Modelling Plasma Using the Single Particle Model and Investigating the Accuracy of the Model

Abstract:

This report sets out to apply the single particle model, with use of the RK4 iterative method, of a plasma within the “magnetic bottle” field. The level of accuracy of the model was investigated to ascertain its validity in comparison to theoretically expected results. The expected outcomes that the magnetic moment and kinetic energy should be conserved where observed to extremely high levels, such that computationally the standard deviations in both were zero. The critical ratio of the system was found to be tremendously accurate, with an error of just 0.01% from the calculated expected value.

Introduction:

Plasma is often called the fourth state of matter, it comes about from the heating of a gas to the point that it becomes ionised.[1] Due to this, their motion is primarily dictated by electromagnetic forces, from both internal fields from particles and particle interactions, and the influence of external fields.[2] One of the current methods of attempting to achieve sustained fusion is through the confinement of plasmas with magnetic fields, the most current design being the Tokomak. As such it is of high importance that the behaviour of plasmas under confinement is understood [3], leading to the requirement for simulations such as that carried out here.

Nuclear fusion reactions are essentially the opposite to fission reactions, in that two light nuclei come together to create a heavier nucleus. Common reactants are deuterium and tritium (hydrogen nuclei with one and two neutrons respectively) which fuse into helium and a neutron, with the higher binding energy of helium causing a huge generation of energy[3], which can then be harnessed in the same way as other methods such as fission reactors and fossil fuels. The confinement of the plasma leads to this by increasing the energy of the particles to the point in which they can begin to fuse.

The simplest method of confining a plasma is the ‘magnetic bottle’, in which two coaxial magnetic mirrors are used to reflect the particles of the plasma back and forth between them, confining it to the space in between the mirrors. These magnetic mirrors are regions of high magnetic field which repels the particles, with the space in between being an area of low magnetic field.[4] This simple magnetic bottle was the focus of the simulations of this experiment.

The magnetic bottle was the first attempt in using confinement to achieve fusion, however as seen from the results of this experiment the magnetic bottle comes with limitations which lead to the development of more complex reactors for confinement. This refinement of the magnetic field within reactors is what has eventually led to the development of the modern Tokomak.

As plasma is comprised of identical ions a single test particle can be simulated in the magnetic field, this is a suitable starting point in simulating a plasma before modifying the simulation to account for the bulk material, and was the approach taken throughout this experiment.[5]

With this situation the primary aims of the simulation were to initially write a code which would simulate the behaviour of a charged particle in an applied magnetic field, through use of the fourth order Runge-Kutta numerical method (RK4). The next aim was then to observe the fluctuation in the energy and magnetic moment of the particle by plotting these values against time, as well as finding how the frequency of oscillation across the bottle changes with velocity. The final aim was to obtain the conditions for which the particle will become confined within the magnetic bottle. All this together will act as a strong indicator of the accuracy of this model in comparison to theoretical expectation, allowing for judgement of the validity of the model.

Theory:

As the system investigated was a single particle system undergoing an external electromagnetic force, the motion upon it was dictated by the Lorentz force [6]:

$$m\frac{d\boldsymbol{v}}{dt} = Ze(\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B}) \tag{1}$$

In which m is the particle mass, Ze is the charge, $\boldsymbol{v}$ is the velocity, $\boldsymbol{B}$ is the external magnetic field at a point and $\boldsymbol{E}$ is the electric field at a point (throughout the experiment there was no external electric field and so $\boldsymbol{E}$ in equation (1) was always zero). The mass of the particle simulated, a deuterium nucleus is, 3.344×10^{-27} kg and the charge it has is 1.602×10^{-19} C.

As equation (1) is a linear, second order differential equation in space, the RK4 iterative method was used to evolve the system in time, as it is a generally stable method, is self-starting and the step size can be adapted at any point during the simulation [7] which was taken advantage of for this simulation.

