The IACT/ATMO package

Introduction to the IACT/ATMO package

This is the README file for the CORSIKA supplements for Cherenkov light by Konrad Bernlöhr. It was last updated for release 1.67 (November 2023).

The Software and data files provided with this package are intended to enhance the CORSIKA air shower simulation program by

a) tabulated atmospheres for different climate zones, including accurate indices of refraction and

b) a flexible interface for arbitrary configurations of Cherenkov light detectors, which don't need to be in a horizontal plane. Although intended for systems of Imaging Atmospheric Cherenkov Telescopes (IACTs), it can be used for any kind of Cherenkov detectors.

c) A machine-independent output format for the 'photon bunches' collected at each of the assumed telescopes or other detector (format and relevant software termed 'eventio').

If you find this software useful for your work, a reference in your publications to

- K. Bernlöhr, Simulation of Imaging Atmospheric Cherenkov Telescopes with CORSIKA and sim_telarray, Astroparticle Physics 30 (2008) 149–158. [arXiv:0808.2253]

would be appreciated.

The main CORSIKA distribution comes with suitable function interfaces which can be selected in the compiler pre-processing step. See the IACT and ATMEXT options in the CORSIKA User's Guide. For successfully compiling and linking CORSIKA when either or both of these options is selected, you need the software provided here. CORSIKA versions 6.4xx to 7.7500 are supported by this package. For support to older CORSIKA versions (5.8xx to 6.3xx), use version 1.47 of the IACT/ATMO package. A number of variables in interfaces and common blocks had different order or different sizes (changes appearing in versions 5.901, 6.032, 6.200, and 6.400) and the current IACT interface will refuse to compile or run with the outdated versions. The `GNUmakefile` used to compile CORSIKA versions before 6.500 has been dropped as well with IACT/ATMO version 1.48.

The build process of CORSIKA 6.5xx and newer also covers the IACT option. However, there is a little utility '`gen_config`' included with this package that can be used to setup and build CORSIKA in a batch-style way (without any questions asked). This utility is well tested with CORSIKA versions up to 6.990 and also with most 7.x version after 7.4000, including CORSIKA 7.7500. See also the '`corsika_build_script`' automating the compilation as much as possible - although it may not work in every environment where CORSIKA can be made to compile.

Note that depending on other CORSIKA options selected, compilation of some of the files provided here might require different preprocessor flags.

Part a) is implemented in '`atmo.c`' and includes files `atmprof1.dat`, `atmprof2.dat`, `atmprof3.dat`, `atmprof4.dat`, `atmprof5.dat`, and

`atmprof6.dat`. Files `atmprof[1-6].dat` contain atmospheric profiles as tabulated in

- F.X.Kneizys et al. (1996), The MODTRAN 2/3 report and LOWTRAN 7 model, Phillips Laboratory, Hanscom AFB, MA 01731, U.S.A.

Indices of refraction were calculated with the effect of water vapour taken into account. An additional file `atmprof9.dat` was constructed for the antarctic winter climate from publicly available radiosonde data for several antarctic stations, in particular the Amundsen-Scott South Pole Station. Beyond 32 km altitude this atmospheric profile is based on extrapolations which might not be very accurate.

Part b) is implemented in its most basic form in '`iact.c`' but for taking full use of it, additional software ('eventio') is required. Instead of a rectangular grid of rectangular detectors, as in previous CORSIKA implementations, individual detectors can be placed individually. Positions don't need to be in a horizontal plane but may stick out of the lowest CORSIKA detection layer. No detector details are needed in the CORSIKA run except for the radius of a sphere containing each detector. Data is recorded in the usual form of CORSIKA 'photon bunches'. With the additional software (see part c) these are recorded in a machine-independent format for later detector and atmospheric transparency simulation. An example skeleton program for such follow-on processing is included in this distribution.

Output in the 'eventio' data structure can be either written into a file or passed directly (piped) to another program. To use such a pipe for immediate processing of the output (from its standard input), use a leading '|' character (perhaps following a '+' if long format is desired) with the TELFIL keyword in the CORSIKA input. Note that no blanks can be part of the file name or pipe supplied due to CORSIKA input processing. The length is also rather limited and any nonprintable or shell metacharacters are forbidden for pipes. If you need argument lists or further redirection, using a shell script for that purpose was the only possibility up to version 1.52. Now you can use the 'IACT TELOPT' input card for that:

```
TELFIL +|my_simulation_script.sh
IACT TELOPT -c my_configuration.cfg
```

If the output file name ends in `.gz`, `.bz2`, `.lzo`, `.lzma`, `.xz`, or `.zst` then files will be compressed on the fly with the gzip, bzip2, lzop, lzma, xz, or zstd programs, respectively. This typically saves on the order of 30% disk space – but at a performance cost, as usual with file compression. Because of its superior compression speed (at lower CPU usage and better compression ratio than gzip), the zstd program ("zstandard", available at `https://github.com/facebook/zstd`) is recommended - but note that while basically all Linux distributions provide installation packages for zstd it is usually not installed by default. The compressed files (of any of the above mentioned types) can also be read on the fly in your analysis program if you replace the fopen() call with fileopen() and the fclose() call with fileclose(), and have the corresponding (de-)compression program installed.

Note also that the TELFIL parameter may contain up to 7 integers appended to the actual name, with colons ':' as separators. Values 1 to 5 determine the amount of information written to standard output for each or every n'th event, while value 6 determines when photon bunches stored in main memory are swapped to temporary files. See the `telfil_` function in `iact.c` for details. Value 7 may be used to override the maximum size for the I/O buffer.

