ADER: Advanced Depletion Extension for Reprocessing

Ver: 1.0 - A SERPENT2 Extension

User Manual

Daniel D. Wooten

June 9, 2019

Contents

1 Preface

This document is intended for users of SERPENT2 wishing to utilize the **A**dvanced **D**epletion **E**Extension for **R**eprocessing (ADER) extension. **This manual assumes the reader is proficient with SERPENT2.** General information about SERPENT2 can be found at the SERPENT2 wiki:

```
serpent.vtt.ft/mediawiki/index.php/Main_Page
```

With regards to citing SERPENT2 and ADER, as stated on the SERPENT2 wiki general reference to SERPENT2 may be provided by - J. Leppanen, M. Pusa, T. Vittanen, V. Valtavirta, T. Kaltiaisenaho. *"The Serpent Monte Carlo code: Status, development, and applications in 2013"*. Ann. Nucl. Energy, **82** (2015) 142 - 150. Reference to ADER may be provided by - D. D. Wooten. *ADER - Advanced Depletion Extension for Reprocessing.* (2019).

ADER makes extensive use of the open-source library, Clp, part of the COIN-OR collection of packages. Supporting documentation and the source code for Clp can be found at:

```
https://github.com/coin-or/Clp
```

Of importance is that Clp is distributed under the Eclipse Public Licence which is not a copy-left licence but a rather forgiving open-source licence. Instructions for installing and linking the Clp libraries can be found in section 12.
Examples of SERPENT2 input are given

```
in this font
```

SERPENT2 and ADER keywords, user inputs which must be a certain word, appear underlined as seen below. **Keywords not enclosed in brackets must appear in the position of the entry they are seen in. Keywords enclosed in brackets may appear anywhere after this opening keyword but must be followed by their input value if they have one.**

```
keyword
```

The following list of symbols are not to be considered "input". Rather, they are delimiters for this manual and should not be considered part of any ADER commands or inputs...

```
[ ] { } < > ( ) ; ...
```

Required user inputs are denoted...

```
[This_entry_is_set_by_the_user]
```

Optional user inputs are denoted...

```
<This_entry_may_be_excluded>
```

Required user choices from a list are denoted...

```
[{option_1; option_2}]
```

Optional user choices from a list are denoted...

```
<{option_1; option_2}>
```

User inputs which are required *if* a previous input was given are denoted...

```
<optional_input> (required_input_due_to_earlier_input)
```

If an example of a user input includes a structure such as...

```
[Entry_1] [some_number]
...
<Entry_n> (some_number)
```

This indicates that at least one instance of `Entry x` must be input by the user but as many instances of type `Entry x` may be input by the user. Additionally the presence of the second bracketed input indicates that each instance of `Entry x` requires an entry of `some_number`. Examples of generic SERPENT2 input, input templates, are given in a light-gray box with black mono-spaced font such as the example seen below...

```
mat [mat_name] [mat_den] <{vol,mass}> [mat_val] <burn> (burn_segments)
[isotope_1_zai].[temp_lib] [iso_frac]
<isotope_n_zai>.(temp_lib) (iso_frac)
```

While any specific use examples, implementations of SERPENT2 input, are given in a black box with white mono-spaced font such as the example seen below…

```
mat water -1.0 vol 1.0 burn 0.0
1001.06c    2
...
1608.06c    1
```

2 Introduction

Please be sure to read the preface for important information.
In the most direct sense ADER seeks to accomplish two tasks: determining an optimal material composition given a set of constraints, and integrating the necessary composition adjustments into a nuclear material evolution model. ADER brings to Serpent 2 the ability to define groups of elements, isotopes, and chemicals; the ability to define relationships between these groups; and the ability to move these groups into, out-of, and between materials. Furthermore, ADER provides the ability to set k_{eff} targets, to prescribe mass transfers within a given system, and to set weighted oxidation state targets for materials. Bringing all of these capabilities together ADER employs the COIN-OR linear optimization (CLP) package to determine the material flows, as set by the user, that will best satisfy an optimization target, also set by the user. Sufficiently simplified - ADER is a material-composition, linear-optimization engine for the SERPENT2 burnup routines.

This manual serves as both a reference for the theory of ADER and as a guide to its usage. As such each section has a "Quick Reference" subsection where direct and simplified answers to common usage questions and input formatting can be found. However, many users will find that a deeper understanding of the theory behind ADER, as well as the nuances of its use, will serve them well. This manual is organized to be red through in a linear fashion by a first time user beginning with the concept of a "group" as it is used in ADER and building up the ADER toolkit from there.

To provide context for the explanation of the components of ADER components of an example simulation are built up throughout this manual. These components only serve as rough examples. That said, for this example consider a fuel salt for a liquid fuel molten salt reactor composed of LiF salt at 71.7 mol-fraction, BeF_2 salt at 16 mol-fraction, UF_3 at 0.023 mol-fraction, UF_4 at 2.277 mol-fraction, and ThF_4 at 10 mol-fraction. Take this salt to have an initial neutron multiplication factor of 1.0 (it does not in reality) and to be in an infinite lattice with a graphite moderator which takes up 50% of the volume. The SERPENT2 material based off this example is shown below. **The keyword "ader" must be included in the definition line of any material which is to be managed by the ADER extensions to SERPENT2 - i.e. the material is either connected to a stream or to a conditions block.** ADER input is entered as a "regular" component of SERPENT2 input files.

```
mat FuelSalt -2.805 vol 1 burn 0 ader
3006.06c    0.00028
3007.06c    0.28323
4009.06c    0.06328
9019.06c    0.60450
90232.06c   0.03954
92233.06c   0.00046
92238.06c   0.00864
```

3 Groups

At the most fundamental level a group in ADER is a list of proportions. This list describes the relative proportions of elements within the group. These elements may or may not have specified isotopic proportions and elements without a specified isotopic composition are permitted in the same group as elements *with* a specified isotopic composition. Consider a group which specifies that fluorine be four times as abundant as uranium, within the group. This group could be used to describe a single molecule of UF_4 or, just as equivalently, one mol of uranium and four mols of fluorine all in a bucket on a lab bench. A group is not defined by a quantity of material, but rather a recipe for a material. Just like defining materials in SERPENT2, the proportions of an element within a group are normalized to unity and the proportions of an isotope within an element are normalized to unity. As such groups in ADER are defined according to four methods A, B, C, or D as seen bellow.

Method A:

```
grp [Name]
[Element_1] [Element_1_frac]
...
<Element_n> (Element_n_frac)
```

Method B:

```
grp [Name]
[Element_1] [Element_1_frac] <isos> (n_isos)
(Iso_1) (Iso_1_frac)
<Iso_n> (Iso_n_frac)
...
<Element_x> [Element_x_frac] <isos> (b_isos)
(Iso_1) (Iso_1_frac)
<Iso_b> (Iso_b_frac)
```

Method C:

```
grp [Name]
[Element_1] [Element_1_frac]
...
<Element_x> [Element_x_frac] <isos> (b_isos)
(Iso_1) (Iso_1_frac)
<Iso_b> (Iso_b_frac)
```

Method D:

```
grp [Name] [sum]
[Name_of_group_1] 1
...
<Name_of_group_n> 1
```

The keyword `grp` begins the group section input. All input following this word, until the next keyword is found, is taken as input for the aforementioned `grp`. The `Name` entry is an identifier by which the group will be located throughout the code - as such all groups require a unique name. In method A a group is defined only by the proportions of its elements. The relative abundance of each element within the group is denoted by `Element_x_frac` while the values for `Element_x` must be the alphabetic periodic table shorthand for that element, Ag for gold as an example. Elements with no prescribed isotopic composition are permitted to take on any combination of their own isotopes.

In method B a group is defined in which each element has a list of specified isotopic proportions. The values for `Iso_n` must be the alphanumeric name of the isotope in the form "Alpha-A", or U-233 for ^{233}U The proportions entered, `Iso_n_frac`, are of the isotope relative to the element as a whole. In method C a group is defined in which some of its elements have a specified list of isotopic proportions and some do not. Those elements which do not have a prescribed list of isotopic proportions are permitted to have any combination of their own isotopes. Lastly, in method D a new type of group is introduced - the summation group. A summation group is defined as the sum of the groups named in its list of component groups. These component groups may be defined using methods A, B, C, or D, allowing for nested summation groups. The value which follows a group name in a summation group definition may be any value greater than 0; however, any value other than 1 could lead to a non-physical answer in the simulation. Please see section 11 for a more detailed explanation.

The `Element_x_frac` values for each group are all summed together and then normalized to this sum. This same procedure is done for all isotopic values within an element's list. For example, consider the group `gUF4` defined below, it is composed of 20% uranium and 80% fluorine. The fluorine is permitted to have any combination of fluorine isotopes while the uranium in this group is specified to be 5% ^{233}U and 95% ^{238}U.This is an example of method C of group definition.

```
grp gUF4
U   1   isos    2
U-233   5
U-238   95
F   4
```

The group `gFLiBe` is defined below as an example of method A, group `gUF3` is defined below as an example of method B, while group `gUF` is defined below as an example of method D. These groups will be used throughout this manual in examples.

```
grp gFLiBe
Li  71.7
Be  16
F   103.7

grp gUF3
F   3   isos    1
F-19    1
U   1   isos    1
U-233   1

grp gUF sum
gUF4    1
gUF3    1
```

The following paragraphs assume some level of familiarity with ADER and will likely be unclear until sections 4 and 5 have been reviewed.

Groups are used to define four structures in ADER; ranges, ratios, streams, and summation groups. These structures are covered in later sections of this manual but their interaction with groups requires a comment here. When a group is used in a range or ratio structure, or is used in a summation group which is part of a range or ratio structure, and this structure is attached to a material by a conditions block, the group in question is added to the material's list of possible recipes. These recipes are the options into which the material's constituent isotopes may be sorted during the optimization step, discussed in section **??**. Range and ratio structures, in the same material, make use of the same group list. That is to say, in the following example seen below, in which the conditions block `example_block` is attached to the material `FuelSalt`, the group `gFLiBe` must **both** be between 20 and 80% of `FuelSalt`'s atomic density **and** be four times as abundant as the group `gUF4`. There are **not** now two copies of the group `gFLiBe` assigned to the material `FuelSalt`. The `rng` and `rto` structures each refer to the **same** group within a single material, in this instance the material `FuelSalt`.

```
conditions example_block
rng gFLiBe min 0.2 max 0.8
rto gFLiBe val 4 grp2 gUF4
```

```
mat FuelSalt −2.805 vol 1 burn 0 ader cnd example_block
3006.06c    0.00028
...
```

The groups used to define group-class streams are used solely as recipes. The groups used to define group-class streams *only* specify the relative proportions of the mass carried by the stream in question. For a stream to be connected to a material, that material *does not* need to have that group in its list of possible recipes. For instance, `FuelSalt` to which the conditions block above, `example_block`, is attached could be the sink for a group-class stream defined using the group `gLi`, a group of elemental lithium, even though there is no usage of the group `gLi` in the conditions block `example_block`. A group-class stream's usage of a group only stipulates what proportions the mass that stream carries must have. If the mass the stream carries originates from a material the isotopes which compose the stream's load may come from any group or no group at all inside of that material. If the mass the stream carries ends in a material the isotopes entering the material may go into any group into which they fit the recipe, or no group at all if the isotope is not a "controlled" isotope - a topic discussed in section 4.3.

