LA-UR-16-24290
Approved for public release; distribution is unlimited.

Title:	New Tools to Prepare ACE Cross-section Files for MCNP Analytic Test Problems
Author(s):	Brown, Forrest B.
Intended for:	MCNP documentation
Issued:	2016-06-17

New Tools to Prepare ACE Cross-section Files for MCNP Analytic Test Problems

Forrest B. Brown

Monte Carlo Codes Group, LANL: PO Box 1663, Los Alamos, NM 87545, fbrown@lanl.gov

INTRODUCTION

Monte Carlo calculations using one-group cross-sections, multigroup cross-sections, or simple continuous-energy cross-sections are often used to: (1) verify production codes against known analytical solutions, (2) verify new methods and algorithms that do not involve detailed collision physics, (3) compare Monte Carlo calculation methods with deterministic methods, and (4) teach fundamentals to students. In this work we describe 2 new tools for preparing the ACE cross-section files to be used by MCNP® for these analytic test problems, *simple_ace.pl* and *simple_ace_mg.pl*.

SIMPLE CONTINUOUS-ENERGY ACE FILES

Description

The *simple_ace.pl* perl script can be used to create continuous-energy ACE cross-section files for analytic test problems. The cross-sections are represented as piecewise linear in energy and can include only capture, fission, nu, and elastic scattering. The fission spectrum $\chi(E)$ is a delta function of energy for prompt neutrons only. P_1 anisotropic scattering is permitted, but only for $|\bar{\mu}| < 1/3$. The ACE file name, ZAID, atomic weight ratio, and temperature can be arbitrarily specified. There is also an option to Doppler broaden the scattering cross-section at each energy point according to the constant cross-section approximation:

$$\sigma_s(E,T) = \sigma_s(E,0)\cdot\left[\left(1+\frac{1}{2x^2}\right)\cdot \mathrm{erf}(x) + \frac{\exp(-x^2)}{\sqrt{\pi}\cdot x}\right]$$

where $x = \mathrm{AWR}\cdot(E/T)$.

simple_ace.pl Usage

```
simple_ace.pl  -file  str   -zaid str   -comment str  \
        -awr x      -tmp x  \
        -e list     -nloge list  \
        -s list     -c list     -f list    -nu list  \
        -mu list    -s1 list  \
        -broaden    -echi x
```

where:

- Arguments can appear in any order. Defaults are noted below for each.
- On Mac OS X and Linux, the line continuation character (if used) is "\". It must be the last character on the line (i.e., no trailing blanks). For Windows the line continuation character is "^".
- *str* denotes a string of characters, surrounded by quotes if the string contains blanks. (Use double-quotes on Windows.)
- *x* denotes a single numerical value.
- *list* may be one numerical value or a list of numerical values.
  - All lists must have the same number of entries as the list of energies. (Except –nloge)
  - List entries must be separated by blanks (not commas).
- For -s, -c, -f, -nu, -mu:
  - If not supplied, set to 0 at each energy
  - If 1 value is supplied in *list*, use it for all energies.
  - Otherwise, the number of values must match the -e *list.*

-file *str*

Use *str* as the filename for the output ACE dataset. [default = zaid string, or 99999.99c if -zaid not supplied]

-zaid *str*

Use *str* as the ACE dataset identifier, ZZAAA.nnc. ZZAAA must be ≤ 99999, and *nn* must be 2 digits. [default = 99999.99c]

-comment *str*

Comment to include in ACE file. Must be quoted if *str* contains blanks or special characters. Length should be ≤ 70 characters. [default = "ACE file created by simple_ace.pl"]

-awr *x*

Use *x* as the atomic weight ratio, $(M / M_{neutron})$. [default = 1e6]

-tmp *x*

Use *x* as the temperature, in units of MeV if $x<1$ or Kelvin if $x>1$. [default = 2.5301e-8 MeV]

-e *list*

List of energy points (MeV). The list must include 2 or more values, & entries must be in increasing order.

[default = 1.0e-11 100.0]

-s *list*

Elastic scattering cross-section.

-c *list*

Capture cross-section (not including fission).

-f *list*

Fission cross-section.

-nu *list*

Number of neutrons from fission, nubar. Set to 0 if the fission cross-section is zero.

-mu *list*

Average cosine of scatter angle, $\bar{\mu}$.

$|\bar{\mu}|$ must be ≤ 1/3.

-s1 *list*

P_1 component of elastic scattering cross-section. May be used instead of –mu, but limit still applies.

-t *list*

IGNORED. The total cross-section is constructed by adding elastic scattering, capture, & fission.

-echi *x*

Use *x* as the single energy (MeV) for delta function $\chi(E)$. Must be supplied if fission is present. [default = none.]

-nloge *list*

Expand the energy mesh supplied by -e, dividing each energy interval into *n* equally-spaced bins in log(E), where *n* is an integer entry in *list*. The number of values in *list* must be 1 less than the number in -e *list* (since the number of intervals is 1 less than the number of points). Linear interpolation in energy is then used to expand the cross-sections to the new energy mesh.

-broaden

Assume each input elastic scattering cross-section is at 0K, and then Doppler broaden each (assuming constant cross-section approximation noted in the previous section). When this option is used, a large number of energy points (via the "–nloge *list*" option) should be used so that linear interpolation of the elastic scattering cross-section is accurate.

Examples

Example 1 shows the usage and input to *simple_ace.pl*, and the screen output that is produced for one of the analytic K_{eff} benchmark problems, ce01 [1]. The "XSn" card information from the *simple_ace.pl* output is then used in an MCNP6 input file to run the calculation using the ACE file produced.

Example 2 shows the usage and input to *simple_ace.pl*, and the screen output that is produced for a benchmark problem for free-gas scattering [2], with AWR=12 and 1001 energy points. The use of "-nloge 1000" expands the energy mesh so that many energy points are used when the Doppler broadening of the scattering cross-section is performed. This retains accuracy for the linear interpolation of the broadened elastic scattering cross-sections.

SIMPLE MULTIGROUP ACE FILES

Description

The *simple_ace_mg.pl* perl script can be used to create multigroup ACE cross-section files for analytic test problems. Multigroup cross-sections for MCNP are represented in a somewhat kludged-up manner, such that scatter from one group to another places the neutrons at the energy-midpoint of the exit group. The cross-sections can include only capture, fission, nu, and scattering.

P_1 anisotropic scattering is permitted for scatter and group-to-group scatter, but only for $|\bar{\mu}| < 1/3$. Due to the limitations on current coding in MCNP6, scattering PDFs are constrained to be equiprobable bins in μ. Unlike continuous-energy scattering where P_1 scattering can be modeled exactly, multigroup scattering must approximate P_1 scattering by using a large number of equiprobable bins.

The ACE file name, ZAID, atomic weight ratio, and temperature can be arbitrarily specified.

***simple_ace_mg.pl* Usage**

```
simple_ace.pl  -file str   -zaid str   -comment str  \
        -awr x    -tmp x  \
        -groups x      -e list      \
        -c list     -f list    -nu list    -t list   -chi list   \
        -s list     -s1 list    -bins x
```

where:

- Arguments can appear in any order. Defaults are noted below for each.
- On Mac OS X and Linux, the line continuation character (if used) is "\". It must be the last character on the line (i.e., no trailing blanks). For Windows the line continuation character is "^".
- *str* denotes a string of characters, surrounded by quotes if the string contains blanks. (Use double-quotes on Windows.)
- *x* denotes a single numerical value.

