



Instituto Tecnológico y de Estudios Superiores de Monterrey

**Modelación de sistemas electromagnéticos**

**F1014B – Grupo 302**

**Profesores:**

Mario Iván Estrada Delgado

Francisco Javier Delgado Cepeda

**“Campo Magnético Terrestre”**

**Segundo Entregable Reto**

Equipo 3



A01752953  
Frida Cano Falcón



A01750150  
Hortencia Alejandra Ramírez Vázquez



A01752789  
Luis Humberto Romero Pérez



A01752785  
David Damián Galán



A01753328  
Miguel Angel Juarez Dorantes

**Fecha de entrega:**  
Viernes 4 de junio de 2021

"Nosotros, como integrantes de la comunidad estudiantil del Tecnológico de Monterrey, somos conscientes de que la trampa y el engaño afectan nuestra dignidad como personas, nuestro aprendizaje y nuestra formación, por ello nos comprometemos a actuar honestamente, respetar y dar crédito al valor y esfuerzo con el que se elaboran las ideas propias, las de los compañeros y de los autores, así como asumir nuestra responsabilidad en la construcción de un ambiente de aprendizaje justo y confiable".

## **Objetivos**

Para la entrega final del reto debemos entregar un programa capaz de simular el campo magnético de la Tierra y el comportamiento de partículas provenientes del sol al entrar en contacto con este además y presentar el trabajo realizado. Considerando los puntos anteriores y continuando con la metodología de trabajo planteada para la primera entrega, nos planteamos objetivos para esta entrega final. Dichos objetivos son los siguientes:

- Codificar funciones capaces de graficar partículas y su movimiento al estar influenciadas por un campo magnético.
- Personalizar el código realizado implementando aspectos de programación orientada a objetos y personalizando la visualización de los campos magnéticos.
- Desarrollar un programa que, con ayuda de las funciones codificadas para el primer entregable, sea capaz de simular el campo magnético terrestre.
- Simular las trayectorias de partículas emitidas por el Sol y el cambio que estas sufrirán al entrar en contacto con el campo magnético terrestre.
- Modelar la trayectoria de una determinada cantidad de partículas en función de velocidades iniciales y las posiciones en las que iniciarán estas partículas.
- Preparar una presentación para exponer los conocimientos desarrollados durante el módulo y presentar nuestra simulación.

## **Introducción**

Para el desarrollo de este reto es indispensable profundizar en temas como son los campos magnéticos, más específicamente el campo magnético terrestre, para comenzar definiremos qué es un campo magnético, de acuerdo con Braun (2003) podemos decir que un campo magnético es generado por el movimiento de cargas eléctricas, este movimiento de cargas ocasiona que partículas a su alrededor y dentro de una distancia proporcional a la fuerza de la carga estén en presencia de una fuerza electromagnética instantánea. Una vez establecida una definición de campo magnético podemos decir que el campo magnético de la Tierra está generado por

movimiento de cargas, pero ¿de dónde provienen estas cargas? De acuerdo con Folguera (2006) “la composición del núcleo terrestre indica que este es un buen conductor eléctrico y su estado, el líquido, indica que este material conductor podría fluir y circular describiendo remolinos y espirales” (p. 20), estos movimientos llevan asociados cargas eléctricas las cuales inducen el campo magnético terrestre.

Una vez comprendido lo que es un campo magnético y el origen del campo magnético terrestre podemos hablar de su importancia y aplicaciones. Por un lado el campo magnético terrestre es indispensable para la vida en la Tierra, Santamaría (2016) menciona que el campo magnético terrestre es esencial para la vida en la Tierra, esto debido a que el campo magnético se extiende hasta el espacio exterior del planeta y es confinado por la acción del fuego solar a una región que llamamos magnetosfera, esta región protege a la superficie terrestre de la radiación solar. Por otro lado el campo magnético terrestre nos ha permitido estudiar el pasado de nuestro planeta con ayuda del paleomagnetismo, “esta disciplina se encarga de determinar entre otras cosas las posiciones de los continentes, o parte de estos, en el pasado” (p. 18), el campo magnético terrestre hace posible transmitir ondas electromagnéticas entre grandes distancias, de acuerdo con Braun (2003) el ingeniero italiano Guglielmo Marconi construyó y desarrolló experimentos con los cuales le fue posible transmitir ondas electromagnéticas trasatlánticas, todo esto posible gracias a que las ondas parecían ser guiadas alrededor de la superficie terrestre, no fue hasta que A. E. Kenelly y O. Heaviside plantearon la teoría de que había una capa de partículas eléctricamente cargadas en la atmósfera que reflejaban las ondas electromagnéticas, esta capa fue nombrada ionósfera por Balfour Stewart en 1882 y se convirtió en la base de las electrocomunicaciones.

Como bien ya mencionamos los campos electromagnéticos, más específicamente el terrestre es de gran importancia en muchos ámbitos de la ciencia, el desarrollo de este reto nos permitirá, entre otras cosas comprender la función del campo magnético terrestre como escudo ante la radiación solar, siendo una simulación computacional el elemento gráfico perfecto para profundizar en el tema.

## *Origen físico de los resultados*

El algoritmo implementado en MATLAB tiene sustento en varias leyes físicas, las cuales se enlistan a continuación:

- **Producto Cruz**

La operación producto cruz de dos vectores A, B, denotada como  $A \times B$ , se calcula como sigue:

$$\begin{aligned} A \times B &= \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ A_x & A_y & A_z \\ B_x & B_y & B_z \end{vmatrix} \\ &= \hat{i}(A_y B_z - A_z B_y) - \hat{j}(A_x B_z - A_z B_x) + \hat{k}(A_x B_y - A_y B_x) \end{aligned}$$

Esta es una operación que implementa el código para poder realizar el cálculo del campo magnético.

- **Ley de Biot-Savart**

La ley de Biot-Savart nos muestra la manera de calcular el campo magnético producido por una línea que mantiene una corriente eléctrica, para un punto específico en el espacio.

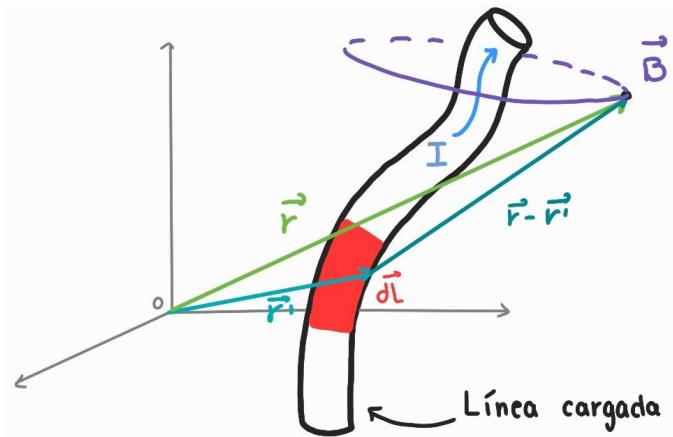
La ecuación de esta ley es

$$\vec{B} = \frac{\mu_0 I}{4\pi} \int_L \frac{\vec{dl} \times (\vec{r} - \vec{r}')}{|\vec{r} - \vec{r}'|^3}$$

, donde  $\mu_0$  es la constante de permeabilidad del vacío, con valor de  $4\pi \times 10^{-7}$ , I

corresponde a la intensidad de corriente eléctrica,  $\vec{dl}$  corresponde a la magnitud vectorial diferencial de línea,  $\vec{r}$  es el vector que apunta del origen al punto donde se pretende calcular el campo, y  $\vec{r}'$  es el vector que apunta del origen a la cola del vector  $\vec{dl}$ . De esta manera, la operación  $\vec{r} - \vec{r}'$  nos permite obtener el vector que apunta de la cola del vector  $\vec{dl}$  al punto donde se pretende calcular el campo. La ley de Biot-Savart nos muestra que la intensidad del campo magnético decae con el cuadrado de la distancia, ya que podemos cancelar un  $\vec{r} - \vec{r}'$  y sustituirlo por el vector unitario  $(\widehat{\vec{r} - \vec{r}'})$ . También nos permite conocer la dirección del campo, que será perpendicular al vector dirección en la que fluye la corriente y al vector distancia al punto en el que se pretende calcular el campo, debido

a la operación producto cruz.



**Figura 1.** Campo magnético de un elemento de corriente.

- **Integración de una función vectorial**

Cuando se tiene una función vectorial, la integral de dicha función queda como el vector de las integrales simples de sus componentes (Stewart, 2017). Si  $\vec{r}(t)$  es una función vectorial con componentes  $< f(t), g(t), h(t) >$ , entonces se cumple:

$$\int \vec{r}(t) dt = < \int f(t) dt, \int g(t) dt, \int h(t) dt >$$

Este resultado es importante porque es la base de la manera en la que se está realizando la integración: después de obtener el vector resultante del producto cruz para la ley de Biot-Savart, obtenemos las ecuaciones paramétricas para cada componente por separado y posteriormente se integran numéricamente.

- **Fuerza de Lorentz**

De acuerdo con Campbell (2019) la fuerza de Lorentz es “la influencia que produce un cambio en una cantidad física. Esta es la fuerza electromagnética  $F$  en una partícula cargada  $q$  con una velocidad  $v$  a través de un campo eléctrico  $E$  y un campo magnético  $B”$

$$F = qE + qv \times B.$$

La fuerza magnética es perpendicular a la velocidad y al campo magnético.

- **Euler**

El método de Euler es un método numérico que nos permite aproximar una solución de una ecuación diferencial con valor inicial:

$$y' = f(x, y) \quad y(x_0) = y_0$$

Es por medio de la recta tangente que pasa por ese punto, es decir vamos a linealizar el problema acercándonos con una recta que pase por un punto dado y que sea tangente a la curva en dicho punto.

La pendiente es igual a  $y' = f(x_0, y_0)$ ; la recta tangente y la curva pasan por el punto  $(x_0, y_0)$  entonces:

$$y_0 = f(x_0, y_0)x_0 + b \Rightarrow b = y_0 - f(x_0, y_0)x_0$$

Así pues la ecuación de la recta tangente es:

$$y = f(x_0, y_0)x + y_0 - f(x_0, y_0)x_0$$

$$y = f(x_0, y_0)(x - x_0) + y_0$$

Entonces:

$$y = y(x_0 + h) \Rightarrow y'(x_0, y_0)(x + h - x_0) + y_0 \Rightarrow y'(x_0, y_0)h + y_0$$

Para encontrar las soluciones aproximadas se pueden usar las siguientes ecuaciones de recurrencia:

$$x_{n+1} = x_0 + nh \quad y_{n+1} = f(x_i, y_i)h + y_n$$

- **Euler mejorado**

Este método se basa en la misma idea del método de Euler, pero este hace un refinamiento en la aproximación, tomando un promedio entre ciertas pendientes:

$$y_{n+1} = y_n + h \frac{f(x_n, y_n) + f(x_{n'}, y_{n'})}{2}$$

## Datos considerados en la simulación

**Radio de la tierra:** Este dato fue indispensable para calcular la aproximación del campo magnético terrestre, con ayuda de nuestra función de campo magnético nos fue posible calcular una aproximación del campo magnético generado por la tierra usando la medida del radio terrestre el cual es de 6371 km.

**Campo magnético de la tierra:** De acuerdo con McLean (s.f.), la intensidad del campo magnético terrestre se encuentra entre los  $2.5 \times 10^{-5}$  y  $6.5 \times 10^{-5}$  teslas, tomando como referencia estas mediciones pudimos corroborar que nuestra función para calcular el campo magnético fuese correcta ya que, el valor obtenido de esta fue de  $5.66 \times 10^{-5}$  T en los polos, estando este entre los valores que investigamos.

**Partículas emitidas por el sol:** Para escoger las partículas con las que estaríamos trabajando investigamos acerca del viento solar y sus componentes, el viento solar es “una nube de gas ionizado que es expulsado de manera constante por el sol a altas velocidades y en todas direcciones, está compuesto principalmente por electrones, protones, partículas alfa y algunos iones más pesados” (Nagi, 2020). Basándonos en la definición anterior decidimos trabajar con aquellas partículas con las que estamos familiarizados, tales como los protones y electrones.

**Propiedades físicas de las partículas elegidas:** Ya que nuestro código necesita la masa, carga y velocidad de las partículas emitidas por el sol, en este caso protones y electrones nos fue necesario investigar y seleccionar los valores de dichas propiedades: en el caso del electrón la masa es de  $9.11 \times 10^{-31}$  y una carga de  $1.6 \times 10^{-19}$ ; en el caso del protón la masa es de  $1.67 \times 10^{-27}$  y tiene una carga de  $1.6 \times 10^{-19}$ ; en el caso de las velocidades encontramos que estas pueden ir de entre los  $2 \times 10^5$  cerca de atmósfera hasta los  $.9 \times 10^9$  km/s en el vacío, en el caso de las velocidades decidimos expresar las velocidades de los electrones en orden  $10^6$  y los protones  $10^9$  de esta forma podríamos apreciar diferentes comportamientos de las partículas al entrar en contacto con el campo magnético terrestre.