For a system that is described by $\boldsymbol{y}(t_0) = \boldsymbol{y}_0$ and $\frac{dy}{dt} = f(t, \boldsymbol{y})$, (in which the 0 subscript denotes the initial position, and for which $\boldsymbol{y}$ is an arbitrary vector dependent upon time, such as velocity or position) as the particle of this system was, the evolution of the system is given by:

$$t_{n+1} = t_n + h \tag{2}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{1}{6}h(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4) \tag{3}$$

From $n = 0$ to the upper limit, h is the step size. The 4 $\boldsymbol{k}$'s are defined as: [7]

$$\boldsymbol{k}_1 = f(t_n, \boldsymbol{y}_n)$$

$$\boldsymbol{k}_2 = f\left(t_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{k}_1\right)$$

$$\boldsymbol{k}_3 = f\left(t_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{k}_2\right)$$

$$\boldsymbol{k}_4 = f(t_n + h, \boldsymbol{y}_n + h\boldsymbol{k}_3) \tag{4}$$

As stated, an advantage of the RK4 is that the step size h can be varied at any point, so after each application of equations (2) – (4) to evolve the system by one step, the Courant-Friedrichs-Lewy (CFL) condition:

$$\frac{v_x h}{\Delta x} + \frac{v_y h}{\Delta y} + \frac{v_z h}{\Delta z} \leq 1 \tag{5}$$

Was applied to change the step size for the next step, which ensured that the system remained stable continuously.

The external magnetic field applied to the particle was described by

$$\boldsymbol{B} = B_x \hat{x} + B_y \hat{y} + B_z \hat{z} \tag{6}$$

For which each component was defined as:

$$B_x = -\frac{2\pi b}{2L} x \sin\left(\frac{2\pi z}{L}\right)$$

$$B_y = -\frac{2\pi b}{2L} y \sin\left(\frac{2\pi z}{L}\right)$$

$$B_z = b\left(2 - \cos\left(\frac{2\pi z}{L}\right)\right) \tag{7}$$

In which b is a constant which describes the field strength (throughout this experiment equal to 0.1 T) and L is the length of the magnetic bottle in meters (throughout this experiment equal to 1.0 m).

For each simulation, the starting coordinates for x and z where always at zero, whilst the starting position in y was given by the Larmor radius, which is given by: [8]

$$r_L = \frac{m v_{perp}}{Zeb} \tag{8}$$

In which v_{perp} is the perpendicular velocity, given by $\sqrt{v_x^2 + v_y^2}$.

If the dot product of equation (1) with $\boldsymbol{v}$ is taken, and noting that this with the cross product in equation (1) gives zero, the result is:

$$m\frac{d\boldsymbol{v}}{dt} \cdot \boldsymbol{v} = \frac{d}{dt}\left(\frac{1}{2} m v^2\right) = 0 \tag{9}$$

Showing that throughout the simulation the kinetic energy of the particle should have remained constant [9], as it is static, this was a core test of if the simulation was working. The magnetic moment of the particle is something else which must have remained constant, as can be clearly seen by the equation for the magnetic moment:

$$\mu = \frac{m v_{perp}^2}{2B} \tag{10}$$

Which clearly contains within it the perpendicular component of the kinetic energy, and so as this should have remained constant, so too should have the magnetic moment of the particle.

For the magnetic bottle there is a critical ratio of velocity, which is the limit at which the particle will be reflected within the magnetic bottle, given by:

$$\frac{v_{perp_a}}{v_a} = \sqrt{\frac{B_a}{B_b}} \tag{11}$$

In which the subscripts a and b represent points at the area of lowest magnetic field (centre of the magnetic bottle) and area of highest magnetic field (top/bottom of the magnetic bottle) respectively. The full derivation of this result is given in appendix A. Using equation(s) (7) the minimum and maximum field for $b = 0.1$ are 0.1 and 0.3, which gives the theoretical critical ratio to be 0.57735026919, to the maximum number of digits displayed upon a calculator screen.

The divide and conquer method was used in finding the critical ratio, this works by having an initial range of values and taking the midpoint. After a simulation had been run, depending upon the possible outcomes (for these simulations this was if the partial had escaped or not) the range is then changed such that the midpoint becomes either the new upper or lower bound. This is then repeated until a chosen degree of accuracy is reached (ideally this would be done to an infinitesimal point that is being tended towards, but this is of course limited by computational ability).

Method:

Initially the code was written with the initial values for the particle required (such as mass, charge, position (set upon the origin) and velocity (-1.0 ms^{-1} in y-direction)) and a subroutine for the RK4, and a uniform magnetic field in the positive z-direction. This was done to allow initial testing of the RK4 to ensure it was working correctly, given that the expected trajectory of a charged particle in a uniform field is easily calculated, and so plotting the trajectory made checking it is working elementary. There was also a simple function written to perform the cross products within the RK4, as Fortran does not have an inbuilt cross product function.