1: number of events for which the photons per telescope are shown
2: number of events for which energy, direction etc. are shown
3: every so often an event is shown (e.g. 10 for every tenth event).
4: every so often the event number is shown even if no. 1 and no. 2 ran out.
5: offset for no. 4 (no. 4=100, no. 5=1: show events 1, 101, 201, ...)
6: the maximum number of photon bunches before using external storage
7: maximum I/O buffer size in Megabytes.
Example: TELFIL +iact.dat:5:15:10

Since this can be rather confusing there are alternatives. Instead of adding these numbers to the TELFIL parameter you can use separate IACT input parameters - which may be less confusing:

```
TELFIL +iact.dat:5:15:10:1000:1:100000:200
```

is equivalent to

```
TELFIL +iact.dat
IACT print_events 5 15 10 1000 1
IACT internal_bunches 100000
IACT io_buffer 200MB
```

Depending on whether your CORSIKA writes Cherenkov light vertical distributions as integral distributions (INTCLONG option) or as light emission distributions (default) you might need to set a '`-DINTEGRATED_LONG_DIST`' preprocessor definition when compiling '`iact.c`'. Since INTCLONG implies a substantial CPU overhead, it is better avoided. With the '`-DINTEGRATED_LONG_DIST`' flag, the Cherenkov light distribution obtained from CORSIKA will be differentiated before writing it. All other vertical distributions are unaffected.

Part c) This additional software referred to as 'eventio' provides a rather general means for machine-independent I/O although with a restricted set of basic data types (implemented in '`eventio.c`' and '`io_basic.h`', requiring also '`warning.c`' and '`warning.h`') and additional files providing higher-level functions for the IACT interface ('`io_simtel.c`', '`mc_tel.h`', '`initial.h`').

The eventio buffers have a user-defined maximum size to avoid that your system gets into trouble when a shower with an unexpectedly large number of photon bunches is encountered. This maximum has a compile-time default, normally 200 Megabytes, that you can change by '`make MAX_IO_BUFFER=500000000`', for example, and which you can also override at run-time with the optional value number 7 of the TELFIL option (see above) or with the `CORSIKA_IO_BUFFER` environment variable, as in:

```
bash> export CORSIKA_IO_BUFFER=500MB
```

or

```
tcsh> setenv CORSIKA_IO_BUFFER 8GB
```

To make use of buffer sizes of 1 GiB and more, you need an eventio version supporting the extended length format. For buffer sizes above 2 GiB you must, in addition, run on a 64-bit system.

Please note the copyright notices (in file '`Copyright`' and in individual source files). If you want to use the software for other purposes than intended (i.e. with CORSIKA), just ask me (`Konrad.Bernloehr@mpi-hd.mpg.de`).

Installation notes

Recent versions of this software were mainly tested under Linux (on i386 and x86_64 architectures) with GNU gfortran and gcc (as part of the GNU Compiler Collection, GCC) version 4.8.5 to 13.2. Note that with gfortan 7.2 the `LD_LIBRARY_PATH` environment variable was needed to include the path of the libfortran.so but that was not reproduced with any other versions, older or newer. On the x86_64 (AMD Opteron and newer Intel CPUs) architecture, both 32 and 64 bit modes are possible. Tests with older compiler versions (GNU g77 and gcc version 3.2 to 3.4.3 as well as GNU gfortran and gcc version 4.0 to 4.7) are not usually done any more. The IACT/ATMO C language code has also been tested to compile with the clang compiler. It can also be compiled by a C++ compiler (g++ or clang++).

Compilation of all the C/C++ code was also tested with clang/clang++ versions 7 to 13, selecting different C (or C++) language standards. For recent compiler versions (same applies to gcc/g++) this covers C99, C11, and C17 (plus experimental C2X) as well as their more relaxed 'gnu' variants plus the 'gnu90' variant (C89/C90 will not work). For the C++ compilers, C++98, C++11, C++14, C++17, C++20, and C++23 were tested (except where older versions of the compilers are not supporting the latest standards). Same with the corresponding gnu++ variants. For historical/unusual compiler versions see footnote.[1]

The 'eventio' software has been tested and used under a wide variety of operating systems and CPUs. although recent version get mainly tested under Linux. A test program is included with this distribution and you are advised to compile and run the test program first, before trying out the higher-level software. Output of this 'testio' test program might look like

```
Write test data to file 'test.dat'.
Default byte order, using mainly vector functions.
Default byte order, using single-element functions.
```

[1] Note that with GCC 4.0.0 the g77 compiler was replaced by gfortran, a Fortran95 compiler which, initially (versions 4.0.x), was not able to compile the CORSIKA and interaction model codes - mainly due to multiple 'SAVE' statements. The incompatibility was classified as a 'regression' by GCC maintainers and is fixed in newer GCC versions (4.1.x). CORSIKA versions 6.7x and up and recent gfortran versions (4.2.x to 13.1) work well together and gfortran compiled code is substantially faster than g77 compiled code. The C code provided with this package used to have no problems with gcc 4.0.x versions. And there are no problems by mixing gcc 4.0.x to gcc 4.4.x as the C compiler with g77 from earlier releases. This has been tested with gcc 4.0.1 and g77 3.3.6 as well as with newer versions.

Older CORSIKA versions were also tested under DEC UNIX 4.0 ('OSF/1'), compiled with DEC f77 and cc and also with GNU g77 and gcc. The older versions were also tested under Linux (i386) with GNU g77 and gcc (egcs 1.1.2 and gcc 2.7.2, 2.95.2, 2.96, and most 3.x versions). However, for gcc versions 2.95.2 and earlier we have seen what appear to be code-generation bugs when compiling Fortran files with optimization enabled, resulting in Not-a-Number values leading in turn to endless loops. If that happens to you, try compiling with '-O0' (optimization disabled). With later gcc versions (2.96 and 3.0.x to 3.4.3 at that time), no such bugs were seen and full optimization appeared to be fine. In recent times, no other systems than Linux in 64-bit mode have been used by the author.