- *list* may be one numerical value or a list of numerical values.
  - Lists for –c, -f, -nu, -t, and –chi must have an entry for each group.
  - Lists for –s and –s1 must have NG^2 entries, where NG is the number of energy groups. The ordering of entries is significant, see below.
  - List entries must be separated by blanks (not commas).
- For -c, -f, -t, -nu, -chi, -s, -s1:
  - If not supplied, set to 0 for each group.
  - If 1 value supplied in *list*, use it for all groups.
  - Otherwise, the number of values must be NG (or NG^2 for –s and -s1).
- **Important**: While continuous-energy data is usually entered from low-energy to high-energy, the multigroup data is entered from group-1 (high-energy) through group-NG (lowest energy). For multigroup ACE files, energies & cross-sections must be ordered from high-energy to low energy.
- For scattering, the entries in lists for –s and –s1 must be in this order:

  1-->1, 1-->2, ..., 1-->NG,
  2-->1, 2-->2, ..., 2-->NG,
  . . .
  NG-->1, NG-->2, ..., NG-->NG,

  For P_1 scatter, abs(sig1/sigs) must be $\leq$ 1/3.
- Negative cross-sections for –t, –c, -f, -nu, -chi, -s are not permitted.

-file *str*

Use *str* as the filename for the output ACE dataset. [default = zaid string, or 99999.99m if -zaid not supplied]

-zaid *str*

Use *str* as the ACE dataset identifier, ZZAAA.nnm. ZZAAA must be $\leq$ 99999, and *nn* must be 2 digits. [default = 99999.99m]

-comment *str*

Comment to include in ACE file. Must be quoted if *str* contains blanks or special characters. Length should be $\leq$ 70 characters.
[default = "multigroup ACE file"]

-awr *x*

Use *x* as the atomic weight ratio, ($M / M_{neutron}$).
[default = 1e6]

-tmp *x*

Use *x* as the temperature, in units of MeV if *x*<1 or Kelvin if *x*>1. [default = 2.5301e-8 MeV]

-groups *x*

Use *x* as the number of energy groups, NG.
[default = 1]

-e *list*

List of energy points at the group boundaries (MeV). If specified, *list* must have NG+1 entries. The list must include 2 or more values, & entries must be in **decreasing** order. Note that the energy bounds are arbitrary and not used directly in MCNP. If NG>2, the energy bounds must be specified in decreasing order.
[defaults:
For NG=1: 100.0, 0.
For NG=2: 100.0, .625e-6, 0.
For NG>2: must specify NG+1 energy bounds]

-c *list*

Capture cross-section (not including fission).

-f *list*

Fission cross-section.

-nu *list*

Number of neutrons from fission, nubar. Set to 0 if the fission cross-section is zero.

-t *list*

Total cross-section.

-chi *list*

Fraction of fission neutrons for each group.

-s *list*

Elastic scattering cross-section. List must have NG^2 entries, from high-energy group through low energy group.

-s1 *list*

P_1 component of elastic scattering cross-section.

-bins *list*

Number of equiprobable bins to use for the PDFs for P_1 scattering.
[default = 1000]

Examples

Example 3 shows the usage and input to *simple_ace_mg.pl*, and the screen output that is produced for one of the analytic K_{eff} benchmark problems, mg01 [1], a 1-group problem. The "XSn" card information from the *simple_ace_mg.pl* output is then used in an MCNP6 input file to run the calculation using the ACE file produced.

Example 4 shows the usage and input to *simple_ace_mg.pl*, and the screen output that is produced for one of the analytic K_{eff} benchmark problems, mg72 [1], a 2-group problem with P_1 scattering. The "XSn" card information from the *simple_ace_mg.pl* output is then used in an MCNP6 input file to run the calculation using the ACE file produced.

LISTINGS FOR *simple_ace.pl* & *simple_ace_mg.pl*

Listings of the perl scripts *simple_ace.pl* and *simple_ace_mg.pl* are provided at the end of this report. The scripts are written in a straightforward manner, first processing input options and creating data in the requested form, and then writing the data in the proper ACE file format. The format specifications for both continuous-energy and multigroup ACE files are available in "Appendix F: Data Table Formats" in Volume III of the MCNP5 manual [3]. The two scripts should be easy to modify to include other options for preparing analytic cross-sections.

REFERENCES

1. A. Sood, R.A. Forster, D.K. Parsons, "Analytic Benchmark Test Set for Criticality Code Verification," *Prog. Nucl. Energy*, **42**, 55-106 (2003).
2. M.A. Gonzales, A. Prinja, F.B. Brown, B.C. Kiedrowski, "An Analytic Benchmark of Neutron Free-gas Scattering Using Continuous-energy Cross Sections in MCNP6", ANS PHYSOR-2016, Sun Valley ID, May 2016, LA-UR-15-26797 (2016).
3. X-5 Monte Carlo Team, "MCNP5 Manual, Volume III: Developer's Guide", LA-CP-03-0284 (April, 2003). [Not available on the MCNP website. This volume of the manual is included with the source-code distributions of MCNP from RSICC.]

Example 1 – Continuous-energy ACE File with Constant Cross-sections for Problem ce01

This example illustrates how to set up & run the analytic Keff benchmark problem "pu-239a", problem ce01 in the MCNP6/Testing/VERIFICATION_KEFF suite, based on LA-UR-01-3082.

Create the ACE file ce01xsecs

```
simple_ace.pl   -zaid  99902.01c    file=ce01xsecs \
      -comment  "la-ur-01-3082, table 2, pu-239a"   \
      -e 1.e-11 100.     -echi  1.  \
      -nu 3.24    -f 0.081600    -c 0.019584  \
      -s 0.225216     -t 0.32640
```

Output from simple_ace.pl

```
=====> simple_ace.pl - create special purpose ACE file

...ignoring input '-t list'; sigt constructed from -s,-f,-c

      zaid = 99902.01c
      za   = 99902
      file = ce01xsecs
       awr = 1000000
       tmp = 2.5301e-08
      echi = 1.
energy pts = 2
  xss size = 60

 e            sigt         sigc         sigs         nu     sigf
1.0000e-11   3.2640e-01   1.9584e-02   2.2522e-01    3.24  8.1600e-02
1.0000e+02   3.2640e-01   1.9584e-02   2.2522e-01    3.24  8.1600e-02

XSDIR Info, to use on XSn card:

   XSn   99902.01c 1e+06 ce01xsecs  0 1 1 60  0 0 2.5301e-08


Creating ACE file:  ce01xsecs
```

MCNP6 Input File Using ce01xsecs

```
la-ur-01-3082, analytic problem 1, PUa-1-0-IN, k=2.612903
 10  100 1.0   -1  imp:n=1

 *1   so    1e6

 kcode  100000  2.0  100 600
 ksrc  0 0 0
 hsrc  1 -1e6 1e6  1 -1e6 1e6  1 -1e6 1e6
 xs1   99902.01c 1e+06 ce01xsecs  0 1 1 60  0 0 2.5301e-08
 m100  99902.01c  1.0
 phys:n j 100
 prdmp 1e7 4r
```

Example 2 – Continuous-energy ACE File for Energy-dependent Cross-sections for Free-gas Scattering Benchmark

This example illustrates how to set up & run an analytic Keff benchmark problem for free-gas scattering. The elastic scattering cross-section is Doppler broadened, and the large number of energy points is needed to preserve accuracy in the linearly-interpolated elastic scattering cross-sections.

