EFICASS Manual

The CHORUS Collaboration

August 30, 1998

Preface

Finally, the long awaited EFICASS manual is released. Although far from being complete, it provides a framework for a manual which eventually should encompass all information on EFICASS which was previousely scattered around on various README files and HTML pages.

The purpose of this manual is twofold. Firstly, it should serve the ordinary user as an useful reference guide and look-up table. Secondly, it should give beginners a "gentle" introduction to the world of EFICASS. Especially, chapters on the *proper* installation and settings of the runcards should be of (vital) interest for future generations of summer students who have had not yet the pleasure to be exposed to the beauty of EFICASS.

In order to make this manual an enriching and fruitful experience, advanced users, who are familiar with some of EFICASS's pitfalls, are welcome to supplement, correct or add chapters.

Contents

Chapter 1

How to install EFICASS on your system

1.1 Where is the code ?

The code can be found in the standard Chorus directories under afs. There is a production version, a "newest", but reasonably trusted version, and a development version, which can be found, respectively in:

```
/afs/cern.ch/user/c/chorus/@sys/old/src/eficass
/afs/cern.ch/user/c/chorus/@sys/pro/src/eficass
/afs/cern.ch/user/c/chorus/@sys/new/src/eficass
/afs/cern.ch/user/c/chorus/@sys/dev/src/eficass
```

or in a shorter notation:

```
$CHORUS/pro/src/eficass
$CHORUS/new/src/eficass
$CHORUS/dev/src/eficass
```

> N.B. The Chorus environmental variable is set to `CHORUS_ROOT=$CHORUS/pro` by default.

In `.../eficass` there are the following files and directories:

`user_GNUmakefile`	- makefile to be used by the user
`librarian_GNUmakefile`	- makefile to be used by code maintainer
`eficass_run`	- shell script to run the executable
`input/`	- subdirectory with run-card files
`digi/`	- subdirectories with fortran source `*.F` files, the latest contain also `#include` and `#if` directives to be (pre)processed.
`draw/`	
`geant/`	
`geom/`	
`init/`	
`kine/`	
`main/`	
`step/`	
`zini/`	
`zout/`	
`include/`	- subdirectory with "include" files for `*.F` source files.
`depends/digi`	
`.../draw`	
`.../zout`	- subdirectories with dependency `*.d` files. They are created when executables are linked and the user should not be concerned about them.

1.2 Installation

In order to install EFICASS on your system, you should go to the directory, in which you intend to install the EFICASS tree. Make sure that the `CHORUS_ROOT` variable points to the desired Chorus level. If not, type at the shell prompt:
`> export CHORUS_ROOT=$CHORUS/pro` (or `/new` or `/dev`)

> N.B. The directory `$CHORUS_ROOT/src/eficass` is referred as "standard place" hereafter.

Then enter:
`> efimake`
If this command is not defined in your path (usually, it is in `$CHORUS/bin`), type the following:
`> gmake install -f $CHORUS/dev/src/eficass/user_GNUmakefile`

This will do the following:

- it creates the subdirectory `eficass`
- it copies there the `eficass_run` script and the `user_GNUmakefile` as `GNUmakefile`
- it creates an (empty) subdirectory `eficass/output` for Eficass output files
- it creates an (empty) subdirectory `eficass/include` for your include files
- finally, it creates the subdirectory `eficass/input` and copies there the following files:

```
efirun.crd,
efigraf.crd,
efiphys.crd;
```

It is supposed that the user normally would need to modify only these files according to his/her needs. The rest of the input card files, namely:

```
efigeom.crd,
eficalo.crd,
efimap.crd,
efiread.crd
```

are not supposed to be changed, so symlinks to them will be established (and removed after the job has finished) by the run script.

If the user wants to change them also, he/she has to copy them manually into the `input/` subdirectory from the standard place and modify them. The run script will automatically use private copies to run the executable.

If you are not going to change some of the data card files in your input directory, it is safer just to delete them. The run script will use the standard ones in this case.

1.3 Modifications of the code

The main tool for code modification is the GNUmakefile in the users' `eficass` directory. Please, consult comments there for details. The procedure should be the following:

- copy the EFICASS `*.F` files you want to modify from the standard place somewhere in the `~/.../eficass` directory or first-level subdirectory. It is strongly recommended (though not obligatory) to resemble the subdirectory structure of the standard EFICASS directory;
- copy "include" files to be modified only in `eficass/include` subdirectory (nowhere else!);
- modify what you want to modify, or create (if you want) new `*.F` files somewhere under `~/.../eficass` or first sub-level and "include" files *only* in `~/.../eficass/include` (One may have `*.F` files in `eficass/include`, too.);
- run `gmake` from within `~/.../eficass` to create your own executable (see section 1.5 on page 11);

Then an `eficass/bin` directory is created (if non-existent) and respective EFICASS executables are there with your modifications replacing standard EFICASS routines or, if new, linked together with them.

1.4 Running EFICASS

There are 4 Eficass executables in `$CHORUS_ROOT/bin`, namely:

`eficass`	- optimised code without graphical functionality.
`eficass_graph`	- optimised code with graphical functionality.
`eficass-g`	- code with debug information, not-optimised, without graphical functionality.
`eficass_graph-g`	- code with debug information, not-optimised, with graphical functionality.

The user may have his/her private set of these 4 (modified) executables in the `~/.../eficass/bin` directory.

To run any of these 8 executables:

- go to the `~/.../eficass` directory.
- make sure that the `$CHORUS_ROOT` variable is set to the path you desire. (If this variable is not set, the run script will set it to `$CHORUS/pro`).
- export the variable `BEAMFILE`=full-path-beam-file-name,

which will be used by the script as the name of the desired beam file (event generator file). The default is:
`/afs/cern.ch/user/j/jvissche/public/beam/jetta.rfz`.

Then type:

```
> eficass_run [option]
```

where `[option]` selects what to run:

option	action
`-std`	run standard version from `$CHORUS_ROOT/bin`
`-std_deb`	run debugger on -g version from `$CHORUS_ROOT/bin`
`-std_graph`	run graphics version from `$CHORUS_ROOT/bin`
`-std_dg`	run debugger on -g graphics version from the directory `$CHORUS_ROOT/bin`
< *none* >	run private standard version
`-deb`	run debugger on private -g version
`-graph`	run private graphics version
`-dg`	run debugger on private -g graphics version
`-help`	prints help message

Files with run cards will be read out from your `~/.../eficass/input` directory and files which are not there from the standard `eficass/input` directory.

Output files will go to `eficass/output` subdirectory and will have names like `$PROCESS.suffix` according to the value of the PROCESS variable, which is defined in the run script. The various suffices stand for:

ps	- ps file from graphics version
out	- not used
deb	- not used
sfz	- simulation output (not implemented)
rfz	- raw data output file
dat	- not used
hbook	- hbook file with generated histograms
evt	- last event generated and seeds for GRNDM

1.5 Running gmake

When the user has modified source files, he/she has to run `gmake` to create/update his/her own executable in the `eficass/bin` directory. This has to be done from within the `~.../eficass` directory, where the `GNUmakefile` resides, though it is not obligatory.

Before using `GNUmakefile` the user may want to edit it to set CERNlibs level different from `pro`, or to set the variable `ROOT_DIR` pointing to the

`~/.../eficass/` directory, if, for some reason, the job is to be executed from another directory.

If you are not in the `~/.../eficass/` directory, you should make the `GNUmakefile` available in the current directory either by coping or linking. This is needed, because for some targets recursive use of `gmake` is implemented.

N.B. Instead of changing the `GNUmakefile` you can redefine any variable which it gets from the environment just by defining it on the command line, e.g.:

```
> gmake [options] [targets] CHORUS_ROOT=$CHORUS/dev
ROOT_DIR=~/.../eficass/
```

When runing on AIX, one may want to set the `LOADMAP` variable to obtain the linkers' map file, when it links the executable with the debug option.

For other platforms this option is not implemented, but the user can do it easily as for AIX, if the corresponding linkers' options are known.

It is instructive to read comments in the `GNUmakefile` before you start using it.
Possible commands to run `gmake` with this `GNUmakefile` are:

`gmake` : creates/updates executable `bin/eficass` (optimised, without graphics)

`gmake debug=yes` : creates/updates executable `bin/eficass-g` (debug, without graphics)

`gmake graphics=yes` : creates/updates executable `bin/eficass_graph` (optimised, with graphics)

`gmake debug=yes graphics=yes` : creates/updates executable `bin/eficass_graph-g` (debug, with graphics)

`gmake install` : (re)installs Eficass from `$CHORUS_ROOT` into subdirectory eficass/ of `$ROOT_DIR` (default: current directory)

`gmake diffs` : creates patches to send to the librarian (Patches have the names of the form `name.patch` for `name.F` or `name.patch_incl` for include file `name` and reside in the directory of their parent files.)

`gmake clean/clean_object/clean_depends` : removes object and/or dependency files of a version, specified with options `debug` and `graphics`.

`gmake clean_lst` : removes `*.lst` compiler output files (produced when compiled with `debug=y` option) `gmake clean_all` : removes object and dependency files of all 4 versions and `*.lst` compiler output file

`gmake clean_patches/clean_output` : removes patches or EFICASS output files, respectively

`gmake help` : prints short help message

> N.B.: One cannot mix targets `clean[suffix]` and `diffs` with other targets in a single invocation of gmake.

Add flag -s for silent work of gmake.

A help message for gmake itself is obtained by entering:

`gmake -help`, or `gmake -h`

When gmake creates/updates executables it creates/updates object (`*.o`) and so called dependency (`*.d`) files. They establish a connection between `*.F` files and "include" files declared in `*.F` files. They are different for each version of the executable. These files reside in directories, where respective `*.F` files are. Do not delete them without good reason. `gmake` will use them in the next run and will remake executable quite faster.

