DETECÇÃO MAGNÉTICA DE MARCADORES UTILIZADOS EM ENSAIOS BIOLÓGICOS

Aluno: Jan Krueger Siqueira
Orientadores: Sônia Renaux Wanderley Louro e Antonio Carlos Oliveira Bruno

Introdução

Ensaios imunológicos em amostras biológicas consistem em métodos que medem a reação antígeno-anticorpo através de um marcador ligado ao anticorpo. Em geral se utilizam marcadores fluorescentes, por radioisótopos, ou enzimáticos. Estes métodos têm limitações dada a instabilidade dos marcadores ou devido a utilização de radiação. Recentemente, métodos magnéticos de detecção têm sido aplicados a estes ensaios através da utilização de marcadores contendo nanopartículas magnéticas em seu núcleo, na tentativa de aumentar a sensibilidade do ensaio. Isto pode levar a um diagnóstico precoce de determinadas patologias como tumores, doenças auto-imunes, etc.

Objetivos

Utilização de um dispositivo supercondutor de interferência quântica (SQUID) e de magnetômetros de efeito Hall de alta resolução de para detecção de marcadores magnéticos comerciais e de nanopartículas magnéticas. Desenvolvimento de uma técnica de calibração para o sistema de forma a relacionar o fluxo magnético medido com o número de marcadores magnéticos na amostra.

Metodologia

O processo geral foi: preparação das amostras, medição de seus campos magnéticos, plotagem dos dados, comparação desses dados com análises teóricas / simulações e documentação dos resultados encontrados. Os detalhes seguem abaixo.

Primeiramente, foram selecionados os tipos de marcadores magnéticos. Tratam-se de partículas de silica ou dextran diluídas em água, contendo no seus núcleos partículas ferromagnéticas ou paramagnéticas. Dentre as diversas opções existentes no mercado, as seguintes foram utilizadas: MagPrep, Sicastar e NanoMag – esta última apresentava suas partículas em três tamanhos: 130 nm, 250 nm e 500 nm. As duas primeiras tinham tamanhos da ordem de 1 μm. Num momento posterior, incluiu-se também partículas de ferrita de cobalto com cerca de 30 nm de tamanho. Assim sendo, criava-se, para cada tipo, amostras de diferentes concentrações no formato de gotas com cerca de 5 μl. Após a evaporação da água, o resíduo de partículas era coberto por uma fita adesiva, a fim de se evitar seu deslocamento durante as magnetizações. Todas as informações do processo foram anotadas.

A medição das amostras se deu, primeiramente, com o SQUID. Neste processo, as amostras eram magnetizadas através de um íma de NdFeB em um campo B = 1500 G. Em seguida, eram postas individualmente num estágio x-y movido por motores de passo e controlado por um computador para o mapeamento do seu campo remanente. O processo era repetido para magnetizações e desmagnetizações sucessivas, na busca de uma constância dos resultados.

A utilização do SQUID permitiu a detecção dos campos remanentes mais fracos, porém o equipamento exigia operação cuidadosa, além de ser operado à hélio líquido. Na tentativa de simplificar o método, utilizou-se uma alternativa: o uso de magnetômetros de

efeito Hall, menos sensíveis porém mais práticos. Para os casos em que o campo remanente medido era baixo, fez-se a medição na presença de um campo magnético externo de 1kG.

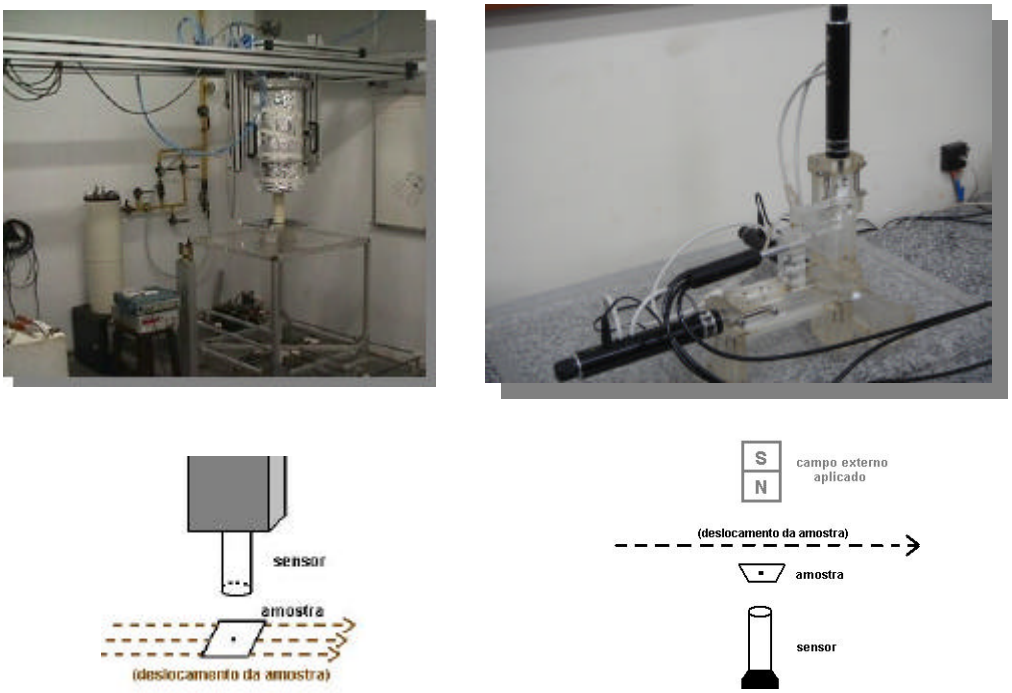


Fig. 1 – O SQUID e o sistema de Efeito Hall, usados para medir a densidade de fluxo magnético resultante das amostras. No primeiro caso, a excursão da amostra foi feita em várias linhas, a fim de garantir a passagem mais próxima ao centro do sensor. No segundo, o percurso foi feito unidimensionamente, graças à facilidade do funcionamento e da visualização.

