CORSIKA and SIM_TELARRAY – A package for the simulation of the imaging atmospheric Cherenkov technique and an investigation of important environmental parameters for such simulations

Konrad Bernlöhr

March 1, 1998
## Contents

1 Introduction ................................................................. 1

2 Features of the programs .................................................. 4
   2.1 The modified CORSIKA ............................................... 4
   2.2 The telescope simulation ............................................. 8

3 Usage of the modified CORSIKA ......................................... 16

4 Usage of sim_telarray ..................................................... 19

5 Data files and important parameters for sim_telarray .............. 23

6 Atmospheric transmission ................................................. 29

7 Lateral light distribution .................................................. 40

8 Image parameters ............................................................ 47
   8.1 Comparison of $\gamma$-showers with data ......................... 47
   8.2 Hadron showers ..................................................... 52

9 Reconstruction accuracy of shower direction .......................... 58

10 Magnetic field .............................................................. 64

11 Atmospheric models in the shower simulation ......................... 68

A Additional CORSIKA control keywords ............................... 71

B sim_telarray configuration parameters ................................ 72

C Internal structures of sim_telarray ...................................... 80

D Source code documentation .............................................. 87
E Data format of output from the modified CORSIKA program 88
E.1 A machine-independent hierarchical data format 88
E.2 Object types in the data file 89
E.3 Object formats 89
## List of Figures

2.1 Cherenkov light of a 300 GeV γ-shower ........................................ 5
2.2 Cherenkov light of a 1 TeV proton shower .................................... 6
2.3 Cherenkov light of a 5 TeV iron shower ........................................ 7
2.4 Structure of gamma and proton showers ........................................ 8
2.5 Structure of proton and iron showers .......................................... 9
2.6 Delay of Cherenkov photons as a function of core distance .......... 10
2.7 Davies-Cotton optics ................................................................. 11
2.8 Photon delay in Davies-Cotton design ........................................ 12
2.9 Gamma-shower as seen in a system of five telescopes .................. 14

5.1 Quantum efficiency and mirror reflectivity ................................ 24
5.2 PM signal shape at discriminator ............................................... 25
5.3 Photo multiplier amplification .................................................. 26
5.4 Single photoelectron response .................................................. 27
5.5 Fraction of pixels triggered ..................................................... 28

6.1 Atmospheric transmission of several absorber types .................... 30
6.2 Atmospheric profiles ............................................................... 31
6.3 Transmission table ................................................................. 32
6.4 Comparison of transmission curves from space .......................... 32
6.5 Comparison of transmission from different altitudes ................. 33
6.6 Comparison of transmission from different altitudes ................. 34
6.7 Profile of molecular and aerosol scatterers ................................ 34
6.8 Transmission from 10 km to observation level (tropical, navy) ..... 35
6.9 Transmission from 10 km to observation level (U.S. standard, rural) 36
6.10 Transmitted spectrum from 10 km altitude at 60° zenith angle .... 36
6.11 Ozone profiles ..................................................................... 37
6.12 Transmission in tropical and Tenerife model ............................ 37
6.13 Spectrum in tropical and Tenerife model .................................. 38
6.14 Transmission with volcanic activity ........................................... 39

7.1 Comparison of lateral distributions (100 GeV) .......................... 41
7.2 Comparison of lateral distributions (1 TeV) ............................... 42
7.3 Comparison of lateral distributions ........................................... 42
7.4 Lateral distributions for 0 m ..................................................... 43
Chapter 1

Introduction

Energetic cosmic-ray particles and $\gamma$-rays entering the atmosphere initiate particle cascades which are called extensive air showers (at least in that energy domain where shower particle reach ground level). Shower particles having a velocity faster than the speed of light in the medium air emit Cherenkov light. The imaging Cherenkov technique uses large optical telescopes and fast imaging devices (at present: cameras of many photomultipliers) to detect the showers by their Cherenkov light. If you are not familiar with the measurement technique, don’t read any further right now since the following text really assumes that you are familiar with it.

For the simulation of the Cherenkov light emission two programs are used within the HEGRA collaboration – the program by A. Konopelko et al. (using the ALTAI code) and the CORSIKA\textsuperscript{1} program [1, 2]. Due to its public availability, due to its rich set of interaction model options and due to the fact that it is used and tested by a large number of physicists, CORSIKA was chosen as the basis of the air shower simulation – despite the fact that the program by A. Konopelko et al. is known to be faster. The base version of CORSIKA, to which the necessary modifications were applied, was 5.20.

The CORSIKA Cherenkov option – developed mainly at the University of Madrid – was aimed at the simulation of the AIROBICC array of non-imaging detectors and requires a rectangular grid of rectangular detectors in a horizontal plane. Photons hitting these detectors are written to an output file for further processing. The photons are not written individually but as photon bunches for which the impact position in the detector plane, the $x$ and $y$ direction cosines, the arrival time (after the first interaction in the shower) and the number of photons (not necessarily an integer) are recorded. The wavelength is undetermined at that time – there is only a wavelength range specified in the configuration and the simulation at present does not take refraction into account. Absorption has to be applied afterwards.

This CORSIKA option was taken as the basis for the simulations described here and was extended such that arrays of telescopes at arbitrary $x, y, z$ positions in space could be simulated. As the basis for this part the code used to simulate arbitrary

\textsuperscript{1}CORSIKA is provided by D. Heck at the Forschungszentrum Karlsruhe.
arrays of tracking or scintillator detectors was taken. That code was originally developed to compare the performance of possible large arrays of CRT detectors to that of other existing, planned, and imaginative arrays of particle detectors. It was run with up to 100000 detectors simulated at the same time and, therefore, required an efficient scheme to decide which detector(s) was (were) hit by a shower particle.

This scheme was extended from a horizontal configuration with circular and rectangular detectors to a 3-D configuration of spheres. At that point the efficiency required an approximation that only Cherenkov light within a given angle (more than $10^\circ$) from the shower axis would be guaranteed to hit detectors out of the ‘detection level’ (as used in CORSIKA). For the anticipated purpose this should not be a problem at all. Photon bunches are now recorded if they pass within a specified radius from a detector (telescope) position. Since the time required for this part of the simulation is negligible compared to CORSIKA itself, a given array of telescopes is not only simulated once but many times with (horizontal) random displacements of each array. In contrast to the AIROBICC-style Cherenkov option where fixed (kind of quasi-random) positions were used, the IACT Cherenkov option generates different pseudo-random offsets for each shower. This way, showers with cores quite far from the telescope array can be easily taken into account.

For recording the photon bunches and other information not the usual (operating system, CPU type, and compiler dependent) CORSIKA data format is used but the eventio [3] format already used for all CRT data and the raw and DST data of the HEGRA telescope system. Item types 1200 to 1210 are defined for that purpose. The CORSIKA run and event header and trailer blocks are also included. These give all the necessary details about the simulated showers. Instead of writing all photon bunches as they emerge from CORSIKA, all bunches are sorted by telescope and array. They are written to the output file only after the simulation of a complete shower. This way, the detector simulation needs to treat only one telescope at a time.

The second step, the simulation of the detector response, is done by a second program called sim_telarray. While the first step is mainly a minor modification of CORSIKA plus an extra extension for the recording, sim_telarray is a completely new program. The sim_telarray program was developed as a test bed for some new ideas in the detector simulation. It is intended to be useful as a general simulation package for telescope arrays and, therefore, the main part of it is designed to be independent of any specific array. The default configuration, however, matches the existing (4-telescope) HEGRA system as closely as possible (or let’s say as good as we know it).

For testing the simulation results and for using them to enhance the analysis with the existing telescope system, the simulation results are converted to the HEGRA telescope system data structures. The resulting raw data can be written either in eventio or FPACK format. Since the existing formats cannot hold all the simulation-specific information the data is written like measured data (MC flag is off). Internally the actual set of HEGRA shower reconstruction software is applied and histograms and individual shower parameters are recorded.
The modified CORSIKA program and the telescope simulation have, among other things, been used to investigate the relevance of environmental parameters like atmospheric transmission, atmospheric density profiles, and geomagnetic field which are in many air-shower simulations either ignored altogether or treated in a very simplified manor. First results of these investigations are presented in this report. The rest of the report is dedicated to descriptions of many technical aspects of the programs and also to comparisons of simulation results to measured data from the present HEGRA stereoscopic system of four Cherenkov telescopes.
Chapter 2

Features of the programs

2.1 The modified CORSIKA

CORSIKA is a program for the simulation of extensive air showers. The physics included in the CORSIKA program is described in [1, 2]. An option for the simulation of Cherenkov light was introduced into CORSIKA by HEGRA collaborators at Madrid. This option was intended for the simulation of the AIROBICC array of non-imaging Cherenkov counters. The Cherenkov photons are not simulated one-by-one but in bunches, which is a (not necessarily integer) number of photons emitted from one place into the same direction. The index of refraction in this scheme has to be required to be wavelength-independent (and corresponds to actual values at 400 nm). No atmospheric refraction and no absorption or scattering is taken into account in this part of the simulation. By modifying the run-time parameters of this CORSIKA version, it can easily be used to write out all photons in a given area and also the particles arriving at ground level. Examples of the light distribution (without absorption etc.) of γ, proton, and iron showers is shown in figures 2.1, 2.2, and 2.3. Note that the co-ordinates in these figures are with respect to the nominal core position and that the charged primaries (in particular iron) are already deflected slightly in the geomagnetic field before the first interaction.

Since recording all photons in a large area would usually imply an extremely large amount of data per simulated shower and since the assumption of a flat detection plane does not fit very well with a system of Cherenkov telescopes in three dimensions, a different option for recording the Cherenkov photons with CORSIKA was developed. A telescope in this context is a fiducial sphere and all photon bunches entering such a sphere are to be recorded. Since even photons emitted very close to the telescopes should be fully taken into account, all telescopes are lifted, if necessary, such that the lower ends of all spheres are above the CORSIKA detection plane. For optimizing the task which telescopes are hit by a photon bunch, each sphere is projected along the shower direction onto the CORSIKA detection plane and enlarged somewhat. The detection plane is divided in a rectangular grid and each telescope (shadow) belongs to one or more grid elements. This way, the more
Figure 2.1: The Cherenkov light of a vertical 300 GeV $\gamma$-shower in a detection plane at an altitude of 2200 m (white corresponds to 80 or more photons/m$^2$ in the 300–450 nm wavelength range). No atmospheric absorption is applied.

Computationally expensive calculations are only needed for the telescopes associated with the grid element where a photon bunch arrives. Except for telescopes far from the detection plane and photon bunches arriving at very large angles with respect to the shower direction (larger than about $10^\circ$), all photons are nevertheless recorded. Since a telescope array can be simulated in multiple instances with random horizontal offsets, the fiducial volumes of telescopes can overlap and a photon bunch can be recorded more than once. Recording of all information is done with the (machine-independent) eventio [3] package. Photon bunches are already sorted by telescope (and telescope array in case of multiple instances) at that stage. This simplifies the stage of telescope and detector simulation to the extent that only one telescope needs to be simulated at a time.

Modifications to the actual CORSIKA code are kept as small as possible in order to allow easy adaptation to forthcoming new versions of CORSIKA and, perhaps, to allow implementing the interface with official CORSIKA versions without automatically distributing the whole simulation code. Most of the telescope-specific components for CORSIKA are contained in separate modules which have to be linked together with the normal CORSIKA modules. Apart from the normal action of recording pho-
Figure 2.2: The Cherenkov light of a vertical 1 TeV proton shower in a detection plane at an altitude of 2200 m (white corresponds to 80 or more photons/m² in the 300–450 nm wavelength range). In the lower panel a negative image with the impact points of electrons (including positrons) and of muons is included.
Figure 2.3: The Cherenkov light of a vertical 5 TeV iron shower in a detection plane at an altitude of 2200 m (white corresponds to 80 or more photons/m² in the 300–450 nm wavelength range). In the lower panel a negative image with the impact points of electrons (including positrons), muons, and protons is included.
Figure 2.4: An illustration of the structure a 300 GeV $\gamma$ and a 1 TeV proton shower. Displayed is the amount of Cherenkov light emitted along the particle tracks. Note the different scales for horizontal and vertical.

For normal production these options are not needed and should be disabled for performance reasons.

2.2 The telescope simulation

The major effort went into the simulation of the response of a system of Cherenkov telescopes with multiple mirror optics in Davies-Cotton [4] design or a variant of
Figure 2.5: An illustration of the structure a 1 TeV proton and a 5 TeV iron shower. Displayed is the amount of Cherenkov light emitted along the particle tracks. Note the different scales for horizontal and vertical.

it. Each telescope is assumed to have a photomultiplier camera of either square or hexagonal design, with equal sizes of all PMs. Although this is not the most general design\(^1\), it matches the current HEGRA system and also the present plans for the HESS system. Each pixel is connected to a discriminator logic and a Flash-ADC system. As far as the electronics design is concerned the current HEGRA system is implemented. Although the speed of the FADCs and the signal shapes can be easily changed in the configuration, the only available configuration is for the current 120 MHz system. A next-neighbour telescope trigger (as for the HEGRA system) is also included and the timing of telescope triggers is modelled up to the central trigger logic.

The simulation starts with random wavelength assignments for the photons of one bunch and a quick check if the photon has a real chance to be detected. For that purpose the atmospheric and telescope transmission probabilities are all multiplied, together with the largest quantum efficiency of any pixel at that wavelength.

The next part is the ray tracing of the multi-mirror optics and PM camera. Shadowing by the camera body is included explicitly but shadowing by other parts (masts etc.) is only included in the genuine telescope transmission above. All photons are

\(^1\)Whipple, for example, has larger pixels surrounding the main field of view.
Figure 2.6: The delay of Cherenkov photons of a vertical 1 TeV $\gamma$-ray shower at 2220 m altitude. The delay is counted with respect to the time at which the original $\gamma$-ray would have arrived at the observation level without any interaction.
Figure 2.7: Multiple mirror telescopes in Davies-Cotton design have spherical mirrors of focal length \( f \) (mirror radius \( 2f \)) mounted on a sphere of radius \( R = f \) around the focus.

checked to have their origin in front of the mirrors. Spherical mirrors are aligned in Davies-Cotton design (see fig. 2.7) with possible variations for performance optimization and with random mirror misalignments. Non-ideal reflection due to small-scale mirror deviations can be taken into account by additional Gaussian smearing of each reflected photon, but that is switched off by default since the quality of the HEGRA mirrors is generally better than the accuracy of the mirror alignment. Light funnels in front of the PMs are used to match the hexagonal or square entrance window for each pixel to the circular photocathode. The reflectance of the funnel coating is presently assumed to be wavelength and angle independent. All reflected rays are assumed to point to the photocathode, i.e. no multiple reflections in the funnel or light reflected out of the funnel again is taken into account (and should not be relevant for a well-designed funnel). The light travel time up to the pixel entrance is fully taken into account. The non-isochronous behaviour of the Davies-Cotton design is therefore fully taken into account. Although that is not very important in the case of the HEGRA telescopes (less than about 1.5 ns, see fig. 2.8), it can be a limiting factor for larger telescopes with smaller \( f/D \) (focal-length to diameter) ratio, for example like the Whipple telescope.

