Measured Magnetic Fields Example

Introduction

The user program example in this directory demonstrates one way to incorporate a data array of measured magnetic fields in a simulation. It is assumed that the data array(s) contain(s) the required B field components (Bx and By for 2D, or Bx, By, and Bz for 3D). You have the option of storing the data in one, two (e.g. 2D), or three (3D) arrays. The choice is yours along with the data addressing issues.

This example uses a single data array with the following 1D format:

bx, by, bx, by, bx, by, …

You could think of it as an array of structures with each structure composed of bx and by.

Creating the data array

Normally the data array would be created with measured values. However, in this example a refined magnetic array serves as the source of information for the data array.

Strategy

The first problem is how to obtain B field information in gauss from a potential array in such a way that a usable data array is automatically created. There are a couple of tricks that can used to do this. First, we will fly neutral particles (no charge). Their trajectories will not be affected by magnetic fields (or by electrostatic fields). Thus we can fly a single neutral along each desired data acquisition line. The first problem is how to take data at the desired intervals along the trajectory. The trick is to use time markers that in conjunction with the neutral's kinetic energy give the exact sample spacing required. Data recording can be used to record Bx and By at every time mark to a delimited data file (be sure to set the number format you desire). You should also record data at the neutral's birth (depress the **Ion's Start** button also). Define the first neutral to start at x = y = 0 in grid units, and fly to the right. In the example a data point is recorded each five gu. Thus the dy for each successive neutral is 5 gu. A single group of neutrals can scan the array and record the needed data.

Important tricks

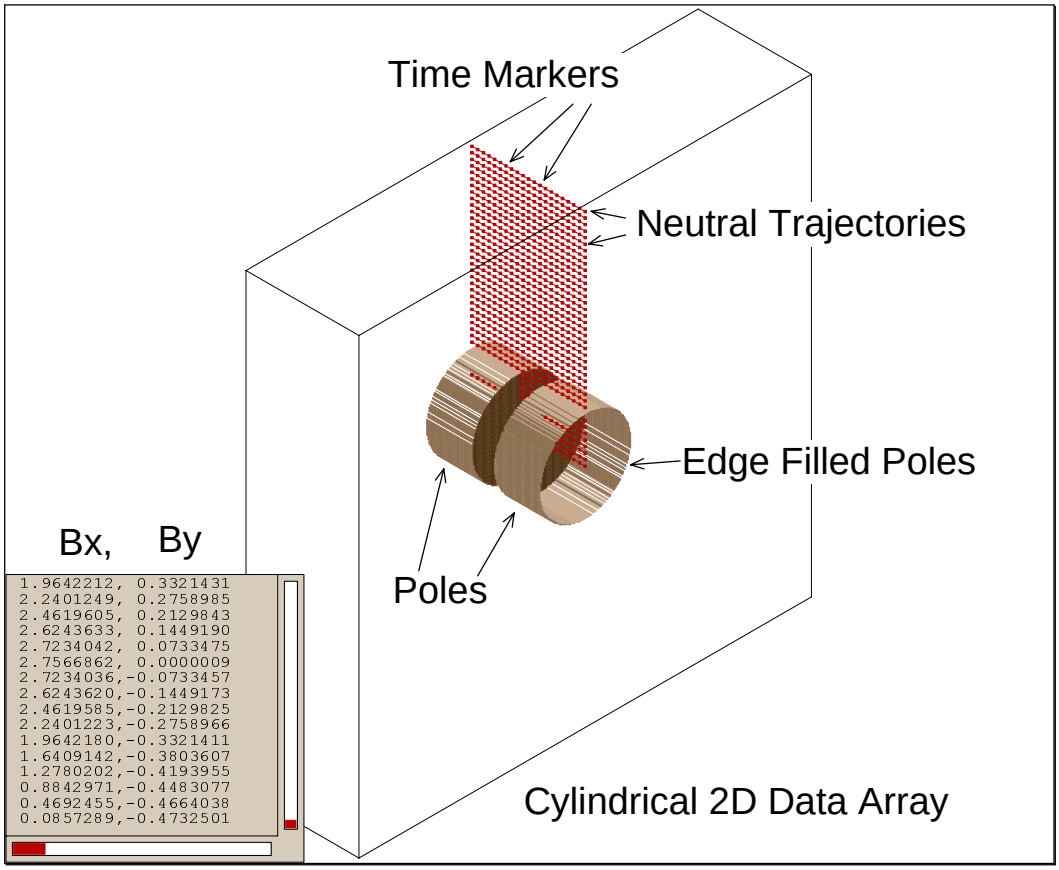
This is all fine and good as long as the neutrals don't splat accidentally on an electrode. This can be avoided by edge filling your electrode or pole definitions. Now the neutrals will fly right through the poles.

