Running Monte Carlo Simulations using WACT and Milagro

Robert Atkins

1 Introduction

Recently I have been working on combining the WACT simulation and the Milagro simulation and I think that I have a production version ready. With earlier versions(v<v6003) of CORSIKA this was not as much of an issue. Frank Samuelson and Tom Stephens had hacked CORSIKA in order to make the Cherenkov routines to include atmospheric absorption and quantum efficiency. Changes were also made to include a "real life" array of detectors as opposed to the regular array of detectors that CORSIKA required. Getting information into the Milagro simulation was easy since all the headers in the Cherenkov files and the particles files (CERNNN and DATNNN) were identical. The major drawback was that the Cherenkov routines in CORSIKA were not very efficient.

As newer versions of CORSIKA were released the Cherenkov routines were greatly improved to take all types of attenuating factors into account at the top of the simulation. Thus you don't waste cpu cycles tracking particles that will not hit your detector, or if they do hit, will not produce a photo-electron. The problem was that the data format of the Cherenkov files changed, and the core positions in the DATNNN file was no longer a real core position.

2 New CORSIKA and Milagro

CORSIKA will generate as usual the DATNNN file that contains all the particle information that is used in the Milagro simulation. CORSIKA will also generate a TelescopeNNN.dat file, if it has been extracted to include the production of Cherenkov light, that contains the Cherenkov photons for all the telescopes (Actually it contains pes, since the quantum efficiency has already been taken into account). CORSIKA keeps any photon that passes through a sphere that contains the entire Cherenkov telescope and records its time intersecting the sphere relative to the time of the primary intersecting the observations height. The TelescopeNNN.dat file is processed by WACTsim_v1.0.c This program traces the photon from the intersection point on the sphere to the WACT mirror and then to the focal plain adjusting the times according to the time of flight. Once in the plain the photons are collected into pixels according to the camera described in camera.txt. WACTsim_v1.0.c creates two files; TelescopeNNN.dat.core and TelescopeNNN.dat.txt. TelescopeNNN.dat.core contains all the core locations for the CORSIKA events. These core locations are used in the Milagro simulation by getparticle.F. By default the Milagro simulation will introduce random offsets for all the particles of a given event thus giving each event a random core position. The versions of CORSIKA used by Milagro does not randomize the core positions and thus Milagro randomizes the core positions with offsets. Coredist.F was modified to include the case where no offset is added to the particles thus using the CORSIKA core position. This change is activated by the setting the dynamic core flag in the detector.txt file to 99. The TelescopeNNN.dat.txt file contains all the WACT pmts hits and the times at which the 5th, 15th, 30th, 50th, and last pes that struck the pmt. This file also includes an event header and Hillas parameters.

Once WACTsim_v1.0.c has been ran, the Milagro simulation can be started. It will require the DATNNN file as usual and the TelescopeNNN.dat.core file. The output from the Milagro simulation is as usual. Once finished the program finishup.c should be run. Finishup.c will read the Milagro output and the TelescopeNNN.dat.txt and combine them into one file. The names of the three files; TelescopeNNN.dat, the Milagro output file, and the combined output file are all passed to the program at the command line. This final file can be read by the Milagro offline code once it has been modified to also read the WACT info.

3 How To Run It

Running these four programs and keeping all the file names straight can be a bit cumbersome. Frank wrote a nice PERL script to handle the Milagro simulation and I have modified it to include all the relevant WACT programs. This script and all the other WACT/Milagro simulations can be found at http://titus.physics.wisc.edu/ atkins/WACT/. The PERL script (mgrbatch2) will have to be modified to contain the appropriate directories for the CORSIKA and Milagro directories. In these directories you will find a CMZ script for extracting the VENUS version of CORSIKA and the appropriate makefiles. The CMZ script will have to be modified such that the correct path to CMZ is included. The CMZ command is located toward the end of the cmz_extract script. To run CORSIKA with a different combination of hadron models the makefile must be changed. To run with VENUS you can simply type "make corsika_venus_iact". To execute the script simply type "./mgrbatch2 10 proton". This command will run all the programs 10 times with a proton as the primary. In mgrbatch2 there is a list of species that can be used.

Also in these directories you will find the appropriate INPUT files and detector.txt file. In the INPUT file you will have to change the directory listed after the DIRECT keyword as well as the TELFIL file. In detector.txt you must change the file names. For example, at Wisconsin the directory that contains all the simulation information is /data01/atkins/sims/wactmilagro_gv31/ and the file name is /data01/atkins/sims/wactmilag_gv31/proton/DAT000017. If you were running at LANL you may want to change this to /disk03/sims/wactmilagro_gv31/pronton/DATNNN. Once you have the correct directory structure the PERL script will automatically update all the files accordingly. After the four programs are done, the PERL script will change the names of the the files so they do not overwrite on the next run.