If the pixels quantum efficiency is less than that of the best pixel in the camera, a photon eventually finding its way to the photocathode has another chance to escape detection (note that the maximum quantum efficiency is taken into account as the very first step). In the other case, the resulting photo-electron can be assumed to find its way through the PM – with a random multiplication factor in the multiplier
Figure 2.8: Arrival of photons from a point source 2° from the zenith in the camera plane after they would cross a reference horizontal plane at the indicated positions (in units of cm). Each dot shows a photon. The circles represent the HEGRA mirrors. Arrival time ranges (in nanoseconds) are indicated by the colours on the right-hand side. Random mirror misalignments are taken into account in this plot.

and a random transit time jitter mainly for the path to the first dynode.

After the raytracing and detection decision the photo-electrons at each PM can be written to an output file instead of the photons, together with all input except the photons. This way, different electronics options can be easily compared without doing the raytracing again – as long as optics and camera layout do not change. In such subsequent simulations reading the photo-electron data, the camera ’sees’ always the same photo-electrons.

Apart from the Cherenkov photons from the air showers, random night sky light is also taken into account. Instead of simulating a spectrum of night sky photons and applying the detector efficiency to it, only a rate of night sky photoelectrons is given. Since the rate of photons is assumed to be constant over the camera field of view, the pixels with an above-average quantum efficiency see a higher rate of night sky photoelectrons and vice versa.

Since pixel high voltages are adjusted such that the measured signals from laser calibrations are about the same for all pixels, those pixels with a randomly better quantum efficiency will generally have a correspondingly lower multiplication for a single photoelectron. This is achieved by a lower voltage which in turn is accompanied by a longer PM transit time. Correlations of such PM properties are included in the simulation.

The measured response at the anode or dynode used for the readout is assumed
to be a linear combination of all the single-photelectron signals. Since signal shapes at the discriminator inputs and as digitized by the FADCs are different, both signal shapes have to be taken into account. It should be noted that measured signal shapes always include the transit time jitter and that the measured FADC signal shape for the HEGRA system (by M. Hess, see [5]) is even based on measurements of Cherenkov light from air showers. These intrinsic spreads thus have to be deconvolved from the measured pulse shapes before using them to configure the simulation.

Discriminators are nominally all operated at the same threshold voltage but actual threshold voltages scatter by a random offset. For coincident logic signals from the discriminators, each input signal above the actual threshold results in a high output signal for an effective gate length. Due to finite rise and decline times of the logic signals and thus the coincidences, the effective gate lengths (for assumed zero rise and decline times) will generally be somewhat smaller than the nominal gate length.

In the case for the HEGRA system, an effective length of 13 ns was suggested by G. Hermann for a nominal gate length of 16 ns. The logic discriminator outputs in the simulation are normally evaluated every 1/8 of an FADC time slice (and compilation options are present for a 1/16 or a 1/32 of it). Two pixels are assumed to trigger coincidentally when the logical AND of the resulting discriminator bit sequences is non-zero. Majority and next-neighbour trigger decisions are evaluated for each of the small discriminator time slices, resulting in an ultimate 0.3 ns resolution of telescope triggers for the present system (or less if the finer bin sizes are used at the expense of additional computations).

After the digitisation of all pixel signals and the evaluation of trigger decisions the data in memory is converted into the data structures used for the present HEGRA telescope system. Since the present implementation of Monte Carlo sub-structures is not well adapted to the CORSIKA program and this kind of telescope simulation, only the simulated experimental raw data is converted. This data can be recorded like ordinary raw data from the telescope system or it can be further processed through the pulse shape analysis and recorded in the FPACK format. As an example of this output see figure 2.9 which was obtained from the normal offline data display of the HEGRA telescope system.

For the purpose of this data conversion, a fixed date and time is used for all events – since the viewing direction of the telescope system is assumed as fixed for one simulation run. Right ascension and declination of the assumed source position are calculated such that they agree with the time and viewing direction. Essentially all configuration parameters of the simulated telescope system are converted to the format normally included in real HEGRA system data. This also includes the shaft encoder values corresponding to the viewing direction.

Further processing with the HEGRA shower analysis software is also included to achieve a full comparison of simulated and reconstructed shower properties. The reconstruction includes the shower direction, the core position and the primary energy (with an algorithm which is, admittedly, only for almost vertical showers). In addition, various parameters used in the selection of $\gamma$-rays are evaluated. Various
Figure 2.9: A simulated $\gamma$-ray shower as seen with a system of five imaging Cherenkov telescopes in a configuration like that of fully installed HEGRA system. Images drawn with the HEGRA offline data display program. The upper five panels show shower images as seen by the five telescopes, the lower left shows the combined view of shower images with the reconstructed shower direction (from a second-moments analysis), and the lower right shows the reconstructed core position from the intersection of projected image major axes on the ground.
histograms of simulated and reconstructed shower properties as well as N-tuples (in text format) are saved for further analysis.
Chapter 3

Usage of the modified CORSIKA

First of all, you need to process the modified CORSIKA source code through CMZ to extract the required code version for simulation of Cherenkov telescope arrays. For this purpose both the CERENKOV and IACT options have to be selected. Other options depend on the wanted interaction options and the machine architecture. Use the cmz_extract shell script for this step. The compilation is best done with the supplied Makefile. Note that this Makefile uses two make levels in order to include machine-dependent compiler options in one file. The syntax is

\[ \text{make TARGETS=corsika} \]

and should work on both the Ultrix (EUx) and the DEC Unix (gustav etc.) clusters. The configuration for Linux in the provided Makefile applies not to the ELIxx cluster but my machine at home and is using g77 (the GNU Fortran compiler). For non-standard wishes like producing shower images, histograms of lateral distributions etc. there are compile-time flags which may be set in iact.c before compiling.

To run CORSIKA make sure to have the following files available:

- EGSDAT2
- NUCNUCCS
- VENUSDAT

These files are from the standard CORSIKA distribution. If an atmospheric model other than the built-in model is used, you also need the files

- atm_profile_model_1.dat
- atm_profile_model_2.dat
- atm_profile_model_3.dat
- atm_profile_model_4.dat
- atm_profile_model_5.dat
- atm_profile_model_6.dat

which hold the atmospheric profiles of MODTRAN atmospheric models 1–6. These files are created by the program atm_models. If you created your own profile, give it a number above 6.
For running you should first prepare an input file (in the following called INPUTS). The following gives an example for generating 1000 vertical gammas between 200 GeV and 30 TeV:

RUNNR 11  
EVTNR 1  
NSHOW 1000  
PRMPAR 1  
ESLOPE -1.67  
ERANGE 0.2E3 30.E3  
THETAP 0. 0.  
PHIP -8.46 -8.46  
SEED 3137 713 0  
SEED 9193 711 0  
SEED 4931 737 0  
SEED 2220.E2  
ELMFLG T T  
RADNKG 200.E2  
ARRANG -8.46  
CERSIZ 5.  
CERFIL F  
CWAVLG 290. 670.  
CSCAT 25 40000. 0.  
TELESCOPE -1173 -2143 287 160  
TELESCOPE 1073 6568 -250 200.  
TELESCOPE 0. 0. 0. 200.  
TELESCOPE 6869 -1297 -531 200  
TELESCOPE -1958 -7175 402 200  
TELESCOPE -7010 320 719 200  
FIXHEI 0. 0  
FIXCHI 0.  
MAGNET 29.56 24.88  
HADFLG 0 0 0 0 0 0  
GHEISH T  
VENUS T  
VENSIG T  
ECUTS 0.3 0.1 0.020 0.020  
MUADDI F  
MUMULT T  
LONGI T 20. T  
MAXPRT 0  
ECTMAP 1.E6  
STEPFC 1.0  
DEBUG F 6 F 1000000  
VENDBG 0  
DIRECT ./  
TELFIL iact_s.dat  
EXIT  

The CORSIKA keywords are described in detail in the CORSIKA User’s Guide. New and modified keywords are described in appendix A. Unused keywords can be commented out with a ‘C’ as the first and a blank as the second character:
C DIRECT /dev/null

Now check that you have enough disk space and memory and you should be ready to start CORSIKA:

./corsika <INPUTS >& something.log

This notation assumes that you are using a C shell. Otherwise redirecting error output looks a bit different:

./corsika <INPUTS > something.log 2>&1

This can normally as well be run in the background. However, under Linux with gcc 2.7.2 and g77 0.5.18 it would not work in the background but dumps core.
Chapter 4

Usage of sim_telarray

The first step again is compiling. This should be easy if you have all necessary source code available:

```
make
```

The `sim_telarray` version produced by this will be able to write telescope raw data in eventio format. If you want to write also FPACK data use

```
make TARGETS=sim_telarray_fpack
```

In both cases the shower reconstruction and filling of histograms is enabled. To create a stripped-down version without shower analysis use

```
make TARGETS=sim_telarray_no_ana
```

There are more bells and whistles for compiling. Modify the Makefile if you want other options for diagnostic output, e.g. `-DDEBUG_TEST_ALL`. But that is the point where you should have a look into the source code.

For running `sim_telarray` you should have the following files available – assuming you don’t change the name of the configuration file on the command line or the other names either on the command line or in the configuration file:

- `imaging.cfg` General configuration file (optional)
- `imaging.cfg.ct1` Telescope-specific configuration (optional)
- `imaging.cfg.ct6` (depending on number of telescopes simulated)
- `atmo_trans.dat` Atmospheric transmission table. For La Palma it is recommended that this is a link to `model1_haze3_season1_vulcan0_alt2220.trans`
- `qe.dat` Quantum efficiency curve of photomultipliers.
- `mirror.dat` Mirror reflectivity curve.
- `spe.dat` Single photo-electron response functions (prompt and including delayed signal).
- `fadc_shape.dat` Shape of FADC pulse for single photo-electron.
- `disc_shape.dat` Shape of PM pulse at discriminator for single p.e.
For more information on these files see section 5. For a complete list of possible parameters in the configuration files see appendix B. In addition you might have a file with star positions, perhaps called stars.dat (by default no such file is read, the name needs to be configured). Such a file would consist of one line per star, giving the azimuth, altitude and flux of the star (the flux being roughly in units of photoelectrons per nanosecond in a typical PM).

Now having all the necessary files starting sim_telarray can be as simple as

\texttt{sim\_telarray}

or somewhat more complicated

\texttt{sim\_telarray -C telescope\_azimuth=180 -C nightsky\_background=all:0.06 \ -h test\_hdata iact\_*.dat}

The following command line options are recognized:

\texttt{-c fname} Use configuration from file \texttt{fname}.

\texttt{-h fname} Write histograms (in eventio format) to file \texttt{fname}. Conversion of these histogram files to HBOOK format is done with the \texttt{cvt2} program from CRT.

\texttt{-i fname} Use input file \texttt{fname}.

\texttt{-p fname} Write a couple of ASCII tables for plots to \texttt{fname}.

\texttt{-o fname} Write the output telescope raw data to file \texttt{fname}. This can be either in eventio or FPACK format, depending on compilation flags and configuration options.

\texttt{-l alpha} Set power law index of spectrum to \texttt{alpha}. By appropriate event weights the generated spectrum is corrected to the desired spectrum as far as the histograms are concerned.

\texttt{-C something} Interpret \texttt{something} as a configuration statement (like it is interpreted in the configuration file).

\texttt{-D flag} Define a flag or variable for the configuration preprocessor \texttt{pfp} which processes the main configuration file.

\texttt{-I path} Include files in the main configuration file are searched in \texttt{path}.

Any remaining command line arguments are interpreted as file names of input files containing photon bunches and other data in eventio format.

Note that the standard output is not just for diagnostic purposes. So you should better not redirect it to /dev/null but to a log-file on disk. After \texttt{sim\_telarray} is finished, you can extract ASCII N-tuples of shower parameters from the log file with

\texttt{egrep ^\@\log\_file | cut -c 4- > ntuple\_file}
For the meaning of the 99 columns see `sim_conv2ct.c` (prf means print floating point number, prd print integer):

```c
prf(array->shower_sim.energy); /* #1 */
prf(array->shower_reco.energy); /* #2 */
prf(array->shower_sim.azimuth); /* #3 */
prf(array->shower_reco.azimuth); /* #4 */
prf(array->shower_sim.altitude); /* #5 */
prf(array->shower_reco.altitude); /* #6 */
prf(array->shower_sim.xcore); /* #7 */
prf(array->shower_reco.xcore); /* #8 */
prf(array->shower_sim.ycore); /* #9 */
prf(array->shower_reco.ycore); /* #10 */
prf(sqrt(square(array->shower_sim.xcore) +
    square(array->shower_sim.ycore))); /* #11 */
prf(array->shower_reco.core_distance); /* #12 */
prf(array->shower_reco.m_scwid); /* #13 */
prd(ed.shwana.ntel); /* #14 */
prd(array->tel_triggered); /* #15 */
for (itc=2; itc<ntel; itc++)
{
    prf(elec[itc].telescope_triggered==0?0:1) +
    ((ed.shwana.tellist & (1<<itc))==0?0:2)); /* #16,34,52,70 */
    prf(elec[itc].photons_all); /* #17,... */
    prf(elec[itc].photons_detected); /* #18,... */
    prd(elec[itc].nn_triggered); /* #19,... */
    prd(elec[itc].triggered); /* #20,... */
    prf(elec[itc].significant_pixels); /* #21,... */
    prf(array->shower_reco.amplitude[itc][0]) /* #22,... */
    prf(array->shower_reco.amplitude[itc][1]) /* #23,... */
    prf(array->shower_reco.width[itc]); /* #24,... */
    prf(array->shower_reco.length[itc]); /* #25,... */
    prf(array->shower_reco.dis[itc]); /* #26,... */
    prf(array->shower_reco.miss[itc]); /* #27,... */
    prf(array->shower_reco.conc[itc]); /* #28,... */
    prf(array->shower_reco.azwidth[itc]); /* #29,... */
    prf(array->shower_reco.azlength[itc]); /* #30,... */
    prf(array->shower_reco.alpha[itc]); /* #31,49,67,85 */
    prf(array->shower_reco.tel_core_distance[itc]); /* #32,50,68,86 */
    prf(sqrt(square(array->shower_sim.xcore-0.01*array->xtel[itc]) +
        square(array->shower_sim.ycore-0.01*array->ytel[itc]))) /* #33,... */
}
/* Add more columns without losing backward compatibility */
prf(array->shower_reco.theta); /* #88 */
prf(array->shower_reco.theta*array->shower_reco.theta); /* #89 */
prf(array->shower_sim.core_dist_3d); /* #90 */
prf(array->shower_reco.core_dist_3d); /* #91 */
for (itc=2; itc<ntel; itc++)
{
    prf(array->shower_sim.tel_core_dist_3d[itc]); /* #92,94,96,98 */
    prf(array->shower_reco.tel_core_dist_3d[itc]); /* #93,95,97,99 */
}
```
I hope that (most of) the variable names are clear enough to understand their meaning. The variable itc is presently only running from 2 to 5 (for CT3 to CT6). Energies are in TeV, distances in meters, angles in degrees. Columns 88 and 89 are the mismatch $\theta$ between actual and reconstructed shower direction (in degrees) and its square $\theta^2$, respectively. Note that these N-tuples include only detected showers while many of the histograms are also filled for the showers not triggering the telescope system. When histogramming from the N-tuples the event weight correction has to be applied 'by hand'. As an example of such histogramming the following Gnuplot input line can be used (as one long line instead of with line break as shown):

```
    gnuplot> plot '<selop 1 "**" -0.93 13 | selop 2 + 0. 1 |
                      whistogram 300 0. 3.' w hist
```

