EM Bonus Homework

Benedict Lee

April 2018

In this document, I will explain the general approach behind the algorithm.

1 Outline of the Approach

For this homework, the only two references used were the ones given in the instructional document, namely the explanatory notes by Cooper and the chapter on Penning trap by Harvard. However, almost the entire algorithm was written from scratch by myself.

To find the trajectory of the particle, the following approach is taken:

1. Find the E field numerically
2. Find the the position of the particle at the next time step.

2 Finding E field

The first step of the algorithm is to find the E field of the system. While the E field can be expressed in an analytically form by differentiating the potential, I wanted to be able to compute any arbitrary potential. Thus, I adopted a numerical approach.

Let $\vec{x}$ be the position vector, and $V(x, y, z)$ be the potential function of the system, by definition, the $\vec{E}$ field is given by:

$$\begin{aligned}\vec{E} &= \nabla V(\vec{x}) \\ &= \begin{bmatrix} \frac{\partial V}{\partial x} \\ \frac{\partial V}{\partial y} \\ \frac{\partial V}{\partial z} \end{bmatrix}\end{aligned}$$

If we look at the x-component of the $\vec{E}$ field, the partial derivative is defined as such:

$$\frac{\partial V(x, y, z)}{\partial x} = \lim_{\Delta x \to 0} \frac{V(x + \Delta x, y, z) - V(x, y, z)}{\Delta x}$$

The above equation gives the right handed limit. Since $V(\vec{x})$ is assumed to be differentiable, the left and right handed limit are equal. Furthermore, since the $V(\vec{x})$ is differentiable, the limit exists. Thus:

$$\begin{aligned}E_x &= \frac{\partial V}{\partial x} \\ &\approx \frac{V(x + \Delta x, y, z) - V(x, y, z)}{\Delta x}, \text{ For sufficiently small } \Delta x\end{aligned}$$

To do this numerically, we define ϵ to be a very small number, in this case $\epsilon = 1 \times 10^{-9}$. Then:

1. Find $V(\vec{x})$, call this V
2. Shift $\vec{x} \to \vec{x}'$ by shifting one component of $\vec{x}$ by ϵ.
3. Compute $V(\vec{x}')$, call this pV
4. The $\vec{E}$ field in that direction is the gradient $(V - pV)/\epsilon$
5. Repeat for the other two components

The code is shown below.

```
function [Ex,Ey,Ez]=efield(x,y,z)
%This function calulates the E field of the Penning trap given any
%coordinate (x,y,z)
global V0 d
s=sqrt(x.^2+y.^2);
V=(V0./2).*(z.^2-(s.^2)./2)./(d.^2);
for i=1:3
    eps=1E-9; %let epsilon be some very small number
    if i==1
        snew=sqrt((x+eps).^2+y.^2); %we first calculate the positive change in x
        pV=(V0/2).*(z.^2-(snew.^2)/2)./(d.^2); %change in potential
        Ex=[-(pV-V)./(eps)]; %E field is approx the gradient at that point if
   eps is small enough
    elseif i==2
        snew=sqrt((x).^2+(y+eps).^2);
        pV=(V0/2).*(z.^2-(snew.^2)/2)./(d.^2);
        Ey=[-(pV-V)./(eps)];

        else
        pV=(V0/2).*((z+eps).^2-(s.^2)/2)./(d.^2);
        Ez=[-(pV-V)./(eps)];
    end
end
end
```

3 Finding the Dynamics

We are interested in finding the dynamics of the system for some length of time, call it Δt. To discretise the system, we have to define an interval of time to segment Δt into finitely many parts, call this time step t. We call the total number of segmented parts N. Of course, as $t \to 0$, we will get back the continuous case.

Let us suppose that we know the position of the particle at some instance in Δt. We call this position $\vec{x}[n]$, $1 < n < N$. We can also calculate the $\vec{E}$ field at this position, denoted as $\vec{E}[n]$, as well as $\vec{B}$ field, denoted as $\vec{B}[n]$. We can make use of finite difference method to find the next positon, $\vec{x}[n+1]$. To do that, we use the definition of acceleration and velocity given by Cooper.

$$\vec{a}[n] = \frac{\vec{x}[n+1] - 2\vec{x}[n] + \vec{x}[n-1]}{t^2}$$
$$\vec{v}[n] = \frac{\vec{x}[n+1] - \vec{x}[n-1]}{2t}$$

From Lorentz force, we know that:

$$m\vec{a} = q(\vec{E} + \vec{v} \times \vec{B})$$

We will now derive the equation to find $\vec{x}[n+1]$.
Let $q' = q/m$, we can rewrite Lorentz force using the finite difference definitions.

