



Lattice

Beginners' FLUKA Course

# Lattice

FLUKA geometry has *replication* (lattice) capabilities

*Only one level is implemented* (No nested lattices are allowed)

In a future release there will be the possibility of a second level

- The user defines lattice positions in the geometry and provides transformation rules from the lattice to the prototype region:
  1. in the input with the ROT-DEFI card
  2. in a subroutine (`lattic.f`)

The lattice identification is available for scoring

Transformations should include:

Translation, Rotation and Mirroring (only through routine).

## WARNING:

Do not use scaling or any deformation of the coordinate system

# In the geometry

- The regions which constitute the **elementary cell** (*prototype*) to be replicated, have to be defined in detail
- The **Lattices** (*replicas*) have to be defined as “**empty**” regions in their correct location.  
**WARNING:** The lattice region **should map exactly** the outer surface definition of the elementary cell.
- The lattice regions are declared as such with a **LATTICE** card at the end of the geometry input
- In the **LATTICE** card, the user also **assigns lattice names/numbers to the lattices**. These names/numbers will identify the replicas in all FLUKA routines and scoring
- Several basic cells and associated lattices can be defined within the same geometry, one **LATTICE** card will be needed for each set
- **Non-replicas carry the lattice number 0**
- Lattices and plain regions can coexist in the same problem

# LATTICE card

After the Regions definition and before the GEOEND card the user can insert the LATTICE cards

- WHAT(1), WHAT(2), WHAT(3)  
Container region range (from, to, step)
- WHAT(4), WHAT(5), WHAT(6)  
Name/number(s) of the lattice(s)
- SDUM  
blank to use the transformation from the *latic* routine  
ROT#nn to use a ROT-DEFI rotation/translation from input  
name use a rotation by names. You can name a rotation using as SDUM in ROT-DEFI any alphanumeric string you like

## Example

<b>LATTICE</b>	Reg: TARGR1 ▼	to Reg: ▼	Step:
Id: 1tra ▼	Lat: 1.0	to Lat: 1.0	Step: 1.0

\*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...

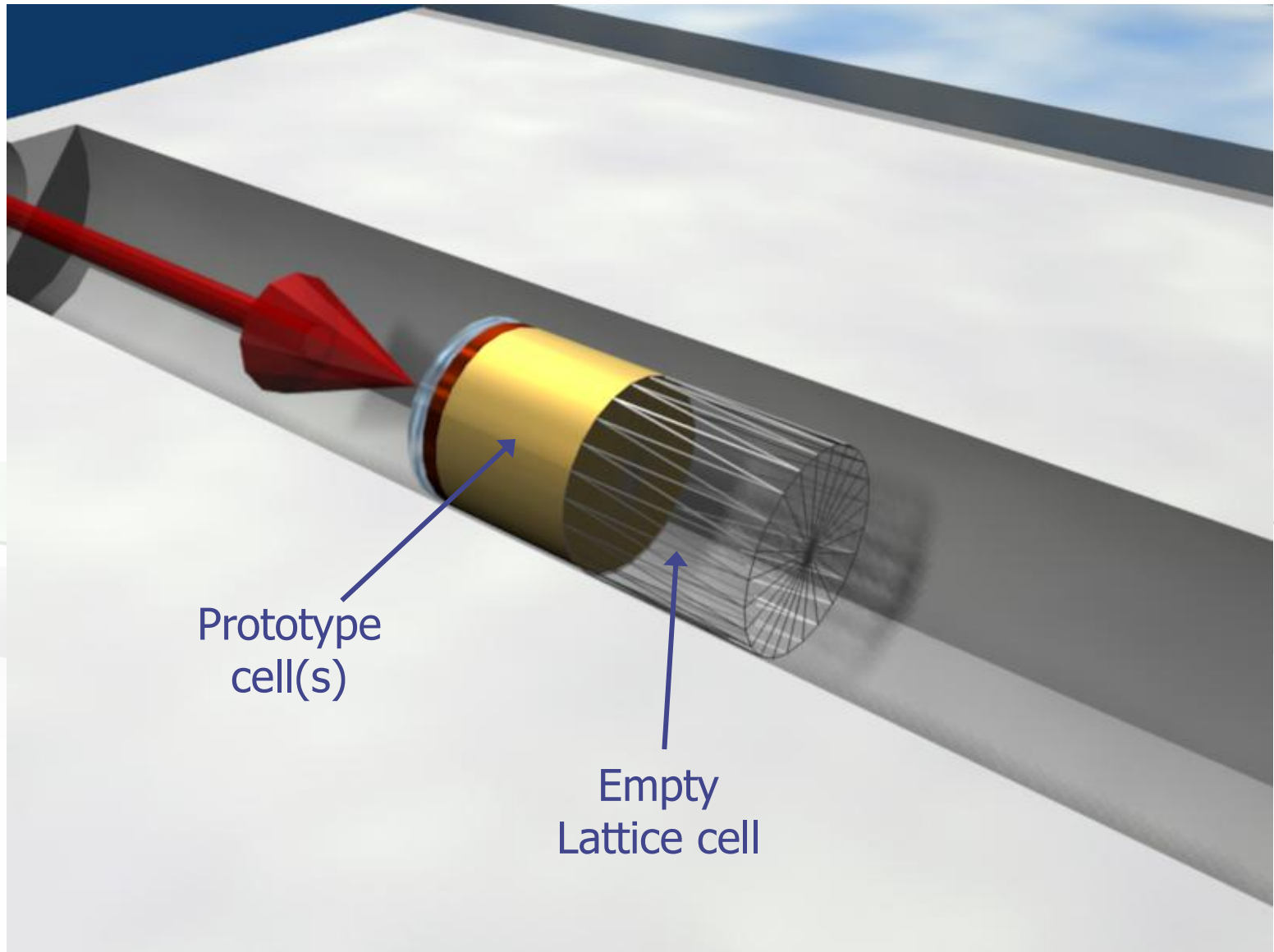
LATTICE	6.00000	19.00000	101.0000	114.00
---------	---------	----------	----------	--------

Region # 6 to 19 are the "placeholders" for the first set replicas. We assign to them lattice numbers from 101 to 114

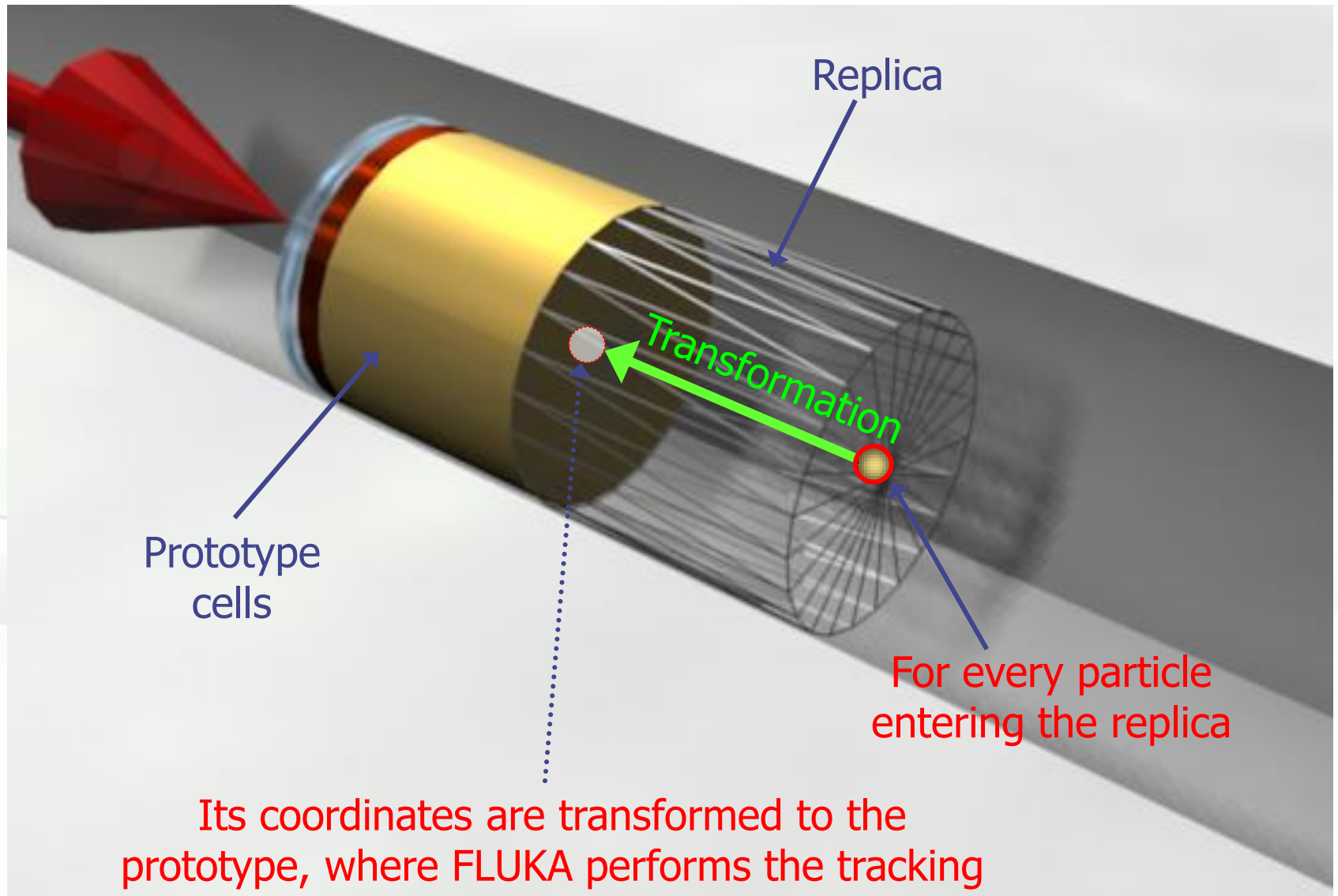
LATTICE	TARGR1	TargRep	1tra
---------	--------	---------	------

