

Tecnológico de Monterrey

Campus Querétaro

**Modelación computacional de sistemas electromagnéticos**

**Grupo 2**

**Actividad: Simulador simbólico de fuerza magnética**

**Actividad Cuestionario magnetismo**

Maestro

Ernesto Momox Beristain

Presenta

Pablo César Jiménez Villeda A01703517

Fecha de entrega: 17 de mayo del 2021

CÓDIGO DE MATLAB Y FUNCIONAMIENTO COMENTADO:

%SIMULADOR DE FUERZA DE LORENTZ - A01703517 - Pablo César Jiménez Villeda

fprintf("Bienvenido al simulador de Fuerza de Lorentz, a continuación ingrese los datos iniciales de su simulación. \n\n")

%El usuario ingresa las condiciones inciales

Q = input("Ingrese el valor de la carga en Coulombs: ");

m = input("Ingrese el valor de la masa de la carga en Kilogramos: ");

%Pedimos al usuario los vectores con las condiciones iniciales

Vo = input("Ingrese el vector de velocidad inicial con el siguiente formato [Vx Vy Vz] en m/s: ");

B = input("Ingrese el vector de la densidad del flujo magnético [Bx By Bz] en T: ");

R = input("Ingrese el vector de la posición inicial de la carga [X Y Z] en m: ");

%El usuario define en qué tiempo desea evaluar el sistema

tiempo = input("Ingrese el tiempo en segundos en el que desea evaluar el sistema: ");

syms Vx(t) Vy(t) Vz(t) t;

%Creamos el vector V a partir de sus componentes simbólicas

V = [Vx(t) Vy(t) Vz(t)];

%De la segunda ley de Newton sabemos que F = ma y que a es igual a dv/dt,

%asimismo, la fuerza de lorentz es F = Q(VxB), al igualar ambas expresiones

%obtenemos que F = m\*dv/dt = Q(VxB), por ende podemos definir a dv/dt como

%dv/dt = (Q/m)\*(VXB)

%dado que dV/dt = Q/m \* V x B

dvdt = (Q / m) \* cross(V, B);

%la función cross() retorna el preducto cruz de dos vectores

%CALCULAMOS velocidad)

%Dado que dvdt es igual a d/dt(Vxax + Vyay + Vzaz) podemos generar

%igualdades que nos permitan obtener un sistema de ecuaciones comparandolo

%con el resultado obtenido en dvdt

%Guardamos en ec1 el valor de la diferencial de Vx correspondiente a la

%dirección ax y generamos una ecuación que representa la condición inicial

%de la velocidad en esa dirección

ec1 = diff(Vx) == dvdt(1,1);

cond1 = Vx(0) == Vo(1,1);

%Guardamos en ec1 el valor de la diferencial de Vy correspondiente a la

%dirección ay y generamos una ecuación que representa la condición inicial

%de la velocidad en esa dirección

ec2 = diff(Vy) == dvdt(1,2);

cond2 = Vy(0) == Vo(1,2);

%Guardamos en ec1 el valor de la diferencial de Vz correspondiente a la

%dirección az y generamos una ecuación que representa la condición inicial

%de la velocidad en esa dirección

ec3 = diff(Vz) == dvdt(1,3);

cond3 = Vz(0) == Vo(1,3);

%Generamos vectores con las ecuaciones y condiciones

ecs = [ec1 ec2 ec3];

conds = [cond1, cond2, cond3];

%Resolvemos el sistema de ecuaciones y obtenemos los valores de las

%velocidades en función del tiempo gracias a dsolve, que recibe como

%argumento un vector de ecuaciones diferenciales y uno de condiciones

[Vx(t), Vy(t), Vz(t)] = dsolve(ecs, conds);

%Creamos un vector velocidad en función del tiempo con las componentes

%obtenidas

V(t) = [Vx(t) Vy(t) Vz(t)];

%CALCULAMOS aceleracion)

%Definimos las ecuaciones simbólicas de los componentes de la aceleración

syms ax(t) ay(t) az(t)

%Se sabe que la aceleración es la razón de cambio de la velocidad, por lo

%que derivando los componentes de velocidad obtenemos los de aceleración

%Obtenemos primera derivada de los componentes de velocidad para definir la

%aceleracion

ax(t) = diff(Vx, 1);

ay(t) = diff(Vy, 1);

az(t) = diff(Vz, 1);

