Advanced Geometry

Universe & Fill

'Like m But' & TRCL

Lattices & Fill

Universe

&

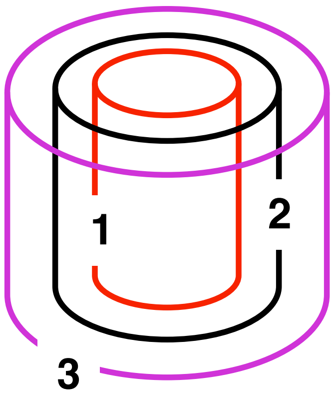
Fill

Universe Card, U

- One or more cells can be grouped together into a collection, called a **universe**.
- A universe is either
  - a lattice cell OR
  - a collection of standard cells
- Form: u=#
  - placed on the cell cards, after the surface info
  - # can be any number, u= numbers need not be sequential
  - # must also appear on a fill= entry on another cell card (container)
  - All cells with the same u=# form a universe that fills another cell.
- Cells of a universe can be finite or infinite, but must fill all of the space inside the container cell they fill.
- Surfaces of a universe CAN be coincident with the cell they fill. (but, avoid this if you can)

Universe Example

- Reactor fuel rod with gap & cladding, surrounded by infinite moderator




```
c   CELLS
10   110    0.069256    -1        u=9    $ fuel
20   0                   1 -2     u=9    $ gap
30   120    0.042910     2 -3     u=9    $ clad
40   130    0.100059     3        u=9    $ water, infinite

c  SURFACES
1   RCC    0. 0. 0.      0. 0. 360.     0.43
2   RCC    0. 0. 0.      0. 0. 360.     0.44
3   RCC    0. 0. 0.      0. 0. 360.     0.49
```

- Universe 9 consists of cells 10, 20, 30, 40 - the fuel, gap, clad, & water
- Note that the Cell 40 (water) is infinite
- Universe 9 can be used to "fill" another cell (container cell), or to create a lattice of fuel rods

Cell Fill Card, FILL

- **Fill a cell or lattice element with a universe**
- **Form: fill=#**
  - **placed on the cell cards, after the surface info**
  - **# is the number of a universe**
  - **Variations:**

fill=# (n)	where n is optional transformation
fill=# (...)	where ... are optional TR entries
***fill=# (...)**	optional TR entries in degrees between this cell and filling universe

- **Usually, the cell being filled will contain a void material, since the material numbers and densities were assigned to the cells in the filling universe**
- **Filled cell is a "window" - clips away any part of the filling universe which extends beyond the cell boundary**
- **Surfaces of filled cell and filling universe can be coincident (but, avoid this if possible)**

Example

Problem puc2

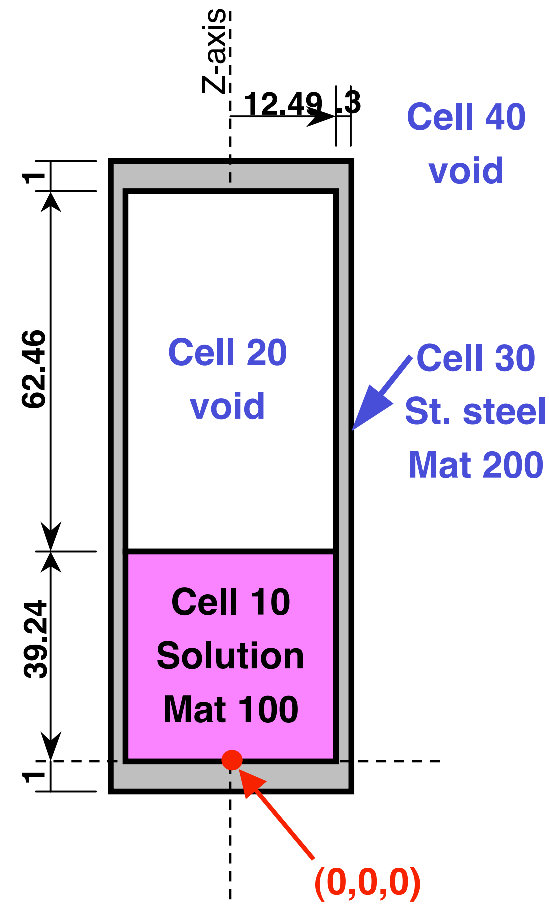
Use UNIVERSE for solution & inner void

&

FILL the steel can with that universe

Problem puc2

- **Copy file puc1 to puc2**
- **Edit file puc2**
  - Modify definitions of cells 10 & 20:
    - Identify cells 10 & 20 as **being in universe 1**
    - Remove surface 1 from their definitions
    - Both cells are infinite (in universe 1)
  - Define a cell (25) for the interior of the container
    - Bounded by the inner surface of the container (1)
    - **Fill it with universe 1**
    - Don't forget to add imp:n=1



- **Run the problem, with**

```
kcode  1000  1.0  25  100
```

- **Note that the answer is identical to the previous run**

Problem puc2 - Comments

- **Universe 1 is infinite**
  - Infinite void above surface 3, infinite solution below surface 3
  - Because cells 10 & 20 are infinite, MCNP can' t compute their volume, & uses Volume=0 in the output

- **Cell 25 is filled with universe 1**
  - Universe 1 is clipped by the container cell (cell 25)
  - The container (cell 25) must be completely filled by the embedded universe (of course it is, since universe 1 is infinite…)

- **Plotting**
  - "XY" plot
  - See what happens when "Level" is changed -- Level 0, Level 1 (must click on "Redraw" to refresh the plot after changing Level)

- **Results**
  - Same as previous runs
  - Not always true when you use universe/fill - might have different roundoff …

'Like m But'

&

TRCL

'Like m But' Card

- "LIKE m BUT" cell description provides shorthand method for repeating similar cells

- Form: j LIKE m BUT list
  - Cell j takes all attributes of cell m except parameters in 'list'
  - Cell m must be defined before "j like m but" in INP file

- Parameters that can make up 'list' include:
  - imp, vol, pwt, ext, fcl, wwn, dxc, nonu, pd, tmp
  - u, trcl, lat, fill
  - mat, rho
  - U and/or TRCL, at minimum, must be in 'list'
  - Examples:

```
17   like 70 but    trcl=( 1 1 2)   u=66
23   like 70 but    mat=13  u=2
```

- Surface numbers cannot be altered with "like m but" format

Translation & Rotation

- Surfaces can be translated/rotated using the **TR** card
- Cells can be translated/rotated using the **TRCL** card
- **Forms:**
  - **translate CELL by (dx,dy,dz):**

    TRCL=(dx dy dz)

