Magnetostatics Project

Von Kaukeano
Tuh42003@temple.edu

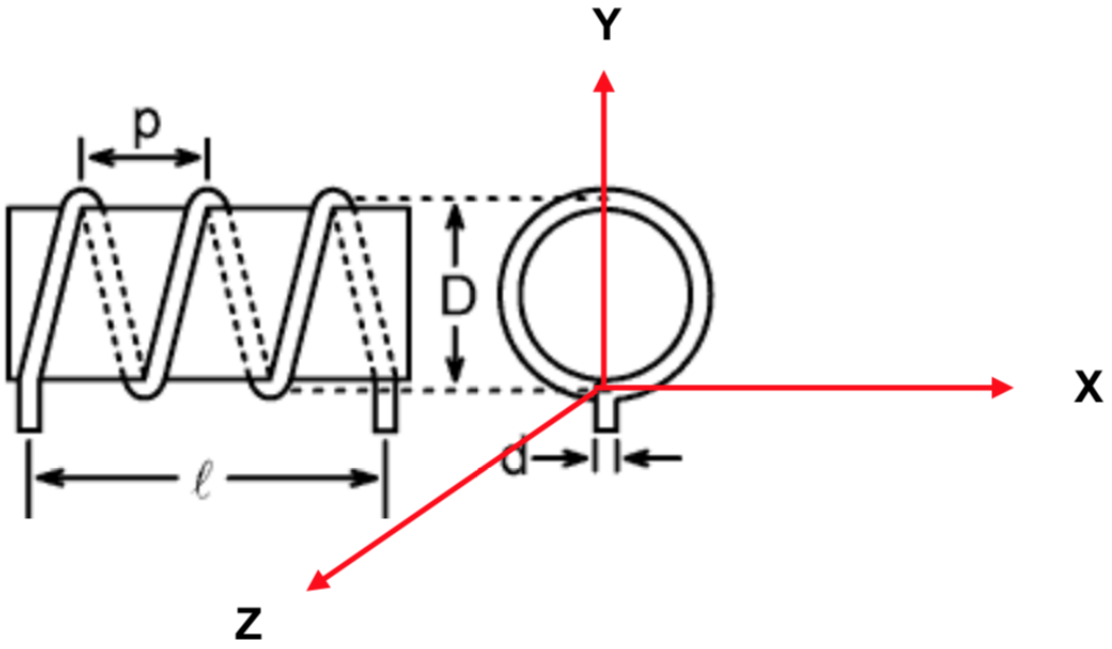

Table of Contents

Table of Figures

Summary
This magneto statics project used the discrete summation solution in Matlab to determine the resulting H at an arbitrary point P(X, Y, Z) for the solenoid. The formula is derived from the Biot-Savart Law and can be expressed as the summation of the current times the cross product of the normal vector and delta L divided by four times pi times the Radius squared[1]. The observation points used were $-2.25 \leq x \leq 2.25$, $-66 \leq z \leq 55$, $2.25 \leq y \leq 6.75$, x as the variable at the line $-2.5 \leq x \leq 2.5$, $y = 4.5$, and $z = 5.5$ and z as the variable at the line $66 \leq z \leq 55$, $y=4.5$, and $x=0$. The delta L was coded infinitesimally small so that it did not affect the H field being that it is an arc of a circle. To determine that our code was correct were removed the pitch of the solenoid and found the resultant vector was only in the z direction like the ring of charge which the textbook has an example to compare to. It was determined that it followed the right hand rule, and the pitch altered calculations and the direction in which the H field pointed. The helix geometry produces an H field that is nonsymmetrical like the classic model of the solenoid.

1. $\left(H = \frac{\sum I * \Delta L \; X \; \alpha_R}{4 * \pi * R^2}\right)$

Introduction

The discrete summation of the Biot-Savart Law is being used for determining of the H field of the solenoid. The current is calculated by the last three digits modulo twelve plus six which became thirteen. The number of N turns was calculated similarly, but was the last two digits modulo eight plus three which became six. The diameter and length of the solenoid was determined by the month and day of our birthday (9/11) in centimeters. The pitch is simply the length of the solenoid divided by the number of turns. Then using the Wikipedia page given, we found formulas that calculated the geometrical points for the helix of the given solenoid. Once the solenoid was created on Matlab, we created the observation point matrix and two instances of x and z as the variable. The H field can then be determined with this information.

