Österreichische Akademie der Wissenschaften
INSTITUT FÜR HOCHENERGIEPHYSIK

HEPHY-PUB-863/08.
Version 2.0 – 5 June 2008

LiC Detector Toy 2.0

Vienna Fast Simulation Tool for Charged Tracks

User's Guide

Meinhard Regler, Manfred Valentan[1] and **Rudolf Frühwirth**

Institute of High Energy Physics, Austrian Academy of Sciences
Nikolsdorfer Gasse 18, A-1050 Vienna, Austria, EU

Abstract

This guide describes the "LiC Detector Toy" (LDT) software tool which has been developed for detector design studies, aiming at investigating the resolution of reconstructed track parameters for the purpose of comparing and optimizing detector set-ups. It consists of a simplified simulation of the detector measurements, based on a helix track model and taking into account multiple scattering, followed by full single track reconstruction using the Kalman filter. The software runs under MatLab and Octave, with an integrated GUI available under MatLab, and may be installed on a desktop or laptop PC. It can easily be used as a black-box tool by non-experts, but can also be adapted to meet individual needs.

[1]Contact: < valentan@hephy.oeaw.ac.at >

Contents

1 Introduction

This simple but powerful software tool for detector design, modification and upgrade studies aims at investigating the resolution of reconstructed track parameters at the inner side of the beam tube for comparing and optimizing the track sensitive devices and the material budgets of various detector set-ups. This is achieved by a simplified simulation of the set-up yielding the measured coordinates, followed by full single track reconstruction.

The detector model corresponds to a collider experiment with a solenoid magnet, and is rotational symmetric w.r.t. the beam axis; the surfaces are either cylinders (barrel region) or planes (forward and rear regions). The magnetic field is homogeneous and parallel to the beam axis, thus implying a helix track model (radius R_H); S-shaped tracks are not foreseen. All material causing multiple scattering is assumed to be concentrated within thin layers; it can either be averaged over a surface, or can be modeled individually.

The simulation generates a charged track from a primary vertex, performs helix tracking in a homogeneous magnetic field with breakpoints due to multiple scattering, and simulates detector measurements including systematic or stochastic inefficiencies and uniform or Gaussian observation errors, *but no other degradation of data.* Supported are *Si* strip detectors (single or double sided, with any stereo angle), pixel detectors, and a time projection chamber (TPC). The simulated measurements are then used to reconstruct the track by fitting its five parameters and calculating their corresponding 5×5 covariance matrix at a given reference cylinder, e.g. the inner surface of the beam tube (optionally being converted to a 6-dimensional Cartesian representation, see section 6). The definition of the track parameters can be found in the Appendix. The method used is a Kalman filter [1, 2] with linear approximation to the track model, the expansion point being defined by the simulated track parameters at the outer surface of the beam tube; process noise is calculated from the material budget as seen by the reference track. Sensitive tests of goodness of the fits (χ^2 distributions and pull quantities) are implemented. The tool may optionally generate vertices with any desired number of tracks, thus being able to deliver input for multiprong track bundling and secondary vertex separation. And it can supply input for pattern recognition studies, for the development of alignment strategies, or for trigger simulation studies.

The algorithms used in the tool are on a solid mathematical basis. The software was originally written in MatLab®, a high-level matrix algebra language and IDE [3]; usage is supported by a graphic user interface (GUI). It has furthermore been adapted to run command line based under Octave, where the GUI is not available [4]. The tool is deliberately kept simple and can without effort be adapted to meet individual needs. Its main purpose, however, is to supply a tool for non-experts, without any knowledge of algorithmic details. Once the detector description ("input sheet") – certainly the most demanding part – has been set up, individual detector layers can easily be added, removed or modified, the result of which can quickly be evaluated after a short running time.

The tool was released at the VCI 2007 [5], and has so far been used for track resolution studies of detectors at the ILC [6] and the BELLE detector at KEK [7].

2 Getting started

The “LDT” software runs under MatLab , furthermore it has been adopted to run under GNU Octave as well. Both are high level languages, which are at present supported on the following platforms: Linux 32/64, Solaris 64, MacOSX PPC/ICD and Windows 32/64. To run the program, MatLab version 7 including the graphics toolbox or GNU Octave version 3.01 is necessary; lower versions may work but have not been tested.

If your institution has no MatLab campus licence, contact “The MathWorks” at `http://www.mathworks.com/`.

GNU Octave is freely redistributable software and can be downloaded at [4] under the terms of the GNU General Public Licence.

Getting started in MatLab : Start MatLab and open the main file `LDT_Matlab.m`. The program is started by running this file in the MatLab interpreter, i.e. opening the file in the editor and clicking the run button, or simply by typing `LDT_Matlab` in the MatLab shell. Usually, MatLab 's current directory (the main search path) will not be the one where the LDT source files reside. Therefore, after starting the program, a window will pop up asking you to change the current directory. Select `Change` MatLab `current directory` and confirm. After the program has been started the main window appears (see figure 1).

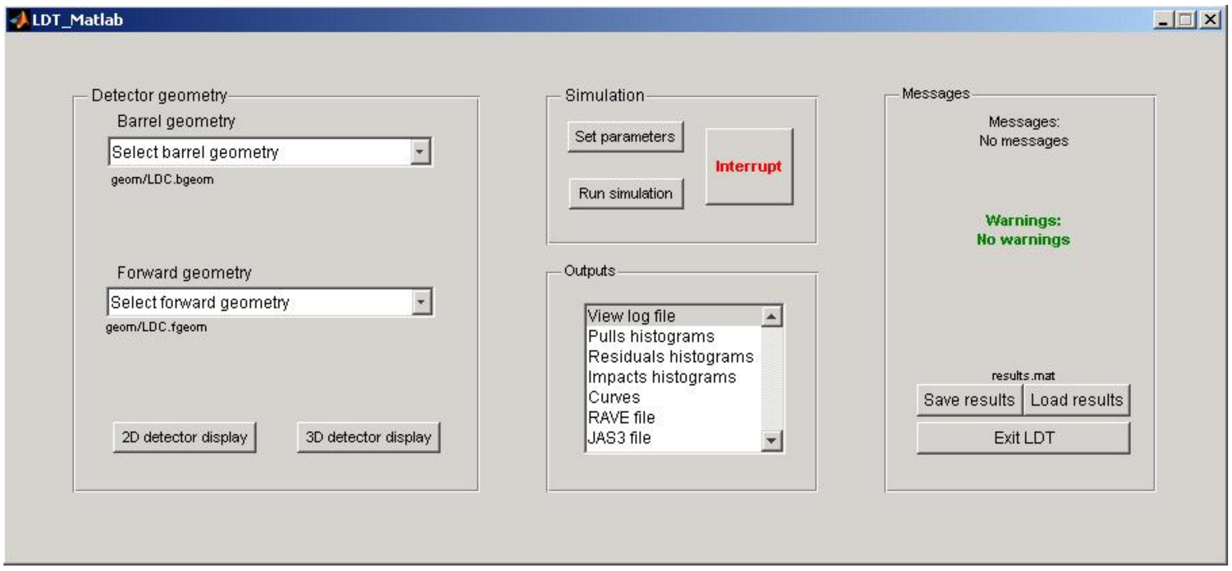
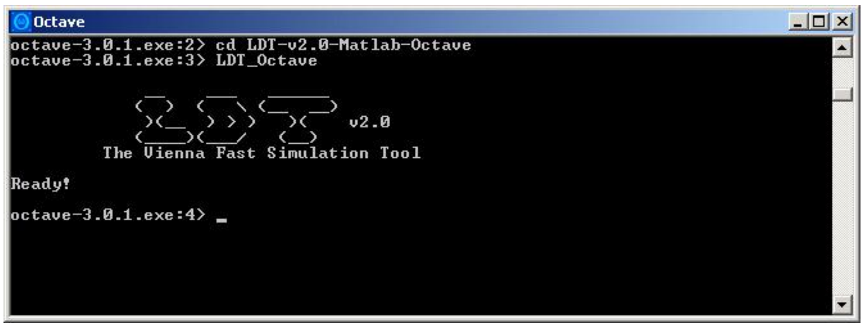


Figure 1: Main window

To end the program, click “Exit LDT” at the lower right corner of the main window.

Getting started in Octave : Start Octave and use the command `cd` (change directory) to navigate to the path where LDT's files are stored. Then type `LDT_Octave` to start the program. The welcome screen is shown (see figure 2) and the main steering file `simulation_parameters_Octave.txt` is opened in the default text editor (see figure 3).



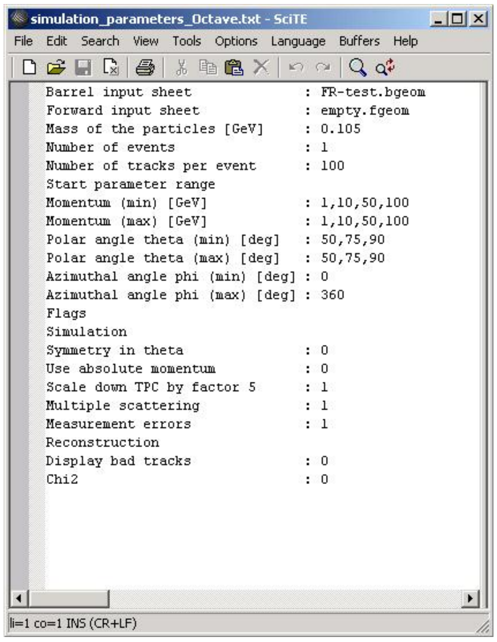
```
octave-3.0.1.exe:2> cd LDT-v2.0-Matlab-Octave
octave-3.0.1.exe:3> LDT_Octave

             __     ___     ______
            <  >   <   \ <__  __>
             ><__   > > >   ><     v2.0
            <____><___/    <__>
        The Vienna Fast Simulation Tool

Ready!

octave-3.0.1.exe:4> _
```

Figure 2: Octave welcome screen



```
Barrel input sheet                  : FR-test.bgeom
Forward input sheet                 : empty.fgeom
Mass of the particles [GeV]         : 0.105
Number of events                    : 1
Number of tracks per event          : 100
Start parameter range
Momentum (min) [GeV]                : 1,10,50,100
Momentum (max) [GeV]                : 1,10,50,100
Polar angle theta (min) [deg]       : 50,75,90
Polar angle theta (max) [deg]       : 50,75,90
Azimuthal angle phi (min) [deg] : 0
Azimuthal angle phi (max) [deg] : 360
Flags
Simulation
Symmetry in theta                   : 0
Use absolute momentum               : 0
Scale down TPC by factor 5          : 1
Multiple scattering                 : 1
Measurement errors                  : 1
Reconstruction
Display bad tracks                  : 0
Chi2                                : 0
```

Figure 3: Octave main steering file

3 Setting up the detector model

The so called “input sheets” contain all input data the program needs for building the detector model. They contain the geometry of the detector layers, their material budget, the accuracy and the kind of the error distribution of the measurements and their efficiency. In addition to that they contain information about passive scattering layers (e.g. the beam tube and support structures) and their material budgets. There are separate input sheets for the barrel and the forward region. They have to be stored in the directory `/geom` with the extension `.bgeom` and `.fgeom`, respectively. There is no difference between the input sheets used by the MatLab version and by the Octave version of LDT.

There are three predefined input sheets, that should not be changed, with the names: `empty.bgeom`, `empty.fgeom` and `beamtube-only.bgeom`. The input sheets `empty.bgeom` and `empty.fgeom` serve as master input sheets for setting up a new detector setup. They can be edited, but have to be stored under a different filename afterwards. The input sheet `beamtube-only.bgeom` is used for detector setups containing only forward layers. Since internally the innermost cylinder layer is used as reference surface for the track parameters, at least *one* cylinder layer has to be considered. This input sheet can be edited to change the radius of this reference layer. Detector setups containing only barrel layers can make use of the forward input sheet `empty.fgeom`.

Handling input sheets in MatLab : The input sheets can be selected using the main window of the program (see figure 1). On the left side of the main window one finds the section “Detector geometry”, divided into two parts, one for the barrel geometry and one for the forward geometry. To select an input sheet, click on the corresponding pull down menu and choose “Select barrel geometry” (“Select forward geometry”). A file dialog will appear, asking you to choose a barrel (forward) input sheet. The name of the selected input sheet will then be displayed beyond the pull down menu.

You can have the program draw a sketch of the selected geometry. By clicking the button “2D detector display”, MatLab opens a figure window showing a 2D sketch of the chosen arrangement, including (by default) the names of the layers and illustrating the chosen start parameter range in ϑ by red lines. By clicking the button “3D detector display” another figure window appears containing a 3D plot of the setup (see figures 4 and 5, respectively). Note, that usually not all layers of the TPC are shown, since there are too many.

To modify an existing geometry, select “Modify barrel (forward) geometry” in the corresponding pull down menu. A file dialog will appear and ask you to choose a barrel (forward) file. MatLab opens the input sheet in the default text editor. Now you can easily change some values of the setup, as will be shown later. Don’t forget to save your changes before running a simulation run!

To open a new input sheet, select “New barrel (forward) geometry” in the corresponding

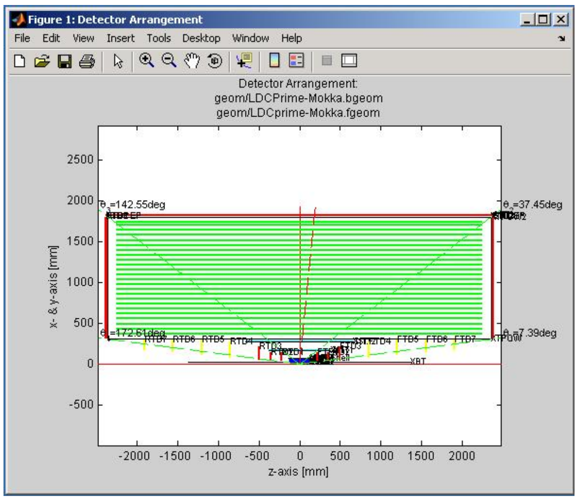


Figure 1: Detector Arrangement
File Edit View Insert Tools Desktop Window Help
Detector Arrangement:
geom/LDCPrime-Mokka.bgeom
geom/LDCprime-Mokka.fgeom
x- & y-axis [mm]
z-axis [mm]

Figure 4: Detector setup in 2D

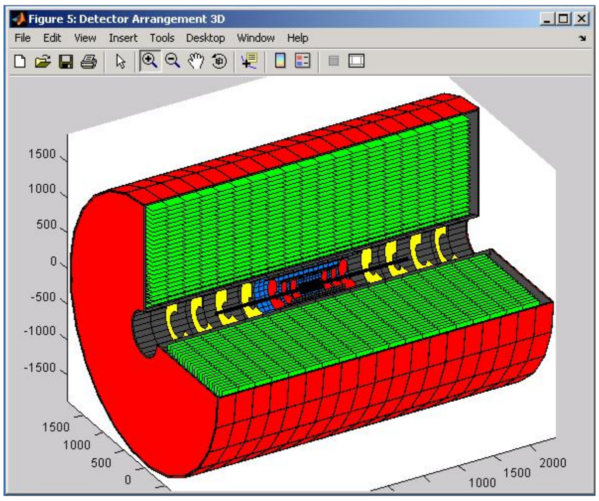


Figure 5: Detector Arrangement 3D
File Edit View Insert Tools Desktop Window Help

Figure 5: Detector setup in 3D

pull down menu. MatLab opens the predefined input sheet `empty.bgeom` (`empty.fgeom`) in the text editor. These input sheets only contain explaining text and may be filled in as desired. Please don't forget to save the file after editing it under a *different* filename!

The pull down menus contain two more items, "Add barrel (forward) geometry to list", and "Remove barrel (forward) geometry from list". They will be explained in section 7.2.

Handling input sheets in Octave : The input sheets can be selected using the main steering file of the program (see figure 3). Enter the filename of the desired barrel input sheet in the first line of the main steering file. The filename of the desired forward input sheet is entered in the second line, both behind the semicolon.

By typing `LDT_display2D` in the Octave command line, Octave opens a figure window showing a 2D sketch of the chosen arrangement, including (by default) the names of the layers and illustrating the chosen start parameter range in ϑ by red lines (see figure 4). The 3D sketch of the detector setup is not available in Octave . Note, that usually not all layers of the TPC are shown, since there are too many.

To modify an existing geometry, type `edit` followed by the name of the desired input sheet in the Octave shell, including the file's extension (`.bgeom` or `.fgeom`, respectively). Octave opens the input sheet in the default text editor. Now you can easily change some values of the setup, as will be shown later. Don't forget to save your changes before running a simulation run!

To open a new input sheet, type `edit empty.bgeom` (`edit empty.fgeom`) in the Octave shell. Octave opens the predefined input sheet `empty.bgeom` (`empty.fgeom`) in the text editor. These input sheets only contain explaining text and may be filled in as desired. Please don't forget to save the file after editing it under a *different* filename!

3.1 Input of barrel region, magnetic field and vertex position

As the entire detector is assumed to have cylindrical symmetry, the detector layers in the barrel region are cylinder surfaces, defined by their radii and their extensions in the z-direction. Each detector layer is a priori assumed to be a double-sided strip layer: the first coordinate measured is the azimuthal arc $R\Phi$, the second one is a helix with arbitrary stereo angle α; for $\alpha = \pi/2$ it measures z. Since inefficiencies are included, a single-sided layer can be modeled by giving one coordinate zero efficiency. Setting the efficiency of the second coordinate to -1 forces both coordinates to fire at the same time (strict correlation). Pixel layers can be modeled by giving the coordinates the corresponding pixel distances, the stereo angle $\alpha = \pi/2$, and strict correlation of the efficiency.

A passive layer is modeled by assigning one coordinate the efficiency zero and by requiring the efficiency of the second coordinate to be strictly correlated. Such a layer doesn't yield measurements at all. At least one passive layer should be included, usually the beam tube. Note that the innermost cylinder layer is used as a reference surface, so that after the reconstruction the fitted parameters are defined on this surface. Also the residuals of the fitted track parameters w.r.t. to the true ones are computed on this surface. This innermost cylinder layer is always assumed to be a passive one. Of course one can define a "virtual" layer (passive, zero thickness) as innermost layer to have the parameters be defined on this surface rather than on the beam tube (for example as virtual measurements for a subsequent vertex fit).

The barrel input sheet is divided into four input sections called Vertex Detector (VTX), Silicon Inner Tracker (SIT), Time Projection Chamber (TPC), and Silicon External Tracker (SET). Note that the layers in the TPC input section have special features. In fact the entire detector can be built using only a single section, e.g the VTX section. The different input sections are prepared because usually it is more convenient to group certain detector layers with similar properties.

In each one of the input sections one can define any number of detector layers and passive layers together with their corresponding geometry and further properties. The input sections VTX, SIT and SET offer the same features (see example below), the only difference being the colour in the detector display.

The input section TPC can be used to define layers with special behaviour. The resolution of TPC measurements may depend on z in order to take into account diffusion effects in the gas:

$$\sigma_{R\Phi} = \sqrt{\sigma^2_{R\Phi,0} + \sigma^2_{R\Phi,1} \cdot \sin(\beta) + C^2_{R\Phi} \cdot \sin(\vartheta) \cdot \frac{6mm}{h[mm]} \cdot \Delta z[m]},$$

$$\sigma_z = \sqrt{\sigma_{z,0}^2 + C_z^2 \cdot \Delta z[m]},$$

where Δz is the drift distance in [m], $\sigma_{R\Phi,0}$ is the lateral resolution at the endplate in [μm], $\sigma_{z,0}$ is the longitudinal resolution in [μm] due to drift time, $C_{R\Phi}$ and C_z are respectively the lateral and longitudinal charge spreadings due to diffusion in [μm/$\sqrt{m}$]. $\sigma_{R\Phi,1}$ takes into account the change of resolution in dependence on the angle β, and $h = \frac{R_{TPC,max} - R_{TPC,min}}{n_{padrows}}$ is the padrow pitch.
Note that this does not imply a particular technology as the diffusion can easily be suppressed by setting $C_{R\Phi}$ and C_z to zero. Currently only Gaussian errors are accepted in the TPC input section.

In the following subsection a part of an example barrel input sheet is shown and explained.

3.2 Example barrel input sheet: Silicon Inner Tracker

The following input lines define two asymmetric passive layers forming support frames, two detector layers (double sided strip layers) and the inner wall of the TPC (passive layer). In the zoomed display (figure 6) passive layers are displayed in black and active layers of the SIT input section are light blue. The dark blue layers are defined in the VTX section and the green layers are defined in the TPC section (not all layers of the TPC section are shown in the zoom).

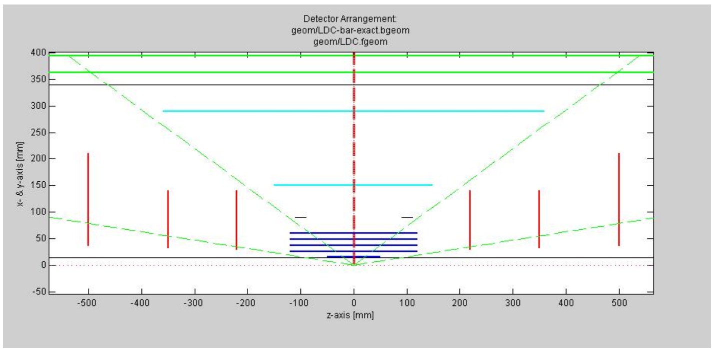


Figure 6: Detector display (zoomed).

```
22 Silicon Inner Tracker (SIT)
23
24 Number of layers:             5
25 Description (optional):      |support rings|-inner tracker-|TPC inner wall|
26 Names of the layers (opt.):   XSR1,  XSR2,  SIT1,    SIT2,    XTPCW
27 Radii [mm]:                   90,    90,    160,     290,     340
28 Upper limit in z [mm]:        110,  -90,    300,     550,     2160
29 Lower limit in z [mm]:        90,   -110,  -300,    -550,    -2160
30 Efficiency RPhi:              0,     0,     0.95,    0.95,    0
31 Efficiency 2nd layer (eg. z):-1,    -1,     0.95,    0.95,   -1
32 Stereo angle alpha [Rad]:     10*pi/180
33 Thickness [rad. lengths]:     0.07,  0.07,  0.0175,  0.0175,  0.14
34 error distribution:           1
35 0 normal-sigma(RPhi) [1e-6m]:
36          sigma(z)    [1e-6m]:
37 1 uniform-d(RPhi) [1e-6m]:    50
38           d(z)    [1e-6m]:    50
```

Line 24: Five layers are defined in this input section.

Line 25: This line is reserved for an optional verbal description of the layers.

Line 26: Here one can define specific names for the layers to identify them in the program. Can be left blank, if done so, the program uses default names, in this case SIT-1 to SIT-5.

Line 27: The radii of the layers. Radii can be equal. If there are only two values, the space between is filled with a number of equidistant layers equal to the number of layers put in above. The radii do not have to be sorted. Internally the program sorts the radii ancending and associates the rest of the values (such as efficiencies, strip distances...) in the same way.

Line 28: Upper limit of the layers in z-direction. A single value is valid for each layer. If there are only two values, the lengths of the layers are calculated such that the ends of the layers all are on the surface of the cone defined by the lower and upper radius and length.

Line 29: Lower limit in z-direction. Same features as before. Layers can be asymmetric with respect to $z = 0$. The first two layers of this example are asymmetric ones.

Line 30: Efficiency of the first coordinate measured, $R\Phi$. Set to zero to model a single or a passive layer.

Line 31: Efficiency of the second coordinate. A value of -1 means that the two coordinates are either both present or both absent (strict correlation). This feature is used to model pixels and passive layers. The first, the second and the last layer of this example are passive ones (two support frames, and the TPC inner wall).

Line 32: The stereo angle is defined as the angle between the strips measuring $R\Phi$ and those measuring the second coordinate. Each (active) layer can have its own angle; in this example the value is valid for all active layers.

Line 33: The thickness in radiation length. A single value is valid for all layers.

Line 34: 0=Gaussian, 1=uniformly (digitally) distributed measurement errors. Each (active) layer can have its own distribution; in this example the value is valid for all active layers.

Line 35,36: Standard deviation in the case of Gaussian distributed measurement errors. If there are no layers with Gaussian errors, this line can be left blank. Each (active) layer can have its own value, or one value can be defined which is valid for all corresponding layers.

Line 37,38: Strip distances (or pad dimensions) in the case of uniform measurement errors. Same features as for the standard deviations.

