

Note: Each individual listing represents an individual .m file, with name given in the header comments of each listing.

The following listing is referenced in Section 3.2.1 of the thesis.

```
1 function [imgOut] = drfconv(imgIn)
2 %DRFCONV convolves an MFM image with an experimentally determined dipole
3 % response function.
4 %
5 %
6 % Input: imgIn - Grayscale MFM image to transform
7 % Output: imgOut - MFM image transformed by DRF
8 %
9 % drfconv.m
10 % Terry Ferrett
11 % 2/12/2008
12
13
14
15
16 % Read input image and cast to double.
17 imgIn = double(imgIn);
18 [imRows imCols] = size(imgIn);
19
20 % if image size is not even, chop it.
21 if mod(imRows,2) == 1,
22     imgIn = imgIn(1:imRows - 1, 1:imCols - 1);
23 end
24
25 % Window the image
26 w_col = ones(imRows);
27
28 % The function kbw generates a Kaiser-Bessel window in a row vector with
29 % the first parameter being the length, and the second parameter being
30 % the alpha value.
31 w_temp = kbw(imRows, 1)';
32
33 for k = 1:imCols,
34     w_col(:,k) = w_col(:,k).*w_temp;
35 end
36 w_row = w_col';
37 window = w_row.*w_col;
38
```

```

39 imgIn = imgIn.*window;
40
41 % Get microns per pixel.
42 um_pp = 90/imRows;
43
44 % length of dipole response function.
45 len = round(50/um_pp);
46
47 % Vector, in microns, of the dipole response function.
48 vec = 0:len;
49 vec = vec*um_pp;
50 vec = vec - vec(end)/2;
51
52
53 % Assume the dipole response function is about 4 um wide.
54 [X,Y] = meshgrid(vec,vec);
55 Z = zeros(length(X),length(X));
56
57 % Create normally distributed cone.
58 z =1*exp((-X.^2 - Y.^2)*2);
59
60
61 % zero padding stuff
62 [rows_B cols_B] = size(imgIn);
63 [rows_px cols_px] = size(z);
64 offr = rows_B + rows_px -1;
65 offc = cols_B + cols_px -1;
66
67
68 % FFT of DRF
69 zp = zeros(offr,offc);
70 zp(1:rows_px, 1:cols_px) = z;
71 zp = zp;
72
73 F_z = fft2(zp,offr,offc);
74
75 % Remove zeros from the FFT of the DRF.
76 F_z = F_z/max(max(abs(F_z)));
77 angles = angle(F_z);
78 F_z = F_z + (cos(angles) + j*sin(angles));
79
80
81 F_img = fft2(imgIn, offr, offc);
82 num1 = max(max(abs(F_img)));
83 F_img = F_img/num1;

```

```

84
85
86 deconvFFT = F_img./conj(F_z);
87
88
89 imgOut = num1*abs(ifft2(deconvFFT));
90
91 % Reverse window the image.
92 inrow = round((offr - imRows)/2);
93 incol = round((offc - imCols)/2);
94 imgOut = imgOut(inrow:end-inrow, incol:end-incol);
95 [a b] = size(imgOut);
96 [c d] = size(window);
97 imgOut = imgOut(1:c, 1:d);
98
99 imgOut = imgOut./window;
100
101
102 % Truncate the edge.
103 in = 10;
104 imgOut = imgOut(in:end-in, in:end-in);
105
106 % Rotate and transpose the image.
107 imgOut = imrotate(imgOut,-0);
108
109 % Map the image into 0-255.
110 imgMapped = imgOut./max(max(imgOut))*255;
111 imgOut = round(imgMapped);
112 imgOut = uint8(imgOut);

```

The following listing is referenced in Section 3.2.2 of the thesis.

```

1 % magnets_3d.oo.m
2 % Terry Ferrett
3 % 2/17/2006
4 %
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 % This function generates a three dimensional lattice of randomly
7 % oriented magnetic dipoles.
8 %
9 % Syntax:
10 % magnets_3d.oo