The uniform field applied was 1.0 T in the z-direction, so by equating the Lorentz force and Newton's law of circular motion (noting that as the field and velocity of the particle are perpendicular, the cross product of the Lorentz fore becomes a simple multiplication) and rearranging for radius, the theoretical radius of the motion of the particle in this system would be 2.08×10^{-8} m.

Upon confirmation that the RK4 was functioning correctly, the main simulation was moved to being in a subroutine and the magnetic bottle field described by equations (7) were implemented as a simple function which could be called within the subroutine for the main simulation. The initial velocity was then set to $(1.0 \times 10^5, 0.0, 1.0 \times 10^5)$ ms^{-1} and the initial position set as described by equation (8). As a way of reducing computation time and increasing efficiency, the position coordinates of the particle were plotted every 1000 steps from here on. The CFL condition, equation (5), was also implemented at this point to ensure stability.

The x, y and z coordinates where written out to a file at this point to be able to produce a 3D plot of the trajectory of the particle, which was the simplest way to observe if the particle remained confined or escaped the bottle.

Equation (10) and the equation for kinetic energy where then implemented, so that each of these may be written to a file and plotted against time. This therefore meant that it was straightforward to observe if these were being held constant (or constant within reason accounting for computation error).

The z-coordinate was also written out to a separate file against time, allowing for the plotting of the particle's oscillation across the magnetic bottle. A Fourier transform could then be applied to this data (this was easily done using the inbuilt tool in xmgrace), transforming into frequency space resulting in a peak (a Dirac delta analytically) at the frequency of the oscillation. From here the initial velocity could be varied to investigate the effect of this upon the different outputs the code was giving.

Next, a subroutine was written which used the divide and conquer method to find the critical ratio for the magnetic bottle, described by equation (11), this was called at the end of the subroutine which runs the simulation. With the dived and conquer run through the new smaller range would be found, as such the simulation would be run again with this new range by calling the main simulation subroutine, then repeating this cycle until the critical ratio was found to a chosen degree of accuracy (here so that the difference between the upper and lower bound was 1.0×10^{-5}).

The full code is given in appendix B, with the module of constants it uses given in appendix C.S

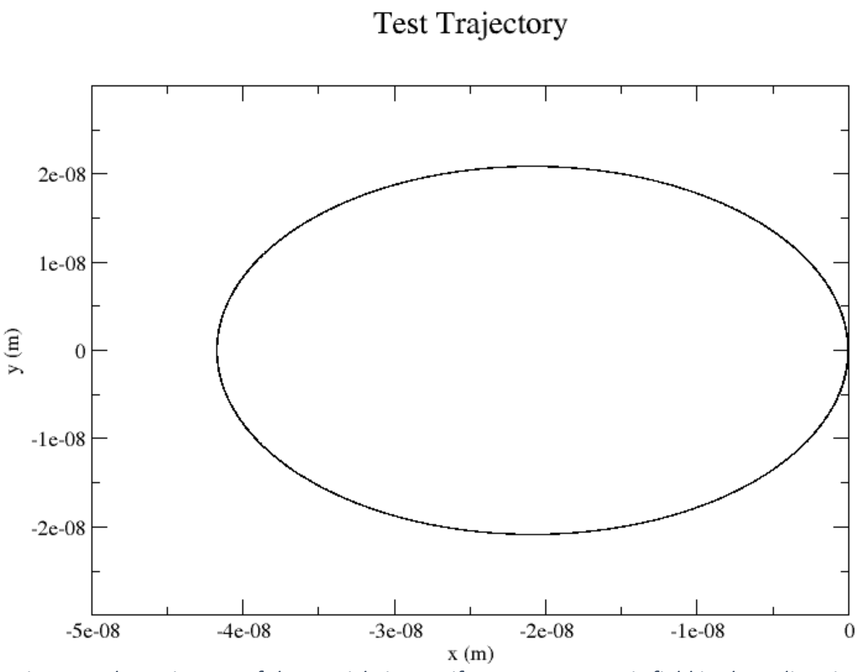

Figure 1: the trajectory of the particle in a uniform, 1 T, magnetic field in the z-direction. The particle underwent circular motion with a radius of $\sim 2.0 \times 10^{-8}$ m, giving confirmation that the RK4 subroutine was functioning. The initial position was on the origin and the initial velocity -1.0 ms^{-1} in the y-direction. There were 1000000 steps performed.