## Clases y Métodos

### Clase LineaCargada

Esta clase permite crear las líneas con flujo de corriente que serán usadas para el cálculo del campo magnético en la simulación. Los objetos de esta clase tienen como atributos las funciones simbólicas que representan la parametrización de la línea ( $lx, ly, lz$ ), sus derivadas ( $dlx, dly, dlz$ ), la variable de parametrización y sus rangos ( $s, min\_s, max\_s$ ), la corriente de la línea cargada ( $I$ ) y el número de particiones para el cálculo de la integral por el método de simpson ( $integral$ ). El único método de esta clase es el constructor, que recibe todos los valores y los asigna a las variables adecuadas. El constructor inicializa las derivadas por medio de la función `diff`.

```
1 classdef LineaCargada
2     properties
3         %Vector de la función paramétrica
4         s, lx, ly, lz;
5         dlx, dly, dlz;
6         %Corriente
7         I;
8         %Limites del parametro s
9         min_s, max_s;
10        integral;
11    end
12    methods
13        function linea = LineaCargada(lx, ly, lz, I, min_s, max_s, integral)
14            linea.lx = lx;
15            linea.ly = ly;
16            linea.lz = lz;
17            linea.I = I;
18            linea.min_s = min_s;
19            linea.max_s = max_s;
20
21            %Derivadas de linea parametrizada
22            linea.dlx = diff(lx);
23            linea.dly = diff(ly);
24            linea.dlz = diff(lz);
25
26            linea.integral = integral;
27        end
28    end
29 end
```

Figura 2. Programación en Matlab de la Clase *LineaCargada*.

### *Clase Vector\_*

Esta clase permite crear puntos en un espacio tridimensional, que son usados para crear la malla de puntos donde se calculará el campo magnético, además de ser usados al momento de calcular y graficar los componentes del campo magnético. Los objetos de esta clase tienen como atributos los valores de posición con respecto a los ejes x, y y z (x,y,z). Esta clase contiene únicamente el método constructor que recibe los valores adecuados y los asigna a sus parámetros, si no se ingresa ningún argumento al constructor, se asigna el valor de 0 a las variables x, y y z.

```
7  classdef Vector_
8  properties
9      x = 0; %Componente x
10     y = 0; %Componente y
11     z = 0; %Componente z
12 end
13 methods
14     %Constructor de la clase Vector_
15     function v = Vector_(x, y, z)
16         if(nargin == 0)
17             v.x = 0;
18             v.y = 0;
19             v.z = 0;
20         else
21             v.x = x;
22             v.y = y;
23             v.z = z;
24         end
25     end
```

**Figura 3.** Programación en Matlab de la Clase *Vector\_*, *atributos y constructor*.

Esta clase contiene los siguientes métodos:

- **Cruz:** Este método regresa el producto cruz de los dos objetos vectores ingresados, colocando el resultado en las componentes de un nuevo vector. Este método es llamado por los métodos de la clase partícula.

```

26 %Funcion de producto cruz entre dos objetos Vector_
27 function v = cruz(a,b)
28     v = Vector_();
29
30     v.x = ((b.z*a.y)-(b.y*a.z));
31     v.y = -((b.z*a.x)-(b.x*a.z));
32     v.z = ((b.y*a.x)-(b.x*a.y));
33 end

```

**Figura 4.** Programación en Matlab de la Clase *Vector\_*, *método cruz*.

- **Sobrecarga de operador mtimes:** Esta sobrecarga (\*) nos permite obtener un nuevo objeto vector al multiplicar un número por un objeto vector, el vector que regresa tendrá como componentes las componentes del vector original multiplicadas por el número indicado.

```

34 %Multiplicar un numero por un vector
35 function v = mtimes(entero,vec)
36     v = Vector_(entero*vec.x, entero*vec.y, entero*vec.z);
37 end

```

**Figura 5.** Programación en Matlab de la Clase *Vector\_*, *sobre carga mtimes*.

- **Sobrecarga de operador plus:** Esta sobrecarga (+) nos permite sumar dos objetos vectores, obteniendo como resultado un nuevo vector cuyas componentes son la suma de las componentes de los vectores ingresados.

```

38 %Sumar dos objetos Vector_
39 function v = plus(vec1,vec2)
40     v = Vector_(vec1.x+vec2.x, vec1.y+vec2.y, vec1.z+vec2.z);
41 end

```

**Figura 6.** Programación en Matlab de la Clase *Vector\_*, *sobre carga plus*.

## Clase Particula

Esta clase permite crear objetos de tipo partícula y calcular su trayectoria dado un cierto campo magnético. Los objetos de esta clase tienen los siguientes atributos:

- **posicion:** Matriz de objetos de la clase Vector\_ que representa la posición de la partícula durante su trayectoria.
- **velocidad:** Matriz de objetos de la clase Vector\_ que representa la velocidad de la partícula en cada punto de su trayectoria.
- **aceleracion:** Matriz de objetos de la clase Vector\_ que representa la aceleración de la partícula en cada punto de su trayectoria.
- **carga:** Atributo que representa la carga de la partícula.
- **masa:** Atributo que representa la masa de la partícula.
- **color:** Matriz de 3 datos representando el color de la partícula al ser graficada.

Los métodos de esta clase son los siguientes:

- **Constructor Particula:** Recibe la posición inicial, la velocidad inicial (como objetos Vector\_), la carga y la masa de la partícula. Una vez recibe estos valores, los asigna a los atributos correspondientes, además genera el color de la partícula de forma aleatoria.

```
1 classdef Particula
2     properties
3         %Matrices de objetos Vector_ para la posicion, velocidad y
4         %aceleracion de la particula
5         posicion = Vector_.empty()
6         velocidad = Vector_.empty()
7         aceleracion = Vector_.empty()
8         %Carga y masa de la particula
9         carga
10        masa
11        color
12    end
13    methods
14        %Constructor de la clase Particula
15        function p = Particula(pos0,v0,q,m)
16            if(nargin == 0)
17                carga = 1;
18                masa = 1;
19            else
20                p.posicion(1) = pos0;
21                p.velocidad(1) = v0;
22                p.carga = q;
23                p.masa = m;
24                p.color = [rand rand rand];
25            end
26        end
```

**Figura 7.** Programación en Matlab de la Clase *Particula*, atributos y constructor.

- **calcularTrayectoria:** Este método calcula la trayectoria de la partícula utilizando la fórmula de la fuerza de Lorentz, recibe como parámetros la línea cargada que genera el campo magnético, la cantidad de puntos que tendrá la trayectoria y el tiempo máximo de esta trayectoria. Una vez ingresados los datos, se calcula el campo y la aceleración en el punto inicial de la trayectoria, además del diferencial de tiempo a usar en los cálculos. Posteriormente, mediante un ciclo for, se calcula la trayectoria mediante dos secciones, la primera calcula una posición, velocidad, campo magnético y aceleración temporales con el método de euler. Después se utilizan los valores temporales para calcular y asignar el valor de la posición, velocidad, campo magnético y aceleración en el siguiente punto de la trayectoria (euler mejorado).

```

20      %Calculo de la trayectoria de la particula
21      function p = calcularTrayectoria(p,linea,n,tmax)
22          %Calcular campo en el punto inicial
23          campo = calcularCampoPunto(linea, p.posicion(1));
24          %Calcular aceleracion inicial
25          p.aceleracion(1) = (p.carga/p.masa)*cruz(p.velocidad(1),campo);
26          %Calcular delta t
27          delta = tmax / n;
28          %Ciclo para calcular la trayectoria mediante euler mejorado
29          for i=2:n
30              %Posicion, velocidad, campo y aceleracion temporales
31              %para calcular los datos en el siguiente punto (euler)
32              p_euler = p.posicion(i-1) + (delta*p.velocidad(i-1));
33              v_euler = p.velocidad(i-1) + (delta*p.aceleracion(i-1));
34              campo_euler = calcularCampoPunto(linea, p_euler);
35              a_euler = (p.carga/p.masa)*cruz(v_euler,campo_euler);
36              %Posicion, velocidad, campo y aceleracion en el siguiente
37              %punto de la trayectoria
38              p.posicion(i) = p.posicion(i-1) + (delta*0.5)*(v_euler + p.velocidad(i-1));
39              p.velocidad(i) = p.velocidad(i-1) + (delta*0.5)*(a_euler + p.aceleracion(i-1));
40              campo = calcularCampoPunto(linea, p.posicion(i));
41              p.aceleracion(i) = (p.carga/p.masa)*cruz(p.velocidad(i),campo);
42          end
43      end

```

**Figura 8.** Programación en Matlab de la Clase *Particula*, método *calcularTrayectoria*.

### *Clase CampoMag*

Esta clase permite crear y calcular las componentes del campo magnético en los puntos necesarios. Los objetos de esta clase tienen dos atributos:

- **mesh:** Matriz de puntos (objetos de la clase Vector\_), que representa el mallado de los lugares donde se calculará el campo magnético.
- **componentes:** Matriz de puntos (objetos de la clase Vector\_), que representa las componentes calculadas del campo magnético en cada punto del mallado.
- **min\_x, max\_x, min\_y, max\_y, min\_z, max\_z:** Límites de la gráfica del campo

```
7  classdef CampoMag
8  properties
9  % Malla
10 mesh;
11 % Vectores del campo magnético
12 min_x;
13 max_x;
14 min_y;
15 max_y;
16 min_z;
17 max_z;
18 end
```

**Figura 9.** Programación en Matlab de la Clase *CampoMag*, sección de atributos.

Los métodos de esta clase son los siguientes:

- **Constructor CampoMag:** Recibe los valores del número de nodos a calcular y los límites en cada uno de los ejes (n, min\_x, max\_x, min\_y, max\_y, min\_z, max\_z). A partir de estos valores, el método genera la matriz de objetos Vector\_ (mesh), y la llena con las coordenadas de cada nodo, que son calculados a partir del número de nodos y los límites del campo. Posteriormente genera una matriz de objetos Vector\_ (componentes) y la deja vacía para ser utilizada por otro método.

```

18     methods
19         % Constructor de CampoMag, genera la malla
20         function campo = CampoMag(n, xmin, xmax, ymin, ymax, zmin, zmax)
21             X=zeros(n);
22             Y=zeros(n);
23             Z = zeros(1,n);
24             campo.min_x=xmin+(xmin*0.1);
25             campo.max_x=xmax+(xmax*0.1);
26             campo.min_y=ymin+(ymin*0.1);
27             campo.max_y=ymax+(ymax*0.1);
28             campo.min_z=zmin+(zmin*0.1);
29             campo.max_z=zmax+(zmax*0.1);
30             %Malla para X y Y
31             for i=1:n
32                 for j=1:n
33                     X(i,j)=xmin+((j-1)*((xmax - xmin)/(n-1)));
34                     Y(i,j)=ymax-((i-1)*((ymax - ymin)/(n-1)));
35                 end
36             end
37             %Malla para Z
38             for i=1:n
39                 Z(i) = zmax - (i - 1) * (zmax - zmin)/(n-1);
40             end
41             campo.mesh = Vector_.empty; % Crea matriz vacia de vectores para la malla
42             %Creando matriz de malla
43             for i = 1:n
44                 for j = 1:n
45                     for k = 1:n
46                         campo.mesh(i, j, k) = Vector_(X(i, j), Y(i, j), Z(k));
47
48                     end
49                 end
50             campo.componentes = Vector_.empty; % Crea matriz vacia de vectores para las componentes
51         end
52
53         %Metodo que calcula el campo de la linea cargada en cada punto de
54         %la malla
55         function campo = calcularCampo(campo, linea)
56             n = size(campo.mesh);
57             n = n(1);
58             for i = 1:n
59                 for j = 1:n
60                     for k = 1:n
61                         %Llamada a la función calcularCampoPunto
62                         campo.componentes(i,j,k) = calcularCampoPunto(linea, campo.mesh(i,j,k));
63                     end
64                 end
65             end
66         end

```

**Figura 10.** Programación en Matlab de la Clase *CampoMag*, constructor de la clase.

- **calcularCampo:** Este método recibe como parámetro la línea cargada que se usará para calcular el campo magnético. El método utiliza 3 ciclos for anidados para ir calculando el campo en cada punto dentro de la matriz mesh, llamando a la función calcularCampoPunto.

```

39
40          %Metodo que calcula el campo de la linea cargada en cada punto de
41          %la malla
42 function campo = calcularCampo(campo, linea)
43 n = size(campo.mesh);
44 n = n(1);
45 for i = 1:n
46     for j = 1:n
47         for k = 1:n
48             %Llamada a la función calcularCampoPunto
49             campo.componentes(i,j,k) = calcularCampoPunto(linea, campo.mesh(i,j,k));
50         end
51     end
52 end
53
54