The program `selop` is a simple utility for calculations between two columns or between one column and a constant plus copying other columns unmodified to output, `whistogram` is a simple program to add up data lines with column one as the abscissa, column two as an additional weight into histograms of given number of bins and limits. In the above example column 1 (primary energy $E$) is first taken to power $-0.93$ (correcting from $E^{-1.67}$ to $E^{-2.6}$ power law) and `mscw` (column 13) is copied as-is, then columns are reordered (first `mscw` and then the event weight $E^{-0.93}$), all resulting lines are added up in a histogram of 300 bins with $0. < \text{mscw} < 3.$ and Gnuplot should plot this histogram with the histogram step plotting style. This is how many of the event plots in this report were produced (before exporting to Fig format and adding labels and comments with `xfig`).
Chapter 5

Data files and important parameters for sim_telarray

If the general configuration file (by default `imaging.cfg`) is not present, the compiled values are used by default. For telescopes, where no telescope-specific configuration file is found, the values from the general configuration file or the compiled values apply. Command-line configuration values override compiled values and values from the general configuration file but not those from the telescope-specific files (since the command line options cannot address individual telescopes). Note that the following configuration (either in the default configuration file or on the command line) is required to reproduce the current 4-telescope system from simulations where CT1 through CT6 were included:

```
ignore_telescopes=1,2
```

This was used in the general configuration file to simulate the present HEGRA 4-telescope system. If you have this in your general configuration file but want to run once with a five telescope system by overriding the statement on the command line (see page 20) make sure to override both numbers (-1 or any other non-existing telescope number to stop ignoring a telescope; see appendix B for details):

```
sim_telarray -C ignore_telescopes=1,-1
```

In the present simulations the following telescope-specific configuration parameters have been used for (CT3 to CT6):

- **CT3**: adjust_gain 1.13
- **CT4**: adjust_gain 0.99
- **CT5**: adjust_gain 1.00
- **CT6**: adjust_gain 1.00
Figure 5.1: Quantum efficiency and mirror reflectivity curves as used in the simulation of the HEGRA telescope system.
These values are motivated by measurements of the conversion factors by M. Hess (and are relative to the nominal values for FADCs, discriminator, and DC current).

Apart from the default transmission table, there is a large set of other such tables available, just in case you wanted to experiment with it (see section 6). The used quantum efficiency and telescope reflectivity curves are shown in figure 5.1. Note that the shading by the camera body is included in the ray-tracing and the extra effect of the plexiglass cover and the reflectivity of the lightguide is taken into account by additional (so far wavelength-independent) parameters.

Note that the supplied `fadc_shape.dat` was generated by deconvolving the intrinsic photon arrival times in showers and the photo-electron transit-time jitter from the measured tables provided by M. Hess (deconvolution done with the DE-CONVOLVE/IMAGE command in MIDAS\(^1\)). It assumes that the HEGRA 120 MHz FADC system is used. The transit-time jitter was also deconvolved from the signal shape at the discriminator as taken from [6] (except that a shorter signal tail is used for efficiency reasons, see figure 5.2).

\(^1\)MIDAS is provided by the European Southern Observatory.
The single photo-electron response functions are based on simulations with Poisson-law multiplication plus background. These simulations have as the important ingredients the multiplication-versus-voltage curves of various dynode materials, the voltage resistor chain settings of the HEGRA telescope system PMs, and the approximate total multiplication of these PMs. Actually none of the literature curves [7] gave a very satisfactory match of the known multiplication at the typical total operating voltages of the PMs at La Palma (see figure 5.3) and the curve used is actually in-between the nominal CuBeO curve and higher-gain dynode materials, corresponding to a first-dynode gain of about 6.4. The width of the measured response functions (at higher than normal voltage) is quite well reproduced if appropriate gains (first dynode gain between 7 and 8) are taken into account.

There are two kinds of backgrounds added. The ion-induced background which has a delay much longer than the integration time is added only to night-sky photo-electrons (where it really does not matter if the signal is prompt or delayed). A much lower background is added for the ‘signal’ photo-electrons. Both kinds of backgrounds have been attempted to match the measurements by C. Köhler [8] and T. Kutter [9] but, unfortunately, all available measurements were made with resistor chains 2R-1.5R-R-...-R instead of 2R-R-R-...-R and with substantially larger voltages (and gains).
than used on La Palma.\textsuperscript{2} The default response functions in \texttt{spe.dat} (one column for ‘signal’ photo-electrons and one for nightsky) are missing one feature of the measured (and literature) response functions: a rise towards very small amplitudes seemingly originating from inelastic scattering (instead of multiplication) of electrons on the first dynodes. There is an alternate file \texttt{spe\_rise.dat} that tries to include this feature on an ad-hoc basis (available data at low amplitudes is not good enough to check that). See figure 5.4 for the shapes of these tables.

In the compiled configuration of \texttt{sim\_telarray} (see section B for full listing) there are a number of parameters which have been adjusted to the appropriate values for the current (4-telescope) HEGRA system on La Palma. Among these parameters there are many which are rather simple to tell (like the focal length, the positions of all mirrors, the number of pixels, their sizes and so on) while others are less well known. There turned out to an internal inconsistency between two important calibration parameters which were separately measured. These two parameters are the FADC sensitivity in terms of amplitude per photoelectron (unit: FADC counts at the peak), taken here to be 0.95 (which is the inverse of the more popular 1.05 p.e. per count value), and the signal amplitude in mV at the discriminator inputs (previously assumed to be about 1.15 mV/p.e. Consistency of these parameters requires that the fraction of pixels with some reconstructed FADC amplitude which were among the triggered pixels agrees between data and simulations. In order to achieve the

\textsuperscript{2}I would conclude that such measurements should be repeated with the actual resistor chain and gain.
Figure 5.5: The fraction of pixels (as a function of the reconstructed FADC amplitude with the `scadc` pulse reconstruction) which had the trigger bit in the discriminator/majority logic set. The simple simulation (where each pixel with more than a given signal in terms of photoelectrons – after applying the response function – is triggered) has a noticeably faster rise than the detailed simulation where signal shapes at the discriminators have been accumulated.

required consistency (see figure 5.5), the first parameter was kept and the second was adjusted to 0.94 mV/p.e. Since the ratio of the 0.95 FADC counts per p.e. to the 0.94 mV per p.e. is only related to the internal gains in the electronics, any change in one parameter has to be related to a proportional change in the other (which is most easily achieved by the `adjust_gain` configuration parameter). A modified gain also results in a change of the DC current without any change of the rate of nightsky photoelectrons (but note: the present calibration of DC current to nightsky is only an estimate and could be improved).
Chapter 6

Atmospheric transmission

The atmospheric transmission/absorption is one source of concern for the energy calibration and to some (though smaller) extend also for the image parameters. The absorption is made up by absorption bands of several molecules (in particular O$_3$ in the relevant wavelength range), by molecular (Rayleigh) scattering, and by scattering and absorption on aerosols. For a detailed introduction into this topic see [10].

Figure 6.1 shows the respective relevance of the most important UV and optical absorbers as a function of wavelength. In all the following, complete absence of clouds is assumed. Apart from clouds, the amount of aerosols (and also their composition, vertical structure etc.) is variable,\(^1\) as well as the concentration of ozone and water vapour. It is, therefore, necessary to find an absorption model relevant for the site to which the simulations apply.

A standard program for calculation of atmospheric transmission and radiance is MODTRAN.\(^2\) MODTRAN has six model atmospheres built in and allows to specify user-defined models. The built-in models are tropical, mid-latitude summer, mid-latitude winter, sub-arctic summer, sub-arctic winter, and U.S. standard atmosphere 1976. Apart from the basic atmospheric model, there is a range of options to determine the amount, composition and vertical structure of aerosols. These include several options for haze (urban, rural, maritime, desert, ...) and for stratospheric dust (volcanic, meteoric dust). Absorption bands are included at moderate resolution (which means well below a nanometer). For scattering processes multiple scattering can be included, but this is not considered important here, because scattered light accounts (except for large zenith angles) only for a small fraction of the total light and because Cherenkov light scattered by an appreciable angle will arrive outside the short integration time used in the imaging atmospheric Cherenkov technique.

For a first look at the relevant general atmospheric models see figure 6.2 with a comparison of the MODTRAN models with measured atmospheric profiles over Tenerife. The best matching model for Tenerife (and due to its proximity also La Palma) is the tropical model. Of course, the vertical profile of the atmosphere is not

\(^1\)The number of aerosol particles per cm$^3$ can be as high as a few millions in major cities or as low as 10 in Greenland during winter.

\(^2\)MODTRAN is patented software released by the U.S. Air Force Phillips Laboratory.
only relevant to the absorption but also for the development of the shower itself and (via the index of refraction) for the amount of Cherenkov light produced. This is discussed in section 11.

For the purpose of the simulation described here, tables of (direct) vertical atmospheric transmission from several starting altitudes to a given observation altitude were obtained with the help of MODTRAN. Instead of a manual setup of the complicated input files for MODTRAN an automated procedure was set up:

loopping [-model 1-6] [-haze 0-10] [-season 0-2] [-vulcan 0-8] [-zenith 0-90] [altitude 0-100]

This procedure sets up the input files for a series of MODTRAN runs, executes MODTRAN and extracts relevant output data. For the later purpose looping calls three small programs (extr, selop, and nano). The transmissions are averaged over wavelength bins of one nanometer. For the further simulation only tables generated with zenith angle 0° are used. Slanted paths are easily obtained in the plane-parallel atmosphere approximation. A resulting transmission table is shown in figure 6.3. The selected transmission table is applied at the detector simulation stage (i.e. in sim_telarray).

Figure 6.4 shows a comparison of the transmission curves for a few models from space to ground (at altitude 2.22 km). Also shown is the curve obtained with the
Figure 6.2: Atmospheric profiles (top: temperature, bottom: pressure scaled by \(\exp(h/7\ \text{km})\)) for the MODTRAN models and measurements by balloons started from Tenerife island. Data kindly provided by Dr. E. Cuevas from the Izana Global Atmospheric Watch (GAW) Observatory, Santa Cruz de Tenerife.
Figure 6.3: An example of a transmission table as obtained with MODTRAN. The observation level $H$ is at an altitude of 2.22 km.

Figure 6.4: A comparison of a few transmission curves from space ($H=100$ km) to ground (2.22 km). The point marks the average transmission measured with the Carlsberg Meridian telescope on La Palma. The two U.S. standard atmosphere curves are essentially indistinguishable as the lower curve.
Figure 6.5: Comparison of the MODTRAN tropical navy maritime (summer) with the Hemberger/Konopelko transmission for different altitudes.

code from M. Hemberger and A. Konopelko. The average visual (about 550 nm) transmission at the Carlsberg Meridian telescope on La Palma is about 0.85 (about 0.17 magnitudes extinction). This is in good agreement with the navy maritime haze models (haze=3, with default values of air mass character and wind speed) and with the absorption model by Hemberger and Konopelko. The absorption in the maritime (haze=4) and rural (haze=1) haze models is somewhat larger than observed. The tropical model with maritime haze is about halfway between U.S. standard atmosphere with maritime haze and tropical atmosphere with navy maritime haze. For sim_telarray the U.S. standard atmosphere with maritime haze model (for summer) was previously used. For best agreement with the Carlsberg data, the tropical atmosphere (model=1) with navy maritime haze model (haze=3) is now recommended.

Although the Hemberger/Konopelko transmission curve from space is in good agreement with the tropical navy maritime model there are noticeable deviations for light emitted in the atmosphere. This is illustrated in figures 6.5 and 6.6. The reason appears to be the following: In the Hemberger/Konopelko transmission there is a combined absorption coefficient for Rayleigh and Mie scattering. This coefficient is matched to observed extinction to space (after correction for ozone absorption). For Rayleigh scattering the optical depth is simply proportional to the amount of air traversed. In the present Hemberger/Konopelko transmission this is applied to

---

The transmission parameters from A. Konopelko apply only to the 300–600 nm wavelength range. For wavelengths above 700 nm transmissions beyond 100% could result. The short-wavelength transmission (below 250 nm) as shown in [11] is also completely wrong!
Figure 6.6: Ratio of MODTRAN tropical navy maritime (summer) to Hemberger/Konopelko transmission for different altitudes.

Figure 6.7: Vertical atmospheric profiles of the amount of scattering material for molecular scattering (blue line, proportional to density) and aerosol scattering for different MODTRAN aerosol models (all for U.S. standard atmosphere 1976). The red line is for a rural model with a sea level visibility of 5 km, the green (dashed) line for a desert model with a sea level visibility of 57 km, and the brown line for a rural model of 23 km visibility with an additional extreme amount of fresh volcanic dust.
the combined Rayleigh+Mie scattering, implying that aerosols are distributed proportional to air density. Actual distributions of aerosols are larger at low altitudes and smaller at high altitudes. This is illustrated in figure 6.7 for several MODTRAN aerosol profiles. The net effect is a 4-7% larger amount of Cherenkov light at the relevant wavelengths (around 400 nm) as compared to the MODTRAN tropical navy maritime model.

