CRMC exercise
**CRMC**

**Cosmic Ray Monte Carlo:** common interface to the hadronic interaction models used in Cosmic Ray physics

Reference: C. Baus, T. Pierog and R. Ulrich. To be published (2016) (please ask colin.baus@alumni.kit.edu when needed)

Available models:
- post-LHC: EPOS-LHC (-m0), QGSJETII-04 (-m7), SIBYLL 2.3c(-m6)
- pre-LHC: EPOS 1.99 (-m1), QGSJET01 (-m2), QGSJETII-03 (-m11), DPMJET 3.06
Where to get CRMC?

CRMC webpage:
https://web.ikp.kit.edu/rulrich/crmc.html

CORSIKA school: distributed on USB/VM

CRMC (Cosmic Ray Monte Carlo package)

CRMC is a package providing a common interface to access the output from event generators used to model the secondary particle production in hadronic collisions. The interface is linked to a wide range of models, however, the unique focus are the models using in the simulation of extensive cosmic ray air showers.

The CR models are all build on top of the Gribov-Regge model. The models include:

- EPOS 1.99/LHC
- SIBYLL 2.1/2.3
- QGSJet 01/II.03/II.04

but there are several more included. When you use any of these models, also cite the original papers!

The project has been incorporated as an interface to the CR models by the ATLAS and CMS Collaborations. In many QCD/Forward publications and data analyses the CR models have been used. The description of background in particular including diffraction is often complementary/better than in the various tunes of Phythia/Herwig/etc. Also the Pierre Auger Observatory, NA61, LHCb, TOTEM used CRMC so far. Please let us know about your project, too!
How to install / compile?

pre-requisites: BOOST, HEPMC, CMAKE
(all present on VM)

Principal steps:
1. unpack
2. configure
3. prepare install directory
4. cmake
5. make
6. make install

⚠️ 2. Configure step

Not all models active by default. E.g. to use QGSjetII04 need to edit ‘CmakeLists.txt’ to activate
Configure

In main directory of crmc-archive: CMakeLists.txt

CMAKE_MINIMUM_REQUIRED (VERSION 2.6)
PROJECT (crmc)

# Enable/Disable models to be built
OPTION (__CRMCSTATIC__ "Build with static library" OFF) # if ON should not combined DPMJET/PHOJET/PYTHIA
because they use different version of pythia (for dynamic library no problem)

OPTION (__QGSJET01__ "Build with model" OFF)
OPTION (__GHEISHA__ "Build with model" OFF)
OPTION (__PYTHIA__ "Build with model" OFF)
OPTION (__HIJING__ "Build with model" OFF)
OPTION (__SIBYLL__ "Build with model" ON)
OPTION (__PHOJET__ "Build with model" OFF)
OPTION (__DPMJET__ "Build with model" ON)
OPTION (__QGSJETII03__ "Build with model" OFF)
OPTION (__QGSJETII04__ "Build with model" OFF)

if (CMAKE_INSTALL_PREFIX INITIALIZED TO DEFAULT)
How to install, step-by-step

1. unpack
   
   tar -xvzf <crmc-archive>
   cd <crmc-dir>

2. configure
   
   edit CmakeLists.txt

3. prepare install & build directory
   
   mkdir build
   mkdir ../<crmc-dir>-install
   cd build

4. cmake -DCMAKE_INSTALL_PREFIX=<path to crmc-dir-install> ../
   
   e.g. on VM for ISAPP school:
   
   cmake -DCMAKE_INSTALL_PREFIX=/home/isapp2018/work/crmc.v1.7.0-install ../
How to install, step-by-step II

5. make
   cross fingers
6. make install

To test crmc:

cd <crmc-dir-install>
./bin/crmc -m6 -T
Configuration file

Default location:

```
<crmc-dir-install>/crmc.param
```

Copy into your working directory!

* Defines stable/unstable particles

* can be used to configure interaction models
How to use?

Input: beam particles, beam momenta

e.g. create LHC p-p events at 7TeV with SIBYLL

```
crmc -m6 -i 2212 -p 3500. -l 2212 -P 3500.
Or
CRM  -m6 -i 2212 -l 2212 -s 7000.
```

(add <crmc-dir>/bin to PATH variable!)