Create the ACE file fgxsecs

```
simple_ace.pl   -zaid  123456.01c    file=fgxsecs \
      -awr  12.0  \
      -comment  "xsecs or fg-scatter benchmark"   \
      -e 1.e-11 100.     -echi  10.  \
      -nu 2.0    -f 0.01    -c 0.01   -s 0.5 \
      -nloge  1000   -broaden
```

Output from simple_ace.pl

```
=====> simple_ace.pl - create special purpose ACE file

      zaid = 123456.01c
      za   = 123456
      file = fgxsecs
       awr = 12.0
       tmp = 2.5301e-08
      echi = 10.
energy pts = 1001
  xss size = 9051

 e              sigt           sigc           sigs           nu    sigf
1.0000e-11     8.2252e+00     1.0000e-02     8.2052e+00     2.00  1.0000e-02
1.0304e-11     8.1037e+00     1.0000e-02     8.0837e+00     2.00  1.0000e-02
1.0617e-11     7.9840e+00     1.0000e-02     7.9640e+00     2.00  1.0000e-02
1.0940e-11     7.8661e+00     1.0000e-02     7.8461e+00     2.00  1.0000e-02
1.1272e-11     7.7499e+00     1.0000e-02     7.7299e+00     2.00  1.0000e-02
1.1614e-11     7.6355e+00     1.0000e-02     7.6155e+00     2.00  1.0000e-02
  . . . . .
9.1411e+01     5.2000e-01     1.0000e-02     5.0000e-01     2.00  1.0000e-02
9.4189e+01     5.2000e-01     1.0000e-02     5.0000e-01     2.00  1.0000e-02
9.7051e+01     5.2000e-01     1.0000e-02     5.0000e-01     2.00  1.0000e-02
1.0000e+02     5.2000e-01     1.0000e-02     5.0000e-01     2.00  1.0000e-02

XSDIR Info, to use on XSn card:

XSn  123456.01c 12      fgxsecs    0 1 1 9051  0 0 2.5301e-08


Creating ACE file:  fgxsecs
```

Example 3 –Multigroup ACE File with Cross-sections for 1-Group Problem mg01

This example illustrates how to set up & run the analytic Keff benchmark problem "pu-239a", problem mg01 in the MCNP6/Testing/VERIFICATION_KEFF suite, based on LA-UR-01-3082.

Create the ACE file ce01xsecs

```
simple_ace_mg.pl   -file  mg01xsecs  -zaid 99902.01m  \
   -comment "la-ur-01-3082, table 2, pu-239a" \
    -groups 1     -chi 1. \
    -nu 3.24  -f 0.081600  -c 0.019584  -s 0.225216  -t 0.32640
```

Output from simple_ace.pl

```
=====> simple_ace_mg.pl - create simple multigroup ACE file

      zaid = 99902.01m
      za   = 99902
      file = mg01xsecs
       awr = 1000000
       tmp = 2.5301e-08
    groups = 1
  xss size = 11
   comment = la-ur-01-3082, table 2, pu-239a
      date = 2016-06-05

  group    Ehi  Elow     total  capture  scatter  fission   nu     chi
      1    100     0    0.3264  0.01958   0.2252   0.0816   3.       1

  XSDIR Info, to use on XSn card:

 XSn   99902.01m 1e+06  mg01xsecs  0 1 1 11  0 0 2.5301e-08

   Creating ACE file:   mg01xsecs
```

MCNP6 Input File Using mg01xsecs

```
la-ur-01-3082, analytic problem 1, PUa-1-0-IN, k=2.612903
 10  100 1.0   -1  imp:n=1

 *1   so    1e6

 kcode  100000  2.0  100 600
 ksrc  0 0 0
 hsrc  1 -1e6 1e6  1 -1e6 1e6  1 -1e6 1e6
 xs1   99902.01m 1e+06 mg01xsecs  0 1 1 11  0 0 2.5301e-08
 m100  99902.01m  1.0
 mgopt f 1
 phys:n j 100
 prdmp 1e7 4r
```

Example 4 –Multigroup ACE File with Cross-sections for 2-Group Problem with P_1 Scattering, mg72

This example illustrates how to set up & run the analytic Keff benchmark problem "d20 Rx", 2-group problem with anisotropic scattering mg72 in the MCNP6/Testing/VERIFICATION_KEFF suite, based on LA-UR-01-3082.

Create the ACE file mg72xsecs

```
simple_ace_mg.pl   -zaid 99953.01m  -file mg72xsecs  \
    -comment "la-ur-01-3082, table 53, d2o Rx" \
    -groups 2     -chi 1.0  0.0    \
    -nu 2.50 2.50   -f 0.0028172 0.097  -c 0.0087078 0.02518  \
    -t 0.33588 0.54628  \
    -s  0.31980  0.004555    0.0 0.42410  \
    -s1 0.06694 -0.0003972   0.0 0.05439
```

Output from simple_ace.pl

```
=====> simple_ace_mg.pl - create simple multigroup ACE file

      zaid = 99953.01m
      za   = 99953
      file = mg72xsecs
       awr = 1000000
       tmp = 2.5301e-08
    groups = 2
  xss size = 3028
   comment = la-ur-01-3082, table 53, d2o Rx
      date = 2016-06-05

  group   Ehi   Elow     total   capture   scatter  fission    nu    chi
      1   100 6.25e-07   0.3359  0.008708  0.3244   0.002817   2.5   1
      2 6.25e-0    0     0.5463  0.02518   0.4241   0.097      2     0

  scattering matrix,   group-I (down) --> group-J (across)
     J -->           1           2
     I --v
         1      0.3198    0.004555
         2           0      0.4241

  XSDIR Info, to use on XSn card:
     XSn   99953.01m 1e+06  mg72xsecs  0 1 1 3028  0 0 2.5301e-08

  Creating ACE file:  mg72xsecs
```

MCNP6 Input File Using mg72xsecs

```
la-ur-01-3082, analytic problem 72, UD2O-2-1-IN, k=1.000227
 10  100  1.0    -1   imp:n=1

*1  so  1.0e6

 kcode  100000  1.0  100 600
 ksrc    0.0    0.0  0.0
 hsrc  1 -1e6 1e6  1 -1e6 1e6  1 -1e6 1e6
 xs1   99953.01m 1e+06 mg72xsecs  0 1 1 3028  0 0 2.5301e-08
 m100  99953.01m  1.0
 mgopt f 2
 phys:n  j 100
 prdmp 1e7 4r
```