%Creamos un vector de velocidad a partir del tiempo

a(t) = [ax(t) ay(t) az(t)];

%CALCULAMOS fuerza)

%dado que F = Q(VxB) --> ya que no hay campo eléctrico

F(t) = Q\*(cross(V(t), B));

%CALCULAMOS energía cinética)

%Dado que Ek = 1/2mV^2

E(t) = (1/2)\*m\*(dot(V(t), V(t)));

%Así ^ obtenemos una ecuación simbólica para calcular la energía en función

%del tiempo

%TEST

% t = 4;

% subs(E(t)) ----> devuelve 9 ya que la energía es constante al no haber

% trabajo

%CALCULAMOS posicion)

syms x(t) y(t) z(t) C1 C2 C3

%Sabemos que la posición se puede obtener a partir de una integral de la

%velocidad, por lo que para obtener la posicion integraremos los

%componentes de la velocidad y calcularemos la constante de integración

%obtenemos componente X de la posicion a partir de la integración de Vx

intX = int(Vx(t), t) ;

X\_t = intX + C1;

%Con la función solve podemos despejar al segundo argumento del primero que

%sería una ecuación en la que con subs evaluamos la protofunción X\_t en 0

C1 = solve(subs(X\_t, 0) == R(1,1), C1);

%Generamos la función final del componente sumando el resultado de la

%integral y la constante de integración

x(t) = intX + C1;

%Repetimos el proceso con los demás componentes:

%obtenemos componente Y de la posicion

intY = int(Vy(t), t) ;

Y\_t = intY + C2;

C2 = solve(subs(Y\_t, 0) == R(1,2), C2);

y(t) = intY + C2;

%obtenemos componente Z de la posicion

intZ = int(Vz(t), t) ;

Z\_t = intZ + C3;

C3 = solve(subs(Z\_t, 0) == R(1,3), C3);

z(t) = intZ + C3;

%Definimos el vector pos(t) a partir de los componentes de posición

pos(t) = [x(t) y(t) z(t)];

%Desplegamos los componentes del sistema a partir del tiempo dado por el

%usuario usando la función subs para sustituir el tiempo dado en cada

%ecuación simbólica

fprintf("\nEn el tiempo de %i segundos los componentes del sistema son: ", tiempo);

posicion = subs(pos(t), tiempo);

fprintf("\n La carga se encuentra en: (%.3f, %.3f, %.3f)m", posicion);

velocidad = subs(V(t), tiempo);

fprintf("\n La carga tiene una velocidad de: (%.3f, %.3f, %.3f)m/s", velocidad);

aceleracion = subs(a(t), tiempo);

fprintf("\n La carga tiene una aceleracion de: (%.3f, %.3f, %.3f)m/s^2", aceleracion);

fuerza = subs(F(t), tiempo);

fprintf("\n La carga tiene una fuerza magnetica de: (%.3f, %.3f, %.3f) N", fuerza);

energia = subs(E(t), tiempo);

fprintf("\n La carga tiene una energia cinetica de: %.3f J \n", energia);

fprintf("Por favor espere mientras generamos la simulación de la trayectoria de la carga... ");

%definimos el vector del tiempo usando linspace, como se puede ver el

%tiempo mostrado en la simulación será de 15 segundos

T = linspace(0, 15, 1500);

%Sustituimos los valores de T en las funciones de posición y luego hacemos

%la conversión a valores del tipo double para poder desplegarlos en la

%gráfica, repetimos esto con las demás ecuaciones simbólicas

posX = double(subs(x, t, T));

posY = double(subs(y, t, T));

posZ = double(subs(z, t, T));

%Sustituimos los valores de T en las funciones de velocidad

velX = double(subs(Vx, t, T));

velY = double(subs(Vy, t, T));

velZ = double(subs(Vz, t, T));

%Sustituimos los valores de T en las funciones de aceleración

acX = double(subs(ax, t, T));

acY = double(subs(ay, t, T));

acZ = double(subs(az, t, T));

%Sustituimos los valores de T en la funcion de fuerza

force = double(subs(F, t, T));