1.6 Info for the librarian

This section is only relevant for the librarians of EFICASS. It briefly describes the steps to be taken to install a new version of EFICASS.

1. Checkout new release of Eficass

   - go to a directory,
   - edit the `librarian_GNUmakefile` (it should be somewhere in the Chorus tree) to set the following variables properly:
     `RELEASE = eficass-99-99-99`← Put here the desired release tag
     `ROOT_DIR = /afs/cern.ch/chorus/offl/@sys/dev/`← Put here the top directory, where the Eficass tree is to be installed under the `../src` subdirectory.

     These variables can be set from the `gmake` command line, instead of editing the makefile.
   - issue the command:
     `> gmake checkout -f /somewhere/librarian_GNUmakefile`

- go to `$ROOT_DIRsrc/eficass` and issue:
  `> ln -s librarian_GNUmakefile makefile`
- edit the makefile to set the `ROOT_DIR` variable properly for further use

2. (Re)build all 4 executables on a certain platform
   - open a window on that platform (e.g. hpplus)
   - go to the `$ROOT_DIRsrc/eficass` directory
   - issue the command
     `> gmake -s bins`
   - go for a coffee...
3. (Re)build one particular executable on a certain platform
   - open a window on that platform (e.g. hpplus)
   - go to the `$ROOT_DIRsrc/eficass` directory
   - issue the command
     `> gmake -s [graphics=yes] [debug=yes]`
   - go for a short coffee...
4. Delete dependency files for a certain version
   - go to the `$ROOT_DIRsrc/eficass` directory (platform doesn't matter)
   - issue:
     `> gmake -s [graphics=yes] [debug=yes] clean_depends`
5. Delete object or object and dependency files for a certain version
   - open a window on the desired platform (e.g. hpplus)
   - go to the `$ROOT_DIRsrc/eficass` directory
   - issue:
     `> gmake -s [graphics=yes] [debug=yes] clean_objects`
     for object files or
     `> gmake -s [graphics=yes] [debug=yes] clean`
     for both.
6. Delete object and dependency files for all 4 version
   - open a window on the desired paltform (e.g. hpplus)
   - go to the `$ROOT_DIRsrc/eficass` directory
   - issue:
     `> gmake -s clean_all`

Chapter 2

Runcards

Runcards are contained in the input files of EFICASS, which configure all necessary parameters you need for your run. For ordinary users of EFICASS this is the only thing they ever need to know about EFICASS. This chapter gives an overview of all runcards in the order of their importance. Although their use is described in the files itself, a more detailed description of them with hints of how to avoid problems is given here.

2.1 Which runcards are there ?

EFICASS reads in 7 `*.crd` files before each run which are located in the subdirectory `input` of EFICASS. Four of them, namely `efigeom.crd`, `eficalo.crd`, `efimap.crd`, `efiread.crd` contain standard settings which should *not be touched* without good reason. Their are not described here. The configuration of the remaining three files, however, is vital for a successful run.

Chapter 3

Realistic vertex positioning in EFICASS

The routines for the primary vertex positioning have recently (8/98, M. Möbius) been replaced by a more general and efficient approach, although the actual Monte Carlo algorithm remained unchanged. In the following the method, the MC algorithm and its implementation into EFICASS are explained. This is followed by a series of plots which illustrate the results for the most important volumes in the setup, namely the target, the calorimeter, the spectrometer and last,but not least the whole setup (FRAM).

3.1 Outline of the algorithm

The MC algorithm for the positioning is fairly straight forward and is outlined below:

1. Obtain ν production point and its direction (in the EFICASS frame)

2. Check, whether the ν intersects the volume under consideration. If not, the ν is simply *rejected.*

3. A number between 0 and `DENSMAX` is sampled, where `DENSMAX` is the maximum of the path-integral $\sum_{Frontplane}^{Backplane}(x_i\rho_i)$, where Front/Backplane denote a point on the front and back plane of the volume, respectively.

4. Starting off from the intersection point of the ν with the front plane of the volume, the neutrino is tracked until the sampled value of the path-density is reached. The vertex is then positioned there.

5. If the ν arrives at the back plane before the sampled path-density is reached, it is *rejected.*

Comments:

- Strictly speaking, the vertex should additionally be weighted with the cross-section of the ν. However, this is already done in the beam generator `gbeam` [1].
- `DENSMAX` can be a bit tricky to find, since the volumes are usually highly inhomogenous. Moreover, if `DENSMAX` is set too high, the Monte Carlo becomes very inefficient; i.e. a lot of neutrinos are rejected.An easy and fairly reliable method to determine a safe value for `DENSMAX` is outlined in section 3.5
- The algorithm is, of course, also applicable to particles other than the neutrino.

3.2 Motivation for a new vertex positioning routine

The position of the primary vertex is weighted with the density of the material which the ν traverses. (see details of the algorithm in section 3.1 on page 17). Hence, a routine for the realistic vertex positioning always involves some sort of scanning of the volume in which the vertex should be placed. Since the structure of the volumes under consideration (e.g. the calorimeter) are rather complicated objects, the scanning becomes non-trivial. The reasons for replacing the existing[2] code were threefold:

- The current routine uses its own, simplified geometry of the target and the calorimeter (other volumes have not been implemented). The Geant volume definitions are *not* used. Subvolumes which are not box-shaped are simply discarded (e.g. the X-ray gun).
- Each beamcard uses a separate routine.→ routines are not general
- Impossible to modify the code, such that it can be used for other volumes, like the spectrometer, for example.

[1] As far as I know, the cross-section weighting in gbeam assumes that the ν traverses the emulsion

[2] By existing or "current" code I mean the vertex positioning routine which was originally implemented in EFICASS and has now, 8/98, been replaced in the `/dev` area by this approach.

The new routine is much more general and accurate, since it uses the Geant volumes which are already implemented into EFICASS. This also means, that it is independent of changes of the internal structure of the volumes. If, for example, the emulsion in the target were replaced with other target material, only `DENSMAX` would have to be adjusted (see section 3.5). The routine is also takes into account the HIDE card settings (e.g. if some emulsion stacks are hidden by the HIDE card). The routine would even work, if the dimensions and position of the volume were changed. The new routine is based on the Geantino approach which is explained in the next section.

3.3 The Geantino approach

The Geantino is a dummy particle provided by Geant. It does not have any interaction with matter and therefore traverses all volumes in a straight line. This property makes it perfectly suitable for scanning purposes. The main idea is summarised below:

> The idea is to track a geantino along the path of the ν. This approach takes advantage of the features provided by the Geant tracking package.
>
> - the computation of the path-density becomes very easy, because Geant provides the information about the current position and density of the medium.
> - The routine is fast, because during tracking the geantino "jumps" from one medium boundary to the next, provided the AUTO run-card is set to 1.

However, this approach involves some problems in terms of the program flow of EFICASS:

- The vertix needs to be positioned during the execution of `gukine`. This means that the tracking routines provided by Geant have to be called directly. Usually all these routines are controlled by the master routine `grun`. So the tracking starts automatically after the execution of `gukine`. Calling these routines directly from a user subroutine requires the resetting of some internal counters after the Geantino tracking has finished. In detail, the counters for the tracks (`NTRACK`, `LTRK`) and vertices (`NVERTX`, `LVTX`) have to be set to zero.

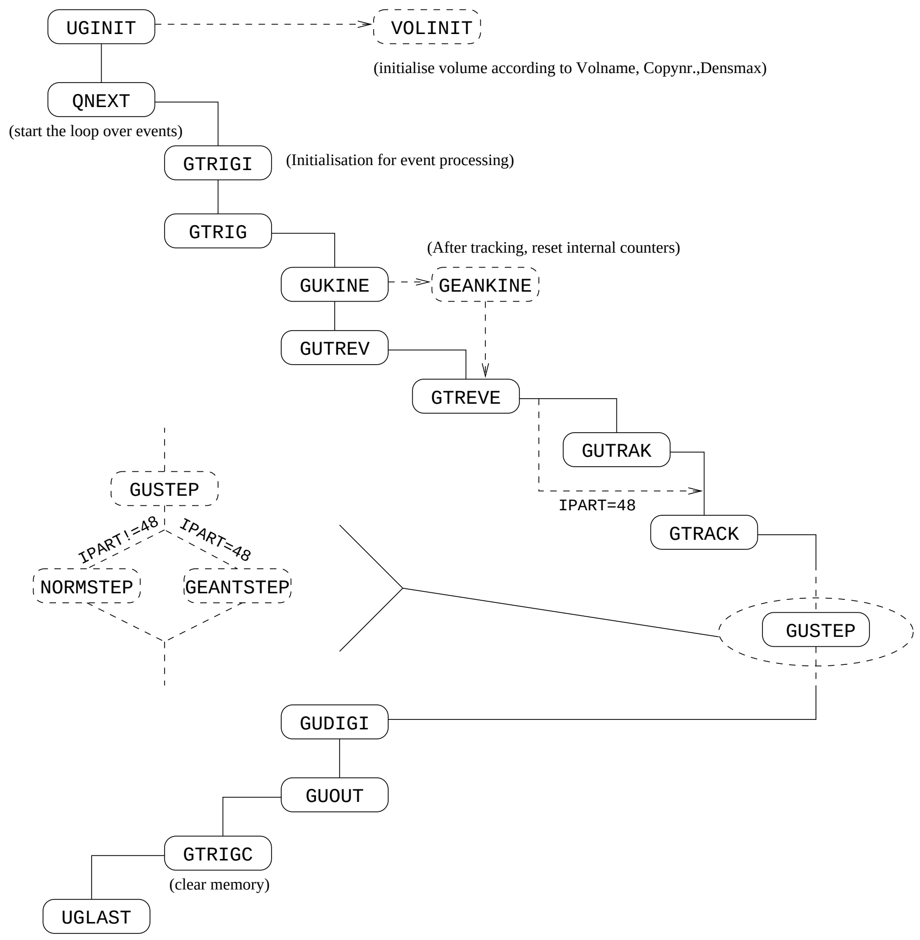

Figure 3.1: This is an simplified flowchart of EFICASS, which illustrates how the vertex positioning routines are implemented into the program.