Again, the groups defined using the `grp` keyword are only recipes. A given recipe can be used to define multiple structures. Streams all create their own copy whereas range and ratio restrictions in the same material refer to the same group in a material's recipe list. As an example, in the input below, there exist four digital versions of the group `gFLiBe`, the original `grp` recipe, the group which has been added to the material `FuelSalt`'s recipe list, and each group created for each stream; and while these two streams represent two distinct stream objects, the optimization line seen in the conditions block will minimize the *sum* of the two streams because they have each been defined with a group of the same name.

```
grp gFLiBe
Li  71.7
Be  16
F   103.7

conditions example_block
rng gFLiBe min 0.2 max 0.8
rto gFLiBe val 4 grp2 gUF4
opt dir min type spec_stream gFLiBe

mat FuelSalt −2.805 vol 1 burn 0 ader cnd example_block
3006.06c    0.00028
...

stream to FuelSalt type feed form cont group gFLiBe
stream to FuelSalt type reac form disc group gFLiBe
```

3.1 Quick Reference

- The proportions of elements within a group are normalized to unity.
- The proportions of isotopes within an element are normalized to unity.
- Only isotopes belonging to the same element may be listed as a component isotope of that element.
- General Input Structure where `Element_x` must be the alphabetic periodic table designation for the element, and `iso_x` must be the alphabetic periodic table designation for the parent element followed by a dash and then the atomic number of the isotope in question. The value which follows a group name in method D of defining a group may be any value greater than 0; however, any value other than 1 could generate a non-physical answer for a given simulation.

```
grp [Name] <sum> [{Element_1; group_1}] [{Element_1_frac; 1}] ...
<isos> (num_isos) (iso_1) (iso_1_frac) <iso_n> (iso_n_frac) ...
```

```
<{Element_n; group_n}> [{Element_n_frac; 1}] ...
<isos> (num_isos) (iso_1) (iso_1_frac) <iso_n> (iso_n_frac) ...
```

4 Conditions Blocks

A conditions block is the structure through which limits on a SERPENT2 material are defined. A conditions block consists of, minimally, the elements seen below where `conditions` is the keyword and `Name` is the unique identifier given to this specific conditions block.

```
conditions [Name]
```

Consider this conditions block `limits_block` as defined below.

```
conditions limits_block
```

A conditions block, and all of the limitations it may carry, are applied to SERPENT2 materials by including the key and value pair, `cnd [Name]`, in the material's definition; the information that follows the `mat` keyword.

```
... [cnd] [Name] ...
```

Building upon the example presented in the introduction, to attach a conditions block with the name of `limits_block` the material definition line would be amended to look like...

```
mat -2.805 FuelSalt vol 1 burn 0 ader cnd limits_block
```

The same conditions block can be attached to multiple materials with each material receiving its own unique set of the limitations - this reduces input duplication. To add limitations to a conditions block there exist six possibilities: ranges, ratios, control tables, oxidation tables, preservation options, and optimization options. Each will be covered below in the following subsections.

4.1 Ranges

Using the `rng` keyword the fraction of a material, by percent atom per cubic centimeter ($\frac{\%}{cm^3}$) which must adhere to the recipe of a specific group may be set. This fraction may be either a single value, seen in method A, or a range of values, as seen in method B. Values less than 0 are not accepted but there is no upper limit for the material fractions. Using a group in a `rng` structure which is attached to a material via a condition block is one of two methods to "attach" a group to a material - the meaning of this is expanded upon in section 4.3
Method A:

```
rng [grp_name] val [value]
```

Method B:

```
rng [grp_name] [min] [min_value] [max] [max_value]
```

Take g_u to be the fraction of a material's atomic density which is attributed to group u, in which case method A sets up the relation seen in equation 4.1 and method B sets up the relation seen in equation 4.2.

$$g_u = \texttt{[value]} \tag{4.1}$$

$$\texttt{[min\_value]} \leq g_u \leq \texttt{[max\_value]} \tag{4.2}$$

Now consider the conditions block shown below which was attached to the material `FuelSalt` at the beginning of this section, 4.

```
conditions limits_block
rng gUF4    min 0.03    max 0.10
```

By adding a `rng` input to the conditions block `limits_block` material `FuelSalt` is now required to have at least 3% and no more than 10% of its atomic density accounted for by isotopes which would fit the recipe of group `gUF4`. Furthermore, these isotopes may not be counted towards inclusion in any other group structure attached to the material `FuelSalt`.

4.2 Ratios

The keyword `rto` is used to describe the permitted relative abundance of groups within a material. Like `rng` structures `rto` entries can be make according to methods A or B seen below. Values equal to or less than 0 are not accepted. Using a group in a `rng` structure which is attached to a material via a condition block is one of two methods to "attach" a group to a material - the meaning of this is expanded upon in section 4.3 Method A:

```
rto [grp1_name] [val] [value] [grp2] [grp_2_name]
```

Method B:

```
rto [grp1_name] [min] [min_value] [max] [max_value] [grp2] [grp_2_name]
```

Method A sets up the relation seen in equation 4.3 and method B sets up the relation seen in equation 4.4.

$$\frac{g_1}{g_2} = \texttt{[value]} \tag{4.3}$$

$$\texttt{[min\_value]} \leq \frac{g_1}{g_2} \leq \texttt{[max\_value]} \tag{4.4}$$

Now, consider adding the following line to the conditions block, `limits_block`, which has served as the example throughout this section.

```
conditions  limits_block
rng  gUF4     min  0.03      max  0.10
rto  gFLiBe   min  4.0  max  99   grp2      gUF
```

This `rto` line specifies that in the material `FuelSalt` the fraction of the material's atomic density which is accounted for by the group `gFLiBe` must be no less than 4 times as prevalent and no more than 99 times as prevalent as the fraction of the material's atomic density which is accounted for by the group `gUF` - which is a summation group.

4.3 Control Tables

In the ADER framework there is no requirement that all of a material's atomic density be accounted for by group structures - groups are permitted to occupy as little or as much of a material's composition as the optimal solution requires. Consider a group, `gNi`, which is defined as elemental nickel and which is limited to be less than 2% of a material's atomic density. One way to meet this requirement is simply to assign none of the nickel isotopes in the material to the group `gNi`. In that case 0% of the material's atomic density is accounted for by the group `gNi`.

Control tables offer a means to prevent the trivial answer reached above. A control table is a list of elements and isotopes. When attached to a material a control table specifies that any isotopes in the material which are listed on the control table, or which are a member of an element listed on the control table, must be fully accounted for by a group structure attached to the material. Groups become attached to a material by being a component of an `rng` or `rto` entry in a conditions block which is assigned to the material in question. In the example above, if there are no other groups for the nickel isotopes to go into, if nickel is a controlled element in the material in question, all of the nickel isotopes in the material would be forced into the `gNi` group. An element or isotope not listed on a control table is referred to as a "free" element or isotope as these constituents are not bound to be in groups.

A control table is defined in the following manner where elements are denoted by their alphabetic periodic table designation and isotopes by adding a dash followed by the atomic number of the isotope in question to the alphabetic periodic table designation for the parent element: i.e. `U` for uranium and `U-233` for ^{233}U.

```
control [table_name]
[element_or_isotope]
...
<additional_element_or_isotope>
```

For example, consider a control table which specifies that all lithium, beryllium, ^{233}U, and thorium must be accounted for by group structures, as seen below.

```
control c_table
Li Be U-233 Th
```

A control table is attached to a conditions block via the entry. . .

```
[cnt] [table_name]
```

Attaching the control table, `c_table`, from above to the conditions block used in this section is seen below.

```
conditions limits_block
rng gUF4     min 0.03     max 0.10
rto gFLiBe   min 4.0  max 99  grp2     gUF
cnt c_table
```

4.4 Oxidation Tables

Oxidation tables are a means of setting a limitation on a weighted sum which is done over all the *elements* in a material. Oxidation tables are constructed as seen below where `value` is multiplied by `weight_value`, if applicable, which produces the weight in the weighted sum done over the elements which are enumerated in the oxidation table. Elements are entered by their alphabetic periodic table designation. Any real number is a valid input for both `value` and `weight_value`.

```
oxidation [Name]
[first_element] [value] <weight> (weight_value)
<nth_element> (value) <weight> (weight_value)
```

As an example consider the oxidation table `oxi_table` defined below. Any elements excluded from an oxidation table's list are given a default value of 0.

```
oxidation oxi_table
H   1
O   -2
```

Oxidation tables are attached to conditions blocks either by method A or method B.
Method A:

```
oxi [Name] val [target_val]
```

Method B:

```
oxi [Name] [min] [min_val] [max] [max_val]
```

Taking ρ_e to be the fraction of a material's atomic density which is attributable to element e an oxidation table, in conjunction with its implementation in a conditions block attached to said material, establishes the relationships seen in equations 4.5 and 4.6 corresponding to methods A and B respectively.

$$\sum_{e}^{E} \rho_e v_e w_e = \texttt{target\_val} \tag{4.5}$$

$$\texttt{min\_val} \leq \sum_{e}^{E} \rho_e v_e w_e \leq \texttt{max\_val} \tag{4.6}$$

Attaching the oxidation table `oxi_table` to the conditions block used throughout this section is demonstrated below using method B.

```
conditions limits_block
rng gUF4    min 0.03    max 0.10
rto gFLiBe  min 4.0 max 99  grp2    gUF
cnt c_table
oxi oxi_table min −0.02 max 0.06
```

4.5 Preservation

Inclusion of the following keyword pair...

```
pres mols
```

into a conditions block adds the limitation to the material that any influx of matter must be equally balanced by a removal of matter with an equal number of atoms; i.e. the atomic density of a material is not permitted to change due to mass flows - only nuclear depletion effects. Only mass flows managed by group-class streams, covered in section 5, are included in this balance. The majority of simulations will find this option to be necessary; otherwise unintended behavior may result.

Including this modifier into the conditions block used throughout this section is seen below.

```
conditions limits_block
rng gUF4    min 0.03    max 0.10
rto gFLiBe  min 4.0 max 99  grp2    gUF
cnt c_table
oxi oxi_table min −0.02 max 0.06
pres mols
```

4.6 Optimization

As ADER is a linear optimization suite, there must be an optimization target. Every material cluster, a concept covered in section 5, with at least one group-class stream, must have one and only one optimization target. Optimization targets are attached to material clusters via a conditions block which is attached to a member of a material cluster. There are nine methods to set optimization targets. The direction of optimization, maximization or minimization, is set in the optimization entry of the conditions table as well. Many of the optimization targets involve concepts related to streams which are covered in section 5.
Method A sets the optimization target as the total value of all feed/reac/redox/remv type streams:

```
opt [dir] [{min; max}] [type] [action] {[feed; reac; redox; remv]}
```

Method B sets the optimization target as the total value of all feed and remv type streams:

```
opt [dir] [{min; max}] [type] [action] [feed_and_remv]
```

Method C sets the optimization target as the total value of all streams:

```
opt [dir] [{min; max}] [type] [action] [streams]
```

Method D sets the optimization target as the total value of all streams which have a valid SERPENT2 material for both the source and sink:

```
opt [dir] [{min; max}] [type] [action] [transfers]
```

Method E sets the optimization target as the total fraction of a material's atomic density which is attributed to the named group. This group *must* have been assigned to a material in the material cluster via either a `rng` or `rto` conditions block entry.

```
opt [dir] [{min; max}] [type] [group] [group_name]
```

Method F sets the optimization target as the total value of all group-class streams defined using the group of name `group_name`.

```
opt [dir] [{min; max}] [type] [spec_stream] [group_name]
```

As an example the optimization target of minimizing all feed type streams is added to the conditions block used throughout this section.

```
conditions limits_block
rng gUF4    min 0.03    max 0.10
rto gFLiBe  min 4.0 max 99  grp2    gUF
cnt c_table
oxi oxi_table min −0.02 max 0.06
pres mols
opt dir min type action feed
```

4.7 Quick Reference

- Conditions blocks are defined as follows...

```
conditions [Name]
```

- Conditions blocks are attached to materials using the `cnd [Name]` key and value pair in a material's definition - i.e. `mat my_mat cnd my_conditions_block`

- Ranges are defined inside a conditions block as follows...

```
rng [grp_name] [{val; min/max}] [value] (max/min) (value)
```

- Ratios are defined inside a conditions block as follows...

```
rto [grp1_name] [{val; min/max}] [value] (max/min) (value)
    grp2 [grp2_name]
```

- Control tables are defined as follows...

```
control [table_name]
[element_or_isotope]
...
<element_or_isotope>
```

- Control tables are added to conditions blocks as follows...

```
cnt [table_name]
```

- Oxidation tables are defined as follows...

```
oxidation [table_name]
[element_x] [value] <weight> (weight_value)
[element_n] [value] <weight> (weight_value)
```

- Oxidation tables are added to conditions blocks as follows...

```
oxi [table_name] [{val; min/max}] [value] (max/min) (value)
```

- To turn on atom-density conservation for a material, add the following line to a conditions block attached to said material...

```
pres mols
```

- Every material cluster with one or more group-class streams must have one and only one `opt` entry attached to one of the conditions blocks which is itself attached to one of the materials in a cluster. `opt` entries are added to conditions blocks as follows...

```
opt [dir] [{min; max}] [type] [{action; group; spec_stream}]
    [{feed; feed_and_remv; reac; redox; remv; streams;
        transfers; group_name}]
```

5 Streams

Streams are the structures by which ADER moves mass into, out of, and between SERPENT2 materials. Streams are defined as seen below...

```
stream <to> (name_of_sink_material) <from> (name_of_source_material)
    [type] [{feed; reac; redox; remv}] [{group; rem}] [name]
    [form] [{cont; disc; prop}] (frac) (frac_value)
```

The `to` and `from` entries specify the stream's source and sink for the mass which the stream moves. Sources and sinks which are given must be valid SERPENT2 materials. Streams do not require *both* a source and a sink, but rather, at least *one* of either a source or sink. In the case of a missing source the mass which is brought into the sink material is assumed to come from an infinite, non-reactive (chemically and neutronically) supply existing outside the simulation bounds. In the case of a missing sink the mass which is removed from the source material is assumed to disappear - it leaves the simulation boundary.

Streams which have elements with unspecified isotopic compositions take these element's isotopic compositions to be equal to that of the elements in the source material at the beginning of the burnup step. These proportions are updated on each ADER criticality search to account for the changes possibly induced by discrete form streams. Streams which have elements with unspecified isotopic compositions and a valid SERPENT2 material as a sink must have a valid SEPRENT2 material as a source.

Material clusters are collections of SERPENT2 materials which are connected via streams. If a stream connects two materials, having a valid SERPENT2 material for each source and sink, these two materials become part of the same material cluster. Because materials may have streams coming from and going to many other materials, not all materials in a material cluster will be directly connected by streams; rather, some materials in a material cluster may happen to only be connected to one another through a series of intermediate streams and materials. Materials in the same cluster share an optimization solution, that is, their compositions are optimized collectively according to the singular optimization target provided, discussed in section 4.6. Of less impact to the user, material clusters also share a collective burnup solution though each material may have its own flux and cross sections. Materials in the same cluster must have the same initial isotopes listed in their definitions whether or not these isotopes exist in each material at the beginning of the simulation. If an isotope needs to be listed in a material for which its concentration is zero, simply include the isotope in the material's definition with a zero value next to its ZAI identifier.

The `type` entry has no effect on stream behavior or implementation. Rather, it serves only as a tag for the `opt` entries (discussed in section 4.6) and as a possible organizational tool for the user.

The following choice of keywords, `group` and `rem`, highlights the most important distinction between streams; the difference between group-class (`group`) and table-class (`rem`) streams. While the details of this distinction will be covered in depth in the following subsections it is sufficient to say, for now, that group-class streams move mass which follows the recipe outlined by a group while table-class streams move mass of which the proportions are determined using a table, a `removal` table (covered in section 5.2). Last but not least, an additional distinction is that group-class streams are variables, the amount of mass which they move is determined by solving the material optimization problem, discussed in section **??** while table-class streams move a fixed amount of mass according to user input. Group-class streams have the name of a group following the keyword `group` while table-class streams have the name of a removal table following the keyword `rem`. **Table-class streams can be used to feed mass into a material. The use of the keyword `removal` is a legacy-term and does not explicitly mean ''removal" in the sense of the English word**.

The `form` keyword and its associated values, `cont`, `disc` and `prop`, designate how a stream behaves in time. The optimization solution, as discussed in section **??**, determines the total mass transfers from the group-class streams interfacing with a material cluster needed to bring the materials in the material cluster to their optimal composition. How this mass is moved over time is up to the user through the use of the `form` keyword. Streams designated as `cont` are "continuous" streams. These streams move mass throughout a burnup step at a constant rate per second. Streams designated as `disc` are "discrete" streams. These streams move mass as an instantaneous input happening before a burnup step begins. Streams designated as `prop` type streams are "proportional" streams and their method of mass transfer is discussed in greater detail in the following sections, 5.1.3 and 5.2.3.

The `frac` keyword and its associated value are discussed in sections 5.1 and 5.2.

An important note, mentioned in several spots throughout this manual, is that only "disc" form streams impact ADER's reactivity iteration scheme. If ADER is instructed to iterate on the material composition to match user specified reactivity targets, ADER will only apply the actions of discrete form streams on each iteration. Any continuous and proportional streams will not have their effects on material reactivity incorporated until after the current burnup step is complete. Additionally, any given material constraint may be out of bounds during some point in a burnup step if streams of mixed type are used together as the optimal conditions for the material rely on the full impact of all streams. As a closing reminder, ADER *does not* account for nuclear burnup in the *current* burnup step. The effects of nuclear burnup on composition are assessed by ADER at the beginning of the next burnup step - i.e. ADER does not predict the effects of nuclear burnup.

5.1 Group-class streams

Group-class streams, those streams using the keyword `group`, operate differently from table-class streams. Group-class streams are options, given by the user to ADER, for moving mass. As discussed in section 11 the amount of mass moved by group-class streams is determined by the optimization solution. That is, the content of group-class streams is determined so that the optimal material composition for the material cluster may be realized.

The mass which a group-class stream moves is described by the group named after the `group` keyword. This named group describes the proportions the mass the stream carries must have. The amount of mass is determined by the optimization solution. The optimization solution operates on a volume normalized basis. As such, take the initial units of any solution for a stream value to be[1] $\frac{\texttt{atoms}}{cm^3}$ where "`atom`" represents a unit of the group-class stream. This unit is composed of fractional isotopes in accordance with the recipe of the group-class stream. Take this value to be denoted $s_{n:x}$ where n denotes different streams and x takes on either c, d, or p representing continuous, discrete, and proportional form streams.

Group-class streams do not make use of the `frac` keyword and as such their input template appears as follows...

```
stream <to> (name_of_sink_material) <from> (name_of_source_material)
    [type] [{feed; reac; redox; remv}] [group] [name]
    [form] [{cont; disc; prop}]
```

5.1.1 Continuous group-class streams

Continuous group-class streams move an equal amount of mass per unit time over the length of a burnup step. Taking Δt to be the total number of seconds in the current burnup step continuous group-class streams deliver or remove $c_n \frac{\texttt{atoms}}{cm^3 s}$ to their sink or source material's respectively where c_n is defined in equation 5.1.

$$c_n = \frac{s_{n:c}}{\Delta t} \tag{5.1}$$

As an example consider below the continuous group-class stream formed using the `gFLiBe` group from section 3 for which there is no source and for which the sink is the material `FuelSalt`.

```
stream to FuelSalt group gFLiBe form cont type feed
```

5.1.2 Discrete group-class streams

Discrete group-class streams move their entire mass load, $s_{n:d}$, collectively and instantaneously following the optimization solution and before the transport sweep preceding the burnup calculation. With further detail available in section 11 discrete group-class streams have a unique aspect different from all other streams. Every burnup step ADER iterates over the optimization solution and a transport sweep until the system analog neutron multiplication factor, k_{eff}^{analog}, is within user defined bounds. Every time an optimization solution is arrived at, if there are discrete streams, these streams will have some value $s_{k,n:d}$ where k denotes

[1]Developers note: The actual units of stream return values in the code are $\frac{\texttt{atoms}}{cm \cdot barn}$

the current iteration over the optimization solution for the current burnup step. The changes in material composition induced by these discrete stream flows, $s_{k,n:d}$, are applied before the next transport sweep meaning that the system compositions reflect the actions of discrete group-class streams. Should another solution of the optimization problem, for the same burnup step, be required, the changes induced by these discrete group-class stream flows, $s_{k,n:d}$, are not undone. Rather, the final $s_{n:d}$ value reported for discrete group-class streams per burnup step is given by equation 5.2.

$$s_{n:d} = \sum_{k}^{K} s_{k,n:d} \tag{5.2}$$

As an example of a discrete group-class stream consider the stream below formed using the group `gUF4`, again, with no source but the material `FuelSalt` as the sink.

```
stream to FuelSalt group gUF4 form disc type reac
```

5.1.3 Proportional group-class streams

Proportional group-class streams attempt to move mass in proportion to the mass already present. Proportional group-class streams approximate a decay-like proportional constant, $\lambda_{i,n:p}$, which in a system with no other effects acting on isotope concentration, would lead to the desired change in the isotope's, i, concentration. This is almost never the case in an ADER simulation and as such the realized change in an isotope's concentration will likely be different from the desired amount of change. As such, the values reported in the output by ADER for all proportional streams, group-class or table-class, are calculated exactly during the burnup solution such that they reflect the actual amount of the isotope moved, not the pre-calculated amount which is likely different. Regardless, considering the role that group-class streams tend to play in ADER simulations the usage of proportional group-class streams should be undertaken with great caution and a thorough understanding of how the simulation parameters will interact to affect the proportional group-class stream constants. Proportional group-class streams may not be formed using summation groups.

