Utilisation du programme simulant le solenoide.c .

1/ **Compilation du programme**

Le programme a été écrit en C et il suffit d'utiliser le compilateur C fournit sur la plupart des distributions unix : gcc.

Exemple : gcc -ggdb -pedantic-errors -lm solenoid.c -o solenoid

2/ **Utilisation du programme**

Après avoir lancer le programme, celui-ci vous demande les caractéristiques du solénoïde que vous souhaitez simuler :

-rayon =(en metre)
-courant =(en Ampere)
-angle que font les spires avec le plan médian=(en radian)
-nombre de spire
-ecart entre chaque spire

et le pas que vous souhaitez utiliser pour les intégrations numériques :

-pas d'integration pour les integrales

Ensuite, le programme propose un menu permettant de choisir le type de résultats que l'on veut obtenir :

-Calculer les champs pour une spire en un point
choix 1
-Calculer les champs pour le solenoide en un point
choix 2
-ecrire les graphes des champs pour une spire (sur le plan median)
choix 3
-ecrire les graphes des champs pour le solenoide (sur le plan median)
choix 4
-ecrire les graphes des champs pour le solenoide (sur l'axe du solenoide)
choix 41
-ecrire les graphes des champs pour le solenoide (sur le plan median; seulement en dehors du solenoide
choix 42
-ecrire l image des champs
choix 5
-calculer la fem en sur une spire placee a l exterieur du solenoide
choix 6
-ecrire les graphes de la fem induite total et de fuite
choix 61

Il suffit alors d'entrer le chiffre correspondant au choix voulu.

Il y a trois possibilités concernant les résultats obtenus :

-1 ils sont directement écrits dans le terminal.

-2 ils sont écrits dans un fichier pour être lu par un logiciel graphique de type gnuplot ou xmgr.
-3 ils sont écrits dans un fichier dans l'optique de fournir des images des champs.

1--Dans le premier cas il n'y a rien a faire.
2--Dans le second il faut lancer un logiciel tel que gnuplot pour lire les fichiers. Les fichiers sont indiqués de manière claires tel que l'on sache à quoi ils font référence.

> *Exemple* :
> Choisissons le quatrième choix : « ecrire les graphes des champs pour le solenoide (sur le plan median) »
> Alors le programme écrit 6 fichiers correspondant aux trois composantes du champ magnétique et aux trois composantes du potentiel vecteur:
> Arho_solenoid Brho_solenoid
> Ateta_solenoid Bteta_solenoid
> Az_solenoid Bz_solenoid