It is expected to run also on other flavours of UNIX, at least with gcc/g77 and perhaps also under some of the other more or less popular 'operating systems' for Intel x86/Pentium PCs, but I never tried. On anything more exotic (from my point of view) I will probably not be able to give any help for porting this software. One area of potential trouble will be interfacing FORTRAN and C functions which is implemented here in the common way on UNIX systems, with an underscore added at C functions called from FORTRAN. All parameters are passed by address, system-specific passing of character string lengths is circumvented by null-terminating the strings before passing them to C functions.

```
Reversed byte order, using single-element functions.
Normal byte order, using single-element functions.
Write tests done.

Read test data from file 'test.dat'.
Default byte order, using single-element functions.
Default byte order, using mainly vector functions.
Reversed byte order, using single-element functions.
Normal byte order, using single-element functions.
Read tests done

Everything is ok. Congratulations!

Note: on this machine you should care about the sign propagation
of 'LONG' data elements to long integer variables.

Note: on this machine you should care about the sign propagation
of 'SHORT' data elements to integer or long integer variables.
```

For a very brief introduction to eventio see the comments at the beginning of 'eventio.c'. A detailed description is available, both in English and in German. Most likely, the basics of 'eventio' should be of less interest to you than the interfaces to the higher-level functions described in comments in 'io_simtel.c' and the skeleton of a telescope simulation program included with this distribution (in sim_skeleton.c). Selected source code comments, function prototypes and cross references of function calls as obtained with the doxygen tool are contained in 'iact_refman.pdf'.

Note that structured 'eventio' data blocks can be listed with the 'listio' tool also provided with this distribution (note: use the '-s' option for showing the structure of nested blocks, use the '-n' and/or '-d' options to names and/or descriptions for the data block type numbers from EventioRegisteredNames.dat). The actual contents of data files written by the IACT interface can be examined with the read_iact program.

The IACT/ATMO package may also contain some 'patches' in the unified diff format which have not yet been merged into the mainline CORSIKA distribution. Patches included with older versions of the IACT/ATMO package introduced also additional features like:

CERWLEN The index of refraction is made wavelength dependent. As a consequence, photon bunches will carry a specific wavelength. Photons of shorter wavelengths (with larger index of refraction) will result in larger Cherenkov cone opening angles and larger bunch sizes. For very fast particles this will generally have a small effect (less than 0.03 deg in the opening angle, for example) but near the Cherenkov threshold the effect can be larger.

This option may also require to use a smaller maximum bunch size (see CERSIZ keyword) since all photons in a bunch are of the same wavelength and, therefore, the peak quantum efficiency rather than the average quantum efficiency determines the maximum acceptable bunch size. (In combination with the CEFFIC option, you should use a maximum bunch size of 1, as usual.)

IACTEXT The interface to the TELOUT function is extended by parameters describing the emitting particle. This extended information is stored as an additional photon bunch (after the normal one) with mass, charge, energy, and emission time replacing the cx, cy, photons, and zem fields, respectively, and are identified by a wavelength of 9999. The compact output format is disabled for making that possible. In addition, all particles arriving at the CORSIKA observation level are included in the eventio format output file, in a photon-bunch like block identified by array and detector numbers 999.
The x, y, cx, cy, and ctime fields keep the normal sense, with coordinates, directions and time counted in the CORSIKA detection level reference frame. The particle momentum is filled into the zem field (negative for upward-moving particles) and the particle ID is filled into the lambda field. If thinning is used, the particle weight is in the photons field.
When compiling iact.c manually (instead of taking advantage of the GNUmakefile), an additional option -DIACTEXT is required to have a consistent interface on both sides.

EXTPRM Instead of the built-in setting of primary particle properties (type, energy, direction) according to PRMPAR, ERANGE, ESLOPE, THETAP, PHIP, VIEWCONE, a user-supplied subroutine EXTPRM (or C function extprm_) can be used to simulate arbitrary mixtures of primary particles, of arbitrary spectra and direction.

These are now part of the default CORSIKA distribution and all patches only applicable to CORSIKA versions before 6.500 have been dropped with IACT/ATMO package 1.48.

Building CORSIKA 6.5xx to 6.7xx

The IACT/ATMO package is provided in the bernlohr sub-directory. If you find a 'GNUmakefile' there, rename it to'GNUmakefile.org'. The build process provided with CORSIKA 6.5xx will ask quite a number of questions for a fresh install - but will remember your preferences for later rebuilds. Users not well familiar with the CORSIKA build options should go through this interactive process:

```
./corsika-install
```

Users who want to get an installation done without risking to have one of the many questions answered in a different way than required, may try the gen_config script:

```
bernlohr/gen_config [ options ]
./corsika-install < /dev/null
```

The possible options include (upper/lower case ignored) most of the CORSIKA build options, for example:

```
bernlohr/gen_config IACT atmext ViewCone volumedet QGSii uRqmd
```

Implied options are added automatically (in the example above the IACT option implies the CERENKOV option).

Building CORSIKA 6.9xx/7.3xxx/7.4xxx/7.5xxx/7.6xxx/7.7xxx

Building CORSIKA versions 6.9xx is very similar to versions 6.5xx to 6.7xx but the configuration utility has changed, and some configuration definitions in the `config.h` file have changed. Support for these versions by the `gen_config` script has been tested at least up to CORSIKA 6.990, 7.3700 (which has issues like crashing), 7.4100 (DO NOT USE 7.4 versions below 7.4005), 7.5700, as well as 7.6300, 7.6400, 7.6900, and 7.7000 to 7.7500. In case of problems you will have to use the interactive configuration utility `coconut`. If all goes well,

```
bernlohr/gen_config [ options ]
./coconut < /dev/null
```

or

```
bernlohr/corsika_build_script [ options ]
```

should be enough to produce a CORSIKA binary with the requested compile-time options. CORSIKA versions 7.xxx can be built just like 6.99x but not all versions have seen as much testing. Note that in particular versions 7.4000 to 7.4004 turned out to be not usable (at least not with the IACT interface). CORSIKA version 7.3700 was seen to fail with floating-point exceptions and cannot be recommended, even though it is the end point for some older interaction models. The 7.7420 package internally sets the version number to 7.7500 - which causes our build tool to fail and therefore that CORSIKA release was not further tested. The 7.7410 version and the actual 7.7500 package work well.

