Magnetic Field Binary File Format
Version 3
April 24, 2018

David Heddle
Christopher Newport University

This describes the binary format used by *ced* and also the general *magfield* package.

The binary file format contains a header of twenty 32-bit words. (The 80 bytes for this header are in the noise when it comes file size.) The header format is:

(int) 0xced (decimal: 3309) magic number—to check for byte swapping
(int) Grid Coordinate System (0 = cylindrical, 1 = Cartesian)
(int) Field Coordinate System (0 = cylindrical, 1 = Cartesian)
(int) Length units (0 = cm, 1 = m)
(int) Angular units (0 = decimal degrees, 1 = radians)
(int) Field units (0 = kG, 1 = G, 2 = T)
(float) q_1 min (min value of slowest varying coordinate)
(float) q_1 max (max value of slowest varying coordinate)
(int) N_{q1} number of points (equally spaced) in q_1 direction including ends
(float) q_2 min (min value of medium varying coordinate)
(float) q_2 max (max value of medium varying coordinate)
(int) N_{q2} number of points (equally spaced) in q_2 direction including ends
(float) q_3 min (min value of fastest varying coordinate)
(float) q_3 max (max value of fastest varying coordinate)
(int) N_{q3} number of points (equally spaced) in q_3 direction including ends
~~Reserved 1~~ High word of creation date (unix time)
~~Reserved 2~~ Low word of creation date (unix time)
Reserved 3
Reserved 4
Reserved 5

The magic word, which should have the hex value ced (i.e. 0xced), is important. The CLAS magnetic field maps are produced by JAVA code which (sensibly) enforces the use of network ordering (big endian) independent of architecture. However the machines we use in CLAS tend to be little endian. If the code reading the maps is also in JAVA, it doesn't matter. If the code reading the maps is in *C* or *C++*, byte swapping will likely be required.

As you see, there used to be five reserved 32-bit slots in the header. Two of them have been used to store the creation date of the field map file, which is a 64-bit (long) quantity. To get the creation data the long has to be reassembled from its two pieces and then, using some sort of language supplied time function, converted into a meaningful string. The details are left as an exercise.

The only ambiguity is the meaning of the triplet $\{q_1, q_2, q_3\}$ For cylindrical coordinates, the triplet means $\{\phi, r, z\}$. It seems most natural that for Cartesian coordinates the triplet maps to: $\{x, y, z\}$. Thus, for a Cartesian field map, *x* would be the outer, slowest-varying grid component.

The total number of field points will be: $N = N_1 \times N_2 \times N_3$ (we will store floats, not doubles)). Each point requires three four-byte quantities. The total size of the binary file will be $80 + 3 \times 4 \times N$.

Noting that the number of points always includes the endpoints, the step size in direction *i* is $(q_{imax} - q_{imin})/(N_i - 1)$

In version 3, two of the reserved words have been allocated to store the creation date in unix time. The remaining reserved fields are available to be used in some manner to be defined later.

The field follows the header, in repeating triplets:

B1
B2
B3

The first three entries correspond to the field components for the first grid point, the next three for the second grid point, etc. The ordering, for consistency, should be:

$\{B_x, B_y, B_z\}$ if the field is Cartesian
$\{B_\phi, B_r, B_z\}$ if the field is Cylindrical

Example

For the binary version of the original torus map (before we encoded creation date) we have for the header:

0xced
0 (grid is cylindrical)
1 (field is Cartesian)
0 (units: cm)
0 (units: decimal degrees)
0 (units: kG)
0.0 (ϕmin)
30.0 (ϕmax, degrees)
121 (N_ϕ)
0.0 (r_{min})
500.0 (r_{max}, cm)
251 (N_r)
100.0 (z_{min}, cm)
600.0 (z_{max}, cm)
251 (N_z)
0 (Reserved 1)
0 (Reserved 2)
0 (Reserved 3)
0 (Reserved 4)
0 (Reserved 5)

Thus the three step sizes are:

$$\Delta\phi = (30-0)/(121-1) = 0.25°$$
$$\Delta r = (500-0)/(251-1) = 2 \text{ cm}$$
$$\Delta z = (600-100)/(251-1) = 2 \text{ cm}$$

Recalling the header is 80 bytes, the total size of the binary is (had better be):

$$80 + 3\times4\times121\times251\times251 = 91{,}477{,}532 \text{ bytes.}$$