$$\vec{a}[n] = q'\left(\vec{E}[n] + \vec{v}[n] \times \vec{B}[n]\right)$$
$$\frac{\vec{x}[n+1] - 2\vec{x}[n] + \vec{x}[n-1]}{t^2} = q'\left(\vec{E}[n] + \frac{\vec{x}[n+1] - \vec{v}[n-1]}{2t} \times \vec{B}[n]\right)$$

Using the distributive property of cross product we can gather the $[n+1]$ terms on the left:

$$\vec{x}[n+1] - \frac{q't}{2}\vec{x}[n+1] \times \vec{B}[n] = q't\left(t\vec{E}[n] - \frac{1}{2}\vec{x}[n-1] \times \vec{B}[n]\right) + 2\vec{x}[n] - \vec{x}[n-1]$$

The right hand side of the equation is a known vector, since $[n]$ and $[n-1]$ values are known. We call this $X1$ Evaluating the left hand side yields 3 equations. For brevity, we omit the $[n+1]'s$ for this section. Written in (x, y, z):

$$\begin{bmatrix} x & -\frac{q't}{2}B_z[n]y & \frac{q't}{2}B_y[n]z \\ \frac{q't}{2}B_z[n]x & y & -\frac{q't}{2}B_x[n]z \\ -\frac{q't}{2}B_y[n]x & \frac{q't}{2}B_x[n]y & z \end{bmatrix} = \begin{bmatrix} 1 & -\frac{q't}{2}B_z[n] & \frac{q't}{2}B_y[n] \\ \frac{q't}{2}B_z[n] & 1 & -\frac{q't}{2}B_x[n] \\ -\frac{q't}{2}B_y[n] & \frac{q't}{2}B_x[n] & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Call the matrix on the right hand side A. To find the $\vec{x}[n+1]$, we simply have to:

$$\vec{x}[n+1] = A^{-1}X1$$

The code is shown below:

```
function x=next(t,B,E,q,m,xn,x1)
%This function calculates the next position of the particle given the
%current one. It is derived from the finite difference method by Ian Cooper
q=q/m;
X1=q.*t.*(t.*E-0.5.*cross(x1,B))+2.*xn-x1;
c=q*t/2;
A=[1 -B(3)*c B(2)*c; B(3)*c 1 -B(1)*c; -B(2)*c B(1)*c 1];
x=(inv(A)*X1')';
   end
```

One point to note is that we require $n > 1$. This is because we require $\vec{x}[n-1]$ to determine the next position. However, the time step for this code is generally very small. We can approximate the first time step as if there were no acceleration. Thus, the following algorithm is used to find the dynamics:

```
for i=1:length(t)
    [Ex,Ey,Ez]=efield(x(i,1),x(i,2),x(i,3));
    if i==1 %assume a=0 for first time step
        x(i+1,:)=v0.*tstep+x(i,:);
    else
        E=[Ex, Ey, Ez];
        [X]=next(tstep,B,E,q,m,x(i,:),x(i-1,:)); %find the next position
        x(i+1,:)=X;
    end
end
```

To plot the trajectories, we simply use the `drawnow` function in MATLAB. From the algorithm above, the position $[n+1]$ is appended to a matrix called `x`. In this $N \times 3$ matrix, each row corresponds to the position of the particle at that time. Thus, we will plot a section of this matrix, refresh the plot, and plot the next section. The code is shown below.

```
N=length(t);
i=1;
rate=50; %This controls how many points to plot per refresh

axis([min(x(:,1)) max(x(:,1)) min(x(:,2)) max(x(:,2)) min(x(:,3)) max(x(:,3))])
xlabel('x')
ylabel('y')
zlabel('z')
while i~=N-rate
    P=plot3([x(i:i+rate,1)],[x(i:i+rate,2)],[x(i:i+rate,3)],'r','displayname','
    Trajectory');
    if P1~=0 %in case E field is not plotted
        legend([P,P1])
    else
        legend(P)
    end
    i=i+rate;
    drawnow
    pause(0.001)
end
```

4 Frequencies and Stability

When a particle is trapped in a Penning Trap, it does three things.

1. Oscillates up and down z-axis with a frequency ω_z
2. Osciilates around z-axis with a frequency ω_m. This is known as the magnetron frequency.
3. Oscillates in a circular motion due to the B field with a frequency $\omega_{c'}$. This is known as the cyclotron frequency.