%Sustituimos los valores de T en la funcion de energía

energy = double(subs(E, t, T));

%Definimos que queremos que se trace una trayectoria a partir de los

%componentes de posición

plot3(posX', posY', posZ', "b");

%Determinamos que queremos que se mantengan los elementos gráficos

hold on;

%Título y etiquetas de la gráfica:

title("Trayectoria de la carga");

ylabel("m");

xlabel("m");

zlabel("m")

%Con la ayuda de un ciclo for iteramos el vector con todos los tiempos para

%animar la posición de la carga en cada momento

for i=1:length(T)

plot3(posX(1,i),posY(1,i),posZ(1,i),'\*r');

pause(0.1);

%Creamos cadenas de texto que desplieguen los componentes de cada

%variable en tiemp real

tiempo = "t: " + T(1,i) + "s";

actualPosition = "X: " + posX(1,i) + "m " + "Y: " + posY(1,i) + "m " + "Z: " + posZ(1,i) + "m ";

actualVelocity = "Vx: " + velX(1,i) + "m/s " + "Vy: " + velY(1,i) + "m/s " + "Vz: " + velZ(1,i) + "m/s ";

actualAcceleration = "Ax: " + acX(1,i) + "m/s^2 " + "Ay: " + acY(1,i) + "m/s^2 " + "Az: " + acZ(1,i) + "m/s^2 ";

actualForce = "Fx: " + force(1,i) + " N " + "Fy: " + force(1,1500+i) + " N " + "Fz: " + force(1,3000+i) + " N ";

actualEnergy= "Ek: " + energy(1,i) + " J";

%Definimos una leyenda que muestre los valores de tiempo, posición, velocidad,

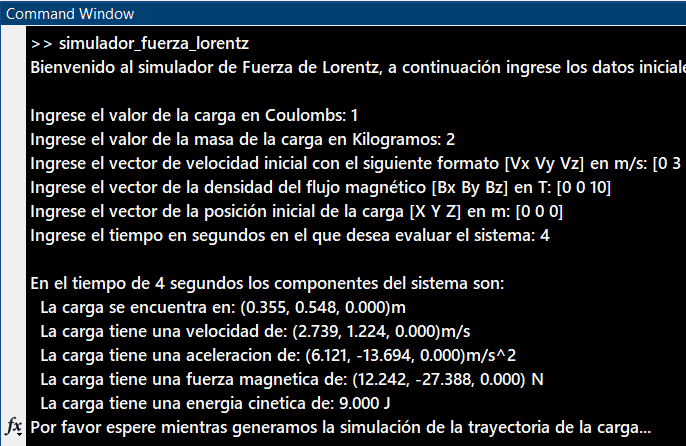
%aceleración, fuerza y energía en tiempo real

legend("Trayectoria", tiempo, actualPosition, actualVelocity, actualAcceleration, actualForce, actualEnergy);