Discussion

This project began by determining our values for the summation as discussed in the introduction. After they were calculated, the points of the solenoid were determined by using the geometrical formulas for a helix. We used the Wikipedia page given.

```
% Birthday: 09/11/1993
% TUID: 915596703

% N = 03 modulo 8 = 3 +3 = 6
% I = 703 modulo 12 = 7 + 6 = 13
dia = 9; %cm
len = 11; %cm
N_turns = 6;
current = 13; %ma
data_points_per_turn = 100;
pitch = len/N_turns;

%% create all points of solenoid
t=0:pi/data_points_per_turn:((2*pi)*N_turns);
r = (dia/2);            % radius
x = (dia/2) * sin(t);
y = (dia/2) * cos(t);
z = pitch/(2*pi) * t;
coil pos matrix = [x' y' z'];
```

Figure 1:Matlab Code to Produce Helix

After the solenoid figure was plotted we created the observation point matrix which was based upon "$-D/4 \leq x \leq D/4$, $-6l \leq z \leq 5l$, $D/4 \leq y \leq 3D/4$". The observation points are where we are observing the H field from. The exact points are $-2.25 \leq x \leq 2.25$, $-66 \leq z \leq 55$, $2.25 \leq y \leq 6.75$ all in centimeters. Then with the observation points, we created two instances where x and z were both the variable and found their corresponding H fields. When x was the variable, $-D/4 \leq x \leq D/4$ and fixed $y = D/2$ and $z = -l/2$ which corresponds to the line $-2.5 \leq x \leq 2.5$, $y = 4.5$, and $z = 5.5$ in centimeters. When z was the variable, $-6l \leq z \leq 5l$ and y=D/2 and x=0 which corresponds to the line $-66 \leq z \leq 55$, y=4.5, and x=0.

```
%% create observation point matrix
observation_point_matrix = [(0.*coil_pos_matrix(:,1))
(0.*coil_pos_matrix(:,2)) coil_pos_matrix(:,3)];
z= linspace((-6*len),(5*len),(data_points_per_turn*2*N_turns)+1);
x= linspace((-1*(dia/4)),(dia/4),(data_points_per_turn*2*N_turns)+1);
y= linspace((-1*(dia/4)),(3*(dia/4)),(data_points_per_turn*2*N_turns)+1);
```

Figure 3: Observation Points

```
observation_point_matrix(:,3) = z';
observation_point_matrix(:,2) = (dia/2);
observation_point_matrix(:,1) = 0;
```

Figure 2: Z Variable

```
observation_point_matrix(:,3) = -1*(len/2);
observation_point_matrix(:,2) = dia/2;
observation_point_matrix(:,1) = x';
```

Figure 4: X Variable

```
%% perform calculations
[row col] = size(coil_pos_matrix);
H_X_Y_Z_NORM = [0 0 0 0];

delta_L_X = pdist2(coil_pos_matrix(1,1), coil_pos_matrix(2,1)); % length in X
delta_L_Y = pdist2(coil_pos_matrix(1,2), coil_pos_matrix(2,2)); % length in Y
delta_L_Z = pdist2(coil_pos_matrix(1,3), coil_pos_matrix(2,3)); % length in Z
delta_L_matrix = [(delta_L_X.*(coil_pos_matrix(:,1).^0))
(delta_L_Y.*(coil_pos_matrix(:,1).^0)) (delta_L_Z.*(coil_pos_matrix(:,1).^0))];

for i = 1:1:row
Radius = pdist2(coil_pos_matrix, observation_point_matrix(i,:));

%distance in X
Radius_X = pdist2(coil_pos_matrix(:,1), observation_point_matrix(i,1));
%distance in Y
Radius_Y = pdist2(coil_pos_matrix(:,2), observation_point_matrix(i,2));
%distance in Z
Radius_Z = pdist2(coil_pos_matrix(:,3), observation_point_matrix(i,3));

Radius_matrix =[Radius_X Radius_Y Radius_Z] ; % vector origin points
A_r = Radius_matrix./Radius;

cross_product = cross(delta_L_matrix, A_r);

H_vector_components = (current.*cross_product)./(4*pi*(Radius(1,1)^2));
% calculate H vector components

H_X_vect = sum(H_vector_components(:,1)); % norm of H
H_Y_vect = sum(H_vector_components(:,2));
H_Z_vect = sum(H_vector_components(:,3));

H_vect = sqrt(H_X_vect^2+H_Y_vect^2+H_Z_vect^2);

H_X_Y_Z_NORM = [H_X_Y_Z_NORM; H_X_vect H_Y_vect H_Z_vect H_vect];
end
```

Figure 5: Calculations of H

The calculations of H consisted of the summation of all the x, y, and z vector components added up. The magnitude was calculated along with the normal vector in order to have the arguments to use for the quiver3 plot used to plot the H field. The delta L was selected to be infinitesimally small by subtracting our data's points from the next point along the solenoid. By changing the amount of data points per turn, we change the magnitude of delta L as well.

```
%% plot of coil with H vectors
figure(1); clf;
subplot(1,2,1);
hold on
scatter3(coil_pos_matrix(:,1),coil_pos_matrix(:,2),coil_pos_matrix(:,
3), 'filled')

scatter3(observation_point_matrix(:,1),observation_point_matrix(:,2),
observation_point_matrix(:,3), 'filled')

xlabel('x')
ylabel('y')
zlabel('z as length of spring')
title('X as a variable for axis of observation')
hold off

subplot(1,2,2);
hold on
scatter3(coil_pos_matrix(:,1),coil_pos_matrix(:,2),coil_pos_matrix(:,
3), 'filled')
q = quiver3(observation_point_matrix(:,1),
observation_point_matrix(:,2), observation_point_matrix(:,3),
H_X_Y_Z_NORM(:,1), H_X_Y_Z_NORM(:,2), H_X_Y_Z_NORM(:,3), 1e1);
xlabel('x')
ylabel('y')

q.ShowArrowHead = 'off';
zlabel('z as length of spring')
hold off
title('X as a variable with resulting magnetic field')
```

Figure 6: Plotting of H Field

The following plots are x and z as variables as delta L changes by an order of three magnitudes. When changing the data points per turn, we change the number of points to plot the solenoid and the delta magnitude changes. The graph visually describes how as you increase the number of points with a smaller delta L, the H field becomes more finite. The number of turns at 1000 points shows that the H field stays within the solenoid. The direction is always changing due to the nature of the pitch. The plots of the x variable show a weak or lower H field compared to the z variable because of the distance from the solenoid. This code is written dynamically that takes global variables as input and finds the H field relative to the data. For example, changing variable D and L would change the pitch but also change the observations points. This changes the need for flagging out of range points.