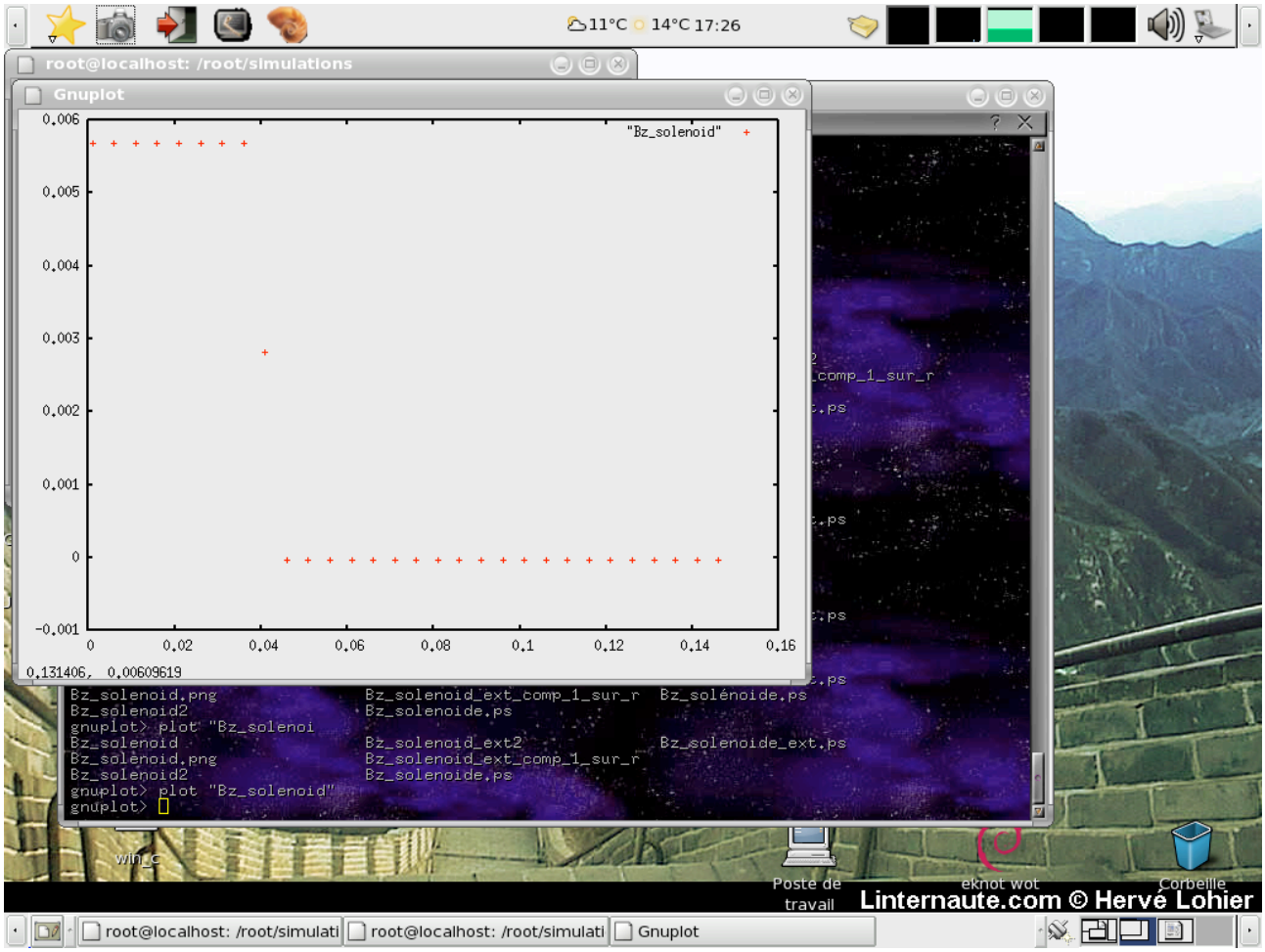
Le choix du repère utilisé (cylindrique ou sphérique) est implicite, dans l'exemple ci-dessus nous avons utiliser un repère cylindrique. Au contraire, lorsque l'on utilise un repère sphérique nous notons les composantes : r,teta,phi.

> *Exemple* :
> Prenons le troisième choix : « ecrire les graphes des champs pour une spire (sur le plan median) »
> Alors les 6 fichiers écrits sont :
> Ar_spire Br_spire
> Ateta_spire Bteta_spire
> Aphi_spire Bphi_spire

Exemple d'utilisation d'un fichier avec gnuplot :

```
[root@mlilom_corp explications]# gnuplot
gnuplot> plot "Bz_solenoid"
```

Ce qui donne un résultat ressemblant au suivant :



3—Dans ce troisième cas il faut utiliser un logiciel open source distribué couramment sous linux : octave. Son rpm est disponible sur rpmfind.net.
De plus il faut ajouter quelques algorithmes à la librairie octave (pour simplifier, vous pouvez simplement les créer dans le répertoire dans lequel vous allez les utiliser, aussi mise à part « file2matrix » ils sont tous distribués sur le site du développeur du logiciel octave):

-file2matrix.m

```
#-------------------------------------------------------------------
#function file2matrix(f)
#
#f : file which contain the matrix
#
#-------------------------------------------------------------------

function y=file2matrix(f)

if (nargin!=1)
usage("file2matrix : file2matrix(f), must have 1 arguments");

elseif(! (isstr(f)))
error("file2matrix : f must be a string");
endif

fp=fopen(f,"r");
n=0;
```

```
A=[];
DIM=fscanf(fp,"%d",2);%getting matrix's dimensions
while (n<DIM(2))
n=n+1;
Z=fscanf(fp,"%g",DIM(1));
A=[A,Z];
endwhile

fclose(fp);

y=A;

endfunction
```