Updates to the IACT/ATMO package

You probably obtained this file through the CORSIKA download area (see `https://www.ikp.kit.edu/corsika/` for instructions). The version there is not updated very often. For more frequent updates to the IACT/ATMO package see also `http://www.mpi-hd.mpg.de/hfm/~bernlohr/iact-atmo/`, where you will need to identify as user '`iact`', password '`corsika`'. If you download a newer version from there, you will have to unpack it in the `bernlohr` sub-directory manually because the CORSIKA build process will only try to unpack exactly the version that came with CORSIKA. You may also have to clean object files in the corresponding build directory (`lib/Linux/bernlohr` or such) if you did build CORSIKA with another version of the IACT/ATMO package before.

This README file, the documentation in the other files of this distribution, and in the CORSIKA User's Guide may answer most of your questions but perhaps not all. In case of further questions or installation problems you are welcome to send e-mail to me, i.e. `Konrad.Bernloehr@mpi-hd.mpg.de` Questions entirely related to CORSIKA itself and neither related to the Cherenkov light emission in CORSIKA nor to the extensions provided with this distribution should be better directed to Tanguy Pierog (`tanguy.pierog@kit.edu`).

If you manage to get the enhancements running on other machines than used by me, I would be glad to hear from you. Other users might benefit from your experience.

Usage notes

Finally, I would like to add a few notes on the usage of CORSIKA for Cherenkov light simulations and some highlights of version changes:

a) If using the LONGI keyword for sampling the 'longitudinal' (vertical) shower development and either taking a CORSIKA version before June 2000 or a later one extracted with INTCLONG option, then of the order of 40% of the CPU time is actually spent in functions CERLDE and CERLDH which are extremely inefficient in their backwards compatible form. If using them that way, the file '`iact.c`' should be compiled with the preprocessor definition '`-DINTEGRATED_LONG_DIST`' to differentiate the Cherenkov light vertical distribution. If you think that you need the longitudinal development for shower particles but not for Cherenkov light and have a pre- June 2000 CORSIKA version, you are advised to remove the calls to CERLDE and CERLDH in subroutine CERENK. With newer versions, this can be accomplished by the NOCLONG option when extracting from CMZ. The default behaviour of new CORSIKA versions should be to sum up photon information only at the level of emission.

b) The Cherenkov photons are stored in 'bunches'. CORSIKA includes an automatic calculation of the bunch size, depending on primary energy. However, this calculation was only optimised for the non-imaging Cherenkov counters (AIROBICC) in the HEGRA array. For Cherenkov telescopes this value will usually be too high. Therefore, you are advised to set an appropriate bunch size for your purposes with the CERSIZ keyword. For imaging telescopes with pixels sensitive at the one photo-electron level, using conventional photomultiplier tubes, a bunch size of 5 to 10 photons seems appropriate. Larger bunch sizes would add artificial 'noise' to your images because, in simple words, you either measure no photo-electron in a pixel or you measure several photo-electrons when the pixel is hit by just one bunch.

   When using the CEFFIC option, where atmospheric absorption, mirror reflectivity and quantum efficiency are already applied in CORSIKA, the bunch size has to be reduced even further. In that case, a bunch size of 5 would mean that, typically, you either measure no or five (or a multiple of about five) photo-electrons. A bunch size of 1 (photo-electron) appears more appropriate then. **Note:** *The CORSIKA data files provided for use with the CEFFIC option (atmabs.dat, mirreff.dat, quanteff.dat) were provided by other CORSIKA users and I am probably not the right person to ask in case of problems with these files.*

   Whether writing photon bunches or photo-electron bunches is more efficient in your case, is best determined by trying out both cases with appropriate bunch sizes. A properly designed telescope simulation program should be able to cope with both options. Photo-electron bunches (i.e. when the CEFFIC option is used) should be marked by a 'wavelength' parameter of -1, while a value of 0 indicates photon bunches (of undetermined wavelength within the given limits). Positive values, indicating a photon wavelength in nanometers are reserved for future enhancements. Negative values other than -1 indicate the wavelength of detected photons with the CEFFIC option in CORSIKA version 6.500 and newer.

c) For zenith angles above some 65 to 70 degrees, the CURVED option should be applied. In that case, the emission altitude of the photons is meant to be the altitude in the common sense, above sea level. Calculation of the amount of air traversed, e.g. for calculating atmospheric transmission, or the distance to the emission point no longer scales exactly with the secant of the zenith angle then.

d) The compact photon bunch output format, requiring 16 bytes per bunch, has several limitations which should probably be of little relevance for current or near-future telescope systems, but should be kept in mind.

   1) Bunch sizes must be less than 327.
   2) photon impact points in a horizontal plane through the centre of each detector sphere must be less than $\pm$32.7 m from the detector centre in both x and y coordinates. Thus,

      $$\sec(z) * R < 32.7 \text{ m}$$

      is required, with 'z' being the zenith angle and 'R' the radius of the detecor sphere. When accounting for multiple scattering and Cherenkov emission angles, the actual limit is reached even earlier than that.
   3) Only times within 3.27 microseconds from the time, when the primary particle propagated with the speed of light would cross the altitude of the sphere centre, can be treated. For large zenith angle observations this limits horizontal core distances to about 1000 m.

   For efficiency reasons, no checks are made on these limits. Starting with version 1.21 of this package, some tests at the beginning of each event were introduced which should catch most violations of the limits, although individual photon bunches still exceeding the limits cannot be fully ruled out. When any of these limits can be exceeded, the longer output format, requiring 32 bytes per bunch, should be used. This is achieved by prefixing the output file name (after the TELFIL keyword) with a '+', without any blanks inbetween, as e.g. in

   ```
   TELFIL +iact.dat
   ```

   or

   ```
   TELFIL +|analysis_program
   ```

   When reading data in compact format it is automatically converted to the longer format, bunch by bunch. Therefore, you need not bother with the internal representation of the data format.

e) If disk space is an important limitation, you may add a `.gz` or `.bz2` to your file names to force compression through the `gzip` or `bzip2` programs, respectively. Note that, due to the efficient eventio file format, compression rates are only of the order of 30%. If your analysis programs opens these files through the `fileopen()` function rather than the standard `fopen()` function, decompression on reading will be automatic.