TARGR1 is the container region using transformation *1tra*

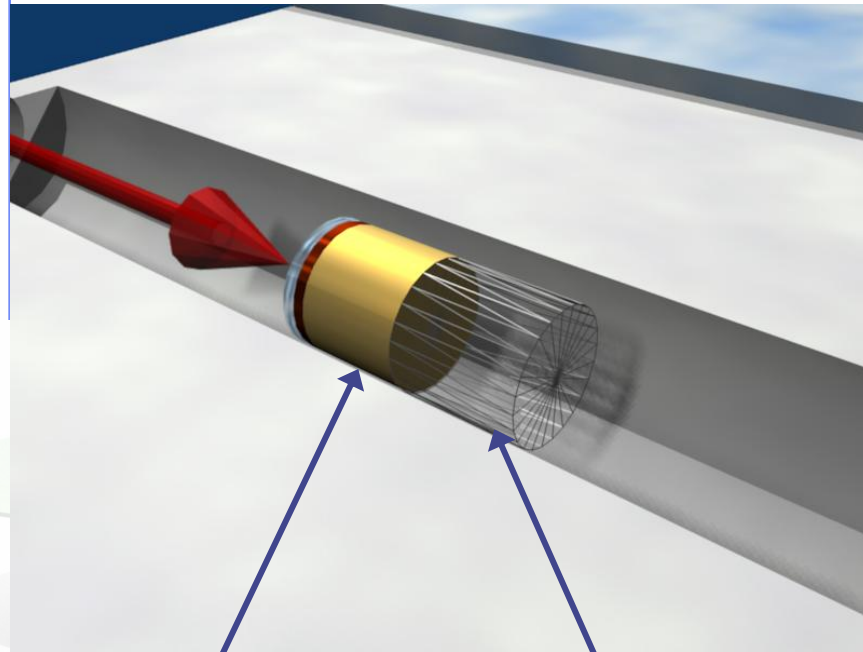
# Plot of the Example



# Plot of the Example

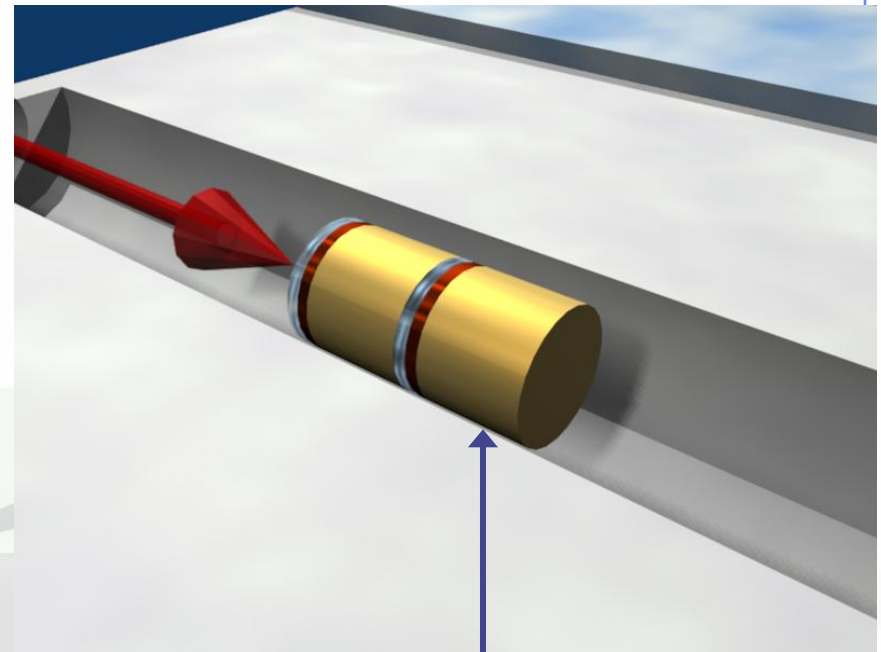


# Plot of the Example



Prototype cell

Empty Lattice cell



Final Replica

# Transformation by input

- Rotations/Translations can be defined with the **ROT-DEFIni** card
- Can be assigned to a lattice by **name** or with **ROT#nnn** SDUM in the **LATTICE** card
- **ROT-DEFIni** cards can be cascaded (using the same **index** or **name**) to define complex transformations

## **WARNING:**

Since matrix multiplication is not commutative the **order** of the Rotation/Translation operations in 3D is important.



# ROT-DEFIni

The ROT-DEFIni defines roto-translations that can be applied to USRBIN, EVENTBIN and LATTICE. It transforms the position of the tracked particle to place of interest scoring or elementary cell (prototype) with the order:

- First applies the translation
- Followed by the rotation on the azimuthal angle
- and finally by the rotation on the polar angle.

$$\mathbf{X}_{\text{new}} = \mathbf{M}_{\text{polar}} \times \mathbf{M}_{\text{az}} \times (\mathbf{X} + \mathbf{T})$$

WHAT(1): assigns a transformation index and the corresponding rotation axis

$\mathbf{I} + \mathbf{J} * 100$  or  $\mathbf{I} * 1000 + \mathbf{J}$

I = index of rotation (WARNING: NOTE THE SWAP OF VARIABLES)

J = rotation with respect to axis (1=X, 2=Y, 3=Z)

WHAT(2): Polar angle of the rotation ( $0 \leq \vartheta \leq 180^\circ$  degrees)

WHAT(3): Azimuthal angle of the rotation ( $-180 \leq \varphi \leq 180^\circ$  degrees)

WHAT(4), WHAT(5), WHAT(6) = X, Y, Z offset for the translation

SDUM: Optional (but recommended) name for the transformation

---

**ROT-DEFI**

Id: 1

Axis: Z ▼

Name: 1tra

Polar: 0.0

Azm:

$\Delta x$ :