  - **Translate & rotate CELL:**

    TRCL=(dx dy dz xx' yx' zx' xy' yy' zy' xz' yz' zz')

    where

    xx' = cosine of angle between original x-axis and new x'-axis

    xy' = … similar …

  - **Translate & rotate CELL:**

    ***TRCL=(dx dy dz xx' yx' zx' xy' yy' zy' xz' yz' zz')**

    where

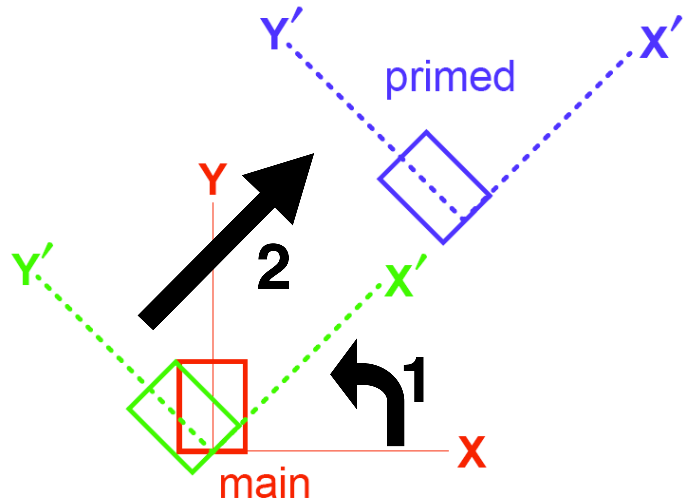
    xx' = angle in degrees between original x-axis and new x'-axis

    xy' = … similar …

- **Rotation is done first, then translation**

Translation & Rotation

- Rotation (red to green axes) is done first (in original coord system)
- Translation (green to blue axes) is done second (in original coord system)



- When TRCL is used, MCNP must create new surfaces
  - The new surfaces are assigned numbers of the form:

    1000 * (new-cell-number) + (original-surface-number)

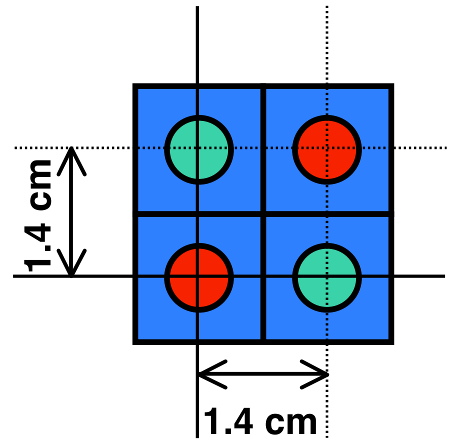
  - Be careful to avoid those surface numbers in the rest of your input
  - If you use TRCL, make sure your surface numbers are <1000 !
- All universes that fill this cell <u>inherit</u> the TRCL

'Like m But' & TRCL - Example

- Cluster of several fuel rods, with different enrichments

```
c Cell Cards
c ----- red universe, 7 -----
1 110  .069  -11  u=7      $ fuel-red
2 120  .100   11  u=7      $ water
c
c ----- green universe, 8 -----
3 130  .069  -11  u=8      $ fuel-green
4 120  .100   11  u=8      $ water
c
c ----- real world -----
5 0       -12  fill=7      $ unit cell,  lower left, at origin
6 like 5 but  fill=8 trcl=(  0 1.4 0) $ upper left
7 like 5 but  fill=8 trcl=(1.4   0 0) $ lower right
8 like 5 but  fill=7 trcl=(1.4 1.4 0) $ upper right

c Surfaces
11   RCC   0. 0. -180.     0. 0. 360.     0.49
12   RPP  -.7 .7   -.7 .7   -180. 180.
```

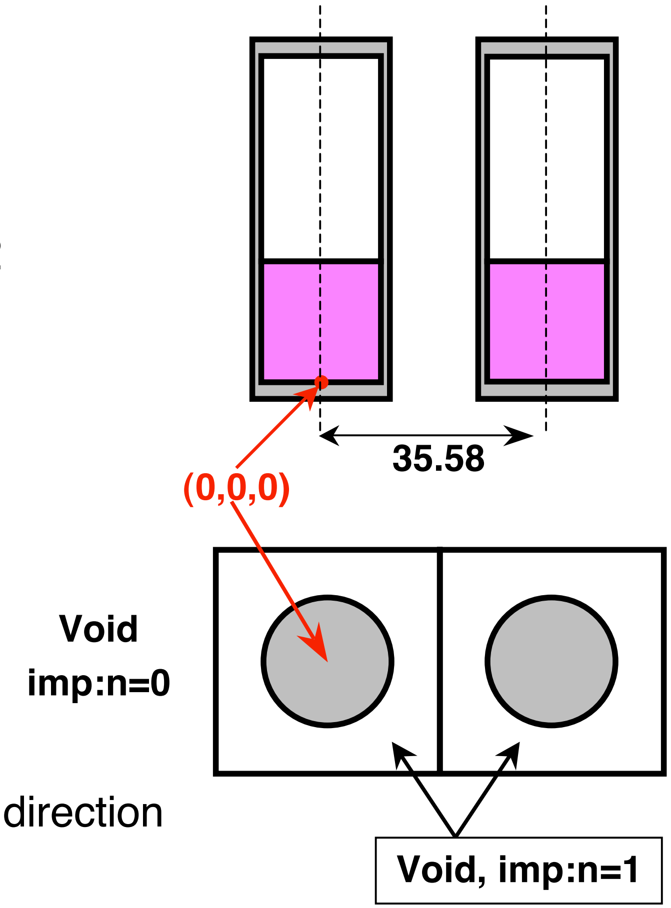


Problem puc3

Two Cans of Solution Using ‘Like m But’ and TRCL

Problem puc3

- **Two cans, 35.58 cm separation between centers**
- **Copy file puc2 to puc3**
- **Edit file puc3**
  - Identify cells 25, 30, 40 as being in universe 2 (change the importance of cell 40 to imp:n=1)
  - Define cell 50 & surface 4
    - A box around the first can (cell 30)
    - Use RPP, x,y in range (-17.79,17.79),
      z in range (-1,102.7)
    - FILL cell 50 with universe 2
    - Importance = 1
  - Define cell 60
    - Same as cell 50, but translated 35.58 cm in +x direction
  - Define cell 99
    - Void, importance = 0, outside of 50 & 60
  - Add another source point to KSRC, at (35.58, 0., 19.62)



Problem puc3

```
puc3 - TWO cylinders
C ----- universe 1, infinite solution & void -----
10  100      9.9270e-2  -3          u=1  imp:n=1   $ infinite solution
20  0                    3          u=1  imp:n=1   $ infinite void
C ----- universe 2, filled can & infinite exterior -----
25  0                   -1 fill=1   u=2  imp:n=1   $ inside of can, filled
30  200      8.6360e-2   1 -2       u=2  imp:n=1   $ can
40  0                    2          u=2  imp:n=1   $ infinite exterior
C ----- real world, 2 boxes (contining cans) & infinite exterior -----
50  0                   -4 fill=2        imp:n=1   $ 1st box at origin, with can
60  like 50 but  trcl=(35.58  0.  0.)              $ 2nd box shifted, with can
99  0                    #50 #60         imp:n=0   $ exterior to both boxes

1       RCC   0.  0.   0.     0.  0. 101.7     12.49
2       RCC   0.  0.  -1.     0.  0. 103.7     12.79
3       pz    39.24
4       RPP  -17.79 17.79   -17.79 17.79   -1. 102.7

kcode   1000 1.0 25 100
ksrc    0.  0.  19.62         35.38 0.  19.62
m100     1001   6.0070e-2     8016   3.6540e-2
         7014   2.3611e-3    94239   2.7682e-4
mt100    lwtr
m200    26056   5.8068e-2
```