- There are some user defined tracking routines, namely **gutrak** and **gutrev** which are not needed for the geantino tracking and thus, have to be bypassed. However, the user defined routine **gustep** is needed. So two different **gustep**'s are required. One for the Geantino tracking (**geantstep**) and the other one for the usual tracking (**normstep** which used to be **gustep**). **gustep** was replaced by a simple conditional statement which calls the appropriate routine.

These problems are illustrated in the flowchart shown in figure 3.1

3.4 Summary of the changes

3.4.1 What does the new routine need ?

For vertex positioning in a certain volume only three parameters are required:

1. Volume name:
   - needs to be a *non-rotated BOX* volume.
   - needs to be in the first sublevel of FRAM.
   - internal structure is arbitrary.
2. Copy number: it is simply the copy number of the Geant volume defined in EFICASS.
3. `DENSMAX`: This is the upper bound of the Monte Carlo as described in section 3.1 on page 17. See details on `DENSMAX` in section 3.5 on page 22.

If the user requires the vertex to be positioned in a volume which is not in the first sublevel (e.g. the emulsion stacks), a special beamcard can be chosen which enables the user to define his/her own box volume, as described below.

3.4.2 Runcard changes

Here, the modifications and extensions of the runcards are briefly described:

- The *realistic* vertex positioning routines for beamcard 5 (calorimeter) and 7 (target) have been replaced.
- Introduced beamcard 9 : Spectrometer.
- Introduced beamcard 10 : User volume with additional parameters in the following order: `DENSMAX`, x/y/z position of volume w.r.t FRAM, dimensions of the box; the default values specify FRAM !
- Introduced beamcard 11 : First sublevel volume. Need to specify the three parameters mentioned in the subsection above; namely the volume name (`CHARACTER*4` Geant name of the volume), the copy number and `DENSMAX`.
- Added DEBG 15 : Debug output from all routines related to the vertex positioning.
- Added DEBG 16 : Position the vertices only, skip subsequent tracking.

- Modified HIST 74: Now, the N-Tuple contains not only the position of the primary vertices, but also includes the density and medium number at that point.
- Modified HIST 76: Changed the axis from radial distance r to r^2
- Deleted HIST 81: Debris from the old routine.
- Added HIST 83: Beam profile at intersection (=front) plane. Since the neutrinos are nearly parallel aligned to the z-axis, this plot looks rather similar to HIST 76.

3.4.3 Deficiencies

Previousely, the user could specify subvolumes of the calorimeter in beamcard 5 . These additional parameters are not supported anymore ! An error message will pop up, if the run card has not been changed in this case. If the user still wants to position vertices in subvolumes of the calorimeter, he should do so via beamcard 10.

In very rare cases, tracking errors can occur. In this case, the pathdensity gives very high values and the neutrino is then rejected by the MC. Since the chance for this to happen is roughly $1 : 10^6$, it does not affect the primary vertex positioning in any way.

The reason for this bug lies in the AUTO run card. The default setting is 0, which means that the tracking steps have a fixed size as specified in the media definitions in EFICASS. In rare cases this can lead to tracking errors, if the Geantino (or any other particle) comes very near to a medium boundary, such that the distance between the particle and the boundary is below the machine precision.

This bug vanishes if the AUTO card is set to 1. In this case the step length is automatically computed by Geant.

3.5 How to find `DENSMAX`

The upper bound of the MC, `DENSMAX`, is hardwired in the code for beamcards 5,7,9 and 10 (if the default is used). Usually, they don't have to be modified. If, however the geometry of these volumes change or if the user wants to use beamcard 10,11 `DENSMAX` needs to be found or adjusted. Since the volumes are usually highly inhomogenous, analytic solutions can be rather cumbersome. However, there exists a simple and fairly reliable method outlined below.

1. Switch on DEBG 15 16.
2. set AUTO to 1 in order to avoid the tracking bug mentioned in section 3.4.3 on page 22.

3. Initially, a high value for `DENSMAX` should be chosen, which is safe (typically 4000 g/m^2).

4. In this case, the MC rejects a lot of neutrinos. The program stores the highest value of the pathdensity it encountered when the neutrino is rejected.

5. After the run, a line at the end of `tau0mu.lis` will say:
`DENSMAX should be greater than` ...

> N.B. In order to make sure that `DENSMAX` is indeed the upper bound, one should add $10 - 20\%$ to this value. Moreover, a beam file with at least 10000 neutrinos should be used.

If `DENSMAX` is too low, the following error message will appear:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!! UPPER BOUND OF MC TOO LOW !!
!! ......... > DENSMAX       !!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

3.6 Results

In this section plots[3] of the vertex distributions are shown for the most relevant volumes, namely TARGet, CALOrimeter, SPECtrometer and FRAM, the whole setup. In addition, the new routine was compared with the old one for the target and the calorimeter. The runs were made with a beam file containing 10000 ν's. The only hidden volume was 'MUTR'. People who intend to reproduce the graphs should note that the plot showing the distribution of vertices in the various media (figures 3.5, 3.6, 3.6, 3.6) are a bit tedious to produce in PAW. For that reason I have included the `*.kumac` file in appendix A.

[3]The plots of the z-profile have still r and not r^2 on the axis—sorry...

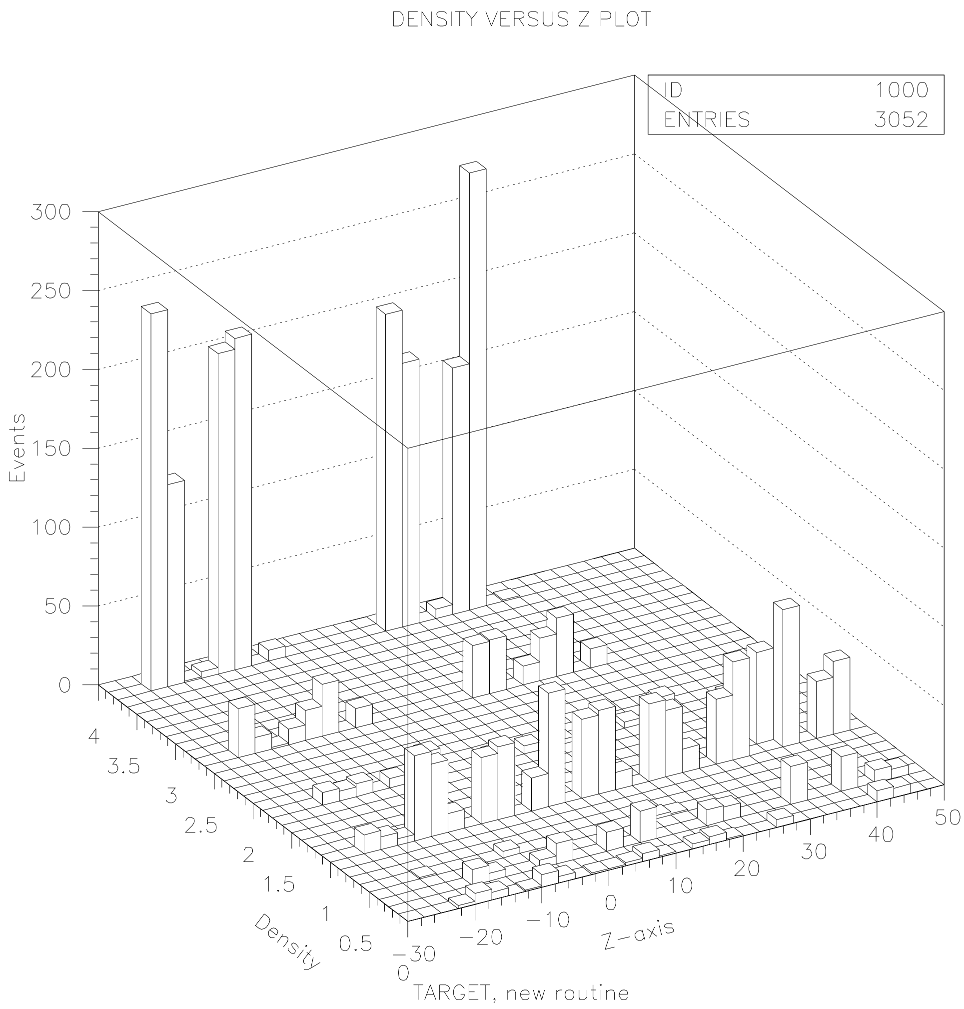
DENSITY VERSUS Z PLOT
ID 1000
ENTRIES 3052
Events
300
250
200
150
100
50
0
Density
4
3.5
3
2.5
2
1.5
1
0.5
0
Z-axis
-30
-20
-10
0
10
20
30
40
50
TARGET, new routine

Figure 3.2:

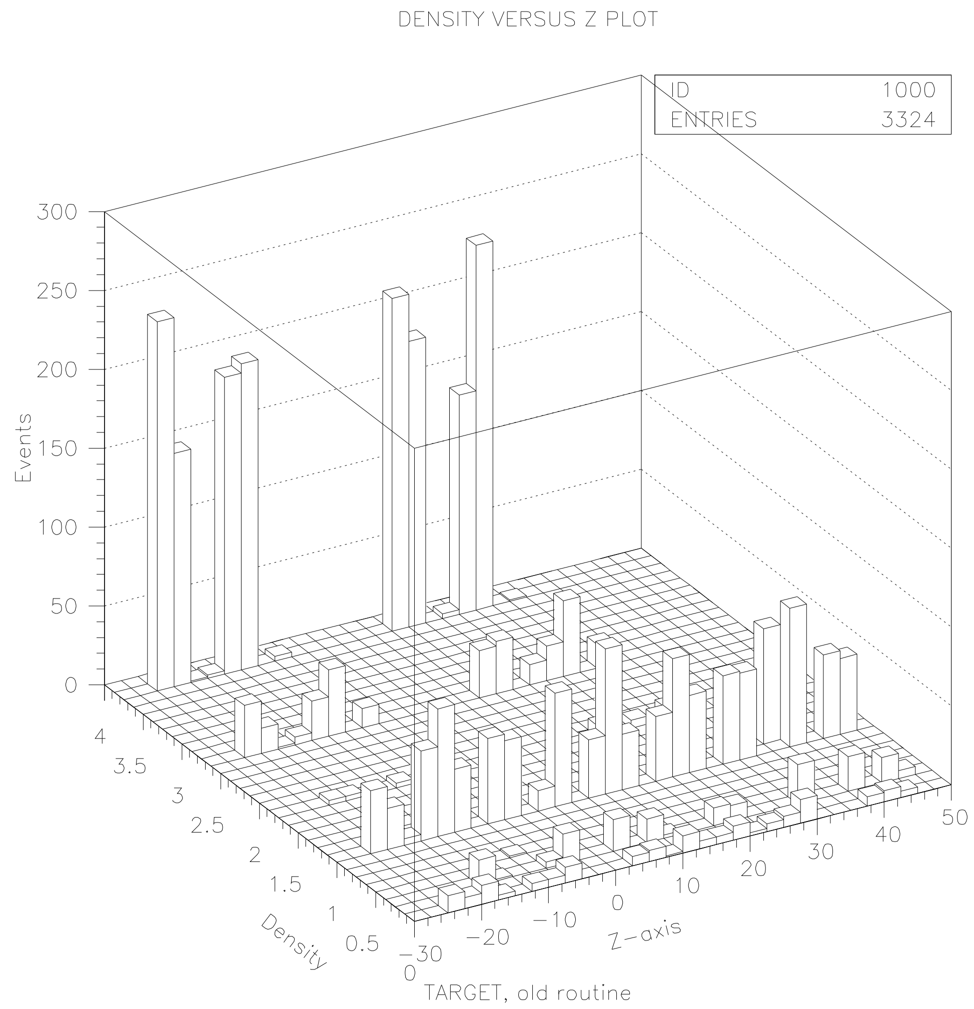
DENSITY VERSUS Z PLOT
ID 1000
ENTRIES 3324
Events
300
250
200
150
100
50
0
4
3.5
3
2.5
2
1.5
1
0.5
0
Density
−30
−20
−10
0
10
20
30
40
50
Z−axis
TARGET, old routine

Figure 3.3:

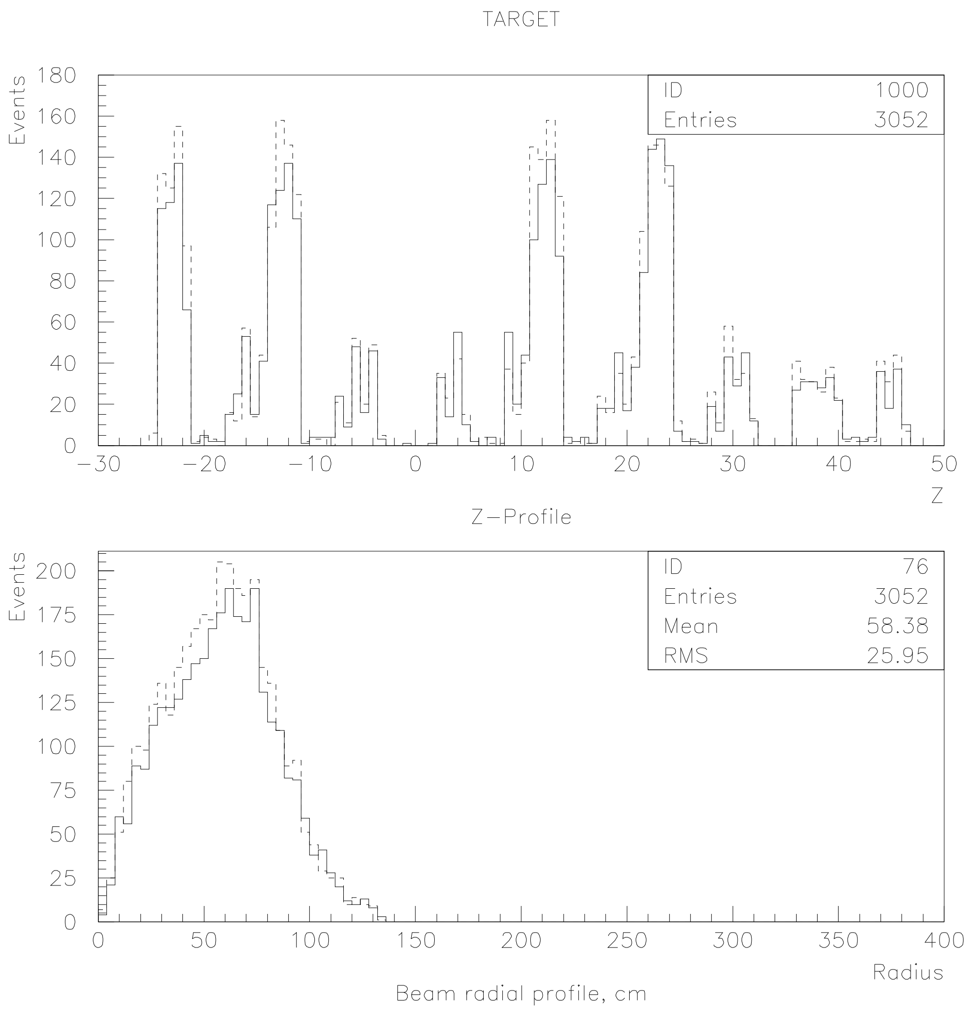

Figure 3.4: The dotted line represents the old routine.

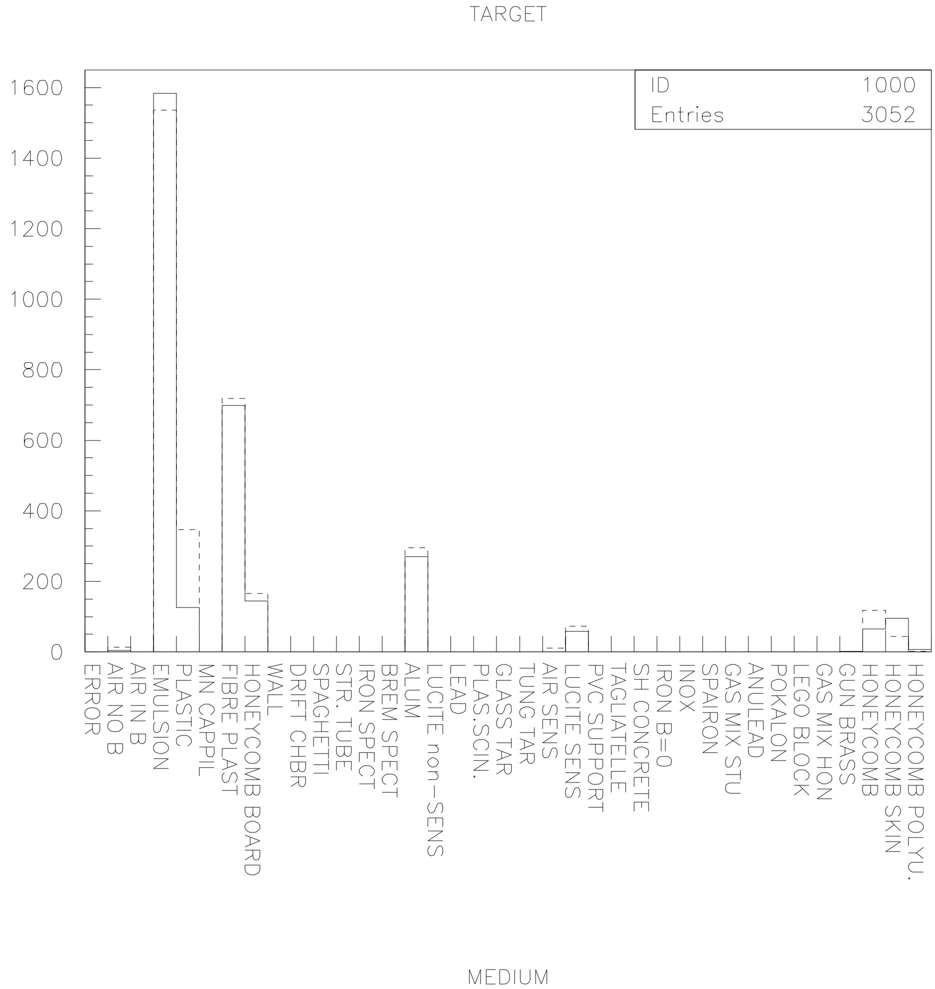

Figure 3.5: The dotted line represents the old routine.

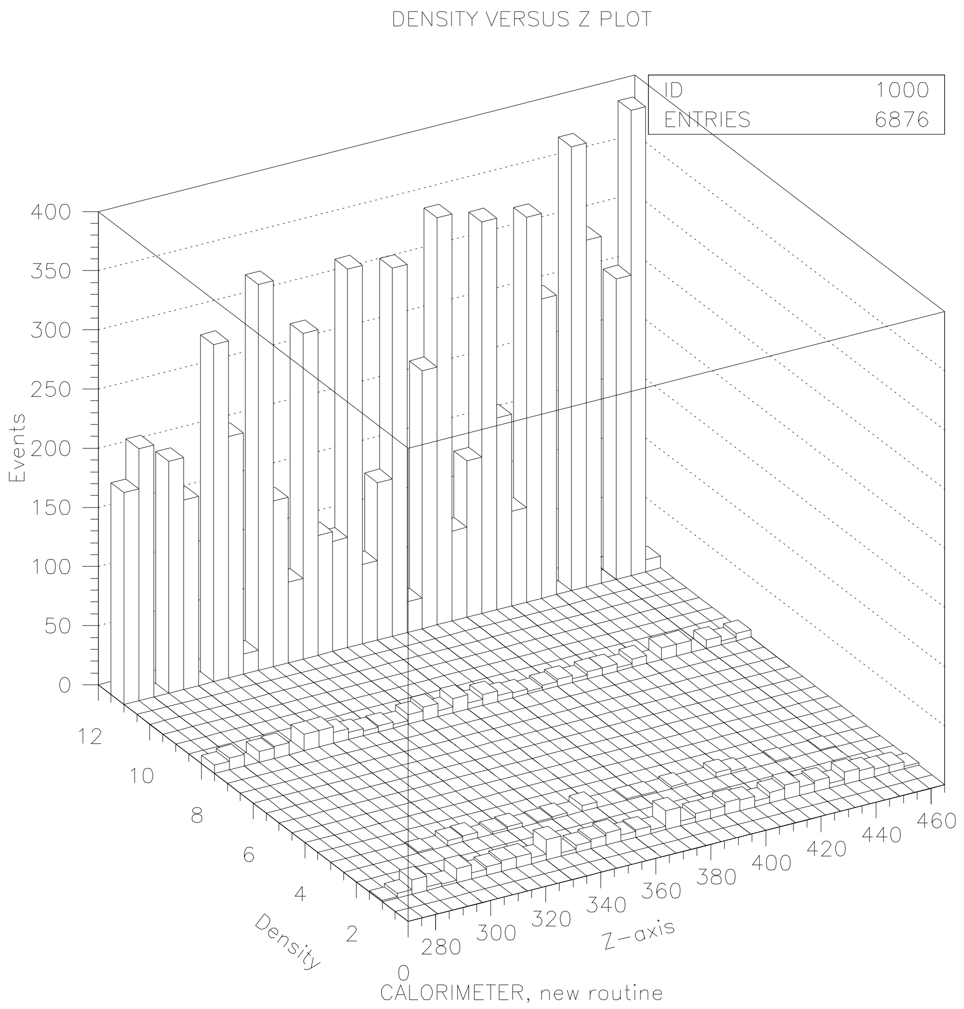
DENSITY VERSUS Z PLOT
ID 1000
ENTRIES 6876
Events
400
350
300
250
200
150
100
50
0
12
10
8
6
4
2
0
Density
280
300
320
340
360
380
400
420
440
460
Z-axis
CALORIMETER, new routine

Figure 3.6:

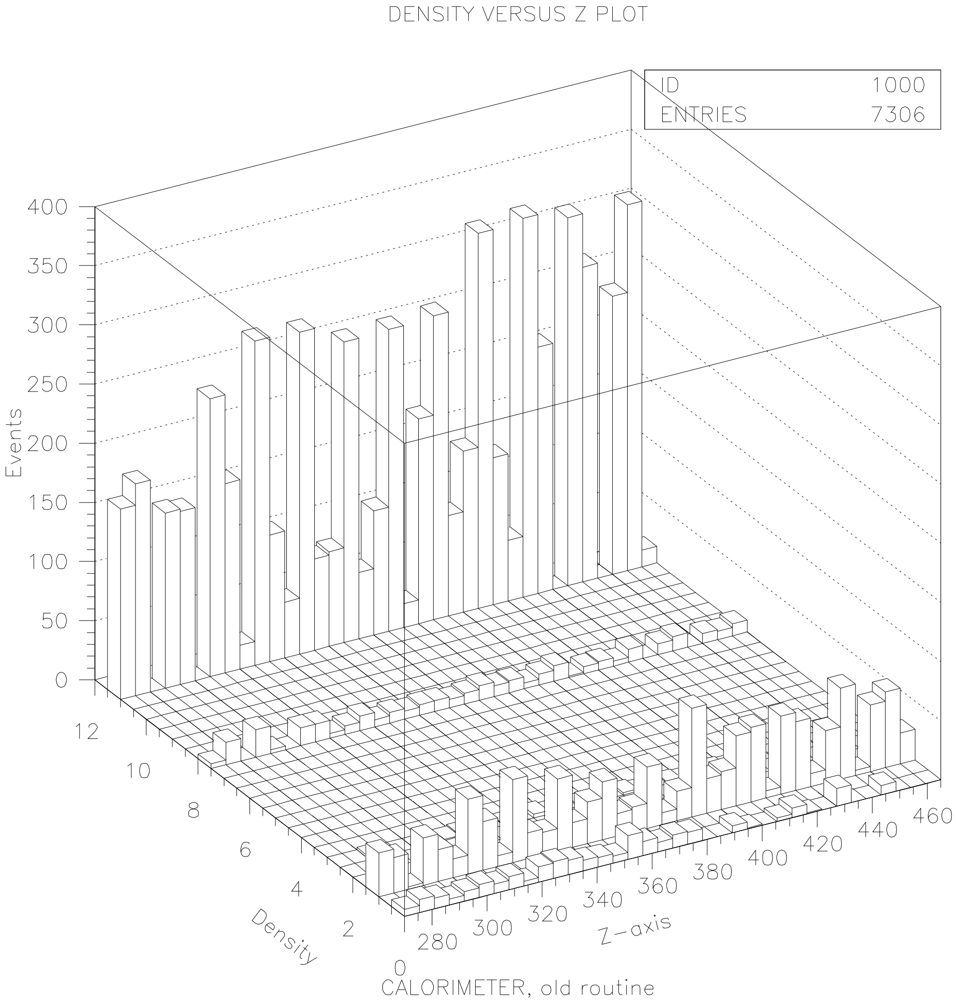
DENSITY VERSUS Z PLOT
ID 1000
ENTRIES 7306
Events
400
350
300
250
200
150
100
50
0
12
10
8
6
4
2
0
Density
280
300
320
340
360
380
400
420
440
460
Z-axis
CALORIMETER, old routine

Figure 3.7:

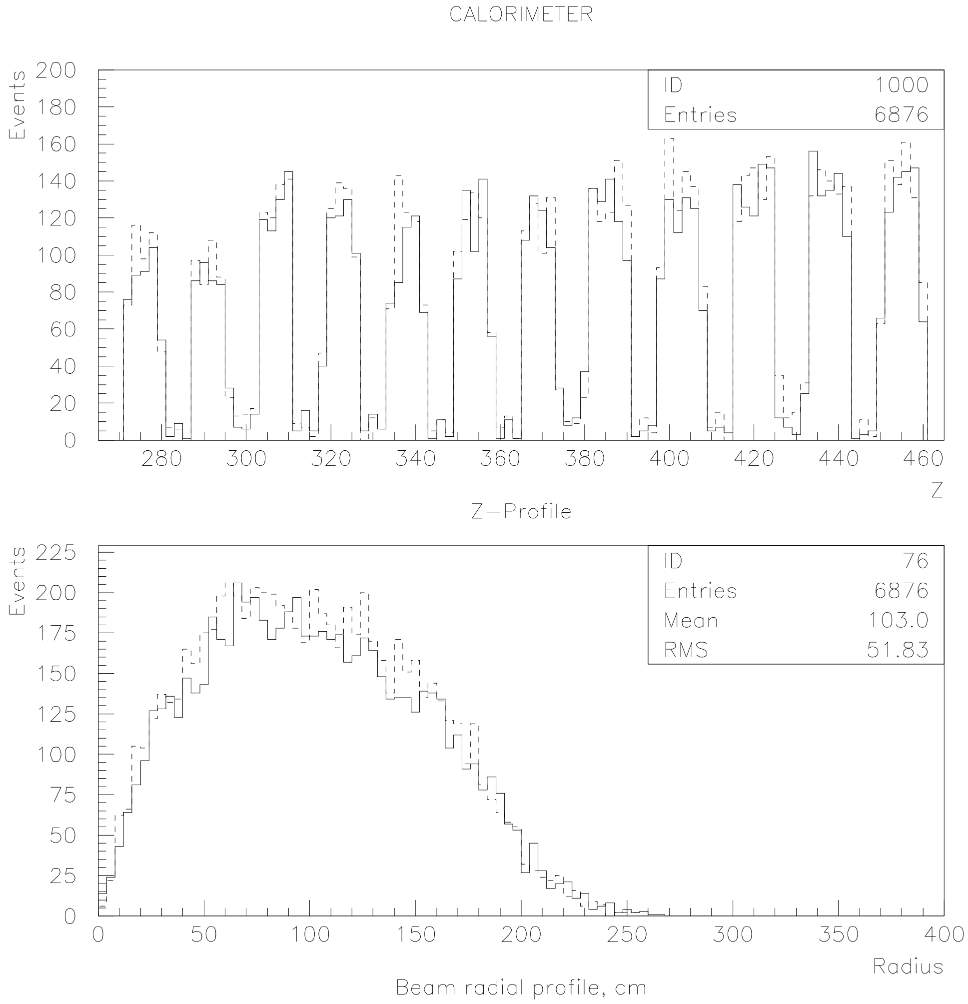

Figure 3.8: The dotted line represents the old routine.

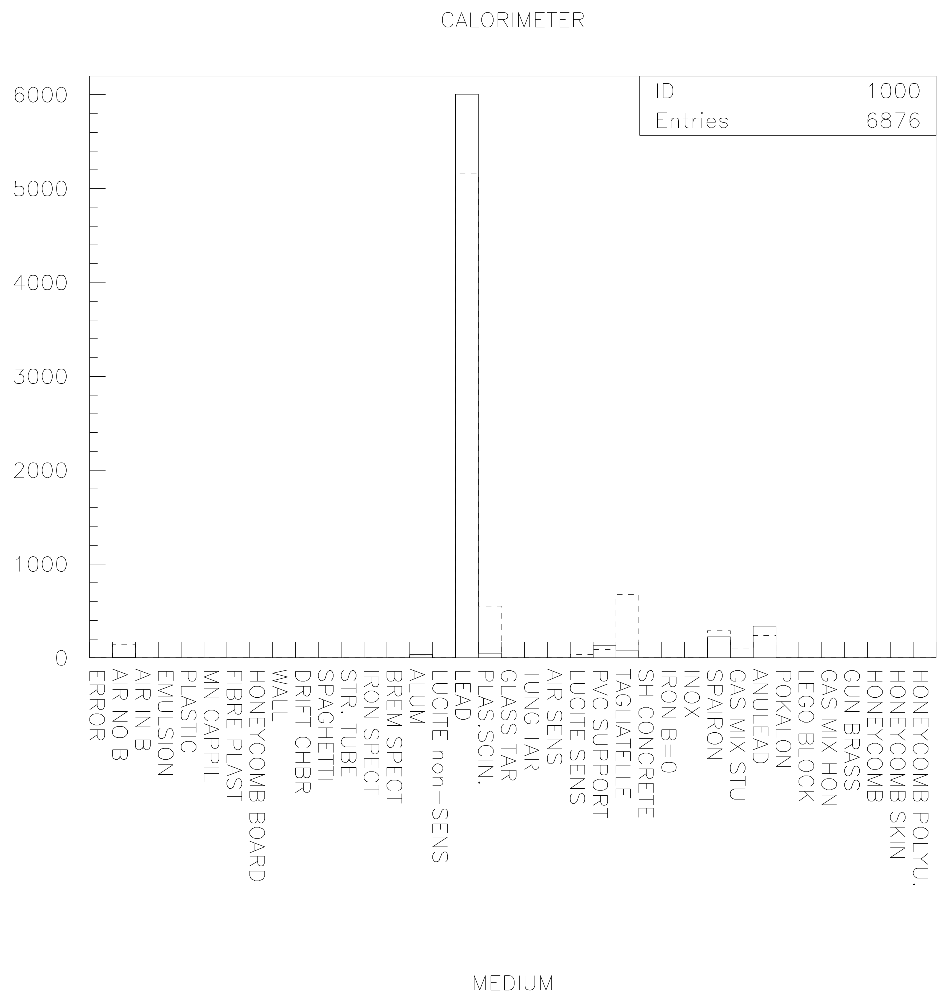

Figure 3.9: The dotted line represents the old routine.

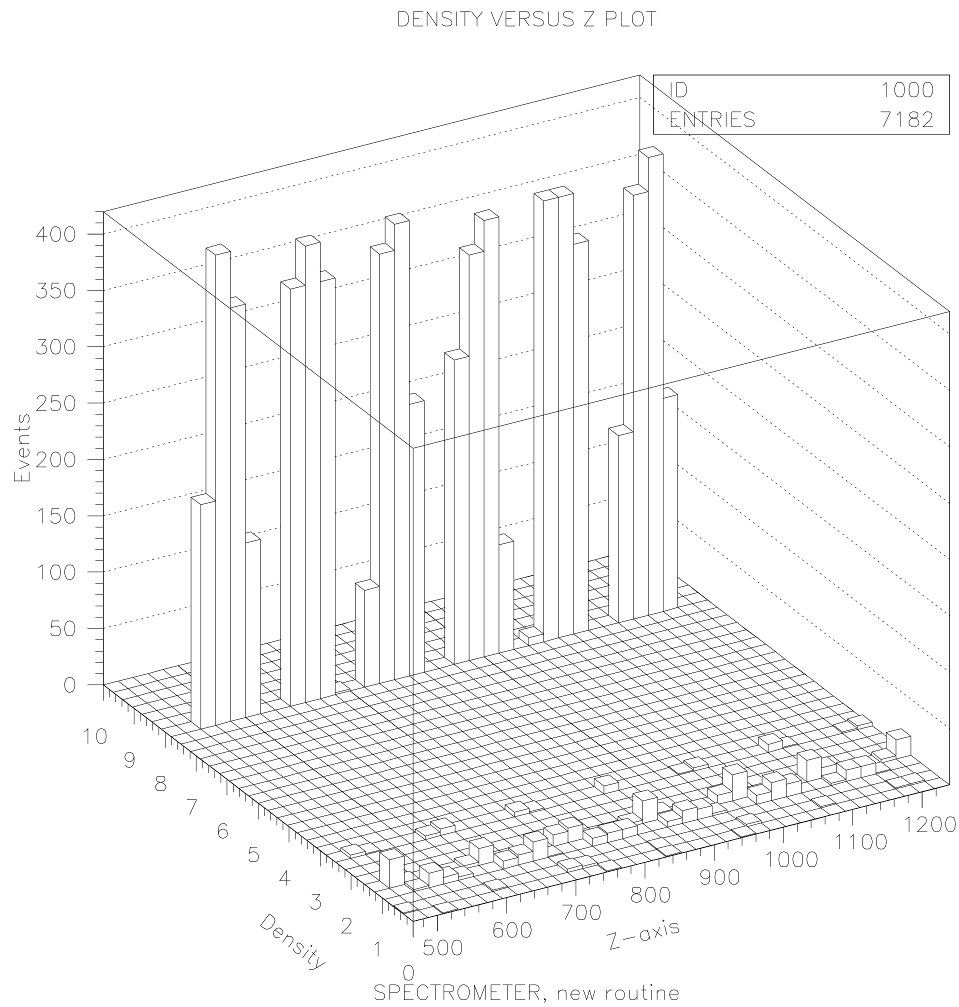
DENSITY VERSUS Z PLOT
ID 1000
ENTRIES 7182
Events
0
50
100
150
200
250
300
350
400
Density
0
1
2
3
4
5
6
7
8
9
10
Z-axis
500
600
700
800
900
1000
1100
1200
SPECTROMETER, new routine

Figure 3.10:

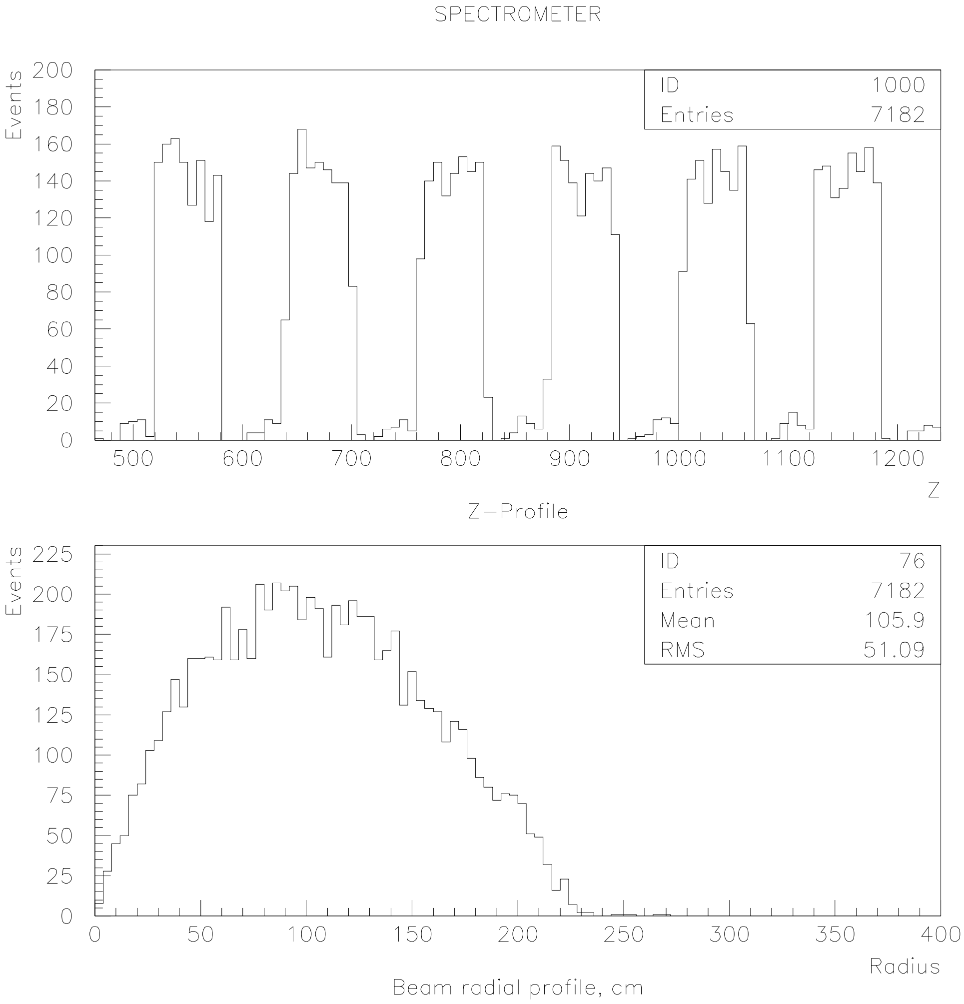
SPECTROMETER
Events
ID 1000
Entries 7182
0 20 40 60 80 100 120 140 160 180 200
500 600 700 800 900 1000 1100 1200
Z
Z−Profile
Events
ID 76
Entries 7182
Mean 105.9
RMS 51.09
0 25 50 75 100 125 150 175 200 225
0 50 100 150 200 250 300 350 400
Radius
Beam radial profile, cm

Figure 3.11:

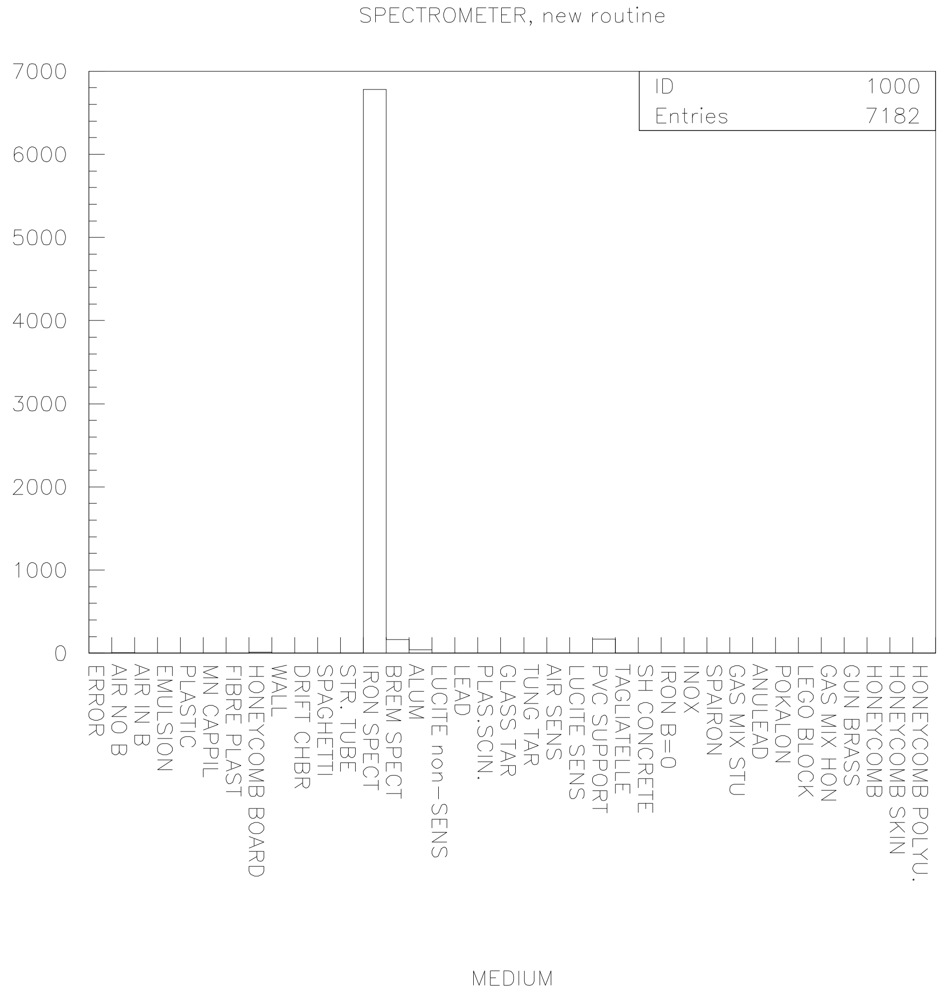
SPECTROMETER, new routine
ID 1000
Entries 7182
7000
6000
5000
4000
3000
2000
1000
0
ERROR
AIR NO B
AIR IN B
EMULSION
PLASTIC
MN CAPPIL
FIBRE PLAST
HONEYCOMB BOARD
WALL
DRIFT CHBR
SPAGHETTI
STR. TUBE
IRON SPECT
BREM SPECT
ALUM
LUCITE non-SENS
LEAD
PLAS.SCIN.
GLASS TAR
TUNG TAR
AIR SENS
LUCITE SENS
PVC SUPPORT
TAGLIATELLE
SH CONCRETE
IRON B=0
INOX
SPAIRON
GAS MIX STU
ANULEAD
POKALON
LEGO BLOCK
GAS MIX HON
GUN BRASS
HONEYCOMB
HONEYCOMB SKIN
HONEYCOMB POLYU.
MEDIUM

Figure 3.12:

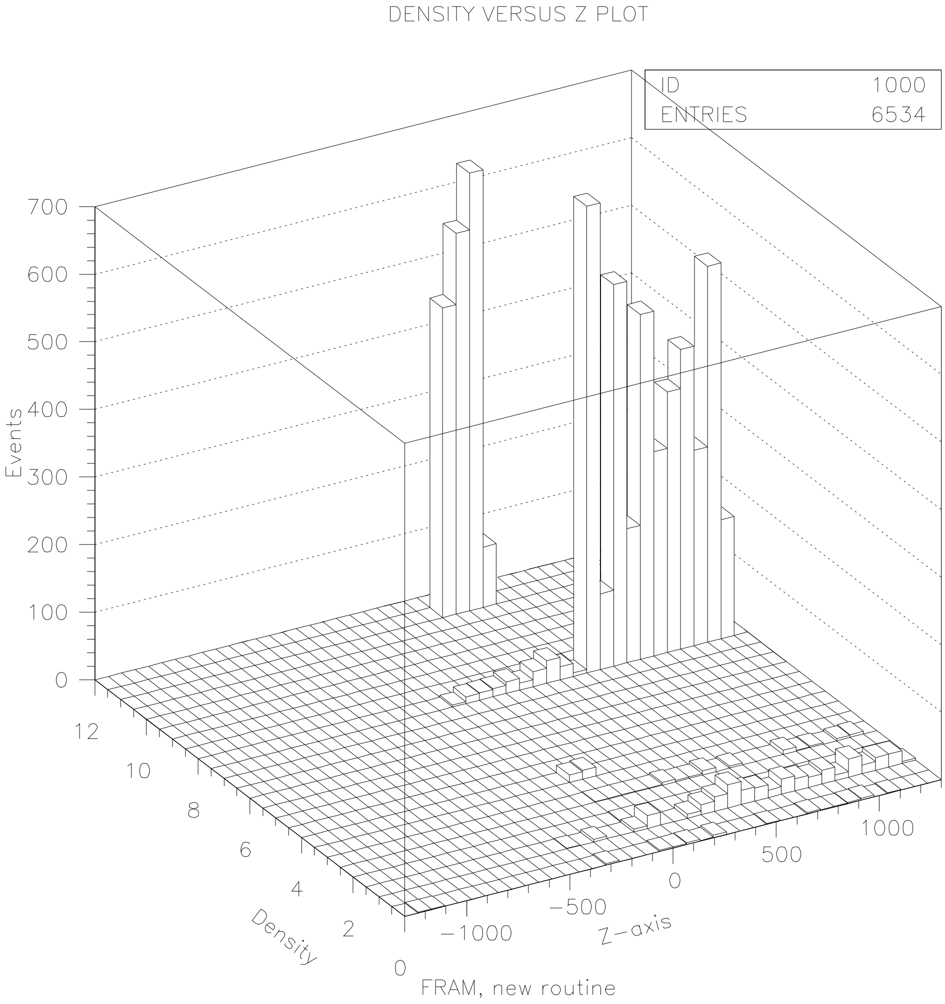
DENSITY VERSUS Z PLOT
ID 1000
ENTRIES 6534
Events
0
100
200
300
400
500
600
700
Density
0
2
4
6
8
10
12
Z-axis
−1000
−500
0
500
1000
FRAM, new routine

Figure 3.13:

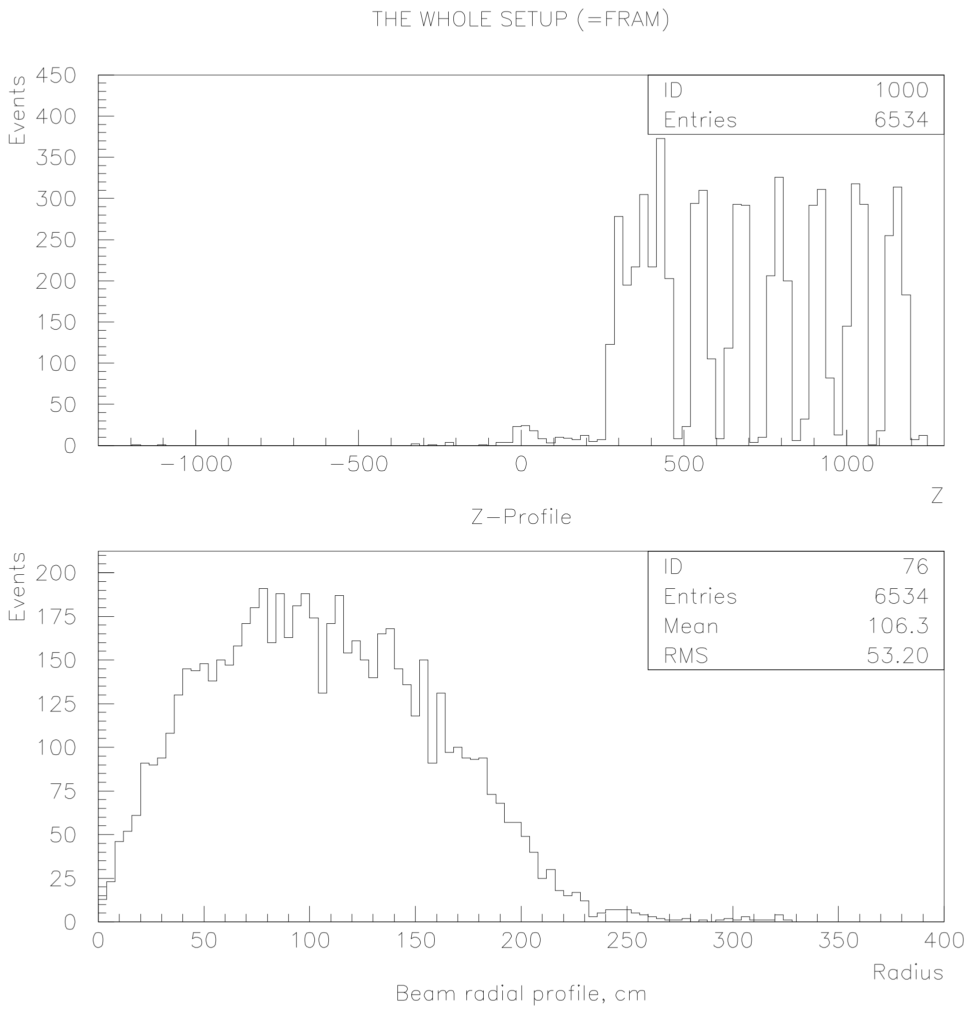
THE WHOLE SETUP (=FRAM)
Events
ID 1000
Entries 6534
450
400
350
300
250
200
150
100
50
0
−1000
−500
0
500
1000
Z
Z−Profile
Events
ID 76
Entries 6534
Mean 106.3
RMS 53.20
200
175
150
125
100
75
50
25
0
0
50
100
150
200
250
300
350
400
Radius
Beam radial profile, cm

Figure 3.14:

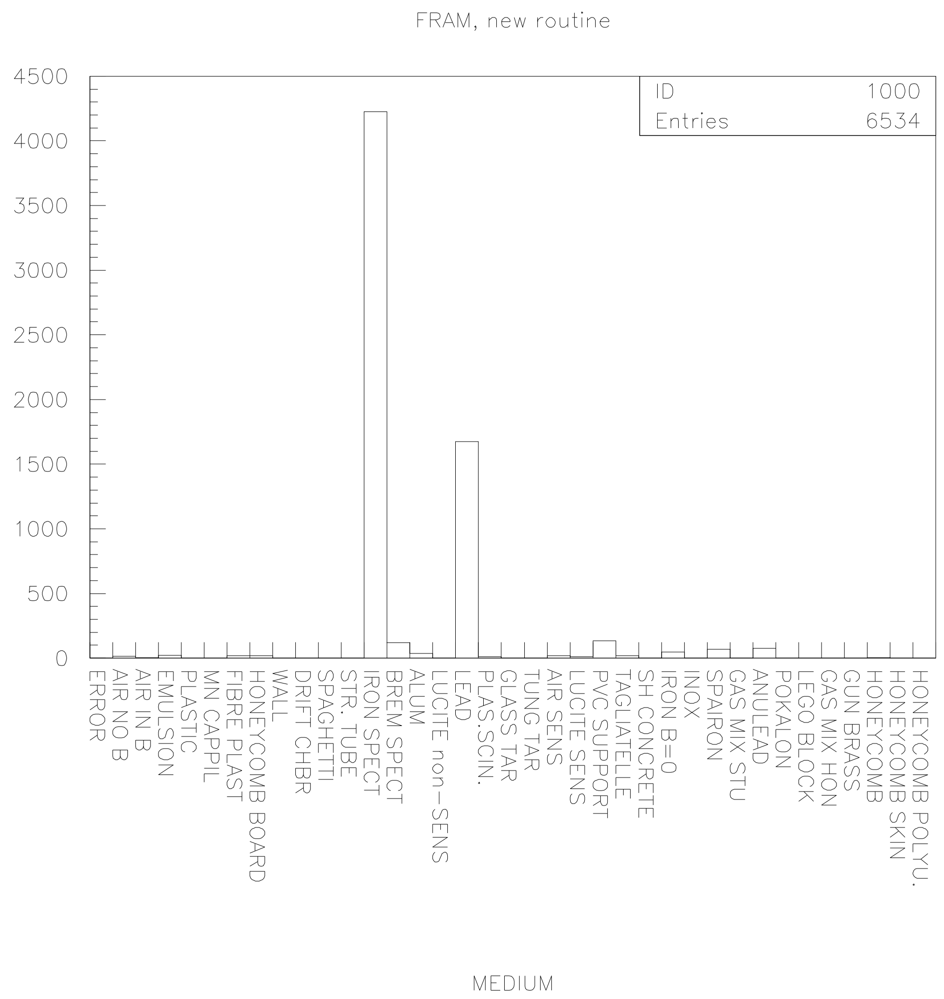
FRAM, new routine
ID 1000
Entries 6534
4500
4000
3500
3000
2500
2000
1500
1000
500
0
ERROR
AIR NO B
AIR IN B
EMULSION
PLASTIC
MN CAPPIL
FIBRE PLAST
HONEYCOMB BOARD
WALL
DRIFT CHBR
SPAGHETTI
STR. TUBE
IRON SPECT
BREM SPECT
ALUM
LUCITE non-SENS
LEAD
PLAS.SCIN.
GLASS TAR
TUNG TAR
AIR SENS
LUCITE SENS
PVC SUPPORT
TAGLIATELLE
SH CONCRETE
IRON B=0
INOX
SPAIRON
GAS MIX STU
ANULEAD
POKALON
LEGO BLOCK
GAS MIX HON
GUN BRASS
HONEYCOMB
HONEYCOMB SKIN
HONEYCOMB POLYU.
MEDIUM

Figure 3.15:

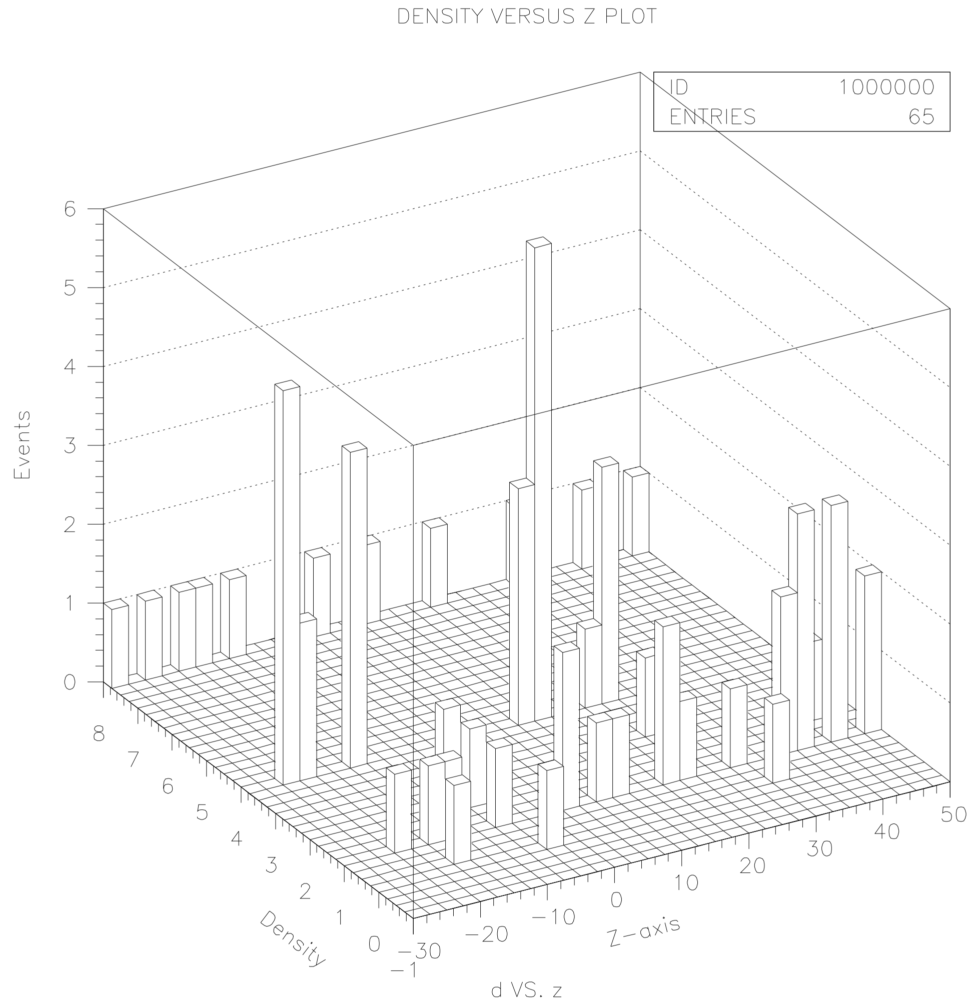

Figure 3.16: This graph shows a zoom of figure 3.13 into the target area.

In this zoom one can clearly see the peaks at the four emulsion stacks as expected (Remember, the emulsion has a density of $\approx 3g/m^3$). The only two major differences between this plot and figure 3.2 is that there is only a handful of vertices positioned in the stacks, since most of them are located in the much denser spectrometer and calorimeter. Moreover, one can see peaks at a density of $\approx 8g/m^3$ where the target is located. This is the lead of the cool-box surrounding the target.

Appendix A

Useful *.kumac file

This macro file was used to produce the vertex distribution in the various media of the setup. The histogram is based on the N-tuple HIST 74. The "error" bin indicates that the medium number is zero; i.e. that vertex is positioned outside the setup. This should not occur at all. If it does, start debugging ...

```
* TITLE= global title, MAX=length of y-axis (events),
* COLOR=1 (black) or 2(red)

MACRO MED TITLE MAX=0 COLOR=0
   SET YGTI 0.8
   COLOR 2 1. 0. 0.
   TITLE_GLOBAL [1]
   H/CR/1DH 1000 'MEDIUM' 37 -1. 36.5
   OPT STAT
   SET STAT 11
   LABEL 1 37 'ERROR' 'AIR NO B' 'AIR IN B' 'EMULSION'
              'PLASTIC' 'MN CAPPIL' 'FIBRE PLAST' 'HONEYCOMB
               BOARD' 'WALL' 'DRIFT CHBR' 'SPAGHETTI' 'STR.
               TUBE' 'IRON SPECT' 'BREM SPECT' 'ALUM' 'LUCITE
               non-SENS' 'LEAD' 'PLAS.SCIN.' 'GLASS TAR'
               'TUNG TAR' 'AIR SENS' 'LUCITE SENS' 'PVC
                SUPPORT' 'TAGLIATELLE' 'SH CONCRETE' 'IRON
                B=0' 'INOX' 'SPAIRON' 'GAS MIX STU' 'ANULEAD'
               'POKALON' 'LEGO BLOCK' 'GAS MIX HON' 'GUN BRASS'
               'HONEYCOMB' 'HONEYCOMB SKIN' 'HONEYCOMB POLYU.'
   SET NDVX -37.13
   SET YMGL 7.
```

```
  SET YHTI 6.
  NT/PRO 1000 74.M
  MAX 1000 [2]
  SET HCOL [3]
  H/PL 1000
  H/DEL 1000
  SET '*'
RETURN
```