Results:

Having performed the initial testing on the RK4, the resulting trajectory is shown in figure 1. Upon inspection the diameter is approximately 4.2×10^{-8} m resulting in an approximate radius of the particles circular motion of 2.1×10^{-8} m, very close to the

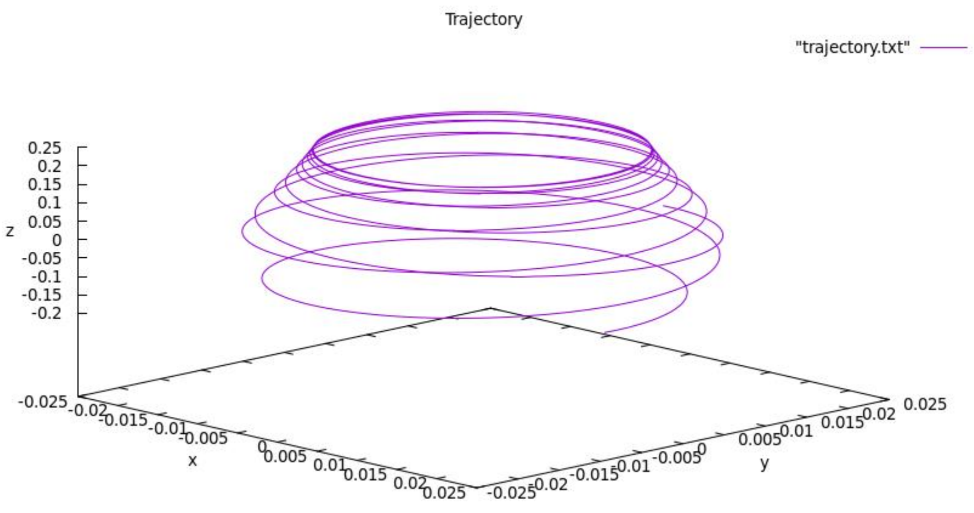

Figure 3: the trajectory of a particle with initial position in x and z at 0, and y given by the Lamour radius, equation 8. The initial velocity was 1.0×10^5 *ms*$^{-1}$*in the x and z direction, and zero in the y-direction. The number of steps was 1000000.*

theoretical value calculated (and possibly even more exact as this was only a brief examination), hence this demonstrated the RK4 to be producing correct results.

Figure 2 shows the trajectory of the particle with the magnetic field now as the magnetic bottle, with the initial conditions of the particle being those previously given. This trajectory also marks the point from which the CFL condition was used to adapt the step size continuously.

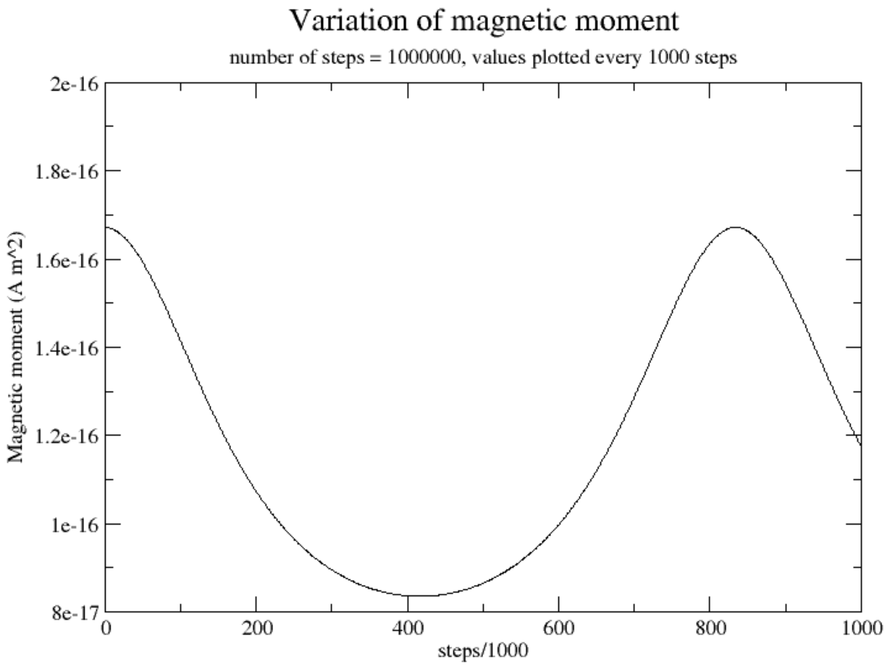

Figure 2: plot of the magnetic moment of the particle against the steps over 1000 (due to plotting only every 1000 steps) with initial conditions corresponding to those of figure 2. The variation in the magnetic moment here is (by inspection) approximately $0.9 \times^{-16}$ *Am*^{2}*, which is small enough to be approximately constant within this simulation.*