Result:

keff = 0.96660 ± 0.00357

Problem puc3

- **Universe 1 is infinite**
  - Same as before, but now appears in 2 different places
  - Clipped by surface 1 when if FILLs cell 25
- **Universe 2 is infinite**
  - Can (containing universe 1) & exterior void
  - Embedded in Cell 50, and also in cell 60
- **Plotting**
  - "XY" plot, "ZX" plot
  - See what happens when "Level" is changed -- Level 0, Level 1, Level 2
    (must click on "Redraw" to refresh the plot after changing Level)
  - Note surface 60004 -- what happened to other translated surfaces?
    (See output file for info on identical surfaces…)
  - Click on "MBODY On" - note the surface "facets" (internal label for body surfaces)
- **Results**
  - Higher Keff, as expected

Lattices

&

Fill

Cell Lattice Card - LAT

- **Defines cell as infinite array or lattice**
  - User input describes central element [0,0,0] in lattice
  - MCNP replicates the central element in all 3 directions

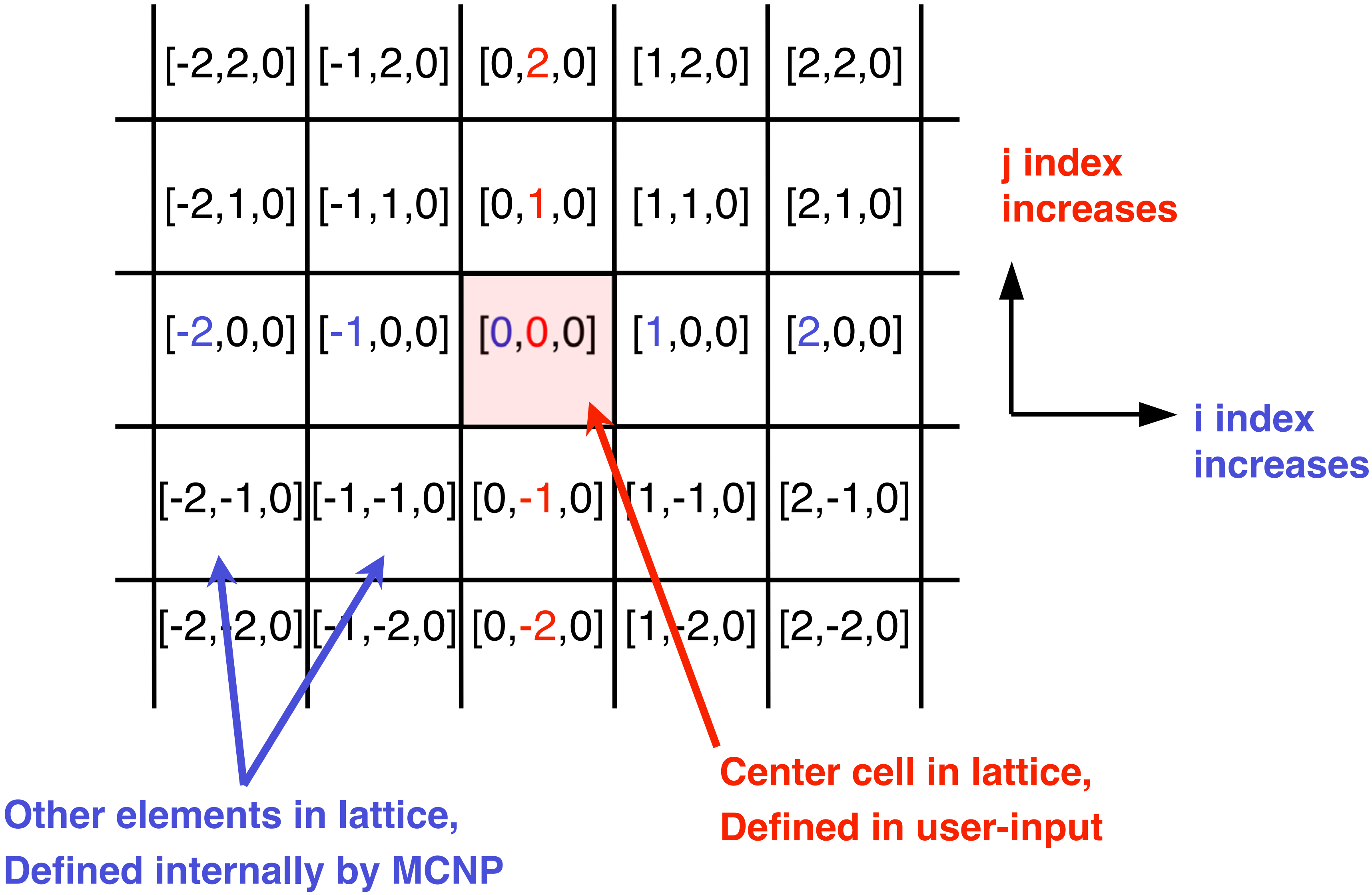
- **Form:** **LAT=1** **hexahedra (six face solid) square**
  LAT=2 **hexagonal prism (eight) triangle**
  - **LAT=# should go on a cell card, after surface info**

- **Space between elements must be filled exactly:**
  - hexahedra need not be rectangular
  - hexagonal prisms need not be regular
  - **Opposite sides of central element must be parallel**

- **Lattice elements can be infinite along 1 or 2 axes**

- **Order of surfaces on the cell card is important**
  - Macrobody will always increment along +axis

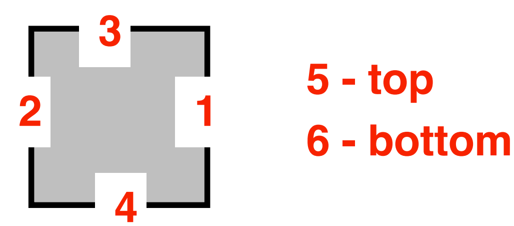
Lattice Indexing



Lattice Element Indexing

- **Elements identified by [i,j,k] labels determined by the <u>order of surface entries on cell card</u>**

- **Cell card specifies the [0,0,0] element**




```
11   0   -1  2    -3  4    -5  6   lat=1
```

