

CENTRE DE DÉVELOPPEMENT TECHNOLOGIQUE
TECHNICAL REPORT

**DRAGON VERIFICATION PLAN:
A PROPOSAL FOR THE MODULES REQUIRED
IN THE AECL CRITICAL PATH**

G. Marleau
Institut de génie nucléaire
Département de génie mécanique
École Polytechnique de Montréal
February, 2001

École Polytechnique de Montréal
Campus de l'université de Montréal
C. P. 6079, succ. Centre-ville
Montréal, Québec
Canada H3C 3A7

SUMMARY

In this report one will find a description of the general verification plan we proposed for the DRAGON routines that are to be called during a typical execution at AECL. The type of verification that is suggested for the various routines will vary from a line by line analysis of the routine to global verification of a set of routines by comparing the results of DRAGON with that expected using an independent analysis of the same problem.

The justification for using such a wide range of verification method resides in the structure of the code itself. On the one hand, a large number of subroutines perform very specific tasks using a limited amount of information and are well suited for line by line verification. On the other hand, one can also find in DRAGON very long routines which are mainly used to control the execution of DRAGON. These routines only perform a limited number of explicit calculations their main goal being to transfer information between the main calculation routines and the data structure. In addition, because of the complexity of some of the calculation techniques used in DRAGON, some routines possess cross-analysis features on which one can rely in the verification plan.

CONTENTS

SUMMARY	i
CONTENTS	ii
1 INTRODUCTION	1
2 VERIFICATION PLAN FOR THE GEOMETRY MODULE	3
3 VERIFICATION PLAN FOR THE EXCELT TRACKING MODULE	5
4 VERIFICATION PLAN FOR THE CP INTEGRATION MODULE	10
5 VERIFICATION PLAN FOR THE FLUX SOLUTION MODULE	13
6 VERIFICATION PLAN FOR THE EDITING MODULE	16
7 CONCLUSION	19
REFERENCES	20

1 INTRODUCTION

The goal of this report is to describe the verification procedure that is proposed in order to ensure that the code DRAGON perform adequately when it is used for the evaluation of the incremental cross sections associated with CANDU reactivity devices.^[1,2] As we described in the report *DRAGON/WIMS–AECL Coupled Execution: Critical Path and Interfaces* such calculations involved only a limited number of DRAGON modules.^[3]

The verification plan proposed here therefore concentrates on the critical DRAGON modules. Moreover, only a subset of the DRAGON routines in a specific module will generally be considered in this initial verification report because of the limitations imposed in the types of calculations that will be performed at AECL using DRAGON.

This report covers the general verification plan for the following modules:

- **GEO**: that reads a DRAGON geometry on the input file;
- **EXCELT**: that analyzes a DRAGON geometry and generates the CP integration lines associated with this geometry;
- **ASM**: that reads the integration lines associated with a geometry and performs the multi-group CP integration;
- **FLU**: that solves the multigroup CP flux equations for fixed, fission or leakage sources;
- **EDI**: that performs the editing of condensed and homogenized macroscopic cross section.

These modules are called successively during a DRAGON supercell calculation.

Some of these modules will not be validated completely. There are many different reasons for this restriction. For example, by limiting the analysis to 3–D geometries there is no need to examine the routines in DRAGON associated with the integration of collision probabilities for 2–D geometries.^[3] Moreover, since the use of the **JPMT**: and **SYBILT**: modules is excluded from the critical path (because of the approximations these methods entail and the fact that they are 1–D or 2–D dimensional modules) all the subroutines associated with these modules will be considered outside the limits of the current verification proposal.

Our first objective here is to identify clearly the complete set of routines that will be verified for each module. Then, we will propose a verification process that will depend strongly on the specific module to be considered. For example, the **GEO**: module of DRAGON does not perform any calculation. It only reads the information provided in the input file and transfers this information directly to a **GEOMETRY** data structure. On the other hand the subroutines associated with the **EXCELT**: module are used to analyze the geometry and to generate the integration lines required for CP evaluation even if it does not contain information specifically related to the collision probability technique. Finally, the worst case scenario can be found in the **EDI**: module which performs cell homogenization and energy condensation because of the presence of the **SPH** homogenization option which requires knowledge of the collision probabilities associated with the homogenization geometry.