From the reference notes from Harvard, we can calculate these frequencies as such:

$$\omega_c = \frac{|e\vec{B}|}{m}$$

This is the original cyclotron frequency. However, this value changes in the presence of an $\vec{E}$ field.

To find the axial frequency:

$$\omega_z = \sqrt{\frac{eV_0}{md^2}}$$

Next we find the magnetron and modified cyclotron frequency.

$$\begin{aligned}\omega_{c'} &= \omega_c - \omega_m \\ &= \omega_c - \frac{\omega_z^2}{2\omega_{c'}} \\ \therefore 2\omega_{c'}^2 - 2\omega_c\omega_{c'} + \omega_z^2 &= 0\end{aligned}$$

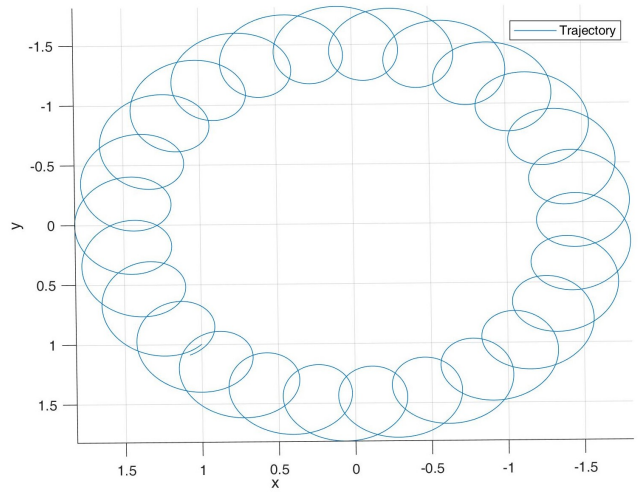
There are two possible $\omega_{c'}$'s. It happens that the lower value corresponds to the slower magnetron frequency, ω_m. More importantly, for the trap to be stable, the following condition must hold:

$$\omega_z \leq \frac{\omega_c}{\sqrt{2}}$$

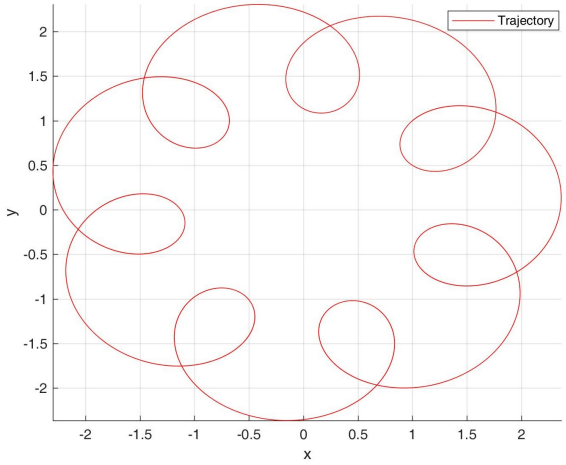
This condition is checked to ensure that the particles are bounded. The rest of the frequencies are printed so that one can cross check with the trajectories.

```
%in this section, the frequencies are calculated
wc=norm(q.*B)/m; %original cyclotron freq
wz=sqrt(q.*V0/(m*d^2)) %Axial freq
wm=-((-2*wc)+sqrt((2*wc)^2-4*(2)*(wz)^2))/4 %magnetron freq
wcnew=-((-2*wc)-sqrt((2*wc)^2-4*(2)*(wz)^2))/4 %new cyclotron fre

stab=wc/sqrt(2)-wz;

if stab<0
    fprintf('The charged particle will not be bounded\n')
    %check that the parameters do indeed result in a trap, else end the
    %code
    return
end
```