  – For LAT=1, at least 4 surfaces or 2 vectors required

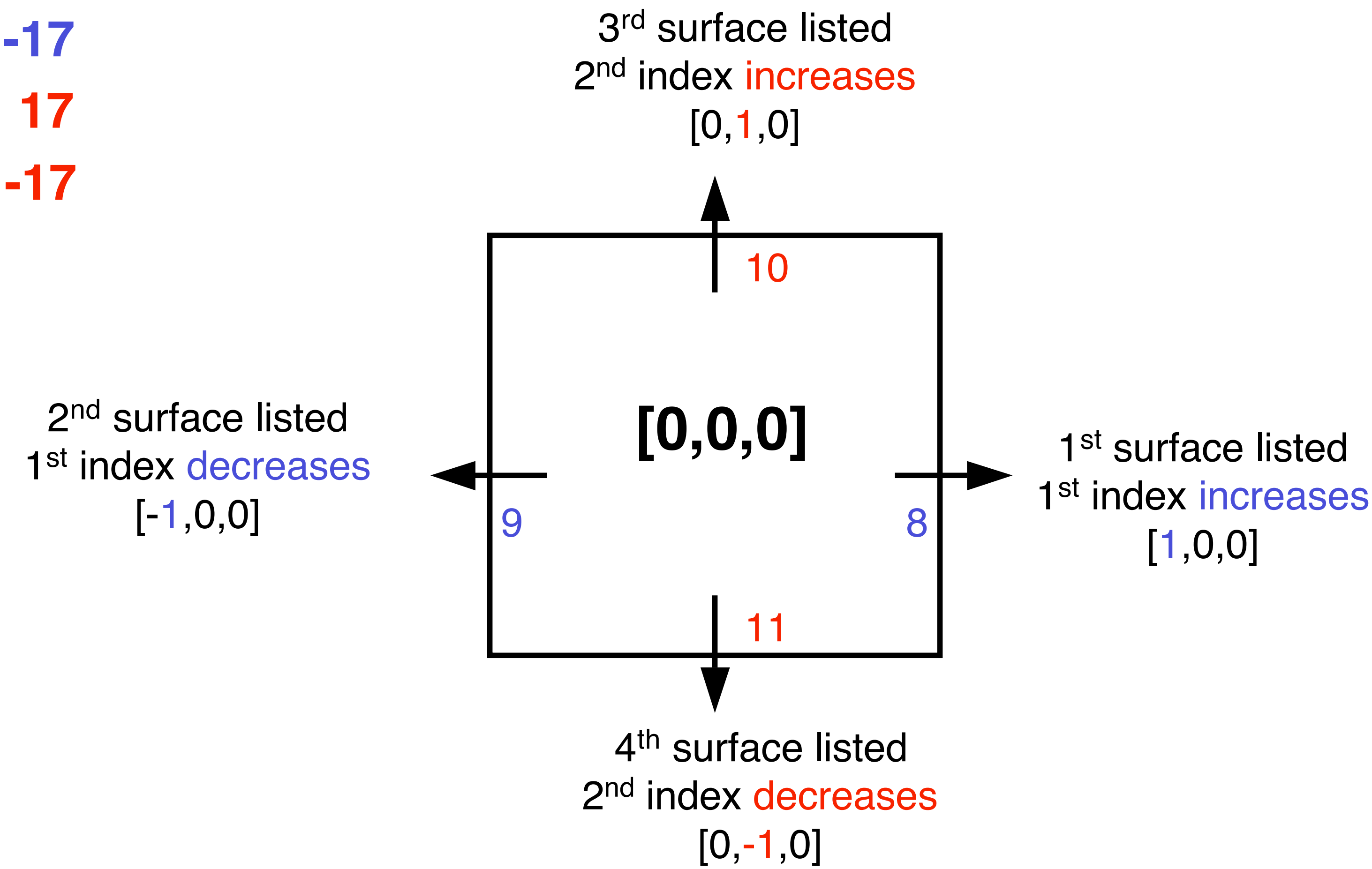
  – On + side of 1st surface = [1, 0, 0] lattice element
  – - side of 2nd [-1, 0, 0]
  – + side of 3rd [0, 1, 0]
  – - side of 4th [0,-1, 0]
  – + side of 5th [0, 0, 1]
  – - side of 6th [0, 0,-1]

- **If you don't list the surfaces in the order shown above, everything will get very confusing & you will have trouble.**

Lattice Element Indexing using Surfaces

5 0 -8 9 -10 11 lat=1

8 px 17
9 px -17
10 py 17
11 py -17



Macrobodies, Facets, & Lattices

- **For macrobodies, MCNP internally replaces the body with a set of surfaces**
  - The surfaces created have the form S.F
  - "S" is the original surface number for the macrobody
  - "F" is a 'facet number', 1, 2, …

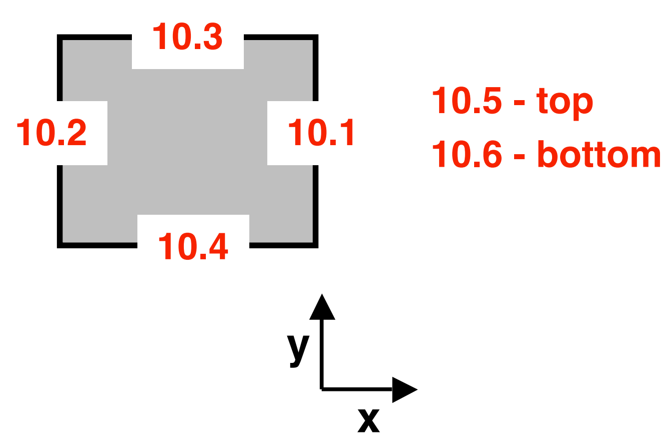
  - These cell & surface cards in MCNP input

```
25   111    -1.0     -10   lat=1     $ cell card

10   RPP   -1 1   -2 2    -3 3       $ surface card (body)
```

generate these surfaces internally

10.1	- "px" plane at x= 1
10.2	- "px" plane at x=-1
10.3	- "py" plane at y= 2
10.4	- "py" plane at y=-2
10.5	- "pz" plane at z=3
10.6	- "pz" plane at z=-3



Lattice Element Indexing using Macrobody

5 0 -12 lat=1

For infinite in z-dir, use 0 0

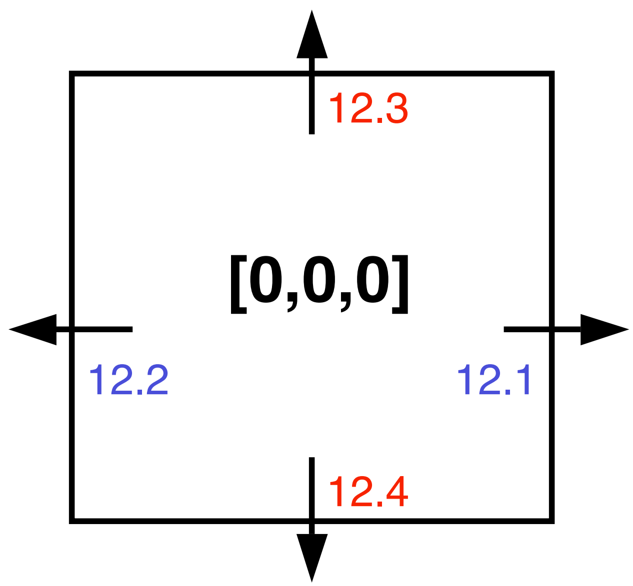
12 RPP -17. 17. -17. 17. -180. 180.

Plane at +Y

2^{nd} index increases [0,1,0]

Plane at -X

1^{st} index decreases [-1,0,0]



plane at +X

1^{st} index increases [1,0,0]

Plane at -Y

2^{nd} index decreases [0,-1,0]

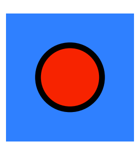
LAT - Example

9 x 9 array of fuel rods

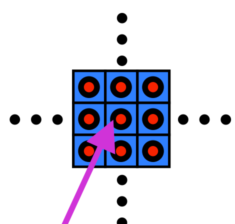
```
c Cells
1 110  .069  -10  u=7      $ fuel
2 120  .100   10  u=7      $ infinite water
c
5 0     -20  fill=7  lat=1  u=9  $ infinite lattice of pins
6 0     -30  fill=9              $ box with 9x9 pins

c Surfaces
10 RCC  0. 0. 0.   0. 0. 360.  0.49    $ cylinder for fuel
20 RPP  -.7 .7   -.7 .7   0 360        $ box for single pin
30 RPP  -6.3 6.3  -6.3 6.3  0 360      $ box holds 9x9 pins
```