-fisim.m

```
function [output_image, bzero, bscale] = fitsim(filename,n_hdu)
%FITSREAD reads a FITS image into a Matlab variable.
%
%  [data, bzero, bscale]=fitsim(filename)
%  [data, bzero, bscale]=fitsim(filename,h_hdu)
%
% The variable 'n_hdu' explicity tells FITSIM how many
% sets of 36 header 'cards' precede the data. One normally
% lets the function figure this out on its own.
%
% Final data values are to be computed by data*bscale+bzero
%
% Known problems
%        Will only deal with 2D fits files.
%
% Version 2.0
% R. Abraham, Institute of Astronomy, Cambridge University
%
% Modified R.G.Lane 18 November 1997 to use strcmp
%                          to return bzero, bscale
% EEE University of Canterbury

%The first few cards of the first set of 36 cards must give information
%in a pre-defined order (eg. the data format, number of axes,
%size etc) but after that the header keywords can come in any
%order. The end of the last card giving information is flagged by
%the keyword END, after which blank lines are added to pad the
%last set of cards so it also contains 36 cards. After the last card
%the data begins, in a format specified by the BITPIX keyword.
%The dimensions of the data are specified by the NAXIS1 and
%NAXIS2 keywords.
%
%Reference:   NASA/Science Office of Standards and Technology
%          "Definition of the Flexible Image Transport System"
%                          NOST 100-1.0
```

```
%
%          This and other FITS documents are available on-line at:
%          <A
HREF="http://www.gsfc.nasa.gov/astro/fits/basics_info.html">http://www.gsfc.nasa.gov/astro/fits/
basics_info.html</A>

%Set flag indicating unknown number of HDUs
if nargin<2
        n_hdu='unknown';
end;

if isstr(n_hdu)
        n_hdu=upper(n_hdu);
end;

%Allow user to specify a few keywords to indicate number of card sets
if strcmp(n_hdu,'HST') | strcmp(n_hdu,'STSCI') | strcmp(n_hdu,'MDS')
  n_hdu=6;
end
if strcmp(n_hdu,'FREI')
  n_hdu=2;
end

%Open the file
fid=-1;
if ~isstr(filename)
        filename=setstr(filename);
end;
if (isempty(findstr(filename,'.'))==1)
        filename=[filename,'.fits'];
end
[file,message] = fopen(filename,'r','l');
if file == -1
        error(message);
end


%First five cards must contain specific information
[d,simple,d]=parse_card(setstr(fread(file,80,'uchar')'));
[d,bitpix,d]=parse_card(setstr(fread(file,80,'uchar')'));
[d,naxis,d]=parse_card(setstr(fread(file,80,'uchar')'));
[d,naxis1,d]=parse_card(setstr(fread(file,80,'uchar')'));
[d,naxis2,d]=parse_card(setstr(fread(file,80,'uchar')'));


if n_hdu=='UNKNOWN'
        %Keep reading cards until one turns up with the keyword 'END'.
        n_card=5;
        keyword=' ';
        while(~strcmp(upper(deblank(keyword)),'END'))
                n_card=n_card+1;
                card=setstr(fread(file,80,'uchar')');
```

```
            keyword=parse_card(card);
        if (strcmp(upper(deblank(keyword)),'BZERO'))
                        [keyword,bzero,comment]=parse_card(card);
                        end
        if (strcmp(upper(deblank(keyword)),'BSCALE'))
                    [keyword,bscale,comment]=parse_card(card);
                        end
        end;
        %Go past the blank lines of padding before the start of the data
        n_blanks = 36 - rem(n_card,36);
        dummy=fread(file,n_blanks*80,'uchar');
else
        dummy=fread(file,((n_hdu*36)-5)*80,'uchar');
end;

%Read the data. Note big-endian switch is used (Mac and UNIX!
if bitpix==-64
  X=fread(file,naxis1.*naxis2,'float32',0,'ieee-be');
elseif bitpix==-32
  X=fread(file,naxis1.*naxis2,'float',0,'ieee-be');
elseif bitpix==8
  X=fread(file,naxis1.*naxis2,'uchar',0,'ieee-be');
elseif bitpix==16
  X=fread(file,naxis1*naxis2,'short',0,'ieee-be');
elseif bitpix==32
  X=fread(file,naxis1*naxis2,'long',0,'ieee-be');
else
  error('data type specified by BITPIX keyword is not -64, -32, 8, 16, or 32');
end;

%Clean up and output data
fclose(file);
 output_image=rot90(reshape(X,naxis1,naxis2));
```

