NRC-CNRC

How to Code Geometry

Writing Subroutine HOWFAR

Walter R. Nelson
Stanford Linear Accelerator Center

Introduction

- An EGS User Code requires
  - SUBROUTINE AUSGAB for scoring results of interest
  - SUBROUTINE HOWFAR to provide information about the nature of the geometry
- The most trivial HOWFAR is a ***homogeneous, infinite*** medium

```
SUBROUTINE HOWFAR;
RETURN;
END;
```

- In this lecture we will show you how to write code that can be used for more complicated geometries

Useful Geometry References

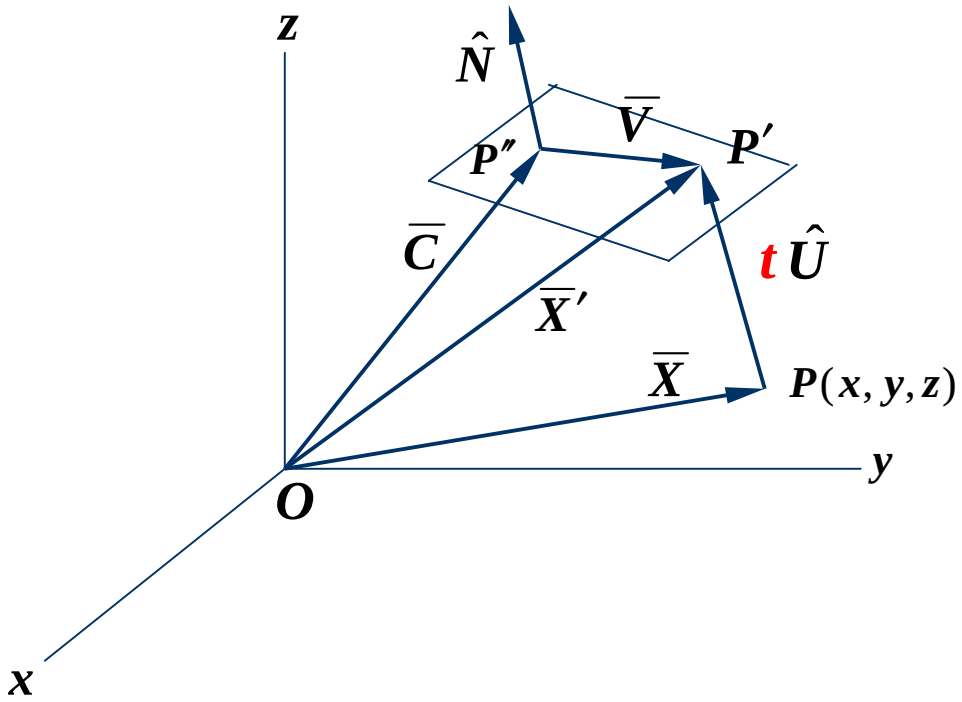
- W. R. Nelson and T. M. Jenkins, "Writing SUBROUTINE HOWFAR for EGS4", SLAC-TN-87-4 (31 August 1988/Rev.)
- W. R. Nelson and T. M. Jenkins, "Geometry Methods and Packages", Chapter 17 in ***MONTE CARLO TRANSPORT OF ELECTRONS AND PHOTONS*** (Plenum Press, 1988)

Note: In this lecture, the word "**EGS**" applies to both EGS4 and EGSnrc

Items Covered in this Lecture

- General mathematical considerations (vectors)
- The role of key variables in EGS
- Special subprograms available with EGS
- Mortran3 ***macro-equivalent*** forms of these subprograms
- Putting all the modules together to form the geometry
- Other geometry packages—e.g., Combinatorial Geometry

Mathematical Considerations



- Particle trajectories are described by the position and direction vectors

$$\overline{X} = x\hat{i} + y\hat{j} + z\hat{k} \quad \text{and} \quad \hat{U} = u\hat{i} + v\hat{j} + w\hat{k}$$

represented by **`X(NP),Y(NP),Z(NP)`** and **`U(NP),V(NP),W(NP)`**, respectively, and are passed in **`COMMON/STACK/`** along with the ***stack pointer***, **`NP`**

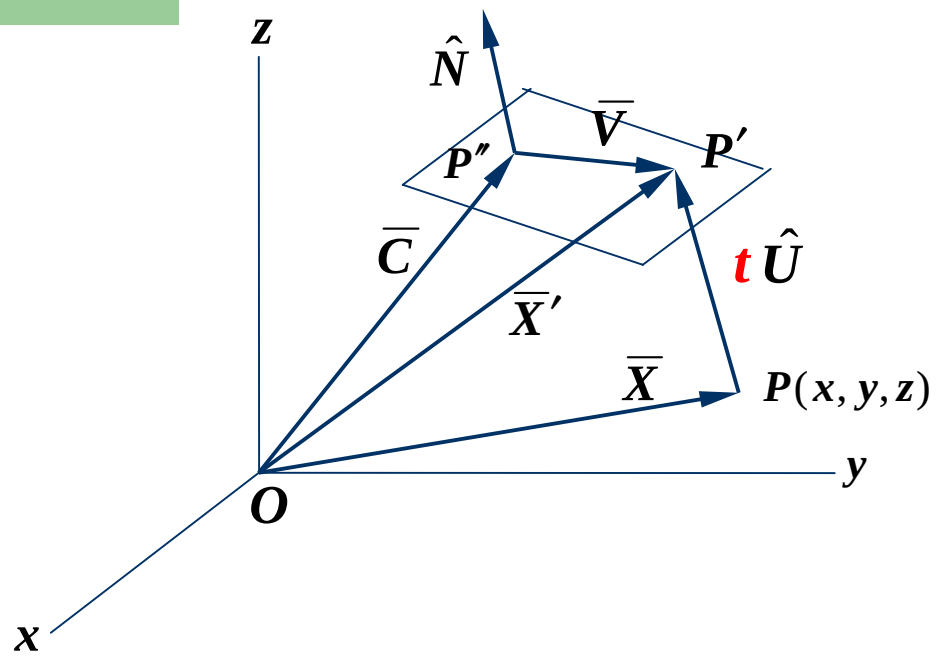
Subroutine HOWFAR

…Math Considerations (cont.)

- The quantities $\overline{\mathbf{X}}(\mathbf{x},\mathbf{y},\mathbf{z})$ **and** $\hat{\mathbf{U}}(\mathbf{u},\mathbf{v},\mathbf{w})$, together with such things as particle **type**, **energy**, **weight**, ***time***, etc., define the ***state function*** of the particle (and are called ***stack variables***)
- In writing SUBROUTINE HOWFAR, the problem becomes one of
  - determining the point of intersection, ***P'***, of the particle trajectory with any given surface,
  - which allows for the extraction the distance ***t*** **,**
  - and comparing ***t*** with **USTEP**—the transport step about to be taken for the current particle being followed

Referring to the figure again…

…Math Considerations (cont.)

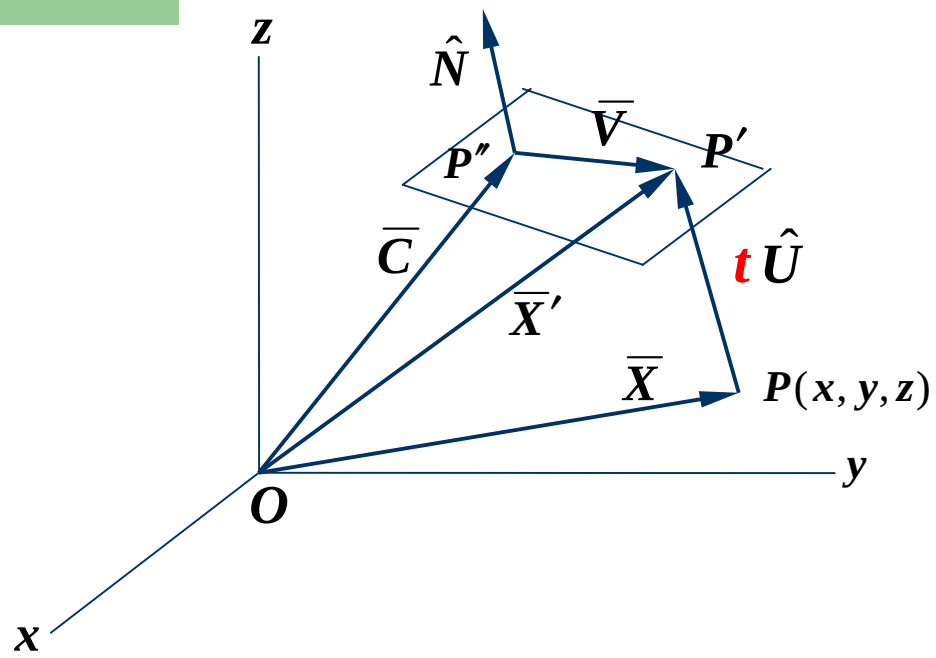


- A plane surface can be described by the vector to a point, ***P"***, on the surface and a unit vector normal to it,

$$\overline{C} = c_1\hat{i} + c_2\hat{j} + c_3\hat{k} \quad \text{and} \quad \hat{N} = n_1\hat{i} + n_2\hat{j} + n_3\hat{k}$$

  which are the arrays **`PCOORD(3:100)`** and **`PNORM(3:100)`**, respectively, that are passed in **`COMMON/PLADTA/`** .

…Math Considerations (cont.)



- The condition for the intersection point (***P'***) to ***lie on the plane*** is obtained from the vector dot product $\overline{V} \cdot \hat{N} = 0.$
- From the diagram we see that

$$\overline{V} = \overline{X}' - \overline{C} = \overline{X} + t\hat{U} - \overline{C} \ ,$$

which leads to the equation…

Subroutine HOWFAR

…Math Considerations (cont.)

$$t = \frac{(\overline{C} - \overline{X}) \cdot \hat{N}}{\hat{U} \cdot \hat{N}} = \frac{(c_1 - x)n_1 + (c_2 - y)n_2 + (c_3 - z)n_3}{un_1 + vn_2 + wn_3}$$

- The following physical situations apply
  - ***a)*** $t > 0$: particle travels ***away from*** the plane
  - ***b)*** $t < 0$: particle travels ***towards*** the plane

- Note: The above equation is indeterminant when the denominator is 0, corresponding to the physical situation in which the particle travels ***parallel to*** the plane