Listing of the simple_ace.pl script:

```
#!/usr/bin/perl
###############################################################################
#
# simple_ace.pl
#
# - create simple ace files for testing & benchmarks
# - consider capture, fission, elastic scatter ONLY
# - no delayed neutrons
# - chi(E) = delta(E-echi)
#
# USAGE:
#   simple_ace.pl  -zaid za   -awr x    -tmp x  -comment str  \
#                  -e list    -nloge    -s list   -c list  \
#                  -f list    -nu list  -mu list  -s1 list  \
#                  -broaden   -echi x   -file str
#  where:
#    -zaid str    = ZZAAA.nnc to use for dataset, default 99999.99c
#    -file str    = filename for output ACE dataset, default zaid
#    -awr  x      = atomic weight ratio,  mass/neutron-mass
#    -tmp  x      = temperature, MeV if <1, Kelvin if >1, default=room
#    -comment str = comment to include in ACE file
#    -echi  x     = energy (MeV) for delta function chi(E)
#
#    -e  list     = energy points (MeV), list must include >= 2 values,
#                   entries must be in increasing order
#
#    for -s, -c, -f, -nu, -mu:
#      = if 0 values supplied, set to 0
#      = if 1 value  supplied, use it for all energies
#      = otherwise, number of values must match the -e list
#
#    -s  list     = scatter xsec, elastic only.
#    -c  list     = capture xsec (not including fission)
#    -f  list     = fission xsec
#    -nu list     = nubar, set to 0 if no fission
#    -mu list     = mubar, average cosine of scatter angle,
#                   abs(mu) MUST BE < 1/3
#    -s1 list     = P1 scatter xsec, may be used instead of mubar
#    -t  list     = IGNORED,  total xsec constructed from s,c,f
#
#    -nloge list  = expand energy mesh, divide each energy interval
#                   into nloge() equally-spaced bins in log(e),
#                   number of values in list = number in -e list -1.
#                   Use linear interpolation for xsec values.
#
#    -broaden     = assume each input scatter xsec is at 0K, and then
#                   Doppler broaden each (assuming constant sig approx)
# History:
#   2015-06-16 - fbb, initial version
#   2015-06-17 - fbb, allow constants or lists for s,c,f,nu
#                single values get propagated to all energies
#   2015-09-09 - fbb, added opts  -sopt   &  -nloge, and functions to
#                permit Doppler broadening of a constant 0-K scatter xsec
#   2015-12-18 - fbb, clean up, generalize -nloge opt,
#                use -broaden instead of "-sopt zero"
###############################################################################
# EXAMPLE
#
# COMMAND:
#
#    ./simple_ace.pl   -nu 3.24 -f .0816 -c .019584 -s .225216 \
#                      -e 1e-11 100 -echi 1
#
# SCREEN OUTPUT:
#
#              zaid = 99999.99c
#              za   = 99999
#              file = 99999.99c
#               awr = 1000000
#               tmp = 2.5301e-08
#              echi = 1
#        energy pts = 2
#          xss size = 60
#
#    e            sigt         sigc         sigs        nu      sigf
#   1.0000e-11   3.2640e-01   1.9584e-02   2.2522e-01   3.24   8.1600e-02
#   1.0000e+02   3.2640e-01   1.9584e-02   2.2522e-01   3.24   8.1600e-02
#
```

```
#   XSDIR Info:
#
#      99999.99c 1e+06     99999.99c  0 1 1 60  0 0 2.5301e-08
#
#
#  Creating ACE file:  99999.99c
#
###############################################################################

  print "\n=====> simple_ace.pl - create special purpose ACE file\n\n";

# =====> get args & expand as needed
  while(   $arg = shift @ARGV ) {
    if(    $arg eq '-file'    ) { $file    = shift(@ARGV); }
    elsif( $arg eq '-zaid'    ) { $zaid    = shift(@ARGV); }
    elsif( $arg eq '-awr'     ) { $awr     = shift(@ARGV); }
    elsif( $arg eq '-comment' ) { $comment = shift(@ARGV); }
    elsif( $arg eq '-tmp'     ) { $tmp     = shift(@ARGV); }
    elsif( $arg eq '-echi'    ) { $echi    = shift(@ARGV); }
    elsif( $arg eq '-e'       ) { @erg     = &get_list;    }
    elsif( $arg eq '-t'       ) { @sigt    = &get_list;    }
    elsif( $arg eq '-s'       ) { @sigs    = &get_list;    }
    elsif( $arg eq '-s1'      ) { @sig1    = &get_list;    }
    elsif( $arg eq '-c'       ) { @sigc    = &get_list;    }
    elsif( $arg eq '-f'       ) { @sigf    = &get_list;    }
    elsif( $arg eq '-nu'      ) { @nu      = &get_list;    }
    elsif( $arg eq '-mu'      ) { @mubar   = &get_list;    }
    elsif( $arg eq '-nloge'   ) { @nloge   = &get_list;    }
    elsif( $arg eq '-broaden' ) { $broaden = 1;            }
    else                        { die("error:  bad arg: $arg"); }
  }

# =====> defaults & checks
  if(  ! $zaid      ) { $zaid    = "99999.99c"; }
  if(  ! $file      ) { $file    = $zaid; }
  if(  ! $awr       ) { $awr     = 1000000.0; }
  if(  ! $comment   ) { $comment = "ACE file created by simple_ace.pl"; }
  if(  ! $echi      ) { $echi    = 0; }
  if(  ! @erg       ) { @erg     = ( 1.e-11, 100.); }
  if(  ! $tmp       ) { $tmp     = 2.5301e-8; }
  if(  $tmp > 1.0   ) { $tmp *= 8.617385e-11; } # if tmp>1, assume Kelvin, convert to MeV
  if(  -s $file                  ) { die("error:  file exists: $file\n"); }
  if(  $za  > 99999              ) { die("error:  za > 99999\n"); }
  if(  @erg < 2                  ) { die("error:  need 2 or more energies\n"); }
  if(  $erg[0] > $erg[1]         ) { die("error:  energies must be increasing order\n"); }
  if(  @nloge>0 && @nloge!=@erg-1 ) { die("error:  list for -nloge\n"); }
  if(  @sig1>0  && @mubar>0      ) { die("error:  use -mu or -s1, not both\n"); }
  ($za=$zaid) =~ s/\..*$//;
  $comment    = substr($comment,0,70);
  $s1_input   = (@sig1>0) ? 1 : 0;        # flag, use sig1 to construct mubars
  $DATE       = &get_date;
  if( @sigt>0 ) {
    undef @sigt;
    print "\t...ignoring input '-t list'; sigt constructed from -s,-f,-c\n\n";
  }
# =====> expand constants to match @erg
  $NE = @erg;
  foreach $sig (\@sigs, \@sigc, \@sigf, \@nu, \@mubar, \@heat, \@sig1) {
    if(    @$sig==0   ) { for($k=0;$k<$NE;$k++) { $sig->[$k]  = 0.0;      } }
    elsif( @$sig==1   ) { for($k=1;$k<$NE;$k++) { $sig->[$k]  = $sig->[0]; } }
    elsif( @$sig!=$NE ) { die("error:  energies & sigs  don't match"); }
  }
# =====> check for fission. if none, set nu=0
  $fission = 0;
  for($k=0;$k<$NE;$k++) { if( $sigf[$k]>0 ) { $fission=1; last; } }
  if( ! $fission ) {
    for($k=0;$k<$NE;$k++) { $nu[$k] = 0.0; }
  }
# =====> check for isotropic, convert sig1 to mubar, & check that abs(mubar) <= 1/3
  $isotropic = 1;
  for($k=0;$k<$NE;$k++) {
    if( $s1_input              ) { $mubar[$k] = $sig1[$k]/$sigs[$k]; }
    if( $mubar[$k] != 0        ) { $isotropic = 0; }
    if( abs($mubar[$k]) > 1/3. ) { die("error:  abs(mubar) > 1/3") };
  }
# =====> if nloge>0, expand each energy interval
  if( @nloge > 0 ) {
    @e =  @erg;
    #---> energy:  interp in equal log(e)