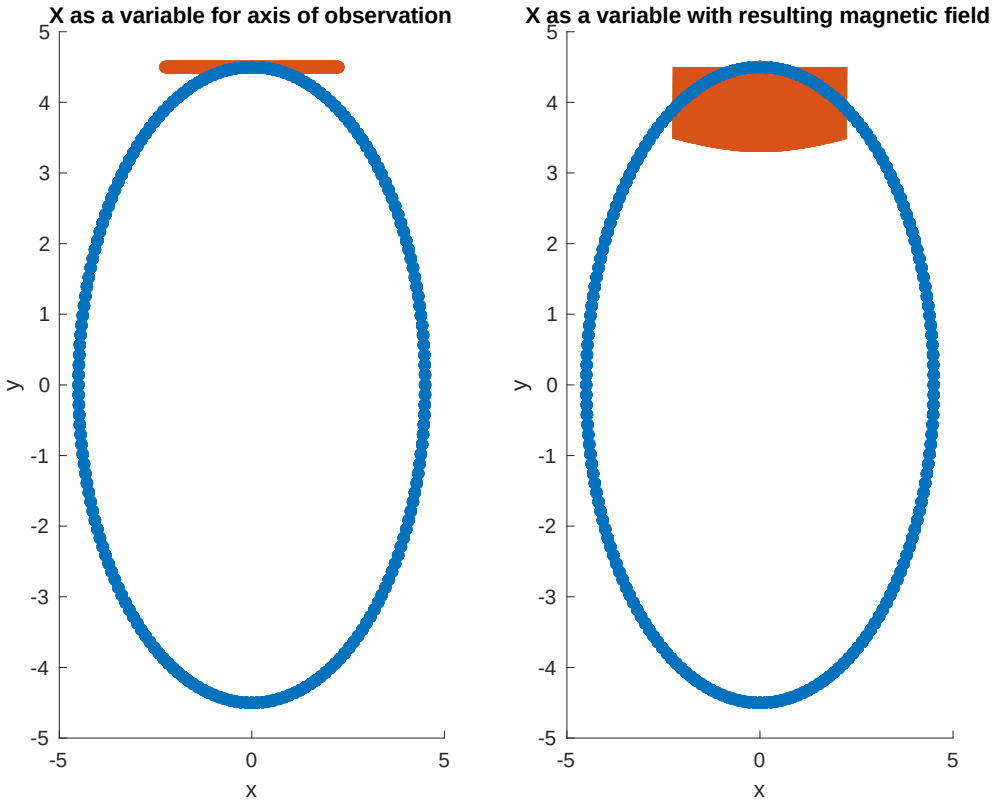

Figure 7: Plot of Solenoid with X Variable and Corresponding H Field

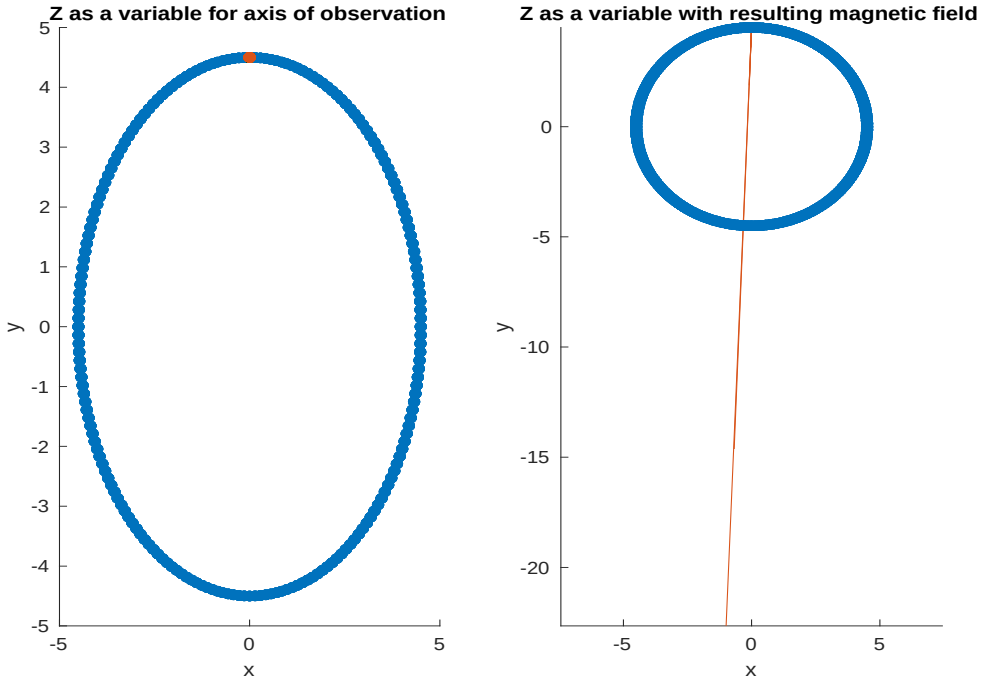

Figure 8: Plot of Solenoid as Z Variable and Corresponding H Field

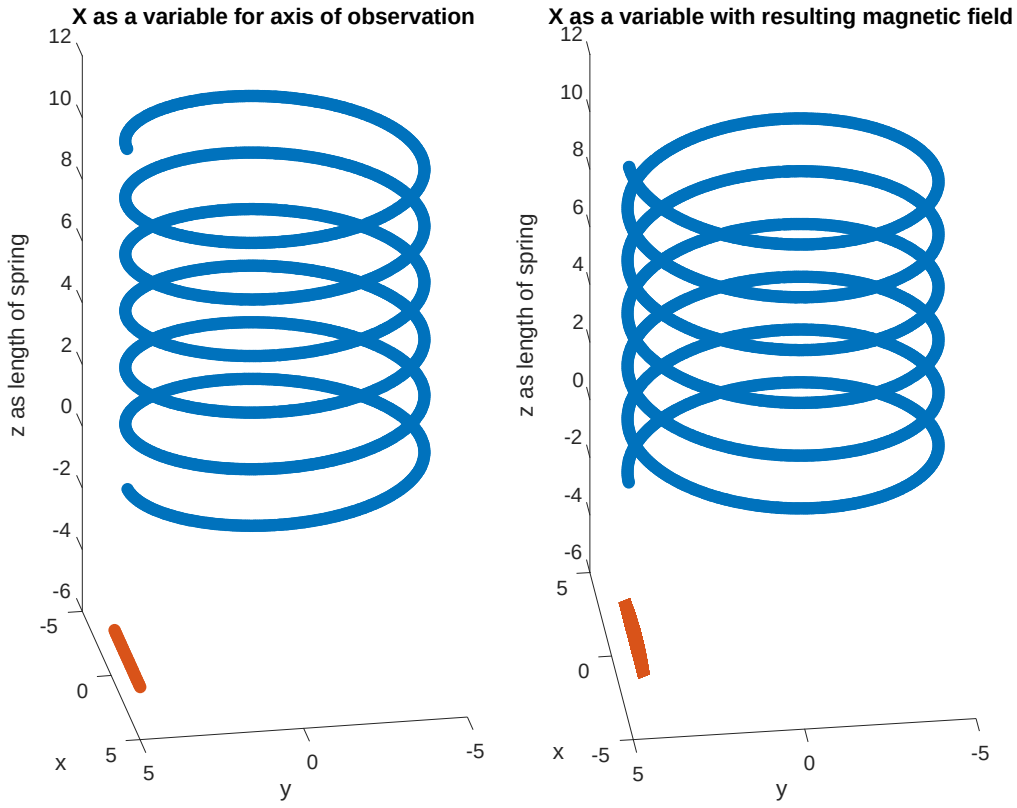

Figure 9: 1000 Data Points Per Turn: X Variable

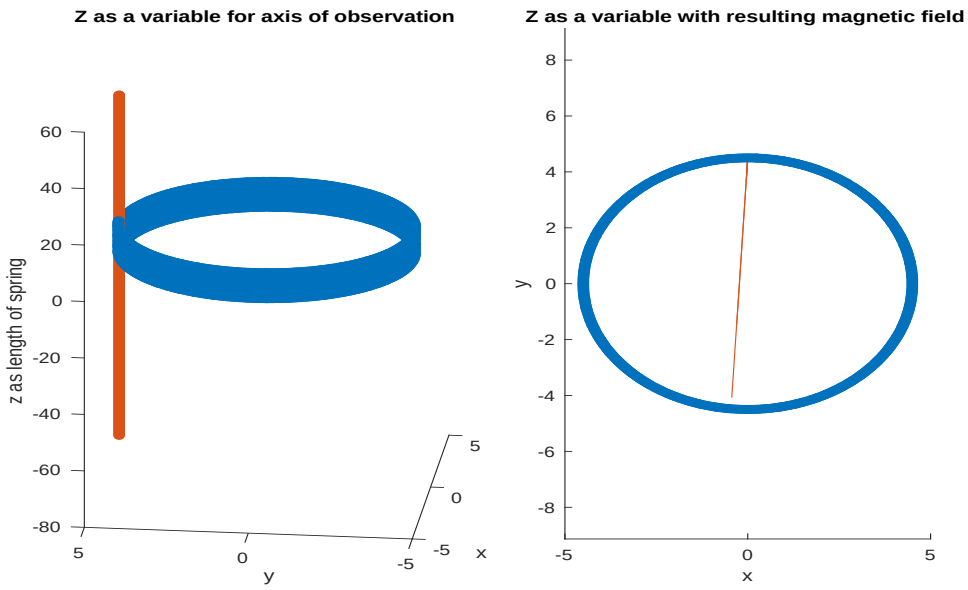