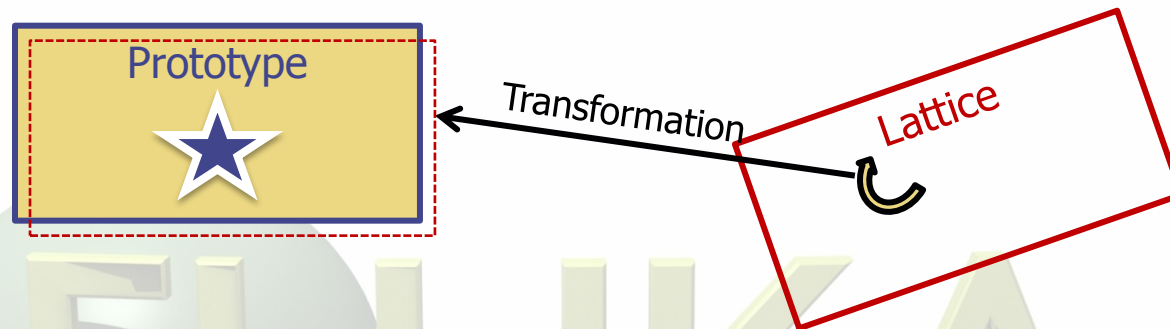
$\Delta y$ :

$\Delta z$ : -10.0

---

# Numerical Precision

- Due to the nature of the floating point operations in CPU even if the transformation is correct the end result could be problematic



This small misalignment between lattice/transformation/prototype could lead to geometry errors

- Use as many digits as possible to describe correctly the prototype and lattice cells as well as the transformation.  
**Is important that the lattice with the transformation corresponds EXACTLY to the prototype**
- In case of need use a **FREE** and **FIXED** card before and after the **ROT-DEFI** to use more than 10 digits
- **GEOBEGIN WHAT(2)** will relax the geometry errors (**USE WITH CAUTION**)

# Input file, and output quantities

- Materials and other properties have to be assigned only to the regions constituting the basic cell(s).
- In all (user) routines the **region number** refers to the corresponding one **in the elementary cell**.
- This applies also to the summary output in the **.out** file, and in the scoring by regions (i.e. the results for a prototype region include also the contribution from its replicas).
- The lattice identity can be recovered by the *lattice number*, as set in the **LATTICE** card, or by GEON2L subroutine if is defined by name
- In particular, the **LUSRBL** routine allows to manage the scoring on lattices in the special **USRBIN/EVENTBIN** structure.

# The USRBIN/EVENTBIN special binning

EVENTBIN or USRBIN with WHAT(1)=8 :

Special user-defined 3D binning. Two variables are discontinuous (e.g. region number), the third one is continuous, but not necessarily a space coordinate.

Variable	Type	Default	Override Routine
1 <sup>st</sup>	integer	region number	MUSRBR
2 <sup>nd</sup>	integer	lattice cell number	LUSRBL
3 <sup>rd</sup>	float	pseudorapidity*	FUSRBV

\* Presently it scores:  $n = -\ln(\tan(0.5 \arctan(\sqrt{x^2 + y^2}/z)))$   
Subject to change to default=**zero** in the next release

# Tips & Tricks [1/2]

- Always remember that the **transformation** refers to the **particles** and not to the geometry!
- You can always divide a transformation into many **ROT-DEFI** cards for easier manipulation.
- Rotations are always around the center of the **geometry**, **and not the center of the object**.
  - To rotate an object, first translate the object to the origin of the axes
  - Perform the rotation
  - Final translation to the requested position.  
Of course with the inverse order since everything should apply to the particle

# Tips & Tricks [2/2]

- The **Geometry transformation editor in flair** can read and write **ROT-DEFI** cards with the transformation requested
- An easy way of creating a replica and the associated transformation is the following:
  1. Select the body defining the outer cell of the prototype
  2. Clone it with (**Ctrl-D**) and change the name of the clones. Click on "**No**" when you are prompted to change all references to the original name.
  3. Open the Geometry transformation dialog (**Ctrl-T**)
  4. Enter the transformation of the object in the listbox
  5. Click on "**Transform**" to perform the transformation on the clone bodies
  6. Click on "**Invert**" button to invert the order of the transformation
  7. Enter a name on the "**ROT-DEFIni**" field and click "**Add to Input**" to create the **ROT-DEFIni** cards
  8. Now you have to create manually the correct **regions** and the **LATTICE** cards