    undef @erg;
    for($k=0; $k<$NE-1; $k++) {
```

```
      $del = (log($e[$k+1]) - log($e[$k])) / $nloge[$k];
      for($i=0; $i<=$nloge[$k]; $i++) {
        $enew = $e[$k] * exp($i*$del);
        push @erg, $enew;
      }
    }
    #---> xsecs:  linear interp to new energy
    foreach $x (\@sigs, \@sigc, \@sigf, \@nu, \@mubar, \@heat) {
      @z  = @$x;
      undef @$x;
      for($k=0; $k<$NE-1; $k++) {
        $del  = (log($e[$k+1]) - log($e[$k])) / $nloge[$k];
        for($i=0; $i<=$nloge[$k]; $i++) {
          $enew = $e[$k] * exp($i*$del);
          $de   = ($enew - $e[$k]) / ($e[k+1] - $e[k]);
          push @$x, (1-$de)*$z[$k] + $de*$z[$k+1];
        }
      }
      undef @z;
    }
    undef @e;
    $NE = @erg;
  }

# =====> broaden: special opt for scatter:  xsec at 0 K, broaden to $tmp
  if( $broaden ) {
    for($k=0;$k<$NE;$k++) { $sigs[$k] *= &broaden( $erg[$k], $tmp, $awr ); }
  }

# =====> total xsec
  for($k=0;$k<$NE;$k++)   { $sigt[$k] = $sigs[$k] + $sigc[$k] + $sigf[$k]; }

#-------------------------------------------------------------------------------
# =====> Set up nxs(), jxs(), xss() -- USE 1-BASED INDEXING

  @nxs[1..16] = 0;
  @jxs[1..32] = 0;

  $nxs[2] = $za;
  $nxs[3] = $NE;
  $nxs[4] = ($fission) ?  1 : 0;  # fission, or not
  $nxs[5] = ($fission) ?  1 : 0;  # fission, or not

  undef @xss;  $xss[0]=0;   # dummy

                                    $NEXT = 1;
  # ---> ESZ block
  $jxs[1] = $NEXT;
  push @xss, @erg;                  $NEXT += $NE;
  push @xss, @sigt;                 $NEXT += $NE;
  push @xss, @sigc;                 $NEXT += $NE;
  push @xss, @sigs;                 $NEXT += $NE;
  push @xss, @heat;                 $NEXT += $NE;

  if( $fission ) {

    # ---> NU block
    $jxs[2] = $NEXT;
    push @xss, (2, 0, $NE, @erg, @nu);       $NEXT += 3 + 2*$NE;

    # ---> MTR block
    $jxs[3] = $NEXT;
    push @xss, 18;                  $NEXT += 1;

    # ---> LQR block
    $jxs[4] = $NEXT;
    push @xss, 0.0;                 $NEXT += 1;

    # ---> TYR block
    $jxs[5] = $NEXT;
    push @xss, 19;                  $NEXT += 1;

    # ---> LSIG block
    $jxs[6] = $NEXT;
    push @xss, 1;                   $NEXT += 1;

    # ---> SIG block
    $jxs[7] = $NEXT;
    push @xss, (1,$NE, @sigf);              $NEXT += 2 + $NE;
  }
```

```
  # ---> LAND block
  $jxs[8] = $NEXT;
  push @xss, 1;                              $NEXT += 1;
  if(     $isotropic && $fission ) {
    push @xss, 6;                    $NEXT += 1;
  }
  elsif( !$isotropic && $fission ) {
    push @xss, 28;                   $NEXT += 1;
  }

  # ---> AND block
  $jxs[9] = $NEXT;
  # ---> elastic scatter
  if( $isotropic ) {
    push @xss, (2, $erg[0],$erg[$NE-1], 0,0);                       $NEXT += 5;
  }
  else {
    # =====> linearly-anisotropic elastic scatter, abs(mu)<=1/3
    $p1 = (1.  - 3.*$mubar[0]) / 2.;
    $p2 = 0.5;
    $p3 = (1.  + 3.*$mubar[0]) / 2.;
    $pq = ($p1+$p2)/2.;
    push @xss, (2, $erg[0],$erg[$NE-1], -6,-17);            $NEXT += 5;
    push @xss, (2, 3, -1.,0.,1., $p1,$p2,$p3, 0.,$pq,1.);   $NEXT += 11;
    push @xss, (2, 3, -1.,0.,1., $p1,$p2,$p3, 0.,$pq,1.);   $NEXT += 11;
  }
  if( $fission ) {
    push @xss, (2, $erg[0],$erg[$NE-1], 0,0);                       $NEXT += 5;
  }

  if( $fission ) {
    # ---> LDLW block
    $jxs[10] = $NEXT;
    push @xss, 1;                                           $NEXT += 1;

    # ---> DLW block
    $jxs[11] = $NEXT;
    push @xss, (0, 1, 10, 0, 2,$erg[0],$erg[$NE-1], 1.0,1.0);       $NEXT += 9;
    push @xss, (0, 2,$erg[0],$erg[$NE-1],
                   2, .999999*$echi, 1.000001*$echi,
                      .999999*$echi, 1.000001*$echi);               $NEXT += 9;

    # ---> FIS block
    $jxs[21] = $NEXT;
    push @xss, (1,$NE);                                             $NEXT += 2;
    push @xss, @sigf;                                       $NEXT += $NE;
  }

  # =====> total size of xss (not counting $xss[0])
  $jxs[22] = $NEXT - 1;
  $nxs[1]  = $NEXT - 1;

#-------------------------------------------------------------------------------
# =====> print info

  print "\t      zaid = $zaid\n";
  print "\t      za   = $za\n";
  print "\t      file = $file\n";
  print "\t       awr = $awr\n";
  print "\t       tmp = $tmp\n";
  print "\t      echi = $echi\n";
  print "\tenergy pts = $NE\n";
  print "\t  xss size = $nxs[1]\n";
  print "\n \t e \t\t sigt \t\t sigc \t\t sigs \t\t nu \t sigf\n";
  for($k=0; $k<$NE; $k++) {
    for($j=0; $j<4; $j++) { printf "\t%-12.4e", $xss[1+$k+$j*$NE]; }
    printf "\t%5.2f\t%-12.4e\n", $nu[$k], $sigf[$k];
  }

  # =====> xsdir info
  print  "\n\tXSDIR Info, to use on XSn card::\n\n";
  printf "\tXSn  %10s %-12.6g %-10s 0 1 1 %d  0 0 %-10.6g \n\n",
              $zaid, $awr, $file, $nxs[1], $tmp;

#-------------------------------------------------------------------------------
# =====> create the ACE file

print  "\n\tCreating ACE file:  $file\n\n";

open(ACE,">$file") || die("error - can't open file: $file\n");
```

```
printf ACE "%10s %11.6g %11.5e %10s\n", $zaid, $awr, $tmp, $DATE;
printf ACE "%-70s%10s\n",               $comment, $DATE;
for($k=0;$k<4;$k++) {
  printf ACE "%7d%11f%7d%11f%7d%11f%7d%11f\n", 0,0, 0,0, 0,0, 0,0;
}
for($k=1;$k<17;$k++) {
  printf ACE "%9d",     $nxs[$k];   if($k%8==0) {print ACE "\n";}
}
for($k=1 ;$k<33;$k++) {
  printf ACE "%9d",     $jxs[$k];   if($k%8==0) {print ACE "\n";}
}
for($k=1; $k<=$nxs[1]; $k++) {
  printf ACE "%20.12g", $xss[$k];   if($k%4==0) {print ACE "\n";}
}
if( $nxs[1]%4 == 1 ) { printf ACE "%60s\n"," "; }
if( $nxs[1]%4 == 2 ) { printf ACE "%40s\n"," "; }
if( $nxs[1]%4 == 3 ) { printf ACE "%20s\n"," "; }

close(ACE);

#-------------------------------------------------------------------------------
sub get_list() {
  my @list;
  while( @ARGV ) {
    $arg = shift(@ARGV);
    if( $arg =~ /^-[a-z]/i ) { unshift(@ARGV,$arg); last; }
    push @list, $arg;
  }
  return @list;
}
#-------------------------------------------------------------------------------
sub get_date {
  my( $y,$m,$d, $date );
  ($y,$m,$d) = (localtime)[5,4,3]; $y+=1900; $m+=1;
  $date = sprintf "%4.4d-%2.2d-%2.2d", $y,$m,$d;
  return  $date;
}
#-------------------------------------------------------------------------------
sub broaden() {
  my $e = $_[0];
  my $t = $_[1];
  my $a = $_[2];
  my $x;
  $x = sqrt( $a*$e/$t );
  return  (1.+.5/$x**2) * &erf($x) + exp(-$x**2)/(1.77245385*$x);
}
#-------------------------------------------------------------------------------
sub erf() {
  #---> erf approx from Hastings
  my $x = $_[0];
  my $a1=.0705230784, $a2=.0422820123, $a3=.00927052721;
  my $a4=.0001520143, $a5=.0002765672, $a6=.00004306381;
  return  1.  - 1./(1.+$x*($a1+$x*($a2+$x*($a3+$x*($a4+$x*($a5+$x*$a6))))))**16;
}
#-------------------------------------------------------------------------------
```