```

**Figura 11.** Programación en Matlab de la Clase *CampoMag*, método *calculaCampo*.

- **graficar:** Este es el método encargado de graficar los objetos de tipo campo, recibe como parámetros la línea cargada, un valor de escala para controlar el tamaño de los vectores y un valor de opacidad para controlar qué tan claros u oscuros se representarán los vectores de acuerdo a su magnitud. Antes de graficar, el método genera una matriz de datos donde almacena el valor de la norma de cada vector y también se calcula la norma máxima, siendo estos valores los que nos ayudan a definir la escala de los vectores y la opacidad que tendrán al graficarse. Para graficar la línea cargada usamos la función fplot3 y las funciones simbólicas de la parametrización de la línea. Para graficar los vectores, utilizamos 3 ciclos for que van dibujando cada vector a partir de los valores en las matrices mesh, componentes, normas y op (nodos donde se calcula el campo, componentes del campo en esos puntos, normas de los vectores y opacidad de los vectores respectivamente) usando la función quiver3.

```

% Método para graficar el campo
function campo = graficar(campo,linea, escala, opacidad)
n = size(campo.mesh);
n = n(1);
normas = zeros(n,n,n);
op = zeros(n,n,n);

figure;
%Gráfica de la linea cargada
fplot3(linea.lx, linea.ly, linea.lz,[linea.min_s linea.max_s], 'r','LineWidth',2);
xlabel("X");
ylabel("Y");
zlabel("Z");

%Graficar Campo
hold on;

%Matriz con las normas del campo magnético
for i=1:n
    for j=1:n
        for k=1:n
            normas(i,j,k) = norm([campo.componentes(i,j,k).x ...
                campo.componentes(i,j,k).y ...
                campo.componentes(i,j,k).z]);
        end
    end
end

%Valor máximo de normas del campo magnético
norMax = max(max(max(normas)));

for i=1:n
    for j=1:n
        for k=1:n
            %Opacidad
            op(i,j,k)= 1*(normas(i,j,k)/(norMax*opacidad));
            if op(i,j,k) > 1
                op(i,j,k) = 1;
            end

            if normas(i,j,k) == 0
                normas(i,j,k) = 1;
            end

            %Graficando campo magnético
            quiver3(campo.mesh(i,j,k).x,campo.mesh(i,j,k).y,campo.mesh(i,j,k).z...
                ,escala*(campo.componentes(i,j,k).x)/normas(i,j,k),...
                escala*(campo.componentes(i,j,k).y)/normas(i,j,k),...
                escala*(campo.componentes(i,j,k).z)/normas(i,j,k),...
                'LineWidth',1.5,'MaxHeadSize',1,'AutoScale','off',...
                'Color',[1-op(i,j,k) 1-op(i,j,k) 1-op(i,j,k)]);
        end
    end
end
view(45,45);
hold off;
end|

```

**Figura 12.** Programación en Matlab de la Clase *CampoMag*, método *graficar*.

- **graficarParticula:** Este método se encarga de graficar trayectorias de partículas en el campo magnético, recibe como parámetro una lista de objetos partículas a graficar en la forma de una vector de matlab. Se utiliza la función de Matlab animatedline para crear la línea de trayectoria y el marcador que representará a cada una de las partículas. Posteriormente, utilizando dos ciclos for, se van dibujando los puntos de cada trayectoria con el comando drawnow.

```

%<<<
%Funcion para graficar la trayectoria de particulas
function campo = graficarParticula(campo,particulas)
    %Creacion de linea y marcador para cada particula
    size = length(particulas);
    for i = 1:size
        f(i)= animatedline('Marker','o','Color',[particulas(i).color(1) particulas(i).color(2) ...
            particulas(i).color(3)],'MarkerSize',5,'MarkerFaceColor',[particulas(i).color(1) ...
            particulas(i).color(2) particulas(i).color(3)]);
        g(i)= animatedline('Linestyle','-', 'Color',[particulas(i).color(1) ...
            particulas(i).color(2) particulas(i).color(3)],'LineWidth',2.5);
    end
    pasos = length(particulas(1).posicion);
    %Graficar y animar cada particula
    for i=1:pasos
        for j=1:size
            clearpoints(f(j))
            addpoints(f(j),particulas(j).posicion(i).x,particulas(j).posicion(i).y,particulas(j).posicion(i).z)
            addpoints(g(j),particulas(j).posicion(i).x,particulas(j).posicion(i).y,particulas(j).posicion(i).z)
            drawnow
        end
    end
end

```

**Figura 13.** Programación en Matlab de la Clase *CampoMag*, método *graficarParticula*.

### Función calcularCampoPunto

Mediante la implementación de esta función dentro del método calcularCampo de la clase CampoMag, se obtiene el campo magnético de la línea cargada con respecto a un punto en el espacio (nodos en la malla).

La función recibe como parámetros la línea cargada y un punto con componentes “x”, “y” y “z”. A través de la implementación de la ley de Biot-Savart, que permite calcular el campo magnético generado por una línea cargada y a partir de los parámetros que recibe la función, se hace el cálculo del campo magnético, antes de aplicar la ley, se calcula el vector r, el cual es el resultante del punto con respecto a la línea parametrizada y se obtiene el producto cruz entre la derivada de la línea parametrizada y el vector r. Con la implementación de la ley de Biot-Savart se obtienen las funciones a integrar de cada componente de donde se generará el campo magnético, se resuelven estas integrales mediante el método de simpson, obteniendo las tres componentes del campo magnético con respecto a un punto.

```

function v = calcularCampoPunto(linea, punto)
    syms s rx ry rz fx fy fz;
    %disp("Calculando campo...");
    %Calculo vector r
    r = [(punto.x - linea.lx) (punto.y - linea.ly) (punto.z - linea.lz)];
    r = r(s);
    rx(s) = r(1);
    ry(s) = r(2);
    rz(s) = r(3);

    %Calculo producto cruz de vector r y derivada de la linea
    cruz = [((rz*linea.dly)-(ry*linea.dlz)) -((rz*linea.dlx)-(rx*linea.dlz)) ...
        |((ry*linea.dlx)-(rx*linea.dly))];

    %Calculo de funciones a integrar con la Ley de Biot-Savart
    funciones_integrandos = (linea.I*(10^(-7))*cruz)/(norm(r)^3);
    funciones_integrandos = funciones_integrandos(s);
    fx(s) = funciones_integrandos(1);
    fy(s) = funciones_integrandos(2);
    fz(s) = funciones_integrandos(3);

    %Calculo de componentes del campo magnético con método de simpson
    try
        vx = intSimpson(fx, linea.min_s, linea.max_s, linea.integral);
        vy = intSimpson(fy, linea.min_s, linea.max_s, linea.integral);
        vz = intSimpson(fz, linea.min_s, linea.max_s, linea.integral);

    catch
        vx = 0;
        vy = 0;
        vz = 0;
    end
    v = Vector_(vx, vy, vz);
end

```

**Figura 14.** Programación en Matlab de la función *calcularCampoPunto*.

#### Función *intSimpson*

Esta función está basada en el método de integración numérica de Simpson 1/3, este método permite hacer una aproximación numérica de integrales definidas que debido a su complejidad no son sencillas de calcular. En este caso la función se ajusta a un polinomio de segundo grado,  $p(x)$ , a la función  $f(x)$ , integrando esta función se obtiene una aproximación a la integral definida.

$$\int_a^b f(x)dx \cong \int_a^b P_n(x)dx$$

Los 3 puntos de ajuste del polinomio son los dos límites,  $x_1 = a$ ,  $x_3 = b$  y el punto medio  $x_2 = (a + b) / 2$ . El polinomio puede ser escrito de la forma

$$p(x) = \alpha + \beta(x - x_1) + \gamma(x - x_1)(x - x_2)$$

y debe satisfacer la condición de que pase por los 3 puntos, es decir,  $p(x_1) = f(x_1)$ ,

$p(x_2) = f(x_2)$ ,  $p(x_3) = f(x_3)$ . Esto implica

$$\alpha = f(x_1)$$

$$\beta = (f(x_2) - f(x_1))/(x_2 - x_1)$$

$$\gamma = \frac{f(x_3) - 2f(x_2) + f(x_1)}{2h^2},$$

con

$$h = \frac{(b - a)}{2}$$

Sustituyendo las constantes en el polinomio y realizando la integral en el intervalo  $[a, b]$ , obtenemos

$$I = \int_{x_1}^{x_3} f(x)dx \approx \frac{h}{3}[f(x_1) + 4f(x_2) + f(x_3)] = \frac{h}{3}[f(a) - 4f(\frac{a+b}{2}) + f(b)]$$

Una vez que conocemos cómo calcular el área de un segmento, se obtienen n particiones, mediante la resta del límite de integración superior menos el inferior entre las n particiones, donde n tiene que ser par, para obtener el valor de h correspondiente a la longitud de cada partición.

$$h = \frac{b - a}{n}$$

Entonces,

$$\int_a^b f(x)dx = \frac{Ax^3}{3} + \frac{Bx^2}{2} + Cx \Big| \frac{b}{a} = \frac{A(b^3 - a^3)}{3} + \frac{B(b^2 - a^2)}{2} + C(b - a)$$

$$f(a) = A * a^2 + B * a + C$$

$$f(b) = A * b^2 + B * b + C$$

$$f\left(\frac{a+b}{2}\right) = A * \left(\frac{a+b}{2}\right)^2 + B * \left(\frac{a+b}{2}\right) + C$$

$$f\left(\frac{a+b}{2}\right) = \frac{A}{4} * (a^2 + b^2 + 2ab) + \frac{B}{2} * (a + b) + C$$

$$f\left(\frac{a+b}{2}\right) = \frac{A}{4} * (a^2 + b^2 + 2ab) + \frac{B}{2} * (a + b) + C$$

$$4f\left(\frac{a+b}{2}\right) = A * \left(a^2 + b^2 + 2ab\right) + 2B * (a + b) + 4C$$

$$\int_a^b f(x)dx = \frac{A(b^3 - a^3)}{3} + \frac{B(b^2 - a^2)}{2} + C(b - a)$$

$$\int_a^b f(x)dx = \frac{A}{3}(b - a)(b^2 + ab + a^2) + \frac{B}{2}(b - a)(b + a) + C(b - a)$$

$$\int_a^b f(x)dx = \left(\frac{b-a}{6}\right)[2A(b^2 + ab + a^2) + 3B(b + a) + 6C]$$

$$\int_a^b f(x)dx = \left(\frac{b-a}{6}\right)[2A(b^2 + ab + a^2) + 2Aab - 2Aab + 3B(b + a) + 6C]$$

$$\int_a^b f(x)dx = \left(\frac{b-a}{6}\right)[2A(b^2 + 2ab + a^2) - 2Aab + 3B(b + a) + 6C]$$

$$\int_a^b f(x)dx = \left(\frac{b-a}{6}\right)[4f\left(\frac{a+b}{2}\right) + A(b^2 + 2ab + a^2) - 2ab + B(b + a) + 2C]$$

$$\int_a^b f(x)dx = \left(\frac{b-a}{6}\right)[4f\left(\frac{a+b}{2}\right) + A(b^2 + a^2) + B(b + a) + 2C]$$

$$\begin{aligned} A(b^2 + a^2) + B(b + a) + 2C &= f(a) + f(b) \\ &= \left(\frac{b-a}{6}\right)[f(a) + f(b) + 4f\left(\frac{a+b}{2}\right)] \end{aligned}$$

Ya que tenemos el área de 3 puntos debemos movernos en grupos de 3 para calcular el área total

$$\int_{x_0}^{x_2} f(x)dx \approx \left(\frac{2\Delta x}{6}\right)[f(x_0) + 4f(x_1) + f(x_2)]$$

$$\int_{x_0}^{x_4} f(x)dx \approx \left(\frac{2\Delta x}{6}\right)[f(x_0) + 4f(x_1) + f(x_2) + 4f(x_3) + f(x_4)]$$

La forma de calcular la aproximación de la integral definida para n segmentos es la siguiente:

$$\int_{x_0}^{x_4} f(x) dx \approx \left( \frac{\Delta x}{3} \right) \left[ f(x_0) + f(x_n) + 4 \sum_{i=0}^{n/2} f(x_{2i-1}) + 2 + 2 \sum_{i=1}^{n/2} f(x_{2i}) \right]$$

```
%Método de Integración por regla de Simpson
function A = intSimpson(f, a, b, n)

deltax = (b-a)/n; %Longitud de las particiones
A=0;
x = a; %x inicial

for i=0:n
    %Para x inicial y x final
    if i == 0 || i == n
        A = A + f(x);

    %Para x par
    elseif mod(i,2) == 0 && i ~= 0 && i ~= n
        A = A + (2*f(x));

    %Para x impar
    elseif mod(i,2) ~= 0 && i ~= 0 && i ~= n
        A = A + (4*f(x));
    end

    x = x + deltax;
end

%Aproximación de la integral
A = (deltax/3)* double(A);
end
```

**Figura 15.** Programación en Matlab de la función *simpson*

#### Main Principal

En este script realizamos cuatro escenarios diferentes, modelamos cuatro líneas cargadas con diferentes parametrizaciones, cada una con su respectivo campo magnético y seis partículas en movimiento con sus trayectorias en función del campo.