Figure 10: 1000 Data Points Per Turn: Z Variable

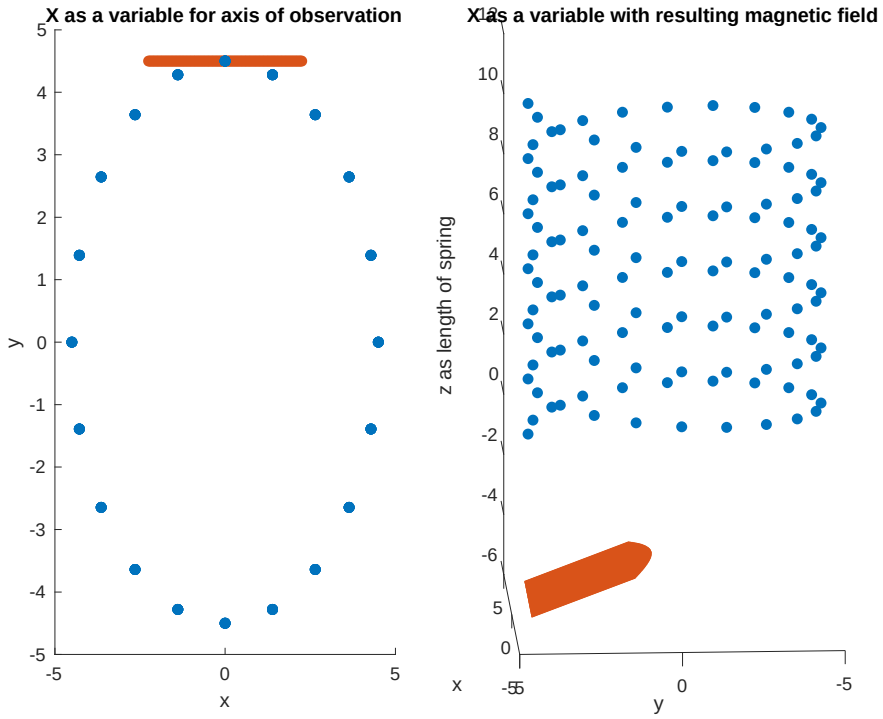

Figure 11: 10 Data Points Per Turn: X Variable

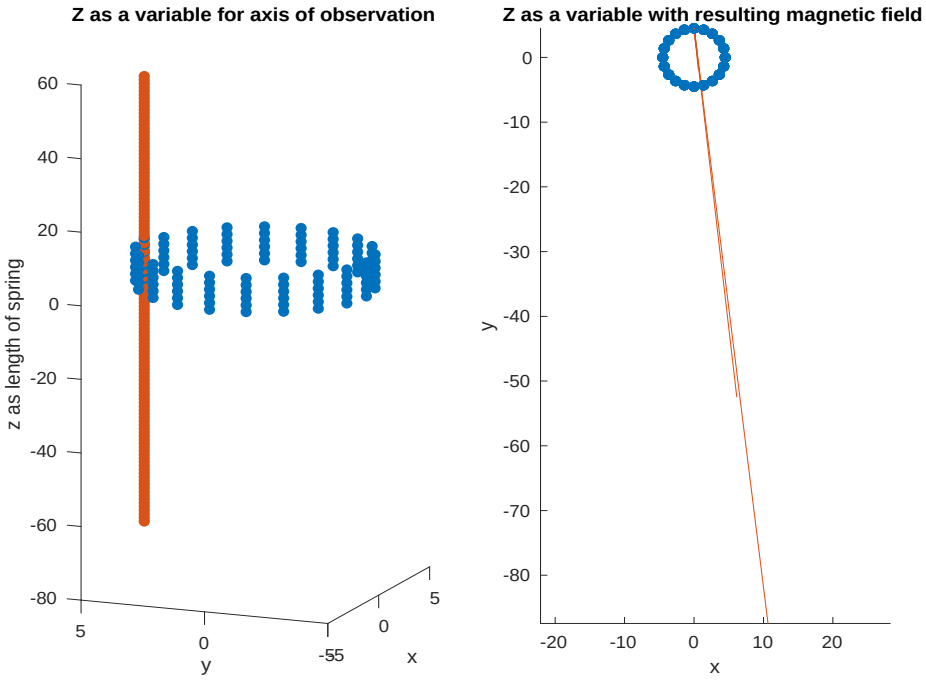

Figure 12: 10 Data Points Per Turn: Z Variable

Conclusion

The outcome of this project was successful and I feel very knowledgeable. Determining real world solutions is interesting because they have different effects and outcomes compared to book solutions. If we were to take in the effect of the size of the wire, then these solutions to the H field would most certainly be changed. A great way to check if the solution is reasonable is to change the pitch to zero and compare the textbook theory of a ring of charge and see if the H field follows the right hand rule.

