ORNL/TM-2016/43

AMPX-6: A Modular Code System for Processing ENDF/B

D. Wiarda
M. E. Dunn
N. M. Greene
M. L. Williams
C. Celik
L. M. Petrie

April 2016

ORNL/TM-2016/43

Reactor and Nuclear Systems Division

AMPX-6: A MODULAR CODE SYSTEM FOR PROCESSING ENDF/B EVALUATIONS

D. Wiarda
M. E. Dunn
N. M. Green
M. L. Williams
C. Celik
L. M. Petrie

April 2016

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-BATTELLE, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

AMPX is a modular system of computer programs used for nuclear analysis with a primary emphasis on tasks associated with the production and use of multigroup (MG), continuous energy (CE) cross sections, depletion/decay libraries, and covariance data. AMPX accepts basic cross-section data from cross-section evaluations in the international Evaluated Nuclear Data File (ENDF/B) Format. AMPX can be used to generate a variety of MG libraries that can be used with modern transport codes to perform nuclear analyses. CE or point cross section libraries can be produced for use in Monte Carlo codes and other applications. Also, AMPX provides cross-section uncertainty or covariance data for use with sensitivity/uncertainty analysis tools. Furthermore, AMPX can be used to process ENDF/B evaluations to produce depletion and decay libraries needed by depletion codes such as ORIGEN. In addition, AMPX has an internal MG format that can be read by various AMPX modules and codes outside the system. For example, the Standardized Computer Analyses for Licensing Evaluations (SCALE) system has the capability to read AMPX formatted cross section libraries. This manual provides procedures for producing nuclear data libraries needed by SCALE and other radiation transport packages.

1. INTRODUCTION

1.1 HISTORY

The AMPX system [1] has existed since the early 1970s and was developed at the Oak Ridge National Laboratory (ORNL) under sponsorship of the Defense Nuclear Agency (DNA). The primary objective of the early system was to produce coupled MG neutron-gamma-ray cross section libraries needed for deep penetration shielding applications.

Prior to AMPX, the generation of coupled cross section libraries required the use of three independent computer codes:

1. XLACS [2], which produces MG neutron cross section libraries,

2. MUG [3], which produces MG gamma-ray cross section libraries, and

3. POPOP4 [4], which produces coefficients needed to determine gamma-ray sources arising from neutron reactions.

The libraries produced by these codes were written in a special format required either by the ANISN [5] and DOT [6] discrete ordinates transport codes or the MORSE [7] MG Monte Carlo code. In the case of XLACS, the library was written in a format that had to be used in a discrete ordinates code known as XSDRN [8] before an ANISN-formatted library could be written. In order to develop the desired MG library, the three classes of data were combined to produce a single coupled neutron-gamma-ray library. The production of a coupled library was laborious and time consuming.

Many similar procedures in the 1960–1970 time period required the use of computer programs that were just as inconvenient and difficult to use. As a result, several organizations developed a new approach for code development known as modular programming. These systems were organized to make it easier to execute sequences of computer programs. Notable among these systems is the DATATRAN [9] system developed at Knolls Atomic Power Laboratory (KAPL), the ARC [10] system developed at Argonne National Laboratory (ANL), and the JOSHUA [11] system developed at the Savannah River Site (SRS).

The AMPX system developed at ORNL borrowed heavily from the ideas and concepts of the early modular cross section processing systems. AMPX is an acronym for Automation of MUG, POPOP4, and

XLACS. During the early 1970s, these three codes comprised the core capabilities needed to generate coupled libraries, but these codes were augmented with a variety of other programs that perform resonance self-shielding, convert library formats, combine collections of cross section data, etc.

The AMPX code system was initially released in 1973 and distributed at ORNL by the Radiation Shielding Information Center (RSIC), the predecessor of the current Radiation Safety Information Computation Center (RSICC). The AMPX code system was distributed by RSIC to many facilities in the United States and throughout the world. Following the initial release, the AMPX code system did not have dedicated funding in the years subsequent to the release. Consequently, further improvements to the AMPX system were supported at a very low level by many projects that dictated the development of new processing procedures.

The development of the Standardized Computer Analyses for Licensing Evaluations (SCALE) system [12] for nuclear analyses was initiated at ORNL under the sponsorship of the Nuclear Regulatory Commission (NRC) in 1973. Many SCALE analysis tools were taken from the AMPX system. Moreover, these tools or codes have been continuously refined and supported over the years. In fact, tasks needed for the SCALE system have indirectly provided a substantial fraction of the support for improvements to the AMPX system over the years. Due to further enhancements of the cross section processing capabilities, an upgraded AMPX code system was released by RSIC in 1978 as code package PSR-63/AMPX-II [13].

For the next fourteen years, AMPX-II was used to produce MG cross section libraries for nuclear applications until the release of AMPX-77 [14] in 1992. The “77” indicates that all modules of the AMPX system were developed under the FORTRAN-77 programming standard. The AMPX-77 development was driven by cross section generation tasks associated with processing Version V of the Evaluated Nuclear Data File (ENDF/B) [15]. In particular, AMPX-77 was used to generate the 238-group LAW [16] library that has cross sections for more than 300 ENDF/B-V isotopes/nuclides.

The ENDF/B release in 1990 used a new collection of formats known as Version 6 [17] that differ from the format used with the release of the ENDF/B-V library in 1978. The ENDF/B-6 Format has been used with all ENDF/B libraries beginning with ENDF/B-VI and continuing through the latest ENDF/B library. Because of a lack of dedicated funding between 1990 and 1996, AMPX had not kept pace with the newer ENDF/B format after the release of the ENDF/B-V library. In an effort to update the AMPX code system, the NRC task ORNL in 1996 to develop a new AMPX cross-section processing system with the capability to process ENDF/B formats through Version 6.

In the past several years, dedicated funding from the US DOE Nuclear Criticality Safety Program (NCSP) and subsequently the NRC enabled ORNL to modernize the AMPX cross-section system. One of the initial objectives of the code modernization effort was to bring AMPX under a formal software quality assurance plan (SQAP). To achieve this objective, ORNL made the decision to merge the AMPX software repository (including software configuration control) and build system with SCALE. Merging the AMPX and SCALE development infrastructure offers many advantages with the primary motivation being software quality assurance. The merger also allows AMPX to use the same continuous integration testing that SCALE now uses [18]. All SCALE development and testing is performed under the SCALE QAP. Moreover, the SCALE QAP specifies the development infrastructure and testing harness that is used to meet the software QA requirements. By merging the AMPX repository and build with SCALE, all of the AMPX development and testing is now performed under the SCALE QA infrastructure. As a result, AMPX development is performed under the SCALE QAP. Although AMPX and SCALE are now developed under the same infrastructure, both code packages will remain separate code packages that can be distributed together or as separate packages. As a result, the development infrastructure merger should be transparent to the user, but the benefit is improved quality and overall efficiency of SCALE and AMPX development.

In addition to the development infrastructure improvements with AMPX, efforts are in progress to modernize the entire cross-section processing system. With the current release of AMPX, portions of the AMPX code system have been converted to C++, taking advantage of modern memory management capabilities and increased computing power. As AMPX capabilities are modernized, these new modern code updates will be released with the AMPX package. Recent modernization efforts have focused on the collision kinematics processing capabilities. Since AMPX has a much longer tradition for generating MG libraries than CE libraries, some processing of the scattering kinematic data was geared more towards MG processing. Previously, kinematic scattering data were processed for use in MG libraries and then converted back for use in CE libraries. Some fidelity was lost in these conversions. The modernization allowed for substantial upgrade of the processing for kinematic data. Since the generation of kinematic data uses many low-level routines, many of these routines have also been updated to C++. This release of the AMPX code package includes modernized collision kinematics processing capabilities. The following section provides an overview of cross-section processing fundamentals along with an overview of the current AMPX capabilities.

2. OVERVIEW

In this section, a qualitative discussion of cross sections is provided, along with various associated cross section operations. In most cases, the discussion is limited to neutron cross sections while most observations apply to other particles.

2.1 WHAT ARE CROSS SECTIONS?

Cross sections are parameters that characterize how particles interact with matter. CE or point cross sections characterize a particle traveling at a particular speed in a particular direction. Upon interaction with a target nucleus, the particle can change speed and/or direction. The total cross section quantifies the probability that a neutron will interact with a target nucleus. The total cross section is the sum of the cross sections that define the probability of having any one of several types of interactions. In the case of neutrons, there may be a hundred or more possible reactions that can be categorized as either absorption or scattering interactions.

If a particle is absorbed by a target nucleus, a new compound nucleus is formed that subsequently decays by emitting an energetic particle(s) such as an (n,γ) reaction. In neutron transport codes, fission is typically treated as an absorption reaction, and the neutron multiplicity data (i.e. $\bar{\nu}$) are used to determine the number of secondary neutrons produced by the fission event. With regard to scattering, a particle may interact with the nucleus without penetrating the nucleus. In this type of collision, the particle scatters in a billiard-ball fashion with the nucleus, and the reaction is referred to as *potential scattering*. The remaining types of scattering events are elastic and inelastic scattering. During an elastic scattering reaction, the particle is absorbed by the nucleus and subsequently re-emitted, leaving the target nucleus in the ground state. As a result, the kinetic energy is conserved for the elastic scattering reaction. For an inelastic scattering reaction, the incident particle is absorbed by the nucleus, and a compound nucleus is formed. The excited nucleus subsequently decays by emitting an energetic particle, thereby leaving the nucleus in an excited state. Unlike the elastic collision, the kinetic energy of the inelastic scattering reaction is not conserved.

A cross section by itself is not enough information to use in a particle transport calculation. The total cross section and its components provide enough information to describe the possible interactions; however, determining the energy and angle of secondary particles requires a knowledge of the reaction kinematics, or the physics must be presented in a structured manner (e.g., tables of energy and angle distributions of secondary particles) to determine the exiting energy and angle.

The AMPX code system can generate MG libraries, whose uses are described in the next section, as well as CE libraries.

2.2 THE MG APPROACH

In the 1940s and most of the 1950s, nuclear analyses were performed without point cross section data because the data simply did not exist. Due to the limited computing resources of the time period, transport calculations that use CE cross sections were not feasible.

Indeed, many calculations were very simple and would not be described as either point or MG calculations. For these calculations, the cross sections were assigned a single value that was selected as the value for the most important energy range for a specific process of a particular isotope. As an example for a thermal reactor calculation, the fission cross section for ^{235}U would be the average value in the thermal region while the fission cross section for ^{238}U would be set to zero.

In the early 1950s, the group diffusion method was developed. This method combines slowing-down theory for energy degradation with diffusion theory for the spatial variation of the neutron flux [19]. In the treatment, the full energy range is divided into a number of energy intervals (or groups), and the neutron is assumed to suffer the average number of elastic scattering collisions required to slow down through an energy group prior to emerging in the next energy group. The number of energy groups and associated group constants that were used in this early method is unknown; however, approximate methods were most likely used to obtain the group constants.

The simplest of the group diffusion methods is the one-group diffusion theory method that served as the basis for many early nuclear analyses. Two-group diffusion theory is well documented and may represent the limit of the number of energy groups that could be handled without computer resources.

The MG approach became prominent in the 1950s, and many MG cross section libraries were introduced, including the 6- and 16-group Hansen-Roach libraries [20,21], the 26-group Russian Bondarenko library [22], the 30-group THERMOS library [23], and the 99-group GAM-II library [24]. Interestingly, the growth of the MG methods paralleled the advancements in computer computational capabilities. The development of the numerical computational methods in parallel with advances in computer capabilities has made the hand calculation methods more feasible.

Four components are required to create a MG cross section library:

point cross section data,

MG energy structure,

flux or weighting spectrum, and

analytic description or tabular description of reaction kinematics.

The subsequent sections discuss the four components needed to generate a MG cross section library.

2.2.1 MG Structure

The inherent assumption of the MG approach is that an energy quadrature can be defined so that the cross section variation in an energy group can be adequately represented by a single average value. Years of calculation experience have revealed that the inherent assumption of the MG approach is valid for many conditions; however, there are some configurations where a particular MG structure does not yield acceptable results. In practice, MG structures are typically classified as either a fine or broad group structure. A fine group structure has a sufficient number of energy groups that can be used to analyze a wide variety of problems. Conversely, a broad group structure has a sufficient number of groups that can be used for a specific class of problems as determined by the developer of the cross section library.

The choice of a group structure and associated weighting spectrum are strongly correlated. In an ideal situation, the importance of the weighting spectrum should be minimized with a larger number of groups because the cross section variation should decrease as the energy band becomes smaller. However, experience with some isotopes has revealed that an ideal group structure does not exist because of the rapid cross section variation as a function of energy.

Fine group structures may require between 100 and 30,000 energy bins, and there are several fine group libraries with 200 to 500 energy groups. These fine group libraries can be used to treat a wide variety of problems; however, some specific energy regions and isotopes/nuclides demand additional treatments for extreme cross section variations as a function of energy.

For many applications, a broad group library is directly applicable and suitable for production analyses. Furthermore, some two-dimensional (2-D) or three-dimensional (3-D) transport calculations with a fine group library can be prohibitive even with current computational capabilities, and a broad group library must be used. Broad group libraries are generated or collapsed from a fine group library and are generally applicable to a limited class of problems. The techniques used to weight the fine group data and subsequently produce a broad group library require a weighting spectrum similar to the spectra of the intended class of problems (e.g., a light-water reactor spectrum).

As noted above, there is not a single fine group energy structure that can be used for all problems that eliminates the need for a weighting spectrum. The selection of a fine group structure is based on the following criteria:

a. thresholds of important reactions for specific isotopes/nuclides in the group structure (e.g., if resonance integrals are frequently calculated, the 0.625 eV boundary should be included to account for the cadmium cutoff value),

b. maintaining the capability to reproduce a previously defined broad group library from the new general purpose library (i.e., retrofitting capability),

c. experience with previous cross section evaluations,

d. treatment of important resonances (e.g., isolate the 6.67 eV ^{238}U resonance into one or more energy groups) is required.

e. treatment of dips or windows in the cross section data (For shielding applications, the low cross section values can provide an energy range for neutron streaming; there is a cross section dip below the 25 keV resonance for ^{56}Fe.), and

f. treatment of resonances at low energies. (For example, ^{239}Pu and ^{241}Pu have resonances in the thermal energy region, and most resonance treatments assume free-atom elastic scattering to calculate the slowing-down effects. Moreover, these resonances are in an energy region where the proper slowing-down kinematics should be used. Therefore, enough groups should be selected to explicitly calculate the effects of the low energy resonances.)

2.2.2 Weighting Spectrum Selection

Selection of an appropriate weighting spectrum is crucial for collapsing point cross section values to a MG cross section library. MG parameters are characterized by three classes of information:

1. group-averaged parameters such as the average value of a fission or elastic scattering cross section for a specific energy group;

2. group-to-group scattering parameters that define the particle scattering between energy groups (These MG parameters describe the scattering transfer as a function of scattering angle. Regarding the angle dependence, the parameters are provided in a matrix form so that the individual terms are the coefficients of a Legendre fit to the scattering terms. As an important note, if a cross section value is flat within an energy group, the scattering distribution and associated terms of the scattering matrix depend on the weighting spectrum. If the weighting spectrum is higher at the low end (in energy) of an energy group, the out-of-group transfer is emphasized. Conversely, if the weighting spectrum is higher at the upper end (in energy) of the energy group, the within-group transfer is emphasized.); and

3. additional MG parameters such as the fraction of fission neutrons born in energy group g, χ_g, or the average number of fission neutrons produced in energy group g, $\overline{\nu}$.

Each of the above parameters is directly affected by the selection of the weighting spectrum, and the use of a fine group structure can reduce the effects of the weighting spectrum on the individual parameters.

Although the selection of a weighting spectrum for generating MG cross sections is important, there is an additional cross section processing consideration that may reduce the potential problems associated with the generation of a MG library using an inapplicable weighting spectrum. In the resonance region, the cross section resonances significantly affect the spectrum that a nuclide will "see." In particular, the resonance of a nuclide will "shield" a nuclide from "seeing" a 1/E flux that would normally be present in the absence of the resonance in the slowing-down region. As a result, there are a variety of procedures that can be used to account for the effects associated with the resonances in the slowing-down region. The treatment of the shielding effects in the resonance region is referred to as "resonance self-shielding."

Most fine-group libraries (at least in AMPX) are generated based on a weighting spectrum that is constructed by splicing 4 simple spectra together. At low energies (arbitrarily chosen to be below 0.125 eV), the weighting function is a Maxwellian spectrum which has a flux shape that assumes the neutron scatters into a region with a free gas scatterer that has no absorption. The Maxwellian flux spectrum has the form:

$$\phi(E) = M(E) = Ee^{-\frac{E}{kT}}$$

Error! Bookmark not defined.1

where

E = energy,

k = Boltzmann constant,

T = temperature in Kelvin.

In the slowing down range, $0.125 < E < E_{cut}$, the weighting spectrum is assumed to be $\phi(E) = 1/E$. The cutoff energy E_{cut} for the slowing down range must be selected and is typically 83 keV by default in the AMPX modules. In the region, $E_{cut} < E < 10^7$ eV, where fission neutrons are born, the following fission spectrum is used:

$$\phi(E) = \chi(E) = E^{0.5} e^{-\frac{E}{\theta}} \quad \text{Error! Bookmark not defined.2}$$

where

θ = temperature of the fission spectrum (e.g., 1.2×10^6 eV).

For energies above 10^7 eV, the particles are considered to be in another slowing down region; hence, the spectrum is assumed to have a l/E shape. For fusion applications, the user may want to include a “fusion peak” that is tied to the 1/E spectrum in the 20 MeV range.

In addition, AMPX allows nuclide and temperature dependent flux spectra.

2.2.3 Point Cross Sections

The ENDF/B evaluations provide CE representations for many cross sections; however, there are energy regions where the point cross section representation is incomplete and may be zero. For example, the tabulated CE data for ^{235}U are zero from 10^{-5} eV through 2.5×10^4 eV. There is a logical explanation for the data deficiency. In particular, the example data range for ^{235}U spans energy ranges known as the resolved and unresolved resonance regions, and the ENDF/B evaluation does not provide CE cross section data in the resonance region. However, the ENDF/B evaluation provides detailed information about the resonance structure in the resonance region, and a processing code must be able to reconstruct the CE cross section representation from the parameters in the resonance region.

2.2.3.1 Resolved Resonance Region

In the resolved resonance region, the evaluation also specifies a mathematical set of formulae that are to be used in conjunction with the resonance parameters to calculate the cross section values as a function of energy. Six different resolved resonance formalisms are available in the Version 6 formats of the ENDF/B system; however, only five formalisms are currently in use:

1. the single level Breit-Wigner (SLBW) representation,
2. the multilevel Breit-Wigner (MLBW) representation,
3. the Adler-Adler (AA) representation, and
4. the Reich-Moore (RM) representation.
5. the full Reich-Moore (RM) representation.

These formalisms are mentioned in order of increasing complexity and are documented in the ENDF/B procedures manual [**Error! Bookmark not defined.**], and are discussed in detail in the POLIDENT user guide [25].

In the resolved resonance region, neutron resonances are described with parameters that define specific characteristics of each resonance (e.g., the resonance energy, size, spin, angular momentum, etc.). Some evaluations may specify more than a thousand resonances. Unfortunately, the ENDF/B formats do not provide an energy mesh or guidance for defining an energy mesh for resonance reconstruction. Due to the lack of specific guidance, many mesh generation schemes have been employed in a variety of codes. The various mesh generation schemes range from very crude approaches that define equal energy bins around the peaks of a resonance to more efficient and accurate expressions that characterize the resonance shape. The more sophisticated methods define a mesh by maintaining a specific ratio of successive cross section values as a fixed constant (e.g., 0.95). (See the discussion of such a technique in the POLIDENT manual [25]). Still, other approaches use adaptive methods that ensure the energy mesh is refined to a point where cross sections can be interpolated within the mesh panels to a user specified tolerance (e.g., POLIDENT module). Techniques to determine the energy mesh within some specified tolerance are expensive and generally require more CPU time than the actual cross section calculation. The increase in CPU time is to be expected since these methods inherently require the cross sections to be calculated as the mesh is being determined.

When cross section resonances are generated, temperature effects must also be taken into account. Doppler broadening is used to account for changes in the resonance structure due to an increase in temperature at thermal energies. During Doppler broadening, the overall cross section of the nucleus does not change as the medium increases in temperature; rather, the effective cross section as seen by the neutron changes with temperature. In particular, at 0 Kelvin, a neutron at an energy, E, "sees" the resonance like the line shape that is calculated from an analytic expression. The cross section is a function of the relative velocity between the neutron and the nucleus. As the temperature of the medium increases, the nucleus experiences thermal motion that is assumed to vary according to a Maxwellian distribution in temperature. Because the motion of the target nucleus causes the neutron to see a distribution of nucleus velocities, the relative velocity and cross section values are functions of the distribution of nucleus velocities. At 0 Kelvin, the cross section peak is the maximum value, and there is no mechanism that would make the cross section value higher than the peak value. Moreover, a neutron at energy E "sees" a cross section value that is averaged over the distribution of relative velocities. The averaging procedure over the distribution of relative velocities is the basis for Doppler broadening. As the temperature of the medium increases, there is a wider distribution of relative velocity values, and the effective peak cross section value will decrease. At energies away from the resonance peak (i.e., "wings" of the resonance), the effective cross section values increase with the associated temperature increase.

2.2.3.2 Unresolved Resonance Region

The unresolved region is an energy regime where the effects of resonances must be treated; however, the resonances are so closely spaced in energy that it is either impossible or impractical to resolve the individual resonance parameters. As a result, the unresolved resonance parameters are averages of resolved resonance parameters over energy.

Resonances fit into families divided according to the angular momentum (ℓ-value) of the nucleus and the angular momentum of the resonance (j-state). The unresolved resonance data in an ENDF/B evaluation are statistical parameters derived from the resonances in each family that can be resolved. These resolved resonances statistically predict how the resonances are spaced in the particular family, and they also the characteristics of the resonances (e.g., the relative size of the fission or elastic scattering component of the resonance).

All of the unresolved data are presented statistically and can be sampled using Monte Carlo techniques to determine ladders of the resolved resonance parameters. Subsequently, the unresolved data can be used in exactly the same fashion as the resolved resonance parameters determined from the measurement and evaluation procedures. It is difficult to prove that the ladders of resonances adequately represent the statistical behavior described in the unresolved resonance data. To alleviate this problem, the AMPX module PURM generates pairs of resonance or levels surrounding the energy of reference given in the ENDF evaluated data files.

Several codes employ a method developed by R. N. Hwang at Argonne National Laboratory for calculating unresolved cross sections as a function of energy [26]. In the unresolved resonance region, the resonance parameters are provided for the SLBW formalism, and the resonance widths are distributed according to a chi-squared distribution with a specified number of degrees of freedom. Flux weighted cross section values can be calculated over an evaluator-specified energy interval using the unresolved resonance parameters. In the averaging process, the method by Hwang makes use of the narrow resonance (NR) approximation, and the resulting expressions for the average cross section values can be expressed in terms of fluctuation integrals that are also defined in terms of the Doppler broadening ψ and χ resonance line shape functions. Despite the simplifications from the NR approximation, the resulting expressions are quite complicated and involve integration of the resonance widths over the evaluator specified chi-square distributions. The method by Hwang makes use of special Gaussian-like quadratures that permit integration over the probability distributions and the ultimate calculation of averaged point cross sections as a function of energy. These averaged point value curves are very smooth functions in energy, and no attempt is made to actually determine the extreme variation in the cross sections that actually are in the real cross sections. This technique is employed in AMPX modules PRUDE and POLIDENT to processes data in the unresolved energy range and form the smooth average cross section functions for selected temperatures and values of a background cross section. The background cross section values are used in the NR approximation to account for other nuclides being mixed in with the nuclide at different concentrations.

2.2.4 Scattering Kinematics

The development of a MG library requires CE cross section data coupled with an appropriate energy group structure and weighting spectrum. In the preceding sections, brief discussions are provided for each of these components; however, a MG library cannot be complete without group-to-group scattering matrices. As expected, the point data, energy-group structure, and weighting spectrum must be used to calculate the terms in a scattering matrix; however, additional information must be provided to describe the physics of a particle collision with a target nucleus. The additional information can be referred to as *scattering kinematics*.

Aside from a specification of the differential cross section as a function of energy and angle, kinematics information is not provided in an ENDF/B evaluation for two body processes (e.g., elastic or discrete level inelastic scattering). All of the parameters necessary to use in the kinematics equations (e.g., the mass ratio and Q values of the inelastic levels) are given in the files, but the evaluation assumes that the cross section processing code accounts for the conservation of energy and momentum in the calculation of transfer matrices.

The kinematics for systems of three or more bodies also must conserve energy and momentum; however, for these cases, there is no general solution since the equations are intractable. Moreover, the kinematics must be presented in some abstract manner in an ENDF/B evaluation (i.e., either as a tabular or analytic fit to some experimental behavior or to some simplified model for the nuclear process). The matrices for multibody interactions can be given in three different ways in the ENDF/B evaluations:

1. The process is assumed to scatter isotropically in the laboratory system. Note that the laboratory system must be used because the transformation from the center-of-mass to the laboratory system would require a solution of the kinematics expressions. Also, the secondary energy distribution of particles is given as a tabular or an analytic distribution.

2. The angular variation of the differential cross section is expressed in a tabular or analytic format that is assumed to be independent of the secondary energy distribution expressed in item one.

3. The angular and energy variation of the scattering distributions are dependent and are presented in a tabular or analytic manner or a combination of both.

AMPX uses procedures developed explicitly for the current release of the system. Furthermore, these new procedures are based on a preference to have a single collection of subroutines with the flexibility and generality to permit the calculation of transfer matrices for any kind of reaction (e.g., two-body or multibody processes for any particle type such as neutrons, gamma-rays, etc.). These procedures are described in the sections of this document that describe the Y12 module.

2.3 CE LIBRARIES

Many codes can now use CE data directly without the need to collapse to a MG first. While the self-shielding sequences in SCALE involving the CENTRM module [37] use point-wise cross section data in the resolved resonance range, CE libraries are also used for criticality methods [12]. The CE libraries contain the same elements as the MG libraries, and except in the case of kinematics data, the point-wise data are simply stored without the need of a flux. The kinematics data are converted to probability distributions. For each incident energy, a cumulative probability distribution with respect to all possible exit angles is calculated based on the information given in ENDF. For each exit angle, a conditional cumulative probability is also given with respect to the exit energy.

2.4 THE MODULAR CONCEPT

The introduction of this manual describes AMPX as a modular programming system. The modular concept isolates a single computing function into a single independent code and provides a communication channel between codes using a standard interface or file. In the limit to this approach, a code would require little or no user interaction. Moreover, the code would perform a single function with an input and an output file specification. There are obvious penalties associated with the limit, because it would require the selection of a formidable number of modules to complete some tasks.

The popular UNIX operating systems prevalent on many modern computing platforms use a similar modular structure. UNIX commands are equivalent to codes with an input and output specification. These specifications can originate from a previous command, or they can be passed to a subsequent command. UNIX commands are generally associated with a single function, but various commands have many options and methods for redirecting output. In addition, the UNIX command execution sequences can be written into scripts that can be re-executed by simply invoking the script.

AMPX uses a similar modular concept. Each module can have options that are specified as part of the input, and ideally, these input options are minimized. A module may require one or more standard input files (e.g., MG or point cross section libraries) and may produce one or more standard output files (e.g., MG or point cross section libraries). The input to AMPX is equivalent to a UNIX script. The modules (commands) are selected, and the options are given in the same stream as input data to the code. Each code can use default locations for required input/output files, or a user can override the default values as part of the input data. Files to be retained can be named and stored using a special module, referred to as “SHELL,” that allows the user to insert UNIX commands into the AMPX execution sequence. As with UNIX scripts, the AMPX code selection and input data information can be retained for subsequent re-execution or modification to perform other similar tasks.

2.4.1 Execution Sequences

This section describes the procedure for constructing an AMPX execution sequence in a schematic fashion. Also, a description of the input data specifications is provided for the system’s driver module.

The AMPX driver module is a variation of the driver module used in SCALE, providing a convenient method for selecting codes collected together in a standard program library, introducing new codes, or importing other codes from standard program libraries. A code is selected by the simple command:

```
=CODENAME
```

For example, if the Y12 module is desired for execution, the command “=Y12” would be entered, followed by the input data for the Y12 module. The input data can be written in any format that the code uses. In previous versions of AMPX, all modules required an input scheme known as “FIDO” (described in section 10). Many of the codes in AMPX still use the FIDO input structure, but newer codes use the keyword free-format driven input, the fixed input scheme, or whatever the code developer prefers. An execution sequence follows the pattern:

```
      =module_name

      input options for the module

      end

      =next_module_name

      input options for the next module

      end

      =next_module_name

      •

      •

      etc.
```

Note that each module is specified with an = sign followed by the name of the module, with no space between the first character of the module name and the equal sign. The input options for the module are specified on the lines following the name of the module. The termination of a module input is specified with an end card starting in column 1. The remaining modules in the sequence are specified in a similar manner as the first module.

A sample AMPX execution script would have the following form:

```
      =shell

      ln -fs /home/centrm_libraries/endfb6/broaden/u235 ft33f001

      end
```

```
=pickeze
0$$ 33 34
1$$ 1 1 1 1 0 0 e t
2$$ 9228
3$$ 3
4$$ 1
5** 0.
t
end
=charmin
single to plot in=34 out=35
end
=shell
cp ft35f001 /home/u235.tot
end
```

In the above example, the PICKEZE and CHARMIN modules are specified in the input file. Note that the SHELL specification is used to permit the execution of UNIX commands during the AMPX execution sequence. In the first SHELL command, the ***u235*** data file is linked to Logical Unit 33 (i.e., ***ft33f001***) in the AMPX working directory prior to executing any AMPX module. Subsequently, the PICKEZE and CHARMIN modules are used to process the data. Based on the CHARMIN input specifications, the output is written to Logical Unit 35. When AMPX is executed on a particular machine, the code system is executed in a temporary working directory on the computing platform. If the user wants to keep a data file produced by a specific module, the SHELL command should be used to copy the file to a desired location as determined by the user. In the above example, the data file produced by CHARMIN (i.e., ***ft35f001)*** is copied to the ***/home*** directory and renamed ***u235.tot***.

The installation procedure will have put a command file *ampxrte* into the installation directory, which takes the above input as a command line argument.

Several points are noted below:

- Since AMPX executes in a temporary directory, all external files accessed during the run must be given using the absolute path. The script sets an environment variable, RTNDIR, which points to the current working directory. Thus files in that directory can also be referenced as ${RTNDIR}/localFile, where localFile is the name of the desired file.

- AMPX uses logical unit numbers to identify files. Logical units are bound to files ftnnf001, where nn denotes the logical unit number to use. The number format is always two digits long.

- Libraries to be used with SCALE must be in big-endian format. By default, AMPX produces data files in native format. However, if the environment variable SCALEXS is set to *yes*, then the following logical unit numbers are written or read in big-endian format: 60-70, 80-89.

3. PROCESSING ENDF DATA

In order to produce MG and CE libraries, various AMPX codes read evaluated data from ENDF formatted files [1]. A collection of ENDF/B formatted evaluation are called *tapes*, and they contain one or more evaluations. Evaluations are made up of files, which can be thought of as sections with specific types of information. The following table gives a brief overview of the file information processed by AMPX at this writing:

File Number	Description
File 1	File 1 contains two parts: • general information about the evaluation in text format (not used for processing), and • number of neutrons per fission, delayed neutron data, and number of prompt neutrons if the nuclide is fissionable. The data are processed by module POLIDENT.
File 2	File 2 contains resolved and unresolved resonance parameters. The information is processed by modules POLIDENT, PRUDE, and PURM.
File 3	File 3 contains point-wise cross section data and Q values for each reaction. The information is processed by modules POLIDENT, PRUDE, PURM and Y12.
File 4	File 4 contains angular distributions of secondary particles for incident neutrons. The information is processed by module Y12.
File 5	File 5 contains energy distributions of secondary particles for incident neutrons. The information is processed by module Y12.
File 6	File 6 contains angular and energy distributions of secondary particles. It is more general than the information that can be given in Files 4 and 5. The information is processed by module Y12.
File 7	File 7 contains thermal neutron scattering data. The information is processed by modules Y12.
File 8	File 8 contains radioactive decay and fission product data.
File 9	File 9 contains multiplicities for production of radioactive nuclides. The information is processed by module POLIDENT, which saves the data for further processing by LIPTON.
File 10	File 10 contains cross sections for production of radioactive nuclides. The information is processed by module POLIDENT, which saves the data for further processing by LIPTON.
File 12	File 12 contains the photon production multiplicities and transition probabilities. The data are processed by module Y12.
File 13	File 13 is similar to File 12, but gives absolute photon production cross sections. The data are processed by module Y12.
File 14	File 14 contains photon angular distributions. The information needs to be combined with Files 13 and 15 data to generate photon yield scattering matrices. The data are processed by module Y12.

File 15	File 15 contains energy distributions for emitted photons. The information must be combined with Files 12 and 14 data to generate photon yield scattering matrices. The data are processed by module Y12.
File 23	File 23 gives the smooth photon interaction cross sections for incident gammas. The information is processed by module Y12.
File 27	File 27 gives the form factors and scattering functions for photons.
File 31	File 31 gives covariance information for the average number of neutrons per fission. The information is processed by module PUFF-IV.
File 32	File 32 gives covariance information for the resonance parameters. The information is processed by module PUFF-IV.
File 33	File 33 gives covariance information for the neutron cross sections. The information is processed by module PUFF-IV.
Fle 35	File 35 gives covariance information for exit energy distributions. The information is processed by module CHICOV.

All ENDF evaluations contain a File 1, which provides textual information about the evaluation.

With respect to AMPX processing, the evaluations can be categorized into four groups:

- A standard incident neutron evaluation contains Files 1, 2 and 3 and possibly Files 4, 5, 6, 12, 13, 14, 15, 31, 32, and 33.
- A standard incident gamma evaluation contains Files 1, 23, and 27.
- A thermal moderator evaluation like ^{1}H in H_2O contains Files 1 and 7 only. Thermal moderator evaluations are the only evaluations containing File 7 data.
- A decay file contains File 8 and is used to generate decay and gamma emission libraries for use in depletion calculations.
- Incident neutron evaluations containing Files 9 and/or 10 and possibly Files 2 and/or 3 are used to generate libraries containing branching information for use in depletion calculations.

At this writing, these are the only files processed by AMPX.

Several kinds of libraries are created by AMPX:

- **MG or CE libraries for incident neutrons.** These libraries are used for neutron transport calculations, and they use information from Files 1, 2, 3, 4, 5, 6, 7, 12, 13,14, and 15. Note that information from files 12–15 is only used in coupled neutron-gamma calculations.
- **MG or CE libraries for incident gammas.** These libraries are also used in transport calculations, and they use information from Files 23 and 27.
- **Covariance libraries for MG data.** The libraries use information from Files 1, 2, 3, 31, 32, 33 and 35.
- **Decay and gamma libraries for use in depletion calculations.** These libraries use information from Files 2, 3, 8, 9, and 10.

As stated above, thermal moderator evaluations only contain File 7 information, which gives cross section data and scattering information in the thermal range. In a CE or MG library, the thermal information is combined with data from a different evaluation outside the thermal range. For example, the thermal moderator information is given for ^{1}H in H_2O in ENDF/B-VII.0. In order to make a full MG or CE library, the thermal moderator information is combined with cross section and scattering information from ^{1}H, which is referred to as the fast evaluation coupled to the thermal moderator H_2O. For a standard evaluation, a free gas treatment is used to calculate the thermal scattering matrices.

Some modules are required to create the libraries supported by the AMPX system. The detailed steps needed to create the various types of libraries are outlined in the next sections. While it is relatively easy to set up the required sequence of modules for one evaluation, a typical library has hundreds of nuclides. A graphical user interface (GUI) called ExSite has been provided to help set up inputs to process all desired evaluations. A detailed description of the GUI is given in Section 4. The following section contains an overview of the steps required. A more detailed description of the functionality of some of the key modules is provided in Sect. 6.

3.1 CREATING AN MG LIBRARY

The general flow of a sequence to generate a neutron MG library is show in Fig. 1, and the flow is described in detail below. Inputs for all desired evaluations are typically created automatically using the AMPX GUI.

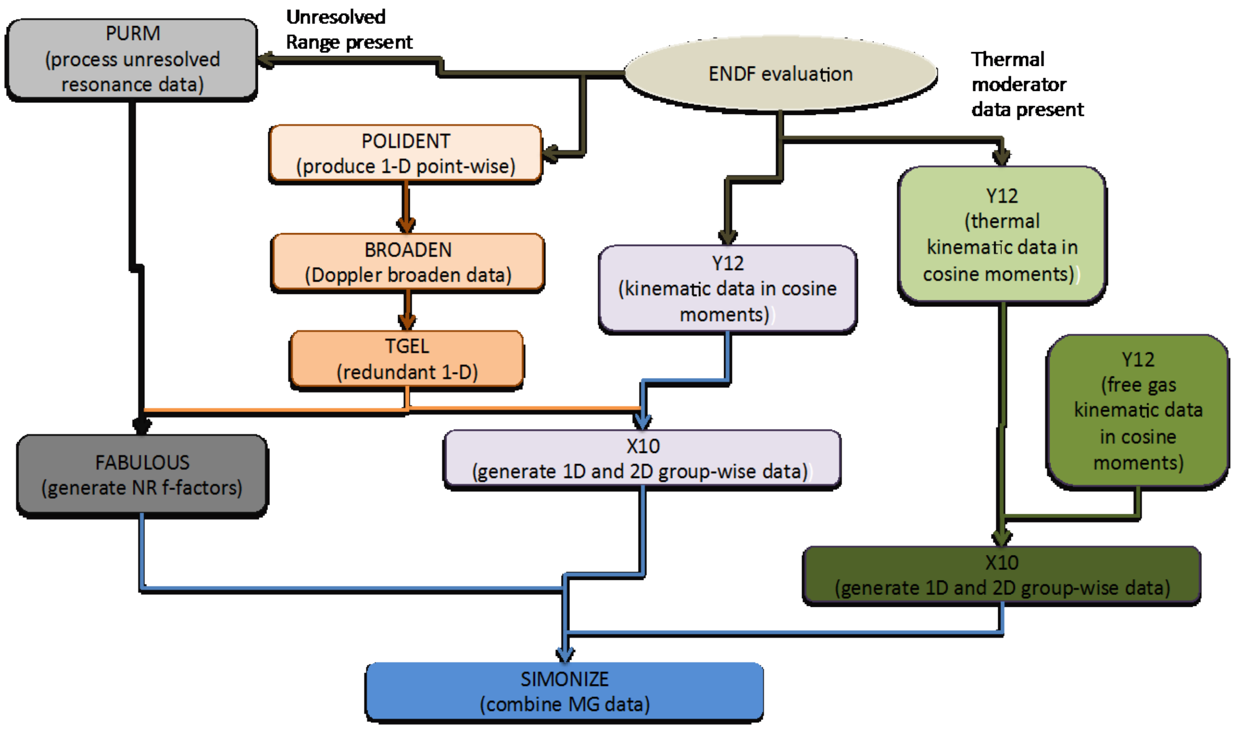

Unresolved Range present
PURM (process unresolved resonance data)
ENDF evaluation
Thermal moderator data present
POLIDENT (produce 1-D point-wise)
BROADEN (Doppler broaden data)
TGEL (redundant 1-D)
Y12 (kinematic data in cosine moments)
Y12 (thermal kinematic data in cosine moments)
Y12 (free gas kinematic data in cosine moments)
FABULOUS (generate NR f-factors)
X10 (generate 1D and 2D group-wise data)
X10 (generate 1D and 2D group-wise data)
SIMONIZE (combine MG data)

Fig. 1. AMPX sequence for producing neutron MG library for one isotope.

The procedure is as follows:

1. The module POLIDENT is used to create the point-wise cross section at 0 K for all neutron evaluations, excluding thermal moderators. The module POLIDENT reconstructs the point-wise data in the resolved and unresolved range if File 2 provides resonance parameters. These data are

combined with the point-wise data given in File 3. The module TGEL is used to reconstruct the total cross section from the partial reactions.

2. The module BROADEN is used to Doppler broaden the point-wise cross section data created in steps 1 and 2. The temperatures at which broadening is required include the temperature at which infinite dilute cross section data are to be saved in the library, as well as all temperatures for which Bondarenko factors are to be generated.

3. The module TGEL is used to reconstruct the total cross section from the partial reactions after broadening.

4. The module Y12 is used to generate the kinematics data for neutron scattering. If a coupled library is desired, Y12 is also used to generate the gamma production scattering data.

5. A suitable weighting function for neutron data is generated using the module JERGENS. The user can also supply a temperature and ZA dependent weighting function.

6. The module PICKEZE is run to select the 1-D cross section at the desired temperatures from the data generated in step 3.

7. The module X10 is run in neutron mode to generate the neutron 1-D cross section data and the neutron scattering matrices. The module uses the point data generated in step 6, the kinematics data from step 4, and the weighting function from step 5.

8. If the library contains thermal moderator data, the module Y12 is used to generate point-wise cross section and kinematics data for the thermal moderators from File 7.

9. The module X10 is run in neutron mode to generate the thermal moderator data. The module uses the point data and kinematics data from step 8 and weighting function from step 5.

10. If thermal moderator data do not exist, the module Y12 is used to generate thermal scattering matrices for free gas scattering. The weighting function generated in step 5 is used for the MG collapse performed in neutron mode in X10.

11. If the MG library should contain gamma production data, module X10 is run in yield mode using the point data generated in step 6, the kinematics data from step 4, and the weighting function generated in step 5.

12. If a coupled library is desired, the module Y12 is used to generate to generate point-wise cross section and kinematics data for all gamma evaluations to be included in the library.

13. A suitable weighting function for gamma data is generated using the module JERGENS.

14. The module X10 is run in gamma mode to generate a gamma MG library. The module uses the point data and kinematics data from step 11 and the weighting function from step 13.

15. If a neutron evaluation contains unresolved resonance data, module PRUDE is used to generate the cross section data in the unresolved resonance range for the desired temperatures and background values. If the probability tables are to be used for the creation of f-factors, module PURM and PURM_UP are used to generate probability tables.

16. The module FABULOUS is used to generate the Bondarenko factors. It uses the point-data generated in steps 3 and 15 (if applicable) and the MG library generated in step 7. The neutron weighting function generated in step 5 is used to collapse point-wise data. The MG library created in step 9 is used to ensure that the infinite-dilute cross sections used to generate the Bondarenko factors are consistent with the 1-D cross section in the library. If the use of probability tables in the unresolved range is desired, FABULOUS_URR is used instead.

17. The module SIMONIZE is used to combine all the parts into one MG library. SIMONIZE recalculates all redundant cross section data and renormalizes scattering matrices as needed.

18. The module RADE is used to ensure that the library is correct.

19. The module AJAX is used to bind the MG libraries for each evaluation produced in step 17 into the final MG library.

3.2 CREATING A CE LIBRARY

As with MG library generation, inputs for all desired evaluations are typically created automatically using the AMPX GUI. The procedure to produce neutron and gamma CE libraries is outlined below.

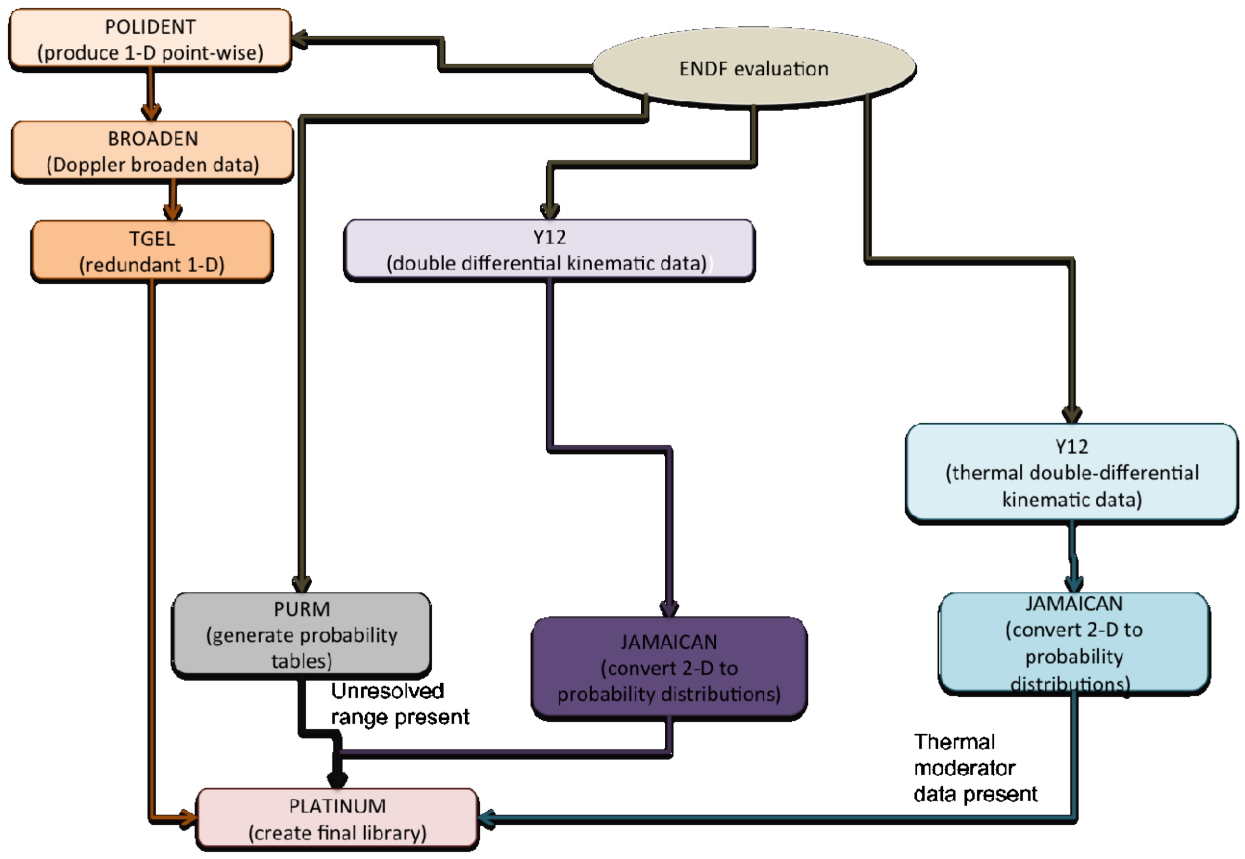

POLIDENT
(produce 1-D point-wise)
ENDF evaluation
BROADEN
(Doppler broaden data)
TGEL
(redundant 1-D)
Y12
(double differential kinematic data)
Y12
(thermal double-differential kinematic data)
PURM
(generate probability tables)
JAMAICAN
(convert 2-D to probability distributions)
JAMAICAN
(convert 2-D to probability distributions)
Unresolved range present
Thermal moderator data present
PLATINUM
(create final library)

Fig. 2. AMPX sequence for producing neutron CE libraries for one isotope.

For a standard neutron evaluation, the process to generate a CE library is as follows:

1. The module POLIDENT is used to create the point-wise cross sections at 0 K for all neutron evaluations, excluding thermal moderators. Module POLIDENT reconstructs the point-wise data in the resolved and unresolved range if File 2 provides resonance parameters. These data are combined with the point-wise data given in File 3. The module TGEL is used to reconstruct the total cross section from the partial reactions.

2. The module BROADEN is used to Doppler broaden the point-wise cross section data created in step 2.

3. The module TGEL is used to reconstruct the total cross section from the partial reactions.

4. The module PICKEZE is run to eliminate the 0 K data for reactions that are Doppler broadened.

5. The module TGEL is run again to re-construct total, capture, inelastic, and absorption cross sections.

6. The module Y12 is used to generate the kinematics data for neutron scattering and gamma-production scattering. The data are produced in a tabulated double-differential form.

7. The module JAMAICAN is run to convert the double differential point-wise distribution into a marginal probability distribution in angle and conditional probability distribution in exit energy.

8. If the evaluation contains unresolved resonance data, the modules PURM and PURM_UP are run for each desired temperature to generate probability tables in the unresolved resonance range.

9. The module PLATINUM is run to create the final CE library. The data produced in steps 5–8 are combined, and the 1-D collision probabilities are calculated.

When processing a thermal moderator, the thermal moderator data given in File 7 must be combined with a suitable evaluation in the higher energy range (fast data). In this case, the procedure is as follows:

1. The module POLIDENT is used to create the point-wise cross section at 0 K for the fast data (for example, ^1H if generating a library for H_2O)

2. The module TGEL is used to reconstruct the total cross section from the partial reactions.

3. The module BROADEN is used to Doppler broaden the point-wise cross section data created in step 2 to the same temperatures used in the thermal evaluation.

4. Steps 4–8 from the procedure for a standard evaluation are run to produce the kinematics data for nonthermal reactions (for example, ^1H if generating a library for H_2O).

5. The module Y12 is run to process the thermal data from File 7. The module JAMAICAN is run to produce a marginal probability distribution in angle and conditional probability distribution in exit energy. In addition, a 1-D cross section is produced for thermal reactions 1007 and 1008.

6. The module TGEL is used to generate the sum of the 1-D thermal cross section data. The result is stored in MT=2, the elastic cross section.

7. The module SPLICER is used to override MT=2 in the data produced in step 3 in the thermal range with the data produced in step 6.

8. The module TGEL is run once more to update the total cross section after the update in the elastic cross section.

9. The module ZEST is run to combine the 1-D thermal data from step 5 with the data produced in step 8 into one file.

10. The module PLATINUM is run to create the final CE library. The data produced in steps 4, 5, and 9 are combined and the 1-D collision probabilities are calculated.

In the case of incident gammas, the ENDF evaluation contains Files 23 and 27. The procedure to process a gamma CE-library is as follows:

1. The module Y12 is run to generate 1-D point-wise gamma data and kinematics data.

2. The module JAMAICAN is run to generate marginal probability distribution in angle and conditional probability distribution in exit energy.

3. The module PLATINUM is run to create the final CE library. The data produced in steps 1 and 2 are combined into one library.

4. USING EXSITE

To generate an MG or CE library, a number of modules have to be run in sequence. Some information from the ENDF evaluation is needed before preparing the input files. For example, some modules are only run if the evaluation contains unresolved resonance data. For thermal evaluation one or more evaluations need to be designated as fast evaluations. While this can be done by the user on a per evaluation basis, it is easier to generate all the information automatically. The program ExSite helps the user in collecting needed information from the ENDF evaluations and generates the input files. In addition ExSite allows the user the flexibility to edit the input for individual modules.

In generating input files ExSite uses two XML-formatted files that can be automatically generated from user selected ENDF tapes. The first file contains information that is automatically extracted from the ENDF tapes. The second file is a configuration file containing information about which fast evaluation(s) are to be combined with a given thermal evaluation and nuclides that need a special ID if the library is to be used with SCALE. The second file is also automatically produced, but the user must review the information before actually using the file. Please note that the ID values for metastable nuclides and evaluations containing thermal moderators have changed between SCALE 6.1 and SCALE 6.2. ExSite tries to automatically generate SCALE 6.2 ID values. A detailed description of the content of the two files can be found in Sect. 8. This section focuses on the use of ExSite.

4.1 CREATE AN XML LISTING

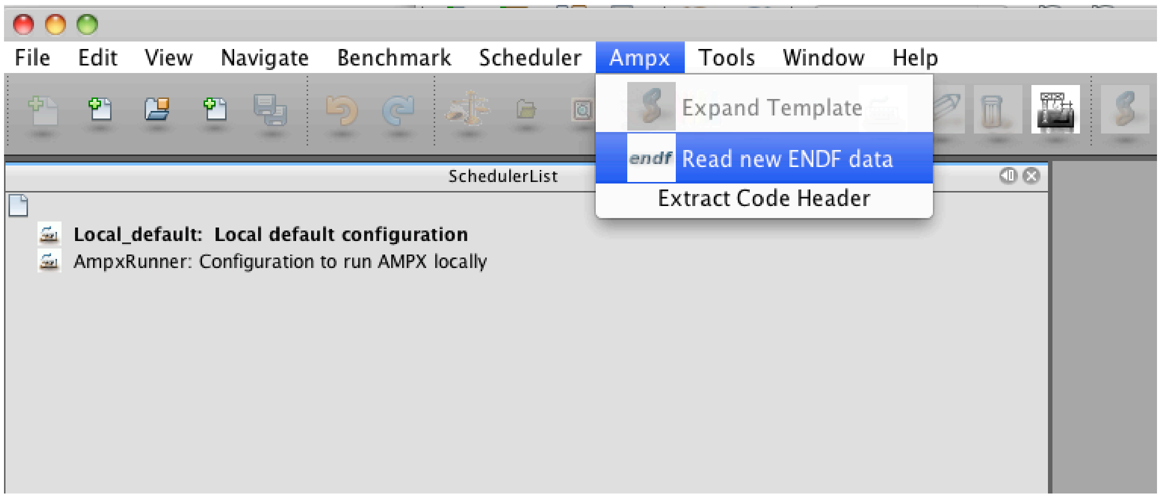
The user must first select “Read new ENDF data” from the “Ampx” menu. See Fig. 3 for details. This will open a wizard to guide the user through the selection of the ENDF tapes. The wizard has three steps:

1. The user selects the ENDF tapes to be parsed. The window is shown in Fig. 4. More files can be added by pressing the “Add” button, which opens a file dialog as shown in Fig. 4. The list of added files can be reviewed, and any unwanted files can be removed. The “Remove” button becomes active when one or more file names are selected.

2. In the next step, the selected files will be parsed, as shown in Fig. 5. The “Next” button becomes available when all ENDF tapes are parsed. If an error occurs, it will be printed on the screen, and the “Next” button will not be enabled.

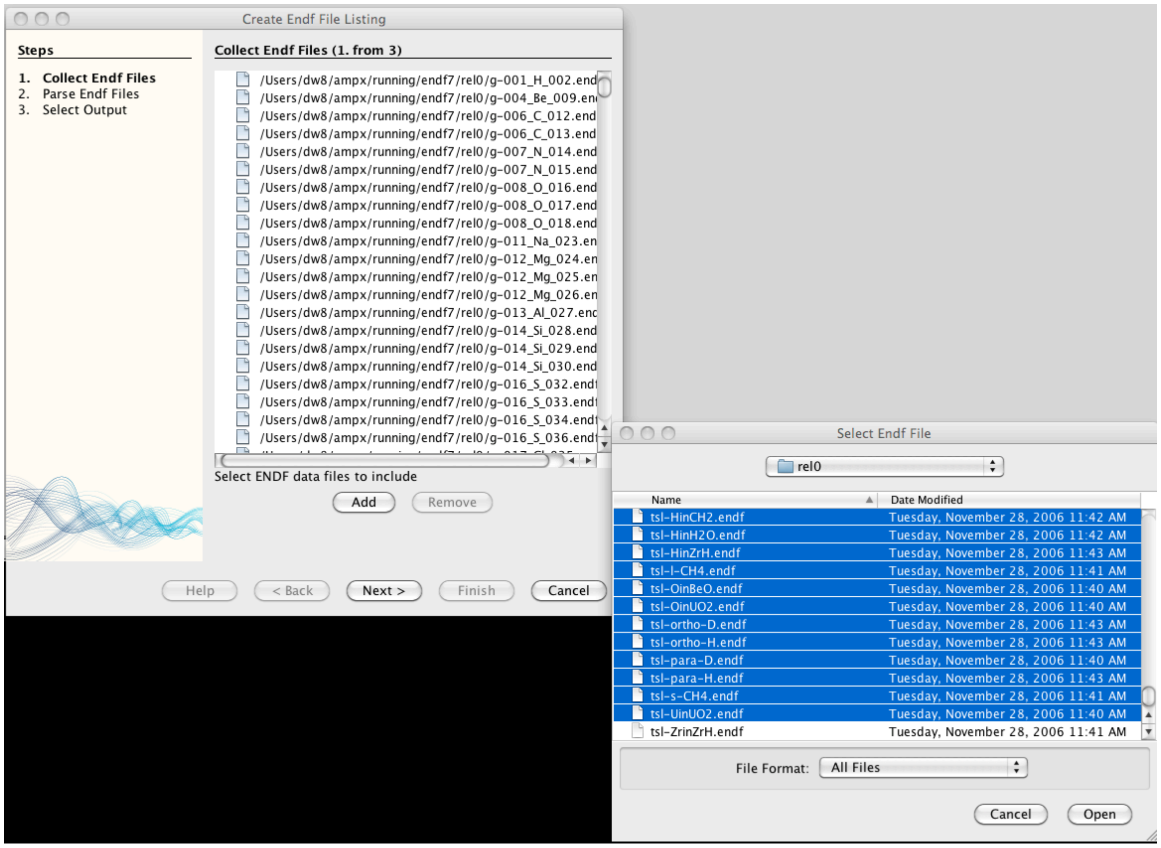
3. The final step is to select the output file. The wizard window is shown in Fig. 6. The “Select Output” button will open a file dialog, allowing the user to select the file name for the ENDF listing. If any of the ENDF tapes selected contains metastable nuclides or a thermal evaluation, the text shown in Fig. 6 will be displayed. In this case, the user should review the configuration file to make sure the correct ID values are chosen.

The ENDF listing generated by the wizard can now be used for expanding templates.