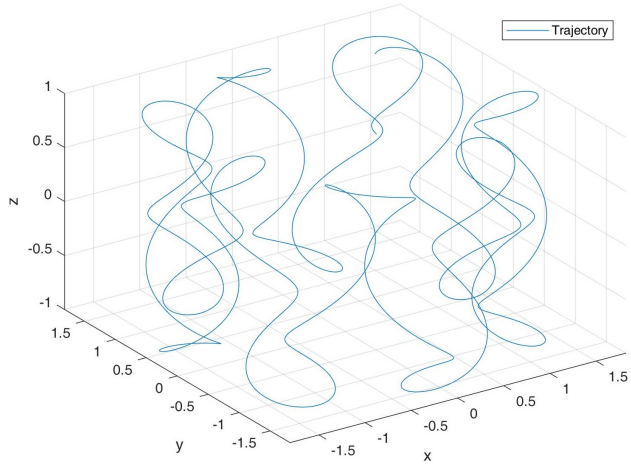
5 Sample Plots



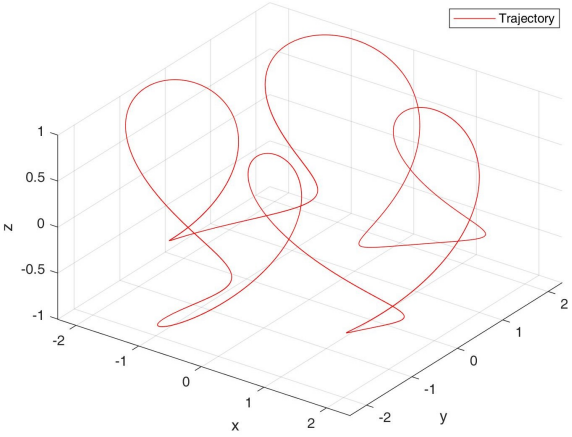
(a)



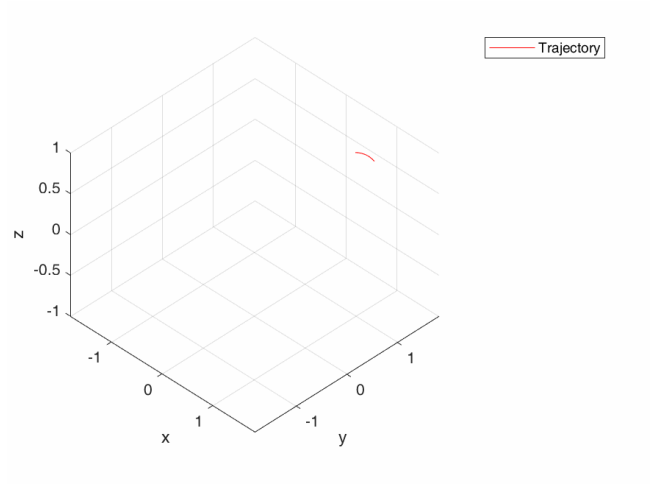
(b)



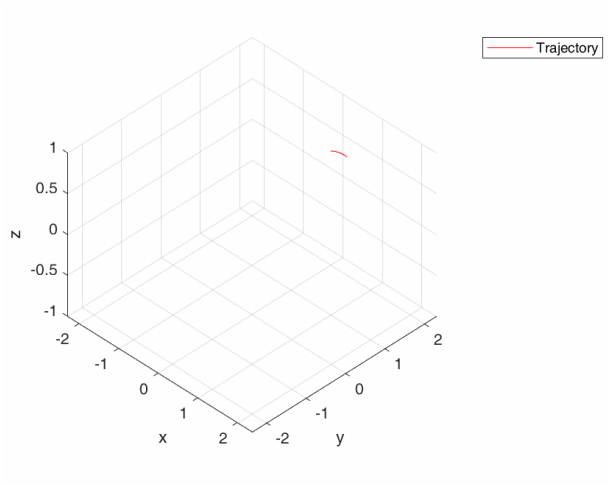
(c)



(d)



(e)



(f)

In the above diagram, (a) and (b) are the top view of the trajectories, (c) and (d) are the corner view of the trajectories and (e) and (f) are the animations of the trajectories (may not play when printed). The left column corresponds to the initial condition of:

$$q = 1.6 \times 10^{-19}$$
$$m = 1.6 \times 10^{-27}$$
$$x_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$
$$v_0 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \times 10^{7}$$
$$\vec{B} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} \times 10^{-1}$$
$$V_0 = 1 \times 10^{6}$$
$$\text{Length of the trap} = 2$$

While the plot on the right has all the parameters equal except:

$$\vec{B} = \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix} \times 10^{-1}$$

