to stdout

OPTIONS:

--full, -f : Produce a full listing of the LNK3DNT contents (can greatly increase runtime)

--help, -h : print usage information

LNK3DNT : LNK3DNT files about which to produce information

AUTHOR: Clell J. (CJ) Solomon [csolomon@lanl.gov]
**l3dscale Widget**

**USAGE:** l3dscale [--help] <LNK3DNT> <FACTOR> [OUTPUT]

**DESCRIPTION:**

l3dscale scales the dimensions of a LNK3DNT file

**OPTIONS:**

--help, -h : print usage information

LNK3DNT : LNK3DNT file to be scaled

FACTOR : Scaling factor to be applied to the file

OUTPUT : Output LNK3DNT file name [Default: LNK3DNT.scaled]

**AUTHOR:** Clell J. (CJ) Solomon [csolomon@lanl.gov]
**mergemctals Widget**

**USAGE:** mergemctals [--help] [--verbose] [--output output] <MCTAL [MCTAL ... ]>

**DESCRIPTION:**

mergemctals statistically merges multiple MCNP MCTAL files into a single MCTAL file.

**OPTIONS:**

--help, -h : print usage information

--verbose, -v : Increase output verbosity

--output, -o : Output MCTAL file name [Default: mergemctals.out]

MCTAL : MCTAL file names to be merged

**AUTHOR:** Clell J. (CJ) Solomon [csolomon@lanl.gov]

**Note:**

This is an alternative, not yet a replacement for, the Perl script that ships with MCNP. However, this binary can be compiled with MPI (requires C++ Boost Libraries), which speeds up merging.
mergemeshtals Widget

**USAGE:** mergemeshtals [--help] [--verbose] [--output output] <MESHTAL [MESHTAL ... ]>

**DESCRIPTION:**
mergemeshtals statistically merges multiple MCNP MESHTAL files into a single MESHTAL file.

**OPTIONS:**

--help, -h : print usage information
--verbose, -v : Increase output verbosity
--output, -o : Output MESHTAL file name [Default: mergemeshtals.out]

MESHTAL : MESHTAL file names to be merged

**AUTHOR:** Clell J. (CJ) Solomon [csolomon@lanl.gov]

**Note:**
This is an alternative, not yet a replacement for, the Perl script that ships with MCNP. However, this binary can be compiled with MPI (requires C++ Boost Libraries), which speeds up merging.
mctal2rad Widget

[TALLY [TALLY ... ]]

DESCRIPTION:

mctal2rad converts an image tally from an MCNP MCTAL file into a TIFF image

OPTIONS:

--direct, -d    : Produce an image of the direct contribution
--help, -h      : print usage information
--log, -l       : Produce an image of the log of the MCTAL values
--transpose, -t : Transpose the image

MCTAL : MCTAL file containing one or more image tallies
TALLY : Tally number for which to produce the images

AUTHOR: Clell J. (CJ) Solomon [csolomon@lanl.gov]

NOTE:

This utility requires the TIFF libraries be installed.
**mctal Overview**

- The Mctal class stores information regarding mctal files
- Tally and Kcode data are stored in the MctalTally and MctalKcode classes, respectively
- Tally and Kcode data are only read into memory when requested via the getTally and getKcode methods of the Mctal class are called
Mctal Class Accessors

- **general Mctal accessors**
  - `GetCode()` – return the generating code name
  - `GetVersion()` – return the code version
  - `GetProbid()` – return the problem id number
  - `GetDump()` – return the corresponding runtpe dump number
  - `GetNps()` – return the number of histories
  - `GetRandoms()` – return the number of random numbers
  - `GetTallyList()` – return a list of the tally numbers
  - `SummaryString()` – return a summary string of the mctal file

- **tally and kcode object accessors**
  - `GetTally(NUM)` – return a `MctalTally` object for tally NUM
  - `GetKcode()` – return a `MctalKcode` object
MctalTally Class Accessors

- **general MctalTally accessors**
  - `ID()` – return the tally id (number)
  - `SummaryString()` – return a summary string of the tally

- **MctalTally bin accessors**
  - `GetFBins()` – return the facet bins
  - `GetDBins()` – return the flagged/direct/collided bins
  - `GetUBins()` – return the user bins
  - `GetSBins()` – return the segment bins
  - `GetMBins()` – return the multiplier bins
  - `GetCBins()` – return the cosine bins
  - `GetEBins()` – return the energy bins
  - `GetTBins()` – return the time bins
MctalTally Class Accessors II