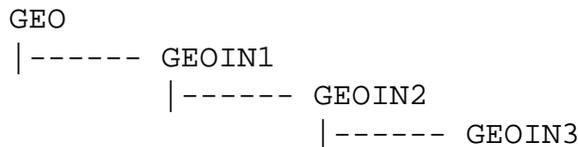
In most case we will not consider a line by line verification of each of the routines in a module. We will generally select one or a list of routines performing a specific task in a given module and verify that these routines perform as expected. Typically, the 3-D tracking routines in the module EXCELT: are used to generate the binary tracking file containing the integration lines. The contents of this tracking file can be verified in various ways, including graphically using a program such as Mathematica. Finally, some of the module can also be verified indirectly by comparison with WIMS-AECL.^[4] For example, one can use in the DRAGON FLU: module the WIMS-AECL computed collision probabilities provided they are reformatted adequately. Since the FLU: module is geometry independent, the results of this module could be compared directly with those obtained using a 2-D WIMS-AECL calculation. This should lead to an overall verification of the FLU: module that should be adequate.

One possibility available in DRAGON which will help in the verification plan is that all the data structures, except for the sequential binary tracking file, can be exported to an ASCII sequential file. Moreover the contents of these data structure is well known and documented.^[5] Accordingly, the various data structures transferred from one module to the next can be examined directly and the errors, if any, identified rapidly. This will be of help for the verification since some data structure can be rebuilt directly from the input requirement of a specific module and compared with the information produced by this module.

2 VERIFICATION PLAN FOR THE GEOMETRY MODULE

The DRAGON module that is used to transfer the geometry description provided as input data to a GEOMETRY data structure is called `GEO :`. This module is controlled by the subroutine `GEO` and involves only three additional subroutines. It has the following structure:^[6]

Structure of the `GEO :` module



The first observation here is that the three subroutines that are called successively by `GEO` represent a FORTRAN-77 interpretation of a recursive call. In fact, `GEOIN1` and `GEOIN2` are identical apart from the fact that their names differ and that the call to `GEOIN2` in `GEOIN1` is replaced by a call to `GEOIN3` in `GEOIN2`. Similarly `GEOIN3` is nearly identical to `GEOIN2`, however, in this case the recursivity is not completed and it is forbidden to define a sub-geometry in `GEOIN3`.

The second observation is that the three routines called by `GEO` are mainly there to read the information provided in the DRAGON input file and transfer it to the GEOMETRY data structure. The analytic capability of the routines presented above is somewhat restricted and a large number of invalid geometries can be defined using this module. In fact some of these geometries may be invalid because they cannot yet be analyzed by one of the DRAGON modules. Other geometries are simply inconsistent or will never be available in DRAGON for processing.

For the 3-D geometries that can be analyzed in DRAGON the `GEOIN3` routine will never be called since these geometries can be defined either in the form of a single block or as an assembly of blocks. For the analysis of CANDU reactivity devices the only option is in fact to use the assembly option of DRAGON. In the case where 3-D end-regions calculations are to be performed using DRAGON then both the one block and the assembly option can be used.

This module could be verified using two straight forward techniques. Since there are only two different routines in this module, namely the routine `GEO` and `GEOIN1` (`GEOIN2` and `GEOIN3` being duplicates of `GEOIN1`), a straight forward line by line verification of the routines can be considered. The use of reverse engineering would also be well suited for this problem since the amount of information transferred to the GEOMETRY data structure is small and nearly identical to the information available in the input file. For example the information following the `MESHX` DRAGON keyword should be identical to the contents of the `MESHX` record on the data structure.^[2,5] This can be verified by comparing the resulting GEOMETRY data structure directly with the DRAGON input file.

The verification plan we propose for the subroutines called by the `GEO :` module is therefore the following:

1. perform a line by line verification of `GEOIN1`;

2. compare the subroutines GEOIN2 and GEOIN3 with GEOIN1 and verify that the differences observed are those expected if one takes for granted that these subroutines are used to simulate a FORTRAN-77 recursive call;
3. verify that the information stored on the GEOMETRY data structure generated by the GEO : module is identical to that provided on the DRAGON input file for typical geometries that can be processed by the EXCELT : module.

