FLUKA
Combinatorial Geometry

FLUKA Beginner’s Course
Introduction

**Principle of Combinatorial Geometry**

Basic objects called *bodies* (such as cylinders, spheres, parallelepipeds, etc.) are combined to form more complex objects called *regions*.

This combination is done using Boolean operations: union, intersection, and subtraction.

1 complex object

3 basic objects
Introduction

Initially, FLUKA Combinatorial Geometry was similar to the package developed at ORNL for the neutron and gamma-ray transport program Morse (M.B. Emmett ORNL-4972 1975.

In the original CG (Morse) bodies were convex solid bodies (i.e. finite portions of space completely delimited by surfaces of first or second degree, i.e. planes or quadrics)

The present FLUKA CG package has been highly improved:
- addition of “infinite bodies”
- possibility of using body and region names instead of numbers and getting rid of alignment rules,
- availability of parentheses
- expansion and roto-translation of bodies
- comments
- repetition of patterns (lattices)
- voxels
Basic Concepts

Four concepts are fundamental in the FLUKA CG:

- **Bodies**: basic convex objects, plus infinite planes (half-spaces), infinite cylinders (circular and elliptical), and generic quadric surfaces (surfaces described by 2nd degree equations)

- **Zones**: sub-regions defined only via bodies intersection and subtraction

- **Regions**: defined as Boolean operations on bodies (union of zones)

- **Lattices**: duplication of existing objects (translated & rotated), will be explained in a separate lecture

Use of “infinite bodies” is encouraged: input are less error-prone and tracking is faster and more accurate.
Basic Concepts: The Black Hole

All particles entering a black-hole are absorbed (they vanish)

**FLUKA geometry MUST be embedded into a BLCKHOLE region**
(to avoid tracking particles to infinity)

The outer surface of the BLCKHOLE region must be a single closed body (e.g. a sphere).

Further black-hole regions can be defined by the user if desired
Combinatorial Geometry Input

CG input must respect the following structure:

**GEOBEGIN** card
- **VOXELS** card (optional, see advanced geometry lecture)
- Geometry title (and reading format options)
  - Body data
- **END** card
- Region data
- **END** card
- **LATTICE** cards (optional, see advanced geometry lecture)
  - Region volumes (optionally requested by a flag in the Geometry title, used together with the **SCORE** command)
- **GEOEND** card

Reminder for txt format input files:
an asterisk (*) in the 1st column comments the line
GEOBEGIN card

WHAT(1) not used
WHAT(2) set accuracy parameter - use with extreme care!
WHAT(3) logical unit for the geometry input.
    Geometry input can be read from file; should be >20.0
WHAT(4) logical unit for the geometry output
    Log feature; should be >20.0; Default is standard output
WHAT(5) Parenthesis optimization level (see FLUKA manual)
WHAT(6) not used
SDUM COMBNAME or COMBINAT
    COMBNAME selects name based format, COMBINAT fixed format
    Default: COMBINAT (!)
    Is overwritten by WHAT(5) of a possible GLOBAL card
Geometry input format

- The input file format for the geometry is different from the one adopted anywhere else in FLUKA (i.e. the number and length of the input fields is different)

- Different formats can be used (due to backward compatibility)
  
  - **Fixed format**
    - Alignment is mandatory
    - Bodies and regions are identified by numbers (rather than names), this makes geometry editing more difficult

  - **Name based format**
    - It is the recommended format
    - It is NOT the default!
Name based format

It has many advantages, the main are:

- input parameter alignment is not necessary
- bodies and regions are identified by names
- possible to edit the input sequence without affecting the region description (e.g. inserting a new body)
- parentheses can be used to perform complex Boolean operations in the description of regions.

Name based format input is used for both body and region

It is activated

either by a **GLOBAL** command at the beginning of the input file
or putting **SDUM = COMBNAME** in the **GEOBEGIN** card
Geometry TITLE card

This card has no keyword, it is the line that follows the GEOBEGIN card (unless voxels are used) and its format is (2I5, 10X, A60)

The card gets three inputs: IVLFLG, IDBG, TITLE

IVLFLG (Input VoLumes FLAG) triggers the normalization of the quantities scored in regions by the option SCORE

IVLFLG= 0 ➞ no normalization applied

IVLFLG= 3 ➞ results divided by region volumes given before GEOEND. Volume value for each region must be provided, format: (7E10.5).