Os resultados adquiridos em ambas as medidas serão listados mais a seguir.

Tanto o controle das posições como o da medição através do SQUID e dos magnetômetros foram realizados em um ambiente LabView já disponível no laboratório. Por outro lado, a plotagem e análises dos dados se deu através de funções e rotinas desenvolvidas por mim em Matlab. Após ajustes dos parâmetros nas curvas Campo x Posição, o formato característico aparecia em todas as ocasiões, e dele se retiravam informações para finalizar o cálculo dos momentos de dipolo magnético das amostras. No caso do método com o SQUID, a informação de interesse foi a intensidade do sinal obtido; e, conhecido o funcionamento do aparelho, o valor do momento de dipolo provinha de um cálculo direto. Já para as medições feitas com os gaussímetros, foram feitas simulações com um modelo de uma espira com corrente (a geometria circular da amostra permitia tal aproximação). Para isso, utilizou-se o programa VF Opera 8.7 e também, como alternativa, uma integração numérica da lei de Biot-Savart no próprio Matlab. Comparando então suas respostas com as experimentais, obtiveram-se os valores para o cálculo do momento magnético das amostras.

Conclusões

As partículas apresentaram boas respostas nas medições, em especial as do tipo MagPrep, que mostraram um bom campo remanente até mesmo em baixas concentrações.

As experiências feitas com os gaussímetros em amostras com baixa concentração apresentaram problemas de sinal-ruído, mesmo se analisando as amostras na presença de campos magnéticos externos. Isto deve-se ao fato de que os gaussímetros não são preparados para realizar medidas sensíveis em ambientes ruidosos.

Cada um dos dois métodos apresentou fatores positivos e negativos de custo/benefício. Todavia, mostraram-se como meios interessantes na detecção dos marcadores magnéticos. Com a continuidade do trabalho, será talvez possível alcançar uma nova e eficiente ferramenta para a medicina e outras áreas da nanotecnologia.

Apêndice 1: Simulação com Biot Savart

Conforme dito anteriormente, a amostra magnetizada pode ser modelada como um dipolo magnético centrado na origem. O motivo é simples: embora o material magnético não esteja presente somente nas bordas, a "corrente" de uma pequena área interna se cancela com a da adjacente. Apenas a corrente do trecho externo se preserva, conforme a figura mais abaixo.

A medição através dos magnetômetros se dá ao longo de várias posições duma reta paralela ao eixo y no plano yz. Tentaremos, pois, aplicar Biot-Savart para calcular a densidade de fluxo magnético no lugar geométrico em questão. É importante mencionar que apenas a componente Z do vetor B será considerada, pois é esta que é medida e usada no cálculo do momento de dipolo magnético.

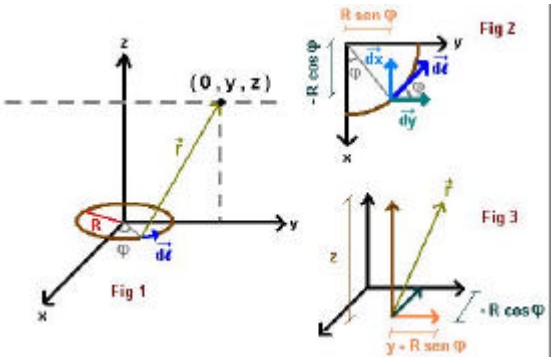


Fig 4: anulação das correntes internas, resultando no modelo da figura 1

Fig 2: $dl = R\,dj \qquad dx = -dl\sin j$

$d\bar{l} = dx\,\bar{i} + dy\,\bar{j} \qquad dy = dl\cos j$

$d\bar{l} = -R\sin j\ dj\ \bar{i} + R\cos j\ dj\ \bar{j}$

Fig 3:

$$\bar{r} = -R\cos j\ \bar{i} + (y - R\sin j)\,\bar{j} + z\,\bar{k}$$

$$r = [R^2\cos^2 j + (y^2 - 2Ry\sin j + R^2\sin^2 j) + z^2]^{1/2} \Rightarrow r = (R^2 + y^2 + z^2 - 2Ry\sin j)^{1/2}$$

Biot-Savart:

$$dB_z = \frac{m_0 I}{4p}\frac{\left|(d\bar{l}\times\bar{r})_z\right|}{r^3} = \frac{m_0 I}{4p}\frac{(-R\sin j\ dj)(y - R\sin j) + R\cos j\ dj\ R\cos j}{(R^2 + y^2 + z^2 - 2Ry\sin j)^{3/2}}$$

$$dB_z = \frac{m_0 I}{4p}\frac{(-Ry\sin j\ dj + R^2\sin^2 dj + R^2\cos^2 dj)}{(R^2 + y^2 + z^2 - 2Ry\sin j)^{3/2}} = \frac{m_0 I}{4p}\frac{(-Ry\sin j + R^2)dj}{(R^2 + y^2 + z^2 - 2Ry\sin j)^{3/2}}$$

$$B_z = \int_0^{2p}\frac{m_0 I}{4p}\frac{(-Ry\sin j + R^2)dj}{(R^2 + y^2 + z^2 - 2Ry\sin j)^{3/2}}$$

$$B_z = \frac{m_0 I}{4p}\left[R^2\int_0^{2p}\frac{dj}{(R^2 + y^2 + z^2 - 2Ry\sin j)^{3/2}} - Ry\int_0^{2p}\frac{\sin j\ dj}{(R^2 + y^2 + z^2 - 2Ry\sin j)^{3/2}}\right]$$

$$dB_y = \frac{m_0 I}{4p}\frac{\left|(d\bar{l}\times\bar{r})_y\right|}{r^3} = \frac{m_0 I}{4p}\frac{z\,R\sin j\;dj}{(R^2+y^2+z^2-2Ry\sin j)^{3/2}}$$

$$B_y = \frac{m_0 I}{4p} z R \int_0^{2p} \frac{\sin j\;dj}{(R^2+y^2+z^2-2Ry\sin j)^{3/2}}$$

$$dB_x = \frac{m_0 I}{4p}\frac{\left|(d\bar{l}\times\bar{r})_x\right|}{r^3} = \frac{m_0 I}{4p}\frac{z\,R\cos j\;dj}{(R^2+y^2+z^2-2Ry\sin j)^{3/2}}$$

$$B_x = \frac{m_0 I}{4p} z R \int_0^{2p} \frac{\cos j\;dj}{(R^2+y^2+z^2-2Ry\sin j)^{3/2}}$$

Vale mencionar que esta última componente (X) resulta em zero, devido à simetria.

Conhecidos R, y, z, e I, "basta resolver" a integral definida para obtermos a expressão do campo ao longo da reta anteriormente desenhada.

A função **CalcularBZBiotSavartReta** resolve a questão numericamente, com auxílio de outras funções (**Integrando1** e **Integrando2**). No entanto, o nosso objetivo é determinar, a partir do campo medido, a distância D ao sensor e a corrente I que circula no dipolo – isto é, o caminho inverso, que tornará possível o campo do momento de dipolo. Faz-se necessário, pois, uma rotina que compare os dados obtidos com o resultado da simulação para distâncias e correntes arbitrárias, até alcançar valores de B que se assemelhem ao experimental. Daí temos a **ObterDI,** que faz uso da função de otimização **MinimizarErro.** Para calcular a distância D, normalizamos ambas as curvas experimental e simulada e comparamos uma com a outra, já que o grau de proximidade ao sensor influencia também no formato do gráfico. Feito isso, a corrente é obtida comparando as curvas sem normalização, já que I é apenas um fator de proporcionalidade na expressão de B_z.

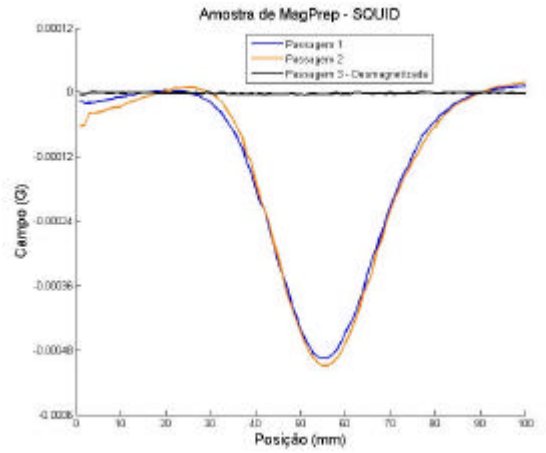
Todas as rotinas e funções citadas estão listadas no Apêndice 3.

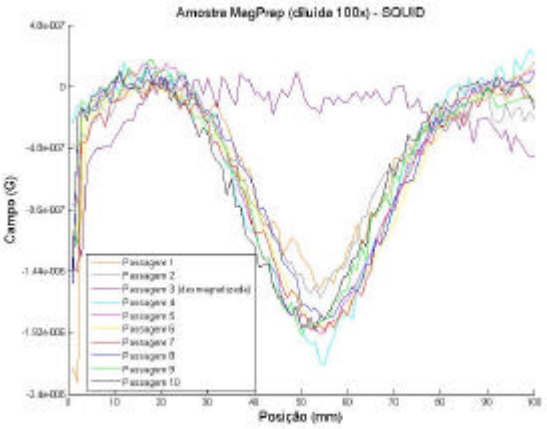
OBS: A resolução numérica da integral em vários pontos pode se tornar lenta. Por tal motivo, foi usado o software VectorField Opera 8.7 como uma ferramenta mais otimizada. Eis por que a rotina ObterDI utiliza, na verdade, as funções **ChamarOperaNorm** e **ChamarOperaNaoNorm**. Caso o programa citado não esteja disponível, a rotina deverá sofrer modificações para utilizar a função CalcularBZBiotSavartReta.