For a proportional stream removing a quantity, $s_{n:p}$, from a material, m, the isotopic proportional constants, $\lambda_{i,n:g}$ for isotope i as modified by group stream n are calculated, before the burnup step and *after* the application of any discrete stream effects (such that the proportional constants are calculated from the updated material composition values), as seen below in equation 5.3 where $f_{i,n:p}$ is the proportion of the stream n accounted for by isotope i, $\rho_{m,k=0}$ is the source material atomic density at the beginning of the zeroth iteration before any discrete stream actions have been applied to any materials, N_i is the current atomic density of isotope i, and Δt is the total time of the current burnup step. In the case of a desired 100% removal of an isotope ADER approximates the term $(s_{n:p} f_{i,n:p} \rho_{m,k=0} N_i^{-1})$ as $(1 - 10^{-8})$.

$$\lambda_{i,n:g} = ln(1 - s_{n:p} f_{i,n:p} \rho_{m,k=0} N_i^{-1})(\Delta t)^{-1} \tag{5.3}$$

For a proportional stream adding a quantity of mass to a material the isotopic proportional constants are calculated, before the burnup step and *after* the application of any discrete stream effects (such that the proportional constants are calculated from the updated material composition values), as seen below in equation 5.4.

$$\lambda_{i,n:g} = ln(1 + s_{n:p} f_{i,n:p} \rho_{m,k=0} N_i^{-1})(\Delta t)^{-1} \tag{5.4}$$

During the construction of the burnup matrix, discussed in section 11.7 these proportional constants are added to the respective decay constants of the isotopes they seek to modify.

As an example of a proportional group-class stream consider the stream below formed using the `gUF4` group and using material `FuelSalt` as a source, having no sink.

```
stream from FuelSalt group gUF4 form prop type remv
```

5.2 Table-class streams

Table-class streams, unlike group class streams, are directions from the user to ADER to move a specific quantity of mass in a specific manner. A stream is classified as a table-class stream when the keyword `rem` is used in the stream definition. This keyword is always followed by the name of a `removal` table. **The use of the word "removal" for the name of this structure is an artifact from earlier code development. These tables can be used to move mass into, out of, and between SERPENT2 materials. The materials listed as their `to` and `from` entries, sink and source respectively, operate as sink and source respectively. The name of this structure `removal` has no relation to the English word "removal" other than spelling.** Removal tables are defined elsewhere in the code using the format below where elements are entered using their alphabetic periodic table designation, so `U` for uranium, while isotopes are entered as elements followed by `-A` where A is the atomic number of the isotope; `U-235` for ^{235}U.

```
removal [Name]
[element_or_isotope_1] [table_value]
...
<element_or_isotope_n> (table_value)
```

Both elements and isotopes may be entered in the same removal table. If a `table_value` ,q, is entered for both an element and one or more of its constituent isotopes, the isotopes for which a `table_value` was entered will retain their corresponding value rather than be overwritten by their parent element's value - i.e. an elemental `table_value` is a default for all child isotopes of that parent element but this default is overwritten by any entry for a child isotope. All q values must be greater than or equal to zero.

As an example, consider the removal table seen below. In this removal table, `rem_table`, all isotopes, i, of uranium have an associated q_i value of 0.1, except for ^{233}U which has a q_i value of zero.

```
removal rem_table
U       0.1
U−233   0.0
```

A single removal table may be used to define multiple streams, each of which will implement the removal table in its own designated fashion.

All table-class streams require the use of the keyword `frac` and the input of its associated value which must be equal to or greater than zero. As such the input template for table-class streams appears as such. . .

```
stream <to> (name_of_sink_material) <from> (name_of_source_material)
    [type] [{feed; reac; redox; remv}] [rem] [name]
    [form] [{cont; disc; prop}] [frac] [frac_value]
```

In the following subsections many equations and parameters will depend on a material value such as density or volume. All material values in reference to table-class streams are taken at the source material if there is one. If there is no source material these material values are taken at the sink material.

5.2.1 Continuous table-class streams

Continuous table-class streams move $s_{n:c}$ amount of material every burnup step where $s_{n:c}$ is defined by equation 5.5 where $q_{i,n:c}$ is the `table_value` given for isotope i by the removal table used to create stream n, N_i is the number density for that isotope (taken after the effects of discrete streams have been applied), r_n is the `frac_value` given for stream n, and V_m is the material volume.

$$s_{n:c} = V_m \sum_{i}^{I} q_{i,n:c} r_n N_i \tag{5.5}$$

From equation 5.5 it is clear that the product of a value from a removal table and the value given for the stream's `frac` entry is the percentage, by atomic density, of that removal table entry to move during each burnup step. Continuous table-class streams move this mass evenly over time. The transfer rate, $h_i \frac{\texttt{atoms}}{s^{-1}}$ for isotope i is given by equation 5.6 where Δt is the length of the current burnup step in seconds.

$$h_i = V_m q_{i,n:c} r_n N_i \Delta t^{-1} \tag{5.6}$$

It is important to remember that isotopes derive their $q_{i,n}$ values from their parent element's q_e value given in the removal table used to define stream n unless the isotope in question was given its own q_i value in the same removal table.

As an example of a continuous table-class stream consider the following...

```
stream from FuelSalt rem rem_table type redox form cont frac 1000
```

5.2.2 Discrete table-class streams

Discrete table-class streams move $s_{n:d}$ amount of material every burnup step where $s_{n:d}$ is defined by equation 5.7 where $q_{i,n:d}$ is the `table_value` given for isotope i by the removal table used to create stream n, N_i is the number density for that isotope (taken before the effects of discrete streams have been applied), and r_n is the `frac_value` given for stream n.

$$s_{n:d} = V_m \sum_{i}^{I} q_{i,n:d} r_n N_i \tag{5.7}$$

From equation 5.5 it is clear that the product of a value from a removal table and the value given for the stream's `frac` entry is the percentage of that removal table entry to move during each burnup step. Discrete table-class streams move this mass instantaneously at the beginning of each burnup step. The transfer amount, h_i for isotope i is given by equation 5.8 where Δt is the length of the current burn up step in seconds.

$$h_i = V_m q_{i,n:c} r_n N_i \Delta t^{-1} \tag{5.8}$$

It is important to remember that isotopes derive their $q_{i,n}$ values from their parent element's q_e value given in the removal table used to define stream n unless the isotope in question was given its own q_i value in the same removal table. With regards to discrete stream behavior throughout ADER criticality iterations, the effects of discrete table-class streams are applied on the zeroth iteration for each burnup step, and no more than that.

As an example of a discrete table-class stream consider the following...

```
stream to FuelSalt rem rem_table type feed form disc frac 0.234
```

5.2.3 Proportional table-class streams

Proportional table-class streams *do not* move a specific quantity of material in a burnup step. Rather, proportional table-class streams modify the decay constant of the isotopes in question - this modification may be slight or may even be of a magnitude to turn a decay constant into a production constant. The stream-constants, $\lambda_{i,n:t} \frac{1}{s^{-1}}$ for isotope i as modified by table-class stream n are determined according to equation 5.9.

$$\lambda_{i,n:t} = q_{i,n:p} r_n \tag{5.9}$$

Concerning the inclusion of $\lambda_{i,n:t}$ into the burnup matrix, while this is covered in detail in section 11.7, it is sufficient to say for now that $\lambda_{i,n:t}$ is added to the isotope's decay constant if there is no source material, this isotope being in the sink material. If there is a source material $\lambda_{i,n:t}$ is subtracted from the source isotope's decay constant and this production term is added to the isotope in the sink material such that the transfer rate is dependent on the source material isotopic concentration.

As an example of a proportional table-class stream consider the following...

```
stream from FuelSalt rem rem_table type remv form prop frac 1.0
```

5.3 Quick Reference

- Streams are defined as follows…

```
stream <to> (name_of_sink_material) <from> (name_of_source_material)
    [type] [{feed; reac; redox; remv}] [{group; rem}] [name]
    [form] [{cont; disc; prop}] (frac) (frac_value)
```

- Removal tables are defined as follows…

```
removal [Name]
[element_or_isotope_1] [table_value]
...
<element_or_isotope_n> (table_value)
```

- Streams require a source or a sink. They may have both, but both are not required.
- Streams with elements lacking an isotopic composition derive the element's isotopic composition from the source material.
- Streams with elements lacking an isotopic composition AND with a valid SERPENT2 material as a sink MUST have a valid SERPENT2 material as a source.
- Material clusters are collections of materials which are connected by ADER streams
- Group-class streams change their delivered mass each burnup step according to the optimization solution.
- Table-class streams move user-defined quantities of mass each burnup step
- Proportional group-class streams may not be formed using summation groups.

6 Criticality Control

In many nuclear burnup simulations the overall neutron multiplication factor of the system in question has some desired value or range it should hold. ADER provides the user the ability to set overall system k_{eff}^{analog} targets, a minimum and a maximum value, set using the below input. The default values for `kmin` and `kmax` are zero and ∞ respectively.

```
kmin [minimum_k_value]
kmax [maximum_k_value]
```

For instance, the below input would place a desired lower bound on k_{eff}^{analog} of 1.0 and an upper bound of 1.001.

```
kmin 1.0
kmax 1.001
```

As discussed in section 11, ADER only considers the neutron multiplication factor of a single SERPENT2 material. This material must be designated by the user using the keyword and value pair, `rhow 1.0`, in the material's definition line before the list of its constituent isotopes such as seen below…

```
mat FuelSalt −2.805 vol 1 burn 0 rhow 1.0 ader
```

Also discussed in section 11 is the "probability of absorption" factor used as an approximation of the effects of the rest of the system outside of the singular material considered for criticality control. While a detailed discussion of this approach is left to section 11 for now it will be said that this method of assessing criticality in systems is expected to produce reasonable and well-behaved results for systems in which there is a single material which is both largely responsible for nuclear criticality and weakly coupled neutronically to its surroundings.

ADER has a limited iteration scheme for criticality search. This scheme is visually depicted as a flowchart in figure **??**. At the beginning of each burnup step ADER builds and solves the optimization problem for all material clusters. Following this the determined actions for discrete type streams, both group-class and table-class, are applied to change material compositions. Following this a new Monte Carlo transport sweep is executed with all the same population and cycle parameters as the transport sweep for a burnup step. Following this transport sweep if the system k_{eff}^{analog} is within the user set bounds, or no iterations remain, the simulation proceeds to the calculation of the nuclear burnup solution. If the k_{eff}^{analog} is out of bounds *and* iterations remain in the criticality search, ADER will build and solve the optimization problem for all material clusters, again. Following this the actions of discrete group-class streams only, discrete table-class streams are only applied on the very initial iteration of the criticality search, are again applied to materials to change their compositions. On every iteration of the criticality search after the initial iteration only the actions of discrete group-class streams are applied to the material compositions. The actions of discrete table-class streams are only applied on the initial iteration of a criticality search. If no criticality search is being done, if the system neutron multiplication factor has not been restricted, after the Monte Carlo transport sweep following the initial application of both discrete group and table-class streams the simulation calculation proceeds on to the burnup solution. To set the maximum number of criticality search iterations the input below is used, the default value of this parameter is 5. The criticality search features of ADER are only activated by the `kmax` and `kmin` keywords - i.e. a default value of 5 will not force a simulation through 5 criticality searches if no criticality search was requested.

```
set ader_trans_iter [value]
```

An important consideration of the interaction of streams with multiple forms, `cont`, `disc`, `prop`, is that if streams of multiple forms are used which significantly affect system reactivity ADER's criticality search will only be aware of the effects due to discrete streams. The effects on criticality from continuous and proportional streams are not assessed until the burnup calculation has been completed. For example if ADER has determined an amount of ^{233}U to inject into a material to keep it critical following a discrete injection of natural Li for chemistry control, and this uranium is from a continuous stream, the Monte Carlo transport sweep run before the burnup calculation will only see the negative reactivity effects of the discrete and instantaneous lithium injection. The positive reactivity effects of the uranium will be observed in the Monte Carlo transport sweep for the *next* burnup step.