6 MATLAB Code

```
%This part is the initialisation of the code
clear all %clear everything in matlab
clf
btm=-1; %specify the dimension of the trap. The trap is modelled to be a cube
top=1;
h=(top-btm)/10;

%Initialise particle's properties
q=1.6E-19; %charge of the particle of interest
m=1.6E-27; %mass
x0=[1,1,1]; %initial position
v0=[1,1,0].*1E7; %initial velocity

%Initialise the trap's parameter
B=[0 0 3].*1E-1; %B field
global V0 d
V0=1E6; %Max potential
z0=top;
p0=top;
d=0.5*(z0^2+(p0^2)/2);%characteristic trap dimension

%in this section, the frequencies are calculated
wc=norm(q.*B)/m; %original cyclotron freq
wz=sqrt(q.*V0/(m*d^2)) %Axial freq
wm=-((-2*wc)+sqrt((2*wc)^2-4*(2)*(wz)^2))/4 %magnetron freq
wcnew=-((-2*wc)-sqrt((2*wc)^2-4*(2)*(wz)^2))/4 %new cyclotron fre

stab=wc/sqrt(2)-wz;

if stab<0
    fprintf('The charged particle will not be bounded\n')
    %check that the parameters do indeed result in a trap, else end the
    %code
    return
end


%This part calculate the dynamics
tstep=1E-9;  %time step (try to keep it around the same order of magnitude or
   the system will diverge)
tEnd=3.4E-6; %End time
t=[0:tstep:tEnd];
x=[x0];
v=[v0];
for i=1:length(t)
    [Ex,Ey,Ez]=efield(x(i,1),x(i,2),x(i,3));
    if i==1 %assume a=0 for first time step
        x(i+1,:)=v0.*tstep+x(i,:);
    else
        E=[Ex, Ey, Ez];
        [X]=next(tstep,B,E,q,m,x(i,:),x(i-1,:)); %find the next position
        x(i+1,:)=X;
    end
end

%Plotting
P1=0; %initialise P1
[rx,ry,rz]=meshgrid([min(x(:,1)):(max(x(:,1))-min(x(:,1)))/10:max(x(:,1))], [min
   (x(:,2)):(max(x(:,2))-min(x(:,2)))/10:max(x(:,2))] ,[min(x(:,3)):(max(x(:,3))
   -min(x(:,3)))/10:max(x(:,3))]); %initialise the vector space

```

```
%This part plots the E field to help visualise the system. It can be
%removed without affecting the code!
%P1=plotE(rx,ry,rz);
hold on


%Animation!!!
N=length(t);
i=1;
rate=20; %This controls how many points to plot per refresh

axis([min(x(:,1)) max(x(:,1)) min(x(:,2)) max(x(:,2)) min(x(:,3)) max(x(:,3))])

grid on
xlabel('x')
ylabel('y')
zlabel('z')

while i~=N-rate
    P=plot3([x(i:i+rate,1)],[x(i:i+rate,2)],[x(i:i+rate,3)],'r','displayname','
    Trajectory');
    if P1~=0 %in case E field is not plotted
        legend([P,P1])
    else
        legend(P)
    end
    i=i+rate;

    drawnow

    %pause(0.001)

end
%}
%plot3(x(:,1),x(:,2),x(:,3),'-') %this plot the entire dynamics without
    animating



function [Ex,Ey,Ez]=efield(x,y,z)
%This function calulates the E field of the Penning trap given any
%coordinate (x,y,z)
global V0 d
s=sqrt(x.^2+y.^2);
V=(V0./2).*(z.^2-(s.^2)./2)./(d.^2);
for i=1:3
    eps=1E-9; %let epsilon be some very small number
    if i==1
        snew=sqrt((x+eps).^2+y.^2); %we first calculate the positive change in x
        pV=(V0/2).*(z.^2-(snew.^2)/2)./(d.^2); %change in potential
        Ex=[-(pV-V)./(eps)]; %E field is approx the gradient at that point if
    eps is small enough
    elseif i==2
        snew=sqrt((x).^2+(y+eps).^2);
        pV=(V0/2).*(z.^2-(snew.^2)/2)./(d.^2);
        Ey=[-(pV-V)./(eps)];

    else
        pV=(V0/2).*((z+eps).^2-(s.^2)/2)./(d.^2);
        Ez=[-(pV-V)./(eps)];
    end
end
end
```

```
119
120 function x=next(t,B,E,q,m,xn,x1)
121 %This function calculates the next position of the particle given the
122 %current one. It is derived from the finite difference method by Ian Cooper
123 q=q/m;
124 X1=q.*t.*(t.*E-0.5.*cross(x1,B))+2.*xn-x1;
125 c=q*t/2;
126 A=[1 -B(3)*c B(2)*c; B(3)*c 1 -B(1)*c; -B(2)*c B(1)*c 1];
127 x=(inv(A)*X1')';
128 end
129
130 function P=plotE(x,y,z)
131 [Ex,Ey,Ez]=efield(x,y,z); %calculate the E field (function below)
132 normE=(sqrt(Ex.^2+Ey.^2+Ez.^2)); %normalise the vector (optional)
133 P=quiver3(x,y,z,Ex./normE,Ey./normE,Ez./normE,'DisplayName','E field'); %plot
    normalised field
134 %P1=quiver3(x,y,z,Ex,Ey,Ez,'DisplayName','E field') %this gives non-normed field
135
136 end
```