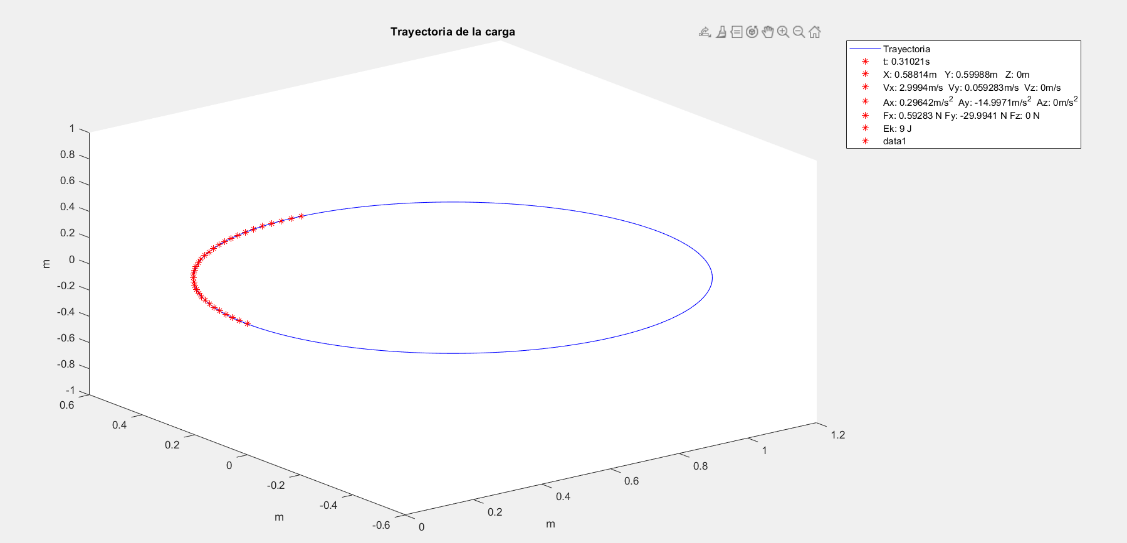
end

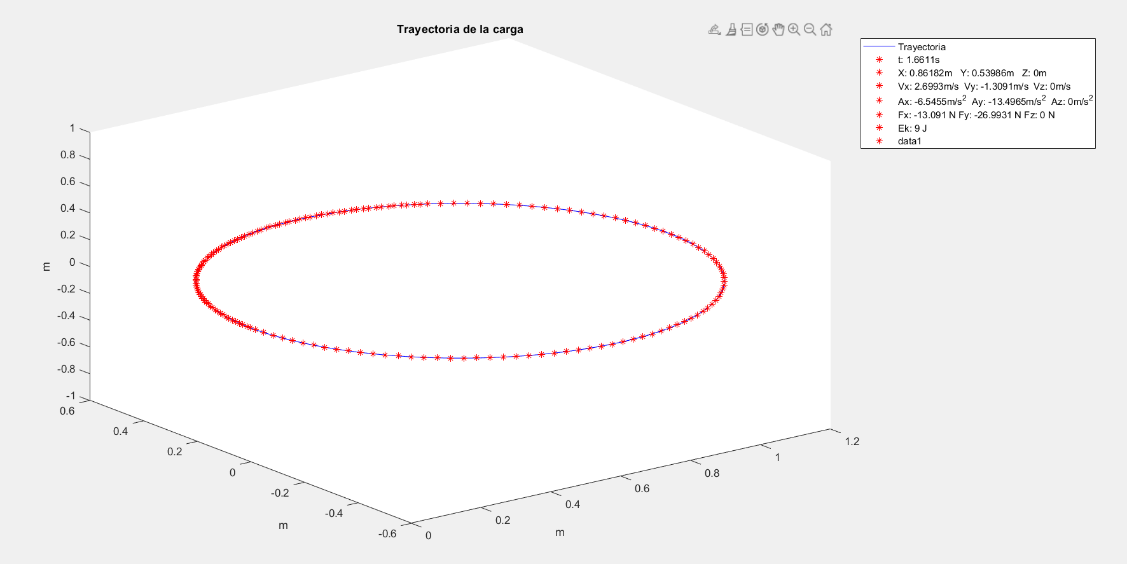
CAPTURAS DE PANTALLA DEL RESULTADO

* Prueba con el mismo planteamiento de la clase:

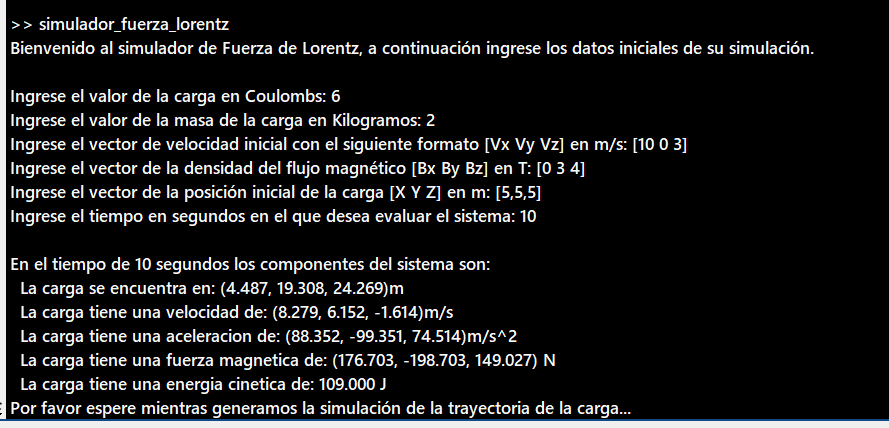


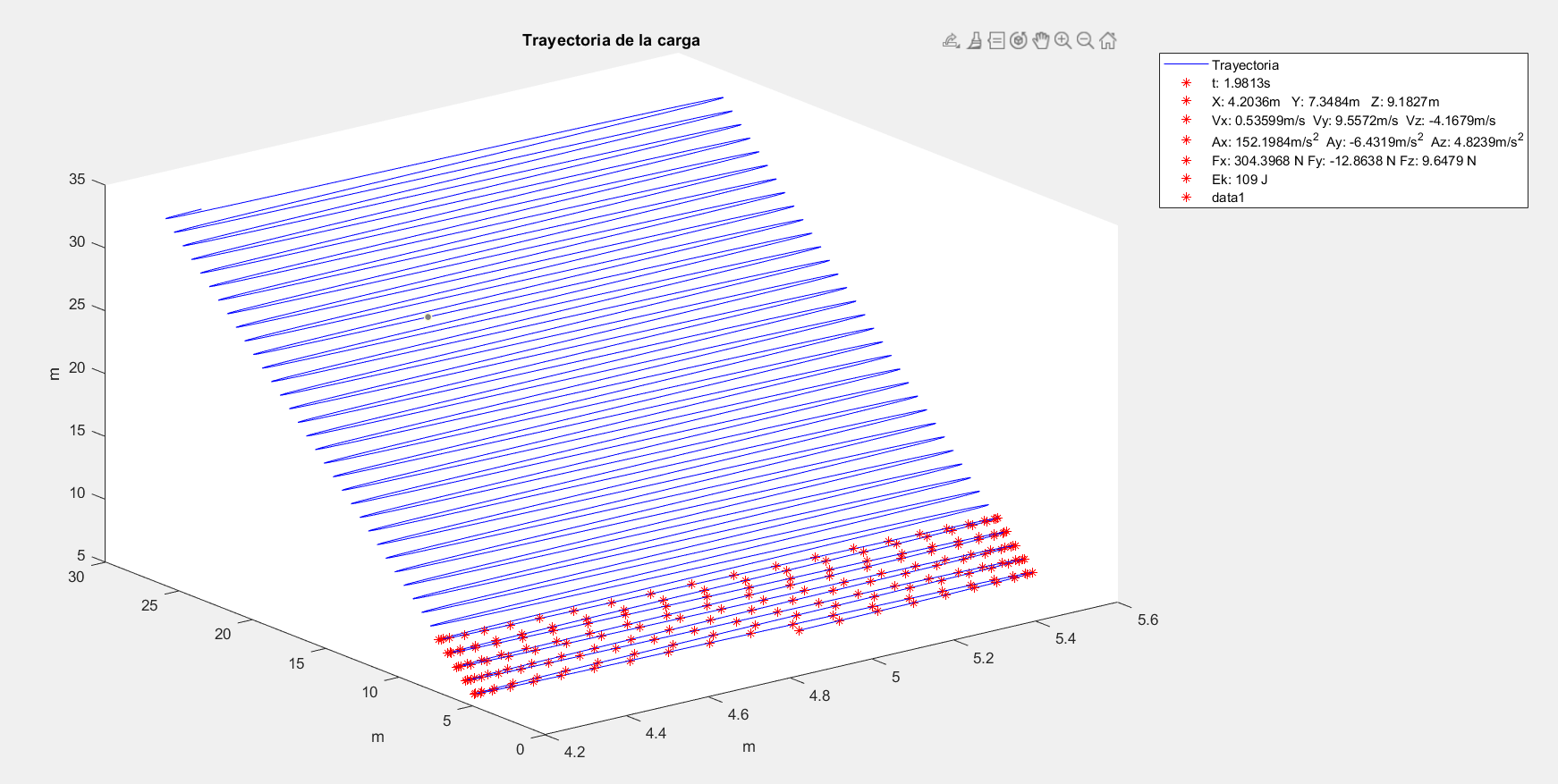
* Simulación:





* Prueba con planteamiento alternativo 1:





* Prueba con planteamiento alternativo 2:

