The axisymmetric magnetic field calculation program package magfield2

F. Glück

Using this program package one can compute magnetic field and vector potential of an axially symmetric coil system, with coils having rectangular shape cross-section in the $z-r$ cylindrical symmetry (meridian) plane. The coils are obtained by rotation of the rectangles around the z-axis, the current in the coils has azimuthal direction, perpendicularly to the $z-r$ meridian plane.

The user has 2 completely different methods for the magnetic field calculation. One method uses elliptic integrals; then one can compute the field everywhere, but the computation time is large. For fast trajectory calculation it is usually better to use the method with Legendre polynomial expansion: in this case the calculation is much faster; nevertheless this method is not appropriate everywhere (see below the details).

Coil parameters (geometry and current)

Before starting the calculation, the user has to define the geometry and current parameters of the coils. For this purpose, the user has to make a coil parameter data (ASCII) file, with the following structure: the first line of the data file contains the integer number of the coils (in the program the variable name Ncoil is used for this number). Then there are Ncoil number of lines, each line contains the parameters of a coil (these are real numbers):

zmid rin thick length current

zmid: middle z value of the coil (m),

rin: inner radius of the coil (m),

thick: radial thickness of the coil (m),

length: length of the coil in the z direction (m),

current: total current of the coil (Amper); total current= number of windings $\times$ current in the coil wire.

The zmin, zmax, rmin, rmax edge parameters of the coil rectangle are: zmin = zmid − length/2., zmax = zmid + length/2., rmin = rin, rmax = rin + thick.

When some of the field computation functions (magfield2 or magfield2_elliptic) is for the first time called, the function coilread is called from inside these functions. Function coilread reads the coil parameters from the abovementioned data file (the name of the data file is defined by the user; see below at the sections magfield2 and magfield2_elliptic). Then this function writes the number of the coils into the global integer Ncoil, and the coil parameters into the global double array coil[Ncoilmax+1][5]. The integer number Ncoilmax should be defined by the user with a #define command, in the beginning of the program which uses the package magfield2 (for example: #define Ncoilmax 100). The number Ncoilmax should be not smaller than the number of the coils Ncoil: otherwise one gets an error message, and the program execution stops.

magfield2_elliptic: field calculation by elliptic integrals

In order to compute the magnetic field by elliptic integrals, the user has to call the function magfield2_elliptic in the following way:

```
magfield2_elliptic(z,r,inputcoil,m,&A,&Bz,&Br);
```

The input and output parameters here are the following:

z,r: cylindrical coordinates of the field point in meter (double real numbers);

inputcoil: this is a char* string constant, it contains the name of the coil parameter data file. One can define this name in the main calling program as: strcpy(inputcoil,"inputcoil.dat");. Of course, one can use arbitrary file name.

m: this integer is necessary for numerical integration. With m=20 one can get usually very good numerical accuracy, if the radial dimension (thickness) of the coils is not too large.

A,Bz,Br: these double real numbers are the results of the calculation. A denotes the vector potential in Teslameter (with axially symmetric coils and azimuthal current the vector potential has only azimuthal component); Bz and Br are the cylindrical coordinates of the magnetic field in Tesla.

The calculation method by the elliptic integrals has the advantage that it works everywhere, for arbitrary field point (even inside the windings of

the coils). Unfortunately, it has also a disadvantage: it is rather slow. Using the first and second complete elliptic integrals one needs for the field calculation a two-dimensional numerical integration for each coil (along the axial and the radial directions). In our method we use also the third complete elliptic integral, which has the advantage that the numerical integration is only in the radial direction necessary; this makes the computation much faster. Nevertheless, even with this improvement the computation is about 1000 times slower than with the Legendre polynomial expansion.

magsource, magfield2: field calculation by Legendre polynomial expansion

The magnetic field calculation by Legendre polynomial expansion is much faster than with the elliptic integrals. In order to understand this method, let us introduce some definitions. Let us fix a point $(z_0, 0)$ on the z axis. It is possible to compute in this axis point the higher derivatives of the magnetic field with high accuracy. From these higher derivatives one can build some numbers B_n $(n = 0, 1, \ldots, nmaxmag)$, where B_0 is the magnetic field in the $(z_0, 0)$ point, B_1 is proportional to the first z-derivative of the field here, B_2 is proportional to the second z-derivative, etc. So all these B_n numbers are defined in the axis point $(z_0, 0)$. It is a special property of the axially symmetric coil system that the magnetic field in an arbitrary off-axis point, which is not far from the axis point $(z_0, 0)$, can be computed by using these numbers B_n. We call the axis point $(z_0, 0)$ source point, and the numbers B_n we call source coefficients; namely, the B_n numbers depend on the parameters (geometry and current) of the coils, which represent the source of the magnetic field.

Let us define the convergence radius ρ_c as the minimal distance of the source point from the coils. The z component of the magnetic field in the field point (z, r) can be computed by the following series:

$$B_z = \sum_{n=0}^{\infty} B_n * \left(\frac{\rho}{\rho_c}\right)^n P_n(u),$$

where ρ is the distance of the field point (z, r) from the source point $(z_0, 0)$, $u = (z - z_0)/\rho$, $P_n(u)$ is Legendre polynomial. The vector potential A and the radial field component B_r can be computed by similar expansions. This series has an important property: it is convergent only for $\rho < \rho_c$, i.e. only

for those field points, which are inside of the convergence circle, defined by the source point as centre and ρ_c as radius.

Let us assume that the user wants to make magnetic field calculation for z values between z0min and z0max, using the above Legendre polynomial expansion method. Then the source points should be defined within this interval. The program uses the simplest configuration: it defines source points in equidistant distribution between z0min and z0max, with delz0 distance between two neighbouring points. The user should define the number delz0, so that it is 2-3 times smaller than the minimal inner coil radius. Then the user defines the name of the coil parameter data file: strcpy(inputcoil,"inputcoil.dat"); (where inputcoil is a *char string constant). In addition, the user has to define 2 integer numbers, nmaxmag and Nspmagmax, by #define command. Here nmaxmag denotes the maximal index value n for the source coefficients B_n (nmaxmag=300 is usually a good choice); Nspmagmax should be defined so that it is not smaller than the number of source points Nsp, where Nsp=(z0max-z0min)/delz0+1).

Then the user calls the function magsource:

magsource(z0min,z0max,delz0,inputcoil);

This function reads first the coil parameter data file by calling function coilread, then it calculates the source points, it computes the source coefficients B_n (n=0,...,nmaxmag) in all source points, and finally it writes the source points and source coefficients into the data file magsource.dat. At this point the user can stop the program running, and continue the field calculation at a later time (since all information necessary for the Legendre polynomial method is saved in the data file magsource.dat).

In order to compute the magnetic field by using the Legendre polynomial expansion, the user has to call the function magfield2 in the following way:

```
magfield2(z,r,inputcoil,n,&A,&Bz,&Br);
```

The input and output parameters here are the same as in the case of magfield2_elliptic. In order to make the Legendre polynomial expansion calculation, the program searches the best source point; this means: it calculates for all source points the ρ distance between field point (z, r) and source point, and it looks for the minimal ρ/ρ_c ratio. The program can use the Legendre series expansion, if this ratio is smaller than 1. If it is larger than 1, the series is not convergent. In this case the Legendre series method

is not possible to employ, and the program computes the magneti field by the elliptic integrals, calling the function magfield2_elliptic. This is the reason why the user has to give the coil parameter file name inputcoil and the integration number n to the function magfield2: if the Legendre series method is available for the computation, the program does not use inputcoil and n, since in this case only the source points and source coefficients from the data file magsource.dat are employed.

An example for the main calling program is the following:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define Ncoilmax 100
#define nmaxmag 300
#define Nspmagmax 3000

void test(char *inputcoil);



#include "magfield2.c"

main()
{
  double z0min,z0max,delz0;
  char inputcoil[100];

  z0min=-0.5;
  z0max=2.5;
  delz0=0.02;
  strcpy(inputcoil,"inputcoil.dat");
  magsource(z0min,z0max,delz0,inputcoil);
//
```

```
  test(inputcoil);
 return 0;
}

////////////////////////////////////////////////

void test(char *inputcoil)
{
  double z,r,A,Bz,Br,Aell,Bzell,Brell,errA,errBz,errBr;
  int i,m;
  m=20;

  z=0.;
  r=0.02;
    magfield2(z,r,inputcoil,m,&A,&Bz,&Br);
    printf("A,Bz,Br= %19.11e %19.11e %19.11e \n \n",A,Bz,Br);
    magfield2_elliptic(z,r,inputcoil,m,&A,&Bz,&Br);
    printf("A,Bz,Br= %19.11e %19.11e %19.11e  \n",A,Bz,Br);
  r=0.;
  for(i=0;i<=20;i++)
  {
    z=i*0.1;
    magfield2(z,r,inputcoil,m,&A,&Bz,&Br);
    printf("z,r,A,Bz,Br= %12.5f %12.5f %17.9e %17.9e %17.9e
       \t \n",z,r,A,Bz,Br);
    magfield2_elliptic(z,r,inputcoil,m,&A,&Bz,&Br);
    printf("A,Bz,Br(elliptic)= %17.9e %17.9e %17.9e \t
    \n",A,Bz,Br);
  }
//
  return;
}
```