Local_default: Local default configuration
AmpxRunner: Configuration to run AMPX locally

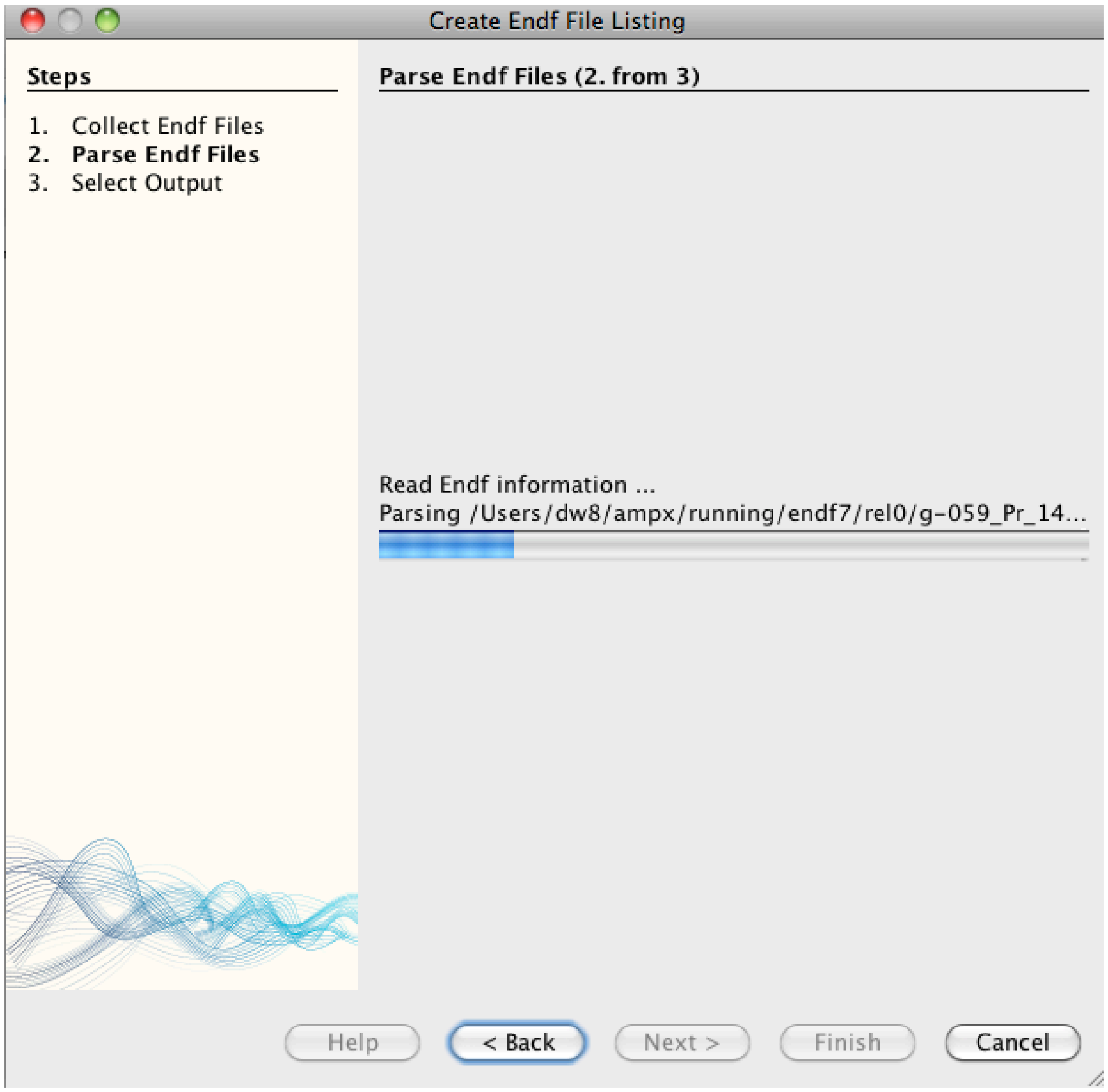
Fig. 3. Start reading abbreviated ENDF tapes.



```
/Users/dw8/ampx/running/endf7/rel0/g-001_H_002.end
/Users/dw8/ampx/running/endf7/rel0/g-004_Be_009.en
/Users/dw8/ampx/running/endf7/rel0/g-006_C_012.end
/Users/dw8/ampx/running/endf7/rel0/g-006_C_013.end
/Users/dw8/ampx/running/endf7/rel0/g-007_N_014.end
/Users/dw8/ampx/running/endf7/rel0/g-007_N_015.end
/Users/dw8/ampx/running/endf7/rel0/g-008_O_016.end
/Users/dw8/ampx/running/endf7/rel0/g-008_O_017.end
/Users/dw8/ampx/running/endf7/rel0/g-008_O_018.end
/Users/dw8/ampx/running/endf7/rel0/g-011_Na_023.en
/Users/dw8/ampx/running/endf7/rel0/g-012_Mg_024.en
/Users/dw8/ampx/running/endf7/rel0/g-012_Mg_025.en
/Users/dw8/ampx/running/endf7/rel0/g-012_Mg_026.en
/Users/dw8/ampx/running/endf7/rel0/g-013_Al_027.enc
/Users/dw8/ampx/running/endf7/rel0/g-014_Si_028.end
/Users/dw8/ampx/running/endf7/rel0/g-014_Si_029.end
/Users/dw8/ampx/running/endf7/rel0/g-014_Si_030.end
/Users/dw8/ampx/running/endf7/rel0/g-016_S_032.endf
/Users/dw8/ampx/running/endf7/rel0/g-016_S_033.endf
/Users/dw8/ampx/running/endf7/rel0/g-016_S_034.endf
/Users/dw8/ampx/running/endf7/rel0/g-016_S_036.endf
```

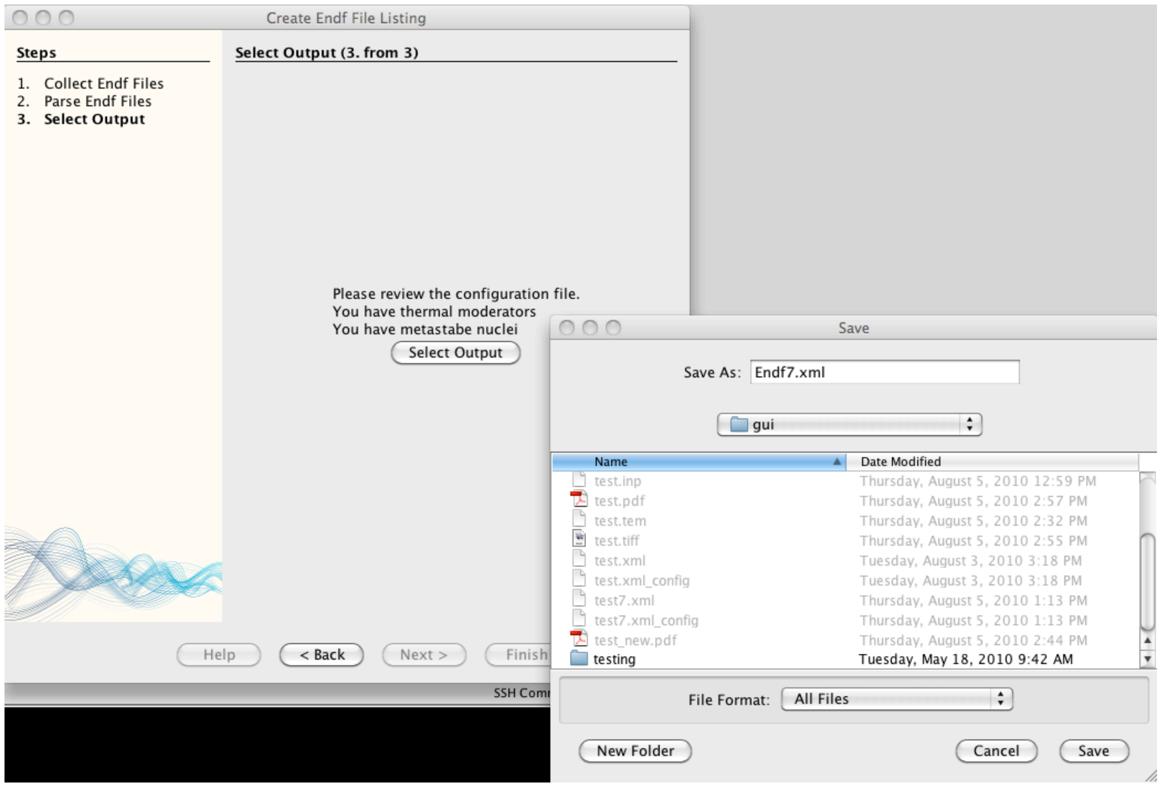
Name	Date Modified
tsl-HinCH2.endf	Tuesday, November 28, 2006 11:42 AM
tsl-HinH2O.endf	Tuesday, November 28, 2006 11:42 AM
tsl-HinZrH.endf	Tuesday, November 28, 2006 11:43 AM
tsl-l-CH4.endf	Tuesday, November 28, 2006 11:41 AM
tsl-OinBeO.endf	Tuesday, November 28, 2006 11:40 AM
tsl-OinUO2.endf	Tuesday, November 28, 2006 11:40 AM
tsl-ortho-D.endf	Tuesday, November 28, 2006 11:43 AM
tsl-ortho-H.endf	Tuesday, November 28, 2006 11:43 AM
tsl-para-D.endf	Tuesday, November 28, 2006 11:40 AM
tsl-para-H.endf	Tuesday, November 28, 2006 11:43 AM
tsl-s-CH4.endf	Tuesday, November 28, 2006 11:41 AM
tsl-UinUO2.endf	Tuesday, November 28, 2006 11:40 AM
tsl-ZrinZrH.endf	Tuesday, November 28, 2006 11:41 AM

Fig. 4. Select the ENDF tapes from which to create the listing.



Read Endf information ...
Parsing /Users/dw8/ampx/running/endf7/rel0/g-059_Pr_14...

Fig. 5. Progress bar to indicate that ENDF tapes are being parsed.



Please review the configuration file.
You have thermal moderators
You have metastabe nuclei

Name	Date Modified
test.inp	Thursday, August 5, 2010 12:59 PM
test.pdf	Thursday, August 5, 2010 2:57 PM
test.tem	Thursday, August 5, 2010 2:32 PM
test.tiff	Thursday, August 5, 2010 2:55 PM
test.xml	Tuesday, August 3, 2010 3:18 PM
test.xml_config	Tuesday, August 3, 2010 3:18 PM
test7.xml	Thursday, August 5, 2010 1:13 PM
test7.xml_config	Thursday, August 5, 2010 1:13 PM
test_new.pdf	Thursday, August 5, 2010 2:44 PM
testing	Tuesday, May 18, 2010 9:42 AM

Fig. 6. Select the output for the ENDF listing.

4.2 USING TEMPLATES

ExSite is used to generate input files by combining templates and ENDF listings. Templates to generate the major types of libraries are included with ExSite. In addition, users can add custom templates. In order to use templates, select “New Template File” from the “File” menu, or open an existing template file by selecting “Open File” from the “File” menu. Template files have an extension of “tem.” After opening the file, a palette appears next to the editor for the input file. Click on one of the templates and drag it into the editor as shown in Figure 7. Once the mouse is released a dialog appears that allows the user to enter data relevant to the selected template. The dialog is shown in Fig. 8. The xml listing generated in the last section needs to be given under the “evals” keyword, and the name of the input files generated is listed under the “input” keyword. Note that the tag name for each evaluation will generally be added to the name of the input. All file names used in the template are interpreted as relative to the template file except if absolute file names are used. The generated input files will use relative file names where possible, except if “absolute” is selected. After pressing okay, the keyword-based input will appear in the editor window. The editor window allows the user to alter any of the wizard-generated input via keyboard input. Selecting “Expand Template” from the “Ampx” menu or pressing the appropriate button on the toolbar will generate the input files. See Fig. 9 for details.

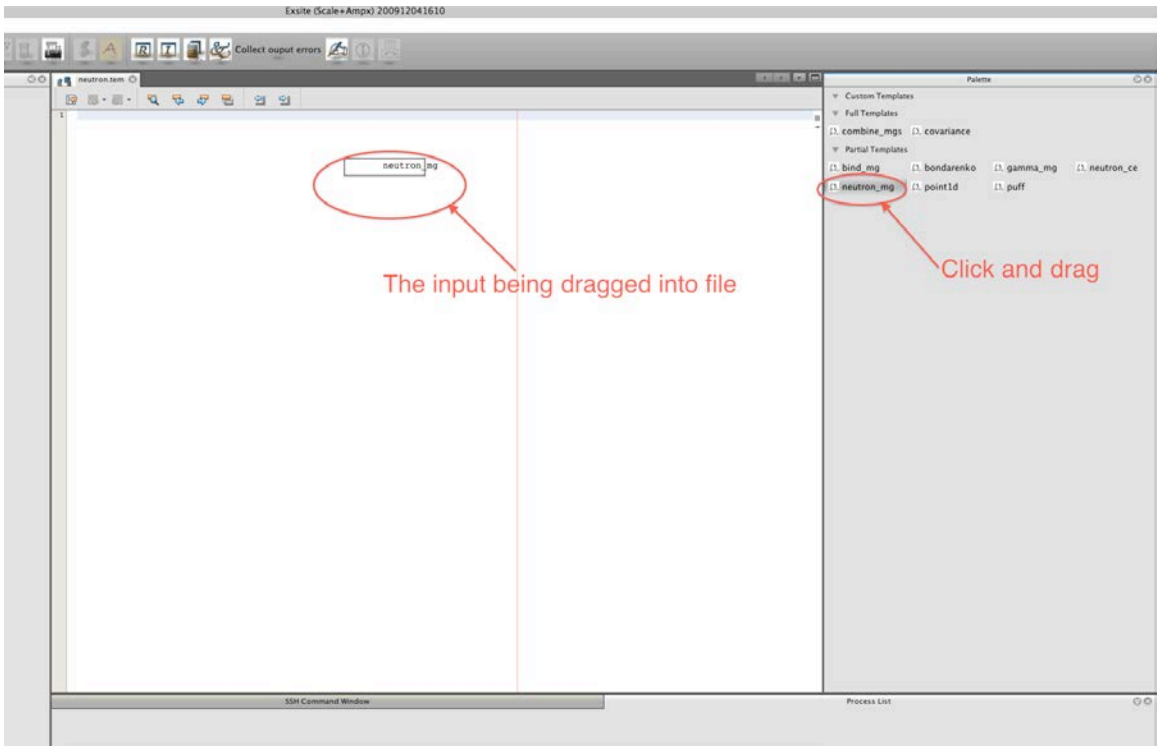
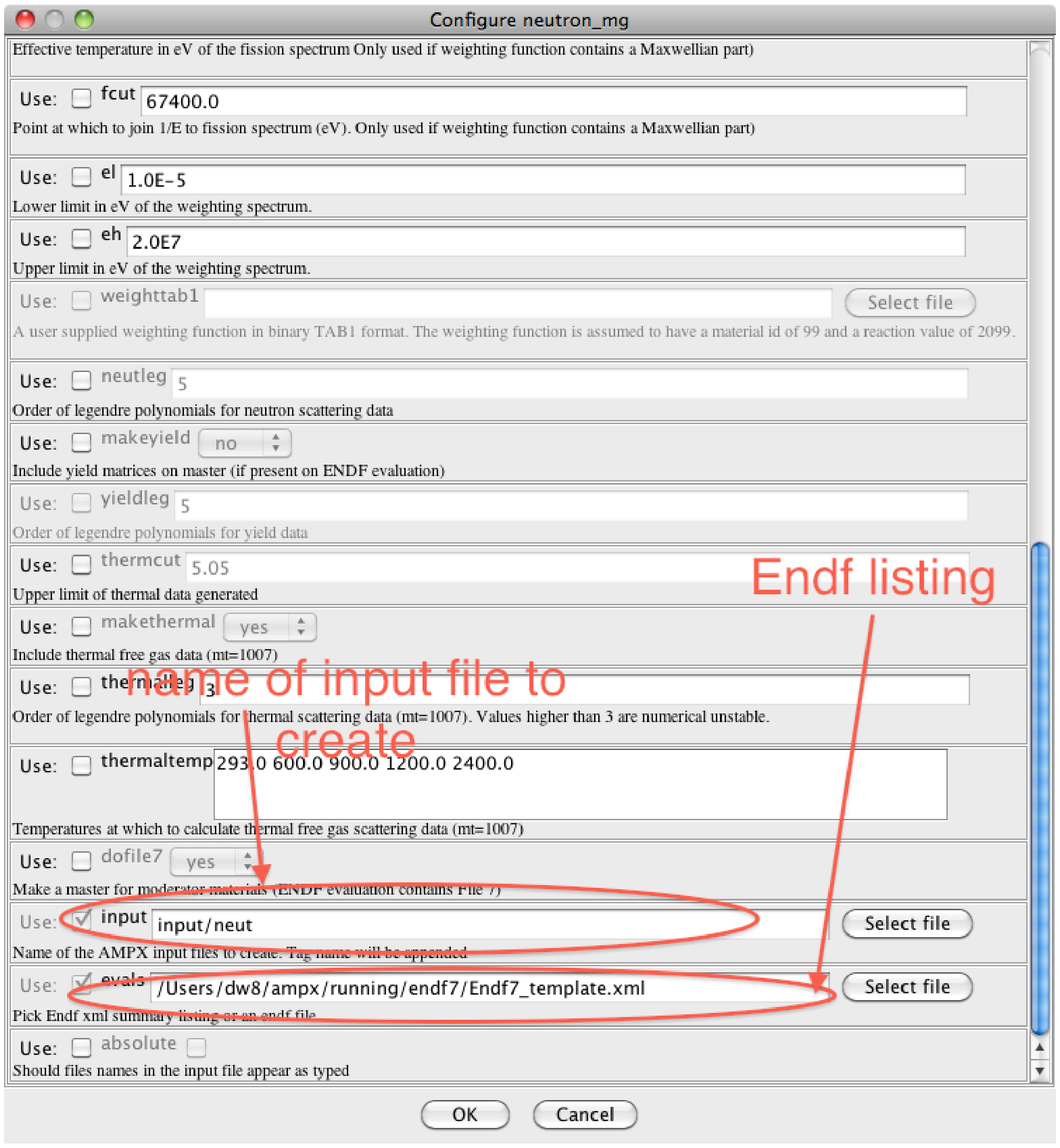


Fig. 7. Drag a template into an template file.



Effective temperature in eV of the fission spectrum Only used if weighting function contains a Maxwellian part)

Use: ☐ fcut 67400.0
Point at which to join 1/E to fission spectrum (eV). Only used if weighting function contains a Maxwellian part)

Use: ☐ el 1.0E-5
Lower limit in eV of the weighting spectrum.

Use: ☐ eh 2.0E7
Upper limit in eV of the weighting spectrum.

Use: ☐ weighttab1
A user supplied weighting function in binary TAB1 format. The weighting function is assumed to have a material id of 99 and a reaction value of 2099.

Use: ☐ neutleg 5
Order of legendre polynomials for neutron scattering data

Use: ☐ makeyield no
Include yield matrices on master (if present on ENDF evaluation)

Use: ☐ yieldleg 5
Order of legendre polynomials for yield data

Use: ☐ thermcut 5.05
Upper limit of thermal data generated

Use: ☐ makethermal yes
Include thermal free gas data (mt=1007)

Use: ☐ thermalleg 3
Order of legendre polynomials for thermal scattering data (mt=1007). Values higher than 3 are numerical unstable.

Use: ☐ thermaltemp 293.0 600.0 900.0 1200.0 2400.0
Temperatures at which to calculate thermal free gas scattering data (mt=1007)

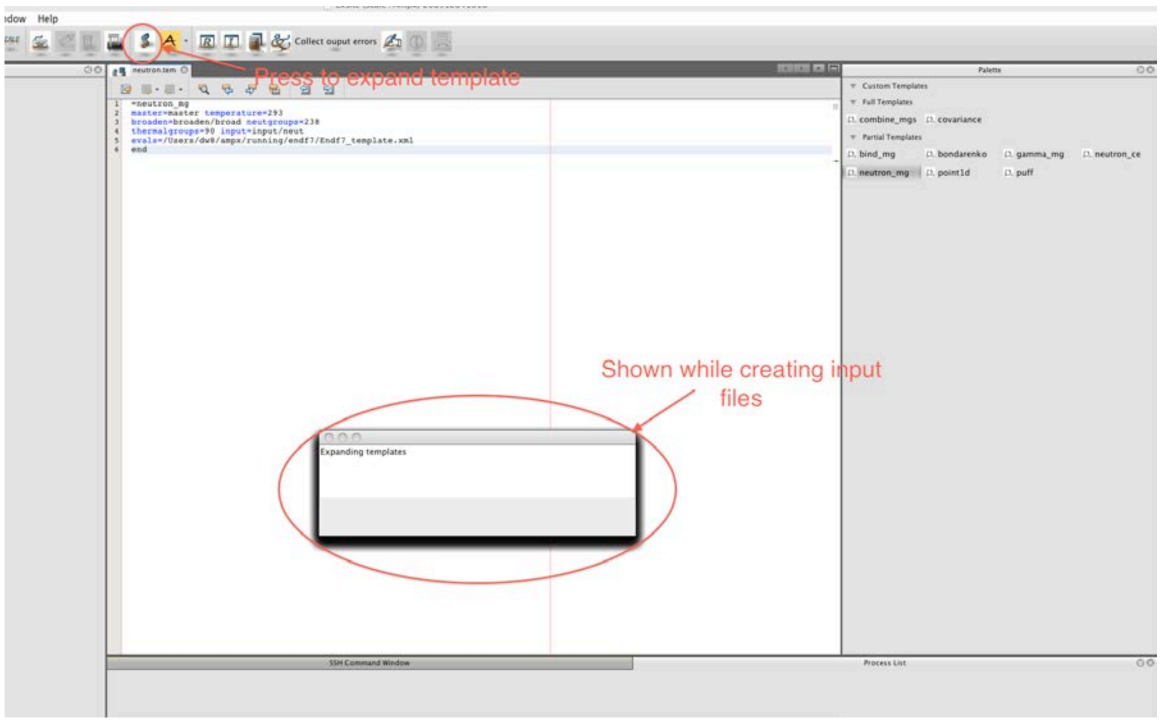
Use: ☐ dofile7 yes
Make a master for moderator materials (ENDF evaluation contains File 7)

Use: ✓ input input/neut
Name of the AMPX input files to create. Tag name will be appended

Use: ✓ evals /Users/dw8/ampx/running/endf7/Endf7_template.xml
Pick Endf xml summary listing or an endf file

Use: ☐ absolute ☐
Should files names in the input file appear as typed

Fig. 8. Add parameters to create the input files.



```
=neutron_mg
master=master temperature=293
broaden=broaden/broad neutgroups=238
thermalgroups=90 input=input/neut
evals=/Users/dw8/ampx/running/endf7/Endf7_template.xml
end
```

Fig. 9. Expand the templates into input files.

To generate an MG library, the input files generated from the templates listed below are run in the order specified:

1. point1d – generates point-wise cross section data. The cross section data are broadened to user-specified temperatures. All temperatures at which Bondarenko data are wanted must included.

2. neutron_mg – generates neutron MG data. This includes free gas data if desired, along with thermal data for thermal moderators if desired. In addition, gamma production data are generated if selected by the user.

3. gamma_mg – generates gamma MG data if a coupled library is desired. This template is used with the corresponding input files.

4. ptable – generates probability tables if Bondarenko factors from the probability table are desired.

5. bondarenko – generates full range Bondarenko factors. Alternately, bondarenko_prob is used to generate the Bondarenko factors using the probability method in the unresolved resonance range.

6. bind_mg – combines the data generated in steps 1–5 into one library. This makes one MG library for each evaluation.

7. combine_mgs – combines the libraries for each evaluation created in step 5 into a final master library.

If the user wants to create a master library file for a single evaluation, the template “master” can be used, which combines steps 1–6 except for thermal moderators. If the library is to contain thermal moderator data, then steps 1–6 must be followed.

To create a neutron CE library, the input files generated from the templates listed below must be run in the order specified:

1. point1d – generates point-wise cross section data. The cross section data are broadened to user-specified temperatures.

2. ptable – generates probability tables for evaluations that contain unresolved resonance data.

3. neutron_ce_partial – generates CE libraries.

The template neutron_ce combines steps 1–3 into one template.

To generate decay libraries, the inputs generated by the template listed below are run in the order specified:

1. origenlib –generates ORIGEN libraries (AMPX working libraries with special MT values encoding the level of the parent and child) for each desired evaluation.

2. combine_mgs – combines the libraries created in step 1 into one library.

To generate covariance libraries, the inputs generated by the following template are run:

1. master – creates a neutron master library.

2. point1d – creates point-wise data.

3. puff – creates the covariance matrices. The input file uses either the data generated in step 1 or step 2, depending on how the cross section data should be passed to module PUFF-IV.

4. combine_cov – combines the covariance matrices produced in step 3 into one library.

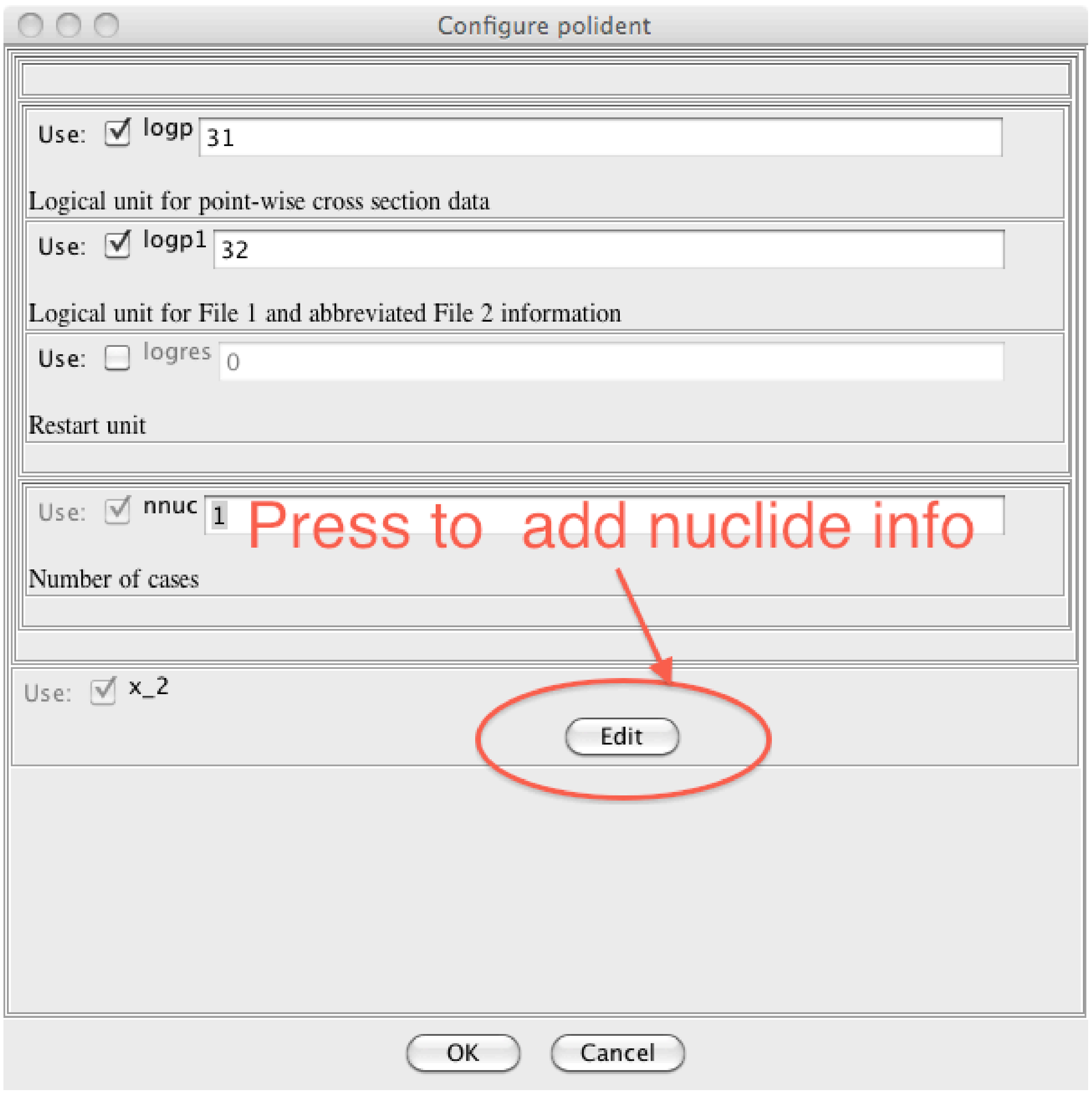
If the user is producing covariance information for one evaluation, the template covariance, which combines steps 1–3 into one template, can also be used.

ExSite provides some templates to generate SCALE input files for infinite dilute test cases that provide a quick check for the generated libraries:

- allnucinf – infinite homogenous medium case
- allnuclat – a lattice case to test data in the thermal range

4.3 EDIT INPUT FILES

ExSite can be used to edit and create AMPX input files. To read an input file, the user opens a new input file or selects an existing input. Once the file is opened, a palette appears. The user drags one of the input modules to the input editor in the same the way the template input palette was used in Figure 7. If an entry repeats, an additional dialog will often appear. An example is the input for POLIDENT. The initial dialog is shown in Fig. 10.



Use: ✓ logp 31

Logical unit for point-wise cross section data

Use: ✓ logp1 32

Logical unit for File 1 and abbreviated File 2 information

Use: ☐ logres 0

Restart unit

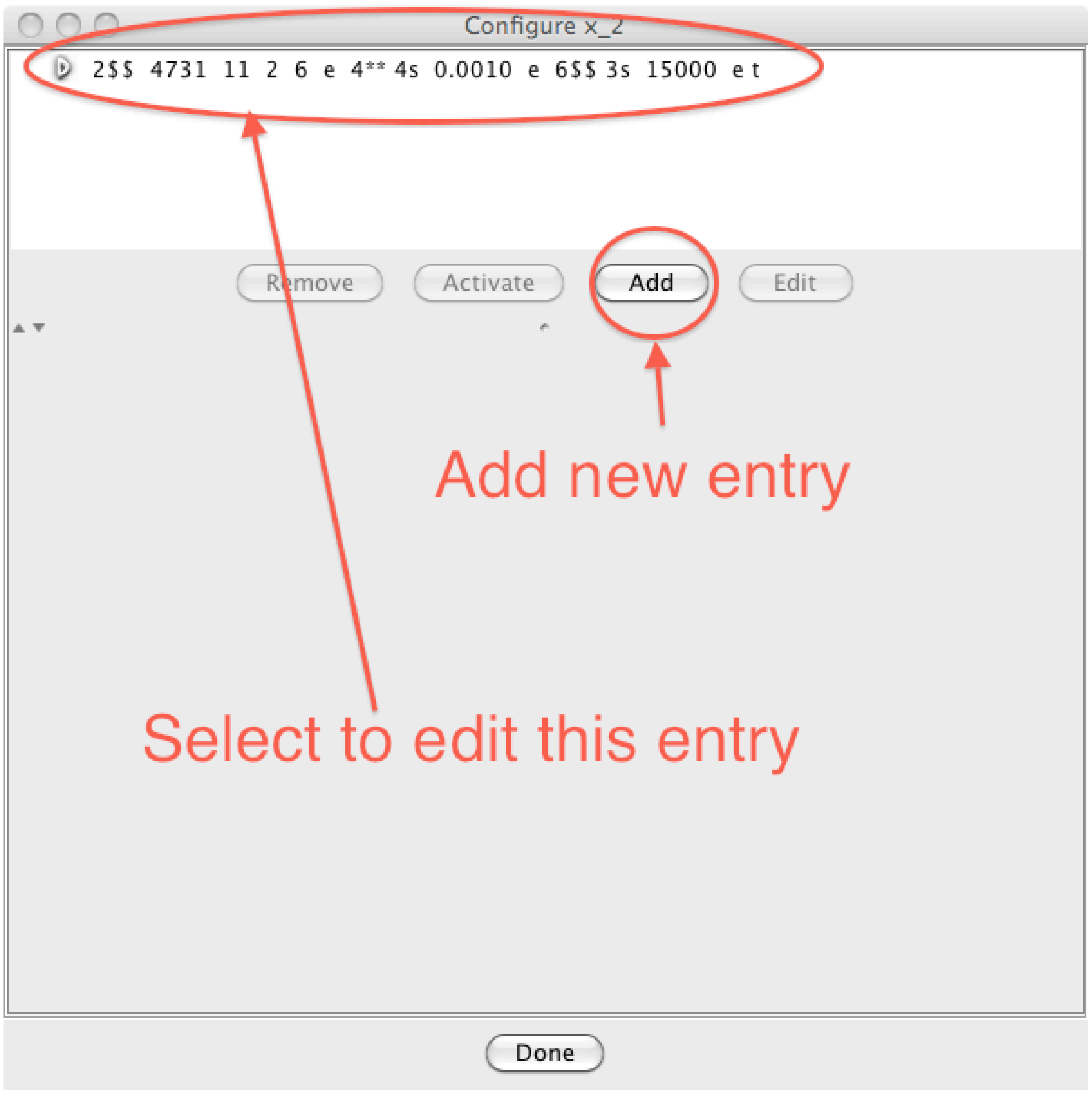
Use: ✓ nnuc 1

Number of cases

Use: ✓ x_2

Fig. 10. Add nuclide info for POLIDENT.

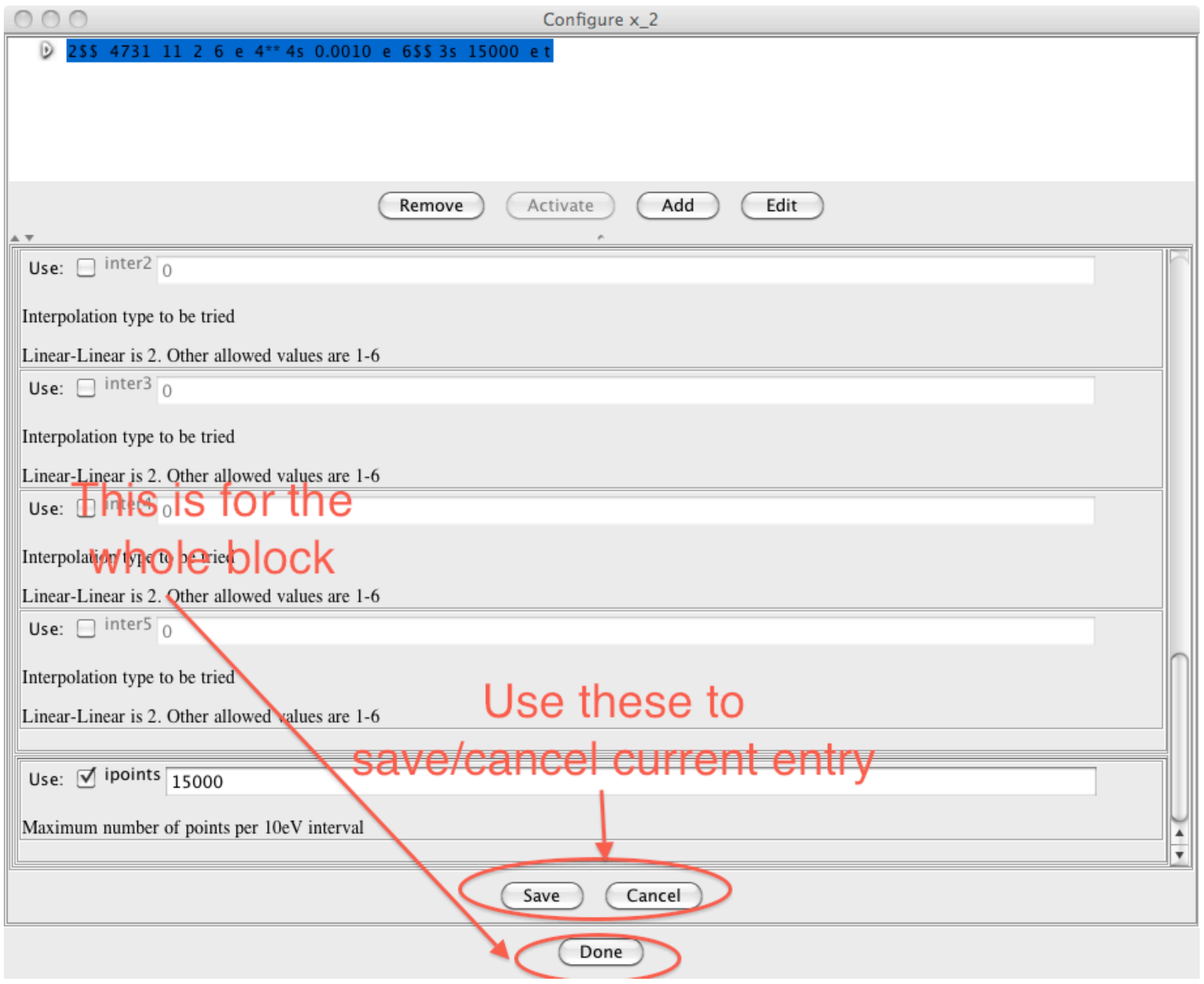
Once the “Edit” button is pressed, an additional dialog appears. The dialog is shown in Figure 11. The “Add” button adds a new entry. If there are already entries, selection of the desired entry will open it for editing. The “Remove” and “Activate” button will also become available, enabling the user to remove or reactivate an entry.



```
2$$ 4731 11 2 6 e 4** 4s 0.0010 e 6$$ 3s 15000 e t
```

Fig. 11. Additional dialog for repeating entries.

Once the entry is open for editing, the user can edit the file, as shown in Fig. 12. Please note that the "Save" and "Cancel" buttons pertain to the current entry, and the "Done" button closes the secondary dialog.



Configure x_2

```
2$$ 4731 11 2 6 e 4** 4s 0.0010 e 6$$ 3s 15000 e t
```

Use: ☐ inter2 0

Interpolation type to be tried

Linear-Linear is 2. Other allowed values are 1-6

Use: ☐ inter3 0

Interpolation type to be tried

Linear-Linear is 2. Other allowed values are 1-6

Use: ☐ inter4 0

Interpolation type to be tried

Linear-Linear is 2. Other allowed values are 1-6

Use: ☐ inter5 0

Interpolation type to be tried

Linear-Linear is 2. Other allowed values are 1-6

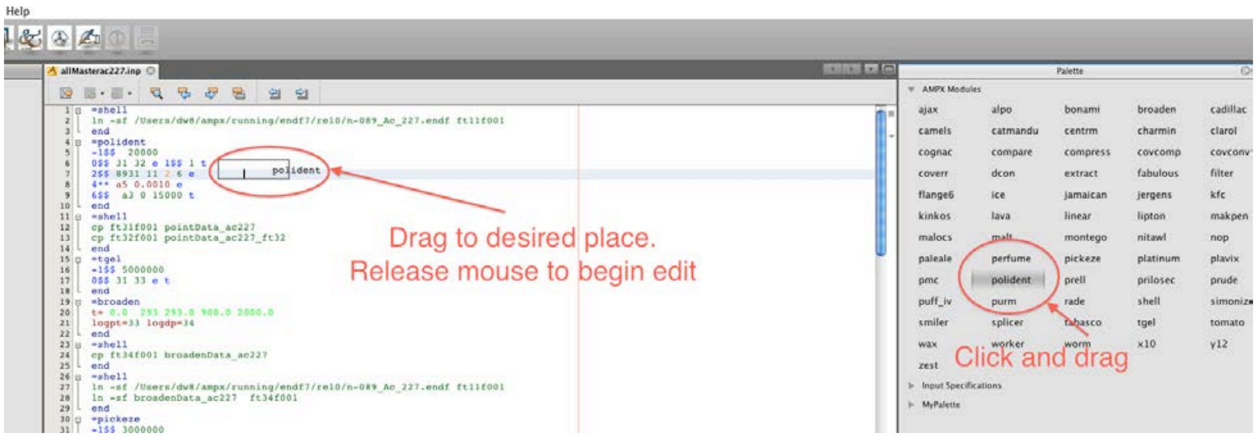
Use: ☑ ipoints 15000

Maximum number of points per 10eV interval

Remove
Activate
Add
Edit
This is for the whole block
Use these to save/cancel current entry
Save
Cancel
Done

Fig. 12. Edit repeating entry.

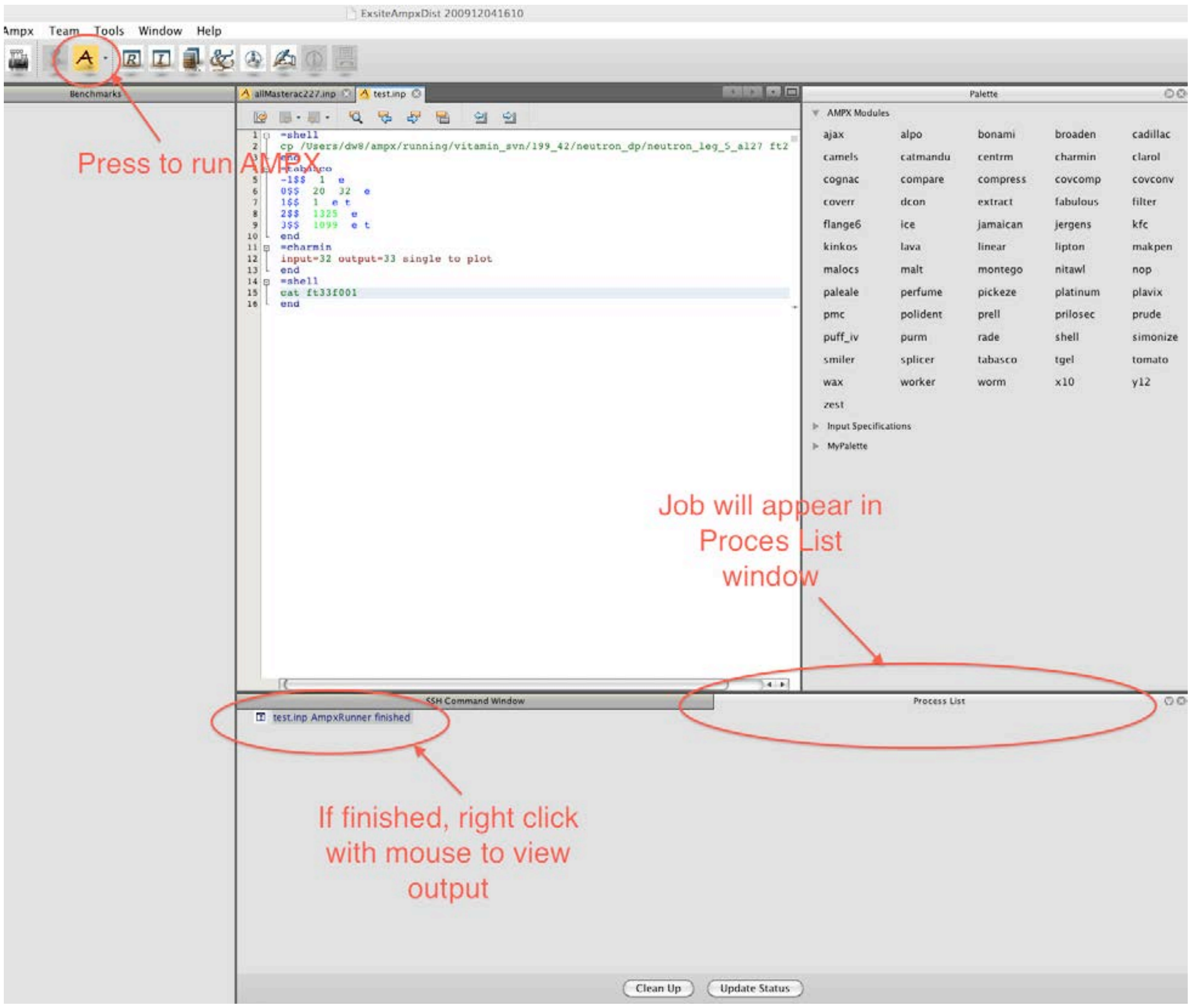
The GUI allows the user to add new input instructions and to modify existing ones. To edit an existing entry, the user simply drags the desired input to the desired position, as shown in Fig. 13.



```
=shell
ln -sf /Users/dw8/ampx/running/endf7/rel0/n-089_Ac_227.endf ft11f001
end
=polident
-1$$  20000
0$$ 31 32 e 1$$ 1 t
2$$ 8931 11 2 6 e
4** a5 0.0010 e
6$$  a3 0 15000 t
end
=shell
cp ft31f001 pointData_ac227
cp ft32f001 pointData_ac227_ft32
end
=tgel
-1$$ 5000000
0$$ 31 33 e t
end
=broaden
t= 0.0  293 293.0 900.0 2000.0
logpt=33 logdp=34
end
=shell
cp ft34f001 broadenData_ac227
end
=shell
ln -sf /Users/dw8/ampx/running/endf7/rel0/n-089_Ac_227.endf ft11f001
ln -sf broadenData_ac227  ft34f001
end
=pickeze
-1$$ 3000000
```

Fig. 13. Drag to edit input instructions.

To run an input file, the user presses the “A” button, as shown in Fig. 14. This will add the task into the “Process List” window. The status of the job will be updated automatically, or the user can press “Update Status” to immediately update. Once the job is finished, the user can right click on the job title and select to view the output or the message file. Input files can also be run directly from the command line using ampxrte.

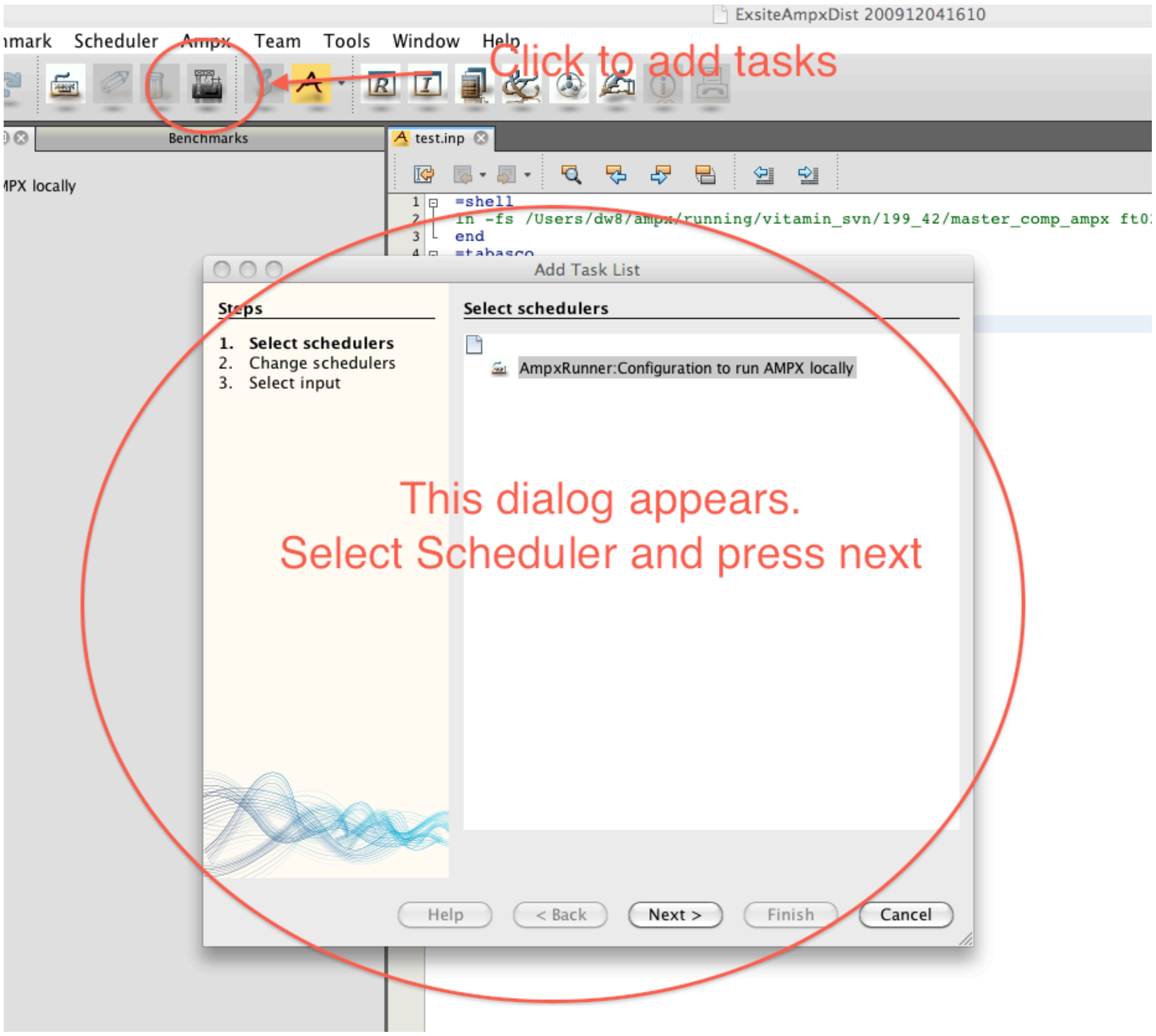

ExsiteAmpxDist 200912041610
Ampx Team Tools Window Help
Benchmarks
allMasterac227.inp
test.inp
Palette
AMPX Modules
ajax alpo bonami broaden cadillac
camels catmandu centrm charmin clarol
cognac compare compress covcomp covconv
coverr dcon extract fabulous filter
flange6 ice jamaican jergens kfc
kinkos lava linear lipton makpen
malocs malt montego nitawl nop
paleale perfume pickeze platinum plavix
pmc polident prell prilosec prude
puff_iv purm rade shell simonize
smiler splicer tabasco tgel tomato
wax worker worm x10 y12
zest
Input Specifications
MyPalette
Press to run AMPX
Job will appear in Proces List window
SSH Command Window
Process List
If finished, right click with mouse to view output
Clean Up
Update Status

```
=shell
cp /Users/dw8/ampx/running/vitamin_svn/199_42/neutron_dp/neutron_leg_5_al27 ft2
end
=tabasco
-1$$  1  e
0$$  20  32  e
1$$  1  e t
2$$  1325  e
3$$  1099  e t
end
=charmin
input=32 output=33 single to plot
end
=shell
cat ft33f001
end
```

test.inp AmpxRunner finished

Fig. 14. Run AMPX input file in ExSite.

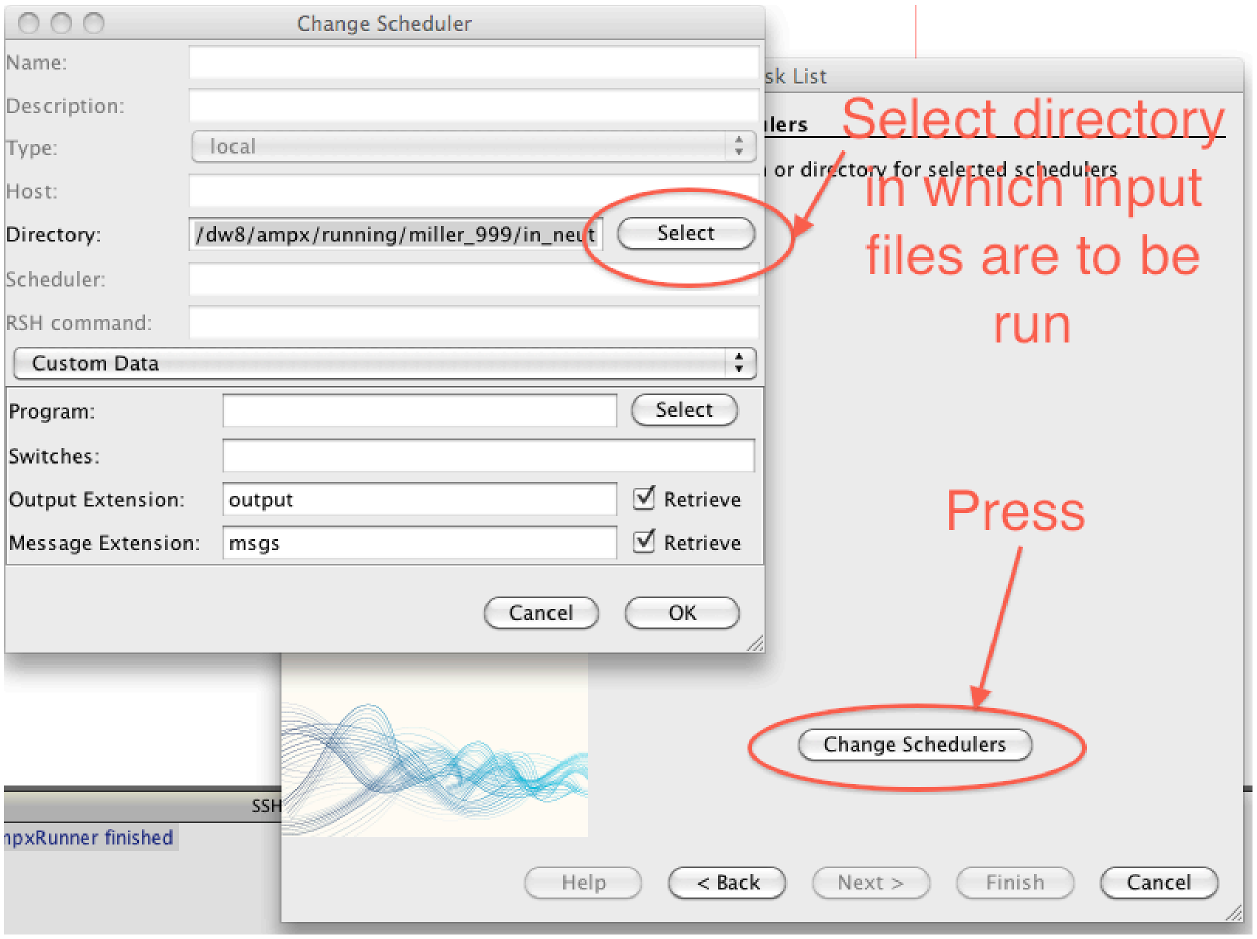
Several tasks can be run in order. This is very helpful if processing all the input files for a library. To do so, select “Add Task List” from the “Scheduler” menu. As shown in Fig. 15, a new wizard appears that will guide the user through the process.



```
=shell
 ln -fs /Users/dw8/ampx/running/vitamin_svn/199_42/master_comp_ampx ft0
end
=tabasco
```

Fig. 15. Start wizard to add more than one task.

To start the wizard for adding more than one task, the user must select the directory in which the input files are to be run. This is not necessarily the directory in which the input files are located. If the input files are not in the same directory in which they will be run, they must be copied to that directory prior to starting AMPX. In a later step, the output and message file generated by AMPX will be returned to the original directory of the input file. As shown in Fig. 16, the user must press the "Change Scheduler" button before moving on to the next step.



Change Scheduler

Name:

Description:

Type: local

Host:

Directory: /dw8/ampx/running/miller_999/in_neut

Scheduler:

RSH command:

Custom Data

Program:

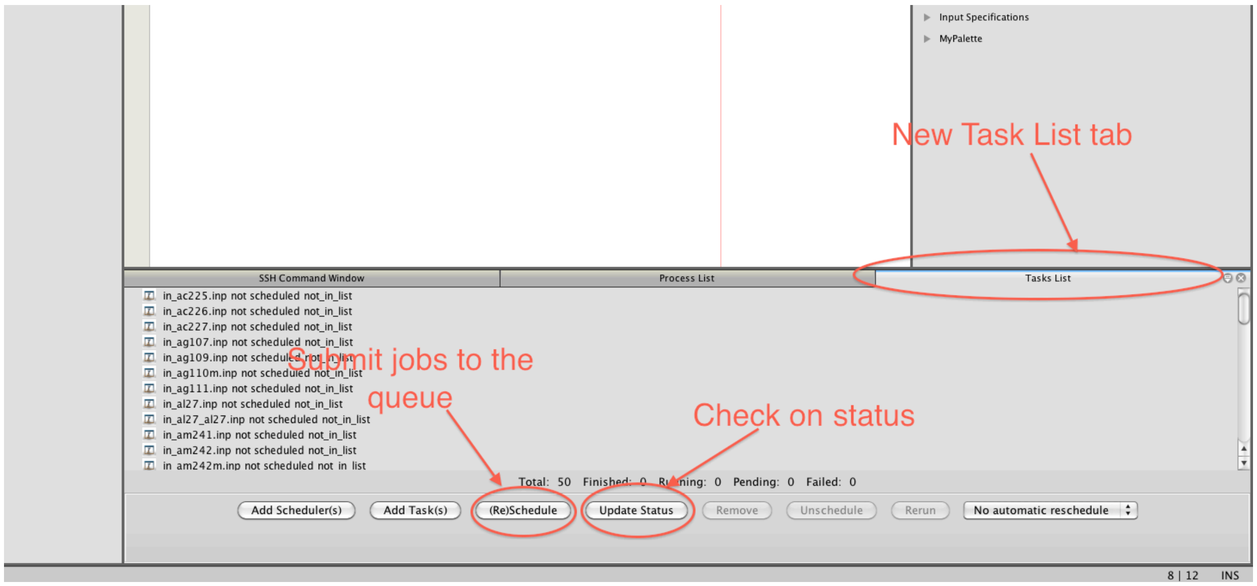
Switches:

Output Extension: output ✓ Retrieve

Message Extension: msgs ✓ Retrieve

Fig. 16. Select the directory in which to run AMPX.

In the next step, the user selects the input files that should be run. After pressing the “Finish” button, the selected tasks are displayed in a new “Task List” bar, as shown in Fig. 17. At this point, the jobs have not yet been started. To start the jobs, press the “(Re)Schedule” button, which will submit the jobs into the queue. To check on the status of the jobs, press the “Update Status” button. On a Unix/Linux or Mac operating system, the program can now be terminated, and the jobs will continue to run in the background.



```
in_ac225.inp not scheduled not_in_list
in_ac226.inp not scheduled not_in_list
in_ac227.inp not scheduled not_in_list
in_ag107.inp not scheduled not_in_list
in_ag109.inp not scheduled not_in_list
in_ag110m.inp not scheduled not_in_list
in_ag111.inp not scheduled not_in_list
in_al27.inp not scheduled not_in_list
in_al27_al27.inp not scheduled not_in_list
in_am241.inp not scheduled not_in_list
in_am242.inp not scheduled not_in_list
in_am242m.inp not scheduled not_in_list

Total:  50   Finished:  0   Running:  0   Pending:  0   Failed:  0
```

Fig. 17. Manage queue of submitted jobs.

5. DATA STRUCTURES

In the processing of nuclear data, AMPX must accommodate different data structures, all of which are encapsulated in a C++ or Fortran object. This allows easy access to the underlying data independent of the disk format of the data. There are three broad categories of data used in AMPX: (1) point-wise data, which includes cross section data and kinematic data, (2) group-averaged data, which includes all cross section data and matrices in a MG library, and (3) probability data, which includes probability tables and kinematic data in a CE library. In a given category, many of the same manipulations are applicable, and similar coding can be shared for all data structures in the same category.

5.1 POINT-WISE DATA

There are two types of point-wise: (1) 1-D data like cross sections as a function of energy, and (2) kinematic data as a function of incident energy, exit energy, and exit angle. The basic functionalities needed to accommodate point-wise data involve interpolation at any desired energy or angle, basic arithmetic combinations of several point-wise data of the same type, and comparison between two different point-wise data of the same type. AMPX includes template classes that capture parts of the procedure common to all data types and then adds specialization as needed. As in ENDF, a unique number describes each reaction. Reaction numbers are the same as in ENDF, augmented with additional values for special data needed in the transport codes.