```

```

11 %
12 % This function calls the function newmag()
13 %
14 % Outputs: Field map plots.%
15 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16
17
18 clear all;
19 close all;
20
21 % Initialize the field.
22 x1 = -10;
23 x2 = 10;
24
25 y1 = -10;
26 y2 = 10;
27
28 z1 = -1;
29 z2 = 1;
30
31 x = x1:0.2:x2;
32 y = y1:0.2:y2;
33 z = z1:0.2:z2;
34
35 [X,Y,Z] = meshgrid(x,y,z);
36
37 [dim_x dim_y dim_z] = size(X);
38
39 x_field = zeros(dim_x,dim_y,dim_z);
40 y_field = zeros(dim_x,dim_y,dim_z);
41 z_field = zeros(dim_x,dim_y,dim_z);
42
43
44 % Magnet orientation
45 orientation = [1 0];
46
47
48 % Specify variance for random pitch and yaw.
49 var = 200;
50 var_center = 0.6^2;
51
52
53 % Place some magnets.
54 for k = x1:1.5:x2,
55     for l = y1:1.5:y2,

```

```

56         for m = z1:1.5:z2,
57             ore = round(rand);
58             orientation = circshift(orientation,[ore 1-ore]);
59             p_rand = sqrt(var)*randn; % Random pitch.
60             y_rand = sqrt(var)*randn; % Random yaw.
61
62
63             k_rand = sqrt(var_center); % Random x-y-z offsets
64             l_rand = sqrt(var_center);
65             m_rand = sqrt(var_center);
66             [out, x_field, y_field, z_field] = newmag3d([(l+l_rand)....
67                 (k+k_rand) (m+m_rand)], orientation(1)*180 + p_rand, ...
68                 y_rand, x_field, y_field, z_field);
69         end
70     end
71 end
72
73 % create MFM image.
74 % Expand out Bz and Bx to create a y vector at z = 0.50 um. 90umx90um
75 % image.
76
77 height = 7;
78 Bx = x_field(:,:,height);
79 By = y_field(:,:,height);
80 Bz = z_field(:,:,height);
81
82
83
84
85 % Create two-dimensional grid of x and y values.
86 dx = 0.2;
87 dy = 0.2;
88
89
90 % Assume the dipole response function is about 600nm wide.
91 [X,Y] = meshgrid(-0.3:dx:0.3,-0.3:dy:0.3);
92 Z = zeros(length(X),length(X));
93
94
95 % Create normally distributed cone.
96 z = exp((-X.^2 - Y.^2)*40);
97 med_d = median(1:length(z));
98 wid_d = med_d - 1;
99
100

```

```

101 % Calculate gradient of z
102 [px,py] = gradient(z);
103 px(px == 0) = 0.001;
104 py(py == 0) = 0.001;
105
106 pz = exp((-X + px).^2 - (Y + py).^2)*5) - exp((-X.^2 - Y.^2)*5);
107 pz = -pz;
108
109 % Determine the amount of zero padding to apply to the 2d FFT.
110 [rows_B cols_B] = size(Bx);
111 [rows_px cols_px] = size(px);
112 offr = rows_B + rows_px -1;
113 offc = cols_B + cols_px -1;
114
115
116
117 % Perform 2d fft on the magnetization patterns.
118 F_Bx = fft2(Bx, offr, offc);
119 F_By = fft2(By, offr, offc);
120 F_Bz = fft2(Bz, offr, offc);
121
122 % Perform 2d fft on the sample magnetization.
123 F_px = fft2(px, offr, offc);
124 F_py = fft2(py, offr, offc);
125 F_pz = fft2(pz, offr, offc);
126
127 % Perform convolution in the frequency domain.
128 F_img = F_Bx.*conj(F_px) + F_By.*conj(F_py) + F_Bz.*conj(F_pz);
129
130 % Deconvolve to find the simulated MFM image.
131 img = real(ifft2(F_img));
132
133
134 % plot the simulated MFM image.
135 T = (0:length(img))*20/length(img);
136 U = (0:length(img))*20/length(img);
137 figure;
138 colormap(gray);
139 imagesc(T,U,img);
140 set(gca,'YDir','normal');
141 xlabel('\mum','Interpreter','tex','FontSize',20);
142 ylabel('\mum','Interpreter','tex','FontSize',20);
143 %title('Simulated MFM Image - Opposite Orientation');
144
145

