Monte Carlo simulation of EPMA measurements on complex specimens using PENELOPE

Francesc Salvat
Xavier Llovet

Facultat de Física (ECM)
Universitat de Barcelona

Outline:

- Introduction
- The simulation code *PENELOPE*
- *PENEPMA*: Simulation of EPMA spectra
- Sample simulation results

Introduction

EPMA: Typical problem of coupled electron-photon transport

Focused monoenergetic e^- beam
➔ x-ray generation and transport within the sample
➔ measured x-ray energy spectrum

Quantitative analysis (x-ray intensities ➔ element concentrations) usually based on approximate semi-empirical algorithms

Difficulties for samples with complex geometries (multilayers, particles, rough surfaces, ...) and for unconventional conditions (low voltages, grazing-angle incidence, ...)

Alternative: Monte Carlo simulation methods (MC)

Components of a Monte Carlo algorithm

- **Interaction model**
  (set of cross sections for the different interaction mechanisms and atomic relaxation data)
  Determines the reliability of the results
  Analytical approximations may not be sufficiently accurate (screened Rutherford DCS, Kirkpatrick-Wiedman-Statham DCS, ...)
- **Transport algorithm**
  Must be as faithful as possible (preferably detailed, interaction by interaction, simulation)
  The continuous slowing-down approximation, CSDA, and other multiple scattering approximations, ... should be avoided
- **Geometry tools**
  Ability to handle systems (samples) with complex geometrical structures
- Some existing codes use **simplifications** in the description of photon transport (e.g., exponential attenuation) which limit the accuracy of calculated fluorescence corrections

The simulation code PENELOPE

*PEN*etration and *E*nergy *LO*ss of *P*ositrons and *E*lectrons (... and photons)

A general-purpose Monte Carlo code system for the simulation of coupled electron-photon transport in arbitrary geometries

Distributed by the OECD-NEA Data Bank (Paris)
(~600 registered users, thoroughly checked... in specific energy ranges)

A very detailed description of the code system (including physical interaction models, sampling and transport algorithms) is given in a NEA report. The PDF file can be downloaded from the web site

http://www.nea.fr/lists/penelope.html

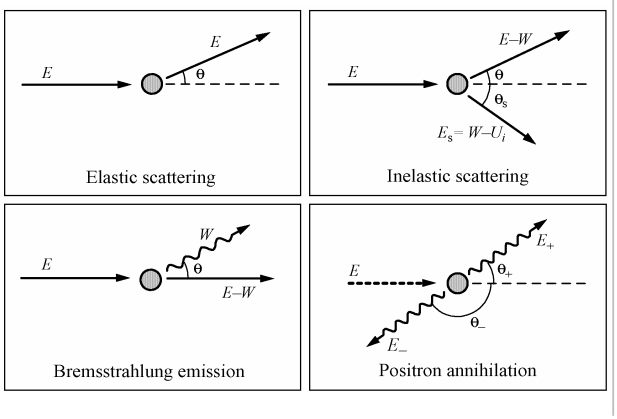
The calculations presented here have been performed using the version 2006 of the code, which was released a few weeks ago

Main features

- All kinds of interactions (except nuclear reactions) in the energy range
  from 10^9 eV down to, nominally, 50 eV
  (covered by the database)
- Implements the most accurate physical models available (limited only by the required generality)
- Simulates electrons and positrons (tunable mixed scheme) and photons (detailed, interaction by interaction)
- Simulates fluorescent radiation from K, L and M-shells
- Includes a flexible geometry package (constructive quadric geometry)
- Electron and positron transport in external magnetic and electric fields (in matter)

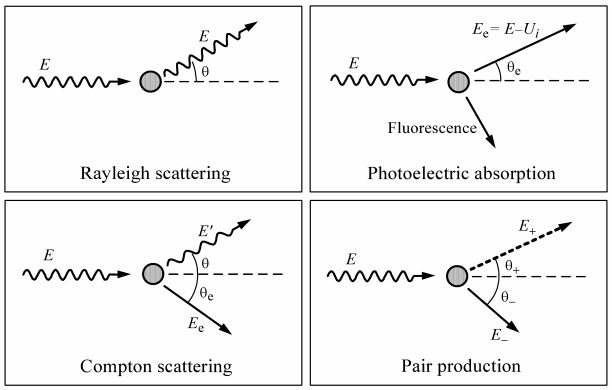
Electron/positron interactions

(Energy: 50 eV to 1 GeV)



Photon interactions

(Energy: 50 eV to 1 GeV)



Atomic relaxation

- **Ionization of inner shells (K, L1-L3, M1-M5)**

The atoms in the sample can be ionized by electron impact, by photoelectric absorption and by Compton scattering of photons

PENELOPE follows all secondary radiations emitted in the relaxation of ions with vacancies in the K shell and in L and M subshells

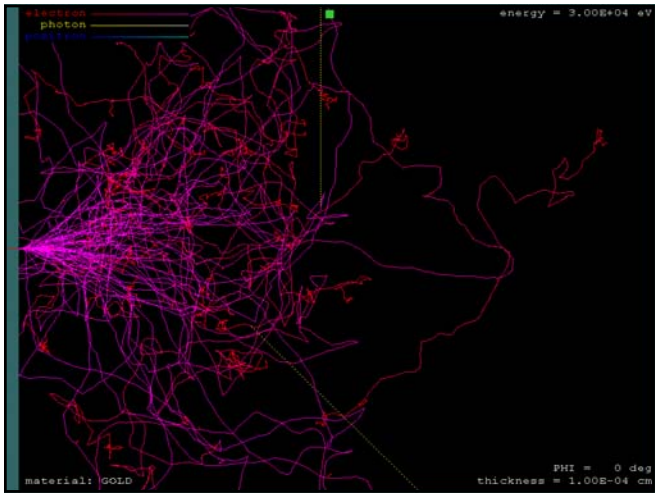
Transition probabilities from LLNL's Evaluated Atomic Data Library (Perkins et al., 1991), which are calculated using **approximate** theoretical models

X-ray energies from compilations by Deslattes at al (2003) (K and L shells) and Bearden (1967) (M shells)
⇒ Simulated characteristic lines have the "correct" energies

Variance reduction

- **Direct MC simulation is very inefficient...**

The next slide shows 50 simulated trajectories of 30-keV electrons in gold. Only one photon is generated



Variance reduction

A convenient figure of merit to measure the performance of a MC algorithm for computing a quantity Q is the efficiency

$$\epsilon_Q \equiv \left(\frac{\sigma_Q}{Q}\right)^{-2} \frac{1}{T}$$

where σ_Q/Q is the relative (statistical) uncertainty and T is the calculation time (on a given computer).

The calculation time to obtain a given statistical accuracy is proportional to $1/\epsilon_Q$. The efficiency can be increased (and T decreased) by using **variance-reduction techniques**

PENELOPE implements three elementary techniques that are safe (i.e. they keep the simulation results unbiased):
- -- **particle splitting**,
- -- **Russian roulette**, and
- -- **interaction forcing** (**or** **method of weights**)

Interaction forcing alone renders EPMA (detailed) simulations feasible

The geometry package PENGEOM

Constructive quadric geometry:
allows describing arbitrary material systems consisting of homogeneous bodies limited by quadric surfaces

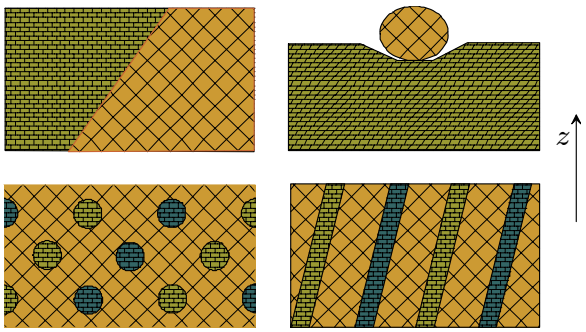
$$\Phi(\mathbf{r}) = A_{xx}x^2 + A_{yy}y^2 + A_{zz}z^2 + A_{xy}xy + A_{yz}yz + A_{zx}zx + A_x x + A_y y + A_z z + A_0 = 0$$

- Versatile enough to describe many real objects,
- Fast analytical calculation of intersections (quadratic equation)

Geometry operations are performed by the subroutine package *PENGEOM*, which automatically tracks particles through the different bodies and materials

NB: *PENGEOM* can be used with any detailed MC code

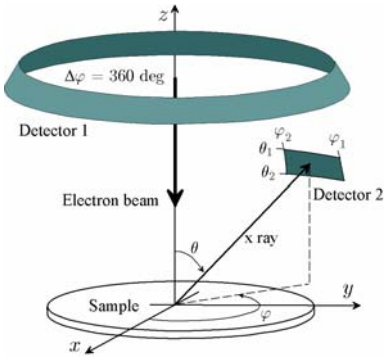
Examples of sample geometries that can be routinely simulated with *PENGEOM*



PENEPMA: simulation of EPMA spectra

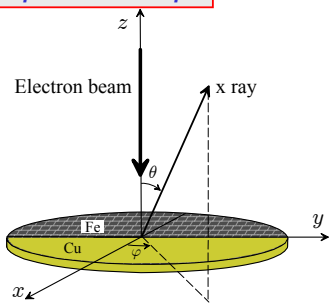
- No programming work is required. The program provides a complete description of electron transport and x-ray generation (including fluorescence)
- You can define the energy, direction and position of the electron beam
- The geometry of the sample (and its environment) is defined by using *PENGEOM*
- PENEPMA implements interaction forcing
- Allows the simulation of up to 25 photon detectors simultaneously
- Provides the x-ray spectrum for each detector, as well as the spectrum of characteristic x-rays (with the fluorescence contribution given separately)

Geometry and detectors

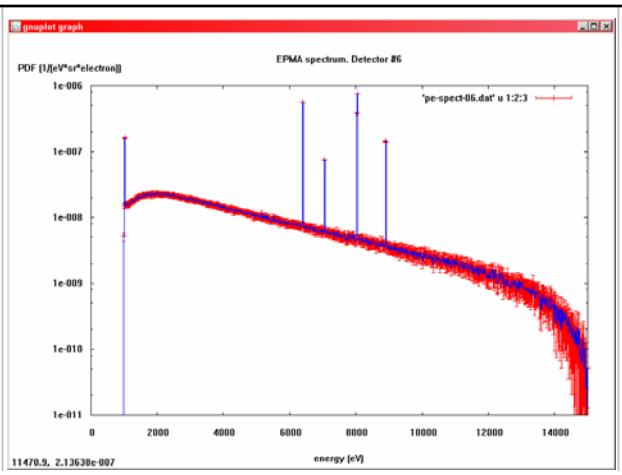


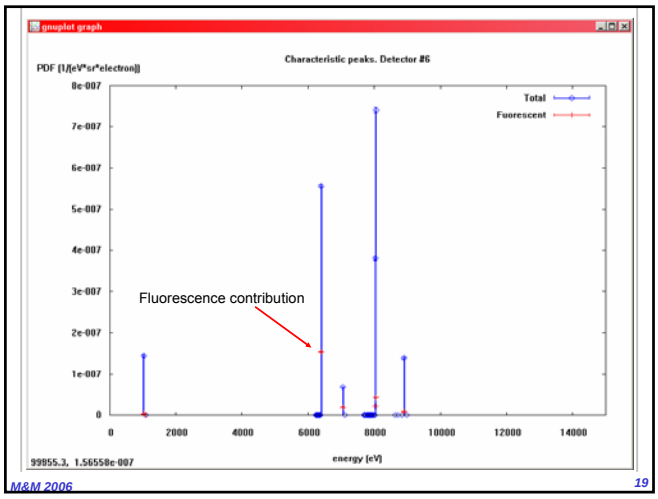
- Detectors are assumed to be ideal. Convolution with the detector's response function is required to compare with measured spectra

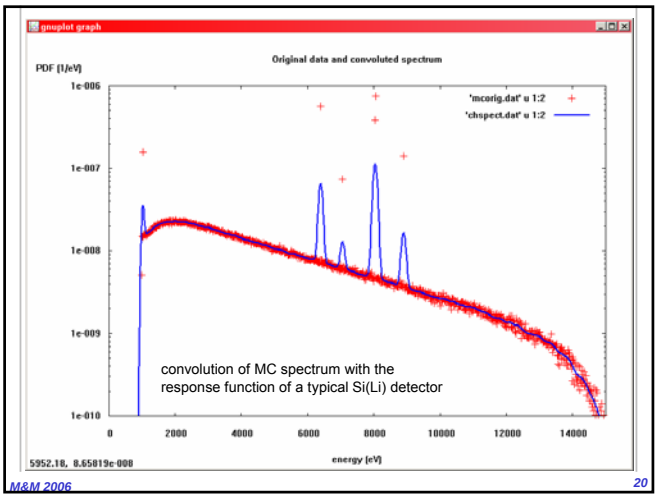
Practical example: Fe-Cu couple



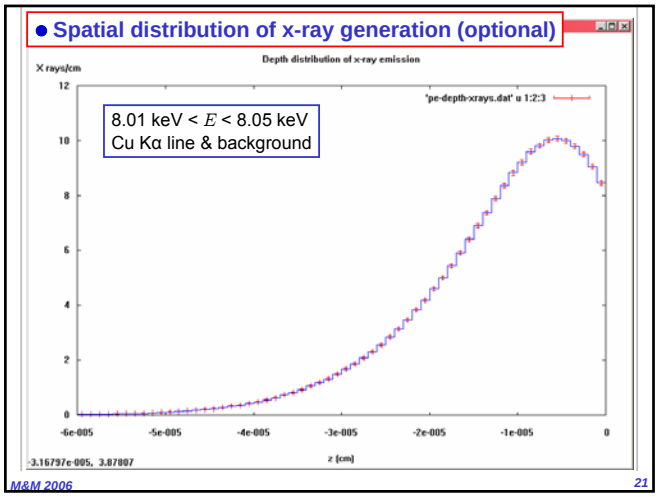
Electron beam characteristics: E = 15 keV, normal incidence
Impact point: on the Cu part, 0.1μm from the interface
Detector # 6: annular, θ = 45-55 deg
Simulation time: ~15 min (characteristic peaks, uncertainty < 5%)
~8 hours (background)

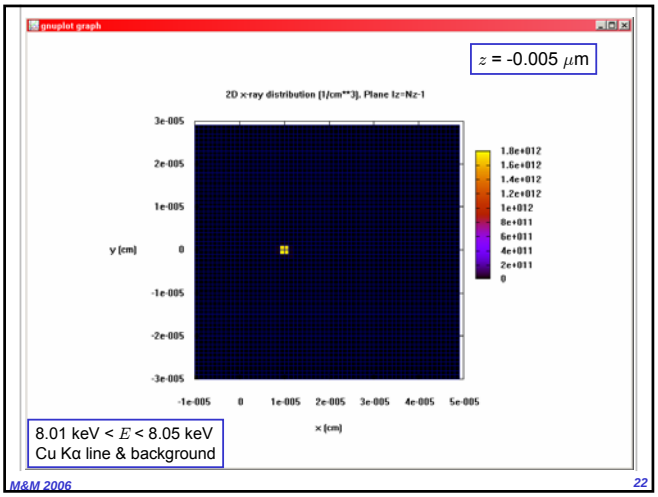


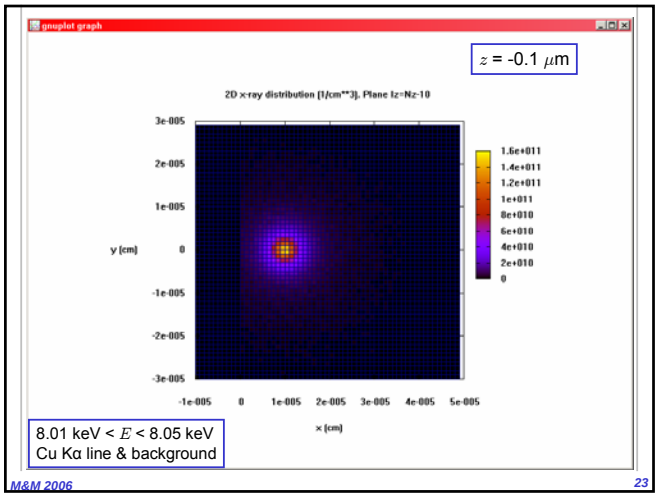


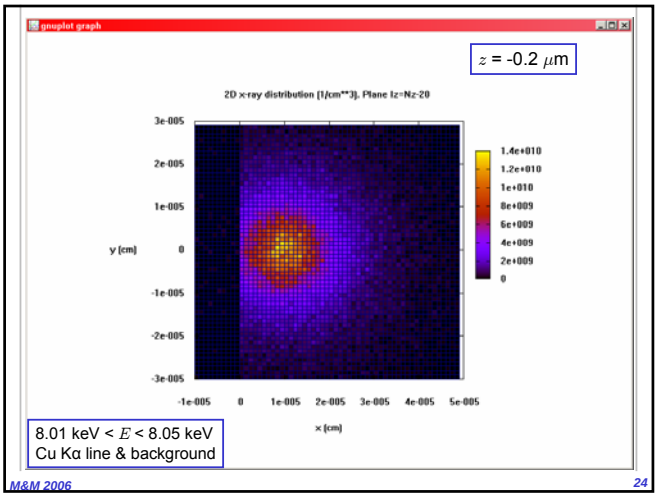


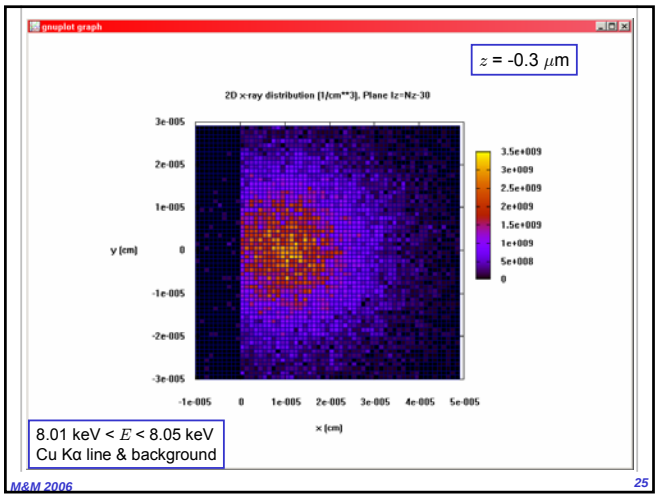
● Spatial distribution of x-ray generation (optional)

