RunMCv3.3 Manual

S. Chekanov

February 8, 2005

HEP division, Argonne National Laboratory, 9700 S.Cass Avenue,
Argonne, IL 60439 USA

1 Contents

Contents

2 Introduction

RunMC is an object-oriented framework aimed to run various FORTRAN Monte Carlo (MC) programs for simulation of the particle production in high-energy collisions between elementary particles. It allows to fill 1D and 2D histograms, as well as to create ntuples (ROOT trees). The program is written in C++, and interfaced with ROOT [1, 2] and CLHEP [3, 4] library classes. To include user-defined histograms, you need a minimum coding. It comes already with a number of histograms, which can be used to look at different event characteristics for many high-energy processes. One can also write a ROOT event tree with selected events for further analysis. This application is open-source licensed, and at this moment, it is available on the LINUX platforms with g++ and f77 compilers.

3 Installation under LINUX

After downloading the file "runmcv3.2.tgz" from `http://www.desy.de/~chekanov/runmc` gunzip and untar this file, i.e. "gunzip runmc.tgz; tar -vxf runmcv3.2.tar". This creates the directory RunMCv3.2. To install the RunMC program from the source file, you need: g++, g77 compilers. Also, you have to have "sed", "zip/unzip" and "acroread" to read the user manual. Normally, all these packages are already installed on Linux PCs.

Before further installation, you should have:

1. ROOT C++ package, `http://root.cern.ch/`;
2. CERNLIB library, `http://wwwasd.web.cern.ch/wwwasd/cernlib/version.html`;
3. CLHEP class library `http://wwwinfo.cern.ch/asd/lhc++/clhep/`;
4. Source-Navigator; `http://sourcenav.sourceforge.net` - optional

Source-Navigator is optional and is usually installed by default in any linux PC. The RunMC program will be functional without it. Source-Navigator just helps a developer to understand complex relationships between elements of a program's source. Check it by typing "snavigator" from the shell prompt. If it is not installed, one may get binary and source code from `http://sourcenav.sourceforge.net`

The simplest way to compile this package is:

- cd RunMC(version)/build i.e. go to the directory containing the package's source code, and edit "install.sh" file: (a) specify the location of the CERNLIB library (default location is CERN_DIR = /cern/pro/lib); (b) specify the location of the CLHEP library, library itself and the include files;
- RUNMC variable should be set to the RunMC(version) directory, such that path gives the location to the RunMC directory. Then, add $RUNMC/bin to PATH and $RUNMC/main/ws/lib to LD_LIBRARY_PATH.

sh family:

```
export RUNMC=<path>/RunMC
export PATH=$RUNMC/bin:$PATH
export LD_LIBRARY_PATH=$RUNMC/main/ws/lib:$LD_LIBRARY_PATH
```

csh family:

```
setenv RUNMC <path>/RunMC
setenv PATH ${RUNMC}/bin:${PATH}
setenv LD_LIBRARY_PATH ${RUNMC}/main/ws/lib:${LD_LIBRARY_PATH}
```

Type "echo $RUNMC" in the bash shell, to be sure that you set this variable properly.

It is also advisable to check your ROOT installation before compiling the RunMC. Type "cd $ROOTSYS/test" and compile ROOT examples by typing "make". If ROOT is installed correctly, you should be able to compile all examples in the "test" directory.

To compile RunMC, go to the "$RUNMC/build" directory and type "./install.sh" to compile the package. This script makes some most elementary checks of your settings. The installation takes 3-10 minutes depending on CPU. After the first compilation, next compilations should take significantly less time, since all Monte Carlo libraries should be installed already.

You can remove the program binaries and object files from the source code directory by typing "make cleanall". To remove only object files, do "make clean". To remove GUI-related files, do "make cleangui". To remove the Monte-Carlo-related files, type "make cleanmc".

Here, "$RUNMC" denotes the installation directory of the RunMC project, and will not be mentioned again; all the directories discussed below are assumed to be located in this directory.

Go to the "proj" directory and type "runmc" to start RunMC GUI. As long as you set $RUNMC, you can execute "runmc" from any directory.

4 Installation under Windows

At present, RunMC can be installed under on Windows with Cygwin/X11.

- Install recent version of Cygwin (free package, see `http://www.cygwin.com`). Install the base packages, and, in addition, the following packages: zip, unzip, rxvt, g++, g77, make, X11, X11lib (X11 library for X11 development, this is necessary for ROOT installation). Use "UNIX" file system for the CYGWIN installation;
- Install ROOT. How to build ROOT under Cygwin is documented on the ROOT web page.
- Download the main RunMC, and gunzip, and untar the package as described in the main RunMC web page;

- Download an additional package called runmc(version)-win32. You can copy it from: `http://www.desy.de/~chekanov/runmc-win32.html`. It includes CLHEP and a few CERNLIB libraries which are necessary to build Monte Carlo models. It also has several scripts to change options in RunMC makefiles.
- "gunzip runmc(version)-win32.tar.gzi"; tar -vxf runmc(version)-win32.tar (Note: this package may have a version number, (version), therefore, use appropriate name for runmc(version)-win32.tar.gz file). This will create the build-win32 directory;
- Now set up the bash enviroment:.

  # set RUNMC variables
  export RUNMC=/home/sergei/work/RunMC # specify location of RunMC
  export RUNMCwin32=‘cygpath -da $RUNMC‘ # do not modify
  export PATH=$RUNMC/bin:$PATH

  if this is done, do "source .bashrc";
- Copy build(version)-win32 directory to $RUNMC/, i.e do "mv build-win32 $RUNMC/";
- Go to $RUNMC/build-win32 and type "./install.sh". This should compile the RunMC program.
- If you plan to use Source Navigator, you should install it under Windows as well. In this case, set correct location of snavigator and acrobat reader in the scripts "$RUNMC/main/runmc/proj_view.sh" and "$RUNMC/main/runmc/show_manual.sh"

See details in `http://www.desy.de/~chekanov/runmc-win32.html`.