The other issue is that you may have to tweak up the starting position, time steps, and dy amounts slightly to insure that the important near-boundary points do not actually fall on the pole's boundary, but rather slightly in the surrounding field to insure you will get a non-zero field.

The data creation example

Setup: Load and refine the **get_data.pa#** file (set convergence objective to 10^{-7}). Remove all pas from ram. Load the **get_data.iob** file into view and click the **Fly'm** button. Notice that the ions

(neutrals) are flown and the B field data is shown (figure below). In order to save the data the data recording file name must be changed from **nul** (try **test.dat** and refly). Look at the file with edy (or some other editor). The first file line (e.g. begin fly'm) must be removed to avoid array reading problems. Also remember that data is appended to these files. If you want a clean copy, get into the file manager and erase the file before the fly'm.



Using the data array in a Mfield_adjust program segment

Now the challenge is to use the data file (saved as **bfield.dat**) to simulate the magnetic fields in a potential array. The following material discusses how an mfield_adjust program segment can be used.

Running the simulation example

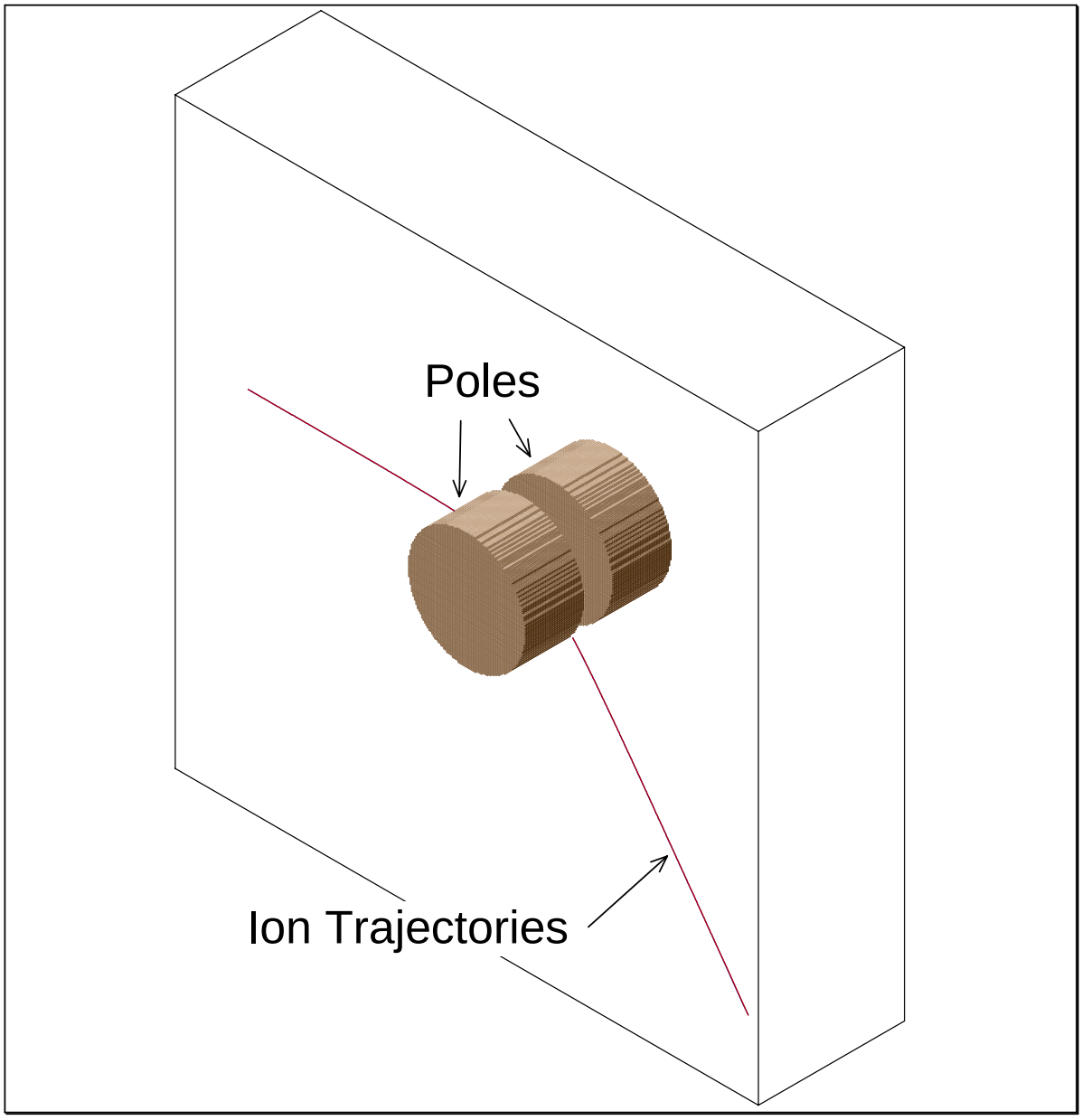
Setup: Load and refine the potential array file bfield.pa#. Remove all pas from memory. Load the file fly_ions.iob into view, turn off time steps and data recording (if they are on), and click fly'm. The ion trajectories are shown in the figure below.