6.1 Quick Reference

- The maximum system k_{eff}^{analog} is set by...

  ```
  kmax [value]
  ```

- The minimum system k_{eff}^{analog} is set by...

  ```
  kmin [value]
  ```

- The system defaults for `kmin` and `kmax` are zero and ∞ respectively. To use the system k_{eff}^{analog} as a constraint designate one and only one material with the keyword-value pair `rhow 1.0` in the material's definition and set appropriate `kmax` and `kmin` targets

- The maximum number of criticality search iterations is set by...

  ```
  set ader_trans_iter [value]
  ```

- The criticality search feature of ADER is only aware of the reactivity impacts of discrete form streams, both group and table-class.

- When a criticality search is activated, the reactivity impacts *of all streams* are assessed as part of the constraints regarding the material designated by the keyword-value pair `rhow 1.0`.

7 Output

ADER output is located in the "[`input_file_name`]_dep.m" output file which SERPENT2 produces. Every material under ADER control with groups assigned via a conditions block or streams, will have output in this file.

7.1 Groups

The fraction of a material's atomic density accounted for by each group is printed out in a vector with the name of "`MAT_m_GRP_g_FRAC`" where `m` is the material name and `g` is the group name for a group which is a member of a material's options list - see section 3 for a description of this list. The value at index j of such a vector corresponds to the value at the end of burnup step j. The values displayed are normalized to the material's atomic density at the beginning of the burnup step.

For example, consider if the group `gFLiBe` accounted for 60% of the material `FuelSalt`'s atomic density during each of two burnup steps. The corresponding output line in the "_dep.m" file would appear as below...

```
MAT_FuelSalt_GRP_gFLiBe_FRAC = [
4.00000E-01, 4.00000E-01 ];
```

Summation groups and their component groups are all reported, in no particular order in the output.

7.2 Streams

The mass moved by a stream on each burnup step j is given in vector form where the vector is named "`MAT_m_STREAM_s_STEP_AMT`" where `m` is the material name and `s` is the name of the group or removal table used to define the group-class or table-class stream in question, respectively. Results for streams with both a valid sink and source are reported by each the sink and the source. Each vector index j corresponds to burnup step j. The units for stream output are $\frac{\texttt{atoms}}{barn \cdot cm}$ where the value represents the stream's impact on a per unit volume basis with the normalizing volume being the material volume for which the stream is reported. This means that if a given stream moves mass between materials of differing volumes, each material will report a different value for that stream for each burnup step because their volumes are different - all thanks to the conservation of mass. In the same context of `atoms` from section 3, `atoms` in this sense refers to a fractional mix of isotopes corresponding to the recipe of the stream in question. For group-class streams these "atomic" units will have the proportions that the group recipe does - this is not necessarily the case for proportional group-class streams which have a bevy of cautions regarding their use presented in section 5.1.3. For table-class streams these "atomic" units take the proportional make up of the table used to define the stream except in the case of proportional table-class streams in which case the "atoms" reported is just that, total atoms. The exact composition of such a stream is not given. Proportional type streams do report a true value, in the sense that the number of atoms is correct, in that the value accounts for nuclear depletion effects. This value is calculated inside the burnup calculation as what is sometimes referred to as a "ghost" nuclide - a false nuclide who's only impact on the solution is to track and measure some parameter of interest, in this case the true amount of mass moved by a proportional stream, group-class or table-class.

As an example, consider if stream `gUF4` had moved $0.001\frac{\texttt{atoms}}{barn \cdot cm}$ per unit volume into material `FuelSalt` on each of two burnup steps - so $0.0002\frac{\texttt{atoms}}{barn \cdot cm}$ of uranium and $0.0008\frac{\texttt{atoms}}{barn \cdot cm}$ of fluorine. The corresponding output line in the "_dep.m" file would appear as below...

```
MAT_FuelSalt_STREAM_gUF4_STEP_AMT = [
1.000000E-02, 1.00000E-02 ];
```

Summation streams and their component streams are all reported in the output in no particular order.

7.3 Quick Reference

- Group fractions of a material's atomic density, normalized to the material's atomic density at the beginning of the burnup step, are reported in a vector with the name of "`MAT_m_GRP_g_FRAC`" where `m` is the material name and `g` is the group name.

- Stream mass transfers are reported on a per unit volume basis of the associated material and are reported in the vector named "`MAT_m_STREAM_s_STEP_AMT`" where `m` is the source or sink material and `s` is the name of the group or removal table used to define the stream. The units are $\frac{atoms}{barn \cdot cm}$.

7.4 Developers Output

In the advanced compilation options for ADER, `ADER_DIAG` and `ADER_INT_TEST`, numerous additional outputs are produced. Any person who is making use of these outputs is expected to be familiar with the documentation of ADER as well as the source code. As such, only a brief summary of the additional files is given for each option. A more thorough understanding may be had by inspection of the functions which produce these outputs.

7.4.1 ADER_INT_TEST

Compiling SERPENT2 and ADER with the "`-DADER_INT_TEST`" flag will cause the following outputs to be produced. In the file name templates given `m` stands for a material name and `n` is a material cluster number.

- "ADER_Clp_Model_Material_`m`_Conformity.test" - contains an element by element comparison of the simplex model in ADER memory with the simplex model in CLP memory.
- "ADER_Cluster_Composition_Matrices.json" - contains all the information needed to construct the simplex problem in a json format.
- "Cluster_`n`_Material_Composition_Matrix.csv" - contains every entry of the simplex matrix for material cluster `n`.

7.4.2 ADER_DIAG

Compiling SERPENT2 and ADER with the "`-DADER_DIAG`" flag will cause the following outputs to be produced. In the file name templates given `m` stands for a material name, `s` stands for the numeric index of a given burnup step, `ss` stands for the numeric index of the sub-step for burnup step `s`, `n` is a material cluster number, and `k` is the index of the criticality search iteration.

- "Cluster_Parent_`m`_Burn_Matrix_Step_`s`_Sub_Step_`ss`.csv" - contains a copy of the burnup matrix for the material cluster who's parent is material `m` for the given burnup step and sub-step.
- "Cluster_Parent_`m`_Pred_Step_`s`_Sub_Step_`ss`_Matrix_Comparison.test" - contains a comparison of every compatible entry in the burnup matrix as generated by SERPENT2 and ADER for the prediction step of a predictor-corrector burnup simulation.
- "Cluster_Parent_`m`_Corr_Step_`s`_Corr_Sub_Step_`ss`_Matrix_Comparison.test" - contains a comparison of every compatible entry in the burnup matrix as generated by SERPENT2 and ADER for the corrector step of a predictor-corrector burnup simulation.
- "ADER_Memory_Lists.test" - contains `WDB` address information for a variety of data.
- "`m`_XS_End_of_Step_`s`.txt" - contains a comparison of the isotopic cross sections (absorption and fission) between SERPENT2 and ADER after all criticality search iterations for a given burnup step.
- "`m`_XS_step_`s`_iter_`k`.txt" - contains a comparison of the isotopic cross sections (absorption and fission) between SERPENT2 and ADER for the given burnup step `s` and inner criticality iteration `k`.

8 Testing

ADER comes complete with a unit test suite and a suite of system tests.

8.1 Unit Tests

To run ADER's unit tests, compile the code with the "-DADER_TEST" flag and run, on a single thread, the input file "test_input.txt" found in the "inputs/Test_Input" directory with the run option "-test". The results of the unit tests will be found in the executing directory in a file titled "TestResults.test".

8.2 System Testing

To run any one of ADER's system tests compile the code anyway except with the "-DADER_TEST" flag. Then, with any number of threads, run any of the system test files found in any of the system test directories found in the "System_Testing" directory. Compare the results with the desired results described in the README files.

9 Parallel Computation

ADER has been developed for threaded operation with the OpenMP interface just as SERPENT2 has. To run ADER with more than one thread do the same thing SERPENT2 asks, add the input seen below to the command line executing the SERPENT2 run. This will enable threaded computation for SERPENT2 as a whole as well, and this input does not need to be duplicated for both SERPENT2 and ADER to run in a threaded manner - only one instance of this command line option should be used.

```
-omp [num_threads]
```

ADER is not compatible with distributed memory computing - i.e. while SERPENT2 can do MPI runs, SERPENT2 with ADER can not. The code as a whole may be compiled in a way to run with MPI but only non-ADER simulations can be run with MPI.

10 General Notes

A few closing notes, restrictions, hints, and tips that didn't fit elsewhere in the manual...

- Any SERPENT2 material with either a conditions block or one or more streams, must have the keyword `ader` appear in the material's definition before the listing of the material's constituent isotopes.
- If the user would like ADER to issue a warning every time the burnup solution produces a negative density for an isotope, which is then corrected to zero, place the following line in the simulation input. The value is arbitrary.

  ```
  set ader_neg_adens [value]
  ```

- ADER is not compatible with divided SERPENT2 materials. These may be used in the same simulation but divided materials, or their children, may not also be materials under ADER control.
- ADER is not compatible with SERPENT2's "`mflow`" structure. Materials affected by `mflow` structures may be used in an ADER simulation but they may not in any way (other than neutronically) be connected with ADER or ADER materials.
- SERPENT2 employs a function, `MaterialBurnup`, which calculates the power produced by a material during a burnup step by the difference in abundances of isotopes in that material between the beginning of the burnup step and the end of the burnup step. This methodology assumes there are no mass flows in the system and has not been configured to incorporate ADER mass flows. AS SUCH - MATERIAL POWER ESTIMATES PROVIDED BY SERPENT2 WHEN RUNNING AN ADER SIMULATION WITH AT LEAST ONE CONTINUOUS OR PROPORTIONAL STREAM ARE LIKELY INCORRECT. The user can fix these power estimates themselves by incorporating the mass moved by streams and the power produced from any fissions on these masses.

11 Theory

The details of ADER's computations and the theory behind its operation are laid out in this section.

11.1 Groups

Take $F_{e|u}$ to be the input values given for each element, e, in a group, u. Take $F_{i|e,u}$ to be the input value given for each isotope, i, as part of element e in group u. The normalized equivalents, $f_{e,u}$ and $f_{i|e,k}$, are defined by equations 11.1 and 11.2 respectively.

$$f_{e,u} = \frac{F_{e,u}}{\sum_{e}^{E} F_{e,u}} \tag{11.1}$$

$$f_{i|e,u} = \frac{F_{i|e,u}}{\sum_{i|e}^{I|e} F_{i|e,u}} \tag{11.2}$$

Concerning the elements in groups which are not given an explicit isotopic composition, referred to as "unfixed" elements - calculations in SERPENT2 are ultimately done on an isotopic basis and so all elements must be composed of isotopes. Groups which are a part of a material's options list, see section 3 for an explanation of this term, derive the isotopic compositions for their unfixed elements using the isotopic composition of the same element in the host material. These $f_{i:e,u}$ values are derived as seen in equation 11.3 where $N_{i:e}$ is the number density of isotope i of element e in the material of interest.

$$f_{i|e,u} = \frac{N_{i|e}}{\sum_{i|e}^{I|e} N_{i|e}} \tag{11.3}$$

Take $f_{i,u}$ to be the normalized fraction of isotope i in group k. These values are derived as seen in equation 11.4.

$$f_{i,u} = f_{e,u} f_{i|e,u} \tag{11.4}$$

Take g_u to be the normalized fraction of a material's atomic density which is taken to ascribe to the recipe for group u - that is, the isotopes composing this subset of the material density all have a normalized abundance of $f_{i,u}$ within the subset. For instance, consider the atomic density and makeup of the material `FuelSalt` from section 2 and consider the group `gFLiBe` from section 3. The g_u value for `gFLiBe` in the material `FuelSalt` could be said to be anywhere from zero to 0.76.