In order to use RunMC together with ROOT, first you should start X-windows on Cygwin, and only then start the RunMC GUI program. Read more details how to start X-windows on Cygwin. The best prompt on Cygwin is RXVT, so do not forget to install it.

5 The RunMC user interfaces: C++ vs Java

Presently, two user interfaces are available: one is written in C++ ("runmc") and a user interface written in Java ("jrunmc"). Try to run them typing "runmc" or "jrunmc". For the latter, you may need to modify "$RUNMC/bin/jrunmc" script if the path to the java is not the standard one. Below, only "runmc" GUI is described.

6 Getting started

The simplest way to learn about RunMC is to run "runmc" program, which executes RunMC GUI. Go to any empty directory and type "runmc". This executes the RUNMC

GUI interface where you can specify details of your Monte Carlo run. However, it is preferable always execute the program from "proj", since this is important requirement when the RMC project files are automatically loaded (which may contain MC control cards).

At present, the following MC models are included to RunMC: PYTHIA6.2 [5], HERWIG 6.5 [6], ARIADNE 4.12 [7], LEPTO 6.5 [8], AROMA 2.2 [9], CASCADE 1.2 [10], PHOJET 1.05 [11], RAPGAP 3.1 [12]. When RUNMC GUI is used, one can choose one of these MC programs. The final state used in your calculation (or/and in the RUNMC output trees) can be: all stable particles, "partons", "charged particles", "all particles" (which is equivalent to the HEPEVT record) , "stable+all" (fill both stable particles and, in addition, complete HEPEVT record), and "user-defined". The option "stable+all" allows to reconstruct some quantity from the final state (PP or PJ vectors, see the definition of the "heplist" class), and at the same time one can preselect events using information on all generated particles (HP vector, which is a copy of HEPEVT event record). Obviously, this option is the slowest. The last option is used to fill the final state in the "proj" directory.

Select the number of events to be processed, type of initial particles, the nucleon structure function (PDF). You also can set kinematic cuts applied on the final state in the transverse momentum (PT) or pseudorapidity (Eta) of particles, if particles were selected, or on jets, if your choice was jet reconstruction. If, instead of the "particles" option, one selects "hadronic jets", all kinematic cuts will be applied on the jets, without any kinematic constrains on the particles used to form jets. For DIS, one can set up the event kinematics in Q2, Bjorken x, y, or W.

Then, you can select the histograms to be filled by clicking on the "Histogram" bottom. You should see the window "Variables". Selecting variables in this window will set the names for future histograms. Then one should set the minimum and maximum range of the histograms, as well as the number of bins. You can also select two-dimensional (2D) histograms using the option from the right side.

You can select up to 8 one-dimensional histograms at the same time (or 4 2D histograms), but in fact, the user can specify up to 500 1D histograms (or 250 2D histograms), but only 8 can be viewed during the run (see below). If you click on "more histograms", you will get a spreadsheet "Histograms", where you can define more than 8 histograms (use the "Variable" window to do this). This spreadsheet should allow you to delete, copy, paste selected histograms.

Then, by clicking on "Options", you can select the size Canvas size, update interval for histograms (the larger number, the less CPU consuming run), color scheme (for 1D or 2D histograms separately, use plain style to look scientific!), and a renice level. Note that the log-scale option at present works only if you set a single histogram for on-line viewing (of course, when histograms will created, one can use ROOT to set log scale). You can also specify whether you want to fill the histograms for events, or use different normalizations. One can normalize histograms to unity, but also one can calculate differential cross sections (in "nb" or "pb") after taking into account the bin size and the luminosity for this MC run. Note that the histogram-normalization option works only for 1D histograms for a moment.

Now you can select the output. Press "Select output". Three types of the output files are possible:

- The histogram file output. It has the extension ".root";
- The HEPEVT event record with output file, which has the extension ".htup"; The HEPEVT output was slightly modified, to make it simpler. Read Monte Carlo manuals for definitions of the variables. Note: the output file can be very big!
- The RUNMC event record. The output file has the extension ".rtup". This is a reduced HEPEVT output. The output contains only particles selected from the RunMC GUI: all particles, stable particles, charged stable particles;

No any kinematic cut is applied for such output, i.e. you cannot set any cut on the final state from RunMC GUI. To keep the output file small, only particle ID, charge (times 3), mother, daughter and Px, Py, Pz, E of particles are filled. The size of the output file is by a factor 4 smaller than that for the complete HEPEVT record. Note that double precision values were replaced by float values, to keep output small.

Now specify the name of your project, and save it by clicking on "Projects→save MC". Another type of projects (which called "RMC", and has the same extension) will be explained latter. The option "Save MC" saves the current status of RunMC GUI. Note, that even if you will not name your project, RunMC will save the present project in the default "analmc.mc" file automatically.

If you do want to preselect events for your RunMC tree, you can do this using three methods:

1. Set appropriate values in the subroutines which initialize the MC programs. All user files are in the "proj" directory (read about this below). FORTRAN subroutines which initialize Monte Carlo models are located in the "proj/ini" directory. For every MC model, there is a separate subroutine, in which new parameters can be specified. (You have to read MC manuals for this, and use FORTRAN style of coding, of course). Then do "make" from the "proj" directory. It will rebuild all MC binary files in the "bin" directory. If you do this, the original MC HEPEVT record will contain the preselected events you want. The event class of RUNMC will also contain only preselected events, since this class is derived from the HEPEVT. Read about the "proj" directory in more details below. Note that this option is not preferable, but should be used if nothing else can help. Instead, use the steering cards (see below).