```

```

146 x = x1:0.2:x2;
147 y = y1:0.2:y2;
148 z = z1:0.2:z2;
149
150
151 figure
152 [X,Y,Z] = meshgrid(x,y,z);
153 %height = 15;
154 quiver(X(:,:,height) + x2,Y(:,:,height) + y2,x_field(:,:,height),...
155         y_field(:,:,height));
156 set(gca,'YDir','normal');
157 xlabel('\mum','Interpreter','tex','FontSize',20);
158 ylabel('\mum','Interpreter','tex','FontSize',20);
159 axis([0 30 0 30])
160
161 figure
162 [X,Y,Z] = meshgrid(x,y,z);
163 quiver(X(:,:,height) + x2,Y(:,:,height) + y2,x_field(:,:,height),...
164         y_field(:,:,height));
165 set(gca,'YDir','normal');
166 xlabel('\mum','Interpreter','tex','FontSize',20);
167 ylabel('\mum','Interpreter','tex','FontSize',20);
168 axis([0 15 0 15])
169
170
171 figure
172 [X,Y,Z] = meshgrid(x,y,z);
173 quiver(X(:,:,height) + x2,Y(:,:,height) + y2,x_field(:,:,height),...
174         y_field(:,:,height));
175 set(gca,'YDir','normal');
176 xlabel('\mum','Interpreter','tex','FontSize',20);
177 ylabel('\mum','Interpreter','tex','FontSize',20);
178 axis([0 5 0 5])

1 % newmag.m
2 % Terry Ferrett
3 % 2/14/2006
4 %
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6 % This function creates a new magnet and adds it to the
7 % specified field.
8 %
9 % Syntax:
10 % out = newmag(center, degrees, strength, x_field, y_field)

```

```

11 %
12 % Inputs:
13 %   center: x-y coordinate of the center of the magnet.
14 %   degrees: rotation of magnet, 0-359
15 %   strength: B-field strength.
16 %   x-field, y-field: components of x and y fields to which
17 %       to add the magnet.
18 %
19 % Outputs:
20 %   out: return code, 0 for success and 1 for failure.
21 %   x-field: new x-vectors for the tape.
22 %   y-field: new y-vectors for the tape.
23 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
24
25 function [out, x-field, y-field, z-field] = newmag(center, pitch, yaw, ...
26     x-field, y-field, z-field);
27
28
29 % Perform some error checking.
30 if(pitch < -360 || pitch > 360)
31     fprintf('Pitch out of range. Please enter a value between 0-360.\n');
32     out = 1;
33     return;
34 end
35
36 if(yaw < -360 || yaw > 360)
37     fprintf('Yaw out of range. Please enter a value between 0-360.\n');
38     out = 1;
39     return;
40 end
41
42 if((center(1) < -45 || center(1) > 45) || (center(2) < -45 || ...
43     center(2) > 45) || (center(3) < -5 || center(3) > 5))
44     fprintf('Center out of range.\n');
45     out = 1;
46     return;
47 end
48
49 % Generate a new field for this magnet.
50 x = -10:0.2:10;
51 y = -10:0.2:10;
52 z = -1:0.2:1;
53 [X,Y,Z] = meshgrid(x,y,z);
54
55 Xs = X;

```



```

56 Ys = Y;
57 Zs = Z;
58
59 % Fixed magnet width, 0.5 um.
60 width.mag = 0.5;
61
62
63 % Apply rotation, first in the x-z plane (pitch), then in the x-y plane
64 % (yaw).
65 x_off_temp = width.mag/2*cos(pitch*pi/180);
66 z_off = width.mag/2*sin(pitch*pi/180);
67 x_off = x_off_temp*cos(yaw*pi/180);
68 y_off = x_off_temp*sin(yaw*pi/180);
69
70 center(1) = center(1);
71 center(2) = center(2);
72 center(3) = center(3);
73
74 % Generate north pole.
75 len1 = sqrt((Xs-x_off-center(1)).^2 + (Ys-y_off-center(2)).^2 + ...
76           (Zs-z_off-center(3)).^2);
77 X1 = (Xs-x_off-center(1))./len1.^3;
78 Y1 = (Ys-y_off-center(2))./len1.^3;
79 Z1 = (Zs-z_off-center(3))./len1.^3;
80
81
82 % Generate south pole.
83 len2 = sqrt((Xs+x_off-center(1)).^2 + (Ys+y_off-center(2)).^2 + ...
84           (Zs+z_off-center(3)).^2);
85 X2 = -(Xs+x_off-center(1))./len2.^3;
86 Y2 = -(Ys+y_off-center(2))./len2.^3;
87 Z2 = -(Zs+z_off-center(3))./len2.^3;
88
89 % Add this magnet to the field.
90 x_field = x_field + X1 + X2;
91 y_field = y_field + Y1 + Y2;
92 z_field = z_field + Z1 + Z2;
93
94 % Return success code.
95 out = 0;

```