Muon Propagation Monte Carlo Written in Java
or
Muon Monte Carlo (mmc)

TABLE OF CONTENTS

---

1. COPYING

---

2. CREDITS

This program (MMC) was originally developed by Dmitry Chirkin at
Bergische Universitaet Gesamthochschule Wuppertal, Germany (2000)
and the University of California at Berkeley, USA (2001).

People who helped to test it, gave useful suggestions and
encouraged its development are (in alphabetical order)

Paolo Desiati
Heiko Geenen
Marek Kowalski
Predrag Miocinovic
Wolfgang Rhode
Kosta Schinarakis
Frank Schroeder

Thank you!

---

3. NEW RELEASES

The latest version of the mmc can be found at the MMC homepage:

http://dima.lbl.gov/~dima/work/MUONPR/
http://icecube.wisc.edu/~dima/work/MUONPR/

(all are the same).

Please refer to the HISTORY file for the latest changes. If your distribution
does not contain the full HISTORY file, it can be obtained from the MMC

homepage, which also contains an extensive description of the algorithm and its precision and summary of formulae used in MMC.

----

4. INSTALLATION NOTES

Since this program does not work with the default jdk included with Redhat (Kaffe), you will need to install the official SUN jdk distribution available at http://java.sun.com/j2se/. Versions 1.3x and 1.2.2 of Java were tested with the mmc v 0.08.7 and up.

Java ports for platforms, other than Solaris SPARC/x86, Linux x86 and Microsoft Windows can be found at http://java.sun.com/cgi-bin/java-ports.cgi. UNIX/Alpha ports are located at http://www.compaq.com/java/download/.

You may want to use Java from IBM instead. It is 1.5-2 times faster than Java from SUN. Download it from http://www.ibm.com/developerworks/java/jdk/. If you have several different Java installations and prefer to use some other version than would normally be chosen by ammc, create a link to the distribution that you would like to use with the name jdk1.x in your MMC or home directory. Choose the version number in the name (1.x in jdk1.x) higher than version numbers of your Java distributions.

To see if you have a Java distribution and/or it is compatible with MMC, run "ammc" from the MMC directory. The script looks for the highest version Java distribution on your computer in one of the standard directories and reports whether it is sufficient to run mmc. If your Java distribution is not found, add the directory in which it resides to the "JAVA_DIR" variable inside ammc script. "ammc -compile" will compile the Amanda program, and "ammc -run" will run it. Add "-frejus" flag to compile or run Frejus program.

If you have gcc v. 3.0 or higher, you can compile static executables for your platform that do not require Java do be run. Change GCC_DIR to point to the location of your gcc distribution and run "ammc -gcj [-frejus]".

Please note that the first time you run the program it will create a file (files) containing parametrization tables. This file will become corrupt if you break the execution of the program at this stage. The program will realize this at the next time you run it, it will re-calculate the tables and overwrite the corrupted file. You **should**, however, explicitly delete the file if you install the new version of the program, or use any of the hidden settings (unless you change the name of the file - see below). By default the parametrization-table file name starts with the dot, therefore, since no other files in the program directory start with the dot, it should be easy to find.

Do not forget to change length=720, radius=180, depth=1730 when running Amanda, if these numbers are different from the settings of your air-shower generator files or do not describe the location of the effective volume of the detector. Just add them as parameters to the ammc script, e.g. "ammc -run -length=800 -radius=400".

----

5. DESCRIPTION

This program is designed to fulfill the goal of propagating a muon through matter. Generally, as a muon travels through matter, it looses energy due to ionization losses, bremsstrahlung, photo-nuclear interaction and pair production. All of these losses have continuous and stochastic components, division between which is artificial and is chosen in the program by selecting an energy cut (ecut) or a relative energy loss cut (vcut). Ideally, all losses should be treated as stochastic. However, that would bring the number of separate energy loss events to infinity, since the probability of such events to occur diverges as 1/E_lost for the bremsstrahlung losses, as the lost energy approaches zero, and even faster than that for the other losses. The value of the cut should be chosen carefully, in such a way that you get the best accuracy with the reasonable calculation time. For example, if the problem is to propagate a muon until it looses all its energy (disappears), then vcut can be set to 0.1−0.2. If, however, one needs to propagate 1 TeV muon beam through only 3 meters of iron, then in order to see a reasonable approximation to the shape of the true final muon energy distribution, the value of vcut may have to be set as low as 0.01−1.e−3.

The program provides classes for the calculation of all components of the muon energy loss separately. To learn their names and parameters, see the javadoc generated manual pages. I suggest you start with the class hierarchy tree − it’s easiest to find the energy loss you’re looking for by name there. An example of how to use the classes is contained in the class Test. Each of the 4 main components has 3 classes: one is the entry class that has some definitions, and provides a function for evaluating of the integration limits. It also creates 2 sub−classes and keeps references to them: one defines continuous part of the loss, and the other stochastic. Since the sub−classes have all functions of the superclass, it could be tempting to call the functions of the superclass (or, rather, the object, standing above) directly. It, however, creates chaos, and the only case where this is used is with the integration limits. All other supplementary functions of the superclass are called with the class pointer dot name of the function. The superclass pointer is created by the superclass and passed along to the subclass at the initialization stage. This strategy keeps user, e.g. from having to set the "bb" variable in the photo−nuclear class three times − once for the superclass, and two times for the sub−classes.