2. Another way to change MC parameters is to use the steering cards. Examples of the steering cards are located in the "example/steer/" directory. Copy necessary steering card to a directory were you execute RunMC GUI (i.e. where you type "runmc". The steering cards will be loaded automatically. Note than in the parameter settings, the functions in "proj/ini" directory are always the last one, i.e. parameters in steering cards can be overwritten by the settings in the "proj/ini" functions. The MC settings is the most complicated part, please carefully check the output log file.

3. One can preselect specific events, while generating fully inclusive events. Again, go to the "proj" directory, and put your user calculations in "user-select.cpp". By default, the function returns "0", i. e. it takes the event. If this function will return any other number, the event will not be taken. Note that this only affects

the RUNMC tree (i.e. file "project_name.rtup"). The event selection done in the "user-select.cpp" file does not change events in the original HEPEVT record, as well as in the simplified HEPEVT tree (i.e. the file "project_name.htup"), which is simply a copy of the FORTRAN HEPEVT common block. In addition, such selection has no affect on the event class used to make histograms on-line. The event class "main/inc/heplist.h" is described below. After a modification in "proj/user-select.cpp", type "make" to recompile the project. Read about the "proj" directory below.

Now select predefined histograms, and set minimum and maximum values and the number of bins. To start the run, click "Start". This will check all your setting, as well as will display the output file options. If everything looks OK, press "Run". After 1-5 seconds, you will see the ROOT canvas with histograms updated in real time. Note that for some models, like AROMA or CASCADE, this takes more time since these programs do some initial integrations before they start looping over events. You will see a blue progress bar indicating how many events were processed.

To verify your settings, click "Log" to see the log file of the MC run. This executes "RunMC Notepad" - an editor with the log file (".analmc.log"). The log file can also be found in your current directory, look at "project_name.log" file.

In addition to the log file, there a file with possible errors "project_name.err". If it is not empty, you will be notified by RunMC GUI.

You can stop the run at any time by clicking on "Stop". (You have to wait for ~ 3 sec after this, since the program needs to finish all calculations. This however depends on type of calculations.). When run is finished, you will see a pop-up window with the run information.

After the run finished, you should see a new button "View output" (below "More histograms" button), which has light color. Click on this button: you should see ROOT Browser with the opened file "analmc.root".

Then, using the ROOT built-in graphics editor, you can modify the canvas, i.e. change scale, put/remove labels etc. To exit, you should click on "Exit". It is advisable to quite the ROOT canvas fist, before you exit RUNMC. All files can be saved as a ROOT file, "project_name.root". If you do not want to look at the histograms latter using "project_name.root file", go to "Select output" to change this setting.

Several presentations options are available. If you select "Options→Only main Canvas", then if you exit the main canvas, the "Info" Canvas will not appear. Normally, howver, when you exit ROOT ("File→Quit ROOT"), a new Canvas will appear with the run information.

If something goes wrong, click on "Cancel". It kills all the runing programs associated with the RunMC program. Note: if you are running some RunMC executables in the background, they will also be killed without saving the results!).

7 Saving and loading the RunMC project file

You can save all you current RunMC GUI settings to a project file (by default, it has has the name "analmc.mc"). The extension "mc" is important. Go to option "Projects" and save this file as "MC".

You can restore this project using the same "Projects" option (click on "Read MC");

There is fast way to load project without using the option "Projects". In the shell prompt, type "runmc project.mc", where "project.mc" is the name of your input file.

8 The RMC physics projects

You cannot do complicated studies just by using RunMC GUI. First of all, only a small number of histograms are available from RunMC GUI. Monte Carlo settings also cannot be changed flexibly enough. Note that if you will copy steering cards to the directory where you execute "runmc", you can change some Monte Carlo parameters by modifying the steering cards. Examples of the steering cards are located in "example/steer" directory.

As was mentioned above, the user can include personal histograms and modifications to the MC settings by changing subroutines in the "proj" directory. This directory contains:

1. At least one "project.mc" file which defines the status of RunMC. This file can be loaded to RunMC GUI via "Projects→read MC", or using the command line: "runmc project.mc"

2. One file "user-comment.txt". It has comments on the structure of this projects, and is shown automatically when an RMC project is loaded. It has some useful information on how to run this project.

3. Directory "ini" with "dummy" subroutines for each MC model. Here you can set MC parameters using FORTRAN codings (read MC manuals how to do this. All MC manuals are located in "main/mcarlo" directory).

4. "user-init.cpp", "user-end.cpp", "user-run.cpp" - project initialisation, analysis and terminations subroutines. The RMC file "par-had-jetLHC.mc" illustrates how to use these files.

5. "user-select.cpp" - here you can set any cuts you want on hadronic final state or original HEPEVT record (if option "stable+all" used).

6. "user_afill.cpp" defines user variables for each event, each particle (1-particle density), or each 2-particle combinations (2-particle density). The same file is used to fill histograms for jets. Read comments in this file. One may also look at a very similar system file "main/src/afill.cpp".

7. This directory can contain steering cards for MC models. Settings in these cards can be overwritten by subroutines in "ini" directory.

8. "Makefile" and "Makefile_in". You need this to compile your project by typing "make". The same make files are used when you load RMC project using "Projects→load RMC";

9. ".snavigator.proj" - invisible file to be used with the Source-Navigator program. When you select "Projects→View RMC source" this file is used by "snavigator".

10. "user-name.txt" file which attributes names to your personal histograms which will be automatically included to RunMC during compilation. All names here should be exactly as in the "user_afill.cpp" file! File "user-pic.xpm" is just a picture illustrating your calculation.

The structure of the "user-name.txt" is:
$(RUNMC)/proj/user-pic.xpm, item1 "comment1"
$(RUNMC)/proj/user-pic.xpm, item2 "comment2"
....

All the subroutines defined above can be modified, and all executable for MC models can be recompiled by typing "make" from the "proj" directory.