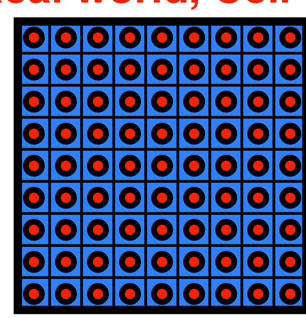
Universe 7



Universe 9



Real world, Cell 6



User defines center cell,
MCNP replicates into infinite lattice

Outer box (surface 30)
truncates infinite lattice

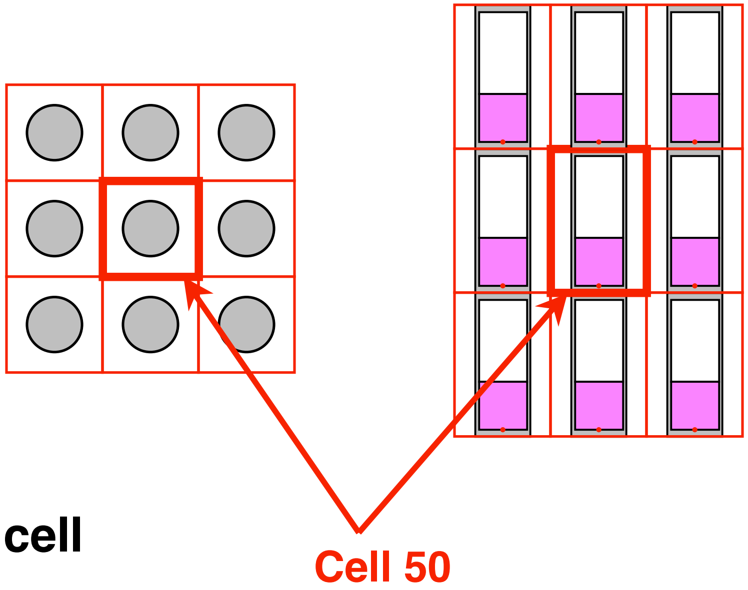
Problem puc4

Infinite Lattice of Cans

(3D Lattice)

Problem puc4

Define the center lattice cell,
filled with universe 2



- **Copy file puc3 to puc4**
- **Edit file puc4**
  - Delete cells 60 & 99
  - **Declare Cell 50 to be the center lattice cell in a hexahedral (box) lattice, LAT=1**
  - **Add this data card (turns off entropy calc for infinite lattice):**

```
hsrc  1 -1.e10 1.e10  1 -1.e10 1.e10  1 -1.e10 1.e10
```

  - **Remove the second point in KSRC**
  - **Plot: mcnp5 i=puc4 ip**
  - **Compute keff: mcnp5 i=puc4**

Comments - puc4

- **Keff is pretty large**
  - Infinite lattice, no leakage, no absorbers, …

- **Note that lattices are:**
  - Defined by creating the center cell & flagging it with LAT=1
  - Filled with 1 or more universes
  - Infinite in extent

- **How do you get a finite lattice?**

  1. **Make an infinite lattice, then give it a universe number**

  2. **Create a container cell to hold some portion of the lattice**

  3. **Then fill that container cell with the lattice universe**
     - **Infinite lattice is clipped (truncated) by the container cell boundaries**
     - **Lattice elements outside the container can never be reached**

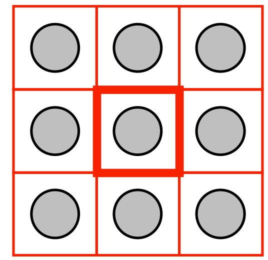
Problem puc5

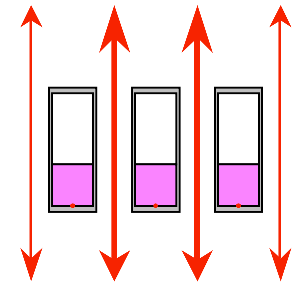
Infinite Lattice of Cans

(2D Lattice)

Problem puc5

Define a 2D lattice (x-y plane) of cans





- Copy file puc4 to puc5
- Edit file puc5
  - Change surface 4 (RPP body that defines center cell)
    - Make the RPP infinite in the z-direction
      - Use "0. 0." for the z top/bottom
      - This tells MCNP that the RPP is ininite in z-direction (no top/bottom)
  - Need to consider neutrons above & below the cans
    - Want void with imp:n=1 <u>between</u> cans
    - Want void with imp:n=0 <u>above & below</u> cans
      (to prevent neutrons from streaming forever…)
    - Need to add Cell 45 to universe 2, above can & below can, imp:n=0
      Could do this with 2 extra surfaces or use existing macrobody facets
  - **Plot:** mcnp5 i=puc5 ip
  - **Compute keff:** mcnp5 i=puc5

Comments - puc5 & puc5m

- **Keff is more reasonable**
  – Infinite lattice in 2D, leakage in Z, no absorbers, …
  – Same result for both puc5 & puc5m
- **File puc5 - straightforward**
  – Extra cell & surfaces to define voids in between cans with imp:n=1, and above/below cans with imp:n=0
- **File puc5m**
  – Similar to puc5, but no extra surfaces needed
  – Use existing top of can (facet 2.2) and bottom of can (facet 2.3)
  – **Major source of confusion: the sign (sense) for facet 2.3**
    - For macrobodies, MCNP internally translates the body definition into a collection of surfaces: infinite cylinder (2.1), top plane (2.2), bottom plane (2.3)
    - **By definition, MCNP considers inside the body to have negative sense, & outside the body to have positive sense**
    - **MCNP alters the surface definitions to match the body sense convention**, hence inside the body has negative sense wrt surface 2.3 and outside the body has positive sense wrt surface 2.3, opposite to the normal surface sense conventions for 2.3

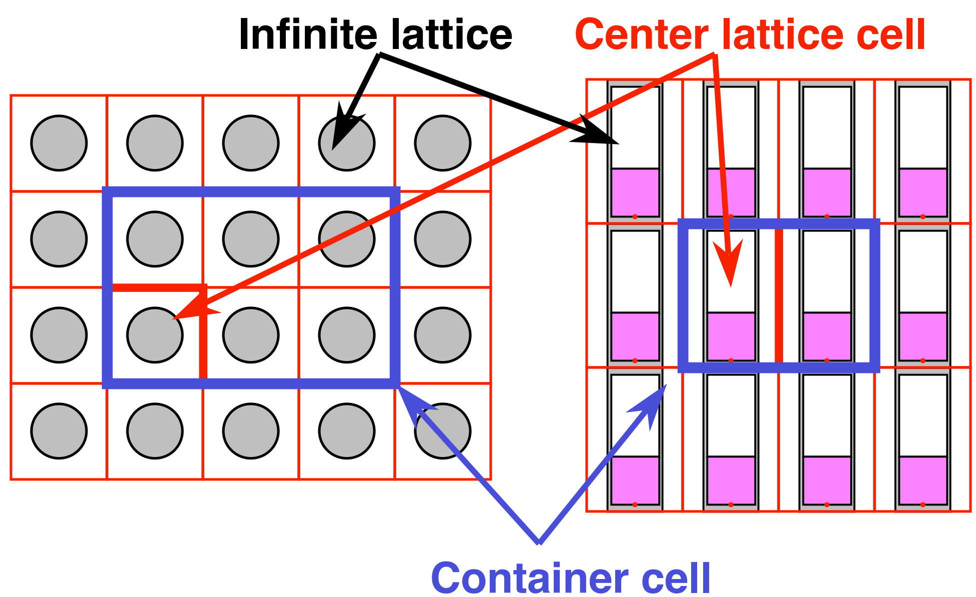
Problem puc6

Finite 2x3 Lattice of Cans

Problem puc6

Start with infinite 3D lattice (Problem puc4) & create a 3x2 array of cans

- **Copy file puc4 to puc6**
- **Edit file puc6**
  - Declare cell 50 as universe 3
  - Define the container cell 60, and an RPP body sized to hold 2x3 array of cell 50
  - Fill cell 60 with universe 3
  - Define cell 99, outside container, imp:n=0