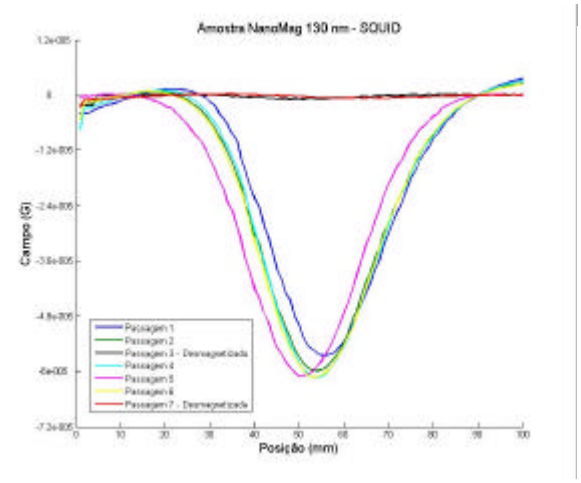
Apêndice 2: Resultados

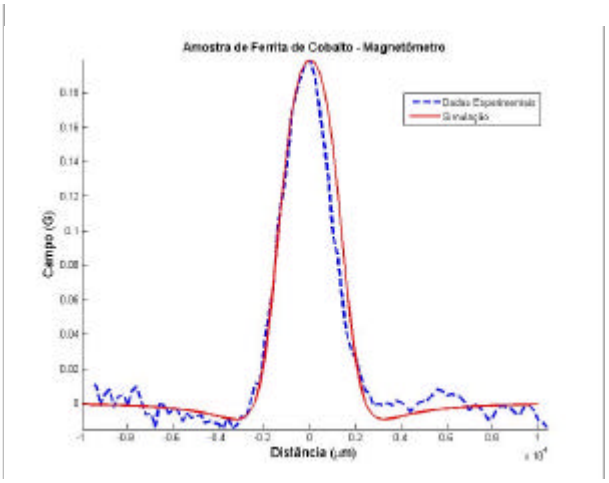
Apresentam-se abaixo os gráficos obtidos em algumas das medições, bem como o valor de momento de dipolo das amostras:

OBS: a medição referente à Ferrita de Cobalto apresenta um momento de dipolo magnético bem superior às demais porque foi medido com o magnetômetro na presença de um campo externo, ao invés do SQUID com o campo remanente.









- MagPrep (50 mg/mL): 9.33 A.m²/kg
- MagPrep (0.5 mg/mL): 3.42 A.m²/kg
- NanoMag 130 nm: $2.03*10^{-2}$ A.m²/kg
- Ferrita de Cobalto: $3.09*10^{3}$ A.m²/kg (em 1000 G)