In fact, the directory "proj" is "unpacked" form of the so called "RMC" file (or project), which contains everything you need to do the analysis. At this moment, this directory has only dummy functions. Go to this directory "proj", type "runmc" and save this directory from "Projects→archive RMC". In the shell prompt you will see the message "archive/analmc.rmc created" (of course, "analmc.rmc" could be any name which you type in RunMC GUI).

The file " analmc.rmc" is nothing but the "proj" directory, archived and compressed using the "zip" utility. This file is stored in the "archive" directory.

In the directory "archive" you can find a number of files with the extension "rmc". These files contain various physics projects (MC settings+histogram definitions as in the "proj" directory), saved exactly as you did before. This is a list of the present projects:

1. "default.rmc" - just "proj" directory with dummy functions;
2. "dis_kinematics.rmc" - project for DIS related studies;
3. "dis_strange.rmc" - project to study strange particles in DIS;
4. "event_shapes.rmc" - project to study the event shapes;
5. "charm_dis.rmc" - project to study D* cross sections in DIS;
6. "jets_HERA.rmc" - project to study jets at HERA (using the longitudinally invariant algorithm in the Breit frame);
7. "jets_LHC.rmc" - project to study jets at LHC; It also illustrates how one can preselect events in the user function "user-select.cpp".
8. "jets+charm_LHC.mc" - project to study jets at LHC. It also requires at least one charm particle in an event. This example illustrates how one can preselect events in "user-select.cpp" function. Note: option "stable+all" are used for the final state (fills both PP and HP vectors from heplist class).
9. "par-had-jetLHC.rmc"- project to study jets at LHC. This calculates jets at parton and hadron levels, and takes the ratio (the so-called hadronisation correction). This example is very important: it illustrates how to set user-defined calculations via "user-run.cpp" file, without using the internal RunMC functions. Also, the user can set some histogram options "user-init.cpp".
10. "view3d.rmc"- project to study top production at NLC. It creates a 3D view of the event, following to a very similar approach of the example tree2.C file in $ROOT-SYS/tutorials. In this example, RUNMC canvas is modified by the user. The settings for MC are done in "ini" directory

Try to load one of this projects to "proj" directory for testing. In RunMC GUI, click on "Projects→load RMC". You will be in the "archive" directory. Select one of the files with the extension "rmc". You can get RMC files from `http://www.desy.de/~chekanov/runmc`. This automatically recreates "proj" directory by copying new files from the selected RMC file. You will see a yellow message "Wait" and yellow status bar indicating the compilation progress: it starts compilations of all MC models using new files from "proj".

Note that if you had important files in "proj", and you have forgotten to save them before loading a new RMC project, such files will not be lost. Check the "main/tmp" directory. It should contain old "proj" directory (in zip format) with the name "proj_date.tmp". Note that files which are one week old are removed automatically.

When the compilation process finished, you can see that there are new histogram in RunMC. A new message will appear which contains information on this project (from the "proj/user-comment.txt" file). Say "OK" to close this window. Then, press histogram bottom, and study the list of the histograms. Now you can do necessary modification via RunMC and can start the run. (Note: if histograms in RunMC GUI were not updated, try to restart "runmc").

You can read comments on the current project using the "Projects→Info RMC" option menu.

The new "proj" file should contain at least one file with the extension "mc" which defines the status of RunMC GUI (which can be loaded as "runmc project.mc" or using "Project→read MC" option.

To debug files in "proj" directory, just type "make" from the shell prompt.

You can study the RMC project which is currently in "proj" using "Project→View RMC source". This executes the "Source-Navigator", which reads the file ".snavigator.proj". Please read the description of Source-Navigator in `http://sourcenav.sourceforge.net` or `http://sourcenav.sourceforge.net/online-docs/index.html`.

From the Source-Navigator program one can compile the RMC project: select "Tool-Build". If you did changes in the project, do not forget to update the project using Source-Navigator – this should update the project file ".snavigator.proj"

The user can define and fill external variables in "user-run.cpp". Load the "par-had-jetLHC.rmc" project to see how this was done. In this example, several histogram files are defined in the user initialization subroutine, and final states are defined in the "user-run.cpp" file. In this subroutine, jets are reconstructed using an external function "ktjet.cpp", and histograms are filled separately for hadrons and partons. The final state in RunMC GUI was set to "user-defined" final state, since the final state is filled by a user.

9 Analyzing ROOT output ntuples and files off-line

RunMC allows to analyse the ntuple files (with the extension "rtup" and "htup") off-line. Just select "Ntuple HEPEVT" or "Ntuple RUNMC", and load the selected files. Always select only one file. Then, if you want to chain over all ntuples, select this option on RunMC GUI. Do not forget to click "check ntuple files". This will show you the list of the files currently selected. All calculations for those files should be done exactly as for real Monte Carlo calculations. (Note that RUNMC ntuple format cannot be used for many calculations since this type of ntuples does not have much information on the event

structure).

Once you have the output files, "project.root", "project.rtup", or "project.htup", you can analyze the output using ROOT TBrowser. The output files can be analyzed latter, calculating complicated event characteristics off-line. Look at the "projexampleproj_hepevt" directory. It contains a demo program which reads "project.htup", and prints all values for each event on the screen. Analogously, "proj/example/proj_runmc" contains a demo program which reads "project.rtup" file. Please read "README" in these directories for more details.