- **MctalTally value accessors**
  - `GetValue(F,D,U,S,M,C,E,T,PERT=NUM)` — return the tally value corresponding to the bin specified by the set of indices F, D, U, S, M, C, E, and T for perturbation NUM (defaults to unperturbed) [a negative value for any bin indicates use of the tally fluctuation chart index]
  
  - `GetError(F,D,U,S,M,C,E,T,PERT=NUM)` — return the tally relative error corresponding to the bin specified by the set of indices F, D, U, S, M, C, E, and T for perturbation NUM (defaults to unperturbed) [a negative value for any bin indicates use of the tally fluctuation chart index]
MctalKcode Class Accessors

- **general MctalKcode accessors**
  - `GetCycles()` — return the number of total kcode cycles
  - `GetSettle()` — return the number of inactive kcode cycles
  - `GetNdat()` — return the number of data elements in a kcode entry
  - `SummaryString()` — return a summary string for the kcode data

- **MctalKcode data accessor**
  - `GetValue(VAL,CYCLE)` — return the kcode value VAL for cycle CYCLE (defaults are averaged combined $k_{eff}$ and last cycle). VAL is an integer between 0 and the number of data elements (see the manual for details)
mctal Examples

1. read tally 4 out of mctal file “my_mctal”, extract the energy bin boundaries, store the tally values for each of the energy bins, and print the tally values for each of the energy bins

2. read the kcode data from mctal file “my_mctal” and print the $k_{eff}$ value and error as a function of active cycle
```python
from mcnptools import Mctal, MctalTally

# construct the mctal class from mctal file "my_mctal"
m = Mctal("my_mctal")

tfc = MctalTally.TFC; # alias for -1

# get tally 4 from the mctal file
t4 = m.GetTally(4);

# get the energy bins of tally 4
t4_e = t4.GetEBins();

# loop over energy bin indices to store and print tally bin value
# using the TFC bin for all other bins

# store the tally values with list comprehension
# f d u s m c e t

t4_evals = [ t4.GetValue(tfc,tf,tcf,tfc,tfc,tfc,i,tfc) for i in range(len(t4_e)) ];

# print the tally values
for i in range(len(t4_evals)):
    print t4_evals[i];
```
from mcnptools import Mctal, MctalKcode

# construct the mctal class from the mctal file "my_mctal"
m = Mctal("my_mctal")

# get the kcode data
kc = m.GetKcode()

# alias for average combined keff
keff = MctalKcode.AVG_COMBINED_KEFF
# alias for average combined keff standard deviation
keff_std = MctalKcode.AVG_COMBINED_KEFF_STD

# loop over active cycles and print
for i in range(kc.GetSettle(), kc.GetCycles()):
    print(i, "", kc.GetValue(keff, i), "", kc.GetValue(keff_std, i))
**meshtal Overview**

- The Meshtal class stores information about the mesh-tally type B data
- The MeshtalTally class stores information about a mesh tally
- The tally data is not read in until the `getTally` method of the Meshtal class is called
Meshtal Class Accessors

- **general Meshtal accessors**
  - `GetCode()` – return the generating code name
  - `GetVersion()` – return the code version
  - `GetProbid()` – return the problem id number
  - `GetComment()` – return the problem comment
  - `GetNps()` – return the number of histories
  - `SummaryString()` – return a summary string
  - `GetTallyList()` – return a list of tallies in the file

- **tally object accessor**
  - `GetTally(NUM)` – return a tally data object corresponding to tally NUM
MeshtalTally Class Accessors I

- general MeshtalTally accessors
  - ID() – return the tally id (number)
  - GetXRBounds() – return the x/r bin boundaries
  - GetYZBounds() – return the y/z bin boundaries
  - GetZTBounds() – return the z/theta bin boundaries
  - GetEBounds() – return the energy bin boundaries
  - GetTBounds() – return the time bin boundaries
  - GetXRBins() – return the x/r bin centers
  - GetYZBins() – return the y/z bin centers
  - GetZTBins() – return the z/theta bin centers
  - GetEBins() – return the energy bins
  - GetTBins() – return the time bins
  - GetVolume(I,J,K) – return the volume of specified element
MeshtalTally Class Accessors II

- **MeshtalTally value accessors**
  - `GetValue(I, J, K, E, T)` – return the tally value for (I,J,K)th element for energy bin E and time bin T. E and T default to the totals if left unspecified.
  - `GetError(I, J, K, E, T)` – return the tally relative error for the (I,J,K)th element for energy bin E and time bin T. E and T default to the totals if left unspecified.
**meshtal Examples**