The impact of the observation level altitude on the fraction of transmitted Cherenkov light is shown in figures 6.8, 6.9, and 6.10. It is evident that a high observation altitude is particularly important for large-zenith-angle observations. On the other hand, the difference between an altitude of 2200 m and one of 1800 m is less than 5%.

It has been noticed by M. Hemberger that the ozone concentration measured in the lower troposphere over Tenerife substantially exceeds standard values. This is also true for all the model atmospheres in MODTRAN (see figure 6.11). The general ozone profile is in good agreement with the tropical model. In order to check for the effect of increased tropospheric ozone, a user-defined atmospheric model was set up for MODTRAN with Tenerife average measured values for all parameters where data was available (pressure, temperature, water vapour, ozone below 32 km altitude) and taking the tropical model otherwise. Transmission for the 'Tenerife' and tropical models is compared in figure 6.12. The resulting Cherenkov light spectra are compared in figure 6.13.

A further potential effect that can be taken into account with MODTRAN is the stratospheric extinction which can be affected by volcanic activity somewhere in the
Figure 6.9: The fraction of vertically transmitted Cherenkov light from an emission point at 10 km altitude to different observation altitudes $H$. A U.S. standard 1976 atmospheric profile and rural haze (with horizontal visibility at sea level of 23 km) was assumed.

Figure 6.10: The transmitted Cherenkov spectrum from an emission altitude of 10 km to different observation altitudes for $60^\circ$ zenith angle (photons per unit wavelength normalized to the unabsorbed spectrum at 400 nm).
Figure 6.11: Atmospheric profiles of ozone as measured over Tenerife and as included in MODTRAN.

Figure 6.12: Atmospheric transmissions from different altitudes for the tropical and Tenerife models.
Figure 6.13: Cherenkov spectrum from different altitudes for the tropical and Tenerife models.

world. Large volcanic eruptions can affect stratospheric extinction for months – in extreme case effects over years have been observed. The effect of different volcanic activity is illustrated in figure 6.14.

A more frequent case of variable atmospheric transmission on La Palma is due to Sahara desert dust. That, of course, is not easily modelled. Records of dust extinction of star light are, nevertheless, available for La Palma.
US standard atmospheric profile with different models of stratospheric extinction

Figure 6.14: Atmospheric transmission for different levels of volcanic dust extinction in the stratosphere.
Chapter 7

Lateral light distribution

The lateral distribution of Cherenkov light in both its absolute intensity and its shape is a simple way to compare different simulations since it is frequently published and does not require any detector simulation. More historic calculations of the lateral light distribution differed quite substantially (see figures 7.1 and 7.2). Although sea-level altitude is a frequent test-case it is not at all typical for real Cherenkov telescopes and – since it is below the inversion layer – is much more subject to haze extinction than altitudes above the boundary layer. As a reference of the present simulations figure 7.3 shows the lateral distribution for the HEGRA site, assuming a U.S. standard atmosphere with maritime haze (same as for the other lateral distributions here). Keep in mind that in the mean-time I found out that the transmission in this model is about 2-3% too small.

An extended comparison of lateral light distributions for the present CORSIKA simulation (again with U.S. standard atmosphere) and simulations by A. Konopelko [11] is shown in figures 7.4, 7.5, 7.6, and 7.7. Unfortunately the simulations by A. Konopelko were only done with a few showers and, therefore, are subject to substantial shower-by-shower fluctuations. The general impression is that (at least for gammas where the fluctuations are not so severe) there is good agreement in the profile and that there is a noticeable difference in intensities at low altitudes. The difference in intensity is connected to the already mentioned atmospheric absorption. At the higher altitudes there appears to remain a difference in the brightness of the proton showers. This could be the result of different hadronic interaction models\(^1\) which can easily lead to differences of the order of 10–20%. This intrinsic uncertainty due to the interaction models ultimately limits the accuracy of an energy calibration which can be achieved by matching the trigger rate of hadronic showers to observations – not to mention the present limited accuracy of the absolute cosmic-ray flux at TeV energies. Note: For the present simulations the trigger rate of vertical hadronic showers with the 4-telescope HEGRA array is about 17 Hz for a nightksy background of 0.049 p.e./ns and about 19 Hz for a background of 0.07 p.e./ns (Galactic plane). Although the 17 Hz are consistent with trigger rate measurement by H. Lampeitl,\(^2\)

\(^1\)The present simulations used the VENUS/GHEISHA interaction options of CORSIKA.
Figure 7.1: Comparison of lateral Cherenkov light distributions for several historic calculations and the present simulations. All calculations are for vertical gammas of 100 GeV at sea level. Present calculations (red lines) are overlaid on a published compilation [12].

the usual event rate is rather 15 Hz. Considering all the uncertainties (calibration constants, cosmic-ray flux, interaction model), the agreement is quite good.
Figure 7.2: Comparison of lateral Cherenkov light distributions for several historic and recent calculations and the present simulations. All calculations are for vertical gammas of 1 TeV at sea level. Present calculations (red lines) are overlaid on a published compilation [13].

Figure 7.3: Lateral distribution of Cherenkov light for the present simulation at the altitude of the HEGRA site.
Figure 7.4: Overlay of several sets of $\gamma$-shower lateral distributions at sea level of A. Konopelko [11] (black solid lines) and the present simulations (coloured lines), all after atmospheric absorption. At this low altitude the present simulations have a substantially larger atmospheric absorption.
Figure 7.5: Overlay of proton shower (black dashed lines: A. Konopelko, light blue lines: present simulations) and γ-shower lateral distributions (black solid lines: A. Konopelko, other coloured lines: present simulations), again after atmospheric absorption. Although differences are smaller than at sea level, they are still noticeable.
Figure 7.6: $\gamma$-shower lateral distributions (as in figure 7.4) at 3500 m. Differences are even smaller than at 2220 m.
Figure 7.7: Proton and $\gamma$-shower lateral distributions (as in figure 7.5) at 5500 m. The $\gamma$-showers agree very well (except at 1 TeV which seems to be a shower fluctuation) but all proton showers are brighter with the present simulations.
Chapter 8

Image parameters

8.1 Comparison of γ-showers with data

A crucial test of the simulations is how well the image parameters used for the selection of γ-showers are reproduced. This test is – among other things – sensitive to the imaging quality of the telescopes but also to the nightsky background. The advantage of the γ-rays are the narrow and regular images and the fact that the physics involved in the shower development is well understood.

For this purpose the Markarian 501 data from the periods 61–65 (summer 1997, between mirror adjustments and the fire at the HEGRA site) were taken. The zenith angle selection used just the zenith angle at the start of runs and no correction was made to have precisely the same average zenith angle as in the simulations (at fixed zenith angles). Nevertheless figures 8.1, 8.2, 8.3, and 8.4 show that the agreement is very good. Note that the scaling of the image width uses the old parametrisation developed before the readjustment of the mirrors in 1997 and, therefore, has an average mean scaled width below 1.

For zenith angles up to 30° each simulated shower was sampled 25 times at random horizontal core offsets up to 400 m, at 45° and 60° each shower was sampled 40 times at core offsets up to 500 m. The viewing direction of the telescope system is 0.5° from the shower direction (corresponding to the wobble mode observations).

The nightsky background used for these simulations is the nominal background of 0.049 photoelectrons per nanosecond and pixel. The image width is increased by about 5% when the background is doubled. The best match of the number of pixels above the zero-suppression threshold – which is a sensitive measure of the amount of nightsky noise – is achieved for about 0.055–0.6 p.e./ns in the case of the Mrk 501 field.
Figure 8.1: Histograms of the image parameters width, mean scaled width, and length for γ-rays of zenith angle 15° in simulations (blue line) and of Mrk 501 measurements (ON–OFF) with zenith angles between 10° and 20° at the beginning of runs. Only data from the measurement periods 61–65 (summer 1997) have been used. Simulations are normalized to measured data.
Figure 8.2: Histograms of the image parameters width, mean scaled width, and length for γ-rays of zenith angle 30° in simulations (blue line) and of Mrk 501 measurements (ON—OFF) with zenith angles between 25° and 35° at the beginning of runs. Only data from the measurement periods 61–65 (summer 1997) have been used. Simulations are normalized to measured data.
Figure 8.3: Histograms of the image parameters width, mean scaled width, and length for γ-rays of zenith angle 45° in simulations (blue line) and of Mrk 501 measurements (ON–OFF) with zenith angles between 40° and 50° at the beginning of runs. Only data from the measurement periods 61–65 (summer 1997) have been used. Simulations are normalized to measured data.
Figure 8.4: Histograms of the image parameters width, mean scaled width, and length for γ-rays of zenith angle 60° in simulations (blue line) and of Mrk 501 measurements (ON–OFF) with zenith angles between 52° and 62° (average about 56°). Only data from the measurement periods 61–65 (summer 1997) have been used. Simulations are normalized to measured data.
8.2 Hadron showers

Hadron showers are far more abundant than the $\gamma$-showers. For the comparison with the simulations it is actually sufficient to use a few runs. Simulations were done for proton, helium, nitrogen, magnesium and iron showers of zenith angles up to $5^\circ$ and telescopes were simulated at zenith angles of $0^\circ$ and (for more efficient usage of simulated showers) $1.5^\circ$. The VENUS and GHEISHA interaction option were selected (for interactions above and below 80 GeV per nucleon, respectively).

Unfortunately, simulations are for the zenith but no measurements so close to zenith were available. The data which was readily available was from the Galactic plane scan. In that case the nightsky background is significantly larger than in the case of the Mrk 501 field and, in particular, the nightsky light is not homogenous over the field of view (because of stars and star clusters). The measured data has an average zenith angle of about $9^\circ$ and reasonable agreement of the number of pixels above zero-suppression (see figure 8.5) is achieved for a nightsky background of 0.073 p.e./ns (assumed as homogenous).

The measurements and simulations of hadronic showers are potentially interesting for the energy-calibration of the telescope system. For that purpose and also for comparing image parameters it is crucial to simulate the mixed composition of cosmic rays. The flux and spectral index parameters fitted by B. Wiebel to directly measured composition data were used for five groups of elements and applied to the simulations of protons, helium, nitrogen (for the CNO group), magnesium (for Ne–S) and iron (for $Z > 17$). As a first check it is important to see that the relevant ranges of core positions and angles are really fully covered. This is shown in figures 8.6 and 8.7.

Figure 8.8 shows the width and length image parameters of the hadron showers in the simulations compared to the measured data from Galactic plane fields at zenith angles of about $9^\circ$. The agreement is not so perfect as for the $\gamma$-showers but it should be kept in mind, that the zenith angles of simulations and measurements do not agree very well and that a homogeneous nightsky background was assumed in the simulations. Miscellaneous other image parameters are compared in figure 8.9.
Figure 8.5: Histograms of the number of significant pixels (above the zero-suppression threshold) for measurements in the Galactic plane (dashed line) and hadron shower simulations. Because in the case of locally triggered telescopes only 12 FADC bins are checked to be above threshold but in the case of not locally triggered telescopes 16 bins are checked, the number of read-out pixels is larger in the case of not locally triggered telescopes. Increased nightsky background (or electronics noise) leads to an increased number of significant pixels. The wide and asymmetric shape of the measured histograms is presumably due to the inhomogenous distribution of the starlight.
Figure 8.6: Distribution of core offsets of hadron showers triggering the present HEGRA 4-telescope system in simulations of the mixed composition (black) and protons only (blue). Note that due to the shallower lateral light distribution of showers from heavy elements, less than half of the showers at core offsets beyond 250 m are induced by protons. The estimated fraction of showers beyond the 400 m limit is below 1%. 
Figure 8.7: The fraction of hadron showers triggering the present HEGRA 4-telescope system in simulations of the mixed composition as a function of the angle between the shower direction and the viewing direction of the telescopes (triggered showers divided by all simulated showers). The overall scale of the fraction is arbitrary. Note that outside the field of view the triggered fraction of showers declines exponentially (by about 0.7 decades per degree). This decline is about the same for all primaries (not shown) but the flat part is somewhat wider for the heavy primaries. The estimated fraction of showers outside the simulated $0^\circ < z < 5^\circ$ range is again well below 1%. 

Angle between shower and viewing direction [°]
Figure 8.8: Width and length parameter distribution of hadron showers as measured (red lines) and as simulated (blue and black lines).
Figure 8.9: Orientation and position dependent classical image parameters for hadron showers (mixed composition), in this case assuming a source candidate at the image centre.
Chapter 9

Reconstruction accuracy of shower direction

The accuracy of the reconstructed shower direction (for example in the case of Markarian 501, see figure 9.1) depends on several ingredients. The first one is the reconstruction of the individual telescope image parameters. The second is the reconstruction of the core position and the third is how the telescope images are combined to the shower direction. As a first approximation one could assume the same Gaussian error in each projection of the shower direction for all showers. That would correspond to an exponential in histograms of the squared space angle $\theta^2$ between point source position and reconstructed direction. Actual tails in the resulting histograms are above the exponential form and are consistent between data and simulations (see figures 9.2, 9.3 and 9.4).

The integrated distributions in terms of the fraction of detected $\gamma$-showers being reconstructed within a given cone half-opening angle $\theta$ is useful to obtain the actual number of showers from the ON–OFF counts. These integrated distributions are quite sensitive to the zenith angle (fig. 9.5), to the number of telescopes with useful images for the analysis (fig. 9.6) and to any core distance, amplitude or energy cuts (fig. 9.7).

For a comparison with data, the fraction of real $\gamma$-showers falling into the OFF region has to be taken into account in general. For small zenith angles this is only a small correction, being about 1% for the integrated event numbers within $0^\circ < \theta < 0.5^\circ$ and is negligible for $\theta < 0.3^\circ$. This correction has not been applied to figures 9.2, 9.3 and 9.4 but to figure 9.8 showing the $\theta^2$ distribution for simulated and measured showers at large zenith angles.
Figure 9.1: The reconstructed shower directions of showers from the direction of Markarian 501 (data obtained with the HEGRA 4-telescope system in 1997). A mean scaled width below 1.0 is required here to suppress the hadronic background.
Figure 9.2: Histogram of the angle between source direction and reconstructed shower direction squared ($0 \leq \theta^2 \leq 0.5 \text{ deg}^2$) for measured $\gamma$-showers from Mrk 501 (ON-OFF, symbols with error bars) and for simulations (blue solid line, normalized to measured data). No cuts applied.