10 RunMC event structure

To include an user-defined histogram or additional calculation, you need to know the event structure. The event structure is defined in the "main/inc/heplist.h" include file.

```
// Class characterizing  a  particle
   public:
   HepLorentzVector p;             // 4-momenta
   HepLorentzVector v;             // vertex
   double           mass;          // mass
   int                  pos;       // position in HEPEVT
   int                  id;        // identity (particle data group)
   int              charge3;       // charge times 3
   int              status;        // status of entry
   int              mother;        // mother
   int              mother2;       // point to second mother
   int              daughter;      // daughter
   int              daughter2;     // point to second daughter
   };

// Class characterizing  a jet (under construction)
    class jet{
    public:
        HepLorentzVector p;     // 4-momenta
        int              in;    // number of particles in this jet
        };

// Event record for Monte Carlo events
   class heplist {
   public:
      heplist();                   // constructor
       ~heplist();                 // destructor
      HepLorentzVector beam1;      // beam 1
      int id1;                     // PID of this particle
      HepLorentzVector beam2;      // beam 2
      int id2;                     // ID of this particle
      HepLorentzVector boson;      // Px,Py,Pz,E of exchange boson
```

```
      int idboson;                 // ID of boson
      HepLorentzVector lepton;     // Px,Py,Pz,E of outgoing lepton(DIS)
      int idlepton;                // ID of lepton

// Information on separate particles
      vector<particle>  PP;        // vector with particles

// Information on separate particles (options stable+all)
// contains complete HEPEVT record
     vector<particle>  HP;         // vector with particles

// information on event
    double      PTtot , PZtot,Etot;   // event PT,PZ,Etot

// information on jets
    vector<jet> PJ;             // vector with jets
    void fill_HEPEVT();         // fill all particles from HEPEVT (stable+all option)
    void fill_All();            // fill all particles from HEPEVT
    void fill_ALLstable();      // fill stable particles from HEPEVT
    void fill_Partons();        // fill partons from HEPEVT
    void fill_CHARGEDstable();  // fill partons from HEPEVT
    int  set_jetPtEta();        // set Pt and Eta cuts on particles
    int  set_particlePtEta();   // set Pt and Eta cuts on jets


// transformations for jets
    int   set_jetLABtoHCMframe();      // move jets to the HCM frame
    int   set_jetHCMtoLABframe();      // move jets to the lab from HCM frame
    int   set_jetLABtoBREITframe();    // move jets to  Breit frame
    int   set_jetBREITtoLABframe();    // move jets to lab frame from Breit frame

// for sepate particles transformation
    int   set_particleLABtoHCMframe();    // move jets to the HCM frame
    int   set_particleHCMtoLABframe();    // move jets to the lab from HCM frame
    int   set_particleLABtoBREITframe();  // move jets to  Breit frame
    int   set_particleBREITtoLABframe();  // move jets to lab frame from Breit frame
    int   set_JADEjets();                 // move to the JADE jets
    int   set_DURHAMjets();               // move to the Durham jets
    int   set_KTjets();                   // move to the KT long invariant  jets
  };
```

11 Structure of HEPEVT and RUNMC outputs

If you decide to write the entire event on the hard disk, you can choose between the HEPEVT record or reduced RUNMC record. Let's remind that the HEPEVT record can only be changed by the FORTRAN initialisation subroutine "proj/ini" (for each MC

separately). But you can do complicated selections of events for a reduced RUNMC record (in "proj/user-select.cpp" file, see above the description, and read the "proj/user-select.cpp" file).

Use the RunMC GUI option "Select output". The HEPEVT output file has "thepevt" tree. It has the following structure:

```
int NEVHEP;                       // read HEPEVT output
int NHEP;                         // NHEP - total number of particles
int ISTHEP[hepevt_local_max];     // ISTHEP(j)
int IDHEP[hepevt_local_max];      // IDHEP(j)
int CHA3[hepevt_local_max];       // charge times 3
int JMOHEP1[hepevt_local_max];    // JMOHEP(1,j)  in fortran
int JMOHEP2[hepevt_local_max];    // JMOHEP(2,j) in fortran
int JDAHEP1[hepevt_local_max];    // JDAHEP(1,j) in fortran
int JDAHEP2[hepevt_local_max];    // JDAHEP(2,j) in fortran
double PHEP_X[hepevt_local_max]; // PHEP(1,j)
double PHEP_Y[hepevt_local_max]; // PHEP(2,j)
double PHEP_Z[hepevt_local_max]; // PHEP(3,j)
double PHEP_E[hepevt_local_max]; // PHEP(4,j) energy
double PHEP_M[hepevt_local_max]; // PHEP(5,j) mass
double VHEP_X[hepevt_local_max]; // VHEP(1,j)
double VHEP_Y[hepevt_local_max]; // VHEP(2,j)
double VHEP_Z[hepevt_local_max]; // VHEP(3,j)
double VHEP_T[hepevt_local_max]; // VHEP(4,j)
```

The RUNMC output contains only particles which you select via the RunMC GUI (all stable, all charged partons etc.). The RUNMC tree is called "trunmc". It has the foolowing structure:

```
const int hepevt_runmc_max =4000;
    int NHEP;                       // total number of particles
    int IDHEP[hepevt_runmc_max];    // particle ID
    int ISTCHA[hepevt_runmc_max];   // 100*ISTHEP+(10 + charge times 3)
    int JMOHEP[hepevt_runmc_max];   // JMOHEP(1,j)  in fortran
    int JDAHEP[hepevt_runmc_max];   // JDAHEP(1,j) in fortran
    float PX[hepevt_runmc_max];     // PHEP(1,j)
    float PY[hepevt_runmc_max];     // PHEP(2,j)
    float PZ[hepevt_runmc_max];     // PHEP(3,j)
    float E[hepevt_runmc_max];      // PHEP(4,j) energy
```

See the example "mrun.cxx" in the "example/proj_runmc" directory which can read "project.rtup" file. Note it uses "float" instead of "double" precision numbers to save the disk space (and there is no reason to think that you really need double precision momenta of final-state particles even for LHC energies).