IDBG select different kinds of geometry fixed format input:

IDBG = 0 : default fixed format

IDBG = -10 or -100 : high accuracy fixed format
Bodies
Bodies

- Each body divides the space into two domains: inside and outside.
- 3-character code of available bodies:
  - RPP: Rectangular Parallelepiped
  - SPH: SPHERE
  - XYP, XZP, YZP: Infinite half space delimited by a coordinate plane
  - PLA: Generic infinite half-space, delimited by a PLANE
  - XCC, YCC, ZCC: Infinite Circular Cylinder, parallel to coordinate axis
  - XEC, YEC, ZEC: Infinite Elliptical Cylinder, parallel to coordinate axis
  - RCC: Right Circular Cylinder
  - REC: Right Elliptical Cylinder
  - TRC: Truncated Right angle Cone
  - ELL: ELLipsoid of revolution
  - QUA: QUAdric
- Other (deprecated) bodies: ARB, RAW, WED, BOX

Do not use them, because they can cause rounding problems
Bodies

The input for each body consists of:

- the 3-letter code indicating the body type (RPP, ZCC...)
- a unique "body name" (alphanumeric identifier, 8 character maximum, case sensitive)
- a set of geometrical quantities defining the body
  (the number depends on the body type, see next slides)

Different items can extend over as many lines as needed
(Maximum 132 characters per line accepted)

Different items have to be separated by one or more blanks,
or by one of the separators , / ; :

All values are in cm!
Infinite half-space parallel to coordinate axis

There are 4 kinds of infinite half-spaces

Three are delimited by planes perpendicular to the coordinate axes:
1. Delimited by a plane \( \perp \) to the \( x \)-axis. Code: \( \text{YZP} \)
2. Delimited by a plane \( \perp \) to the \( y \)-axis. Code: \( \text{XZP} \)
3. Delimited by a plane \( \perp \) to the \( z \)-axis. Code: \( \text{XYP} \)

All defined by a single number:
\( Vx \) (resp. \( Vy \), or \( Vz \)),
coordinate of the plane on the corresponding perpendicular axis

Points for which:
\[ x < Vx \] (resp. \( y < Vy \), or \( z < Vz \))
are “inside the body”
A PLA defines the infinite half space delimited by a generic plane.

A PLA is defined by 6 numbers:
Hx, Hy, Hz (vector ⊥ to the plane, arbitrary length);
Vx,Vy,Vz (any point lying on the plane)

The half-space “inside the body” is that from which the vector is pointing (i.e. the vector points "outside").
Rectangular Parallelepiped: RPP

An RPP has its edges parallel to the coordinate axes. It is defined by 6 numbers in the following order:

\[ X_{\text{min}}, X_{\text{max}}, Y_{\text{min}}, Y_{\text{max}}, Z_{\text{min}}, Z_{\text{max}} \]

(min and max coordinates delimiting the parallelepiped)
An **RPP** definition extends over one single card in default fixed format, or over two cards in high-accuracy body fixed format.

Example in **fixed format** (the comment lines shown are allowed input lines):

```
*... +....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
RPP  4     -20.0     +20.0     -50.0     +50.0     -38.5     +38.5
* (a parallelepiped centered on the origin)
```

High-accuracy fixed format

```
*... +....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
RPP  4     -20.0     +20.0     -50.0     +50.0     -38.5     +38.5
```

---

**Rectangular Parallelepiped:** RPP
A SPH is defined by 4 numbers:
Vx, Vy, Vz (coordinates of the centre), R (radius)
Circular cylinder

Right circular cylinder: RCC

An RCC can have any orientation in space
Limited by a cylindrical surface and two plane faces $\perp$ to its axis.
Each RCC is defined by 7 numbers:
$V_x, V_y, V_z$ (centre of one face);
$H_x, H_y, H_z$ (vector corresponding to the cylinder height, pointing toward the other face);
$R$ (cylinder radius).