As stated previously, a key way to test the accuracy of the system being simulated was to observe the change in magnetic moment and kinetic energy of the deuterium nucleus. Figure 3 shows a plot of the magnetic moment against the steps taken (analogous to time) and figure 4 the same for the kinetic energy.

The variation in the magnetic moment was very low (on the order of 10^{-16}) and there

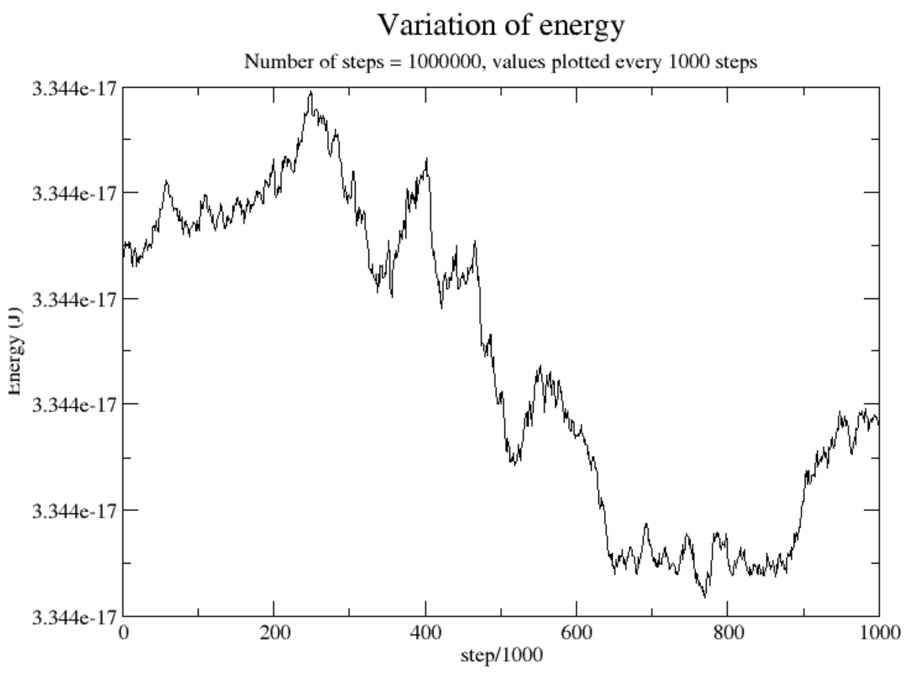

Figure 4: plot of the variation of the kinetic energy of the particle against the steps over 1000, with initial conditions corresponding to those of figure 2. By the scale of the axis there is negligible variation visible in the kinetic energy.

was not a high enough degree of accuracy in the data for the kinetic energy to show any