More information on CORSIKA can be found in the the same directory as the source code. Information about the improved Cherenkov routines can be found in src_cmnts.ps and eventio_en.ps of the source code directory.

4 WACT MC Data Format

The format of the WACT portion of the mc files is as follows.

```
npmt_WACT
tube_number pes  time1  time2  time3  time4  time5
****
****
****
****
hillas_Nmax  hillas_Xo  hillas_Xmax  hillas_a hillas_b hillas_c hillas_chi^2
```

This data is appended to the end of each Milagro event. The Milagro format is unchanged.

5 CORSIKA INPUT file

```
**********************************************************************
RUNNR  134  number  of  run
EVTNR  1  number  of  first  shower  event
NSHOW  20  number  of  showers  to  generate
PRMPAR  402  particle  type  of  prim.  particle
ESLOPE  -1.5  slope  of  primary  energy  spectrum
ERANGE  1.E3  1.E6  energy  range  of  primary  particle  (GeV)
THETAP  0.  30.  range  of  zenith  angle  (degree)
PHIP  0.  360.  range  of  azimuth  angle  (degree)
SEED  539  0  0  seed  for  1.  random  number  sequence
SEED  540  0  0  seed  for  2.  random  number  sequence
SEED  541  0  0  seed  for  3.  random  number  sequence
SEED  542  0  0  seed  for  IACT  core  postions
OBSLEV  265.E3  observation  level  (in  cm)
ELMFLG  T  T  em.  interaction  flags  (NKG,EGS)
ATMOSPHERE  6  T  use  US  atmosphere  from  Bernlhor.
RADNKG  200.E2  outer  radius  for  NKG  lat.dens.distr.
CERSIZ  1.  bunch  size  Cherenkov  photons
CERFIL  N  Cherenkov  output  to  extra  file
CWAVLG  200.  665.  Cherenkov  wavelength  band
CSCAT  1  50000.  50000.  scatter  Cherenkov  events
CERQEF  T  T  T  Take  quant  eff,  absorb,  reflec.into  acount.
TELESCOPE  6294.5  -961.5  0.  120.
TELESCOPE  -1399.8  -6147.2  0.  120.
TELESCOPE  -7050.  -10700.  0.  120.
TELESCOPE  -13264.1  -876.  0.  120.
```

```
TELESCOPE  3400.  5900.  0.  120.
TELESCOPE  5254.  11700.  0.  120.
TELFIL  +/data01/atkins/sims/wactmilagro_gv31/helium/Telescope.134.dat
FIXHEI  0.  0  first  interaction  height  &  target
FIXCHI  0.  starting  altitude  (g/cm**2)
MAGNET  22.9  46.3  magnetic  field  centr.  MILAGRO
HADFLG  0  1  0  1  0  2  flags  hadr.interact.  &  fragmentation
ECUTS  0.1  0.15  0.00085  0.001  energy  cuts  for  particles
VENUS  T  0  use  venus  for  high  energy  hadrons
VENSIG  T  use  VENUS  hadronic  cross  sections
MUADDI  T  additional  info  for  muons
MUMULT  T  muon  multiple  scattering  angle
LONGI  T  5.  T  T  longit.distr.  &  step  size  &  fit  &  out
MAXPRT  1000  max.  number  of  printed  events
ECTMAP  1.E4  cut  on  gamma  factor  for  printout
STEPFC  2.0  mult.  scattering  step  length  fact.
DEBUG  F  6  F  1000000  debug  flag  and  log.unit  for  out
DIRECT  /data01/atkins/sims/wactmilagro_gv31/helium/  output  directory
DATBAS  T  write  .dbase  file
USER  atkins  user
EXIT  terminates
**********************************************************************
```