Apêndice 3: Funções e Rotinas do MatLab Desenvolvidas

```
%%%%%%%%%%%%%%%%%%%%%%% FUNÇÃO CalcularBiotSavartEixo %%%%%%%%%%%%%%%%%%%%%%%
%
%  Autor: Jan Krueger Siqueira, jun/2006
%
%  Descrição da Função:
%    Calcula o campo dum dipólo magnético no eixo que passa no seu centro.
%    Por simetria, tal campo possuira apenas componente Z.
%
%  Chamada da Função:
%    B = CalcularBiotSavartEixo( distancia , raio , corrente ) ;
%
%  Parâmetros Recebidos:
%    distância - distância ao centro            -> em metros
%    raio      - raio da espira (dipólo)        -> em metros
%    corrente  - corrente que circula no dipólo -> em Ampères
%
%  Valor retornado:
%    B         - valor do campo                 -> em Gauss
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function B = CalcularBiotSavartEixo ( distancia , raio , corrente )

% checar número de parâmetros
    if nargin ~= 3
        error('*** ERRO *** CalcularBiotSavartEixo requer 3 parâmetros!') ;
    end

% constante
    mi0 = 4e-7 * pi ;
```

```
% Biot-Savart -> B em Tesla
    B = ( mi0 .* corrente .* ( raio ) .^ 2 ) / ...
           ( 2 .* ( distancia .^ 2 + raio .^ 2 ) .^ ( 3 / 2 ) ) ;

% em Gauss
    B = B * 1e4 ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% FIM DA FUNÇÃO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%% FUNÇÃO CalcularBZBiotSavartReta %%%%%%%%%%%%%%%%%%%%%%
%
%  Autor: Jan Krueger Siqueira , jun/2006
%
%  Descrição da Função:
%    Calcula a componente Z campo dum dipólo magnético centrado na origem
%     (sob o plano xy), nos pontos duma reta paralela ao eixo y no plano yz.
%    Esta funçao resolve a integral de Biot-Savart (referente a este caso)
%     numericamente.
%    Para muito pontos, o algorítmo fica lento; as funções ChamarOpera”...”
%     são mais rápidas. Mas esta funçao aqui não precisa, ao contrário das
%     outras, de recursos e programas externos ao MatLab.
%
%  Chamada da Função:
%    Bz = CalcularBZBiotSavartReta( z , R , I , vetY ) ;
%
%  Parâmetros Recebidos:
%    z         - distância da reta à origem         -> em metros
%    R         - raio da espira (dipolo)            -> em metros
%    I         - corrente que circula no dipólo     -> em Ampères
%    vetY      - vetor com posicoes y da reta aonde
%                se deseja calcular o campo         -> em metros
%
%  Valor retornado:
%    Bz        - vetor com os valores de campo em Z -> em Gauss
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Bz = CalcularBZBiotSavartReta ( z , R , I , vetY )

% checar número de parâmetros
    if nargin ~= 4
        error('*** ERRO *** CalcularBZBiotSavartReta requer 4 parâmetros!') ;
    end

% constante
    mi = 4e-7 * pi ;              % H / m

% Tentiva frustrada de resolver sem aproximações:

    % colocando no prompt o cálculo do campo por Biot-Savart
        % syms phi y z R mi I ;
        % ( ( mi * I ) / ( 4 * pi ) ) * ( R * int( 1 / ( R^2 + y^2 + z^2 - 2 * R * y * sin(
phi ) ) ^ ( 3 / 2 ) , phi , 0 , 2*pi ) - R * y * int( sin ( phi ) / ( R^2 + y^2 + z^2 - 2 * R
* y * sin( phi ) ) ^ ( 3 / 2 ) , phi , 0 , 2*pi ) )

% Infelizmente, o MatLab daqui não soube como calcular as funçoes que ele
%   mesmo deu como resultado da integral... logo, vamos de outro jeito

% integral numérica com funções auxiliares
    for i = 1 : length( vetY )
        Bz(i) = ( ( mi * I ) / ( 4 * pi ) ) * ...
                ( R^2 * quad( @Integrando1 , 0 , 2 * pi , [] , [] , ...
                              vetY( i ) , z , R ) ...
                  - R * vetY( i ) * quad( @Integrando2 , 0 , 2 * pi , ...
                                          [] , [] , vetY( i ) , z , R ) ) ;
    end

    % OBS: fiz um "for" porque "quad" não gosta de receber e passar
    %   argumentos vetores para a função de integração (ele faz isso em seu
    %   algorítmo, o que pode entrar em conflito com os parâmetros do
    %   usuário)

% de Tesla para Gauss
    Bz = Bz .* 1e4 ;

% Testando a função (DESATIVADO):
%{
    % plotando
        plot( y , Bz , 'b' ) ;
        title( 'Campo Magnético x Posição da Amostra' ) ;
        xlabel( 'Distância ( m )' ) ;
        ylabel( 'Campo (G)' ) ;

        hold on ;
```

```
    % plotando o simulado no Opera e copiado manualmente num arquivo
        load comparando.txt ;

        % convertendo x para metros
            comparando( : , 1 ) = comparando( : , 1 ) / 100 ;
        % consertando curva para cima
            comparando( : , 4 ) = - comparando( : , 4 ) ;

        plot( comparando( : , 1 ) , comparando( : , 4 ) , 'r') ;
        title( 'Campo Magnético x Posição da Amostra' ) ;
        xlabel( 'Distância ( m )' ) ;
        ylabel( 'Campo (G)' ) ;
        legend( 'Biot-Savart Numérico' , 'Opera' ) ;
%}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% FIM DA FUNÇÃO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%% FUNÇÃO ChamarOperaNaoNorm %%%%%%%%%%%%%%%%%%%%%%%%
%
%  Autor: Jan Krueger Siqueira , jun/2006
%
%  Descrição da Função:
%     Chama programa Opera 8.7 e simula solenóide pré-estabelecida.
%     Obtem valores de campo em Z a uma distância passada como parâmetro.
%     NÃO NORMALIZA curva de valores.
%     Retorna esses valores de campo num vetor.
%
%  Chamada da Função:
%     Bz = ChamarOperaNaoNorm ( J , d , x , r ) ;
%
%  Parâmetros Recebidos:
%     J - densidade de corrente que circula na amostra   -> em Ampères/cm²
%     x - posições onde se deseja calcular o campo       -> em centímetros
%     d - distância (vertical) do sensor à amostra       -> em centímetros
%     r - raio da amostra                                -> em centímetros
%
%  Valor retornado:
%     Bz - vetor com os valores de campo em Z das posições passadas em "x"
%          (sensor se desloca) -> em Gauss