3 VERIFICATION PLAN FOR THE EXCELT TRACKING MODULE

The DRAGON module that is used for 3-D geometry analysis is called EXCELT: and is controlled by the subroutine EXCELT. It has the following general structure:^[6]

General structure of the EXCELT: module

```

EXCELT
|----- LDRASS  ->
|----- XELDRV
|
|----- LELCHK
|----- XELCMP
|----- XELCOP
|----- XELNTR
|----- XELTRK
|
|----- XELBIN
|----- XELDCL
|----- XELPRC
|----- XELEDC
|----- XELETR
|----- XELGPR
|----- XELMRG
|----- XELPRP
|----- XELTI2
|----- XELEQN
|----- XELLIN
|----- XELLSR
|----- XELTI3
|----- XELEQN
|----- XELLIN
|----- XELLSR
|----- XELTRP
|----- KELMRG
|----- KELRNG
|----- LELCRN
|----- KELSYM
|----- XELGRD
|----- XELVOL
|----- LELCRN
|----- XELCRN
|----- XELPSC
|----- XELPSI
|----- XELTS2

```

```

|           |           |----- XELLIN
|           |           |----- XELLSR
|           |           |----- XELTSA
|           |           |----- XELTSW
|----- XCWTRK
|           |----- XCGBCM
|           |----- XCGDIM
|           |----- XCGGEO
|           |           |----- XCGROD
|----- XCWICL
|           |           |----- XCWHEX
|           |           |----- XCWREC
|           |           |----- XCWROD
|           |           |----- XCWSRT
|           |----- XCWSCl
|           |           |----- XCWREC
|           |           |----- XCWROD
|           |           |----- XCWSRT
|           |           |----- XELTSA
|           |           |----- XELLSW
|----- XHXTRK
|           |----- LHXUNH
|           |           |----- DEPLIT
|           |           |----- DUTURN
|           |----- MESHST
|           |           |----- NEIGHB ->
|           |----- NEIGHB ->
|           |----- TRKHEX

```

The first observation is that this module contains three different tracking routine subsets, namely the XELTRK, XCWTRK and the XHXTRK tracking options. The subset of routines associated with XCWTRK is required for 2-D cluster geometry analysis. From the point of view of AECL, such 2-D calculations will be performed using WIMS-AECL.^[3] Accordingly, the verification of the XCWTRK subset of routines will not be considered here. Similarly, we will not verify the subset of routines associated with XHXTRK since they are only required for 2-D and 3-D hexagonal geometry analysis. Finally, because DRAGON will be used at AECL primarily for 3-D mixed Cartesian/cylindrical geometry analysis, the main subset of routines we will need to consider is that associated with XELTRK.

A second observation is that the XELTRK routine has two different functions. First, it is used to analyze a mixed Cartesian/cylindrical 2-D or 3-D assembly of cell. In DRAGON the distinction between 2-D and 3-D cells at this level is somewhat rhetoric since a 2-D geometry is simulated by a 3-D geometry of spatial extension $z = 1$. The main difference between the 2-D and 3-D geometry processing arises at the level of integration line generation. In the

EXCELT: module this tracking is performed by the routines XELTI2 and XELTS2 for 2-D geometries with isotropic and specular integration lines respectively and by XELTI3 for 3-D geometries with isotropic integration lines. As a result, we will mainly concentrate our efforts here on the verification of the XELTI3 routine.

From the point of view of AECL use, the EXCELT: module will have the following structure:

Structure of the module EXCELT: as used by AECL

```

EXCELT
|----- LDRASS  ->
|----- XELDRV
|
|----- LELCHK
|----- XELCMP
|----- XELCOP
|----- XELNTR
|----- XELTRK
|
|----- XELBIN
|----- XELDCL
|           |----- XELPRC
|----- XELEDC
|----- XELETR
|----- XELGPR
|----- XELMRG
|----- XELPRP
|----- XELTRP
|           |----- KELMRG
|           |----- KELRNG
|           |           |----- LELCRN
|           |----- KELSYM
|           |----- XELGRD
|           |----- XELVOL
|           |           |----- LELCRN
|           |           |----- XELCRN
|           |           |           |----- XELPSC
|           |           |           |----- XELPSI
|----- XELTI3
|           |----- XELEQN
|           |----- XELLIN
|           |----- XELLSR

```

The overall verification process we suggest here is relatively simple since the EXCELT: module already contains a large number of self-verification features. For example, the function

LELCHK is used to compare the contents of a reference TRACKING data structure with that generated by EXCELT: after the analysis of a new GEOMETRY. This feature will be used to verify if the geometry analysis performed by EXCELT: is consistent with an independent analysis of the same geometry. For such a verification, we can select a few geometries that will be representative of the type of calculations to be performed at AECL. For these geometries we will generate independently the TRACKING data structure expected using the information provided in the reports IGE-232 and IGE-233.^[5,7] The region volume and surface area will be computed analytically.^[8] This verification process will involve a global verification of all the routines called by XELTRK except for XELTI3 which is used to generate the binary tracking file using the information stored on a TRACKING data structure.

The EXCELT: module contains an additional self-verification option for the computed regional volume and surface area which is already included in the subroutine XELNTR that is used for the track normalization. In fact, the track normalization process available in XELNTR relies on a comparison between the numerically computed region volume and surface area (using the information stored on the binary tracking file) and those evaluated analytically and stored on the TRACKING data structure. By comparing these volumes and surface areas, it is possible to verify the consistency between the tracking routine, the binary tracking file and a reference TRACKING data structure. Moreover, since the routine XELNTR can process both a new binary and an old binary tracking file, it is possible to use it on an already normalized binary tracking file to verify if the normalization process was performed adequately.

We will also perform an independent verification of the integration line generating routines XELTI3, XELEQN, XELLIN and XELLSR. Here, for a finite number of line directions and starting points we will evaluate manually the intersection between these lines and the various regions in the geometry and compare the result with the equivalent integration line stored on the binary tracking file.

Finally here is the complete verification process we propose for the EXCELT: module.

1. A line by line verification of the following subroutines:
 - XELPRC and XELPRP that read a GEOMETRY data structure and checks if this geometry can be processed by the EXCELT: module;
 - LELCHK that compares two TRACKING data structures;
 - XELCOP that is used to duplicate a binary tracking file;
 - XELEQN that generates the integration direction and weights;
 - LELCRN, XELCRN, XELPSI and XELPSC that are used to identify and compute the surface of intersection between a 2-D Cartesian region and an annular region.
2. A global verification of the geometry analysis routines:
 - a comparison of the TRACKING data structure generated by DRAGON for a variety of geometries with reference TRACKING data structures generated manually;
 - an automatic DRAGON comparison of the TRACKING data structure generated by DRAGON for a variety of geometries with reference TRACKING data structures generated manually;

3. A verification of the contents of the binary tracking file:

- by generating manually integration lines and comparing with those present on the binary tracking file;
- by using the subroutine XELNTR to compare automatically the regional volume evaluated using the binary tracking file with those present on the TRACKING data structure;
- by using the subroutine XELNTR to validate automatically the track normalization process (by using XELNTR on a binary tracking file that is already normalized).

4 VERIFICATION PLAN FOR THE CP INTEGRATION MODULE

The DRAGON module that is used for collision probability integration is ASM: and is controlled by the subroutine ASM. It has the following general structure:^[6]

General structure of the ASM: module

```

ASM
|----- ASMDRV
|
|----- PIJCPL
|----- PIJNOS
|----- PIJNRM
|----- PIJSMD
|----- EXCELP
|
|----- PIJAAA
|----- PIJABC
|----- PIJCMP
|----- PIJI2D
|----- PIJI3D
|----- PIJKST
|----- PIJRDG
|----- PIJRGL
|----- PIJRHL
|----- PIJRNL
|----- PIJS2D
|----- PIJS3D
|----- PIJWPR
|----- JPMA   ->
|----- JPMP   ->
|----- SYBILP ->

```

As we mentioned in the introduction the routines JPMA, JPMP and SYBILP will never be called for 3-D problems since these routines are all restricted to the computation of CP matrices for 1-D and 2-D problems.

The routines PIJI2D and PIJS2D which are the general CP integration routines for 2-D geometries with isotropic and specular tracking respectively are not critical to the use of DRAGON at AECL. Similarly, the routines PIJAAA and PIJKST which are required for the evaluation of the directional collision probabilities and the routine PIJCMP which is used to transform the general CP matrix in a symmetric format are restricted to 2-D geometries and will not be considered in the current verification program. The subroutine PIJNRM is only used if the albedo leakage model is to be considered in the flux solution module. Such a leakage model is rarely used in standard DRAGON executions and is present mainly for academic purposes. Accordingly, the verification of this subroutine is not necessary at this stage in the verification. Finally

the subroutine PIJS3D which is programmed for the integration of 3-D collision probabilities based on a specular tracking cannot be used in the current version of DRAGON since such a 3-D tracking is not yet available in DRAGON.