The PLANE1 Algorithm

```
SUBROUTINE PLANE1(IPLN,INPT,IHIT,TPLN);
        "-------------------------------------------------------------------"
        "  Input:  IPLN      Plane identification number                    "
        "          INPT = 1  Surface normal points AWAY from current region "
        "               =-1  Surface normal points TOWARDS current region   "
        "  Output: IHIT = 0  Particle travels AWAY from surface (a miss)     "
        "               = 1  Particle travels TOWARDS surface (a hit)        "
        "               = 2  Particle travels PARALLEL TO surface (a miss)   "
        "          TPLN      Distance to surface (when IHIT=1)              "
        "-------------------------------------------------------------------"
        COMMON/STACK/E(40),X(40),Y(40),Z(40),U(40),V(40),W(40),DNEAR(40),
                     WT(40),IQ(40),IR(40),NP;
        DOUBLE PRECISION E;
        COMMON/PLADTA/PCOORD(3,100),PNORM(3,100);

        UDOTN=PNORM(1,IPLN)*U(NP) + PNORM(2,IPLN)*V(NP) + PNORM(3,IPLN)*W(NP);
        UDOTNP=UDOTN*INPT;

        IF (UDOTNP.EQ.0.0) [ IHIT=2; "Parallel to z-axis (indeterminant)" ]
        ELSEIF (UDOTNP.LT.0.0) [ IHIT=0; "Traveling away from surface"]
        ELSE [ "Traveling towards surface---determine distance (TPLN)"
          IHIT=1;
          TPLN=PNORM(1,IPLN)*(PCOORD(1,IPLN)-X(NP))
                  + PNORM(2,IPLN)*(PCOORD(2,IPLN)-Y(NP))
                     + PNORM(3,IPLN)*(PCOORD(3,IPLN)-Z(NP));
          TPLN=TPLN/UDOTN;
          ]
        RETURN;
        END;
```

Except for parameter **INPT**, which allows for more efficient determination of ***t***, algorithm above is based precisely on the previous equation

Subroutine HOWFAR

Specifications for HOWFAR

- For every `CALL SHOWER` invocation in the MAIN program of the User Code, particles that are being transported are placed on a ***stack***
- The ***current*** particle being tracked is identified on the ***stack*** by the pointer, `NP`
- There are three EGS variables that play an important role in HOWFAR: `USTEP`, `IDISC`, and `IRNEW`
- These variables are passed in `COMMON/EPCONT/`

…Specifications for HOWFAR (cont.)

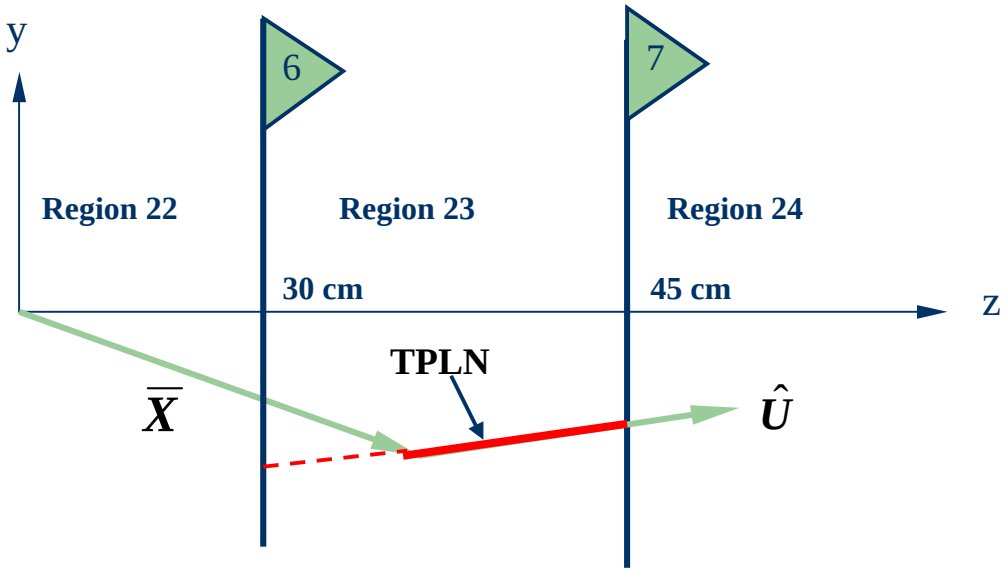
- On entry to HOWFAR, EGS has predetermined that it would like to transport the current particle by a straight-line distance, **USTEP**
- HOWFAR must then determine if **USTEP** will carry the particle past the boundary towards which it is heading
- If it does carry it past, HOWFAR must do two things:
  - Shrink **USTEP** to the distance to the boundary, t
  - Set **IRNEW** to the "new" region in which the particle is expected to end up
- Otherwise, a **RETURN** is simply made to the subroutine that called HOWFAR (i.e., ELECTR or PHOTON)

…Specifications for HOWFAR (cont).

- On occasion a particle will end up in a region designated by the user as a *discard region*
- In such cases, the flag **IDISC** is set equal to **unity** in HOWFAR and a **RETURN** is made to the calling subprogram
- The distance to the next boundary, t, can be determined by calling one of the following geometry routines:

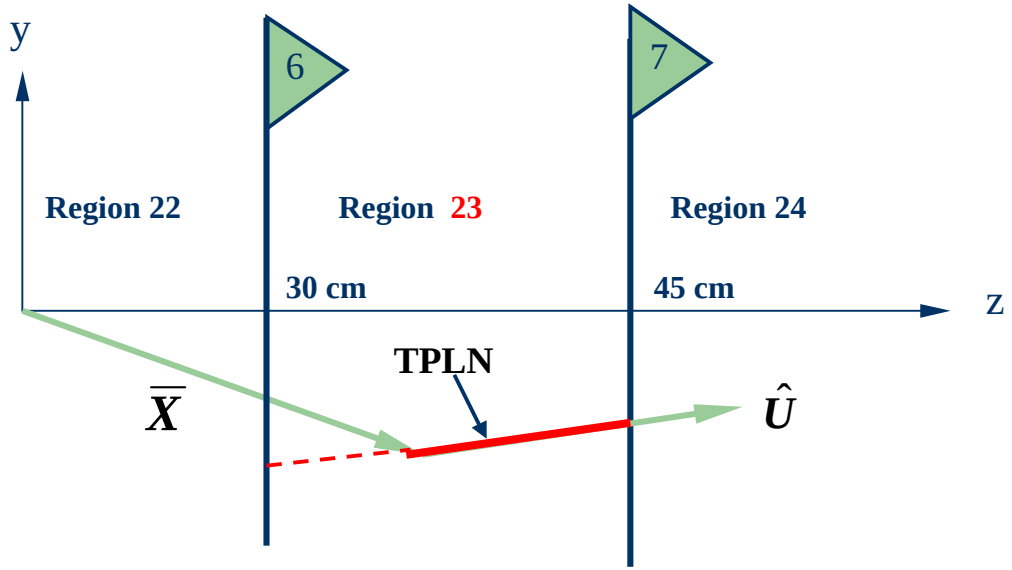
  PLANE1, CYLNDR, CONE, SPHERE, PLAN2P, PLAN2X, CYL2, CON2, SPH2
- Two other geometry subprograms are available to help in this task:
  - CHGTR for changing **USTEP** and **IRNEW** if needed
  - FINVAL for getting the coordinates at the end of a projected transport

Using PLANE1 and CHGTR



- Consider the two parallel planes separating three regions
- The regions are identified by the numbers 22, 23 and 24 and the planes by the numbers 6 and 7
- The ***triangles*** enclosing the numbers 6 and 7 have a purpose—they point in the direction of the unit normal vector and the user must define them

…PLANE1 and CHGTR (cont.)

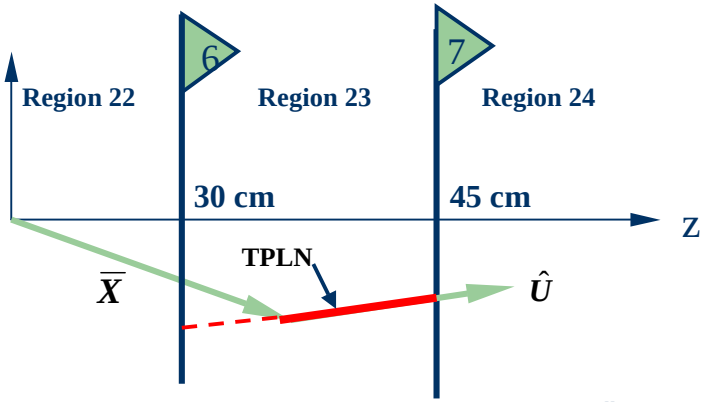


- Assume that planes 6 and 7 are located at z = 30 and 45 cm, respectively

```
PCOORD(1,6)=0.0;    PCOORD(2,6)=0.0;    PCOORD(3,6)=30.0;
PNORM(1,6)=0.0;     PNORM(2,6)=0.0;     PNORM(3,6)=1.0;
PCOORD(1,7)=0.0;    PCOORD(2,7)=0.0;    PCOORD(3,7)=45.0;
PNORM(1,7)=0.0;     PNORM(2,7)=0.0;     PNORM(3,7)=1.0;
```

- If particles are initially started in region 23 and discarded when they leave this region, the following **HOWFAR** will work nicely with EGS

…PLANE1 and CHGTR (cont.)




```
SUBROUTINE HOWFAR;
     "-------------------------------------------------------------------"
     " Two parallel planes (6,7) separating three regions (22,23,24).    "
     " Transport in region 23 and discard in regions 22 and 24.          "
     "-------------------------------------------------------------------"
     COMIN/EPCONT,STACK/;  "To supply IDISC and IR(NP)"

     IF (IR(NP).NE.23) [ IDISC=1; "Discard particles outside region 23" ]
     ELSE [ "Track particles within region 23"
       CALL PLANE1(7,1,IHIT,TPLN);  "Check upstream plane first"
       IF (IHIT.EQ.1) [ "Surface is hit---make changes if necessary"
         CALL CHGTR(TPLN,24);
         ]
       ELSEIF (IHIT.EQ.0) [ "Heading backwards"
         CALL PLANE1(6,-1,IHIT,TPLN);  "To get TPLN-value (IHIT=1, a must)"
         CALL CHGTR(TPLN,22);  "Make changes if necessary"
         ]
       ]
     RETURN;
     END;
```

In the above, **SUBROUTINE CHGTR** does the following:

- If **TPLN.LE.USTEP** → **USTEP=TPLN** and **IRNEW=24** (or **22**)
- Otherwise nothing is done

Subroutine HOWFAR

$CHGTR – The Macro Equivalent of CALL CHGTR

```
SUBROUTINE CHGTR(TVALP,IRNEWP);
  COMIN/EPCONT/;
  IF(TVALP.LE.USTEP) [ USTEP=TVALP; IRNEW=IRNEWP; ]
  RETURN;
  END;
                        -versus-

  REPLACE {$CHGTR(#,#);} WITH
     {;IF({P1}.LE.USTEP) [USTEP={P1}; IRNEW={P2};]}

  "NOTE:   EVERYWHERE $CHGTR IS USED ONE MUST
           INCLUDE COMIN/EPCONT/"
```