Listing of the simple_ace_mg.pl script:

```
#!/usr/bin/perl
#########################################################################
#
# simple_ace_mg.pl -- Create a simple multigroup ACE file with NG groups
#
# USAGE:
#    simple_ace_mg.pl  -zaid za   -groups n    -awr x  -tmp x   \
#                      -e list    -t  list  -s  list   -c list  \
#                      -f list    -nu list  -s1 list -chi list  \
#                      -comment str     -file str
#
# INFO:
#  * ZAID must be of the form   nnnnn.iim,  nnnnn,ii = integers
#  * consider capture, fission, isotropic scatter ONLY
#  * no delayed neutrons
#
#  * Group 1 is highest energy, group NG is lowest energy
#  * NG+1 energy bounds must be supplied in DECREASING order
#  * Default energy group structure (in MeV):
#      1 group:       100,  0.
#      2 group:       100,  .625e-6,  0.
#      other:         no default, error if not supplied
#
#  * Must include entries for xsecs. If all are constant, only
#    1 entry is needed; otherwise, must have NG entries in list:
#      -t   list      total   xsec
#      -c   list      capture xsec (not including fission)
#      -f   list      fission xsec
#      -nu  list      nubar, assumed 0 if fission is 0
#      -chi list      chi,   assumed 0 if fission is 0
#      -s   list      SEE BELOW
#      -s1  list      SEE BELOW
#
#  * FOR NOW, ONLY ISOTROPIC OR P1 SCATTER IS PERMITTED.
#      The list supplied for -s must have NG**2 entries
#      for the P0 group-to-group scatter xsec in this order:
#             1-->1,  1-->2,  ..., 1-->NG,
#             2-->1,  2-->2,  ..., 2-->NG,
#             ...
#             NG-->1, NG-->2, ..., NG-->NG,
#       FOR P1 SCATTER, ABS(sig1/sigs) MUST BE < 1/3
#
# HISTORY:
#   2015-12-14 - fbb, initial version
#########################################################################
# EXAMPLE
#
#  ./simple_ace_mg.pl  -zaid 22089.01m  \
#                      -comment 'la-ur-12-22089 analytic problem 1' \
#                      -groups 2   -e  100. 10. 0.      \
#                      -t   2. 3.   -c  .5 1.  -f .5 1. -nu .75 4.5 \
#                      -chi 1. 0.   -s  .5 .5  0. 1.
#
#########################################################################

print "\n=====> simple_ace_mg.pl - create simple multigroup ACE file\n\n";

# =====> get args & expand as needed
  while(   $arg = shift @ARGV ) {
    if(    $arg eq '-file'    ) { $file     = shift(@ARGV); }
    elsif( $arg eq '-zaid'    ) { $zaid     = shift(@ARGV); }
    elsif( $arg eq '-groups'  ) { $NG       = shift(@ARGV); }
    elsif( $arg eq '-comment' ) { $comment  = shift(@ARGV); }
    elsif( $arg eq '-e'       ) { @erg      = &get_list;    }
    elsif( $arg eq '-t'       ) { @sigt     = &get_list;    }
    elsif( $arg eq '-c'       ) { @sigc     = &get_list;    }
    elsif( $arg eq '-f'       ) { @sigf     = &get_list;    }
    elsif( $arg eq '-nu'      ) { @nu       = &get_list;    }
    elsif( $arg eq '-chi'     ) { @chi      = &get_list;    }
    elsif( $arg eq '-s'       ) { @sigs     = &get_list;    }
    elsif( $arg eq '-s1'      ) { @sig1     = &get_list;    }
    elsif( $arg eq '-awr'     ) { $awr      = shift(@ARGV); }
    elsif( $arg eq '-tmp'     ) { $tmp      = shift(@ARGV); }
    elsif( $arg eq '-bins'    ) { $NBINS    = shift(@ARGV); }
    else                        { die("error:  bad arg: $arg"); }
  }

# =====> defaults & checks
```

```
  if(  ! $zaid      ) { $zaid    = "99999.99m"; }
  if(  ! $file      ) { $file    = $zaid; }
  if(  ! $NG        ) { $NG      = 1; }
  if(  ! $NBINS     ) { $NBINS   = 1000; }
  if(  ! $comment   ) { $comment = "multigroup ACE file"; }
  if(  ! $awr       ) { $awr     = 1000000.0; }
  if(  ! $tmp       ) { $tmp     = 2.5301e-8; }
  if(  $tmp > 1.0   ) { $tmp    *= 8.617385e-11; } # Kelvin if >1, convert to MeV
  if(  !@erg && $NG==1 ) { @erg     = ( 100., 0. ); }
  if(  !@erg && $NG==2 ) { @erg     = ( 100., .625e-6, 0. ); }
  ($za=$zaid) =~ s/\..*$//;         # remove the suffix
  $comment    =  substr($comment,0,70);
  $DATE       =  &get_date;
  if(  $za  > 99999             ) { die("error:  za > 99999\n"); }
  if(  -s $file                 ) { die("error:  file exists: $file\n"); }
  if(  substr($zaid,-1,1) ne "m" ) { die("error:  zaid must end in m\n"); }
  if(  !@erg                    ) { die("error:  must specify energies\n"); }
  if(  @erg != $NG+1            ) { die("error:  bad count for energies\n"); }
  if(  $erg[0] < $erg[$NG]      ) { die("error:  energies are not decreasing\n");}

# =====> number of entries in scatter matrix
  $NS        = $NG * $NG;
  $ISOTROPIC = (@sig1>0) ? 0 : 1;     # true if scatter is isotropic