12 How to include user-defined histograms

The easiest, just load any "RMC" project from "archive" directory using "Project→load RMC". You will see changes in "proj" directory, and can study functions copied from

RMC. Below are some additional details:

You can include additional histograms in a few steps: Go to the user project directory, "proj", and edit "user-name.txt" file. You have to put the line:

$(RUNMC)/proj/user-pic.xpm,Something "description"

Here, "user-pic.xpm" is an icon which corresponds to this particular calculation (you can use other icons, of course). Then, "Something" is a unique name of the calculation.

The names should be defined as following:

- Any calculation which does not involve looping over all particles/jets should not have the symbol "@" at the beginning;
- If you calculate a single-particle density (looping over all particles in an event), your variable name should have "@" at the beginning. Note that "@" will be removed automatically for histogram titles etc.
- If you calculate two-particle density, i.e. calculate a variable for each particle pair, your variable name should have "@@" at the beginning.

Once you defined the title of your calculation, put the description of the calculation after a space. You can put up to 500 histogram definitions. Each definition should be on a new line.

There are a few types of histograms:

1. A histogram for a value characterizing the entire event. You have to define the variable "*getval" (the star in front is important!) to do this in the function "user_afill.cpp". It is very important that the name in "if" statement of this function should match the histogram name in "user-name.txt". You can include as many calculations as you like, but they all should be included using "if-else" statements.
2. A histogram representing a variable for each particle. Do this exactly as before, only specify particle counting number in your variable in "user_afill.cpp" function. See examples in the RMC files. The title of a calculation should contain "@" as explained above.
3. A histogram for each particle pair. Do this exactly as before, but specify additional 2 arguments in this variable representing particle number in "user_afill.cpp" function. The name of the calculation should contain "@@" as explained above.
4. A histogram for each jet. In this case use additional variable "jet" string in "if" statement of the function "user_afill.cpp".

When you did this, you have to recompile the project, type "make clean" and then "make".

You can find a number of examples in "main/src/afill.cpp". You should not modify this file. Better, load one of the physics projects from "archive" as described before.

Note that the name convention described above is not needed when a user fills the histograms in "user-run.cpp" file. See "par-had-jetLHC.rmc" project. In this example,

the final state and well as histograms are filled without using RunMC directly. In addition, user can initialize the calculations in "user-init.cpp"

If you need more histograms, select "More histograms". You will see the spreadsheet "Histograms" and the window "Variables". Click on the variable name in the "Variable" list, and this histogram will be added to the spreadsheet. This should allow you to add and edit as many histograms as you like. If you need to replace some entry on the spreadsheet, click on the raw you would like to replace, and select new new from the histogram selector. If you select "none", this will create an empty histogram which you can replace with some other histogram. Then close the spreadsheet ("OK") and save the histograms.

If you now save the project as "Project→save MC", you will see that your histograms are added to the end of the file "project.mc".

In total, 500 histograms can be included to your project. Instead of using the spreadsheet "Histograms", one can use any your favourite editor and correct the file "project.mc". Go to the end of this file. For each new histogram, you need to add 2 lines:

H-NAME "Name of the histogram" "comment or definition" H-INFO "Dimension" Xmin Xmax "No of bins" "weight"

where "Dimension" means "1" or "2", i.e. 1D or 2D histogram, Xmin and Xmax are the minimum and maximum values for histogram, and "No of bins" is the number of bins (should be always integer). Xmin and Xmax, weight are real numbers. Note: do not use more than 20 characters to for the title of your histogram.

If you want to use "2D" histograms, set "2" after "H-INFO". The number of histogram with this option should be always even, otherwise RunMC will fail to calculate how many 2D histograms you want and will report an error.

You can also change any option in "project.mc" by hand. To learn about this, use RunMC GUI and study what it does when you create the "project.mc" file.

13 User-defined settings for MC models

At this moment, all Monte Carlo models are initialised (via the RunMC GUI) for fully inclusive events (AROMA- for charm events). You can change this if you will include your Monte Carlo settings into the files located in "proj/ini". You have to use FORTRAN coding for this. Please refer to Monte Carlo manuals on how to specify your new Monte Carlo settings (they are in "main/mcarlo/" directories. The names of the subroutines in "proj/ini" are self-explanatory. The subroutine "set-lepto-ariadne.f" corresponds to LEPTO with ARIADNE (for ep DIS only!). When you did this,go to the "proj" directory and recompile it by typing "make". Note: PYTHIA does not work well for DIS, use it for photoproduction.

Another way to change MC parameters is to use steering cards (i.e initialisation files). Examples of the steering cards are located in "example/steer/" directory. Copy necessary steering card to a directory were you run RunMC GUI (i.e. where you execute "runmc"). The steering cards will be included automatically. Note that in the parameters settings, the functions in "proj/ini" directory are always the last one, i.e. parameters in steering cards can be overwritten by the settings in "proj/ini functions.

The steering cards can be loaded and edited using RunMC steering card editor. Use "MC settings". You should see a spreadsheet-type of editor for the steeringcards, where you can type new parameters for MC models. Note: you loaded RMC file, which may

contain some steering cards, it is important to start RunMC from the "proj" directory, otherwise the RunMC steering-card editor will never find the current RMC steering card.

Note that the steering card for HERWIG is still rather simple, i.e. only a few parameters were implemented so far. Look at the "main/mcarlo/herwig-65/RUNMC-herwig.steer.f" to check which parameters can be included. Therefore, the best way to change the HERWIG parameters is to use more flexible function in "proj/ini".

14 How to include user-defined RMC projects

There are three functions in the "proj" directory: "user-init.cpp" (initialisation), "user-run.cpp" (for event calculation), "user-end.cpp" (for job termination). You can access the HEPLIST class for every events and do any calculation you want to, including filling of a ntuple. If you have an external function you want to link to the project, you should add an additional line in the file "Makefile_in" with the name of new file. Then you have to recompile the project from the "proj" directory.