Take E_j to be the number density of element j in a material. Take I_k to be the atomic density of isotope i in a material. Take μ_j to be the portion of element j's number density not attributed to group structures. Take μ_k to be the portion of isotope k's atomic density which is not attributed to group structures. In the case of a "controlled" element or isotope, a concept covered in section 4.3, the μ value would be zero. The relationships then seen in equations 11.5 and 11.6 may be established.

$$E_j = \mu_j + \sum_{u}^{U} g_u f_{e,u} \tag{11.5}$$

$$I_k = \mu_k + \sum_{u}^{U} g_u f_{i,u} \tag{11.6}$$

In section 4.2 the concept of a relative abundance constraint between two groups was developed. Equation 4.4 can be expressed linearly as two inequalities as shown in equations 11.7 and 11.8 where r_m is the input `min_value` and r_M is the input `max_value`.

$$-\infty \leq -g_1 + r_m g_2 \leq 0 \tag{11.7}$$

$$0 \leq -g_1 + r_M g_2 \leq \infty \tag{11.8}$$

Summation groups, those groups defined according to method D in section 3, have their g_k values - denoted $g_{k:Y}$ - determined according to equation 11.9 where $g_{k|Y}$ is the g_k value for a group used in the definition of summation group Y and $w_{k|Y}$ is the weight value entered for group k used in the definition of summation group Y. In section 3 this value is shown as a mandatory input of "`1`" though in reality it may take any value greater than 0 - but any value other than 1 may lead to a non-physical solution in the optimization problem.

$$g_{k:Y} = \sum_{k|Y}^{K|Y} g_{k|Y} w_{k|Y} \tag{11.9}$$

11.2 Group-class Streams

For group-class streams take s_v to be the atomic density of this mass load where an "atomic" unit is composed of isotopes in proportion to the $f_{k,u}$ values of the group, which are denoted $f_{k,v}$ for groups which are used to form streams.

Take $E_{j,\Delta}$ to be the number density of element j in a given stream's mass load. Take $I_{k,\Delta}$ to be the atomic density of isotope i in a given stream's mass load. The relationships then seen in equations 11.10 and 11.11 may be established.

$$E_{j,\Delta} = s_v f_{e,v} \tag{11.10}$$

$$I_{k,\Delta} = s_v f_{i,v} \tag{11.11}$$

Concerning the elements in streams which are not given an explicit isotopic composition, referred to as "unfixed" elements - calculations in SERPENT2 are ultimately done on an isotopic basis and so all elements must be composed of isotopes. Streams derive the isotopic compositions for their unfixed elements using the isotopic composition of the same element in the source material. These $f_{i|e,v}$ values are derived as seen in equation 11.12 where $N_{i|e}$ is the number density of isotope i of element e in the source material of interest. Streams with unfixed elements and no SERPENT2 material as a source are not permitted.

$$f_{i|e,v} = \frac{N_{i|e}}{\sum_{i|e}^{I|e} N_{i|e}} \tag{11.12}$$

Summation streams, group-class streams constructed with summation groups have their s_v values - denoted $s_{v:Y}$ - determined according to equation 11.13 where $s_{v|Y}$ is the s_v value for a stream used in the definition of summation stream Y and $q_{v|Y}$ is the weight value entered for group v used in the definition of summation stream Y. In section 3 this value is shown as a mandatory input of "`1`" though in reality it may take any value greater than 0 - but any value other than 1 may lead to a non-physical solution in the optimization problem.

$$s_{v:Y} = \sum_{v|Y}^{V|Y} s_{v|Y} q_{v|Y} \tag{11.13}$$

11.3 Table-class Streams

The change in an isotope's, i, atomic density, $z_{i,v}$, induced by a continuous or discrete table-class stream, v is given by equation 11.14 where $q_{i,v}$ is the `table_value` given for isotope i by the removal table used to create stream v, N_i is the number density for that isotope, and r_v is the `frac_value` given for stream v.

$$z_{i,v} = q_{i,v} r_v N_i \tag{11.14}$$

The change in an isotope's, i, atomic density, $z_{i,v}$, induced by a proportional table-class stream, v is given by equation 11.15 where $q_{i,v}$ is the `table_value` given for isotope i by the removal table used to create stream v, N_i is the number density for that isotope, and r_v is the `frac_value` given for stream v. If the proportional stream in question, v, has only a single valid SERPENT2 material as a sink, the sign of $q_{i,v}$ is positive. Otherwise, the sign of $q_{i,v}$ is negative.

$$z_{i,v} = N_i e^{q_{i,v} r_v \Delta t} \tag{11.15}$$

The total change in an isotope's, i, atomic density due to all tale-class streams is denoted r_i is given by equation 11.16. It should be noted, that when this summation involves values for a proportional stream the r_i value becomes an approximation as the proportional stream will move more or less mass based on nuclear depletion and the action of other streams.

$$r_i = \sum_{v}^{V} z_{i,v} \tag{11.16}$$

There are of course $z_{e,v}$ and r_e equivalents for elements as well.

11.4 Criticality Control

A weighted sum over isotopes in a material forms the reactivity constraint that may be applied. This constraint is derived from the expression for the multiplication factor as found in Equation 11.17:

$$k_{eff} = P_{NL} \frac{\sum_{m}^{M} \phi_m \omega_m \nu \Sigma_f^m}{\sum_{m}^{M} \phi_m \omega_m \Sigma_a^m} \tag{11.17}$$

where ϕ_m, ω_m, $\nu\Sigma_f^m$ and Σ_a^m are, respectively, the scalar neutron flux, the volume fraction, the spectrum averaged neutron production cross section, and the macroscopic absorption cross section for each material m. P_{NL} is the neutron non-leakage probability. In ADER, the ability to control k_{eff} is limited to the case of a single neutron multiplying material; take $\nu\Sigma_f = 0$ for every material but the multiplying material M and as such Equation 11.17 can be rewritten as follows:

$$k_{eff} = P_{NL} \frac{\phi_M \omega_M \Sigma_a^M}{\sum_{m}^{M} \phi_m \omega_m \Sigma_a^m} \frac{\nu \Sigma_f^M}{\Sigma_a^M} \tag{11.18}$$

The probability of a neutron being absorbed in the multiplying material is defined as follows:

$$P_A = P_{NL} \frac{\phi_M \omega_M \Sigma_a^M}{\sum_{m}^{M-1} \phi_m \omega_m \Sigma_a^m} \tag{11.19}$$

Then k_{eff} can be calculated as:

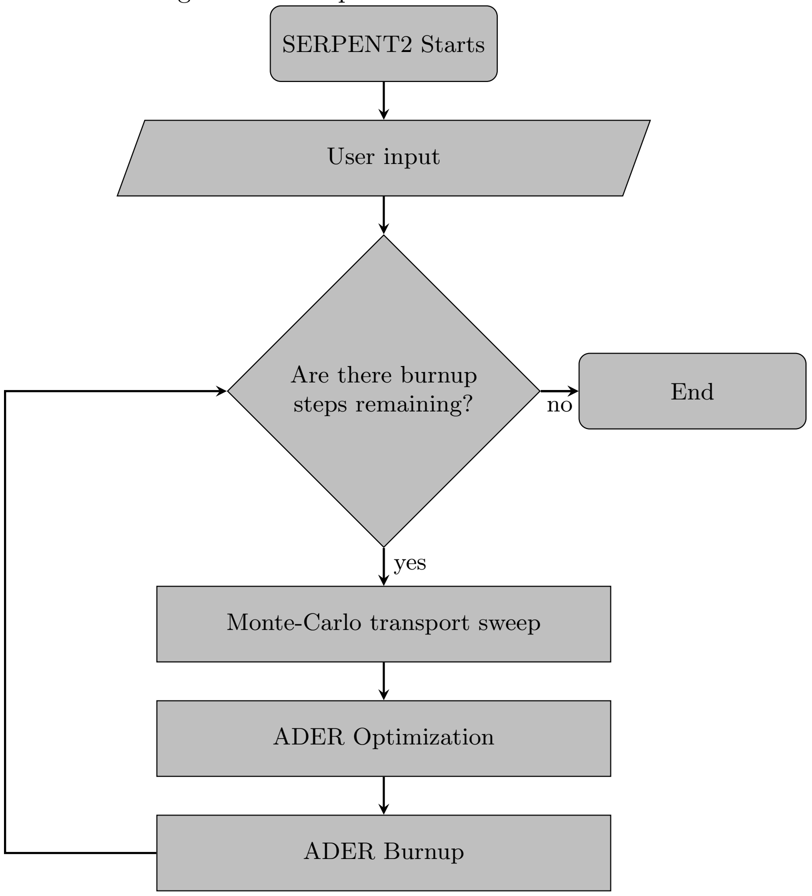
$$k_{eff} = P_A \frac{\nu \Sigma_f^M}{\Sigma_a^M} = P_A \frac{\sum_{i}^{I} \nu \Sigma_f^i}{\sum_{i}^{I} \Sigma_a^i} \tag{11.20}$$

where $\nu\Sigma_f^i$, and Σ_a^i are, respectively, the spectrum averaged neutron production cross section, and the absorption cross section for every isotope i in the multiplying material M. This relation is expected to hold for simulations in which there is a dominant reactive material and for which $\nu\Sigma_f \approx 0$ for all other materials.

In this case, given lower and upper bounds for the multiplication factor of the system, k_{eff}^{min} and k_{eff}^{max} respectively, Equation 11.20 can be made linear as in Equations 11.21 and 11.22.

$$0 \geq \frac{k_{eff}^{min}}{P_A} \sum_{k}^{K} \sigma_a^k I_k - \sum_{k}^{K} \nu^k \sigma_f^k I_k \tag{11.21}$$

Figure 1: A simplified schematic of interactions between SERPENT2 and ADER.



$$0 \leq \frac{k_{eff}^{max}}{P_A} \sum_{k}^{K} \sigma_a^k I_k - \sum_{k}^{K} \nu^k \sigma_f^k I_k \tag{11.22}$$

A key assumption of this linearization process is that $\frac{\partial P_A(m \ldots M)}{\partial M} = 0$ when in truth P_A is a function of the composition of material M. The impacts of this approximation are expected to be quite small but it will affect all simulations, more so those with strong leakage effects.

