VMC^{++} , a quick guide

Iwan Kawrakow

February 13, 2002

Abstract

The following is a quick guide for installing and running VMC^{++} .

1 Installation

1. Unzip the VMC^{++} distribution file. This will create the directory `vmc++` with subdirectories `bin, data, spectra, runs, phantoms`. The executable `vmc.exe` is in `bin`.
2. Define environment variables `vmc_home, vmc_dir, xvmc_dir`. `vmc_home` must point to the VMC^{++} installation directory (*e.g.* `c:/vmc++`), `vmc_dir` to the sub-directory `runs` where all input files are to be placed. `xvmc_dir` is only relevant for dose output in the format of `vmc_show` but it must be defined, otherwise VMC^{++} will not execute. To define these variables in Windows NT, right-click on the "My computer" icon, select "Properties" from the menu and then the "Environment" tab. I think, under XP the environment variable definition is also under "My computer" but is hidden somewhere in "Advanced". If you want to have a separate directory for VMC^{++} input and output files, you can do so by defining `vmc_dir` to point to this directory instead of `vmc_home/runs`.

2 Running VMC^{++}

Open a DOS shell, go to the VMC^{++} bin directory and type

```
vmc.exe input_file
```

where `input_file` is the name of the VMC^{++} input file without extension (all VMC^{++} input files have the extension `.vmc`). Information about the progress of the simulation will be written to the standard output. After completion, VMC^{++} will generate various output files. The type and amount of output is specified in the input file, see section 4.8.2.

3 Data files

Data files that are necessary for the initialisation of a VMC^{++} calculation are in the `data` sub-directory. The two files that are relevant for the user are the `media.data` and the `ct_ramp.data` files.

The `media.data` file can be used to specify additional materials that are not already included in the distribution (Note: there is no need to specify elemental materials, data for all elements from 1-100 is found in `element.data`). The information necessary to specify a material is as follows: the first line contains medium id, medium name (case sensitive), mass density in g/cm^3, mean ionization energy in eV, number of elements and a dummy variable which should be set to zero. The medium id has to be > 100 and different than the id's of materials already in the file. The medium name must also be different than names previously used. Both, the medium id and the medium name can be used in a VMC^{++} input file when specifying a geometry. The next `N` lines, where `N` is the number of elements, contain atomic number, atomic weight and fraction by weight for each of the elements that make up the medium.

The file `ct_ramp.data` specifies the mass density ranges in which a certain material is to be used, if the geometry is determined via a CT file. The format of the file is easy, each line contains a medium name, the minimum and maximum density for which the material is to be used and the default material density. **The CT ramp defined in the `ct_ramp.data` file is from recent comparisons with the MMC group. Please adapt the data to your needs**.

Note that, unlike all other MC codes out there, no separate material data preparation is required for VMC^{++} . Once the materials involved in the simulation, the maximum source energy and the particle production thresholds are known, VMC^{++} will generate necessary interpolation tables on-the-fly. This initialization step takes of the order of one second per material on my Linux box, but unfortunately of the order of 3 seconds per material on my Windows NT machine using the same CPU (the simulation itself is 10-20% slower under Windows).

4 Input files

The operation of VMC^{++} is controlled via input files, which must have the extension `.vmc` and must be placed in the directory specified by the environment variable `vmc_dir`. In order to make the understanding of the syntax of an input file easier, a few sample input files have been included in the distribution (in the inputs sub-directory). A VMC^{++} input file must specify the simulation geometry, scoring options, output options, the source, run time control such as number of histories and batches, and possibly MC control parameter (*e.g.* cut-off energies and step sizes) and variance reduction parameter. A VMC^{++} input file is parsed for appropriate input using a set of syntax rules explained in the next sub-section. Following sub-sections deal with the "code words" that the various VMC^{++} objects understand.