You can also include user-defined labraries, use variable "$(PROJ_LIBS)" in Makefile_in to do this. Look at the example "hztool.rmc".

The example "par-had-jetLHC.rmc" illustrates how to fill user-defined histograms. It does not used the final state option from RunMC GUI (it is set to "user-defined"). Instead, the final states (partons and hadrons) are filled inside "user-run.cpp". Histograms are also filled inside this function. Drawing options and some protection conditions are redefined in the "user-init.cpp" function.

You can save and restore the "proj" directory using "Project→archive RMC" and "Project→read/build RMC". This will create a file with the extention "rmc" in the "archive" directory. You can restore this file latter using "Project→read/build RMC".

It is very easy to make a personal RMC project. First, load the default RMC project file using "Project→load RMC" and select "default.rmc". Then start Source Navigator using "Project→view RMC source". Now you can edit the project. Then "Update" the project from the Source Navigator and exit. Finally, you should rename the project on the RunMC GUI and save it as RMC file. Alternatively, just go to "proj" and edit the RMC source using any editor.

15 How to access some MC parameters

During MC running, sometimes you need to access MC parameters and plot them. To do this, you should copy them in the array REAL rUSER(50) (from "getevkin.inc"). This should be done in the main loop of the FORTRAN program.

For example, for PYTHIA6, copy necessary variables to the rUSER array in "main/mcarlo/pythia-62/RUNMC-pythia6.f" after the line "nev=nev+1". Then you can access this variable in any of your function (like "user_afill.cpp" under "proj") if you specify the include file " #include<getevkin.h>" and put the line "extern getevkincommon getevkin_ ;", You variable should be seen as "getevkin_.rUSER[0]", "getevkin_.rUSER[1]" etc. Look at the example in "dis_kinematics.rmc" where the kinematic variables Q^2 etc were taken directly from the LEPTO model.

Similarly, you can do the same with any MC model.

16 How to use RunMC without GUI

You can run any MC model without the GUI. You need GUI only to create "name.mc" file and to be able to communicate with the MC. But this can be done without the graphical interface as well.

Before doing this, it is advisable first to run RunMC GUI and create a project file. Go to "Options" and select "Do not update" and "Do not show plots". Then you can save this project (the default name is "analmc.mc").

Next step is to run this MC model. Look at "bin". This directory has binary files for each MC model. These files read the input file ".analmc.ln" located in your current directory. This is just a link to your input project file, say "analmc.mc". One can edit this file, to set additional settings you want. Then, execute the MC binary file in shell. For example, to run ARIADNE, just type: "analmc.ariadne analmc.log"

It is likely that you will get the message in the log file:

PROCESS PID=2331 renice 10 2331

2331: old priority 0, new priority 10 output file: analmc

$RUNMC/pipes/pipe_<date>.stop has 1

where "date" is a string containing the time when your job was started. This means that the communication file "pipe_<date>.stop" forbids this run since it has "1". To start the run, you should replace "1" in "pipe_<date>.stop" by "0". If you set "0" in this file, you can run this MC. If you will want to stop the MC run, again set "1" to "pipes/pipe_date.stop". After 3-5 seconds, the run should be terminated and all histograms will be saved. Since all pipe files are not identical, you can run as many jobs as you like at the same time.

If you get the message that "pipe_date.stop" is not find, you should create this file in the "pipes" directory and make sure that the name of this file was set correctly in the "analmc.mc" file (it should not contain any extension like ".stop"!).

The easiest way to learn this – use first RunMC GUI and analyse changes in the project file "analmc.mc" file.

In addition to the "pipe_<date>.stop" file, each run has another file called "pipe_<date>.events", which contains the number of generated events.

The RUNMC will clean the "pipes" directory from the old files. At present, it will clean pipe files which more than 20 days old.

17 How to load RMC projects without GUI

First, remove your current "proj" directory. Copy a RMC file to the top $RUNMC directory, for example "cp $RUNMC/archive/default.rmp $RUNMC/". Then unzip it, "unzip default.rmp". This will create new "proj" directory with new files. Then recompile the project by executing "make" from the "proj" directory.

18 Random seeds for MC models

Every time when you start new MC run, seeds for random numbers used by the MC models are different. Seeds are calculated from many numbers which include the current

time, job id etc. Just check few first lines in "project.log" file to see that the numbers are indeed different every time you star the run.

19 Specific issues how to run MC

19.1 PYTHIA, HERWIG, LEPTO, AROMA models

For these models, just include your settings to $RUNMC/proj/ini" and recompile the project from "proj". In addition to this, you can use steering cards in your current directory. Examples of the steering cards are in "example/steer/(MC model)" for each MC model. Note: settings defined in "proj/ini" are set at the very end, i. e. your parameters defined in the steering cards can be overwritten via files in "proj/ini".

19.2 CASCADE model

For this model, you need a steering card and a file with unintegrated gluon densities. Unintegrated densities can be found in `http://www-h1.desy.de/~jung/cascade`. You should copy one of it to your current directory and rename it to "ccfm.dat". One file with example unintegrated PDF is in the "main/mcarlo/cascade-12/steer" directory.

RunMC contains example steering cards for several processes in "main/mcarlo/cascade-12/steer". You should copy one steering card to your current directory and rename it to "cascade.cards". The file "cascade.cards" should be in the same directory where you execute "runmc".

19.3 PHOJET model

This model is not supported by the author, therefore, the PHOJET model was implemented as it is, with a minimal coding necessary for RunMC. The cards files for initialisations, located in "example/steer/PHOJET", are not standard and are different from other Monte Carlo models. Because of this reason, it is impossible to edit these steering cards using RunMC GUI. This model also requires input PDF file, "fitpar.dat" (located in "main/mcarlo/phojet/files").