6 Milagro detector.txt

```
**********************************************************************
'/data01/atkins/sims/wactmilagro_gv31/proton//DAT000017'
'/data01/atkins/sims/wactmilagro_gv31/proton/Telescope.17.dat.core'
'/data01/atkins/sims/wactmilagro_gv31/proton/proton.17.45520611.dat'
0               ! view (0=no graphic, 1=top, 2,3=side, 4=persp., (1)5=PMT)
1               ! detector [1=Milagro, 2=Milagrito]
670.            ! waterlevel        [cm] (Milagrito: 180 cm, Milagro: 670 cm)
680.            ! coverlevel        [cm]
1               ! baffles ?     [0=no,1=yes]
1               ! outrigger array as described in outrigger.txt [0=no,1=yes]
99              ! "dynamic" core position [0=no, otherwise area factor,99 for wact]
100.  100.      ! maximum range of core position (x,y) [m] >  10 GeV
100.  100.      ! maximum range of core position (x,y) [m] > 100 GeV
100.  100.      ! maximum range of core position (x,y) [m] >   1 TeV
100.  100.      ! maximum range of core position (x,y) [m] >  10 TeV
```

```
0.      0.      ! common core offset in (x,y) [m]
0.1             ! minimum total energy [GeV] for particle output (gammas,e+,e-)
1               ! first event
10              ! last event
0               ! number of times to reuse corsika shower
0               ! minimum number of tubes in air shower layer
outrigger_001mc.txt ! outrigger survey file
**********************************************************************
```