11.5 Constructing the Optimization Problem

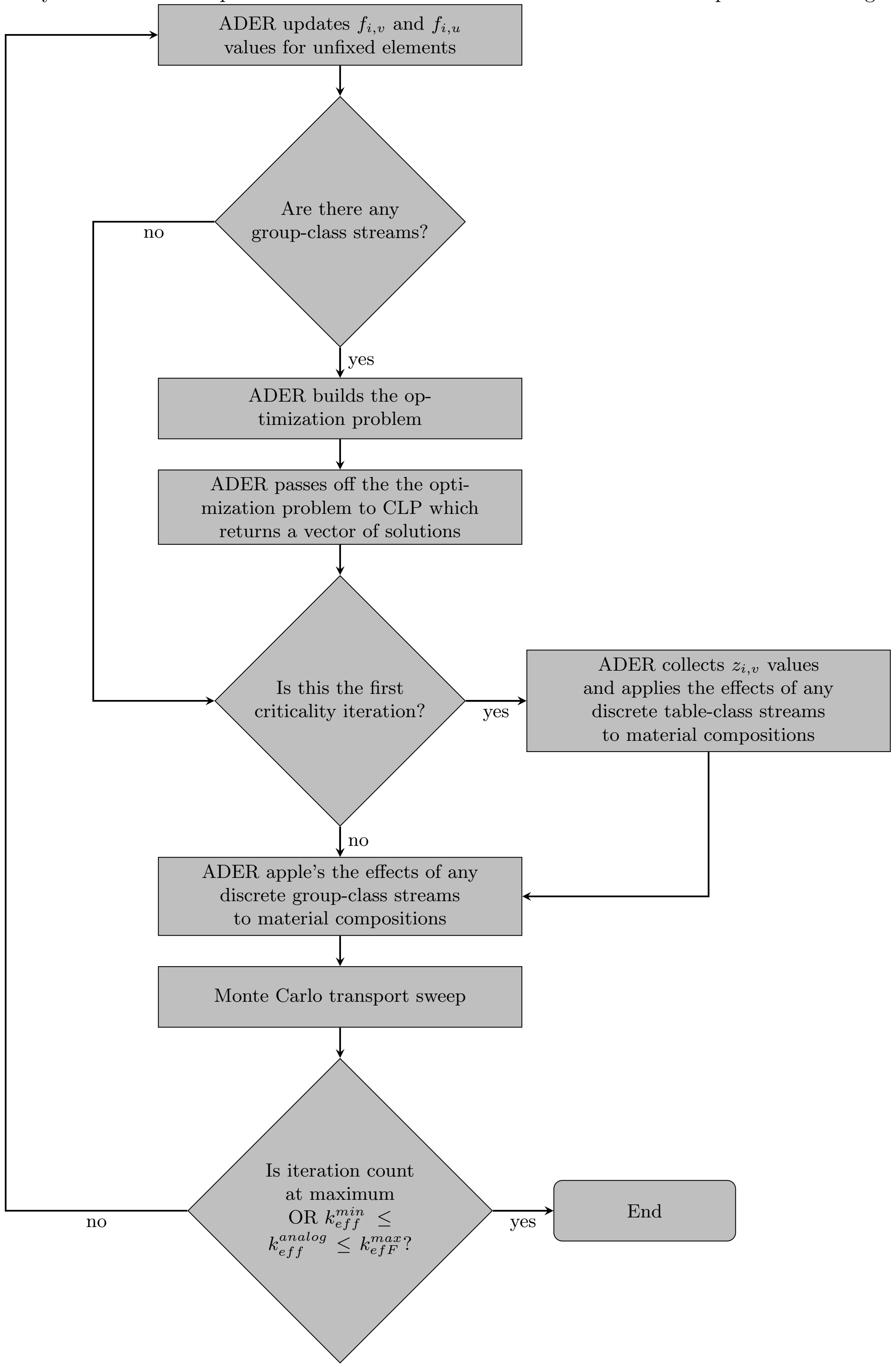
A discussion of the construction of the optimization problem would not be complete without a discussion of ADER's interactions with SERPENT2 and a view of the overall program flow. Figure 11.5 provides a simplified view of ADER's interactions with SERPENT and the overall simulation progression. In figure 11.5 the "ADER Optimization" box from figure 11.5 is expanded as a process into its own flowchart.

In a SERPENT2 burnup simulation, following the initial Monte Carlo transport sweep done at the beginning of every burnup step, ADER enters its criticality search iterations - even if a criticality search has not been asked for. The default values for `kmin` and `kmax` will cause the criticality check to pass regardless.

The first consequential action ADER takes in each iteration is to determine the $f_{i|e,u}$ and $f_{i|e,v}$ values for the isotopes of unfixed elements in groups and group-class streams. Following this assessment all of the $z_{i,v}$ values are calculated for table-class streams.

This point marks the first major divergence of any ADER simulation. *If and only if* the only streams entered by the user are table-class streams the simulation proceeds to the application of the effects of discrete form streams. There is no optimization process in this case because the user has only given direct orders to

Figure 2: A detailed look of the “ADER Optimization” box from figure 11.5. This diagram picks up at the entry to the “ADER Optimization” box and exits out to the “ADER Burnup” box also in figure 11.5.



ADER updates $f_{i,v}$ and $f_{i,u}$ values for unfixed elements
Are there any group-class streams?
no
yes
ADER builds the optimization problem
ADER passes off the the optimization problem to CLP which returns a vector of solutions
Is this the first criticality iteration?
yes
ADER collects $z_{i,v}$ values and applies the effects of any discrete table-class streams to material compositions
no
ADER apple’s the effects of any discrete group-class streams to material compositions
Monte Carlo transport sweep
Is iteration count at maximum OR $k_{eff}^{min} \leq k_{eff}^{analog} \leq k_{efF}^{max}$?
no
yes
End

ADER in the form of table-class streams, there are no choices for ADER to make in the form of group-class streams.

In the case that at least one group-class stream, attached to a material, exists in the simulation, the simulation would then proceed to build and solve the optimization problem for each material cluster.

The CLP library expects a linear programming matrix from ADER. Figure 11.5 depicts the scheme for constructing the linear programming matrix. Column bounds are presented above the appropriate column whereas row bounds are presented to the left of the appropriate row. Below the column bounds are the variables which the columns represent and to the right of the row bounds are the equation number, if any, of the equation that row is modeled after. For the sake of brevity the matrix in Figure 11.5 is for one material only though many materials may be involved in such a matrix should they be linked together by shared mass transfers. In which case the only variables shared between materials in the same material cluster are the group-class streams and the stream equations they are a part of are the only coupling equations; aside from transfers by table-class streams but those are only represented in the linear programming matrix, they are handled by other routines all together. If a second material were to be included in this matrix then, perhaps, the stream entries in the fourth, fifth, and sixth columns would have non-zero coefficients for some E^d_j and I^d_k rows of the second material. The coefficients used for the construction of these matrices are normalized to the atomic density from the beginning of the optimization process for the host material.

Working down the matrix row by row the first row encountered represents equation 11.7 with arbitrary groups g_1 and g_2 whereas the next row down represents equation 11.8. The third row, what will be referred to as an elemental future row represents the atom balance for element j where fe^u_j is the fractional proportion of element j in group u. The novel column involved here is an elemental future column whose inclusion in the same row closes the equation where an “elemental future value”, E^f_j, is the fraction of a material's atomic density (relative to the density at the beginning of the step) that is taken up by element e_j. The bounds for this row are those for a free element, a concept covered in section 4.3, those elements which are permitted to have portions of the element not tied up in declared group structures. In the case of a controlled element, those who's complete abundance must be accounted for by group structures, the lower bound is changed to zero. The fourth row, an elemental delta row, represents the change in the abundance of element j, E^d_j, as caused by all group-class streams where fe^v_j is the fractional proportion of element j in stream v. Of course the elemental delta column is involved to close the balance. The fifth row, or balance row, is what ties together E^f_j and E^d_j. The bounds, α and β, are equal and represent $E^c_j + r_{e_j}$ constituting an element balance “in time where E^c_j represents the present fractional abundance of element j. The fifth row requires, straightforwardly, that the future amount of an element be equal to the current amount plus any delta, or change, in the element's abundance. The sixth row is an isotopic balance row requiring that the abundance of an element be equal to the abundance of its constituent isotopes. The following three rows, the seventh, eighth, and ninth, are the isotopic versions of the elemental future, delta, and balance rows where f values are for the isotopic fractional proportions. γ and δ are equal and represent $I^c_k + r_i$ where I^c_k represents the present fractional abundance of isotope k. In the tenth and eleventh rows Equations 11.22 and 11.21 find representation with η and θ respectively representing terms of the expanded sum found in the referenced equations; $\frac{k^{min}_{eff}}{P_A}\sigma^k_a - \nu^k\sigma^k_f$ and $\frac{k^{max}_{eff}}{P_A}\sigma^k_a - \nu^k\sigma^k_f$. The twelfth row represents equation 4.6, accounting for the contributions of the future quantity of an element to a material's averaged oxidation state (or however the weighted sum is interpreted). The thirteenth row, or Pres row, exists when the user instructs ADER to balance inflows with outflows as discussed in section 4.5. The pres row requires that the net stream transfers in a material come to zero. The effects of table-class streams are captured in υ and ω as seen in equation 11.23. The fourteenth row represents a closure for a group defined with method D from section 3; a summation group. In this case the summation group is g_3 which was defined as follows...

```
grp g3 sum
g1  w1
g2  w2
```

The fifteenth row represents a closure relationship for a summation stream, a group-class stream built using a summation group. In this case the summation stream is s_3. The final row is the optimization, or Opt row. This row indicates to the simplex routine which variables to minimize or maximize. In figure 11.5 the opt row is indicating that g_1 is the optimization target.

$$\upsilon = \omega = \sum_{i}^{I} r_i \tag{11.23}$$

$$\begin{array}{cc|cccccccccc}
 & & [b_m, b_M] & [0,\infty) & [0,\infty) & [0,\infty) & [0,\infty) & [0,\infty) & [0,\infty) & (-\infty,\infty) & [0,\infty) & (-\infty 0,\infty) \\
 & & g_1 & g_2 & g_3 & s_1 & s_2 & s_3 & E_j^f & E_j^d & I_k^f & I_k^d \\
\hline
(-\infty,0] & \text{Eq.11.7} & -1 & r_m & & & & & & & & \\
[0,\infty) & \text{Eq.11.8} & -1 & r_M & & & & & & & & \\
(-\infty,0] & E_j^f & f_{e_j^1} & f_{e_j^2} & & & & & -1 & & & \\
[0,0] & E_j^d & & & & f_{e_j^1} & f_{e_j^2} & & & -1 & & \\
[\alpha,\beta] & E_j^b & & & & & & & 1 & -1 & & \\
[0,0] & E_j^i & & & & & & & -1 & & 1 & \\
(-\infty,0] & I_k^f & f_{i_k^1} & f_{i_k^2} & & & & & & & -1 & \\
[0,0] & I_k^d & & & & f_{i_k^1} & f_{i_k^2} & & & & & -1 \\
[\gamma,\delta] & I_k^b & & & & & & & & & 1 & -1 \\
[0,\infty) & \text{Eq.11.22} & & & & & & & & & \eta & \\
(-\infty,0] & \text{Eq.11.21} & & & & & & & & & \theta & \\
[O_m,O_M] & \text{Eq.4.6} & & & & & & & v_e w_e & & & \\
[\upsilon,\omega] & \text{Pres} & & & & 1 & 1 & & & & & \\
[0,0] & \text{Eq.11.9} & -w_{1|3} & -w_{2|3} & 1 & & & & & & & \\
[0,0] & \text{Eq.11.13} & & & & -q_{1|3} & -q_{2|3} & 1 & & & & \\
 & \text{Opt} & 1 & & & & & & & & &
\end{array}$$

11.6 Solving the Optimization Problem

Once the matrix seen in figure 11.5 has been constructed by ADER it is converted into a dense column-major format and passed off to the CLP library which solves this matrix as a simplex problem. CLP then returns a vector containing the atomic density of each group in a material, normalized to the material's density at the beginning of the optimization process. This vector also contains all the group-class mass load values needed to bring the material cluster to the optimal state. If the simulation is in the first criticality search iteration for a burnup step, the actions of all discrete form streams,both group-class and table-class, are applied to materials. If the simulation is past the first criticality search iteration for the current burnup step only the actions of discrete group-class streams are applied to materials. Following these actions a Monte Carlo transport sweep is run with all the same cycle and batch parameters as specified by the user in the SERPENT2 input files. At this point, if the system k_{eff}^{analog} is within the bounds as set by the user (or the default bounds) or if the criticality search has already used up all of its iterations, the simulation progresses on to the burnup calculation. If not, another criticality search iteration is launched starting with the determination of the isotopic composition for all unfixed elements in groups and group-class streams.