<table>
<thead>
<tr>
<th>RCC</th>
<th>eg_RCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_x$</td>
<td>0.0</td>
</tr>
<tr>
<td>$V_y$</td>
<td>0.0</td>
</tr>
<tr>
<td>$V_z$</td>
<td>0.0</td>
</tr>
<tr>
<td>$H_x$</td>
<td>0.0</td>
</tr>
<tr>
<td>$H_y$</td>
<td>10.0</td>
</tr>
<tr>
<td>$H_z$</td>
<td>10.0</td>
</tr>
<tr>
<td>$R$</td>
<td>2.0</td>
</tr>
</tbody>
</table>
Infinite cylinders

Infinite Circular Cylinder parallel to coordinate axis: XCC, YCC, ZCC
Each XCC (YCC, ZCC) is defined by 3 numbers:
Ay, Az for XCC
(Az, Ax for YCC, Ax, Ay for ZCC)
(coordinates of the cylinder axis), R (cylinder radius)

Infinite Elliptical Cylinder parallel to coordinate axis: XEC, YEC, ZEC
Each XEC (YEC, ZEC) is defined by 4 numbers: Ay, Az for XEC
(Az, Ax for YEC, Ax, Ay for ZEC)
(coordinates of the cylinder axis);
Ly, Lz for XEC
(Lz, Lx for YEC, Lx, Ly for ZEC)
(semi-axes of the ellipse)
Arbitrary generic quadric: **QUA**

A **QUA** allows to insert a quadric surface. This is defined by a 2\textsuperscript{nd} degree equation \( F(x,y,z) = 0 \)

Each **QUA** is defined by 10 numbers: 

\[ A_{xx}, A_{yy}, A_{zz}, A_{xy}, A_{xz}, A_{yz}, A_x, A_y, A_z, A_0 \]

Corresponding to the equation:

\[
A_{xx} x^2 + A_{yy} y^2 + A_{zz} z^2 + A_{xy} xy + A_{xz} xz + A_{yz} yz + \\
+ A_x x + A_y y + A_z x + A_0 = 0
\]

A **QUA** definition extends over two cards in default fixed format, and over 4 cards in high-acuracy body fixed format.
Regions
Concept

Regions are defined as combinations of bodies obtained by boolean operations:

<table>
<thead>
<tr>
<th>Method</th>
<th>Union</th>
<th>Subtraction</th>
<th>Intersection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name based format</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixed format</td>
<td>OR</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>Mathematically</td>
<td>∪</td>
<td>0</td>
<td>∩</td>
</tr>
</tbody>
</table>

Regions but must be of homogeneous material composition.

Each point of space must belong to one and only one region!
Regions

Input for each region starts on a new line and extends on as many continuation lines as are needed. It is of the form:

REGNAME NAZ boolean-zone-expression

or

REGNAME NAZ boolean-zone-expression | boolean-zone-expression | ...

- REGNAME is the region “name” alphanumeric identifier, 8 character maximum, case sensitive. Must start by an alphabetical character
- NAZ See next slide
- “boolean-zone-expression” See next slides
Regions

- **NAZ** (Number of Adjacent Zones) is a rough estimate of the number of zones a particle can enter when leaving the current region zones (5 by default). What actually matters is the NAZ sum over all regions, defining the size of the *contiguity list*.

- While tracking, the program searches in the contiguity list for the neighbor zones of each zone. If the zone is not yet in the list, the whole geometry is scanned and it is added to the list with its neighbor zones.

- When the NAZ limit is reached (i.e. the list is full) the code prints a warning: GEOMETRY SEARCH ARRAY FULL. This is not lethal: the calculation continues but with a reduced efficiency.

- If you have more than 1000 regions, you must issue a **GLOBAL** card putting in **WHAT(1)** a higher limit (not beyond 10000).
**Boolean zone expressions (+/- operators)**

- **Zones** are defined by intersections and/or subtraction of bodies
  - Zones are described by a sequence of one or more bodies each being preceded by its + or - sign
  - **+body**: only the inner part of the body can belong to the zone (means that the zone being described is fully contained inside this body)
  - **-body**: only the outside of the body can belong to the zone (means that the zone being described is fully outside this body)

\[
\text{Zone1} = +\text{Body1} +\text{Body3}
\]

(Zone1 is the space common to Body1 and Body3)

\[
\text{Zone2} = +\text{Body2} -\text{Body1}
\]

(Zone2 is the space inside of Body2 but outside of Body1)

**Zones** must be finite: normally in the description of each zone and hence of each region the symbol + must appear at least once.
Boolean zone expressions ( +/- operators)
Combining zones (| operator)

- The | (or OR) operator is used as a Boolean union operator in order to combine zones.

- Such combination of zones forms a region:
  - In its simplest form, a region just consists of one zone.
  - Regions are not necessarily simply connected, i.e., they can consist of zones which are not contiguous.
  - On the other hand, zones belonging to the same region can be partially overlapping.

Example:

\[
\text{Ground 5 | +Body9 | +Body15 | +Body1 | +Body8 - Body2 | +Body8 - Body3}
\]

\[
\text{*} \\
\text{< 1st - > 2nd - > 3rd - > 4th - > 5th - >}
\]

\[
\text{| +Body8 +Body18 | +Body12 - Body10 - Body11 - Body13 - Body14}
\]

\[
\text{*} \\
\text{< 6th - > 7th and last zone}
\]

In name-based format, one can also use parentheses to form more complex Boolean operations.

In evaluating the expressions, the highest operator precedence is given to parentheses, followed by +, - and the | operator.
Example
Geometry Example “F” shaped target

F-shaped target made of Copper

Surrounded by vacuum

Black hole: spherical shell from 1000 to 10000 cm radius
Geometry example “F”: Bodies

Several possibilities for bodies, e.g.:

(A) Separated bodies
(B) Separated bodies
(C) overlapping bodies
(D) subtraction

The 4 options are equivalent. We will use C.
Geometry example “F”: input file

```
GEOBEGIN
  0  0          The copper F
  SPH SPIN     0.0 0.0 0.0 1000.
  SPH SPOUT    0.0 0.0 0.0 10000.
  RPP VERTI    0.0 1. 0.0 5. 0.0 1.
  RPP TOP      0.0 1. 4. 5. 0.0 3.
  RPP MIDDLE   0.0 1. 2. 3. 0.0 3.
END

* Black hole
  BLKBODY 5 +SPOUT -SPIN
* Void around
  VOID 5 +SPIN -TOP -VERTI -MIDDLE
* Target
  TARGET 5 +TOP | +VERTI | +MIDDLE
END
GEOEND

ASSIGNMA BLCKHOLE BLKBODY
ASSIGNMA VACUUM VOID
ASSIGNMA COPPER TARGET
```

1 cm in x
Geometry example “F”: input file

```
GEOBEGIN
   0    0          The copper F
   SPH SPIN   0.0 0.0 0.0 1000.
   SPH SPOUT  0.0 0.0 0.0 10000.
   RPP VERTI  0.0 1.0 0.0 5.0 0.0 1.
   RPP TOP    0.0 1.4 5.0 0.0 3.
   RPP MIDDLE 0.0 1.2 3.0 0.0 3.
END

* Black hole
   BLKBODY 5 +SPOUT -SPIN

* Void around
   VOID     5 +SPIN -TOP -VERTI -MIDDLE

* Target
   TARGET5 +TOP | +VERTI | +MIDDLE
END

GEOEND

ASSIGNMA    BLCKHOLE   BLKBODY
ASSIGNMA      VACUUM      VOID
ASSIGNMA      COPPER    TARGET
```
Geometry example “F”: input file

```
GEOBEGIN

0 0          The copper F
SPH SPIN     0.0 0.0 0.0 1000.
SPH SPOUT    0.0 0.0 0.0 10000.
RPP VERTI    0.0 1.0 0.0 5.0 0.0 1.
RPP TOP      0.0 1.4 5.0 0.0 3.
RPP MIDDLE   0.0 1.2 3.0 0.0 3.

END

* Black hole
BLKBODY      5  +SPOUT -SPIN
* Void around
VOID         5  +SPIN -TOP -VERTI -MIDDLE
* Target
TARGET       5  +TOP | +VERTI | +MIDDLE

END

GEOEND

ASSIGNMA    BLCKHOLE    BLKBODY
ASSIGNMA    VACUUM      VOID
ASSIGNMA    COPPER      TARGET
```
Geometry example “F”: input Flair
Geometry example “F”: input Flair
Remarks
Important Notes

- Whenever it is possible, the following bodies should be preferred:
  PLA, RPP, SPH, XCC, XEC, XYP
  XZP, YCC, YEC, YZP, ZCC, ZEC, QUA

These bodies make the tracking faster, since for them extra coding ensures that unnecessary boundary intersection calculations are avoided when the length of the next step is smaller than the distance to any boundary of the current region.

