TECHNICAL REPORT
IGE–260

NEW GEOMETRIES PROCESSING IN DRAGON:

The NXT: Module

G. Marleau

Institut de génie nucléaire
Département de génie physique
École Polytechnique de Montréal
December 2005

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

The tracking modules of DRAGON are used to perform two successive tasks, namely:[?,?,?,?]

1. to analyze the geometry provided and generate the information required for the tracking procedure to take place;
2. to generate, for a specific geometry, the integration lines resulting from a specific numerical quadrature.

The result of the first task is the TRACKING data structure where one can find the records that are essential for all the other calculations in DRAGON involving the geometry, including the regional volumes, the mixture associated with each region, the flux index cross referencing the elements in the flux/current array with elements in the volume array and the boundary conditions in the form of an albedo array. Depending on the type of geometry and on the specific tracking module considered, one can also find in this data structure additional records that can be used to rebuilt the geometry or associate a graphical image with the geometry. The result of the second task is the creation of the tracking file that contains all the information required to process the numerical quadrature specified in the tracking module.[?] The format of this tracking file does not depend on the tracking module selected even if the explicit contents of the file does. For example, the `NXT:` tracking file generated for a given geometry will in general differ from the information generated using the `EXCELT:` module for the same geometry.

The `NXT:` tracking module retains most of the properties of the tracking procedures already implemented in DRAGON, namely it contains a geometry analysis procedure as well as an integration line generation procedure. However, these procedures are programmed in such a way that they can be called by other modules of DRAGON, therefore making DRAGON more modular. The `NXT:` module has also been developed using a structure parallel to that used in the `EXCELT:` module (justifying the name NXT for New Excell Tracking). The main difference is the rationalisation of the information stored in the TRACKING data structure. The new records in the TRACKING data structure associated with the `NXT:` module are defined in such a way that they can accomodate new types of geometry as well as additional geometry levels, thereby simplifying the process of code maintenance and updating. In this report we will describe the `NXT:` tracking procedure which, in the long term, could replace the `EXCELT:` tracking procedure that has been in use for the last 20 years.

The primary goal of this report is to describe the `NXT:` tracking procedure. This will include:

- the theory manual for the `NXT:` module.
- the programmer's guide for the `NXT:` module;
- the user guide for the `NXT:` module;
- a description of the TRACKING data structure generated by `NXT:` including a presentation of the contents of the /NewExcellTrk/ subdirectory included in this data structure;

2 GEOMETRY CONSIDERATIONS

The `NXT:` module, while being quite general, can only process a limited number of geometry out of the full set that one can define using the DRAGON `GEO:` module. The main constraints on the geometries that can be currently analyzed by this module are the following :

1. A maximum of three sub-levels of geometry will be processed (see Figure 1 for a 2-D exemple of such a geometry construction). The levels are classified using the following hierarchy by the `NXT:` tracking module:

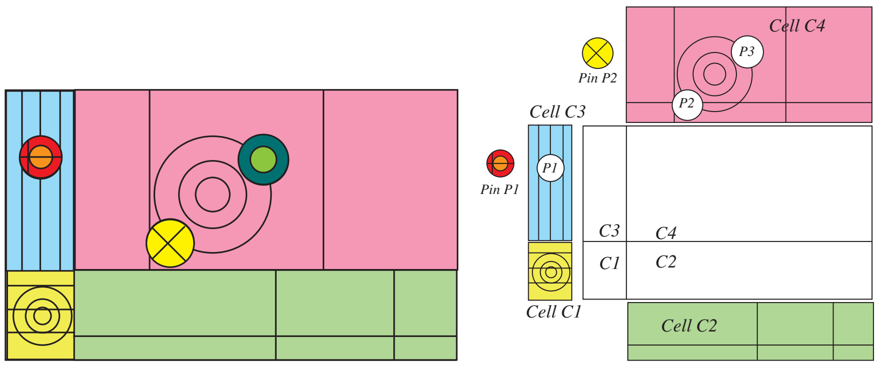

Figure 1: Example of a multilevel 2-D geometry. Explicit geometry (left) and description in terms of cells and pins (right).

   (a) Main assembly level that will be filled with cell and to which are associated the boundary conditions (see Figure 2).

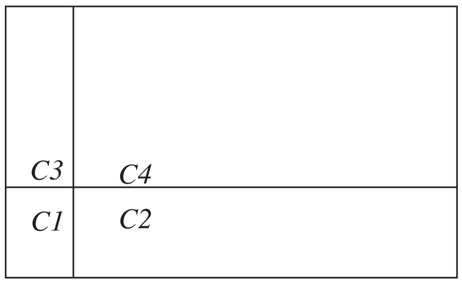

Figure 2: Global 2-D geometry.

   (b) Intermediate level defining the cells that will be used to fill the assembly (see Figure 3). In the current version of `NXT:` the cells used to define these assemblies cannot themselves contain cell sub-assemblies.

   (c) Optional upper level that corresponds to the cell geometries added using the `CLUSTER` keyword (see Figure 3). The cell geometry on this level covers the region defined by lower level geometries.

   In the case where the global geometry is not an assembly (a sigle cell) it is automatically generated by `NXT:` based on the cell properties.

2. Limited to 2-D and 3-D geometries having a rectangular boundary (see Figure 5). This means that each geometry in 2-D must be located between 2 lines parallel to the y axis and two lines parallel to the x axis.

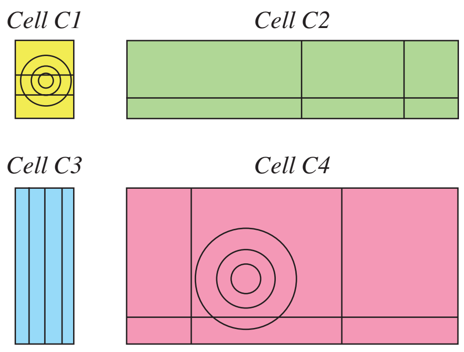

Figure 3: Cells inserted in global geometry.

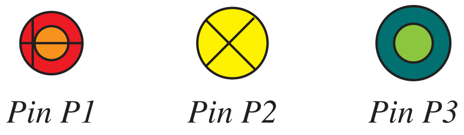

Figure 4: Pins superimposed on cells

In 3-D, the geometry must be locates between 2 plane normal to the x axis, 2 plane normal to the y axis and finally 2 plane normal to the z axis .

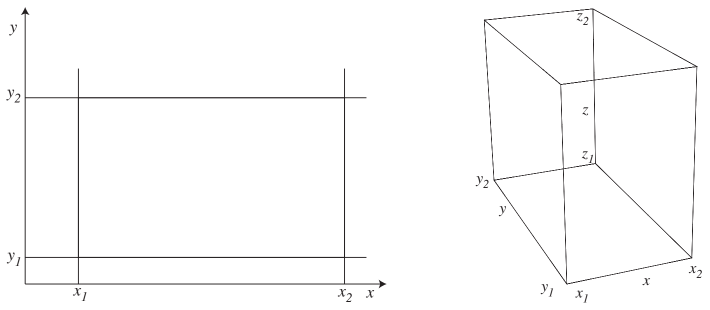

Figure 5: Geometry with 2-D (left) and 3-D (right) rectangular boundaries.

3. The boundary conditions are applied on the external faces of the geometry with three exceptions:

   - `SYME`, a mirror symmetry applied at the center of the cell (single cell geometry) or at the center of the cells closest to the direction specified for an assembly geometry (see Figure 6).
   - `SSME`, a mirror symmetry applied at the boundary of the cell or the assembly.
   - `DIAG` which is applied on a $x = y$ diagonal passing through the center of the cell or the assembly (see Figure 6). The combination `X-` and `Y+` means that the regions under and to the right of the diagonal are tracked while for the combination `X+` and `Y-` the regions over and to the left of the diagonal are tracked.

The external surfaces associated with a geometry are also generated assuming that the pin surfaces associated with an external boundary cover completely the external surfaces associated with the cell geometries.

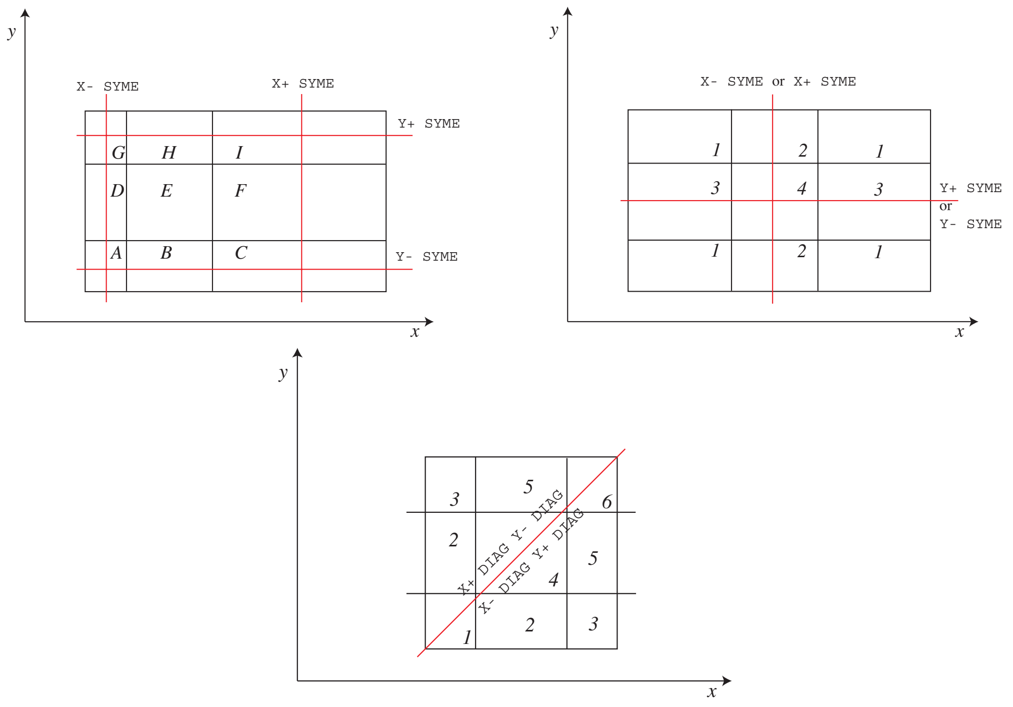

Figure 6: Central cell symmetry (`SYME`) as applied for assembly of cells (top left) or for individual cells (top right) and diagonal symmetry (`DIAG`) for square cell or assembly (bottom).

Similarly, the cell surfaces associated with an external boundary are superimposed over the assembly surfaces.

Typical 2-D and 3-D geometries that can be processed by the `NXT:` module are illustrated in Figure 7. For the 2-D geometry, each color is associated with a diffetent region number. One can immediatly see the hierarchy of region numbering with namely the pins hiding the cells hiding the assembly level. This hierarchy is also used for 3-D geometry both for the region and external surface numbering. However, the cells and pins surfaces not in contact with an external surface are never considered.