4.1 Input file syntax

Sought input is identified with "code words". The input following a code word is then extracted and passed to the object looking for the input. Every VMC^{++} object looks for relevant input only within a section of the input file that is separated by a start delimeter and a stop delimeter. For instance, the delimeter of the beamlet source object is `beamlet source` and so, the beamlet source object searches for relevant information only within the part of the input file that is enclosed by

```
:start beamlet source:
:stop beamlet source:
```

The following general syntax rules apply for any VMC^{++} input

1. The order in which code words and sections (a part of the input file enclosed by a pair of `start` and `stop` delimeter) appear in the input file is irrelevant.
2. Code words must be followed by `'='` or `':'`, blanks between the actual code word and the = or : sign are permitted, *i.e.*
   ```
   my code word =
   my code word=
   my code word     :
   ```
   are all the same.
3. The parser is not case sensitive, *i.e.*
   ```
   my code word =
   My CoDe WOrd =
   MY CODE WORD =
   ```
   are all the same.
4. Input following `'#'` (Unix comment) or `'!'` (Fortran comment) is ignored.
5. Input can be integers, floating numbers or strings
6. If multiple input per code word is to be parsed, the input tokens must be separated by commas or blanks *i.e.*
   ```
   my code word = 1, 2, 3
   ```
   or
   ```
   my code word = 1  2  3
   ```
   or
   ```
   my code word = aluminum, water  bone
   ```
7. If input is to be continued on the next line, the line must end with a comma or backslash (Unix line continuation), *e.g.*
   ```
   my code word =  aluminum,
                   water,
                   bone
   ```

(**Note: because the above rules are hard coded, one has to use "/" instead of backslash as a directory separator under Windows**).

4.2 Source

This version of VMC^{++} comes only with a "beamlet point source" implementation, this should be sufficient for the initial phase of the investigations. The incident particles may be photons, electrons or positrons, they may have a mono energy or a spectrum. The beamlet source may have a single beamlet or multiple beamlets. **Note: if a source has multiple beamlets specified, a separate dose distribution for each beamlet will be calculated, see below**. The following input options are recognised:

1. `mono energy`: if this code word is present, the source is assumed to be monoenergetic and the subsequent input determines the energy. If not present, the source object will check for a spectrum (see below)

2. `spectrum`: if this input is present, a file name is expected. VMC^{++} will search for the file in the directory `$vmc_home/spectra` and so, the absolute path to the file should not be given. Two examples of spectrum files are included in the distribution.
3. `charge`: may be -1 (electrons), 0 (photons) or 1 (positrons)
4. `virtual point source position`: 3 floating numbers defining x-,y- and z-position of the point source.
5. `beamlet edges:` 3 times 3 floating point numbers defining the 3 edges of a rectangle in counter-clockwise order. If we denote them by $\vec{x}_1, \vec{x}_2$ and $\vec{x}_3$, the forth rectangle edge is calculated using

$$\vec{x}_4 = \vec{x}_1 - \vec{x}_2 + \vec{x}_3$$

Note: VMC^{++} checks that the tree vectors really form a rectangle (*i.e.* $(\vec{x}_2 - \vec{x}_1) \cdot (\vec{x}_3 - \vec{x}_2) \equiv 0$) and aborts execution if this is not the case. Multiple beamlets may be specified by simply repeating the `beamlet edges` code word the required number of times. During the simulation, points $\vec{x}$ are randomly sampled within the rectangle specified by $(\vec{x}_1, \vec{x}_2, \vec{x}_3)$ and the direction $\vec{u}$ of the particle is set to

$$\vec{u} = \frac{\vec{x} - \vec{x}_0}{|\vec{x} - \vec{x}_0|}$$

where $\vec{x}_0$ is the point specified in the `virtual point source position` code word. All positions must be in the same co-ordinate system that is used for the phantom definition. The co-ordinate system itself is arbitrary.

The calculated dose is normalized as dose per incident "fluence" Φ_0. The "fluence" is defined as number of particles divided by the beamlet area. There is a cosine missing in this definition (that's why the quotation marks). But at the end, the final normalization, precise definition of beamlet weights, etc., is up to you and therefore I decided to not go into the trouble implementing a "proper" dose normalization.

4.3 Transport parameter

This part of the input file, specified with the delimeter `MC Parameter`, can be used to input MC transport parameter such as particle production threshold energies, step-sizes, treatment of track-ends, etc. The complete description of the available settings and the way they affect execution speed and accuracy goes beyond the scope of this short description. It is therefore best to use as input

```
automatic parameter = yes
```

(see provided input files). With this input present, VMC^{++} will analyze phantom resolution, incident beam energy and size, etc., and determine some reasonable set of MC transport parameters. The execution speed will be slightly slower than possible but at the same time unlikely to produce inaccurate results (one can make ANY MC code fail by selecting an inappropriate set of transport parameters).