In addition, there is a rather different concept to fill histograms. In case when PHOJET steering cards are used, RunMC first fills a temporary binary file "runmc.tmp" with events, and only then it start to read it and fill histograms. Thus, one may notice some delay at the very beginning of the run, but this compensated by a very fast histogram filling. Make sure that you have enough disk space (50k events usually require 60M for HERA experiment).

20 Look & Feel of RunMC GUI

By default, it uses "modern" style. You can change this by copying one of the files located in the "main/ws/styles" to your home directory, renaming it to " .wsrc". Fonts for RunMC GUI could be changed, see the Section "Possible problems".

21 User Preferences

If you do not like to use the "snavigator" to work with the RMC project files, one can specify your favourite IDE or text editor in the file "main/runmc/proj_view.sh". This script is called when you select "View RMC source" from RunMC GUI.

You can also change "acroread" to "xpdf" in a script located in the directory "main/runmc/"

All other preferences to view the RMC files should be set directly from the Source Navigator.

22 How to share RunMC physics projects

All user-defined settings and calculations (including those necessary to update RunMC GUI) are in the "proj" directory. You should create RMC file of from this directory using RunMC GUI "Project→archive RMC". Then you can send just created RMC file located in the "archive" directory.

23 Possible problems

23.1 ROOT canvas corrupted:

ROOT canvas with updated histograms is corrupted when overlapped with another window. Answer: you have to activate "BackingStore" in X11: (see PAW howto) LINUX: It depends which daemon is in charge of putting up the login box on the X-Windows screen, and start the X-Window session at login time. It can be xdm, kdm (provided with the KDE environment) or gdm. If it is xdm or kdm the line starting the X server in the file /usr/lib/X11/xdm/Xservers should not contain the option "-bs" it should look something like:

```
:0 local /usr/X11R6/bin/X +bs
```

For XFree86 4.0.1, the Screen section of the file /etc/X11/XF86Config (or /etc/X11/XF86Config-4) should be something like:

```
Section "Screen"
            .
            .
             Option "backingstore"
            .
             .
          EndSection
```

23.2 Text fonts on RunMC GUI

If there are no any text on RunMC GUI then the problem is with X-Windows fonts. (this could happen for some Linux PC). The reason: X-fonts with size "14" are not available. This size was used for all text labels on RunMC GUI. This problem can easily be fixed without installing additional fonts. Look at the file "main/gui/runmc_gui.cpp".

You will see the lines:

```
char* fon[] = {
,"14 iso8859 * bold * r *", ! font was used as default for RunMC GUI
"10 iso8859 * * * r *",
"18 iso8859 * * * r *",
"20 iso8859 * * * r *",
"22 iso8859 * * * r *",
"24 iso8859 * * * r *",
"26 iso8859 * * * r *",
"30 iso8859 * * * r *",
NULL}
```

Edit the first line, replacing "14" by "15" or "16" (best quality!). One can remove "bold" by "*" etc. Then recompile RunMC GUI, typing "make -f Makefile-linux" in the "main/gui/" directory, and copy the binary file to your bin directory, i.e. "mv runmc_gui $RUNMC/bin/runmc_gui" . Fonts should appear when you will restart runmc. If not, run "xfontsel" and check what fonts are available.

24 Tabular Summary

Program name:	RunMC
Version:	3.2
Date of last version:	November 15, 2004
Author:	Sergei Chekanov
Size:	11M (Linux/Windows-Cygwin), 12M addition (Windows-Cygwin)
Operating system:	Linux, Windows/Cygwin
Additional packages needed:	CLHEP, ROOT, CERNLIB with PDFLIB (only for Linux)
Program requirements:	g77, g++, make, zip/unzip, sed, X11
Programming language:	C++, C, Fortran, bash
Availability:	`http://www.desy.de/~chekanov/runmc`

25 Bibliography

References

[1] R. Brun, F. Rademakers, ROOT: An object oriented data analysis framework, Nucl. Instrum. Meth. A389 (1997) 81.

URL http://root.cern.ch/

[2] R. Brun, F. Rademakers, P. Canal, M. Goto, Root status and future developments, ECONF C0303241 (2003) MOJT001.

[3] L. Lönnblad, CLHEP: A project for designing a C++ class library for high-energy physics, Comput. Phys. Commun. 84 (1994) 307.

[4] M. Fischler, A. Pfeiffer, CLHEP - new developments and directionsThe proceedings of International Conference on Computing in High Energy Physics and Nuclear Physics (CHEP 2000), Padova, Italy, 7-11 Feb 2000.
URL http://wwwasd.web.cern.ch/wwwasd/lhc++/clhep/

[5] T. Sjöstrand, L. Lönnblad, S. Mrenna, Pythia 6.2: Physics and manual.

[6] G. Corcella, et al., HERWIG 6.5 release note.

[7] L. Lönnblad, ARIADNE version 4: A program for simulation of qcd cascades implementing the color dipole model, Comput. Phys. Commun. 71 (1992) 15.

[8] G. Ingelman, A. Edin, J. Rathsman, LEPTO 6.5 - a monte carlo generator for deep inelastic lepton-nucleon scattering, Comput. Phys. Commun. 101 (1997) 108.

[9] G. Ingelman, J. Rathsman, G. Schuler, AROMA 2.2 - a monte carlo generator for heavy flavour events in *ep* collisions, Comput. Phys. Commun. 101 (1997) 135.

[10] H. Jung, The CCFM monte carlo generator CASCADE, Comput. Phys. Commun. 143 (2002) 100.

[11] R. Engel, PHOJET: manual, Univ. Siegel preprint 95-05.
URL http://www-ik.fzk.de/~engel/phojet.html

[12] H. Jung, Hard diffractive scattering in high energy ep collisions and the monte carlo generator RAPGAP, Comm. Phys. Commun. 86 (1995) 147.