- **Modify the HSRC card (optional):**

```
hsrc   3 -17.79 88.95   2 -17.79 53.37   1 -1. 102.7
```

- **Plot:** **mcnp5 i=puc6 ip**
- **Compute keff:** **mcnp5 i=puc6**

Comments - puc6

- **Keff is larger than 1.0 ….**

- **Note that lattices are:**
  - Defined by creating the center cell & flagging it with LAT=1
  - Filled with 1 or more universes
  - Infinite in extent

- **To get a finite lattice:**

  1. **Make an infinite lattice, then give it a universe number**

  2. **Create a container cell to hold some portion of the lattice**

  3. **Fill the container cell with the lattice universe**
     - **Infinite lattice is clipped (truncated) by the container cell boundaries**
     - **Lattice elements outside the container can never be reached**

FILL Card Fully Specified for Lattices

- **When 'filling' a cell which is a lattice, you can specify which universe goes into each individual lattice element**

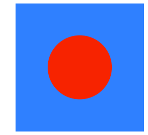
- **Infinite Lattice Form: fill = n**
  - Fill **all** lattice elements with universe **n**

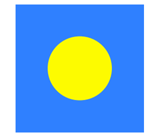
- **Finite Lattice Form: fill = i1:i2 j1:j2 k1:k2 N_1 (...) N_2 (...) etc**
  - **i1:i2 j1:j2 k1:k2**
    defines which elements of the lattice exist (i1≤ i2, j1≤j2, k1≤k2)
  - **N_1, N_2, etc**
    list of filling **universe numbers** that specify what universe fills each lattice element
  - **Order of array entries follows FORTRAN convention:**
    (i1,j1,k1), (i1+1,j1,k1), (all i,j1,k1), ... (all i,j2,k1)..... (all i,j3,k1) …..
  - For this fill card: `fill= -2:2 0:1 0:0`
    5 x 2 x 1 entries are required, as in

```
fill= -2:2 0:1 0:0  1   2   3   2   1
                    2   1   1   1   2
```

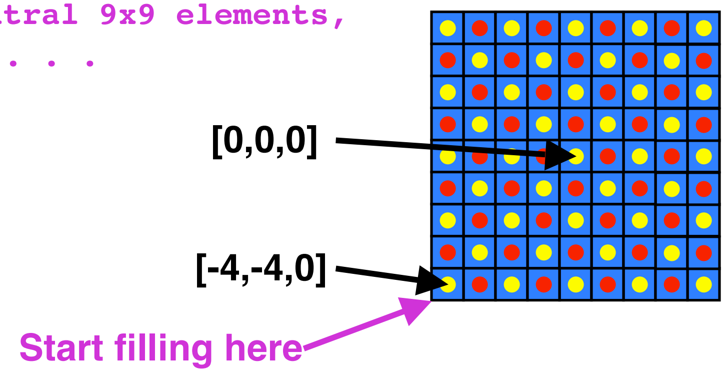
LAT - Example

- **Finite lattice, 9x9 checkerboard arrangement**





```
c Cell Cards
1 110  .069  -10  u=7      $ fuel-red
2 120  .100   10  u=7      $ infinite water
c
3 130  .069  -10  u=8      $ fuel-yellow
4 120  .100   10  u=8      $ infinite water
c
5 0          -20  u=9 lat=1   $ lattice of pin-cells
         fill= -4:4  -4:4  0:0          $ only fill central 9x9 elements,
          8 7 8 7 8 7 8 7 8   $   start at bottom-left . . .
          7 8 7 8 7 8 7 8 7
          8 7 8 7 8 7 8 7 8
          7 8 7 8 7 8 7 8 7
          8 7 8 7 8 7 8 7 8
          7 8 7 8 7 8 7 8 7
          8 7 8 7 8 7 8 7 8
          7 8 7 8 7 8 7 8 7
          8 7 8 7 8 7 8 7 8
6 0     -30  fill=9           $ box with 9x9 pins,
                              $   chops off unused lattice locations
```




```
c Surfaces
10  RCC  0. 0. 0.     0. 0. 360.     0.49
20  RPP  -.7 .7    -.7 .7    0 360
30  RPP  -6.3 6.3  -6.3 6.3  0 360  $ box holds 9x9 pins
```

Fill Card - Special Case

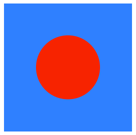
- Special value for the fill array: own universe number means element is filled with material on cell card

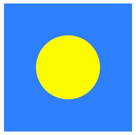
```
11  300  -1.0 -22  lat=1  u=9  fill= -1:1 -1:1 0:0
                                        9  9  9
                                        9  1  9
                                        9  9  9
```

  - The lattice elements in Cell 11 that are filled with Universe 9 (the universe assigned to this cell) are actually filled with Material 300
  - Note that when this special case is used, there can be no geometric detail inside of the lattice element

LAT - Example Using Special Case of Fill

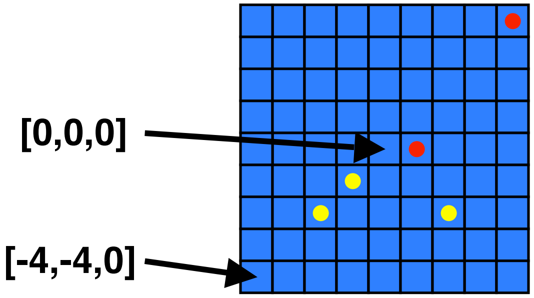
- **Finite lattice, 9x9 arrangement with only a few fuel pins**





```
c Cell Cards
1 110  0.069  -10  u=7     $ fuel-red
2 120  0.100   10  u=7     $ infinite water
c
3 130  0.069  -10  u=8     $ fuel-yellow
4 120  0.100   10  u=8     $ infinite water
c
5 120  -1.0  -20  u=9 lat=1          $ lattice of pin-cells
        fill= -4:4  -4:4  0:0        $ only fill central 9x9 elements,
     9 9 9 9 9 9 9 9 9   $   start at bottom-left . . .
     9 9 9 9 9 9 9 9 9
     9 9 8 9 9 9 8 9 9
     9 9 9 8 9 9 9 9 9
     9 9 9 9 9 7 9 9 9
     9 9 9 9 9 9 9 9 9
     9 9 9 9 9 9 9 9 9
     9 9 9 9 9 9 9 9 9
     9 9 9 9 9 9 9 9 7
6 0     -30  fill=9      $ box with 9x9 pins,
                         $   chops off unused lattice locations

c Surfaces
10  RCC  0.0 0.0 0.0     0.0 0.0 360.0     0.49
20  RPP  -0.7 0.7  -0.7 0.7  0 360
30  RPP  -6.3 6.3  -6.3 6.3  0 360  $ box holds 9x9 pins
```