4.4 MC Control

Input that controls the MC simulation should be placed between the delimeter `MC Control`. Available options are

1. `ncase`: Number of particle tracks to be simulated
2. `nbatch`: Number of batches to be used for the statistical analysis
3. `rng seeds`: Initial random number seeds. This input can be safely omitted in which case the random number generator (the one by Zaman and Marsaglia is used) will initialise itself using Zaman and Marsaglia's default values.

4.5 Variance reduction parameters

At this point there is no automatic selection for these parameters available (best variance reduction strategy depends on the situation, *e.g.* in-phantom dose calculations *vs* treatment head simulations, etc). For your purposes they should be left as in the sample input files. The only thing to take into account is that for electron beams the input following `photon split factor` should be 2 but (normally) -40 for photon beams.

4.6 Quasi random numbers

Input that controls the use of quasi random numbers should be between the delimeter `quasi`. VMC^{++} can use simultaneously several quasi random generators. Again, the discussion of this feature goes beyond the scope of this quick guide and so, quasi random number input should be simply copied from the provided input files.

4.7 Geometry input

Input that specifies the geometry (or geometries) should be placed between the delimeter `geometry`. VMC^{++} can run simulations that involve several geometries at once. The geometry class in VMC^{++} is an abstract class[1] and an arbitrary number of geometry objects can be created by including the necessary input. The provided version of VMC^{++} was truncated to handle XYZ geometries only. To define a XYZ geometry one must include input that is enclosed by the delimeter `XYZ geometry`. One also must specify a name for the geometry (this is true for any other geometry) as this name is used to distinguish between scoring and output options in different geometries. The easiest way to define a XYZ geometry is to include the input

```
method of input = CT-PHANTOM
phantom file = ct_file
```

where `ct_file` is a density matrix. The format of this matrix, which you need to generate from your CT data sets, is best explained by including the portion of the code that reads-in the matrix:

```
FILE *fp = fopen(the_file_name,"rb");
if( !fp ) fatalError("Can not open density matrix file
%s",the_file_name);
int nx,ny,nz;
fread(&nx,1,sizeof(int),fp);
fread(&ny,1,sizeof(int),fp);
```

[1] The move to abstract geometry and scoring classes was done after the ICCR 2000 conference. The use of abstract classes for geometry and scoring comes with a penalty of about 20% in terms of execution speed. Given the greatly increased flexibility, this decrease in execution speed was found acceptable.

```
fread(&nz,1,sizeof(int),fp);
float *tmpx = new float [nx+1];
fread(tmpx,nx+1,sizeof(float),fp);
float *tmpy = new float [ny+1];
fread(tmpy,ny+1,sizeof(float),fp);
float *tmpz = new float [nz+1];
fread(tmpz,nz+1,sizeof(float),fp);
float *the_dens = new float [ nx*ny*nz ];
fread(the_dens, sizeof(float), nx * ny * nz, fp);
fclose(fp);
```

The other portion of information necessary to construct the density matrix is that the voxel `(ix,iy,iz)` has the address `ix + iy*nx + iz*nx*ny` in the density and dose arrays. Upon encountering this input, VMC^{++} will use the density to material conversion convention specified in the `ct_ramp.data` file to assign a material to each voxel.

If the above input is not present, VMC^{++} will search for the definition of a set of x-, y-, and z-planes, a set of materials and input that assigns a material number to the voxels.

Input for x-, y-, or z-planes is to be placed between the delimeter `x-planes` (or `y-planes` etc.). A set of planes can be defined in two different ways:

- Individual plane co-ordinates. The following is an example for this method:
  ```
  :start z-planes:
  method of input = individual
  co-ordinates = 0, 0.5, 1.1, 5, 12, 12.1
  :stop z-planes:
  ```
  The above defines 5 regions in z-direction that are 0.5, 0.6, 3.9, 7, 0.1 cm thick.
- Groups of regions. The input
  ```
  :start y-planes:
  method of input = groups
  front face = -3.5
  number of slabs = 5, 3, 5
  slab thickness = 0.2, 0.5, 0.2
  :stop y-planes:
  ```
  defines a set of 14 y-planes that divides space into 13 regions, the first 5 regions being 0.2 cm thick, the next 3 regions 0.5 cm and finally five 0.2 cm thick regions. The first plane is at y=-3.5 and the last at y=0.

