

MATLAB Electromagnetism simulations

Main file

```

Tierra3D.m  x  +
1 - clear; clc; clf; close all;
2 - n = 7;
3 - % Paulo Ogando, Erika Marlene Garcia, Jose Luis Madrigal
4 - % Maximiliano Carrasco, Alan Said Martinez, Christian Parrish Gutierrez
5 - %Parametrización
6 - syms s px py pz dpx dpy dpz
7 - disp('¿Que quieres graficar?');
8 - opcion = input('1. Línea  2. Anillo  3. Solenoide  4.- Solenoide esferico: ');
9 - if opcion == 1
10 -     [q,m,xinicial,vinicial,dominio,l]=basicas(n);
11 -     I = 1;
12 -     Mew = 4*pi*I;
13 -     Const = (Mew*I)/4*pi;
14 -     smin = -20; smax = 20;
15 -     px(s) = 0;
16 -     py(s) = 0;
17 -     pz(s) = s;
18 - end
19 - if opcion == 2
20 -     [q,m,xinicial,vinicial,dominio,l]=basicas(n);
21 -     delta = .01;
22 -     I = 1;
23 -     Mew = 4*pi*I;
24 -     Const = (Mew*I)/4*pi;
25 -     smin = 0; smax = 2*pi;
26 -     px(s) = cos(s);
27 -     py(s) = sin(s);
28 -     pz(s) = 0;
29 -
[posx,posy,posZ] = malladoT3D(n,dominio);
[Bx, By, Bz] = camposM3D(posx,posy,posZ,Const,px,py,pz,dpx,dpy,dpz,doms);
[BxN, ByN, BzN,Mx] = NormalizarB(Bx,By,Bz,l);
[posiciones] = particulas(q, m, xinicial, vinicial, delta, Const,px,py,pz,dpx,dpy,dpz,doms);
GraficarB(posx,posy,posZ,BxN,ByN,BzN,Mx,px,py,pz,doms,posiciones);

```

```

Tierra3D.m  x  +
function [q,m,xinicial,vinicial,dominio,l]=basicas(n)

- xmin = -2; xmax = 2;
- ymin = -2; ymax = 2;
- zmin = -2; zmax = 2;
- dominio = [xmin,xmax,ymin,ymax,zmin,zmax];
- l = (xmax-xmin)/(2*n);
- q = 1;
- m = 1;
- Xi = 0;
- Xj = 0;
- Xk = 0;
- xinicial = [Xi,Xj,Xk];
- Vi = 1;
- Vj = 1;
- Vk = 1;
- vinicial = [Vi,Vj,Vk];
- end