11.7 Nuclear Burnup Calculations

ADER's burnup routine rides along the iteration scheme employed by the SERPENT2 burnup routine. That is to say that ADER is compatible with any burnup correction scheme that the user employs through SEREPNT2. In truth, the only affect that a burnup iteration scheme has on ADER is to change the cross sections used in any criticality control rows in the optimization matrices. ADER extends the burnup capabilities of SERPENT2 to include the effects of both group and table-class streams. The coefficients in the burnup matrix are those from the Bateman equation as seen in equation 11.24 for the one energy group and zero dimensionality case, where N is the number density of nuclide n, t is time, $b_{m\to n}$ is the branching ratio for the decay of nuclide m into n, λ is the decay constant for its sub-scripted nuclide, q goes over all neutron induced absorption reactions for a given isotope, $a_{m\to n}^{q}$ is the branching ratio for isotope m into n due to reaction q, σ_x^y is the effective microscopic cross section of reaction x for isotope y, ϕ is the scalar neutron flux, d denotes all transmutation reactions for a given isotope, $R_n(t)$ is a fractional removal (or addition) rate for isotope n at time t, and $F_n(t)$ is a feed (or removal) amount for isotope n at time t. These last two terms in equation 11.24 account for proportional and continuous form streams respectively.

A highly truncated burnup scheme can be seen in figure 11.25 in which there are two isotopes, ^{233}U and ^{135}Xe, and two streams; S_c representing a continuous stream with a constant injection rate and S_p representing a proportional stream with a transfer rate dependent upon the concentration of the substances to be transferred. There are, of course, two matrices as well. The burnup matrix to the left holding the coefficients of the Bateman equation and the second, to the right, holding the initial concentrations of isotopes and the values for the streams. The first column of the first row gives the creation and destruction of ^{233}U which is dependant on the concentration of ^{233}U with Γ representing nuclear destruction as seen in equation 11.26. The third column of the first row holds the fraction of stream S_c that ^{233}U comprises. These entries together describe the evolution of ^{233}U in the given system. In the second row Ξ, as seen in equation 11.27, represents the production of ^{135}Xe from ^{233}U. In the second column of the second row are the processes dependant on the concentration of ^{135}Xe. Υ represents the proportional rate constant as determined by the multiplication of $q_{i,v}$ and r_v whereas Θ is given by equation 11.28. The third row is blank as the abundance of a continuous type stream, c_n, does not change over a burn step. The fourth row is an addition specific to ADER and not found in the Bateman equations; rather, this line, and the lines it represents, exists to keep track of the amount of an isotope that a proportional stream moves simply to provide this information to the user. The system of matrices seen in figure 11.25 is solved by SERPENT2 using the CRAM methodology providing updated isotopic abundances and proportional stream transfer amounts.

$$\begin{aligned}\frac{\mathrm{d}N_n(t)}{\mathrm{d}t} = & \sum_m^M b_{m\to n}\lambda_j N_j(t) + \\ & \sum_m^M \sum_q^Q a^q_{m\to n}\sigma^k_q \phi(t) N_k(t) - \\ & N_n(t)\lambda_i - \sum_d^D \sigma^n_d \phi(t) N_n(t) - \\ & R_n(t)N_n(t) + F_n(t)\end{aligned} \tag{11.24}$$

$$\begin{array}{cc} & \begin{array}{cccccc} {}^{233}\mathrm{U} & & {}^{135}\mathrm{Xe} & S_c & S_p & \mathbb{N} \end{array} \\ \begin{array}{c} {}^{233}\mathrm{U} \\ {}^{135}\mathrm{Xe} \\ S_c \\ S_p \end{array} & \begin{bmatrix} -\lambda_{^{233}\mathrm{U}} + \Gamma & & f^{S_c}_{^{233}\mathrm{U}} & \\ \Xi & -\lambda_{^{135}\mathrm{Xe}} + \Upsilon + \Theta & & \\ & & & \\ & \Upsilon & & \end{bmatrix} \begin{bmatrix} N_{^{233}\mathrm{U}} \\ N_{^{135}\mathrm{Xe}} \\ c_n \\ 0 \end{bmatrix} \end{array} \tag{11.25}$$

$$\Gamma = -\sum_d^D \sigma_d^{^{233}\mathrm{U}}\phi \tag{11.26}$$

$$\Xi = b_{^{233}\mathrm{U}\to{}^{135}\mathrm{Xe}}\lambda_{^{233}\mathrm{U}} + \sum_q^Q a_{^{233}\mathrm{U}\to{}^{135}\mathrm{Xe}}\sigma_q^{^{233}\mathrm{U}}\phi \tag{11.27}$$

$$\Theta = -\sum_d^D \sigma_d^{^{135}\mathrm{Xe}}\phi \tag{11.28}$$

Following the solution of the burnup problem the material compositions are updated accordingly and the simulation moves on to the next burnup step.

12 Installation

Installing ADER is a five step process: Downloading ADER covered in section 12.1, installing the CLP libraries covered in section 12.2, editing the necessary lines in the SERPENT2 base code covered in section 12.3, compiling the code covered in section 12.4, and testing the code, already covered in section 8. These instructions assume the user is operating inside of a linux-like environment.

12.1 Downloading ADER

ADER is available as a public repository hosted at...

```
some_web_address
```

Download ADER using your preferred client. The directories contained in the ADER parent directory, call this folder `ADER_Dir`, are...

- `docs` - where this user manual and the API can be found in the subdirectories `docs/UM` and `docs/API`, respectively.
- `inputs` - where in the subdirectory, `inputs/Test_Input`, the file `test_input.txt` can be found.
- `src` - where all of ADER's source files can be found.
- `System_Testing` - where all of ADER's system tests can be found in their respective subdirectories titled after the test which they contain. Inside of each subdirectory is the test input file and a README file explaining the operation of the test.

12.2 Downloading and Installing CLP

CLP is available as a public repository hosted at...

```
https://github.com/coin-or/Clp
```

Download CLP using your preferred client. Inside this directory, call it `Clp_Dir`, there is one important subdirectory, `Clp_Dir/Clp`. Inside of this subdirectory executing the following commands in a linux-like environment should install the Clp libraries on your system...

```
./configure
./install-sh
make all
```

Copy the file `Clp_C_Interface.h`, found in `Clp_Dir/Clp/src`, into your SERPENT2 build directory. After this process is complete do not forget to add the path to this subdirectory to all applicable system paths, most likely just your system `PATH`.

12.3 Editing SERPENT2

These instructions assume that the base version of SERPENT2 being modified is version 2.1.30. ADER DOES NOT WORK WITH EARLIER VERSIONS OF SERPENT2. Compatibility with future versions of SERPENT2 and installation directions are not guaranteed. To configure SERPENT2 to interact with ADER three steps are necessary. First, copy all of the files found in `ADER_Dir/src`, to the SERPENT2 `src` directory (or wherever you build SERPENT2). This might look something like the following...

```
cd ~/SERPENT2_Dir
cp ~/ADER_Dir/src/* ./src/
```

In the directory `ADER_Dir/docs/UM` there is a file named "Source_mod.txt". "Ln" is short for "line number". The contents of this file have the following format...

```
[function_name].c: Line number or description of location in file
{
#ADER MOD BEGIN#
contents to add to file at the designated location
#ADER MOD END#
}
```

For each listing, add the code contents found between the `ADER MOD BEGIN` and `ADER MOD END` tags to the actual SERPENT2 source file given at the location described: this is either inserting new code lines following an existing SERPENT2 source line given as either an insertion listing or as a single line number or as a range of line numbers in which case this range is to be fully replaced by the ADER code sample. A suggestion is to execute these file modifications from the modifications closest to the end of the file to the modification closest to the beginning of the file - in this way the line numbers you look for are unchanged by the addition of the ADER code. It is considered good practice to include the tags themselves ,`ADER MOD BEGIN` and `ADER MOD END`, but this is not strictly necessary. Consider the file, "main.c", seen below...

```
#include header.h

void main(args)
{
long j;
long i = 0;

i = 1;

return(i);
}
```

Now consider that one of the entries in the file "Source_mod.txt" looks like the below...

```
Main: Ln 6-8
{
#ADER MOD BEGIN#
for(j=0; j<10; j++)
{
    i++;
}
#ADER MOD END#
}
```

The correctly modified main.c file would look like the below where lines 6 through 8 of the base file were replaced with the content between and including the ADER mod tags: "`ADER MOD BEGIN`" and "`ADER MOD END`". Please use your best discretion when editing these files. If a line number direction would cut off a `for` loop and cause compilation or logic errors, obviously that line number is incorrect and should be slightly different.

```
#include header.h

void main(args)
{
long i = 0;
#ADER MOD BEGIN#
for(j=0; j<10; j++)
{
    i++;
}
#ADER MOD END#
return(i);
}
```

12.3.1 Modifying the Makefile

Add the following lines to the SERPENT2 Makefile before the `OBJS` list but replace `[absolute/path/to/the/CLP/library]` with the actual path to the Clp library on your system.

```
# ADER MOD BEGIN #
##############################################################################

# Below should be "-L[absolute/path/to/the/CLP/library] -lclpsolver
#     and -L[absolute/path/to/the/CLP/library] -lclp

LDFLAGS += -L/Clp_Dir/Clp/lib -lClpSolver -L/Clp_Dir/Clp/lib -lClp

# Enable ADER_TEST to run the test_input.txt file to execute the
#     unit and integration test suite

#CFLAGS += -DADER_TEST

# Enable ADER_INT_TEST to ouput the files necessary for integration
#     testing. Many of these files are human readable and useful
#     for manual debugging

#CFLAGS += -DADER_INT_TEST

# Enable ADER_DIAG to output a copious amount of debugging data

#CFLAGS += -DADER_DIAG

##############################################################################
# ADER MOD END #
```

In the directory `ADER_Dir/docs/UM` there is a file named "Object_list.txt" - add the contents of this file to the SERPENT2 Makefile `OBJS` list. You may alphabetize the entires if you care to but it is not necessary.

In the directory `ADER_Dir/docs/UM` there is a file named "Build_list.txt" - add the contents of this file to the SERPENT2 Makefile build list - the long list of makefile build instructions at the bottom of the file. You may alphabetize the entries if you care to but it is not necessary.

12.4 Compiling ADER

To compile ADER with SERPENT2, after having followed the steps in all previous subsections of this section, give the command below inside of your `SERPENT2_Dir/src` directory...

```
make all
```

To compile ADER for unit testing, uncomment the line with the phrase "`CFLAGS += DADER_TEST`". When compiled this way the entire code will work for no other purpose than ADER unit testing conducted with the file "`test_input.txt`" found inside the "`ADER_Dir/inputs/Test_Input`" directory.

The compilation flags "`DADER_DIAG`" and "`DADER_INT_TEST`" are covered in section 7.4.