3.2.1 Magnetic field and beam spot

At the end of the barrel input sheet there are four more input lines. Here one defines the solenoidal magnetic field in [T] and the vertex position in [mm]. For every event a vertex is generated uniformly in the ranges entered in the three lower lines. All tracks of one event start at the same vertex.

```
72 Magnetic field and beam spot
73
74 Solenoid magnetic field [T]: 5
75 Range in x [mm]:            -0.0021   0.0021
76 Range in y [mm]:            -1e-5     1e-5
77 Range in z [mm]:            -1.04     1.04
```

3.3 Input of forward region

The forward and rear regions are both treated as “forward” due to reasons of similarity. The detector layers are planar wheels defined by their z-position and their inner and outer radius. Again each detector layer is a priori assumed to be a double strip layer measuring two coordinates u and v. These coordinates are defined by two angles δ_1 and δ_2 dependent on the intersection point. Keep in mind that the strip directions are perpendicular to the directions of the coordinates (see figure 7). Passive and single layers can be modeled using the inefficiency. Pixel detectors are built the same way as before.

The forward input sheet is divided into four input sections, two for the forward and two for the rear region. They are called Forward Module 1 (FM1), Forward Module 2

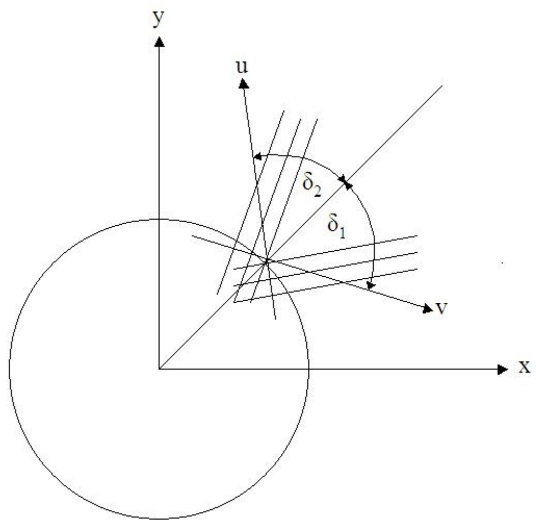


Figure 7: Definition of forward coordinates u and v.

(FM2), Rear Module 1 (RM1) and Rear Module 2 (RM2). In each one of these input sections one can define any number of detector layers and passive layers, together with their corresponding geometry and further properties. All input sections of the forward and rear region offer the same features (see example below), the only difference being the colour in the detector display. In principle the entire forward and rear region of the detector can be put into a single input section. But usually it's more convenient to divide the setup into groups of similar detector layers. In the following section a part of an example forward input sheet is shown and explained.

3.4 Example forward input sheet

The following input lines define three detector layers (pixel layers). In the detector display (figure 6) these layers are displayed in red on the right hand side. The yellow layers are defined in the FM2 input section (they can't be seen as the picture is zoomed). On the left hand side one can see the same configuration for the rear region, defined in RM1 and RM2. In this example the forward and rear region are symmetric, but they may be asymmetric.

```
04 Forward module 1
05
06 Number of layers:             3
07 Description (optional):     |forward pixel discs|
08 Names of the layers (opt.):  FPD1, FPD2, FPD3
09 z positions [mm]:            180,  300,  450
10 Inner radius [mm]:           40,   47.5, 57.5
11 Outer radius [mm]:           138,  140,  280
12 Efficiency u:                0.95
13 Efficiency v:               -1
14 Angle 1st coord. (u) [Rad]:  0
15 Angle 2nd coord. (v) [Rad]:  pi/2
16 Thickness [rad. lengths]:    0.01
17 error distribution:          1
18 0 normal-sigma(u) [1e-6m]:
19          sigma(v) [1e-6m]:
20 1 uniform-d(u) [1e-6m]:      50
21           d(v) [1e-6m]:      300
```

Line 06:	Three layers are defined in this input section.
Line 07,08:	Descriptions and specific names for the layers.
Line 09–11:	These lines contain the geometry of the layers. They offer the same features as the corresponding lines of the barrel input sheet.
Line 12,13:	Efficiencies, also used for modeling passive, single and pixel layers.
Line 14,15:	The angles defining the strip directions. In this example the directions are perpendicular to each other, together with the strictly correlated efficiency this models pixel discs.
Line 16:	The thickness in radiation lengths.
Line 17:	0=Gaussian, 1=uniformly (digitally) distributed measurement errors.
Line 18,19:	Standard deviation in the case of Gaussian errors. In this example, no layers have this property, so the lines are left blank.
Line 20,21:	Strip distances (or pad dimensions) in the case of uniform errors. The values here are valid for all layers.

The input sections Rear Module 1 (RM1) and Rear Module 2 (RM2) offer a special feature. If you enter the value -1 in the line 'Number of layers', the program internally copies the values of FM1 to RM1 and those of FM2 to RM2 while substituting the z positions by their negative value. When setting up a symmetric detector w.r.t. the plane $z = 0$ (in the forward/rear region), this feature can be used to skip filling in the input sections of the rear region.

4 Input of simulation parameters

Input of simulation parameters in MatLab : By selecting "Set parameters" in the main window's section "Simulation" the window "Edit simulation parameters" is opened (see figure 9). Here one puts in some general parameters, the start parameters, and sets the flags to steer the simulation.

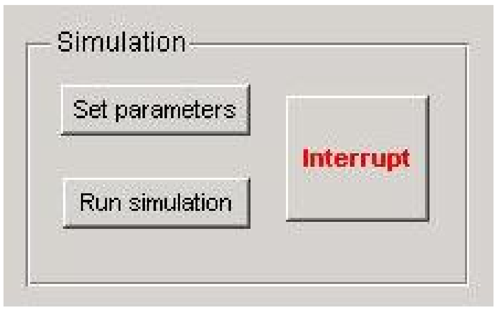


Figure 8: The section "Simulation" in the main window

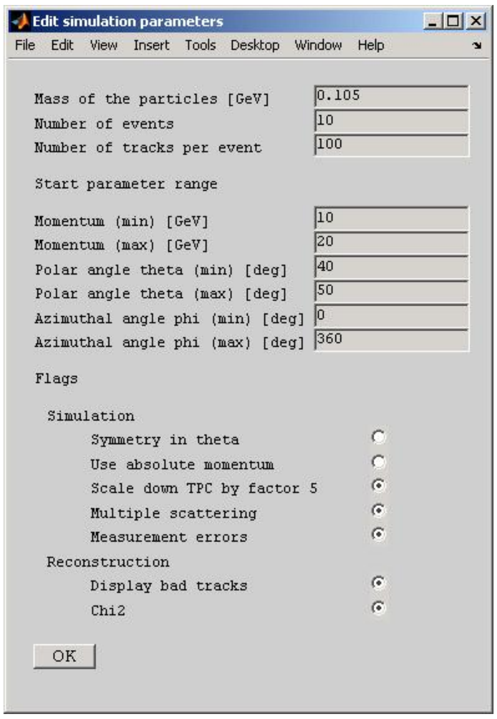
Input of simulation parameters in Octave : The Octave version of LDT reads the simulation parameters from the main steering file `simulation_parameters_Octave.txt` (see figure 3), which is opened automatically when starting LDT in Octave . Except for the first two lines, where the chosen input sheets are entered, it contains the same simulation parameters and flags as the window 'Edit simulation parameters' of the MatLab version. In fact, the MatLab version writes the simulation parameters entered in this very window to a very similar text file, which is later read out when running a simulation.

4.1 General parameters:

- Mass of the particles: The unit is $[GeV/c^2]$. If zero mass is entered, all particles are treated relativistically in terms of multiple scattering ($\beta P \approx P$).
- Number of events and tracks: The number of events to be simulated can be entered, as well as the the number of tracks per event. Currently each event contains the same number of tracks.

4.2 Start parameters

In the input section "Start parameter range" one can define ranges for the track's momentum and the track's direction. These start parameters do not obey any physics, the tracks of one event do not have any relation among each other, exept that they start at the same vertex. Here you only define the phase space of possible start parameters.


Edit simulation parameters
File Edit View Insert Tools Desktop Window Help
OK

Mass of the particles [GeV]	0.105
Number of events	10
Number of tracks per event	100
Start parameter range	
Momentum (min) [GeV]	10
Momentum (max) [GeV]	20
Polar angle theta (min) [deg]	40
Polar angle theta (max) [deg]	50
Azimuthal angle phi (min) [deg]	0
Azimuthal angle phi (max) [deg]	360
Flags	
Simulation	
Symmetry in theta	○
Use absolute momentum	○
Scale down TPC by factor 5	●
Multiple scattering	●
Measurement errors	●
Reconstruction	
Display bad tracks	●
Chi2	●

Figure 9: Window 'Edit simulation parameters'

- Momentum: Here you can put in the minimum and maximum values of the track's momentum in [GeV/c].
- Polar angle theta: Put in here the minimum and maximum values of the polar direction angle ϑ in [°]. Values from 0 to 180 are accepted. But make sure that the angle is not too small to hit any detector layer!
  Attention: ϑ should not be mixed up with the dip angle $\lambda = 90° - \vartheta$!
- Azimuthal angle phi: Here one defines the minimum and maximum values of the azimuthal direction angle φ in [°]. Since up to now the whole detector is assumed to have rotational symmetry, changes of this parameter won't affect the results.

For every track, the momentum and the azimuth angle φ are generated randomly from a uniform distribution between the given limits. In contrast, the polar angle ϑ is chosen randomly from a uniform distribution of $\cos(\vartheta)$ between the limits $\cos(\vartheta_{min})$ and $\cos(\vartheta_{max})$. If the minimum and maximum value of a range coincide, the corresponding starting value is fixed.
Remark: The momentum is either the transverse, or the absolute momentum, depending on the flag "Use absolute momentum" (see below).

4.3 Control flags

The flags are used for steering the program and selecting simulation and reconstruction features. There are two groups of flags, one group of flags each to influence the simulation and the reconstruction. In the OCTAVE version a flag is enabled if the value behind the semicolon is 1, and disabled, if the value is 0.

- Symmetry in theta: If this flag is set, the ϑ-range of the track is mirrored at the plane $z = 0$. Both ϑ-limits must be larger or smaller than $\pi/2$, respectively, that means one obtains two slices of ϑ-ranges. The number of tracks is uniformly distributed in both slices.
- Use absolute momentum: If "Use absolute momentum" is set the momentum range mentioned above is treated as range for the absolute momentum. Otherwise the values are the limits for the transverse momentum.
- Scale down TPC by factor 5: To increase the simulation speed, one can scale down the TPC using the flag "Scale down TPC by factor 5". The number of layers in the TPC is devided by 5, the sigmas are multiplied by $\sqrt{5}$, and the radiation length of each remaining layer is multiplied by 5. This corresponds to replacing five detector layers by one layer with statistically corresponding properties.
- Multiple scattering; Measurement errors: Multiple scattering and measurement errors can be switched off using the corresponding flags, mainly for debugging purposes.
- Display bad tracks: The flag "Display bad tracks" forces the program to write information about badly reconstructed tracks to the log file.
- Chi2: The computation of the test quantity χ^2 can be disabled separately using the flag "Chi2" since it is a rather time consuming part of the fit procedure. If enabled, the average number of degrees of freedom, the mean value of the χ^2 and the ratio of both numbers are written into the log file.

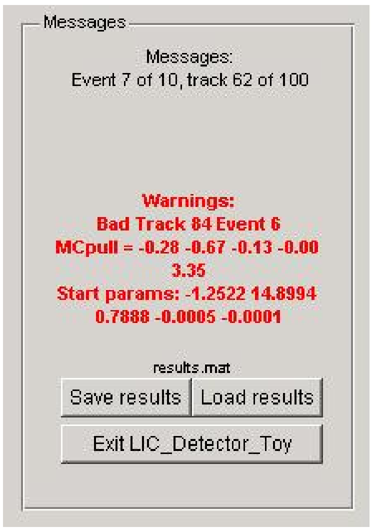
5 Running a simulation

Having set up the detector arrangement and defined the start parameters, one can run the simulation in the MatLab version by clicking the button "Run simulation" in the main window's simulation section (see figure 8). In the Octave version, a simulation run is started by typing `LDT_run` in the Octave shell.
Then, the following steps are carried out:

- The selected input sheets are read out and the internal detector model is set up.
- The start parameters and flags are read.
- Event loop
  - The vertex is chosen randomly within the given limits.
  - Track loop
    - The start parameters are chosen randomly within the given limits.
    - The simulation of the track going through the detector is carried out, yielding the measured space points.
    - The reference track is computed. This track is used as expansion point for the Kalman filter.
    - The track is reconstructed, yielding the fitted track parameters at the inner side of the innermost cylinder layer.
    - Test quantities are calculated.
    - The results are stored.
  - The results of all tracks of one event are stored.
- The results of all events are stored.
- The residuals and impact parameters are computed and written to the log file.
- All results are saved to the binary file `results.mat` to be available later on.