Creating and loading the data array

The following command defines and automatically loads the data array into an adjustable variable array called bfield:

Adefa bfield 1681 ; “bfield.dat” create a 1681 element array and load bfield.dat into it.

The data array contains 40 lines of points with 21 point pairs in each line (840 pairs total or 1680 points total). The extra point of dimension was for good luck (no real use).



Comparing with normal pa trajectories

It is desirable that ion trajectories can be compared between normal pa fields and those created by the array data. In order to do this the first line of the user program segment flies ion number oneusing the pa’s fields by exiting the segment.

```
Rcl ion_number 1 x=y exit          ;exit from data array simulation if ion number one
```

Calculating data array index from ion location

The data array has one data point for each five potential array points in x and each five potential array points in y. Thus the index points are:

```
Rcl ion_px_abs_gu 5 / sto x      ; x index including fractional part
int sto xint                     ; x integer index
rcl x frac sto xfrac             ; x fractional index
1 x<>y – sto xlfrac              ; (1- x fractional index)
Rcl ion_py_abs_gu 5 / sto y      ; y index including fractional part
int sto yint                     ; y integer index
rcl y frac sto yfrac             ; y fractional index
1 x<>y – sto ylfrac              ; (1 – y fractionsl index)
```

Calculating lower left corner index

```
Rcl yint 21 * rcl xint +         ;linear index of point pair
2 * 1 + sto llnBx                ;index of lower left corner Bx value in pair
1 + sto llnBy                    ;index of lower left corner By value in pair
```

Calculating B field in x by linear interpolation

The following is used to calculate the B field in the array's x direction

```
rcl llnbx       arcl bfield rcl xlfrac * rcl ylfrac *    ;ll corner contribution
rcl llnbx 2 +   arcl bfield rcl xfrac * rcl ylfrac * +   ;lr corner contribution
rcl llnbx 42 +  arcl bfield rcl xlfrac * rcl yfrac * +   ;ul corner contribution
rcl llnbx 44 +  arcl bfield rcl xfrac * rcl yfrac * +    ;ur corner contribution
sto ion_bfieldx_gu                                       ;pass Bx to SIMION
```

Calculating B field in r (y and z) by linear interpolation

The following is used to calculate the B field in the r direction (y and z components)

```
rcl llnby       arcl bfield rcl xlfrac * rcl ylfrac *    ;ll corner contribution
rcl llnby 2 +   arcl bfield rcl xfrac * rcl ylfrac * +   ;lr corner contribution
rcl llnby 42 +  arcl bfield rcl xlfrac * rcl yfrac * +   ;ul corner contribution
rcl llnby 44 +  arcl bfield rcl xfrac * rcl yfrac * +    ;ur corner contribution
sto bfieldr                                              ;save r component
```

Resolving the x and z components

The following makes use of the ion's non-absolute y and z components to resolve the By and Bz components:

```
Rcl ion_pz_gu                    ; load pz as if it were y
rcl ion_py_gu                    ; load py as if it were x
>P                               ; convert to polar coordinates y = angle x = r
rlup                             ; roll up pointer to y register
rcl bfieldr                      ; replace r with bfieldr
>R                               ; convert back to rectangular coordinates
sto ion_bfieldy_gu               ; pass By component to SIMION
rlup                             ; roll up pointer to y register
sto ion_bfieldz_gu               ; pass Bz component to SIMION
```

Extending the example

The above example could be extended to 3D data arrays by packing the array as 2D pages with the first page set for z = 0 (or 1 in user array conventions). This would require the computing of zint, zfrac, and zlfrac much as the equivalent 2D factors were calculated. Moreover, the linear interpolation would require 8 lines using the six fractional weighting factors appropriately.

The only limit here is your imagination and the need to do something like this in the first place.