f) When the IACT option is used together with the THIN option, all bunch sizes are simply multiplied by the weight, with no changes in the output format. Since the limit in d2) above is then easily violated, the compact bunch format should not be used together with the THIN option. Well, for imaging atmospheric Cherenkov telescopes, the thinning option might be not really appropriate at all and this final limitation, therefore, irrelevant.

g) You may pipe the IACT output data directly into a telescope (or other detector) simulation program by using a setting like

```
TELFIL |analysis_program
```

or

```
TELFIL +|analysis_program
```

in the CORSIKA inputs file. Note that due to restrictions of CORSIKA's processing of inputs files, you cannot add any command line option to your analysis program there. In case you need such options, use an intermediate shell script or similar to call your analysis program with options. The analysis program should read the data from its standard input then.

h) Note that the positions given on the 'TELESCOPE' lines in CORSIKA inputs file are in centimeters with respect to the nominal core position and the CORSIKA detection level. They are NOT counted from sea level.

i) Starting with version 1.21 of this package, the VOLUMEDET option of CORSIKA is now supported. While it is determined at the time of CMZ extraction for CORSIKA, it is fully dynamic for the IACT option and determined at run time. When CORSIKA is extracted/compiled without the VOLUMEDET option, all random shower core offsets (see the CSCAT configuration keyword in the CORSIKA User's Guide) are in a horizontal plane. Before version 1.21 of this package, this was always the case. If CORSIKA is now extracted/compiled with the VOLUMEDET option, the random core offsets are in the shower plane, defined as perpendicular to the shower axis.

j) Starting with version 1.25, the package has been prepared for importance sampling of core position offsets. This would mean that actual core offsets can be generated in a non-uniform distribution and can extend to different distances, depending on primary energy, primary type, zenith angle and so on. This package, however, does not provide a real implementation of importance sampling (other than for testing that the later stages of the processing properly get the weights for each event). If you do nothing about it, you will get uniformly distributed core offsets as before. If you plan to make use of importance sampling, you have to replace the file 'sampling.c' with an implementation of your choice.

k) Starting with version 1.34, the deflection of charged primary particles in the geomagnetic field is accounted for (with TSTART on). Thus the nominal core position is no longer

on the extrapolation of the original velocity vector but approximately where the primary particle would intersect the detection level, assuming no interaction and no multiple scattering take place. This is useful for single muons in calibration events but also for low-energy electrons. This can be disabled in CORSIKA 6.5xx through a configuration line

```
IACT impact_correction off
```

Users of CORSIKA 6.5xx need at least version 1.35 for getting the correction to work.

l) Starting with version 1.36, an experimental interface for setting primary particle type, energy, and direction has been added. A patch has to be applied to corsika.F in the meantime (as of CORSIKA 6.600):

```
(cd src && patch -p1 <extprm.patch)
```

The actual implementation has to be provided by the user before adding any

```
IACT extprim ...
```

data cards to the CORSIKA inputs file. Through this interface it is possible have mixed primary compositions, non-power-law spectra, and/or arbitrary angular distribution, for example simulating Cherenkov telescopes which follow a target along the sky.

m) Starting with version 1.38, the dynamic range in a telescope is basically unlimited due to automatic thinning of bunches. When a detector sphere is hit by more than the given maximum number of bunches, the actual number of bunches is reduced by increasing powers of two, by discarding every second bunch and increasing the bunch size of the remaining bunches by factors of two. Very large eventio buffers are now possible on 64-bit machines. They were formerly limited to less than 1 GiB per telescope array and event and now to 2 GiB on 32-bit machines but can be increased to 4 Terabytes on 64-bit machines.

n) Starting with version 1.41, the photon bunch data can be written in individual blocks for each telescope instead of one block for a full array. The default behaviour is still one block per array. To force writing individual blocks use

```
IACT split-always
```

in the CORSIKA inputs file. To split only data of events with many bunches, use

```
IACT split-auto [maxbunches]
```

where the optional `maxbunches` number is the threshold in the sum of photon bunches across the whole array where splitting starts should take effect. The number can have a 'k' or 'M' postfix to indicate thousands or millions of bunches. If the number is missing, a default (normally 10 million bunches) applies.

o) Version 1.43 fixes a bug with running CORSIKA with the THIN option, and adds automatic disabling of the compact bunch format when finding that the THIN option is actually used.

p) Starting with version 1.44 the IACT/ATMO package is mostly provided under the terms of the GNU Lesser General Public License, version 2.1 or newer (lgpl2.1+, for short), except for a couple of small utility programs provided under different terms (gpl3+).

p) Since version 1.46 the MARK_DIRECT_LIGHT and STORE_EMITTER compiler otions are available (but you may need to activate them manually in the source code, as passing them via the CFLAGS through CORSIKA build tools is rather tricky).

q) The charge sign of the emitting particle is available for CORSIKA versions before 7.400 only after applying a patch (relevant only with IACTEXT option).

r) Support for CORSIKA versions before 6.500 has been dropped with version 1.48. Latest CORSIKA version tested is now 7.4100.

s) The maximum length for file names provided with the TELFIL parameter has been extended (patch included for some 7.3 and 7.4 versions which is only needed for 'TELFIL' parameter card but not for 'IACT TELFIL' parameter card).

t) The limitation of the Cherenkov emission wavelength to 700 nm in CORSIKA is (and always has been) irrelevant with the IACT option and no CEFFIC option. Since semiconductor sensors (SiPMs) are generally sensitive to longer wavelengths, this has become an issue and can be solved with some patches. CORSIKA version 7.5000 has the wavelength range opened up to 2000 nm by default.

u) An example program has been added to show how IACT-style output can be produced for artificial light sources without CORSIKA. A more user-friendly and more capable version of it is distributed with a sim_telarray package.

v) Starting with version 1.50 the IACT_NO_GRID compiler option allows to bypass the assignment of detector 'shadows' to grid cells at the CORSIKA detection level. While not recommended for most productions, for performance reasons, it can be mandatory to certain test scenarious like, for example, a 10 km high string of telescopes used to illustrate the difference of shower images at different altitudes, or the artificial light sources which can be simulated without using CORSIKA.