Point-wise data allow discontinuities (i.e., there can be two identical energy or angle values with different function values). This frequently occurs if two different evaluations are used in different energy regions. Special care needs to be taken at these points if interpolating or manipulating the data to preserve these discontinuities.

5.1.1 1-D data

1-D point-wise data are stored in a format very similar to the TAB1 structure in the ENDF manual [**Error! Bookmark not defined.**], which supports energy and cross section values, along with a prescription on how to interpolate the cross section on grid points not given. The same interpolation values as in ENDF are allowed: (1) histogram, in which y is constant in x, (2) linear-linear, in which y is linear in x, (3) linear-log, in which y is linear in ln(x), (4) log-linear, in which ln(y) is linear in x, and (5) log-log, in which ln(y) is linear in ln(x). Internally, AMPX always uses a linear-linear interpolation. Each TAB1 record also includes a header record that contains the reaction number and the temperature of the cross sections data.

5.1.2 Kinematic data

Kinematic data are used in three different forms in AMPX:

$$f(E,E',\mu) = \frac{d^2\sigma}{dE'd\Omega}(E \to E',\mu)$$

double-differential

$$f(E,E',\mu) = \sum_l \frac{2l+1}{2} a_l(E,E')P_l(\mu)$$

Legendre coefficients, and

$$f(E,E',\mu) = \sum_l c_l\,(E,E')$$

cosine moments

where

E is the	incident particle energy,
E' is the	exit particle energy,
μ is the	cosine of the exit angle,
$P_l(\mu)$ represents the	Legendre polynomials of order l,
$a_l(E,E')$ represents the	Legendre coefficients of order l, and
$c_l(E,E')$ represents the	cosine coefficients, calculated as $\int_{-1}^{1} f(E,E',\mu)\mu^l \, d\mu$

The kinematic data are stored in C++ classes that allow easy manipulation of the underlying data. The angular distribution for a given incident and exit energy is encapsulated in an ExitEnergy object that stores the exit energy and the angular distribution. An integer flag indicates whether the distribution is given in Legendre moments, cosine moments, or as a tabulated cosine/value distribution. The distribution can be flagged as being discrete, as would be encountered for coherent elastic scattering in crystalline materials. A list of ExitEnergy objects is collected in an IncidentEnergy object, which also stores the value of the incident energy. A flag indicates whether the exit energy distribution describes an elastic or discrete-inelastic reaction, information needed for conversions between center-of-mass and laboratory system. The exit-energy distribution can also be marked as discrete, as would be encountered for discrete gamma lines. A list of IncidentEnergy objects is collected in a KinematicBlock object, which also stores reaction parameters like the reaction type, mass ratios of incident and exit particles, and the Q value. A flag indicates whether the data are given in the center-of-mass or laboratory frame of reference and whether the data are absolute or relative (i.e., divided by the 1-D cross section at the given incident energy).

5.1.3 Mesh generation

When combining and reconstructing point-wise data, it is often necessary to create an energy or angle mesh for the new distribution. The new mesh is generated by first calculating an initial mesh, most often the union mesh of the distributions to be used in the operation. If any of the distributions has a discontinuity, they are added as a discontinuity to the union mesh. If the distributions do not all start or end at the same energies, the starting and end points are added as discontinuities as well. For example if combining cross section data in the resolved range that ends at E_r with cross section from the unresolved range that starts at E_r, the energy E_r is added twice to the mesh. The mesh is then refined using a halving scheme between two energies or angles: interpolate the distribution at the middle energy or angle and calculate the distribution from an appropriate function and compare the two resulting distributions. If the two distributions agree within a user-supplied precision, the mesh between the two points is dense enough, and the process is repeated with the next set of points. If the two distributions are not the same, the middle energy or angle is added to the mesh, and the process is repeated between the low value and the middle value. A similar procedure is applied to thin the number of mesh points, where the actual distribution at a given energy or angle point is compared to the interpolated distribution. If those two distributions are equal within a user-specified precision, the intermediate energy or angle point is eliminated from the mesh. Mesh points with a discontinuity are never eliminated during a thinning of the mesh. The resulting distribution is calculated on the final mesh. If the new distribution results from an arithmetic operation on other distributions, such as adding or subtracting, special care needs to be taken at discontinuities. If any of the distribution has a discontinuity, it is added as a discontinuity on the final mesh and the calculation is performed twice: once by interpolating the values on the initial distribution on the left side and once on the right side of the discontinuity, which will return the same value if the distribution has no discontinuity. As pointed out, if the contributing distributions do not start and end at the same energy or angle, a discontinuity is added to the final mesh at the starting and ending points and the calculation is performed twice: once with the distribution that starts or ends and once without that distribution.

AMPX has very general codes that combine point-wise data from different distributions of the same type or that generate distributions from a function. The general code calls on specialized functions that interpolate and compare the distribution at a given energy or angle, which are described in more detail below. In addition, the user can supply arithmetic functions that are to be performed on the initial distributions to form the desired distribution or theoretical functions (such as a Maxwellian) to generate a distribution and a suitable mesh.

The advantage of generalized code is that mesh generation and treatment of discontinuities is localized.

5.1.4 Interpolation

In the case of 1-D data, interpolation at any energy value not given simply uses the interpolation rules (1–5 as described above) to calculate the intermediate value. The same is true if interpolating for a desired angle if the kinematic data are given in double-differential format. All point-wise data can contain discontinuities and interpolation at a discontinuity point can be ambiguous. Therefore, all routines that interpolate point-wise data have a flag indicating whether the left, right, or intermediate value is wanted at the point of the discontinuity. The flag has no effect at other energy or angle values.

5.1.4.1 Interpolating in exit energy

If the angular distribution is given as Legendre coefficients or as cosine moments, the moments are simply interpolated for the desired exit energy at the correct order using the specified interpolation law. If the angular distribution is given in tabulated form (i.e., as a function of angle cosine), a union mesh of the angles from the two exit energies is formed and refined via a halving scheme. The value for each angle on the resulting mesh is calculated by interpolating the value for the angle at the two exit energies and interpolated at the desired exit energy.

5.1.4.2 Interpolation in incident energy

For interpolation in incident energy, ENDF and AMPX define additional interpolation laws besides the one defined for other point-wise data: corresponding point interpolation (11–15) and unit-based interpolation (21–25). Assuming there are two distributions—$f_1(E_1, E', \mu)$ and $f_2(E_2, E', \mu)$—where $f_1(E_1, E', \mu)$ extends from E'^{low}_1 to E'^{high}_1, and $f_2(E_2, E', \mu)$ extends from E'^{low}_2 to E'^{high}_2. The objective is to interpolate $f_m(E_m, E', \mu)$. For interpolation schemes 1–5, one proceeds as in the case of angular data, generating an initial union mesh of exit energies as refined by a halving scheme. For each exit energy on the union mesh, the new exit energy distribution is interpolated as described above.

For unit-based and corresponding point interpolation, the lowest exit energy at the intermediate incident energy is

$$E'^{low}_m = E'^{low}_1 + \frac{E_m - E_1}{E_2 - E_1}\left[E'^{low}_2 - E'^{low}_1\right], \qquad \textbf{(Error! Bookmark not defined.}1)$$

and the highest exit energy is

$$E'^{high}_{m} = E'^{high}_{1} + \frac{E_m - E_1}{E_2 - E_1}\left[E'^{high}_{2} - E'^{high}_{1}\right], \tag{2}$$

with a difference in calculating the distributions for intermediate exit energies. In both cases, corresponding exit energies E'$_1$ in the lower incident energy panel and E'$_2$ in the upper incident energy panel are determined. From these, a unit-based value is calculated:

$$x'_1 = E'_1 - E'^{low}_{1}$$ (**Error! Bookmark not defined.**3)

and

$$x'_2 = E'_2 - E'^{low}_{2},$$ (**Error! Bookmark not defined.**4)

which is then interpolated to a value of

$$x'_m = x'_1 + \frac{x'_2 - x'_1}{E_2 - E_1}(E_m - E_1)\ ,$$ (**Error! Bookmark not defined.**5)

and the exit energy in the intermediate panel is

$$E'_m = \left(E'^{high}_{m} - E'^{low}_{m}\right)x'_m + E'^{low}_{m}.$$ (**Error! Bookmark not defined.**6)

The value of the distribution at E'$_m$ is then calculated using the interpolation law (1–5), obtained by subtracting 10 or 20 from the given interpolation scheme. The difference is in the determination of the corresponding points. If using the unit based approach (21–25), points are chosen so that:

$$\frac{E'_1 - E'^{low}_{1}}{E'^{high}_{1} - E'^{low}_{1}} = \frac{E'_2 - E'^{low}_{2}}{E'^{high}_{2} - E'^{low}_{2}}.$$ (**Error! Bookmark not defined.**7)

For the method of corresponding points (11–15), values E'1 and E'2 are chosen so that the normalized cumulative integral over the exit energies agree:

$$\frac{\int_{E'^{low}_{1}}^{E'_1} f_1\left(E_1, E', \mu\right)dE'}{\int_{E'^{low}_{1}}^{E^{high}_{1}} f_1\left(E_1, E', \mu\right)dE'} = \frac{\int_{E^{low}_{1}}^{E'_2} f_2\left(E_2, E', \mu\right)dE'}{\int_{E'^{low}_{2}}^{E'^{high}_{2}} f_2\left(E_2, E', \mu\right)dE'}$$ (**Error! Bookmark not defined.**8)

.

AMPX supports all of the interpolation laws and uses them as given in the evaluation. However, internally ,an interpolation value of 22 is used, and additional mesh points are added if needed to describe the distribution as given by the evaluator. The unit-based approach is sensitive to small tails in the distribution since the exit energy range must be well defined. AMPX provides a method that cuts lower or upper exit energies from a distribution if these energies do not substantially contribute to the total integral of the distribution. To determine the new lower exit energy, the integral is calculated first:

$$I_{tot} = \int_{E'_{low}}^{E'_{high}} f(E, E', \mu)\, dE', \qquad \text{(**Error! Bookmark not defined.**9)}$$

and then the partial integral

$$I_{part} = \int_{E'^{new}_{low}}^{E'_{high}} f(E, E', \mu)\, dE'. \qquad \text{(**Error! Bookmark not defined.**10)}$$

If $\left(\frac{|I_{tot} - I_{part}|}{I_{tot}}\right)$ is smaller than a user-defined precision, the energy range is changed to the new lower exit energy. A similar procedure is performed for the upper exit-energy limit.

In addition, the unit-based interpolation is slightly modified if the distribution has upscatter terms in order to preserve the correct upscatter in the interpolated panel. Two units are being used—one extending from $[E'_{low}, E]$, and the other from $[E, E'_{high}]$, where E is the incident energy. Depending on the distribution and the distance between the provided incident energies, the shape of the exit energy distribution can change, as seen in Fig. 18.

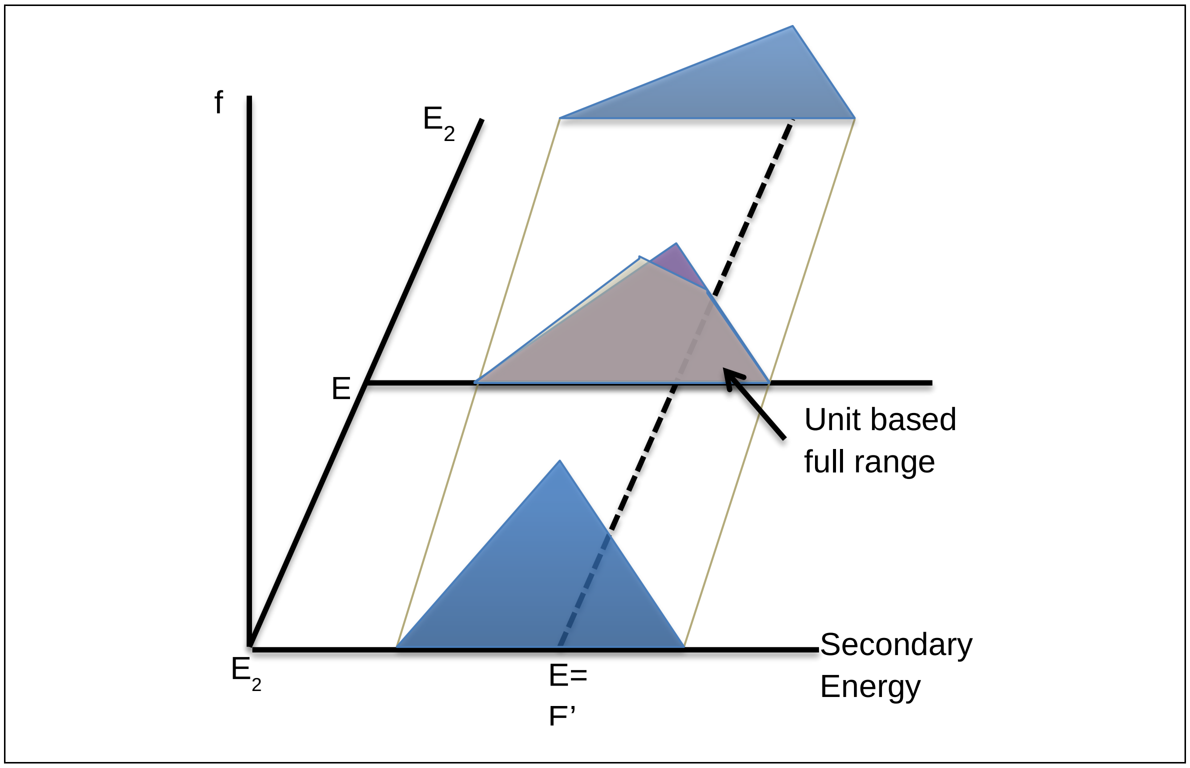


Fig. 18. Difference between using unit-based over the full range or using unit-based between low exit energy and incident energy and incident energy and high incident energy.

5.1.5 Comparison

It is frequently required to compare two values or distributions to determine whether additional data are needed to accurately describe the distribution. Comparison is always made by interpolating a new distribution at the desired energy or angle. While this may be computationally intensive, it provides more precise results. To ensure that two distributions are equal, one steps over a union mesh of energy or angle and interpolates two auxiliary distributions, one from each of the two distributions, to be compared at each union mesh point. These two distributions can now be directly compared. If any of the distributions has a discontinuity, the comparison is made first with the left value then with the right value.

The easiest comparison is for 1-D data or angular distributions. As outlined above, values from each point on the union grid are interpolated from both distributions. If any absolute value of those two values is larger than a user-defined cut-off, the two values are assumed to be the same if the relative difference is smaller than a user-defined precision. Otherwise, the absolute difference is compared, which in effect assumes that the two values are zero. The cut-off value ensures that very small values are treated as zero and for example avoids adding unnecessary points to the mesh if comparison is used in connection with a halving scheme. Care needs to be taken so that the cut-off value does not eliminate major features of the distribution. A good choice is a value which, integrated over all energies and/or angles, does not change the total integral more than a user-defined precision. This is the default value chosen in AMPX, but it can be overridden by the user. Two 1-D data or angular distributions are equal within in a user-specified precision if all values on the union grid satisfy the above comparison.

Two exit energy distributions are compared by comparing the angular distributions interpolated on the union grid. Two incident energy distributions are compared by comparing the exit energy distributions on a refined union mesh.

Comparison is often used to generate a suitable mesh in incident and exit energy and angle. This procedure works well for exit energy and angles, but it can lead to an extremely fine mesh for the incident energies. Therefore, if an incident energy mesh needs to be refined, then only the exit energy range is compared. If the exit energy range can be interpolated via unit-based interpolation, then the mesh is assumed to be dense enough. In order to thin an incident energy mesh, the full distribution is always used for the comparison. This procedure relies on a good initial grid for the incident energy being given, which is true if the grid is supplied by the evaluator.

5.1.6 Conversion for kinematic data

Angular distributions for a given exit energy can be given in Legendre or cosine moments or in tabulated form. Conversion functions to convert between these formats are provided in AMPX. Conversion between cosine and Legendre coefficient is a simple transformation of the coefficients: the order does not change. Conversion to tabulated form is always based on Legendre moments. An initial equidistant cosine grid between -1 and 1 is selected on which the Legendre moments are expanded. The grid is refined using the halving scheme described above.

The final kinematic distributions are desired in the laboratory system. Therefore a conversion from the center-of-mass system is often needed. Conversion is only performed for tabulated double-differential formats; a conversion to tabulated format is performed before the conversion if needed.

For elastic and discrete-inelastic incident neutron reactions, there is only one exit energy in the center-of-mass system for a given incident energy, and in the laboratory system, each exit energy has exactly one exit angle. To convert the kinematic data for a given incident energy, AMPX generates an initial grid of exit energies in the laboratory system by stepping over equidistant points in the angle distribution in the center-of-mass system. The grid is then further refined by a halving scheme used on the center-of-mass angle until the exit and angle distribution in the laboratory system can be reproduced to a user-defined precision. For some nuclides and reactions, two different exit energies in the laboratory system can have the same exit angle. In these cases, the Jacobian that transforms the value of the distribution from center-of-mass to laboratory system can be infinite, and the above halving system does not work correctly when close to the discontinuity. In these cases, the angle in the center-of-mass system where the discontinuity occurs is determined, and two laboratory angles on either side of the value are chosen so that the Jacobian can still be calculated within numerical precision. The halving scheme is then employed on the center-of-mass angles below the lower of the two angles and above the higher of the two angles. In addition, a much denser initial grid is chosen for the lower angular region. If the final format is moments instead of tabulated, the Jacobian for the transformation from center-of-mass to laboratory system is not required for the angle, as integration is performed over the whole angle range. Therefore, the Jacobian to convert between center-of-mass and laboratory system angle is not applied, and the distribution is immediately converted to cosine moments. The Jacobian to convert between center-of-mass and laboratory system exit energies is still applied. If the distribution is desired in tabulated form, both Jacobians are applied.

The incident energy grid used in the center-of-mass system can often be coarser than is required in the laboratory system, especially for threshold reactions. Additional incident energies in the laboratory system are calculated by interpolating a distribution for the desired incident energy from the center-of-mass kinematic data and then converting the distribution to the laboratory system as described above. The additional incident energies are chosen so that the exit energy range can be determined via unit-based interpolation. The exit energy distributions themselves are not compared. The laboratory incident energy grid is thinned if possible, but this time it is based on a full comparison of the exit energy and angle distribution.

The procedure to convert other reactions from center-of-lab to the laboratory system is similar to the conversion for elastic incident neutron reactions except that there can be more than one exit angle for a given exit energy. In this case, the distribution in the center-of-mass and laboratory system will always be tabulated; there is no special treatment if the final desired format is an angular distribution in moments.

If the kinematic data for a threshold reaction are given in the center-of-mass system, ENDF sometimes contains exit energy distributions at incident energy slightly below or at the threshold. Conversion of these panels to the laboratory system is numerically unstable. Therefore, only incident energies slightly above the threshold are converted. In order to preserve the distribution, additional incident energies above the threshold are inserted into the distribution in the center-of-mass system before conversion to the lab system.

5.2 GROUP-AVERAGED DATA

Group-averaged data used by AMPX are categorized in three broad groups: (1) 1-D cross section data, which includes f-factors and subgroup data, (2) scattering matrices and (3) group-averaged covariance data. 1-D cross section data and scattering matrices are normally contained in a MG library. AMPX and SCALE share a C++ resource that keeps all MG data in memory and provides classes to access the data.

An integral over one set of 1-D data can always be calculated analytically since the interpolation is known. AMPX library functions provide methods for this simple integral.

The two point-wise data structures described above can be converted to group-averaged data using AMPX library functions. In the case of 1-D cross section data, a point-wise cross section TAB1 object is multiplied with a flux, which is also given as a TAB1 object, and then it is integrated over all energies in a given group:

$$\frac{1}{\int_{E_1}^{E_2} \varphi(E)dE} \int_{E_1}^{E_2} \sigma(E)y(E)\varphi(E)\,dE$$

(**Error! Bookmark not defined.**11)

where $\sigma(E)$ is the cross section and $\varphi(E)$ is the flux. The yield $y(E)$ is only different from 1 if fission yields are calculated. This integral can be solved numerically if the two distributions use linear interpolation.

Scattering matrices are calculated for a given incident energy group and exit energy group, and they are always in Legendre moments. Please note that AMPX includes a $(2l + 1)$ factor for each matrix, where l is the order of the Legendre moment. Assuming the incident energy group i has boundaries $\lfloor E_1, E_2 \rfloor$ and the exit energy group j has boundaries $\lfloor E'_1, E'_2 \rfloor$, the scattering matrix element at $[i, j]$ for Legendre moment l is calculated as:

$$\frac{1}{\int_{E_1}^{E_2} \varphi(E)dE} \int_{E_1}^{E_2} dE y(E)\sigma(E)\varphi(E) \int_{E'_1}^{E'_2} f_l\Big(\quad E, E'\Big)\, dE',$$

(**Error! Bookmark not defined.**12)

where $\sigma(E)$ is the cross section, $\varphi(E)$ is the flux, and $f_l(E, E')$ is the exit energy distribution at incident energy E for the lth Legendre order. The yield $y(E)$ is only different from 1 if fission yield matrices are calculated. The inner integral at a given energy E is calculated using unit-based interpolation. Assuming the desired incident energy $E_m$lies between two incident energies present in the distribution— $f_l(E_1, E')$

and $f_l(E_2, E')$ the exit energy range $[E_m^l, E_m^h]$ is calculated at E_m using Eqs. (1) and (2), from which the unit-based exit group boundaries $x_1' = \frac{E_1' - E_m^l}{E_m^h - E_m^l}$ and $x_2' = \frac{E_2' - E_m^l}{E_m^h - E_m^l}$ are calculated in the intermediate panel. From this the corresponding x values can be calculated, as well as the integral boundaries in the two outer distributions. The integrals for the two outer distributions are then calculated, and a value at $E_m$is calculated by linear interpolation of the integrals. The outer integral is performed for all cosine orders at the same time using a fourth order Runge-Kutta method with adaptive step size [27], except when the angular distribution is discrete and the integral can be calculated analytically.

5.3 PROBABILITY DATA

For CE libraries, the kinematic data are needed in marginal (with respect to angle) and conditional (with respect to exit energy) cumulative probability functions. In the case of elastic and discrete inelastic reactions, the conversion is straightforward, as there is only one possible exit energy for a given exit angle. Thus the marginal cumulative probability function is simply the normalized angular distribution. For each angle, the conditional cumulative probability function has one value.

For other reactions, the code first determines an angular mesh for each incident energy that is dense enough so that the angular distribution for each exit energy can be represented on that mesh. On that incident energy-dependent angle mesh, integrate $f(E, E', \mu)$ is integrated over all E' values. The integral of this function in angle is the yield. This function, normalized so that the integral is 1, is the cumulative distribution in angle for the given incident energy. The cumulative distribution function is divided into a user-defined number of equiprobable bins, usually 32. If the distribution function can be described with less than the user-defined number of bins that are not equiprobable, then that grid is used instead to conserve disk space. For each bin, the conditional cumulative probability distribution for the exit energies is determined and saved.

6. NOTES ON SOME OF THE MODULES

Some modules in AMPX are shared with SCALE. Detailed description for these modules can be found in the SCALE6 manual [12]. The shared modules are: AJAX, AIM, CADILLAC, COGNAC, LAVA, ICE, PALEALE, MALOCS and RADE.

The functionality of some of the key modules used to generate MG and CE libraries are described below.

6.1 POLIDENT

The module POLIDENT is used to generate point-wise cross section data at 0 K. A more detailed description of the module can be found in the separate POLIDENT manual [25]. The most complicated procedure performed by POLIDENT is the mesh generation in the resolved resonance range, where the cross section data can vary rapidly as a function of energy. Consequently, the cross section function can be extremely dependent on the energy grid. If certain energy values or grid points are omitted, the structure of the resonances may not be represented correctly during reconstruction from the resonance parameters. Historically, energy-mesh-generation schemes begin with a very coarse grid, and points are added to the coarse grid using the halving scheme described above. Based on experience, this halving scheme, coupled with a coarse initial energy grid, can lead to inadequate representation of the resonance structure particular near inflection points. In an effort to avoid potential problems associated with the halving scheme, POLIDENT uses an adaptive mesh scheme largely based on a very fine energy mesh that is subsequently collapsed to an auxiliary structure using a user-defined convergence tolerance. In the fine mesh generation, a significant amount of effort is devoted to ensure that the detailed resonance structure is represented faithfully. Due to the very large number of fine grid points, the steps are only performed for slices of the whole resolved resonance energy range. The procedure is outlined below:

1. Within a given slice, determine the level spacing <D> and mean neutron-line width $<\Gamma_n>$, and calculate an initial step size of $\Delta E = 0.1\sqrt{\frac{\langle \Gamma_n \rangle}{\langle D \rangle}}$. If there are no resonances in the slice, at least 10 steps are required over the interval, and the minimal initial step size is 0.01. A slice starts with a width of 10eV but is extended in 10eV increments until it contains at least one resonance. This allows for handling isotopes with resolved resonance ranges that extend to very high energies.

2. Calculate the absorption, capture, fission, scattering and total cross section on the initial grid determined in step 1 using the appropriate resonance formalism on the grid generated in step 1. This is the initial fine grid mesh.

3. Numerically calculate the first and second derivative on the fine grid for total, capture, and fission. On a new auxiliary grid, add maxima and minima, defined by a sign change in the first derivative and inflection points, and defined by a sign change in the second derivative. In addition, add resonance energies to the auxiliary grid.

4. Add points to the fine grid between the inflection and maxima points. The number of points, N, added between inflection point E_i and maxima E_m is based on the following empirical criteria:

$$\tan^{-1}\left(\left|\frac{\sigma_i - \sigma_f}{E_i - E_f}\right|\right) < \frac{\pi}{6} \quad , \quad N = 10$$

$$\frac{\pi}{6} \le \tan^{-1}\left(\left|\frac{\sigma_i - \sigma_f}{E_i - E_f}\right|\right) < \frac{\pi}{3} \quad , \quad N = 15$$

$$\frac{\pi}{3} \le \tan^{-1}\left(\left|\frac{\sigma_i - \sigma_f}{E_i - E_f}\right|\right) < \frac{\pi}{2} \quad , \quad N = 20$$

(**Error! Bookmark not defined.**13)

1. Using a halving scheme, add more points to the fine mesh until all cross section data can be interpolated linearly.

2. Collapse the fine grid to the user grid. The user grid starts with all the critical points on the auxiliary grid. Then add enough points from the fine grid so that all cross section values on the fine grid can be linearly interpolated from points on the user grid within a user-specified precision.

After the mesh is determined, the cross section is calculated for each mesh energy using the resonance formalism given by the evaluator.

In the unresolved resonance range, the resonance self-shielding must be handled on a statistical basis, and the resonance parameters are given as averaged values for a given energy range. Only the single-level Breit-Wigner formalism is currently allowed in ENDF.

Resonances fit into families divided according to the angular momentum (l-value) of the nucleus and the angular momentum of the resonance (j-state). The unresolved resonance data in an ENDF/B evaluation are statistical parameters derived from the resonances in each family that can be resolved. These resolved resonances statistically predict how the resonances are spaced in the particular family and also the characteristics of the resonances (e.g., the relative size of the fission or elastic scattering component of the resonance). POLIDENT uses a method developed by R. N. Hwang at ANL for calculating unresolved cross sections as a function of energy [28]. In the unresolved resonance region, the resonance parameters are provided for the SLBW formalism, and the resonance widths are distributed according to a chi-squared distribution with a specified number of degrees of freedom. Flux weighted cross section values can be calculated over an evaluator-specified energy interval using the unresolved resonance parameters. In the averaging process, Hwang's method makes use of the narrow resonance (NR) approximation, and the resulting expressions for the average cross section values can be expressed in terms of fluctuation integrals that are also defined in terms of the Doppler broadening ψ and χ resonance line-shape functions. Despite the simplifications from the NR approximation, the resulting expressions are quite complicated and involve integration of the resonance widths over the evaluator-specified chi-square distributions. Hwang's method makes use of special Gaussian-like quadratures that permit integration over the probability distributions and the ultimate calculation of averaged point cross sections as a function of energy. These averaged point value curves are very smooth functions in energy, and no attempt is made to actually determine the extreme variation in the cross sections that appear in the real cross sections.

6.2 TGEL

The module recalculates redundant cross section data from the partial cross section values. Great care needs to be taken at discontinuities, which frequently arise at the transition between resolved and unresolved resonance range. A detailed description of the method used to add two point-wise data vectors is given in the POLIDENT manual [25]. The functions used are in the ampx90lib library and are general enough to add, subtract, divide, and multiply point-wise data.

The redundant cross sections should be calculated before the data are used in transport calculations. Note that, in contrast to the NJOY RECONR module [29], POLIDENT does not reconstruct redundant cross sections. The module TGEL has to be run before AMPX-generated redundant point data can be compared to NJOY-generated point data.

6.3 Y12

The ENDF formatted data files contain scattering matrix information for many reactions. To keep the actual ENDF tape brief, a variety of formats is used. The module Y12 converts all formats into a unified kinematics format suitable for use in other AMPX modules. The module Y12 produces scattering matrices where the angular distribution is either given in Legendre moments, cosine moments, or tabulated form. The distribution of exit energy and angle is always given in the laboratory system. Data from Files 3, 4, 5, 6, 7, 12, 13, 14, 15, 23, and 27 are processed. In the case of thermal neutron scattering data (File 7) and incident gamma data (Files 23, 26, and 27) Y12 also generates point-wise 1-D cross section data. For all other cases, point-wise 1-D cross section data are generated in module POLIDENT.

6.3.1 Processing of ENDF Tapes

ENDF data are read into intermediate objects before further processing. This shields the processing methods from reading the ENDF formatted files. Each reaction with kinematic data is encapsulated in a KinematicReaction object, which stores reaction specific data such as incident and exit particle, temperature, and Q value, and a list of KinematicYield objects. KinematicYield objects encapsulate the kinematic data contained in one or more KinematicData object and yield information. The yield information contained in a YieldData object is a list of zero or more 1-D cross section data that need to be summed and multiplied with the kinematic data to give the full desired kinematic data. Kinematic data can sometimes be separated into angular distribution and exit energy distribution, so the KinematicData object has a flag indicating whether the exit energy distribution, the angular distribution, or the full double differential distribution is given. If the kinematic data are already given in double differential form, either tabulated or as Legendre moment, they are stored as one or more KinematicBlock objects in the KinematicData object. In some cases the data are given in a form that needs further processing to convert it to double-differential form, and for these cases, special classes are given in the KinematicData object, as outlined below. Y12 first converts these special objects into double-differential format and stores them in a KinematicBlock object. All KinematicBlock data of the same type are then added together. This is done by creating an initial union grid of all incident energies on which each value of the distributions to be summed is interpolated and then added together. Additional points are added by using the halving scheme. If exit energy and angle distribution are given separately, they are combined into one KinematicBlock object. As a final step, the kinematic data are multiplied by the yield data.

Before returning, Y12 converts any data given in the center-of-mass system into the laboratory system and converts to Legendre, cosine, or tabulated depending on the user’s selection.

6.3.1.1 Energy distributions

If angular and exit energy distribution are given separately, the energy distribution can often be given as a function. ENDF currently allows two evaporation spectra, as well as a Maxwellian Fission spectrum, a Watt spectrum, and a Madland-Nix Fission spectrum. A special class takes parameters for these functions as read from the ENDF formatted file. Y12 translates the functions into a KinematicBlock object. The appropriate grids for incident and exit energies are constructed via a halving scheme as described above.

6.3.1.2 Kalbach-Mann formalism

If the angular distribution are given in the Kalbach-Mann formalism [30,31], the distribution for a given incident and exit energy is given in terms of parameters r, a and f_0. The parameter r is a pre-compound fraction given by the evaluator, and the parameter a is a simple parameterized function that depends mostly on the center-of-mass emission energy. The parameter f_0 is the integral over the angular distribution. The parameter a is either given by the evaluator, or it can be calculated. The distribution is always given in the center-of-mass system. The initial Kalbach-Mann parameters are stored in a KinematicBlock object using the zero to second Legendre order to store f_0, r and a. The advantage of using a KinematicBlock object is that existing interpolation routines can be used to interpolate values for f_0, r and a at any desired incident and exit energy. Y12 converts the parameters to a double differential distribution in the center-of-mass system. The incident energy grid is the union grid of the incident energies given by the evaluator for the parameters and for the yield data. Additional energies are added close to the threshold energy.

6.3.1.3 N-Body Phase-space distribution

For N-Body reactions, the phase space distribution is sometimes used in ENDF evaluations. A PhaseSpace object stores the parameters, APSX for the total mass of the NPSX number of particles treated by the distribution. Y12 then calculates double-differential cross section from these parameters and determines a suitable grid.

6.3.1.4 Coherent elastic thermal neutron scattering

For some crystal structures such as graphite, neutrons can scatter without loss of energy at the Bragg edges. For each incident energy, one or more discrete exit angles may exist. File 7 lists the angles and a value proportional to the structure factors as a function of incident energy for different temperatures. These data are encapsulated into a CoherentElastic object which is processed by Y12. The incident energy grid is determined by first generating the 1-D cross section on a grid dense enough to interpolate linearly. The kinematic data are then generated on the same grid. The angles are marked as discrete.

6.3.1.5 Incoherent elastic thermal neutron scattering

Incoherent elastic scattering is described by the following formula:

$$\frac{\sigma_b}{4\pi} e^{-2EW(T)}(1-\mu)\delta(E-E'),$$

(**Error! Bookmark not defined.**14)

where $\sigma_b$is the characteristic bound cross section and W' Debye-Waller integral divided by the atomic mass, both of which are given by the evaluator. The two values are stored in an IncoherentElastic object for processing by Y12. Y12 first calculates the 1-D cross section and determines a grid dense enough to linearly interpolate in the cross section data. The kinematic data are then generated on the same incident energy grid.

6.3.1.6 Incoherent inelastic thermal neutron scattering

The preponderance of ENDF/B thermal scattering data use this data form. For most materials, it is the only form needed, and data are given as a list of S(α,β) data, with $\alpha = \left[E' + E - 2\mu\sqrt{EE'}\right]/A_0kT$ as the momentum transfer and $\beta = (E' - E)/kT$ as the energy transfer. The variable T denotes temperature, k is Boltzman's constant, and A_0 is the mass of the principle scattering atom in the material. A SalphaBeta object is created from the ENDF data for processing in Y12. In addition, the evaluator gives an upper incident energy E_{max} above which the of S(α,β) data are not to be used, and treatment switches to a short collision time approximation, where S(α,β) is given as:

$$S^{SCT}(\alpha,\beta) = \frac{exp\left[-\frac{(\alpha-|\beta|^2)T}{4\alpha T_{eff}} - \frac{|\beta|}{2}\right]}{\sqrt{4\pi\alpha\frac{T_{eff}}{T}}},$$ (**Error! Bookmark not defined.**15)

and the double differential cross section is calculated as:

$$\frac{\sigma'_b}{4\pi kT}\sqrt{\frac{E'}{E}}e^{-\beta/2}S(\alpha,\beta)$$ (**Error! Bookmark not defined.**16)

from the S(α,β) data, where σ'_b and T_{eff} are given in the ENDF evaluation. Y12 uses the S(α,β) given by the evaluator below E_{max}, above which the short collision time approximation is used. For incident energies below E_{max} and exit energies and angles for which the β is outside the range given by the evaluator, the short collision time approximation is used. If α is lower than the range given by the evaluator, the lowest value given by the evaluator is used. If α is higher than the range given by the evaluator, the short collision time approximation is used. For large negative values of β, the formulas for the cross section become numerically unstable due to the exponent. Therefore, for $\beta < -1200$, the double differential cross section value is set to zero. For a given incident energy, the lower range of the exit energy is set to $10^{-9}eV$, and the upper exit energy is set so that the integral over the exit angle is zero. For the exit energies and angles, the halving scheme is used in conjunction with a starting grid. The lower and upper exit energies are then eliminated if the relative difference in the integral of exit energy and angle does not change more than a user specified precision. For the incident energies, first a grid is generated in which energy points are spaced by $E_{i+1} = E_i\frac{ln(10)}{20}$ in the desired energy range. Then a sufficient number of points is added to interpolate linearly in the 1-D cross section calculated as the integral over exit energy and angle. Finally, the incident energy grid is thinned so that cross section can by linearly interpolated and the exit energy range can be determined by unit-based interpolation.

For nuclides that do not have evaluator-provided S(α,β) data, the free gas approximation is used. In the case of CE transport calculations, the transport codes generate the free gas data during transport. For the MG calculations, the free gas data are provided on the library. Y12 uses the same procedure for calculating the free gas kinematic data using the expression

$$\frac{1}{\sqrt{4\pi\alpha}}exp\left\{-\frac{\alpha^2+\beta^2}{4\alpha}\right\}$$ (**Error! Bookmark not**

defined.17)

to calculate a value for $S(\alpha,\beta)$.

6.3.1.7 Incoherent Scattering for incident gammas

The cross section for incoherent scattering of incident gammas is given by

$$\frac{d\sigma(E,E',\mu)}{d\mu} = S(q)\frac{d\sigma_{KN}(E,E',\mu)}{d\mu},$$

(**Error! Bookmark not defined.**18)

where $S(q)$ is the incoherent scattering cross section listed in the ENDF evaluation as a function of the momentum of the recoil electron (in inverse angstrom units):

$$q = \frac{E}{\alpha}\left[1+\left(\frac{\alpha\prime}{\alpha}\right)^2 - 2\mu\left(\frac{\alpha\prime}{\alpha}\right)\right]^{1/2},$$

(**Error! Bookmark not defined.**19)

where

$$\alpha = \frac{E}{m_0c^2}, \alpha' = \frac{E\prime}{m_0c^2},$$

m_0 rest mass of the electron,

c speed of light,

and

$$\frac{d\sigma_{KN}(E,E',\mu)}{d\mu} = \frac{3\sigma_{TH}}{16}\left(\frac{E'}{E}\right)^2\left(\frac{E}{E'}+\frac{E'}{E}+\mu^2-1\right)d\Omega$$

(**Error! Bookmark not defined.**20)

is the Klein-Nishina equation [32] where solid angle can be converted to the cosine via:

$$d\Omega = -2\sin\theta d\Theta = -2d(\cos\theta) = -2d\mu,$$

(**Error! Bookmark not defined.**21)

and $\sigma_{TH} = 0.66524485$ is the classical Thompson cross section for the electron. This formula can be directly used if generating data in tabulated form for use in CE libraries, as the first integral is over the cosine. Therefore, if tabulated data are requested, Y12 calculates the double differential cross section from Eq. (18). However, if calculating cosine moments or Legendre orders, the first integral is over exit energy. Therefore, a Jacobian must be applied to convert to an integral over exit energy. If the photon energy is expressed in electron rest mass units (i.e. $e = E_\gamma/0.5110034$), where E_γ is given in units of MeV, then energy and momentum conservation leads to

$$\frac{e\prime}{e} = \frac{1}{1+e(1-\mu)},$$ (**Error! Bookmark not defined.**22)

from which the Jacobian is derived as:

$$d\mu = \frac{de\prime}{e\prime^2}.$$ (**Error! Bookmark not defined.**23)

If the user desires cosine or Legendre moments, then this transformation is applied to Eq. (18) before the cross section data are converted to cosine moments.

The incident energy grid is determined by the energy grid used for the cross section data as given by the evaluator. The exit energy grid for a given incident energy is determined by first adding 100 equally spaced exit energies in the energetically allowed range. This grid is then refined by a halving scheme and finally thinned of exit energy points not needed to describe the distribution. Since there is only one angle per exit energy, no grid is necessary for the angular distribution.

6.3.1.8 Coherent Scattering for incident gammas

The cross section for incoherent scattering of incident gammas is given by:

$$\frac{d\sigma(E,E',\mu)}{d\mu} = \delta(E-E')\pi r_0^2(1+\mu^2)\{[F(q)+F'(E)]^2 + F''(E)^2\},$$ (**Error! Bookmark not defined.**24)

where:

$q = \alpha[2(1-\mu)]^{1/2}$	is the recoil moment of the atom (in inverse angstroms),
	$\alpha = \frac{E}{m_0 c}$,
m_0	rest mass of the electron,
c	speed of light, and
$r_0 = {e^2}/{m_0 c^2}$	classical radius of the electron.

The remaining quantities are listed in the ENDF.

The incident energy grid is determined by the energy grid used for the cross section data as given by the evaluator. The angular distribution is determined by first adding 50 equally angle cosines over the full range. This grid is then refined by a halving scheme and is finally thinned.

6.3.1.9 Pair production for incident gammas

Since there are no kinematic data for pair production given in ENDF, an isotropic distribution is assumed with fixed exit energy of 511.0034 keV for all incident energies. The range of the incident energies is determined by the range of the cross section data given for pair production.

6.3.1.10 Photon production for incident neutrons

Photon production kinematic data can be given in many of the formats discussed above. These kinematic data are processed as discussed above. In addition, photon multiplicity data can be given as either multiplicities or as transition probability arrays. In both cases it is assumed that the angular dependence and energy dependence are separable. The use of transition arrays is primarily of use in discrete level inelastic scattering. Here the situation is characterized by having a neutron interacting with a nucleus that can be excited to one of many excited quantum states. A nucleus in an excited state will decay to the ground state by emitting photons or other particles, and it can conceptually transfer to all of the energy levels below the initial excited level. When the nucleus transfers between levels, energy must be conserved. Therefore, particles are expelled from the nucleus whose energies must be the difference between the energies of the levels. If a transition array is used, the energy levels of the decay are explicitly given, and Y12 will reconstruct the decay all the way to the ground state. If multiplicity data are given, Y12 assumes that the evaluator already provided the decay to the ground state.

Photon production data in ENDF can be given in units of cross section (absolute) or relative to the incident neutron cross section. Like ENDF, Y12 normally assumes that the kinematic data are given relative to the cross section. Since Y12 does not have access to the cross section data itself, it cannot change the kinematic data from absolute to relative. In order to be consistent with the existing AMPX code base, Y12 uses a modified reaction number to indicate that the kinematic data are given in units of cross section.

6.4 JERGENS

The JERGENS module is used to generate the flux used to collapse point-wise data to MG format. A variety of predefined functions are available; see JERGENS input instructions for more details. JERGENS cannot construct a flux based on existing data, like the total cross section of a given material. If an arbitrary function based on existing point-wise data is desired, a combination of ZEST and FUNCCAL is likely needed.

6.5 PURM AND PURM_UP

Because of the statistical nature of the unresolved resonance parameters, probability tables can be used to provide cross section probability distribution functions for energy ranges at specific temperatures. The module PURM in AMPX uses an approach described in detail in Dunn 2002 [33] which is different from the ladder approach used in NJOY. PURM generates pairs of resonance or levels surrounding the energy of reference given in the ENDF evaluated data files. For each (l,J) value, pairs of resonance are sampled from the Wigner distribution with the level density given for this pair in ENDF. Once the distribution of energy levels is sampled, the resonance width is sampled from a Chi-square distribution for each resonance. The number of degrees of freedom is given in ENDF. PURM uses the Δ_3 statistic test developed by Dyson and Mehta [34,35] to determine the appropriate number of pairs of resonances. Once all the parameters are sampled, the single-level Breit-Wigner formalism is used to calculate the total, fission, capture, and elastic cross section data at all desired temperatures. The process is repeated a sufficient number of times to generate statistical information of the cross section data. Since cross section data cannot be negative, and since an elastic cross section of zero causes the transport codes to abort, PURM rejects any sample that generates negative cross sections or zero elastic cross section values. PURM generates probability tables for each desired energy and temperatures. The cross section boundaries in the table are chosen so that the probability is equiprobable in the total cross section.

As in the resolved range, the ENDF format allows the evaluator to give cross section data in point-wise format in File 3 that are to be combined with the cross section calculated from the unresolved resonance parameters. However, due to the statistical nature, two different formats are allowed, as distinguished by a value of zero or one for a flag called LSSF. If LSSF is zero, the cross section of File 3 is to be added to File 2 to form the total cross section data. If LSSF is one, File 2 unresolved resonance parameters are to be used to compute a slowly varying self-shielding factor that may be applied to the rapidly varying File 3 values. In either case, two different energy grids may be used for File 2 and File 3 data. Since the generation of a probability table for a given energy is a computationally intensive step, PURM only generates probability tables at the energies of reference used for the unresolved resonance data in File 2. However, since the probability tables are generated on equiprobable bins, probability tables at other desired energies can easily be interpolated. A second module PURM_UP is used to generate the probability tables on the union grid of File 2 and 3 data. In PURM_UP, the point-wise cross section from File 3 is retrieved and Doppler broadened to the desired temperatures. If LSSF is one, the probability tables generated in module PURM are interpolated on the union grid and multiplied by the Doppler-broadened File 3 cross section value. If LSSF is zero, the probability tables without the File 3 contribution are interpolated on the union grid, and the Doppler-broadened File 3 cross section data are added. Since the File 3 data do not contain any statistical variations, the above operations will not change any distributions described by the probability tables except for a constant factor or a constant value.

6.6 PRUDE

The module PRUDE is used to calculate the temperature and background-dependent cross section data in the unresolved resonance range based on methods developed by R. N. Hwang at ANL [36] as outlined in Sect. 2.2.4. As in the case of PURM, the cross section data given in File 2 are either added or multiplied with the temperature- and background-dependent cross section data.

6.7 X10

The module X10 is used to generate the group-wise 1-D data and scattering matrices. The module does not read ENDF/B data files; instead, it takes the following three data types:

1. a file containing the point-wise 1-D data at all desired temperatures,
2. a file containing a smooth weighting function, and
3. a kinematics file generated by Y12 that contains the data in cosine moment form.

X10 does not include physics. The physics associated with a particular type of scattering are all contained in the kinematics file produced by Y12. Therefore, X10 can use the same integration routines whether neutron matrices, gamma matrices, or gamma production matrices are produced. The integration routines used are described in Sect. 5.2. In addition to the 1-D cross section data and kinematic data given in the ENDF evaluation, X10 generates several specific reactions for use in the transport codes.

For fissionable materials, X10 will produce several specialized scattering matrices as listed in Sect. 7.3.2.

6.8 FABULOUS

The module FABULOUS computes data needed for resonance self-shielding of MG cross sections with the Bondarenko method [37]. The Bondarenko method represents self-shielded cross sections in terms of temperature and a background cross section parameter σ0 which indicates the degree of self-shielding:

$$\sigma_{x,g}^{(j)}(\sigma_0,T)=\frac{\int_g \sigma_x^{(j)}(E,T)\,\varphi(E,\sigma_0,T)\,\mathrm{dE}}{\int_g \varphi(E,\sigma_0,T)\,\mathrm{dE}}, \qquad (25)$$

where $\sigma_{x,g}^{(j)}(\sigma_0,T)$ is the self-shielded MG cross section in group g for reaction type x and nuclide j, corresponding to background cross section σ0 and temperature T. The Doppler broadened CE cross section data $\sigma_x^{(j)}(E,T)$ appearing in Eq. (25) is processed as described in previous sections, and the weighting function $\varphi(E,\sigma_0,T)$ approximates the fine-structure flux spectrum for a mixture containing resonance nuclide j at varying dilutions as defined by the value of σ0. Bondarenko shielding factors, also known as f-factors, are computed from the expression

$$f_{x,g}^{(j)}(\sigma_0,T)=\frac{\sigma_{x,g}^{(j)}(\sigma_0,T)}{\sigma_{x,g}^{(j)}(\varphi_{ref},T_{ref})},$$

(**Error! Bookmark not defined.**26)

where the denominator corresponds to an arbitrary reference cross section evaluated with a specified reference flux energy spectrum and temperature. Often the reference spectrum is characteristic of a fission source in a moderating medium containing an infinitely dilute concentration of the resonance material. However, rather than a generic infinitely dilute spectrum, it may be advantageous to use a more tailored reference spectrum for some applications.

To use the library Bondarenko data to obtain shielded cross sections for transport calculations, the value of σ0 is computed for the system (e.g., lattice) of interest, and the corresponding factor f(σ0 ,T) is interpolated from tabulated values on the library. The shielded cross section is obtained by multiplying the shielding factor by the reference (i.e., infinitely dilute) cross section.

The salient feature of the Bondarenko method is that the flux weighting spectrum is parameterized by a function that depends only on the background cross section and temperature, which allows Eqs. (25) and (26) to be evaluated for a specified set of background cross sections and temperatures that span the range of self-shielding conditions. For example, the background cross sections used to produce the ^{238}U f-factors for ENDF/B-VII libraries in the SCALE system are

$$\sigma_0=\left\{0.01, 1.0, 10.0, 50.0, 100.0, 1000.0, 10000.0, 10^6, 10^{10}\right\}.$$

(**Error! Bookmark not defined.**27)

Group-dependent f-factors vs. background cross sections and temperature, as well as the reference cross section values, are included for each nuclide in the AMPX master library.

AMPX provides three methods to parameterize the flux for processing Bondarenko data: (1) analytical approximation, (2) numerical solution of the slowing-down equation for an infinite homogeneous medium, and (3) numerical solution for a heterogeneous unit cell in an infinite lattice. It is usually desirable to use the analytical representation for the unresolved resonance range and higher energies, while the numerical solutions are usually more accurate for the resolved resonance and thermal ranges. Numerical solutions obtained with the homogeneous model are adequate for most resonance nuclides, but the heterogeneous model may produce more accurate results for important nuclides in the fuel region of a reactor lattice. Only the analytical method is provided in the AMPX distribution. The full AMPX/SCALE distribution, along with the SCALE data directory, is needed for the remaining two methods.

The AMPX module FABULOUS computes Bondarenko factors with an analytical expression for flux based on the narrow resonance (NR) and intermediate resonance (IR) approximations:

$$\varphi(E,\sigma_0,T) = \frac{\sigma_0 \varphi_{ref}(E)}{\sigma_t^{(j)}(E,T)+\sigma_0}$$

(**Error! Bookmark not defined.**28)

In this case Eq. (25) reduces to

$$\sigma_{x,g}^{(j)}(\sigma_0,T) = \frac{\int_g \frac{\sigma_x^{(j)}(E,T)\,\varphi_{ref}(E)}{\sigma_t^{(j)}(E,T)+\sigma_0}\mathrm{dE}}{\int_g \frac{\varphi_{ref}(E)}{\sigma_t^{(j)}(E,T)+\sigma_0}\mathrm{dE}}.$$

(**Error! Bookmark not defined.**29)

It can be seen that the limit for $\sigma_{x,g}^{(j)}(\sigma_0 \to \infty, T_{ref})$ is the value of the reference cross section weighted with φ_{ref}.

The integrals in Eq. (29) are solved using the Runge-Kutta method with adaptive step size [27].

As outlined above, there are two methods available in AMPX to calculate the cross section data in the unresolved resonance range. If the module PRUDE is used to generate temperature and background cross section dependent cross section, Eq. (29) is used. However, in the unresolved resonance range, values for $\sigma_x^{(j)}(E,T)$ is substituted by $\sigma_x^{(j,urr)}(E,T,\sigma_o)$ as calculated by PRUDE.

If the probability table is used, $\sigma_{x,p}^{(j)}(E,T)$ is the cross section for the band p of the probability table for reaction x and nuclide j. The probability for a given band is given by P_p. In this case, FABULOUS calculates $\sigma_{x,g}^{(j)}(\sigma_0,T)$ as

$$\sigma_{x,g}^{(j)}(\sigma_0,T) = \frac{\int_g \sum_p P_p \frac{\sigma_{x,p}^{(j)}(E,T)\varphi_{ref}}{\sigma_{t,p}^{(j)}(E,T)+\sigma_0} dE}{\int_g \sum_p P_p \frac{\varphi_{ref}}{\sigma_{t,p}^{(j)}(E,T)+\sigma_0} dE}.$$

.(**Error! Bookmark not defined.**30)

If the unresolved resonance range starts or ends within a group, the numerator and denominator each will have two terms, one containing the sum over the probability table bands, and one using the integral with the point-wise cross section data.

The AMPX module IRFfactor obtains a parameterized CE flux spectrum using numerical solutions computed with the CENTRM/PMC deterministic transport modules in SCALE [12]. These SCALE modules CENTRM and PMC have application program interfaces (APIs) which are called directly from IRFfactor. The PMC module evaluates Eq. (25) using a weight function calculated by CENTRM, a deterministic transport solver that uses CE data. CENTRM determines a problem-specific energy mesh based on cross section variations for materials in the problem, resulting in an energy grid of ~60,000–100,000 points for the flux calculation. Either homogeneous or simple heterogeneous models can be used.

The point-wise flux (PW) for the homogeneous model is computed from the slowing-down equation for an infinite-medium mixture containing resonance nuclide *j* mixed with hydrogen:

$$\left(\sigma_t^{(j)}(E,T)+\sigma_0\right)\varphi(\mathrm{E},\sigma_0,T)=\int_E^{E/\alpha^{(j)}}\frac{\sigma_s^{(j)}(E',\mathrm{T})\,\varphi(E',\sigma_0,\mathrm{T})}{(1-\alpha^{(j)})E'}dE'+\sigma_0\int_E^{\infty}\frac{\varphi(E',\sigma_0\,\mathrm{T})}{E'}dE'$$

(**Error! Bookmark not defined.Error! Bookmark not defined.**31)

where

$1-\alpha^{(j)}$ = the maximum fractional energy loss in an elastic collision with nuclide *j*,

$$\sigma_0=\frac{N^{(H)}}{N^{(j)}}\sigma_p^{(H)} \quad \text{(the background cross section)},$$

$\sigma_p^{(H)}$ = the potential cross section for hydrogen,

(**Error! Bookmark not defined.Error! Bookmark not defined.**32)

and $N^{(H)}$, $N^{(j)}$ are number densities of hydrogen and nuclide *j*, respectively.

Note that the solution to Eq. (31) is parameterized in terms of σ_0 and T as required by the Bondarenko method. IRFfactor uses CENTRM to numerically solve Eq. (31) for a given resonance material, with the specified sets of temperatures and background cross sections. The ratio $N^{(H)}/\,N^{(j)}$ in the CENTRM homogeneous medium calculation is adjusted to give the desired background cross section using Eq. (32). IRFfactor executes the PMC module to compute self-shielded MG cross sections by evaluating Eq. (25) with the calculated CE flux at the corresponding (σ_0,T). The module PMC also updates the elastic scattering matrix, and the self-shielded removal cross section data are the diagonal elements of the elastic scattering matrix for Legendre order zero. The shielded cross sections are converted into f-factors for the library.

IRFfactor can also be used to compute heterogeneous shielding factors for resonance nuclide *j* contained in a uniform lattice consisting of cylindrical fuel pins, surrounded by cladding, and contained in a moderating material. In this case, CENTRM calculates the space-dependent flux in a unit cell using either the discrete ordinates method for a 1-D Wigner-Seitz model or the method of characteristics for 2-D unit cell model (i.e., cylindrical fuel pin in a rectangular moderator region). Although CENTRM can treat an arbitrary mixture of resonance absorbers, IRFfactor by default neglects resonance interference effects and represents the total cross sections by the potential scattering for all nuclides other than *j*. This approximation allows Bondarenko data for each resonance material to be processed independently. For self-shielding with the heterogeneous model, the following transport equation is solved to obtain the CE angular flux spectrum for a given heterogeneous cell configuration:

$$\left(\nabla\cdot\Omega+N^{(j)}(r)\sigma_t^{(j)}(r,E,T)+\sum_{m\neq j}N^{(m)}(r)\sigma_p^{(m)}\right)\Psi(r,E,\Omega;T)=$$
$$N^{(j)}\int_E^{E/\alpha^{(j)}}\frac{\sigma_s^{(j)}(E',T)\,\varphi(r,E',T)}{(1-\alpha^{(j)})E'}dE'+\sum_{m\neq j}\left\{N^{(m)}(r)\sigma_p^{(m)}\int_E^{E/\alpha^{(m)}}\frac{\varphi(r,E',T)}{(1-\alpha^{(m)})E'}dE'\right\}.$$

(**Error! Bookmark not defined.Error! Bookmark not defined.**33)

The zone-averaged scalar flux $\varphi^{(Z)}(E)$, which is obtained by integrating the CE angular flux over all directions, is used as the weighting function to compute shielded MG cross sections. Cell parameters such as fuel radius, pin pitch, moderator void fraction, and material concentrations (i.e., $N^{(j)}$ and $N^{(m)}$) are varied to obtain a range of heterogeneous self-shielding conditions for solving Eq. (33). The self-shielded cross sections are assigned to the background cross section for the corresponding heterogeneous cell configuration. Two options are provided to define the background cross section of the heterogeneous cell: (1) the embedded self-shielding method (ESSM) described in [38], and (2) conventional equivalence theory.

The ESSM calculates the background XS in group *g* for resonance nuclide *j* in zone *Z* from the expression [38]:

$$\sigma_{0,g}^{(j)}=\frac{\sigma_{a,g}^{(j)}\,\varphi_g^{(Z)}}{1-\varphi_g^{(Z)}}$$

(**Error! Bookmark not defined.Error! Bookmark not defined.**34)

In Eq. (34), the variable $\sigma_{a,g}^{(j)}$ is the self-shielded group cross section weighted with the CE flux in zone *Z*, and $\varphi_g^{(Z)}$ is the MG flux for zone *Z*, which is calculated by solving the following one-group slowing-down equation for the same heterogeneous cell:

$$\left(\nabla\cdot\Omega+N^{(j)}(r)\sigma_{a,g}^{(j)}(r)+\sum_m N^{(m)}(r)\sigma_p^{(m)}\right)\Psi_g(r,\Omega)=\sum_m N^{(m)}(r)\,\lambda^{(m)}\sigma_p^{(m)},$$

(**Error! Bookmark not defined.Error! Bookmark not defined.**35)

where $\lambda^{(m)}$ is the IR parameter discussed the next section.