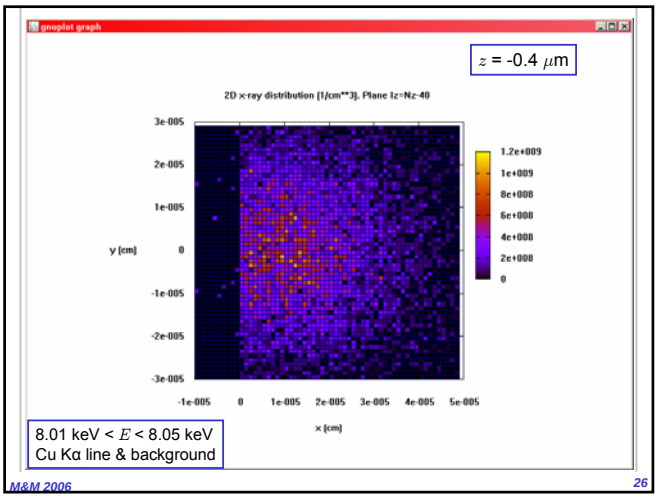


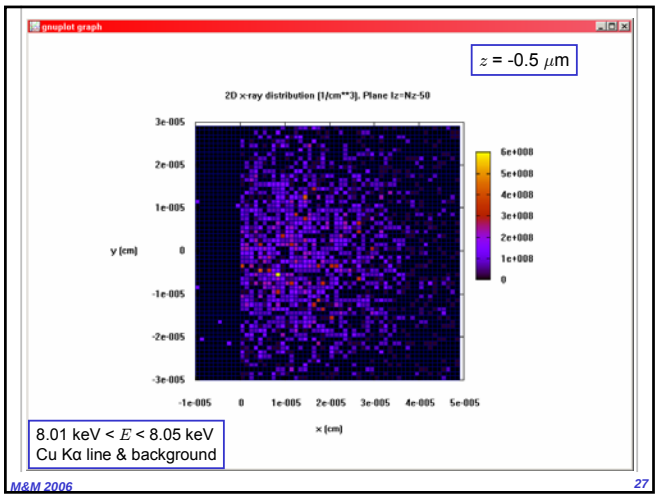












● Main limitations

Our knowledge of interaction and relaxation properties is limited

- Need of ionization cross sections more accurate than the PWBA
  Distorted wave methods seem to work well, calculations in progress
- Electrons with energies below ~500 eV in condensed matter
  (aggregation and coherence effects are disregarded)
- Photon interaction cross sections
  (EXAFS and anomalous scattering are not considered)
- Fluorescence and Coster-Kronig yields of L and outer shells
  (affected by substantial uncertainties)

Nevertheless, MC simulation is a valuable tool for understanding specific spectral features and the physical processes in our sample,

... and to gain information on fundamental parameters from EPMA spectra

CHAPTER 6: Structure and operation of PENELOPE

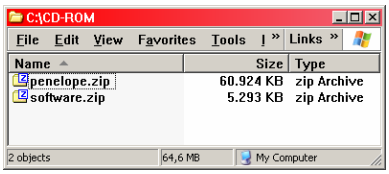
Outline:

- Contents of the CD-ROM
- Installation of PENELOPE
- Structure of the main program

Contents of the CD ROM

Two Zip files:

- **`penelope.zip`** : The PENELOPE (version 2006) code system
- **`software.zip`** : GNU Fortran 77 compiler and GNUPLOT
  (both for MS Windows; they are also available for LINUX)



NOTE: on Linux and other Unix OSs, the option "-a" may be required to unzip properly

(Just in case, we have also a CD with the whole package uncompressed)

Installation of the GNU FORTRAN 77 compiler

freeware: http://www.geocities.com/Athens/Olympus/5564

1) Copy the directory **`\CD-ROM\software.zip\g77d`** to **`c:\g77d`**

2) Copy the file **`\CD-ROM\software.zip\gnusetup.bat`** to c:**`\gnusetup.bat`** (or to any other directory in your **`PATH`**)


```
PATH=c:\g77d\bin;c:\gnuplotd;%PATH%
SET LIBRARY_PATH=c:\g77d\lib
```


3) Test your installation of the g77 compiler:

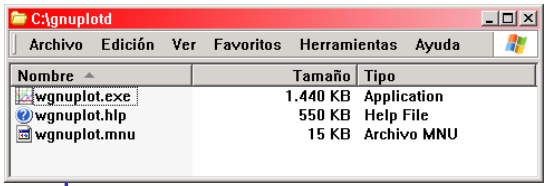
- Open a command window (aka MS-DOS window)
- Execute **`"gnusetup.bat"`**
- Go to directory **`c:\g77d`** and execute

```
g77 -Wall -O  mytest1.f -o mytest1.exe
```

Installation of GNUPLOT

freeware: http://www.gnuplot.info

- Copy the directory **`CD-ROM\software.zip\gnuplotd`** to **`c:\gnuplotd`**

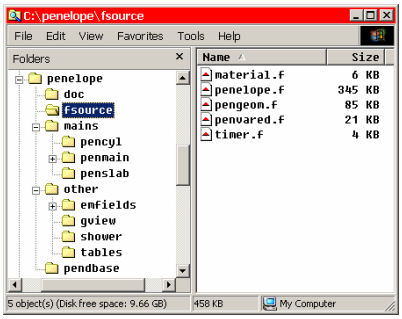


`gnusetup` includes **`gnuplotd`** in your "path"

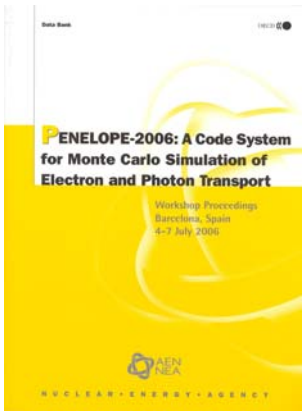
```
PATH=c:\gnuplotd;%PATH%
```

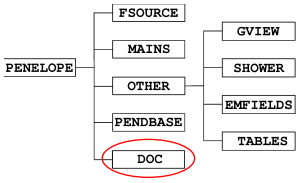

Installation OF PENELOPE

- Copy **`CD-ROM\penelope.zip`** to your hard disk and unzip it.
  This creates a new directory **`\penelope`** on your hard disk
  NOTE: on Linux and other Unix OSs, the option "-a" may be required to unzip properly



Directory `doc`. Documentation

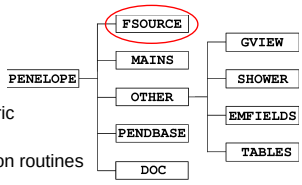




`tutorial.pdf`
`penelope-2006-NEA.pdf`

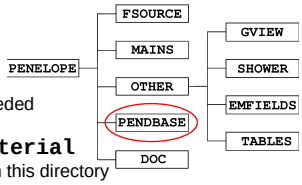
This is the document to have at hand in the initial stages of the use of PENELOPE

Directory `fsource`



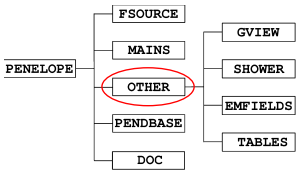
- **`penelope.f`** : simulation package
- **`pengeom.f`** : constructive quadric geometry package
- **`penvared.f`** : variance reduction routines
- **`material.f`** : main program to generate material data files
- **`timer.f`** : portable timing routines

Directory `pendbase`



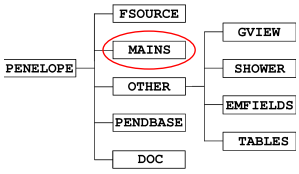
- 797 ASCII files containing material data (composition, interaction cross sections, etc.) needed to build the material definition files
- The binary executables files of **`material`** and **`shower`** have to be placed in this directory

Directory `other`



- **`\gview`**: geometry viewers **GVIEW2D** and **GVIEW3D** (MS Windows only), with examples of geometry-definition files.
- **`\shower`**: The program **`shower`** generates electron-gamma showers within a slab and displays the resulting tracks on the computer screen (MS Windows only). An educational tool.
- **`\emfields`**: subroutine package **`penfield`** for transporting electrons/positrons under external static electric and magnetic fields.
- **`\tables`**: auxiliary main program **`tables`** to generate numerical tables of radiation transport properties (cross sections, mean free paths, stopping powers, ...) for arbitrary materials. It allows the user to access the physics models underlying the simulations.