All superclasses for the cross sections are united under another superclass CrossSections. In addition to the 4 energy losses mentioned above this class contains also subclass Decay which is there to make sure that the tracking integral is non−zero (it may also prove helpful when tracking taus). If initialized, class CrossSections creates all sub−classes and keeps references to them. It also defines some auxiliary functions for the displacement integral.

The main class of the program is Propagate. You can create this class by calling the Propagate(w, ecut, vcut) constructor. It creates the Particle class, CrossSections, and Medium. All parameters are passed along to the constructor of the Medium class. They are: name of the medium (e.g. "Water"), value of the ecut and vcut. If both ecut and vcut make sense (ecut>0 and 0<vcut<=1), then the lower one is chosen for each particular energy of the primary (muon). If only one of the cuts makes sense, then only that one is used. If both parameters are out of their ranges of reasonable values, then vcut=1 is used instead. Class Particle contains everything this program needs to know about the particle, like it’s location, direction, energy and mass. All variables except energy show the particle’s actual state. Energy and momentum are used for the all−around calculations, however.

The main routine propagateTo(r, e) of the class Propagate returns the energy of the particle after passing distance r if the particle has survived, or the traveled distance to the point of disappearance with a minus sign otherwise. As this routine returns, the Particle object, referenced by Propagate, describes the final state of the particle.

Since the program needs to evaluate up to 3−fold integrals, calculation time may become quite large (minutes) even for one muon. In its simplest form the program is designed to be used in the SYMPHONY framework (see class MPMCWJ for an example of how to do that). For more practical applications, however, parametrization (interpolation) routines are implemented. Enable them by calling interpolate("all") after the Propagate class is created. You may choose to enable just some of the parameterizations by replacing the word "all" with the list of things to parametrize. E.g. "bremsE trackX" will enable parametrization of the continuous part of bremsstrahlung loss and the displacement integral. This example will, however, take a long, long time to initialize, since trackX will call other (not parametrized) functions many, many times. In such a case you may be better off by omitting the trackX parametrization altogether. You can also save parametrization tables in a file and read them at in at a later time. Just call interpolate with a second parameter, e.g. interpolate("all", ""). The file will have a unique name based on the parameters you gave while initializing Propagate class. If you use any of the "less obvious" features listed below, or do some other changes in the code, the parametrization tables will become invalid. In such case you should delete the file with tables, or give it a different name by setting the second parameter of interpolate to something other than just an empty string.

Most of the features of the program can be switched on by using the command-line options of the front-end programs (Amanda, Frejus or Test). The name of the file to which the parametrization tables are saved changes when changing these options. By selecting appropriate options you can choose one of 5 photonuclear cross sections, turn on the LPM effect, enable Moliere scattering, continuous randomization, exact time of flight calculation and more. To obtain the list of options run the appropriate front-end program with the "--help" option.

Among the less obvious (hidden) features is the possibility to change e_low setting at which the muon is considered lost by the program. By default it is set to the muon rest mass. Stopped muon decay will also be considered if you use option "-sdec". It is possible to change the highest energy treated by the program (with parameterizations on). By default it is set to bigEnergy=1.e14 MeV. You can also choose to enable the high energy cutoff for the bremsstrahlung photons. It should to some extent approximate the behavior of the photons if the Lorenz invariant were not conserved. Enable it by setting Propagate.s.b.lorenz=true and Propagate.s.b.lorenzCut=1.e6, with the second expression setting the high energy cutoff in MeV. You need to change it to a more realistic value before using. If you choose to enable any of these options, you should of course do so before any parameterizations are done.

You will find programs Frejus, Amanda and Test in the package. The first one gives you an interactive way to set the energy and the distance from the keyboard after it initializes. This could be useful for using the program in UNIX pipes. The second program Amanda is an attempt to tailor this mmc package to the purposes of propagating the muons for the AMANDA mass production chain. It reads and outputs data in F2000 compliant format to and from the standard input/output. The last of the three front-end programs is Test. It can be used to perform a number of tests and is a useful tool for plotting cross sections and energy losses.

We hope that you enjoy using this program and welcome any comments (and bug reports). The primary design goals while writing this program were uncompromising computational precision and code clarity (for ease of future changes and corrections). Since the program was designed to be used in the SYMPHONY framework, computational speed was only a secondary issue. However, with full optimization (parameterizations) this code is at least as fast or even faster than the competition. We hope that the combination of precision, code clarity, speed and stability will make this program package a useful tool in the research connected with high energy particles propagating through matter.

Last modified on Sep 07 2001.