2.1 Assembly level

As already mentionned, at the main assembly level the geometry must have a purely Cartesian structure. This level can be built directly by the user using the `CAR2D` and `CAR3D` geometry types and filling each region in the geometry with cells as proposed in the following 2-D and 3-D examples

```
Geo2DA := GEO: :: CAR2D 1 2
  CELL  C1 C2
  X- REFL X+ REFL Y- REFL Y+ REFL
  ::: C1 := GEO: CAR2D 2 1
    MESHX <<X1>> <<X2>> <<X3>> MESHY <<Y1>> <<Y2>>
    MIX 1 2 ;
  ::: C2 := GEO: CARCEL 2 1 1
    MESHX <<X1>> <<X3>> MESHY <<Y2>> <<Y3>>
    RADIUS 0.0 <<R1>> <<R2>>
    MIX 3 4 5 ;
;
```

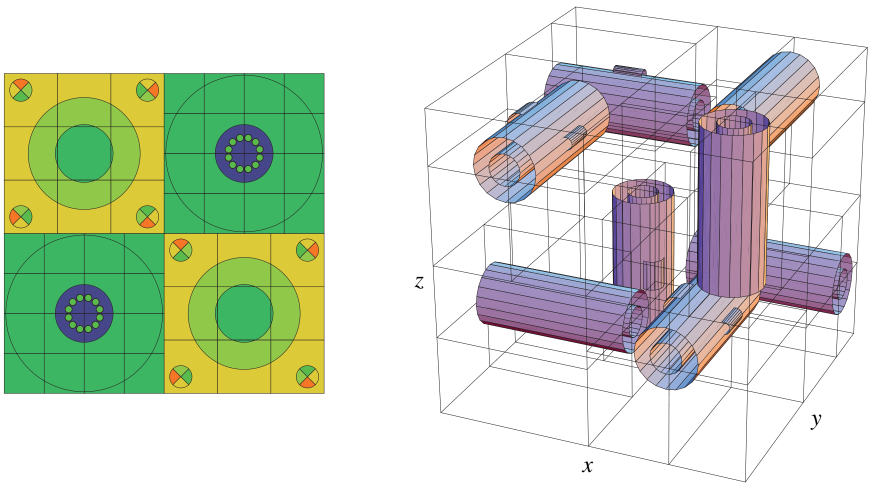

Figure 7: Typical 2-D (left) and 3-D (right) geometries that can be processed by the `NXT:` module.

```
Geo3DA := GEO: :: CAR3D 2 2 2
  CELL  C1 C2 C3 C4 C5 C6 C7 C8
  X- REFL X+ REFL Y- REFL Y+ REFL Z- REFL Z+ REFL
  ::: C1 := GEO: CAR3D 1 1 1
    MESHX <<X1>> <<X2>> MESHY <<Y1>> <<Y2>> MESHZ <<Z1>> <<Z2>>
    MIX 1 ;
  ::: C2 := GEO: CARCELZ 1 1 1 1
    MESHX <<X2>> <<X3>> MESHY <<Y1>> <<Y2>> MESHZ <<Z1>> <<Z2>>
    RADIUS 0.0 <<R1>>
    MIX 12 2 ;
  [ ... ]
;
```

The exact dimensions of the assembly are extracted from the dimensions of the individual cells, which must be defined in such a way as to form a uniform Cartesian mesh in the X, Y and Z directions.

In the case where the assembly is made up of a single cell, as in

```
Geo2DC := GEO: :: CARCEL 2 1 1
  X- REFL X+ REFL Y- REFL Y+ REFL
  MESHX <<X1>> <<X3>> MESHY <<Y2>> <<Y3>>
  RADIUS 0.0 <<R1>> <<R2>>
  MIX 3 4 5
;
Geo3DC := GEO: :: CAR3D 2 2 2
  X- REFL X+ REFL Y- REFL Y+ REFL Z- REFL Z+ REFL
  MESHX <<X1>> <<X2>> <<X3>>
  MESHY <<Y1>> <<Y2>> <<Y3>>
  MESHZ <<Z1>> <<Z2>> <<Z3>>
  MIX 1 2 3 4 5 6 7 8
;
```

the `NXT:` module will automatically generate, using the information associated with the cell, the required `CAR2D` or `CAR3D` assembly that will be filled with the cell specified. For the 2-D and 3-D cases above, `NXT:` will work as if it was seeing the following equivalent geometries:

```
Geo2DCR := GEO: :: CAR2D 1 1
  X- REFL X+ REFL Y- REFL Y+ REFL
  CELL Geo2DC
  ::: Geo2DC := GEO: CARCEL 2 1 1
    MESHX <<X1>> <<X3>> MESHY <<Y2>> <<Y3>>
    RADIUS 0.0 <<R1>> <<R2>>
    MIX 3 4 5 ;
;
Geo3DCR := GEO: :: CAR3D 1 1 1
  X- REFL X+ REFL Y- REFL Y+ REFL Z- REFL Z+ REFL
  CELL Geo3DC
  ::: Geo3DC := GEO: CAR3D 2 2 2
    MESHX <<X1>> <<X2>> <<X3>>
    MESHY <<Y1>> <<Y2>> <<Y3>>
    MESHZ <<Z1>> <<Z2>> <<Z3>>
    MIX 1 2 3 4 5 6 7 8 ;
;
```

A consequence of this reconstruction feature of `NXT:` is that in this case the cell geometry options are limited to `CAR2D` or `CARCEL` in 2-D and to `CAR3D`, `CARCELX`, `CARCELY` or `CARCELZ` in 3-D.

This procedure is straightforward for the cases where only the `REFL`, `VOID` and `ALBE` boundary conditions are used. If the first level geometry is a cell, the reconstruction process described above remains valid and the symmetry is applied directly to the cell of interest. On the other hand, if the first main level geometry is an assembly, two problem arise when the `SYME`, `SYMM`, and `DIAG` boundary are used:

1. These symmetries are used to simplify the input file and the GEOMETRY data structure thereby created. This means that the GEOMETRY data structure does not explicitly represent the geometry that will be treated.

2. The final geometry provided in the GEOMETRY data structure does not necessarily have the purely Cartesian mesh required by the `NXT:` module (primarly when diagonal boundary conditions are selected but also for the `SYME` geometry which may be cutting cylindrical regions in half).

In order to illustrate this problem let us consider the 2-D examples of Figure 8 generated using the following instructions in DRAGON:

```
Geo2DADSY := GEO: :: CAR2D 2 2
  CELL  C1 C2 C3
  X- DIAG X+ REFL Y- SYME Y+ DIAG
  ::: C1 := GEO: CAR2D 1 1
    MESHX <<X1>> <<X2>> MESHY <<X1>> <<X2>>
    MIX 1 ;
  ::: C2 := GEO: CARCEL 2 1 1
    MESHX <<X1>> <<X3>> MESHY <<X1>> <<X2>>
    RADIUS 0.0 <<R1>> <<R2>>
    MIX 3 4 5   ;
  ::: C3 := GEO: CARCEL 2 2 2
    MESHX <<X1>> <<X2>> <<X3>> MESHY <<X1>> <<X2>> <<X3>>
    RADIUS 0.0 <<R1>> <<R2>>
    MIX 6 7 8 9 10 11 9 10 11 12 13 14 ;
;
Geo2DADSS := GEO: :: CAR2D 2 2
  CELL  C1 C2 C3
  X- DIAG X+ REFL Y- SYME Y+ DIAG
  ::: C1 := GEO: CAR2D 1 1
```

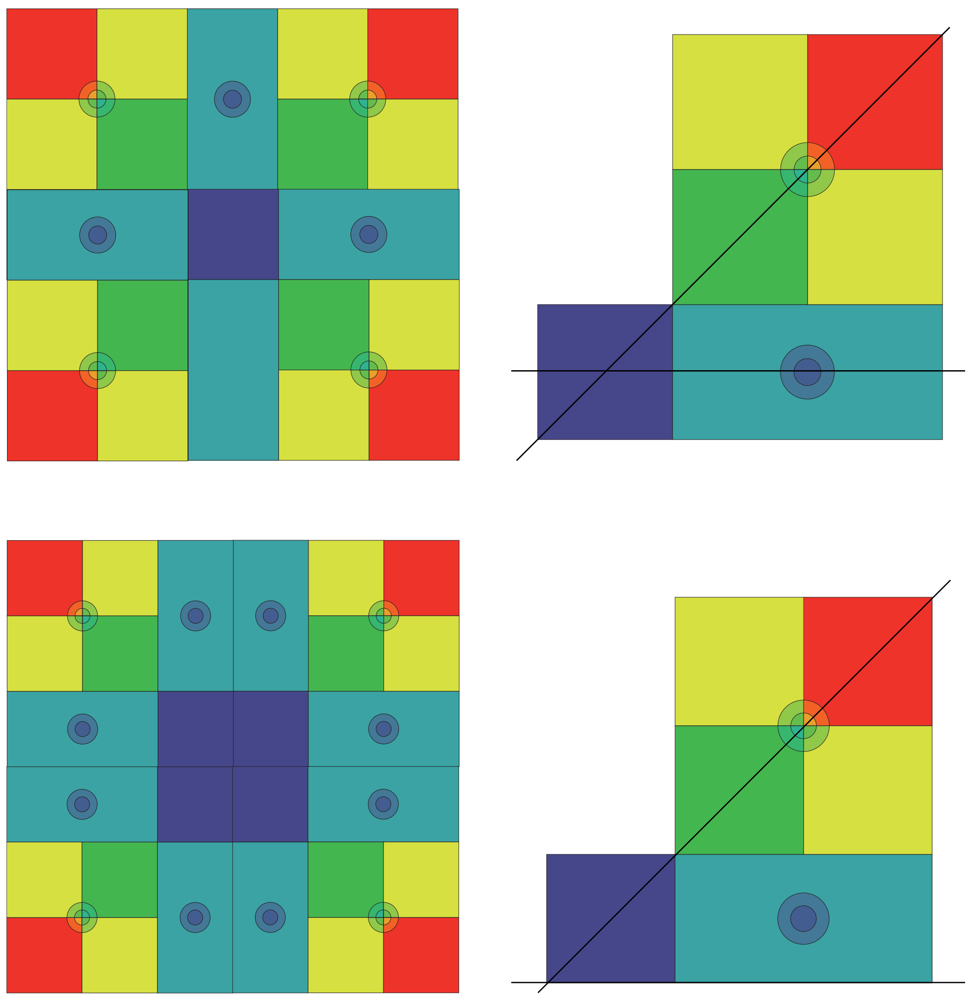

Figure 8: Example of a 2-D geometry with diagonal $x = y$ and mirror symmetry with respect to the X and Y axis passing through the center (top figures with `SYME` symmetry) or located at the bottom (bottom figure with `SSYM` symmetry) of the geometry. The figure on the left is the exact geometry and that to the right is the geometry provided as input to DRAGON.

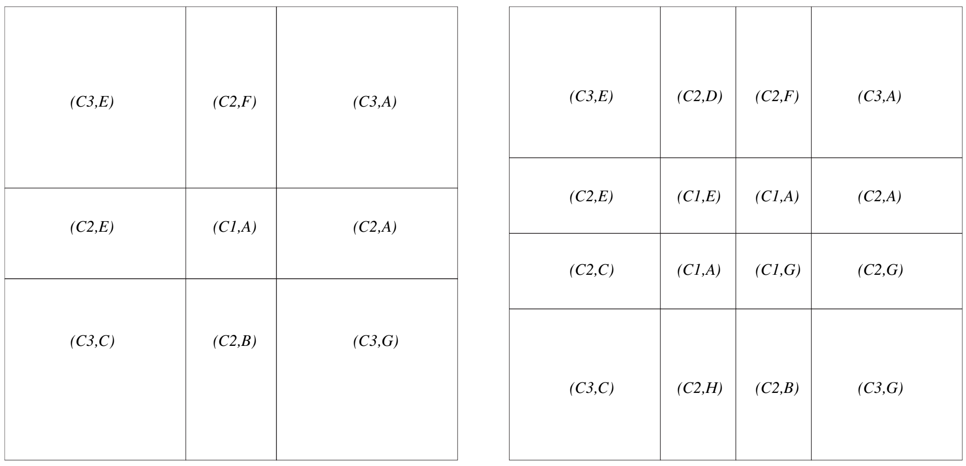

Figure 9: Unfolded assembly with cell contents and `TURN` option for `Geo2DADSY` (left) and `Geo2DADSS` (right).

```
        MESHX <<X1>> <<X2>> MESHY <<X1>> <<X2>>
        MIX 1 ;
      ::: C2 := GEO: CARCEL 2 1 1
        MESHX <<X1>> <<X3>> MESHY <<X1>> <<X2>>
        RADIUS 0.0 <<R1>> <<R2>>
        MIX 3 4 5   ;
      ::: C3 := GEO: CARCEL 2 2 2
        MESHX <<X1>> <<X2>> <<X3>> MESHY <<X1>> <<X2>> <<X3>>
        RADIUS 0.0 <<R1>> <<R2>>
        MIX 6 7 8 9 10 11 9 10 11 12 13 14 ;
;
```

For such geometries, the `NXT:` module first unfolds the geometry according to the instruction, generating a 3×3 assembly for geometry `Geo2DADSY` and a 4×4 assembly for geometry `Geo2DADSS` and then fills these assemblies with the adequatly rotated and reflected cells (see Figure 9) where the rotation indices (`A` to `H`) are defined in the DRAGON users manual.[?] It also test if each cells possesses the intrinsic symmetry (cell `C1` for example) required by the geometry description.

2.2 Cell level

Two types of cells can be used to fill 2-D assemblies:

- `CAR2D`, that describes a rectangular cell.
- `CARCEL`, a rectangular cell that contains an embedded set of concentric annular regions. The center of these annular region coincides with the center of the cell unless specified otherwise (keyword `OFFCENTER`). In addition, the annular regions must all be located within the boundary of the rectangle.

The explicit rectangular submesh is provided using the `MESHX` and `MESHY` keywords for both the `CAR2D` and `CARCEL`. In addition the explicit radial submesh for a `CARCEL` geometry is specified using the `RADIUS` keyword. Implicit submeshing is also possible at the tracking level with the use of the keywords `SPLITX` (X submesh), `SPLITY` (Y submesh) and `SPLITR` (radial submesh).

The four types of cells that can be used to fill 3-D assemblies are:

- `CAR3D`, that describes a rectangular parallelipiped.

- CARCELX, that describes a rectangular parallelipiped that contains an embedded set of concentric cylindrical regions parallel to the X axis that extend over the full X range of the cell.
- CARCELY, that describes a rectangular parallelipiped that contains an embedded set of concentric cylindrical regions parallel to the Y axis that extend over the full Y range of the cell.
- CARCELZ, that describes a rectangular parallelipiped that contains an embedded set of concentric cylindrical regions parallel to the Z axis that extend over the full Z range of the cell.

The explicit rectangular submesh is provided using the MESHX, MESHY and MESHY keywords for both these geometries while the radial submesh for the last three geometries is specified using the RADIUS keyword. As for 2-D cells, implicit submeshing is also possible at the tracking level with the use of the keywords SPLITX (X submesh), SPLITY (Y submesh), SPLITZ (Z submesh) and SPLITR (radial submesh).

2.3 Pin level

The only of pin geometry permitted for 2-D cells is a TUBE that describes a cell with circular boundaries. In 3-D, TUBEX, TUBEY or TUBEZ pin geometry that represents cylindrical cells in the X, Y and Z directions are generally permitted with the following restrictions:

- The direction of all the pins in a cell must be identical.
- The cylindrical regions associated with a cell, if any, must be identical to that of the pins that are inserted in this cell.

Finally all the pins (both in 2-D and 3-D) must be fully enclosed in the cell.

The radial mesh associated with a tube and specified using the RADIUS can be refined implicitely at the tracking level using the SPLITR option. Similarly cylinders in the X, Y and Z directions will be defined using the (MESHX, SPLITX), (MESHY, SPLITY) and (MESHZ, SPLITZ) pairs of keywords. Note that is also possible to superimposed a Cartesian mesh on the radial mesh using the following options

- MESHX, SPLITX, MESHY and SPLITY for TUBE geometries.
- MESHY, SPLITY, MESHZ and SPLITZ for TUBEX geometries.
- MESHZ, SPLITZ, MESHX and SPLITX for TUBEY geometries.
- MESHX, SPLITX, MESHY and SPLITY for TUBEZ geometries.

3 GEOMETRY ANALYSIS

Here, we will describe successively how DRAGON identifies the region and surfaces associated with a multilevel geometry (an assembly) similar to those described in Section 2. One may recall that such assemblies are always based the superposition of geometries containing cells (keyword `CELL`), pin clusters (keyword `PIN`) or mixtures (keyword `MIX`). Here we will consider the case of first pure geometries (Section 3.1) which are defined as a geometry that is filled only with mixtures. These can be defined using the following keywords:

- `CAR2D` for a 2-D Cartesian cell.
- `CAR3D` for a 3-D Cartesian cell.
- `CARCEL` for a 2-D Cartesian geometry with embedded annular regions,
- `CARCELX` for a 3-D Cartesian geometry with embedded cylindrical regions directed along the X axis.
- `CARCELY` for a 3-D Cartesian geometry with embedded cylindrical regions directed along the Y axis.
- `CARCELZ` for a 3-D Cartesian geometry with embedded cylindrical regions directed along the Z axis.
- `TUBE` for an annular 2-D geometry which may contain an embedded 2-D Cartesian mesh.
- `TUBEX` for an X directed 3-D cylindrical geometry which may contain an embedded 3-D Cartesian mesh.
- `TUBEY` for a Y directed 3-D cylindrical geometry which may contain an embedded 3-D Cartesian mesh
- `TUBEZ` for a Z directed 3-D cylindrical geometry which may contain an embedded 3-D Cartesian mesh

In Section 3.2 we will study the effect of pin clusters superposition on these geometries. Finally in Section 3.3 we will study how these pure geometry are combined to create an assembly.

3.1 Region and surface identification for pure geometries

3.1.1 Pure Cartesian cells

Pure Cartesian cells (see Figure 10) in 2-D and 3-D may be created using the following DRAGON input data structures

- 2-D Cartesian cell

  `Geometry := GEO: CAR2D` $n_x\ n_y$
  `MESHX` $(x_i,\ i=0,n_x)$
  `MESHY` $(y_j,\ j=0,n_y)$
  `MIX` $((m_{i+n_x(j-1)},\ i=1,n_x),\ j=1,n_y)$ `;`

- 3-D Cartesian cell

  `Geometry := GEO: CAR3D` $n_x\ n_y\ n_z$
  `MESHX` $(x_i,\ i=0,n_x)$
  `MESHY` $(y_j,\ j=0,n_y)$
  `MESHZ` $(z_k,\ k=0,n_z)$
  `MIX` $(((m_{i+n_x(j-1+n_y(k-1))},\ i=1,n_x),\ j=1,n_y)\ k=1,n_z)$ `;`

where x_i, y_j and z_k are the local position of the lines (in 2-D) or planes (3-D) defining the subregions that can be found in the cell along the X, Y and Z direction, $n_x + 1$, $n_y + 1$ and $n_z + 1$ representing the number of such lines or planes in each direction. Note that for simplicity we will assume that the 2-D geometry is a 3-D geometry with $n_z = 1$ and

$$\begin{aligned} z_0 &= 0 \\ z_1 &= 1 \end{aligned}$$

even though the 3-D extension is in principle from $-\infty$ to ∞. This choice is in fact dictated by the fact that 2-D volumes are in fact identical to 3-D volumes of regions with a height $z_1 - z_0 = 1$.

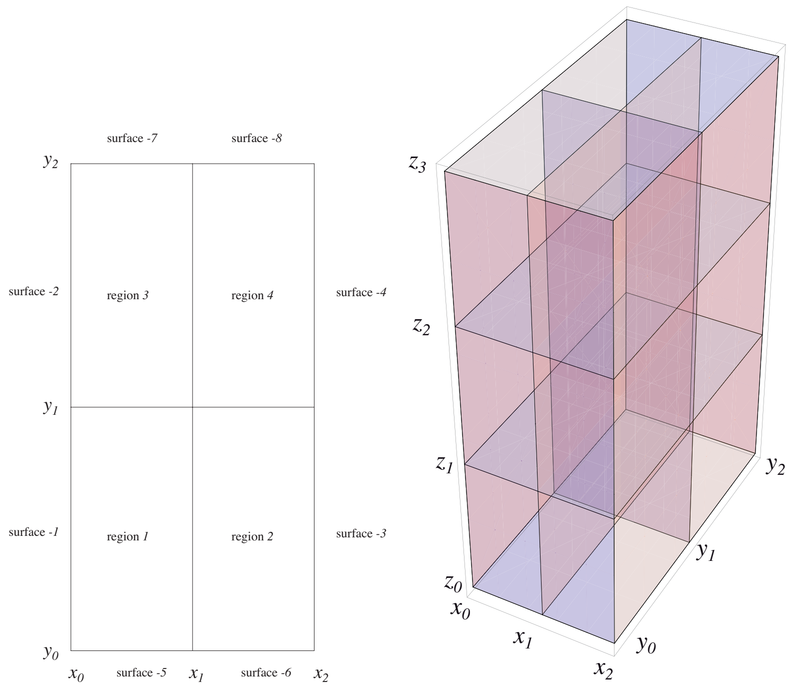

Figure 10: Example of pure Cartesian cells in 2-D (left) and 3-D (right).

Such a Cartesian cell should contain

$$N_r = n_x n_y n_z$$

subregions, the mixture associated with these subregions being provided by m_l. In DRAGON, each subregion l with volume V_l (see Appendix A.1) of a cell can be associated with a position (i, j, k) in the Cartesian mesh according to (see Figure 10 for an example in 2-D):

$$l = i + n_x \left(j - 1 + n_y(k - 1)\right) \tag{3.1}$$

$$V_l = (x_i - x_{i-1})(y_j - y_{j-1})(z_k - z_{k-1}) \tag{3.2}$$

this notation being similar to that used in the input data structure to associate a mixture with a subregion.

The numbering ℓ (negative values) of the surfaces with area S_ℓ follows the following algorithm:

1. $n_{x,s} = n_y n_z$ surfaces at $x = x_0$

$$\ell = -(j + n_y(k - 1)) \tag{3.3}$$

$$S_\ell = (y_j - y_{j-1})(z_k - z_{k-1}) \tag{3.4}$$

2. $n_{x,s}$ surfaces at $x = x_{n_x}$

$$\ell = -(j + n_y(k-1)) - n_{x,s} \tag{3.5}$$
$$S_\ell = (y_j - y_{j-1})(z_k - z_{k-1}) \tag{3.6}$$

3. $n_{y,s} = n_z n_x$ surfaces at $y = y_0$

$$\ell = -(k + n_z(i-1)) - 2n_{x,s} \tag{3.7}$$
$$S_\ell = (z_k - z_{k-1})(x_i - x_{i-1}) \tag{3.8}$$

4. $n_{y,s}$ surfaces at $y = y_{n_y}$

$$\ell = -(k + n_z(i-1)) - 2n_{x,s} - n_{y,s} \tag{3.9}$$
$$S_\ell = (z_k - z_{k-1})(x_i - x_{i-1}) \tag{3.10}$$

5. $n_{z,s} = n_x n_y$ surfaces at $z = z_0$

$$\ell = -(i + n_x(j-1)) - 2n_{x,s} - 2n_{y,s} \tag{3.11}$$
$$S_\ell = (x_i - x_{i-1})(y_j - y_{j-1}) \tag{3.12}$$

6. $n_{z,s}$ surfaces at $z = z_{n_z}$

$$\ell = -(i + n_x(j-1)) - 2n_{x,s} - 2n_{y,s} - n_{z,s} \tag{3.13}$$
$$S_\ell = (x_i - x_{i-1})(y_j - y_{j-1}) \tag{3.14}$$

for a total of

$$\begin{aligned} N_S &= 2n_{x,s} + 2n_{y,s} + 2n_{z,s} \\ &= 2n_y n_z + 2n_z n_x + 2n_x n_y \end{aligned} \tag{3.15}$$

surfaces.

3.1.2 Pure annular and cylindrical cells

Pure annular (2-D) and cylindrical (3-D) cells (see Figure 11) may be created using the following DRAGON input data structures

- 2-D annular cell

  `Geometry := GEO: TUBE` n_r
  `MESHR` $(r_g,\ g = 0, n_r)$
  `MIX` $(m_g,\ g = 1, n_r)$ `;`

  containing $N_r = n_r$ subregions.

- 3-D X directed cylinders cell

  `Geometry := GEO: TUBEX` $n_r\ n_x$
  `MESHR` $(r_g,\ g = 0, n_r)$
  `MESHX` $(x_i,\ i = 0, n_x)$
  `MIX` $((m_{g+n_r(i-1)},\ g = 1, n_r),\ i = 1, n_x)$ `;`

that contains $N_r = n_r n_x$ subregions.

- 3-D Y directed cylinders cell

    ```
    Geometry := GEO: TUBEY $n_r$ $n_y$
    MESHR ($r_g,\ g = 0, n_r$)
    MESHY ($y_j,\ j = 0, n_y$)
    MIX (($m_{g+n_r(j-1)},\ g = 1, n_r$), $j = 1, n_y$) ;
    ```

  that contains $N_r = n_r n_y$ subregions.

- 3-D Z directed cylinders cell

    ```
    Geometry := GEO: TUBEZ $n_r$ $n_z$
    MESHR ($r_g,\ g = 0, n_r$)
    MESHZ ($z_k,\ k = 0, n_z$)
    MIX (($m_{g+n_r(k-1)},\ g = 1, n_r$), $k = 1, n_z$) ;
    ```

  that contains $N_r = n_r n_z$ subregions.

where r_i is the radius of the n_r concentric annular or cylindrical regions in the cell such that $r_0 = 0.0$. For 3-D cylinders, x_i, y_j and z_k are respectively the local position of the planes normal to the cylindrical axis defining the extent of the n_x, n_y or n_z cylinders. Note that for simplicity we will assume that the 2-D geometry is a 3-D geometry with $n_z = 1$ and an extension

$$z_0 = 0$$
$$z_1 = 1$$

even though the 3-D extension is in principle from $-\infty$ to ∞. The mixture associated with these subregions are provided provided by m_l.

For the 2-D annular cells, each subregion l with volume V_l can be associated with a position g in the radial mesh according to (see Appendix A.2):

$$l = g \tag{3.16}$$
$$V_g = \pi(r_g - r_{g-1})^2 \tag{3.17}$$

For 3-D cylindrical cells we will use

- X directed cylinders

$$l = g + n_r(i-1) \tag{3.18}$$
$$V_l = \pi(r_g - r_{g-1})^2(x_i - x_{i-1}) \tag{3.19}$$

- Y directed cylinders

$$l = g + n_r(j-1) \tag{3.20}$$
$$V_l = \pi(r_g - r_{g-1})^2(y_j - y_{j-1}) \tag{3.21}$$

- X directed cylinders

$$l = g + n_r(k-1) \tag{3.22}$$
$$V_l = \pi(r_g - r_{g-1})^2(z_k - z_{k-1}) \tag{3.23}$$

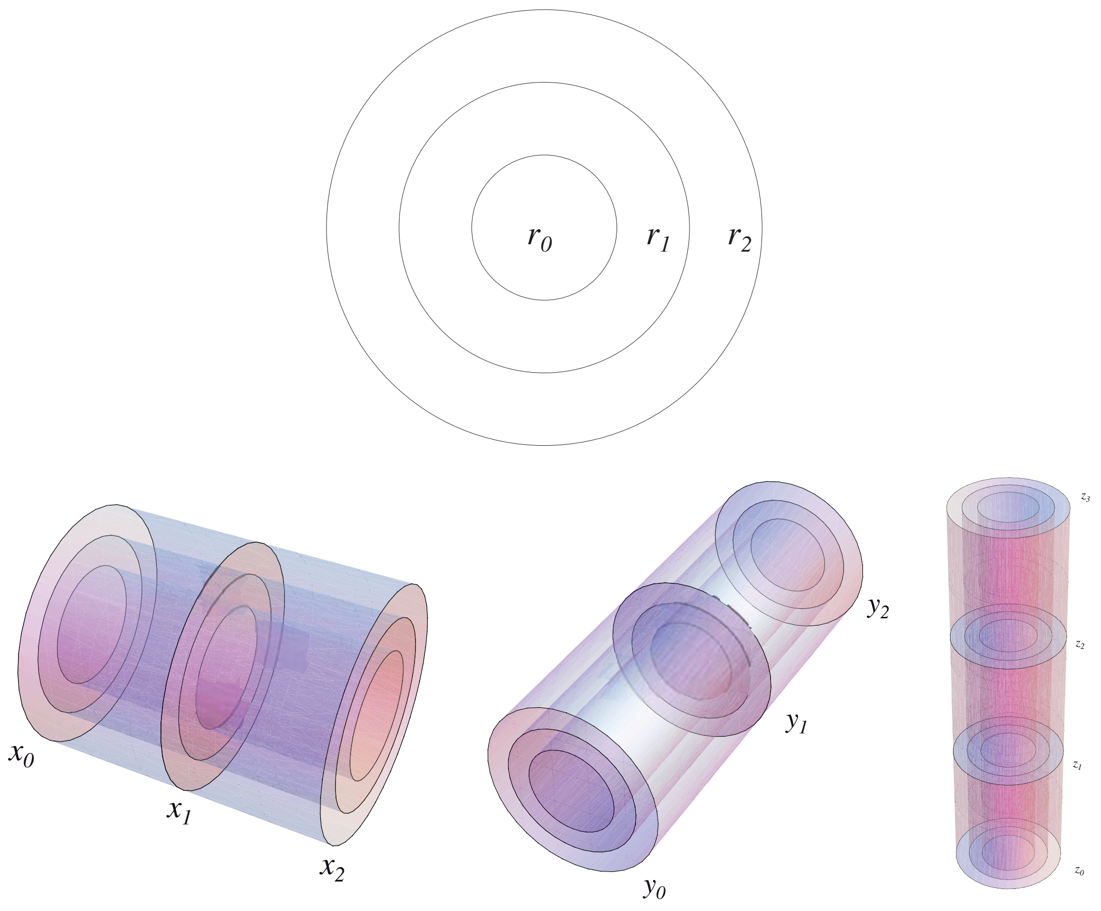

Figure 11: Examples of `TUBE` (top), `TUBEX` (bottom left), `TUBEY` (bottom center) and `TUBEZ` (bottom right) geometries.

The volume ordering being consistent with the mixture ordering.

In 2-D, a single outer surface $\ell = -1$ will be considered with area

$$S_1 = 2\pi r_{n_r} \tag{3.24}$$

while for 3-D cylinders we will use

- X directed cylinders
  1. n_r surfaces at $x = x_0$
  $$\ell = -g \tag{3.25}$$
  $$S_\ell = \pi(r_g - r_{g-1})^2 \tag{3.26}$$
  2. n_r surfaces at $x = x_{n_x}$
  $$\ell = -g - n_r \tag{3.27}$$
  $$S_\ell = \pi(r_g - r_{g-1})^2 \tag{3.28}$$
  3. n_x radial surfaces (in the $Y - Z$ plane) ar $r = r_{n_r}$
  $$\ell = -i - 2n_r \tag{3.29}$$
  $$S_\ell = 2\pi r_{n_r}(x_i - x_{i-1}) \tag{3.30}$$

  for a total of $N_S = 2n_r + n_x$ surfaces.
- Y directed cylinders
  1. n_r surfaces at $y = y_0$
  $$\ell = -g \tag{3.31}$$
  $$S_\ell = \pi(r_g - r_{g-1})^2 \tag{3.32}$$
  2. n_r surfaces at $y = y_{n_y}$
  $$\ell = -g - n_r \tag{3.33}$$
  $$S_\ell = \pi(r_g - r_{g-1})^2 \tag{3.34}$$
  3. n_y radial surfaces (in the $Z - X$ plane) ar $r = r_{n_r}$
  $$\ell = -j - 2n_r \tag{3.35}$$
  $$S_\ell = 2\pi r_{n_r}(y_j - y_{j-1}) \tag{3.36}$$

  for a total of $N_{S,Y} = 2n_r + n_y$ surfaces.
- Z directed cylinders
  1. n_r surfaces at $z = z_0$
  $$\ell = -g \tag{3.37}$$
  $$S_\ell = \pi(r_g - r_{g-1})^2 \tag{3.38}$$
  2. n_r surfaces at $z = z_{n_z}$
  $$\ell = -g - n_r \tag{3.39}$$
  $$S_\ell = \pi(r_g - r_{g-1})^2 \tag{3.40}$$

3. n_z radial surfaces (in the $X-Y$ plane) ar $r=r_{n_r}$

$$\ell = -k - 2n_r \tag{3.41}$$

$$S_\ell = 2\pi r_{n_r}(z_k - z_{k-1}) \tag{3.42}$$

for a total of $N_{S,Z}=2n_r+n_z$ surfaces.

3.1.3 Cartesian cell with annular subregions

Cartesian 2-D cells containing annular subregions and Cartesian 3-D cells with cylindrical (3-D) subregions (see Figure 12) may be created using the following DRAGON input data structures

- 2-D Cartesian cell with annular mesh

  `Geometry := GEO: CARCEL` $n_r\ n_x\ n_y$
  `MESHR` $(r_g,\ g=0,n_r)$
  `MESHX` $(x_i,\ i=0,n_x)$
  `MESHY` $(y_j,\ j=0,n_y)$
  `MIX` $(((m_{g+(n_r+1)(i-1+n_x(j-1))},\ g=1,n_r+1),\ i=1,n_x),\ j=1,n_y)$ `;`

  containing a maximum of $N_r=(n_r+1)n_xn_y$ subregions. Note that the a value of (g,i,j) with $g\le n_r$ correspond to the part of an annular ring located between r_g and r_{g-1} that is included in the Cartesian region identified by (i,j) while a set (g,i,j) with $g=n_r+1$ corresponds to the part of a Cartesian region identified by (i,j) completely outside the annular ring of radius r_{n_r}. Some of these regions may not exists when $g\le n_r$ since the intersection of the ring and the Cartesian region may vanish. In this case, the a mixture number is still required on input even if it will not be used in the cell description.

- 3-D Cartesian cells containing X directed cylinders

  `Geometry := GEO: CARCELX` $n_r\ n_x\ n_y\ n_z$
  `MESHR` $(r_g,\ g=0,n_r)$
  `MESHX` $(x_i,\ i=0,n_x)$
  `MESHY` $(y_j,\ j=0,n_y)$
  `MESHZ` $(z_k,\ k=0,n_z)$
  `MIX` $((((m_{g+(n_r+1)(j-1+n_y(k-1+n_z(i-1)))},\ g=1,n_r{+}1),\ j=1,n_y),\ k=1,n_z),\ i=1,n_x)$ `;`

  that contains a maximum of $N_r=(n_r+1)n_xn_yn_z$ subregions. Note that the a value of (g,i,j,k) with $g\le n_r$ correspond to the part of an cylindrical ring located between r_g and r_{g-1} that is included in the Cartesian region identified by (i,j,k) while a set (g,i,j,k) with $g=n_r+1$ corresponds to the part of the Cartesian region identified by (i,j,k) completely outside the cylindrical ring of radius r_{n_r}. Some of these regions may not exists when $g\le n_r$ since the intersection of the ring and the Cartesian region may vanish. In this case, the a mixture number is still required on input even if it will not be used in the cell description.

- 3-D Cartesian cells containing Y directed cylinders

  `Geometry := GEO: CARCELY` $n_r\ n_x\ n_y\ n_z$
  `MESHR` $(r_g,\ g=0,n_r)$
  `MESHX` $(x_i,\ i=0,n_x)$
  `MESHY` $(y_j,\ j=0,n_y)$
  `MESHZ` $(z_k,\ k=0,n_z)$
  `MIX` $((((m_{g+(n_r+1)(k-1+n_z(i-1+n_x(j-1)))},\ g=1,n_r{+}1),\ k=1,n_z),\ i=1,n_z),\ j=1,n_y)$ `;`

that contains a maximum of $N_r = (n_r + 1)n_x n_y n_z$ subregions. The same comment on mixtures input as that provided for Cartesian cells with X directed cylinders remains valid here.

- 3-D Cartesian cells containing Z directed cylinders

  ```
  Geometry := GEO: CARCELZ $n_r$ $n_x$ $n_y$ $n_z$
  MESHR ($r_g,\ g = 0, n_r$)
  MESHX ($x_i,\ i = 0, n_x$)
  MESHY ($y_j,\ j = 0, n_y$)
  MESHZ ($z_k,\ k = 0, n_z$)
  MIX (((($m_{g+(n_r+1)(i-1+n_x(j-1+n_y(k-1)))}$, $g = 1, n_r + 1$), $i = 1, n_x$), $j = 1, n_y$), $k = 1, n_z$)
  ;
  ```

  that contains a maximum of $N_r = (n_r + 1)n_x n_y n_z$ subregions. Again some mixtures may not be used in the calculations as described for Cartesian cells with X directed cylinders.

Here r_i is the again radius of the n_r concentric annular or cylindrical regions in the cell such that $r_0 = 0.0$. For 3-D cylinders, x_i, y_j and z_k are respectively the local position of the planes normal to the cylindrical axis defining the extent of the n_x, n_y or n_z cylinders. Note that for simplicity we will again assume that the 2-D geometry is a 3-D geometry with $n_z = 1$ and $z_0 = 0$ and $z_1 = 1$ even though the 3-D extension is in principle from $-\infty$ to ∞.

For the 2-D CARCEL geometries, each possible subregion l of the cell can be associated with a position (g, i, j) according to (see Appendix A.2):

$$l = g + (n_r + 1)(i - 1 + n_x(j - 1)) \tag{3.43}$$

for a maximum of N_r regions. The volume V_l of these regions can be evaluated using the procedures described in Appendix A. Note that some of these volumes may turn out to vanish since not all the radial regions will have an intersection with each of the Cartesian region. Thus, the final region identification will correspond to $\{l_c\}$, a compressed version of the set $\{l\}$ that contains N_V terms where the elements with vanishing volume have been removed:

```
lc=0
DO l=1,N_{r}
  IF(V_{l} .NE. 0) THEN
    lc=lc+1
    V_{lc}=V_{l}
    m_{lc}=m_{l}
  ENDIF
ENDDO
N_{V}=lc
```

where the mixture identification vector is compressed in a fashion similar to V_l.