Figure 9.3: Histogram of the angle between source direction and reconstructed shower direction squared ($0 \leq \theta^2 \leq 0.1 \text{ deg}^2$) for measured $\gamma$-showers from Mrk 501 (ON-OFF, symbols with error bars) and for simulations (blue solid line, normalized to measured data). No cuts applied.
Figure 9.4: Histogram of the angle between source direction and reconstructed shower direction squared ($0 \leq \theta^2 \leq 0.1 \text{ deg}^2$) for measured γ-showers from Mrk 501 (ON-OFF, symbols with error bars) and for simulations (blue solid line, normalized to measured data). A cut of a mean scaled width below 1.0 has been applied. For comparison the simulated showers failing the cut, i.e. with mscw > 1.0 are also shown. The direction of these broader showers is generally less well reconstructed than of the narrow showers.

Figure 9.5: The integrated distribution of $\theta^2$ in simulated showers for zenith angles of 15°, 30°, 45°, and 60° (fraction of showers within given half opening angle $\theta$).
Figure 9.6: Integrated $\theta^2$ distribution for all detected showers and for all showers where images from 2, 3, or 4 Cherenkov telescopes could be used for the shower reconstruction. It is evident that the two-telescope showers (mainly low-energy) are rather poorly reconstructed.

Figure 9.7: Integrated $\theta^2$ distribution for all 'detected' showers in the simulation (black), for showers where the energy reconstruction succeeded (blue), and for showers with reconstructed energies above 1 TeV (green) and above 3 TeV (red). A cut on reconstructed core distance below 200 m was applied.
Figure 9.8: Histogram of the angle between source direction and reconstructed shower direction squared \((0 \leq \theta^2 \leq 0.25 \text{ deg}^2)\) for measured high-zenith-angle \(\gamma\)-showers from Mrk 501 (ON-OFF, symbols with error bars) and for simulations (blue solid line, normalized to measured data). A cut of a mean scaled width below 1.0 has been applied.
Chapter 10

Magnetic field

The geomagnetic field turns out to have an impact on the amount of light found in the ‘disc’ (the flat region of the lateral light distribution) of several percent. This is shown in figure 10.1 for the magnetic field configuration at La Palma with showers from different azimuth angles. Without magnetic field the amount of light is larger by about 9% than with the actual magnetic field (in the inner part). With La Palma magnetic field the light yield is largest (by 2–3%) for showers from the south (shower direction parallel to magnetic field). This rather small difference has a dramatic effect at the detection threshold on the core distribution of showers triggering the telescope system. This is illustrated in figure 10.2 for an assumed magnetic field of 0.3 G at 45° zenith angle.

Even for these showers at the detection threshold the impact on the image parameters is rather minute (see figure 10.3), perhaps with an indication of slightly narrower and longer images of showers parallel to the magnetic field (from south in this case) than for showers perpendicular to the magnetic field.
Figure 10.1: Lateral distributions of Cherenkov light from \( \gamma \)-showers of 30° zenith angle and different azimuth angles (two independent sets of simulations to show the statistical relevance of the effect).
Figure 10.2: Core distance distributions of 1 TeV $\gamma$-ray showers from south and north, respectively, in a magnetic field configuration of 45° inclination (for a 5-telescope system similar to the HEGRA system but in a horizontal and north-south symmetric configuration).
Figure 10.3: Normalized histograms of image parameters width and length for 1 TeV γ-ray showers parallel and perpendicular to the magnetic field (for a 5-telescope system similar to the HEGRA system but in a horizontal and north-south symmetric configuration).
Appendix A

Additional CORSIKA control keywords

The following keywords have been added to the keywords documented in the CORSIKA User’s Guide:

**ATMOSPHERE**  
Use MODTRAN atmospheric model $i$ (in terms of density and refraction index) instead of the CORSIKA built-in model.  
This requires a file `atm_profile_model_i.dat`.

**TELESCOPE** $x$ $y$ $z$ $r$  
Add a new telescope at $(x, y, z)$ with fiducial radius $r$.

**TELFIL** name  
The telescope-specific data in eventio format are to be written to a file `name`. If this file exists and is writable new data is appended. After the program is finished the file will be read-only to avoid accidental overwriting. The file name may be `/dev/null` to avoid output.

The following existing CORSIKA keywords have been changed:

**CERFIL** name  
To avoid creation of the normal CORSIKA Cherenkov output file (which will not be filled) set `name` to `/dev/null` or set CERFIL F (in which case it is diverted to the particle output file).

**DIRECT** name  
The directory or prefix for the CORSIKA output files may now be `/dev/null` to avoid any of the normal CORSIKA output.

Other CORSIKA keywords with parameters passed to the telescope simulation:

**CSCAT** $n$ $x$ $y$  
Randomly simulate $n$ telescope arrays in the specified area which is a circle of radius $x$ if $y$ is zero or within a rectangle of area $2x \times 2y$.

The following CORSIKA keyword is ignored when compiled with the IACT option:

**CERARY**  
(Useful only for the AIROBICC simulation).
Appendix B

sim_telarray configuration parameters

All sim_telarray configuration is processed with the CRT module config.c. Several lists of structures of type CONFIG_ITEM are declared for this purpose. Each structure holds

1. Keyword. The uppercase part must be given full, the lowercase part can be abbreviated.
2. Type. Like 'I' or 'Int' for integer, 'Double', 'Func', ...
3. Number of elements (-1 for functions).
4. Pointer to the variable holding the configuration (NULL for functions).
5. Pointer to configuration functions (NULL for variables).
6. Initial value (initial argument for functions).
7. Lower bound (if present).
8. Upper bound (if present).

See the source code documentation for details. Setting a configuration value in the configuration file can be done by a line like

telescope_altitude 100

or

Telescope_Altitude=100

The case of the entered keywords is ignored. The equal sign is just an optional placeholder. Setting a vector of variables may look like any of the following:

mirror_opt = -0.0025, -0.10, 0.0
nightsky_background_all: 0.049
mirror_x (9,10,13-14): 53
mirror_opt 3*0
The first form specifies each individual value. Values not entered are unchanged. The second form sets all elements of a vector, the third form a selected subset of a vector and the fourth a given number (3) of elements starting with the first.

The following (somewhat long) listing shows the variable declarations used for the configuration. See the CONFIG_ITEM structures for the names of all possible configuration parameters. Note that all lengths are in units of centimeters (as in CORSIKA), all angles are in degrees.

```c
/* ====================== Configuration section ======================== */
static int max_used_telescopes;

struct imaging_setup
{
    /* --- source --- */
    double source_azimuth; // Azimuth angle of source or 0.
    double source_altitude; // Altitude angle of source or 0.

    /* Reference position */
    double reference_position[3]; // Reference position with respect to
    // the observation level.

    /* --- telescopes --- */
    int base_telescope_number; // Number of first telescope, e.g.
    // 2 if starting with CT2.
    double telescope_phi; // Azimuth angle of the telescope(s)
    // Unit: degrees from north towards east.
    double telescope_theta; // Zenith angle of the telescope
    // Unit: degrees.
    double telescope_random_angle; // Random misalignment of the telescope.
    int reverse_mode; // Is 1 if reverse positioning is used.

    /* --- mirrors --- */
    int mirrors; // Number of mirrors on the telescope.
    double focal_length; // Nominal spherical mirror focal length.
    double mirror_flen_grading; // Gradual change of focal length from
    // inner to outer mirrors.
    double mirror_flen_random; // Random error in focal lengths.
    double mirror_flen_scale[MAX_MIRRORS]; // List of focal length scaling factors.
    double mirror_x[MAX_MIRRORS]; // X position of all mirrors. Unit: cm.
    double mirror_y[MAX_MIRRORS]; // Y position of all mirrors. Unit: cm.
    double mirror_diameter[MAX_MIRRORS]; // Diameters of all mirrors. Unit: cm.
    double mirror_offset; // Offset of mirror backplane from
    // fixed point of telescope mount
    // (positive if fixed point is between
    // mirror and focus). Unit: cm.
    double mirror_rnd_align_angle; // Fluctuations of the mirror alignment
    // angle with respect to nominal
    // alignment. Unit: degrees.
    double mirror_rnd_align_distance; // Fluctuations in the aligned
    // distance from the focus. Unit: cm.
    double mirror_dc_opt[3]; // Optimisation parameters relative
    // to simple Davies-Cotton design.
```
double mirror_rnd_ref_angle; // Gaussian random fluctuations of reflection angles, due to small-scale surface deviations.

double mirror_degraded_reflection; // Mirror reflectivity is by the given factor worse than the nominal table (at all wavelengths).

/* --- cameras --- */
int camera_type; // Camera geometry type (1: hexagonal, 2: square).
int camera_pixels; // Number of pixels in camera (allowed values depend on geometry type).
double camera_body_diameter; // Diameter of circular camera body (for shadowing). Unit: cm
double pixel_size; // Pixel size (flat to flat). U: cm.
double pixel_depth; // Depth of PM below pixel entrance.
double cathode_diameter; // Photocathode diameter. Unit: cm.
double lightguide_gap; // Insensitive gap between the lightguides of neighbouring pixels. Unit: fraction of pixel size.
double camera_transmission; // Transmission of camera (including the plexiglass window).
double lightguide_reflectivity; // Reflectivity of the lightguide material in front of the camera.

/* electronics */
char *spe_fname; // Name of file with single photo electron amplitude spectrum.
int nspe; // Size of lookup table for creating random numbers according to the given single p.e. spectrum (only used with the VERY_FAST_SPE compile-time option).
char *fadc_pulse_fname; // The file with FADC pulse shape.
char *disc_pulse_fname; // The file with discriminator pulse.
double fadc_frequency; // FADC frequency. Unit: MHz.
int fadc_bins; // No. of FADCs bins to be filled.
int fadc_per_channel; // How many FADCs work in parallel with corrensponding delays.
double fadc_noise; // Gaussian noise in digitisation.
double transit_time_jitter; // Jitter in nanoseconds.
double fadc_amplitude; // Signal amplitude in mV for single photoelectrons after PM, preamplifier, cable, and shaper at the inputs of the FADC.
double disc_amplitude; // Similar after amplifier at the input of the discriminators.
double adjust_gain; // Common multiplicative adjustment for FADC and discr. ampl. and PM gain (for small tel. to tel. variations).
double disc_threshold; // Discriminator threshold. Unit: mV.
double disc_threshold_variation; // Channel to channel variations [mV].
double disc_gate_length; // Effective discr. gate length [ns]
double disc_gate_length_variation; // and variation of it [ns].
double gain_variation;  // By how the gain may vary between
                    // PMs after the voltage has been
                    // adjusted for approximately
                    // the same gains. Unit: fraction.
double qe_variation;  // Quantum efficiency variation
                    // between PMs. Unit: fraction.
double fadc_pedestal;  // Nominal FADC pedestal value.
double fadc_pedestal_dev;  // Deviation of FADCs for same channel.
double fadc_pedestal_var;  // Channel-to-channel variation.
double fadc_pedestal_err;  // Assumed error in initial calibration.
double fadc_pedestal_sysvar;  // Systematic (e.g. due to temperature)
                    // variation of baselines.
double fadc_sensitivity;  // FADC counts per mV voltage.
double fadc_sensitivity_variation;  // Relative variations in sensitivity
                    // (even for FADCs of the same channel).
double pm_gain;  // Only used for DC current.
double pm_transit_time;  // Total transit time of the PM.
double pm_voltage_variation;  // Variations of transit times
                    // are usually caused by needing
                    // different voltages to reach the
                    // same gains. Unit: fraction.
double pm_gain_index;  // Gain rises as given power of
                    // the PM voltage.
double nightsky_background[MAX_PIXELS];  // Number of photoelectrons per
                    // nanosecond.
int fadc_ac_coupled;  // Set to 1 if FADCs are AC coupled.
int disc_ac_coupled;  // Similar for the discriminators.
/* Camera calibration */
double laser_photons;  // Number of laser photons at each PM.
int laser_events;  // Laser events at start of run.
int closed_pedestal_events;  // Pedestal events at start of run
int opened_pedestal_events;  // with camera lid closed/opened.
/* --- trigger --- */
double simple_threshold;  // Simple threshold. Unit: p.e.s
                    // (used only without full sim.)
double trigger_current_limit;  // Pixels above this limit are
                    // excluded from the trigger. [uA]
int trigger_pixels;  // Number of pixels required for
                    // single telescope trigger.
int trigger_neighboured;  // Number of neighboured pixels
                    // required for single telescope
                    // trigger.
int trigger_telescopes;  // Number of telescopes required
                    // for the system trigger.
int telescope_ignore[MAX_IGNORE];  // List of telescopes ignored
                    // (numbers starting at 1,
                    // i.e. CT 1, ...).
int full_simulation;  // Full simulation of FADC signals
                    // if > 0, of discriminators if > 1.
};