General Suggestions

- **Don't set up geometry all at once**
  - start with small pieces,   plot each as you go along
- **Always plot geometry !!!!!**
  - To see if it's correctly defined
  - To see if it's what you intended to define
- **Keep cells reasonably simple**
- **Use parentheses freely for clarity**
- **Only as much geometry detail as required for accuracy**
- **Check MCNP-calculated mass and volume against hand-calculated values**
- **2-D slices thru more than one plane**
  - Move plot plane origin around
  - Don't put plot plane directly on a surface
- **If all else fails…**
  - Use VOID card with inward-directed source
  - Lost particle: set plot origin to **xyz** location of lost particle,
    use **uvw** for plot basis vector, zoom-in with plotter

Hexagonal Geometry

(Advanced Topic - Time Permitting)

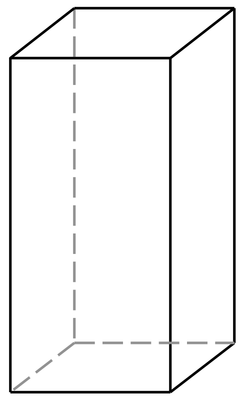
Hexagonal Lattice

"Criticality Calculations with MCNP: A Primer," LA-UR-04-0294

Hexahedra

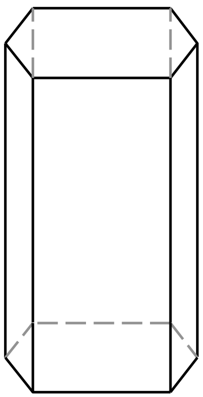
(Quadrilateral

Prism)



Hexagonal

Prism



- **Opposite sides must be identical and parallel.**
- **Hexagonal prism cross section must be convex**
- **Height of hexagonal prism can be infinite (if defined with surfaces)**

Macrobody - Right Hexagonal Prism

- RHP and HEX are the same card

- RHP or HEX Card:

RHP *v1 v2 v3* *h1 h2 h3* *r1 r2 r3* *s1 s2 s3* *t1 t2 t3*

v1 v2 v3 = x, y , z coordinates of the bottom center of hex

h1 h2 h3 = vector from bottom to top, magnitude = height
for a z-hex with height h, h1 h2 h3 = 0 0 h

r1 r2 r3 = vector from the axis to the middle of the 1st facet,
for a pitch 2p facet normal to y-axis, r1 r2 r3 = 0 p 0

s1 s2 s3 = vector to center of the 2nd facet

t1 t2 t3 = vector to center of the 3rd facet

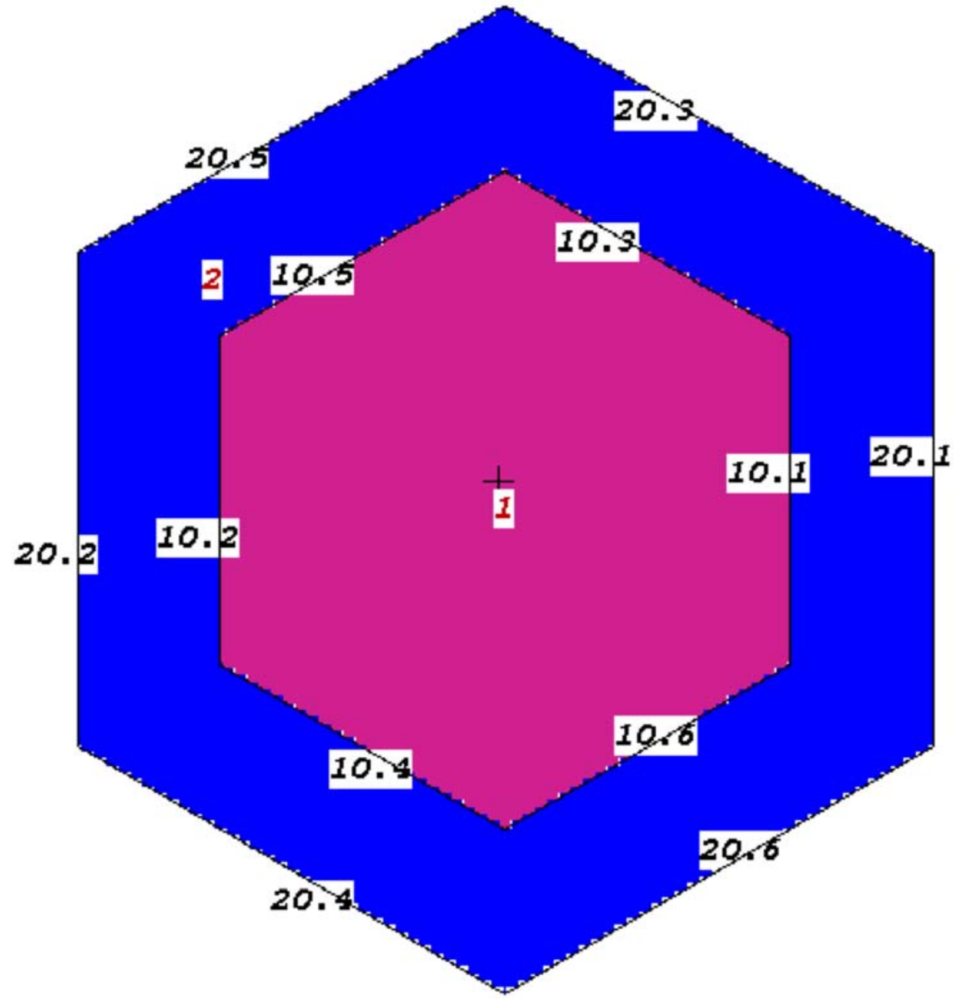
Macrobody - Right Hexagonal Prism Example

- **Example:**

```
1 101 -1.0   -10
2 102 -7.8    10  -20

10 rhp 0 0 -4   0 0 8   2. 0 0
20 rhp 0 0 -4   0 0 8   3. 0 0
```

Center of base = (0,0,-4)
Height = 8, "out of the paper"
Surface 10.1 is x=2, 10.2 is x=-2
Surface 20.1 is x=3, 20.2 is x=-3