On ISAPP VM:

`export PATH=$PATH:/home/isapp2018/crmc.v1.7.0-install/bin'`
Run the program by executing

```
./bin/crmc -h
```

to get the following help:

Options of CRMC:
- `h` [ --help ]
  description of options
- `v` [ --version ]
  show version and exits
- `o` [ --output ] arg
  output mode: hepmc (default), hepmc.gz, root, lhe, lhe.gz
- `r` [ --seed ] arg
  random seed between 0 and 1e9 (default: random)
- `n` [ --number ] arg
  number of collisions
- `m` [ --model ] arg
  model (0=EPOS LHC, 1=EPOS 1.99, 2=PYTHIA 8.23, 3=QGSJETII-04)
- `p` [ --projectile-momentum ] arg
  momentum/(GeV/c)
- `P` [ --target-momentum ] arg
  momentum/(GeV/c)
- `s` [ --sqrtS ] arg
  sqrt(s)/(GeV*2)
- `i` [ --projectile-id ] arg
  PDG or Z+10000+A+10
- `I` [ --target-id ] arg
  PDG or Z+10000+A+10
- `c` [ --config ] arg
  config file
- `f` [ --out ] arg
  output file name (auto if none provided)
- `t` [ --produce-tables ] [=arg[=1]]
  create tables if none are found
- `T` [ --test ] [=arg[=1]]
  test mode
- `x` [ --cross-section ] [=arg[=1]]
  calculate and print cross section only

For projectile and target id the following shortcuts are allowed:

- `1` = PDG(2212) = proton
- `2` = PDG(2212) = anti-proton
- `12` = PDG(1000000120) = Carbon
- `120` = PDG(211) = pion+ (not for -1)
- `120` = PDG(211) = pion- (not for -1)
- `130` = PDG(321) = kaon+ (not for -1)
- `130` = PDG(321) = kaon- (not for -1)
- `208` = PDG(1000002080) = Lead

Using these shortcuts with automatic output file name generation will create more human readable names.

The environment variable $CRMC_DOUT can be set to define the path of the output files, otherwise $PWD is used as default path.

**Example to generate 100 7 TeV pp collisions with EPOS LHC:**
```
bin/crmc -o hepmc -57000 -n100 -m0
```

**Example to generate 100 1.38 ATeV PbPb collisions with EPOS 1.99:**
Exercises

Get exercise sheet and crmc-exercises.tar.gz from the school webpage

What we are going to learn:

1. How to run CRMC in different configurations
2. How to run CRMC and create event files, run RIVET
3. How to write your own analysis in HEPMC
PDG ids

Proton 2212
Pi+ 211

Nuclei:  Z*10000+A*10

More:


Or see .pdf in crmc-exercise directory

Common options are printed by help: “crmc -h”
RIVET

https://rivet.hepforge.org/

Easily compare theory with experiment.

Experimental analyses pre-implented

‘rivet -h’
‘rivet - - list’ to see all analyses

‘rivet-mkhtml’

See exercise 2

Rivet

The Rivet toolkit (Robust Independent Validation of Experiment and Theory) is a system for validation of Monte Carlo event generators. It provides a large (and ever growing) set of experimental analyses useful for MC generator development, validation, and tuning, as well as a convenient infrastructure for adding your own analyses.

Rivet is the most widespread way by which analysis code from the LHC and other high-energy collider experiments is preserved for comparison to and development of future theory models. It is used by phenomenologists, MC generator developers, and experimentalists on the LHC and other facilities.

Features

- Object-oriented C++ framework for analysis algorithms
- Ever-increasing collection of analyses, more than 400 so far...
- Python interface and suite of user-friendly data handling scripts
- Large collection of generator-independent event analysis tools
- Automatic caching of expensive calculations, for efficiently running many analyses on each event
- Flexible system for fast detector effect simulation in BSM analyses
- Close matching of standard observables to experimental analysis definitions
- Reference data connection to HepData, avoid hard-coding

The Rivet user manual is kept up to date on the arXiv (1003.0694 [hep-ph]).

The C++ MC generators Herwig and Sherpa have convenient user interfaces for producing input events for Rivet analysis, as well as built-in Rivet support. Users may find the Sacrifice interface convenient for running Pythia 8, and the AGiLe steering interface useful for older Fortran generators like PYTHIA6 and HERWIG6.
The output format of CRMC is stored in HEPMC file format.

These events are stored in human-readable ASCII.

C++ library with same name that reads these files and provides iterators and things..

→ see exercise 3

For example: each particle in the file is a GenParticle object.

GenParticle has a member “momentum” of type FourVector.

http://hepmc.web.cern.ch/hepmc/