Directory `mains`



- Examples of main programs:
  `penslab.f` a single slab
  `pencyl.f` cylindrical geometries
  `penmain.f` generic quadric geometries
- **`*.in`**: input data files for the respective programs
- **`*.mat`**: material definition files used by these programs
- **`*.gnu`**: GNUPLOT scripts for visualizing simulated distributions

Final installation steps

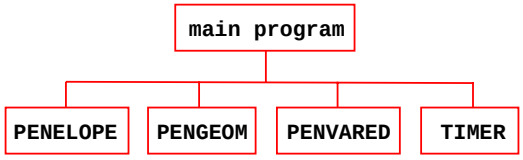
- Compile the program **`material`**,
  `g77 -Wall –O material.f -o material.exe`
  Move **`material.exe`** to directory **`\pendbase`**
- Compile **`tables`** :
  `g77 -Wall –O tables.f -o tables.exe`
  This program delivers basic transport data, such as mean free paths, photon mass attenuation coefficients, etc, without needing to run a full simulation
  Copy the file **`tables.exe`** into directory **`\pendbase`**
- Copy the file **`shower.exe`** into directory **`\pendbase`**

NOTE: the modules used by each main program are defined through "**`include`**" statements (they do not have to be declared to the compiler)

Structure of the main program

The user must provide a **steering main program** that generates the initial states of primary particles, controls the evolution of the simulated tracks and keeps score of relevant quantities.

NOTE: the generic main program **`penmain`** can handle a wide variety of problems, but yours may require some special treatment **...**



The communication between the main program and the simulation routines is through a few **common blocks**

All real variables are `double precision` and integers are all `integer*4`

Most of the information flux between the main program and the simulation and geometry subroutines is through the common block

COMMON/TRACK/E,X,Y,Z,U,V,W,WGHT,KPAR,IBODY,MAT,ILB(5)

which contains the following quantities:

KPAR ... kind of particle (1=electron, 2=photon, 3=positron)
E ... current particle's energy (eV)
X,Y,Z ... position coordinates (cm)
U,V,W ... direction cosines
WGHT ... particle's weight (used only with variance reduction)
IBODY ... index of body where the particle is moving
MAT ... corresponding material
ILB(5) ... describes origin of secondary particles (pp. 215-216)

In the I/O of the PENELOPE and PENGEOM routines, all energies are in eV and all lengths are in cm

To start the simulation of a particle, its initial state variables must be set by the main program. PENELOPE modifies the energy and direction cosines only when the particle undergoes an interaction. The position coordinates are updated by PENGEOM.

The simulation parameters, which are needed to initialize PENELOPE, are provided through the common block

COMMON/CSIMPA/EABS(3,MAXMAT),C1(MAXMAT),C2(MAXMAT), WCC(MAXMAT),WCR(MAXMAT)

For each material **MAT** the required parameters are

EABS(3,MAT) ... Absorption energies
C1(MAT) ... Average angular deflection
C2(MAT) ... Maximum average energy loss
WCC(MAT) ... Cutoff energy loss of hard inelastic collisions
WCR(MAT) ... Cutoff energy loss of hard bremsstrahlung emission

NOTE: The number label of each material is determined by its position in the input material-data file.

To allow repeatability of the simulation, the seeds of the random-number generator (L'Ecuyer) are specified by the main program.

COMMON/RSEED/ISEED1,ISEED2

● PENELOPE subroutines

SUBROUTINE PEINIT(EPMAX,NMAT,IRD,IWR,INFO)

Reads material data file and prepares tables to be used in simulation

◆ Input arguments

EPMAX ... maximum energy of particles
Warning: for positrons, add ~$2m_ec^2$ = 1,022 MeV
NMAT ... number of different materials
IRD,IWR ... input/output units (open in the main program)
INFO ... amount of information written to output (1, minimal; 4, very detailed)

NOTE: Simulation parameters must have been defined and loaded in common **CSIMPA**

SUBROUTINE CLEANS

Initializes the secondary stack (sets **NSEC=0**)
Must be called before starting the simulation of a new primary particle

SUBROUTINE START

Forces next interaction to be a soft one; it also verifies that **E>EABS.**
Must be called before starting a new particle (primary or secondary) and after each interface crossing.

SUBROUTINE JUMP(DSMAX,DS)

Determines the distance **DS** to the next interaction

DSMAX ... input, maximum allowed path length for electrons and positrons. Recommended: < body "thickness"/10
DS ... output, actual distance to travel

SUBROUTINE KNOCK(DE,ICOL)

Simulates the next interaction event and modifies the state variables in common **TRACK** (energy and direction)

DE ... (output) deposited energy in the material
ICOL ... (output) type of interaction

ICOL	electrons (KPAR=1)	photons (KPAR=2)	positrons (KPAR=3)
1	artificial soft event (random hinge)	coherent (Rayleigh) scattering	artificial soft event (random hinge)
2	hard elastic collision	incoherent (Compton) scattering	hard elastic collision
3	hard inelastic collision	photoelectric absorption	hard inelastic collision
4	hard bremsstrahlung emission	electron-positron pair production	hard bremsstrahlung emission
5	inner-shell impact ionisation		inner-shell impact ionisation
6			annihilation
7	delta interaction	delta interaction	delta interaction
8	auxiliary interaction	auxiliary interaction	auxiliary interaction

SUBROUTINE SECPAR(LEFT)

This subroutine delivers the initial state of a secondary particle produced during the previous simulation of the shower. This particle is removed from the secondary stack

LEFT ... output, number of particles in secondary stack at calling time. When **LEFT**=0, the simulation of the shower is completed.

The particle state variables are directly loaded in

COMMON/TRACK/E,X,Y,Z,U,V,W,WGHT,KPAR,IBODY,MAT,ILB(5)

◆ **ILB(1)** ... generation of the particle (1=primary, 2=direct descendants, ...)
◆ **ILB(2)** ... kind **KPAR** of the parent particle, only if **ILB(1)>1**
◆ **ILB(3)** ... Interaction mechanism **ICOL** that originated the particle
◆ **ILB(4)** ... $= Z \cdot 10^6 + \mathrm{IS1} \cdot 10^4 + \mathrm{IS2} \cdot 100 + \mathrm{IS3}$, when applicable
◆ **ILB(5)** ... can be defined by the user; it is transferred to all descendants of the particle

Numerical labels used to designate atomic electron shells.

In the case of non-radiative transitions, the label 30 indicates shells beyond the N7 shell.

label	shell	label	shell	label	shell
1	K $(1s_{1/2})$	11	N2 $(4p_{1/2})$	21	O5 $(5d_{5/2})$
2	L1 $(2s_{1/2})$	12	N3 $(4p_{3/2})$	22	O6 $(5f_{5/2})$
3	L2 $(2p_{1/2})$	13	N4 $(4d_{3/2})$	23	O7 $(5f_{7/2})$
4	L3 $(2p_{3/2})$	14	N5 $(4d_{5/2})$	24	P1 $(6s_{1/2})$
5	M1 $(3s_{1/2})$	15	N6 $(4f_{5/2})$	25	P2 $(6p_{1/2})$
6	M2 $(3p_{1/2})$	16	N7 $(4f_{7/2})$	26	P3 $(6p_{3/2})$
7	M3 $(3p_{3/2})$	17	O1 $(5s_{1/2})$	27	P4 $(6d_{3/2})$
8	M4 $(3d_{3/2})$	18	O2 $(5p_{1/2})$	28	P5 $(6d_{5/2})$
9	M5 $(3d_{5/2})$	19	O3 $(5p_{3/2})$	29	Q1 $(7s_{1/2})$
10	N1 $(4s_{1/2})$	20	O4 $(5d_{3/2})$	30	outer shells

Auger IS1-IS2-IS3: $\mathtt{ILB}(4) = Z \cdot 10^6 + \mathtt{IS1} \cdot 10^4 + \mathtt{IS2} \cdot 100 + \mathtt{IS3}$

X-ray IS1-IS2: $\mathtt{ILB}(4) = Z \cdot 10^6 + \mathtt{IS1} \cdot 10^4 + \mathtt{IS2} \cdot 100$

Example: x ray K-L2 of Cu (Z=29), $\mathtt{ILB}(4) = 29010300$

SUBROUTINE STORES(E,X,Y,Z,U,V,W,WGHT,KPAR,ILB)

Stores the initial state of a new particle in the secondary stack. The input arguments are the state variables of the particle and the array of labels **ILB**. The variables **IBODY** and **MAT** are read from common **TRACK**

Used for particle splitting

FUNCTION RAND(DUMMY)

Random number generator. The 'seeds' **ISEED1** and **ISEED2** must be initialized in the main program and transferred through the named common block **/RSEED/**
(to be declared as an external function in each subprogram that uses it)

PENGEOM subroutines

SUBROUTINE GEOMIN(PARINP,NPINP,NMAT,NBOD,IRD,IWR)

Reads geometry data from the input file, initializes the geometry package and prints the files **geometry.rep** and **pengeom-tree.rep**

SUBROUTINE LOCATE

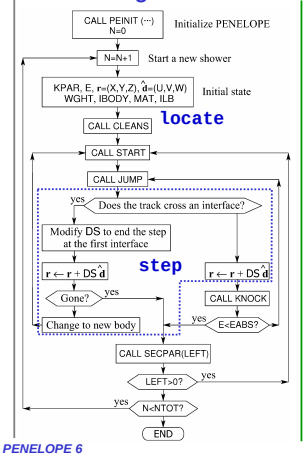
Determines the body **IBODY** that contains a point of coordinates **(X,Y,Z)** and its material **MAT** (values delivered though common **TRACK**)

SUBROUTINE STEP(DS,DSEF,NCROSS)