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Bz = ChamarOperaNaoNorm ( J , x , d , r )

% checar parâmetros
    if nargin ~= 4
        error( '*** ERRO *** ChamarOperaNaoNorm requer 4 parâmetros!' ) ;
    end

    if r < 0
        error( '*** ERRO *** Raio negativo!!!' ) ;
    end

% criar opera.comi com "x", "d" e "r" para simular dipolo magnético
    arq_txt = fopen( 'opera.comi' , 'wt' ) ;
    fprintf( arq_txt , 'CONDUCTOR ACTION=DEFINE TYPE=SOLENOID, X1=%f Y1=-5.0E-04 X2=%f Y2=-
5.0E-04 X3=%f Y3=5.0E-04 X4=%f Y4=5.0E-04 CU1=0 CU2=0 CU3=0 CU4=0 CURD=%f,\n' , ...
             r + 0.001 , r , r , r + 0.001 , J ) ;
    fprintf( arq_txt , 'TOLERANCE=1 PHASE=ONE XCEN=0 YCEN=0 ZCEN=0 THETA1=0 PHI1=0 PSI1=0 X0=0
Y0=0 Z0=0 T=0 P=0 S=0 IRXY=0 IRYZ=0 IRZX=0 SYMMETRY=1\n' ) ;
    fprintf( arq_txt , 'LINE X1=%f Y1=%f Y2=Y1 Z1=0 Z2=Z1 X2=%f NP=%f\n' , ...
             x( 1 ) , d , x( length( x ) ) , length( x ) - 1 ) ;
    fprintf( arq_txt , 'PLOT STYLE=AUTO COLOUR=AUTO FILE=TEMP COMP=By ERASE=YES YMINIMUM=*
YMAXIMUM=* LOCAL=YES ORDINATE=NUMBER PRINT=YES\n' ) ;
    fprintf( arq_txt , 'END' ) ;
    fclose( arq_txt ) ;

% executa opera pos (esse programa fechará sozinho)
    !c:\program files\vector fields\opera 8.7\post\post.exe

% ler o arquivo lp
    fid = fopen( 'c:\\Jan\\2006.1\\opera_logs\\Opera3d_Post_1.lp' , 'rt' );
    line = fgetl( fid ) ;        % não existe "do while" no MATLAB >(
    while line( 4 ) ~= 'X'
       line = fgetl( fid ) ;
    end
    dados = fscanf( fid , '%f' ) ;
    fclose( fid ) ;

% deletar arquivos .lp e .txt do diretório
    !del c:\Jan\2006.1\opera_logs\*.lp
    !del c:\Jan\2006.1\opera_logs\*.log

% obter vetor by (elementos múltiplos de 4 do vetor dados)
    for i = 4 : 4 : length( dados )
        by( i / 4 ) = dados( i ) ;
```

```
    end

% garante que curva "chapéu" esteja para cima
    if by( round( length( by ) / 2 ) ) < 0
        by = - by ;
    end

% retornar by
    Bz = by ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% FIM DA FUNÇÃO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%% FUNÇÃO ChamarOperaNorm %%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Autor: Jan Krueger Siqueira , mai/2006
%
%  Descrição da Função:
%     Chama programa Opera 8.7 e simula solenóide pré-estabelecida.
%     Obtem valores de campo em Z a uma distância passada como parâmetro.
%     A densidade de corrente na amostra é arbitrada para um valor qualquer
%     NORMALIZA curva de valores.
%     Retorna esses valores de campo num vetor.
%
%      Chamada da Função:
%         B = ChamarOperaNorm( d , x , r ) ;
%
%   Parâmetros Recebidos:
%     d - distância (vertical) do sensor à amostra       -> em centímetros
%     x - posições onde se deseja calcular o campo       -> em centímetros
%     r - raio da amostra                                -> em centímetros
%
%   Valor retornado:
%     Bz - vetor com os valores de campo em Z das posições passadas em "x"
%         (sensor se desloca) -> em Gauss
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Bz = ChamarOperaNorm ( d , x , r )

% checar parâmetros
    if nargin ~= 3
        error( '*** ERRO *** ChamarOperaNorm requer 3 parâmetros!' ) ;
    end

    if r < 0
        error( '*** ERRO *** Raio negativo!!!' ) ;
    end

% criar opera.comi com "x" e "d" para simular dipolo magnético
    arq_txt = fopen( 'opera.comi' , 'wt' ) ;
    fprintf( arq_txt , 'CONDUCTOR ACTION=DEFINE TYPE=SOLENOID, X1=%f Y1=-5.0E-04 X2=%f Y2=-
5.0E-04 X3=%f Y3=5.0E-04 X4=%f Y4=5.0E-04 CU1=0 CU2=0 CU3=0 CU4=0 CURD=1,\n' , ...
             r + 0.001 , r , r , r + 0.001 ) ;
    fprintf( arq_txt , 'TOLERANCE=1 PHASE=ONE XCEN=0 YCEN=0 ZCEN=0 THETA1=0 PHI1=0 PSI1=0 X0=0
Y0=0 Z0=0 T=0 P=0 S=0 IRXY=0 IRYZ=0 IRZX=0 SYMMETRY=1\n' ) ;
    fprintf( arq_txt , 'LINE X1=%f Y1=%f Y2=Y1 Z1=0 Z2=Z1 X2=%f NP=%f\n' , ...
             x( 1 ) , d , x( length( x ) ) , length( x ) - 1 ) ;
    fprintf( arq_txt , 'PLOT STYLE=AUTO COLOUR=AUTO FILE=TEMP COMP=By ERASE=YES YMINIMUM=*
YMAXIMUM=* LOCAL=YES ORDINATE=NUMBER PRINT=YES\n' ) ;
    fprintf( arq_txt , 'END' ) ;
    fclose( arq_txt ) ;

% executa opera pos (esse programa fechará sozinho)
    !c:\program files\vector fields\opera 8.7\post\post.exe

% ler o arquivo lp
    fid = fopen( 'c:\\Jan\\2006.1\\opera_logs\\Opera3d_Post_1.lp' , 'rt' );
    line = fgetl( fid ) ;        % não existe "do while" no MATLAB
    while line( 4 ) ~= 'X'
        line = fgetl( fid ) ;
    end
    dados = fscanf( fid , '%f' ) ;
    fclose( fid ) ;

% deletar arquivos .lp e .txt do diretório
    !del c:\Jan\2006.1\opera_logs\*.lp
    !del c:\Jan\2006.1\opera_logs\*.log

% obter vetor by (elementos múltiplos de 4 do vetor dados)
    for i = 4 : 4 : length( dados )
        by( i / 4 ) = dados( i ) ;
    end

% normaliza curva
    by = by / max( abs ( by ) ) ;
```

```
% garante que curva "chapéu" esteja para cima
    if by( round( length( by ) / 2 ) ) < 0
        by = - by ;
    end

% retornar by
    Bz = by ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% FIM DA FUNÇÃO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%% FUNÇÃO Integrando1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Autor: Jan Krueger Siqueira , jun/2006
%
%  Descrição da Funçao:
%    Função auxiliar da "CalcularBZBiotSavartReta".
%    Passada como argumento para a funçao "quad" (integral numérica).
%
%  Chamada da Função:
%    result = Integrando1( phi , y , z , R ) ;
%
%  Parâmetros Recebidos:
%    phi - ângulo pelo qual se integra a espira           - em radianos
%    y   - distância y do ponto onde se calcula o campo   - em metros
%    z   - distância z do ponto onde se calcula o campo   - em metros
%    R   - raio do dipólo magnético                       - em metros
%
%  Valor retornado:
%    result - valor do 1º integrando (aplicado aos valores) da expressão