Once the x-, y-, and z-planes of the XYZ geometry are defined, one must define the material composition. This is accomplished by including a sub-section, enclosed by the `media input` delimeter. With the code word

`media = medium1, medium2, ...  mediumN`

the materials that make up the phantom are made known to VMC^{++} . An arbitrary number of materials is permitted. Materials can be identified using their names (as defined in `media.data`), id's, or chemical symbols (if elements). **Material names and chemical symbols are case sensitive**. An id that is less or equal than 100 is interpreted as the atomic number of an element. When VMC^{++} reads in the materials, all regions are set to contain the first material in the medium list. One can then modify the material composition by using

```
method of input = planes
start x-plane = ib1, ib2, ib3, ...
stop x-plane = ie1, ie2, ie3, ...
start y-plane = jb1, jb2, jb3, ...
stop y-plane = je1, je2, je3, ...
start z-plane = kb1, kb2, kb3, ...
stop z-plane = ke1, ke2, ke3, ...
medium = m1, m2, m3, ...
relative density = r1, r2, r3, ...
```

All regions with x-index greater or equal than `ib1` and less than `ie1`, y-index greater or equal than `jb1` and less than `je1`, z-index greater or equal than `kb1` and less than `ke1` will be set to medium `m1` and the default mass density of medium `m1` scaled by `r1`, etc. If the `relative density` code word is missing, the default material mass density will be used[2]

4.8 The scoring manager

Because there may be several geometries involved in the MC simulation, there is a "scoring manager", which is responsible for constructing scoring and output objects, objects that handle particles exiting a geometry, etc., as needed and defined in the input file. Input that controls the operation of the scoring manager is enclosed by the delimeter `scoring options`.

The first thing the scoring manager needs to know is in which geometry the particles start their transport as they come from the source[3]. The start geometry is made known to the scoring manager by the code word

```
start in geometry :  my_geometry
```

where `my_geometry` is the name of a defined geometry. This input MUST be present even if there is only a single geometry defined in the input file.

Scoring and output options are defined in separate sub-sections within the delimeters `scoring options`.

4.8.1 Dose scoring

Recording the energy deposited by various particles during the simulation is only done in geometries for which this is requested by the user. To request dose scoring during the simulation, one needs to include a sub-section enclosed by the delimeter `dose options`. One then defines the geometries in which dose scoring is to be done using

```
score in geometries:  geometry1, geometry2, ...
```

where `geometry1`, etc., are the names of defined geometries. By default VMC^{++} scores energy deposition. However, it is possible to request "dose to water" calculation. In this case, energy is multiplied by the ratio of the restricted mass stopping power in water to the restricted mass stopping power in the medium at the energy of the electron depositing the energy. This option is turned on by including

[2]Note, however, that the default material density is used for the calculation of the density effect correction. The mass density modification in the input file is merely a multiplicative factor for stopping powers and cross sections. If you want a density effect correction for a non-default mass density to be used, you have to define a separate medium in `media.data` that has the required density.

[3]Making this known to the scoring manager avoids checking all defined geometries and also removes ambiguities if a particle from the source may enter more than 1 geometry.

`score dose to water: yes`
into the dose options section. In this VMC^{++} version, the dose deposited by each beamlet is scored in a separate array. Keep in mind that 16 bytes per voxel are needed for scoring dose and doing statistical analysis when requesting a simulation with a large number of beamlets.

4.8.2 Output options

One can request dose output for each geometry where dose scoring was done during the simulation. To do so, one needs to include a sub-section within the scoring options section of the input file that is enclosed by
`:start output options my_geometry:`
`:stop output options my_geometry:`
where `my_geometry` is the name of a geometry where dose scoring was requested. Each scoring geometry requires its own output options sub-section. In addition, because dose output may be requested in several geometries, the output file names are constructed from the name of the input file and the name of the geometry. For instance, if `inpfile.vmc` is the name of the input file and if dose scoring and output was requested in geometries with names `geom1` and `geom2`, VMC^{++} will produce various output files with different extensions named
`inpfile_geom1`
for results in geometry `geom1` and
`inpfile_geom2`
for results in geometry `geom2`. The number and extensions of such files depends on the requested output as explained below.

1. 3D dose dump is requested by including
   `dump dose = integer`
   into the output options. The integer number can be 0,1 or 2. If zero (or if the code word is missing), no 3D dose output will be done. If 1 or 2, the dose distribution will be dumped into a binary file with the extension `.dos` using 2 different formats:

   - `dump dose = 1`: The dose distribution and its statistical uncertainties will be written to the file using single precision floating point numbers (4 bytes per number).
   - `dump dose = 2`: The dose distribution only will be written to the file using unsigned short integers (2 bytes per number).

   The format is best explained by the code used to write the data. The following is the same for the 2 formats:

   ```
   std::ofstream dumpd(the_dose_file);
   dumpd.write((char *) &nreg,sizeof(int));
   dumpd.write((char *) &ncase,sizeof(long));
   dumpd.write((char *) &nbatch,sizeof(long));
   dumpd.write((char *) &n_beamlet,sizeof(int));
   dumpd.write((char *) &dump_dose,sizeof(int));
   ```

   where `nreg` is the number of regions (`nx*ny*nz`), `ncase` the number of particle

sets used in the simulation, `nbatch` the number of statistical batches[4], `n_beamlet` the number of beamlets in the simulation (and therefore the number of subsequent dose distributions) and where `dump_dose` is the format specifier (1 or 2). The code writing the actual data looks as follows:

```
for(int i=0; i<n_beamlet; i++) {
    analyze_dose(i);
    if( dump_dose == 1 ) {
        dumpd.write((char *) dose_array,nreg*sizeof(float));
        dumpd.write((char *) error_array,nreg*sizeof(float));
    } else {
        dumpd.write((char *) &dmax,sizeof(double));
        unsigned short *tmp = new unsigned short [ nreg ];
        for(int j=0; j<nreg; j++) {
            double aux = dose_arry[j]/dmax*65534;
            tmp[j] = (unsigned short) aux;
        }
        dumpd.write((char *) tmp,nreg*sizeof(unsigned short));
        delete [] tmp
    }
}
```

where `dose_array` and `error_array` are pointers to float arrays of size `nreg`, which, after the call to `analyze_dose(i)`, contain the dose deposited by the i'th beamlet. `dmax` is the maximum dose in each of the `n_beamlet` distributions and can be used to convert the unsigned short integers to floating numbers in the case `dump_dose=2`. In all cases, the dose per incident "fluence" is output in units of 10^{-10} Gy cm^2.

2. A dose profile (or scan along a given line) is requested by including
   `scan xo =` $\vec{x}_1, \vec{x}_2, ...$
   `scan uo =` $\vec{u}_1, \vec{u}_2, ...$
   where $\vec{x}_i$ and $\vec{u}_i$ are 3D vectors (*i.e.* 3 numbers) defining a line with the parametrisation $\vec{x} = \vec{x}_i + t\vec{u}_i$. If such a card is present in the input file `inpfile` for the geometry `geom`, VMC^{++} will produce files
   `inpfile_geom.prof0, inpfile_geom.prof1, ...`
   one file for each scan requested. The output in these files contains 3 columns which are position, dose, dose uncertainty. The "position" is defined as follows: if $\vec{a}_1, \vec{a}_2, ..., \vec{a}_n$ are the intersections of the line $\vec{x} + t\vec{u}$ with the boundaries of the geometry, the i'th "position" is the distance between the point $(\vec{a}_i + \vec{a}_{i+1})/2$ and the point $\vec{x}$. This somewhat cumbersome procedure for requesting dose profiles is motivated by (i) the desire for generality and the capability to output dose profiles in arbitrary geometries (not just XYZ) and by (ii) the desire to be able to look at dose scans along arbitrary lines (not just along the x-, y-, or z-direction). If there are several beamlets in the simulation, the output will contain the dose for each beamlet individually and the total dose assuming equal weights for the beamlets, one after the other.

[4] This information is included in the output in order to be able to do statistical analysis in a restarted simulation.