```

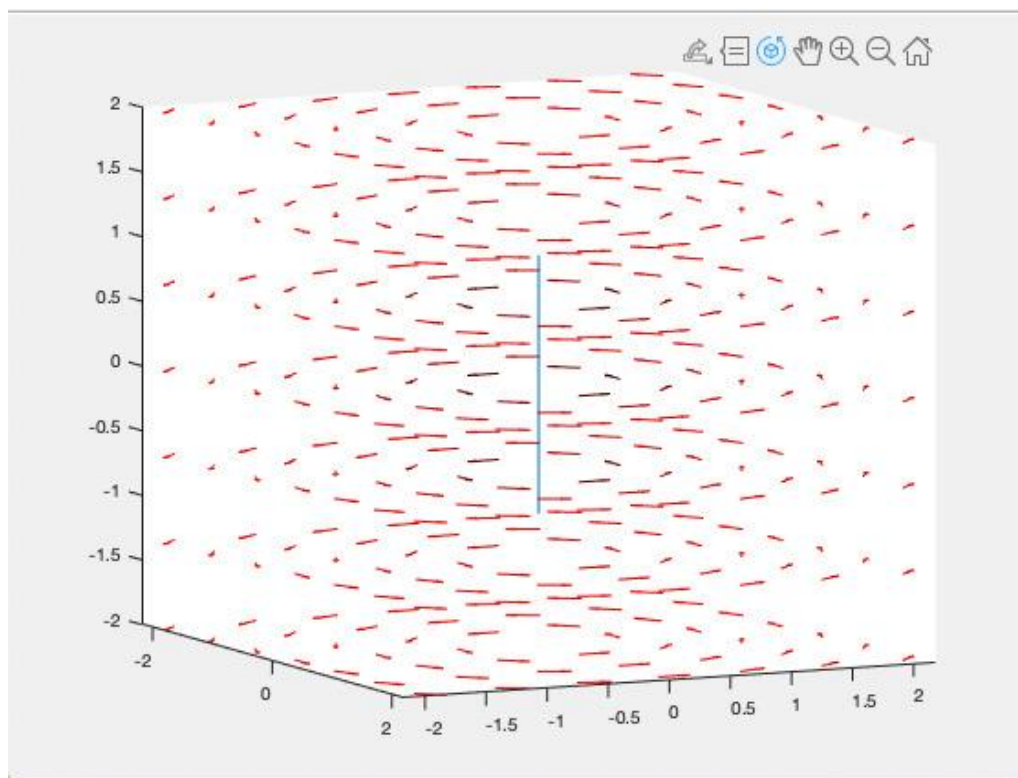
Planet data

```
Tierra3D.m  x +
- if opcion == 4
-     xmin = -7000000; xmax = 7000000;
-     ymin = -7000000; ymax = 7000000;
-     zmin = -7000000; zmax = 7000000;
-     dominio = [xmin,xmax,ymin,ymax,zmin,zmax];
-     l = (xmax-xmin)/(2*n);
-     q = -1.6*10^-19;
-     m = 9.11*10^-31;
-     Xi = 300000;
-     Xj = 149000;
-     Xk = 210000;
-     xinicial = [Xi,Xj,Xk];
-     Vi = 400000;
-     Vj = 400000;
-     Vk = 400000;
-     vinicial = [Vi,Vj,Vk];
-     delta = .000000525;
-     I = 12*10^5;
-     Mew = 4*pi*10^-7;
-     Const = (Mew*I)/4*pi;
-     smin = -99; smax = 99;
-     px(s) = 6371000*(sqrt(1-(0.01*s)^2).*cos(s));
-     py(s) = 6371000*(sqrt(1-(0.01*s)^2).*sin(s));
-     pz(s) = 6371000*(0.01.*s);
- end
```