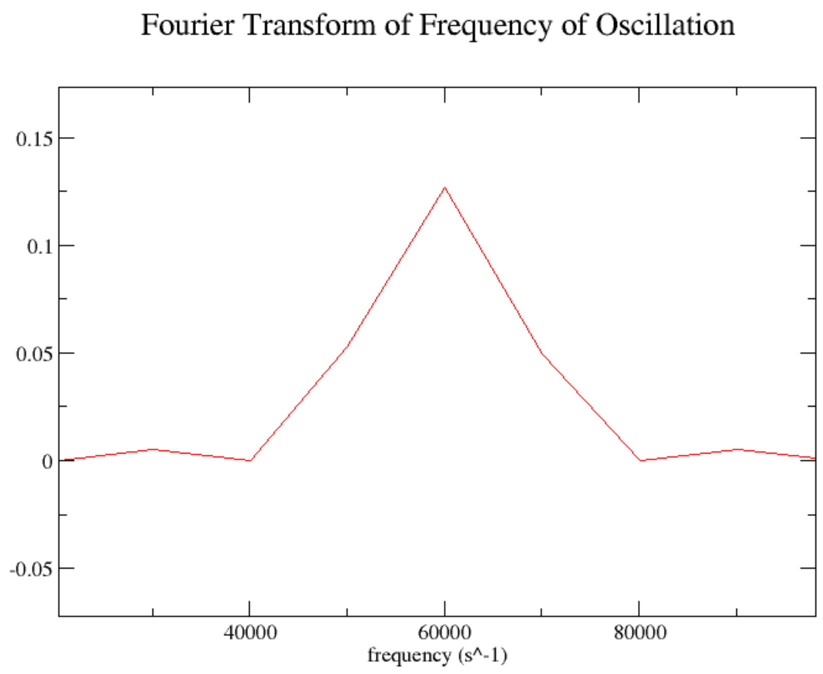

Figure 5: Fourier transform of the z-component of position against time with conditions matching the particle in figure 2. By using the inbuilt abilities of xmgrace this was found to be 60066.8 s^{-1}.

variation, suggesting both to be behaving as would be expected. As a second test, a subroutine was written to calculate the standard deviation of both of these, the output value for both came out as zero within the terminal.

Figure 5 shows the plot containing the approximation for the Dirac delta in frequency space, to give the frequency of oscillation for the deuterium nucleus in figure 2. The number of time steps taken was increased by 10 as the simulation in figure 2 did not complete a full oscillation, and the accuracy was increased by having a full wave like function to Fourier transform in xmgrace. Simply by zooming in very close on the peak within xmgrace and rolling the curser over the point of the peak, the value of the frequency could be found to be 60066.8 s^{-1}. It should be noted that here the x-axis was plotted as true time rather than steps, by continuously summing the time step h and also writing this to file, to allow the frequency to be in correct units.

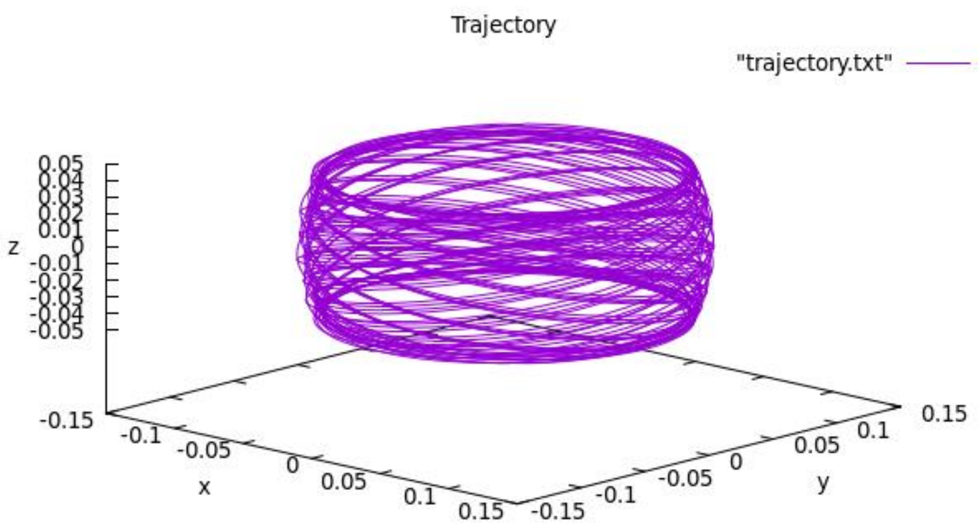

Figure 4: trajectory of a deuterium nucleus with the initial velocity changed from that of figure 2, such that the x-component is 5 times greater. The number of steps was 10000000 to ensure there was full oscillations for the frequency calculation.

Figure 6 shows the trajectory of a deuterium nucleus with all initial conditions matching those for figure 2, with the exception of the initial velocity in which the x-component was increased by 5 to be 5.0×10^5 ms^{-1}.

Figure 7 shows the approximation of the Dirac delta for the frequency of the system in figure 6. By using the same means as with figure 5, the frequency for this system was found to be 350206 s^{-1}.

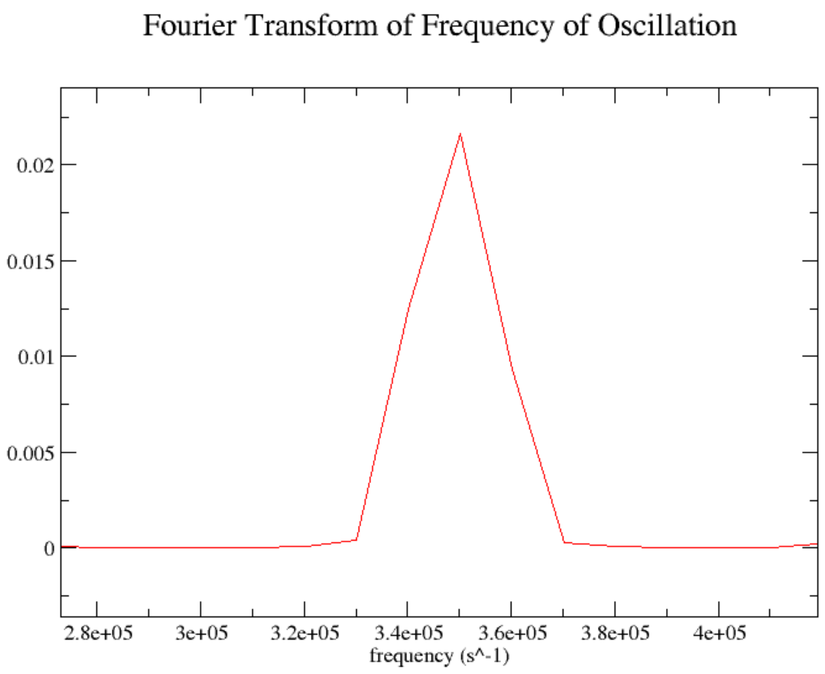

Figure 5: Fourier transform of the z-component of position against time with conditions matching the particle in figure 6. The frequency here was found to be 350206 s^{-1}.

The x-component was then set back to as it was for figure 2, and the z-component set to be 5 times greater than in figure 2, so that it was 5.0×10^5 ms^{-1}. This caused the particle to escape from the magnetic bottle, illustrated in figure 8, and as such there was of course no frequency of oscillation for this system. The same was found by reducing the x-component by 5 to 0.2×10^5 ms^{-1}.

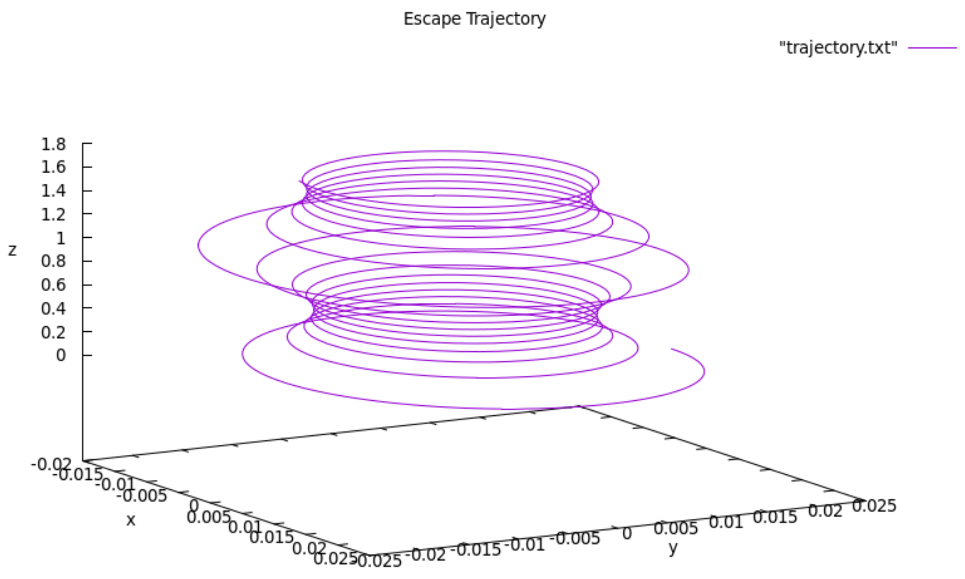

Figure 6: trajectory of a deuterium nucleus with the initial z-component of the velocity increased by 5 from the particle in figure 2, causing the particle to escape the magnetic bottle.

Upon implementation of the divide and conquer technique, along with use of equation (11), with a chosen level of accuracy being a difference of less than 1.0×10^{-5} between the upper and lower bounds for the critical ratio, the critical ratio (using the system illustrated in figure (2)) was then found to be 0.5767822656250000 using the maximum number of significant figures shown in the terminal.