As a result the structure of the ASM: module that will be considered here is:

Structure of the ASM: module as used by AECL

```

ASM
|----- ASMDRV
      |----- PIJCPL
      |----- PIJNOS
      |----- PIJSMD
      |----- EXCELP
                |----- PIJABC
                |----- PIJI3D
                |----- PIJRDG
                |----- PIJRGL
                |----- PIJRHL
                |----- PIJRNL
                |----- PIJWPR

```

The overall verification process we suggest here will be divided into three parts.

1. A line by line verification for the following routines:

- the collision probabilities re-normalization routines PIJRDG, PIJRGL, PIJRHL and PIJRNL. These are documented in Part 1 of the Dragon theory manual and in various publications;^[9-11]
- the routine PIJABC that takes into account the boundary conditions to build the complete collision probability matrix;^[8,9]
- the routine PIJSMD that generates the scattering modified CP matrix from the reduced CP matrix stored in a symmetric format;
- the routine PIJNOS that generates the standard CP matrix from the reduced CP matrix stored in a symmetric format;
- the routine PIJCPL that generates the total leakage matrix.

2. A numerical verification for the collision probability matrices. Here we will consider the following analysis:

- for selected integration lines we will evaluate independently the contribution of this line to the various components of the CP matrix and compare these contributions with those computed in the subroutine PIJI3D;
- for selected simple geometry perform an independent evaluation of the CP matrix and compare with that generated by PIJI3D;

3. The use of the self-verification option already present in the PIJWPR subroutine. This subroutine is generally used for printing the collision probability matrix. However, it contains two features which are of interest for the verification process:
 - it verifies the conservation relations associated with the collision, leakage and escape probabilities;^[9]
 - it is called both before and after the collision probability re-normalization procedure as a means to evaluate the efficiency of these procedures and to detect the presence of unwanted negative probabilities which may be generated by the subroutines PIJRDG and PIJRGL.

5 VERIFICATION PLAN FOR THE FLUX SOLUTION MODULE

The DRAGON module that is used to solve the multigroup CP equations for the flux and eigenvalue is called FLU: and is controlled by the subroutine FLU. It has the following general structure:^[6]

General structure of the FLU: module

```

FLU
|----- FLUGET
|         |----- FLUGPI
|         |----- FLURFL
|----- FLUDRV
|         |----- FLUASR
|         |----- FLUADN
|         |----- FLUGFL
|         |----- FLUSFL
|         |----- FLUQFC
|         |----- FLUINR
|         |----- FMODUL
|         |----- SMODUL
|         |----- FLUACV
|         |         |----- SMODUL
|         |----- FLUBAL
|         |----- FLUDB2
|         |         |----- FMODUL
|         |         |----- FLUALB
|         |----- FLUFUI
|----- FLUQFE
|         |----- FLUGFL
|         |----- FLUSFL
|         |----- FLUBLN
|         |----- FLULBD
|         |----- FLUQFB
|         |----- FLUQFS
|         |----- FLUQFX
|         |----- B1HOM
|         |         |----- B1DIF
|         |         |----- B1BETA
|         |         |----- B1SOL

```

The first observation here is that instead of using the generic flux solution and residual calculation external function names (FMODUL and SMODUL respectively), we will insert directly in

the above structure the explicit subroutine names used for the solution of all collision probability problems, namely the TRFICF and TRFICS subroutines respectively. In fact the only other options that is available in DRAGON involves a response matrix solution to the JPM transport problem that is not pertinent to the current discussion.

There are also five subroutines that appear in this module which are reserved for B_1 heterogeneous leakage calculations, namely FLUFUI, FLUBLN, FLULBD, FLUQFB and FLUQFC. Since this leakage model is implemented only for 2-D calculations, these routines will not be considered as part of the validation. Similarly, the subroutine FLUALB will not analyzed since it is required when a leakage model with an albedo approximation is used. As we mentioned in Section 4 this option is there mainly for academic purposes and is never used for 3-D transport calculations. Finally the subroutines FLUASR and FLUADN are only required for adjoint and generalized adjoint flux calculations and will not be included in our validation.