The above one-group fixed source transport equation is solved independently for each group using CENTRM. In order to obtain shielded cross sections with ESSM for a reactor lattice mixture, a similar fixed source is solved (perhaps in 2-D or 3-D geometry); Eq. (34) is evaluated to obtain the background cross section, and then the shielded cross section is interpolated from the tabulated values on the library. The ESSM approach is used in the POLARIS [39] and MPACT [40] lattice physics codes

The other option for defining the background cross section of a heterogeneous cell is the conventional method based on the IR approximation and equivalence theory [41], as implemented in the BONAMI module of SCALE:

$$\sigma_0 = \frac{\sum_{m \neq j} \lambda^{(m)} \sigma_p^{(\mathrm{m})} N^{(\mathrm{m})} + \Sigma_{esc}}{N^{(j)}}$$ (**Error! Bookmark not defined.Error! Bookmark not defined.**36)

where $\lambda^{(m)}$ is the IR parameter (discussed below) for nuclide *m*, and Σ_{esc} is the "escape" cross section. In IRFfactor the escape cross section for the heterogeneous cell is computed from the expression

$$\Sigma_{esc} = \frac{A(1-c)}{\overline{\ell}},$$ (**Error! Bookmark not defined.Error! Bookmark not defined.**37)

where c = Dancoff factor for the fuel, $\overline{\ell}$ = average chord length in the fuel zone, and A= Bell correction factor. The techniques used to evaluate these parameters are discussed in the BONAMI section of SCALE. In addition to BONAMI, several other codes employ Eq. (36) for Bondarenko self-shielding calculations, although the expressions for the Dancoff and Bell factors may differ somewhat.

After all cell calculations are completed, shielded cross sections corresponding to the desired set of σ_0 values on the library are obtained by interpolation from the shielded data computed for the various cell background cross sections and then are converted to f-factors.

The modules FABULOUS and IRFFACTOR can also calculate the f-factors for the removal cross section, which is a measure of elastic scattering staying in the group. The removal cross section is defined as

$$\frac{1}{\int_g \varphi(E,T,\sigma_0) dE} \int_g dE \sigma_2^{(j)}(E,T) \varphi(E,T,\sigma_0) \int_{E'=E_g}^{E} f_0\left(E, E'\right) dE',$$ (**Error! Bookmark not defined.**38)

where the definitions are as given in Eq. (12), and E_g is the lower group boundary. Outside the thermal range, elastic scattering does not scatter to exit energies larger than the incident energy; therefore, the inner integral can be extended to the upper energy bound. This makes the removal cross section in the infinite diluted limit the same as the diagonal elements of the scattering matrix for the elastic cross section at Legendre order zero.

Using the narrow resonance flux defined in Eqs. (26) and (25), the narrow resonance removal cross section can be calculated as a function of temperature and background value:

$$\frac{1}{\int_g \frac{\varphi_{ref(E)}}{\sigma_t^{(j)}(E,T)+\sigma_0} dE} \int_g \frac{\sigma_2^{(j)}(E,T)\varphi_{ref(E)}}{\sigma_t^{(j)}(E,T)+\sigma_0} dE \int_{E'=E_g}^{E} f_0\left(E, E'\right) dE'.$$ (**Error! Bookmark not defined.**39)

As before, two options are available to calculate the cross section data for the elastic or total cross section in the unresolved resonance range. The first option is to use $\sigma_x^{(j,urr)}(E,T,\sigma_o)$ as calculated by PRUDE, and the second is to use the probability table data, in which case Eq. (39) in the unresolved resonance range becomes

$$\frac{1}{\int_g \sum_p P_p \frac{\varphi_{ref(E)}}{\sigma_{t,p}^{(j)}(E,T)+\sigma_0} dE} \int_g \sum_p P_p \frac{\sigma_{2,p}^{(j)}(E,T)\varphi_{ref(E)}}{\sigma_{t,p}^{(j)}(E,T)+\sigma_0} dE \int_{E'_1=E_g}^{E} f_0\left(E, E'\right) dE'.$$ **(Error! Bookmark not defined.**40)

The f-factors for the removal cross section are calculated as a ratio between Eq. (39) and the diagonal elements of the scattering matrix for the elastic cross section at Legendre order zero. In the case of IRFACTOR, the module PMC updates the elastic scattering matrix, and the removal cross section as a function of background cross section and temperature is retrieved as the diagonal element of the updated elastic scattering matrix.

6.9 LAMBDA

The AMPX module LAMBDA computes group-dependent values for the λ factors used in the IR approximation, which is widely used in Bondarenko self-shielding codes. The λ parameters also are used in the AMPX IRFfactor module to determine background cross sections for the heterogeneous cells used in computing the shielding factors, as shown in see Eqs. (35) and (36).

The IR approximation represents the moderating properties of a scatterer nuclide by the parameter λ, which varies over the range of 0.0 to 1.0. The lower limit value of λ=0.0 corresponds to a wide resonance (WR) scatterer, and the upper limit of λ=1.0 corresponds to a narrow resonance (NR) scatterer. A neutron loses a negligible amount in a collision with a WR nuclide compared to the resonance energy widths of a specified reference absorber material usually chosen to be ^{238}U. Conversely, the resonance energy widths of the reference absorber are negligibly small compared to the neutron energy lost in a collision with an NR scatterer. Only an infinite mass nuclide strictly satisfies the requirements for a WR scatterer, while hydrogen closely approximates an ideal NR scatterer in the epithermal energy range. The scattering properties of all other materials are intermediate between these two limits, and they correspond to fractional values of λ. Since the amount of energy lost per collision—as well as resonance widths of the reference absorber—depends on the neutron energy, the IR parameters in general are functions of energy group.

The AMPX module LAMBDA uses the hydrogen-equivalent method [41] to determine IR parameters for a specified scatterer nuclide and reference absorber, assumed here to be ^{238}U. First, a reference set of self-shielded cross sections is computed for several infinite media mixtures of ^{238}U and hydrogen, in which the hydrogen-to-^{238}U ratio is varied to span the range from minimal to infinite dilution. The flux spectra used to weight the shielded ^{238}U cross sections are calculated with the CENTRM transport code which uses point-wise cross sections ^{238}U and hydrogen. Next, some of the hydrogen atoms in the media are replaced by a corresponding number of atoms of a given scatterer nuclide *j*. The CENTRM calculations are performed again for the media, which now consists of ^{238}U, hydrogen, and scatterer *j*. For this application, the CENTRM calculations are performed using only the potential cross section for *j*. The shielded cross section for the mixture will be different than the original mixture containing only hydrogen because the slowing-down properties of *j* are different. However instead of one-to-one replacement, the number of *j* atoms can modified to obtain the same shielded cross section as obtained with hydrogen. This is called the hydrogen equivalent number density of *j*, and from this value, the corresponding lambda value for nuclide *j* can be found. In practice, the hydrogen-equivalent value is obtained by matching the self-shielded cross section with the corresponding value for the pure hydrogen moderated case. The process is repeated for each nuclide and energy group to obtain group-dependent lambdas. This procedure is completely automated in the module.

6.10 SIMONIZE

The module SIMONIZE is used to combine partial master libraries generated by X10 and FABULOUS into a cohesive AMPX MG master library for a given evaluation. The module combines the data into one AMPX MG master library after recalculating and renormalizing the data as follows:

1. 1-D cross section values below a user-specified value are set to 0. This includes negative cross section values.

2. For thermal moderators like ^{1}H in H_2O, the elastic cross section (MT=2) is set to the 1-D collapse of the thermal scattering matrices in the thermal range. The first thermal group defines the extent of the thermal range.

3. If the evaluation uses the free-gas approximation for the thermal scattering matrices, the thermal scattering matrices are renormalized such that when collapsed they result in the elastic cross section as calculated from the ENDF information.

4. The elastic scattering matrix (MT=2) is set to zero in the thermal range, where the first thermal group again defines the thermal range. The assumption is that the free gas scattering matrix should be used in this range.

5. An AMPX master library should only contain up-scatter in the thermal range. Thus, scattering matrices are corrected to not contain any up-scatter terms outside the thermal range. Any up-scatter terms are added to the closest diagonal matrix element.

6. All redundant 1-D cross section values are recalculated from the partial cross section values.

7. If the library contains gamma production information, the module checks whether all production matrices are given relative to the cross section. If not, the relevant scattering matrices are renormalized to be relative to the cross section.

8. If a scattering matrix has a (P_l,l>0) term that is non-zero and the P_0 term is zero, then the (P_l,l>0) is also set to zero.

6.11 JAMAICAN

The modules generating kinematics data generate the double differential point-wise data. The CE transport codes require the data in the form of marginal probabilities with respect to angle, and conditional probability with respect to exit energy. The module JAMAICAN performs the conversion from double differential point-wise to marginal, and conditional probabilities and can also eliminate redundant points from the distribution and processes discrete gamma lines if processing gamma production data.

In the case of elastic or discrete inelastic reactions (MT=2, 51-90) for incident neutrons, the module uses the same number of angles and exit energies as given in the double differential point-wise data. The necessary integrations are performed using standard AMPX integration routines, and the distribution is normalized to one. In some cases (for example ^{1}H, which has a mass ratio with respect to the incident neutron of less than 1.0), elastic or discrete inelastic reactions can scatter to the same lab angle for different exit energies. In these cases, the marginal probability for the angle lists the same angle twice with different cumulative probabilities. The integration routine used in JAMAICAN steps through the point-wise distribution, inverting integration boundaries where necessary, instead of using driver routines, thus ensuring correct integration in the double-valued region.

In some cases, like coherent elastic scattering for thermal moderators, the double-differential distribution can be discrete in angle. Scattering only occurs in specific directions determined by Bragg edges in the crystal structure. JAMAICAN produces a special distribution for the marginal angle probabilities: for each discrete angle, two angles are added to the distribution: one at the Bragg angle with the correct probability, and the other offset with a probability of zero. This causes the CE transport codes to pick only the discrete angles.

In all other cases, the angle distribution is assumed to be continuous. In most cases, equiprobable angle bins are used for the marginal angle distribution, where the number of desired bins is given by the user. A union grid of all angles for a given incident energy and all exit energies is first created, and the bin boundaries are determined from the cumulative probability distribution on that grid. Unless not requested by the user, an attempt is made to thin the union angle grid while preserving all values of the double differential distribution. If the union grid can be thinned to use a number of angles smaller than the requested number of equiprobable angles, this smaller number is used, and the angle distribution is tagged as not equiprobable. Subsequently, the conditional energy distribution for each selected angle is determined by interpolation from the full double differential function. Unless not requested by the user, an attempt is made to thin the conditional energy distribution if the conditional probability can be determined by linear interpolation.

6.12 PLATINUM

The module PLATINUM is used to create a cohesive CE library for a given evaluation from all the partial parts generated by other modules. However, in contrast to SIMONIZE, which performs a similar task for AMPX MG master libraries, the redundant cross section data are not recalculated, and the elastic cross section is not changed in the thermal range for thermal moderators. These tasks should be performed by other modules before passing the data to PLATINUM. The module unionizes the 1-D cross section data as needed and calculates the collision probabilities. Depending on the availability of cross section data, the following collision probabilities, which are typically needed by Monte Carlo transport codes, are calculated:

MT	**Definition**
2006	$\frac{\sigma_{total} - \sigma_{absorption}}{\sigma_{total}}$
2018	$\frac{\sigma_{nubar} \times \sigma_{fission}}{\sigma_{total}}$
2016	$\frac{\sigma_{n2n}}{\sigma_{total}}$
2017	$\frac{\sigma_{n3n}}{\sigma_{total}}$

6.13 PUFF-IV

The module PUFF-IV is used to generate covariance matrices with respect to the group-averaged cross section data. A detailed description is available in a separate PUFF-IV manual [42].

ENDF contains covariance information for the point-wise data in Files 31 and 33. Covariance information for resonance parameters in the resolved and unresolved range is given in File 32. Covariance information for exit energy probability distributions is given in File 35, and data on exit angle are in File 34.

A covariance matrix is given with respect to two parameters:

$$COV(x,y) = \langle \delta x \delta y \rangle,$$ (**Error! Bookmark not defined.Error! Bookmark not defined.**41)

where

$$\delta x = x - \langle x \rangle,$$ (**Error! Bookmark not defined.Error! Bookmark not defined.**42)

where $\langle x \rangle$ is the expected value of parameter x and δx is the deviation from the expected value from the true value. The standard deviation or uncertainty of the parameter x is then defined as

$$s(x) = \sqrt{\langle \delta x \delta x \rangle}.$$ (**Error! Bookmark not defined.**43)

In addition the correlation between two parameters is defined as

$$\rho(x,y) = \frac{COV(x,y)}{s(x)s(y)}.$$ (**Error! Bookmark not defined.Error! Bookmark not defined.**44)

All the above equations refer to absolute quantities. However, in the libraries produced by PUFF-IV, relative quantities are used in which the covariance and the standard deviation are divided by the expected values.

Assuming that a derived quantity $f(x_1, x_2, x_3, \dots, x_n)$ is to be calculated and covariance information for the parameters is available, then the standard deviation for f will be

$$s(f) = \sqrt{\sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i} \delta x_i \right)^2},$$ (**Error! Bookmark not defined.Error! Bookmark not**

defined.45)

and the covariance matrix for two different values of the function f would be

$$COV(f_1, f_2) = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial f_1}{\partial x_i} \langle \delta x_i \delta x_j \rangle \frac{\partial f_2}{\partial x_j}.$$

(**Error! Bookmark not defined.Error! Bookmark not defined.**46)

The above formulas are derived via the definition of the covariance in the case of a linear function with respect to the individual parameters. The desired function, which is not likely to be linear in the parameters, is converted to a linear function by using a first order Taylor expansion. Higher order effects are neglected in the processing of the covariance information and the propagation of uncertainties. For continuous function, the sums above are substituted by integrals.

6.13.1 Point-wise covariance for cross section data

In Files 31 and 33, covariance matrices for cross section data with respect to the incident energy are given. Some covariance matrices are given directly, and some are given as derived matrices. If given as direct covariance matrices, the covariance matrix is given on a fairly broad bin structure for the incident energy. For a given bin, the covariance value is assumed to be constant. Covariance matrices are given for incident energy, material, and reaction. ENDF contains partial covariance matrices:

$$COV\left(\sigma^{react_1}(E_1), \sigma^{react_2}(E_2)\right) = \langle \delta\sigma^{react_1}(E_1), \delta\sigma^{react_2}(E_2) \rangle,$$

(**Error! Bookmark not defined.Error! Bookmark not defined.**47)

where $react_1$ describes one material and reaction, and $react_2$ describes another material and reaction. Since the matrices are partial covariance matrices, they do not need to be symmetric, and the bin structures for the two incident energies can differ. PUFF-IV processes these point-wise covariance matrices into covariance matrices with respect to group-averaged cross section values, which are defined by eq. (11). Define

$$\begin{gathered} \phi_I = \int_{E \in I} \varphi dE \\ x_I^{react_1} = \frac{1}{\phi_I} \int_{E \in I} \varphi \sigma^{react_1}(E) dE \end{gathered}.$$

(**Error! Bookmark not defined.Error! Bookmark not defined.**48)

Note that

$$\begin{aligned} \frac{\partial(x_I)}{\partial\sigma(E_1)} &= \frac{1}{\phi_I}\left(\varphi(E_1^2)\sigma(E_1^2)\frac{\partial E_1^2}{\partial E_1} - \varphi(E_1^1)\sigma(E_1^1)\frac{\partial E_1^1}{\partial E_1} + \int_{E_1^1}^{E_1^2} \varphi \frac{\partial\sigma(E_1)}{\partial\sigma(E_1)} dE\right), \\ &= \frac{1}{\phi_I}\int_{E_1^1}^{E_1^2} \varphi\delta(E - E_1)\, dE = \frac{1}{\phi_I}\varphi(E) \end{aligned}$$

(**Error! Bookmark not defined.Error! Bookmark not defined.**49)

where $[E_1^1, E_1^2]$ is the energy range for group I. Then the group-averaged covariance matrix becomes

$$COV\left(x_I^{react_1}, x_J^{react_2}\right) = \int_{E_1 \in I} dE_1 \int_{E_2 \in J} dE_2 \frac{\partial\left(x_I^{react_1}\right)}{\partial\sigma^{react_1}(E_1)} \langle\delta\sigma^{react_1}(E_1), \delta\sigma^{react_2}(E_2)\rangle \frac{\partial\left(x_J^{react_2}\right)}{\partial\sigma^{react_2}(E_2)} = \frac{1}{\phi_I \phi_J} \int_{E_1 \in I} dE_1 \int_{E_2 \in J} dE_2\, \varphi(E_1)\langle\delta\sigma^{react_1}(E_1), \delta\sigma^{react_2}(E_2)\rangle\varphi(E_2).$$

(Error! Bookmark not defined.Error! Bookmark not defined.50)

The covariance matrix $\langle\delta\sigma^{react_1}(E_1), \delta\sigma^{react_2}(E_2)\rangle$ is a point-wise matrix. However it is given as a constant over evaluator defined incident energy bins. For ease of calculation, a super grid is generated which contains the user-defined group structure and every evaluator-defined bin boundary for all explicitly given matrices. Since the evaluator-supplied covariance is point-wise, the value is the same in any subgroup of an evaluator defined group. Also, since all energy boundaries are included, partial group values will not occur. Equally, $x_I^{react_1}$ and $\phi_I$can be divided into these subgroups. The covariance matrix then becomes

$$\frac{1}{\phi_I^s \phi_J^s} \sum_{E_1^s \in I} \sum_{E_2^s \in J} COV\left(\sigma^{react_1}\left(E_1^S\right), \sigma^{react_2}\left(E_2^S\right)\right) \phi_I^s \phi_J^s,$$

(Error! Bookmark not defined.Error! Bookmark not defined.51)

where E_1^s denotes the energy groups on the super grid. If the point-wise cross section covariance is given as a relative covariance, the equation on the super-grid becomes:

$$\frac{1}{\phi_I^s \phi_J^s} \int_{E_1 \in I^s} dE_1 \int_{E_2 \in J^s} dE_2\, \varphi(E_1)\varphi(E_2)\sigma^{react_1}\sigma^{react_2} COV\left(\sigma^{react_1}(E_1), \sigma^{react_2}(E_2)\right) = \sum_{E_1^s \in I} \sum_{E_2^s \in J} COV\left(\sigma^{react_1}\left(E_1^S\right), \sigma^{react_2}\left(E_2^S\right)\right) x_I^{react_1} x_J^{react_2}.$$

In order to collapse back to the user-defined group structure, it is noted that

$$x_I = \frac{1}{\phi_I} \sum_{I_s \in I} \phi_I^s\, x_I^s.$$

(Error! Bookmark not defined.Error! Bookmark not defined.52)

Therefore, the covariance matrix between two broad groups is:

$$\langle\delta x_I, \delta x_J\rangle = \frac{1}{\phi_I \Phi_J} \sum_{I_s \in I} \phi_I^s \sum_{J_s \in J} \phi_J^s \frac{\partial x_I^s}{\partial x_I^s} \langle x_I^s, x_J^s\rangle \frac{\partial x_J^s}{\partial x_J^s} = \frac{1}{\phi_I \Phi_J} \sum_{I_s \in I} \phi_I^s \sum_{J_s \in J} \phi_J^s \langle x_I^s, x_J^s\rangle$$

(Error! Bookmark not defined.Error! Bookmark not defined.53)

In addition to direct covariance matrices with respect to point-wise cross section data, ENDF includes derived matrices. For the first case, the evaluator simply defines that a given cross section is defined as the sum over several other reactions. Not only does this define the covariance matrix for the given reaction, the reaction is now also correlated with all the partial reactions that make up the defined reaction. PUFF_IV automatically adds these extra cross correlation matrices. Another option open to the evaluator is to define the cross section as being given as a ratio with respect to the cross section of another material and reaction. The ratio is assumed to have no uncertainty, and a covariance matrix for the dependent cross section can therefore easily be calculated.

All the choices open to the evaluator—i.e. explicit or derived covariance matrices—can be used over defined energy ranges. If a matrix is given for a partial energy range, all matrix elements are set outside the range to zero. The final matrix can then simply be formed by adding the various partial energy range matrices together. This relies on the fact that ENDF in these cases does not give a correlation between the matrices for the partial energy ranges.

The group-averaged cross section on the super grid that is needed to calculate the group-averaged covariance matrices is calculated from the point-wise cross section and flux data. PUFF_IV also has the capability to use cross section data from an MG library. In this case, the cross section data for any subgroup of a user-defined MG group is simply assumed to be identical to the cross section of the large group. This ensures that the latter collapse from fine group to broad group results on the original MG cross section value. The flux is treated similarly if given on user-defined groups.

6.13.2 Resolved resonance covariance matrix

In the resolved range, the covariance matrices are given for all resonance parameters. In order to propagate the covariance information to the point-wise cross section, partial derivatives of the cross section with respect to a given resonance parameter are needed. PUFF_IV uses SAMRML, a part of SAMMY [43], to calculate these derivatives. The SAMRML library, which is part of SAMMY, uses analytical formulas to calculate the derivatives for the full R-matrix Reich-Moore formalism. If the resonance parameters in ENDF are given for single- or multi-level Breit-Wigner formalism, they are converted to the full R-matrix Reich-Moore formalism prior to derivatives being calculated. The covariance matrix with respect to the point-wise cross section thus becomes

$$\langle \sigma^{react_1}(E_1), \sigma^{react_2}(E_2) \rangle = \sum_{kn} \frac{\partial \sigma^{react_1}}{\partial P_k} \langle P_k, P_n \rangle \frac{\partial \sigma^{react_2}}{\partial P_n},$$

(**Error! Bookmark not defined.Error! Bookmark not defined.**54)

where the sum is over all resonance parameters P_n. The formula for the point-wise covariance matrix with respect to the cross section can be inserted into Eq. (50) to yield

$$COV\left(x_I^{react_1}, x_J^{react_2}\right) = \frac{1}{\phi_I \phi_J} \int_{E_1 \in I} dE_1 \int_{E_2 \in J} dE_2\, \varphi(E_1) \sum_{kn} \frac{\partial \sigma^{react_1}}{\partial P_k} \langle P_k, P_n \rangle \frac{\partial \sigma^{react_2}}{\partial P_n} \varphi(E_2).$$

(**Error! Bookmark not defined.Error! Bookmark not defined.**55)

If the user defines

$$D_{Ik}^{react_1} = \int_{E_1 \in I} dE_1 \varphi(E_1) \frac{\partial \sigma^{react_1}}{\partial P_k},$$

(**Error! Bookmark not defined.Error! Bookmark not defined.**56)

then the formula becomes

$$COV\left(x_I^{react_1}, x_J^{react_2}\right) = \sum_{kn} D_{Ik}^{react_1} \langle P_k, P_n \rangle D_{Jn}^{react_2}.$$

(**Error! Bookmark not defined.Error!**

Bookmark not defined.57)

The integral in Eq. (56) is solved using a fourth-order Runge-Kutta method with adaptive step size [27]. Since the number of resonance parameters can be large, solving Eq. (57) can take a long time. Therefore, PUFF_IV can be compiled with the BLAS library [44], which has functions for the type of matrix operation needed in Eq. (57) to speed up the process considerably.

6.13.3 Unresolved resonance covariance matrix

The covariance data given in File 32 for the unresolved range are independent of energy in the range they are given. This is the case, even if the unresolved resonance parameters given in File 2 are energy dependent. Therefore, the resonance parameters given in File 32 are used to calculate the covariance matrix with respect to the cross section instead of the parameters used to calculate the point-wise cross section data for the library. The parameters for the unresolved resonance range are given as an average value and a distribution function for those values, which is a χ^2 distribution with an evaluator-defined number of degrees of freedom. The formula for the average cross section at a temperature of 0 K involves a statistical integral, which is calculated using a quadrature integration as outlined in Steen 1969 [45] using the weights and abscissa given in Henryson 1976 [46]. Since all resonance parameters are independent of energy, the derivative of the statistical integral becomes straightforward, and the derived integrant is integrated using the same quadrature weights and abscissa as the original integral. After taking the point-wise derivative, the formulas for calculating the covariance matrix with respect to the group averaged cross section data are identical to the ones used in the resolved resonance range.

6.13.4 Exit energy covariance matrix

The ENDF evaluations can contain covariance matrices for the exit energy. The covariance matrices are independent of incident energy in a given range of incident energies. Covariance matrices may be given for several ranges of incident energies, but correlation between these ranges are not supported in the ENDF formatted files. For a given incident energy range, a covariance matrix with respect to the exit energy is given. The covariance matrix elements are given on an exit energy bin structure. The value given is the covariance matrix on the bin probability and not on the bin-averaged probability function. This is equivalent to the group-averaged covariance on an evaluator defined group structure. Therefore, if the group-averaged one-dimensional χ is defined as

$$\chi_{g'} = \int_{E' \in g'} f(E, E')\, dE',$$

(**Error! Bookmark not defined.Error! Bookmark not defined.**58)

and:

$$\int_0^\infty f(E, E')\, dE' = 1,$$

(**Error! Bookmark not defined.Error! Bookmark not defined.**59)

then the covariance matrix given is:

$$\int_0^\infty f(E,E')\,dE' = 1$$ (**Error! Bookmark not defined.Error! Bookmark not defined.**60)

$$\int_0^\infty f(E,E')\,dE' = 1$$ (**Error! Bookmark not defined.Error! Bookmark not defined.**61)

$$\langle \delta\chi_{g'_I}, \delta\chi_{g'_J} \rangle$$ (**Error! Bookmark not defined.Error! Bookmark not defined.**62)

on an evaluator-defined group structure, which needs to be converted to the user-defined group structure. Since χ is normalized to 1, combining two groups into one group simply implies adding the values for the two groups. Therefore, the same applies for the covariance:

$$\int_0^\infty f(E,E')\,dE' = 1$$ (**Error! Bookmark not defined.Error! Bookmark not defined.**63)

$$\int_0^\infty f(E,E')\,dE' = 1$$ (**Error! Bookmark not defined.Error! Bookmark not defined.**64)

$$\langle \delta\chi_{g'_I}, \delta\chi_{g'_J} \rangle = \sum_{I'\in I, J'\in J} \langle \delta\chi_{g'_{I'}}, \delta\chi_{g'_{J'}} \rangle.$$ (**Error! Bookmark not defined.Error! Bookmark not defined.**65)

If splitting a larger group into several groups, χ scales with the bin width, i.e.

$$\int_0^\infty f(E,E')\,dE' = 1$$ (**Error! Bookmark not defined.Error! Bookmark**

$$\int_0^\infty f(E,E')\,dE' = 1$$ (**Error! Bookmark not defined.Error! Bookmark**

$$\Delta E_I = \sum_{i\in I} \Delta E_i$$
$$\chi_{g'_I} = \sum_{i\in I} \chi_{g'_i} \frac{\Delta E_i}{\Delta E_I},$$ (**Error! Bookmark not defined. Error! Bookmark not defined.**68)

not define d.66)	**not define d.**67)

and the covariance elements scale similarly with the bin width.

7. MISCELLANEOUS USEFUL INFORMATION

7.1 PROCESSING OF ENDF TAPES

ENDF/B formatted files are read in some of the AMPX modules. A number of functions to read and in some cases write parts of ENDF/B formatted files are collected in the *EndfLib* library. A more modern and modular version of this library is provided in *EndfCLib*, which is used in AMPX modules written in C++. The basic building blocks of an ENDF/B formatted file are control records, text records, list records, tab1 records, tab2 records, and intg records (see ENDF-102 manual for details [**Error! Bookmark not defined.**]). The older library functions implemented in FORTRAN read the nuclear data files into memory, but they still require the calling program to be aware of the ENDF structure. The new library functions implemented in C++ store the ENDF data in structures that are closely tied to the actual data. In the case of the new C++ library, an interface layer between the ENDF data files and the processing code allows easy support for new formats of the nuclear data files like the GND format developed by Lawrence Livermore National Laboratory (LLNL) [47]. The new library is mostly used in Y12, the module used to process kinematic data. The module PUFF-IV, which is used to process covariance information, uses its own ENDF reading library that predates the two AMPX libraries.

7.2 FILE FORMATS USED IN AMPX

7.2.1 Tab1 formats

Many programs that need cross section vs. energy data for different reaction and temperatures use a binary TAB1 format for the files. The format closely resembles the format for File 3 in an ENDF formatted file, except that the definitions for some of the fields in the control records have different meanings, depending on the data stored in the file. The energy and cross section data can be given in single or double precision. Most programs will automatically detect the difference. An exception is the module CHARMIN that converts between different forms of the tab1 formatted files. The file format consists of one or more Type 1 records and ends in a Type 2 control record.

Type 1 record

```
MAT,MF,MT,AWR, ZA, L11,L21,N11,N21

MAT,MF,MT,TEMP,SIG0,L12,L22,N21,N22, (NBT(I),INT(I),I=1,N21), (X(I),Y(I),I=1,N22)

MAT,MF,MT,0.0, 0.0, 0,0,0,0
```

The values for AWR,ZA,TEMP and SIG0 are always single precision float values. The values for X and Y can be single or double precision float values. However, all X and Y values in a given file have to be of the same type.

Type 2 record

```
MAT,0,0,0.0,0.0,0,0
```

The C++ class *Tab1Container* in *endfCLib* can be used to access the data. Child classes implement the disk I/O for the data. For an example on how to use *Tab1Container_m*, look at the PICKEZE module. The module CHARMIN will convert between allowed formats of the Tab1 file.

7.2.2 Kinematics file

Y12 writes a kinematics file in a native format. For historical reasons, there are a couple of other kinematic file formats used in AMPX. However, these file formats will be phased out as the modernization of AMPX progresses. The only other format still in use is the legacy Y12 format, which is used by the CRAWDAD module in SCALE to read scattering kernel data for thermal moderators. The C++ class *KinematicContainer* in *endfCLib* holds the data in memory. Child classes manage the disk I/O.

Record 1: MT, NTEMP, ZAI				
	Loop over NTEMP Temperatures			
	Record 2: T, NSUB			
		Loop over NSUB Subsections		
		Record 3: NE, MT, ZAI, ZAP, AWP, AWR, Q, LAB		
			Loop over NE incident energies	
			Record 4: E, NF, UNION, DISCRETE, ELASTIC	
				Loop over NF Exit Energies
				Record 5: LEG, M, EF
				Record 6a (if LEG = 2): (COS_i, i=1,M)
				Record 6: (VAL_i, i=1,M)

The terms used above are defined as follows:

MT	the process identifier
NTEMP	the number of temperatures at which data are given
ZAI	the ZA value of the incident particle
T	the temperature at which data are given (K)
NSUB	the number of subsections given for a temperature
ZAP	the ZA value of the outgoing particle
AWP	the mass ratio of the outgoing particle
AWR	the mast ratio of the target particle
Q	the Q value of the reaction
LAB	1 if data are given in laboratory frame of reference, 0 otherwise
NE	the number of incident energies
E	the incident energies at which kinematics data are given
NF	the number of sink energies
UNION	integer, not used
DISCRETE	1 if the exit energy distribution is discrete, 0 otherwise
ELASTIC	1 if the reaction is elastic or discrete inelastic, 0 otherwise
LEG	1: distribution is given in Legendre moments
	2: distribution is tabulated as a function of cosine of the exit angle
	3: distribution is given in cosine moments
M	number of moments (if LEG=1 or LEG=2) or number of exit angles
EF	the exit energy
COS	the cosine values of the exit angle (only used if LEG=2)
VAL	the value of the distribution

Note that AWP and ZAP are used to describe multiple exit particles that may be produced by a particular reaction (e.g., if the exit particle is a neutron, then AWP=1.0; if it is a gamma, then AWP=0.0).

All float values are stored in double precision.

While there are sections that collected data for a given reaction and temperatures, there can be more than one section for a given reaction or temperature.

The module KINKOS can be used to transform the kinematics file to one of the legacy formats if needed.

7.2.3 MASTER LIBRARY AND WORKING LIBRARY FORMATS

The AMPX MG formats have been designed to allow a generality paralleling that of the ENDF/B point libraries. For example:

1. The formats can accommodate neutron libraries, gamma libraries, or coupled neutron-gamma libraries.

2. An arbitrary number of reaction cross sections can be included with ENDF/B identifiers used for processes where possible.

3. An arbitrary order of anisotropy can be treated which can vary from nuclide-to-nuclide or even from process-to-process in the case of the master library. Temperature dependence is allowed on the master library.

4. The master library can contain Bondarenko data for resonance self-shielding by BONAMI.

5. The master library can include scattering matrix data for an arbitrary number of processes.

In the case of the resonance data, partial energy-range data can be specified. For example, on some of the SCALE libraries, the Bondarenko data are only for the unresolved region, which will vary from nuclide-to-nuclide.

Potentially the most space-consuming data on a cross section library are the transfer matrix data. AMPX uses so-called magic word arrays for these data which help to eliminate zero and/or impossible data elements. This procedure is especially important for the master library, where the library may contain data for more than 50 separate processes represented to an arbitrary level of anisotropy.

The master and working library formats are written using a combination of seven kinds of information, each of which has one or more record types associated with it:

1. Header information – written on the front of the library to specify the number of neutron and/or gamma groups, the number of nuclides, etc., contained in the library (Record Type 1).

2. Energy structure information – contains the group boundaries (Record Type 2).

3. Nuclide directory information – 50 words that give a title for the nuclide, along with other parameters that specify the kinds of information included for the nuclide, such as number of records in the library for the nuclide and how much neutron and gamma data are given (Record Type 3).

4. Bondarenko data – four record types are used for this information:

    a. A record that gives the values of σ_0 and T at which the factors are tabulated, along with cutoff energies for the Bondarenko calculation. The parameter σ_0 is called the background cross section and represents the cross section value for all nuclides mixed with the nuclide being calculated, and T is the temperature value (Record Type 5).

    b. A directory record containing information about the specific processes for which the Bondarenko factor data apply, such as the process, the energy groups for which data are given, etc. (Record Type 6).

c. A record containing infinite dilution values for a process (Record Type 7).

d. A record containing the Bondarenko factors for a process (Record Type 8).

5. A record containing average cross sections by process (Record Type 9).

6. Three record types are used to present transfer matrices:

    a. A directory record that specifies the processes, orders of anisotropy, lengths, units, etc. (Record Type 10).

    b. A record to specify temperatures when the matrices are temperature dependent (Record Type 11).

    c. A magic-word record to store a transfer matrix (Record Type 12).

A discussion of the structure of each of the various record types follows.

Record Type 1 (Header Record)

The header record is the first record on a master and a working library and always contains 110 words:

1.	IDTAPE	An identification number for the library
2.	NNUC	The number of sets of data on the library
3.	IGM	The number of neutron energy groups on the library.
4.	IFTG	The first thermal neutron group on the library (i.e., the first group that **receives** an upscatter source)
5.	MSN	Master library version type (2 for NITAWL-II resonance processing compatibility)
6.	IPM	The number of gamma-ray energy groups on the library
7.	I1	Zero
8.	I2	(0/1 = no/yes) A trigger that specifies that this library was produced by weighting a working library in the XSDRNPM module.
9.	I3	Zero
10.	I4	Zero
11. – 110.	(TITLE(I), I=1,100)	100 words of text describing the cross section library.

Record Type 2 (Energy Boundaries)

This record is on both a master library and a working library and specifies the energy boundaries in eV of the neutron groups and/or gamma groups, followed by the corresponding lethargy boundaries. The energy boundaries are arranged in descending order, followed by the lethargy boundaries in ascending order. The lethargy zero is normally taken at 10 MeV. The structure is

(EB(I),I=1,IGP), (UB(I),I=1,IGP) ,

where IGP is the number of groups plus 1.

Record Type 3 (Cross section Set Directory Record)

Each set of data on a master or working library has a 50-word directory record that specifies certain parameters needed to determine dimensions required to process the data and to describe the make-up of the set of data. Table 1 describes these data.

Note that the 50-word records are made up of integer, character, and floating-point words. Words 1–18 and 49 are character data. Words 29, 30, 34, 35, and 43 are floating point. All other words are integers. For both types of libraries, many parameters may have no meaningful interpretation for a particular set of data. This situation is especially true of the working library. For example, words 20, 21, 22, 25, and 26 only have meaning if the working library has been produced by weighting a previous working library. Zero values will be used when a parameter is not applicable.

Record Type 4 (Resonance Parameters)

No longer used.

Table 1. Record type 3 (cross section directory record)

Word(s)	Master library	Working library
1–18	18 words of text describing the set	18 words of text describing the set
19	Identifier of the set	Identifier of the set
20	Number of 6-parameter sets of resolved resonance data	Identifier of the set from which this set derived
21	Number of energies at which to evaluate unresolved values	Zone number in which the nuclide occurred
22	Number of neutron processes for which group-averaged values are given (temperature independent)	Number of zones in problem which produced this set
23	Number of neutron processes with scattering arrays	Length of P_0 total scattering matrix
24	Zero	Order of expansion of total scattering matrix
25	Number of gamma processes for which group-averaged values are given	Sequence of this set in all zone-weighted sets
26	Number of gamma processes with scattering arrays	Number of zone-weighted sets for this nuclide
27	Number of neutron-to-gamma processes	Maximum length of any scattering array in with scattering arrays
28	(Maximum order of scattering)*32768 + (total number of separate scattering arrays for this set)	Number of neutron processes which have group-averaged values
29	A – neutron equivalent mass number	A – neutron equivalent mass number
30	ZA – 1000*Z + A	ZA – 1000*Z + A
31	Zero	Zero
32	Zero	Zero
33	Zero	Zero

34	Power per fission in watt-sec/fission	Power per fission in watt-sec/fission

Table 2. Record type 3 (cross section directory record) (continued)

Word(s)	Master library	Working library
35	Energy release per capture in watt-sec/capture	Energy release per capture in watt-sec/capture
36	Maximum length of any scattering array in the set	Zero
37	Number of sets of Bondarenko data	Zero
38	Number of σ_0 values in Bondarenko data	Zero
39	Number of temperature values in Bondarenko data	Zero
40	Maximum number of groups in Bondarenko data	Zero
41	Zero	Number of gamma processes that have group-averaged values
42	Zero	Zero
43	σ_p — potential scattering cross section	Zero
44	Zero	Zero
45	ENDF MAT for fast neutron data	ENDF MAT for fast neutron data
46	ENDF MAT for thermal neutron data	ENDF MAT for thermal neutron data
47	ENDF MAT for gamma data	ENDF MAT for gamma data
48	ENDF MAT for gamma production data	ENDF MAT for gamma production data
49	Source: 0=ENDF	Source: 0=ENDF
50	Number of records in this set	Number of records in this set

Record Type 5 (First Record of Bondarenko Block)

This record is used to specify the σ_0 and temperature values at which all Bondarenko factors for the nuclide will be presented. It also specifies the upper and lower energies for which factors can apply in the case where they do not span all energy groups. The number of σ_0 values, NSIG0, is specified in the 38th word in the set directory, and the 39th word specifies the number of temperatures, NT. The record structure is

$$(\sigma_0(\mathrm{I}),\ \mathrm{I=1,\ NSIG0}),\ (\mathrm{T(I),\ I=1,\ NT}),\ \mathrm{ELO,\ EHI.}$$

The σ_0 values can either ascend or descend; the temperatures are expressed in Kelvin in ascending order. The parameter σ_0 is the cross section value for the other nuclides mixed with a nuclide in a particular situation.

Record Type 6 (Directory for Bondarenko Data)

This record type is used to specify the processes that have Bondarenko data in the set. Its length is six times NBOND, the number of Bondarenko processes, specified in the 37th word in the set directory. The structure is the following:

(MT(I), I=1, NBOND),

(NF(I), I=1, NBOND),

(NL(I), I=1, NBOND),

(ORDER(I), I=1, NBOND),

(IOFF(I), I=1, NBOND), and

(NZ(I), I=1, NBOND).

The parameters have the following interpretation: MT is the identifier of the process (e.g., MT = 2 is for elastic scattering, as in ENDF/B). NF is the number of the first energy group for which parameters are given. NL is the last group for which parameters are given. ORDER is used to specify lower group of homogeneous or heterogeneous f-factors and IOFF the upper group. NZ is presently unused and has a zero value.

Record Type 7 (Infinite Dilution Values for Bondarenko Data)

Each process that has Bondarenko data has one of these records which contain the infinite-dilution values for the process. Its structure is

$$(\sigma^{\infty}(I), I=NF, NL),$$

where NF and NL, are the first and last groups with data for the process.

Record Type 8 (Bondarenko Factors)

This record is a 3-D array and contains the Bondarenko factors for a process. Its structure is

$$(((BF(I,J,K), I=1, NSIGO), J=1, NT), K=NF, NL).$$

Record Type 9 (Temperature-Independent Average Cross Sections)

This record type is used to present average cross sections, sometimes called 1-D cross sections on the library.

Its structure is

MT_1, (σ_1(I), I=1, IGM)

MT_2, (σ_2(I), I=1, IGM)

.

.

.

MT_{LAST}, (σ_{LAST}(I), I=1, IGM),

where the MTs are the process identifiers, and the cross sections, σ, are given for all groups (**NOTE:** The MTs are given as floating point numbers).

Record Type 10 (Scattering Matrix Directory)

An AMPX master library always provides a directory that identifies the scattering matrices which are given for a nuclide. The structure is

(MT(I), I=1, N2D),

(L(I), I=1, N2D),

(NL(I), I=1, N2D), and

(NT(I), I=1, N2D),

where N2D is the number of scattering (2-D) processes, MT is the process identifier, L is the maximum length of any of the scattering matrices for the process, NL is the order of Legendre fit to the scattering matrix, and NT is a parameter whose definition depends on the type of data (whether neutron, gamma production, or gamma) given as follows:

1. For neutron-neutron data, NT is the number of temperatures at which scattering matrices are given. If only one temperature is present it may be 0.

2. For gamma production data, NT is zero if the data are in yield units and is unity if they are in cross section units.

3. For gamma-gamma data, NT is zero.

Record Type 11 (Scattering Matrix Temperatures)

This record type is only used on a master library and specifies the temperatures (in eV) of the scattering matrices. It is only used for neutron-neutron data and is given when NT > 0 (see Record Type 10). The temperatures are in ascending order as follows:

(T(I), I=1, NT).

Record Type 12 (Scattering Matrix)

This record type is used to store scattering-matrix data (sometimes called 2-D data). As will be illustrated, it has provisions for truncating zero and/or impossible elements from the array. It exists in two forms: (1) a self-defining form used for gamma production data on a master library and for all scattering matrices on a working library, and (2) a form that is not self-defining. The only difference is that the self-defining form specifies the length as the first word, while the other does not; that is,

L, (X(I), I=1, L)

or

(X(I), I=1, L).

The structure of the X-array is as follows:

magic word for a group,

terms for scattering to the group,

magic word for the next group,

terms for scattering to this group,

etc..

In some cases, a negative or zero magic word is used to specify the end of data in the record.

A magic word is used to define:

1. the sink group number, III,
2. the first group number, JJJ, which scatters to this group, and
3. the last group number, KKK, which scatters to this group.

The magic word is then defined as

$$\mathrm{MW} = 1000000^*\mathrm{JJJ} + 1000^*\mathrm{KKK} + \mathrm{III},$$

such that it is composed of three 3-digit integers:

$$\mathrm{MW} : \mathrm{JJJKKKIII}\ .$$

The scattering terms below a magic word are in reverse ordering (following typical practice for transport theory programs); that is, the scattering term for scattering from the last group is first, etc.:

MW for group III

$\sigma(\mathrm{KKK}\rightarrow\mathrm{III})$

$\sigma(\mathrm{KKK}\text{-}1\rightarrow\mathrm{III})$

.

.

.

$\sigma(\mathrm{JJJ}\rightarrow\mathrm{III})$

The scattering matrix record will contain one P_ℓ matrix for a process.

Consider an elastic scattering matrix for hydrogen which will be a full triangular matrix and assume three energy groups. The scattering matrix will look as follows:

1001001

$\sigma(1\rightarrow1)$

1002002

$\sigma(2\rightarrow2)$

$\sigma(1\rightarrow2)$

1003003

$\sigma(3\rightarrow3)$

$\sigma(2\rightarrow3)$

$\sigma(1\rightarrow3)$

NOTE: The record is a mixture of integer and floating-point terms.

AMPX Master Library Format

The overall structure of an AMPX master library is given as follows:

	Record type
Header record	1
Nuclide directory (one record per nuclide)	3
Neutron energy boundaries	2
Gamma energy boundaries	2
Records for nuclide 1	
Records for nuclide 2	
etc.	

The structure of the records of a nuclide is:

	Record type
Nuclide directory record	3
Bondarenko data (one set per nuclide)	5,6,7,8
Temperature-independent group-averaged neutron cross sections	9
Scattering matrix data for neutrons	10,11,12
Scattering matrix data for gamma production	10,12
Group-averaged gamma cross sections	9
Scattering matrix data for gammas	10,12

The internal structure for Bondarenko data is:

	Record type
(σ_0(I),I=1,NSIGO),(T(J),J=1,NT),EL,EH	5
Bondarenko data directory (MT(I),I=1,NBOND), (NF(I),I=1,NBOND), (NL(I),I=1,NBOND), (ORDER(I),I=1,NBOND), (IOFF(I),I=1,NBOND), (NZ(I),I=1,NBOND)	6
The following records are given in pairs for all NBOND Bondarenko processes.	
Infinite dilution values (σ_∞(i),I=NF,NL)	7
Bondarenko factors (((BF(I,J,K),I=1,,NSIGO),J=1,NT),K=NF,NL)	8

The internal structure of the scattering matrix data for neutrons is:

	Record type
Scattering matrix directory (MT(I),I=1,N2D), (L(I),I=1,N2D), (NL(II),I=1,N2D), (NT(I),I-1,N2D)	10

The following structure is repeated N2D times.

Temperature values (NT>0) (T(I),I=1,NT)	11
The following record is repeated MAX(1,NT)*(NL+1) times.	
Scattering matrix (X(I),I=1,L)	12

The internal structure for gamma production scattering is:

	Record type
Scattering matrix directory (MT(I),I=1,N2D), (L(I),I=1,N2D), (NL(I),I=1,N2D)	10
For each process, I=1, N2D, the following record type is given NL+1 times corresponding to the P_0, P, ..., P_{NL} matrices.	
Self-defining scattering matrix length LENGTH,(X(I),I=1,LENGTH)	12

The internal structure for gamma-gamma scattering is:

	Record type
Scattering matrix directory (MT(I),I=1,N2D), (L(I),I=1,N2D), (NL(I),I=1,N2D), (NT(I),I=1,N2D)	10
For each process, I=1, N2D, the following record type is given NL+1 times corresponding to the P_0, P_1, ..., P_{NL} matrices.	
Scattering matrix (X(I),I=1,L)	12

AMPX Working-Library Format

The overall structure of a working library is given below:

	Record type
Header records	1
Neutron energy boundaries	2
Gamma energy boundaries	2
Nuclide directory (one record per nuclide)	3
Records for nuclide 1	
Records for nuclide 2	
etc.	

The structure of the records for a nuclide is:

	Record type
Nuclide directory record	3
Group-averaged neutron cross sections	9
Group-averaged gamma cross sections	9
The P_0, P_1, ..., P_{NL} total scattering matrices are presented in self-defining records.	
L,(X(I),I=1,L)	12

The AMPX and SCALE code system uses a C++ class *AmpxLibrary* with FORTRAN bindings to read, save, and store MG library data.

7.2.4 CE library format

The following description provides the cross section formatting details for a single isotope/nuclide in a CE KENO cross section library. Note that the description only applies to the library format and does not reflect how the data are stored in the code during a calculation.

7.2.4.1 Cross section file format

The cross section file for each nuclide/isotope is composed of multiple blocks of data that describe the physics of radiation transport for a specific isotope/nuclide. The organizational structure of the various blocks of data within the library is presented in Table 2.

Table 3. Continuous-energy cross section file organization

Block	Description	Zero temperature file	Temperature -dependent file
-0	AMPX/SCALE Header Block	yes	yes
1	Header Information	yes	yes
2	$\overline{V}$ Data	yes	
3	MT Data	yes	yes
4	Unionized energy grid for total, elastic scattering, fission and capture	yes	yes
5	CE cross section data [σ(E)]	yes	yes
6	Energy-dependent collision probabilities		yes
7	Forward kinematics data (secondary angle and energy distributions)	yes	yes
8	Probability table data		yes
9	Macroscopic cross sections (not used)		yes
10	Adjoint source (not used)		yes
11	Adjoint kinematics data (not used)	yes	yes
12–20	Not used		

Each block can have multiple records that are used to describe the physics associated with each type of data block. A description of the records within each data block is provided in the subsequent sections.

Each nuclide/isotope has one zero-temperature file (also referred to as a temperature-independent file) and multiple temperature-dependent files. Zero-temperature files contain the reactions that do not change as a function of temperature. Temperature-dependent files contain the reactions that have been Doppler broadened for the specified temperature. The cross section files for the nuclides with thermal scattering data have been generated at the temperatures that are provided in the corresponding ENDF/B evaluation files. All other nuclides/isotopes have multiple temperature-dependent files.

7.2.4.2 AMPX/SCALE header information block

The first block contains the header information related to data generation (i.e., date, code version etc.). The format of the AMPX/SCALE header block is provided in Table 3. Note that record number 1, which contains the information about the number of records of type character, real, and integer is not included in the counter for the number of records. For example, if NC is 10, then the last record of character type is record number 11.

Table 4. AMPX/SCALE header information block format

Record	Parameter	Type	Description
1	NC	Integer	Number of records of character variables
1	NR	Integer	Number of records of real variables
1	NI	Integer	Number of records of integer variables
2	FILENAME	Character*80	Filename prefix used for this nuclide
2	AMPXDATE	Character*80	Date AMPX modules were created
2	SCALEDATE	Character*80	Date SCALE modules were created
2	ICFILEDATE	Character*80	Date input creater was created
3	ICVERSION	Character*80	Version of input creater
3	AXVERSION	Character*80	Version of AMPX modules used
3	SCVERSION	Character*80	Version of SCALE modules used
3	FILEDATE	Character*80	Date this file was created
4	FVERSION	Character*80	Version of this file format
4	UNION	Character*80	Flag to signal unionized cross sections
4	DUM	Character*80	Dummy place holder
4	DUM	Character*80	Dummy place holder
	NEXT (NC-4) RECORDS HAVE THE FOLLOWING STRUCTURE		
5 to NC	DUM,TEMPSUFFIX,DUM,DUM	Character*80	DUM – character dummy place holder TEMPSUFFIX – suffix in the filename for each temperature
NC+1	BLANK		
	NEXT (NR) RECORDS HAVE THE FOLLOWING STRUCTURE		
NC+2 to NC+NR	DUMR,TEMP,DUMR,DUMR	Real	DUMR – real dummy place holder TEMP – temperature at which cross sections are available
NC+NR+1	DUMR,DUMR,DUMR,DUMR	Real	DUMR – real dummy place holder
	NEXT (NI) RECORDS HAVE THE FOLLOWING STRUCTURE		
NC+NR+2 to NC+NR+NI+1	DUMI,DUMI,DUMI,DUMI	4Integer	DUMI – integer dummy place holder

7.2.4.3 Header information block

The header block is used to provide the generic information about the library and subsequent data blocks. The format of the header block is provided in Table 4. Note that the first record in the header block identifies the material, and a mixture parameter is provided to associate the isotope/nuclide with a mixture number. During the generation of a CE KENO library, the mixture ID is set to zero. If a problem-dependent library is prepared in the future, the mixture ID will be set to the appropriate mixture number. The header block always has 12 records and exists in zero-temperature and temperature-dependent files.

Table 5. Header block format

Record	Parameter	Type	Description
1	MAT	Integer	ENDF material identifier
1	ID	Integer	ID Source
1	MIX	Integer	mixture ID (> 0 for problem dependent library, 0 otherwise)
1	ZA	Integer	ZA number for isotope/nuclide (Z*1000+A)
2	LENGTH	Integer	Length of TITLE array
2	TITLE	Character*100	Character*LENGTH title that includes source ENDF identification
3	LFI	integer	Fission flag (LFI = 0/1 does not fission/does fission)
3	LPTAB	Integer	Probability table flag (LPTAB = 0/1 no tables/tables present)
3	ISO	Integer	Isotope flag (0/1 no/yes)
3	LSAB	Integer	S(α,β) data exist (0/1 no/yes)
3	NMT	Integer	Number of 1-D reactions
3	NUM_FAST	integer	Number of 2-D temperature independent MTS
3	NUM_THERM	integer	Number of 2-D temperature dependent MTS
3	MAX_ANGLES	Integer	Maximum number of angles in 2-D kinematics block
3	MAX_EXITE	Integer	Maximum number of exit energies in 2-D kinematics block
3	MAX_TEMPS	Integer	Maximum number of temperatures
3	CHANCE	Integer	2-D 1st, 2nd, 3rd, or 4th chance fission data exist
3	METASTATE	Integer	State of the nuclide, 0 indicates ground state
4	AWR	Double	Atomic weight ratio
4	ELR	Double	Lower energy boundary for resolved-resonance region (eV)
4	EHR	Double	Upper energy boundary for resolved-resonance region (eV)
4	ELU	Double	Lower energy boundary for the unresolved-resonance region (eV)
4	EHU	Double	Upper energy boundary for the unresolved-resonance region (eV)
4	TEMP(i), i=1,MAX_TEMPS	Double	Temperature (K) for the cross section data
5	SIGP	Real	Potential cross section
6–12			Not used

7.2.4.4 $\bar{\nu}$ block (if LFI=1)

The format of the $\bar{\nu}$ record within the data block is provided in Table 5. Note that this block can have up to four different $\bar{\nu}$ records. The first record provides the total $\bar{\nu}$ as a function of energy, and the format has provisions to provide the delayed and prompt $\bar{\nu}$ as a function of energy. The final record provides the ratio of delayed $\bar{\nu}$ to total $\bar{\nu}$ as a function of energy. This block exists in zero degree files only.

Table 6. $\overline{\nu}$ data block format

IZA	**MB**	**NU_COUNT**	**MAX_NR**	**MAX_NP**				
452	NR	NP	NBT(n)	INT(n)	n=1,NR	*E*(i)	$\overline{\nu}$ (i)	i=1,NP
455	NR	NP	NBT(n)	INT(n)	n=1,NR	*E*(i)	$\overline{\nu}_d$(i)	i=1,NP
456	NR	NP	NBT(n)	INT(n)	N=1,NR	*E*(i)	$\overline{\nu}_p$(i)	i=1,NP
459	NR	NP	NBT(n)	INT(n)	N=1,NR	*E*(i)	$\overline{\nu}_d$(i)/ $\overline{\nu}$ (i)	i=1,NP

The variables for the $\overline{\nu}$ record are defined as follows:

IZA	integer form of ZA number [integer],
MB	block number (MB = 2) [integer],
NU_COUNT	Number of types of $\overline{\nu}$ data (possible MTs listed below) [integer],
MAX_NR	Maximum number of NR values [integer],
MAX_NP	Maximum number of NP values [integer],
MT	452/455/456/459 total/delayed/prompt/ratio of delayed to total [integer],
NR	number of interpolation regions [integer],
NP	number of points [integer],
NBT(n)	end point of interpolation region n [integer],
INT(n)	interpolation type for region n (ENDF interpolation types) [integer],
E(i)	ith energy point [double],
$\overline{\nu}$ (i)	value of total $\overline{\nu}$ corresponding to *E*(i) [double],
$\overline{\nu}_d$(i)	value of delayed $\overline{\nu}$ corresponding to *E*(i) [double],
$\overline{\nu}_p$(i)	value of prompt $\overline{\nu}$ corresponding to *E*(I) [double], and
$\overline{\nu}_d$(i)/ $\overline{\nu}$ (i)	ratio of delayed $\overline{\nu}$ to total $\overline{\nu}$ at *E*(i) [double].

7.2.4.5 MT block

Both zero degree and temperature-dependent files contain all MT numbers for all temperatures. Table 6 lists the structure of the two MT records.

Table 7. Reaction identifier (MT) data block format

IZA	MB	0	C1	C2	L1	L2	N1	NUM_FAST_1D	MT(i)	i=1,NUM_FAST_1D
IZA	MB	0	C1	C2	L1	L2	N1	NUM_THERM_1D	MT(i)	i=1,NUM_THERM_1D

IZA	integer form of ZA number [integer]
MB	block number (MB = 3) [integer]
C1	place holder for real quantity (typically 0.) [double]
C2	place holder for real quantity (typically 0.) [double]
L1	place holder for integer quantity (typically 0) [integer]
L2	place holder for integer quantity (typically 0) [integer]
N1	place holder for integer quantity (typically 0) [integer]
NUM_FAST_1D	number of temperature-independent reactions [integer]
NUM_THERM_1D	number of temperature-dependent reactions [integer]
MT(i)	identifier for the ith reaction [integer]

7.2.4.6 Unionized energy grid block

The energy mesh is for MT=1 (total). Since all temperature-dependent MTs are unionized, the mesh is the same for all MTs. Note that even though the temperature-independent MTs are not unionized, the mesh contains the points for temperature-independent MTs, as well. This allows quick mapping of the temperature-independent MTs before execution in KENO. The unionized energy grid data are listed in Table 7. Note that the collision probabilities have a different energy mesh that is also included in the energy grid block. The zero degree file contains all energy grids for all temperatures (MAX_TEMPS sets of records), whereas the files for specific temperatures contain only the energy grid for that temperature.

Table 8. Unionized energy grid data block format

IZA	MB	TEMP	NE	NE_CP
E(i)	i=1,NE			
E_CP(i)	i=1,NE_CP			

IZA	integer form of ZA number [integer]
MB	block number (MB = 4) [integer]
TEMP	Temperature (K) [double]
NE	number of energy points [integer]
NE_CP	number of energy points for collision probabilities [integer],
E(i)	ith Energy point [double]
E_CP(i)	ith Energy point for collision probability arrays [double]

7.2.4.7 CE cross section block

This block exists in each file with temperature-dependent files being slightly different. Since the energy grid for the temperature-dependent MTs is already in BLOCK 4 (MB=4), the energy points, E(i), are not included in the temperature-dependent files. Table 8 shows the records and their structure.