1. read in a meshtal file, extract tally 4, and print the 2-D slice for the z index of 5
```python
from mcnptools import Meshtal, MeshtalTally
from sys import stdout

# construct the meshtal class from meshtal file "my_meshtal"
m = Meshtal("my_meshtal")

# get tally 4 from the meshtal file
t4 = m.GetTally(4)

# get the x and y bin centers
x = t4.GetXRBins()
y = t4.GetYZBins()

# loop over x and y bins indices and print the tally value for
# z index of 5
for i in range(len(x)):
    for j in range(len(y)):
        stdout.write("{:12.5e}".format(t4.GetValue(i,j,5)))
        stdout.write("\n")
```

*Los Alamos National Laboratory*

Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA
Outline

Intrinsic Source Constructor
  Overview
  Using MISC to Generate Source Terms for MCNP
  ISC Library Usage

MCNPTools
  Overview
  Widgets
  MCNPTools Accessor Classes
    Mctal Class
    Meshtal (mesh-tally type B) Class
    Ptrac Class
  Interactive MCNPTools Demo (time permitting)
ptrac Overview

- The Ptrac class manages data for MCNP’s ptrac files and can handle BOTH text and binary versions (assumes binary as default)
- The PtracHistory class manages data for an individual history
- The PtracNps class handles the starting information for a history
- The PtracEvent class handles information for particle events
Ptrac Class Accessors

- ReadHistories(NUM) – reads NUM histories and all associated events into memory as a list; allows “chunk” processing of a file without filling memory
PtracHistory Class Accessors

- `GetNPS()` – return a `PtracNps` class to the history (NPS) information
- `GetNumEvents()` – return the number of events recorded for the history
- `GetEvent(NUM)` – return a `PtracEvent` class to the history’s NUMth event
PtracNps Class Accessors

- **NPS()** – return the history number
- **Cell()** – return the filtering cell from CELL keyword (if present)
- **Surface()** – return the filtering surface from SURFACE keyword (if present)
- **Surface()** – return the filtering tally from TALLY keyword (if present)
- **Value()** – return the tally score from TALLY keyword (if present)
PtracEvent Class Accessors

- Type() – return the event type
- Has(DATA) – return true if the event has data type DATA
- Get(DATA) – return the value of the data type DATA
ptrac Examples

1. Open a ptrac file named “my_ptrac”, read in histories, and print the x, y, and z location and energy of all bank events (this is when particles are pulled from the bank)

2. Open a ptrac file named “my_ptrac”, read in histories, and, for all surface crossings, print the x, y, and z location and the angle (in degrees) of the crossing with respect to the surface normal
ptrac Example 1 – Python

```python
from mcnptools import Ptrac
from sys import stdout

# explicitly open the file as a binary ptrac
p = Ptrac("my_ptrac", Ptrac.BIN_PTRAC)

# initialize counter
cnt = 0

# read histories in batches of 10000
hists = p.ReadHistories(10000)
while hists:
    # loop over all histories
    for h in hists:
        # loop over all events in the history
        for e in range(h.GetNumEvents()):
            event = h.GetEvent(e)

            if event.Type() == Ptrac.BNK:
                cnt += 1

                stdout.write("{:13d}{:13.5e}{:13.5e}{:13.5e}{:13.5e}\n".format(
                    cnt,
                    event.Get(Ptrac.X),
                    event.Get(Ptrac.Y),
                    event.Get(Ptrac.Z),
                    event.Get(Ptrac.ENERGY)
                ))

    hists = p.ReadHistories(10000)
```
from mcnptools import Ptrac
from sys import stdout

# explicitly open the file as a binary ptrac
p = Ptrac("my_ptrac", Ptrac.BIN_PTRAC)

# read histories in batches of 10000
hists = p.ReadHistories(10000)

while hists:
    # loop over all histories
    for h in hists:
        # loop over all events in the history
        for e in range(h.GetNumEvents()):

            event = h.GetEvent(e)

            if event.Type() == Ptrac.SUR:
                stdout.write("{:13.5e}{:13.5e}{:13.5e}{:13.5e}
".format(
                    event.Get(Ptrac.X), \
                    event.Get(Ptrac.Y), \
                    event.Get(Ptrac.Z), \
                    event.Get(Ptrac.ANGLE) \
                )
            )

    hists = p.ReadHistories(10000)
Interactive MCNPTools Demo