- The **$CHGTR** macro can be explained as follows:
  - The Mortran macro-processor recognizes and processes specific ***strings***
  - **$CHGTR**(***string1, string2***), in its entirety, is such a string–and it also contains the substrings: ***string1,string2***
  - The ***string1*** that is located at the position of the first # is assigned to parameter **{P1}**
  - The ***string2*** that is located at the position of the second # is assigned to parameter **{P2}**
  - The macro is then expanded out accordingly
- Example: **$CHGTR(TPLN,24);** produces

```
IF(TPLN.LE.USTEP) [USTEP=TPLN; IRNEW=24;]
```

which is equivalent to **CALL CHGTR(TPLN,24);**

$PLANE1 – The Macro Equivalent of CALL PLANE1

- In a very similar fashion we can create a macro called **$PLANE1** that will replace **CALL PLANE1**
- We won't go through the details here, but will simply demonstrate how both **$PLANE1** and **$CHGTR** are used in the previous example of **SUBROUTINE HOWFAR**

```
SUBROUTINE HOWFAR;
      "-------------------------------------------------------------------"
      " Two parallel planes (6,7) separating three regions (22,23,24).    "
      " Transport in region 23 and discard in regions 22 and 24.          "
      "-------------------------------------------------------------------"
      COMIN/EPCONT,PLADTA,STACK/;  "To supply many things now"

      IF (IR(NP).NE.23) [ IDISC=1; "Discard particles outside region 23" ]
      ELSE [ "Track particles within region 23"
        $PLANE1(7,1,IHIT,TPLN);  "Check upstream plane first"
        IF (IHIT.EQ.1) [ "Surface is hit---make changes if necessary"
          $CHGTR(TPLN,24);]
        ELSEIF (IHIT.EQ.0) [ "Heading backwards"
          $PLANE1(6,-1,IHIT,TPLN);  "To get TPLN-value (IHIT=1, a must)"
          $CHGTR(TPLN,22);  "Make changes if necessary"
          ]
        ]
      RETURN;
      END;
```

- Note that the **COMIN** statement now contains **PLADTA** in addition to **EPCONT** and **STACK**

Subroutine HOWFAR

The $PLAN2P Macro*

(* Mnemonic for *two parallel* planes)

- The **HOWFAR** example that we have been following can be simplified even further with the aid of **$PLAN2P**

```
SUBROUTINE HOWFAR;
       "-------------------------------------------------------------------"
       " Two parallel planes (6,7) separating three regions (22,23,24).    "
       " Transport in region 23 and discard in regions 22 and 24.          "
       "-------------------------------------------------------------------"
       COMIN/EPCONT,PLADTA,STACK/;  "To supply many things now"

       IF (IR(NP).NE.23) [ IDISC=1; "Discard particles outside region 23" ]
       ELSE [ "Track particles within region 23"
         $PLAN2P(7,24,1,6,22,-1);
         ]
       RETURN;
       END;
```

- First group of numbers **(7,24,1)** corresponds to checking the ***downstream*** plane and is equivalent to **$PLANE1(7,1,IHIT,TPLN)** followed by **$CHGTR(TPLN,24)**
- The second group **(6,22,-1)** corresponds to checking the ***upstream*** plane and is equivalent to **$PLANE1(6,-1,IHIT,TPLN)** followed by **$CHGTR(TPLN,22)**
- **$PLAN2P** is efficient in that the second plane is only checked if necessary; namely, if the particle is really heading towards it (i.e., it makes sense to **query the *downstream* plane first**)

Subroutine HOWFAR

Multislab Example

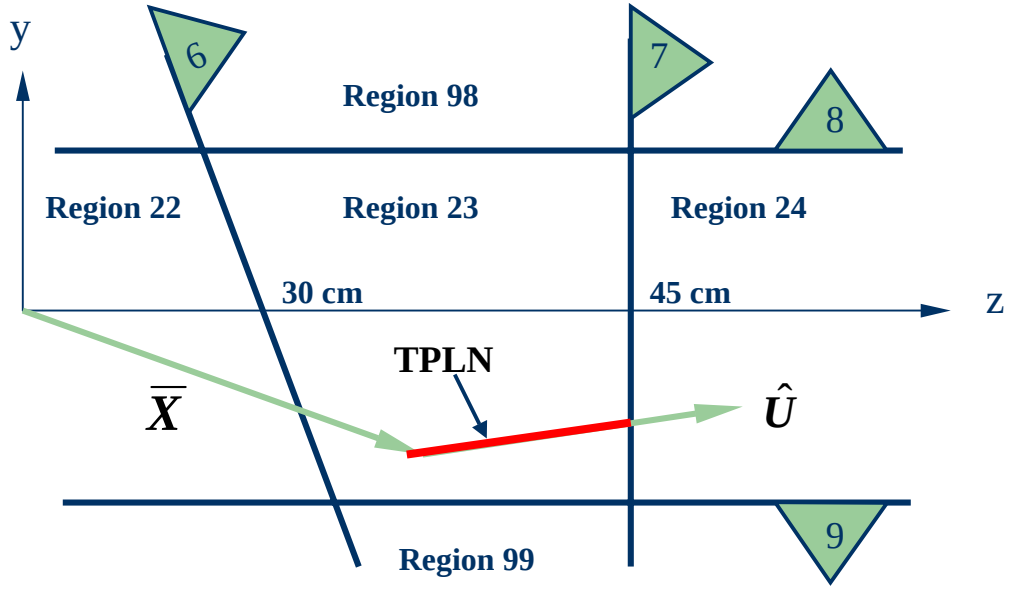
- It is very simple to extend the previous HOWFAR for many slabs

```
SUBROUTINE HOWFAR;
     "-------------------------------------------------------------"
     " Multislab (NREG-1) shower calorimeter.                      "
     "-------------------------------------------------------------"
     COMIN/EPCONT,PLADTA,STACK/;
     COMMON/PASSIT/NREG;                          "To make the number"
                                        "of regions (NREG) available"

     IRL=IR(NP);                              "Create a local variable"

     IF (IRL.EQ.1.OR.IRL.EQ.NREG) [ IDISC=1;   "Upstream/downstream"
                                                    "regions"
                                  ]
     ELSE [                "Track particles within calorimeter proper"
       $PLAN2P(IRL,IRL+1,1,IRL-1,IRL-1,-1);
       ]
     RETURN;
     END;
```