At any time, the simulation run can be aborted by clicking the button "Interrupt" in the MatLab version. This forces the program to stop the run after the current track and drop all up to now stored results. In the Octave version, the key combination `Ctrl+c` has the same effect.

While running the MatLab version, the elapsed event number and track number is displayed in the main window's section "Messages". Figure 10 shows this section during a simulation run. Up, one can see which track is beeing processed. Down, printed in red, the program shows a warning because of a badly reconstructed track. The standardized residuals (the so called "MC pulls") and the start parameters of this very track are displayed. These warnings are enabled by setting the flag "Display bad tracks" in the window "Edit


Messages

Messages:
Event 7 of 10, track 62 of 100

Warnings:
Bad Track 84 Event 6
MCpull = -0.28 -0.67 -0.13 -0.00
3.35
Start params: -1.2522 14.8994
0.7888 -0.0005 -0.0001

results.mat

Figure 10: The section "Messages"

simulation parameters" (see figure 9).
In the OCTAVE version, the simulation progress and possible warnings are displayed in the shell.

Having finished a simulation run, the message "Simulation successfully finished" appears and the results are written to a so called "results file". Per default, all results of a simulation run are written to a results file named `results.mat`. All output routines access a results file, e.g. to create histograms from the reconstructed track parameters. A more detailed explaination of the contents of the results file can be found in section 8.2. After each simulation run, `results.mat` is overwritten by the results of the new run.

Results files in MATLAB : To have access to the results of a past run, one can save the results file by clicking "Save results" in the lower part of the "Messages" section. A file dialog will appear asking you for a filename. A copy of `results.mat` is generated under the quoted filename.
The current results file's name is displayed right above the buttons "Save results" and "Load results". When generating an output, the results file quoted here will be accessed. After saving a results file, the filename here is changed from `results.mat` to the filename of the saved results file, and all output routines will acces the new results file.
By clicking the button "Load results" an existing results file can be loaded. A file dialog appears asking you to choose a results file. Again, the current results file is changed to the selected one.

Results files in OCTAVE : To have access to the results of a past run, one can save the results file by typing `LDT_save('filename.mat')`. A copy of `results.mat` is generated under the quoted filename. Don't forget to specify the extension `.mat` as well!

The output routines in OCTAVE can be called with or without an input parameter. For example, the command `LDT_residuals` will make use of the results stored in the default results file `results.mat`, whereas calling this output routine like `LDT_residuals('filename.mat')` will use the results file `filename.mat`.

As already said, the results of a simulation run are used to generate different outputs from. These outputs are explained in section 6.

6 Description of the output

Outputs in MatLab : In the lower middle region of the main window one finds the section "Outputs" (see figure 11). Here, one can view the results of the simulation run and create outputs from the reconstructed tracks.

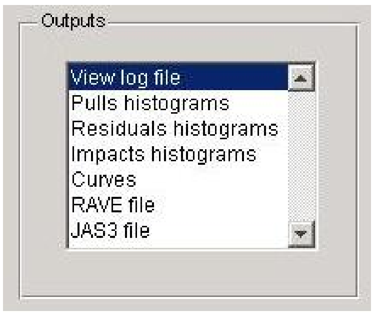


Figure 11: The section "Outputs"

With exception of "View log file", selecting any of the outputs here calls a subroutine, which accesses the current results file displayed in the section "Messages" (see figure 10). In the following, each one of the outputs seen in figure 11 is explained. The selection "Curves" will be explained in section 7.

Outputs in Octave : The Octave version of LDT can generate the same outputs as the MatLab version. The corresponding commands are:

- `edit LDT.log`
- `LDT_pulls`
- `LDT_residuals`
- `LDT_impacts`
- `LDT_curves`
- `LDT2rave`
- `LDT2jas3`

The output routines in Octave can be called with or without an input parameter. For example, the command `LDT_residuals` will make use of the results stored in the default results file `results.mat`, whereas calling this output routine like `LDT_residuals('filename.mat')` will use the results file `filename.mat`.

6.1 The log file

By selecting “View log file” in the MatLab version or by typing `edit LDT.log` under Octave , the file `LDT.log` is opened in the default text editor. It contains useful information about the latest simulation run. A sample log file can be found below:

```
Action: run simulation
Bad Track 72 Event 10
   MCpull = -0.00 3.16 1.02 -0.14 0.51
   Start params: -1.1764 -8.5039 2.0865 0.0009 0.0001
Bad Track 95 Event 10
   MCpull = -0.88 -3.61 -2.88 0.88 -0.14
   Start params: 2.6582 33.1675 0.4247 -0.0007 -0.0001
-----------------------------------------------------------------
-------------------   SIMULATION PARAMETERS   -------------------
-----------------------------------------------------------------
Pt:[Gev]     [   1.0000 100.0000 ]
phi:[deg]    [   0.0000 360.0000 ]
Theta:[deg]  [   6.0000 174.0000 ]
x:[mm]       [  -0.0000   0.0000 ]
y:[mm]       [  -0.0000   0.0000 ]
z:[mm]       [  -0.0000   0.0000 ]

Magnetic Field Bz:       3.5000  [T]
Mass of the particles    0.1050  [GeV]
(plays only a role in multiple scattering =>
zero mass means all particles are treated relativisticly!)

Flags:
Symmetry in Theta         : off
Use absolute momentum     : off
Scale down TPC by factor 5: on
Multiple scattering       : on
Measurement errors        : on
Display bad tracks        : on
Chi2                      : on

Barrel geometry:  geom/LDCPrime_02Sc_11.04.08.bgeom
Forward geometry: geom/LDCPrime_02Sc_11.04.08.fgeom
-----------------------------------------------------------------

Number of events:        10
Number of tracks/event:  100
barrel region:          800
forward region:         6
rear region:            0
intermediate region:    194

-----------------------------------------------------------------
-------------------------   RESULTS   -------------------------
-----------------------------------------------------------------
1000 out of 1000 tracks are used for test statistics

Pulls greater than 3 are considered bad

Bad pulls:
4  3  3  2  5

Total number of bad tracks:
14

Standardized residuals
      Phi           z          theta        beta       kappa
```

```
mean: 0.013977    0.042478   0.0015655   -0.004582  -0.0058704
std:  1.0065      1.0278     0.99438      1.0026      1.0179

Pulls of innermost hit
      Barrel tracks  Fwd/Rear tracks  -----Intermediate tracks-----
       RPhi    z       u       v       RPhi    z       u       v
mean: +0.00   -0.05   +0.05   +0.93   -0.02   +0.07   -0.05   -0.01
std:  +1.01   +1.02   +0.87   +1.02   +1.08   +1.00   +0.80   +0.93

R.M.S. values of the residuals and momenta [mu] and [mrad] resp.
      RPhi          z      theta        phi     dpt/pt   dpt/pt^2
   4.14593   10.44711    0.06764    0.11345    0.00800    0.00025

Impact paramters [mu]
          2d    3d (abs)
mean:  -0.0182   5.0237
std:    5.2248   4.6466

Chi^2
ndf:    86.774, average
mean:   86.6087
ratio:  1.001909

Mean of the MC-Chi^2 at the beamtube:
5.155256

Simulation successfully finished!
```

If the flag “Display bad track” is enabled, the first entries in the log file contain the event number and the track number of badly reconstructed tracks as well as their Monte-Carlo pulls (standardized residuals) and their start parameters. In the sample log file quoted here not all bad tracks are listed due to lack of space. The log file printed by the program of course contains all bad tracks.

The bad tracks are followed by the log file's proper header. It contains the simulation parameters, the magnetic field, the mass of the particles, the status of the flags and the selected input sheets.

The header is followed by a little track statistic. It contains the number of tracks and events simulated and the number of tracks gone through the forward, intermediate, barrel and rear region of the detector.

The main part of the log file contains the results of the simulation run, namely the test quantities and the residuals. This part of the log file is again opened by a statistic, containing the number of tracks that could be used for test statistics. If due to inefficiencies or geometric reasons some tracks did not yield enough measurements to be reconstructed properly, those tracks are ignored when computing test quantities and residuals.
If the flag “Display bad track” is enabled, the statistic of bad tracks is printed. This statistic contains the number of bad estimates for each track parameter seperately, i.e. $[\Phi, z, \vartheta, \beta = \varphi - \Phi, \kappa = \pm 1/R_H]$, as well as the total number of bad tracks.

The next section of the log file contains the standardized residuals (the “MC pulls”) and the pull quantities. The standardized residuals are computed from the simulated start parameters (Monte-Carlo truth) and the fitted parameters at the end of the Kalman filter for every track parameter. Their mean value should be compatible with 0 and their standard deviation should be compatible with 1.
The pulls of the innermost hit of every track are computed from the simulated measurements and the fitted track at that point. Again, the mean value should be compatible with 0 and the standard deviation with 1. If the innermost hit is in a cylinder layer, the measurements are $R\Phi$ and the second coordinate (e.g. z), otherwise the measurements are u and v. The pulls are computed seperately for barrel tracks, forward/rear tracks, and for intermediate tracks. In case of intermediate tracks, the two cases (innermost hit = barrel layer; innermost hit = forward layer) are declared seperately.

The next part of the log file contains the main output of the program, the residuals and the impact parameters. The values printed here are the root mean square values of the residuals of $R\Phi$, z, ϑ, and φ, as well as the root mean square values of $\frac{\Delta p_\mathrm{t}}{p_\mathrm{t}}$ and $\frac{\Delta p_\mathrm{t}}{p_\mathrm{t}^2} = \Delta\frac{1}{p_\mathrm{t}}$.
The next section contains the mean values and the standard deviations of the projected impact parameters and of the impact parameters in space.

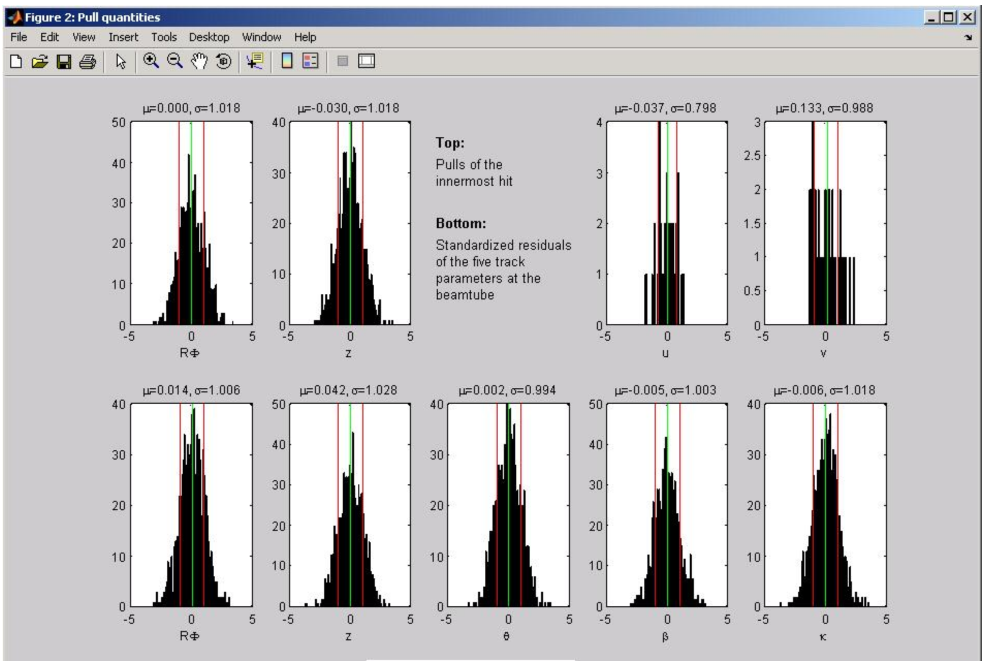
If the computation of χ^2 is enabled, the mean value and the average number of degrees of freedom of the total χ^2 is written to the log file. The mean value should be compatible with the average number of degrees of freedom, and so the quoted ratio should be around 1. In addition to that, the mean value of the χ^2 at the beamtube (computed from the start parameters, the fitted parameters and the covariance matrix at the beamtube) is printed. This value should be compatible with 5, the number of track parameters.