%             do campo
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function result = Integrando1 ( phi , y , z , R )

% checar número de parâmetros
    if nargin ~= 4
        error( '*** ERRO *** Integrando1 requer 4 parâmetros!' ) ;
    end

% aplicar expressão
    result = 1 .* ( ( R.^2 + y.^2 + z.^2 - 2 .* R .* y .* sin( phi ) ) ...
                  .^ ( -3 / 2 ) );

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% FIM DA FUNÇÃO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%% FUNÇÃO Integrando2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Autor: Jan Krueger Siqueira , jun/2006
%
%  Descrição da Funçao:
%    Função auxiliar da "CalcularBZBiotSavartReta"
%    Passada como argumento para a funçao "quad" (integral numérica).
%
%  Chamada da Função:
%    result = Integrando2( phi , y , z , R ) ;
%
%  Parâmetros Recebidos:
%    phi - ângulo pelo qual se integra a espira           - em radianos
%    y   - distância y do ponto onde se calcula o campo   - em metros
%    z   - distância z do ponto onde se calcula o campo   - em metros
%    R   - raio do dipólo magnético                       - em metros
%
%  Valor retornado:
%    result - valor do 2º integrando (aplicado aos valores) da expressão
%             do campo
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function result = Integrando2 ( phi , y , z , R )

% checar número de parâmetros
    if nargin ~= 4
        error( '*** ERRO *** Integrando2 requer 4 parâmetros!' ) ;
    end

% aplicar expressão
    result = sin( phi ) .* ...
             ( ( R.^2 + y.^2 + z.^2 - 2 .* R .* y .* sin( phi ) ) ...
             .^ ( -3 / 2 ) );

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% FIM DA FUNÇÃO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%% FUNÇÃO LocalizarIndex %%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%
%   Autor: Jan Krueger Siqueira , jun/2006
%
%   Descrição da Funçao:
%     Localiza index (num vetor) dum elemento cujo valor melhor se aproxima
%       do procurado.
%     Caso haja empate, o primeiro dos elementos é retornado.
%
%   Chamada da Função:
%     index = LocalizarIndex( val , vet ) ;
%
%   Parâmetros Recebidos:
%     val - valor procurado
%     vet - vetor de elemento
%
%   Valor retornado:
%     index - index do elemento
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function index = LocalizarIndex ( val , vet )

    erroMin = Inf ;
    index = 0 ;

    for i = 1 : length(vet)

        erro = abs( vet( i ) - val ) ;

        if erro < erroMin
            erroMin = erro ;
            index = i ;
        end

    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%% FIM DA FUNÇÃO %%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%% FUNÇÃO MinimizarErro %%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Autor: Jan Krueger Siqueira , jun/2006
%
%  Descrição da Funçao:
%    Parecida com a funçao lsqcurvefit.
%    Dada uma função Y = f(X), tenta encontrar um valor "X" que resulte no
%      (ou seja próximo do) valor "Y" desejado.
%    "X" e "Y" podem ser vetores.
%    O erro do "X" encontrado varia bastante dependendo da funçao.
%
%  Chamada da Função:
%    X = MinimizarErro( funcao , valorInf , valorSup , respEsp , P1 ,
%                          P2 , ... ) ;
%
%  Parâmetros Recebidos:
%    funcao   - função a ser "otimizada" (entre aspas ou com @ na frente)
%    valorInf - limite inferior de onde se deseja encontrar a resposta
%    valorSup - limite superior de onde se deseja encontrar a resposta
%    respEsp  - resposta ("Y") que se espera encontrar em "funçao"
%               com o parâmetro procurado ("X")
%    P1,P2,...- (OPCIONAL) parâmetros extras a serem passados para "função"
%
%    A funçao passada será chamada: funcao( X , P1 , P2 , ... )
%                                ou: funcao( X )
%
%  Valor retornado:
%    X        - parâmetro procurado
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function X = MinimizarErro ( funcao , valorInf , valorSup , respEsp , ...
                             varargin )

% checar parâmetros
    if nargin < 4
        error( '*** ERRO *** MinizarErro2 requer 4 ou mais parâmetros!' ) ;
    end

    if valorInf >= valorSup
        error( '*** ERRO *** Limite inferior >= Limite Superior!!!' ) ;
    end

% dados iniciais
    erroCima     = 100000000 ;
    erroBaixo    = 100000000 ;

    erroAnterior = 100000000 ;
```

```
    resultCima   = 0 ;
    resultBaixo  = 1 ;

% minimizar quadrados por "busca binária"
    %while erroAnterior > 0.01
    for i = 1:10

        valorMeio = ( valorInf + valorSup ) / 2 ;

        resultCima = funcao( valorSup , varargin{ : } ) ;
        resultBaixo = funcao( valorInf , varargin{ : } ) ;

        erroCima = sum( ( resultCima - respEsp ) .^ 2 ) ;
        erroBaixo = sum( ( resultBaixo - respEsp ) .^2 ) ;

        if erroBaixo < erroCima
            erroAnterior = erroBaixo ;
            valorSup = valorMeio ;
        elseif erroBaixo > erroCima
            erroAnterior = erroCima ;
            valorInf = valorMeio ;
        else
            %X = valorMeio ;
            %break ;
        end

    end

    X =  valorMeio ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% FIM DA FUNÇÃO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ROTINA ObterDI %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Autor: Jan Krueger Siqueira , jun/2006
%
%  Descrição da Rotina:
%    Com base nos dados experimentais, calcula a corrente da amostra (I) e
%      sua distância ao sensor (D).
%    Antes de se iniciar a rotina, o gráfico experimental deverá já estar
%      aberto.
%    Num dado momento, espera-se que o usuário selecione o pico do gráfico.
%    Esta rotina chama o programa VectorField Opera 8.7.