- Always use as many digits as possible in the definition of the body parameters, particularly for body heights (RCC, REC, TRC), and for direction cosines of bodies with slant surfaces. The name based format or the high-accuracy fixed format should always be used in these cases.
Precision Errors

Modeling assumptions

Avoid touching surfaces!
When floating point operations are involved.

- Use cutting surface B instead
- Or force partial overlap of bodies
Tracking accuracy

- FLUKA uses systematically double precision mathematic (i.e. 16 significant digits).

- GEOBEGIN’s WHAT(2) * 10^{-6} cm is the absolute accuracy (AA) requested for tracking and boundary identification.

- The relative accuracy (RA) achievable in double precision is of the order of 10^{-14} - 10^{-15}.

- AA should be larger than RA*L, being L the largest coordinate value in the considered problem, i.e. the whole geometry size (The outer blackhole shell containing the system does not count).

- For very large and very small geometries, you may, respectively, need to increase or decrease the WHAT(2) default value of 0.0001.
Geometry Errors and Debugging
Geometry Errors

During execution the code always needs to know the region where a particle is located at every step

- The program will **stop** only if a particle position **does not** belong to any region. An error message will be printed in the .err file with the particle position.

- **IMPORTANT!** It will not stop if a particle position belongs to more than one region. It will accept the first region it finds but the results will completely unreliable!!
Geometry Errors

Further types of errors

- Problem space not enclosed by a black body region
- Never start a primary particle along a surface. You could get a geometry error even if the geometry is correct because FLUKA cannot determine the region
- Precision errors
- Lattice replica $\leftrightarrow$ basic cell mismatch
  (see advanced geometry lecture)

Debugging Tools

- GEOEND card with the DEBUG option
  (handled through a dedicated Flair frame)
- Error messages during simulation in the .err file
- Geometry plotting by Flair
  (automatically invoking the PLOTGEOM card)
- FLAIR Geometry Editor (very powerful! See dedicated lecture)
- Simplegeo
Debugging with FLUKA

**GEOEND** card activates the geometry debugger. Detects both undefined or multiple defined points in a selected X,Y,Z mesh

- Two cards are needed
  - **First card**
    - WHAT(1) = X_{max}
    - WHAT(2) = Y_{max}
    - WHAT(3) = Z_{max}
    - WHAT(4) = X_{min}
    - WHAT(5) = Y_{min}
    - WHAT(6) = Z_{min}
    - SDUM = DEBUG
  - **Second Card**
    - WHAT(1) = Nx
    - WHAT(2) = Ny
    - WHAT(3) = Nz
    - SDUM = &

**WARNING!**

The program stops if too many errors are found
A message will be issued on the output unit
Debugging with FLUKA

- If no error is found, no .err file will be created

- REMINDER: If the debugger does not find any error it does not mean that the geometry is error free!

- One has to test, changing the GEOEND settings especially for critical and complicated regions

- Errors will be listed in the .err file in the form:

  **** Lookdb: Geometry error found ****
  **** The point: -637.623762 -244.554455 -96.039604 ****

  - Point is contained in more than one region
    **** is contained in more than 1 region ****
    **** (regions: 6 7) ****

  - Not contained in any region
    **** is not contained in any region
Auxiliary program: *SimpleGeo*
Auxiliary program: *SimpleGeo*

- *SimpleGeo* is an interactive solid modeler which allows for flexible and easy creation of the models via drag & drop, as well as on-the-fly inspection.
- Imports existing geometries for viewing.
- Creating new geometries from scratch.
- Export to various formats (FLUKA, MCNP, MCNPX).
- Download, Tutorials, etc.: [http://theis.web.cern.ch/theis/simplegeo](http://theis.web.cern.ch/theis/simplegeo)
- Operating system: Windows only.