For the 3-D CARCELX, CARCELY and CARCELZ geometries, a similar process is used where each possible subregion l of the cell can be now associated with a position (g, i, j, k). Here we use:

$$l = g + (n_r + 1)(i - 1 + n_x(j - 1 + n_y(k - 1))) \tag{3.44}$$

for a maximum of N_r regions. The global ordering here is different from that used for the mixture ordering. The volumes V_l can again be evaluated using the procedures described in Appendix A. As for the CARCEL geometry, some of the volumes may turn out to vanish. Thus, the final region identification will correspond to $\{l_c\}$, a compressed version of the set $\{l\}$ that contains N_V terms where the elements with vanishing volume can be removed using the procedure described above.

In 2-D, the outer surfaces ℓ are numbered according to the following procedure

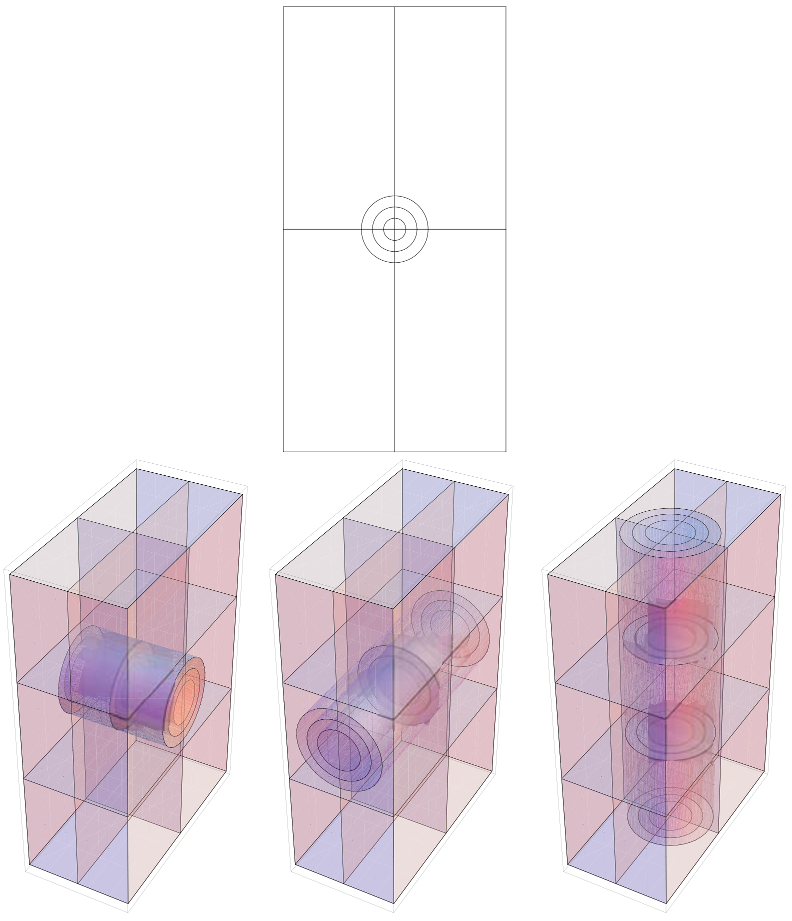

Figure 12: Examples of CARCEL (top), CARCELX (bottom left), CARCELY (bottom center) and CARCELZ (bottom right) geometries.

1. n_y surfaces at $x = x_0$

$$\ell = -j \tag{3.45}$$
$$S_\ell = (y_j - y_{j-1}) \tag{3.46}$$

2. n_y surfaces at $x = x_{n_x}$

$$\ell = -j - n_y \tag{3.47}$$
$$S_\ell = (y_j - y_{j-1}) \tag{3.48}$$

3. n_x surfaces at $y = y_0$

$$\ell = -i - 2n_y \tag{3.49}$$
$$S_\ell = (x_i - x_{i-1}) \tag{3.50}$$

4. n_x surfaces at $y = y_{n_y}$

$$\ell = -i - 2n_y - n_x \tag{3.51}$$
$$S_\ell = (x_i - x_{i-1}) \tag{3.52}$$

for a total of

$$N_S = 2n_y + 2n_x \tag{3.53}$$

surfaces.

For 3-D Cartesian geometries with embedded cylindrical regions, the surfaces are numbered according to the following procedure

- `CARCELX`

  1. A maximum of $n_{r,x} = (n_r + 1)n_y n_z$ surfaces at $x = x_0$

     $$\ell = -g - (n_r + 1)(j - 1 + n_y(k - 1)) \tag{3.54}$$

     with S_ℓ evaluated using the procedure of Appendix A.
  2. A maximum of $n_{r,x}$ surfaces at $x = x_{n_x}$

     $$\ell = -g - (n_r + 1)(j - 1 + n_y(k - 1)) - n_{r,x} \tag{3.55}$$

     with S_ℓ evaluated using the procedure of Appendix A.
  3. $n_x n_z$ surfaces at $y = y_0$

     $$\ell = -k - n_z(i - 1) - 2n_{r,x} \tag{3.56}$$
     $$S_\ell = (x_i - x_{i-1})(z_k - z_{k-1}) \tag{3.57}$$

  4. $n_x n_z$ surfaces at $y = y_{n_y}$

     $$\ell = -k - n_z(i - 1) - 2n_{r,x} - n_x n_z \tag{3.58}$$
     $$S_\ell = (x_i - x_{i-1})(z_k - z_{k-1}) \tag{3.59}$$

  5. $n_x n_y$ surfaces at $z = z_0$

     $$\ell = -i - n_x(j - 1) - 2n_{r,x} - 2n_x n_z \tag{3.60}$$
     $$S_\ell = (x_i - x_{i-1})(y_j - y_{j-1}) \tag{3.61}$$

6. $n_x n_y$ surfaces at $z = z_{n_z}$

$$\ell = -i - n_x(j-1) - 2n_{r,x} - 2n_x n_z - n_x n_y \tag{3.62}$$
$$S_\ell = (x_i - x_{i-1})(y_j - y_{j-1}) \tag{3.63}$$

for a maximum of

$$N_S = 2(n_{r,x} + n_x(n_z + n_y)) \tag{3.64}$$

surfaces.

- `CARCELY`

1. $n_y n_z$ surfaces at $x = x_0$

$$\ell = -j - n_y(k-1) \tag{3.65}$$
$$S_\ell = (y_j - y_{j-1})(z_k - z_{k-1}) \tag{3.66}$$

2. $n_y n_z$ surfaces at $x = x_{n_x}$

$$\ell = -j - n_y(k-1) - n_y n_z \tag{3.67}$$
$$S_\ell = (y_j - y_{j-1})(z_k - z_{k-1}) \tag{3.68}$$

3. A maximum of $n_{r,y} = (n_r + 1)n_x n_z$ surfaces at $y = y_0$

$$\ell = -g - (n_r + 1)(k - 1 + n_z(i-1)) - 2n_y n_z \tag{3.69}$$

with S_ℓ evaluated using the procedure of Appendix A.

4. A maximum of $n_{r,y}$ surfaces at $y = y_{n_y}$

$$\ell = -g - (n_r + 1)(k - 1 + n_z(i-1)) - 2n_y n_z - n_{r,y} \tag{3.70}$$

with S_ℓ evaluated using the procedure of Appendix A.

5. $n_x n_y$ surfaces at $z = z_0$

$$\ell = -i - n_x(j-1) - 2n_{r,y} - 2n_y n_z \tag{3.71}$$
$$S_\ell = (x_i - x_{i-1})(y_j - y_{j-1}) \tag{3.72}$$

6. $n_x n_y$ surfaces at $z = z_{n_z}$

$$\ell = -i - n_x(j-1) - 2n_{r,y} - 2n_y n_z - n_x n_y \tag{3.73}$$
$$S_\ell = (x_i - x_{i-1})(y_j - y_{j-1}) \tag{3.74}$$

for a maximum of

$$N_S = 2(n_{r,x} + n_y(n_z + n_x)) \tag{3.75}$$

surfaces.

- `CARCELZ`

1. $n_y n_z$ surfaces at $x = x_0$

$$\ell = -j - n_y(k-1) \tag{3.76}$$
$$S_\ell = (y_j - y_{j-1})(z_k - z_{k-1}) \tag{3.77}$$

2. $n_y n_z$ surfaces at $x = x_{n_x}$

$$\ell = -j - n_y(k-1) - n_y n_z \tag{3.78}$$
$$S_\ell = (y_j - y_{j-1})(z_k - z_{k-1}) \tag{3.79}$$

3. $n_x n_z$ surfaces at $y = y_0$

$$\ell = -k - n_z(i-1) - 2n_y n_z \tag{3.80}$$
$$S_\ell = (x_i - x_{i-1})(z_k - z_{k-1}) \tag{3.81}$$

4. $n_x n_z$ surfaces at $y = y_{n_y}$

$$\ell = -k - n_z(i-1) - 2n_y n_z - n_x n_z \tag{3.82}$$
$$S_\ell = (x_i - x_{i-1})(z_k - z_{k-1}) \tag{3.83}$$

5. A maximum of $n_{r,z} = (n_r+1)n_x n_y$ surfaces at $z = z_0$

$$\ell = -g - (n_r+1)(i-1+n_x(j-1)) - 2n_y n_z - 2n_x n_z \tag{3.84}$$

   with S_ℓ evaluated using the procedure of Appendix A.

6. A maximum of $n_{r,z}$ surfaces at $z = z_{n_z}$

$$\ell = -g - (n_r+1)(i-1+n_x(j-1)) - 2n_y n_z - 2n_x n_z - n_{r,z} \tag{3.85}$$

   with S_ℓ evaluated using the procedure of Appendix A.

The maximum number of surfaces in this case is

$$N_S = 2(n_{r,x} + n_z(n_x + n_y)) \tag{3.86}$$

Again some of these surfaces will vanish and the final index associated with the surfaces will be compressed using a procedure similar to that used for the volumes.

3.1.4 Annular cell with Cartesian subregions

Annular (2-D) and cylindrical (3-D) cells (see Figure 13) with Cartesian subregions may be created using the following DRAGON input data structures

- 2-D annular cell

  `Geometry := GEO: TUBE` $n_r\ n_x\ n_y$
  `MESHR` $(r_g,\ g=0,n_r)$
  `MESHX` $(x_i,\ i=0,n_x)$
  `MESHY` $(y_j,\ i=0,n_y)$
  `OFFCENTER` $x_c\ y_y$ `MIX` $(((m_{g+n_r(i-1+n_x(j-1))},\ g=1,n_r),\ i=1,n_x),\ j=1,n_y)$ `;`

containing a maximum of $N_r = n_r n_x n_y$ subregions. Here the conditions:

$$x_0 < -r_{n_r} + \frac{x_0 + x_{n_x}}{2} + x_c$$
$$x_{n_x} > r_{n_r} + \frac{x_0 + x_{n_x}}{2} + x_c$$
$$y_0 < -r_{n_r} + \frac{y_0 + y_{n_y}}{2} + y_c$$
$$y_{n_y} > r_{n_r} + \frac{y_0 + y_{n_y}}{2} + y_c$$

must be satisfied.

- 3-D X directed cylinders cell

  `Geometry := GEO: TUBEX` $n_r\ n_x\ n_y\ n_z$
  `MESHR` $(r_g,\ g=0,n_r)$
  `MESHX` $(x_i,\ i=0,n_x)$
  `MESHY` $(y_j,\ i=0,n_y)$
  `MESHZ` $(z_k,\ k=0,n_z)$
  `OFFCENTER` $x_c\ y_y\ z_c$ `MIX` $((((m_{g(n_r(j-1+n_y(k-1+n_z(i-1)))},\ g=1,n_r),\ j=1,n_y),\ k=1,n_z),\ i=1,n_x)$ `;`

  that contains a maximum of $N_r = n_r n_x n_y n_z$ subregions. Here the conditions:

$$y_0 < -r_{n_r} + \frac{y_0+y_{n_y}}{2} + y_c$$
$$y_{n_y} > r_{n_r} + \frac{y_0+y_{n_y}}{2} + y_c$$
$$z_0 < -r_{n_r} + \frac{z_0+z_{n_z}}{2} + z_c$$
$$z_{n_z} > r_{n_r} + \frac{z_0+z_{n_z}}{2} + z_c$$

  must be satisfied.