Enseguida se presenta la explicación de la modelación de una línea cargada con seis partículas aleatorias, esta es replicada en los tres escenarios (recta vertical, espiral y círculo) cambiando solamente los nombres de los objetos y la parametrización de las líneas cargadas:

Se declaran variables simbólicas ( $lx$ ,  $ly$ ,  $lz$ ,  $s$ ) que servirán para determinar la trayectoria parametrizada de la línea cargada en cada uno de los ejes en el espacio. Enseguida se crea un objeto de tipo *LíneaCargada*, el cual contiene como parámetros las trayectorias ( $lx$ ,  $ly$ ,  $lz$ ), la intensidad de la línea cargada, los límites de parametrización (mínimo y máximo) y el número de iteraciones que servirán para calcular el campo magnético. Posteriormente, creamos un objeto de tipo *CampoMag* con el número de nodos que queremos representar y los límites de la malla en los tres ejes, para así calcular el campo de la *LíneaCargada* anteriormente creada.

La segunda parte consiste en la creación de las seis partículas debido a que queremos que sus posiciones y velocidades iniciales sean aleatorias creamos vectores para cada posición en el espacio y velocidad respectivamente con la función rand. Creamos un vector vacío de partículas. Gracias a un iterador, creamos una por una las partículas, con los valores de posiciones y velocidades iniciales dadas por rand y las almacenamos en el vector de partículas antes creado. Finalmente, graficamos todos los elementos: línea cargada, su campo magnético y las seis partículas en movimiento.

```

%% Línea cargada: Recta Vertical
lx(s)= 0;
ly(s) = 0;
lz(s) = s;
linea_recta = LineaCargada(lx, ly, lz, 2, -3, 3, 20);
campo_recta = CampoMag(8, -5, 5, -5, 5, -5, 5);
campo_recta = campo_recta.calcularCampo(linea_recta);

linea_pos_x = (1-(-1)).*rand(6, 1)-1;
linea_pos_y = (1-(-1)).*rand(6, 1)-1;
linea_pos_z = (1-(-1)).*rand(6, 1)-1;
linea_vel_x = (1-(-1)).*rand(6, 1)-1;
linea_vel_y = (1-(-1)).*rand(6, 1)-1;
linea_vel_z = (2-(-2)).*rand(6, 1)-2;

linea_listaP = Particula.empty();
for i=1:6
    linea_listaP(i) = Particula(Vector_(linea_pos_x(i), linea_pos_y(i), ...
        linea_pos_z(i)), Vector_(linea_vel_x(i), linea_vel_y(i), linea_vel_z(i)),...
        -1, 1e-6);
    linea_listaP(i) = linea_listaP(i).calcularTrayectoria(linea_recta, 10, 0.1);
end
campo_recta = campo_recta.graficar(linea_recta,0.5, 0.5);
campo_recta = campo_recta.graficarParticula(linea_listaP);

```

**Figura 16:** Programación de Matlab de la modelación de una Línea recta vertical cargada, su campo magnético y seis partículas en movimiento

```

%% Linea cargada: espiral
clear;
syms s lx ly lz;
lx(s)= cos(s);
ly(s) = sin(s);
lz(s) = 0.03*s;
linea_espiral = LineaCargada(lx, ly, lz, 1, -10*pi, 10*pi, 100);
campo_espiral = CampoMag(7, -2, 2, -2, 2, -2, 2);
campo_espiral = campo_espiral.calcularCampo(linea_espiral);

espiral_pos_x = (.5-(-1)).*rand(6, 1)-1;
espiral_pos_y = (.5-(-1)).*rand(6, 1)-1;
espiral_pos_z = (.5-(-1)).*rand(6, 1)-1;
espiral_vel_x = (2-(-2)).*rand(6, 1)-2;
espiral_vel_y = (2-(-2)).*rand(6, 1)-2;
espiral_vel_z = (2-(-2)).*rand(6, 1)-2;

espiral_listaP = Particula.empty();
for i=1:6
    espiral_listaP(i) = Particula(Vector_(espiral_pos_x(i), espiral_pos_y(i), ...
        espiral_pos_z(i)), Vector_(espiral_vel_x(i), espiral_vel_y(i), espiral_vel_z(i)),...
        -1, 1e-6);
    espiral_listaP(i) = espiral_listaP(i).calcularTrayectoria(linea_espiral, 10, 0.1);
end
campo_espiral = campo_espiral.graficar(linea_espiral,0.5, 0.1);
campo_espiral = campo_espiral.graficarParticula(espiral_listaP);

```

**Figura 17:** Programación de Matlab de la modelación de una Línea en espiral cargada, su campo magnético y seis partículas en movimiento

```

%% Linea cargada: circulo
lx(s)= cos(s);
ly(s) = sin(s);
lz(s) = 0;
linea_circulo = LineaCargada(lx, ly, lz, 1, 0, 2*pi, 20);
campo_circulo = CampoMag(8, -2, 2, -2, 2, -2, 2);
campo_circulo = campo_circulo.calcularCampo(linea_circulo);

circulo_pos_x = (.5-(-1)).*rand(6, 1)-1;
circulo_pos_y = (.5-(-1)).*rand(6, 1)-1;
circulo_pos_z = (.5-(-1)).*rand(6, 1)-1;
circulo_vel_x = (2-(-2)).*rand(6, 1)-2;
circulo_vel_y = (2-(-2)).*rand(6, 1)-2;
circulo_vel_z = (2-(-2)).*rand(6, 1)-2;

circulo_listaP = Particula.empty();
for i=1:6
    circulo_listaP(i) = Particula(Vector_(circulo_pos_x(i), circulo_pos_y(i), ...
        circulo_pos_z(i)), Vector_(circulo_vel_x(i), circulo_vel_y(i), circulo_vel_z(i)),...
        -1, 1e-6);
    circulo_listaP(i) = circulo_listaP(i).calcularTrayectoria(linea_circulo, 10, 0.1);
end
campo_circulo = campo_circulo.graficar(linea_circulo,0.3, .6);
campo_circulo = campo_circulo.graficarParticula(circulo_listaP);

```

**Figura 18:** Programación de Matlab de la modelación de una Línea cargada circular, su campo magnético y seis partículas en movimiento

## Modelación del campo magnético terrestre

Al igual que para las líneas cargadas anteriormente mencionadas, en este caso se modeló una espiral que sigue la trayectoria de la superficie de una esfera en los tres ejes del espacio y se calculó su campo magnético. Sin embargo, las partículas utilizadas para esta simulación buscan tener una relación con las dimensiones de partículas reales tales como los protones y los electrones, por ello se inicializan cada una en variables independientes y se en listan en un sólo vector para graficarse.

```
%> Linea cargada: tierra
clear;
syms s lx ly lz;
lx(s)= 6371000*sqrt(1-(0.01*s)^2)*cos(s);
ly(s) = 6371000*sqrt(1-(0.01*s)^2)*sin(s);
lz(s) = 6371000*0.01*s;
linea_esfera = LineaCargada(lx, ly, lz, -3*10^7, -99, 99, 50);
campo_esfera = CampoMag(8, -7000000, 7000000, -7000000, 7000000, -7000000, 7000000);
campo_esfera = campo_esfera.calcularCampo(linea_esfera);

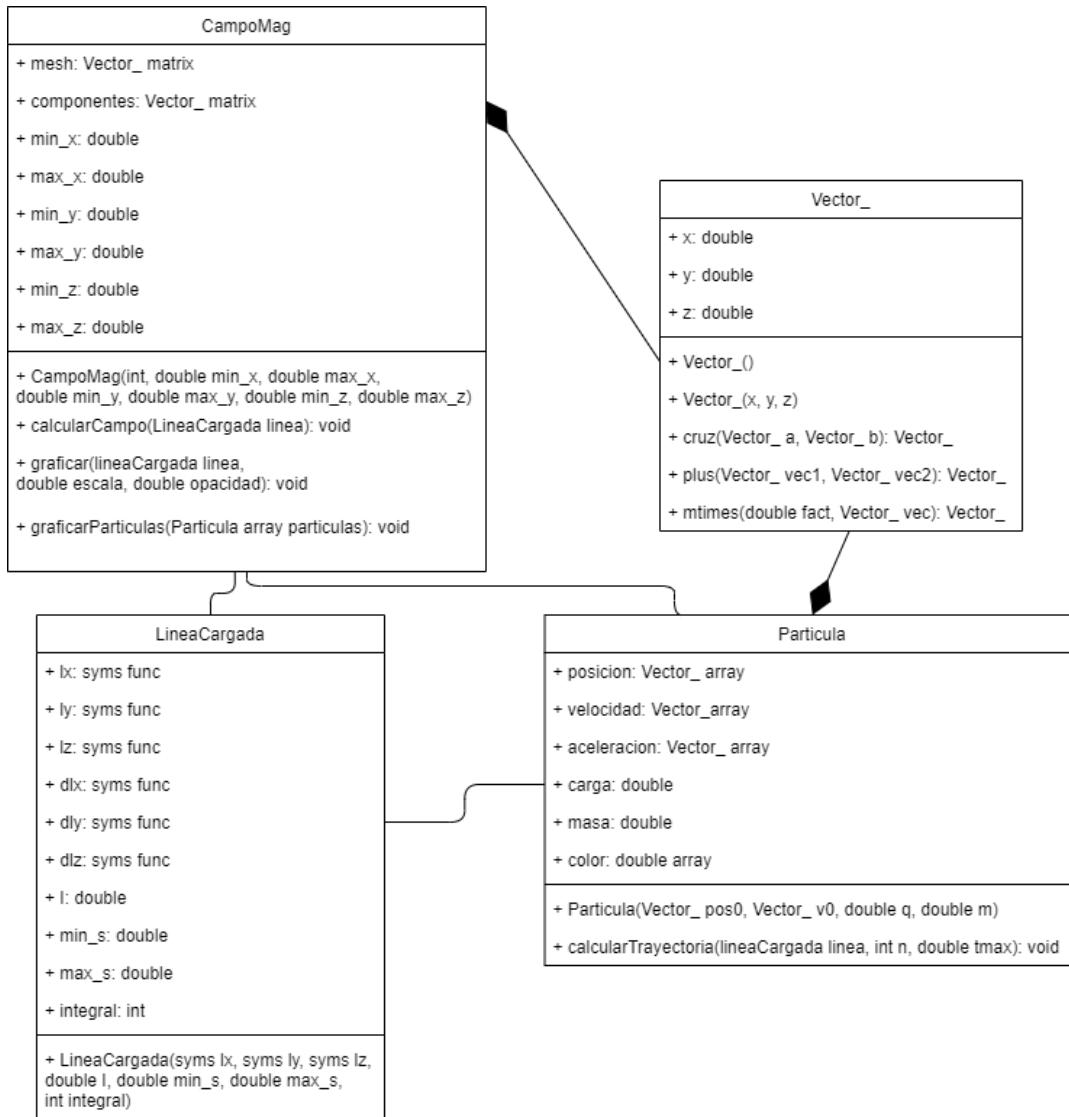
proton1 = Particula(Vector_(6*10^6,6*10^6,6*10^6),...
    Vector_(-0.1875*10^9,-0.1875*10^9,-0.1875*10^9),1.6*10^-19,1.67*10^-27);
proton1 = proton1.calcularTrayectoria(linea_esfera,200,10^-4);
proton2 = Particula(Vector_(-7*10^6,-5*10^6,2*10^6),...
    Vector_(0.6*10^9,0,0),1.6*10^-19,1.67*10^-27);
proton2 = proton2.calcularTrayectoria(linea_esfera,200,10^-4);
proton3 = Particula(Vector_(-7*10^6,2*10^6,0),...
    Vector_(0.75*10^9,0,0),1.6*10^-19,1.67*10^-27);
proton3 = proton3.calcularTrayectoria(linea_esfera,200,10^-4);
proton4 = Particula(Vector_(-6*10^6,4*10^6,2*10^6),...
    Vector_(0.6*10^9,0,0),1.6*10^-19,1.67*10^-27);
proton4 = proton4.calcularTrayectoria(linea_esfera,200,10^-4);
electron1 = Particula(Vector_(6*10^6,6*10^6,6*10^6),...
    Vector_(-0.75*10^6,-0.75*10^6,-0.75*10^6),-1.6*10^-19,9.11*10^-31);
electron1 = electron1.calcularTrayectoria(linea_esfera,200,5.5*10^-7);
electron2 = Particula(Vector_(6*10^6,6*10^6,6*10^6),...
    Vector_(-0.1875*10^6,-0.1875*10^6,-0.1875*10^6),-1.6*10^-19,9.11*10^-31);
electron2 = electron2.calcularTrayectoria(linea_esfera,200,5.5*10^-7);
electron3 = Particula(Vector_(-6*10^6,-5*10^6,3*10^6),...
    Vector_(1*10^6,0.1875*10^6,-0.1875*10^6),-1.6*10^-19,9.11*10^-31);
electron3 = electron3.calcularTrayectoria(linea_esfera,200,4.375*10^-7);
electron4 = Particula(Vector_(-6*10^6,-3*10^6,-3*10^6),...
    Vector_(1*10^6,-0.1875*10^6,-0.1875*10^6),-1.6*10^-19,9.11*10^-31);
electron4 = electron4.calcularTrayectoria(linea_esfera,200,2.125*10^-7);
listaP = [proton1 proton2 proton3 proton4 electron1 electron2 electron3 electron4];
campo_esfera = campo_esfera.graficar(linea_esfera,10^6,0);
campo_esfera = campo_esfera.graficarParticula(listaP);
```

**Figura 19:** Programación de Matlab de la modelación del campo magnético terrestre.

## Personalización

Para la simulación y la estructura del código, se utilizó el paradigma de Programación Orientada a Objetos. De esta manera, es posible mantener las características que comparten diferentes clases, como la línea en la que fluye corriente, o bien el campo magnético. Además, al utilizar este paradigma, es más sencillo modificar el código y agregar nuevas funcionalidades sin comprometer a las ya existentes. En concreto, se creó una nueva clase que modela el movimiento de las partículas y que funciona perfectamente con el código creado para el primer entregable (gráfica de un campo magnético).