7 mgrbatch2*

```
#!/usr/bin/perl
##########################################################################
#
#       mgrbatch2
#
# This script takes two arguments and a corsika input file and a geant milagro
# input file.  It uses these resources to generate new output files
# and then run the corsika and  geant programs in succession.
# You must modify this script to include your milagro and corsika
# directories and the executables therein.
# The number of runs to simulate and the primary particle type used
# in the runs have to be given on the command line.
# Run the script with the following command
#             mgrbatch2 number_of_runs  particle_type
#
#  Particle_type can be proton,neutron,antiproton,gamma,electron,
#  positron,muon,muon+,helium,oxygen,carbon,nitrogen,iron,pi0,
#  pi-,or pi+.
#  Remember, the number of events that you generate is
#  the number of runs multiplied the number of events specified
#  in the corsika input file.
#  The proper particle ID is inserted into the corsika input file,
#  the random number seeds are appropriately incremented, and the
#  the output directory is changed depending on the output particle
#  type.  The run number is properly incremented.
#  All other options remain the same as in the original input
#  file.   Corsika is then run.  A new geant input file is then
#  generated with the corsika output file as the new input file.
#  The geant output file is modified too.   All other options remain
```

```
#  the same as in the original geant input file.  The milagro Geant simulation
#  is then run and the corsika file is deleted to save space.
#
#  Note!!  The input files are locked when they are modified.  This means
#  that the script is safe to use on  multiple machines with the same
#  input files simultaneously.  This means you need to maintain only
#  one corsika input file and one geant input file, and no run numbers
#  or random seeds will be reused, and you don't have to keep track of them.
#
#
#  This script requires milagro version 25 or later!!
#  If you want to make changes to this script please do, and notify me if
#  they're useful.
#  Frank Samuelson Mon Jul 17 16:40:28 MDT 2000


################### Modify these SETUP variables  ############################

### Home directory where the corsika executable and input file are
$HOMECORSIKA  = "/usr/users/atkins/corsika/v6003";
######### The corsika  executable which resides in the above directory;
$corsika = "corsika";
$wact_read = "wact_read"; # Corsika program file
$finishup  = "finishup";
######### The reference input file which resides in the above directory
$cinfile = "INPUTS";         # Corsika program file


### Home directory where the milagro executable and input file are
$HOMEMILAGRO  = "/usr/users/atkins/wact/sims/milagro/milagro_geant_v31";
######### The milagro executable which resides in the above directory;
$geant   = "milagro";     # Geant program file
######### The reference input file which resides in the above directory
$ginfile = "detector.txt";         # Geant input file

##################### END  of  SETUP ###################################

 use Fcntl;
#require('sys/fcntl.ph');

$HOST = $ENV{"HOSTNAME"}; # get host name
```

```
$USER = $ENV{"USER"}; # get the users login name

########################################## Arguments to the script
$MAXRUNS   = $ARGV[0];                                # number of runs to generate
$type = $ARGV[1];
$corsika = "$HOMECORSIKA/$corsika";         # Corsika program file
$geant   = "$HOMEMILAGRO/$geant"; # Geant program file
$ginfile="$HOMEMILAGRO/$ginfile";
$cinfile="$HOMECORSIKA/$cinfile";
$wact_read = "$HOMECORSIKA/WACT/$wact_read";
$finishup = "$HOMECORSIKA/WACT/$finishup";

# Determine the correct number to go into the particle field based on
# the command line argument given above
if ($type =~ /proton/)     { $typen=14; }
elsif  ($type =~ /neutron/) { $typen=13; }
elsif  ($type =~ /antiproton/) { $typen=15; }
elsif  ($type =~ /gamma/) { $typen=1; }
elsif  ($type =~ /electron/) { $typen=3; }
elsif  ($type =~ /positron/) { $typen=2; }
elsif  ($type =~ /muon/) { $typen=6; }
elsif  ($type =~ /muon+/) { $typen=7; }
elsif  ($type =~ /helium/) { $typen=4*100+2; }
elsif  ($type =~ /oxygen/) { $typen=16*100+8; }
elsif  ($type =~ /carbon/) { $typen=12*100+6; }
elsif  ($type =~ /nitrogen/) { $typen=14*100+7; }
elsif  ($type =~ /iron/) { $typen=56*100+26; }
elsif  ($type =~ /pi0/) { $typen=7; }
elsif  ($type =~ /pi-/) { $typen=9; }
elsif  ($type =~ /pi+/) { $typen=8; }
else {
    print STDERR " particle not understood\n";
    exit(-2);
}


##########################################################################
# Here we loop over all the allowed run numbers, generating input files #
# and running the various simulations for each run number. #
##########################################################################
```

```
for($i=0; $i<$MAXRUNS; $i++){#start

#
# Create a new input file for Corsika
#
#   Open and lock current Corsika input file

    open(TEMP,"+< $cinfile") || die "Couldn't open the silly file $cinfile\n";
    &lockfile(TEMP) || die " Could not lock file $cinfile\n";

#  Determine the output directory by reading what is in the input
#  file, stripping off the last directory name and adding our new one

    do {
$ln=<TEMP>;
@el=split(/ +/,$ln);
if ($el[0] =~ /RUNNR/)  { $RUN= $el[1]; }
    } while ($el[0]!~ /DIRECT/);  # Read until we get the right line
    @sdnms=split(/\//, $el[1]);   # Split the line
    $sdnm="";
    for ($j=0; $j<($#sdnms); $j++) {   # Reconstruct the line
$sdnm=$sdnm.$sdnms[$j].'/';
    }
    $sdnm=$sdnm.$type.'/';         #Put the directory name in.
    $robnm='+'.$sdnm;

#  If the output directory does not exist, create it.
    if ( ! -d $sdnm ) { system("mkdir $sdnm"); }


#  Determine the highest run number already existing in the
#  directory
    if (!open(INPUT,"ls -1 $sdnm |")) {  #listing on output directory.