- 3-D Y directed cylinders cell

  `Geometry := GEO: TUBEY` $n_r\ n_x\ n_y\ n_z$
  `MESHR` $(r_g,\ g=0,n_r)$
  `MESHX` $(x_i,\ i=0,n_x)$
  `MESHY` $(y_j,\ i=0,n_y)$
  `MESHZ` $(z_k,\ k=0,n_z)$
  `OFFCENTER` $x_c\ y_y\ z_c$ `MIX` $((((m_{g+n_r(k-1+n_z(i-1+n_x(j-1)))},\ g=1,n_r),\ k=1,n_z),\ i=1,n_z),\ j=1,n_y)$ `;`

  that contains a maximum of $N_r = n_r n_x n_y n_z$ subregions. Here the conditions:

$$x_0 < -r_{n_r} + \frac{x_0+x_{n_x}}{2} + x_c$$
$$x_{n_x} > r_{n_r} + \frac{x_0+x_{n_x}}{2} + x_c$$
$$z_0 < -r_{n_r} + \frac{z_0+z_{n_z}}{2} + z_c$$
$$z_{n_z} > r_{n_r} + \frac{z_0+z_{n_z}}{2} + z_c$$

  must be satisfied.

- 3-D Z directed cylinders cell

  `Geometry := GEO: TUBEZ` $n_r\ n_x\ n_y\ n_z$
  `MESHR` $(r_g,\ g=0,n_r)$
  `MESHX` $(x_i,\ i=0,n_x)$
  `MESHY` $(y_j,\ i=0,n_y)$
  `MESHZ` $(z_k,\ k=0,n_z)$
  `OFFCENTER` $x_c\ y_y\ z_c$ `MIX` $((((m_{g+n_r(i-1+n_x(j-1+n_y(k-1)))},\ g=1,n_r),\ i=1,n_x),\ j=1,n_y),\ k=1,n_z)$ `;`

that contains a maximum of $N_r = n_r n_x n_y n_z$ subregions. Here the conditions:

$$\begin{aligned}
x_0 &< -r_{n_r} + \frac{x_0 + x_{n_x}}{2} + x_c \\
x_{n_x} &> r_{n_r} + \frac{x_0 + x_{n_x}}{2} + x_c \\
y_0 &< -r_{n_r} + \frac{y_0 + y_{n_y}}{2} + y_c \\
y_{n_y} &> r_{n_r} + \frac{y_0 + y_{n_y}}{2} + y_c
\end{aligned}$$

must be satisfied.

Only the part of each Cartesian region totally inside the annular or cindrical region will be considered. The mixture

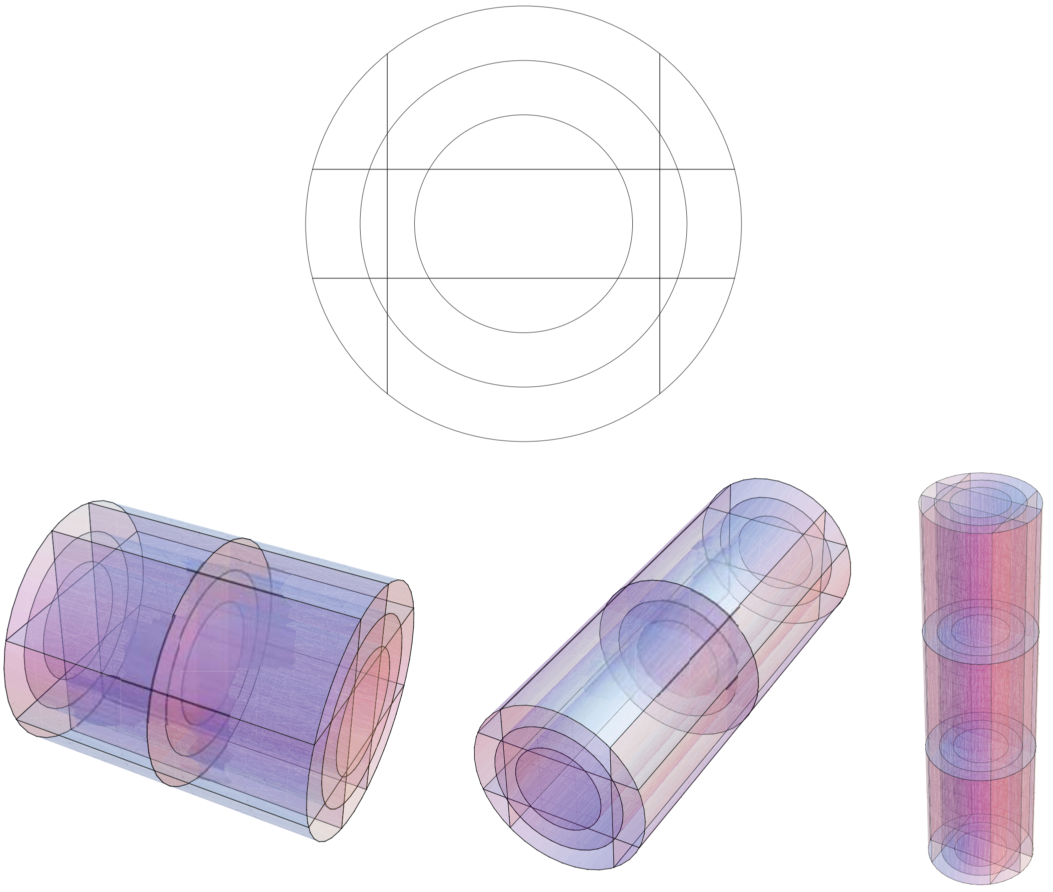

Figure 13: Examples of `TUBE` (top), `TUBEX` (bottom left), `TUBEY` (bottom center) and `TUBEZ` (bottom right) geometries with Cartesian subregions.

associated with these subregions are provided provided by m_l.

For the 2-D `TUBE` geometries, each possible subregion l of the cell can be associated with a position (g, i, j) according to:

$$l = g + n_r(i - 1 + n_x(j - 1)) \tag{3.87}$$

for a maximum of N_r regions. The volume V_l of these regions can be evaluated using the procedures described in Appendix A. Note that some of these volumes may turn out to vanish since not all the radial regions will have an intersection with each of the Cartesian region. Thus, the final region identification will correspond to $\{l_c\}$, a compressed version of the set $\{l\}$ that contains N_V terms where the elements with vanishing volume have been removed:

```
lc=0
DO l=1,N_{r}
  IF(V_{l} .NE. 0) THEN
    lc=lc+1
    V_{lc}=V_{l}
    m_{lc}=m_{l}
  ENDIF
ENDDO
N_{V}=lc
```

where the mixture identification vector is compressed in a fashion similar to V_l.

For the 3-D `TUBEX`, `TUBEY` and `TUBEZ` geometries, a similar process is used where each possible subregion l of the cell can be now associated with a position (g, i, j, k). Here we use:

$$l = g + n_r(i - 1 + n_x(j - 1 + n_y(k - 1))) \tag{3.88}$$

for a maximum of N_r regions. The global ordering here is different from that used for the mixture ordering. The volumes V_l can again be evaluated using the procedures described in Appendix A. As for the `TUBE` geometry, some of the volumes may turn out to vanish. Thus, the final region identification will correspond to $\{l_c\}$, a compressed version of the set $\{l\}$ that contains N_V terms where the elements with vanishing volume can be removed using the procedure described above.

In 2-D, the outer surfaces ℓ are numbered according to the following procedure

$$\ell = -i - n_x(j - 1) \tag{3.89}$$

for a maximum of $n_y n_y$ radial surfaces at r_{n_r} (for example in Figure 13, there is no outer surface associated with the Cartesian region $(i, j) = (2, 2)$).

For 3-D Cylindrical geometries with embedded Cartesian regions, the surfaces are numbered according to the following procedure

- `TUBEX`
  1. A maximum of $n_{r,x} = n_r n_y n_z$ surfaces at $x = x_0$
  $$\ell = -g - n_r(j - 1 + n_y(k - 1)) \tag{3.90}$$
  with S_ℓ evaluated using the procedure of Appendix A.
  2. A maximum of $n_{r,x}$ surfaces at $x = x_{n_x}$
  $$\ell = -g - n_r(j - 1 + n_y(k - 1)) - n_{r,x} \tag{3.91}$$
  with S_ℓ evaluated using the procedure of Appendix A.
  3. A maximum of $n_x n_y n_z$ radial surfaces at $r = r_{n_r}$
  $$\ell = -i - n_x(j - 1 + n_y(k - 1)) - 2n_{r,x} \tag{3.92}$$

  for a maximum of
  $$N_S = 2n_{r,x} + n_x n_y n_z \tag{3.93}$$
  surfaces.

- `TUBEY`
  1. A maximum of $n_{r,y} = n_r n_x n_z$ surfaces at $y = y_0$
  $$\ell = -g - n_r(i - 1 + n_z(k-1)) \tag{3.94}$$
  2. A maximum of $n_{r,y}$ surfaces at $y = y_{n_y}$
  $$\ell = -g - n_r(i - 1 + n_x(k-1)) - n_{r,y} \tag{3.95}$$
  3. A maximum of $n_x n_y n_z$ radial surfaces at $r = r_{n_r}$
  $$\ell = -i - n_x(j - 1 + n_y(k-1)) - 2n_{r,x} \tag{3.96}$$

  for a maximum of
  $$N_S = 2n_{r,y} + n_x n_y n_z \tag{3.97}$$
  surfaces.
- `TUBEZ`
  1. A maximum of $n_{r,z} = n_r n_x n_y$ surfaces at $z = z_0$
  $$\ell = -g - n_r(i - 1 + n_x(j-1)) \tag{3.98}$$
  2. A maximum of $n_{r,z}$ surfaces at $z = z_{n_z}$
  $$\ell = -g - n_r(i - 1 + n_x(j-1)) - n_{r,z} \tag{3.99}$$
  3. A maximum of $n_x n_y n_z$ radial surfaces at $r = r_{n_r}$
  $$\ell = -i - n_x(j - 1 + n_y(k-1)) - 2n_{r,x} \tag{3.100}$$

  for a maximum of
  $$N_S = 2n_{r,z} + n_x n_y n_z \tag{3.101}$$
  surfaces.

Again some of these surfaces will vanish and the final index associated with the surfaces will be compressed using a procedure similar to that used for the volumes.

3.2 Pins in cells

Pin cells can be inserted in a pure geometry using the command `CLUSTER`. A 2-D example can be found in Figure 14 corresponding to

```
Geometry := GEO: CARCEL $n_r$ $n_x$ $n_y$
MESHR ($r_g,\ g = 0, n_r$)
MESHX ($x_i,\ i = 0, n_x$)
MESHY ($y_j,\ j = 0, n_y$)
MIX ((($m_{g+(n_r+1)(i-1+n_x(j-1))},\ g = 1, n_r + 1),\ i = 1, n_x),\ j = 1, n_y$)
CLUSTER GeoPin
   ::: GeoPin := GEO: TUBE $n_{r,l}$ $n_{x,l}$ $n_{y,l}$
   MESHR ($r_{g,l},\ g = 0, n_{r,l}$)
   MESHX ($x_{i,l},\ i = 0, n_{x,l}$)
   MESHY ($y_{j,l},\ i = 0, n_{y,l}$)
   MIX ((($m_{g+n_r(i-1+n_x(j-1))},\ g = 1, n_{r,l}),\ i = 1, n_{x,l}),\ j = 1, n_{y,l}$)
   NPIN 2 RPIN $r_p$ APIN $\theta_p$ ;
;
```

where two pin geometry `GeoPin` are inserted in the cell `Geometry` at positions

$$(x_1, y_1) = \left(r_p \cos(\theta_p) + \frac{x_0 + x_{n_x}}{2}, r_p \sin(\theta_p) + \frac{y_0 + y_{n_y}}{2} \right)$$
$$(x_2, y_2) = \left(r_p \cos(\theta_p + \pi) + \frac{x_0 + x_{n_x}}{2}, r_p \sin(\theta_p + \pi) + \frac{y_0 + y_{n_y}}{2} \right)$$

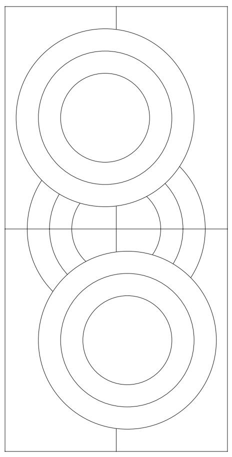

Figure 14: Annular pins in `CARCEL` geometries.