static struct imaging_setup setup;
static char spe_fname[1024];
static char fadc_pulse_fname[1024];
static char disc_pulse_fname[1024];
static char mirror_ref_fname[1024];
static char qe_fname[1024];
static CONFIG_ITEM cfgitems[] =
{
    /* Source */
    { "SOURCE_AZIMUTH", "Double", 1, &setup.source_azimuth, NULL, "0." },
    { "SOURCE_ALTITUDE", "Double", 1, &setup.source_altitude, NULL, "0." },
    /* Reference position */
    { "REFERENCE_POSITION", "Double", 3, setup.reference_position, NULL,
      "0., 0., 731." },
    /* Telescopes */
    { "MAXIMUM_TELESCOPES", "Int", 1, &max_used_telescopes, NULL, "6", "1",
      NULL /*, CFG_REJECT_MODIFICATION */ },
    { "BASE_TELESCOPE_NUMBER", "Int", 1, &setup.base_telescope_number, NULL, "1" },
    { "TELESCOPE_PHI", "Double", 1, &setup.telescope_phi, NULL, "0" },
    { "TELESCOPE_AZIMUTH", "Double", 1, &setup.telescope_phi, NULL, "0" },
    { "TELESCOPE_THETA", "Double", 1, &setup.telescope_theta, NULL, "0" },
    { "TELESCOPE_ZENITH_ANGLE", "Double", 1, &setup.telescope_theta, NULL, "0" },
    { "TELESCOPE_ALTITUDE", "Func", -1, NULL, (PFV) set_tel_altitude },
    { "TELESCOPE_RANDOM_ANGLE", "Double", 1, &setup.telescope_random_angle,
      NULL, "0.005" },
    { "REVERSE_MODE", "Int", 1, &setup.reverse_mode, NULL, "0", "0", "1" },
    /* Mirrors */
    { "MIRRORS", "Int", 1, &setup.mirrors, NULL, "30", "1", "30"/+MAX_MIRRORS*/ },
    { "FOCAL_Length", "Double", 1, &setup.focal_length, NULL,
      "492", "10", "10000" },
    { "GRADING_OF_FOCAL_Length", "Double", 1, &setup.mirror_flen_grading,
      NULL, "6." },
    { "RANDOM_FOCAL_Length", "Double", 1, &setup.mirror_flen_random,
      NULL, "1.5" },
    { "MIRROR_OFFSET", "Double", 1, &setup.mirror_offset, NULL, "100." },
    { "MIRROR_F_SCALE", "Double", MAX_MIRRORS, setup.mirror_flen_scale, NULL,
      "30*1.", "0.5", "2" },
    { "MIRROR_X", "Double", MAX_MIRRORS, setup.mirror_x, NULL,
      "53, 53, -53, 0, 0" },
    { "MIRROR_Y", "Double", MAX_MIRRORS, setup.mirror_y, NULL,
      "-30, 30, -30, 30, 120, -120, 120, -120, 150, -150, 150, -150,"
      "90, -90, 90, -90, 60, -60, 60, -60, 120, -120, 0, 0, 30, "
      "-30, 30, -30, 30, 60, -60}" },
    { "MIRROR_Diameter", "Double", MAX_MIRRORS, setup.mirror_diameter, NULL,
      "30*60" },
    { "MIRROR_ALIGN_RANDOM_ANGLE", "Double", 1, &setup.mirror_rnd_align_angle,
      NULL, "0.015" },
    { "MIRROR_ALIGN_RANDOM_DISTANCE", "Double", 1, &setup.mirror_rnd_align_distance, NULL, "0.5" },
};
Parameters for optimizing relative to Davies-Cotton design */
/* R.M.S. optimized (for HEGRA configuration): */
/* -0.007, -0.03, -0.0033 */
/* Max. light in 5 mm radius from nominal focus optimized: */
/* -0.003, -0.05, -0.003 */
/* -0.0025, -0.10, 0.0 */

{ "MIRROR_REFLECTION_RANDOM_Angle", "Double", 1, &setup.mirror_rnd_ref_angle, NULL, "0" },
{ "MIRROR_REFLECTIVITY", "Text", sizeof(mirror_ref_fname)-1, mirror_ref_fname, NULL, "mirror.dat" },
{ "MIRROR_DEGRADED_REFLECTION", "Double", 1, &setup.mirror_degraded_reflection, NULL, "1.0" /* 1.0 means as good as in reflectivity table */, "0.0", "1.0" },
/* Camera */
{ "CAMERA_TYPE", "Int", 1, &setup.camera_type, NULL, "1", "1", "2" },
{ "CAMERA_PIXELS", "Int", 1, &setup.camera_pixels, NULL, "271", "1", "1024" },
{ "CAMERA_BODY_DIAMETER", "Double", 1, &setup.camera_body_diameter, NULL, "45", "0", "1000" },
{ "PIXEL_SIZE", "Double", 1, &setup.pixel_size, NULL, "2.1", "0.1", "100" },
{ "PIXEL_DEPTH", "Double", 1, &setup.pixel_depth, NULL, "2.0", "0.0", "100" },
{ "CATHODE_DIAMETER", "Double", 1, &setup.cathode_diameter, NULL, "1.5", "0.1", "100." },
{ "LIGHTGUIDE_GAP", "Double", 1, &setup.lightguide_gap, NULL, "0.01", "0.", "1.""},
{ "QUANTUM_EFFICIENCY", "Text", sizeof(qe_fname)-1, qe_fname, NULL, "qe.dat" },
{ "CAMERA_TRANSMISSION", "Double", 1, &setup.camera_transmission, NULL, "0.92", "0.01", "1.00" },
{ "LIGHTGUIDE_REFLECTIVITY", "Double", 1, &setup.lightguide_reflectivity, NULL, "0.85", "0.", "1.""},
/* Photomultiplier */
{ "PM_AVERAGE_GAIN", "Double", 1, &setup.pm_gain, NULL, "1.6e5", "1e4", "3e7"},
{ "ADJUST_GAIN", "Double", 1, &setup.adjust_gain, NULL, "1.0", "0.1", "10."},
{ "PM_GAIN_INDEX", "Double", 1, &setup.pm_gain_index, NULL, "7.9", "5.0", "15.0"},
{ "GAIN_VARIATION", "Double", 1, &setup.gain_variation, NULL, "0.02", "0.0"},
{ "PM_VOLTAGE_VARIATION", "Double", 1, &setup.pm_voltage_variation, NULL, "0.10", "0.0"},
{ "QE_VARIATION", "Double", 1, &setup.qe_variation, NULL, "0.1", "0.0"},
{ "PM_TRANSIT_TIME", "Double", 1, &setup.pm_transit_time, NULL, "20.", "0"},
{ "TRANSIT_TIME_JITTER", "Double", 1, &setup.transit_time_jitter, NULL, "0.75", "0.0"},
/* Discriminator and trigger */
{ "SIMPLE_THRESHOLD", "Double", 1, &setup.simple_threshold, NULL, "8.4" /* was 7.3 */},
{ "TRIGGER_CURRENT_LIMIT", "Double", 1, &setup.trigger_current_limit, NULL, "9.0" },
{ "TRIGGER_PIXELS", "Int", 1, &setup.trigger_pixels, NULL, "2" },

77
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIGGER_NEIGHBOURS</td>
<td>Int</td>
<td>1</td>
<td>NULL</td>
<td>setup.trigger_neighboured,</td>
</tr>
<tr>
<td>TRIGGER_TELESCOPES</td>
<td>Int</td>
<td>1</td>
<td>setup.trigger_telescopes, NULL</td>
<td>&quot;2&quot;</td>
</tr>
<tr>
<td>IGNORE_TELESCOPEs</td>
<td>Int</td>
<td>5</td>
<td>setup.telescope_ignore</td>
<td>&quot;-1&quot;, -1, -1, -1, -1, NULL</td>
</tr>
<tr>
<td>PM_PHOTOELECTRON_SPECTRUM</td>
<td>Text</td>
<td>&quot;1023&quot;</td>
<td>spe_fname</td>
<td>NULL</td>
</tr>
<tr>
<td>PM_SPE_TABLE_SIZE</td>
<td>Int</td>
<td>1</td>
<td>setup.nspe</td>
<td>&quot;10000&quot;, &quot;1000&quot;, &quot;20000&quot;</td>
</tr>
<tr>
<td>DISCRIMINATOR_PULSE_SHAPE</td>
<td>Text</td>
<td>&quot;1023&quot;</td>
<td>disc_pulse_fname</td>
<td>NULL</td>
</tr>
<tr>
<td>DISCRIMINATOR_AMPLITUDE</td>
<td>Double</td>
<td>1</td>
<td>setup.disc_amplitude, NULL</td>
<td>&quot;0.94&quot; /* needed to fit data; nominal was 1.2, source ??*/, &quot;0.0&quot;</td>
</tr>
<tr>
<td>DISCRIMINATOR_THRESHOLD</td>
<td>Double</td>
<td>1</td>
<td>setup.disc_threshold, NULL</td>
<td>&quot;8.&quot; /* A. Daum, priv. comm. */, &quot;0.0&quot;</td>
</tr>
<tr>
<td>DISCRIMINATOR_VAR_THRESHOLD</td>
<td>Double</td>
<td>1</td>
<td>setup.disc_threshold_variation</td>
<td>NULL</td>
</tr>
<tr>
<td>DISCRIMINATOR_GATE_LENGTH</td>
<td>Double</td>
<td>1</td>
<td>setup.disc_gate_length, NULL</td>
<td>&quot;13.&quot; /* effective length, G. Hermann, priv. comm. */, &quot;1.&quot;, &quot;100&quot;</td>
</tr>
<tr>
<td>DISCRIMINATOR_VAR_GATE_LENGTH</td>
<td>Double</td>
<td>1</td>
<td>setup.disc_gate_length_variation</td>
<td>NULL</td>
</tr>
<tr>
<td>DISC_AC_COUPLED</td>
<td>Int</td>
<td>1</td>
<td>setup.disc_ac_coupled</td>
<td>NULL</td>
</tr>
<tr>
<td>FADC_PULSE_SHAPE</td>
<td>Text</td>
<td>&quot;1023&quot;</td>
<td>fadc Pulse_fname</td>
<td>NULL</td>
</tr>
<tr>
<td>FADC_MHZ</td>
<td>Double</td>
<td>1</td>
<td>setup.fadc_frequency, NULL</td>
<td>&quot;120&quot;</td>
</tr>
<tr>
<td>FADC_BINS</td>
<td>Int</td>
<td>1</td>
<td>setup.fadc_bins, NULL</td>
<td>&quot;16&quot;, &quot;1&quot;, &quot;128&quot;</td>
</tr>
<tr>
<td>FADC_PER_CHANNEL</td>
<td>Int</td>
<td>1</td>
<td>setup.fadc_per_channel, NULL</td>
<td>&quot;2&quot;, &quot;1&quot;, &quot;2&quot;</td>
</tr>
<tr>
<td>FADC_NOISE</td>
<td>Double</td>
<td>1</td>
<td>setup.fadc_noise</td>
<td>NULL</td>
</tr>
<tr>
<td>FADC_AMPLITUDE</td>
<td>Double</td>
<td>1</td>
<td>setup.fadc_amplitude, NULL</td>
<td>&quot;0.95&quot; /* G. Hermann: 1/1.05 */, &quot;0.0&quot;</td>
</tr>
<tr>
<td>FADC_AC_COUPLED</td>
<td>Int</td>
<td>1</td>
<td>setup.fadc_ac_coupled</td>
<td>NULL</td>
</tr>
<tr>
<td>FADC_PEDESTAL</td>
<td>Double</td>
<td>1</td>
<td>setup.fadc_pedestal, NULL</td>
<td>&quot;20.0&quot;, &quot;0.0&quot;, &quot;255&quot;</td>
</tr>
<tr>
<td>FADC_DEV_PEDESTAL</td>
<td>Double</td>
<td>1</td>
<td>setup.fadc_pedestal_dev</td>
<td>NULL</td>
</tr>
<tr>
<td>FADC_VAR_PEDESTAL</td>
<td>Double</td>
<td>1</td>
<td>setup.fadc_pedestal_var</td>
<td>NULL</td>
</tr>
<tr>
<td>FADC_ERR_PEDESTAL</td>
<td>Double</td>
<td>1</td>
<td>setup.fadc_pedestal_err</td>
<td>NULL</td>
</tr>
<tr>
<td>FADC_SYSVAR_PEDESTAL</td>
<td>Double</td>
<td>1</td>
<td>setup.fadc_pedestal_sysvar</td>
<td>NULL</td>
</tr>
<tr>
<td>FADC_SENSITIVITY</td>
<td>Double</td>
<td>1</td>
<td>setup.fadc_sensitivity, NULL</td>
<td>&quot;1.0&quot;, &quot;0.0&quot;</td>
</tr>
<tr>
<td>FADC_VAR_SENSITIVITY</td>
<td>Double</td>
<td>1</td>
<td>setup.fadc_sensitivity_variation</td>
<td>NULL</td>
</tr>
<tr>
<td>NIGHTSKY_BACKGROUND</td>
<td>Double</td>
<td>MAX_PIXELS</td>
<td>setup.nightsky_background</td>
<td>NULL</td>
</tr>
<tr>
<td>LASER_PHOTONS</td>
<td>Double</td>
<td>1</td>
<td>setup.laser_photons, NULL</td>
<td>&quot;500&quot;, &quot;1&quot;</td>
</tr>
<tr>
<td>LASER_EVENTS</td>
<td>Int</td>
<td>1</td>
<td>setup.laser_events</td>
<td>NULL</td>
</tr>
<tr>
<td>CLOSED_PEDESTAL_EVENTS</td>
<td>Int</td>
<td>1</td>
<td>setup.closed_pedestal_events</td>
<td>NULL</td>
</tr>
<tr>
<td>OPENED_PEDESTAL_EVENTS</td>
<td>Int</td>
<td>1</td>
<td>setup.opened_pedestal_events, NULL</td>
<td>&quot;0&quot;, &quot;0&quot;</td>
</tr>
</tbody>
</table>
static char input_fname[1024];
static char plot_fname[1024];
static char histogram_fname[1024];
static char pe_list_fname[1024];
static char output_fname[1024];
static char stars_fname[1024];
static double power_law;
static int only_triggered_arrays;
static int only_triggered_telescopes;
static int output_format;

static CONFIG_ITEM cfg_main[] =
{
    { "INPUT_FILE", "Text", sizeof(input_fname)-1, input_fname, NULL, "iact.dat"},
    { "PLOT_FILE", "Text", sizeof(plot_fname)-1, plot_fname, NULL, "plot.gpl"},
    { "HISTOGRAM_FILE", "Text", sizeof(histogram_fname)-1, histogram_fname, NULL, "ctsim.hdata"},
    { "PHOTOELECTRON_FILE", "Text", sizeof(output_fname)-1, pe_list_fname, NULL, "iact_pe.dat"},
    { "OUTPUT_FILE", "Text", sizeof(output_fname)-1, output_fname, NULL, "ctsim.dat"},
    { "POWER_LAW", "Double", 1, &power_law, NULL, "2.68" },
    { "STARS", "Text", sizeof(stars_fname)-1, stars_fname, NULL, "none" },
    { "ONLY_TRIGGERED_ARRAYS", "Int", 1, &only_triggered_arrays, NULL, "1" },
    { "ONLY_TRIGGERED_TELESCOPES", "Int", 1, &only_triggered_telescopes, NULL, "0" },
    { "OUTPUT_FORMAT", "Int", 1, &output_format, NULL, "0" },
    { NULL_CONFIG_ITEM }
};

static char setup_trans_fname[1024]; /* File name of transmission table */

static CONFIG_ITEM cfgitems[] =
{
    { "ATMOSPHERIC_TRANSMISSION", "Text", sizeof(setup_trans_fname)-1, setup_trans_fname, NULL, "atmo_trans.dat" },
    { NULL_CONFIG_ITEM }
};