Table 9. CE microscopic cross section data block format

IZA	MB	TEMP	MAX_NR	MAX_NP									
MT_1	Q	TEMP	EMIN	EMAX	NOUT	NR	NP	NBT(n)	INT(n)	n=1,NR	*E*(i)	σ(i)	i=1,NP
MT_2	Q	TEMP	EMIN	EMAX	NOUT	NR	NP	NBT(n)	INT(n)	n=1,NR	*E*(i)	σ(i)	i=1,NP
MT_{NUM_MTX}	Q	TEMP	EMIN	EMAX	NOUT	NR	NP	NBT(n)	INT(n)	n=1,NR	*E*(i)	σ(i)	i=1,NP

IZA	integer form of ZA number [integer]
MB	block number (MB = 5) [integer]
TEMP	Temperature (K) [double]
EMIN	Minimum energy of the reaction (includes additional point with zero cross section value for interpolation purposes) [double]
EMAX	Maximum energy of the reaction (includes additional point with zero cross section value for interpolation purposes) [double]
MAX_NR	Maximum of NR values [integer]
MAX_NP	Maximum of NP values [integer]
MT	Reaction identifier [integer]
NUM_MTX	Number of MTs (NUM_FAST_1D in temperature-independent file, NUM_THERM_1D in temperature-dependent file) [integer]
Q	reaction energy or Q value [double]
NOUT	number of secondary neutrons produced by the reaction (Note: if MT=18,19,20, 21 or 38, the number of secondary neutrons is determined from the $\overline{\nu}$ data block, and NOUT will be set to zero. For neutron disappearance reactions, NOUT is also set to zero) [integer]
L1	place holder for integer quantity (typically 0) [integer]
NR	number of interpolation regions [integer]
NP	number of points [integer]
NBT(n)	end point of interpolation region n [integer]
INT(n)	interpolation type for region n (ENDF interpolation types) [integer]
E(i)	ith energy point [double]
σ(i)	microscopic cross section value corresponding to *E*(i) [double].

7.2.4.8 Energy-dependent collision probabilities

The following block of data provides the energy-dependent collision probabilities. The energy-dependent collision probabilities that are needed for the Monte Carlo random walk are the nonabsorption [Pinabs(E)], absorption [Piabs(E)] and fission [Pif(E)] probabilities:

$$P^i_{nabs}(E) = \frac{\sigma^i_s(E)}{\sigma^i_t(E)}$$

$$P^i_{abs}(E) = \frac{\sigma^i_a(E)}{\sigma^i_t(E)}$$

$$P^i_f(E) = \frac{\nu^i(E)\sigma^i_f(E)}{\sigma^i_t(E)}$$

where

$\sigma_s^i(E)$	=	scattering cross section for isotope/nuclide *i*,
$\sigma_t^i(E)$	=	total cross section for isotope/nuclide *i*,
$\sigma_a^i(E)$	=	absorption cross section for isotope/nuclide *i*,
$\sigma_f^i(E)$	=	fission cross section for isotope *i*, and
$\bar{\nu}^i(E)$	=	average number of neutrons released per fission at energy *E* for isotope *i*.

In addition, (n,2n) and (n,3n) reaction probabilities are also saved if those reaction cross sections exist for the nuclide. The energy-dependent absorption and fission probabilities can be used in both the forward and adjoint modes of transport; however, the nonabsorption probability defined above is not the same in the adjoint mode of transport. Therefore, an adjoint nonabsorption probability can be defined as follows:

$$P^{i*}_{nabs}(E') = \frac{\int_E dE \int_\mu d\mu \sigma^i_s(E' \rightarrow E, \mu)}{\sigma^i_t(E')},$$

where

$\sigma_s^i(E' \rightarrow E, \mu)$	=	isotope/nuclide *i* differential scattering cross section for scattering from *E′* to *E* through angle μ, and
$\sigma_t^i(E')$	=	isotope/nuclide *i* total cross section at energy *E′*.

For each isotope/nuclide, the energy-dependent collision probability block may have up to five different records that correspond to the five different collision probabilities. If the LFI flag is zero, the fission probability record is zero for the isotope. Each collision probability record is provided in the format shown in Table 9.

This block is in temperature-dependent files only. Note that since the unionized energy grid data block already includes the points for the collision probability data, energy points are not included.

Table 10. Energy-dependent collision probability data block format

IZA	MB	NREAD	TEMP	MAX_NR	MAX_NP						
2006	C1	TEMP	L1	L2	NR	NP	NBT(n)	INT(n)	n=1,NR	P_{2006}(i)	i=1,NP
2007	C1	TEMP	L1	L2	NR	NP	NBT(n)	INT(n)	n=1,NR	P_{2007}(i)	i=1,NP
2016	C1	TEMP	L1	L2	NR	NP	NBT(n)	INT(n)	n=1,NR	P_{2016}(i)	i=1,NP
2017	C1	TEMP	L1	L2	NR	NP	NBT(n)	INT(n)	n=1,NR	P_{2017}(i)	i=1,NP
2018	C1	TEMP	L1	L2	NR	NP	NBT(n)	INT(n)	n=1,NR	P_{2018}(i)	i=1,NP
2027	C1	TEMP	L1	L2	NR	NP	NBT(n)	INT(n)	n=1,NR	P_{2027}(i)	i=1,NP

IZA	integer form of ZA number [integer]
MB	block number (MB = 6) [integer]
NREAD	Number of collision probability records (e.g., nonfissile nuclides don't have 2018, pure-scatterers don't have 2027) [integer]
TEMP	Temperature (K) [double]
MAX_NR	Maximum of NR values for all collision probabilities [integer]
MAX_NP	Maximum of NP values for all collision probabilities [integer]
MT	collision probability identifier [integer]
	MT = 2006: nonabsorption probability
	MT = 2007: adjoint nonabsorption probability
	MT = 2016: (n,2n) reaction probability
	MT = 2017: (n,3n) reaction probability
	MT = 2018: fission probability
	MT = 2027: absorption probability
C1	place holder for real quantity (typically 0.) [double]
TEMP	Temperature (K) [double]
L1	place holder for integer quantity (typically 0) [integer]
L2	place holder for integer quantity (typically 0) [integer]
NR	number of interpolation regions (NR = 1) [integer]
NP	number of points [integer]
NBT(n)	end point of interpolation region n [NBT(NR) = NP] [integer]
INT(n)	interpolation type for region n [INT(NR) = 2] [integer]
P_{MT}(i)	collision probability at energy E(i) [double]

7.2.4.9 Forward kinematics data block

The kinematics section of the library provides the information for determining the exiting energy and angle of a particle emerging from a collision with a target isotope/nuclide. Because of the complexity of collision kinematics, different types of collision representations may be provided depending upon the type of reaction to be processed.

Zero-temperature files contain extra records that are used to dimension the required arrays. These records are listed in Table 10. Record number 2 in Table 10 is repeated for each temperature (including zero) so that the total number of records is MAX_TEMPS+2.

Table 11. Header records in zero-temperature file only

IZA	MB	NUM_FAST+ NUM_THERM	MAX_NR	MAX_ANGLES	MAX_EXITE
MAX_MU(n)	n=1, NUM_FAST+NUM_THERM		MAX_EOUT(n)	n=1,NUM_FAST+NUM_THERM	

7.2.4.10 Forward kinematics block

The kinematics data are provided for each reaction that has a secondary angle and energy distribution The format of the kinematics data for each reaction is provided in Table 11. The definitions for each parameter in Table 11 are provided in the description that follows the table. The data structure in Table 11 appears to be rather complex; however, the structure is needed to accommodate coupled energy-angle distributions. A universal kinematics data structure is desired to accommodate all possible secondary energy-angle distributions. The most complex structure is the coupled energy-angle data for thermal scattering and ENDF/B File 6 distributions. By addressing the most complex structure with the kinematics format, the less complex distributions can be treated by default. Therefore, the data structure outlined in Table 11 was developed to address the coupled distributions. In terms of Monte Carlo, the data structure represents the joint CDF for selecting the secondary angle and energy. As with any method, there are pros and cons to the structure in Table 11. The data structure has the advantage of uniformity; however, there is a storage penalty associated with the representation of non-coupled energy-angle distributions. For the purposes of CE KENO, the uniform data structure advantage outweighs the added storage cost that is incurred.

As shown in Table 11, the kinematics data structure has a header record that specifies the number of sections (NSECT) used to describe the secondary distributions for the reaction. The kinematics structure is divided into NSECT incident energy blocks, and the first record within each section specifies the incident energy range and number of incident energies (NE) for the section. For example, the first section in Table 11 is characterized with incident energies between E11 and E1NE, and the last section is defined for incident energies between ENSECT1 and ENSECTNE. Note that the NE values can vary between sections.

Each section is subsequently divided into multiple blocks of data that describe the secondary energy-angle distributions. The first block of data has a secondary angular cosine distribution for each incident energy; therefore, there are NE angular distributions in the (E, μ) data block. In terms of Monte Carlo, each angular distribution record in the (E, μ) data block corresponds to the marginal CDF for selecting the secondary angle at an incident energy. The format specifies NPU secondary cosines for each incident energy, and the number of secondary cosines can vary between incident energies. In other words, the NPU variable can change between incident energies within a section. The NPU angle cosines correspond to NPU - 1 cosine bins. For example, the first angle bin has a lower boundary of μ_1 and an upper boundary of μ_2.

Each μ distribution is defined with NPU angle cosines and the corresponding cumulative probability, Cμ, for each cosine bin. In addition, the value of the PDF at each angle, Pμ, is also provided in the format. For the secondary angular distribution, the Pμ values are needed for interpolation purposes. Because there are NPU - 1 cosine bins, there are NPU - 1 Cμ values provided with the distribution; however, there are NPU values of the PDF that are provided for each angular cosine. As a result, the NPU location in the cumulative probability distribution is not needed and is zero.

Following the (E, μ) data block in a section are multiple blocks of data that describe the secondary energy distributions. As noted previously, each incident energy has a secondary distribution of NPU angle cosines. For each (E, μ) pair, there is a corresponding secondary energy distribution that can have NPE secondary energies. As indicated in Table 11, there is an (E, μ, E′) data block for each incident energy, and there are NPU secondary energy distribution records within each (E, μ, E′) data block. Each secondary energy distribution record within an (E, μ, E′) block represents a conditional CDF for selecting the exit energy for a given incident energy E and secondary angle μ. The term *conditional CDF* implies that the exit cosine has been selected, and the corresponding CDF for exit energy defines the probability of selecting the exit energy for the given secondary angle cosine.

The format of each secondary energy distribution record is analogous to the secondary angular distribution representation. For each (E, μ) pair, there are NPE secondary energies that correspond to NPE - 1 energy bins. Moreover, the cumulative probabilities, CE′, for each energy bin are also provided with the distribution. For each secondary energy, there is a location that can be used to store the value of the PDF for the exit energy, PE′, that is used for interpolation during the sampling process. If the reaction is elastic or discrete-level inelastic scattering, the PE′ values will be interpolation parameters for the power-interpolation method that was initially developed for the BONAMI module.

There are two special cases that can be described with the kinematics structure in Table 11. If the secondary angular distribution is isotropic at an incident energy E, there will be a single exit cosine specified (i.e., NPU =1) in the (E, μ) block with a value of -2.0 and corresponding probability of 1.0. As a result, the exit cosine will be sampled uniformly between -1.0 and 1.0. Since there is only one exit cosine specified for isotropic scattering, the (E, μ, E′) block will have one record that specifies the secondary energy distribution for any secondary angle at an incident energy E. If the interaction mechanism is coherent or incoherent elastic scattering, there is no change in energy resulting from the collision (i.e., E′ = E). Therefore, each secondary energy distribution in the (E, μ, E′) block will only have one exit energy with a value of E and corresponding probability of 1.0.

The following table shows the kinematics data structure. Zero degree file contains temperature-independent reactions only (NUM_FAST MTs). Temperature-specific files contain the temperature-dependent MTs for that temperature (NUM_THERM MTs).

Table 12. Forward kinematics data structure for a single reaction

MT	**TEMP**	**NSECT**								**Header record**
E^1_1	E^1_{NE}	**NR**	NE	**AWP**	**LD**	**ZAP**	**YIELD**			**1st Section**
E^1_1	**C2**	**LMU**	**L2**	**NR**	NPU	**μ(1:NPU)**	C_μ**(1:NPU)**	P_μ**(1:NPU)**	YIELD_1	
E^1_2	**C2**	**LMU**	**L2**	**NR**	NPU	**μ(1:NPU)**	C_μ**(1:NPU)**	P_μ**(1:NPU)**	YIELD_2	
			.							
			.							
			.							(E^1, μ) Block Marginal CDF
E^1_{NE}	**C2**	**LMU**	**L2**	**NR**	NPU	**μ(1:NPU)**	C_μ**(1:NPU)**	P_μ**(1:NPU)**	YIELD_{NE}	
E^1_1	**μ(1)**	**1**	**LE**	**NR**	NPE	E'**(1:NPE)**	$C_{E'}$**(1:NPE)**	$P_{E'}$**(1:NPE)**		
E^1_1	**μ(2)**	**2**	**LE**	**NR**	NPE	E'**(1:NPE)**	$C_{E'}$**(1:NPE)**	$P_{E'}$**(1:NPE)**		
			.	.	.					(E^1_1, μ, E') Block Conditional CDF
			.	.	.					
			.	.	.					
E^1_1	**μ(*NPU*)**	NPU	**LE**	**NR**	NPE	E'**(1:NPE)**	$C_{E'}$**(1:NPE)**	$P_{E'}$**(1:NPE)**		
E^1_2	**μ(1)**	**1**	**LE**	**NR**	NPE	E'**(1:NPE)**	$C_{E'}$**(1:NPE)**	$P_{E'}$**(1:NPE)**		
E^1_2	**μ(2)**	**2**	**LE**	**NR**	NPE	E'**(1:NPE)**	$C_{E'}$**(1:NPE)**	$P_{E'}$**(1:NPE)**		
			.	.	.					
			.	.	.					
			.	.	.					(E^1_2, μ, E') Block Conditional CDF
E^1_2	**μ(*NPU*)**	NPU	**LE**	**NR**	NPE	E'**(1:NPE)**	$C_{E'}$**(1:NPE)**	$P_{E'}$**(1:NPE)**		
							.			.

							.			.
							.			.
							.			.
E^1_{NE}	μ(1)	1	LE	NR	*NPE*	E'(1:NPE)	$C_{E'}$(1:NPE)	$P_{E'}$(1:NPE)		
E^1_{NE}	μ(2)	2	LE	NR	*NPE*	E'(1:NPE)	$C_{E'}$(1:NPE)	$P_{E'}$(1:NPE)		
			.	.	.					($E^{1}{}_{\mathrm{NE}}$, μ , E') Block
			.	.	.					Conditional CDF
			.	.	.					**End 1st Section**
E^1_{NE}	μ(*NPU*)	*NPU*	LE	NR	*NPE*	E'(1:NPE)	$C_{E'}$(1:NPE)	$P_{E'}$(1:NPE)		
$E^{\mathrm{NSECT}}{}_1$	$E^{\mathrm{NSECT}}{}_{NE}$	NR	*NE*	AWP	LD	ZAP	YIELD			**NSECT Section**
$E^{\mathrm{NSECT}}{}_1$	C2	LMU	L2	NR	*NPU*	μ(1:NPU)	C_μ(1:NPU)	P_μ(1:NPU)	YIELD_1	
$E^{\mathrm{NSECT}}{}_2$	C2	LMU	L2	NR	*NPU*	μ(1:NPU)	C_μ(1:NPU)	P_μ(1:NPU)	YIELD_2	
			.							
			.							(E^{NSECT} , μ) Block
			.							Marginal CDF
$E^{\mathrm{NSECT}}{}_{NE}$	C2	LMU	L2	NR	*NPU*	μ(1:NPU)	C_μ(1:NPU)	P_μ(1:NPU)	$\mathrm{YIELD}_{\mathrm{NE}}$	
$E^{\mathrm{NSECT}}{}_1$	μ(1)	1	LE	NR	*NPE*	E'(1:NPE)	$C_{E'}$(1:NPE)	$P_{E'}$(1:NPE)		
$E^{\mathrm{NSECT}}{}_1$	μ(2)	2	LE	NR	*NPE*	E'(1:NPE)	$C_{E'}$(1:NPE)	$P_{E'}$(1:NPE)		
			.	.	.					($E^{\mathrm{NSECT}}{}_1$, μ , E') Block
			.	.	.					Conditional CDF
			.	.	.					
$E^{\mathrm{NSECT}}{}_1$	μ(*NPU*)	*NPU*	LE	NR	*NPE*	E'(1:NPE)	$C_{E'}$(1:NPE)	$P_{E'}$(1:NPE)		
							.			.
							.			.

							.		.
							.		.
E^{NSECT}_{NE}	**μ(1)**	**1**	**LE**	**NR**	***NPE***	E'**(1:NPE)**	$C_{E'}$**(1:NPE)**	$P_{E'}$**(1:NPE)**	
E^{NSECT}_{NE}	**μ(2)**	**2**	**LE**	**NR**	***NPE***	E'**(1:NPE)**	$C_{E'}$**(1:NPE)**	$P_{E'}$**(1:NPE)**	(E^{NSECT}_{NE} , μ , E') Block Conditional CDF
			.	.	.				
			.	.	.				
			.	.	.				**End NSECT Section**
E^{NSECT}_{NE}	**μ(*NPU*)**	***NPU***	**LE**	**NR**	***NPE***	E'**(1:NPE)**	$C_{E'}$**(1:NPE)**	$P_{E'}$**(1:NPE)**	

IZA	integer form of ZA number [integer]		
MB	block number (MB = 7) [integer]		
MT	reaction identifier [integer]		
NSECT	number of angle-energy distribution sections [integer]		
AWP	atomic weight [double]		
LD	flag to indicate discrete reaction [0=no;1=discrete]		
ZAP	particle charge [double]		
YIELD	total yield for this incident energy [double]		
LMU	flag for secondary angular distribution data [integer]		
	LMU	= 0:	secondary angle data provided in equiprobable cosine bins
		= 1:	secondary angle data provided in nonequiprobable cosine bins
LE	flag for secondary energy distribution data [integer]		
	LE	= 0:	secondary energy data provided in equiprobable bins
		= 1:	secondary energy data provided in nonequiprobable bins

E^n_i	ith incident energy point (eV) for the nth section [double]
C2	place holder for real quantity (typically 0.) [double]
TEMP	Temperature (K) [double]
L2	place holder for integer quantity (typically 0) [integer]
NR	place holder for integer quantity (typically 0) [integer]
NE	number of incident energy points [integer]
NPU	number of secondary μ values [integer]
NPE	number of secondary energy (E') values [integer]
μ(j)	jth secondary angle cosine [double]
C_μ(j)	cumulative probability for the jth secondary angle cosine bin [double]
P_μ(j)	value of the PDF for the jth secondary angle cosine [double]
Yield	Yield for the given incident energy
E'(j)	jth secondary energy point (eV) [double]
$C_{E'}$(j)	cumulative probability for the jth secondary energy bin [double]
$P_{E'}$(j)	value of the PDF for the jth secondary energy point or power interpolation parameter [double]

7.2.4.11 Probability table block

Probability tables are provided if the isotope has an unresolved-resonance region (URR). Note that the AMPX module PURM does not process multi-isotope nuclides with unresolved-resonance data. Consequently, a nuclide evaluation with unresolved-resonance data will not have probability table information; however, for practically all ENDF nuclide evaluations with unresolved data, there are corresponding individual isotope evaluations available in ENDF to construct the appropriate nuclide.

Table 12 provides the format for the probability tables for a particular isotope. This block exists in temperature-dependent files only. In the URR, a probability table is defined for a range of energies between E_i and E_{i+1}. Moreover, the probability table provides possible cross section values for the total, elastic scattering, fission and capture reactions in the unresolved range. The table is constructed based on the total cross section. Therefore, the values in the table define the probability for the total cross section value between Ei and Ei+1. During the random walk, the i^{th} bin is selected to obtain the total cross section, and the corresponding values of the elastic scattering, fission and capture reactions are also obtained from the i^{th} bin.

Note that the probability table block allocates four separate records for each reaction within a table. If the isotope is fissionable (LFI = 1), four separate reaction identifiers will be present in each table; however, for nonfissionable isotopes (LFI = 0), the fission cross section for each table is zero.

Based on the probability table structure, the format has a cross section band flag LBND that describes the structure of the table. If LBND is 0, the cross section bands are equiprobable, and if LBND is 1, the cross section bands are not equiprobable.

Table 13. Probability table structure for unresolved region

IZA	MB	0	ELR	EUR	SIG0	N1	NTAB			Header record
MT_1	$E_{1,1}$	$E_{1,2}$	$LTABLE_1$	LBND	NB	$\sigma_{MT}(i)$	$C_{MT}(i)$	$P_{MT}(i)$	$i=1,NB$	
MT_2	$E_{1,1}$	$E_{1,2}$	$LTABLE_1$	LBND	NB	$\sigma_{MT}(i)$	$C_{MT}(i)$	$P_{MT}(i)$	$i=1,NB$	Table 1
MT_3	$E_{1,1}$	$E_{1,2}$	$LTABLE_1$	LBND	NB	$\sigma_{MT}(i)$	$C_{MT}(i)$	$P_{MT}(i)$	$i=1,NB$	
MT_4	$E_{1,1}$	$E_{1,2}$	$LTABLE_1$	LBND	NB	$\sigma_{MT}(i)$	$C_{MT}(i)$	$P_{MT}(i)$	$i=1,NB$	
.										
.										
.										
MT_1	$E_{NTAB,1}$	$E_{NTAB,2}$	$LTABLE_{NTAB}$	LBND	NB	$\sigma_{MT}(i)$	$C_{MT}(i)$	$P_{MT}(i)$	$i=1,NB$	
MT_2	$E_{NTAB,1}$	$E_{NTAB,2}$	$LTABLE_{NTAB}$	LBND	NB	$\sigma_{MT}(i)$	$C_{MT}(i)$	$P_{MT}(i)$	$i=1,NB$	Table NTAB
MT_3	$E_{NTAB,1}$	$E_{NTAB,2}$	$LTABLE_{NTAB}$	LBND	NB	$\sigma_{MT}(i)$	$C_{MT}(i)$	$P_{MT}(i)$	$i=1,NB$	
MT_4	$E_{NTAB,1}$	$E_{NTAB,2}$	$LTABLE_{NTAB}$	LBND	NB	$\sigma_{MT}(i)$	$C_{MT}(i)$	$P_{MT}(i)$	$i=1,NB$	

IZA	integer form of ZA number [integer]
MB	block number (MB = 8) [integer]
ELR	threshold for URR (eV) [double]
EUR	upper boundary for URR (eV) [double]
L1	place holder for integer quantity (typically 0) [integer]
L2	place holder for integer quantity (typically 0) [integer]
N1	place holder for integer quantity (typically 0) [integer]
NTAB	number of probability tables in block [integer]
MT_i	reaction identifier [integer]
	MT_1 = 1: total
	MT_2 = 2: elastic scattering
	MT_3 = 18: fission
	MT_4 = 102: capture
$E_{n,1}$	lower energy bound for table n (eV) [double]
$E_{n,2}$	upper energy bound for table n (eV) [double]
$LTABLE_n$	probability table number corresponding to nth table [integer]
LBND	cross section band flag [integer]
	LBND = 0: cross section bands are equiprobable
	LBND = 1: cross section bands are not equiprobable
NR	place holder for integer quantity (typically 0) [integer]
NB	number of cross section bands [integer]
NBT(n)	dummy array that is not used for probability table data [integer]
INT(n)	dummy array that is not used for probability table data [integer]
$\sigma_{MT}(i)$	cross section value for reaction MT corresponding to the ith cross section band [double]
$C_{MT}(i)$	cumulative probability for reaction MT corresponding to the ith cross section band [double]
$P_{MT}(i)$	probability for reaction MT corresponding to the ith cross section band [double]

7.3 REACTION TYPE IDENTIFIERS

Reaction types in ENDF data are identified by integers called MT numbers. Within the AMPX system, the ENDF MT numbers are used where possible to identify the appropriate cross section data. AMPX processed data with no ENDF MT number counterpart are identified by integers outside the range of the ENDF MT numbers. Special values are listed in the following sections.

7.3.1 Multiplicity matrices

A MG library produced by the module X10 may contain several 1-D and 2-D data combining fission and multiplicity data. It is assumed that the number of source groups and sink groups is identical. For each pair of primary fission reaction f (mt=18, 19, …) and multiplicity $\overline{\nu}$(mt=452, 455, 456), the following 2-D and 1-D data are produced:

1. A scattering matrix using the appropriate kinematics distribution. It is multiplied by $f \times \overline{\nu}$
2. A normalized scattering matrix, which is the matrix produced in step, divided by $f \times \overline{\nu}$
3. A 1-D vector of the length of sink groups: $\chi_{isnk} = \sum_{isrc} a_{isrc,isnk} flux_{isrc}$, where the sum is over all source groups and $flux_{isrc}$ is the flux used for the given source group. The 1-D vector is normalized so that $\sum_{isnk} \chi_{isnk} = 1$.

The same MT value is used for items 2 and 3. Table 13 lists the MT values used for items 1-3:

Table 14. Total and fractional multiplicity MT values

Primary MT	Secondary MT	Distribution	Total matrix (Item 1)	Fraction matrix (Items 2 and 3)	1-D value for $f \times \overline{\nu}$
18	452	18	1452	1018	452
18	456	18	1456	1056	455
18	455	455 (if no present 18)	1455	1055	456
19	456	19 (if not present 18)	1419	1019	4561
20	456	20 (if not present 18)	1420	1020	4562
21	456	21 (if not present 18)	1421	1021	4563
38	456	38 (if not present 18)	1438	1038	4564

7.3.2 Additional Reaction values used in AMPX

The reaction types given in Appendix B of the ENDF manual [**Error! Bookmark not defined.**] are augmented where necessary to describe identifiers assigned to AMPX processed data. See Table 13 for definitions of χ values. In addition to the matrices and 1-D cross sections described in Sect. 7.3.1, the values given in Table 14 are used.

Table 15. Reaction numbers used in the AMPX code system

MT	Description
1-1000	Same as ENDF
1007	Thermal scattering matrix-may contain coherent and incoherent data
1008	Thermal scattering matrix for coherent data
1099	Group integral of the weighting function for neutron cross sections
1599	Group integral of the weighting function for gamma cross section
2000	Lambda factor
2022	Removal cross section
3002	Initially the same as MT=2, may be update during transport calculations
3018	Initially the same as MT=18, may be update during transport calculations
3102	Initially the same as MT=102, may be update during transport calculations

7.4 MISCELLEANEOUS USEFUL INPUT FILES

7.4.1 Print 1-D cross section data from AMPX master or working library

The module PALEALE will print detailed information concerning all data from an AMPX master or working library. However, it is useful to occasionally generate a list of (x,y) data for a selected cross section for plotting.

```
=shell

cp  /scale/scale6.svn/data/scale.rev06.xn238v7 ft31f001

end

=tabasco

0$$  31  32  e

1$$  1  e t

2$$  92235  e

3$$  102  e t

end

=charmin

input=32 output=33 single to plot

end

=shell

cp ft33f001 ${RTNDIR}/u235_capture

end
```

The above input selects the capture cross section (MT=102) for ^{235}U (ID=92235) from an AMPX master library and prints it as a list of (x,y) histogram values in file u235_capture.

7.4.2 Convert (x,y) data into a weighting function file

In order to create MG libraries, a flux file is needed. If a custom flux is used in the form of (x,y) pairs, the following input can be used to convert to a TAB1 formatted file that can be used in the sequences (please note that the flux needs to be per unit of energy, not per unit of lethargy):

```
=shell

cp ${RTNDIR}/weight.dat ft30f001

end

=charmin

in=30 out=10 fidas to double

end

=shell

cp ft10f001 ${RTNDIR}/weighting

end
```

The file weight.dat contains the weighting function in FIDO format, as shown below:

```
2## e

2$$ 9.900000E+01 3.000000E+00 2.099000E+03 0 0 0 0 1 n e

t

2## e

2$$ 9.900000E+01 3.000000E+00 2.099000E+03 0 0 0 0 1 n e

t

3$$ n 2.000000E+00

4##

 1.000000E-05 0.000000E+00

 3.162280E-05 1.458894E-02

 1.000000E-04 2.551780E-02

 1.019370E-04 2.599223E-02

 1.039120E-04 2.824284E-02

….

Nth data point

t

2$$ 9.900000E+01 3.000000E+00 0

    0.000000E+00 0.000000E+00 0.000000E+00 0 0 0

2## a4 0.000000E+00 0 e t

2$$ 9.900000E+01 0.000000E+00 0
```

```
   0.000000E+00 0.000000E+00 0.000000E+00 0 0 0
2## a4 0.000000E+00 0 e t
2$$ 0.000000E+00 0.000000E+00 0
    0.000000E+00 0.000000E+00 0.000000E+00 0 0 0
2## a4 0.000000E+00 0 e t
```

Where n is the number of data points desired. The first pair of 2## and 2$$ arrays gives the integer and floating points values of the first control record. The first three numbers are the MAT, MF and MT values. The neutron sequences expect MAT=99 and MT=2099, the gamma sequences expect MAT=99 and MT=1599. The last two numbers simply give the number of interpolation regions (1) and the number of data points (n). The pair of 2## and 2$$ arrays are repeated to conform to the definition of a data block in File 3 in the ENDF-102 standard. The 3$$ array lists the interpolation table, which in this case is linear-linear for all data points. The 4## array lists the data points. The final block after the customary “t” ending of fidas input simply lists the SEND and FEND control records.

7.5 INTEGRATION ROUTINES IN AMPX6

In constructing MG cross sections, functions or products of two or more functions must frequently be integrated. Some applications, such as that of generating Bondarenko factors (See the Sect. 6.8 on the FABULOUS module), involve considerably more complicated expressions and functions to be simultaneously treated. Over the development of AMPX, integration routines have evolved. Several integration routines support all interpolation codes allowed in the ENDF-102 standard, but it is generally expected that the point-wise data are given with linear-linear interpolation. While some of the older methods for integration are still in use, all the new modules use one of four methods:

1. The Tab1 class provides routines to integrate one-dimensional data.

2. A general class *NumIntegrate* uses a Fifth-order Runge-Kutta method with adaptive step-size as described in Numerical Recipes [27]. The user must extend this class to provide the function that needs to be integrated.

3. *IntegrateCross* is used to generate 1-D group-averaged cross section data. It assumes that the point-wise data use linear-linear interpolation and forms a union grid, and then it solves the integral analytical as a product of two or three straight lines.

4. *IntegrateMatrix* is the class used to generate MG scattering matrices. It uses a combination of option 1 and 2 in conjunction with a unit-based interpolation to calculate the scattering matrix.

Some older AMPX modules use other integration routines; these will be phased out as the modules are modernized and converted to C++.

8. INPUT FILE GENERATION

8.1 AUTOMATIC INPUT FILE GENERATION

It is frequently desired to create a MG or CE library based on a selected set of ENDF/B formatted files. The auxiliary AMPX code ExSite greatly helps with the task of preparing the input files. This section focuses on the background of input file generation, while the use of ExSite is described in Sect. 4.

ExSite first parses the desired ENDF/B formatted file and generates an abbreviated list of information pertinent to library creation. The list is stored in XML format to allow easy human and machine readability. The list is used in conjunction with templates that contain the input descriptions for the AMPX modules necessary to generate the desired library. In general, more than one template is needed to generate the final library. The next section describes the creation of the abbreviated ENDF list in more detail. The format of the templates is described later.

8.1.1 ENDF listing

The templates used to generate AMPX input files require a list of ENDF information which is assumed to be in xml format and must contain a root element named `Materials`. The information is automatically extracted from ENDF/B formatted files. The given file can contain one or more evaluations. For each evaluation, an XML element named `Material` is added to the XML file. The element contains attribute/value pairs for every piece of information extracted from the ENDF evaluation. The available attributes are listed in Table 15. Since an extensible XML format is used, more attribute value pairs may be added in the future.

Table 16. Field names automatically extracted from the ENDF evaluation

Name	Description
tape	The full name of the file containing the evaluation
filename	The name (without the path) of the file containing the evaluation
endf	The endf material number
za	The standard material charge number Retrieved from C1 position in first record of File 1
awr	The mass ratio of the material Retrieved from C2 position in first record of File 1
dbrcnuclide	Set to “yes” if a >= 200.
tag	A name that uniquely describes the material Constructed from columns 1–11 of the fifth record of File 1 (Denoted ZYNAM in ENDF manual)
mod	The ENDF evaluation modification number. Retrieved from N2 position in first record of File 1 (Denoted NMOD in ENDF manual)
rel	The ENDF evaluation release number. Retrieved from L1 position in third record of File 1 (Denoted LREL in ENDF manual)
nlib	The library source (ENDF, JEFF, etc.)
eval	The date of the ENDF evaluation Retrieved from columns 23–32 of the fifth record of File 1 (Denoted EDATE in ENDF manual)

Table 15. Field names automatically extracted from the ENDF evaluation (continued)

Name	Description
awi	Mass ratio of the incident particle
zai	ZA value of the incident particle
neutron	Set to “yes” if incident particle is a neutron. Otherwise set to “no” Based on C1 value in third record of File 1 (Denoted AWI in ENDF manual)
gamma	Set to “yes” if incident particle is a gamma. Otherwise set to “no” Based on C1 value in third record of File 1 (Denoted AWI in ENDF manual)
author	The name of the authors of the ENDF evaluation Retrieved from columns 34–66 of the fifth record of File 1 (Denoted AUTH in ENDF manual)
lab	The name of the originating laboratory Retrieved from columns 12–22 of the fifth record of File 1 (Denoted ALAB in ENDF manual)
dist	The original distribution date of the ENDF evaluation Retrieved from columns 23–32 of the sixth record of File 1 (Denoted DDATE in ENDF manual)
rdate	Date of the last revision of the ENDF evaluation Retrieved from columns 34–43 of the sixth record of File 1 (Denoted DDATE in ENDF manual)
rev	The version number of the library Retrieved from the N2 value of the third record of File 1 (Denoted NVER in ENDF manual)
metaStable	Set to “true” if the material is metastable, otherwise set to “false” Determined from columns 1–11 of the fifth record of File 1 (Denoted ZYNAM in ENDF manual)
version	Library format Retrieved from N2 position in second record of File 1 (Denoted NFOR in ENDF manual)
lis	State number of the target nucleus. Retrieved from L2 position in second record of File 1
fission	Set to “yes” if File 1 contains any of the following reaction: 452, 455, 456, 458
file2	Set to “yes” if the evaluation contains a File 2
nis	Contains the number of isotopes given in File 2
resonance	If File 2 is present and contains resonance parameters for the resolved resonance rangeIt is set to: • “SLBW” if the resonance parameters are given in Single Level Breit Wigner format. (ENDF parameters LRF=1) • “MLBW” if the resonance parameters are given in Multi Level Breit Wigner format. (ENDF parameters LRF=2) • “RM” if the resonance parameters are given in Reich-Moore format. (ENDF parameters LRF=3) • “AA” if the resonance parameters are given in Adler-Adler format. (ENDF parameters LRF=4) • “RML” if the resonance parameters are given in R-Matrix Limited format. (ENDF parameters LRF=7) If more than one resolved resonance region is given for the evaluation, the value reflects the format for the last resonance region given.

Table 15. Field names automatically extracted from the ENDF evaluation (continued)

Name	Description
res	If File 2 is present and contains resonance parameters for one or more resolved resonance regions the value is set to “yes”
unres	If File 2 is present and contains resonance parameters for one or more unresolved resonance regions the value is set to “yes”
scattering	If File 2 is present set to the potential scattering cross section. It is determined from the first resolved resonance range for each isotope given in the evaluation as follows (if the file only contains an unresolved resonance range, the radius given in that section is used): 1. Set the scattering radius ap to AP (see ENDF manual) if NRO = 0. If NRO > 0, set ap to the first radius given in the table of energy radius pair. If LRF=3 and ap=0.0, select first APL (see ENDF manual) 2. If only an unresolved region is present, select AP from it 3. If LRF=7 select APE from the first group of J^{π} values and the second channel 4. Calculate the potential scattering cross section as: $12.56637061 \times ap^2$ 5. If more than one isotope is present, add the cross section data calculated in step 2 according to the abundance of each isotope
file3	Set to “yes” if the evaluation contains a File 3
file4	Set to “yes” if the evaluation contains a File 4
file5	Set to “yes” if the evaluation contains a File 5
totalFission	Set to “yes” if the evaluation contains a File 5 and File 5 contains a section for MT=18 (total fission)
partialFission	Set to “yes” if the evaluation contains a File 5 and File 5 contains a section for MT=19, MT=20, MT=21 or MT=38 (second chance fission)
file6	Set to “yes” if the evaluation contains a File 6
AWP0	Set to “yes” if File 6 contains photon production yield matrices. This is true if any of the sections has a mass ratio of 0 for the product. In addition if LAW=2, the mass ratio can contain the gamma energy if the charge for the outgoing particle is 0
file7	Set to “yes” if the evaluation contains a File 7
file7temps	If File 7 exists, it contains the list of temperatures at which the moderator data were evaluated
file12	Set to “yes” if the evaluation contains a File 12
file13	Set to “yes” if the evaluation contains a File 13
file23	Set to “yes” if the evaluation contains a File 23
covariance	Set to “yes” if the evaluation contains File 31, File 32 or File 33
chicov	Set to “yes” if the evaluation contains File 35

The XML list containing the parameters automatically extracted from the ENDF evaluation is not sufficient to generate the input files. An additional XML file is used containing information concerning metastable nuclei and thermal moderator data. This configuration file contains a root element `ConfigFile` and three sections. If parsing one or more ENDF formatted files, a configuration file is automatically created with some default values. **The values should be reviewed before actually using the configuration file.** In the final library, the nuclei are normally identified by their ZA value and not by the ENDF material number. This makes it necessary to use a different ID in some cases in order to avoid duplication.

8.1.1.1 Special nuclei

The correct ZA value can be easily retrieved from the ENDF evaluation itself. However, it may be desired to give a different identifier to some nuclei. The section is contained in an element with name `specialNuclei`. The element contains one or more `nuclei` elements. The `nuclei` element has the following attributes:

- `endf` contains the ENDF MAT number of the evaluation for which the final library ID should be changed.
- `realza` contains the actual ZA value of the nucleus
- `scaleza` contains the ID to be used in the final library
- `name` label used in the SCALE standard composition library

By default, the section contains new ID value for H1 (changed to 8001001) and H2 (changed to 8001002).

8.1.1.2 Meta stable nuclei

Since the ZA value for meta-stable nuclei is the same as for the stable form, the ID in the final library needs to be changed in order to avoid duplicate entries with the same ID value. The section is contained in an element named `metastable`, which contains one or more `nuclei` elements. The `nuclei` element as three attributes:

- `endf` contains the ENDF MAT number of the evaluation for which the final library ID should be changed
- `realza` contains the actual ZA value of the nucleus
- `scaleza` contains the ID to be used in the final library

By default, an entry is created for each meta-stable nuclei contained on the ENDF tape being parsed. The new ID is set to LISO*1000000+ZA

8.1.1.3 Thermal moderators

In the creation of the final library, thermal moderators must be treated separately, as they only contain data in the thermal range. The data must be combined with one or more evaluation in the fast range. In addition the final ID should depend on the thermal data, as well as the fast data connected to it. The section is contained in an element named `thermal`, which contains one or more `nuclei` elements. Each `nuclei` element has two attributes:

- `endf` contains the ENDF MAT number of the evaluation for which the final library ID should be changed
- `realza` contains the actual ZA value of the nucleus

In addition, each `nuclei` element has one or more `fastMat` elements listing the evaluations to be used in the fast region. Each `fastMat` element contains the following attributes:

- `endf` contains the ENDF MAT number evaluation to be used in the fast region

- `scaleza` contains the ID to be used in the final library for the combined thermal and fast data
- `name` specifies the name used in the SCALE standard composition library

An evaluation is recognized as a thermal moderator if File 7 is present. In this case, an entry is automatically added. Table 16 lists the values that are automatically selected. Please note that these are the material numbers as given in the ENDF manual. Some of the actual material numbers used in ENDF/VI and ENDF/VII differ from this designation. Please review the automatically generated configuration file for consistency. Comments containing the tag for each thermal and fast evaluation are included in the configuration file and should help in this task.

Table 17. Thermal material numbers recognized (See ENDF manual for detail)

MAT number	Description	Fast evaluation	SCALE ID
1	Water	Bound with H1	1001
2	Para Hydrogen	Bound with H1	5001001
3	Ortho Hydrogen	Bound with H1	4001001
7	H in ZrH	Bound with H1	7001001
11	Heavy Water	Bound with H2	1002
11	Para Deteurium	Bound with H2	5001002
13	Ortho Deuterium	Bound with H2	4001002
26	Be	Bound with Be9	3004009
27	BeO	Bound with Be9	5004009
		Bound with O16	5008016
28	Be_2C	Bound with Be9	7004009
		Bound with C	7006000
29	Be in BeO	Bound with Be9	5004009
31	Graphite	Bound with C	3006000
33	l-Methane	Bound with H1	7001001
34	s-Methane	Bound with H1	2001001
37	Polyethylene	Bound with H1	9001001
40	Benzene	Bound with h1	6001001
		Bound with C	5006000
46	O in BeO	Bound with O16	5008016
58	Zr in ZrH	Bound with Zr	1040000
		Bound with Zr90	1040090
		Bound with Zr91	1040091
		…	
75	UO_2	Bound with O16	7008016
		Bound with U232	1092232
		Bound with U233	1092233
		…	
76	UC	Bound with C	8006001
		Bound with U232	1092232
		Bound with U233	1092233

If the ENDF material number for a thermal moderator is not found in Table 16, it is bound with an evaluation with the same Z and A value. This may not lead to the desired result if the ZA value is not set to a realistic value in ENDF.

8.1.2 Templates

In order to generate input files, the XML lists described previously are combined with templates in XML format. The templates are usually part of a user template used in ExSite to automatically generate input files. This section describes the use of the template only.

The template is an XML file containing elements instructing the parser on how to interpret the input. The main branching elements are:

- `openFile` with attribute `name`, opens a file for writing. The `name` attribute contains the name of the file. Output will only be written to files that have previously been opened. This element can appear anywhere in the template. The element can contain an attribute `newInput` with values yes or no. If the value is no, the output file is opened in append mode, otherwise a new file is created.

- `closeFile` with attribute `name`, closes a previously opened file. The `name` attribute contains the name of the file. This element can appear anywhere in the template.

- `writeFile` with attribute `name`, opens a file for writing. The `name` attribute contains the name of the file. The only children allowed are XML CDATA sections and `text` elements (see below for more information). Any information contained in one of these child elements will be written into the output file. This element can appear anywhere in the template. Please note that any `writeFile` element needs to be closed before starting or closing any other element but a `text` element or a CDATA section.

- `text` contains text to be written to the output file. It may contain an attribute `restrict`. Please see below for more information. This element can only appear inside a `writeFile` element.

- `loop` start a loop over all evaluations. It may contain an attribute `restrict`. Please see below for more information. Looping elements may be nested. The element can appear anywhere in the templates except inside a `writeFile` element.

- `last` is an element that can appears anywhere in a `loop` element. It is assumed that the element contains one or more `writeFile` elements which will be written to the file only if this is the last evaluation to be processed in the parent `loop` element.

- `first` is an element similar to the last element except that its content is only used if the first evaluation of the parent `loop` element is processed.

- `notLast` is an element that can appear anywhere in a `loop` element. It is assumed that the element contains one or more `writeFile` elements which will be written to the file for anything but the last evaluation to be processed in the parent `loop` element.

- `notFirst` is an element similar to the `notLast` element, except it is processed for anything but the first evaluation to be processed in the parent `loop` element.

- `arrayLoop` with attribute `value` is a special loop that loops over all values listed in the `value` attribute. Values are assumed to be separated by space. This loop type cannot be nested and cannot contain any `last`, `first`, `notLast` or `notFirst` elements.

- `thermalLoop` loops over all fast evaluations that may be bound with the current thermal moderator. If the evaluation is not a thermal moderator, this loop will be ignored. This loop type cannot be nested and cannot contain any `last`, `first`, `notLast` or `notFirst` elements.

- `exclude` with attribute `run`. If the value of `run` is `no`, this section of the template is ignored.

Inside a text element or in any attribute value, the function *writeData()* will be substituted by values retrieved from the XML file listing the abbreviated ENDF information. The function *writeData* may have several arguments separated by commas. The first argument is always a reference to the field to be written, and the last two arguments are always optional and contain the length of the field and the fill character. If the first argument is any of the attribute names listed in Table 15, the value of that argument will be substituted. For example if processing ^{232}U, then *writeData(endf)* will be substituted by 9219 and *writeData(endf,6,*)* will be substituted by **9219. In addition to the values listed in Table 15, the values listed in Table 17 can also be used.

Table 18. Additional substitution values that can be used in templates

z	The atomic number of the current evaluation
a	The mass number of the current evaluation
chem	The chemical name of the current evaluation
source	The source of the library (endf, jeff, etc.). This is just an alias for nlib.
library	A string representation of source which is used in some filenames
scaleid	A special ID value used in the library, calculated as: **endf + 10000 * rel + 1000000 * rev** (See Table 15 for definition of **endf**, **rel** and **rev**.)
changedza	The ZA value of the ID or the ID designated in the configuration file
thermalscaleid	The same as scaleid except calculated with the endf, rel and rev number for the thermal moderator (If requesting scaleid for a thermal moderator, the output will be the scaleid for the first fast evaluation bound to the moderator.)
thermalEvals	The number of evaluations to be used in the fast region if the current evaluation is a thermal moderator (In all other cases it is 0.)
date	The current date (The function can contain a second argument giving the desired format for the date.)
loopindex	This substitution value that can only appear inside a loop element (It is the index of the element in the XML listing currently being processed. The element can have an additional argument giving the offset. For example, if the current index is 2 and the offset is 60, the value 62 is printed.)
loopnumber	A substitution value that can only appear inside a loop element, equal to the number of times the content of the loop will be executed. (The element can have an additional argument giving the offset.)
loopcount	A substitution value that can only appear inside a loop element; the number of times the loop is being processed, including the current execution. (This is different from loopindex if the

	loop has a restrict attribute. The element can have an additional argument giving the offset.)
array_value	A substitution value that can only appear inside an arrayLoop element; substituted by the current value of the loop variable
array_number	A substitution value that can only appear inside an arrayLoop element; substituted by the current index of the array loop (The function can have an additional argument given the offset to be added to the index. For example, if the current index is 2 and the offset is 60, the value 62 is printed.)
array_length	A substitution value for the number of items in the value of the element listed as the second argument.

In the case of a thermal moderator, the values for the fast evaluations to be bound with the moderator are often needed in the input file. Therefore, each of the function arguments listed in Tables 15 and 17 can have a postfix of “fast1,” “fast2” . . . . If this postfix is found, the corresponding value is retrieved not from the current evaluation but from the first, second . . . fast evaluation to be bound with this thermal moderator. If the current evaluation is not a thermal moderator, the postfix is ignored, and the value for the current evaluation is substituted. The field changedza can have postfix values of thermal1, thermal2 . . . in order to retrieve the value to use for the ID in the final library (changedzafast1, changedzafast2. . . would retrieve the ZA value of the evaluation used in the fast region). As the number of fast evaluations to be bound with a thermal moderator is not known while writing the template, the loop element `thermalLoop` is provided. In this loop, each argument can have a further postfix “_thermNum” which will be substituted by 1,2 . . . until all fast evaluations have been processed. For example, to list the endf mat number and the new scale ID for all fast evaluations of a given thermal moderator, the following snippet is used:

```
<writeFile name="InputFileName">

<text>For thermal moderator writeData(tag) with mat=writeData(endf) use

</text>

</writeFile>

<thermalLoop>

    <writeFile name="InputFileName">

<text>      Fast: writeData(tagfast_thermalNum) with endf
mat=writeData(endffast_thermalNum) and new id of writeData(changedzafast_thermalNum)

</text>

   </writeFile>

</thermalLoop>
```

If preparing a coupled library, it is necessary to find a gamma evaluation associated with the current neutron evaluation. If the XML list contains neutron evaluations and gamma evaluations, the template parser will attempt to pair the two together based on the Z value. An evaluation is recognized as a gamma evaluation if the incident particle is a gamma and the evaluation contains File 23 data. An evaluation is recognized as a neutron evaluation if the incident particle is a neutron and the evaluation contains File 3 data. If an associated gamma evaluation is found, all the tags listed in Tables 15 and 17 can have a postfix “gamma.” If the postfix is found, the corresponding value is retrieved not from the current evaluation, but from the associated gamma evaluation. If the current evaluation is a thermal moderator, then the gamma evaluation is associated with the evaluations to be used in the fast region. In this case, a postfix of "gammathermal1," "gammathermal2" . . . must be used.

The `loop` and the `text` element take an optional attribute `restrict` that selects only certain evaluations. The `restrict` attribute value is a comma-separated list of restrictions to be applied. If a restriction is prefixed with a "-" then the `loop` or `text` will only be executed for evaluations that do not fulfill the requested property. If a restriction is prefixed with a "+" then the `loop` or `text` will only be executed for evaluation that do fulfill the requested property. The "+" prefix is only needed if one or more restrictions are given in a restrict argument; otherwise, "+" is assumed. If more than one restriction is given and all restrictions are prefixed with a "+" then the body will only be executed for evaluations that fulfill **ALL** of the restrictions: i.e., a logical and is assumed; otherwise a logical or is assumed. A restriction can be constructed from all of the tags given in Tables 15 and 17. The postfix operators for thermal moderators and associated gamma evaluations are also allowed. The restriction is created by giving the desired tag followed by the desired value. For example "+file3(yes) +neutron(yes)" selects all neutron evaluations excluding thermal moderators. In addition, the following shortcut restrictions are also allowed:

- `gamma` restricts to neutron evaluations that have an associated gamma evaluation
- `thermal` restricts to thermal moderator evaluations
- `metastable` restricts to meta stable nuclei
- special restricts to nuclei for which the library ID is changed (These are the nuclei listed in the `specialNuclei` section of the configuration file.)
- `fastforthermal` restricts to nuclei that are used as a fast evaluation for any of the thermal moderators in the listing

Examples describing custom templates for ExSite are provided later in this section.

8.2 EXSITE FILES

ExSite is used to generate the XML listings described above and to process the templates. It also serves as a GUI to read and create AMPX input files. The user input for the various AMPX modules and the templates is described in XML formatted files. The XML files for the module input are created from input descriptions in the module source files. The user template input is usually much simpler than for the module input and is created by hand. This section describes the xml formats used by ExSite, concluding with some examples of custom template files. Instructions on how to run ExSite are given in Sect. 4.

Module descriptions are given in an XML file with elements for each input parameter. The input parameters are subdivided into groups, where the parent group is the module itself. The input parameters are described in an element giving the name of the parameter. All input parameters can have the attributes listed in Table 18. Additional attributes may be available depending on the type. Each input element also can contain the elements listed in Table 19. Additional elements may be needed depending on the type of the input parameter.

Table 19. Attributes available for input parameters

Name	Required	Possible choices	Description
type	yes	listed below	The type of the variable
required	no	yes no	Indicates whether this language element is required *Default is no.*
keyword	no	yes no	Indicates whether a keyword is required to start the parameter, as in **keyword=value** *Default is no.*
nameUsed	no		Value to use instead of the name if the parameters are **keyword=value** (Required when spaces are needed in the keyword; the attribute can contain one or more values in a space-separated list, and in this case, all values in that list are recognized as valid keywords.)
attachMeaning_*	no		If **nameUsed** supplies a space-separated list of keywords, attaches different descriptions to each of these keywords (If desired, a corresponding attribute **attachMeaning_keyword** must appear for each keyword listed in **nameUsed**)
keywordSeparator	no		The separator between **keyword=value** *Default is =*
caseSensitive	no	yes no	The comparison for keywords to be entered as case sensitive. *Default is no.*
before	no	Name of input	If this input object is used, it must appear before the indicated input object. The indicated input object must be a required one.
after	no	Name of input	If this input object is used, it must appear after the indicated input object. The indicated input object must be a required one
appendEol	no	yes no	Used when the input must end in an end of line *Default is no.*
hide	no	yes no	Used when the indicated input must be hidden from the user *Default is no.*
deprecated	no	yes no	Used when the parameters are deprecated (used in the text but not in the GUI) *Default is no.*
depends	no		Name of the input value this input depends on (The value can contain +, -, *, and /. In this case it is assumed that the input parameters contain numbers combined to calculate the final value.

Table 18. Attributes available for all input parameter descriptions (continued)

Name	Required	Possible choices	Description
compValue	no		Value used to compare with resulting value if **depends** is set
compare	no	eq ne gt lt	Used to compare the value of the dependent input object to **compValue**: eq: compares **for equality** ne: compares **that not equal** gt: converts **compValue** and **value** of the dependent input object to a float and compares **as greater than** lt: converts **compValue** and **value** of the dependent input object to a float and compares **as less than**
postfix	no		A postfix to be added after the value
repeat	no	exsite_unlimited	If **exsite_unlimited**, the block can be repeated any number of times (If a number, gives the number of times this block repeats. Otherwise it is the name of a valid input object. The value of this input object is converted to a number which gives the number of times this block repeats. The value can contain +, -, *, and /. In this case, it is assumed that the input parameters contain numbers combined to calculate the final value.)
repeatIsUpper	no	yes no	Indicates that the value given in **repeat** is an upper limit only
repeatAtLeastOnce	no	yes no	Indicates that the value given in **repeat** is an upper limit only, but the value must be given at least once
lineLength	no		The allowed length of line. If -1, no line length limit is assumed. Inherited from parent if not set. *Default is -1.*
allowComments	no	yes no	Allows comments in the coding (Inherited from parent if not set.) *Default is yes.*
groupTag	no		A name that groups a number of input elements into a logical group; used to display the input parameters together in the GUI
listPanel	no	yes no	Determines how the input parameter is displayed in the GUI if it is a group of input parameters
automaticEnable	no	yes no	If this input parameter depends on the value of another parameter, it will be unchecked in the GUI. If this attribute is set to "yes," then the input parameter will be selected in the GUI as soon as it is applicable

Table 20. Child elements available for all input descriptions.

Name	Required	Description
exsite_description	no	Relative path to an html description of the input object
exsite_default	no	Default value for this input object (Value must be valid in the context of the type attribute.)
exsite_required_position	no	Used when input object is required in a certain position (Position counting starts from 0.)
exsite_summary_line	no	A short description used in the GUI
exsite_icon_path	no	Relative path to an icon that can be used to indicate this input object in the GUI

The following values are allowed for the type attribute:

exsite_string: The input object described by this tag is assumed to be a string. It takes two additional attributes:

- spaceAllowed, which can have values of yes and no and indicates whether the string can contain spaces
- equalAllowed, which can have values of yes and no and indicates whether the string can contain equal signs

exsite_flag: The input object describes a Boolean value. If the keyword is present, the Boolean value is set to *true*, otherwise to *false*. No additional attributes or elements are available for this type.

exsite_float/exsite_integer: The input object describes a float or an integer object. No additional attributes are available. Additional elements available are:

- exsite_max_required, which gives the upper bound for the number
- exsite_min_required, which gives the lower bound for the number

exsite_key: The input object describes a bare keyword that has different meanings depending on its value. It is different from exsite_flag since the result value must preserve the actual value given by the user.

exsite_boolean: An input object that describes a Boolean value. The values *yes* or *true* are both recognized as setting the value to *true*. The values *no* or *false* both set the value to *false*. All other values are not accepted.

exsite_file: This input object is similar to exsite_string; however, it denotes that this object is a file. The GUI can then display this object with a file selection box. The object has several additional attributes:

- spaceAllowed, which can have values of *yes* and *no*; indicates whether the filename can contain spaces.
- isNewFile, which can have values of *yes* and *no*; indicates whether the input refers to an existing file or whether it will create a new file. Based on this information, the GUI can determine whether to display a load or save dialog.

- substituteDrive, which can have values of *yes* and *no*, and only takes effect if the underlying operating system is a windows system. If set to *yes*, then the drive letter of the default value is changed to the same drive where the Exsite program resides.

exsite_enum: The input describes an enumeration type of input parameter. It takes one additional optional attribute and several required elements. The additional attribute is allowUser, and it takes a value of *yes* or *no*. If true, users are allowed to add values in addition to choosing one of the predefined options. The required elements are:

- enum_type with attribute type. The value of type defines which types of values are allowed for the enumeration.

- One or more exsite_enum_option elements listing the available choices. The element contains one additional element named exsite_description, which contains the description of this option. The remaining text is the value of the option.

exsite_array: An input type describing an array of data. The element contains several optional attributes and one required element. The attributes are:

- length giving the number of elements in the array. The value can take the same arguments as the repeat attribute listed in Table 18. The default is exsite_unlimited.

- arrayStart character string indicating the start of array values. A value of “\\n” will be substituted by an end of line character. The default is to use no start value.

- Separator gives a list of characters that separate array elements. The value “\\n” is substituted by a new line character, “\\t” by a tab character, and “\\r” by a hard return character.

- arrayEnd is the same as arrayStart, except it denotes the end of array values. The default is to use no end value. The array terminates after the first character string that cannot be parsed as an array element.

- arrayMarkeRequired is used if arrayEnd and/or arrayStart are given but are not required.

- lengthAbsolute uses the absolute value stored in length to calculate the desired length of the array.

In addition, the element takes one required element, array_type with required attribute type to indicate the type of data to be stored in the array.

exsite_close_depend: An input value that can be used inside a group input element. It is used if group is closed by a certain character string. In some special cases, the occurrence of this character string does not indicate the end of the group input. This is a marker input not displayed in the GUI and depends on the attribute values of the group of which it is a part. This element takes one required attribute endBlockValue, giving the character string that can potentially end the group input.

exisite_group: an input element that groups together other input elements. The available attributes are listed in Table 20.

exsite_fidas: This is a special case of exsite_group for use with fidas type input. It describes a fidas array. The readBlockValue and readBlock attributes need to be explicitly given. Like other exsite_group elements, it contains input elements describing the various entries into a fidas array. The input elements can be any of the elements describing numbers of the correct type of the fidas array.