As a result the final structure of the FLU module we will consider here is:

Structure of the FLU: module as used by AECL

```

FLU
|----- FLUGET
|         |----- FLUGPI
|         |----- FLURFL
|----- FLUDRV
|         |----- FLUGFL
|         |----- FLUSFL
|         |----- FLUINR
|         |         |----- TRFICF
|         |         |----- TRFICS
|         |         |         |----- TRFICF
|         |         |----- FLUACV
|         |         |         |----- TRFICS
|         |         |         |         |----- TRFICF
|         |         |----- FLUBAL
|         |         |----- FLUDB2
|         |         |         |----- TRFICF
|----- FLUQFE
|         |----- FLUGFL
|         |----- FLUSFL
|         |----- FLUQFS
|         |----- FLUQFX
|         |----- B1HOM
|         |         |----- B1DIF
|         |         |----- B1BETA
|         |         |----- B1SOL

```

The overall verification process we suggest here will be divided into four parts.

1. A line by line verification for the following routines:

- FLUGET and FLUGPI that are used to read the FLU: processing options;
- TRFICF that is used to compute the one group flux ϕ^g associated with a fixed source S^g using:

$$\phi_i^g = \sum_j p_{ij}^g S_j^g;$$

- TRFICS that is used to compute the one group residual $\Delta\phi^g$ associated with a fixed source S^g and a reference flux ϕ^g using:

$$\Delta\phi_i^g = \phi_i^g - \sum_j p_{ij}^g S_j^g;$$

- FLUQFX and FLUQFS that add to the current source the contribution from fission and an external fixed source respectively;
- FLUGFL and FLUSFL that read and write the flux to or from the FLUXUNK data structure.

2. A global numerical verification for the following routines:

- TRFICF and TRFICS that are used to compute the one group flux ϕ^g and residual $\Delta\phi^g$ as described above;
- FLUQFX and FLUQFS that add to the current source the contribution from fission and an external fixed source respectively;
- the B1DIFF set of subroutines that computes homogeneous B_0 and B_1 diffusion coefficients and the corresponding buckling;

3. A self-verification process for the following routines:

- for FLUDB2 and B1DIFF by comparing the case where convergence on buckling is considered with that where a k_{eff} convergence is used with imposed leakage.

4. A comparison with WIMS–AECL:

- for FLUINR that controls the thermal iteration and relies on a rebalancing and variational acceleration;
- for FLUQFE that controls the power iteration.

6 VERIFICATION PLAN FOR THE EDITING MODULE

The DRAGON module that is used to edit the homogenized and condensed reaction rates and cross sections is called EDI : and is controlled by the subroutine EDI. It has the following general structure:^[6]

General structure of the EDI : module

```

EDI
|----- EDIGET
|----- EDIENE
|----- EDITIS
|----- EDIDRV
      |----- EDIBAL
      |----- EDIDST
            |----- EDIDEL
            |----- EDISTA
      |----- EDIDTX
            |----- EDIPRR
            |----- EDIPXS
            |----- EDILBD
            |----- EDIRAT
            |----- EDISCT
            |----- EDILBD
      |----- EDIHFC
      |----- EDIISO
      |----- EDIMIC
            |----- EDIMPR
            |----- EDIMRR
            |----- EDIMXS
            |----- EDIUPS
            |----- XDRLGS
            |----- XDRLPR
            |----- XDRNED
      |----- EDITXS
      |----- SPHDRV ->

```

The main observations here is that a large number of these routines, namely EDIMIC, EDITXS, EDITIS, EDIISO are related to microscopic cross sections condensation and homogenization. Since the information that will be transferred from WIMS-AECL to DRAGON involves only macroscopic cross sections, these subroutine are of no interest in the current verification process. A second observation is that we will only consider a direct flux/volume homogenization. In that case the SPH factors are all be imposed to 1.0 and the SPHDRV subroutine is never

called. Here the EDIHFC subroutine is used to compute the H -factors. Unless the information provided on the MACROLIB that will be generated from WIMS–AECL contains the required information these H -factors will not be produced. Moreover, a more convenient method to generate the same information as that stored in the H -factors is to use an additional macroscopic cross section $\kappa\Sigma_f$ (κ is the energy produced by fission) which can be condensed and homogenized like any other vector cross sections. Accordingly, the EDIHFC subroutine will not be used in typical DRAGON executions at AECL. Finally, the EDILBD subroutine can only be used if a B_1 current consistent homogenization of the linearly anisotropic scattering cross section is required. In AECL application, this will not be the case and the subroutine EDILBD will never be called.