Performs the geometrical part of the track simulation. Moves the particle and changes body and material numbers as necessary. New values of the variables **X,Y,Z,IBODY,MAT** are delivered though common **TRACK**

DS ... path length to travel
DSEF ... travelled path length before leaving the initial material or completing the jump
NCROSS ... number of interfaces crossed

Flow diagram



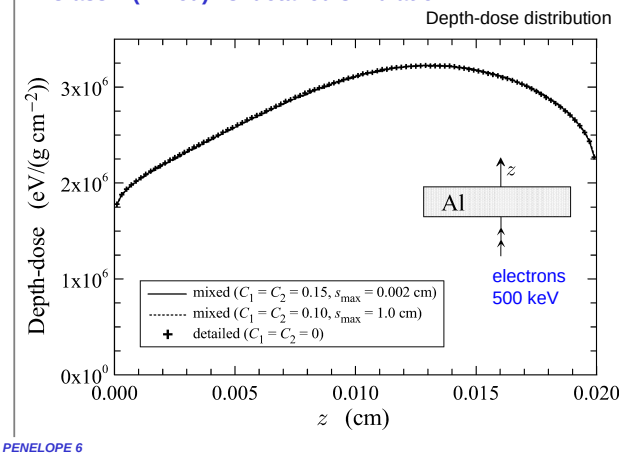

```
C ** MAIN PROGRAM
      CALL PEINIT(EPMAX,NMAT,5,6,INFO)
      CALL GEOMIN(PARINP,NPIP,NM,NB,7,6)
      N=0
C ** Start a new shower
 10   N=N+1
C Set the initial state variables.
      CALL LOCATE
      CALL CLEANS
C ** Start tracking in the material
 20   CALL START
 30   CALL JUMP(DSMAX(IBODY),DS)
      CALL STEP(DS,DSEF,NCROSS)
      IF(MAT.EQ.0) THEN
        GOTO 40  ! Scoring?
      ENDIF
      IF(NCROSS.GT.0) GOTO 20
      CALL KNOCK(DE,ICOL)
C Score relevant quantities
      IF(E.GT.EABS(KPAR,MAT)) GOTO 30
 40   CONTINUE
C Score relevant quantities
      CALL SECPAR(LEFT)
      IF(LEFT.GT.0) THEN
        GOTO 20  ! Scoring?
      ENDIF
      IF(N.LT.NTOT) GOTO 10
C Compute averages and write results
      END
```


Selecting the simulation parameters

$$E_{\mathrm{abs}}, C_1, C_2, W_{\mathrm{cc}}, W_{\mathrm{cr}}, s_{\max}$$

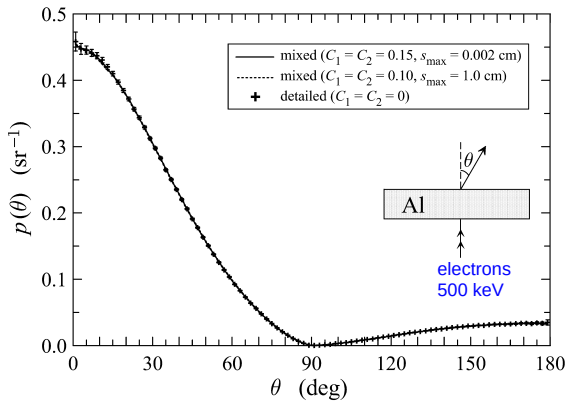
- E_{abs} such that $R(E_{\mathrm{abs}})$ < typical dimensions of volume bins
- C_1 ~ 0.05
- C_2 ~ 0.05 (effective only at very high energies)
- W_{cc} and W_{cr} "estimated" from experiment/scoring details W_{cc} , $W_{\mathrm{cr}} < E_{\mathrm{abs}}$ and energy resolution
- $s_{\max}$ (to ensure multiple soft interactions, to allow energy-loss corrections, transport in EM elds)

Class II (mixed) vs. detailed simulation



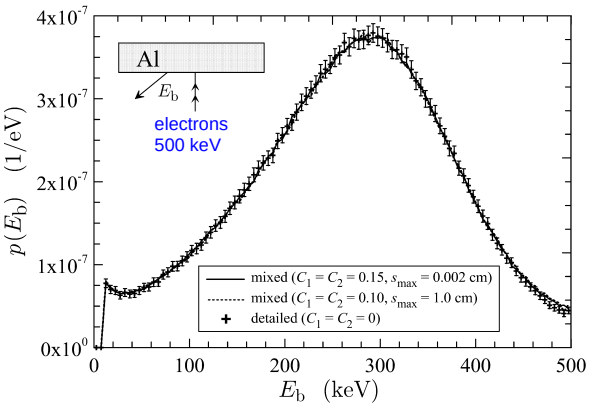
Class II (mixed) vs. detailed simulation

Angular distribution of emerging electrons



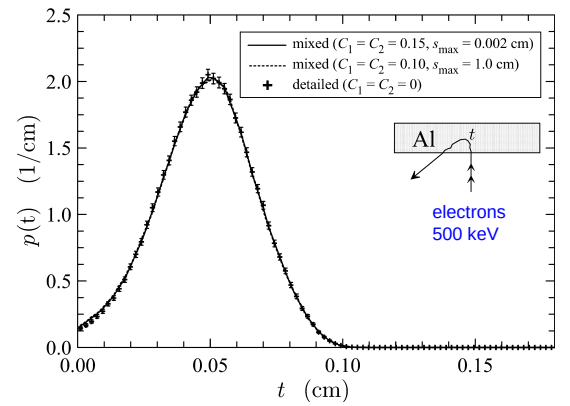
Class II (mixed) vs. detailed simulation

Energy distribution of backscattered electrons



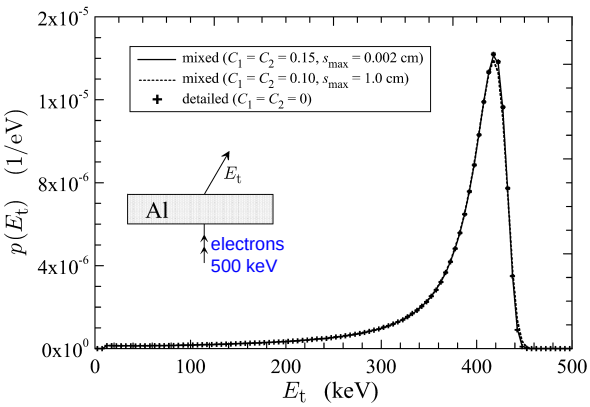
Class II (mixed) vs. detailed simulation

Path-length distribution of backscattered electrons



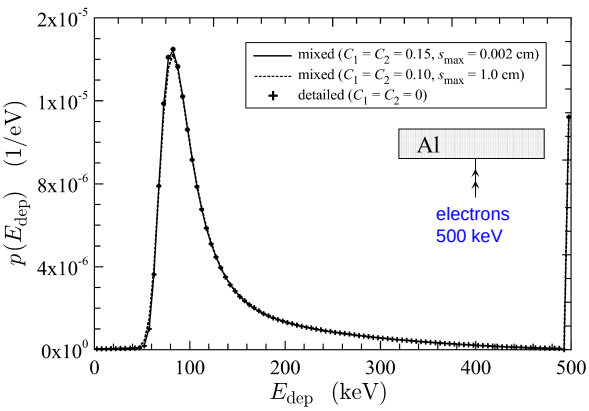
Class II (mixed) vs. detailed simulation

Energy distribution of transmitted electrons



Class II (mixed) vs. detailed simulation

Distribution of deposited energy



Practical simulation

Let's assume that your problem can be solved by using the example main program **penmain**

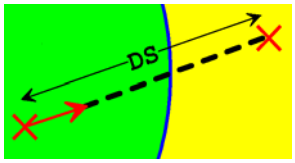
- Prepare your geometry definition file. Check that the geometry is properly defined using the viewing-debugging programs **gview2d** and **gview3d**
- Run the program **material** to generate the material data file. It must include all the materials present in your geometry, in the order assumed in the geometry definition file
- Edit the **penmain** input file. Assign tentative values to the simulation parameters for each material
- Perform preliminary short simulations, and check that the results do make sense
- Consider the possibility of applying variance reduction techniques
- Run the simulation. Make sure that statistical uncertainties are small enough

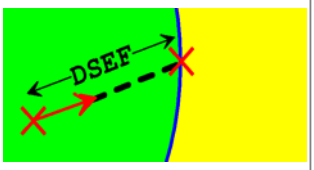
CHAPTER 5: Constructive quadric geometry

Outline:

- Tracking particles through complex geometries
- Quadric surfaces
- Constructive geometry: bodies, modules
- Geometry viewers
- The subroutine package **PENGEOM**

Tracking particles through complex geometries





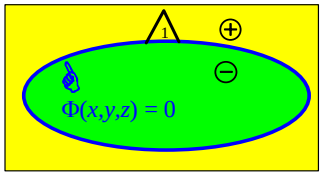
Physics routines determine the distance that the particle would travel (in an infinite medium)

Geometry routines move the particle and stop it when the trajectory intersects an interface

The job of geometry routines

- Determine where (body and material) is the particle
- Move particles along straight trajectories
- Find intersections with the surfaces that limit the body
- Determine the body and material after the interface

Quadric surfaces



- A surface divides the space into three mutually exclusive regions:
  1) $\Phi(\mathbf{r}) = 0 \rightarrow$ the surface itself
  2) $\Phi(\mathbf{r}) > 0 \rightarrow$ "outside", **side pointer = +1**
  3) $\Phi(\mathbf{r}) < 0 \rightarrow$ "inside" , **side pointer = –1**
- The "side pointer" of a region is not an absolute property. The eq. $-\Phi(\mathbf{r}) = 0$ defines the same surface but with the inner and outer regions exchanged

Intersections

- Intersection of the (straight) trajectory of a particle, $\mathbf{r} = \mathbf{r}_0 + \hat{\mathbf{v}}s$, with a surface $\Phi(\mathbf{r}) = 0$. The path length s is determined by solving the equation

$$\Phi(\mathbf{r}_0 + \hat{\mathbf{v}}s) = 0$$

This must be done many times (and limits the possible forms of Φ).

Quadric surfaces

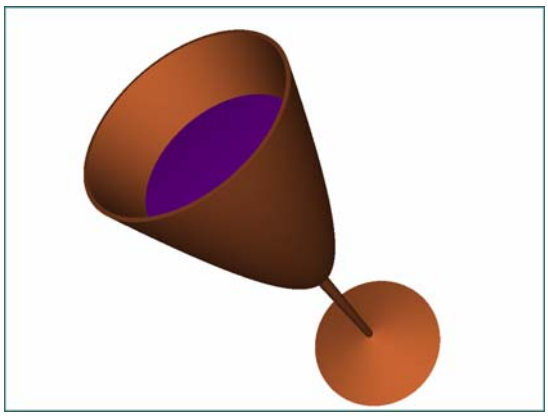
$$\Phi(\mathbf{r}) = A_{xx}x^2 + A_{yy}y^2 + A_{zz}z^2 + A_{xy}xy + A_{yz}yz + A_{zx}zx + A_x x + A_y y + A_z z + A_0 = 0$$

- Versatile enough to describe many real objects,
- Allow fast analytical calculation of intersections (quadratic equation)
- Some non-quadric surfaces give intersection equations that are also analytically solvable, but require much more CPU time. For instance, the torus,

$$\left[\sqrt{x^2 + y^2} - a\right]^2 + z^2 - b^2 = 0$$

gives a 4th order algebraic equation for the distance to the interface

With quadric surfaces we can describe virtually any object... approximately



Definition of quadric surfaces

- **Implicit form** (the list of non-zero coefficient values)

$$\Phi(\mathbf{r}) = A_{xx}x^2 + A_{yy}y^2 + A_{zz}z^2 + A_{xy}xy + A_{yz}yz + A_{zx}zx + A_x x + A_y y + A_z z + A_0 = 0$$

- **Transformations** (optional, in the quoted order):
- Scaling along the directions of the coordinates axis (the surface is "expanded" by a factor a, b, c in the direction x, y, z)

$$x \rightarrow ax, \qquad y \rightarrow by, \qquad z \rightarrow cz$$

- Rotation (**Euler angles**)

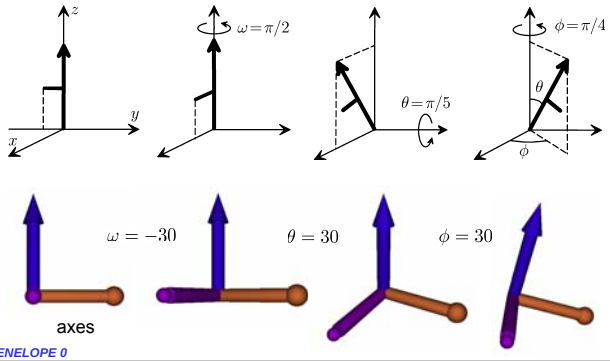
$$\mathbf{r} \rightarrow \mathcal{R}(\omega, \theta, \phi)\,\mathbf{r}$$

- Translation

$$\mathbf{r} \rightarrow \mathbf{r} + \mathbf{a}$$

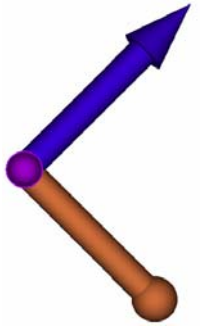
● Rotations and Euler angles

$$\mathcal{R}(\omega,\theta,\phi) = \mathcal{R}_z(\phi)\,\mathcal{R}_y(\theta)\,\mathcal{R}_z(\omega)$$



● Rotations and Euler angles

axes



$$R_x(\theta) = R_z(90^o)\;R_y(-\theta)\;R_z(-90^o)$$

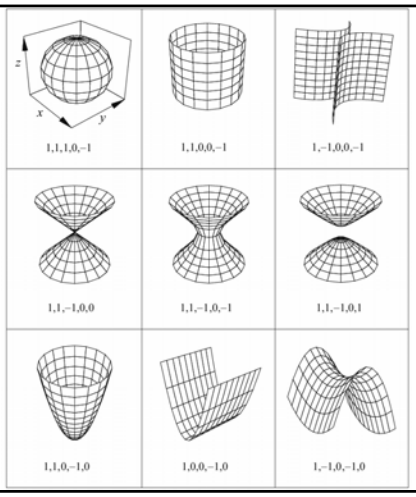
♦ **Reduced form** (the kind of surface is defined by its indices)

$$\Phi(\mathbf{r}) = I_1x^2 + I_2y^2 + I_3z^2 + I_4z + I_5 = 0$$

Reduced form	Indices					Quadric
$z-1=0$	0	0	0	1	−1	plane
$z^2-1=0$	0	0	1	0	−1	pair of parallel planes
$x^2+y^2+z^2-1=0$	1	1	1	0	−1	sphere
$x^2+y^2-1=0$	1	1	0	0	−1	cylinder
$x^2-y^2-1=0$	1	−1	0	0	−1	hyperbolic cylinder
$x^2+y^2-z^2=0$	1	1	−1	0	0	cone
$x^2+y^2-z^2-1=0$	1	1	−1	0	−1	one sheet hyperboloid
$x^2+y^2-z^2+1=0$	1	1	−1	0	1	two sheet hyperboloid
$x^2+y^2-z=0$	1	1	0	−1	0	paraboloid
$x^2-z=0$	1	0	0	−1	0	parabolic cylinder
$x^2-y^2-z=0$	1	−1	0	−1	0	hyperbolic paraboloid

the size, orientation and position of the quadric are defined by means of appropriate transformations:
expansion, **rotation**, and **translation** (in this order)

Non-planar quadrics in reduced form



● Syntax

♦ **Surfaces in reduced form**

```
0000000000000000000000000000000000000000000000000000000000000000000000000
SURFACE (    1)   reduced form
INDICES=( 1, 1, 1, 1, 1)
X-SCALE=(+1.0000000000000000E+00,    0)                (DEFAULT=1.0)
Y-SCALE=(+1.0000000000000000E+00,    0)                (DEFAULT=1.0)
Z-SCALE=(+1.0000000000000000E+00,    0)                (DEFAULT=1.0)
  OMEGA=(+0.0000000000000000E+00,    0) DEG            (DEFAULT=0.0)
  THETA=(+0.0000000000000000E+00,    0) DEG            (DEFAULT=0.0)
    PHI=(+0.0000000000000000E+00,    0) RAD            (DEFAULT=0.0)
X-SHIFT=(+0.0000000000000000E+00,    0)                (DEFAULT=0.0)
Y-SHIFT=(+0.0000000000000000E+00,    0)                (DEFAULT=0.0)
Z-SHIFT=(+0.0000000000000000E+00,    0)                (DEFAULT=0.0)
0000000000000000000000000000000000000000000000000000000000000000000000000
```

- Rotations are described by means of the **Euler angles**
- Lines defining default values can be removed from the definition file

NOTE: The integer label following each parameter,

```
X-SCALE=(+1.0000000000000000E+00,    0)                (DEFAULT=1.0)
```

allows the parameter value to be modified from the main program (selected geometry elements can be made mobile)

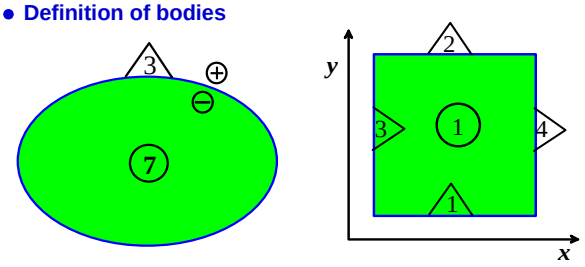
♦ **Surfaces in implicit form**

```
0000000000000000000000000000000000000000000000000000000000000000000000000
SURFACE (    1)   implicit form
INDICES=( 0, 0, 0, 0, 0)
    AXX=(+0.0000000000000000E+00,    0)                (DEFAULT=0.0)
    AXY=(+0.0000000000000000E+00,    0)                (DEFAULT=0.0)
    AXZ=(+0.0000000000000000E+00,    0)                (DEFAULT=0.0)
    AYY=(+0.0000000000000000E+00,    0)                (DEFAULT=0.0)
    AYZ=(+0.0000000000000000E+00,    0)                (DEFAULT=0.0)
    AZZ=(+0.0000000000000000E+00,    0)                (DEFAULT=0.0)
     AX=(+0.0000000000000000E+00,    0)                (DEFAULT=0.0)
     AY=(+0.0000000000000000E+00,    0)                (DEFAULT=0.0)
     AZ=(+0.0000000000000000E+00,    0)                (DEFAULT=0.0)
     A0=(+0.0000000000000000E+00,    0)                (DEFAULT=0.0)
1111111111111111111111111111111111111111111111111111111111111111111111111
  OMEGA=(+0.0000000000000000E+00,    0) DEG            (DEFAULT=0.0)
  THETA=(+0.0000000000000000E+00,    0) DEG            (DEFAULT=0.0)
    PHI=(+0.0000000000000000E+00,    0) RAD            (DEFAULT=0.0)
X-SHIFT=(+0.0000000000000000E+00,    0)                (DEFAULT=0.0)
Y-SHIFT=(+0.0000000000000000E+00,    0)                (DEFAULT=0.0)
Z-SHIFT=(+0.0000000000000000E+00,    0)                (DEFAULT=0.0)
0000000000000000000000000000000000000000000000000000000000000000000000000
```

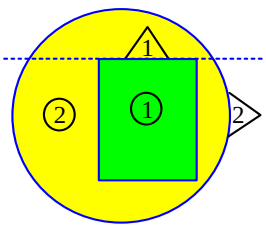
- Lines defining default values can be removed from the definition file

Constructive geometry; bodies and modules

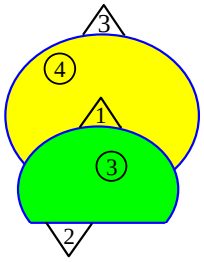
Definition of bodies




```
0000000000000000000000000000000000000000000000000000000000000000000000
BODY      (    1)    text
MATERIAL(    1)
SURFACE   (    1), SIDE POINTER=( 1)
SURFACE   (    2), SIDE POINTER=(-1)
SURFACE   (    3), SIDE POINTER=( 1)
SURFACE   (    4), SIDE POINTER=(-1)...
0000000000000000000000000000000000000000000000000000000000000000000000
```

Some bodies can not be defined with only surfaces

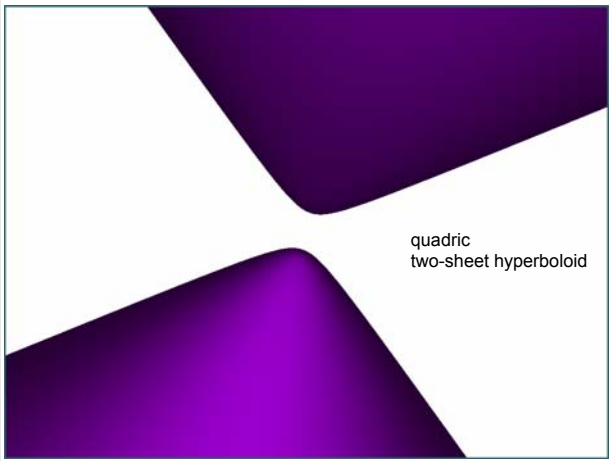


... but avoid using limiting bodies if you can do with only surfaces

```
0000000000000000000000000000000000000000000000000000000000000000000000
BODY      (    2)    text
MATERIAL(    1)
SURFACE   (    2), SIDE POINTER=(-1)
BODY      (    1)
0000000000000000000000000000000000000000000000000000000000000000000000
```


Example

```
-----------------------------------------------------------------
         Visualization of reduced quadric surfaces.
Define the desired quadric (surface 1) by entering its indices.
The region with side pointer -1 (inside the quadric) corresponds
to MATERIAL=1.
0000000000000000000000000000000000000000000000000000000000000000000000
SURFACE   (    1)    Reduced quadric
INDICES=( 1, 1,-1, 0, 1)
0000000000000000000000000000000000000000000000000000000000000000000000
BODY      (    1)    The interior of the quadric
MATERIAL(    1)
SURFACE   (    1), SIDE POINTER=(-1)
0000000000000000000000000000000000000000000000000000000000000000000000
END       0000000000000000000000000000000000000000000000000000000000000000000000
```

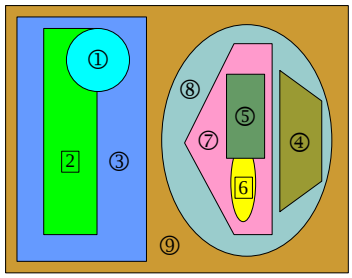
quadric
two-sheet hyperboloid

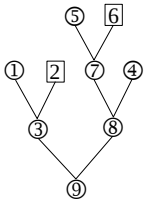
Modules

- Module ≡ Connected volume limited only by quadrics
  The particle can enter or leave the module only through the limiting surfaces (this is not true for bodies)
- A module may contain bodies and other modules (daughters). These have to be **fully contained** in the parent module
  The "cavities" of a module can be filled with a given material
- A module cannot overlap other modules or bodies other than its daughters
- A module can be rotated/translated as a whole

```
0000000000000000000000000000000000000000000000000000000000000000000000
MODULE    (    3)    text
MATERIAL(    2)
SURFACE   (    5), SIDE POINTER=( 1)
BODY      (    1)
MODULE    (    2)
1111111111111111111111111111111111111111111111111111111111111111111111
  OMEGA=(+0.0000000000000000E+00,    0) DEG           (DEFAULT=0.0)
  THETA=(+0.0000000000000000E+00,    0) DEG           (DEFAULT=0.0)
    PHI=(+0.0000000000000000E+00,    0) RAD           (DEFAULT=0.0)
X-SHIFT=(+0.0000000000000000E+00,    0)               (DEFAULT=0.0)
Y-SHIFT=(+0.0000000000000000E+00,    0)               (DEFAULT=0.0)
Z-SHIFT=(+0.0000000000000000E+00,    0)               (DEFAULT=0.0)
0000000000000000000000000000000000000000000000000000000000000000000000
```


Genealogical tree

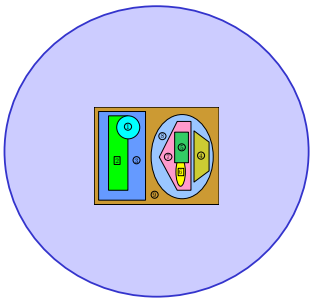




○ → modules (may have descendants, or not) □ → bodies

To speed up the simulation, each module should have a small number of daughters (small number of branches in each node).
With a properly defined genealogical tree, simulation speed is almost independent of the complexity of the geometry.

Genealogical tree



Enclosure: The whole geometry is assumed to be contained within a sphere of 10^7 cm radius (root module).

Bodies that extend beyond the enclosure are “truncated”

Other geometry definition options

♦ **INCLUDE** ≡ inserts an existing geometry definition file
(useful to split a long geometry definition file into smaller pieces)

```
0000000000000000000000000000000000000000000000000000000000000000000000
INCLUDE
FILE=   (filename.ext)
0000000000000000000000000000000000000000000000000000000000000000000000
```

♦ **CLONE** ≡ copies one module and moves it
(very useful to define repetitive structures)

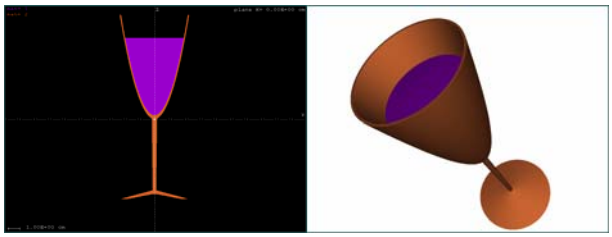
```
0000000000000000000000000000000000000000000000000000000000000000000000
CLONE    ( yyy)   copies one module and moves it
MODULE   ( xxx)
1111111111111111111111111111111111111111111111111111111111111111111111
  OMEGA=(+0.000000000000000E+00,   0) DEG          (DEFAULT=0.0)
  THETA=(+0.000000000000000E+00,   0) DEG          (DEFAULT=0.0)
    PHI=(+0.000000000000000E+00,   0) RAD          (DEFAULT=0.0)
X-SHIFT=(+0.000000000000000E+00,   0)              (DEFAULT=0.0)
Y-SHIFT=(+0.000000000000000E+00,   0)              (DEFAULT=0.0)
Z-SHIFT=(+0.000000000000000E+00,   0)              (DEFAULT=0.0)
0000000000000000000000000000000000000000000000000000000000000000000000
```