Similarly, the 3-D example illustrated in Figure 15 corresponds to

`Geometry := GEO: CARCELX` $n_r\ n_x\ n_y\ n_z$
`MESHR` $(r_g,\ g = 0, n_r)$
`MESHX` $(x_i,\ i = 0, n_x)$
`MESHY` $(y_j,\ j = 0, n_y)$
`MESHZ` $(z_k,\ k = 0, n_z)$
`MIX` $((((m_{g+(n_r+1)(j-1+n_y(k-1+n_z(i-1)))},\ g = 1, n_r + 1),\ j = 1, n_y),\ k = 1, n_z),\ i = 1, n_x)$
`CLUSTER` GeoPin
    `:::` GeoPin `:= GEO: TUBEX` $n_{r,p}\ n_{x,p}\ n_{y,p}\ n_{z,p}$
    `MESHR` $(r_{g,p},\ g = 0, n_{r,p})$
    `MESHX` $(x_{i,p},\ i = 0, n_{x,p})$
    `MESHY` $(y_{j,p},\ i = 0, n_{y,p})$
    `MESHZ` $(z_{k,p},\ k = 0, n_{z,p})$
    `MIX` $((((m_{g+(n_r)(j-1+n_y(k-1+n_z(i-1)))},\ g = 1, n_r),\ j = 1, n_y),\ k = 1, n_z),\ i = 1, n_x)$
    `NPIN` 2 `RPIN` r_p `;`
`;`

The main feature of the pure geometry that are inserted in a cell using this command is the fact that the pins hide completely the part of the cell geometry that is located completely inside the limits of the pins (see Figure **??** for a 2-D example)

3.3 Cells in assemblies

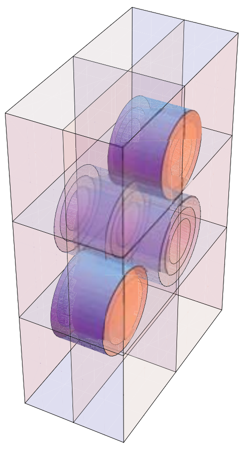
Figure 15: X directed cylindrical pins in CARCELX geometries.

APPENDICES

APPENDIX A

ANALYTICAL VOLUME EVALUATION

Here we will only be dealing with combinations of 3-D rectangles and 3-D cylinders or 2-D rectangles and circles. One restriction in DRAGON is that only 3-D cylinders within a cell can intersect and that these cylinders must all have parallel axes that are co-linear with the x, y or z directions. One consequence of this observation is that all the volume for 3-D geometry can be computed using the surfaces associated with a 2-D projection of the cell along the axis of the cylinders multiplied by the height of the cell or cylinder. Accordingly, we will concentrate on the problem of evaluating the surfaces associated with each region possible when rectangles and circles intersect with a maximum of 2 non-concentric circles being permitted in DRAGON.

A.1 Rectangle

A 2-D Cartesian cell is identified by a point (x, y) that satisfies (see Figure 16):

$$x_1 \leq x \leq x_2 \qquad\qquad y_1 \leq y \leq y_2 \tag{A.1}$$

where x_1 and x_2 are the locations with respect to the origin of the left and right boundary while y_1 and y_2 are the locations with respect to the origin of the bottom and top boundary. The 2-D volume (3-D surface) of this rectangle is given by:

$$V^C = (x_2 - x_1)(y_2 - y_1)$$

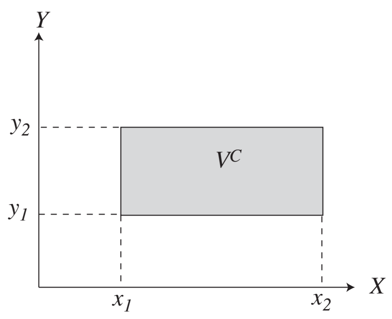

Figure 16: Volume of a rectangle

A.2 Concentric Circles

Concentric annular rings are identified by the position of their center (R, θ) with respect to the origin and the inner r_1 and outer radius r_2 of the rings (see Figure 17)

$$r_1 \leq r \leq r_2$$

The 2-D volume of this ring is:

$$V^R = \pi(r_2^2 - r_1^2)$$

The volume of a circle of radius r can be obtained from the above using $r_2 = r$ and $r_1 = 0$.

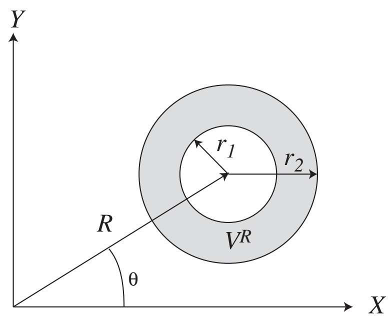

Figure 17: Volume of concentric annular circles

A.3 Overlapping Circles

The 2-D volume V^A associated with the intersection of two non-concentric circles overlapping is somewhat more difficult to obtain. Here we will assume for simplicity that each circle i ($i = 1, 2$) is identified by (R_i, θ_i, r_i) as illustrated in Figure 18. The first step is then to locate the center $(x_{i,c}, y_{i,c})$ of each circle i

$$(x_{i,c}, y_{i,c}) = (R_i \cos\theta_i, R_i \sin\theta_i)$$

the equation for the line delimiting each circle being

$$(x - x_{i,c})^2 + (y - y_{i,c})^2 = r_i^2$$

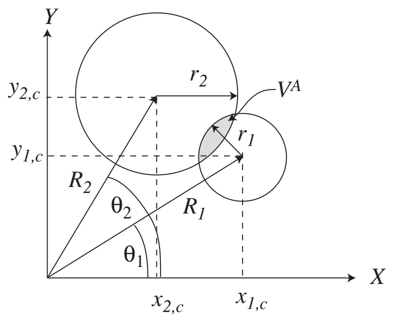

Figure 18: Volume of overlapping circles

The first step in evaluating the surface of intersection between circle 1 and 2 is to find the intersection points, if any between the two lines defining the circle. This can be readily obtained in the following way. First, one displaces the system origin in such a way that it is located at the center of circle 1 ($x \to x + x_{1,c}$ and $y \to y + y_{1,c}$) (see Figure 19). The equations for the line defining each circle in the translated system of reference is

$$x^2 + y^2 = r_1^2$$
$$(x - x_{2,d})^2 + (y - y_{2,d})^2 = (r_2)^2$$

with

$$x_{2,d} = (x_{2,c} - x_{1,c})$$
$$y_{2,d} = (y_{2,c} - y_{1,c})$$

Every point on the first circle must then satisfy:

$$y^2 = r_1^2 - x^2 \tag{A.2}$$

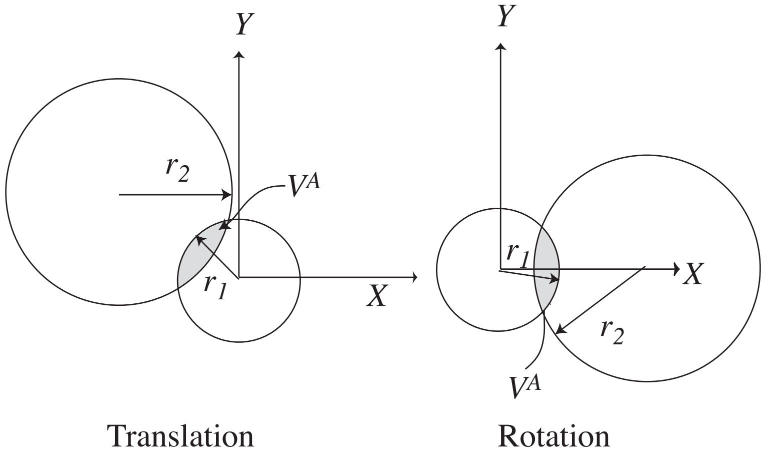

Figure 19: Translation and rotation of overlapping circles

this equation being invariant under rotation of the axis with respect to the origin of circle 1. The second step consists in rotating the axis in such a way that the center of the circle 2 is located on the x axis (see Figure 19). Since circle 2 is located at position $(x_{2,d}, y_{2,d})$ with respect to the origin of circle 1, we will have:

$$R_{2,d} = \sqrt{(x_{2,d})^2 + (y_{2,d})^2}$$
$$\theta_{2,d} = \arctan\left(\frac{y_{2,d}}{x_{2,d}}\right)$$

Note that for $|y_{2,d}/x_{2,d}| > 1$, one generally uses

$$\theta_{2,d} = \text{arccotan}\left(\frac{x_{2,d}}{y_{2,d}}\right)$$

for increased precision when $x_{2,d}$ becomes small with respect to $y_{2,d}$.

After a rotation of the axis by $-\theta_{2,d}$, the equation for circle 2 becomes:

$$(x - R_{2,d})^2 + y^2 = r_2^2 \tag{A.3}$$

Circles 1 and 2 then intersect only if

$$R_{2,d} < r_1 + r_2$$

Using Eq. (A.2) in Eq. (A.3) one obtains

$$(x - R_{2,d})^2 + r_1^2 - x^2 = r_2^2$$

which has for solution:

$$x_s = \frac{R_{2,d}^2 - r_2^2 - r_1^2)}{2R_{2,d}} \tag{A.4}$$

The intersection points are then given by $(x_s, -y_s)$ and (x_s, y_s) with

$$y_s = \sqrt{r_1^2 - x_s^2} \tag{A.5}$$

Now for circle 1, these two points cover an angular sector of width

$$\alpha_1 = 2\arctan\left(\frac{y_s}{x_s}\right)$$

while for circle 2 the angular sector covered is

$$\alpha_2 = 2\arctan\left(\frac{y_s}{x_s - R_{2,d}}\right)$$

still being careful to use the $\mathrm{arccotan}$ instead when either x_s or $x_s - R_{2,d}$ are much smaller than y_s.

Using this information, the 2-D volume of the intersection region can be computed in the following way. It is the sum the part of the angular sector of circle 1 to the left of the line define by x_s given by:

$$V^{1,l} = \frac{\alpha_1}{2} r_1^2 - x_s y_s$$

and the part of the angular sector of circle 2 to the right of the line define by x_s:

$$V^{2,r} = \pi r_2^2 - \left(\frac{\alpha_2}{2} r_2^2 - (x_s - R_{2,d}) y_s\right)$$

for a total of

$$V^A = V^{1,l} + V^{2,r}$$

A.4 Circles Overlapping a Rectangle

First let us consider the intersection of a rectangle defined as in Eq. (A.1) and a circle defined by (R, θ, r) or (x_c, y_c, r) as illustrated in Figure 20:

$$(x_c, y_c) = (R\cos\theta, R\sin\theta)$$

The first step here consists of determining whether the rectangle is located totally outside or inside the circle or intersects it.

In the case where the rectangle is totally located outside the cylinder the volume of intersection of these two regions vanishes identically. If the Cartesian region is an outer cell, the surface area of each face is just the lenght of the lines composing the sides of the Cartesian cell. In the case where the rectangle is located totally inside the circle, the volume of intersection is V^C while if the circle is located totally inside the circle the volume of intersection is V^R and the surface area is the lenght of the circle, namely $2\pi r$ (for a circle corresponding to an outer boundary). Finally when some of the faces defining the rectangle intersect the circle, one can compute the volume of intersection V^O between the rectangle and the circle using

$$V^O = V_{2,2} - V_{2,1} - V_{1,2} + V_{1,1}$$

where the volume $V_{i,j}$ represents the intersection between the cylinder and the plane located to the left of surface x_i and below y_j (see Figure 21 for example).