6.2 Histograms of pull quantities and standardized residuals

By selecting “Pulls histograms” in the “Outputs” section of the main window (typing `LDT_pulls` under Octave), the pull quantities and the standardized residuals are displayed in histograms, as shown in figure 12. The histograms in the top row show the distribution of the pulls of the innermost hits, barrel representation left and forward representation right. The bottom row shows the histograms of the standardized residuals, one histogram for each track parameter. The mean values are plotted as green lines and the standard deviations as red lines. In addition, these values are printed on top of the the corresponding histogram.

6.3 Histograms of the residuals with respect to the true track parameters

Selecting “Residuals histograms” in the “Outputs” section of the main window (typing `LDT_residuals` under Octave) opens the *main output of the program* (see figure 13).



Top:
Pulls of the innermost hit

Bottom:
Standardized residuals of the five track parameters at the beamtube

Figure 12: Histograms of pulls and standardized residuals.

The residuals of the fitted state vectors are displayed in form of histograms, together with the corresponding root mean square value. The root mean square of the deviation from the Monte Carlo truth serves as a measure for the detector resolution. The program draws histograms of the residuals of $R\Phi$, φ, z, ϑ, and of $\Delta p_\mathrm{t}/p_\mathrm{t}$ and $\Delta p_\mathrm{t}/p_\mathrm{t}^2$. The root mean square is illustrated by red lines.

The four histograms in the upper half of the window display the residuals of $R\Phi$, φ, z and ϑ. For better readability, the displayed RMS values of $R\Phi$ and z are shown in micrometers, and the displayed RMS values of φ and ϑ are shown in milliradians. The histograms in the lower half display $\frac{\Delta p_\mathrm{t}}{p_\mathrm{t}}$ (dimensionless) and $\frac{\Delta p_\mathrm{t}}{p_\mathrm{t}^2}$ (in units of c/GeV).

The relative deviation $\frac{\Delta p_\mathrm{t}}{p_\mathrm{t}}$ can be parametrized in the following way [1]:

$$\left(\frac{\Delta p_\mathrm{t}}{p_\mathrm{t}}\right)^2 = c_1 p_\mathrm{t}^2 + c_2\sqrt{1 + m^2/p^2},$$

where the first term is due to the detector resolution and the second term takes into account multiple scattering. Neglecting $m^2/p^2$and dividing by p_t yields.

$$\frac{\Delta p_\mathrm{t}}{p_\mathrm{t}^2} = \sqrt{c_1 + c_2/p_\mathrm{t}^2}.$$

For increasing p_t, $\frac{\Delta p_\mathrm{t}}{p_\mathrm{t}^2}$ is therefore nearly independent of p_t, and for this reason its root mean square value is often used for comparisons of different detector setups.

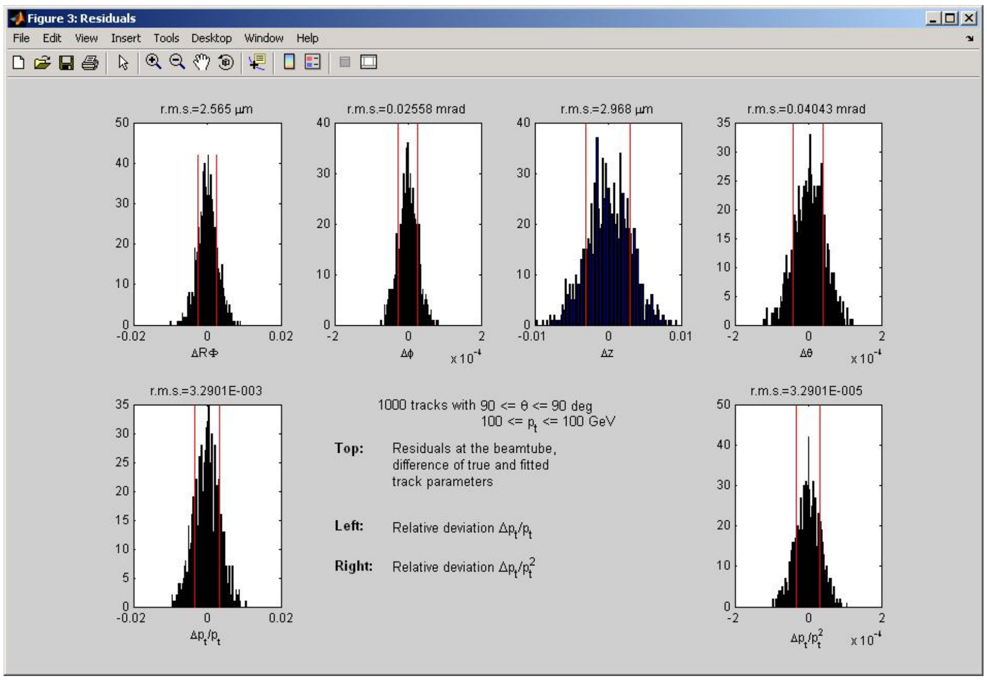


Figure 13: Residuals histograms (difference of reconstructed and simulated track parameters).

6.4 Impact parameters

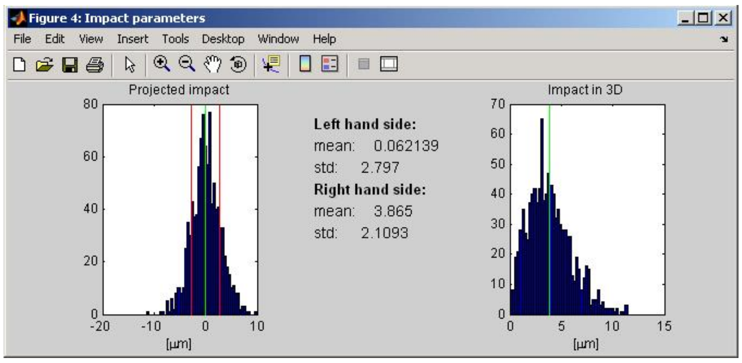
By selecting “Impacts histograms” in the “Outputs” section of the main window (or by typing `LDT_impacts` under Octave), the program displays the projected (transverse) impact parameters and the impact parameters in space in form of histograms (see figure 14). The corresponding mean values and standard deviations are displayed as well.

6.5 RAVE file

The program is able to produce input for a subsequent vertex fit, using the VERTIGO framework and RAVE vertex reconstruction toolkit [8].

MatLab : By selecting “RAVE file” a file dialog is asking you for the name of the file to be written. The routine uses the results stored in the current results file.

Octave : The routine `LDT2rave` can be called with either one or two input parameters. The call `LDT2rave('filename.txt')` will use the results stored in the default results file `results.mat` and write it to the file specified by the given filename, whereas calling it like `LDT2rave('filename.txt','myresultsfile.mat')` will use the results stored in the results file `myresultsfile.mat`.



Left hand side:
mean: 0.062139
std: 2.797
Right hand side:
mean: 3.865
std: 2.1093

Figure 14: Histograms of the impact parameters.

As demanded by RAVE, the reconstructed track parameters and their covariances are first converted to a Cartesian 6D representation (with a 6×6 covariance matrix of rank 5) and then written as text in the VERTIGO Data Harvester's CSV (comma separated variables) format.

6.6 JAS3 file

For doing further studies with the reconstructed and simulated tracks, the program can write a simple text file for a special Java program embedded in the JAS3 analysis tool [9].

MatLab : By selecting "JAS3 file" a file dialog is asking you for the name of the file to be written. The routine uses the results stored in the current results file.

Octave : The routine `LDT2jas3` can be called with either one or two input parameters. The call `LDT2jas3('filename.txt')` will use the results stored in the default results file `results.mat` and write it to the file specified by the given filename, whereas calling it like `LDT2jas3('filename.txt','myresultsfile.mat')` will use the results stored in the results file `myresultsfile.mat`.

Then, the program collects the demanded data and writes the output.

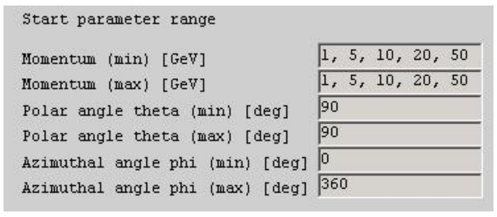
7 Calculating and displaying curves

In the section “Start parameter range” (section 4.2) was explained how to define the phase space of possible start parameters and the number of events and tracks. Having carried out a simulation run, one obtains rms-values of the deviations of the track parameters from the Monte Carlo truth, as well as the impact parameters (projected and in space), and the relative momentum resolution values $\frac{\Delta p_t}{p_t}$ and $\frac{\Delta p_t}{p_t^2} = \Delta\frac{1}{p_t}$. Now one will probably want to know, how these values change in dependence on e.g. the momentum, or, how they change when using different detector setups.

7.1 Curves with different start parameters

The program is able to carry out a sequence of simulation runs with different start parameters, using a “hidden” feature in the input of the start parameters. Instead of defining *one* pair of minimum and maximum values, you simply put in more pairs of minimum and maximum values. In the MatLab version this is done in the window “Edit simulation parameters”, while in the Octave version one uses the main steering file `simulation_paramters_Octave.txt`. In the following, only the input within the MatLab version is shown, the corresponding input in the Octave version looks the same.

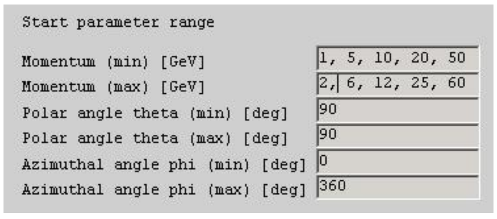
Example: If you want to determine the detector performance for the momenta of 1, 5, 10, 20 and 50 GeV/c, simply put these values in the corresponding input lines of minimum and maximum momentum, in the window “Edit simulation parameters” (see figure 15). The first value of these lines defines the momentum range of the first point of the curve, the second one defines the range of the second point, etc.
Just like before, if you want to simulate the points with a momentum range rather than with fixed values for the momentum, you simply put in different values for the minimum and maximum value of p_t. An example for this is shown in figure 16.



Start parameter range

Momentum (min) [GeV]	1, 5, 10, 20, 50
Momentum (max) [GeV]	1, 5, 10, 20, 50
Polar angle theta (min) [deg]	90
Polar angle theta (max) [deg]	90
Azimuthal angle phi (min) [deg]	0
Azimuthal angle phi (max) [deg]	360

Figure 15: Curve with fixed values of p_t



Start parameter range

Momentum (min) [GeV]	1, 5, 10, 20, 50
Momentum (max) [GeV]	2, 6, 12, 25, 60
Polar angle theta (min) [deg]	90
Polar angle theta (max) [deg]	90
Azimuthal angle phi (min) [deg]	0
Azimuthal angle phi (max) [deg]	360

Figure 16: Curve with ranges of p_t

Having carried out the simulation runs, the program opens two figure windows. One displays the rms values of the deviations from the Monte Carlo truth of $R\Phi$, z, ϑ and φ in dependence on p_t. The other one shows the standard deviation of the projected impact parameter, the mean value of the impact parameter in space and the root mean square values of the relative momentum resolution values $\frac{\Delta p_t}{p_t}$ and $\frac{\Delta p_t}{p_t^2}$.

As an example, the rms value of $\frac{\Delta p_t}{p_t^2}$ in dependence on p_t can be seen in figure 17. The curves in dependence on p_t are plotted in logarithmic scale.

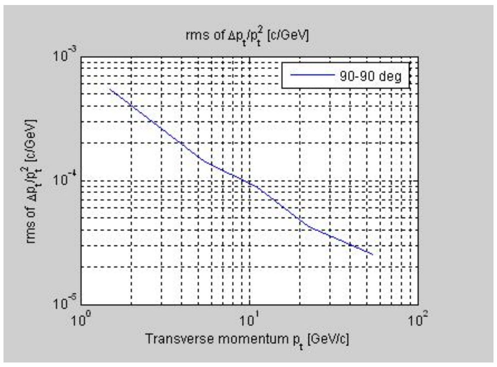
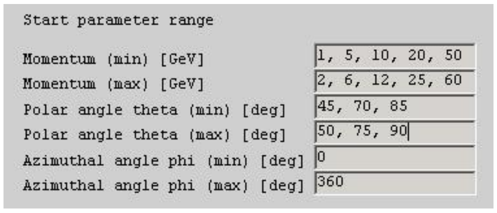


Figure 17: Example: $rms\left(\frac{\Delta p_t}{p_t^2}\right)$ in dependence on p_t

You can compute multiple curves at once, for example with different values of ϑ. To do this, just put in multiple pairs of minimum and maximum values of ϑ as start parameters (see figure 18). Having carried out the simulation for the first range of ϑ and all ranges of p_t, the program continues the calculation with the next ϑ range and the first p_t range. Now, in the output plots, more curves will appear, displaying the results for the different ϑ ranges (see figure 19).