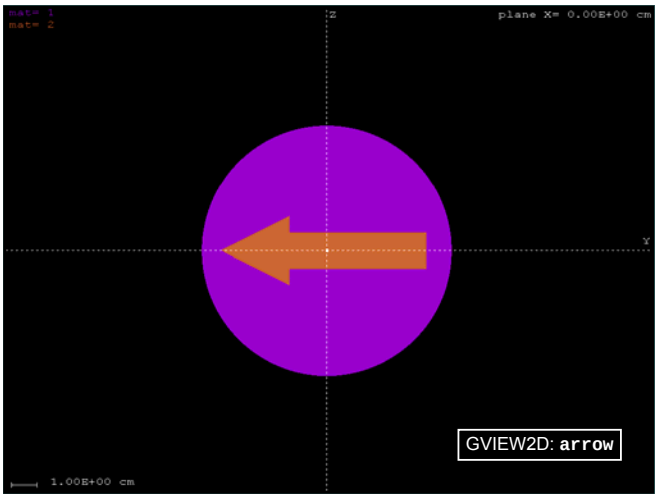

Geometry viewers

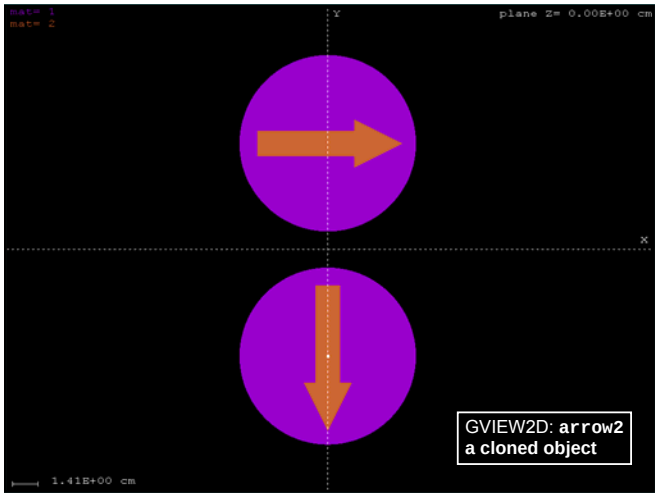
GVIEW2D two-dimensional (material and body maps)
GVIEW3D three-dimensional images (quite realistic)

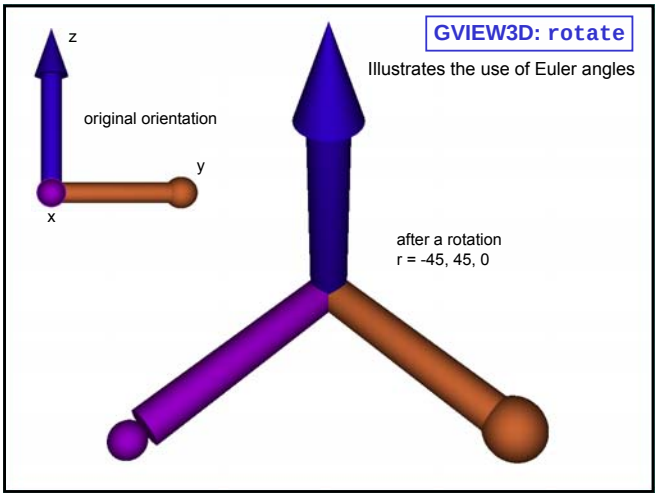
GVIEW2C cylindrical geometries. Similar to GVIEW2D.

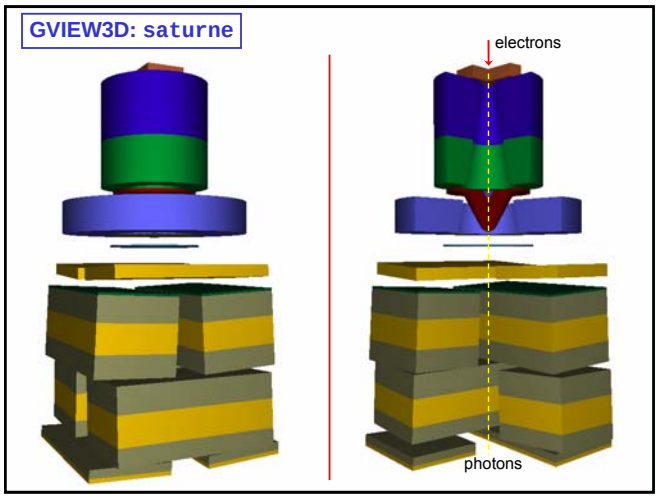
They run only under MS Windows.

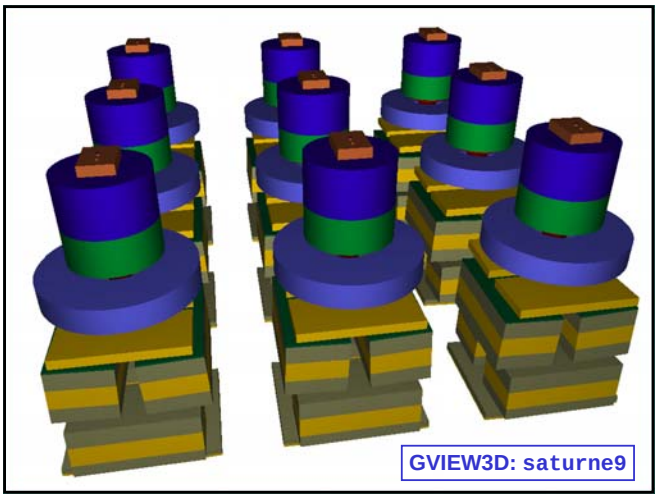












Debugging

- GVIEW generates a debugging file named **"geometry.rep"**, as well as an auxiliary file **"pengeom_tree_rep"** that contains details on the genealogical tree structure
- GVIEW stops when a syntax error or an obvious inconsistency is found The offending input datum usually appears in the last printed lines of the debugging file
- Errors or inconsistencies in the definition of the geometry are usually identified by visual inspection of the images created by GVIEW

WARNING: PENGEOM assigns its own **labels** to all geometry elements. To specify a body or module, we need to know its "internal" label.

- The surface and body labels used by PENGEOM during simulation are printed in file **"geometry.rep"**. The body labels can also be identified by running GVIEW2D with **display mode** = 2

The subroutine package PENGEOM

User-callable routines

SUBROUTINE GEOMIN(PARINP,NPINP,NMAT,NBOD,IRD,IWR)

Reads geometry data from the input file, initializes the geometry package and prints the files **geometry.rep** and **pengeom_tree.rep**

SUBROUTINE LOCATE

Determines the body that contains a point of given coordinates and its material

SUBROUTINE STEP(DS,DSEF,NCROSS)

Performs the geometrical part of the track simulation. Moves the particle and changes body and material numbers as necessary

Making geometry parts "mobile"

Input values of pre-selected geometry parameters can be modified from the main program.

```
SUBROUTINE GEOMIN(PARINP,NPINP,NMAT,NBOD,IRD,IWR)
DIMENSION PARINP(NPAR)      NPINP ≤ NPAR
```

Example: The input file element

```
0000000000000000000000000000000000000000000000000000000000000000000000
SURFACE (   1)   sphere, R=1, centered at (0,0,0)
INDICES=( 1, 1, 1, 0,-1)
X-SHIFT=(+0.000000000000000E+00,   1)              (DEFAULT=0.0)
Y-SHIFT=(+0.000000000000000E+00,   2)              (DEFAULT=0.0)
0000000000000000000000000000000000000000000000000000000000000000000000
```

with **NPINP=2, PARINP(1)=1., PARINP(2)=3.**
is transformed (internally) into

```
0000000000000000000000000000000000000000000000000000000000000000000000
SURFACE (   1)   sphere, R=1, centered at (1,3,0)
INDICES=( 1, 1, 1, 0,-1)
X-SHIFT=(+1.000000000000000E+00,   0)              (DEFAULT=0.0)
Y-SHIFT=(+3.000000000000000E+00,   0)              (DEFAULT=0.0)
0000000000000000000000000000000000000000000000000000000000000000000000
```


Impact detectors

SUBROUTINE STEP does not stop particles when they cross an interface between adjacent bodies of the same composition. That is, these interfaces are not visible from the main program

COMMON/QKDET/KDET(NB)

KDET(KB) is initialised to zero. Body **KB** is made part of impact detector **IDET** by setting **KDET(KB)=IDET** (in main program). When particles enter body **KB** coming from a body that is not part of detector **IDET**, they are halted at the surface of the active body and control is returned to the main program.

Thus, detector surfaces are made visible from the main program.

Impact detectors are useful for visualising bodies (**GVIEW2D**), for simulating real detectors and for generating phase-space files (**PENMAIN**).

● Limitations of PENGEOM

♦ To avoid being caught in an endless loop, STEP does not leave the particle exactly on the surface, but some 10^{-8} cm farther.

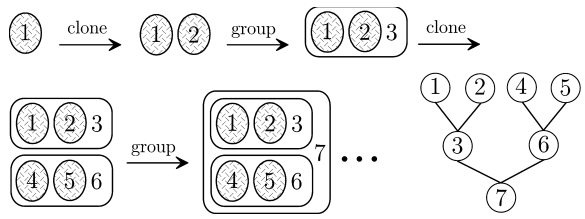
♦ Consequently, geometrical details smaller than ~10^{-7} cm (i.e. 10 Å) are not guaranteed to be described correctly.

♦ PENGEOM admits up to 10,000 surfaces and 5000 bodies (and modules). The maximum number of bodies in a module or the number of surfaces defining a body is set to 250. These parameters can be changed by editing the source file **`pengeom.f`** (but **with extreme care!**).

♦ Note that the speed of the simulation depends strongly on the structure of the genealogical tree. **The responsibility of setting it appropriately rests with the user.**

● Example of "optimized" modular tree

Repetitive structure



Each step duplicates the existing tree and adds a common root module