# =====> expand constants to match $NG
  foreach $sig (\@sigt, \@sigc, \@sigf, \@nu, \@chi) {
    if(    @$sig==0   ) { for($k=0;$k<$NG;$k++) { $sig->[$k]  = 0.0;       } }
    elsif( @$sig==1   ) { for($k=1;$k<$NG;$k++) { $sig->[$k]  = $sig->[0]; } }
    elsif( @$sig!=$NG ) { die("error:  groups & sig  don't match"); }
  }
  foreach $sig (\@sigs, \@sig1) {
    if(    @$sig==0   ) { for($k=0;$k<$NS;$k++) { $sig->[$k]  = 0.0;       } }
    elsif( @$sig==1   ) { for($k=1;$k<$NS;$k++) { $sig->[$k]  = $sig->[0]; } }
    elsif( @$sig!=$NS ) { die("error:  groups & sigs  don't match"); }
  }

# =====> set flag for fission or not, then normalize chi()
  $FISSION = 0;
  for($k=0;$k<$NG;$k++) { if( $sigf[$k]>0 ) { $FISSION=1; last; } }
  if( $FISSION ) {
    $s = 0.0;
    for($k=0; $k<$NG; $k++) { $s       += $chi[$k]; }
    for($k=0; $k<$NG; $k++) { $chi[$k] /= $s;       }
  }
  else {
    for($k=0; $k<$NG; $k++) { $chi[$k]=0.; $nu[$k]=0.; }
  }

# =====> make sure no xsec data is < 0
  foreach $x (\@erg, \@sigt, \@sigc, \@sigf, \@nu, \@chi, \@sigs) {
    for($k=0; $k<@$x; $k++) {
      if( $x->[$k] < 0.0 ) { die("error:  negative xsec data not allowed\n"); }
    }
  }
#-------------------------------------------------------------------------------
# =====> Set up nxs(), jxs(), xss() -- USE 1-BASED INDEXING

  # nxs[1] set at end....                 # size of xss()
  $nxs[ 2] = $za;                         # 1000*z + a
  $nxs[ 3] = ($ISOTROPIC)? 0 : $NBINS+1;  # angular distrib vars, nleg
  $nxs[ 4] = 0;                           # number of edit reactions, nedit
  $nxs[ 5] = $NG;                         # groups
  $nxs[ 6] = $NG - 1;                     # upscatter   groups
  $nxs[ 7] = $NG - 1;                     # downscatter groups
  $nxs[ 8] = 0;                           # no secondary particles
  $nxs[ 9] = 0;                           # angular:  equiprobable bins
  $nxs[10] = 1;                           # total nu onlY
  $nxs[11] = 0;                           # standard transport, no BFP
  $nxs[12] = 1;                           # neutrons
  @nxs[13..16] = 0;                       # unused

  $jxs[ 1] = 1;                           # lerg,  energy structure
  $jxs[ 2] = $jxs[ 1] + 2*$NG;            # ltot,  loc of total   xsecs
  $jxs[ 3] = $jxs[ 2] +   $NG;            # lfiss, loc of fission xsecs
  $jxs[ 4] = $jxs[ 3] +   $NG;            # lnu,   loc of nu-bar data
  $jxs[ 5] = $jxs[ 4] +   $NG;            # lchi,  loc of chi    data
  $jxs[ 6] = $jxs[ 5] +   $NG;            # labs,  loc of absorption (capture)
  @jxs[7..12] = 0;
  $jxs[13] = $jxs[ 6] +   $NG;            # lpol,  loc of P0 locators
  @jxs[14,15] = 0;
  $jxs[16] = $jxs[13] +   $NG**2 + 1;     # lxpnl, loc of XPn locators
```

```
  if( $ISOTROPIC ) {
    $jxs[17] = $jxs[16] + 1;
  }
  else {
    $jxs[17] = $jxs[16] + $NG**2+1;
  }
  @jxs[18..32] = 0;

  # ---> xss() array
  for($k=0; $k<$NG; $k++) {
    # ---> energy group midpoints & widths, DESCENDING ORDER
    $xss[ $jxs[1]    +$k ] = .5 * ($erg[$k] + $erg[$k+1]);
    $xss[ $jxs[1]+$NG+$k ] =       $erg[$k] - $erg[$k+1] ;

    # ---> multigroup xsec data
    $xss[ $jxs[2]    +$k ] = $sigt[$k];
    $xss[ $jxs[3]    +$k ] = $sigf[$k];
    $xss[ $jxs[4]    +$k ] = $nu[$k];
    $xss[ $jxs[5]    +$k ] = $chi[$k];
    $xss[ $jxs[6]    +$k ] = $sigc[$k];
  }

  # ---> scatter matrix block
  $xss[ $jxs[13]  ] = $jxs[13] + 1;

  for($i=0; $i<$NG; $i++) {      # init group
    for($j=0; $j<$NG; $j++) {    # exit group
      $xss[ $jxs[13]+1+$i*$NG+$j  ] = $sigs[$i*$NG+$j];
    }
  }
  if( $ISOTROPIC ) {
    $xss[ $jxs[16]  ] = 0;
    $xss[ $jxs[17]  ] = 0;
    $nxs[1]           = $jxs[17];
  }
  else {
                                                $NEXT16  = $jxs[16] + 1;
    $xss[ $jxs[16]  ] = $NEXT16;
                                                $NEXT17  = $jxs[17] + 1;
    $xss[ $jxs[17]  ] = $NEXT17;
                                                $LOFF    = 1;
    for($i=0; $i<$NG; $i++) {      # init group
      for($j=0; $j<$NG; $j++) {    # exit group
        $k = $i*$NG + $j;
        if( $sigs[$k] == 0.0 ) {;
          $xss[$NEXT16] = -1.0;                         $NEXT16 += 1;
          next;
        }
        if( $sig1[$k] == 0.0 ) {;
          $xss[$NEXT16] = 0.0;                          $NEXT16 += 1;
          next;
        }
        $xss[$NEXT16] = $LOFF;                          $NEXT16 += 1;
                                                $LOFF   += $NBINS+1;
        $A[0] = 1.0;
        $A[1] = $sig1[$k] / $sigs[$k];
        $N    = 1;
        @MUS  = &get_equiprob_bins( $NBINS, $N, @A );
        for($m=0;$m<=$NBINS;$m++) {