Start parameter range

Momentum (min) [GeV]	1, 5, 10, 20, 50
Momentum (max) [GeV]	2, 6, 12, 25, 60
Polar angle theta (min) [deg]	45, 70, 85
Polar angle theta (max) [deg]	50, 75, 90
Azimuthal angle phi (min) [deg]	0
Azimuthal angle phi (max) [deg]	360

Figure 18: Input of curves with different ϑ ranges

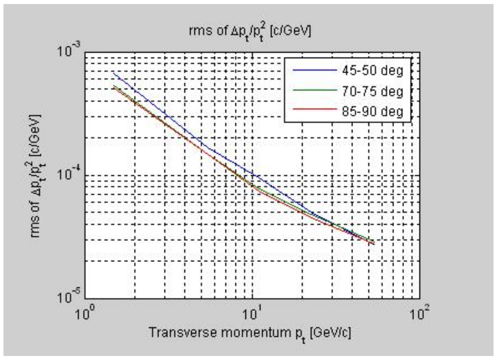
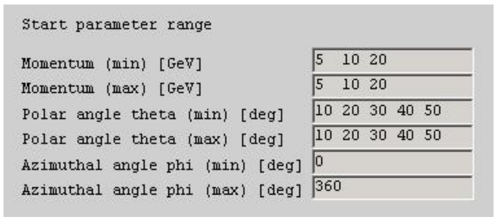


Figure 19: Curves with different ϑ ranges

If you want to determine the detector's behaviour in dependence on ϑ for a fixed value of p_t, just do the whole thing the other way 'round. Put in more pairs of minimum and maximum values for ϑ and only one pair of values for p_t. Again, you can compute curves for different values of p_t, too (see figure 20).

If there are more value pairs for ϑ than for p_t, the program plots curves for the different values of p_t in dependence on ϑ. That means, ϑ instead of p_t occurs in the ordinate and the plot will be a linear one rather than a logarithmic one (see figure 21). As long as there are at least as many value pairs for p_t as for ϑ, the plot will be in dependence on p_t.



Start parameter range

Momentum (min) [GeV]	5 10 20
Momentum (max) [GeV]	5 10 20
Polar angle theta (min) [deg]	10 20 30 40 50
Polar angle theta (max) [deg]	10 20 30 40 50
Azimuthal angle phi (min) [deg]	0
Azimuthal angle phi (max) [deg]	360

Figure 20: Input of curves dependent on ϑ

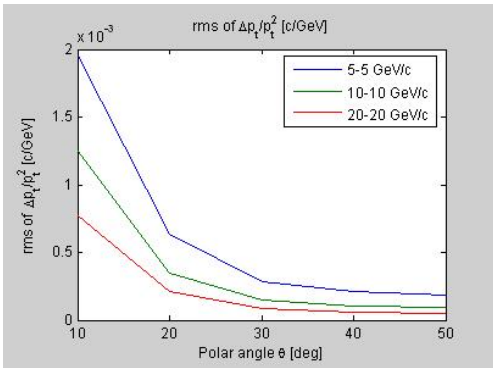


Figure 21: Curves dependent on ϑ

Similar to the normal case, the different ϑ ranges are plotted as red lines in the 2D detector display, as can be seen in figure 22.

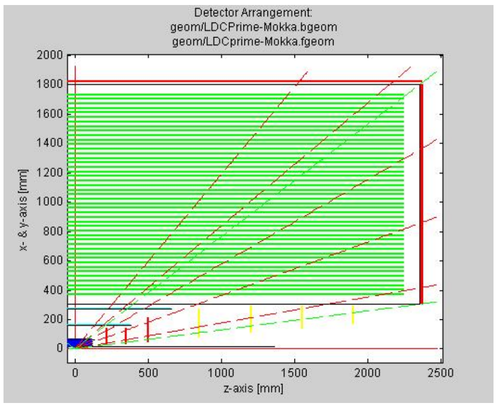


Figure 22: Different ϑ ranges in the detector display

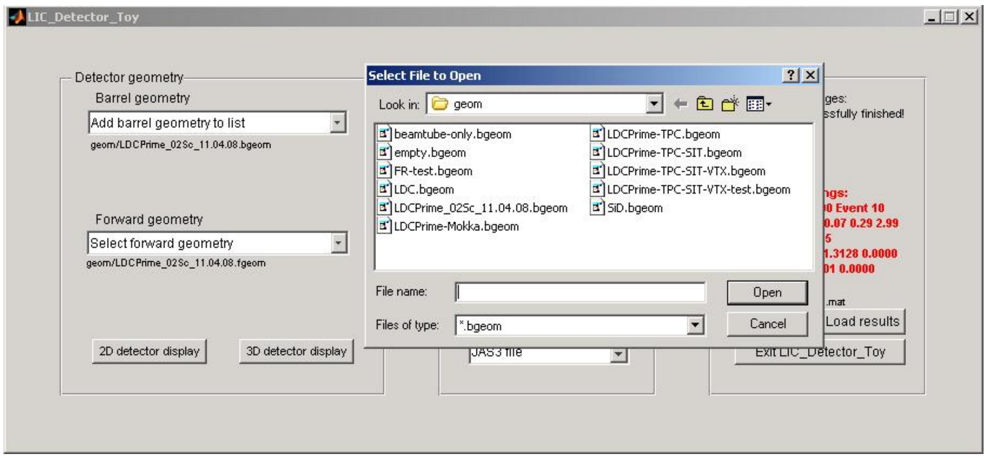
When running a normal simulation run, without calculating curves, the program writes information and results of that run to the log file "LDT.log". In case of curves, this log file contains the corresponding information of each single run. Thus, one can look up e.g. the values of the test quantities of every calculated point. The single runs can be identified by the information in the header of the log file.

Like in the normal case, the results of the repeated simulation run are written to the results file (see section 5). But since the default results file `results.mat` is overwritten each time a new simulation run is carried out, only the detailed information of the *last* run is available for generating histograms or other outputs from. Since the interesting values obtained from a curve run are the rms values of the track parameters, the impact parameters and the momentum resolutions, those values are collected in each run and written to the results file. When saving the results after a curve run, on can recall the plots of the curves by loading the corresponding results file and clicking "Curves" in the main window's "Output" section.

7.2 Curves with different detector setups

In literature, one often finds plots showing e.g. the relative momentum resolution $\sigma\left(\frac{\Delta p_{\mathrm{t}}}{p_{\mathrm{t}}^2}\right)$ in dependence on the transverse momentum p_{t} for different detector setups. Such plots allow to resolve the effect of changing e.g. some geometric parameters for a wide range of momentum.
LDT is able to produce such plots in a very easy way, similar to what was explained in the previous section.

MatLab : In the main window's section "Detector geometry" one can define a so called "geometry list" of barrel and forward input sheets, each, which are processed in sequence. To establish such a list, one has to use the selections "Add barrel (forward) geometry to list" and "Remove barrel (forward) geometry from list" in the corresponding pull down menus.
For normal applications, said list of input sheets only consists of one single input sheet, namely the one selected by "Select barrel (forward) geometry". Selecting "Add barrel (forward) geometry to list" now has the effect of extending this list by an additional input sheet. This input sheet is selected by a habitual file dialog window. This can be seen in figure 23.



Barrel geometry: Add barrel geometry to list
geom/LDCPrime_02Sc_11.04.08.bgeom
Forward geometry: Select forward geometry
geom/LDCPrime_02Sc_11.04.08.fgeom

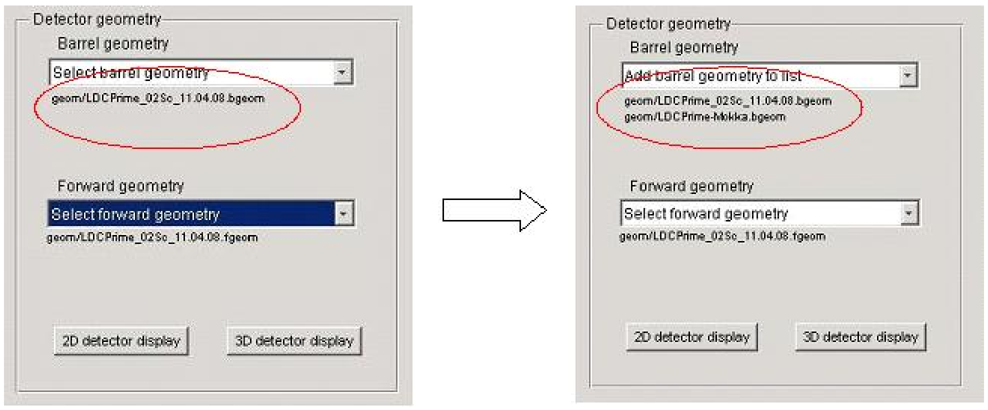
Look in: geom

beamtube-only.bgeom
empty.bgeom
FR-test.bgeom
LDC.bgeom
LDCPrime_02Sc_11.04.08.bgeom
LDCPrime-Mokka.bgeom
LDCPrime-TPC.bgeom
LDCPrime-TPC-SIT.bgeom
LDCPrime-TPC-SIT-VTX.bgeom
LDCPrime-TPC-SIT-VTX-test.bgeom
SiD.bgeom

File name:
Files of type: *.bgeom

Figure 23: Adding a barrel input sheet to the list

Beyond the pull down menu, where the selected input sheet's name is displayed, the name of the new input sheet is added, as can be seen in figure 24.


Detector geometry
Barrel geometry
Select barrel geometry
geom/LDCPrime_02Sc_11.04.08.bgeom
Forward geometry
Select forward geometry
geom/LDCPrime_02Sc_11.04.08.fgeom
2D detector display
3D detector display

Detector geometry
Barrel geometry
Add barrel geometry to list
geom/LDCPrime_02Sc_11.04.08.bgeom
geom/LDCPrime-Mokka.bgeom
Forward geometry
Select forward geometry
geom/LDCPrime_02Sc_11.04.08.fgeom
2D detector display
3D detector display

Figure 24: Adding a barrel input sheet to the list

Repeating this process, one can establish a list of any number of input sheets. Note, that although the main window only offers space enough to display the names of 5 input sheets, the list can contain more.
Of course, the lists of barrel and forward input sheets have to contain *the same number* of corresponding input sheets. Selecting "Remove barrel (forward) geometry from list" deletes the last entry in the list. An example of correctly set up geometry lists is shown in figure 25. As one can see, it's possible to add the same input sheet multiple times to the list.