w) Starting with version 1.53 there is an additional 'IACT TELOPT' parameter card feature to add command line options in case that 'TELFIL' (or 'IACT TELFIL') specifies that output is to be written to a pipe.

x) Version 1.54: Optimisation of the interpolation functions involved in the translation between height and thickness or density or index of refraction, and also in the refraction corrections is in progress. A first set of optimized functions is available if FAST_INTERPOLATION2 is defined during compilation (note: FAST_INTERPOLATION is defined by default).

y) Starting with version 1.54 an optional hook into checking if a track segment may result in detectable Cherenkov light was added (as a patch for CORSIKA 7.6400) but the checking code in the IACT module still has to be worked out before the hook gets recommended for general use.

z) Starting with version 1.55 negative bunch sizes are accepted as a way to distinguish two classes of origin of the photon bunches, e.g. Cherenkov versus fluorescence light or direct versus scattered light. Both the IACT module as well as the sim_telarray program now use fabs() of the reported bunch size.

aa) Starting with version 1.56 the TELFIL name may contain `${RUNNR}`, `${NSHOW}`, `${OBSLEV}`, and/or `${CORSIKA_VERSION}` which will be replaced with the corresponding values from the run header before opening the output file (note that OBSLEV gets converted here to units of meters). Setting of environment variables is now possible with the 'IACT setenv variable value' inputs card - which takes effect before environment variables are expanded for the IACT output file name, independent of the order in the inputs file.

ab) Starting with version 1.57 a new utility program got included to print the contents of the output produced by the IACT interface, called `read_iact`. Compile with

```
make -f GNUmakefile.org read_iact
```

or

```
gcc read_iact.c fileopen.c io_simtel.c eventio.c \\
  straux.c warning.c -lm -o read_iact.
```

The `listio` program, built with '`make -f GNUmakefile.org listio`' and used to show the data block structure of the IACT interface output data, has been augmented with block type number to text registry. To make this available globally copy the file `EventioRegisteredNames.dat` to `~/.EventioRegisteredNames` in your HOME directory. (Note: you will need the '-n' or '-d' options with listio to show the registered names.)

ac) Also in version 1.57 the atmo.c code has seen significant reworking in a number of areas. 1) The fast interpolation can now be (and is, by default) used for all routine interpolations, including those for the refraction correction. 2) An extra source code file, rpolator.c, plus the corresponding header file has been included – and, until the CORSIKA build procedure gets adapted, that really means an `#include` – with additional interpolation features. 3) Cubic spline interpolation instead of linear interpolation is used now, except for fast interpolation where it is only used in the initialization. The changes to areas 1) to 3) are, for the time being, implemented as compile-time options, to allow for further tests and optimization before down-selecting the different interpolation schemes. 4) The fit for the CORSIKA EGS4 part has now several sets of starting layers and finishes the fit for the most promising set of starting layers.

ad) Version 1.58 goes one step further in run-time optimization, using a more finely sampled set of equidistant supporting points for the atmospheric profiles 'fast interpolation', actually now so finely sampled that the exponential function calls can be avoided. Using

cubic splines on top of that has only a marginal accuracy improvement but costs some 10% in CPU time and is thus not enabled by default. Using tabulated atmospheres is now (marginally) faster than built-in profiles.

ae) Version 1.59 enforces the use of rpolator and cubic splines in the initial interpolation of atmospheric profiles and the fast interpolation scheme later-on (matching the default settings in version 1.58).

af) Version 1.60 enables including the atmospheric profile data table into the IACT interface output. On the detector simulation side, there is no longer a need to find an 'atmprof¡n¿.dat' file with '¡n¿' matching the number encoded in the first event header block (although this will continue to work as a fall-back method). Because the coconut build procedure and its 'Makefile.in' in the 'bernlohr' sub-directory have not been adapted to the new files, both repolator.c and mc_atmprof.c get included from atmo.c, for the time being.

ag) Version 1.61 checks that an event header was emitted before encountering an event end. This could happen with electron/proton primaries that are heavily deflected or lose too much energy before they can have a first interaction. There is also an experimental CORSIKA patch available that will make sure that missing event headers get written before the event end, also for the non-IACT output files.

ah) Version 1.62 includes support for the vectorized version of the raybnd function (best with SSE/AVX instruction sets and vectorized math library). So far a custom build procedure is needed for that. Thanks to L. Arrabito and colleagues for the work put into this performance improvement.

ai) Instead of passing custom definitions for the pre-processor through the coconut/configure/make steps, it is also possible to create an optional include file "iact_config.h" (best in the same place as "iact.c" and "atmo.c") which a not too old compiler (gcc: 4.9.x or newer) will recognize and include, if present. Older compilers would ignore it.

aj) Fixing a problem for muons injected at the top of the atmosphere where a rounding error could result in returning a density of zero rather than that at the top support point, and a zero step size in muon tracking (infinite loop).

ak) Version 1.63 comes with a useful implementation of importance sampling, to be more precise: a three-zone parametrization with a flat distribution $p(r) = \text{const.}$ for $0 \le r < r_1$, a power-law distribution $p(r) = r^k$ (usally with $k \le 0$) for $r_1 \le r < r_2$, and again a flat distribution for $r_2 \le r \le r_3$. The overall distribution is continuous at r_1 and r_2. With all generated core offsets scaled by the CSCAT radius, r_3 is fixed to one, and r_1 as well as r_2 are effectively provided as fractions of the CSCAT radius. The parameters (for a fixed, i.e. energy-independent sampling) or name of the parameter file (for energy-dependent or energy/angle-dependent sampling) are provided in the CORSIKA inputs on a line like one of:

```
IACT TELSAMPLE fixed:k,r1,r2
IACT TELSAMPLE 1d:fname-with-E-k-r1-r2
IACT TELSAMPLE 2d:fname-with-E-theta-k-r1-r2
```