Hexagonal Lattice Element Indexing

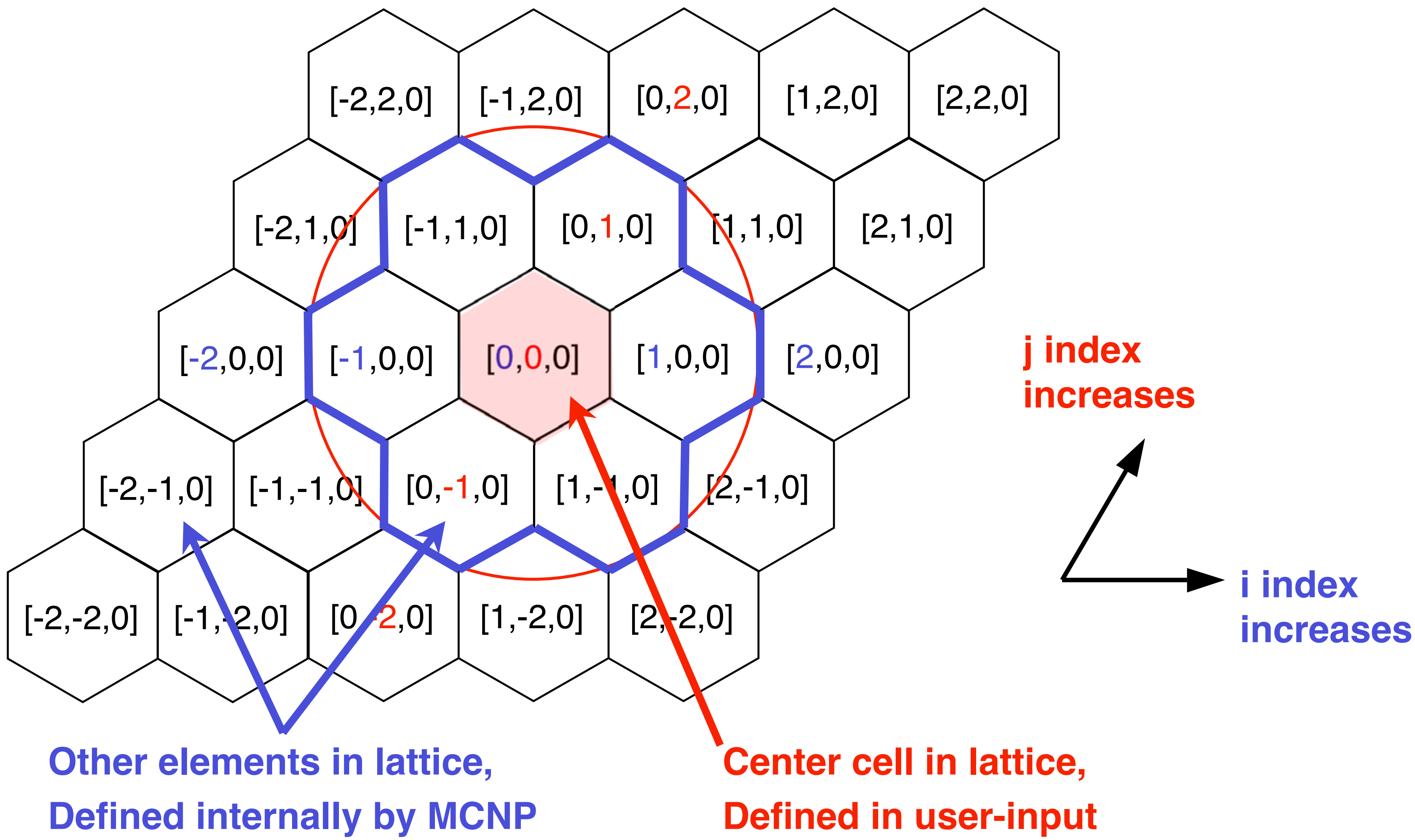
- **At least v1 v2 v3 h1 h2 h3 r1 r2 r3 required**

- **i,j,k lattice element indexing determined by RHP vectors**

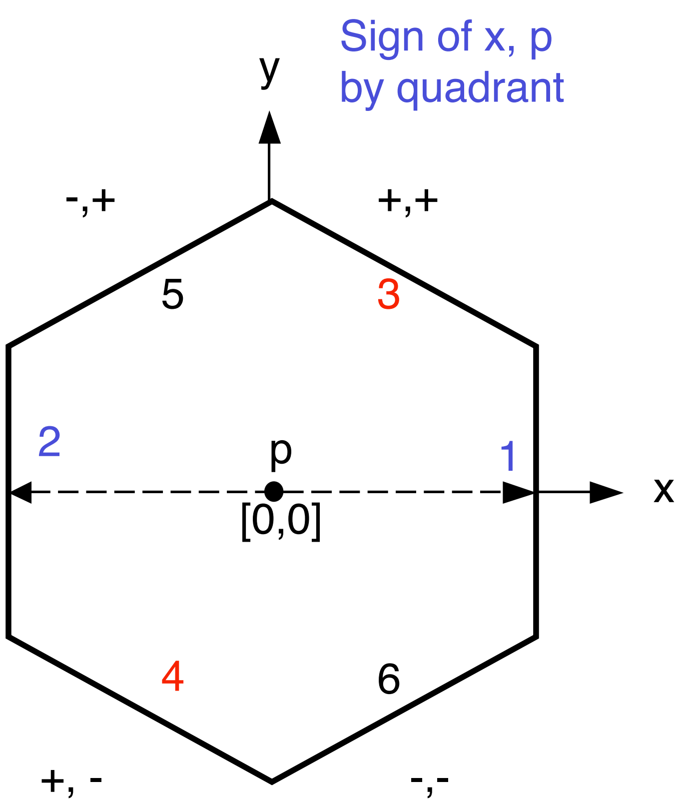
Beyond:

		i	j	k	
– Plane normal to END of	r1 r2 r3 = [	1,	0,	0]	element
– Plane normal to BEGINNING of	r1 r2 r3 = [	-1,	0,	0]	element
– Plane normal to END of	s1 s2 s3 = [	0,	1,	0]	element
– Plane normal to BEGINNING of	s1 s2 s3 = [	0,	-1,	0]	element
– Plane normal to END of	t1 t2 t3 = [	-1,	1,	0]	element
– Plane normal to BEGINNING of	t1 t2 t3 = [	1,	-1,	0]	element
– Plane normal to END of	h1 h2 h3 = [	0,	0,	1]	element
– Plane normal to BEGINNING of	h1 h2 h3 = [	0,	0,	-1]	element

Hexagonal Lattice Element Indexing



Hex, Using Surfaces



Equilateral Hexagonal Lattice

MCNP Surface Equation

($Ax + By + Cz - D = 0$)

1: $x - p/2 = 0$

2: $x + p/2 = 0$

3: $x + \sqrt{3}y - p = 0$

4: $x + \sqrt{3}y + p = 0$

5: $-x + \sqrt{3}y - p = 0$

6: $-x + \sqrt{3}y + p = 0$

MCNP Surfaces

1 px p/2

2 px -p/2

3 p 1.0 1.73205 0.0 p

4 p 1.0 1.73205 0.0 -p

5 p -1.0 1.73205 0.0 p

6 p -1.0 1.73205 0.0 -p

Hexagonal Lattice Element Indexing

- **At least 6 surfaces are required**

- **i,j,k lattice element indexing determined surface order:**

  Beyond:

	i j k
– Beyond 1st surface is	[1, 0, 0] element
– Beyond 2nd surface is	[-1, 0, 0] element
– Beyond 3rd surface is	[0, 1, 0] element
– Beyond 4th surface is	[0, -1, 0] element
– Beyond 5th surface is	[-1, 1, 0] element
– Beyond 6th surface is	[1, -1, 0] element
– Beyond 7th surface is	[0, 0, 1] element
– Beyond 8th surface is	[0, 0, -1] element

- **Suggest i, j, k indices increase along +x, +y, +z**

Lattice Example

5 0 -8 11 -7 10 -12 9 lat = 2

```
7  p   1 1.732 0   23.1
8  px    11.55
9  p  -1 1.732 0  -23.1
10 p   1 1.732 0  -23.1
11 px -11.55
12 p  -1 1.732 0   23.1
```