Table 21. Attributes available for exsite_group element

Name	Required	Possible Values	Description
endBlock	no	yes no closeOnRequired	Indicates whether the end of the group input is indicated by a certain combination of characters If **closeOnRequired i**s selected, then the group input is closed if all required input elements have been given. *Default is no.*
readBlock	no	yes no	Indicates whether the start of the group is input, as indicated by a certain combination of characters. *Default is no.*
canFold	no	yes no	If set to “yes.” then the group input can be folded into the GUI. *Default is no.*
readBlockValue*	no		If **readBlock** indicates that a certain character combination starts the group, this attribute gives that character combination. The value \\n is substituted by a new line character. There can be more than one character string that opens the group. If so, they must be supplied in more than one instance of readBlockValue. Since XML only allows unique attribute names, substitute * with any value to make the attribute name unique. *The default is read <groupName>.*
endBlockValue*	no		If endBlock indicates that a certain character combination ends the group, this attribute gives that character combination. The value \\n is substituted by a new line character. There can be more than one character string that ends the group. If so, they must be supplied n more than one instance of endBlockValue. Since XML only allows unique attribute names, substitute * with any value to make the attribute name unique. *The default is end <groupName>.*
matchTag	no	yes no	If there is more than one value for readBlockValue and endBlockBalue, then ensure that matching pairs are used. Pairs are generated by order of appearance in the xml file. *Default is no.*

Table 20. Attributes available for exsite_group element (continued)

Name	Required	Possible Values	Description
xmlOrder	no	yes no	If set to "true," the order of the elements in the input file must be the same order as given in the xml description. *Default is no.*
discardBlockEnd	no	yes no	If set to "true," the group is terminated after the endBlockValue is read. However, the character string is not consumed and thus is available for parsing for the next input element. *Default is no.*
noEmptyOnDiscard	no	yes no	If set to "true," a group can consist of no input elements in the input file. Otherwise, an empty group is flagged as an error. *Default is yes.*
closeDepends	no		The name of the field is defined here if the endBlockValue is present but does not indicate the end of the group unless a certain field is set (The parser will automatically create a field of type exsite_close_depend. This input type needs to be defined in the xml input description.)
closeDependCompValue	no		If closeDepends is set, the value of the field given in closeDepends that keeps the group open
closeDependCompare	no		If closeDepends is set, the means to compare the field

8.2.1.1 Examples

The following example is the input for the compare module, which uses FIDO style input:

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

<compare type="exsite_group" endBlock="yes" endBlockValue="\n end"

         lineLength="72" displayMode="page">

    <exsite_summary_line>MODULE TO COMPARE FUNCTIONS ON TWO TAB1 FILES

    </exsite_summary_line>

    <centrm appendEol="true" type="exsite_string" spaceAllowed="yes"

            required="yes" hide="true">

        <exsite_summary_line>Centrm parameters</exsite_summary_line>

        <exsite_required_position>0</exsite_required_position>
```

```
</centrm>

<x_1 type="exsite_group" endBlock="yes" endBlockValue="t \n"

     xmlOrder="yes" displayMode="page" required="yes">

    <exsite_required_position>1</exsite_required_position>

    <Core_Allocation type="exsite_fidas" nameUsed="Core Allocation"

                     displayMode="page" readBlock="yes"

                     noEmptyOnDiscard="yes" readBlockValue="-1$$">

        <ICORE hide="yes" type="exsite_integer" nameUsed="ICORE ">

            <exsite_summary_line>Number of words of

                   core to allocate.</exsite_summary_line>

            <exsite_default>500000</exsite_default>

        </ICORE>

    </Core_Allocation>

    <Logical_Unit_Assignments type="exsite_fidas"

            nameUsed="Logical Unit Assignments"

            displayMode="page" readBlock="yes"

            noEmptyOnDiscard="yes" readBlockValue="0$$">

        <LOG1 type="exsite_integer" nameUsed="LOG1 ">

            <exsite_summary_line>Logical unit on which the first

                  TAB1 File is located&lt</exsite_summary_line>

            <exsite_default>1</exsite_default>

        </LOG1>

        <LOG2 type="exsite_integer" nameUsed="LOG2 ">

            <exsite_summary_line>Logical unit on which the second

                  TAB1 File is located</exsite_summary_line>

               <exsite_default>2</exsite_default>

        </LOG2>

        <LOG3 type="exsite_integer" nameUsed="LOG3 ">

            <exsite_summary_line>Logical unit where the

                   difference TAB1 File </exsite_summary_line>

            <exsite_default>3</exsite_default>

        </LOG3>

    </Logical_Unit_Assignments>
```

```
        </x_1>

    </compare>
```

The element compare gives the name of the module and indicates that an end block is required, and it consists of end on the start of a new line. The centrm element is required for almost all modules. It indicates that all input following the name of the module until the end of the line is to be read into the input element centrm and to be hidden from the user. The input consists of one block of data terminated by the customary "t" at the end of the fidas block. This is defined in the element named x_1. The input consists of one fidas array (-1$$) which is described in the element named Core_Allocation. It indicates that the fidas array start after the character string "-1$$" is found. The fidas array group closes automatically if all children have been given. The attribute value for noEmptyOnDiscard indicates that the group fidas array can be omitted if all default values are desired. Finally, all the input elements are listed. They are all of type exsite_integer and need to appear in the same order as given in the xml description.

8.2.1.2 Template files

Template files to be used in ExSite are a combination of user input descriptions and templates used to generate input files. They consist of two sections. The first section contains an input description as used for module input description. It contains input parameters needed to execute the template. The second part consists of a template using a selected XML listing and the parameters supplied in the first section.

The first section is an input description for keyword-based input. In addition to the fields supplied by the input description, the program automatically adds descriptions for:

- "evals," an input field that takes file name of the file containing the XML listing of the abbreviated ENDF information

- "input," an input field that describes the name of the input file to be created

- If the input description contains a field named "neutgroups" and a field called "thermalgroups," a field name "iftg," denoting the first thermal group, is automatically created.

In the template the fields of the user input are available in two ways:

- A postfix of "_user" is added to list of arguments available to the writeData function listed in

  Tables 15 and 17.

- XML entities are added to the template created. XML entities can be accessed in an XML file by using "&entity;". Common examples are the entities normally provided for the ampersand and the lesser than sign, "&amp;" and "&lt;" respectively. However, since the template in given in Exsite template file does not have these values defined, "entity_fieldname" is used, which will be translated to an entity reference if expanding the template.

writeData applies to the data for a given evaluation. Therefore, writeData only works inside a loop. On the other hand, entities are available throughout the template.

In addition, if an input field of type "exsite_array" is encountered, additional user fields are created:

- fieldname_number contains the number of fields in the array

- fieldname_strip creates a list of the array with array markers, commas and end of line values stripped and substituted by spaces

- fieldname_eol is the same as fieldname_strip, except instead of a space an eol is used

There are two types of ExSite templates: one that gives a template, and one that uses previously defined templates. The later allows the user to string previously defined templates together.

Inside the root element, which can have any desired name, the first type of template has two elements:

- An element named *InputParameters* that contains the description for the user input. It is structured like a module input description.

- An element named InputData which contains the template to be used.

The following example creates a tabular list of all the evaluations (line numbers are not part of the template):

```
<?xml version="1.0"?>


<customTemplate>


<InputParameters>

  <table type="exsite_group" endBlock="yes" endBlockValue="\n end">

      <exsite_summary_line>Create a table of  the xml
data.</exsite_summary_line>

   <centrm type="exsite_string" appendEol="true" spaceAllowed="yes"

                required="yes" hide="true">

          <exsite_summary_line>Centrm parameters</exsite_summary_line>

             <exsite_default>

         </exsite_default>

         <exsite_required_position>0</exsite_required_position>

    </centrm>




    <addgamma type="exsite_enum" required="no" keyword="yes">

       <enum_type type="exsite_string"/>
```

```
        <exsite_default>yes</exsite_default>

        <exsite_enum_option>yes</exsite_enum_option>

        <exsite_enum_option>no</exsite_enum_option>

       <exsite_summary_line> Do you want to list the gamma
data</exsite_summary_line>

    </addgamma>


  </table>

</InputParameters>


<InputData>


<openFile name="entity_input"/>


  <!-- add non-moderator evaluations first  -->

 <writeFile name="entity_input">

<text>Tag name   endf         scale id   yield </text>

 </writeFile>


<exclude run="entity_addgamma">

 <writeFile name="entity_input">

<text>  gamma eval </text>

</writeFile>

</exclude>


 <writeFile name="entity_input">

<text>

----------------------------------------------------------

</text>

 </writeFile>
```

```
<!-- restrict to non-moderator data -->

<loop restrict="+file3(yes) +neutron(yes) -file7(yes)">

<writeFile name="entity_input">

<text>writeData(tag,10) writeData(endf,10) writeData(changedza,10) </text>

<text restrict="AWP0(yes) file12(yes) file13(yes)"> yes </text>

<text restrict="-AWP0(yes) -file12(yes) -file13(yes)"> no  </text>

<text restrict="+gamma +addgamma_user(yes)">writeData(taggamma,10) </text>

<text>

</text>

    </writeFile>

</loop>




  <!-- add moderator evaluations   -->

<writeFile name="entity_input">

<text>

Tag name      endf     fast tag  fast endf    scale ID    has yield </text>

</writeFile>

<exclude run="entity_addgamma">

 <writeFile name="entity_input">

<text>  gamma eval </text>

</writeFile>

</exclude>


<writeFile name="entity_input">

<text>

-----------------------------------------------------------------------------
-----
```

```
</text>

 </writeFile>




<loop restrict="+file7(yes)">

<thermalLoop>

 <writeFile name="entity_input">

<text> writeData(tag,10)   writeData(endf,10) </text>

<text> writeData(tagfast_thermalNum,10) writeData(endffast_thermalNum,10)
writeData(changedzathermal_thermalNum,10) </text>

<text restrict="AWP0fast_thermalNum(yes) file12fast_thermalNum(yes)
file13fast_thermalNum(yes)"> yes </text>

<text restrict="-AWP0fast_thermalNum(yes) -file12fast_thermalNum(yes) -
file13fast_thermalNum(yes)"> no  </text>

<text restrict="+gammathermal_thermalNum +addgamma_user(yes)">
writeData(taggammathermal_thermalNum,10) </text>

<text>

</text>

</writeFile>

</thermalLoop>

</loop>




</InputData>



</customTemplate>
```

The first element (InputParameters, line 5-28) lists parameters under the user control. The user can select whether the associated gamma evaluations should be listed. The second element (InputData, line 30-107) lists the template. The input file is opened on line 32. Please note that the “entity_input” type of referring to user input must be used here. The form “writeData(input_user)” will only work inside a loop. Line 35-37 writes the first part of the title. The *text* elements must be enclosed in a *writeFile* element, and “entity_input” type of referring to user input must be used outside a loop construct. The title should contain a reference to the gamma evaluation only if the user requests it. This is done with the *exclude* element in line 93-43. Inside a loop construct the more convenient *restrict* attribute for the text element could have been used. The *loop* element in line 53-63 writes the data for non-moderator evaluations. The *restrict* argument is used to restrict the loop to evaluations pertaining to incident neutrons and containing File 3 data but no File 7 data. Lines 54–55 are used to write out quantities available for all evaluations. Line 56 prints a *yes* if the evaluation contains photon production data. An evaluation contains photon production data if AWP0 is yes or if it contains File 12 or File 13 data. Since neither of the restrict arguments is prefixed by a "+,"evaluations are selected for which AWP0 is *yes* and/or which File 12 and/or File 13 exists. Line 57 prints a *no* if no photon production data exist. The "-" before each of the restrict arguments should be marked. Lines 58–59 print the tag name of the associated gamma evaluation if the user requested it and if an associated gamma evaluations exists. Lines 64–104 repeat the same tables, but they only select thermal moderators. In order to print all associated evaluations in the fast range, a *thermalLoop* construct is used in lines 88–103. The postfix "fast_thermalNum" that appears on many of the quantities refers to the desired quantity in the evaluation to be used in the fast region. The postfix "gammathermal_thermalNum" selects the gamma evaluation associated with the fast evaluation.

Additional examples can be found in the exsite folder, which contains templates to create all the major library types.

8.3 GENERATING MODULE INPUT AND PDF INPUT

Input instructions for modules are automatically translated from information supplied in the file containing the main program for a module. The comments are given in lines starting with "!!*" for Fortran files and "*!!" for C or C++ files. All lines starting with this special comment character are extracted and interpreted as an XML file.

9. INSTALLATION

The AMPX build is based on CMake from KitWare, which supports a consistent experience on LINUX, Mac, and Windows.

AMPX requires:

- Intel Fortran/C++ 13.1 or higher
  or
  GNU GCC/G++/GFORTAN 4.6.1 or higher

- CMake – Platform independent build configuration

- QT 4.7 or higher

Optionally, AMPX requires LAPACK and BLAS libraries.

There are four main steps to create an AMPX installation:

1. CMake configuration creates a native build tree
2. Compilation compiles all executables and libraries
3. Optionally run test cases
4. Installation – This deploys all executables into a configuration ready for running

'CMakeLists.txt' files are found throughout AMPX. From the AMPX root directory, these CMakeLists.txt files create a tree of included directories called the SOURCE TREE.

To configure a build, 'cmake' is called on the root CMakeLists.txt file, namely `ampx_dir/CMakeLists.txt`. CMake takes the SOURCE TREE and creates a BUILD TREE. The BUILD TREE contains or will contain the build configuration, Make files, and all compilation output (i.e., object files, archive libraries and binary executables). Then `make install` in the build directory will install the packages, and an optional `ctest` command will run several tests to ensure that AMPX was built correctly.

AMPX requires QT and optionally LAPACK and BLAS routines. If these libraries reside in a known location (i.e., installed by yum or macports [see below]), the installation procedure will automatically find the libraries. Otherwise the location can be given at configuration time.

9.1 RECOMMENDED INSTALLATION PROCEDURE

1. Navigate to the root scale directory, where CMakeLists.txt, PackagesList.cmake and CTestConfig.cmake are shown. This is the root of the source tree that at which CMake will be pointed.
2. Make a build directory:
   ```
   mkdir build
   cd build
   ```
   and create a cmake script in this directory called *configure* with content similar to:

   ```
   #!/bin/bash

   INSTALL_PATH=${PWD}/install
   ```

```
OPTIONS=${PWD}/../script/options_ampx_packages.cmake

cmake \

        -D SCALE_ENABLE_TESTS:BOOL="ON" \

        -D SCALE_CONFIGURE_OPTIONS_FILE:FILEPATH=${OPTIONS} \

         -D DART_TESTING_TIMEOUT:STRING=6500 \

        -D CMAKE_INSTALL_PREFIX:FILEPATH=${INSTALL_PATH} \

        -D CMAKE_BUILD_TYPE:STRING=RELEASE \

        -D CMAKE_Fortran_COMPILER:STRING=gfortran \

        -D CMAKE_CXX_COMPILER:STRING=g++ \

        -D CMAKE_C_COMPILER:STRING=gcc \

        $*
```

and make it executable:

```
    chmod u+x configure
```

An example script file is provided in the script directory: *configure_ampx_gcc* for gnu compilers and *configure_ampx_intel* for intel compilers.

3. Create the build tree by running

```
./configure ../
```

4. Install the packages by running
   ```
   make install
   ```
   and optionally run the test suite:
   ```
   ctest
   ```
5. If third party libraries are not found, the user may need to specify the path to them. For QT, add
   ```
   -D CMAKE_PREFIX_PATH=/Path_to_QT_directory/bin \
   ```

   to the configure script. For lapack and blas, add:
   ```
   -D TPL_LAPACK_LIBRARIES:STRING=/Path_to_lapack_libs/lib/liblapack.so\
   -D TPL_BLAS_LIBRARIES:STRING=/Path_to_lapack_libs//lib/libblas.so\
   ```
   to the configure script.

Every library and executable is a TARGET. Calling 'make' on Linux and Mac will build ALL targets. For example, typing *make x10* will only build the x10 executable.

CMAKE_Fortran_COMPILER, CMAKE_Fortran_FLAGS, CMAKE_C_COMPILER, CMAKE_C_FLAGS, CMAKE_CXX_COMPILER, and CMAKE_CXX_FLAGS may be modified in the cmake command invocation line to update the compilers or compiler flags to the desired settings.

9.2 MAC OSX

XCode is needed to compile AMPX on a Mac OS X computer. It is available free from the App Store. In addition, the command line options from XCode must be installed. This can be accomplished by navigating to XCode->Preferences->Downloads->Components->Command Line Tools, and clicking "Install.'"

The latest version of macports, downloadable from http://www.macports.org, is also required.

If the rsync is blocked by a firewall, macports can be synchronized over http by changing the file */opt/local/etc/macports/sources.conf*, changing the line:

```
rsync://rsync.macports.org/release/tarballs/ports.tar [default]
```

to

```
http://www.macports.org/files/ports.tar.gz [default]
```

The following commands will install all necessary components from macports:

If rsync port is blocked use:

```
sudo port –d sync
```

If rsync port is open:

```
sudo port selfupdate
```

Regardless of rsync status, the following commands should be executed:

```
sudo port install gcc48
sudo port install qt4-mac
sudo port install cmake
```

These commands upgrade the default compilers from gcc-4.2.2 to gcc-4.8.2, and they install qt4.8.4 and cmake. Newer versions may be used if available from macports. Once the new version is installed, the default gcc version must be selected. First the available versions are listed:

```
port select --list gcc
```

And then one of the available ports should be selected via:

```
sudo port select --set gcc mp-gcc48
```

Installation will then proceed in the same manner as on Linux.

10. FIDO INPUT

10.1 INTRODUCTION

The FIDO input method is specially devised to allow entering or modifying large data arrays with minimal effort. Patterns of repetition or symmetry are used wherever possible. The FIDO system was patterned after the input method used with the FLOCO coding system at Los Alamos and was first applied to the DTF-II code. Since that time, numerous features requested by users have been added, a free-field option has been developed, and the application of FIDO has spread to innumerable codes.

The data are entered in units called "arrays." An array comprises a group of contiguous storage locations to be filled with data at the same time. These arrays usually correspond one-to-one with FORTRAN arrays used in the program. A group of one or more arrays read with a single call to the FIDO package forms a "block," and a special delimiter is required to signify the end of each block. Arrays within a block may be read in any order with respect to each other, but an array belonging to one block must not be shifted to another. The same array can be entered repeatedly within the same block. For example, an array could be filled with "0" using a special option, and then a few scattered locations could be changed by reading in a new set of data for that array. If no entries to the arrays in a block are required, the delimiter alone satisfies the input requirement.

Three major types of input are available: fixed-field input, free-field input, and user-field input.

10.2 FIXED-FIELD INPUT

The fixed-field input option is documented here for completeness. The use of fixed-field input is NOT recommended. Use the free-field input option documented in Sect. 10.3.

Each card is divided into six 12-column data fields, each of which is divided into three subfields. The following outline illustrates a typical data field. The three subfields always comprise 2, 1, and 9 columns, respectively.

To begin the first array of a block, an *array originator field* is placed in any field on a card:

Subfield 1:	An integer *array identifier* <100 specifying the data array to be read in.
Subfield 2:	An *array-type indicator*: "$" if the array is integer data "*" if the array is real data "#" if the array is double-precision data
Subfield 3:	Blank

Data are then placed in successive fields until the required number of entries has been accounted for.

In entering data, it is convenient to think of an "index" or "pointer" as a designator that is under the control of the user and that specifies the position in the array into which the next data entry is to go. The pointer is always positioned at array location #1 by entering the array originator field. The pointer subsequently moves according to the data operator chosen. Blank fields are a special case in that they do not cause any data modification and do not move the pointer.

A *data field* has the following form:

Subfield 1:	The *data numerator*, an integer <100, referred to as N_1 in the following discussion
Subfield 2:	One of the special *data operators* listed below
Subfield 3:	A nine-character *data entry* to be read in F9.0 format. It will be converted to an integer if the array is a "$" array or if a special array operator such as Q is being used. Note that an exponent is permissible but not required. Likewise, a decimal is permissible but not required. If no decimal is supplied, it is assumed to be immediately to the left of the exponent, if any; and otherwise to the right of the last column. This entry is referred to as N_3 in the following discussion.

A list of data operators and their effects on the array being entered is provided below:

"Blank" indicates a single entry of data. The data entry in the third subfield is entered in the location indicated by the pointer, and the pointer is advanced by one. However, an entirely blank field is ignored.

"+" or "–" indicates exponentiation. The data entry in the third field is entered and multiplied by 10^{+N_1} or $10^{\pm N_1}$,where N_1 is the data numerator in the first subfield, given the sign indicated by the data operator itself. The pointer advances by one. For cases in which an exponent is needed, this option allows the entering of more significant figures than the blank option.

"&" has the same effect as "+."

"R" indicates that the data entry is to be repeated N_1 times. The pointer advances by N_1.

"I" indicates linear interpolation. The data numerator, N_1, indicates the number of interpolated points to be supplied. The data entry in the third subfield is entered, followed by Nj interpolated entries equally spaced between that value and the data entry found in the third subfield of the next nonblank field. The pointer is advanced by $N_1 + 1$. The field following an "I" field is then processed normally according to its own operator. The "I" entry is especially valuable for specifying a spatial mesh. In "$" arrays, interpolated values will be rounded to the nearest integer.

"L" indicates logarithmic interpolation. The effect is the same as that of "I" except that the resulting data are evenly separated in log-space. This feature is especially convenient for specifying an energy mesh.

"Q" is used to repeat sequences of numbers. The length of the sequence is given by the third subfield, N_3. The sequence of N_3 entries is to be repeated N_1 times. The pointer advances by N_1*N_3. If either N_1 or N_3 is 0, then a sequence of $N_1 + N_3$ is repeated one time only, and the pointer advances by $N_1 + N_3$. This feature is especially valuable for geometry specification.

The "N" option has the same effect as "Q" except that the order of the sequence is reversed each time it is entered. This feature is valuable for the type of symmetry possessed by S_n quadrature coefficients.

"M" has the same effect as "N" except that the sign of each entry in the sequence is reversed each time the sequence is entered. For example, the entries 1 2 3 2M2 would be equivalent to 1 2 3 -3 -2 2 3. This option is also useful in entering quadrature coefficients.

"Z" causes $N_1 + N_3$ locations to be set at 0. The pointer is advanced by $N_1 + N_3$.

"C" causes the position of the last array entered to be printed. This is the position of the pointer less 1. The pointer is not moved.

"O" causes the print trigger to be changed. The trigger is originally off. Successive "O" fields turn it on and off alternately. When the trigger is on, each card image is listed as it is read.

“S” indicates that the pointer is to skip N_1 positions, leaving those array positions unchanged. If the third subfield is blank, the pointer is advanced by N_1. If the third subfield is noy blank, that data entry is entered following the skip, and the pointer is advanced by $N_1 + 1$.

“A” moves the pointer to the position, N_3 specified in the third subfield.

“F” fills the remainder of the array with the datum entered in the third subfield.

“E” skips over the remainder of the array. The array length criterion is always satisfied by an E, no matter how many entries have been specified. No more entries to an array may be given following an “E” except that data entry may be restarted with an “A.”

The reading of data to an array is terminated when a new array origin field is supplied or when the block is terminated. If an incorrect number of positions has been filled, an error edit is given, and a flag is set which will later abort execution of the problem. FIDO then continues with the next array if an array origin was read. Otherwise, control is returned to the calling program.

A *block termination* consists of a field having “T” in the second subfield. Entries following “T” on a card are ignored, and control is returned from FIDO to the calling program.

Comment cards can be entered within a block by placing an apostrophe (') in column 1. Then columns 2–80 will be listed, with column 2 being used for printer carriage control. Such cards have no effect on the data array or pointer.

10.3 FREE-FIELD INPUT

With free-field input, data are written without fixed restrictions as to field and subfield size and positioning on the card. The options used with fixed-field input are available, although some are slightly restricted in form. In general, fewer data cards are required for a problem, the interpreting print is easier to read, a card listing is more intelligible, the cards are easier to keypunch, and certain common keypunch errors are tolerated without affecting the problem. Data arrays using fixed- and free-field input can be intermingled at will within a given block,

The concept of three subfields per field is still applicable to free-field input, but if no entry for a field is required, no space for it needs to be left. Only columns 1–72 may be used as with fixed-field input. A field may not be split across cards.

The *array originator field* can begin in any position. The array identifiers and type indicators are used as in fixed-field input. The type indicator is entered twice to designate free-field input (i.e., “$$,” “**,” or “##”). The blank third subfield required in fixed-field input is not required. For example,

```
31**
```

indicates that array 31, a real-data array, will follow in free-field format.

Data fields may follow the array origin field immediately. The data field entries are identical to the fixed-field entries with the following restrictions:

1. Any number of blanks may separate fields, but at least one blank must follow a third subfield entry if one is used.

2. If both first- and second-subfield entries are used, no blanks may separate them (i.e., 24S, but not 24 S).

3. Numbers written with exponents must not have imbedded blanks (i.e., 1.0E+4, 1.0-E4, 1.0+4, or even

1+4, but *not* 1.0 E4). A zero should never be entered with an exponent. For example, 0.00-5 or 0.00E-5 will be interpreted as -5 × 10^{-2}.

4. In third-subfield data entries, only 9 digits—including the decimal but not including the exponent field—can be used (i.e., 123456.89E07, but *not* 123456.789E07).

5. The Z entry must be of the form: 738Z, *not* Z738 or 738 Z.

6. The + or - data operators are not needed and are not available.

7. The Q, N, and M entries are restricted: 3Q4, 1N4, M4, but *not* 4Q, 4N, or 4M.

10.4 USER-FIELD INPUT

If the user follows the array identifier in the array originator field with the character "U" or "V," the input format is to be specified by the user. If "U" is specified, the FORTRAN FORMAT to be used must be supplied in columns 1–72 of the next card. The format must be enclosed by the usual parentheses. Then the data for the entire array must follow on successive cards. The rules of ordinary FORTRAN input as to exponents, blanks, etc., apply. If the array data do not fill the last card, the remainder must be left blank.

"V" has the same effect as "U" except that the format read in the last preceding "U" array is used.

10.5 CHARACTER INPUT

If the user wishes to enter character data into an array, at least three options are available. The user may specify an arbitrary format using a "U" and reading in the format. The user may follow the array identifier by a "/." The next two entries into subfield 3 specify the beginning and ending indices in the array into which data will be read. The character data are then read starting with the next data card in an 18A4 format.

Finally, the user may specify the array as a free-form "*" array and then specify the data entries as "nH" character data, where n specifies how many characters follow H.

11. REFERENCES

1. N. M. Greene, J. L. Lucius, L. M. Petrie, W. E. Ford, III, J. E. White, and R. Q. Wright, *AMPX, A Modular Code System for Generating Coupled Multi-Group Neutron-Gamma Libraries from ENDF/B*, ORNL/TM-3706, Union Carbide Corp., Nucl. Div., Oak Ridge National Laboratory, March 1976.

2. N. M. Greene, J. L. Lucius, J. E. White, R. Q. Wright, C. W. Craven, Jr., and M. L. Tobias, XLACS: A Program to Produce Weighted Multi-Group Neutron Cross-sections from ENDF/B, ORNL-TM-3646, Union Carbide Corp., Nucl. Div., Oak Ridge National Laboratory, April 1972.

3. J. R. Knight and F. R. Mynatt *MUG: A Program for Generating Multi-Group Photon Cross-sections*, CTC-17, Union Carbide Corp., Nucl. Div., Oak Ridge National Laboratory, January 1970.

4. W. E. Ford, III and D. H. Wallace, *POPOP4 — A Code for Converting Gamma-Ray Spectra to Secondary Gamma-Ray Production Cross-sections*, CTC-12, Union Carbide Corp., Nucl. Div., Oak Ridge National Laboratory, May 1969.

5. W. W. Engle, Jr., *A User's Manual for ANISN, A One-Dimension Discrete Ordinates Transport Code with Anisotropic Scattering*, K-1693, Union Carbide Corp., Nucl. Div., Oak Ridge National Laboratory, July 1982.

6. W. W. Engle, Jr., *A User's Manual for ANISN, A One-Dimension Discrete Ordinates Transport Code with Anisotropic Scattering*, K-1693, Union Carbide Corp., Nucl. Div., Oak Ridge National Laboratory, July 1982.

7. M. B. Emmett, *MORSE-CGA, A Monte Carlo Radiation Transport Code with Array Geometry Capability*, ORNL-6174, Martin Marietta Energy Systems, Inc., Oak Ridge National Laboratory, April 1985.

8. N. M. Greene and C. W. Craven, Jr., XSDRN: A Discrete Ordinates Spectral Averaging Code, ORNL/TM-2500, Union Carbide Corp., Nucl. Div., Oak Ridge National Laboratory, July 1969.

9. H. J. Kopp and D. S. Selengut, "DATATRAN—A Data-Handling Computer Language for a Large Modular Reactor Design System," pp. 1460–1472 in *Proceedings of the International Conference on the Utilization of Research Reactors and Reactor Mathematics and Computation*, CNM-R-2, Knolls Atomic Power Laboratory, May 1967.

10. H. J. Kopp and D. S. Selengut, "DATATRAN—A Data-Handling Computer Language for a Large Modular Reactor Design System," pp. 1460–1472 in *Proceedings of the International Conference on the Utilization of Research Reactors and Reactor Mathematics and Computation*, CNM-R-2, Knolls Atomic Power Laboratory, May 1967.

11. H. C. Honeck, "The JOSHUA System," DPSTM-500, Du Pont De Nemours & Co., Savannah River Lab., November 1969.

12. *SCALE: A Comprehensive Modeling and Simulation Suite for Nuclear Safety Analysis and Design*, ORNL/TM-2005/39, Version 6.1, June 2011. Available from the Radiation Safety Information Computational Center at Oak Ridge National Laboratory as CCC-785.

13. "AMPX-II, Modular Code System for Generating Coupled Multi-Group Neutron-Gamma-Ray Cross-Section Libraries From Data in ENDF Format," RSIC documentation for PSR-63, Union Carbide Corp., Nucl. Div., Oak Ridge National Laboratory, November 1978.

14. N. M. Greene, W. E. Ford, III, L. M. Petrie and J. W. Arwood, *AMPX-77: A Modular Code System for Generating Coupled Multigroup Neutron-Gamma Cross-Section Libraries from ENDF/B-IV*

and/or ENDF/B-V, ORNL/CSD/TM-283, Martin Marietta Energy Systems, Inc., Oak Ridge National Laboratory, October 1992.

15. N. M. Greene, W. E. Ford, III, L. M. Petrie and J. W. Arwood, *AMPX-77: A Modular Code System for Generating Coupled Multigroup Neutron-Gamma Cross-Section Libraries from ENDF/B-IV and/or ENDF/B-V*, ORNL/CSD/TM-283, Martin Marietta Energy Systems, Inc., Oak Ridge National Laboratory, October 1992.

16. N. M. Greene, J. W. Arwood, R. Q. Wright and C. V. Parks, *The LAW Library -- A Multi-Group Cross-Section Library for Use in Radioactive Waste Analysis Calculations*, ORNL/TM-12370, Martin Marietta Energy Systems, Inc., Oak Ridge National Laboratory, August 1994.

17. "ENDF/B-6 Formats Manual: Data Formats and Procedures for the Evaluated Nuclear Data Files ENDF/B-VI and ENDF/B-VII," BNL-90365-2009, Rev. 1, CSEWG Document ENDF-102, M Herman and A. Trkov, Ed., Brookhaven National Laboratory (2009).

18. B. T. Rearden, M. T. Sieger, S. M. Bowman, and J. P. Lefebvre, *Quality Manual for the SCALE Code System,* SCALE-QAP-005, Rev. 4, Oak Ridge National Laboratory, Oak Ridge, Tenn. (2013).

19. A. M. Weinberg and E. P. Wigner, *The Physical Theory of Neutron Chain Reactors*, University of Chicago Press, 1958, pp. 508–514.

20. G. E. Hansen and W. H. Roach, *Six- and Sixteen-Group Cross-sections for Fast and Intermediate Critical Assemblies*, LAMS-2534, Los Alamos Natl. Lab., September 1963.

21. C. M. Hopper and J. P. Renier, "Expanded and Applied Sixteen-Neutron-Energy-Group Cross-Section Library," *Trans. Am. Nucl. Soc.*, **61**, 186, 1990.

22. I. I. Bondarenko, Ed., *Group Constants for Nuclear Reactor Calculations*, Consultants Bureau, New York, 1964.

23. H. C. Honeck, *THERMOS, A Thermalization Transport Theory Code for Reactor Lattice Calculations*, BNL-5826, Brookhaven National Laboratory, 1961.

24. G. D. Joanou and J. S. Dudek, "Gam-II: A B_3 Code for the Calculation of Fast-Neutron Spectra and Associated Multigroup Constants," GA-4265, General Atomic, Sept. 1963.

25. "POLIDENT: A Module for Generating Continuous-Energy Cross Sections From ENDF Resonance Data", M.E. Dunn and N. M. Greene, NUREG/CR-6694 and ORNL/TM-2000/035, Oak Ridge National Laboratory (2000).

26. R. Hwang, "A Rigorous Pole Representation of Multilevel Cross-sections and Its Practical Applications," *Nuc. Sci. Eng*., **96**, 192, 1987.

27. W. H. Press, W. T. Vetterling, S. A. Teukolsky and B. P. Flannery, *Numerical Recipes in FORTRAN The Art of Scientific Computing*, Second Edition, Cambridge University Press, 1992.

28. R. Hwang, "A Rigorous Pole Representation of Multilevel Cross-sections and Its Practical Applications," *Nuc. Sci. Eng*., **96**, 192, 1987.

29. R. E. MacFarlane and A. C. Kahler, Methods for processing ENDF/V-VII with NJOY, Nuclear Data Sheets **111** (2010), 2739.

30. C. Kalbach and F.M. Mann, *Phenomenolgy of continuous angular distributions. I. Systematic and Parameterization*, Phys. Rev. C, **23** (1981), 112.

31. C. Kalbach, *Systematics of Continuous Angular Distributions: Extensions to Higher Energies*, Phys. Rev. C, **37** (1988), 2350.

32. O. Klein, Y. Nishina, Z. Phys. **52**, 853 (1929).

33. M. E. Dunn, L. C. Leal, "Calculating Probability Tables for the Unresolved-Resonance Region Using Monte Carlo Methods," PHYSOR 2002 # A0107, May 2002.

34. F. J. Dyson and M. L. Mehta, "Statistical Theory of the Energy Levels of Complex Systems," *J. Math. Phys*, **4**, 701 (May 1963).

35. L.C. Leal and N.M. Larson, SAMDIST: *A Computer Code for Calculating Statistical Distributions for R-Matrix Resonance Parameters*, ORNL/TM-13092, Lockheed Martin Energy Research Corporation, Oak Ridge National Laboratory, September 1995.

36. R. Hwang, "A Rigorous Pole Representation of Multilevel Cross-sections and Its Practical Applications," *Nuc. Sci. Eng*., **96**, 192, 1987.

37. M. L. Williams, "Resonance Self-Shielding Methodologies in SCALE 6," *Nucl. Technol*. **174**, 149 (2011).

38. M. L. Williams, K. S. Kim, "The Embedded Self-Shielding Method," *Proceedings of PHYSOR 2012 – Advances in Reactor Physics – Linking Research, Industry, and Education*, Knoxville, Tennessee, April 15–20 (2012).

39. M. A. Jesse et al., "POLARIS: A New Two-Dimensional Lattice Physics Analysis Capability for the SCALE Code System," *Proceedings of PHYSOR 2014 International Conference: The Role of Reactor Physics toward a Sustainable Future* , Kyoto, Japan, September 28–October 2 (2014).

40. B. Kochunas, et al., *Overview of Development and Design of MPACT: Michigan Parallel Characteristics Transport Code*, Proc. M&C 2013, Sun Valley, ID, USA. May 5-9 (2013).

41. R. J. J. Stamm'ler and M. J. Abbate, *Methods of Steady-State Reactor Physics in Nuclear Design*, Academic Press, London (1983).

42. D. Wiarda, G. Arbanas, L. Leal, M. E. Dunn, *Recent Advances with the AMPX Covariance Processing Capabilities in PUFF-IV*, 2008 Workshop on Neutron Cross Section Covariances, Port Jefferson, New York, June 24–27 (2008).

43. N.M. Larson, *Updated User's Guide for SAMMY: Multilevel R-Matrix Fits to Neutron Data Using Bayes' Equations*, ORNL/TM-9179/R6 (July 2003).

44. J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling, "A Set of Level 3 Basic Linear Algebra Subprograms," A*CM Trans. Math. Soft.*, 16 (1990), pp. 1–17.

45. N. M. Steen, G. D. Byrne and E. M. Gelbard, "Gaussian quadrature of the Integrals. $\int_0^\infty exp(-x^2)\, f(x) dx$ and $\int_0^b exp(-x^2)\, f(x) dx$," *Mathematics of Computation*, **23**, 661, (1969).

46. H. Henryson II, J. Toppel and C.G. Stenberg, *MC2-2: A Code to Calculate Fast Neutron Spectra and Multigroup Cross Section*, ANL-8144, Argonne National Laboratory, June 1976.

47. C. M. Mattoon, B. R. Beck, N. R. Patel, N. C. Summers, G. W. Hedstrom, *Generalized Nuclear Data: A New Structure (with Supporting Infrastructure) for Handling Nuclear Data*, Nuclear Data Sheets **113(12)**, pp. 3145–3171 (2012).

APPENDIX A. APMX INPUT INSTRUCTIONS

APPENDIX A. AMPX INSTRUCTIONS

A-1. AJAX - MODULE TO MERGE, COLLECT, ASSEMBLE, REORDER, JOIN, COPY SELECTED NUCLIDES FROM AMPX WINTERFACES

AJAX is a module to combine data on AMPX master or working libraries. Options are provided to allow merging from any number of files in a manner that will allow the user to determine the final nuclide ordering if desired.

Input Data

Block 1

-1$ Core assignment [1]

1.	NWORD	number of words to allocate (50,000)

0$ Logical assignments [2]

1.	MWT	logical number of new library (1)
2.	NWAX	not used (0)

1$ Number of files [1]

1.	NFILE	number of files from which data will be selected

Terminate Block 1 with a T.

Stack Block 2 and 3 one after the other NFILE times.

Block 2

2$ File and option selection [2]

1. NF — logical number of file considered
1. IOPT — nuclide treatment

   The following choices are available:

   - -N: deletes N nuclides from NF to create the new file on MWT
   - 0: adds all nuclides to the new file on MWT
   - N: adds N nuclides from NF to create the new file on MWT Sets with duplicate identifiers will not be entered on MWT. The first occurrence of an identifier selects that set for the new library.

5$ Sequence number [1]

1.	SEQ	sequence number to use for working library

Terminate Block 2 with a T.

Only use Block 3 if IOPT != 0.

Block 3

3$ Nuclides selected [IOPT]

1. ID — identifiers of nuclides to be added or deleted from NF

   Only used if IOPT != 0.

4$	New identifiers [IOPT]		
	1.	IDNEW	allows changing the identifier given in the 3$ array for the new library. Only used if IOPT > 0.
6$	Zone id to select [IOPT]		
	1.	ZONE	zone id of the the nuclides to select. A negative value selects all (-1) Only used if IOPT != 0.
7$	New zone identifiers [IOPT]		
	1.	NZONE	allows changing the identifier given in the 6$ array for the new library. (0) Only used if IOPT > 0.

Terminate Block 3 with a T.

Repeat block title optionally up to five times.

Block Title

title: title card for the AMPX working library Type: Character*72

Sample Input

```
    0$$ 40 1$$ 3 T
    2$$ 13 T
    3$$ 92235 92238 94249 T
    2$$ 20 T
    2$$ 31 T
    3$$ 100000 T
```

This input creates a library on logical unit 4 using data from logical units 1, 2, and 3, as follows: three nuclides—92235, 92238, and 94249—are taken from logical unit 1; all nuclides from logical unit 2 are copied unless they use one of the three identifiers already copied. Finally, a data set identified by 100000 is copied from logical unit 3. Please note that AJAX does not check to determine whether the commands have been fully completed. In other words, if logical unit 1 does not have a 92235, it cannot be copied, but the code will not produce any errors. The AJAX output, however, does list the nuclides copied and their data sources.

Logical Unit Parameters

Variable	Unit number	Type	Description
NF		binary	
MWT		binary	logical number of new library
	18	binary	scratch
	19	binary	scratch

A-2. ALPO - MODULE FOR PRODUCING ANISN LIBRARIES FROM AMPX WORKING LIBRARIES

ALPO (ANISN Library Production Option) is a module for producing ANISN libraries from AMPX working libraries. Several working libraries can be accessed in a given run. The ANISN library can be produced in either binary or BCD format.

Input Data

Block 1

0$ Logical Assignments [2]

1.	MAN	logical unit for the ANISN library (use a 7 when a punched card output is desired) (20)
2.	MAX	start of ANISN IDs (1)

1$ Primary Options [9]

1.	NFILE	number of working libraries to be accessed (0)
2.	IHT	position of the total cross section in the ANISN tables (3)
3.	IHS	position of the within-group cross section in the ANISN tables (3) IHT + IGM - IFTG + 1, where IGM is the number of neutron energy groups; IFTG is the first thermal group
4.	ITL	table length of the ANISN tables (0) IHS + IGM + IPM - 1, where IPM is the number of gamma-ray groups
5.	MAXPL	maximum order of scattering to be written on the ANISN library (20)
6.	IOPTID	option to print label with each block of ANISN cross sections (0) 0: no printing 1: print data
7.	IOPT2D	option to print scattering matrices (0) 0: no printing 1: print data
8.	ITRANS	transport correction option (0) ☐ -N - truncate PN and above matrices and correct all lower ordered within-group terms by subtracting (2l+1)*sigmaN(g->g')/(2N+1) ☐ 0 - no transport correction ☐ N - replace sigmat with sigmatr = sigmaa + (1 - mu)* sigmas, where mu is calculated by summing the Pl matrix and dividing by the P0 sum, or by 2/(3*A), when Pl is not given. The within-group term is also adjusted.
9.	ICORE	number of words to allocate (50,000)

Terminate Block 1 with a T.

Stack Block 2 and 3 one after the other NFILE times

Block 2

2$ File selection options [2]

1. NF logical number of the working library (0)
2. IOPT nuclide selection (0)
   - -N - accepts all nuclides from the working library except the N designated in the 3$ array below
   - 0, accepts all nuclides from the working library
   - N - accepts 0 nuclides designated in the 3$ array below

Terminate Block 2 with a T.

Only use Block 3 if IOPT != 0.

Block 3

3$ nuclides to be selected or ignored [IOPT]

1. NUCS nuclides to be selected or ignored (0) only used if IOPT != 0

Terminate Block 3 with a T.

Sample Input

```
0$$ 20 E 1$$ 1 4 10 30 3 0 0 0 500000 T
2$$ 4 5 T
3$$ 92235 92238 8016 1001 26000 T
```

This discussion assumes that data are being accessed from a 50 group AMPX working library on logical unit 4. Input indicates that an ANISN library on logical unit 20 should be created. The total cross section is in position 4 in the ANISN cross section tables, which implies (since default values were not overriden using the 1$ array) that nu times the fission cross section is in position 3, the absorption cross section is in position 2, and the fission cross section is in position 1. Furthermore, by specifying that the within-group scattering cross section is in position 10, only 6 upscattering terms are possible. If upscatters are found on the working library that scatter by more than 6 groups up, those terms are "summed" into the source group number less 6 scattering terms. This keeps the scattering matrices balanced and allows the scatter to be put in the highest place available in the matrix. It is also specified that the "table length" is 30, which means that the table has slots for 30-10 or 20 downscattering terms. As is the case with upscattering, if terms are encountered which scatter down by more than 20 groups, they are added to the lowest transfer terms available in the table. Five nuclides were selected from the working library: ^{235}U (92235), ^{238}U (92238), ^{16}O (8016), ^{1}H (1001), and Fe (26000).

Logical Unit Parameters

Variable	Unit number	Type	Description
MAN		binary	logical unit for the ANISN
NF		binary	logical number of the working library
	14	binary	scratch

A-3. BROADEN - MODULE TO DOPPLER BROADEN TAB1 FUNCTIONS

BROADEN reads data on a double or single precision binary TAB1 library, and Doppler broadens the total, elastic-scattering, fission, first-chance fission, and capture cross sections. It writes the Doppler-broadened data onto a binary TAB1 library. Optionally, it will Doppler broaden processes other than those just mentioned. This code is based on two subroutines written by Dermit E. “Red” Cullen of LLNL, called HUNKY and FUNKY. These routines use numerical integrations to perform Doppler broadening.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
INPUT=	LOGPT, NENDF, NTAP1	31	logical unit of the input TAB1 library
OUTPUT=	LOGDP, NDOP, NOUT	32	logical unit of the output TAB1 library
T=			space-separated list of temperature(s) in Kelvin at which data should be broadened
MAT=			space-separated list of material identifiers (if not present, all are Doppler broadened)
MT=			space-separated list of reaction identifiers to broaden (if not present, only default MT values are broadened)
addMT=			adds the list of indicated MT values to the list of MTs being broadened
icekeno=		1	option to also broaden 3, 20, 21 and 38 0 - do not broaden 3, 20, 21 and 38 1 - broaden 3, 20,21 and 38
outmode=		0	manner in which the output should be saved 0 - select the same mode as the input 1 - save as single precision -1 - save as double precision
oldBroaden			option to not add extra points
eps=		0.001	precision level at which the adaptive mesh should be created

Notes

The numerical integration routines used in BROADEN were developed by Dermit E. Cullen at LLNL. A characteristic of these routines is that they assume the input cross section is linear in energy. The module POLIDENT constructs the cross section data on a suitable dense linear-linear mesh. In addition, the BROADEN module will add points as needed.

Sample Input

```
INPUT=1 OUTPUT=2
T= 300 900 2100 MAT= 1000 2000
```

This input indicates that data should be read from the TAB1 library on logical unit 1, and that data should be written to a new file on logical unit 2. The data will be Doppler broadened for temperatures of 300, 900, and 2100 Kelvin for the materials identified by 1000 and 2000.

Logical Unit Parameters

Variable	Unit number	Type	Description
INPUT		binary	logical unit of the input TAB1 library
OUTPUT		binary	logical unit of the output TAB1 library
14		binary	scratch
99		binary	scratch

A-4. CADILLAC - (COMBINE ALL DATA IDENTIFIERS LISTED IN LOGICAL AMPX COVERX- FORMAT)

CADILLAC (Combine All Data Identifiers Listed in Logical AMPX COVERX-format) is an AJAX-like module that can be used to combine multiple covariance data files in COVERX format into a single covariance data file. The user can change the material IDs as needed. CADILLAC reads and exports data in a binary format native to the computing platform.

Input Data

Block Output

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
out=			logical unit for final COVERX output file
directory=		no	Creates a contents directory of input COVERX files without assembling an output file. no - creates an output file yes - creates directory and exit

Repeat block Input as often as needed.

Block Input

Block starts on first encounter of a keyword in the block.

Block terminates on encountering the next occurrence of keyword in or end.

Keyword	Alternate	Default	Definition
in=			logical unit number for input COVERX file
file=		1	old or new COVERX formatted input file 0 - old COVERX input file 1 - new COVERX input file All COVERX files produced by the current AMPX code system are in new COVERX format.
dec=		no	Dec formatted BIG_ENDIAN COVERX file yes - old style DEC generated COVERX file no - not an old style DEC generated COVERX file all COVERX files produced by the current AMPX code system are in the new COVERX format.
delete=			space-separated list of materials to delete from input library

Keyword	Alternate	Default	Definition
add=			space-separated list of materials to add from input library The following special options are available: add=0 selects all nuclides from "in" if add > 0 and a 0 is specified as the last value in the array, all the nuclides on the input file will be selected; however the ids explicitly specified can be changed using the "new" array input.
new=			space-separated new material ids for materials given in add The number of new materials must match the number of materials given in add exactly.
secondary=			space-separated list of secondary id values that will be changed refers to the material id in cross material covariance matrices
matid=			space-separated list of new secondary id values for values given in secondary

Notes

- For all newly created COVERX files, file=1 and dec=no.
- There is a one-to-one correspondence between values in the "add" array and the "new" array. There is also a one-to-one correspondence between values in "secondary" and "matid".
- After "in" is specified, the keywords governing the operations on "in" must be specified prior to entering another "in".
- The minimum input following in specification is add=0 that specifies all nuclides from "in" will be copied to "out".
- The same keyword can only be entered once on a line of input. For example, the following input is invalid:

  add=id1 id2 in=33 add=id3 id4
  Entering the keyword "add" twice on the same line is invalid, but it is acceptable to have two different key words on the same line. In other words, there is no problem having the keywords "in" and "add" on the same line.

Sample Input

```
=cadillac out=23
in=20 file=1 delete=92233 end
=cadillac out=24
in=23 add=0 file=1
in=22 add=9222 new=92233 file=1 secondary=9228 9427
matid=92235 94239 end
```

The first input deletes the mat=92233 from the COVERX file on logical unit 20 and generates a new COVERX file on logical unit 23. The second CADILLAC input takes all materials from logical unit 23 and adds them to the new COVERX file on logical unit 24. In addition, the covariance matrices from the COVERX file on logical unit 22 that correspond to material id 9222 are added to the new COVERX file on logical unit 23 after first changing the material id to 92233. If cross material matrices with a second material id of 9228 or 9427 exist, the ids for the second material are changed to 92235 or 94239, respectively.

Logical Unit Parameters

Variable	Unit number	Type	Description
out		binary	logical unit for final COVERX output file
in		binary	logical unit number for input COVERX file
14		binary	scratch
15		binary	scratch

A-5. CAMELS - MODULE TO COMPARE AMPX MASTER OR WORKING LIBRARIES

CAMELS (Compare AMPX Master Libraries) compares two cross section collections on two AMPX master libraries (master or working). The two libraries must use the same neutron and/or gamma-ray group structures.

Comparisons are made for the 1-D data (group-averaged cross sections), Bondarenko factors, and the 2-D data (group-to-group transfer matrices). There is no requirement that the two libraries use the same ordering in the manner in which data are written. CAMELS keys on the identifiers of all classes of data and makes comparisons when it finds matches. The two libraries to be compared have to be of the same type.

The primary output from CAMELS is a file written in the AMPX master or working library format, depending on the input, containing values defined by (A-B)/B, where A represents the values on the first library, and B respresents the values on the second library. The second library is the reference library, and the values are the relative difference of the values on the first library relative to the reference library. Since the output is in the AMPX master or working format, it can be listed, plotted, etc., using any appropriate AMPX utility module, such as the PALEALE module.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
log1=	in1	1	logical unit of the first AMPX master/working library
log2=	in2	2	logical unit of the second AMPX master/working library
log3=	out	3	logical unit of the output AMPX master/working library
eps=		1e-5	the precision to compare
worker			If present, use to compare working libraries; otherwise master libraries are compared.
print=			data printing options 1dn - print 1-D neutron differences 1dg - print 1-D gamma differences 2dn - print 2-D neutron differences or transfer matrices 2dy - print 2-D yield matrices differences 2dg - print 2-D gamma matrices differences bond - print Bondarenko data differences

Sample Input

```
    log1=91 log2=92 eps=1e-3 print=1dn print=2dn print=bond
```

The example requests comparing the two data collections located on logical units 92 and 92. The differences with absolute values greater than 0.001 (0.1%) will be written on logical unit 3 in the AMPX master library format. In addition, all differences for 1-D and 2-D neutron data and for Bondarenko factors will be written on the screen.

Logical Unit Parameters

Variable	Unit number	Type	Description
log1		binary	logical unit of the first AMPX master/working library
log2		binary	logical unit of the second AMPX master/working library
log3		binary	logical unit of the output AMPX master/working library

A-6. CEEXTRACT - EXTRACT DATA OUT OF A CE LIBRARY

CEEXTRACT allows for extracting 1-D, kinematic data and probability tables in a format suitable for use in PLATINUM to make a new library. The 1-D data contain collision cross sections, which PLATINUM will override. The collision cross section data can be deleted from the TAB1 formatted files prior to a reprocessing with PLATINUM using module ZEST.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
zero=			the prefix of the library file
dir=			the directory containing the library files
1d=		0	unit of the file in which to save the 1-D data (If less or equal to zero, 1-D data are not exported)
2df=		0	unit of the file in which to save the 2-D temperature-independent data (If less or equal to zero, 2-D data are not exported.)
2dt=		0	unit of the file in which to save the 2-D temperature-dependent data (If less or equal to zero, 2-D data are not exported.)
prob=		0	unit of the file in which to save the probability table data (If less or equal to zero, probability table data are not exported.)
unit =		60	unit on which to read the CE library files

A-7. CHARMIN - MODULE TO CONVERT TAB1 LIBRARIES FROM SINGLE TO DOUBLE PRECISION, TO TEXT, OR FROM ANY OF THESE FORMATS TO ANY OF THE OTHERS

CHARMIN (Change and Re-Make INput File) is a code that converts between different TAB1-file formats.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
input=	inpu inp in i	31	logical unit of input file
output=	outpu outp out ou o	32	logical unit of output file
Select one of these			
single			Input file is single precision binary.
double			Input file is double precision binary.
fido			Input file is in FIDO format.
cen			Input file contains a CENTRM flux.
to	t	to	Keywords before this flag are for the input file. After this flag they are for the output file.
Select one of these			
single			Output file is single precision binary.
double			Output file is double precision binary.
bcd			BCD Tab1 format
ploth			XY columns with headers
plot			XY columns without headers
fido			output file in FIDO format
mat=			material number to use if reading centrm flux data
mt=			reaction number to use if reading centrm flux data

Repeat block zone descriptions as often as needed

Block Zone descriptions

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
zone=			zone which to read
ztemp=			temperature for the zone
sig0=			background cross section value for the zone
za_l=			lowest value of za for which to use this flux
za_h=			highest value of za for which to use this flux

Sample Input

```
    INPUT=1 OUTPUT=2 SINGLE TO DOUBLE
```

This indicates that the single precision binary file on logical unit 1 should be read and a double precision file on logical unit 2 should be created.

Logical Unit Parameters

Variable	Unit number	Type	Description
INPUT		binary	
OUTPUT		BCD or binary	

A-8. CLAROL - A MODULE TO REPLACE CROSS SECTIONS ON AN AMPX MASTER INTERFACE

CLAROL (Correct Libraries and Replace Old Labels) is a module that replaces or adds data in an AMPX master library at the lowest level (e.g., it can replace individual elements in either 1-D or transfer arrays). It also has provisions for modifying entries in the table of contents on a master library and for overriding the title cards associated with each data set on a master library. Because this module operates at such a detailed level, it is recommended that the user be familiar with the idiosyncrasies of the AMPX master interface format before attempting to use CLAROL.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
in=			logical unit of the input master/working library
out=			logical unit of the output master/working library
worker			If present, working library should be corrected.
nototal			If present, totals are not summed again.
noratio			If present, mt=1007 is not renormalized. Otherwise, for non-moderator materials, the free gas scattering matrix (MT=1007) is normalized to MT=2. For moderators, the elastic cross section (MT=2) is substituted by the sum value of the thermal scattering matrix. Moderators have a nonzero value in id=46.
noup			If present, scattering matrices are not corrected for upscatter.
nobond			If present, Bondarenko data are not renormalized to resummed totals. Not currently used, as Bondarenko data are not updated.
nosmall			If present, small values in 1-D are retained.
noyield			If present, yield matrices are not converted to units of yield.
nocompact			If present, scattering matrices are not compacted, and small values in the scattering matrix are set to zero, If a l>0 matrix has a non-zero term, where the l=0 matrix does not, it is set to zero
smallcut=		1.0d-12	cut-off value to set 1-D and scattering matrix values to zero If nosmall is not set, all 1-D cross section smaller than smallcut are set to zero. If nocompact is not set, all scattering matrix elements smaller than smallcut will be set to zero.

Repeat block Data as often as needed.

Block Data

Block starts on first encounter of a keyword in the block.

Block terminates on encountering the next occurrence of keyword end or neutron or gamma or yield or resonance or bondarenko or 1dn or 2dn or 1dg or 2dg or 2dy or sumn or sumg or title or end.

Keyword	Alternate	Default	Definition
Select one of these			
1dn			lists changes for 1-D neutron data
1dg			lists changes for 1-D gamma data
2dn			lists changes for 2-D neutron data
2dg			lists changes for 2-D gamma data
2dy			lists changes for yield matrices
bond			lists change for bondarenko factors
refbond			lists change for reference bondarenko cross sections.
trans			lists changes for transfer matrix
title			lists an new title for the indicated nuclide
sumn			additional user sum rules for neutron data
sumg			additional user sum rules for gamma data
ido=			the id of the old set If ido and idn are given and no records for idn exist, ido is copied and renamed to idn.
idn=			the id of the new set See ido. If idn is not given, the value of ido is used.
mt=			the reaction for which to change the data
nf=			the first group for which to apply changes If changing a scattering matrix, this is the source group
nl=			the last group for which to apply changes If changing a scattering matrix, this is the source group
nsink=			the sink group if changing scattering matrix data
lval=			the order of the matrix to update if changing scattering data
temp=			the temperature of the matrix to update
data=			values or text listing the desired changes The data section is enclosed between < and > signs. Multiple lines are allowed. For 1dn, 1dg, 2dn, 2dg, 2dy and trans, FIDO style array input is allowed.

Logical Unit Parameters

Variable	Unit number	Type	Description
in		binary	logical unit of the input master/ working library
out		binary	logical unit of the output master/working library

A-9. COGNAC - CONVERSION OPERATIONS FOR GROUP-DEPENDENT NUCLIDES IN AMPX COVERX-FORMAT

COGNAC (Conversion Operations for Group-Dependent Nuclides in AMPX COVERX-format) is a module used to convert COVERX formatted libraries from bcd to binary and vice versa.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
in=			logical unit number for input COVERX file
out=			logical unit number for output COVERX file
Select one of these			
bcd			Input file is in ASCII format.
binary			Input file is binary.
to	t	to	Keywords before this flag are for the input file. After this flag, they are for the output file.
Select one of these			
bcd			Output file is in ASCII format.
binary			Output file is binary.
new=		no	process an input binary COVERX in the new file format yes - characters were printed as characters no - characters were printed as floats All files produced with the current AMPX code system are of type new
dec=		no	processing old COVERX binary file generated on a DEC Alpha in BIG_ENDIAN format yes - DEC Alpha in BIG_ENDIAN format no - Not DEC Alpha in BIG_ENDIAN format dec=yes should only need to be specified with the option new=no. All files produced with the current AMPX code system are of type new
strip		no	If present, strip undesired reaction values. If present, only retain covariances matrices where the mt values are 1, 2, 4, 16, 18, 101, 102, 103, 104, 105, 106, 107, 452, 1018.

Logical Unit Parameters

Variable	Unit number	Type	Description
in		BCD or binary	logical unit number for input COVERX file
out		BCD or binary	logical unit number for output COVERX file

A-10. COMBINE - ADD, SUBTRACT, MULTIPLY OR DIVIDE TAB1 FILES

COMBINE is used to add, subtract, multiply or divide TAB1 files.

Input Data

Block Specifications

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
in1=		31	input TAB1 file
in2=		32	input TAB1 file
out=		33	output TAB1 file
con=		1.0	constant with which to multiply data in in2
option=		1	procedure to perform add - add the two tab1 files sub - subtract in2 from in2 mul - multiply the two tab1 files div - divide in1 by in2

Logical Unit Parameters

Variable	Unit number	Type	Description
out		binary	output TAB1 file
in in		binary	
		binary	

A-11. COMPARE - MODULE TO COMPARE FUNCTIONS ON TWO TAB1 FILES

COMPARE is a module to read two TAB1-formatted single precision binary files and compare the functions with the same identifiers (MAT, MF, MT). It writes a TAB1 single precision binary file that contains difference functions, (Function 1 - Function 2) / Function 2, identified by the original identifiers. This module can be used to compare two pointwise cross section files. For example, COMPARE can be used to compare point cross sections from AMPX with point cross sections generated by NJOY. Note that the AMPX module EXTRACT would be used to convert an NJOY-PENDF to TAB1 format. Subsequently, COMPARE would be used to compare the two TAB1 files.

Input Data

Block 1

-1$ Core allocation [1]

	1.	ICORE	number of words of core to allocate (500000)

0$ Logical unit assignments [3]

	1.	LOG1	logical unit on which the first TAB1 file is located (1)
	2.	LOG2	logical unit on which the second TAB1 file is located (2)
	3.	LOG3	logical unit where the difference TAB1 file will be written (3)

Terminate Block 1 with a T.

Sample Input