A continuación se muestra el diagrama UML del diseño.



**Figura 20.** Diagrama UML del diseño del programa

En este caso, la clase CampoMag está en relación de composición con la clase Vector\_, debido a que tiene en sus propiedades las matrices mesh y componentes, las cuales son de tipo Vector\_. La clase Particula también tiene una relación de composición con la clase Vector\_ porque contiene matrices 1xn (arreglos) de este tipo de dato, correspondientes al registro de su posición, velocidad y aceleración.

Por otro lado, la clase CampoMag se encuentra relacionada con la clase LineaCargada, aunque en una relación de asociación, ya que se utiliza la última clase como parámetro de algunos métodos y hacen uso de ella, a pesar de que no se encuentra declarada dentro de los atributos de CampoMag. Las clases LineaCargada y Particula también tienen una relación de asociación, ya que Particula hace uso de un objeto línea cargada para poder realizar el cálculo de la trayectoria.

La accesibilidad de los atributos y métodos para todas las clases es pública. Los constructores de las clases se utilizan para crear objetos con los atributos correspondientes. El caso especial es el constructor de la clase CampoMag, ya que inicializa la malla mesh dada la cantidad de puntos n, y los límites introducidos para x, y, z, dentro de los parámetros del constructor. Además, se utiliza sobrecarga de operadores para realizar suma de objetos tipo Vector\_ y multiplicación de un objeto Vector\_ por un escalar. Dentro de la clase Vector\_ también se implementa la función para realizar el producto cruz de dos vectores, dado que se utiliza en diferentes lugares de la implementación.

La principal ventaja de utilizar esta forma de programación es que podemos tener diferentes objetos CampoMag al mismo tiempo, e incluso partículas, ambos guardados en la memoria, sin que estos tengan que graficarse al instante o interfieran uno con otro. Esto es muy útil al poder recuperar la información de los campos y las trayectorias al únicamente tener que calcularlos una vez, considerando el enorme tiempo de cómputo que requieren, y posteriormente utilizarlos varias veces con diferentes opciones de graficación.

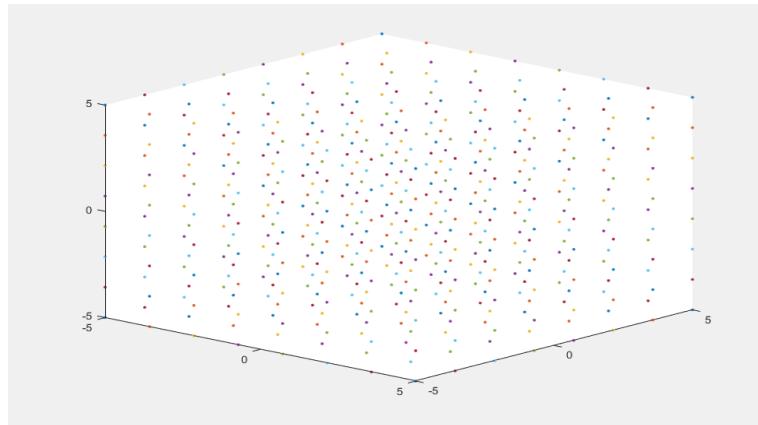
Para tener un manejo más accesible de la graficación de los vectores del campo magnético y que se adecue a la parametrización de línea cargada, se agrego la opción para que la opacidad y escala con la que se muestra el campo magnético, sea un parámetro de entrada, con el fin de adecuar la graficación conforme a su visualización. Otro parámetro de entrada que puede ser modificado es el número de particiones que se hará de acuerdo a cada caso para el cálculo de la aproximación de la integral por el método de Simpson.

## Resultados

### Línea finita

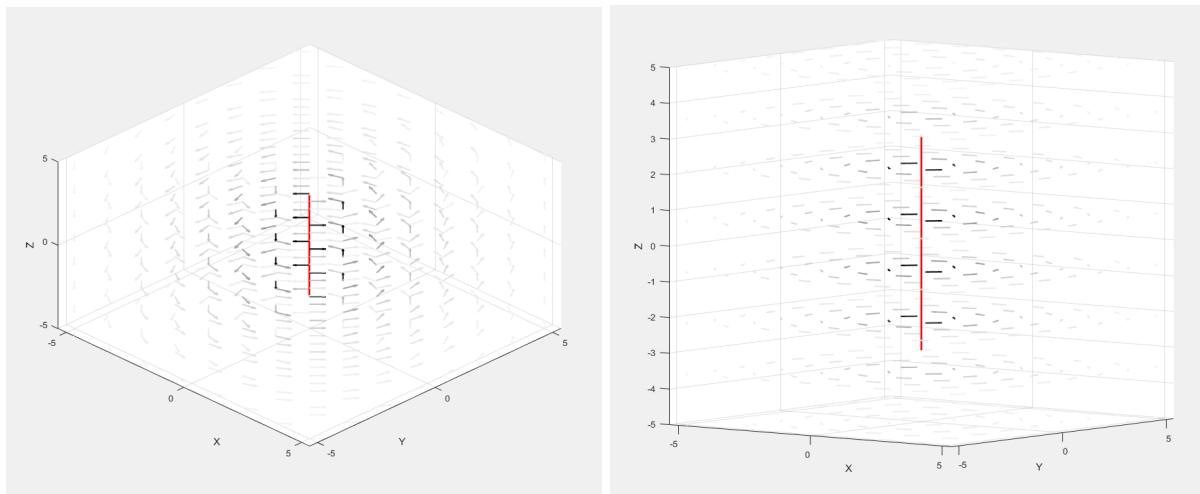
Se parametrizó una línea finita, con carga de  $I = 1$ , mediante los siguientes parámetros  $lx(s) = 0$ ,  $ly(s) = 0$  y  $lz(s) = s$ , con intervalos de  $s \in [-3,3]$ , de la que se generó su malla y el campo magnético sobre la malla generada.

### Malla

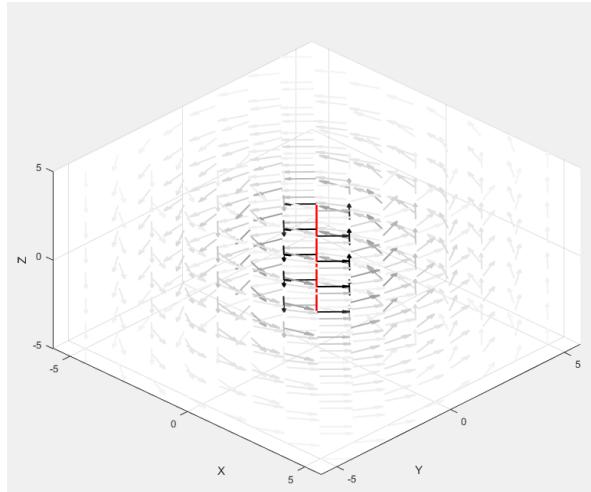


**Figura 21.** Malla generada entre los intervalo de  $x_{\min}=-3$ ,  $x_{\max}=3$ ,  $y_{\min}=-3$ ,  $y_{\max}=3$ ,  $z_{\min}=-3$ ,  $z_{\max}=3$  con 8 nodos.

### Campo magnético



**Figura 22.** Campo magnético de la línea finita con parametrización  $lx(s) = 0$ ,  $ly(s) = 0$  y  $lz(s) = s$ , con intervalos de  $s \in [-3,3]$  y carga igual a 1, con escala de 0.5 y opacidad de 0.5.

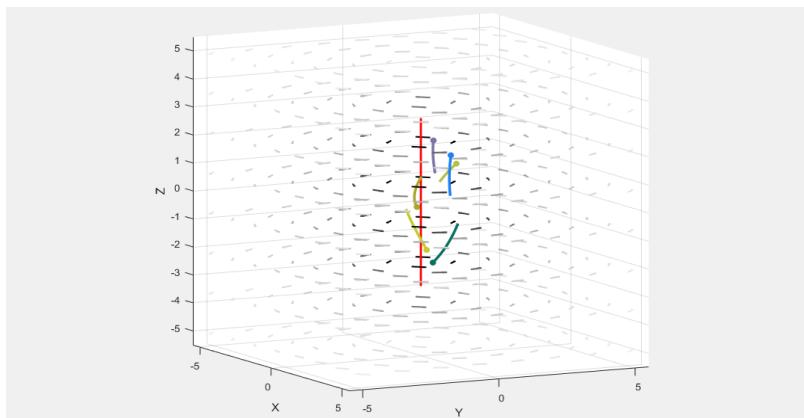


**Figura 23.** Campo magnético de la línea finita con parametrización  $l_x(s) = 0$ ,  $l_y(s) = 0$  y  $l_z(s) = s$ , con intervalos de  $s \in [-3,3]$  y carga igual a 1, con escala de 1 y opacidad de 1.

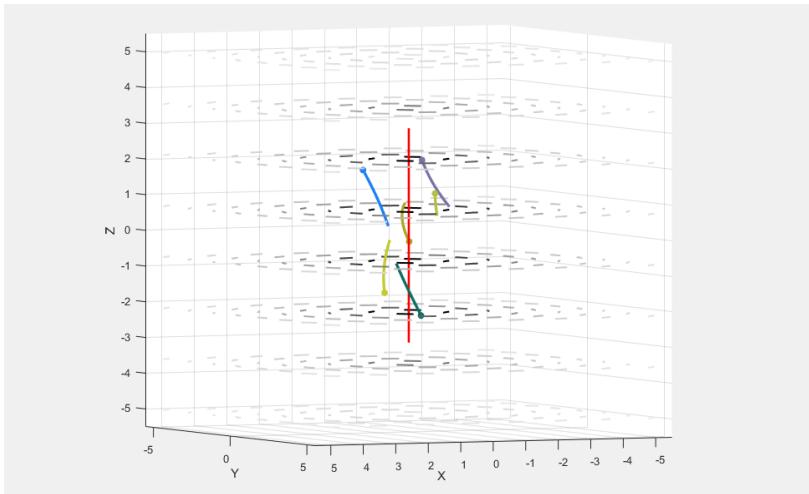
En la figura 22 y 23 se observa el campo magnético generado por la misma línea finita, sin embargo se muestra con escalas y opacidades diferentes la graficación del campo, siendo más visible una de las personalizaciones que se hizo en la creación del programa, en el que la opacidad y escala de los vectores puede ser modificada.

En ambas figuras se puede observar que los vectores en el centro de la malla tienen menor opacidad y los que se encuentran más alejados de la línea cargada tienen mayor opacidad, esto debido a que el campo magnético es mayor en el centro de la malla y va disminuyendo conforme se aleja de línea.

Se hizo la simulación de 6 partículas que se generan de forma aleatoria, como se muestra a continuación:



**Figura 24.** Vista lateral del campo magnético generado por una línea recta con 6 partículas en posiciones y con velocidades aleatorias.



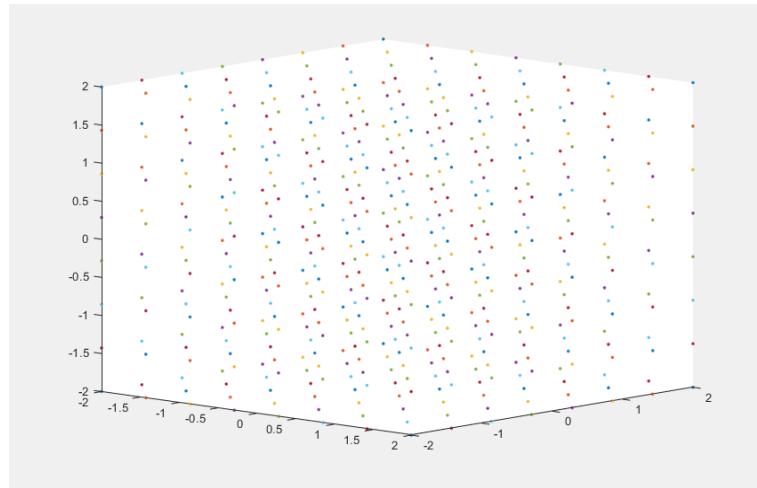
**Figura 25.** Vista lateral del campo magnético generado por una línea recta con 6 partículas en posiciones y con velocidades aleatorias.

En la figura 24 y 25 se pueden observar 6 de las partículas generadas de forma aleatoria con parámetros que se encuentran entre posiciones de -1 a 1 y con velocidades de entre -2 a 2, ambos parámetros en coordenadas ‘x’, ‘y’ y ‘z’, el movimiento que tiene la trayectoria de la partícula sigue las condiciones de la fuerza de Lorentz descritas anteriormente, también se observa que estas trayectorias se mueven de acuerdo al campo magnético generado por la línea recta, algunas de ellas se encuentran girando alrededor de la línea cargada, mientras que otras tratan de escapar del campo generado. Debido a que tienen velocidades y posiciones diferentes las trayectorias son diferentes en cada una de las partículas.

## Círculo

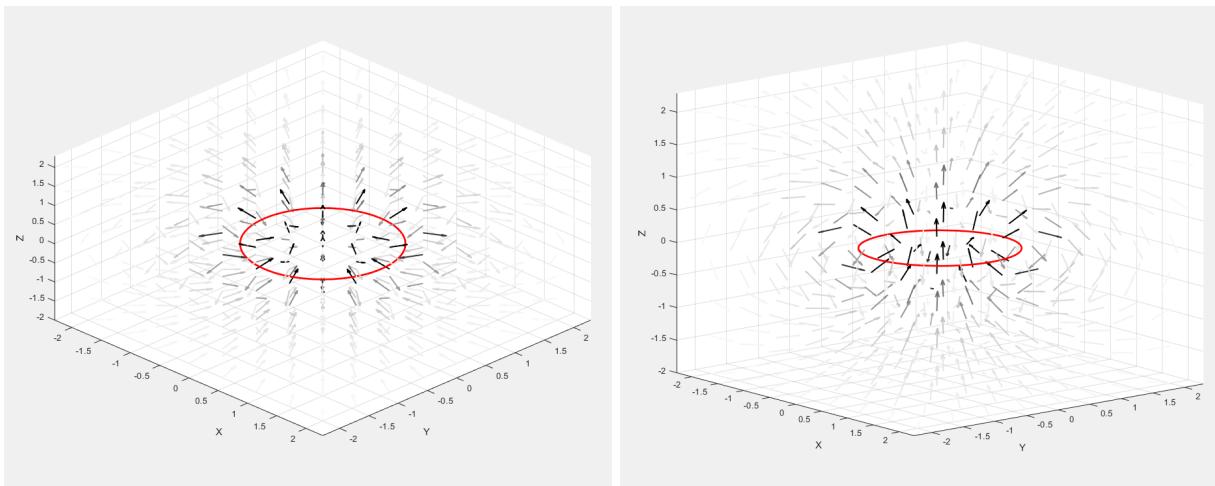
Se parametrizó un círculo con carga de  $I = 1$ , mediante los siguientes parámetros  $lx(s) = \cos(s)$ ,  $ly(s) = \sin(s)$  y  $lz(s) = 0$ , con intervalos de  $s \in [0, 2\pi]$ , de la que se generó su malla y el campo magnético sobre la malla generada.

## Malla



**Figura 26.** Malla generada entre los intervalo de  $xmin=-2$ ,  $xmax=2$ ,  $ymin=-2$ ,  $ymax=2$ ,  $zmin=-2$ ,  $zmax=2$  con 8 nodos.

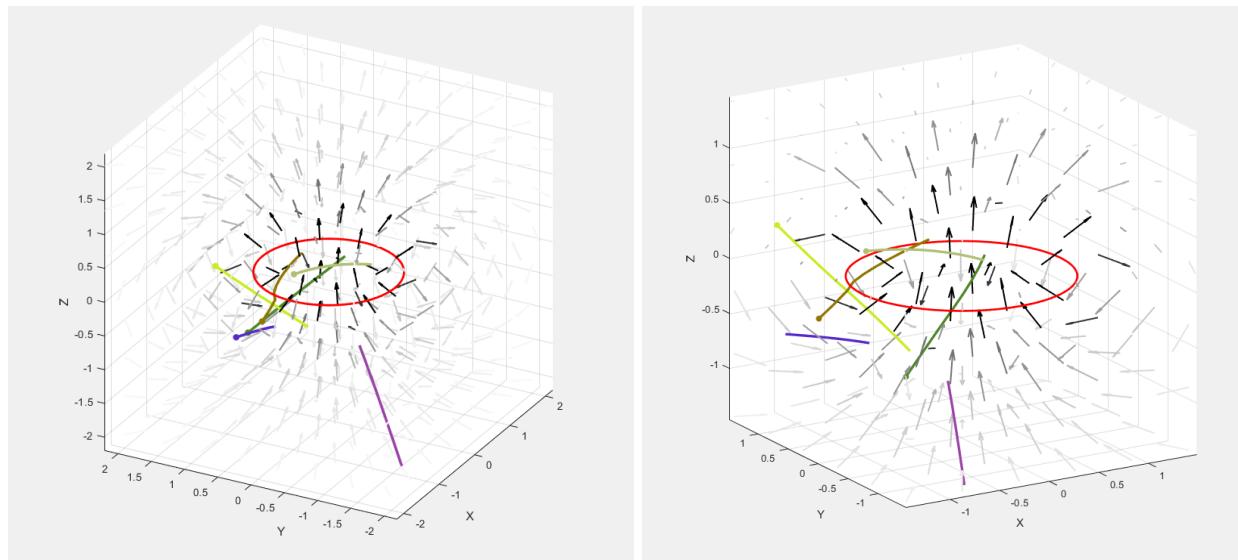
## Campo magnético



**Figura 27.** Campo magnético del círculo con parametrización  $lx(s) = \cos(s)$ ,  $ly(s) = \sin(s)$  y  $lz(s) = 0$ , con intervalos de  $s \in [0, 2\pi]$  y carga igual a 1.

Se puede observar en la figura 27 que los vectores en el centro de la malla tienen menor opacidad y los que se encuentran más alejados del círculo cargado tienen mayor opacidad, esto debido a que el campo magnético es mayor cerca del círculo con carga y va disminuyendo conforme se aleja de él.

Para el campo magnético generado anteriormente por un círculo con la misma parametrización y carga, se hizo la simulación de 6 partículas que se generan de forma aleatoria, como se muestra a continuación:



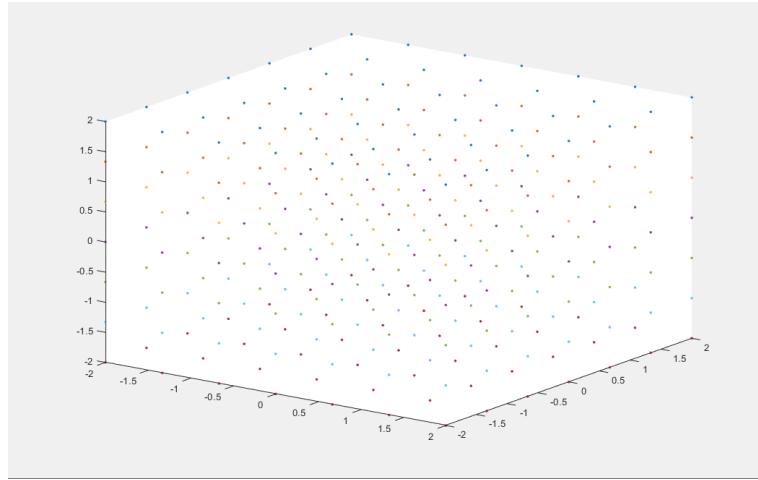
**Figura 28..** Vista superior y lateral con zoom del campo magnético generado por un círculo cargado, con 6 partículas en posiciones y con velocidades aleatorias.

En la figura 28 se pueden observar 6 de las partículas generadas de forma aleatoria con parámetros que se encuentran entre posiciones de -0.5 a 1 y con velocidades de entre -2 a 2, ambos parámetros en coordenadas ‘x’, ‘y’ y ‘z’, el movimiento que tiene la trayectoria de la partícula sigue las condiciones de la fuerza de Lorentz, se puede observar que estas trayectorias se mueven de acuerdo al campo magnético generado por el círculo cargado. Debido a que tienen velocidades y posiciones diferentes las trayectorias son diferentes en cada una de las partículas, se observa que algunas de ellas cruzan cerca del círculo cargado pasando por debajo , mientras que otras están tratando de escapar del campo generado.

## Hélice circular

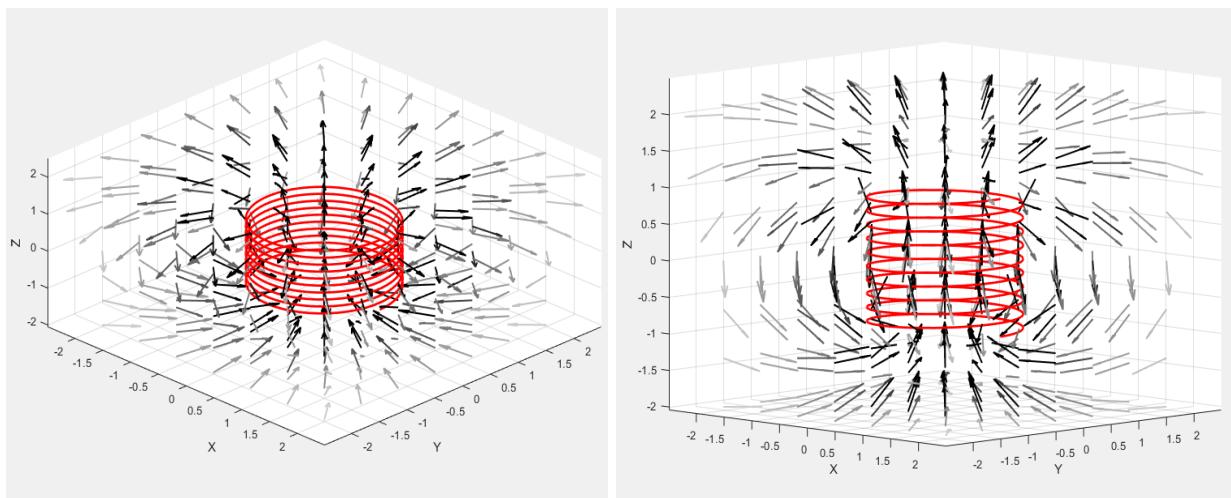
Se parametrizó una hélice circular con carga de  $I = 1$ , mediante los siguientes parámetros  $lx(s) = \cos(s)$ ,  $ly(s) = \sin(s)$  y  $lz(s) = 0.03s$ , con intervalos de  $s \in [-7\pi, 10\pi]$ , de la que se generó su malla y el campo magnético sobre la malla generada.

### Malla



**Figura 29.** Malla generada entre los intervalo de  $x_{min}=-2$ ,  $x_{max}=2$ ,  $y_{min}=-2$ ,  $y_{max}=2$ ,  $z_{min}=-2$ ,  $z_{max}=2$  con 7 nodos.

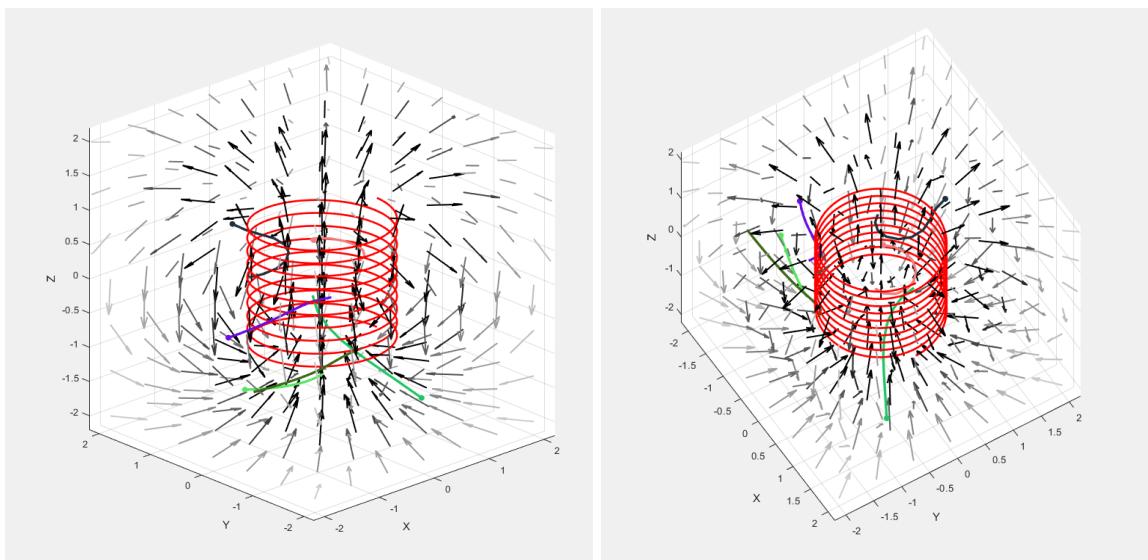
### Campo magnético



**Figura 30.** Campo magnético de una hélice circular con parametrización  $lx(s) = \cos(s)$ ,  $ly(s) = \sin(s)$  y  $lz(s) = 0.03s$ , con intervalos de  $s \in [-10\pi, 10\pi]$  y carga igual a 1.