If the keywords fixed/1d/2d are missing, any parameter starting like a number gets assumed as 'fixed', otherwise as the name of a file for 1-D interpolation (depending on energy only). Energies are listed in units of GeV, zenith angles (theta, for 2-D only) in degrees. Example file for 1-D interpolation (comment line not necessary):

```
# E    k   r1   r2
1.0   -5   0.1  0.3
10.   -5   0.15 0.3
100.  -5   0.2  0.4
1e3   -4   0.3  0.6
1e4   -3   0.5  1.0
1e5   -2   1.0  1.0
```

The example results in mainly small offsets for low energies and approaches a flat distribution at (and beyond) 100 TeV. Although energies E are given in units of GeV, the actual interpolations, separately for k, r_1, and r_2, are done by linear (bi-linear) interpolation in $log(E)$ for 1-D ($log(E)$ and θ for 2-D). The generated events are stored with an event weight used to compensate for the non-uniform distribution. Since this weight has a value of π times the CSCAT radius squared for a flat distribution, it is also termed *area weight*. Radii over-represented in the generated distribution have a weight less than that, radii under-represented have a weight larger than in a flat distribution. Histograms of core positions with weights applied should correspond to the flat distribution for any energy (and zenith angle) range, in the plane perpendicular to the shower axis (VOLUMEDET option required).

al) Version 1.63 also comes with a simple but useful implementation of the external setup of the primary particles (ID, energy, and direction). The implementation reads text files with parameters for the primary of one shower per line, with four to six values: the particle type code (as used internally in CORSIKA), the total energy in units of GeV, the zenith angle in degrees, the azimuth angles in degrees, plus an optional field where a value of '1' would indicate that the azimuth angle should be counted in the astronomical convention rather than the default CORSIKA convention. The final optional value is an event weight to be multiplied on top of the 'area weight' as used for non-uniform core position sampling (importance sampling). To activate that feature, the CORSIKA inputs must include a line

```
IACT EXTPRIM [text:]filename
```

where the format specifier `text:` in front of the filename is optional, as text mode is the only format currently supported. An example file:

```
# A 300 GeV gamma ray coming from South (z.a. 20 deg.):
1    0.3e3  20    180  1
# PHI in CORSIKA units, with event weight 0.5:
1    0.2e3  20.1  355  0 0.5
# Implicitely PHI is in CORSIKA units:
1    0.25e3 20.2  356
# Proton from a little bit West of South:
14   1e3    20    182  1
# And an iron nucleus from a bit East of South:
5626 5e3    20    178  1 2.0
```

If the specified file contains fewer lines than NSHOW requests, the remaining events will result in errors and/or events of too low energy to be detected. Concerning the optional event weight, keep in mind that the further processing has no knowledge (without statistical analysis of the resulting distributions) if the events were generated as CORSIKA is made to believe (in parameters ERANGE, ESLOPE, THETAP, PHIP, VIEWCONE), or from a different distribution. Also keep in mind that the THETAP and PHIP parameters are typically used in further processing for the telescope alignment. Drawing external showers corresponding to an entirely different direction might not be a good idea.

am) The package 'bernlohr-1.63.tar.gz' unfortunately was made available under the same name with slightly different ingredients. In particular, an iact.c version from 2020-12-04 made it into the CORSIKA distribution, including a bug in subroutine TELPRM which got fixed three days later. This bug becomes relevant if compiled with the IACTEXT option (which is needed to activate particle output through the IACT interface). In order to resolve the mess of different packages under the same 1.63 name, please upgrade to 'bernlohr-1.64.tar.gz' - which is basically the same as the latest 1.63 except that it should resolve a few more compiler warnings, if you use a modern compiler.

an) The identification and handling of 2D importance sampling tables in versions 1.63 and 1.64 had some errors. There is work-around for these version using the 'rpolist' tool and standard system tools to first interpolate a 2D table into a 1D table for a specific zenith angle and then use that file in 'IACT TELSAMPLE' implicitly as 1D. Version 1.65 has these errors fixed and can use the 2D tables directly. It also includes auto-detection for 1D (four data columns) or 2D (five data columns). The work-around will continue to work.

ao) The IACT module can decide on its own if particle data should be written to the IACT output, starting with version 1.66. The dependency on the IACTEXT compile-time option and the FPAROUT run-time flag is no longer needed (patch `corsika-77410-telprt.patch`). It is off by default. (Behavior would not be backwards-compatible either way.) Use the 'IACT STORE-PARTICLES onoff' input card to set it at run-time ('onoff' can take on any of the usual values for booleans, 0/1/N/Y/On/oFF/NO/yeS/false/true/...).

ap) The IACT module can decide at run-time if extra information about the particles emitting Cherenkov light should be written, starting with version 1.66 - as long as the IACTEXT compile-time option is used. For backward-compatibility, it is off by default without STORE_EMITTER and on with STORE_EMITTER define at compile-time. (Pretty much backwards-compatible.) Use the 'IACT STORE-EMITTER onoff' input card to set it at run-time ('onoff' can take on any of the usual values for booleans, 0/1/N/Y/On/oFF/NO/yeS/false/true/...).

aq) Version 1.67 passes the actually configured run number along (as a 32-bit integer) rather than relying on the floating-point value included in the RUNH data (which has only 23 bits in its mantissa). As long as the run number gets used from the ID field of the run header data block rather than from the data field (sim_telarray and various tools adapted accordingly), run numbers up to $2^{31} - 1$ are supported, in theory independent from the NRREXT option in the CORSIKA code itself – but CORSIKA itself will still refuse any run numbers above 999 999 unless compiled with NRREXT option. Eventio-related code was refreshed.

ar) The preference for the compact bunch format is being phased out. While in prior versions the long bunch format is only chosen when explicitely asked for (by prefixing the TELFIL name with '+'), or when limitations in the compact format are expected to cause problems, you may now need to prefix the TELFIL name with '-', if you prefer the compact format or compile with PREF_COMPACT_BUNCH defined. Note that this '-' prefix will not work with IACT versions before 1.67.