Appendix

```
clear;
clc;

% Birthday: 09/11/1993
% TUID: 915596703

% N = 03 modulo 8 = 3 +3 = 6
% I = 703 modulo 12 = 7 + 6 = 13
dia = 9; %cm
len = 11; %cm
N_turns = 6;
current = 13; %ma
data_points_per_turn = 100;
pitch = len/N_turns;

%% create all points of coil
t=0:pi/data_points_per_turn:((2*pi)*N_turns);
r = (dia/2);            % radius
x = (dia/2) * sin(t);
y = (dia/2) * cos(t);
z = pitch/(2*pi) * t;
coil_pos_matrix = [x' y' z'];

%% create observation point matrix
observation_point_matrix = [(0.*coil_pos_matrix(:,1))
(0.*coil_pos_matrix(:,2)) coil_pos_matrix(:,3)];
z= linspace((-6*len),(5*len),(data_points_per_turn*2*N_turns)+1);
x= linspace((-1*(dia/4)),(dia/4),(data_points_per_turn*2*N_turns)+1);
y= linspace((-
1*(dia/4)),(3*(dia/4)),(data_points_per_turn*2*N_turns)+1);

observation_point_matrix(:,3) = -1*(len/2);
observation_point_matrix(:,2) = dia/2;
observation_point_matrix(:,1) = x';

%% perform calculations
[row col] = size(coil_pos_matrix);
H_X_Y_Z_NORM = [0 0 0 0];
delta_L_X = pdist2(coil_pos_matrix(1,1), coil_pos_matrix(2,1)); %
length in X
delta_L_Y = pdist2(coil_pos_matrix(1,2), coil_pos_matrix(2,2)); %
length in Y
delta_L_Z = pdist2(coil_pos_matrix(1,3), coil_pos_matrix(2,3)); %
length in Z
delta_L_matrix = [(delta_L_X.*(coil_pos_matrix(:,1).^0))
(delta_L_Y.*(coil_pos_matrix(:,1).^0))
(delta_L_Z.*(coil_pos_matrix(:,1).^0))];

for i = 1:1:row
Radius = pdist2(coil_pos_matrix, observation_point_matrix(i,:));

%distance in X
Radius_X = pdist2(coil_pos_matrix(:,1),
```

```
%distance in Y
Radius_Y = pdist2(coil_pos_matrix(:,2),
observation_point_matrix(i,2));
%distance in Z
Radius_Z = pdist2(coil_pos_matrix(:,3),
observation_point_matrix(i,3));

Radius_matrix =[Radius_X Radius_Y Radius_Z] ; % vector origin points
A_r = Radius_matrix./Radius;

cross_product = cross(delta_L_matrix, A_r);

H_vector_components =
(current.*cross_product)./(4*pi*(Radius(1,1)^2));
% calculate H vector components

H_X_vect = sum(H_vector_components(:,1)); % norm of H
H_Y_vect = sum(H_vector_components(:,2));
H_Z_vect = sum(H_vector_components(:,3));

H_vect = sqrt(H_X_vect^2+H_Y_vect^2+H_Z_vect^2);

H_X_Y_Z_NORM = [H_X_Y_Z_NORM; H_X_vect H_Y_vect H_Z_vect H_vect];
end
H_X_Y_Z_NORM(1, :) = [];
%% plot of coil with H vectors
figure(1); clf;
subplot(1,2,1);
hold on
scatter3(coil_pos_matrix(:,1),coil_pos_matrix(:,2),coil_pos_matrix(:,
3), 'filled')
scatter3(observation_point_matrix(:,1),observation_point_matrix(:,2),
observation_point_matrix(:,3), 'filled')
xlabel('x')
ylabel('y')
zlabel('z as length of spring')
title('X as a variable for axis of observation')
hold off
subplot(1,2,2);
hold on
scatter3(coil_pos_matrix(:,1),coil_pos_matrix(:,2),coil_pos_matrix(:,
3), 'filled')
q = quiver3(observation_point_matrix(:,1),
observation_point_matrix(:,2), observation_point_matrix(:,3),
H_X_Y_Z_NORM(:,1), H_X_Y_Z_NORM(:,2), H_X_Y_Z_NORM(:,3), 1e1);
xlabel('x')
ylabel('y')

q.ShowArrowHead = 'off';
zlabel('z as length of spring')
hold off
title('X as a variable with resulting magnetic field')
```

Figure 13: Entire MATLAB Code