79
Appendix C

Internal structures of sim_telarray

The following listing shows a large part of the file mc_aux.h which holds definitions of constants and structures used across several modules of sim_telarray. This is included here to illustrate the many details which are included in the simulation program.

```
#define OVERSAMPLING 10        /* Sampling of signal shape */
#define MAX_SHAPE_LENGTH 100    /* Maximum length of shape in FADC bins */
#define MAX_MIRRORS 30          /* Changing that requires changes in cfgitems[] */

#define MAX_PIXELS 1024
#define MAX_FADC_BINS 128       /* Maximum number of FADC time bins */
#define MAX_FADC_SIGNAL 255     /* Largest possible FADC value */
#define MAX_PER_CHANNEL 2       /* Maximum no. of coupled FADC per channel. */

#define MAX_TEL 6              /* The largest no. of telescopes/array. */
#define MAX_ARRAY 100           /* The largest no. of arrays to be handled */
#define MAX_IGNORE 5            /* The size of the list of telescopes to be ignored */

/* Refraction index of air as a function of height in km (0km<=h<=8km) */
#define Nair(hkm) (1.+0.0002814*exp(-0.0947982*(hkm)-0.00134614*(hkm)*(hkm)))

struct mc_options
{
    char *input_fname;       // Input file name.
    char *pe_list_fname;     // Output file with list of photoelectrons in
                             // eventio format.
    char *output_fname;      // Output file of simulation in whatever format.
    char *plot_fname;        // ASCII output file with tables relevant for plotting.
    char *histogram_fname;   // ASCII output file for histograms.
    double power_law;       // Wanted power law (e.g. -2.67).
    int only_triggered_arrays; // Is non-zero if only triggered arrays should
                                // be written to the output file.
    int only_triggered_telescopes; // Is non-zero if only data from triggered
                                     // telescopes should be written to the output file.
    int output_format;       // See 'sim_conv2ct.c' for interpretation.
};
```
struct simulated_shower_parameters
{
    double energy;     // Shower energy [TeV]
    double azimuth;    // Shower direction azimuth [deg]
    double altitude;   // Shower direction altitude above horizon
    double xcore, ycore, zcore; // Shower core position [m]
    double core_dist_3d; // Distance of core from reference point
    double tel_core_dist_3d[MAX_TEL]; // Offset of telescopes from shower axis
};

struct reconstructed
{
    double energy;     // Shower energy [TeV]
    double azimuth;    // Shower direction azimuth [deg]
    double altitude;   // Shower direction altitude above horizon
    double xcore, ycore, zcore; // Shower core position [m]
    double m_scwid;    // Mean scaled width of images
    double amplitude[MAX_TEL][2]; // Image amplitude and halo
    double width[MAX_TEL]; // Image parameter 'width'
    double length[MAX_TEL]; // Image parameter 'length'
    double dis[MAX_TEL];   // Image parameter 'dis' (distance)
    double miss[MAX_TEL];  // Image parameter 'miss'
    double conc[MAX_TEL];  // Image parameter 'conc' (concentration)
    double azwidth[MAX_TEL]; // Image parameter 'azwidth'
    double azlength[MAX_TEL]; // Image parameter 'azlength'
    double alpha[MAX_TEL]; // Image parameter 'alpha'
    double tel_core_distance[MAX_TEL]; // Offset of telescopes from shower core
    double core_distance; // Offset of CT3 from shower axis
    double core_dist_3d; // Offset of telescopes from shower axis
    double theta; // Offset of reconstructed from simulated shower
         // direction [deg]
};

struct telescope_array
{
    double longitude;  // Geographic longitude (positive towards east) [deg]
    double latitude;   // Geographic latitude (positive towards north) [deg]
    double obs_height; // Height of observation level [cm]
    double refpos[3];  // Reference position with respect to obs. level [cm]
    int ntel;          // Number of telescopes simulated per array
    int max_tel;       // Maximum number of telescopes acceptable
    int narray;        // Number of arrays with random shifts per shower
    double xtel[MAX_TEL]; // X positions of telescopes ([cm] -> north)
    double ytel[MAX_TEL]; // Y positions of telescopes ([cm] -> west)
    double ztel[MAX_TEL]; // Z positions of telescopes ([cm] -> up)
    double rtel[MAX_TEL]; // Radius of spheres enclosing telescopes [cm]
    double toff;        // Time offset from first interaction to the moment
         // when the extrapolated primary flying with the vacuum
         // speed of light would be at the observation level.
    double xoff[MAX_ARRAY]; // X offsets of the randomly shifted arrays [cm]
}
double yoff[MAX_ARRAY]; // Y offsets of the randomly shifted arrays [cm]
double azimuth; // Nominal azimuth angle of telescope system [deg].
double altitude; // Nominal altitude angle of telescope system [deg].
double source_azimuth; // Azimuth of assumed source.
double source_altitude; // Altitude of assumed source.
int min_tel_trigger; // No. of triggered telescopes needed for array trigger
int tel_triggered; // No. of telescopes triggered in current array
int array_triggered; // Is 1 if current array was triggered
int telescope_ignore[5]; // List of telescopes to be ignored in simulation
int closed_pedestal_events; // Number of pedestal events with camera closed
int opened_pedestal_events; // Number of pedestal events with camera opened
int laser_events; // Number of laser events
struct simulated_shower_parameters shower_sim;
struct reconstructed shower_reco;
typedef unsigned char fadc_data_t; // Sufficient for an 8-bit FADC.
// Note: use unsigned char to have discriminator outputs at 1/8th of FADC bins,
// unsigned short for 1/16th of FADC bins and unsigned int for 1/32nd of it.
/* #define DISC_BITS_PER_BIN (8*sizeof(disc_data_t)) */
#if defined(DISC_BITS_16)
typedef unsigned short disc_data_t; // Bit field for discriminator output.
#define DISC_BITS_PER_BIN 16
#elif defined(DISC_BITS_32)
#define DISC_BITS_PER_BIN 32
#else /* Normally 8 bits are enough: */
typedef unsigned char disc_data_t; // Bit field for discriminator output.
#endif
struct channel_calibration
{
    double pedestal_sum; // Pedestal sum of all FADCs per channel.
    double sigma_pedestal_sum; // Variation of that sum.
    double pedestal[MAX_PER_CHANNEL]; // Pedestals in individual FADCs.
    double sigma_pedestal[MAX_PER_CHANNEL]; // Variation of pedestal bin contents.
    double laser; // Laser amplitude.
    double sigma_laser; // Variation of laser amplitude.
    double laser_time; // Average time offset of laser pulses. [ns]
};
struct pm_and_fadc_channel
{
    double qe_rel; // Quantum efficiency relative to avrg.
    double fadc_amplitude; // In mV at peak of single photoelectron.
    double disc_amplitude; // Same for signal at discriminator.
    double pedestal[MAX_PER_CHANNEL]; // Actual pedestal in FADC units.
double sensitivity[\text{MAX\_PER\_CHANNEL}]; // ADC counts per mV of signal.
double transit_delay; // Transit time delay of PM.
double background; // Photoelectrons per nanosecond.
double current; // DC current [\text{uA}] due to background.
fadc\_data\_t signal[\text{MAX\_FADC\_BINS}]; // Digitized FADC signal.
double ideal_signal; // Without digitisation and background.
double median_time; // Median time (not weighted) after PM.
double disc_threshold; // Discriminator threshold [mV].
disc\_data\_t trigger[\text{MAX\_FADC\_BINS}]; // Above/below trigger at bin fraction
int trigger\_disabled; // Non-zero if pixel cannot trigger.
int triggered; // Non-zero if the pixel has triggered.
int triggered\_in\_time; // If trigger is coincident.
int nn\_triggered; // If pixel has NN triggered at same time.
int gate\_length; // Gate width as no. of
// 1/\{DISC\_BITS\_PER\_BIN\}th of bins.
struct channel\_calibration calib; // Calibration parameters.
// The following data is only set after the conversion to CT data format:
int significant; // Is 1 if pixel signal is significant
// in conversion to CT format.
double peak\_simple; // Simple pulse peak (max. bin).
double peak\_pp; // Pulse peak from pulse analysis.
double peak\_sc; // Similar but at time of pixel.
};

struct camera\_electronics
{
    int telescope; // The 'official' number for the telescope.
    int pixels; // Number of pixels.
    struct pm\_and\_fadc\_channel *channels; // Per channel data.
    int fadc\_bins; // Number of digitized bins.
    int fadc\_per\_channel; // Number of FADCs per readout channel
    double *xspe\_prompt; // Random number lookup tables for single
    double *yspe\_prompt; // photoelectron amplitude spectrum
    // for 'prompt' pulses.
    double *xspe\_bkgrnd; // Same for nightsky background including ion
    double *yspe\_bkgrnd; // feedback.
    int nspe\_prompt; // Size of preceding tables for signal
    int nspe\_bkgrnd; // and background.
    double shape[OVERSAMPLING\*\text{MAX\_SHAPE\_LENGTH}]; // Single photoelectron
    // pulse shape with peak set to 1.
    int shape\_length; // Actual length of pulse shape in bins.
    double bkg\_shape[\text{MAX\_SHAPE\_LENGTH}]; // Simpler treatment for background
    int bkg\_shape\_length; // from nightsky.
    double disc\_shape[\text{MAX\_SHAPE\_LENGTH}]; // Pulse shape at discriminators
    int disc\_shape\_length; // Length of signal at discriminator
    // [# of bins * DISC\_BITS\_PER\_BIN]
    double frequency; // FADC frequency in MHz;
    double interval; // Corresponding time interval in nanoseconds.
    double fadc\_noise; // Dark noise / digitising noise (FADC counts).
    double quantum\_efficiency[1000]; // Q.E. times mirror reflectivity times
    // camera transmission as a vector where the
    // index is wavelength in steps of nanometers.
double max_qe_rel; // Maximum of relative quantum efficiencies of all channels.
double lightguide_reflectivity; // Reflectivity of the 'funnel' before PMs.
double transit_time_jitter; // Jitter of transit times for each pm [ns].
double simple_threshold; // Simple 'discriminator' threshold in units of photoelectrons (only used when discriminator signals not fully simulated).
double signal_area; // Area under signal pulse shape.
double bkgrnd_area; // Area under background pulse shape.
double disc_area; // Area under discriminator pulse shape.
int fadc_ac_coupled; // Is 1 if FADCs are AC coupled.
int disc_ac_coupled; // Is 1 if discriminators are AC coupled.
double time_offset; // Time offset of start of FADC memory with respect to shower core crossing observation level [ns].
double nominal_delay; // Nominal value of the delay of telescope trigger signals at the central station depending on 'global' alt/az and assuming a plane light front propagating with the speed of light in air.
double pedestal_sysvar; // Systematic variation of pedestals.
double nn_offset; // Time offset of 2-fold NN trigger relative to start of FADC memory. [ns]
double nom_disc_threshold; // Nominal discriminator threshold.
double pe_conversion; // P.e. per FADC peak ampl.
double trigger_current_limit; // Pixels are excluded from trigger if the current is above this threshold [uA].
int triggered; // No. of triggered pixels.
int nn_triggered; // No. of next-neighbour triggered pixels.
int trigger_pixels_req; // No. of pixels required for telescope trigger.
int trigger_nn_pixels_req; // No. of pixels with a next-neighbour required.
int telescope_triggered; // Is 1 if the telescope has triggered.
int simulated; // Is 1 if the signal simulation was done.
int full_simulation; // Configured simulation level:
// Level 0: Only integral signals.
// Level 1: FADC simulation but not discriminator.
// Level 2: FADC and discriminator fully simulated.
double laser_photons; // Number of laser photons per pixel.
double median_time; // Median arrival time of all photons in camera with respect to shower core at obs. level. [ns]
double fadc_delay; // Delay added, taking into account of random FADC phase at time of arrival. [ns]
double photons_all; // All photons hitting telescope fiducial sphere.
double photons_detected; // Number of photons detected (photo-electrons).
int significant_pixels;
long amplitude_histogram[356];
};

struct pm_camera
{
    int telescope; // The 'official' number for the telescope.
    int camera_type; // 1: hexagonal, 2: square
    int pixels; // Number of pixels in the camera.
    double pixel_size; // Flat-to-flat 'size' (=separation) of pixels [cm].
double pixel_depth;  // Depth of PM below pixel entrance [cm].
double pixel_x_pos[MAX_PIXELS]; // X position of pixel [cm] as seen from
  // the camera front, X->north in zenith
  // with telescope at azimuth=0.
double pixel_y_pos[MAX_PIXELS]; // Y position of pixel [cm].
int hexagon_pixel_index[MAX_PIXELS];// (p,q) values -> pixel number.
int hexagon_p_pixel[MAX_PIXELS]; // pixel number -> p
int hexagon_q_pixel[MAX_PIXELS]; // pixel number -> q
int hexagon_pixel_rings;  // Number of 'rings' of hexagons.
int camera_pixel_assignment_initialized; // Is 1 if initialisation is done.
int square_camera_pixel_rows; // No. of columns/rows of square camera.
double pixel_cathode_r_squared; // Square of visible cathode radius [cm**2].
double pixel_lightguide_extend; // Fraction of pixel 'size' to which the
  // lightguide is working; the rest is dead.
double lightguide_reflectivity; // Reflectivity of the lightguide which is
  // assumed as wavelength independent.
};

struct transform_struct {
  double offset[3];       // Offset between coordinate frames.
  double rot[3][3];      // Rotation matrix.
};

struct mirror_struct {
  double x;               // X position of mirror centre [cm]
    // with respect to telescope frame.
  double y;               // Y position of mirror centre [cm]
  double z;               // Z position of mirror centre [cm]
  double r;               // Radius of curvature [cm]
  double f;               // Focal length [cm]
  double d;               // Mirror diameter [cm]
  double distance;        // Distance from camera centre [cm]
  double inclination;     // Inclination of mirror to tel. axis.
  double phi;             // Azimuth angle of inclined mirror.
  struct transform_struct trans; // Telescope <-> mirror transformation.
};
typedef struct mirror_struct Mirror;

struct telescope_optics {
  int telescope;  // The 'official' no. for the telescope.
  struct transform_struct tel_trans; // Ground <-> telescope transformation.
  struct transform_struct cam_trans; // Telescope <-> camera transformation.
  Mirror mirror_setup[MAX_MIRRORS]; // Setup/alignment of individual mirrors.
  int mirrors;  // No. of mirrors present.
  double camera_body_diameter;  // Outer diameter of camera body. [cm]
  double focal_length;  // Nominal focal length of mirrors. [cm]
  double offset;  // Offset of the centre of the point
    // where the optical axis intersects the
    // sphere of mirror centres with respect
double mirror_rnd_ref_angle; // Microscopic random reflection angle.
double azimuth; // Azimuth angle of telescope orientation.
double altitude; // Altitude angle of telescope orientation.
int reverse_mode; // Reverse or normal mode of alt-az mount.
};
Appendix D