-parse_card.m

```
function [keyword,value,comment] = parse_card(s)
%PARSE_CARD Parses a FITS header card.
%Reference:
%             NASA/Science Office of Standards and Technology
%         "Definition of the Flexible Image Transport System (FITS)"
%                    NOST 100-1.0    June 19, 1993

%Set defaults

keyword=[];value=[];comment=[];

%Get keyword in bytes 1 - 8
keyword=s(1:(min(max(size(s)),8)));
%keyword=s(1:8);
if nargout==1
  return;
```

```
end

%If keyword is blank then the line is a comment
if strcmp(keyword,'        ')
   keyword=[];
         value=[];
         comment=deblank(s(11:80));
         return;
end;


%Keyword is non-blank. Check if there is a corresponding value/comment.
%If not then the only possibilities are that bytes 11:80 are a comment
%or that they are blank
if ~strcmp(s(9:10),'= ')
   keyword=deblank(keyword);
         value=[];
         comment=deblank(s(11:80));
         return;
end;

%Card is a standard keyword/value/comment structure. Break the value/comment
%string (bytes 11 - 80) into separate strings by tokenizing on "/" character.
%Remove the leading and trailing blanks on the value and the trailing blanks
%on the comment.

keyword=deblank(keyword);
[value,comment]=strtok(s(11:80),'/');
comment=deblank(comment);
value=fliplr(deblank(fliplr(deblank(value))));

%Now figure out whether to output the value as a string or as a number.
%The FITS standard requires all values that are strings to be in single
%quotes like this: 'foo bar', so I can simply look for occurences of a
%single quote to flag a string. However, logical variables can take the
%values T or F without having any single quotes, so I'll have to look
%out for those also.

%Test for logical. Return logical as a string.
if strcmp(upper(value),'T') | strcmp(upper(value),'F')
         return;
end;

%Test for string. Return string unconverted.
if length(findstr("",value)) ~= 0
         return;
end;

%Only thing left is a number. Convert string to number.
value=str2num(value);
```

strtok.m :

```
%
% usage : [substring1,substring2] = strtok(input_string,char)
%
% Exemple :
%
%                   string='Bonjour / la France';
%
%     [zozo,zaza]=strtok(string,'/')
%     >> zozo = 'Bonjour '
%     >> zaza=' la France'
%
%     [zozo,zaza]=strtok(string,'a')
%     >> zozo = 'Bonjour / l'
%     >> zaza=' France'
%
%     [zozo,zaza]=strtok(string,'z')
%      >> zozo='Bonjour / la France';
%      >> zaza='';

function [substring1,substring2]=strtok(input_string,char)

qq=findstr(char,input_string);
if (length(qq)==0)
  substring1=input_string;
  substring2='';
else
  substring1=input_string(1:qq(1)-1);
  substring2=input_string(qq(1)+1:length(input_string));
endif
```

Ensuite il faut procéder de la manière suivante pour obtenir des graphes des champs:

Après avoir sélectionné le choix 5 : « ecrire l image des champs », le programme demande la définition (en pixel) que vous souhaitez obtenir pour votre image. (attention l'image obtenue n'est pas à l'echelle !).

> *exemple* :
> Quelle definition pour l image?
> nombre de lignes :
> 100
> nombre de collonnes
> 100

Une fois le calcul accompli il faut lancer octave (les fichiers dans lesquels se trouvent les images sont précédés de « carte_ » exemple « carte_Bz » pour l'image de la composante z du champ magnétique).

> *Exemple* :
> [root@baltazar solenoid]# octave

```
octave:1> A=file2matrix("carte_Bz");
octave:2> imagesc(A);
```

L'image s'affiche alors avec le logiciel « imagemagick » pour les distributions redhat, suze et mandrake.
Il est probable que cela ne marche pas sous d'autres distributions.

Conclusions :

J'espère avoir été clair sur l'utilisation de ce programme, si vous avez le moindre problème, vous pouvez m'écrire sur ma boîte : mlilom@hotmail.com.

Ce programme est une première version est peut être largement amélioré :
-intégrations numériques (qui dans cette version est une simple méthode de Newton)
-modèle de spire non plane (idée : 1 – modéliser une fraction de spire plane avec un delta de dirac puis sommer plusieurs de ces fractions avec un petit angle entre chacune permettant d'obtenir une spire « s'enroulant » avec la précision voulue (en contrôlant la taille des fractions de spires). 2 – modéliser directement une spire « s'enroulant » avec un delta de dirac dépendant de l'éloignement au centre mais aussi de la hauteur (voir Toshiharu Tominaka, « Vector Potential for a single helical current conductor », Nuclear Instruments and methods in Physics Research A 523 (2004) 1-8).