The $PLAN2X Macro*

(* Mnemonic for *two crossing* planes)

- Consider the geometry below consisting of five regions formed by a pair of *parallel* and a pair of *crossing* planes:



- We will impose the conditions that all particles start out in region 23, but are ***discarded*** when they leave it

…the following HOWFAR can be written

Subroutine HOWFAR

…$PLAN2X Macro (cont.)

```
SUBROUTINE HOWFAR;
        "-------------------------------------------------------"
        " Two crossing planes (6,7) separating three regions "
          "(22,23,24), with "
        " second set of parallel planes (8,9) separating three regions"
          "(98,99)"
        "-------------------------------------------------------"
        COMIN/EPCONT,PLADTA,STACK/;

        IF (IR(NP).NE.23) [ IDISC=1; "Discard particles outside "
                                     " region 23"
          ]
        ELSE [                      "Track particles within region 23"
          $PLAN2X(7,24,1,6,22,-1);
          $PLAN2P(8,98,1,9,99,1);
          ]
        RETURN;
        END;
```

- It is instructive for to convince oneself that the following statements are true:
  - **$PLAN2X** and **$PLAN2P** must **both** be called
  - The **order** in which **$PLAN2X** and **$PLAN2P** are called is **not important**
  - If one has preknowledge that the radiation field moves ***essentially*** towards one or the other parallel plane, then **$PLAN2P** can be made more efficient by **checking the preferred plane first**
  - There is **no preferred order** for checking planes with **$PLAN2X**
  - **USTEP** and **IRNEW** will **always be properly selected**

Subroutine HOWFAR

The $CYLNDR, $CONE and $SPHERE Macros*

- The conic surface algorithms are basically all the same and **$CONE** and **$SPHERE** may be used in **SUBROUTINE HOWFAR** in the same manner as **$CYLNDR**; therefore, only **$CYLNDR** will be described here
- We will skip the math here, but the intersection of a vector with a conic surface leads to a quadratic equation, the solutions of which are both real and imaginary and correspond to actual ***physical*** solutions
- The following figure shows possible trajectories intersecting a cylinder

Key:

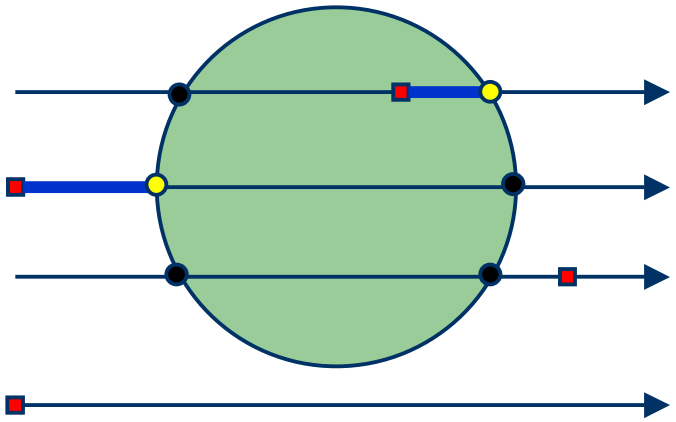
Start

Useful intersection

Useful distance

Non-useful intersection

* Currently defined along the z-axis only



Subroutine HOWFAR

The $CYLNDR Macro

- **The algorithm in** `$CYLNDR` was designed to take all these possibilities into account
- To accomplish the task, the user must determine whether the current particle location is ***inside*** or ***outside*** of the cylinder
- For `$CYLNDR(ICYL,INCY,IHIT,TCYL)` the parameters are explained as follows

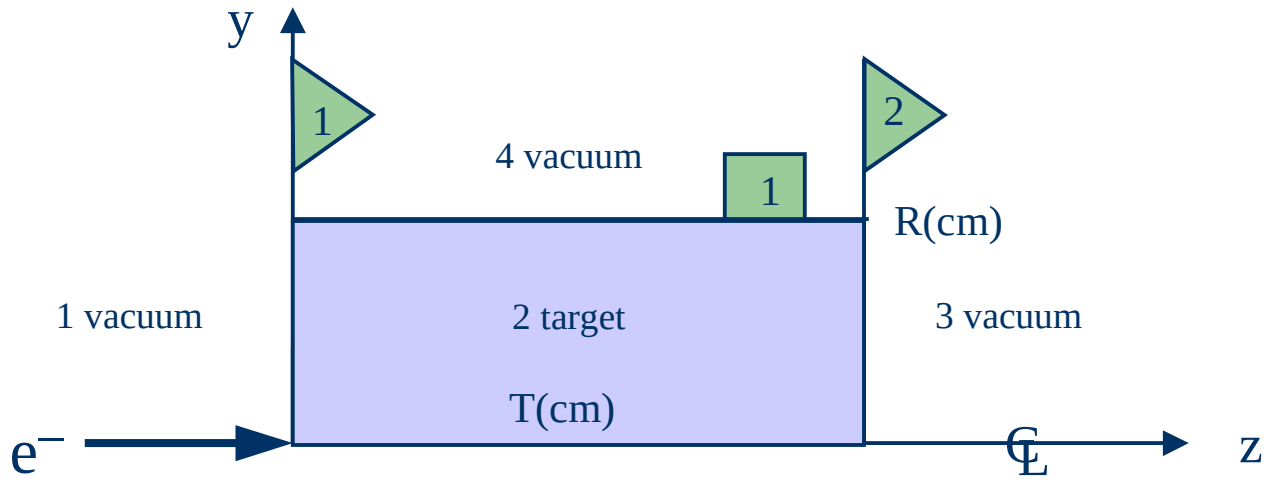
```
Input:  ICYL       Cylinder identification number
        INCY = 1   Current particle position is INSIDE cylinder
             = 0   Current particle position is OUTSIDE cylinder
Output: IHIT = 1   Particle trajectory HITS surface
             = 0   Particle trajectory MISSES surface
        TCYL       Distance (shortest) to surface (when IHIT=1)
```

- `TCYL` is the ***useful distance*** shown by the blue line segment in the previous slide

Subroutine HOWFAR

Example: A Cylinder-Slab

- Consider a cylindrical target struck by an electron beam



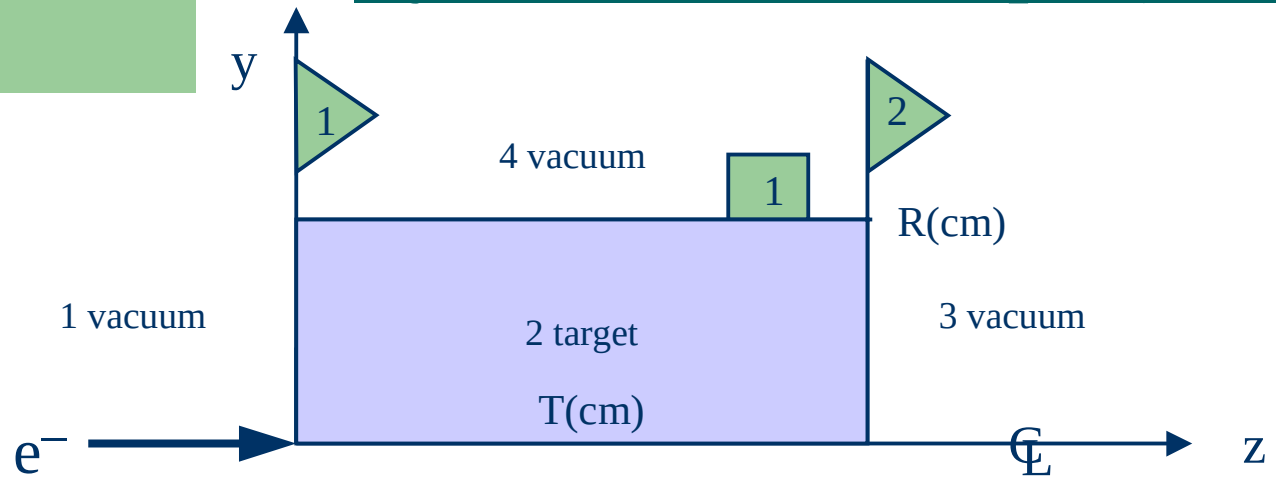
  - The cylinder of rotation about the z-axis is identified by box 1 and radius R
  - Planes 1 and 2 define the length of the target of thickness T
  - There are four regions of interest**:** the target (region 2) and three vacuum regions– upstream (region 1), downstream (region 3) and surrounding the target (region 4)

- The radius (squared) of the cylinder is defined in **MAIN** and passed to **HOWFAR** via **COMIN/CYLDTA/**, and the following macro (in **egsnrc.macro**) is required

```
REPLACE {;COMIN/CYLDTA/;} WITH
        {;COMMON/CYLDTA/CYRAD2($MXCYLS);}
```

Subroutine HOWFAR

…Cylinder-Slab Example (cont.)



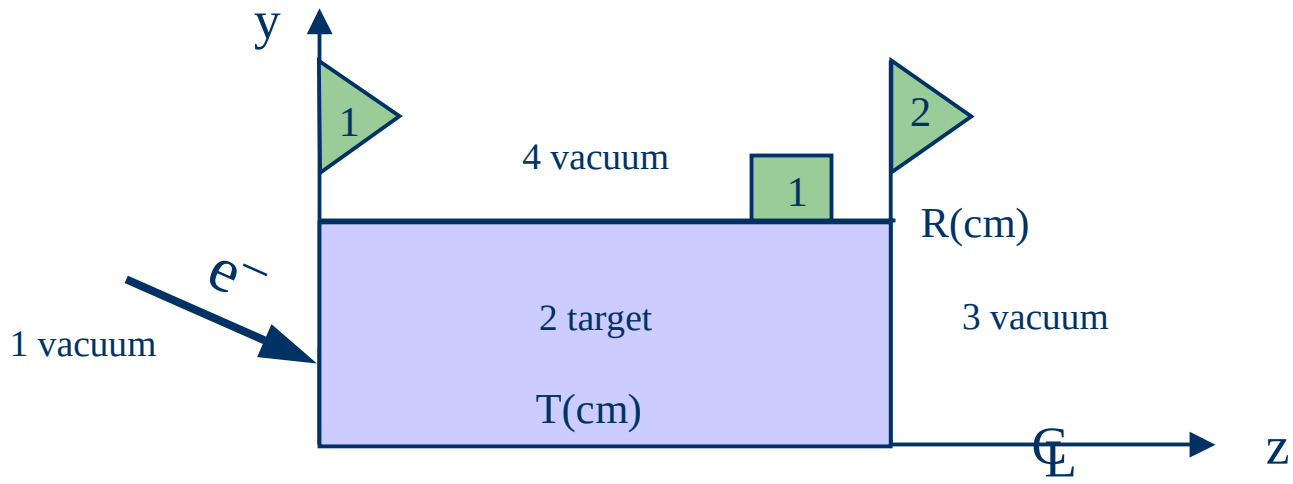
- The following will work for the above geometry:

```
SUBROUTINE HOWFAR;
    "-----------------------------------------------------------------"
    " Cylinder of rotation about the z-axis bounded by two planes.      "
    "-----------------------------------------------------------------"
    COMIN/CYLDTA,EPCONT,PLADTA,STACK/;

    IF (IR(NP).NE.2) [ IDISC=1; "Discard particles outside the target" ]
    ELSE [ "Track particles within the target"
      $CYLNDR(1,1,IHIT,TCYL);  "Check the cylinder surface"
      IF (IHIT.EQ.1) [ $CHGTR(TCYL,4); "Change if necessary" ]
      $PLAN2P(2,3,1,1,1,-1);  " Check the downstream plane first and"
                              " then the upstream one if necessary  "
      ]
    RETURN;
    END;
```

Subroutine HOWFAR

The $FINVAL Macro

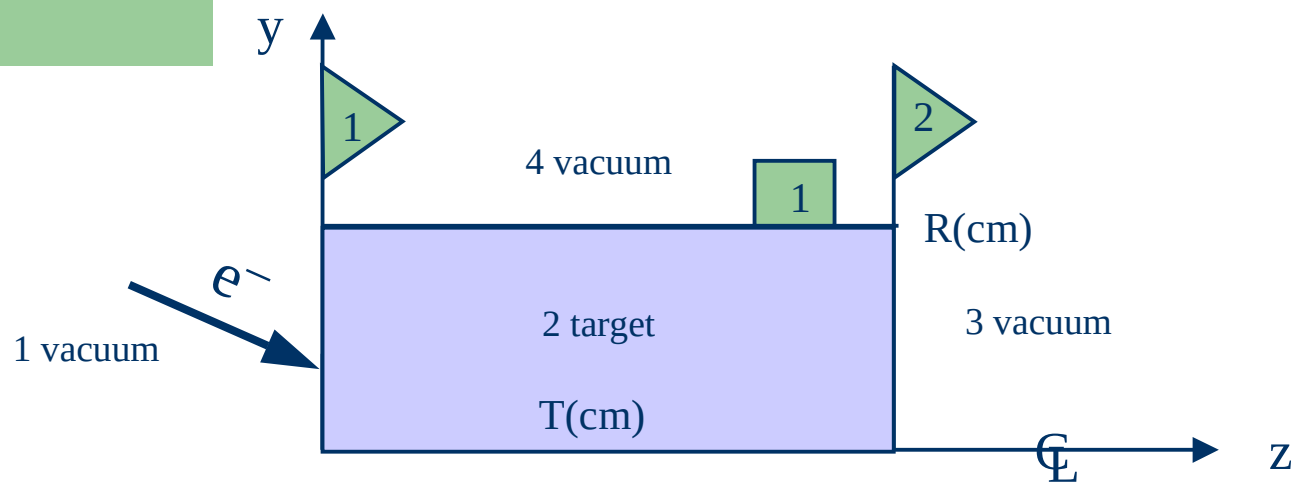
- The **$FINVAL** macro is useful for determining the final coordinates of a particle–e.g., the coordinates at the ***point of intersection*** of a trajectory and a geometric surface
- To illustrate this, consider the cylinder-slab geometry of the previous example:



Assume that we have a beam to the left in region 1 and are allowing the particles to be ***transported to the target***

Subroutine HOWFAR

$FINVAL Macro (cont.)



- The following additional code could be used with our **HOWFAR** to take care of this situation

```
IF (IR(NP).EQ.1) [
      $PLANE1(1,1,IHIT,TPLN);
      IF (IHIT.EQ.1) [
        $FINVAL(TPLN,XF,YF,ZF);  "Get final coordinates"
        IF ((XF*XF + YF*YF).LT.CYRAD2(1)) [ IRNXT=2; ]
        ELSE [ IRNXT=4; ]
        $CHGTR(TPLN,IRNXT);
        ]
```

Note that a square root was purposely avoided in the **IF** statement above for efficiency reasons

Subroutine HOWFAR

Closing Remarks

- A set of geometry macros is available for defining **HOWFAR**
- This lecture demonstrated how one uses these macros to create a relatively simple geometry
- But these macros (or their subroutine equivalents) can be used in a modular way to define very complex geometries
- The references at the beginning of this lecture provide more complex examples