Se puede observar en la figura 30 que los vectores cerca de la hélice circular tienen menor opacidad y los que se encuentran más alejados del círculo cargado tienen mayor opacidad, esto debido a que el campo magnético es mayor cerca de la hélice con carga y va disminuyendo conforme se aleja de él. También se observa que el comportamiento de los vectores del campo magnético salen de la parte superior de la hélice circular y entran por la parte inferior.

Para el campo magnético generado anteriormente por una hélice circular con la misma parametrización y carga, se hizo la simulación de 6 partículas que se generan de forma aleatoria, como se muestra a continuación:



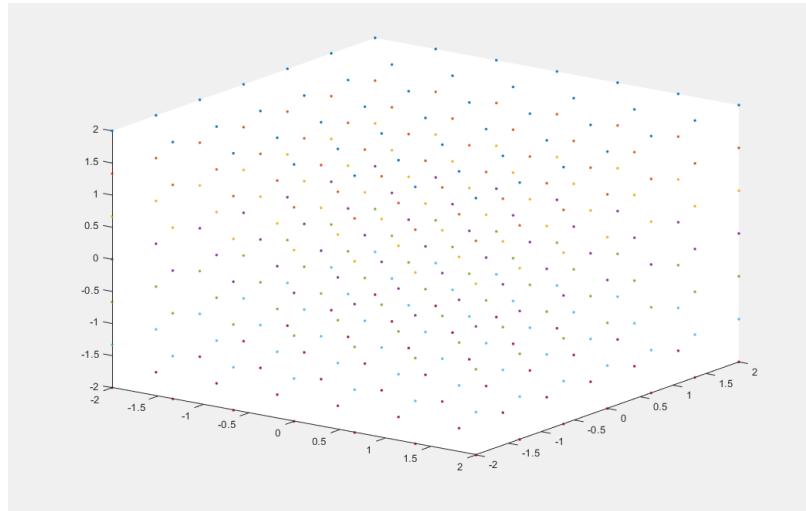
**Figura 31.** Vista superior y lateral del campo magnético generado por una hélice circular, con 6 partículas en posiciones y con velocidades aleatorias.

En la figura 31 se pueden observar 6 de las partículas generadas de forma aleatoria con parámetros que se encuentran entre posiciones de -0.5 a 1 y con velocidades de entre -2 a 2, ambos parámetros en coordenadas ‘x’, ‘y’ y ‘z’, el movimiento que tiene la trayectoria de la partícula sigue las condiciones de la fuerza de lorentz, se puede observar que estas trayectorias se mueven de acuerdo al campo magnético generado por la hélice cargada. Se generan velocidades y posiciones, por lo que las trayectorias de cada una de las partículas será diferente según su posición y velocidad , se observa que algunas de ellas que se generan en el origen se encuentran girando dentro de la espira, otras de ellas tratan de escapar del campo, mientras que en otro caso, las partículas siguen el campo magnético de abajo dirigiéndose hacia arriba de la hélice.

## *Esfera*

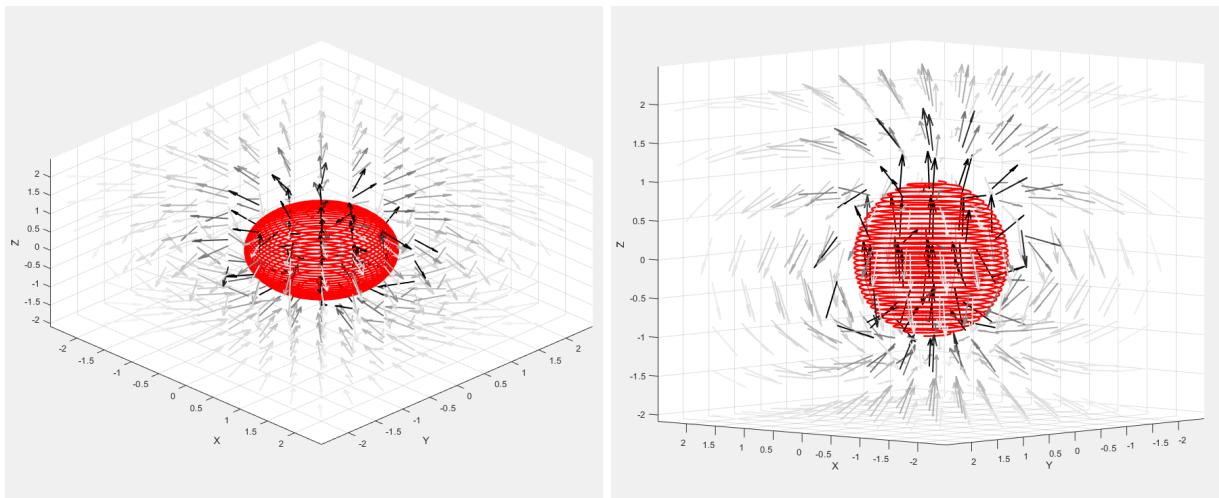
Se parametrizó una esfera hueca con carga de  $I = 1$ , mediante los siguientes parámetros  $lx(s) = \sqrt{1 - (0.01s)^2} \cos(s)$ ,  $ly(s) = \sqrt{1 - (0.01s)^2} \sin(s)$  y  $lz(s) = 0.01s$ , con intervalos de  $s \in [-99, 99]$ , de la que se generó su malla y el campo magnético sobre la malla generada.

## *Malla*



**Figura 32.** Malla generada entre los intervalo de  $x_{\min}=-2$ ,  $x_{\max}=2$ ,  $y_{\min}=-2$ ,  $y_{\max}=2$ ,  $z_{\min}=-2$ ,  $z_{\max}=2$  con 8 nodos.

## *Campo magnético*



**Figura 33.** Campo magnético de una hélice circular con parametrización  $lx(s) = \sqrt{1 - (0.01s)^2} \cos(s)$ ,  $ly(s) = \sqrt{1 - (0.01s)^2} \sin(s)$  y  $lz(s) = 0.01s$ , con intervalos de  $s \in [-99, 99]$  y carga igual a 1.

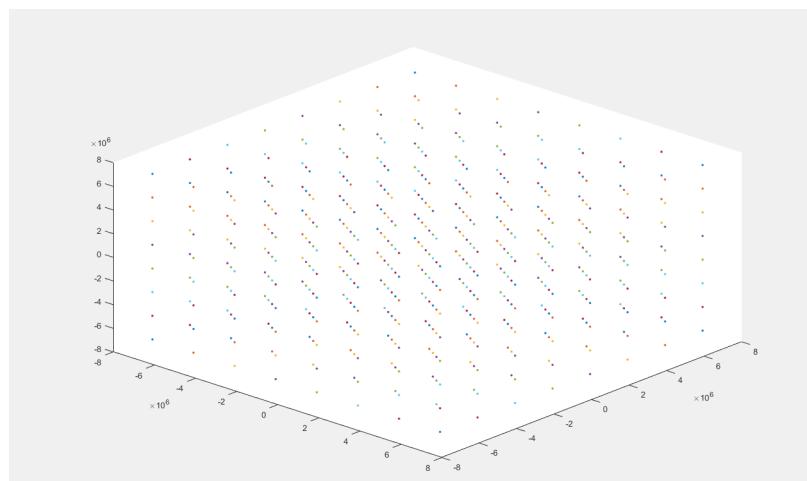
Se puede observar en la figura 33 que los vectores cerca de la esfera hueca cargada tienen menor opacidad y los que se encuentran más alejados de ella tienen mayor opacidad, esto debido a que el campo magnético es mayor cerca de la parametrización de la esfera y va disminuyendo conforme se aleja de ella. También se observa que el comportamiento de los vectores del campo magnético salen de la parte superior y entran por la parte inferior.

Este caso presentado anteriormente fue necesario para comprender adecuadamente, el caso del campo magnético generado por la tierra, a partir de este ejemplo se partió para obtener la simulación del campo magnético terrestre y del movimiento y trayectoria que siguen las partículas cargadas que llegan del sol cerca de nuestro planeta.

### ***Campo magnético terrestre***

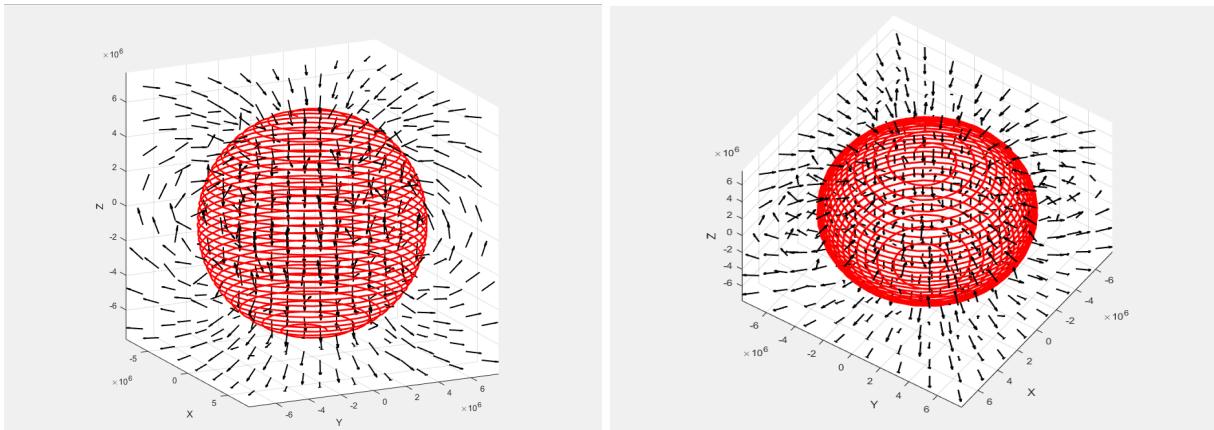
Se parametrizó una esfera con radio de 6371000 (simulando el valor real del radio de la tierra), con una intensidad de corriente de  $I = -3 \times 10^7$ , mediante los siguientes parámetros  $lx(s) = 6371000 * \sqrt{1 - (0.01s)^2} * \cos(s)$ ,  $ly(s) = 6371000 * \sqrt{1 - (0.01s)^2} * \sin(s)$  y  $lz(s) = 6371000 * 0.01s$ , con intervalos de  $s \in [-99, 99]$ , de la que se generó su malla y el campo magnético sobre la malla generada.

### ***Malla***



**Figura 34.** Malla generada entre los intervalo de  $x_{\min} = -7 \times 10^6$ ,  $x_{\max} = 7 \times 10^6$ ,  $y_{\min} = -7 \times 10^6$ ,  $y_{\max} = 7 \times 10^6$ ,  $z_{\min} = -7 \times 10^6$ ,  $z_{\max} = 7 \times 10^6$  con 8 nodos.

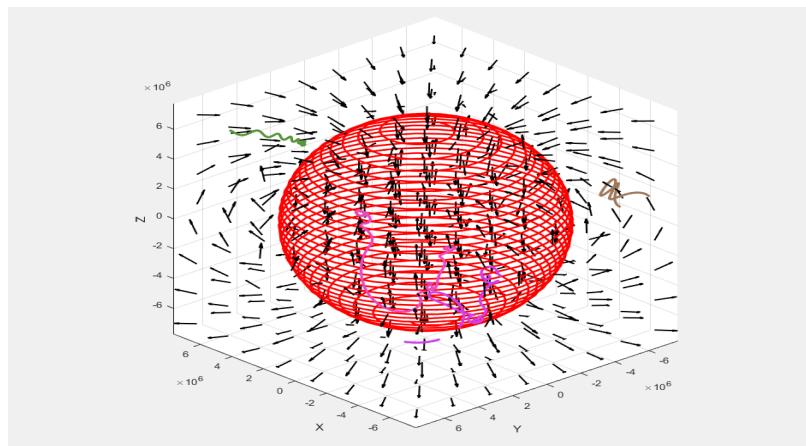
## Campo magnético



**Figura 35.** Campo magnético de la tierra.

En la figura 35 podemos observar el campo magnético, cuyo valor del campo en los polos es de  $5.66 \times 10^{-5}$  T, el cual es un valor correcto, porque sabemos que el campo magnético de la tierra puede variar entre  $2.5 \times 10^{-5}$  T y  $6.5 \times 10^{-5}$  T, siendo más intenso en los polos norte y sur de la tierra, podemos observar que este campo sigue una dirección correcta, debido a que el polo magnético de la tierra se encuentra en el polo sur geográfico, lo cual es lo que se observa en la simulación del campo terrestre. Vemos que sale del sur geográfico y entra por el norte geográfico de la tierra.