Konrad Bernlöhr [November 2023]

Files included in this distribution:

File	Size	Comments
Copyright	1432	[Copyright notice]
lgpl-2.1.txt	26434	[GNU LGPL version 2.1]
lgpl-3.txt	7637	[GNU LGPL version 3]
gpl.txt	35147	[GNU GPL version 3]
GNUmakefile.org	23115	[used to compile optional utility and test programs]
README	58032	[this file as plain text]
README.pdf	~142000	[this file as PDF file]
EventioRegisteredNames.dat	16780	[shows names for known eventio block type numbers]
atmo.c	113375	[source code for tabulated atmospheres and refraction]
atmo.h	3259	[header file in modules linking with atmo.c]
atmprof1.dat	2749	[atmospheric profile: tropical]
atmprof2.dat	2655	[atmospheric profile: midlatitude summer]
atmprof3.dat	2691	[atmospheric profile: midlatitude winter]
atmprof4.dat	2748	[atmospheric profile: subarctic summer]
atmprof5.dat	2763	[atmospheric profile: subarctic winter]
atmprof6.dat	2748	[atmospheric profile: U.S. standard 1976]
atmprof9.dat	2838	[atmospheric profile: antarctic winter; fixed density column
corsika_build_script	8807	[shell script for building CORSIKA]
current.c	6814	[insert current time string into warnings]
current.h	1690	[hqeader file for optional current time add-on]
eventio.c	164383	[source code for basic 'eventio' functions]
eventio_en.pdf	129942	[description of basic eventio data format (in English)]
eventio_registry.c	7005	[optional code for loading and showing block names]
eventio_registry.h	1476	[include file for eventio_registry.c]
fake_corsika.c	11765	[example showing how to write IACT data without CORSIKA]
fileopen.c	44942	[open files found in one of several paths]
fileopen.h	2105	[header file in modules linking with fileopen.c]
gen_config	41700	[shell script for configuring CORSIKA 6.5xx and newer]
hconfig.h	11605	[header file included by io_history.c]
iact3d.ps	427991	[plot illustrating photon bunch selection method]
iact.c	168931	[source code for IACT interface to CORSIKA]
iact.h	3405	[IACT interface definitions for C/C++ programs]
iact_config.h	844	[Any IACT-related definitions we may want to set up]
iact_refman.pdf	~1.0 MB	[automatically generated source code documentation]
initial.h	12445	[required include file for system-specific things]
io_basic.h	14632	[required include file for 'eventio']
io_history.c	30909	[required extra file for 'read_iact' tool]
io_history.h	4062	[required include file for 'read_iact' tool]
io_simtel.c	100393	[source code for higher-level IACT I/O functions]
listio.c	6357	[source code for utility to list 'eventio' blocks]
mc_atmprof.c	10839	[sharing atmospheric profile in IACT/ATMO and later stages]
mc_atmprof.h	4010	[a structure for storing the tabulated atmospheric profile]
mc_tel.h	12292	[required include file for IACT eventio interface]
read_iact.c	11800	[utility program for showing the data written by iact.c]
rpolator.c	104952	[source code for advanced table interpolation]
rpolator.h	6563	[header file for rpolator]
sampling.c	17447	[implement importance sampling of core offsets]
sampling.h	1275	[interface definition for importance sampling]
sim_skeleton.c	55576	[skeleton of a program reading the IACT eventio data]

```
   straux.c             4933   [auxilliary string functions]
   straux.h             1561   [header file for straux.c]
   testio.c            45709   [source code for 'eventio' test program]
   trapfpe.c            1685   [helps trapping floating point errors with g77/gcc]
   unused.h             2644   [declare macros to mark unused function parameters]
   warning.c           18429   [required auxilliary source code for 'eventio']
   warning.h            2927   [required include file for 'eventio']

Optional patches to CORSIKA (typically applied with: "patch -p1 < xyz.patch",
from the top-level corsika-nnnn directory) include:

   extprm.patch        Adding the EXTPRM subroutine call that may (depending
                       on data cards, requiring user-provided code) allow
                       user-defined selection of primary particle properties
                       (for CORSIKA 6.600; should also work for 6.502).
   corsika-6720.patch  In most 6.7xx CORSIKA versions one OPEN statement
   corsika-6735.patch  fails with gfortran because of a non-standard 'name='
                       parameter (instead of 'file='). Apply the patch closest
                       to your version; some offset is normally not a problem.
   corsika-6990.patch  Store also the sign of the charge of the emitting particle
                       with the IACTEXT option. Not relevant without IACTEXT.
                       Now incudes the 700 nm patch (see below).
   corsika-73700.patch Extending the wavelength range should now be standard procedure.
   corsika-74005-700nm.patch  Cherenkov emission to a 2000 nm upper limit.
                       (Remember to keep away from versions 7.4000 to 7.4004.)
   corsika-74100.patch and longer TELFIL file names need a little change as well.
   corsika-75600.patch Add emission for electrons before first interaction.
   corsika-75700.patch Avoid double throwing of wavelengths with CERWLEN.
   corsika-76300.patch Fix ignoring a second CSCAT input card with misleading warning.
   corsika-76400-checktrack.patch An experimental patch to call the iact_check_track function.
                       Not ready for general use.
   corsika-76900-bernlohr-Makefile.am.patch A patch for automake configuration with
                       the additional rpolator and mc\_atmprof code compiled separately
                       rather than included by atmo.c for the IACT/ATMO interface.
                       Requires automake to update Makefile.in (tricky and version
                       dependent; building works fine without this patch).
   corsika-76900-cerlde.patch A patch to the CERLDE subroutine, reducing the number
                       of THICK() function calls most of the time from two to one,
                       without SLANT and CURVED options at least.
   corsika-77100-write_evth.patch Try enforcing an event header even if no interaction
                       happened for a photon/electron/positron primary. EXPERIMENTAL!
   corsika-77400.patch Fix a problem that can arise from a rounding error when
                       injecting muons at the top of the atmosphere.
   corsika-77410-telprt.patch Always call TELPRT if CERENKOV and IACT are enabled.
                       The IACT interface can decide on its own if particles
                       get actually written to the IACT output (for version >= 1.66).

No patch is currently used with CORSIKA 7.7500.
```