As a result the final structure of the EDI module we will consider here is:

Structure of the EDI : module as used by AECL

```

EDI
|----- EDIGET
|----- EDIENE
|----- EDIDRV
      |----- EDIDST
      |             |----- EDIDEL
      |             |----- EDISTA
      |----- EDIDTX
      |             |----- EDIPRR
      |             |----- EDIRAT
      |             |----- EDIPXS
      |             |----- EDISCT

```

The overall verification process we suggest here will be divided into three parts.

1. A line by line verification for the following routines:

- EDIGET and EDIENE that are used respectively to read the EDI : processing options and to select the condensation group limit associated with a specific energy;
- EDIRAT that is used to evaluate the reaction rates associated with various type of vector reactions;
- EDISCT that is used to evaluate the scattering rates.

2. A global numerical verification for the following routines:

- EDIDTX and EDIPRR and EDIPXS that are used to compute the reaction rates to print them and to print the associated macroscopic homogenized and condensed cross section;

- EDIDTS, EDISTA and EDIDEL that are used to compare reaction rates and to evaluate incremental cross sections.

3. A self-verification process:

- for EDIDTX using EDIDTS by comparing the homogenized and condensed cross sections generated by DRAGON with reference values generated manually.

7 CONCLUSION

In this report we described the minimal verification procedure that should be considered in order to ensure that the code DRAGON perform adequately when it is used at AECL. Even if this verification program looks relatively simple, it is in fact quite extensive both from the point of view of the manpower it will require and the amount of information that will have to be generated and analyzed. Moreover, for some of the modules described above, this verification does not only imply running DRAGON on typical problems, but also generating the information that is required for a line by line or a global verification of various subroutines.

A final comment concerns the extent of the verification plan we propose. It is in fact much more general than one can suspect at a first glance since a large number of the routines that will be verified here can be called by other DRAGON modules. One example of such modules is EXCELL: which relies on the subroutines of EXCELT: and ASM: for tracking and collision probability integration. As a result, the report that will result from this proposal may include the verification of some modules which are not necessarily part of the DRAGON critical path.

REFERENCES

- [1] G. Marleau, R. Roy and A. Hébert, *DRAGON: A Collision Probability Transport Code for Cell and Supercell Calculations*, Report IGE-157, Institut de génie nucléaire, École Polytechnique de Montréal, Montréal, Québec (1994).
- [2] G. Marleau, A. Hébert and R. Roy, *A User Guide for DRAGON. Version DRAGON_000331 Release 3.04*, Report IGE-174 Rev.5, Institut de génie nucléaire, École Polytechnique de Montréal, Montréal, Québec, (2000).
- [3] G. Marleau, *DRAGON/WIMS-AECL Coupled Execution: Critical Path and Interfaces*, Report IGE-249, Institut de génie nucléaire, École Polytechnique de Montréal, Montréal, Québec (2001).
- [4] J. Griffiths, *WIMS-CRNL Users Manual*, Report RC-1176, COG-94-52, Atomic Energy of Canada Limited, Chalk River, Ontario (1994).
- [5] A. Hébert, G. Marleau and R. Roy, *A Description of the DRAGON Data Structures. Version DRAGON_000331 Release 3.04*, Report IGE-232 Rev. 3, Institut de génie nucléaire, École Polytechnique de Montréal, Montréal, Québec (2000).
- [6] G. Marleau, *DRAGON Programmer's Manual*, Report IGE-251, Institut de génie nucléaire, École Polytechnique de Montréal, Montréal, Québec (2001).
- [7] G. Marleau, *The EXCELL Geometries Numbering Scheme in DRAGON*, Report IGE-233, Institut de génie nucléaire, École Polytechnique de Montréal, Montréal, Québec (1997).
- [8] G. Marleau, "Fine Mesh 3-D Collision Probability Calculations Using the Lattice Code DRAGON", *International Conference on the Physics of Nuclear Science and Technology*, Long Island, New York, October 5-8, 1998,
- [9] G. Marleau, *DRAGON Theory Manual Part 1: Collision Probability Calculations*, Report IGE-236, Institut de génie nucléaire, École Polytechnique de Montréal, Montréal, Québec (1999).
- [10] R. Roy and G. Marleau, "Normalization Techniques for Collision Probability Matrices", *PHYSOR-90*, pp IX.40-IX.49, Marseille, France, April 23-27, 1990.
- [11] E.A. Villarino, R.J.J. Stamm'ler, A.A. Ferri and J.J. Casal, "HELIOS: Angularly Dependent Collision Probabilities", *Nucl. Sci. Eng.*, **112**, 16-31 (1992).