Para el campo magnético generado de la tierra, se hizo la simulación de 8 partículas que llegan del sol, 4 de ellas son electrones y las otras 4 corresponden a características de un protón, como se muestra a continuación:



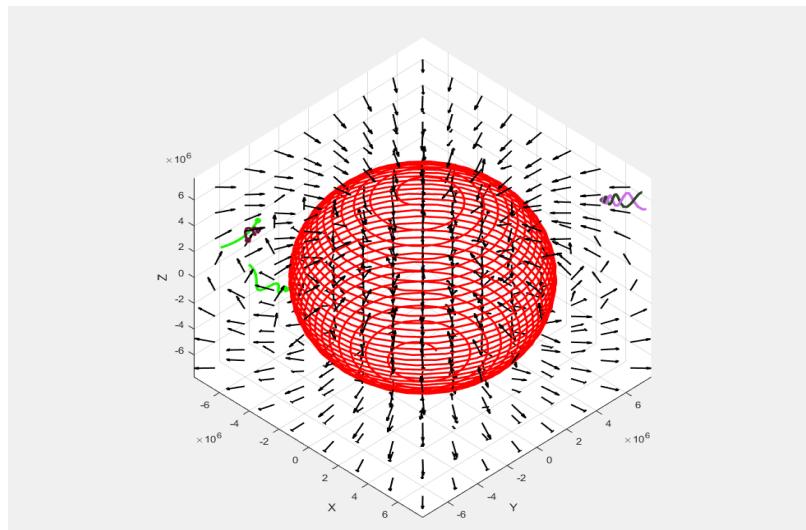
**Figura 36.** Vista lateral del campo magnético generado por la tierra, con 4 partículas con características de un protón que llegan del sol.

En la figura 36 se pueden observar 4 partículas con las características de un protón, es decir, tienen una masa equivalente a  $1.67 \times 10^{-27}$ , una carga de  $1.6 \times 10^{-19}$ , las velocidades y posiciones utilizadas para generar la trayectoria de las partículas fueron las siguientes:

**Tabla 1.** Valores de posición y velocidad para los protones dentro de la simulación

	Posición	Velocidad
Proton 1	$(6 \times 10^6 i + 6 \times 10^6 j + 6 \times 10^6 k) \text{ m}$	$(-0.1875 \times 10^9 i - 18.75 \times 10^9 j - 18.75 \times 10^9 k) \text{ m/s}$
Proton 2	$(-7 \times 10^6 i - 5 \times 10^6 j + 2 \times 10^6 k) \text{ m}$	$(0.6 \times 10^9 i + 1.6 \times 10^9 j + 1.67 \times 10^9 k) \text{ m/s}$
Proton 3	$(-7 \times 10^6 i + 2 \times 10^6 j) \text{ m}$	$(0.75 \times 10^9 i + 1.6 \times 10^9 j + 1.67 \times 10^9 k) \text{ m/s}$
Proton 4	$(-6 \times 10^6 i + 4 \times 10^6 j + 2 \times 10^6 k) \text{ m}$	$(0.6 \times 10^9 i + 1.6 \times 10^9 j + 1.67 \times 10^9 k) \text{ m/s}$

El movimiento que tiene la trayectoria de las partículas sigue las condiciones de la fuerza de Lorentz, además de seguir el movimiento que tienen los protones al estar cerca del campo magnético de la tierra, tienen un trayectoria que forma espirales alrededor del campo terrestre, otro movimiento que se puede observar en las partículas es acercarse, pero debido a la velocidad que llevan, son expulsadas por el campo magnético.



**Figura 37.** Vista lateral del campo magnético generado por la tierra, con 4 partículas con características de un electrón que llegan del sol.

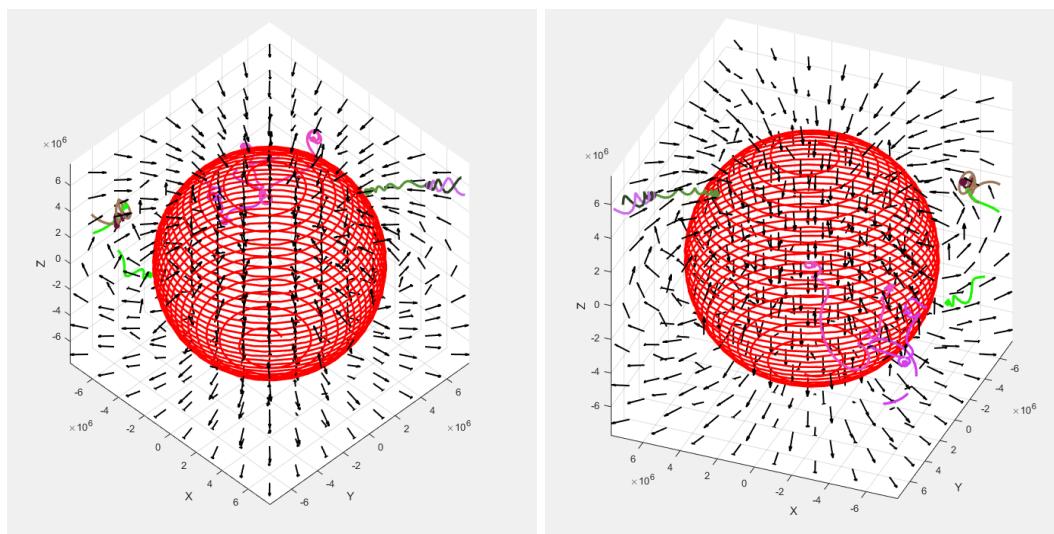
En la figura 37 se pueden observar 4 partículas con las características de un electrón, tienen una masa equivalente a  $9.11 \times 10^{-31}$ , una carga de  $-1.6 \times 10^{-19}$ , las velocidades y posiciones utilizadas para generar la trayectoria de las partículas fueron las siguientes:

**Tabla 2.** Valores de posición y velocidad para los electrones dentro de la simulación

	Posición	Velocidad
Electron 1	$(6 \times 10^6 i + 6 \times 10^6 j + 6 \times 10^6 k) \text{ m}$	$(-0.75 \times 10^6 i - 0.75 \times 10^6 j - 0.75 \times 10^6 k) \text{ m/s}$
Electron 2	$(6 \times 10^6 i + 6 \times 10^6 j + 6 \times 10^6 k) \text{ m}$	$(-0.1875 \times 10^6 i - 0.1875 \times 10^6 j - 0.1875 \times 10^6 k) \text{ m/s}$
Electron 3	$(-6 \times 10^6 i - 5 \times 10^6 j + 3 \times 10^6 k) \text{ m}$	$(1 \times 10^6 i + 0.1875 \times 10^6 j - 0.1875 \times 10^6 k) \text{ m/s}$
Electron 4	$(-6 \times 10^6 i - 3 \times 10^6 j - 3 \times 10^6 k) \text{ m}$	$(1 \times 10^6 i - 0.1875 \times 10^6 j - 0.1875 \times 10^6 k) \text{ m/s}$

El movimiento que tiene la trayectoria de las partículas sigue las condiciones de la fuerza de Lorentz, además de seguir el movimiento que tienen los electrones al estar cerca del campo magnético de la tierra, podemos observar una trayectoria en algunos de los electrones que forman espirales alrededor del campo terrestre, otras realizan movimiento oscilatorios, dando vueltas sobre determinada zona.

En la figura 38 podemos observar el movimiento de las 8 partículas en conjunto en una sola gráfica.



**Figura 38.** Vista lateral del campo magnético generado por la tierra, con las 8 partículas (electrones y protones) que llegan del sol.

## **Dispositivo Estudiado**

Los campos magnéticos son estudiados en diferentes disciplinas debido a su apoyo para diferentes técnicas de trabajo: en medicina suelen ser utilizadas para la proyección de resonancias magnéticas; se utiliza para innovar los medios de transporte con el fin de eliminar la fricción y mejorar la velocidad de los vehículos; en física se utiliza para investigar las altas energías, entre otros ejemplos (World Health Organization, 2016).

En cuanto a sus aplicaciones en medicina existen investigaciones interesantes, una de ellas es el reciente caso realizado por la Universidad de Guanajuato (UG), México. Los investigadores y estudiantes del Laboratorio de Superconductividad Aplicada de la UG están trabajando en prototipos de manipulación de campos magnéticos como terapias no invasivas para tratar el Alzheimer (Diario Milenio, 2021). El objetivo de esta investigación es evitar el desprendimiento de las moléculas que causan el daño cerebral a través de campos magnéticos, fomentando que las moléculas se queden adheridas a la célula con fuerzas magnéticas.

En cuanto a sus aplicaciones en el transporte, el magnetismo se ha utilizado como herramienta para evitar el uso de energías no renovables y contaminantes que alimentan los transportes. A través de la “levitación magnética” (fenómeno por el cual un material dado puede suspenderse gracias a la repulsión entre polos iguales de dos imanes) se han desarrollado trenes Maglev (Magnetic levitation), provocando que los trenes sean suspendidos en el aire. De acuerdo con el Instituto Mexicano Madero, los sistemas Maglev son ecológicos y veloces, ya que la energía empleada para su movimiento es eléctrica, su fuerza de propulsión es mínima y debido a la minoración de fricción se contrarresta el deterioro del tren (Instituto Mexicano de Madero, 2017).

Estas investigaciones serían imposibles sin el apoyo de simulaciones computacionales de campos magnéticos, ya que estas sirven para predecir el comportamiento de los campos en función de la corriente eléctrica en ambientes controlados. Por ello es de gran relevancia innovar los métodos de simulación, volviéndolos más eficaces, precisos y comprensibles.

## **Conclusión**

Los campos magnéticos y los fenómenos del electromagnetismo han sido observados en la naturaleza desde hace mucho tiempo, pero su estudio formal tiene un tiempo relativamente reciente. Podemos decir que nuestra civilización actual está en funcionamiento gracias a dichos estudios e innovaciones, ya que, como revisamos anteriormente, encontramos aplicaciones de los campos magnéticos e inducción electromagnética en muchísimas áreas, que van desde las más simples, como los motores eléctricos, hasta las más complejas y sofisticadas, como las imágenes por resonancia magnética o aceleradores de partículas.

Como fue estudiado en el presente trabajo, el campo magnético terrestre sirve de escudo protector contra las partículas cargadas de alta energía provenientes de nuestra estrella, el Sol, pues las desvía de impactar con la superficie de la Tierra, lo cual podría ser nocivo para la salud de las especies en la Tierra.

## Referencias

- Stewart, J. (2017). Cálculo de varias variables: trascendentes tempranas (8a. ed.). Cengage Learning. <https://0-elibro-net.biblioteca-ils.tec.mx/es/lc/consorcioitesm/titulos/93303>
- Marcos, M. P. (2009). Latitud. La verdadera historia del descubridor del magnetismo terrestre. (Spanish). Estudios Filosóficos, 58(168), 389–390
- Santamaría, J. J. V. (2016). La historia del campo magnético terrestre registrada en las rocas. Fundamentos del Paleomagnetismo. Enseñanza de las Ciencias de la Tierra, 24(3), 261-261.
- Lasfargues, P. (1995). Magnetismo en Geología y la prospección magnética terrestre. Instituto Politécnico Nacional.  
<https://0-elibro-net.biblioteca-ils.tec.mx/es/ereader/consorcioitesm/72209?page=26>
- Folguera, A. (2006). Introducción a la geología; el planeta de los dragones de piedra. Eudeba.  
<https://0-elibro-net.biblioteca-ils.tec.mx/es/ereader/consorcioitesm/101394?page=27>
- Braun, E. (2003). Electromagnetismo: de la ciencia a la tecnología (3a. ed.). FCE - Fondo de Cultura Económica.  
<https://0-elibro-net.biblioteca-ils.tec.mx/es/ereader/consorcioitesm/71968?page=19>
- World Health Organization. (2016, 4 agosto). Campos electromagnéticos y salud pública. Organización Mundial de la Salud.  
<https://www.who.int/peh-emf/publications/facts/fs299/es/>
- Diario Milenio. (2021, 23 abril). En UG exploran aplicaciones de campos magnéticos para atender Alzheimer. Milenio Diario Digital.  
<https://www.milenio.com/politica/comunidad/ug-exploran-campos-magneticos-atender-alzheimer>

Instituto Mexicano de Madero. (2017). *Magnetismo como energía viable en el transporte*.  
<https://dspace.umad.edu.mx/bitstream/handle/11670/288/4-CartelMagnetismo%20como%20energ%c3%ada%20viable%20en%20el%20transporte.pdf?sequence=1&isAllowed=y>

Campbell, J. (2019). Lorentz force. *Salem Press Encyclopedia of Science*.

García Hernández, A. (2015). *Ecuaciones diferenciales*. Grupo Editorial Patria.  
<https://0-elibro-net.biblioteca-ils.tec.mx/es/ereader/consorcioitesm/39438?page=39>

McLean, S. (s. f.). *Geomagnetism Frequently Asked Questions*. NOAA. Recuperado 5 de junio de 2021, de <https://www.ngdc.noaa.gov/geomag/faqgeom.shtml>

Nagy, T. A. (2020). Solar wind. *Salem Press Encyclopedia of Science*.