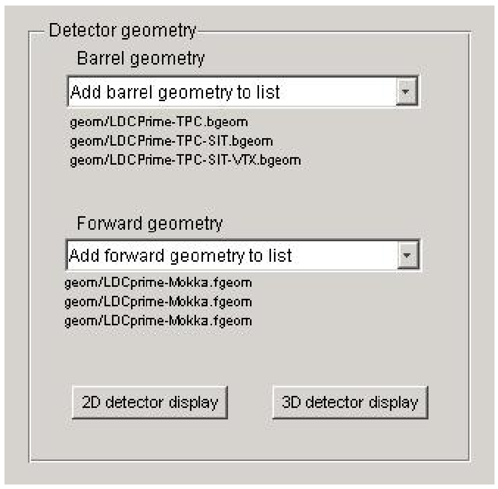
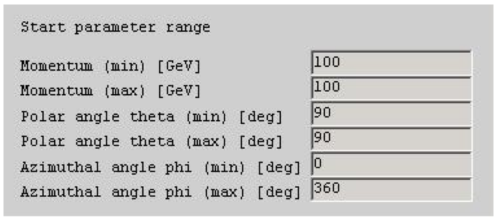

Detector geometry
Barrel geometry
Add barrel geometry to list
geom/LDCPrime-TPC.bgeom
geom/LDCPrime-TPC-SIT.bgeom
geom/LDCPrime-TPC-SIT-VTX.bgeom
Forward geometry
Add forward geometry to list
geom/LDCprime-Mokka.fgeom
geom/LDCprime-Mokka.fgeom
geom/LDCprime-Mokka.fgeom
2D detector display
3D detector display

Figure 25: Example for geometry lists

Using the shown example geometry lists, evaluation at $p_t = 100$ [GeV/c] and $\vartheta = 90$

[deg] (as shown in figure 26) would result in three simulation runs in sequence, the first with the first input sheet of the barrel geometry list and the first input sheet of the forward geometry list. After the first run, the program continues with the second input sheets on the lists, and so on. Having finished the three runs, it opens the same two figure windows as before, see the example of $rms\left(\frac{\Delta p_t}{p_t^2}\right)$ in figure 27.



```
Start parameter range

Momentum (min) [GeV]            100
Momentum (max) [GeV]            100
Polar angle theta (min) [deg]   90
Polar angle theta (max) [deg]   90
Azimuthal angle phi (min) [deg] 0
Azimuthal angle phi (max) [deg] 360
```

Figure 26: Sample start parameters (only one point per detector setup)

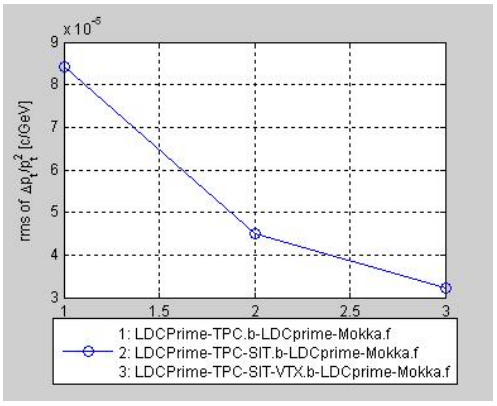


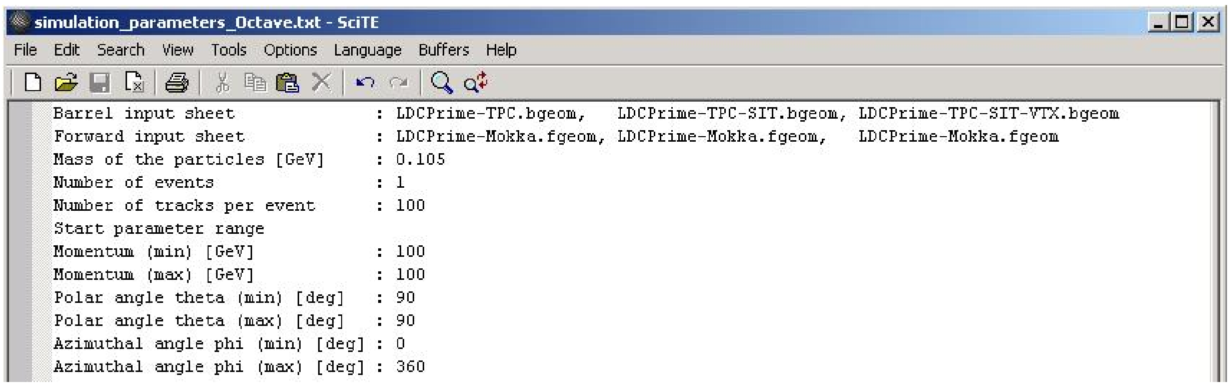
Figure 27: Results of a simulation run with different input sheets

The three plotted points show the result of the simulation runs for the different input sheets. From the legend beyond the plot one can tell, which point belongs to which input sheets. Due to space saving reasons, the extensions of the input sheets, `.bgeom` and `.fgeom` are abbreviated with `.b` and `.f`.

OCTAVE : Under OCTAVE , setting up geometry lists is done in the main steering file `simulation_parameters_Octave.txt`. As explained in section 3, the first two lines of the steering file contain the selected input sheets for the barrel region and the forward region. A geometry list can easily be established by adding more input sheets in the corresponding lines, seperated by commas. The OCTAVE geometry list and start parameters corresponding to figure 25 and 26 can be found in figure 28.

Using the shown example geometry lists, evaluation at $p_t = 100$ [GeV/c] and $\vartheta = 90$ [deg] (as shown in figure 28) would result in three simulation runs in sequence, the first with the first input sheet of the barrel geometry list and the first input sheet of the forward geometry list. After the first run, the program continues with the second input sheets on the lists, and so on. Having finished the three runs, it opens the same two figure windows as before, see the example of $rms\left(\frac{\Delta p_t}{p_t^2}\right)$ in figure 29.

Since OCTAVE is not able to display multilined legend entries, the legend doesn't look as nice as in MATLAB . To identify the points, one has to have a look at the main steering file. Point 1 corresponds to the first pair of input sheets, point 2 to the second, etc.



```
Barrel input sheet             : LDCPrime-TPC.bgeom,   LDCPrime-TPC-SIT.bgeom, LDCPrime-TPC-SIT-VTX.bgeom
Forward input sheet            : LDCPrime-Mokka.fgeom, LDCPrime-Mokka.fgeom,   LDCPrime-Mokka.fgeom
Mass of the particles [GeV]    : 0.105
Number of events               : 1
Number of tracks per event     : 100
Start parameter range
Momentum (min) [GeV]           : 100
Momentum (max) [GeV]           : 100
Polar angle theta (min) [deg]  : 90
Polar angle theta (max) [deg]  : 90
Azimuthal angle phi (min) [deg] : 0
Azimuthal angle phi (max) [deg] : 360
```

simulation_parameters_Octave.txt - SciTE
File Edit Search View Tools Options Language Buffers Help

Figure 28: Example for geometry lists in OCTAVE

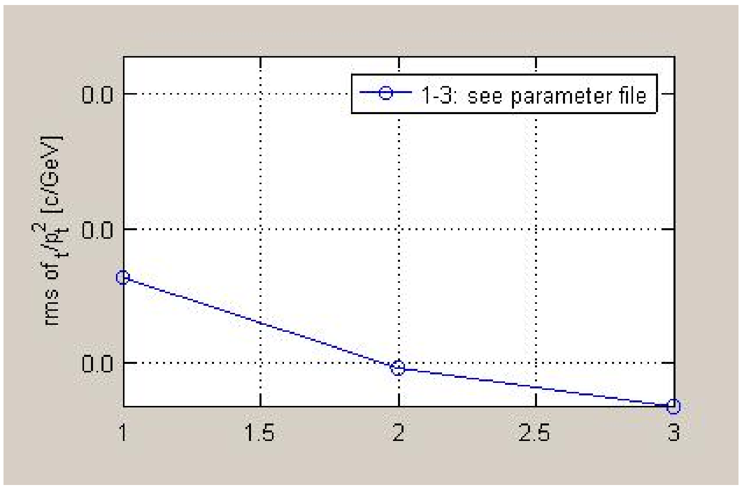
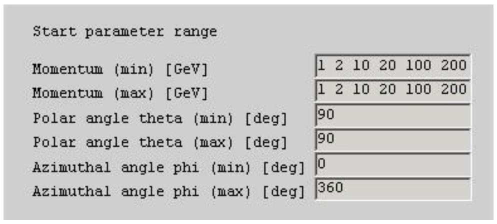


Figure 29: Results of a simulation run with different input sheets in OCTAVE

In the previous example we resolved the difference between the three detector setups at only one point in the phase space of possible start parameters, we used only $p_{\mathrm{t}} = 100$ [GeV/c] and $\vartheta = 90$ [deg]. Similar to what was presented in the previous section, we can do the comparison of the detector setups for different momenta and polar angles, too. A sample input of start parameters is shown in figure 30 and figure 32, the result of the simulation can be seen in figure 31 and 33.



Start parameter range

Momentum (min) [GeV]	1 2 10 20 100 200
Momentum (max) [GeV]	1 2 10 20 100 200
Polar angle theta (min) [deg]	90
Polar angle theta (max) [deg]	90
Azimuthal angle phi (min) [deg]	0
Azimuthal angle phi (max) [deg]	360

Figure 30: Sample start parameters (5 different momenta per detector setup)

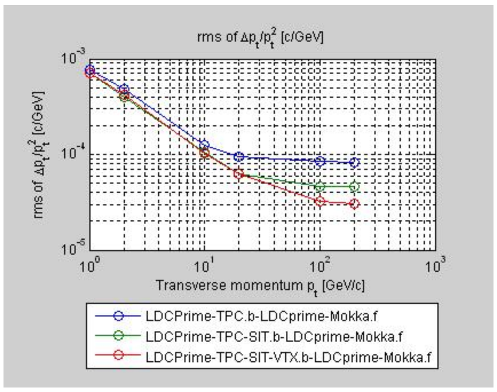
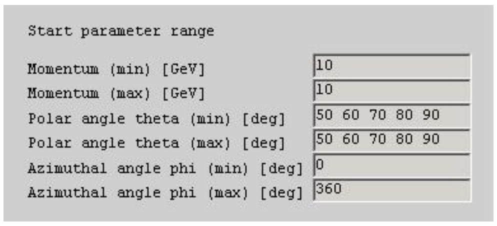


Figure 31: Results of a simulation run with different input sheets, in dependence on p_{t}



Start parameter range

Momentum (min) [GeV]	10
Momentum (max) [GeV]	10
Polar angle theta (min) [deg]	50 60 70 80 90
Polar angle theta (max) [deg]	50 60 70 80 90
Azimuthal angle phi (min) [deg]	0
Azimuthal angle phi (max) [deg]	360

Figure 32: Sample start parameters (5 different polar angles per detector setup)

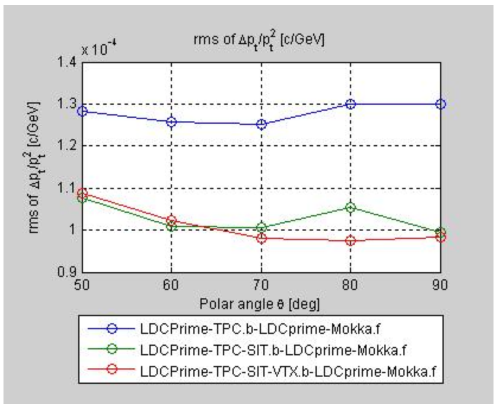


Figure 33: Results of a simulation run with different input sheets, in dependence on ϑ

Now, the programs executes several simulation runs using the same input sheets, for the different values of p_{t} and ϑ, respectively. Having processed the runs for one detector geometry, it loads the next one and calculates the same points with the new geometry. Therefore, in the plot now appear three curves representing the detector behaviour of the different setups in dependence on p_{t} and ϑ, respectively.

When computing curves, the program plots e.g. the relative momentum resolution in dependence on two parameters. As long as you use only *one* barrel input sheet and *one* forward input sheet (and no further entries in the geometry list), these parameters can be the momentum and the polar angle. As seen in section 7.1, the plot will be in dependence on the quantity with more start parameters, while the other one is used to identify the different curves.

When a geometry list of more than one input sheets is defined, this list is treated with priority. That means, that the different input sheets are used to define the different curves, while the plot is done in dependence on *one* of the other quantities. So you cannot compare different detector setups depending on the momentum *and* the polar angle at the same time. But as shown at the beginning of section 7.2, if the program is assigned to calculate curves using different input sheets with the same start parameters (only one momentum and one polar angle), the plot will be dependent on the input sheets. This avoids getting a plot with single points, one above the other, at the same momentum or polar angle.

If for both the momentum and the polar angle multiple start values are defined, the simulations will be done depending on the quantity with more entries. In this case, the first entry of the other quantity will be used, the further ones will be ignored. Again, if the two quantities have the same number of entries, p_{t} will be used rather than ϑ.

8 Additional features

In this section some features are listed that are not enabled by default. All these features are thought for people who know what they are doing, i.e. knowledge of the MatLab - programing language is required to use this features.

8.1 Some key variables

When it comes to length units, the program internally uses [mm]. This can be changed by changing the variable `unit` in the file `LDT_Matlab.m` or `LDT_Octave.m`, respectively. `unit` holds the internally used length unit in units of [mm]. That means, for computation in [mm] `unit` holds the value 1. For [cm], it holds 10, for [m] it holds 1000. Since `unit` is defined to be a global variable, the whole simulation process and all subroutines have access to it and rescale all affected values automatically.