%    No final, plota-se um gráfico do experimental x simulado, com os
%      valores de otimizaçao obtidos.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% limpar tela
    clc ;

% opção do MATLAB para exibir com mais casas decimais
    format long ;

% definir quantidade de pontos a serem capturados (não conta ponto do meio)
    QUANT = 26 ;

% definir RAIO DA AMOSTRA (em cm)
    RAIO = 0.15 ;

% deletar arquivos ".lp", ".comi" ".txt" do diretório
    !del c:\Jan\2006.1\opera_logs\*.lp
    !del c:\Jan\2006.1\opera_logs\*.log
    !del c:\Jan\2006.1\*.comi

% captura pontos do gráfico atual (já desenhado!)
    XData = get( get( gca , 'children' ) , 'XData' ) ;
    YData = get( get( gca , 'children' ) , 'YData' ) ;

    if isempty( XData )
        error('*** ERRO *** Erro na leitura do gráfico experimental!' ) ;
    end

    % se gráfico estiver invertido, joga pra cima
        if YData( round( length( YData ) / 2 ) ) < 0
            YData = - YData ;
        end

    % localizar pico manualmente
        disp( 'Escolha o pico com o mouse (e aperte ENTER):' ) ;
        [X_max Y_max] = getpts ;

        if length( X_max ) ~= 1
            error( '*** Erro *** Escolha exatamente 1 ponto!' ) ;
        end
```

```
    % localizar índice mais próximo do ponto selecionado
        i_max = LocalizarIndex( X_max , XData ) ;

    % obter valor Y desse ponto
        Y_max = YData( i_max ) ;

    % ajustando x( i_max ) para 0
        XData = XData - XData( i_max ) ;

    % restringir quantidade de pontos (em torno do pico)
        XData = XData( i_max - QUANT / 2 : i_max + QUANT / 2 ) ;
        YData = YData( i_max - QUANT / 2 : i_max + QUANT / 2 ) ;

    % de microns para centímetros
        XData = XData / 10000 ;

% otimizaçao de d
    % normalizando YData
        YDataNORM = YData / Y_max ;

    % valores iniciais ( para d )
         valorInf = 0.08 ;
        valorSup = 0.5 ;

    d = MinimizarErro( @ChamarOperaNorm , valorInf , valorSup , ...
                         YDataNORM , XData , RAIO )

% otimizaçao de J
    % valores iniciais ( para J )
        valorInf = 1e4 ;
        valorSup = 1.5e7 ;

    J = MinimizarErro( @ChamarOperaNaoNorm , valorInf , valorSup , ...
                       YData , XData , d , RAIO )

% plotagem comparativa ( experimetal x simulado )
    figure ;
    hold on ;

    plot( XData , YData , 'b-x' , 'LineWidth' , [2] ) ;
    plot( XData , ChamarOperaNaoNorm( J , XData , d , RAIO ) , 'k-x' , ...
          'LineWidth' , [2] ) ;

    grid ;
    xlabel( 'Distância (\mum)' , 'fontsize' , 13 ) ;
    ylabel( 'Campo (G) ' , 'fontsize' , 13 ) ;
    title( strcat( 'Distância encontrada: ' , num2str( d ) , ...
                   'cm // J encontrado: ' ,  num2str( J ) , 'A/cm²' ) ) ;
    legend( 'Experimental' , 'Simulação com VF Opera 8.7' ) ;

    set( gca , 'fontsize' , 13 ) ;
    set( gcf , 'Position' , [200 200 800 600] ) ;
    axis tight ;

% limpar variáveis
    clear ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% FIM DA ROTINA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%% ROTINA PlotarDados %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  Autor: Jan Krueger Siqueira , abr/2006
%
%  Descrição da Rotina:
%    Lê dados experimentais de campo duma amostra num arquivo e plota-os
%      num gráfico.
%    Usuário entrará com o nome do arquivo quando solicitado.
%    Num dado momento, será pedida a seleçao de pontos, num gráfico
%      parcial, para alinhar a base da curva para a horizontal.
%    Se o gráfico for aceito pelo usuário, será pedida a digitação de um
%      título e o nome para o arquivo do gráfico (será salvo em jpg e fig)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% limpa tela de comandos
    clc ;

% carrega arquivo e guarda nome de sua variável num string
    arquivo = input( 'Digite o nome do arquivo: ' , 's' ) ;
    arq = load( arquivo ) ;

% nova tela de gráficos, para não sobrescrever nenhum outro
    figure ;

% plota gráfico sem correção
```

```
    plot( arq( : , 5 ) , arq( : , 3 ) ,  '-x' , 'LineWidth' , [2] ) ;
    grid on ;
    xlabel( 'Distancia (\mum)' , 'fontsize' , 13 ) ;
    ylabel( 'Campo (G)' , 'fontsize' , 13 ) ;
    set( gcf , 'Position' , [200 200 800 600] ) ;

% pede para usuário selecionar pontos
    [x y] = getpts ;

    if length( x ) < 2
        error( '*** ERRO *** Selecione 2 pontos ou mais!' ) ;
    end

% conserta possivel rampa, calculando a reta entre os ponto selecionados
%  e depois subtraindo gráfico dessa reta
    [P , Lixo] = polyfit( x , y , 1 ) ;

    arq( : , 3 ) = 3 * ( arq( : , 3 ) - polyval( P , arq( : , 5 ) ) ) ;

% plota, coloca grade, dá nome aos eixos, ajeita janela e mostra pico na
%  legenda
    plot( arq( : , 5 ) , arq( : , 3 ) , '-x' , 'LineWidth' , [2] ) ;
    grid on ;
    xlabel( 'Distancia (\mum)' , 'fontsize' , 13 ) ;
    arquivo = strrep( arquivo , '_' , '-' ) ;
    ylabel( strcat( 'Campo (G) - Arquivo->' , arquivo ) , 'fontsize' , ...
            13 ) ;
    set( gca , 'fontsize' , 13 ) ;
    set( gcf , 'Position' , [200 200 800 600] ) ;
    axis tight ;

    pico = max( abs( arq( : , 3 ) ) ) ;
    pico = strcat( 'Pico: ' , num2str( pico ) ) ;
    pico = strrep( pico , ':' , ': ' ) ;
    legend( pico ) ;
    plotedit on ;

% salva arquivo
    resposta = input( 'Salvar figura?(s/n) ' , 's' ) ;

    if ( resposta == 's' || resposta == 'S' )
        titulo = input( 'Digite o titulo do gráfico: ' , 's' ) ;
        title( titulo , 'fontsize' , 13 ) ;
        salvador = input( 'Digite o nome a ser salvo: ' , 's' ) ;
        saveas( gcf , salvador , 'fig' ) ;
        saveas( gcf , salvador , 'jpg' ) ;
    end

% deleta variáveis usadas acima
    clear ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% FIM DA ROTINA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```