print STDERR "Can't do listing on directory $sdnm\n";
exit(-2);
    }

    $greatest=0;
    while ( $ln=<INPUT> ) {    # read the name of each file in the directory
if ($ln =~ /DAT/  ) { # If it is a data file
```

```
    $lnn=substr($ln,3,6);
    if ($lnn> $greatest ) {      # See if it has the highest run num.
$greatest=$lnn;

    }
}
    }
    close(INPUT);

# Rewind back to the beginning of the inputs file
    seek (TEMP,0,0);           # Go back to beginning of the inputs file

    sleep 3;
# Makeup a new inputs file name
    $tt=time-963531000;
    if ($RUN<=$greatest ) {
$RUN=$greatest+1;
    } else {
$RUN++;

    }
    $makeup="$cinfile.$tt.$HOST.$RUN";

# open and lock the new input file
    open(FILE,">$makeup") || die "Couldn't open $makeup\n";
    &lockfile(FILE) || die "Could not lock file ";

 #   $wactpos=undef;
# Read the old input file, replacing the correct values as we go
    while($temp=<TEMP>){
@fields = split(/ +/,$temp);
if ($fields[0] eq "RUNNR"){
    $fields[1]=$RUN;
} elsif ($fields[0] eq "DIRECT"){
    $fields[1]=$sdnm;
    $directr=$fields[1];

}elsif($fields[0] =~ /SEED/){
    $fields[1]+=4;           # Set the seed
} elsif ($fields[0] =~ /PRMPAR/ ) {
    $fields[1]=$typen;   # Set the particle type
```

```
          } elsif ($fields[0] =~ /TELFIL/ ) {
              $filename ='Telescope.'.$RUN.'.dat';
      $fields[1] =$robnm.$filename;
      $wactnm = $sdnm.$filename;
#       print STDERR "$fields[1]\n";
    }


if ( $#fields) {
  print FILE join("  ",@fields);  # print every line
          }
  }
#  }
# make the new inputs file and the old one the same
    close(FILE);
    system("cp $makeup  $cinfile");
    close(TEMP);        # Close and unlock the files everything


    $clgfile = "$directr/Clog.$RUN"; # Corsika log file


    # Construct the name of the corsika output file as corsika does
    $ghn=6-length($RUN);
    $zpad="";
    for ($j=0; $j< $ghn; $j++) { $zpad="0$zpad"; }
    $coufile="$directr/DAT$zpad$RUN";         # particle file


#    $choufile="$directr/CER$zpad$RUN";        # cherenkov light file

# run corsika  All output goes to Clog.runnumber.gz
    if (system("cd $HOMECORSIKA; $corsika < $makeup 2>&1  | gzip > $clgfile.gz  ")!=0){
print STDERR "problem executing $corsika\n";
exit(-23);
    }
# run the wact_read program on the telescope.dat file
print STDERR "$wactnm\n";
    if (system("cd $HOMECORSIKA/WACT; $wact_read $wactnm")!=0){
        print STDERR "Problem executing !! $wact_read\n";
        exit(-23);
```

```
    }


    if (system("rm $makeup")!=0 ) {   # remove the temporary input file
print STDERR "problem removing $makeup\n";
exit(-23);
    }


     sleep 3;
############################ MILAGRO GEANT #############################
#
# Create a new input file for Geant
#

# Open the input GEANT input file and lock it.
    open(FILE,"+<$ginfile") || die "Couldn't open $ginfile\n";
    &lockfile(FILE) || die "Could not lock file $ginfile\n";

#  Create a new GEANT input file name
    sleep 3;
#    $tt=time-963531000;
#    $makeup="$ginfile.$tt.$HOST.$RUN";
    $makeup="$HOMEMILAGRO/detector.txt.$tt.$HOST.$RUN";

#  Open the new file an lock it, just in case
    open(TEMP,">$makeup") || die "Couldn't open $makeup\n";
    &lockfile(TEMP) || die "Could not lock file $makeup";


    $temp=<FILE>; # eat a line
    print TEMP "'$coufile'\n";  # Put name of particle file into input file


    $temp=<FILE>;
    $telfile = $wactnm;
    $wact_info = $wactnm.'.txt';
    $wactnm = $wactnm.'.core';
    print TEMP "'$wactnm'\n";

    $temp=<FILE>; # eat a line
```

```
# Create the output file name.  Put it in the same directory as
# previous output files.
    $temp =~ /'(.+)'/;

    @el=split(/\//,$1);              # Chop up the directory name




    $goufile = $directr."$type.$RUN.$tt.dat";
    print TEMP "'$goufile'\n";

    while($temp=<FILE>) { print TEMP $temp;  }

    close(FILE);
    close(TEMP);

# geant log  file
    $glgfile= $directr."/Glog.$RUN.gz";


#  Run Geant, sending standard output and standard error to a zipped file.
    system("cd $HOMEMILAGRO ; $geant $makeup < card 2>&1  | gzip> $glgfile;sync");
    $newtelfile = "$type.$RUN.$tt.CER";

    $wact_dat = $directr."$type.$RUN.$tt.wact.dat";
    system("cd $directr;  mv $wact_info $wact_dat");
    $final_dat =$directr. "$type.$RUN.$tt.wact_milagro.dat";








    sleep 3;
    system("cd $HOMECORSIKA/WACT/; $finishup $wact_dat $goufile $final_dat");
    system("cd $directr; mv $telfile $newtelfile");
```

```
    system("cd $directr;  gzip $final_dat");

    system("cd $directr; gzip $goufile");              # zip up the output file
    system("/bin/rm $makeup");             # remove the input file
    system("cd $directr; /bin/rm -f $coufile");
    system("cd $directr; /bin/rm -f $coufile.dbase");
    system("cd $directr; /bin/rm -f $coufile.long");# remove corsika output
    system("cd $directr; /bin/rm -f $wactnm");
}


#########################################################################
# Finally, we notify the user that the job is done. #
#########################################################################

$string = "Completed runs $MAXRUNS on $HOST";
system("echo '$string' | mail $USER");


#########################################################################
#     Subroutines #
#########################################################################




 sub lockfile {
     # We pass fnctl a F_WRLCK and a bunch of 0's
     # This works on Linux and IRIX and should work on most other machines
     # if the first element of the fnctl flock structure is the type of
     # lock.  Is this a POSIX standard?

     $bub=$_[0];
     $lockz = pack('s s l l s', &F_WRLCK, 0, 0, 0, 0);
     $gg=fcntl($bub, &F_SETLKW,$lockz ) || -1;   # do a blocking write lock

     return $gg;
 }
```