In the new version of the program a lower limit for the transverse momentum has been implemented. By default it is set to $0.5GeV$. The limit is hard coded in `LDT_run.m`, if one wants an other value, the variable `pt_limit` has to be changed.
Note: The limit for the transverse momentum only applies if the flag for the use of the absolute momentum is set.

The subroutine `LDT_display2D.m` displays the detector arrangement by default including the names of the detectors and the values of the limiting angles, which devide the ϑ range in forward, intermediate, barrel and rear region. One can disable these text outputs in the sketch by setting the variables `drawtheta` and `drawnames` to 0 at the beginning of the code of `LDT_display2D.m`.
Since a TPC usually consists of a high number of layers, the program by default only displays every tenth layer of the TPC. To have every layer be displayed, set the variable `drawwholeTPC` to 1. If "Scale down TPC by factor 5" is selected, every layer is displayed by default.

It is possible to simulate one single track several times with a different sequence of random numbers. This is realized by moving the start of the track loop (`for ITrack=1:N.Track`) after the point where the start parameters for the track are generated. An appropriate section in the source code is commented out.
One can influence the initial state of the random generators by changing the variable `runnumber` in the main program `LDT_run.m`. For reasons of simplicity, the run number is restricted to integer values >0.

The variable `minmeas` defines the minimum number of measurements per coordinate. By default it is set to 3 in the main program (`LDT_run.m`). If a track has less measurements of one or both coordinates, it is not considered in computing pull quantities and residuals.

8.2 The results file

At several points in this manual the results file was mentioned, which contains the collected results of a simulation run. Since there is no physical difference between the tracks of different events, the tracks from all events are gathered together. That means, a simulation of 10 events with 100 tracks each results in a collection of 1000 tracks. In the following, the "number of tracks" N_{tracks} is understood the number of events times the number of tracks per event of the considered simulation run.
The results file is a MatLab workspace file, with help of which variables used in the program can be stored. The variables stored in `results.mat` are loaded to the workspace using the command `load results`, and can afterwards be viewed and manipulated by normal MatLab commands.

The most important variables of the results file are:

- `Aparam_start` [$N_{tracks} \times 5$ double]: The start parameters of each track, defined at the inner side of the innermost cylinder layer. The track parameters are $[\Phi, z, \vartheta, \beta(= \varphi - \Phi), \kappa_{signed}]$. Angles are measured in [Rad], lengths are measured in the currently used length unit [unit] (see above).
- `Avertex` [$N_{events} \times 3$ double]: The space point where the tracks of each event came from, in cartesian coordinates x, y and z, in [unit].
- `Aparam_fit` [$N_{tracks} \times 5$ double]: The fitted track parameters of each track, defined at the inner side of the innermost cylinder layer.
- `ACf_store` [$N_{tracks} \times 5 \times 5$ double]: The 5×5 covariance matrix of each track, defined at the inner side of the innermost cylinder layer.
- `Ameas` [$N_{tracks}\times$ maximum number of traversed detectors, double]: Information about how many coordinates have been measured by a layer, for every track. 0 means no measurement, 1 means first coordinate measured, 2 means second coordinate measured, 3 means both coordinated measured.
- `Ahitpattern` [$N_{tracks}\times$ maximum number of traversed detectors, cell]: The layer names of traversed layers for each track. Together with `Ameas` one can look up which detector measured which coordinate.
- `Apull` [$N_{tracks} \times 2$ double]: The real pull quantities at the first nonpassive layer, for every track and the two coordinates.
- `AMC_pull` [$N_{tracks} \times 5$ double]: The Monte-Carlo pull quantites (the standardized residuals) at the innermost cylinder layer, for every track and the five track parameters.

- `AMC_res` [$N_{tracks} \times 5$ double]: The deviations of the reconstructed track parameters from the Monte-Carlo truth (`Aparam_start`) at the innermost cylinder layer.
- `AMCpullhit` [$1 \times N_{tracks}$ logical array]: Indicates the tracks, that could be used for computation of pull quantities and residuals. Tracks indicated with 1 could be used, those with 0 were rejected.
- `chi2` [$N_{tracks} \times 1$ double]: The total (accumulated) χ^2 test quantity at the innermost cylinder layer, for every track indicated to be useful by `AMCpullhit`.
- `ndf` [$N_{tracks} \times 1$ double]: The number of degrees of freedom, with which the total χ^2 test quantity at the innermost cylinder layer should be compatible, for every track indicated to be useful by `AMCpullhit`.
- `nameb, namef`: These variables are cell arrays containing the name(s) of the used barrel and forward input sheets.
- `resultsofcurves` [$8 \times N_{curves} \times N_{points}$]: This is the array where the results of a curve run are stored. The dimension depends on the number of curves and the number of points per curve, as determined by the geometry lists and the number of start parameters. In `resultsofcurves` the rms values of the residuals (the deviations of the reconstructed track parameters from the MC truth), the impact parameters and the rms values of the relative momentum resolutions are stored:
  - `resultsofcurves`$(1,:,:)$: rms value of $\Delta R\Phi$
  - `resultsofcurves`$(2,:,:)$: rms value of Δz
  - `resultsofcurves`$(3,:,:)$: rms value of $\Delta\vartheta$
  - `resultsofcurves`$(4,:,:)$: rms value of $\Delta\varphi$
  - `resultsofcurves`$(5,:,:)$: rms value of $\frac{\Delta p_t}{pt}$
  - `resultsofcurves`$(6,:,:)$: rms value of $\frac{\Delta p_t}{pt^2}$
  - `resultsofcurves`$(7,:,:)$: standard deviation of projected impact parameters
  - `resultsofcurves`$(8,:,:)$: mean value of impact parameters in space

Acknowledgements

The "LDT" software was designed and developed in the Vienna ILC Project Group. Thanks are due to R. Höfler for valuable contributions. Efficient helix tracking was actively supported by W. Mitaroff (HEPHY Vienna), who also contributed to this manuscript. The corresponding author is M. Valentan.

Please send bug reports to `valentan@hephy.oeaw.ac.at`.

References

[1] R. Frühwirth et al.: *Data Analysis Techniques for High-Energy-Physics*, 2^{nd} edition (eds. M. Regler and R. Frühwirth), Cambridge University Press (2000).

[2] M. Regler, R. Frühwirth and W. Mitaroff: Int.J.Mod.Phys. **C7, 4** (1996) 521.

R. Frühwirth et al.: Nucl.Instr. Meth. **A 334** (1993) 528.

[3] D. Hanselmann and B. Littlefield: *Mastering* MatLab *7*, Pearson Prentice-Hall (2005).

See also `http://www.mathworks.com/`.

[4] Octave on the web: `http://www.gnu.org/software/octave/`

[5] M. Regler, M. Valentan and R. Frühwirth: *The LiC Detector Toy Program*, Proc. 11^{th} Vienna Conf. on Instrumentation (VCI) 2007, Nucl.Instr.Meth. **A 581** (2007) 553.

[6] W. Mitaroff, M. Regler, M. Valentan and R. Höfler: *Track Resolution Studies with the "LiC Detector Toy" Monte Carlo Tool*, Proc. 10^{th} Int. Linear Collider Workshop (LCWS) 2007, DESY Hamburg (to be published).

M. Regler, W. Mitaroff, M. Valentan, R. Frühwirth and R. Höfler: *The "LiC Detector Toy" Program*, Proc. Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP) 2007, Victoria, Canada (to be published).

[7] M. Leitgab and M. Regler: Study of the effects of higher beam luminosities on the impact parameter resolution of the 2006 BELLE Inner Tracker using the LiC Detector Toy, HEPHY Lab Report "Projektarbeit Hochenergiephysik" part 2, Vienna 2007.

[8] W. Waltenberger, F. Moser and W. Mitaroff: *RAVE – a detector-indepenent vertex reconstruction toolkit*, Proc. 11^{th} Vienna Conf. on Instrumentation (VCI) 2007, Nucl.Instr.Meth. **A 581** (2007) 549.

[9] Java Analysis Studio (JAS3): `http://jas.freehep.org/`

W. Mitaroff: private communication.

Appendix: The global coordinate system

The detector setup is assumed to be rotational symmetric w.r.t. the z-axis, but not necessarily mirror symmetric w.r.t. the origin $z = 0$. The axes (x, y, z) define a right-handed orthogonal basis. By convention, the x-axis is chosen to be horizontal w.r.t. the ground, and the y-axis to point upwards.

Surfaces are either cylinders of radius R and borders $z_{lower} \leq z \leq z_{upper}$ ("barrel region"), or planes normal to the z-axis with circular borders $R_{inner} \leq R \leq R_{outer}$ ("forward and rear regions"). They may be real or virtual.

Besides Cartesian coordinates, cylinder coordinates and spherical polar coordinates are defined for space points and/or momenta:

Space points $\vec{x} = [x, y, z]_{\mathrm{cart}} = [R, \Phi, z]_{\mathrm{cyl}}$

$x = R \cdot \cos\Phi$	$R = \sqrt{x^2 + y^2}$	
$y = R \cdot \sin\Phi$	$\Phi = \arctan(y/x)$,	azimuth angle $0 \leq \Phi < 2\pi$

Momenta $\vec{p} = [p_x, p_y, p_z]_{\mathrm{cart}} = [P, \vartheta, \varphi]_{\mathrm{sph}} = [p_T, \varphi, p_z]_{\mathrm{cyl}}$

$p_x = P \cdot \sin\vartheta \cdot \cos\varphi$	$P = \sqrt{p_x^2 + p_y^2 + p_z^2}$	$= \sqrt{p_T^2 + p_z^2}$
$p_y = P \cdot \sin\vartheta \cdot \sin\varphi$	$\vartheta = \arccos(p_z/P)$,	polar angle $0 \leq \vartheta \leq \pi$
$p_T = P \cdot \sin\vartheta$	$\lambda \equiv \frac{\pi}{2} - \vartheta$,	dip angle $\frac{\pi}{2} \geq \lambda \geq -\frac{\pi}{2}$
$p_z = P \cdot \cos\vartheta$	$\varphi = \arctan(p_y/p_x)$,	azimuth angle $0 \leq \varphi < 2\pi$

The **magnetic field** is assumed to be homogeneous and aligned parallel or anti-parallel to the z-axis. It is defined by the flux density $\vec{B} = [0, 0, B_z]_{\mathrm{cart}}$. This implies a helix track model, with the helix axis being parallel to z.

The following **units** are used: [*length*] = mm, [*angle*] = rad, [*momentum*] = GeV/c, [*B field*] = T (Tesla), and [*charge*] = e (elementary charge). For a particle with momentum P and charge Q, the radius of the helix and its signed inverse are

$$r_H = \frac{1}{K_{\mathrm{mm}}} \cdot \frac{P \cdot \sin\vartheta}{\mid Q \cdot B_z \mid} \qquad \kappa = -\mathrm{sign}(Q \cdot B_z) \cdot \frac{1}{r_H}$$

with the unit-dependent constant $K_{\mathrm{mm}} = 10^{-15}\, c = 0.00029979... \frac{\mathrm{GeV}/c}{\mathrm{T \cdot mm}}$. Our sign convention corresponds to $\mathrm{sign}(\kappa) = \mathrm{sign}(\frac{\mathrm{d}\varphi}{\mathrm{d}s}) \equiv$ sense of rotation in (x, y)-projection. Note that in the absence of matter, P and ϑ are constants of motion.

The **helix equations** for a starting point $[x_S, y_S, z_S]$ and a starting azimuthal direction angle φ_S, as functions of the running parameter φ, are:

$$x(\varphi) = x_S + \tfrac{1}{\kappa} \cdot (\sin\varphi - \sin\varphi_S)$$
$$y(\varphi) = y_S - \tfrac{1}{\kappa} \cdot (\cos\varphi - \cos\varphi_S)$$
$$z(\varphi) = z_S + \tfrac{1}{\kappa} \cdot \cot\vartheta \cdot (\varphi - \varphi_S) \qquad \text{path length } s(\varphi) = \tfrac{1}{\kappa} \cdot (\varphi - \varphi_S)/\sin\vartheta$$

The **track parameters** are defined as follows:

$[\Phi, z, \vartheta, \beta = \varphi - \Phi, \kappa]$	at $R = R_S$	LDT "barrel region"
$[x, y, \vartheta, \varphi, \kappa]$	at $z = z_S$	LDT "forward/rear region"
$[\frac{x}{10}, \frac{y}{10}, \frac{z}{10}; p_x, p_y, p_z]$		RAVE "6D Cartesian" (unit length = cm)