Here for simplicity we will compute $V_{i,j}$ in terms of V_j which represents either the cylinder surface located to the left of the plane defined by x_j or the cylinder surface located below the plane defined by surface y_j. These can be obtained using the relation

$$V_j = \begin{cases} 0 & \text{for } u_j < -r \\ \pi r^2 & \text{for } u_j > r \\ \alpha_j r^2 + u_j\sqrt{r^2 - (u_j)^2} & \text{otherwise} \end{cases}$$

$$\alpha_j = \arccos\left(-\frac{u_j}{r}\right)$$

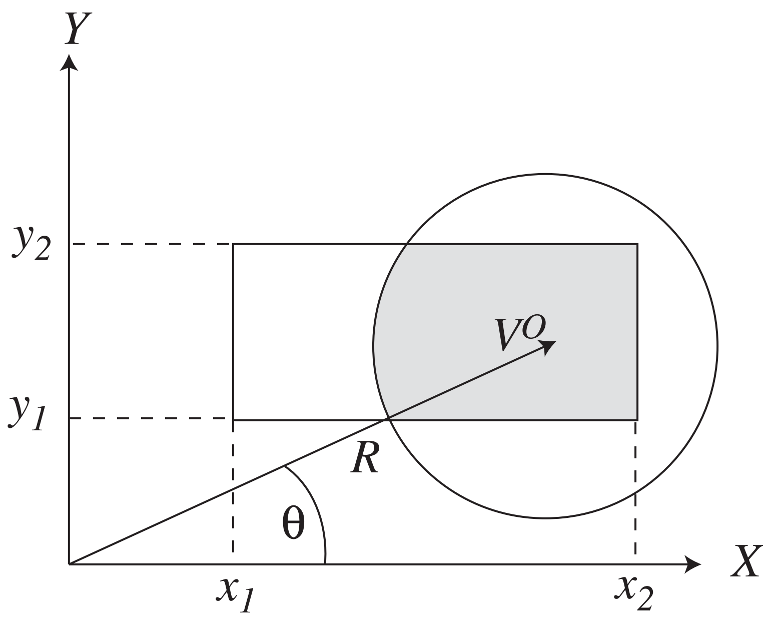

Figure 20: Volume for the overlapp of a rectangle with an annular region

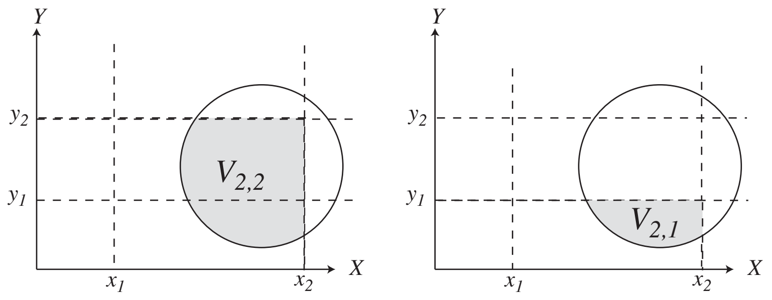

Figure 21: Decomposition of a rectangle circle region overlap

where

$$u_j = x_j - x_c$$

for the surface to the left of x_j or

$$u_j = y_j - y_c$$

for the surface below y_j.

In the case where the point of intersection of lines x_i and y_j is located inside the cylinder of radius r, namely

$$u_{i,j} = \sqrt{(x_i - x_c)^2 + (y_j - y_c)^2} < r$$

V_{ij} is given by:

$$V_{i,j} = \frac{1}{2}(V_i - V_j) + u_i u_j + \frac{1}{4}\pi r^2$$

For all the other cases, depending on the location of the various planes with respect to the center of the cylinder we will use:

$$V_{i,j} = \begin{cases} 0 & \text{if } u_i < 0 \text{ and } u_j < 0 \\ V_j & \text{if } u_i < 0 \text{ and } u_j > 0 \\ V_i & \text{if } u_i > 0 \text{ and } u_j < 0 \\ V_i + V_j - \pi r^2 & \text{if } u_i > 0 \text{ and } u_j > 0 \end{cases}$$

When Cartesian regions intersect annular cell, only the surfaces associated with the annular boundary can be outer boundaries (`TUBE`). In this case, the area correspond to

$$S_{\text{radial}} = r(\theta_{\max} - \theta_{\min})$$

where $\theta_{\max}$ is the maximum angle covered by the circular arc and $\theta_{\min}$ the minimum angle.

Finally, for the case where a rectangle is intersected with 2 non concentric annular regions as illustrated in Figure 22 the volume evaluation process to be considered is a combination of rectangle/circles intersections (see above) and circle/circle intersections (see Appendix A.3).

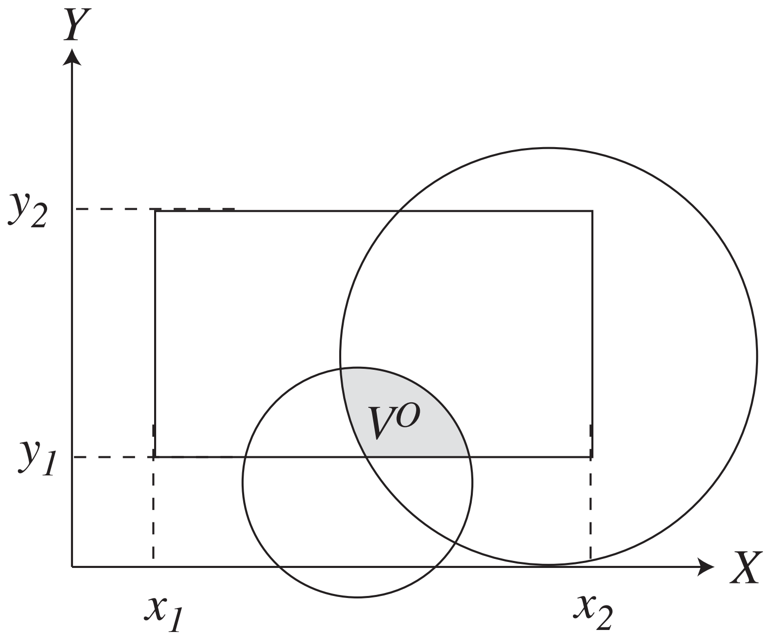

Figure 22: Volume for the overlapp of a rectangle with two non concentric annular regions

APPENDIX B

PROGRAMMING GUIDE FOR THE NXT MODULE

B.1 Structure of the program

Structure of the NXT tracking driver:

```
NXT
 |------ NXTGET
 |------ NXTACG
 |        |------ NXTBCG
 |        |------ NXTBRT
 |        |        |------ NXTETS
 |        |------ NXTCUA
 |        |        |------ NXTTRS
 |        |------ NXTCVS
 |        |        |------ NXTAVS
 |        |------ NXTGMD
 |        |        |------ NXTTPO
 |        |        |        |------ NXTIAA
 |        |        |        |------ NXTIRA
 |        |        |------ NXTTRM
 |        |------ NXTMCD
 |        |        |------ NXTEGI
 |        |        |------ NXTRCS
 |        |        |------ NXTRIS
 |        |        |        |------ NXTTRS
 |        |        |------ NXTRPS
 |        |        |------ NXTSGI
 |        |        |------ NXTTPS
 |        |        |------ NXTVOL
 |        |        |        |------ NXTPCA
 |        |        |        |        |------ NXTIRA
 |        |        |        |        |------ NXTIRR
 |        |        |        |        |        |------ NXTPRR
 |        |        |        |------ NXTPCC
 |        |        |        |        |------ NXTIAA
 |        |        |        |        |------ NXTIRA
 |        |        |        |        |------ NXTIRR
 |        |        |        |        |        |------ NXTPRR
 |        |        |        |        |------ NXTPRA
 |        |        |        |        |------ NXTPRR
 |        |        |        |------ NXTVCA
 |        |        |        |------ NXTVCC
 |        |        |        |        |------ NXTVCA
 |        |        |        |        |------ NXTIRA
 |------ NXTTCG
 |        |------ NXTQAC
 |        |------ NXTQAS
```

```
   |         |         |------ NXTQEW
   |         |         |------ NXTQLC
   |         |         |------ NXTQLT
   |         |------ NXTQSC
   |         |------ NXTQSS
   |         |------ NXTTLC
   |         |         |------ NXTLCA
   |         |         |------ NXTTCR
   |         |         |         |------ NXTLCA
   |         |         |         |------ NXTLCU
   |         |         |         |------ NXTLCY
   |         |         |         |------ NXTLRS
   |         |         |         |------ NXTRTL
   |         |------ NXTTLS
   |         |         |------ NXTLCA
   |         |         |------ NXTQPS
   |         |         |------ NXTTCR
   |         |         |         |------ NXTLCA
   |         |         |         |------ NXTLCU
   |         |         |         |------ NXTLCY
   |         |         |         |------ NXTLRS
   |         |         |         |------ NXTRTL
   |         |------ NXTTNS
```

Structure of EXCELP inline driver for NXT:

```
   EXCELP
   |------ NXTTGC
   |         |------ NXTLCA
   |         |------ NXTTCR
   |------ NXTTGS
   |         |------ NXTLCA
   |         |------ NXTTCR
```

B.2 Use of each routine

The `NXT:` routines perform the following tasks:

- `NXT` is the main subroutine of the module NXT.
- `NXTACG` is the main routine to analyze a geometry.
- `NXTAVS` adds current cell information to global surfaces and volumes for geometry.
- `NXTBCG` reads boundary conditions and symmetries and verify for consistency.
- `NXTBRT` builts surface reflection/transmission coupling array.
- `NXTCUA` creates the array for testing the geometry in an assembly for internal symmetries and unfolding the assembly according to these symmetries.
- `NXTCVS` computes final volumes and surfaces.
- `NXTEGI` extracts cell or pin geometry information.

- `NXTETS` builts equivalent surface array for translational symmetry.
- `NXTGET` reads `NXT:` input data.
- `NXTGMD` evaluates global mesh for assembly.
- `NXTIAA` computes the volume of intersection between two `TUBE`.
- `NXTIRA` finds the intersection between a rectangular region and an annular pin.
- `NXTIRR` finds intersection between a rectangular region and a Cartesian pin.
- `NXTLCA` tracks a Cartesian 2-D or 3-D geometry.
- `NXTLCU` merges two sets of tracks.
- `NXTLCY` tracks an annular geometry.
- `NXTLRS` stores line segments in tracking vector with global region and surface identification.
- `NXTMCD` creates a multicell description for the geometry.
- `NXTPCA` removes volumes or surfaces of the overlapping pins.
- `NXTPCC` removes from the volumes or surfaces of a cell the volumes or surfaces of the overlapping pins.
- `NXTPRA` compute the volume of intersection between a `CARCEL` and a `TUBE` centered at the origin.
- `NXTPRR` finds rectangle representing the intersection of two rectangles.
- `NXTQAC` defines quadrature angles for cyclic tracking.
- `NXTQAS` defines quadrature angles for a given standard tracking option.
- `NXTQEW` defines EQ_N quadrature angles.
- `NXTQLC` defines Legendre-Chebyshev quadrature angles.
- `NXTQLT` defines Sanchez-Mao-Santandrea (Legendre-Trapezoidal) quadrature angles.
- `NXTQPS` generates directions defining the planes normal to a solid angle.
- `NXTQSC` defines spatial quadrature for cyclic tracking.
- `NXTQSS` defines standard spatial quadrature.
- `NXTRCS` renumbers cell surfaces.
- `NXTRIS` rotates geometry according to reference turn.
- `NXTRPS` renumbers pin cluster surfaces.
- `NXTRTL` rotates tracking line according to reference turn.
- `NXTSGI` discretizes geometry according to splitting options.
- `NXTTCG` is the main routine to track a geometry.
- `NXTTCR` tracks a cell rotated according to its explicit position in the assembly.
- `NXTTLC` generates the cyclic tracking lines for a geometry.
- `NXTTLS` generates the standard tracking lines for a geometry

- `NXTTNS` normalizes tracking lines and save on tracking data structure.
- `NXTTPO` tests cluster pins overlapp.
- `NXTTPS` tests if pins satisfy required symmetry.
- `NXTTRM` determines the final mesh of a cell after turn.
- `NXTTRS` applies Cartesians symmetry to TURN factors.
- `NXTVCA` computes volume and surfaces for Cartesian geometry.
- `NXTVCC` computes volumes for a mixed `CARCEL` geometry.
- `NXTVOL` computes regional volumes.

In addition the following lines are called by the `EXCELP` routine when inline tracking is requested:

- `NXTTGC` generate a specific cyclic tracking line for a geometry (inline tracking).
- `NXTTGS` generate a specific standard tracking line for a geometry (inline tracking).