```
0$$ 23 24 25 T
```

This input indicates that the identical functions on logical units 23 and 24 should be compared and the difference file should be written in the TAB1 format on logical unit 25.

Logical Unit Parameters

Variable	Unit number	Type	Description
LOG1		binary	logical unit on which the first
LOG2		binary	TAB1 file is located in the logical unit on which the second TAB1 file is located
LOG3		binary	logical unit where the difference TAB1 file will be written

A-12. COMPRESS - MODULE TO COMPRESS FUNCTIONS WRITTEN IN TAB1 FORMAT

COMPRESS is a module which reads a point TAB1 data file written by a program such as POLIDENT and reduces the number of points in the functions on the file by eliminating points which can be interpolated to within a user-specified tolerance. For example, POLIDENT typically generated functions that are accurate (in terms of generating a function according to ENDF/B specifications, not according to physical correctness) to within 0.1%. Many applications may only need functions that are accurate to a much coarser tolerance, such as 1%. COMPRESS allows this operation to be performed.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
LOGIN=	IN	1	logical unit of input file
LOGOUT=	OUT	2	logical unit of output file
EPS=			tolerance to which points are tested to see if they can be eliminated Note that EPS is the relative difference, (A-B)/A, not the percentage difference. A value of 0.01 is equivalent to 1%.

Sample Input

```
IN=1 OUT=2 EPS=0.005 END
```

This input incicates that data should be read from the TAB1 file on logical unit 1 and that a TAB1 file should be written on logical unit 2 with functions accurate to within 0.5% of the original ones.

Logical Unit Parameters

Variable	Unit number	Type	Description
LOGIN		binary	logical unit of input file
LOGOUT		binary	logical unit of output file

A-13. COVCOMP - COMPARE TWO COVERX FILE OR ADD COVERX FILES ACCORDING TO A GIVEN PERCENTAGE

CPVCOMP compares two COVERX files or adds/subtracts COVERX file data. If comparing, the program compares the files and writes the differences into a new COVERX formatted file. In addition, it writes summary information to the screen.

Input Data

Block Keyword-Based Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
in=			space-separated list of input file logical units If negative, file is assumed to be binary.
perc=			Space-separated list of percentages associated with those units If not given, 1 is assumed for all input matrices. It is only used if adding covariance matrices.
fac=			space-separated list of factors to apply to covariance If not given, 1 is assumed. This is only useful when subtracting covariances matrices. If negative percentages are given, cross section data are subtracted, but covariance data are added. It is only used if adding covariance matrices.
out=		3	Logical unit of the output COVERX file
nytype=		-1	Type of the output COVERX file -1 - Use the type used in the first COVERX file 1 - Covariance matrix, standard deviation 2 - Relative covariance matrix and deviation 3 - Correlation matrix, standard deviation
eps=		1e-5	Precision to which to compare COVERX file data.
nullVal=		-9999	Value to substitute if matrix not found.
all			If comparing two COVERX files, print all differences.
convert			If comparing two COVERX files, always convert to ntype=1.
skip			Skip cross section data and matrices unless on all COVERX files.
add			If present, add the covariance matrices.

Notes

File format for the output file is a COVERX file with the following features:

- ☐ All cross section, uncertainties and covariance data are written out as abs(a1-a2)/abs(a1), where a1 is the value in file 1, and a2 is the value in file 2. If a1 is null, the value abs(a1-a2) is used instead.
- ☐ If a cross section or matrix does exist in one file but not the other, -9999 is written for all the values.

☐ If the group structures in the two files do not agree then the files cannot be compared. In this case, the COVERX file does contain a header but contains 0 cross section and covariance matrix data.

Sample Input

```
in=1 -2 out=-3 eps=1e-5 all
```

The BCD formatted COVERX files on logical units 1 and the binary COVERX file on unit 2 are compared, and differences are printed in binary format on logical unit 3. In addition, all differences larger than 1e-5 are printed to the screen

Logical Unit Parameters

Variable	Unit number	Type Description
log1		BCD or binary
log2 out		BCD or binary
		BCD or binary logical unit of the output COVERX file

A-14. COVCONV - PROGRAM TO CONVERT FILE 32 RESONANCE DATA INTO FILE 33 FORMAT

The program takes a COVERX file and converts the data pertaining to the File 32 information into the File 33 format. The group structure of the COVERX file is expected to contain all energy range end points from File 32. In addition, File 33 cannot contain any covariance information that overlaps with the covariance information in File 32.

Input Data

Block 1

0$ Logical unit assignment [8]

1.	cov	logical unit for the coverx file containing File 32 covariance data (-1) If negative, file is assumed to be binary.
2.	endf	logical unit for endf (2) any file 33 data in this file will be combined with the newly created File 33 data
3.	inmode	ENDF library format (2) 1: binary 2: BCD
4.	mat	material identifier
5.	out	logical unit for output of new File 33 data (2)
6.	outmo	ENDF library format for output file (2) 1: binary 2: BCD
7.	unres	option for whether unresolved parameter matrix get translated (0) 0: yes 1: no
8.	lty0	specifies how to treat lty=0 sections (0) 0: Do not allow to extend lty=0. Any energy of an overlapping lty=0 section will be automatically adjusted 1: Allow to extend lty=0

Terminate Block 1 with a T.

Logical Unit Parameters

Variable	Unit number	Type Description
cov		BCD or binary logical unit for the coverx file containing File 32 covariance data
endf		BCD or binary logical unit for endf
out		BCD or binary logical unit for output of new File 33 data

A-15. COVERR - PROGRAM TO CONVERT COVERX FILES TO ERRORR COVARIANCE FILES

COVERR is a program to convert coverx files to errorr covariance files. It can only convert COVERX files that contain one nuclide. Module CADILLAC should be used to select the desired material prior to running COVERR if the coverx formatted file contains more than one nuclide.

Input Data

Block 1

0$ Logical unit assignment [2]

1.	log1	logical unit for the first COVERX file (1) If negative, file is assumed to be binary.
2.	out	logical unit for errorr file (2)
3.	mat	ENDF mat number to use in errorr file (0)

1* ENDF header information [2]

1.	za	ZA value for the nucleus (0) This is the value written in the BOXER file.
2.	awr	mass ratio for the nucleus (0) This is the value written in the BOXER file.

Terminate Block 1 with a T.

Logical Unit Parameters

Variable	Unit number	Type	Description
log1		BCD or binary	Logical unit for the first coverx file
out		BCD	Logical unit for errorr file

A-16. FABULOUS_URR - MODULE TO PRODUCE BONDARENKO FACTOR TABLES

FABULOUS is a module that produces full-range Bondarenko factor tables from ENDF/B evaluations. It does not read the ENDF/B evaluation directly, but instead uses Doppler-broadened point data produced by modules POLIDENT and BROADEN. If the evaluation contains unresolved resonance data, the unresolved point data must be created at the desired temperatures and background values by module PRUDE if factors from statistical integrals are desired. Alternatively, probability tables generated by PURM and PURM_UP can also be used in the URRR. In order to produce infinite dilute cross section data consistent with the 1-D neutron data, an AMPX master library containing group-averaged neutron cross section data is required. FABULOUS does not perform any Doppler broadening; instead it assumes that all point data have been created at the desired temperatures.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
out=		1	unit for the output file containing the Bondarenko factors
in=		19	unit for the master containing the 1-D and 2-D neutron data
idlib=			identifier of the nuclide on the input master library
idpoint=			identifier of the material for the point data, the probability tables and the kinematic data
matwt=		99	material identifier of the flux data
mtwt=		2099	reaction identifier of the flux data
flux=		46	unit for the file containing the flux data
resol=			unit for the file containing the temperature dependent point-wise data This file contains all point-wise data in broadened to the desired temperatures. If no file is given, Bondarenko factors will not be calculated. This can be used to calculate f-factors in the URR only.
urrpoint=			unit for the file containing the temperature and background dependent point-wise data in the URR This file contains all point-wise data in broadened to the desired temperatures and calculated at the desired background cross section values.
urrprob=			unit for the file containing the temperature dependent probability tables in the URR
kin=			unit for the file containing point-wise kinematic data This file is needed if removal f-factor values are desired
temps=			space-separated list of temperature(s) in Kelvin at which f-factors should be generated
sig0=			space-separated list of background cross section values at which f-factors should be generated
mts=			space-separated list of additional reactions at which f-factors should be generated By default f-factors are generated for mt=1, 2, 18, 102, 1007, 1008, 2022 provided the required data. are available. If additional reactions are desired, they can be added in this array.

A-17. FABULOUS - MODULE TO PRODUCE BONDARENKO FACTOR TABLES

FABULOUS is a module that produces full-range Bondarenko factor tables from ENDF/B evaluations. It does not read the ENDF/B evaluation directly, but instead uses Doppler-broadened point data produced by modules POLIDENT and BROADEN. If the evaluation contains unresolved resonance data, the unresolved point data must be created at the desired temperatures and background values by module PRUDE. In order to produce infinite dilute cross section data consistent with the 1-D neutron data, it is strongly advised to supply an AMPX master library containing group-averaged neutron cross section data. FABULOUS does not perform any Doppler broadening; instead it assumes that all point data have been created at the desired temperatures and will terminate otherwise.

Input Data

Block 1

TITLE: Title to describe the Bondarenko factor set Type: Character*72

-1$	Core allocation [1]		
	1.	ICORE	number of words of core to allocate (500,000)
0$	Logical unit assignments [4]		
	1.	MMT	logical unit of the AMPX master library (1)
	2.	MXS	logical unit of the Doppler-broadened point data file (31)
	3.	MWS	logical unit of the Weighting Spectrum (46)
	4.	MUN	logical unit of the unresolved data from PRUDE (0)
1$	Primary parameters [5]		
	1.	IDSET	identifier of the Bondarenko factors in the master library
	2.	MAT	material identifier of the nuclide to be processed
	3.	NTEMP	number of temperatures in the Bondarenko factor tables
	4.	NSIG0	number of sig0-values in the Bondarenko factor tables
	5.	IGM	number of neutron energy groups
2$	Weight function selection parameters [2]		
	1.	MATWT	the MAT number for the weighting function
	2.	MTWT	the MT number for the weighting function
3$	Additional options [11]		
	1.	NEXTRA	number of extra cross sections for which Bondarenko factors are to be made (0) By default, Bondarenko factors for total, elastic scattering, fission, and capture will be produced.
	2.	LIST1D	option to print the 1-D cross section (0) 0: no 1: yes
	3.	LISTBF	option to print the Bondarenko factors (0) 0: no 1: yes
	4.	IDEBUG	option to print debug information (0) 0: no 1: yes

5.	master	unit of master to use for reference cross section data if desired (0). If not given, the group-averaged data calculated from the point data are used.
6.	masterID	Nuclide id on master to use as reference cross section (MAT)
7.	moderator	Option to select to indicate whether this a moderator (0) 0: no 1: yes For a moderator, the Bondarenko factors in the thermal range are set to 1.
8.	iftg	position of first thermal group (0)
9.	IOPT5	not used (0)
11.	IOPT7	not used (0)
19.	IOPT6	not used (0)

4* Energy range vver which Bondarenko factors are generated [2]

1.	ELO	lower energy of range (1e-5)
2.	EHO	upper energy of range (2e7)

5* Floating point parameters [2]

1.	AWR	mass ratio for nuclide
2.	EPS	accuracy to which integration is to be converged (0.0001)

Terminate Block 1 with a T.

Block 5

7* Energy group limits [IGM+1]

1.	IGMS	energy group limits The boundaries are not needed if a standard AMPX group structure is used. Enter values high to low in energy in eV

8* Temperatures [NTEMP]

1.	TEMPS	temperatures at which Bondarenko factors are desired

9* Sig0s [NSIG0]

1.	SIG0S	Sig0 values at which Bondarenko factors are desired

10$ EXTRA_CROSS [NEXTRA]

1.	extras	extra cross sections for which to generate Bondarenko data

Terminate Block 5 with a T.

Sample Input

```
0$$ 1 31 46 32 1$$ 1000 1395 3 8 238 2$$ 8000 99
5** 235.0 E T
8** 300 900 2100
9** 1.0E8 1.0E5 1.0E4 1.0E3 1.0E2 10.0 1.0 1.0E-6
T
```

This input indicates that a master library should be writtenon logical unit 1, point data should be read on logical unit 31, a weighting function should be read in TAB1 format on logical unit 46, and point unresolved resonance data should be read on logical unit 32. The tables included on the AMPX master library will be identified by 1000; the MAT number is 1395; Bondarenko factors are to be produced at three temperatures and eight values of the background cross section. The mass ratio is 235.0. The temperatures are 300, 900, and 2100 K. The background cross sections are 1.0E8, 1.0E5,......1.0E-6. (Note that the group structure for 238 groups is not specified since it is a standard AMPX group structure and will be automatically accessed).

Logical Unit Parameters

Variable	Unit number	Type	Description
MMT		binary	logical unit of the AMPX master library
MXS		binary	logical unit of the Doppler-broadened point data file
MWS		binary	logical unit of the weighting spectrum
MUN		binary	logical unit of the unresolved data from PRUDE
	77	binary	Scratch

A-18. FILTER - SELECT SPECIFIC DATA FROM A MASTER OF WORKING LIBRARY

FILTER allows for selection of a specific data type from a master or working library.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
work			processes a working library If present, a working library is processed
in=		1	logical unit of the input library
out=		2	logical unit of the output library

Repeat block data as often as needed.

Block Data

Block starts on first encounter of a keyword in the block.

Block terminates on encountering the next occurrence of keyword end or neutron or gamma or yield or resonance or bondarenko or 1dn or 2dn or 1dg or 2dg or 2dy or end

Keyword	Alternate	Default	Definition
Select one of these			
NEUTRON			includes all neutron data
GAMMA			includes all gamma-ray data
YIELD			includes gamma-ray yield data
RESONANCE			includes resolved resonance parameters
BONDARENKO			includes Bondarenko factor data
1DN			includes 1-D neutron data
2DN			includes neutron scattering matrices
1DG			includes 1-D photon data
2DG			includes photon scattering matrices
2DY			includes photon production matrices
mt=			List of reaction values to include or exclude If all mt values are positive, the listed mt values will be selected from the partial library and added to the new library. If all mt values are negative, the listed mt values are excluded from the new library.

Notes

Data selected in the data block will only be included in the new library if they are present on the old library. If processing a working library, either 2dn or 2dg will select the transfer matrix.

Logical Unit Parameters

Variable	Unit number	Type	Description
IN		BCD	
OUT		binary	

A-19. FUNCCALC - CALCULATE ARBITRARY FUNCTION

FUNCCALC calculates an arbitrary function using the data given on a tab1-formatted data file. The function is calculated using up to three sets of (mat,mt) values from the tab1-formatted data file. The (mat,mt) values are assumed to be unique. The module PICKEZE can be used to select the desired sets. These sets are denoted as function values 1,2, or 3 below. Values are created over the range el to eh using as many points as needed to create the function up to precision eps.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
in=		10	logical unit of the input library
out=		11	logical unit of the output library
el=		1e-5	lower limit of the function to create
eh=		2e7	upper limit of the function to create
eps=		1e-4	precision to which to create the function
id1=		99	material id of the function to create
id2=		1099	reaction id of the function to create

Repeat Block Command as often as needed

Block Command

Block starts on first encounter of a keyword in the block.

Block terminates on encountering the next occurrence of keyword com or end or mat

Keyword	Alternate	Default	Definition
com=			calculation to perform vr - sets register ireg to value in creg sr - sets register ireg to function value at creg ar - adds value in register creg to value in register ireg and stores in ireg mr - multiplies value in register creg with value in register ireg and stores in ireg dr - divides value in register ireg by value in register creg and stores in ireg cr - sets ireg to 0 er - takes exponential of register ireg and stores in ireg lr - takes logarithm of register ireg and stores in ireg sv - takes the value in register ireg as the final function value
creg=			register name, see description for com
ireg=			register name, see description for com

Repeat block functions up to 3 times.

Block Functions

Block starts on first encounter of a keyword in the block.

Block terminates on encountering the next occurrence of keyword mat or end or com.

Keyword	Alternate	Default	Definition
mat=			Material value of function to access on the tab1 formatted file
mt=			Reaction value of function to access on the tab1 formatted file

Sample Input

```
in=10 out=20 id1=99 id2=1099
mat=9237 mt=18 mat=9347 mt=1 mat=99 mt=2099 com=sr ireg=1 creg=1
com=sr ireg=2 creg=2 com=sr ireg=3 creg=3 com=mr ireg=1 creg=2 com=vr
ireg=4 creg=1e10 com=ar ireg=3 creg=4 com=dr
ireg=1 creg=3 com=sv
```

Load the cross section for (9237,18) in function 1, (9347,1) in function 2 and the flux (99,2099) in function.

3. For each point to be calculated, the registers are filled as follows:

1. Load (9237,18) or function 1 into register 1.
2. Load (9437,1) or function 2 into register 2.
3. Load (99,1099) or function 2 into register 3.
4. Multiply register 1 by register 2 and store in register 1.
5. Store a user supplied value of 1e10 in register 4.
6. Add register 3 and register 4 and store in register 3.
7. Divide register 1 by register 3 and store in register 1.
8. Save the value in register 1 as the final function value.

A-20. IRFFACHOMO - MODULE TO PRODUCE HOMOGENOUS F-FACTORS

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
out=		77	unit for the output file containing the final Bondarenko factors
in=		78	unit for the master containing the 1-D and 2-D neutron data
fnuc=			identifier of the resonance nuclide to use
bnuc=			identifier of the background nuclide to use
dens=			density value to use for the resonance nuclide in the infinite medium calculation
ehres=			upper limit of the RR of the resonance nuclide
bcut=		1e-4	lowest possible density for the background nuclide
low=		0	lowest group for which to generate homogenous f- factors If 0, the group containing the upper end of the RR is selected
high=		0	highest group for which to generate homogenous f- factors If 0, the last group is selected

A-21. IRFFACTOR - MODULE TO CALCULATE INTERMEDIATE RESONANCE F-FACTORS BASED ON HETERO CELLS

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
in=			the unit number of the input cross section library
out=			the unit number of the output cross section library
fnuc=			resonance absorber nuclide for which to calculate the ffactors
mopt=		1	option for treating moderator XSs (see parameter ircalc in I_Crawdius input) 0 - include energy-dep PW XS's [standard CENTRM lib data] 1 - treat as IR moderator => no abs.; elastic=lambda*sigp 2 - treat as IR absorber => has abs.; elas=lambda*sigp; tot=abs+elas]
absopt=		0	option for treating absorber lam*sigp (see parameter ircalc in I_Crawdius input) 0 - do NOT include absorber lambda*sigp in background XS] 1 - include absorber lambda*sigp in background XS]
medit=		0	option for treating moderator XSs (see parameter ircalc in I_Crawdius input) 0 - no edits 1 - edit background XS's obtained for cells 2 - also edit final f-factors]
nterp=		1	interpolation method for f-factors 0 - Segev interpolation 1 - Spline interpolation taken from GSL TPL
check=		no	If yes, only perform checking; if no, perform full execution yes - input checked and background XS values are edited if medit > 0 no - input not checked
essm=		yes	yes - background XS is computed using essm method no - Bonami background XS is used yes - the background XS is computed using essm method no - use Bonami background XS
iter=		yes	yes - inner iterations are performed in the computation of essm background XS yes - essm background XS is computed using inner iterations in MG flux calc) no - essm background XS is computed using NO inner iterations (=> within grp XS=0)
cut=		1e-9	lower cut-off value for the density of background nuclide
bcut=		1e-5	lower cut-off value for f-factors (Values will be set to previous sig0 value.)
elow=		1e-3	lowest energy for which to calculate f-factors
ehigh=		2e+5	highest energy for which to calculate f-factors

Keyword	Alternate	Default	Definition
			If zero, the highest energy of the input master is used.
ehres=		0.0	upper energy of the resolved resonance range A value of 0 indicates that energy ehigh is to be used as upper energy bound.
removal=		yes	options for computing within-group scatter f-factors yes - add removal f-factors no - do not add removal f-factors
irmt=		2000	mt value for the lambda factors
cellfil=			full path to file containing scale csas input defining heterogeneous cases The input string must be enclosed in quotes.

A-22. JAMAICAN - MODULE TO THIN POINT-WISE 2-D DATA

The point-wise 2-D data created by module MONTEGO can contain a dense mesh of exit energies and angles. If converting to marginal and conditional probabilities, the mesh can often be thinned. This module thins the mesh and writes the data out in a format suitable for use in PLATINUM. The program typically uses equiprobable angle bins except for elastic and discrete inelastic reactions. However, if the distribution can be described with a number of non-equiprobable angle bins using less than nang angles, non-equiprobable angle bins are used.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
mon=		20	file containing kinematic data in double differential form
out=		21	output file in platinum format
nang=		32	maximum number of equiprobable energy bins
eps=		1e-3	precision to which to calculate the distributions.
nothin			(If present, no thinning is performed.)
form=		native	file format of the input file native - native format

A-23. JERGENS - MODULE TO GENERATE WEIGHT FUNCTIONS AND TO COMBINE ENDF/B TAB1 RECORDS

JERGENS (Just an Excellent Routine to Generate Strings) is the module used to construct a weighting spectrum such as that needed by X10 or PRILOSEC. The output from the module is a file containing the new functions written in TAB1 format. JERGENS is used to generate certain predefined functions. If arbitrary functions are needed, use module FUNCCAL.

Input Data

Block 1

-1$ Options [20]

1.	intlow	index for the lowest interpolation value (2) The list of allowed endf interpolation values (1-5) is given in inter1, inter2, ..., inter5. The value intlow gives the index of the lowest interpolation value to try. The user should typically set intlow and inthigh to 2 and inter1=1, inter2=2, ... This allows only linear-linear interpolation in the generated weighting function.
2.	inthigh	index for highest interpolation value (2) (see intlow for more detailed explanation)
3.	IOPT3	not used (100)
4.	mat	MAT number for the weighting functions (99)
5.	mf	MF (file) number for the weighting function (3)
6.	inter1	first interpolation scheme used (1)
7.	inter2	second interpolation scheme used (2)
8.	inter3	third interpolation scheme used (3)
9.	inter4	fourth interpolation scheme used (4)
10.	inter5	fifth interpolation scheme used (5)
11.	ICORE	not used (100000)
12.	OPTS	not used

0$ Logical assignments [3]

1.	NDFB	All external functions required by JERGENS must reside here. The current version of JERGENS does not allow the use of external functions.
2.	MWT	the logical unit of the output file
3.	MSC	not used (18)

1$ Problem information [1]

1.	NMWT	number of functions to be written on MWT

2* Energy Range [2]

1.	ELO	low-energy cutoff of functions to be generated (in eV) (0.00001)
2.	EHI	high-energy cutoff of functions to be generated (in eV) (2.0e7)

Terminate Block 1 with a T.

Repeat Block 2 NMWT times.

Block 2

3$ Identifier and option selectors [3]

1.	IDWT	identifier for function to be created (equivalent to the MT number in ENDF/B)	
2.	NC	number of commands associated with the construction of this function (0) The current version of JERGENS only allows creation of predefined dose and weighting functions.	
3.	IW	Options for the desired dose function or weighting function	
		0:	1/E
		1:	flat
		2:	Maxwellian - 1/E - fission spectrum
		3:	E
		4:	Maxwellian - 1/E - fission spectrum - 1/E above 10 MeV
		5:	neutron dose factors per ANSI/ANS 6.1.1-1977
		6:	gamma-ray dose factors per ANSI/ANS 6.1.1-1977
		7:	1/V (normalized to 1.0 at 2200 m/s)
		8:	Henderson neutron dose factors in (Rads/hr)/((photons/cm**2)/sec)
		9:	silicon gamma dose factors in (Rads/hr)/((photons/cm**2)/sec)
		10:	Claiborne-Trubey gamma dose factors in (Rads/hr)/((photons/cm**2)/sec)
		11:	1/E function with high and low cutoffs
		12:	Watt fission spectrum
		9031:	ANSI 6.1.1-1992 Neutron Dose Factors
		9032:	air neutron kerma factors in (Gr/hr)/((neutrons/cm**2)/sec)
		9033:	air neutron kerma factors in (Rad/hr)/((neutrons/cm**2)/sec)
		9034:	dose equivalent factors in (Sv/hr)/((neutrons/cm**2)/sec)
		9035:	dose equivalent factors in (Rem/hr)/((neutrons/cm**2)/sec)
		9036:	neutron effective dose factors in (Sv/hr)/((neutrons/cm**2)/sec)
		9037:	neutron effective dose factors in (Rem/hr)/((neutrons/cm**2)/sec)
		9505:	ANSI 6.1.1-1991 gamma dose factors in (Rads/hr)/((photons/cm**2)/sec)
		9502:	Henderson gamma dose factors in (Rads/hr)/((photons/cm**2)/sec)
		9506:	gamma air kerma factors in (Greys/hr)/((photons/cm**2)/sec)
		9507:	gamma air kerma factors in (Rad/hr)/((photons/cm**2)/sec)
		9508:	dose equivalent factors in (Sv/hr)/((photons/cm**2)/sec)
		9509:	dose equivalent factors in (Rem/hr)/((photons/cm**2)/sec)
		9510:	gamma Effective Dose Factors in (Sv/hr)/((photons/cm**2)/sec)
		9511:	gamma effective dose factors in (Rem/hr)/((photons/cm**2)/sec)
		9029:	neutron dose factors per ANSI/ANS 6.1.1-1977
		9504:	gamma-ray dose factors per ANSI/ANS 6.1.1-1977
		9027:	Henderson Neutron Dose Factors in (Rads/hr)/((photons/cm**2)/sec)
		9503:	Claiborne-Trubey Gamma Dose Factors in (Rads/hr)/((photons/cm**2)/sec)

4* Constants [6]

1.	TMAX	temperature of the Maxwellian spectrum (K) (300) If a Watt fission spectrum is generated, then this is the value of a in exp(-e/a)*sinh(sqrt(b) e) in units of MeV.
2.	AKT	multiplier on KT to determine Maxwellian to 1/E join point (5) If a Watt fission spectrum is generated, then this is the value of b in exp(-e/a)*sinh(sqrt(b) e) in units of MeV.
3.	THETA	effective temperature in eV of the fission spectrum (1.27e6)
5.	FCUT	point at which to join I/E to fission spectrum (67.4e3)
6.	SIGD	not used
6.	EPS	accuracy to which functions are to be generated (0.0001)

Terminate Block 2 with a T.

Notes

The combination of IOPT1, IOPT2 with INT1, INT2, ... INT5 allows a very flexible method of selecting the kinds of interpolation schemes allowed in the functions produced. The interpolation schemes are as follows:

Code	Type
1	histogram
2	linear x - linea y
3	linear x - log y
4	log x - linear y
5	log x - log y

intlow points to the word in inter1...inter5 which contains the first interpolation code to be tried. inthigh points to the last word in the string containing the code to be tried. JERGENS cycles through the codes specified inter(intlow) to inter(inthigh) to determine the best code to use. By default, intlow and inthigh are both 2, indicating that a linear-linear function is being constructed. intlow = 2 and inthigh = 5 would try types 2, 3, 4, and 5 in exactly that order. Making intlow= 1 and inthigh = 2, and inter = 5 with inter2 = 2 would cycle between a log-log and a linear-linear scheme, etc. Attempting an interpolation of 1 (histogram) would be fruitless because accuracy specifications could never be satisfied. Therefore, it should be avoided.

Logical Unit Parameters

Variable	**Unit number**	**Type**	**Description**
NDFB		binary	all external functions required by JERGENS must reside here
MWT		binary	The logical unit of the output file

A-24. KINKOS - KINEMATICS KONVERSION SYSTEM

Kinematics Konversion System (KINKOS) is a module to convert kinematics files generated by module Y12 into different formats.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
INPUT=	IN	31	logical unit of input file
OUTPUT=	OUT	32	logical unit of output file
nl=		5	if converting from cosine moment to cosine moment format, the maximum number of cosine moments to use
eps=		1e-3	precision to which the cosine grid is to be constructed
Select one of these			
y12_d			input file is double precision y12 format
y12_s			input file is single precision y12 format
kfc			input file is kfc format
mon			input file is montego format
native			input file is native format
to			Keywords before this flag are for the input file. After this flag, they are for the output file.
Select one of these			
y12_d			Output file is double precision y12 format.
y12_s			Output file is single precision y12 format.
kfc			Output file is kfc format.
mon			Output file is montego format.
native			Output file is native format.
ascii			Output file in ascii.
format=		cos	if saving in native format, the format to which the data should be converted cos - Save cosine moments leg - Save as Legendre moments tab - Save in tabulated form
fbot=		1e-5	if lopping is switched on, the fraction to remove from the bottom of the distribution
ftop=		1e-5	if lopping is switched on, the fraction to remove from the top of the distribution
upscatter			correct mt=1007 for upscatter
lop			lops small fraction from the exit energy distribution
id=		0	the new id to use for the data if id change is desired
eup=		3.0	if applying upscatter correction, the highest energy which can have upscatter
eterm=		5.0	if applying upscatter correction, the highest energy for thermal matrices

Keyword	Alternate	Default	Definition
cross=		0	unit for the cross section data in TAB1 format
awi=		1.0	mass ratio of incident particle (needed if converting com to lab)

Logical Unit Parameters

Variable	Unit number	Type	Description
input		binary/ASCII	
output		binary/ASCII	

A-25. KINZEST - MODULE TO MANAGE KINEMATIC LIBRARIES

KINZEST (Zippy Ensembler of Strings) is a module analogous to ZEST, except it treats kinematic libraries.

Input Data

Block 1

0$ Logical assignments [2]

1.	LOG	logical unit of library to be written (31)
2.	NLOG	number of commands (or libraries) required to create LOG (1)

Terminate Block 1 with a T.

Stack Blocks 2 and 3 one after the other NLOG times.

Block 2

2$ Input library selection [2]

1.	NLIN	logical number of input library
2.	NC	Options for how the strings are to be treated (0) ☐ -N: deletes N strings from NLIN to create LOG ☐ 0: accepts all strings from NLIN ☐ N: adds N strings from NLIN to create LOG

Terminate Block 2 with a T.

Only use Block 3 if NC != 0.

Block 3

3$ MAT numbers from NC [NC]

1.	MAT	material identifier(s) of nuclides to be added or deleted. (0) Only used if NC != 0. There must be exactly NC values.

4$ New MAT numbers from NC [NC]

1.	MATnew	new material identifier(s) of nuclides to be added. (0) Only used if NC > 0. A zero leaves the identifier unchanged.

5$ MT numbers from NC [NC]

1.	MT	reaction identifiers of nuclides to be added or deleted (0) Only used if NC != 0. There must be exactly NC values.

6$ New reaction numbers from NC [NC]

1.	MATnew	New reaction identifier(s) of nuclides to be added. (0) Only used if NC > 0 A zero leaves the identifier unchanged.

7*	awp values to preserve/delete [NC]	
	1. awp	values of awp to keep or to delete. (0) Only used if NC > 0
8*	zap values to preserve/delete [NC]	
	1. zap	values of zap to keep or to delete. (0) Only used if NC > 0

Terminate Block 3 with a T.

Logical Unit Parameters

Variable	Unit number	Type	Description
LOG		binary	logical number of library to be written
NLIN		binary	

A-26. LAMBDA - MODULE TO PRODUCE LAMBDA FACTORS

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
out=		77	unit for the output file containing the final Bondarenko factors
in=		78	unit for the master containing the 1-D and 2-D neutron data
fnuc=			identifier of the fissionable nuclide to use
bnuc=			identifier of the background nuclide to use
dens=			density value to use for the resonance nuclide in the infinite medium calculation
bdens=			density value to use for background nuclide in the reference case
iddens=			density value to use for resonance nuclide in the reference case
feps=		1e-3	lower limit to determine there is no fluctuation If the standard deviation of the values for different background densities falls below this value, it is assumed that lambda for this group cannot be calculated, so it is set it to 1.
eps=		1e-3	used to determine whether enough background values have been added for calculation
bcut=		1e-4	lowest possible density for the background nuclide
temp=		293	temperature at which to perform the calculation
low=		1	lowest group for which to generate lambda factors
high=		0	highest group for which to generate lambda factors If 0, the last group is selected.
lcut=		1e-5	lowest possible number density for the background nuclide
hcut=		1e5	highest possible number density for the background nuclide
irmt=		2000	reaction value to use for the generated lambda factors

A-27. LAVA - AMPX MODULE TO MAKE AN AMPX WORKING LIBRARY FROM AN ANISN LIBRARY

LAVA (Let ANISN Visit AMPX) is a module that converts an ANISN library (neutron, gamma, or coupled neutron-gamma) to an AMPX working library such as those used in XSDRNPM. ANISN cross sections can be entered on cards (fixed or free-form FIDO format) or on a binary library.

Input Data

Block 1

-1$ Core assignment [1]

	1.	NWORD	number of words to allocate (50,000)

0$ Logical definitions [4]

	1.	N1	ANISN library (20)
	2.	N2	AMPX working library (4)
	3.	N3	scratch (18)
	4.	N4	scratch (19)

1$ ANISN library parameter data [8]

	1.	NNUC	number of isotopes to be put on new library
	2.	IGM	number of neutron groups
	3.	IHT	position of sigma_{total}
	4.	IHS	position of sigma_{g->g'}
	5.	IHM	table length
	6.	IFTG	first thermal group
	7.	IPM	number of gamma groups
	8.	IFM	format of ANISN library -1: binary 0: free-form BCD 1: formatted BCD
	9.	IFLAG	flag that selects the method for calculating scattering cross sections from scattering matrices (1) 0: sets elastic cross section to sum_{g'}(sigma_{g->g'}) 1: attempts to calculate the correct elastic cross section See notes for more details

Terminate Block 1 with a T

Block 2

2$ Identifiers of block of data for the nuclide on the ANISN library [NNUC]

	1.	NUCIDS	Identifiers of Block of Data for the Nuclide on the ANISN Library

3$ Order of scattering for the nuclide on the ANISN library [NNUC]

	1.	SCAT	Order of scattering for the nuclide on the ANISN library: If an order of scattering for a nuclide is negative, the P(l > 0) matrices for the nuclide will be multiplied by (2l+1) to account for differences in the way different computer programs require these to be normalized.

4$ AMPX identifiers for the nuclides selected from ANISN library [NNUC]

1. AMPXID — AMPX identifiers for the nuclides selected from ANISN library

5$ Process identifiers for the top positions in the ANISN cross section tables [IHT]

1. PROCID — Process identifiers for the top positions in the ANISN cross section tables
   The order is from position IHT to position 1 (i.e., backwards from the way it is in the cross section tables). ANISN always expects sigma_{total} in position IHT, with nu*sigma_{f} above that, and sigma_{a} a above that. The contents of the other positions are arbitrary.

6* Fission spectrum [IGM]

1. FISSION — Fission spectrum
   If a nuclide has a nonzero fission cross section, and no fission spectrum (MT=1018) is specified in the ANISN library or the fission spectrum (CHI) flag for that nuclide has been set in the 9$ array, then the fission spectrum specified in the 6* array is used for that nuclide.

7* Neutron energy group boundaries [IGM+1]

1. IGMS — Neutron energy group boundaries
   Read high to low in energy (eV)

8* Gamma-ray energy group boundaries [IPM+1]

1. IPMS — Gamma-ray energy group boundaries
   Read high to low in energy (eV)

9$ Nuclide CHI flags [NNUC]

1. CHIS — Nuclide CHI flags
   If 0, use the fission spectrum from the ANISN library; if 1, use the fission spectrum from the 6* array

Terminate Block 2 with a T

Notes

ANISN matrices are the sum of the individual scattering matrices (elastic, inelastic, n2n, n3n, etc.) for processes possible for the particular nuclide. LAVA attempts to arbitrarily determine values for an elastic (MT = 2) and an n2n (MT = 16) cross section, recognizing that elastic scattering is generally the most dominant scattering process, and that n2n is the most common scattering process that yields more than a single exit neutron. In order to accomplish this, the absorption cross section in the ANISN data must be the true absorption value (not an energy absorption cross section as in some older gamma- ray sets, or whatever alternative value). When IFLAG = 1, requiring the correct absorption, the elastic value is taken as

sigma_{el}^{g} = sigma_{t}^{g} - sigma_{a}^{g}
while the n2n is taken from
sigma_{n2n}^{g} = sum_{g'} sigma_{0}(g -> g') - sigma_{el}^{g}
When IFLAG = 0, no attempt is made to calculate an n2n value, and the elastic value is simply
sum_{g'}(sigma_{g->g'})

Sample Input

```
0$$ 20 4 18 19 1$$ 5 16 3 4 16 16 0 0 0 T
2$$ 1 5 9 13 17
3$$ 333 3 3
4$$ 92235 92238 26000 1001 8016
6** Put in 16 numbers for a Fission Spectrum
T
```

This input will create an AMPX working library on logical unit 4 from an ANISN binary library on logical unit 20. The ANISN library is for 16 neutron energy groups and has the total cross section in position 3, the within-group scattering in position 4 with a table length of 16. There are no gamma groups. The ANISN identifiers are 1, 5, 9, 13, and 17 for the P0 parts of a P3 fit to ^{235}U, ^{238}U, Fe, ^{1}H, and ^{16}O, respectively. The energy group boundaries are not read since the 16-group structure is one of the standard AMPX structures.

Logical Unit Parameters

Variable	Unit number	Type	Description
N1		binary	ANISN library
N2		binary	AMPX working library
N3		binary	scratch
N4		binary	scratch
N5		binary	
N6		binary	
	47	binary	scratch

A-28. LINEAR - MODULE TO LINEARIZE FUNCTIONS WRITTEN IN TAB1 FORMAT

LINEAR is a module that will read a point TAB1 data file, which is written by a program such as POLIDENT, and linearize the data to within a user-specified tolerance.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
IN=		1	logical unit of input file
OUT=		2	logical unit of output file
FORCE=		yes	flag to force linearization even if it is already linear yes - Force linearization no - Do not force linearization
MORE=		no	flag to print arrays before and after linearization yes - print no - do not print
EPS=		0.001	tolerance to which points are tested to see if they can be linearly interpolated Note that EPS is the relative difference (A-B)/A, not the percentage difference. A value of 0.01 is equivalent to 1%.

Sample Input

```
    IN=23 OUT=24 EPS=0.005 END
```

This input indicates that data from the TAB1 file on logical unit 23 should be read, and a TAB1 file on logical unit 24 should be written with functions that can be linearly interpolated to within 0.5% of the original ones.

Logical Unit Parameters

Variable	Unit number	Type	Description
LOGIN		binary	
LOGOUT		binary	

A-29. LIPTON - CONVERT ASCII ENDF/B FILE THAT CONTAINS FILE 3, 9 AND 10 TO BINARY

LIPTON is a program to read an ASCII ENDF/B File that contains File 3, 9 and 10 data and create Tab1 binary records for File 3, 9 and 10 records. The resultant file can then be passed to PRILOSEC for processing. For Files 9 and 10, the MTs are redefined as MT*10000+LFS*100+LIS, and the functions are written as TAB1 triplets instead of using the multiple subsection scheme. File 9 functions are constructed by multiplying the appropriate cross sections from File 3 by the File 9 values. No attempt is made to form the product functions to a user-specified precision at present, though it may be done later.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
file3=		3	binary tab1 file containing File 3 data
file9=		9	binary tab1 file containing File 9 and 10 data
out=		10	result of combining File 3, 9 and 10 in binary tab1 format.

Logical Unit Parameters

Variable	Unit number	Type	Description
ndfb		ASCII	
tab1		binary	
	14	binary	scratch
	15	binary	scratch
	16	binary	scratch

A-30. MAKPEN - MODULE TO GENERATE CROSS SECTION DATA IN A PENDF FORMAT

MAKPEN (MAKe PENDF) is module that reads CE cross section data in a TAB1 format and generates a Point ENDF cross section file (PENDF). MAKPEN reads the File 1and abbreviated File 2 information from the POLIDENT logical output LOGP1. Subsequently, MAKPEN reads the user-specified TAB1 formatted cross section data and constructs a PENDF library. The code input is freeform with keyword definitions.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
title=			title line for the PENDF tape
in1=		31	TAB1 formatted cross section file
in2=		32	POLIDENT output file with File 1 and File 2 information this is the LOGP1 file generated by POLIDENT
iout=		40	PENDF cross section output file
tol=			POLIDENT convergence tolerance for energy mesh construction

Sample Input

```
title=u-238 endf6, polident generated cross sections title=processed by m.
e. dunn
in1=35 in2=32 iout=36 nnuc=1 tol=0.001
```

The input above can be used to convert an AMPX TAB1 file for ^{238}U to a PENDF format that can be processed by the NJOY code system. For the example case, the ENDF/B-6 evaluation for ^{238}U was processed through POLIDENT with an energy-mesh generation tolerance of 0.001 (i.e., 0.1%), and the TAB1 pointwise cross section file from POLIDENT is stored on logical unit 35. In the input for MAKPEN above, a title description is provided to define the point cross section file. Subsequently, the input TAB1 file is specified to be on logical unit 35. In addition, POLIDENT provides an output file with ENDF/B File 1 information, and the input in2=32 specifies that this information is located on logical unit 32. The PENDF created by MAKPEN will be produced on logical unit 36. Moreover, the sample input indicates that a single isotope/nuclide will be processed. Note that the energy-mesh generation tolerance is also specified in the MAKPEN input (i.e., tol=0.001).

Logical Unit Parameters

Variable	Unit number	Type	Description
in1		binary	TAB1 formatted cross section file
in2		binary	POLIDENT output file with File 1 and File 2 information
iout		binary	PENDF cross section output file
ltab1		binary	
	14	binary	scratch
	15	binary	scratch
	16	binary	scratch

A-31. MALOCS - MODULE TO COLLAPSE AMPX MASTER CROSS SECTION LIBRARIES

MALOCS (Miniature AMPX Library of Cross Sections) is a module to collapse AMPX master cross section libraries. The module can be used to collapse neutron, gamma-ray, or coupled neutron-gamma master libraries.

Input Data

Block 1

0$ Library Logical Unit Numbers [2]

1.	NOLD	logical number of device containing fine-group AMPX master library (1)
2.	NNEW	logical number of device containing broad-group AMPX master library (22)

1$ Case Description [6]

1.	NNEUT	number of neutron fine group
2.	IGMF	number of neutron broad groups
3.	NGAM	number of gamma-ray fine group
4.	IPMF	number of gamma-ray broad groups
5.	IWN	Neutron weighting option (0) 0: Input neutron weighting spectrum in the 5* array 1: Use MT=1099 1-D neutron data from each fine-group master data set for the neutron weighting spectrum. other: Use the 1-D data identified with an MT number of IOPT2 For values < 0, see 3$ array. Use the 1-D data identified with an MT number of IOPT2 spectrum for all neutron data sets being collapsed.
6.	IWG	gamma weighting option (0) 0: Input gamma-ray weighting spectrum in the 7* array 1: Use MT=1099 1-D gamma-ray data from each fine-group master data set for the gamma-ray weighting spectrum. other: Use the 1-D data identified with an MT number of IOPT6 For values < 0, see 3$ array. Use the 1-D data identified with an MT number of IOPT6 spectrum for all gamma-ray data sets being collapsed.

3$ Option Triggers [10]

1.	IOPT1	if > 0, identification number of master data for neutron weighting (0) Identification number of master data set from which the neutron weighting spectrum (IOPT2 data) will be obtained
2.	IOPT2	if > 0, process identifier (MT number) of neutron weighting spectrum in IOPT1 master data set (0)
3.	IOPT3	trigger to print broad-group 1-D cross section (0) 1: print data 0: do not print data
4.	IOPT4	trigger to print broad-group transfer matrices (0) 0: do not print data other: print arrays through order N

5.	IOPT5	auxiliary gamma-ray weighting spectrum trigger (0) if > 0, identification number of master data set from which the gamma-ray weighting spectrum (IOPT6 data) will be obtained.
6.	IOPT6	process identifier (MT number) of gamma-ray weighting spectrum in IOPT5 master data set (0)
7.	IOPT7	trigger to collapse out upscatter terms if nonzero (0) 0: no upscatter truncation (recommended) 1: XSDRNPM method of upscatter truncation 2: ANISN method of upscatter truncation 3: simple sum method of upscatter truncation 4: Non-negative ANSIN method of upscatter truncation
8.	IOPT8	trigger to truncate downscatters to a maximum of IOPT8 terms below the within group if IOPT8 is nonzero (0)
9.	IOPT0	not used (0)
10.	IOPT10	weighting spectrum printing option (0) 0: Do not print weighting spectrum 1: print weighting spectrum

Terminate Block 1 with a T.

Block 2

4$ Neutron broad-group numbers by fine group [NNEUT]

1.	NNEUTS	neutron broad-group numbers by fine group only used if NNEUT > 0 a zero "suppresses" a fine group.

5* Neutron weighting spectrum [NNEUT]

1.	NNEUTW	neutron weighting spectrum only used if IWN = 0

6$ Gamma-ray broad-group numbers by fine group [NGAM]

1.	NGAMS	gamma-ray broad-group numbers by fine group only used if NGAM > 0 When collapsing the gamma groups in a coupled master library, the 6$ entries are the actual group numbers and do not need to include the number of neutron groups.

7* Gamma-ray weighting spectrum [NGAM]

1.	NGAMW	gamma-ray weighting spectrum only used if IWG = 0

Terminate Block 2 with a T.

Sample Input

```
1$$ 16 4 0 0 0 0 2$$ 1 2 T
4$$ 4R1 4R2 4R3 4R4
5**
1.234E-7 5.697E-7 8.724E-6 9.412E-5
9.269E-5 8.193E-4 3.627E-4 8.463E-4
3.492E-4 8.624E-3 7.999E-4 3.224E-5
1.947E-5 2.333E-5 8.387E-5 4.417E-6
T
```

This input produces a collapsed AMPX master library on logical unit 2 with 4 neutron energy groups, starting with a master library in 16 energy groups on logical unit 1. Groups 1–4 become broad group 1, 5–8 become broad group 2, 9–12 become broad group 3, and 13–16 broad become broad group 4.

Logical Unit Parameters

Variable	Unit number	Type	Description
NOLD		binary	logical number of devices containing fine-group AMPX master library
N1		binary	
N2		binary	
	17	binary	scratch
	18	binary	scratch
	19	binary	scratch

A-32. MALT - MAKE ANISN LIBRARY TRANSFORMATION

This program converts a binary ANISN library to the ASCII format and vice versa.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
in=		31	logical unit of input file
out=		32	logical unit of output file
i1=			number of columns in the ANSIN output file
i2=			number of rows in the ANSIN output file
id=			ANISIN ID to use if reading an AMPX master
istart=			start of data If 0, the user assumes that only neutron 1-D data are wanted. If gamma data with the same mt value exist, they are added after the neutron data. If larger than 0, only the gamma data are added, and istart is the number of neutron groups.
Select one of these			
fixed			fixed ANISN library format
free			free ANISN library format
binary			binary ANISN library format
ampx			reads 1-D data from AMPX master
to		to	keywords before this flag are for input file. After this flag for output file
Select one of these			
fixed			fixed ANISN library format
free			free ANISN library format
binary			binary ANISN library format

Logical Unit Parameters

Variable	Unit number	Type	Description
log		binary or BCD	

A-33. MG_TO_KIN - CONVERT TOTAL MG SCATTERING MATRIX TO CE

This module converts a total scattering matrix given in a working format AMPX library into a double differential format suitable for processing in JAMAICAN. It is easier to generate a total scattering matrix in MG format, as the elastic and discrete inelastic scattering matrices are given in Legendre format, which is easily added together. This MG total scattering matrix can then be added to a CE library for use with point detectors.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
in=		31	logical unit of input MG library
out=		32	logical unit of output kinematic file in native format
worker			If present, the MG library is a working library
scratch=		14	scratch unit used during processing
mat=			ID of the nuclide to process

Logical Unit Parameters

Variable	Unit number	Type	Description
in		binary	logical unit of input MG library
scratch		binary	scratch
out		binary	logical unit of output kinematic file in native format

A-34. PALEALE - IMPROVED MODULE FOR PRINTING DATA FROM AMPX LIBRARIES

PALEALE is an extension of the ALE module that is provided to list data from AMPX master and working libraries. In addition to allowing a user more flexibility in selecting the information to be printed, some of the output formats have been improved, and a significant improvement is to allow a user to control the line lengths so that the printed information is easier to view on an 80-character terminal display. The input to PALEALE uses the ALE input as its basis (that is to say that a user can use exactly the same input and get improved outputs); however, additional parameters can be supplied in new arrays to give a user more control over how information is printed and to allow for reducing the volume of the output normally produced by an ALE run.

Please understand that all of the reports that can be printed by ALE have not been upgraded to give the user all the additional control. For example, the resonance information, whether it is resolved resonance parameters or Bondarenko factors still give the same output and are difficult to read on a terminal. The two areas that have been revised are the group-averaged data edits, and, especially, the transfer matrix edits. In the latter case, the edits have progressed from something that is virtually unreadable and hard to understand to something which is simple and well-labeled. Sample outputs will be given in a later Section.

PALEALE will be modified as time is available to allow more user control over the edits it produces.

Input Data

Block 1

0$ Logical unit assignments [2]

1.	MMT	logical unit of AMPX master library (1)
2.	MWT	logical unit of AMPX working library (0)

1$ Number of nuclides for which edits are wanted [1]

1.	NEDIT	number of nuclides for which edits are wanted

2$ Data classes to be printed [10]

1.	ICLASS1	group-averaged neutron data (0) 0: Do not print 1: Print
2.	ICLASS2	group-averaged gamma data (0) 0: Do not print 1: Print
3.	ICLASS3	resonance parameter data (resolved data or Bondarenko factors) (0) 0: Do not print 1: Print
4.	ICLASS4	not used
5.	ICLASS5	not used
6.	ICLASS6	not used
7.	ICLASS7	not used
8.	ICLASS8	not used
9.	ICLASS9	not used
10.	ICLASS10	not used

3$ Carriage control characters to be used in printing classes of data [25]

1.	JCLASS1	option whether to start the data for a nuclide on a new page (0)	
		0:	Do not start the data for a nuclide on a new page.
		1:	Start the data for a nuclide on a new page.
2.	JCLASS2	option whether to start the group-averaged data on a new page (0)	
		0:	Do not start the group-averaged neutron cross sections on a new page.
		1:	Start the group-averaged neutron cross sections on a new page.
3.	JCLASS3	printing of group-averaged gamma cross (0)	
		0:	Do not start the group-averaged gamma cross sections on a new page.
		1:	Start the group-averaged gamma cross sections on a new page.
4.	JCLASS4	printing of transfer matrices (0)	
		0:	Do not start transfer matrices for each process selected on a new page.
		1:	Start transfer matrices for each process selected on a new page.
5.	JCLASS5	not used	

4$ Process identifiers of transfer matrices to be printed [100]

1.	MTID	process identifiers of transfer matrices to be printed Input up to 100 process identifiers (MT-numbers) for the transfer matrices that should be printed. Note that a working library has only one transfer matrix, the "total" transfer matrix, which is selected by entering a 1.

5$ Maximum order of Legendre coefficient of transfer matrix to be printed [100]

1.	MAXOLC	maximum order of Legendre coefficient of transfer matrix to be printed Enter up to 100 values in one-to-one correspondence with the 4$ array

6$ Temperature for the scattering matrices to be printed [100]

1.	MAXTEMP	temperature for the scattering matrices to be printed Enter up to 100 temperatures in Kelvin for the scattering matrices to be printed. These must be entered in a one-to-one correspondence with the 4$ and 5$ arrays.

7$ Neutron process selection [200]

1.	MAXNPROC	neutron process selection Enter up to 200 process identifiers (MT-numbers) for the processes to be included in the print of the neutron group-averaged data. For example, if 1, 2, 4, 16, 18, and 27 are entered and filled with zeroes, the printout will include the total, elastic scattering, inelastic scattering, n2n, fission, and absorption cross sections, respectively.

8$ Gamma process selection [200]

1.	MAXGPROC	gamma process selection Enter up to 200 process identifiers for the processes to be included in the print of the gamma group-averaged data.

9$ Page format parameters [3]

1.	NLMAX	order of scattering to be printed (10)
2.	NTMAX	number of temperatures at which scattering matrices will be printed (10)
3.	LINE	line length that will be printed (80)

Note that this parameter only applies to group-averaged cross section and scattering matrix edits at present.

Terminate Block 1 with a T.

Only use Block 2 if NEDIT > 0.

Block 2

11$ Identifiers of the Nuclides [NEDIT]

1.	IDS	identifiers of the nuclides for which the user wants data printed Only used if NEDIT > 0. Enter NEDIT nuclide identifiers. Note that when NEDIT=0, the information selected in the first block will be printed for all nuclides in the library. To avoid having data for some nuclides included in the printout, the AJAX module should be used to select the nuclides desired.

12$ Zone of the Nuclides [NEDIT]

1.	IDZS	zone of the nuclides for which the user wants data printed (-1); only used if NEDIT > 0 Enter NEDIT nuclide zone identifiers. A -1 selects all zones

Terminate Block 2 with a T.

Notes

Note that one cannot produce edits from a master and a working file in the same execution.

Sample Input

```
    -1$$ 500000 0$$ 10 E 1$$ 1 4$ 2 F0
    5$ 3 F0 7$ 1 2 4 18 27 E T
    11$$ 1000 T
```

This input says to allocate 500,000 words of core to PALEALE and to read data from the AMPX master library on logical unit 10 for 1 nuclide, whose identifier is 1000. The scattering matrix for elastic scattering up to order P3 will be listed, along with the group-averaged data for MT=1 (total), MT=2 (elastic scattering), MT=4 (inelastic scattering), MT=18 (fission), and MT=27 (absorption).

Logical Unit Parameters

Variable	Unit number	Type	Description
MMT		binary	logical unit of AMPX master library
MWT		binary	logical unit of AMPX working library
	14	binary	scratch

A-35. EXTRACT - MODULE TO READ AN NJOY PENDF AND CREATE A TAB1 FILE

EXTRACT is a module that will read an ASCII PENDF that NJOY creates, and select the tabulated cross sections for a nuclide. These cross sections are written to a binary TAB1 File. File 1 and 3 reactions are copied.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
IN=		31	logical unit of PENDF library
OUT=		32	logical unit of the output TAB1 library
mode=		1	format of the output TAB1 file 1 - single precision tab1 format -1 - double precision tab1 format
T=			space-separated list of temperature(s) wanted on the output file. If not given, all temperatures are included.
MAT=			space-separated list of material identifiers wanted.
MT=			space-separated list of reaction identifiers. If not present, all are included.

Sample Input

```
    in=32 out=34 mat=9237 mode=-1
```

This example requests copying file 1 and 3 data for ^{238}U (MATNO=9237). The input PENDF is on logical unit 32, and the output TAB1 file is on logical unit 34 and is in double precision.

Logical Unit Parameters

Variable	Unit number	Type	Description
IN		BCD	logical unit of PENDF library
OUT		binary	logical unit of the output TAB1 library scratch
	14	binary	

A-36. PICKEZE - MODULE TO PICK FUNCTIONS FROM A TAB1 FILE

PICKEZE is a module that selects functions or classes of data on a library written as a TAB1 file and writes a new TAB1 file that contains the selected data. For example, if a file with the total cross section at 300 K is needed, PICKEZE can be used to extract the desired cross section data. There are other AMPX modules that will perform similar operations, such as ZEST, but none at the level of detail allowed by PICKEZE.

Input Data

Block Parameters

-1$ Core allocation [1]

1.	ICORE	not used (500000) START HEREA	

0$ Logical unit assignments [2]

1.	LOGIN	logical unit of the input TAB1 file (31)	
2.	LOGOUT	logical unit of the output TAB1 file (32)	

1$ Selection option control parameters [7]

1.	NMAT	number of materials to select (0)	
		Zero selects all materials.	
2.	NMF	number of file types to select (0)	
		Zero selects all file types.	
3.	NMT	number of processes to select (0)	
		Zero selects all processes.	
4.	NT	number of temperatures to select (0)	
		Zero selects all temperatures.	
5.	NSIG0	number of background cross section to select (0)	
		Zero selects all background cross sections.	
6.	temp_sel	exclusively selects temperature (0)	
		1:	select exclusively
		0:	also select non-broadened
			If exclusive selection is chosen, only processes with the desired temperature are selected. If non-broadened is selected, processes with only one temperature (0K) are also selected. Usually only a subset of processes is broadened; the remaining processes have only one temperature.
7.	sig_sel	exclusively selects background cross section values (0)	
		1:	select exclusively
		0:	also select sig0=0
			If exclusive selection is chosen, only processes with the desired background cross section are selected. If Also select sig=0 is chosen, processes with only one value of sig0 on the file are also selected.

Terminate block parameters with a T.

Block arrays

2$ Selected material identifiers [NMAT]

1.	MATS	material identifiers for the desired processes

3$ Selected file identifiers [NMF]

1.	MFS	file identifiers for the desired files

4$ Selected Process Identifiers [NMT]

1.	MTS	reaction identifiers for the desired reactions. If negative, then the specified reactions will be removed

5* Selected temperatures [NT]

1.	NTS	values for the desired temperatures

6* Selected background cross sections [NSIG0]

1.	NSIG0S	values for the desired sig0 values

Terminate block arrays with a T.

Sample Input