          $xss[$NEXT17] = $MUS[$m];                 $NEXT17 += 1;
        }
      }
    }
    $nxs[1] = $NEXT17 - 1;
  }
#-------------------------------------------------------------------------------
# =====> print info:

  print "\t      zaid = $zaid\n";
  print "\t      za   = $za\n";
  print "\t      file = $file\n";
  print "\t       awr = $awr\n";
  print "\t       tmp = $tmp\n";
  print "\t    groups = $NG\n";
  print "\t  xss size = $nxs[1]\n";
  print "\t   comment = $comment\n";
  print "\t      date = $DATE\n\n";

  $HEAD= "  %5s%9s%9s%12s%12s%12s%12s%12s%12s\n";
  $NUMS= "  %5d%9.3g%9.3g%12.4g%12.4g%12.4g%12.4g%12.4g%12.4g\n";
  printf $HEAD, "group", "Ehi", "Elow", "total", "capture",
```

```
              "scatter", "fission", "nu", "chi";
  for($k=0; $k<$NG; $k++) {
    $ehi  = $erg[$k];
    $emid = $xss[ $jxs[1]+$k ];
    $elow = $erg[$k+1];
    $t    = $xss[ $jxs[2]+$k ];
    $f    = $xss[ $jxs[3]+$k ];
    $n    = $xss[ $jxs[4]+$k ];
    $x    = $xss[ $jxs[5]+$k ];
    $c    = $xss[ $jxs[6]+$k ];
    $s = 0; for($i=0; $i<$NG; $i++) { $s += $xss[ $jxs[13]+1+$k*$NG+$i ]; }
    printf $NUMS, $k+1, $ehi,$elow, $t, $c, $s, $f, $n, $x;
  }
  if( $NG > 1 ) {
    printf "\n  scattering matrix,   group-I (down) --> group-J (across)\n";
    printf   "     J -->";
    for($k=0;$k<$NG;$k++) { printf "%12d",$k+1; }
    printf "\n     I --v\n";
    for($k=0;$k<$NG;$k++) {
      printf "%10d",$k+1;
      for($i=0;$i<$NG; $i++) { printf "%12g", $xss[$jxs[13]+1+$k*$NG+$i]; }
      printf "\n";
    }
  }
  $TOL = 1.e-5;
  for($k=0;$k<$NG;$k++) {
    $t    = $xss[ $jxs[2]+$k ];
    $f    = $xss[ $jxs[3]+$k ];
    $c    = $xss[ $jxs[6]+$k ];
    $s = 0; for($i=0; $i<$NG; $i++) { $s += $xss[ $jxs[13]+1+$k*$NG+$i ]; }
    if( abs($t-($c+$f+$s)) > $TOL ) {
      print "\n\tGroup %6d:\tCap+Fis+Scat differs from Total by >%6g\n",$k+1,$TOL;
    }
  }
  # =====> xsdir info
  print  "\n  XSDIR Info, to use on XSn card:\n\n";
  printf "\tXSn  %10s %-12.6g %-10s 0 1 1 %d  0 0 %-10.6g \n\n",
              $zaid, $awr, $file, $nxs[1], $tmp;
#-------------------------------------------------------------------------------
# =====> ACE file creation

  print  "\n  Creating ACE file:  $file\n\n";

  open(ACE,">$file") || die("error - can't open file: $file\n");

  printf   ACE "%10s %11.6g %11.5e %10s\n", $zaid, $awr, $tmp, $DATE;
  printf   ACE "%-70s%10s\n",               $comment, $DATE;
  for($k=0; $k<4;$k++) {
    printf ACE "%7d%11f%7d%11f%7d%11f%7d%11f\n", 0,0, 0,0, 0,0, 0,0;
  }
  for($k=1; $k<17;$k++) {
    printf ACE "%9d",     $nxs[$k];  if($k%8==0) {print ACE "\n";}
  }
  for($k=1; $k<33;$k++) {
    printf ACE "%9d",     $jxs[$k];  if($k%8==0) {print ACE "\n";}
  }
  for($k=1; $k<=$nxs[1]; $k++) {
    printf ACE "%20.12g", $xss[$k];  if($k%4==0) {print ACE "\n";}
  }
  if( $nxs[1]%4 == 1 ) { printf ACE "%60s\n"," "; }
  if( $nxs[1]%4 == 2 ) { printf ACE "%40s\n"," "; }
  if( $nxs[1]%4 == 3 ) { printf ACE "%20s\n"," "; }
  close(ACE);
#-------------------------------------------------------------------------------
# =====> utility routines
sub get_list() {
  my( $arg, @list );
  while( @ARGV ) {
    $arg = shift(@ARGV);
    if( $arg =~ /^-[a-z]/i ) { unshift(@ARGV,$arg); last; }
    push @list, $arg;
  }
  return @list;
}
#-------------------------------------------------------------------------------
sub get_date {
  my( $y,$m,$d, $date);
  ($y, $m, $d) = (localtime)[5,4,3]; $y+=1900; $m+=1;
  $date = sprintf "%4.4d-%2.2d-%2.2d", $y,$m,$d;
  return $date;
}
```

```
#-------------------------------------------------------------------------------
sub get_equiprob_bins {
  my( $N, @A, $nbins, $nx, $dx, @x, @cdf, $j, $k, $pp, $prob, @mu );
  $nbins = shift( @_ );
  $N     = shift( @_ );
  @A     = @_;
  #---> get crude CDF, may not be strictly increasing
  $nx = 1000;
  $dx = 2./$nx;
  for($k=0; $k<=$nbins; $k++) {
    $x[$k] = -1.0 + $k*$dx;
    #---> compute cdf(x) for Nth order Legendre expansion
    $cdf[$k] = 1. + $x[$k];
    if( $N > 0 ) {
      $P[0] = 1.0;
      $P[1] = $x[$k];
      for($j=2;$j<=$N+1;$j++) { $P[$j] = ((2.*$j-1.)*$x[$k]*$P[$j-1]
                                          - ($j-1.)*$P[$j-2])/$j;}
      for($j=1;$j<=$N;$j++)   { $cdf[$k] += $A[$j] * ($P[$j+1] - $P[$j-1]); }
    }
    $cdf[$k] *= 0.5;
  }
  #---> find equiprob step boundaries
  $prob = 1. / $nbins;
  $mu[0] = -1.;
  for($k=1;$k<=$nbins;$k++) {
    $pp = $prob*$k;
    for($j=$nbins;$j>=0;$j--) {
      if( $cdf[$j] < $pp ) {
        $mu[$k] = $x[$j] + ($pp-$cdf[$j])*($x[$j+1]-$x[$j])/($cdf[$j+1]-$cdf[$j]);
        last;
      }
    }
  }
  return @mu;
}
#-------------------------------------------------------------------------------
```