Source code documentation

This part is available as a separate file [14] and is produced automatically with the comments utility (see CRT source code collection for sources). Note that the full Makefile rules for it require sccs, ctags, a modified version of cproto, \LaTeX with a non-standard indexing style, C compiler and preprocessor and other things (which are all installed on the EUx cluster but may be missing on other machines). For modules taken unmodified from the CRT source code collection, the corresponding CRT source documentation should be used. The actual source code contains many more comments to explain how and why things are done the way they are. These comments are not reproduced since that would effectively require to print the whole source code. For details, you are therefore encouraged to inspect the actual source code.
Appendix E

Data format of output from the modified CORSIKA program

E.1 A machine-independent hierarchical data format

The data from the modified CORSIKA program is written with the eventio package [3] which was already used for the CRT and HEGRA telescope system data. For a general description of this package see also the CRT documentation. Among the features of the eventio package are

machine independence Both big-endian and little-endian byte order is supported. Floating point representation is IEEE and conversion of VAX and general internal floating point representations to and from IEEE is included. The package has been tested under Ultrix (MIPS CPU), DEC Unix (cc and gcc), OS-9 (68040 CPU), Lynx OS (68040 and PowerPC) and also (but not recently) VAX/VMS, AIX, and MS-DOS. Since both byte-orders are equally supported, there is no need for forward and backward byte order reversal when the writing and reading is usually done on machines of the same byte order. The byte order for writing can be selected at run-time (and could, in principle at least, be changed from one block to the next).

hierarchical structure A hierarchical data structure is supported where one item may contain either several sub-items or atomic data (bytes, integers, floating point numbers). As long as sub-items are not mixed with atomic data at the same level, the structure of each data block can be listed without knowing anything about the data format (except that it is eventio format). For such listings the listio program is available from the CRT source code collection.

transmission failure recovery Even in the case that errant data is transmitted or recorded, the beginning of the next data block is found by looking for a four-
byte marker sequence (which also serves to tell the byte order of the recorded data).

### E.2 Object types in the data file

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1200</td>
<td>CORSIKA run header</td>
</tr>
<tr>
<td>1201</td>
<td>Positions and sizes of telescopes within telescope array</td>
</tr>
<tr>
<td>1202</td>
<td>CORSIKA event header</td>
</tr>
<tr>
<td>1203</td>
<td>Offsets of multiple telescope arrays for the present event</td>
</tr>
<tr>
<td>1204</td>
<td>Top level item for data from one array in one event</td>
</tr>
<tr>
<td>1205</td>
<td>Photons hitting one telescope</td>
</tr>
<tr>
<td>1206</td>
<td>Camera layout in the telescope simulation</td>
</tr>
<tr>
<td>1207</td>
<td>(not yet implemented)</td>
</tr>
<tr>
<td>1208</td>
<td>Photo-electrons after ray-tracing and detection</td>
</tr>
<tr>
<td>1209</td>
<td>CORSIKA event end</td>
</tr>
<tr>
<td>1210</td>
<td>CORSIKA run end</td>
</tr>
</tbody>
</table>

### E.3 Object formats

The following detailed list of the object formats start with the object type number (as before), the version described, and the meaning of the ident number in the block. Variables of type *Long* are 4-byte integers, variables of type *Real* are 4-byte floating point numbers. *Short* variables are 2-byte integers.

#### E.3.1 CORSIKA run header

This block (like the other CORSIKA blocks) contains just the same data as in the original CORSIKA data files (but in a machine-independent format, of course). For descriptions of their contents consult the CORSIKA users guide (for the CORSIKA version in use).

Object type: 1200  
Version: 0  
Identifier: run number

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>runh</td>
<td>Long</td>
<td>1</td>
<td>FORTRAN string RUNH (may be reversed)</td>
</tr>
<tr>
<td>data</td>
<td>Real</td>
<td>272</td>
<td>see CORSIKA users guide</td>
</tr>
</tbody>
</table>
### E.3.2 Positions and sizes of telescopes

Object type: 1201  
Version: 0  
Identifier: 0

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ntel</td>
<td>Long</td>
<td>1</td>
<td>Number of telescopes in an array</td>
</tr>
<tr>
<td>x</td>
<td>Real</td>
<td>ntel</td>
<td>$x$ pos. (measured towards north, unit: cm)</td>
</tr>
<tr>
<td>y</td>
<td>Real</td>
<td>ntel</td>
<td>$y$ pos. (measured towards west, unit: cm)</td>
</tr>
<tr>
<td>z</td>
<td>Real</td>
<td>ntel</td>
<td>$z$ pos. (from detection level, unit: cm)</td>
</tr>
<tr>
<td>r</td>
<td>Real</td>
<td>ntel</td>
<td>Radii of spheres around tel. (unit: cm)</td>
</tr>
</tbody>
</table>

### E.3.3 CORSIKA event header

Object type: 1202  
Version: 0  
Identifier: event number

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>runh</td>
<td>Long</td>
<td>1</td>
<td>FORTRAN string EVTH (may be reversed)</td>
</tr>
<tr>
<td>data</td>
<td>Real</td>
<td>272</td>
<td>see CORSIKA users guide</td>
</tr>
</tbody>
</table>

### E.3.4 Offsets of telescopes

With usually multiple instances of an array of telescopes being simulated, the arrays are randomly offset in each event.

Object type: 1203  
Version: 0  
Identifier: 0

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>narray</td>
<td>Long</td>
<td>1</td>
<td>Number of arrays</td>
</tr>
<tr>
<td>toff</td>
<td>Real</td>
<td>1</td>
<td>Time delay since first interaction (already subtracted, unit: ns)</td>
</tr>
<tr>
<td>xoff</td>
<td>Real</td>
<td>narray</td>
<td>Offsets of arrays in $x$ (unit: cm)</td>
</tr>
<tr>
<td>yoff</td>
<td>Real</td>
<td>narray</td>
<td>Offsets of arrays in $y$ (unit: cm)</td>
</tr>
</tbody>
</table>

### E.3.5 Data top-level block for one array

The data for one array are stored in one top-level block. Within this top-level block either the photons arriving at the telescope detection plane or the photo-electrons...
registered in the PM camera are contained. Note that since the data for one array is all within one top-level block, sufficient memory must be available to buffer this block. Data from the top-level block is extracted until the end of the block is reached.

Object type: 1204  
Version: 0  
Identifier: Array number

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>either photons</td>
<td>1205</td>
<td>(any)</td>
<td>Photons arriving at one telescope</td>
</tr>
<tr>
<td>or photo_electrons</td>
<td>1208</td>
<td>(any)</td>
<td>Detected photo-electrons in one camera</td>
</tr>
</tbody>
</table>

**E.3.6 Photon bunches arriving at the telescope places**

The CORSIKA photon bunches are sorted by telescope and array and are written after a full shower has been simulated. Photon bunches are actually recorded in two different formats: a long format with 32 bits per value and a short format with just 16 bits per value. Format conversion is done by the software automatically and is transparent at the user level – only the numerical accuracy is different and the amount of storage needed. *This block is only present in data read directly from the shower simulation.* It is always a subblock of type 1204.

**Long format:**

Object type: 1205  
Version: 0  
Identifier: 1000 × array number + telescope number
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Short</td>
<td>1</td>
<td>Array number of telescope</td>
</tr>
<tr>
<td>tel</td>
<td>Short</td>
<td>1</td>
<td>Telescope number</td>
</tr>
<tr>
<td>Photons</td>
<td>Real</td>
<td>1</td>
<td>Sum of photons in all bunches</td>
</tr>
<tr>
<td>bunches</td>
<td>Long</td>
<td>1</td>
<td>Number of photon bunches following</td>
</tr>
<tr>
<td>per bunch:</td>
<td>(bunches)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>Real</td>
<td>1</td>
<td>$x$ pos. relative to telescope (unit: cm)</td>
</tr>
<tr>
<td>y</td>
<td>Real</td>
<td>1</td>
<td>$y$ pos. relative to telescope (unit: cm)</td>
</tr>
<tr>
<td>cx</td>
<td>Real</td>
<td>1</td>
<td>$x$ direction cosine</td>
</tr>
<tr>
<td>cy</td>
<td>Real</td>
<td>1</td>
<td>$y$ direction cosine</td>
</tr>
<tr>
<td>time</td>
<td>Real</td>
<td>1</td>
<td>Arrival time relative to time when the primary travelling at $v = c$ would arrive at the core in the CORSIKA detection plane (unit: ns)</td>
</tr>
<tr>
<td>zem</td>
<td>Real</td>
<td>1</td>
<td>Altitude of emission (unit: cm a.s.l.)</td>
</tr>
<tr>
<td>photons</td>
<td>Real</td>
<td>1</td>
<td>Photons in this bunch</td>
</tr>
<tr>
<td>lambda</td>
<td>Real</td>
<td>1</td>
<td>So far always zero = unspec. (unit: nm)</td>
</tr>
</tbody>
</table>

**Short format:**

Object type: 1205  
Version: 1000  
Identifier: $1000 \times$ array number + telescope number

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>Short</td>
<td>1</td>
<td>Array number of telescope</td>
</tr>
<tr>
<td>tel</td>
<td>Short</td>
<td>1</td>
<td>Telescope number</td>
</tr>
<tr>
<td>Photons</td>
<td>Real</td>
<td>1</td>
<td>Sum of photons in all bunches</td>
</tr>
<tr>
<td>bunches</td>
<td>Long</td>
<td>1</td>
<td>Number of photon bunches following</td>
</tr>
<tr>
<td>per bunch:</td>
<td>(bunches)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>Short</td>
<td>1</td>
<td>$x$ position (unit: 0.1 cm)</td>
</tr>
<tr>
<td>y</td>
<td>Short</td>
<td>1</td>
<td>$y$ position (unit: 0.1 cm)</td>
</tr>
<tr>
<td>cx</td>
<td>Short</td>
<td>1</td>
<td>$x$ direction cosine (unit: 1/30000)</td>
</tr>
<tr>
<td>cy</td>
<td>Short</td>
<td>1</td>
<td>$y$ direction cosine (unit: 1/30000)</td>
</tr>
<tr>
<td>time</td>
<td>Short</td>
<td>1</td>
<td>Arrival time (unit: 0.1 ns)</td>
</tr>
<tr>
<td>log_zem</td>
<td>Short</td>
<td>1</td>
<td>$1000 \times \log_{10}(zem[cm])$</td>
</tr>
<tr>
<td>photons</td>
<td>Short</td>
<td>1</td>
<td>Photons in this bunch</td>
</tr>
<tr>
<td>lambda</td>
<td>Short</td>
<td>1</td>
<td>So far always zero = unspec. (unit: nm)</td>
</tr>
</tbody>
</table>
**E.3.7 Camera layout in the telescope simulation**

For the detected photo-electrons which are recorded from `sim_telarray`, the assumed camera layouts (at least in its principal aspects) are recorded at the beginning of a telescope simulation (one block per telescope). Note that for a telescope aligned to the zenith at azimuth angle zero, the $x$ positions are given towards North, $y$ positions towards West.

*This block is not present in data read directly from the shower simulation.*

Object type: 1206  
Version: 0  
Identifier: Telescope number

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Short</td>
<td>1</td>
<td>Camera type (1: hexagonal, 2: square)</td>
</tr>
<tr>
<td>pixels</td>
<td>Short</td>
<td>1</td>
<td>Number of pixels in camera</td>
</tr>
<tr>
<td>xp</td>
<td>Real</td>
<td>pixels</td>
<td>$x$ positions of pixels in camera (unit: cm)</td>
</tr>
<tr>
<td>yp</td>
<td>Real</td>
<td>pixels</td>
<td>$y$ positions of pixels in camera (unit: cm)</td>
</tr>
</tbody>
</table>

**E.3.8 Photo-electrons after ray-tracing and detection**

The `sim_telarray` writes (if wanted) the detected photoelectrons to an output file. This output file contains all blocks from the input file – but with photo-electrons instead of photons – plus the assumed camera layout. The photons are sorted by arrival time at the PM. *This block is not present in data read directly from the shower simulation.* It is a subblock of type 1204.

Object type: 1208  
Version: 0  
Identifier: $1000 \times$ array number + telescope number

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pixels</td>
<td>Long</td>
<td>1</td>
<td>Number of pixels in camera</td>
</tr>
<tr>
<td>per non-empty pixel:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ipix</td>
<td>Short</td>
<td>1</td>
<td>Pixel number</td>
</tr>
<tr>
<td>npe</td>
<td>Short</td>
<td>1</td>
<td>Number of photo-electrons</td>
</tr>
<tr>
<td>time</td>
<td>Real</td>
<td>npe</td>
<td>Times of photo-electrons (unit: ns)</td>
</tr>
</tbody>
</table>

**E.3.9 CORSIKA event end block**

Object type: 1209  
Version: 0  
Identifier: Event number
### Variable Type Number Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>evte</td>
<td>Long</td>
<td>1</td>
<td>FORTRAN string EVTE (may be reversed)</td>
</tr>
<tr>
<td>data</td>
<td>Real</td>
<td>272</td>
<td>see CORSIKA users guide</td>
</tr>
</tbody>
</table>

#### E.3.10 CORSIKA run end block

Object type: 1210  
Version: 0  
Identifier: Run number

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rune</td>
<td>Long</td>
<td>1</td>
<td>FORTRAN string RUNE (may be reversed)</td>
</tr>
<tr>
<td>data</td>
<td>Real</td>
<td>2</td>
<td>see CORSIKA users guide</td>
</tr>
</tbody>
</table>
Bibliography


Erratum

Due to an error in implementing the tabulated model atmospheres in CORSIKA, the tabulated values never reached the EGS part in CORSIKA (which insists on a piecewise exponential atmosphere). As a consequence the gamma-shower lateral distributions shown for different model atmospheres (chapter 11) only account for different indices of refraction while the shower development was always calculated with the CORSIKA built-in U.S. standard atmosphere approximation. This is fixed in CORSIKA 5.8 but the report has not been updated.