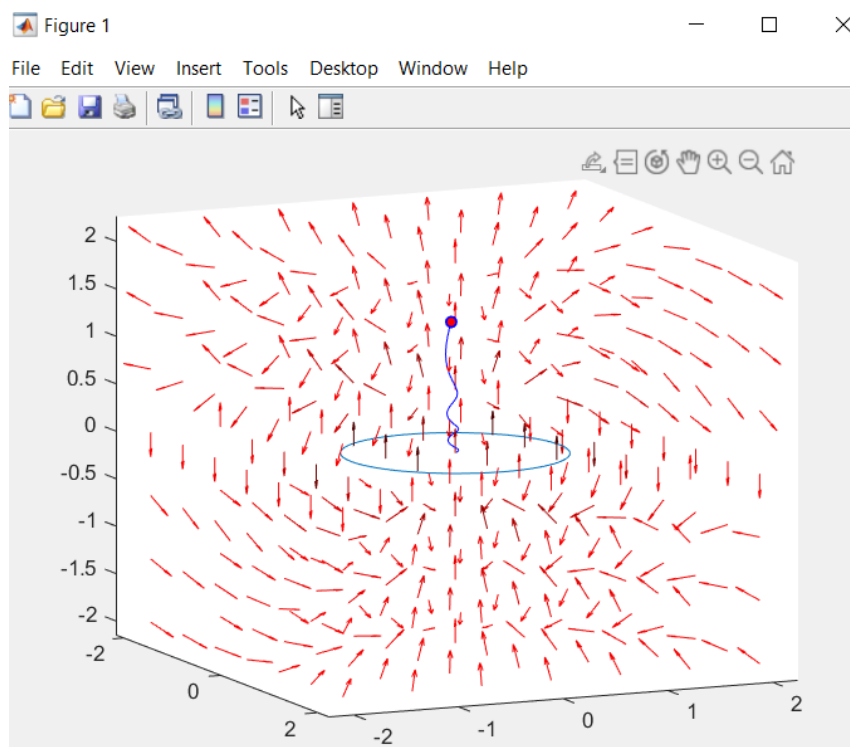
File of one function: Biot Savart

```
BiotSavart.m  x +
1 function [Bx, By, Bz] = BiotSavart(Const,posx,posy,posZ,px,py,pz,dpx,dpy,dpz,doms)
2 - syms s
3 - %Vector r
4 - rx = posx - px;
5 - ry = posy - py;
6 - rz = posZ - pz;
7
8 - %Magnitud del vector r
9 - Magr = (rx.^2 + ry.^2 + rz.^2).^(3/2);
10
11 - %Producto cruz
12 - Cruzei = (dpy*rz - ry*dpz);
13 - Cruzej = -(dpx*rz - rx*dpz);
14 - Cruzek = (dpx*ry - rz*dpy);
15
16 - %Integrando
17 - dBx(s) = Cruzei / Magr;
18 - dBy(s) = Cruzej / Magr;
19 - dBz(s) = Cruzek / Magr;
20
21 - %Integracion
22
23 - Bx = Const * TrapezoidaTierra(dBx,20,doms(1),doms(2));
24 - By = Const * TrapezoidaTierra(dBy,20,doms(1),doms(2));
25 - Bz = Const * TrapezoidaTierra(dBz,20,doms(1),doms(2));
```

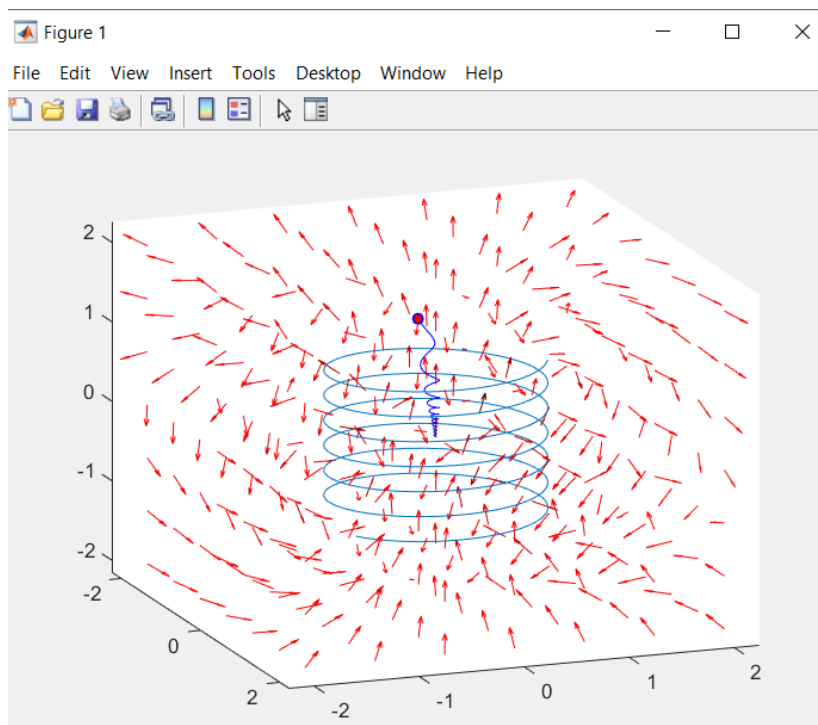
Option 1: Current line (we couldn't get the correct delta to see the trajectory, so this is a screenshot of the first deliverable, just the magnetic field)



Option 2: Ring (remember this is an animation)



Option 3: Solenoid



Option 4: Spherical Solenoid (Earth)