```
0$$ 23 24 1$$ 0 0 1 1 0 E T
4$$ 1
5** 300
T
```

The input TAB1 file is on logical unit 23, and the output TAB1 file is on logical unit 24. One process and 1 temperature are selected: MT=1 (total cross section) and 300K, respectively.

Logical Unit Parameters

Variable	Unit number	Type	Description
LOGIN		binary	logical unit of the input TAB1 file
LOGOUT		binary	logical unit of the output TAB1 file
	14	binary	scratch

A-37. PLATINUM - PKENO LIBRARY ASSEMBLER TOOL IN A USEABLE MODULE

PLATINUM (Pkeno Library Assembler Tool in a Useable Module) is a module that assembles a pointwise data library for CEKENO. PLATINUM reads 1-D pointwise data, kinematics data, and probability table data in order to assemble a cross section library for a single isotope/nuclide for CEKENO. It can also create a gamma cross section file for an element from 1-D pointwise data and kinematics data.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
gamma			processes gamma data
identifier=			id for data set to be created for SCALE: MAT+10000*MOD+1000000*Version
vers=		7	evaluation version number - used to construct identifier if identifier not specified
source=		0	id for data evaluation source (max 2 digit integer) -1 - unknown 0 - ENDFB 1 - JEF 2 - JENDL 3 - BROND 4 - CENDL other: user-defined source
title=			100-character title for data set (Title should be only entry on a line.)
out=		60	starting logical unit number for CE library (Code will increment the unit number for each temperature.)
maxtemp=		100	maximum number of temperatures possible on file
outtemp=		0	If not entered, all temperatures will be put out.
sigp=		0.0	potential scattering cross section
centrm=			corresponding thermal scattering kernel filename
debug			Prints extra debug output
eps=		0.0001	tolerance used for combining functions
icversion=		none	version of input creator used to create the input files
filever=		1.1	version of xsecs created by the latest input files
fileid=			output filename prefix. (Output filename will be fileid_temp where temp is the temperature.)
filedate=			date on which the input file is created by the input creator
ampxver=		6.0	version of AMPX being used
ampxdate=			date on which the AMPX module is created
scalever=		6.0	version of SCALE being used
scaledate=			date on which the SCALE package is created
union=		no	yes - turn on unionization no - co not turn on unionization

Keyword	Alternate	Default	Definition
fixnegatives=		no	yes - fix large negatives no - do not fix large negatives
outdetail=		normal	output detail level normal - print all useful information more - print more than just useful information
gyield=		no	yes - put gamma yield data onto the final library no - do not put gamma yield data onto the final library If filever is 2.0 or larger, gyield is set to yes.
debug=			prints extra debug information
gamma=			creates gamma library file

Repeat block cross section 1 time.

Block Cross Section

Block starts on first encounter of a keyword in the block. Block end is reached if all required parameters are given.

Keyword	Alternate	Default	Definition
n1d=			1-D CE cross sections (neutron or gamma)
id=		0	Material identifier for 1-D data

Repeat block info file 1 time.

Block Info File

Block starts on first encounter of a keyword in the block. Block end is reached if all required parameters are given.

Keyword	Alternate	Default	Definition
info=			logical unit for information file
id=		0	material identifier for info data

Repeat block fast kinematics data 1 time.

Block Fast kinematics data

Block starts on first encounter of a keyword in the block. Block end is reached if all required parameters are given

Keyword	Alternate	Default	Definition
n2d_fast=			logical unit for fast kinematics data (neutron or gamma)
id=		0	material identifier for 2-D fast kinematics data

Repeat block thermal kinematics data as often as needed.

Block Thermal kinematics data

Block starts on first encounter of a keyword in the block. Block end is reached if all required parameters are given

Keyword	Alternate	Default	Definition
Select one of these			
n2d_free		0	logical unit for free-gas kinematics data (neutron)
n2d_sab		0	logical unit for thermal scattering law kinematics (neutron) If thermal scattering law data are specified, n2d-free must be 0.
id=		0	material identifier for 2-D thermal kinematics data

Repeat block probability table data as often as needed.

Block Probability Table Data

Block starts on first encounter of a keyword in the block. Block end is reached if all required parameters are given.

Keyword	Alternate	Default	Definition
ptable=			logical unit for probability table data (neutron)
id=		0	material identifier for probability table data

Sample Input

```
identifier=xxxxxxxx source=xx vers= output=log
title=tttttttttttttttttttttttttttttttttttttttttttttttt n1d=log
id=xxxxxxxx
info=log id=xxxxxxxx n2d_fast id=xxxxxxxx n2d_free id=xxxxxxxx

n2d_sab id=xxxxxxxx sigp=sigp centrm=fname
ptable id=xxxxxxxx icversion=icv
debug end
```

Logical Unit Parameters

Variable	Unit number	Type	Description
out		binary	starting logical unit number for CE library (Code will increment the unit number for each temperature.)
n1d		binary	1-D CE cross sections (neutron or gamma)
info		binary	logical unit for information file
n2d_fast		binary	logical unit for fast kinematics data (neutron or gamma)
n2d_free		binary	
n2d_sab		binary	

Variable	Unit number	Type	Description
ptable		binary	logical unit for probability table data (neutron)
	14	binary	scratch
	15	binary	scratch
	16	binary	scratch
	17	binary	scratch

A-38. POLIDENT - MODULE TO PRODUCE POINT DATA FROM RESONANCE DATA

POLIDENT (Point Libraries of Data from ENDF/B Tapes) is a module that accesses the resonance parameters from File 2 of an ENDF/B library and constructs the CE cross sections in the resonance region. The cross sections in the resonance range are subsequently combined with the File 3 background data to construct the complete cross section representation as a function of energy. POLIDENT has the following notable features:

- ☐ processes all resonance reactions that are identified in File 2 of the ENDF/B library
- ☐ processes single- and multi-level Breit - Wigner, Reich-Moore and Adler-Adler resonance formalisms
- ☐ provides a robust energy mesh generation scheme that determines the minimum, maximum and points of inflection in the cross section function
- ☐ processes all CE cross section reactions identified in File 3 of the ENDF/B library and outputs all reactions in an ENDF/B TAB1 format that can be accessed by other AMPX modules
- ☐ processes multi-isotope nuclides with different resonance ranges
- ☐ treats discontinuities in cross section data by taking the limit of the function from both sides of the discontinuity
- ☐ provides ENDF/B File 1 and abbreviated File 2 data that can be used to construct a PENDF (Point ENDF) file by the AMPX module MAKPEN

Input Data

Block 1

-1$	File9Processing [1]		
	1.	File9	if not 0, unit in which to save file 9 and file 10 data (0)
0$	Output library [3]		
	1.	LOGP	logical unit for point-wise cross section data (31)
	2.	LOGP1	logical unit for File 1 and abbreviated File 2 information (32)
	3.	LOGRES	restart unit (0)
1$	Number of cases [1]		
	1.	NNUC	number of cases (1)

Terminate Block 1 with a T.

Repeat Block 2 NNUC times.

Only use Block 2 if NNUC > 0.

Block 2

2$ ENDF/B Data Source [4]

1.	MAT	ENDF material identifier for nuclide to be processed
2.	NDFB	logical unit number for ENDF library (11)
3.	MODE	ENDF library format (2) 1: binary 2: BCD
4.	NVERS	not used (0)
5.	mesheps	convergence tolerance for energy mesh generation (0.001)

4* Floating Point Parameters [14]

1.	EPS	epsilon to combine data from Files 3 and 2 (0.001)
2.	R	the ratio factor used in a cross section energy mesh (0.99) value that is used only for nuclides using the Adler-Adler parameterization in the resolved resonance range
3.	XNP	the number of points taken equally spaced in lethargy between resonance bodies (50.0) value used only for nuclides using the Adler-Adler parameterization in the resolved resonance range
4.	XGT	the multiplier on the total width above and below a resonance over which the ratio mesh scheme is used (50.0) value used only for nuclides using the Adler-Adler parameterization in the resolved resonance range
6.	OPT2	not used (0)
7.	OPT3	not used (0)
8.	OPT4	not used (0)
9.	OPT5	not used (0)
10.	OPT6	not used (0)
11.	OPT7	not used (0)
12.	OPT8	not used (0)
13.	OPT9	not used (0)
14.	OPT10	not used (0)

5$ Options [8]

1.	intstart	starting value for interpolations to be tried (1) the values for interpolation to be tried start at inter1 and go through inter6, listing the default endf interpolation values. Usually lin-lin is the only one desired for point-wise cross section data.
2.	intstop	ending value for interpolations to be tried (1) the values for interpolation to be tried start at inter1 and go through inter6, listing the default endf interpolation values. Usually lin-lin is the only one desired for point-wise cross section data.
3.	IOPT3	maximum number of interpolation regions allowed in the output (1)
4.	inter1	Interpolation type to be tried (2) Linear-Linear is 2. Other allowed values are 1-5.
5.	inter2	Interpolation type to be tried (0) Linear-Linear is 2. Other allowed values are 1-5.
6.	inter3	Interpolation type to be tried (0) Linear-Linear is 2. Other allowed values are 1-5.

7.	inter4	interpolation type to be tried (0) Linear-Linear is 2. Other allowed values are 1-5
8.	inter5	interpolation type to be tried (0) Linear-Linear is 2. Other allowed values are 1-5

6$ Function parameters [4]

1.	AddMt51	If not zero, add MT=51 from URR range if applicable. (0)
2.	N2MAX	not used (0)
3.	MLBW	not used (0)
4.	IPOINTS	maximum number of points per 10eV interval (5000)

Terminate Block 2 with a T.

Notes

Parameters R, XNP, and XGT in Block 2, Array 4 are only used for generating an energy mesh for nuclides with the Adler-Adler formalism.

intstart, instop, and inter1 through inter5 are used to specify the interpolation types and their order which will be in combining two or more ENDF/B functions. The types are as follows:

1. Histogram
2. Linear x, linear y
3. Linear x, log y
4. Log x, linear y
5. Log x, log y

instart and intstop specify which entries in the five-position table are to be used (e.g., the default values of 1 indicate that only the first entry in the table should be used, or linear-linear interpolation by default).

Sample Input

```
0$$ 31 32 1$$ 5 T
2$$ 9228 11 2 T
2$$ 9231 11 2 T
2$$ 2637 11 2 T
2$$ 125 11 2 T
2$$ 825 11 2 T
```

This input tells POLIDENT to access an ENDF/B file on logical unit 11 that is in BCD format and that contains the data for nuclides ^{235}U, ^{238}U, Fe, ^{1}H, and ^{16}O, identified by 9228, 9237, 2631, 125, and 825, respectively. The data will be written to logical unit 31.

Logical Unit Parameters

Variable	Unit number	Type	Description
NDFB		BCD	logical unit number for ENDF library
LOGRES		binary	restart unit
LOGP		binary	logical unit for point-wise cross section data
LOGRES		binary	restart unit
	14	binary	scratch
	15	binary	scratch
	18	binary	scratch

A-39. PRELL - MODULE TO PRODUCE AND MANIPULATE AN ENERGY LIMITS LIBRARY

PRELL (Produce Reordered Energy Limits Library) is an AMPX module to create copy, modify, punch, or list an AMPX energy-group-limits library. The new library will be reordered in an increasing number of groups for neutron structures, then in the order of increasing groups for gamma structures. The new library may be printed. Modification features include adding new structures to an existing library and changing boundaries in an existing structure.

Input Data

Block 1

0$ Logical unit assignments [3]

	1.	NO	logical unit number of the old library (77) If a new group limits library is being created, a 0 is entered for this parameter.
	2.	NW	logical unit number of the new library (18)
	3.	NS	not used (0)

1$ Options [2]

	1.	NOPT	Print option (0) 0: prints only new or updated group structures 1: prints all group structures
	2.	NSETS	number of sets to be added/deleted and/or modified (0)

Terminate Block 1 with a T.

Stack Block 2 and 3 one after the other NSETS times

Block 2

3$ FLAGS [3]

	1.	IG	number of groups in set If negative, the group structure is deleted from the file
	2.	ITYPE	type of group structure (0) 0: neutron-group-structure 1: gamma-group-structure
	3.	IVER	version of group structure; only 0 is currently allowed (0)

Terminate Block 2 with a T.

Only use Block 3 if IG > 0.

Block 3

7* Group boundaries [IG+1]

	1.	IGB	group boundaries in eV

Terminate Block 3 with a T.

Notes

The structure of the group limits file is very simple. It consists of one header record that indicates how many group structures are included and what they are. This is followed by one record for each group structure indicating the energy boundaries (in eV). The structure of record 1 is:

```
Record 1: NS, (INDEX(J,I), J=1,3), I=1,NS)
```

The NS records that follow this use:

```
Records 2, NS+1: IGM, (EBDRY(I), I=1,IGM+1)
```

The usage of the INDEX array is as follows:

INDEX(1,I) number of energy groups of the Ith structure,
INDEX(2,I) particle type (0 for neutrons, 1 for photons) of the Ith structure, and
INDEX(3,I) version of the Ith structure (this term has never been activated.

IGM is a repeat of the number of energy groups, and the EBDRY array contains the group limits in eV arranged in descending order.

Sample Input

```
0$$ 47 48 0 1$$ 0 2 T
3$$ 537 0 0 T
7** (Put in 538 energy boundaries for a 537 group neutron structure.)
T
3$$ 10 0 0 T
7** (Put in 11 energy boundaries for a 10 group neutron structure.)
T
```

This input shows how the user would update the standard energy group library on logical unit 47, to include new 538 and 10 energy group neutron structures. The new group library will be written on logical unit 48 and will contain all of the older structures, in addition to the two new structures.

Logical Unit Parameters

Variable	Unit number	Type	Description
NO		binary	logical unit number of the old library
NW		binary	logical unit number of the new library

A-40. PRILOSEC - MODULE TO PRODUCE ORIGEN CROSS SECTION LIBRARIES

PRILOSEC (Produce Incredible Libraries of Cross Sections) is a module that reads a file with TAB1 records and creates an AMPX master library for each material that it finds on the library. The ZA number will be used for an identifier unless the noorig keyword is supplied. Only one temperature for each nuclide is allowed. If the file contains more than one temperature, module PICKEZE should be used to select the desired temperature value.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
master=		1	logical unit for final library
tab1=		32	logical unit of point data
logwt=		32	logical unit for weighting function file
logebdry=		47	logical unit for file containing energy boundaries The file containing the standard AMPX energy boundaries is by default linked to unit 47. If a non-standard group structure is preferred, the prell module should be used.
title=			title for the nuclides nuclide id added automatically
matwt=			material number of the weighting function
mtwt=			reaction number of the weighting function
eps=		1e-5	precision to which to calculate the integral
igm=			number of neutron groups
noorig			If set, mt values and ids will not be reset for ORIGEN.
nowork			If set, an AMPX master file will be produced.
old			If set, the old format for ORIGEN libraries will be used.

A-41. PRUDE - AMPX MODULE TO CREATE CROSS SECTIONS FOR THE UNRESOLVED RESONANCE ENERGY REGION

PRUDE (Process Unresolved Data on ENDF/B) is a module that accesses the unresolved resonance data in file 2 of an ENDF/B library and writes out a file which gives the energy variation of average cross sections for several important processes as a function of temperature and the weighting parameters, sigma_0. Its primary use is to pass these data to the TABU module, which creates Bondarenko factors that ultimately become part of an AMPX master interface. The Bondarenko factors are used by the BONAMI module for performing self-shielding in the unresolved region.

In the development of the Bondarenko treatment, a narrow-resonance weighting of the form

If f(E) = PHI(E) / (sigma_T + sigma_0) sigma_{gr}(sigma_0,T) = (int_g[sigma(E,T) f(E) dE]) / (int_g[f(E) dE])

is used, where PHI(E) is a smooth weighting function (generally 1/E in the unresolved region), T is the temperature at which the cross sections were Doppler broadened, and sigma_0 is the cross section that accounts for the cross sections of other nuclides in the mix with the resonance nuclide.

PRUDE accepts an arbitrary number of temperatures and sigma_0 values as input. At each pair of values, T and sigma_0, it makes a calculation to determine shielded cross sections. The energy mesh is chosen to be either the energy mesh at which the unresolved parameters are specified in the ENDF/B library or at 100 points equally spaced in lethargy over the unresolved region when the parameters are constant. The output from PRUDE is a file of records written in the ENDF/B "TABL" format, as follows:

```
Record 1: MAT, MF, MT, 0, 0, 0, 0, 0, 0
Record 2: MAT, MF, MT, T, sigma0 , 0, 0, NR, NP, (NBTi, JNTi, i=1, NR),
( E(i), sigma( T, sigma0 ), i=l, NP)
Record 3: MAT, MF, 0, 0, 0, 0, 0, 0, 0
```

where MAT is the material identifier, MF is the file number, MT is the process identifier, T is the temperature in Kelvin, sigma_0 is the background cross section, NR is the number of interpolation regions, NBTi, JNTi comprise the interpolation table, and E, sigma are the energy cross section values.

Each T, sigma_0 pair will generate the three records shown above for each of six processes.

MT = 1, total cross section
MT = 2, elastic scattering
MT = 102, (n,gamma)
MT = 18, fission
MT = 1000, transport cross section
MT = 4, inelastic scattering

Input DataData

Block Units

0$ Point file assignment [1]

1.	LOGP	logical unit in which point data are to be written (31)

1$ Processing Option [1]

1.	NNUC	Number of materials to process

Terminate block units with a T.

Stack block parameters and values one after the other NNUC times.

Block Parameters

2$ Data source and problem options [5]

1.	MATNO	material number for the ENDF/B data
2.	NSIG0	number of background values
3.	NTEMP	number of temperatures
4.	NDFB	logical unit containing the ENDF/B data (11)
5.	MODE	format of ENDF/B data (1) 1: binary 2: BCD

Terminate block parameters with a T.

Block Values

3* Background cross sections [NSIG0]

1.	NSIG0S	Background cross sections (sigmma_0,i=1,NISG0) Specify these values in descending order

4* Temperatures [NTEMP]

1.	TEMPS	TEMPERATURE (T_i,i=1,NTEMP); Specify these values in ascending order

5* Processing options [2]

1.	EPS	precision with which to combine data (1.0e-3)
2.	NEW	not used (0)

Terminate block values with a T.

Sample Input

```
Sample Input
0$$ 31 1$$ 2 T
2$$ 9228 8 5 11 2 T
3** 1.0E10 1.0E6 1.0E4 1000 100 10 1 1.0E-5
4** 300 600 1000 1500 2000 T
2$$ 9237 8 5 12 2 T
3** 1.0E10 1.0E6 1.0E4 1000 100 10 1 1.0E-5
4** 300 600 1000 1500 2000 T
```

This input illustrates how to use PRUDE to create a point library of data for the unresolved energy regions of ^{235}U (MAT=9228) and ^{238}U (MAT=9237). The ^{235}U data are accessed from the BCD ENDF/ B library on logical unit 11, while the ^{238}U data are from the BCD ENDF/B library on logical unit 12. In both cases, numbers for eight background cross sections and five temperatures will be produced. (Note that PRUDE is programmed to discard any background-temperature combinations producing negative cross section values that arise due to approximations used in the scheme that calculate self-shielded values).

Logical Unit Parameters

Variable	Unit number	Type	Description
LOGP		binary	logical unit where point data are to be written
NDFB		BCD	
	14	binary	scratch
	15	binary	scratch
	16	binary	scratch
	17	binary	scratch
	18	binary	scratch

A-42. PUFF_IV - MODULE TO GENERATE MG CORRELATION MATRICES

PUFF-IV is a module that reads the cross section uncertainty data from an ENDF/B library and constructs MG correlation matrices on a user specified energy grid structure. PUFF-IV has the following features:

☐ Processes ENDF/B uncertainty data through Version VI

☐ Provides output correlation matrices in the COVERX format

☐ Processes short-range variance formats, as well as lumped reaction covariance formats that were introduced in ENDF/B-V and could not be processed by PUFF-III

☐ Has a directory feature that provides a list of the explicitly and implicitly defined covariance matrices from ENDF/B Files 31 and 33; also, determines if resonance parameter uncertainty information from ENDF/B File 32 is available

☐ Calculates eigenvalues for each correlation matrix and tests for positive definiteness

Input Data

Block 1

-1$	Core allocation [1]		
	1.	LENGTH	number of words to allocate (500,000)
0$	Directory flag [1]		
	1.	LDIR	directory option (0) If a logical unit (ldir > 0) is given, the program generates the directory output and exits. For ldir=0 (the default), covariance matrices are generated.
1$	Integer parameters [18]		
	1.	NO28	unit for COVERX formatted output (-/+ = Binary/BCD) (-1)
	2.	ISS	unit number for standard deviations in user group structure from standard cross section file (0) only used if processing LTY=1 NC sub-subsection, and the standard cross section uncertainties must be processed apriori. A suitable file is normally generated on unit 16 if processing a standard uncertainty file.
	3.	I19	unit number for standard material ENDF uncertainty file in BCD format (0) only used if processing LTY=1 or 2 NC sub-subsections. I19 cannot equal IO32
	4.	IO11	unit number for AMPX master library if IXSOP=1 or TAB1 file if IXSOP=2(0) only used if IXSOP > 0
	5.	IO32	unit number for ENDF uncertainty file in BCD format (32)
	6.	MATUSE	MAT number of material to process (0)
	7.	IUSER	number of user groups for covariance calculation (-12) -2: 240 group CSEWG -3: 99 group GAM2 -4: 620 group SAND2

		-5:	30 group LASL
		-6:	68 group GAMI
		-7:	171 group VITAMIN-C
		-8:	26 group ORNL-5517
		-9:	100 group GE
		-10:	6 group ORNL-5318
		-12:	44 group AMPX
		other:	Otherwise user input in 2# array if greater than 0. If negative and not one of the above choices, use the number of groups in a standard AMPX group.
8.	IXS	number of groups of input cross sections (-11)	
		-2:	240 group CSEWG
		-3:	99 group GAM2
		-4:	620 group SAND2
		-5:	30 group LASL
		-6:	68 group GAMI
		-7:	171 group VITAMIN-C
		-8:	26 group ORNL-5517
		-9:	100 group GE
		-10:	6 group ORNL-5318
		-11:	Read from AMPX master library
		-12:	44 group AMPX
		other:	Otherwise user input in 3# array if greater than 0. Set only to give cross section data explicitly in the 4## array.
9.	IWT	weighting function	
		1:	1/E
		2:	1/(E * sum_{T}
		3:	(1/E)*INPUT (INPUT placed in 5# array)
		4:	INPUT (placed in 5# array)
		other:	If less than 0 the unit of a Flux file in tab1 format If a flux file is given, the material and reaction value of the flux is read in the 5## array
10.	IXSOP	cross section input (1)	
		0:	User input in 4# array
		1:	AMPX master library
		2:	TAB1 file containing point-wise cross section data
11.	JOPT1	Files 31 and 33 processing options (2)	
		0:	processes File 33
		1:	processes File 31
		2:	processes Files 31 and 33
		3:	processes neither File 31 nor File 33
12.	JOPT2	File 32 processing options (2)	
		0:	does not process File 32
		1:	processes File 32 as sensitivity data
		2:	full resonance calculation of File 32
13.	JOPT3	option for matrix to be collapsed to user group (0)	
		0:	Yes
		1:	No
			For normal operation, the covariance matrix should be collapsed to the user group structure. The collapsing is not wanted if a File

32 covariance matrix should be processed in preparation for converting to File 33 format.

14.	NOX	maximum number of covariance matrices in COVERX file (500)
15.	NOCVX	not used
16.	NMT	number of MAT-MT reaction pairs to process (-1) If larger or equal to 0 the number of covariances to process. They are given in Block 4. If -1 process all reaction pairs on ENDF tape
17.	NDM1	Reads integral constants (0) 0: uses standard integral constants 1: reads integral constants from Block 6
18.	LD8FL	How an LB=8 section gets calculated (0) 0: calculated as described in ENDF standard 1: assumes that ratio (Delta E_{k})/(Delta E_{I}) = 1 for all k and I 2: ignores all contributions from LB=8 sections

Terminate Block 1 with a T.

Only use Block 2 if IUSER > 0.

Block 2

2# usergrid [IUSER+1]

1.	UserGrid	USER energy grid Only used if IUSER > 0.

Terminate Block 2 with a T.

Only use Block 3 if IXS > 0.

Block 3

3# IXS_ARRAY [IXS+1]

1.	CrossGrid	cross section energy grid Only used if IXS > 0.

Terminate Block 3 with a T.

Only use Block 4 if NMT > 0.

Block 4

4# MAT-MT pairs and cross section data [NMT*2 + NMT*IXS]

1.	MATInfo	material and reaction value of covariances to be calculated. Only used if NMT > 0
2.	CrossSections	cross section data for MAT-MT pairs Only used if IXSOP = 0.

Terminate Block 4 with a T.

Only use Block 5 if ABS(IWT) > 2.

Block 5

5# IWT_ARRAY [IXS]

1.	Weights	user-defined weighting factors Only used if ABS(IWT) > 2. If IWT is negative, the material id and the reaction id of the weighting function to use should be given. If using an AMPX cross section library the number of weights given has to be the same as the number of groups on the cross section library. If using point-wise cross section data, the number of weights must be the same as the number of groups in the super group structure. It is recommended to use IWT<0 in this case.

Terminate Block 5 with a T.

Only use Block 6 if NDM1 = 1.

Block 6

6# THERMAL_VALUES [3]

1.	ThermalEner	energy for thermal cross section in eV (0.0253) Only used if NDM1 = 1.
2.	LowRes	lower energy for resonance integral in eV (0.5) Only used if NDM1 = 1.
3.	UppRes	upper energy for resonance integral in eV (5500) Only used if NDM1 = 1.

Terminate Block 6 with a T.

Block Title Cards

COVERX_TITLE: COVERX Title card Type: Character*72

Notes

If JOPT1=3 and File 31 or 33 are not present in the file, only the available information is processed, and PUFF-IV prints a warning message about the missing file. Similarly, if JOPT2=2 or JOPT2=3 is specified and File 32 is not present, the calculation proceeds as if JOPT2=0 was specified. A warning message is printed.

Sample Input

```
    -1$$ 400000000 e
    1$$ -1 0 0 11 32 9222 -12 -11 1 1 2 2 a16 -1 e t coverx file for u233
```

This input prompts puff-iv to process File 31, 32 and 33 from ENDF/B file on logical unit 32. The cross section data are taken from the AMPX library on logical unit 11. The covariances format id=9222 are generated for the 44 AMPX group structure with a weighting of 1/E. The title for the COVERX file is "coverx file for u233"

Logical Unit Parameters

Variable	Unit number	Type	Description
NO28		BCD or binary	unit for COVERX formatted output (-/+ = Binary/BCD)
ISS		binary	unit number for standard deviations in user group structure from standard cross section file
I19		binary	unit number for standard material ENDF uncertainty file in BCD format
IO32		bcd	unit number for ENDF uncertainty file in BCD format
	15	random access	scratch
	16	binary	scratch
	17	binary	scratch
	18	binary	scratch
	19	binary	scratch
	20	random access	scratch
	21	binary	scratch
	22	binary	scratch
	23	binary	scratch
	25	binary	scratch

A-43. PURM - MODULE TO PRODUCE PROBABILITY TABLES FROM UNRESOLVED RESONANCE DATA USING MONTE CARLO

Purm (probability tables for the unresolved region using Monte Carlo) is a module that uses a Monte Carlo approach to calculate probability tables on an evaluator-defined energy grid in the unresolved-resonance region (urr). For each probability table, purm samples pairs of resonances surrounding the reference energy. The resonance distribution is sampled for each spin sequence (i.e., l-j pair), and purm uses the Delta$_3$-statistics test to determine the number of pairs of resonances for each spin sequence. For each resonance, purm samples the resonance widths from a chi-square distribution for a specified number of degrees of freedom. Once the resonance parameters are sampled, purm calculates the total, capture, fission and scatter cross sections at the reference energy using the single-level Breit-Wigner formalism with appropriate treatment for temperature effects. The cross section calculation constitutes a single iteration or history. The calculation is repeated for a user-specified number of histories and batches. After completing the specified number of histories for a batch, a batch estimate for the probability for each cross section band within a table is obtained by dividing the number of tallies for the band by the total number of histories processed. Additional batches are processed until the user-specified number of batches is complete. Due to the nature of the calculational procedures, purm provides a mechanism for monitoring the convergence of the cross section calculation. For each reaction, a plot of the calculated cross section is provided by the batches run. Additional statistical checks are provided for each cross section calculation.

Note that purm should only be used to process individual isotope evaluations, and it should not be used to process nuclide evaluations with multiple isotopes with unresolved-resonance regions.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
logp=		31	output unit for the probability table
bond=		32	output unit for the Bondarenko factors

Repeat block nuclide as often as needed.

Block nuclide

Block starts on encountering nuc.

Block terminates on encountering enuc.

Keyword	Alternate	Default	Definition
nbatch=		300	number of batches to run
iter=		600	number of iterations per batch
nband=		20	number of bands to create
mat=			ENDF material number
ndfb=			logical unit of ENDF file to process
temp=			space-separated list of temperature(s) in Kelvin at which probability tables are desired
sig0=			space-separated list of background values for the Bondarenko factors
equ			If present, bands are equiprobable.
eps=		0.001	precision to which to create the mesh if adding cross section data
extra=		0	number of points to add between energies given in the ENDF file

Logical Unit Parameters

Variable	Unit number	Type	Description
ndfb logp		binary binary	output unit for the probability
logb		binary	table

A-44. PURM_UP - CORRECT PROBABILITY TABLES FOR FILE 3 CONTRIBUTIONS.

The module purm generates probability tables at the energies of references given in the ENDF/B formatted file. It does not take File 3 (smooth) cross section contribution into account. This module adds or multiplies with the File 3 cross section data, depending on the flag set in the ENDF/B formatted file.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
in=		31	input unit for tables generated by PURM
out=		32	output unit for updated probability tables
ndfb=		11	logical unit of ENDF file to process
matf=			material number of tables generated by purm and in endf
matp=			material number desired on output file
eps=		0.001	precision to which to create the mesh if adding cross section data

Logical Unit Parameters

Variable	Unit number	Type	Description
ndfb		binary	logical unit of ENDF file to process
in		binary	input unit for tables generated by purme
out		binary	output unit for updated probability tables

A-45. RADE - MODULE TO CHECK AMPX MASTER CROSS SECTION LIBRARIES

RADE (Rancid AMPX Data Exposer) is provided to check AMPX- and ANISN-formatted MG libraries. It will check neutron, gamma, or coupled neutron-gamma libraries. Some of the more important checks are made to ensure that:

☐ sigma_{t} = sigma_{a} + sigma_{s}

☐ sigma_{in} = sum(sigma_{in}^{partial})

☐ sigma_{a} = sigma_{c} + sigma_{f}

☐ sigma_{c} = sigma_{n,gamma} + sigma_{n,alpha} +sigma_{n,p} + sigma_{n,d} + ...

☐ sigma_{el}^{g} = sum(sigma_{el}(g -> g')

☐ sigma_{0}(g -> g') > 0

☐ sigma_{t}, sigma_{a}, sigma_{f}, sigma_{n,gamma}, sigma_{n,p},... > 0

☐ f_{l}^{min} < f_{l}(g -> g') <= 1.0
where f_{l}(g -> g' = [sigma_{l}(g -> g')]/[(2l+1) sigma_{0}(g -> g')]
and f_{l}^{min} = -1.0 for all odd l and for even l

☐ l=2 - f_{l}^{min} = -0.5

☐ l=4 - f_{l}^{min} = -0.433

☐ l=6 - f_{l}^{min} = -0.419

☐ l=8 - f_{l}^{min} = -0.414

In addition to these checks, the code will compute an estimate of the capture-binding energy for each neutron group in a coupled neutron-gamma set. On option, one can request a display of differential cross sections.

Input Data

Block 1

-1$ Core assignment [1]

1.	NWORD	number of words to allocate (100,000)

1$ Checking commands [4]

1.	MMT	checks the AMPX master interface on logical MMT (0) (can be a neutron, gamma, or a coupled neutron-gamma library)
2.	MWT	checks the AMPX working/weighted interface on logical MWT (0)
3.	MAN	checks the ANISN binary-formatted library on logical MAN (0)
4.	IFM	formats of the ANISN library -1: ANISN library is binary formatted. 0: ANISN library is BCD free form. 1: ANISN library is BCD fixed form

2$ Options [20]

1.	numAng	number of angles at which a display of differential cross sections is desired (0) These angles will be equally spaced in the cosine range, -1 to +1. These edits are for the group-integrated cross sections and not for each group-to-group transfer
2.	eps	the epsilon in 1/1000s of a percent, to which checks are made (1) That is eps=1 is equivalent to 0.001% checking. This is the default value when eps is not entered or when a zero value is entered.
3.	printbind	print option 0: prints the estimated binding energy table 1: suppresses printing the estimated binding energy tables for processes with gamma production data
3.	OPT3	not used

3$ ANISN Options [7]

1.	NSET	number of ANISN nuclides to check
2.	IHT	position of sigma_{T} if checking ANISN library
3.	IHS	position of sigma_{g} if checking ANISN library
4.	ITL	table length if checking ANISN library
5.	NL	maximum order of scattering if checking ANISN library
6.	IGM	number of neutron groups if checking ANISN library
7.	IPM	number of photon groups if checking ANISN library

Terminate Block 1 with a T.

Only use Block 2 if MAN != 0.

Block 2

4$ Identification numbers of P0 sets [NSET]

1.	IDPO	identification numbers of P0 sets on ANISN binary library on logical MAN Only used if MAN != 0.

5$ Order of scattering for sets [NSET]

1.	ISOR	order of scattering for sets of ANISN data on logical MAN Only used if MAN != 0.

7* Neutron group structure [IGM+1]

1.	NGS	neutron group structure Only used if MAN != 0. order high to low in eV

8* Neutron group structure [IPM+1]

1.	GGS	gamma group structure Only used if MAN != 0.

order high to low in eV Terminate Block 2 with a T.

Sample Input

```
    1$$ 1 E T
```

This input instructs RADE to perform consistency checks on the data on the master library on logical unit 1.

Logical Unit Parameters

Variable	Unit number	Type	Description
MMT		master library	checks the AMPX master interface on logical MMT
MWT		working library	checks the AMPX working/ weighted interface on logical MWT
MAT		ANISN library	
	18	binary	scratch
	19	binary	scratch

A-46. SIMONIZE

SIMONIZE is a module that collects classes of data (resonance parameters, neutron data, gamma data, gamma production data, thermal scattering matrices, etc.) from an arbitrary number of AMPX master formatted data sources and combines them into a single comprehensive collection (i.e., a master library that contains all the data wanted) for a nuclide. At the same time, SIMONIZE normalizes and rearranges data to make a set of data that are ready for use in transport calculations or other applications.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
IDENTIFIER=			identifier of the collection of data on the master library
MASTER=		1	logical unit onto which the data will be written
TITLE=			title to use for the data SIMONIZE will use the title from the NEUTRON data files if a TITLE is not supplied, which will be the most common situation.
za=			overrides ZA value of the nuclide
fastid=			overrides id identification for fast data
thermid=			overrides id identification of thermal data
gamid=			overrides id identification of photon data
yieldid=			overrides id identification of photon yield data
source=		0	source of the data as defined in the ENDF manual
small1d=		1.0e-12	1-D cross sections smaller than this are set to zero
small2d=		1.0e-12	2-D cross sections smaller than this are set to zero
kipratio			If not present, apply correction to MT=1007 if not a moderator, to mt=2 otherwise.
skipnorm			does not recalculate redundant cross sections
skipscatter			does not correct matrices for upsscatter
oldza			If present, convert the za value to the za values used for SCALE 6.1 and earlier.

Repeat block data descriptions as often as needed.

Block Data descriptions

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
Select one of these			
NEUTRON			unit containing a collection of neutron data
GAMMA			unit containing a collection of gamma-ray data
YIELD			unit containing a collection of gamma-ray yield data
BONDARENKO			unit containing Bondarenko factor data
1DN			unit containing averaged neutron data
2DN			unit containing neutron scattering matrices
1DG			unit containing averaged photon data
2DG			unit containing photon scattering matrices
2DY			unit containing photon production matrices
MODERATOR			special keyword used to signal that the data originate from a thermal ENDF/B evaluation
ID19=			the identifier of the data on the logical unit that currently processed
mt=			list of reaction values to include or exclude If all mt values are positive, the listed mt values will be selected from the partial library and added to the new library. If all mt values are negative, the listed mt values are excluded from the new library.

Sample Input

```
Identifier=92234 master=1 source=0
Neutron=20 id19=92234
Bondarenko=36 id19=9221
2dn=21 id19=92234
```

This input tells SIMONIZE to combine NEUTRON data produced by X10 that is located on logical unit 20 and identified by 92234 with thermal scattering data located on logical unit 21 identified by 92234, and with Bondarenko Factor Data on logical unit 36 identified by 9221 to create an AMPX master on logical unit 1 with a nuclide identifier of 92234.

Logical Unit Parameters

Variable	Unit number Type	Description
MASTER	binary	logical unit onto which the data will be written
NEUTRON	binary	
GAMMA	binary	
YIELD	binary	
BONDARENKO	binary	
1DN	binary	
2DN	binary	
1DG	binary	
2DG	binary	
2DY	binary	

A-47. SMILER - AMPX MODULE TO CONVERT NJOY GENDF FILES TO AMPX MASTER LIBRARIES

The SMILER module (Second MILER*) was written to circumvent inefficiencies observed in the use of the original code.

MILER1 provides a means of converting group-averaged cross sections from the NJOY2 system for use by modules written for the AMPX system. By default, NJOY writes these data in a format called the GENDF format, which is an ENDF/B-like format.

SMILER is not a revision of MILER, but it is a response to the observation that many situations will cause MILER to require exorbitant I/O operations to convert between GENDF format and the AMPX master library format. SMILER uses procedures that take advantage of the current large-computer memories, allowing the user to liberally use core-size allocations. This is in contrast to previous processes in which the user shuttled data in and out of the core to accommodate many problems. Because of this change in programming style, SMILER uses simpler procedures than previously employed, thereby making it more compact and easier to maintain.

As with MILER, SMILER requires little input over simply specifying the GENDF files to be combined and converted. Like MILER, a SMILER run produces cross sections for only one nuclide. These one-nuclide master libraries can be easily collected by the AJAX module. SMILER accepts the BCD or binary formats of GENDF files.

Note that no code which prepares an AMPX master library should include an array identified by 1452 in the 1-D arrays. SMILER does not include it and should never be modified to do this, as it will result in completely erroneous results when used in some code combinations.

Input Data

Block 1

0$ Logical Assignments [3]

1.	MMT	logical unit of AMPX master interface (1)
2.	MG1	first GENDF file (0)
3.	MG2	second GENDF file(0)
4.	MG3	third GENDF file(0)

Note that because photon-only GENDF files do not strictly follow the GENDF format specifications and specify the number of photon groups in the word designation for the number of neutron groups, MG3 is reserved as the location for this type of file. Logical units MG1 and MG2 can both contain either neutron-only or coupled neutron-gamma data. Borrowing an idea from MILER, positive values for MG1, MG2, MG3 are used for BCD files, whereas negative values are used for binary files.

1$ Nuclide identifier and direct-access file status [2]

1.	ID19	identifier of the set of data produced by SMILER (1)
2.	N9STAT	not used (0)

2$ NJOY/AMPX thermal identifier correspondence list [100]

1.	NJID	NJOY/AMPX thermal identifier correspondence list (221 1007 222 1008 e) Up to 50 doublets give the NJOY identifiers for a thermal-scattering process, followed by its corresponding AMPX identifier. By default, this array contains 221 1007 222 1008, followed by 96 zeroes, which indicates that an AMPX identifier of 1007 should be used on the arrays which NJOY identifies with 221 and 1008 on those identified by 222.

Terminate Block 1 with a T.

Notes

Converting between different MG cross section formats is a very common requirement, but there is a wide variety of choices that one can make in designing a format.

The differences in GENDF and AMPX formats clearly demonstrate areas that can be different.

1. The ordering of energy groups differs. Traditionally, group 1 is the highest energy group, as it is in the AMPX master interface. In GENDF, group 1 is the lowest energy group.

2. The Legendre coefficients in scattering matrices in the AMPX master interface include the (2l + 1) multiplier following conventions established for the ANISN and DOT programs in the mid-1960s. GENDF does not.

3. The matrices for reactions that produce multiple secondary particles, such as n2n, contain the multiplicity on GENDF. In AMPX, they do not.

4. The units of temperatures associated with scattering matrices are in eV in AMPX vs. Kelvin in GENDF.

5. Various process identifiers for averaged cross sections must be carefully monitored in order to interact properly with various AMPX modules. For example, the GENDF-scattering matrices for fission are identified by MT = 18, but use of this identifier on the AMPX master interface would lead to undesirable results, and it is redefined to be 9018. Likewise, MT = 221 . . . for thermal-scattering processes are converted to MT = 1007, 1008. . . to interface with the AMPX procedures.

6. The fission spectrum on the GENDF file is in scattering-matrix form (a more correct form), whereas it is generally expected to be a single array on the AMPX interface.

The basic procedure in SMILER is very simple. Note that even though a GENDF file can contain many (up to the number of groups) collections of records for a process at a single temperature, these can be collected into a single record before they are shuttled to a direct-access scratch file. Furthermore, if one chooses a procedure that constructs all of the matrices for Legendre coefficients of scattering processes in core prior to writing to the direct access file, the requisite I/O operations are minimized.

Sample Input

```
0$$ 1 2 0 0 1$$ 92235 E T
```

This input will create an AMPX master library on logical unit 1 for ^{235}U data taken from the NJOY GENDF file on logical unit 2. (Note that SMILER only processes one nuclide at a time so that the identifier of the data on GENDF is not required; i.e., the GENDF file must be for only one nuclide.)

Logical Unit Parameters

Variable	Unit number	Type	Description
MMT		binary	logical unit of AMPX master interface
MG1		binary	first GENDF file
MG2		binary	second GENDF file
MG3		binary	third GENDF file
	9	direct access	scratch

A-48. SPLICER - SETS THE FUNCTIONS ON A TAB1 FILE TO ZERO BETWEEN EL AND EH, OR CHOP TO THE GIVEN RANGE, OR SPLICE WITH DATA

SPLICER sets the functions on a TAB1 file to zero between el and eh. It is recommended to use the DCON module following the use of SPLICER to make sure the partials sum to the appropriate total values.

Input Data

Block Specifications

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
in1=		31	input TAB1 file
in2=		0	input TAB1 file
out=		33	output TAB1 file
el=		1.0e-5	lower energy value of range to zero negative lower energy causes the lowest energy of the tab1 data in unit IN1 to be picked
eh=		3.0e7	upper energy value of range to work on
option=		1	procedure to perform 1 - splices the data on in2 between el and eh 0 - zeroes data between el and eh -1 - only copys data between el and eh

Sample Input

```
in1=31 el=1e-5 eh=3.0 out=33 option=0
```

All values between 1e-5 eV and 3.0 eV should be set to zero on the data on logical unit 31 and saved on logical unit 33.

Sample Input

```
in1=31 el=1e-5 eh=3.0 option=-1 out=33
```

Only the values between 1e-5 eV and 3.0 eV are copied on the data on logical unit 31, and they are saved on logical unit 33.

Sample Input

```
in1=31 in2=33 el=1e-5 eh=3.0 option=1 out=35
```

If material, reaction and temperature match, data on logical unit 33 are spliced into the data on file 31 in the range 1e-5 eV to 3 eV. The output data are saved on logical unit 35.

Logical Unit Parameters

Variable	Unit number	Type	Description
out		binary	output TAB1 file
in in		binary	
		binary	

A-49. TABASCO - MODULE TO READ FUNCTIONS FROM AN AMPX MASTER LIBRARY AND WRITE THEM TO A TAB1 FILE AS HISTOGRAMS

TABASCO (TAB1 functions from AMPX/SCALE Master Libraries originally) is a module one can use to extract data from an AMPX master library and have it written onto a single precision binary TAB1 library as histograms equivalent to the averaged values in the master library. These histograms can then be plotted or used in other applications that need these data.

Input Data

Block 1

-1$ Indicates whether worker [1]

1.	worker	If negative, a working library is read. (1)

0$ Logical unit assignments [2]

1.	MMT	the logical unit of the input AMPX Master file (31)
2.	LOGOUT	the logical unit of the output TAB1 file (32)

1$ Number of classes of data to select [1]

1.	NCOM	the number of classes of data to select

Terminate block 1 with a T.

Block 2

2$ Identifiers of materials selected [NCOM]

1.	MATS	Identifiers of Materials selected Zero entry selects everything

3$ Process identifiers selected [NCOM]

1.	MTS	Process Identifiers selected Zero entry selects everything

4$ Not used [NCOM]

1.	NNUSED1	Not used

5* Temperatures selected [NCOM]

1.	NNUSED2	Not used

6* Sig0s selected [NCOM]

1.	NNUSED3	Not used

Terminate Block 2 with a T.

Sample Input

```
0$$ 23 24 1$$ 2 T
2$$ 1395 1398 3$$ 1 0 T
```

This input indicates that data should be read from the AMPX master library on logical unit 23, and data should be written to a TAB1 file on logical unit 24. The total cross section (MT=1) for MAT=1395 is selected, and all processes are selected for MAT=1398.

Logical Unit Parameters

Variable	Unit number	Type	Description
NMT		binary	logical unit of the output
LOGOUT		binary	TAB1 file

A-50. TGEL - MODULE TO CALCULATE TOTAL CROSS SECTIONS FOR FUNCTIONS WRITTEN IN TAB1 FORMAT

The module tgel is a program that will add up partial cross sections to form either an elastic (MT=1007+MT=1008), inelastic, capture, absorption, nonelastic, or total cross section. This ensures that values for redundant cross sections are consistent.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
input=		1	single or double precision input TAB1 file
output=		2	single or double precision output TAB1 file
eps=		1e-4	precision to which to calculate cross section data
total			reconstruct total cross section
capture			reconstruct capture cross section
absorption			reconstruct absorption cross section
inelastic			reconstruct inelastic cross section
thermal			reconstruct thermal cross section
nonelastic			reconstruct nonelastic cross section

Logical Unit Parameters

Variable	Unit number	Type	Description
input		binary	single or double precision input TAB1 file
output		binary	single or double precision output TAB1 File
	99	binary	scratch

A-51. TOMATO - MODULE TO CHANGE MATERIAL IDENTIFIERS (MAT NUMBERS) ON A TAB1 FILE

TOMATO (Toss MAT numbers on a TAB1 file) is a module that allows the user to change the material identifiers (MAT numbers) on a TAB1 file. Isolating this simple functionality makes it much easier to develop and use other modules, such as the SPLICER module.

Input Data

Block 1

-1$ Core allocation [1]

	1.	ICORE	not used (50000)

0$ Logical unit assignments [2]

	1.	LOGIN	logical unit of the input TAB1 file (31)
	2.	LOGOUT	logical unit of the output TAB1 file (32)

1$ Number of materials to change [3]

	1.	NMAT	number of materials whose identifiers should be changed (0)
	2.	NMT	number of reactions whose identifiers should be changed (0)
	3.	NMF	number of file numbers whose identifiers should be changed (0)

Terminate block 1 with a T.

Block 2

2$ Identifiers of materials whose identifiers should be changed [NMAT]

	1.	NMATold	identifiers of materials whose identifiers should be changed only used if NMAT > 0

3$ New identifiers for the material [NMAT]

	1.	NMATnew	new identifiers for the material. only used if NMAT > 0

4$ Identifiers of reactions whose identifiers should be changed [NMT].

	1.	NMTold	identifiers of reactions whose identifiers should be changed only used if NMT > 0

5$ New identifiers for the reactions [NMT]

	1.	NMTnew	new identifiers for the reactions only used if NMT > 0

6$ Identifiers of file numbers whose identifiers should be changed [NMF]

	1.	NMFold	identifiers of file numbers whose identifiers should be changed only used if NMF > 0

7$ New Identifiers for the file numbers [NMF]

	1.	NMFnew	new identifiers for the file numbers only used if NMF > 0

Terminate block 2 with a T.

Sample Input

```
    -1$$ 500000 0$$ 23 24 1$$ 2 T
    2$$ 1395 1398 3$$ 1495 1498 T
```

This input indicates that 500,000 words of core should be allocated to TOMATO and that data should be read from the TAB1 library on logical unit 23, and data should be written to a new file on logical unit 24. The identifiers on the original library are changed from 1395 to 1495 and from 1398 to 1498. Other than these two nuclides, all will be copied with their identifiers unchanged.

Logical Unit Parameters

Variable	Unit number	Type	Description
LOGIN		binary tab1 file	logical unit of the input TAB1 file
LOGOUT		binary tab1 file	logical unit of the output TAB1 file

A-52. WORM - AMPX MODULE TO CONVERT AN AMPX WORKING LIBRARY TO AN AMPX MASTER LIBRARY

WORM (Working to Master Converter) is an AMPX module that converts a binary AMPX working library into a binary AMPX master library. WORM works with any working library to produce a master library containing neutron and/or gamma and/or gamma-production information. In the case of the working library containing more than one of the above types of data, WORM automatically splits the transfer matrices so that all neutron data are carried together and identified by MT = 1, gamma production data are carried together and identified by MT = 1, and, likewise, gamma data are carried together and identified by MT = 501. One-dimensional (reaction averages) cross sections are carried on a process-by-process basis, exactly as in the master library. Only the total transfer matrices are available, since it is impossible to split out individual transfer processes in a general manner once they are added together to produce the working library.

Input Data

Block 1

-1$ Core allocation [1]

	1.	ICORE	number of words to allocate to WORM (50,000)

0$ Logical unit assignments [1]

	1.	MMT	master library is written on this logical unit. (1)
	2.	MWT	working library is mounted on this logical unit. (4)
			Terminate Block 1 with a T.

Sample Input

```
0$$ 1 2 T
```

This input instructs WORM to convert the sets of data on the AMPX working library on logical unit 2 into the formats used on an AMPX master library and to write them on logical unit 1.

Logical Unit Parameters

Variable	Unit number	Type	Description
MMT		binary	Master library is written on this logical unit.
MWT		binary	Working library is mounted on this logical unit.
	17	binary	scratch
	18	binary	scratch

A-53. X10 - MODULE TO PRODUCE MG LIBRARIES FROM THREE TABULAR FILES

X10 is the AMPX module for generating MG libraries. In its present form, it only generates neutron interaction, gamma-ray yield, or gamma-ray interaction cross sections.

Because there were three independent situations (neutron-neutron, gamma-gamma, and neutron-gamma) to address and because production capabilities were expected to accommodate other types of coupling (for example, neutrons produced by gamma rays), the design choice was a single system that is expandable to cover situations not previously addressed.

X10 can accomplish the above goals by accepting data from three tabular files:

1. a file in TAB1 format that contains point cross sections,

2. a file in TAB1 format that contains a smooth weighting function, and

3. a tabular kinematics file generated by Y12.

X10 does not do physics; all kinematic data are in the LAB and in fully double differential form given in Legendre order or cosine moments. Since all processes for all particles use the same coding, it can be argued that the code will be easier to maintain since, in most situations, one can argue that the code will either work correctly or will always work incorrectly.

Another difference in the way X10 calculates transfer matrices is that it always uses a group structure for the source particle and another for the sink particle. The coding is performed in a manner that always considers the source group structure when making integrations for the interacting particle, and it always considers the sink group structure when making integrations for the particle that is produces, even when the two particles are the same. Nothing informs the integration routines that they are dealing with neutrons, photons, protons, etc.. In fact, the simple observation is made that these routines can be used to produce energy- absorption coefficients (or dose factors) simply by specifying the energy absorbed when a particle undergoes a particular type of reaction. Though it may have no practical application, one could go further and specify a group structure for the dose factors, which would also be a functions of scattering angle, just like typical scattering matrices.

The same routines that calculate scattering matrices also calculate averaged cross sections and multiplicities, such as nu-bar (which must be weighted by a combination of a cross section times a flux). This procedure makes it easier to ensure consistency between group-averaged values and transfer matrices.

Note that X10 never reads an ENDF/B library. Other modules (POLIDENT and Y12, for example) read these and produce point cross section files and kinematics files.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
type=		neutron	execution mode for x10 neutron - generates neutron interaction data yield - generates gamma-ray yield data gamma - generates gamma-ray interaction data
tab1=		32	logical unit containing the Doppler-broadened point data
logwt=		30	logical unit containing the pointwise weighting function
matwt=		99	material for the weighting function
mtwt=		1099	reaction for the weighting function
logebdry=		77	logical unit for file containing energy boundary information
master=		1	logical unit for output AMPX master
title=			title to use for the AMPX master
kin=		0	logical unit containing kinematic data
id=		0	id of material to process on point-wise and kinematic file
nl=		5	maximum Legendre order in final master
igm=		0	number of neutron groups to use
iftg=		0	number of first thermal group
ipm=		0	number of gamma groups to use
eps=		1e-5	Precision at which to construct the mesh
pot=		0.0	potential scattering cross section to write into master
upscatter			If present, add an upscatter correction for thermal point-wise data
eup=		3.0	if performing upscatter correction, the energy at which correction starts
eterm=		5.0	if performing upscatter correction, the energy at which all upscatter is eliminated

Logical Unit Parameters

Variable	Unit number	Type	Description
tab1		binary	logical unit containing the Doppler-broadened point data
logwt		binary	logical unit containing the pointwise weighting function
logebdry		binary	logical unit for file containing energy boundary information
kin		binary	logical unit containing kinematic data
master		binary	logical unit for output AMPX master

A-54. Y12 - CREATE KINEMATIC DATA FILES

This module creates kinematic files for incident neutron and gamma data, as well as thermal moderators. Y12 saves the kinematic file in cosine moments or Legendre moments for use in MG processing or as point-wise kinematic data for use in CE library processing.

Input Data

Block Input

Block starts on first encounter of a keyword in the block.

Keyword	Alternate	Default	Definition
ndf=		11	logical unit of ENDF file to process
mat=			material number to process
kin=		31	logical unit of output kinematic file
point=		-1	logical unit of file containing 1-D point data If less or equal to 0, the point wise data will not be generated.
id=		-1	ID to be used on the kinematic and 1-D point file If less or equal to 0, this will be the same as the mat number on ENDF.
eps=		1e-3	precision at which to generate the grid
nl=		5	if saving in Legendre coefficients of cosine moments, the number of moments to generate
emax=		5.05	if processing thermal moderator data or free gas data, the upper energy limit
emin=		1e-5	if processing thermal moderator data or free gas data, the lower energy limit
free			if present, generates free gas data
awr=			if processing free gas, the mass ratio to use
pot=			if processing free gas, the free atom scattering cross section
temp=			space-separated list of temperature(s) in Kelvin at which to generate free gas data
coform=		yes	option to apply the form factor for Klein-Nishina scattering yes - applies the factor no - does not apply the factor
awp=			If given, kinematic data should only be processed for particles with this mass ratio.
zap=			If given, kinematic data should only be processed for particles with this ZA value.
for=		tab	desired output format tab - generates tabulated double differential data cos - generates data in cosine moments leg - generates data in Legendre coefficients

Logical Unit Parameters

Variable	Unit number	Type	Description
kin		binary	logical unit of output kinematic file
ndf		binary	logical unit of ENDF file to process

A-55. ZEST - MODULE TO MANAGE STRING LIBRARIES

ZEST (Zippy ensembler of strings) is a module analogous to AJAX, except ZEST uses string libraries such as those produced by POLIDENT. A string is a TAB l record in ENDF nomenclature. Options are provided to allow merging from any number of files in a manner to allow the user to determine the final nuclide ordering, if desired.

Input Data

Block 1

-1$ Core assignment [1]

1.	NWORD	not used (50000)

0$ Logical assignments [2]

1.	LOG	logical number of library to be written (31)
2.	LBIG	writes out in single or double precision (0) 0: double 1: single

1$ Library Selector [1]

1.	NLOG	number of commands (or libraries) required to create LOG (1) Terminate block 1 with a T.

Stack Block 2 and 3 one after the other NLOG times.

Block 2

2$ Input library selection [2]

1.	NLIN	logical number of input library
2.	NC	how the strings are to be treated (0) ☐ -N: deletes N strings from NLIN to create LOG. ☐ 0: accepts all strings from NLIN. ☐ N: adds N strings from NLIN to create LOG

Terminate Block 2 with a T.

Only use Block 3 if NC != 0.

Block 3

3$ MAT numbers From NC [NC]

1.	MAT	material identifier(s) of nuclides to be added or deleted. (0) Only used if NC != 0 There must be exactly NC values.

4$ MT numbers from NC [NC]

1.	MT	reaction identifier(s) of nuclides to be added or deleted. (0) Only used if NC != 0. There must be exactly NC values.

5$ MF numbers from NC [NC]

1.	MF	File identifier(s) of nuclides to be added or deleted. (0) Only used if NC != 0. There must be exactly NC values.

6$ New MAT numbers from NC [NC]

1.	MATnew	New material identifier(s) of nuclides to be added. (0) Only used if NC > 0. A zero leaves the identifier unchanged.

7$ New MT numbers from NC [NC]

1.	MTnew	New reaction identifier(s) of nuclides to be added. (0) Only used if NC > 0. A zero leaves the identifier unchanged.

8$ New MF numbers from NC [NC]

1.	MFnew	New file identifier(s) of nuclides to be added. (0) Only used if NC > 0. A zero leaves the identifier unchanged

Terminate Block 3 with a T.

Sample Input

```
0$$  31  0     1$$ 10 T
2$$  21  0     T
2$$  22  0     T
2$$  23  0     T
2$$  24  0     T
2$$  25  0     T
2$$  26  0     T
2$$  27  0     T
2$$  28  0     T
2$$  29  0     T
2$$  30  0     T
```

This input instructs ZEST to combine the contents of the ten-point cross section libraries on logical units 21–30 onto a single point cross section library on logical unit 31. Note that the order in which the point cross section libraries are accessed determines the ordering on the output library so that a case like this can be used to force an ordering.

Logical Unit Parameters

Variable	Unit number	Type	Description
LOG		Binary	Logical number of library to be written
NLIN		Binary	