Discussion:

The trajectory shown in figure 2 represents the path of a charged particle that would be expected from a magnetic bottle field, as it followed the characteristic shape of this field, illustrated in figure 9.

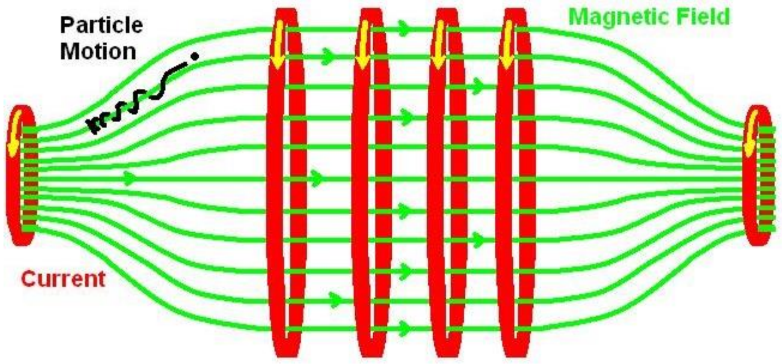

Figure 7: an illustration of the characteristic shape of a magnetic bottle. [10]

Figures 3 and 4 illustrate the variation, respectively, of the magnetic moment and kinetic energy of the deuterium particle with respect to the step number of the simulation, for a simulation made up of 1000000 steps. The magnetic moment in figure 3 had a total variation of approximately 0.9×10^{-16} Am^2, which being much less than one is clearly an indicator of very little variation in the magnetic moment. The scale of the y-axis in figure 4 didn't show any readable variation in the kinetic energy of the particle, and further examination of the raw data revealed that the value for the kinetic energy remained constant up to thirteen significant figures, showing a very clear high degree of accuracy in the conservation of the kinetic energy. The implementation of the calculation to find the standard deviations of these, both of which coming out as zero, further indicated the high degree of accuracy the model had for conservation of both of these values. These results combined showed the model to very closely follow what was expected by equations (9) and (10).

The frequencies represented in figures 5 and 7 clearly indicate a correlation between the magnitude of the velocity and the frequency of the particles oscillations across the bottle, as would be expected by equation (1) as the Lorentz force upon the particle will increase with its velocity, a greater force resulting in faster movement across the bottle and hence a more rapid oscillation.

Of course, this wasn't the only factor affecting the particle, as the critical ratio given by equation (11) also dictates if the particle will remain confined, regardless of the oscillation rate the particle may undergo. The experimental result found for the critical ratio was 0.5767822656250000, which can be compared to the analytic result which can be derived from equation (11), as earlier discussed, to be 0.57735026919. This gives an error in the experimental result from the simulation of 0.01%, further showing the high degree of accuracy for this simulation.

These results clearly show that this simulation, using the RK4, is a highly precise model for projecting the trajectory of a particle within a plasma for a magnetic field. However, a real plasma is of course not a single particle but a bulk of ionised gas, as well as the free electrons which have been removed from the ions. Hence one failing of this particular model is that it does not take into account the Coulomb interactions that happen within the plasma, which will of course have an impact upon the trajectory of a single particle, there are alternative models to this single particle approach (which require a more complex code) which will take into account these interactions, for instance the fluid description of plasma. [11]

Following from this experiment, which has acted to show the high degree of accuracy (within discussed limit) the single particle model has for plasma modelling, further study could be elementarily carried out by building upon this. Simple adaptation to the model used here to introduce the magnetic field of a more recent plasma confinement reactor (such as that of a tokamak, which is a combination of a toroidal field and a poloidal field) to then investigate what happens under these conditions.

Conclusion:

The overall conclusion that can be made from the results of this experiment is that the single particle model is a highly accurate model for simulating the behaviour of a plasma under influence of a magnetic field, of course accepting the neglecting of the inter-particle Coulomb interactions. The primary indicators of this being the conservation of the magnetic moment and kinetic energy of the test particle, and most importantly the very high degree of accuracy in which the model predicted the critical ratio for the deuterium nucleus within this particular magnetic field. The models success here means it would be straightforward to incorporate the fields of more modern reactor designs to perform investigations upon these.

References:

[1] – Chen, F.F. (2016). Introduction. In: *Introduction to Plasma Physics and Controlled Fusion*. Springer, Cham

[2] – Bittencourt, J.A. (2004) *Fundamentals of Plasma Physics*. 3rd ed. New York; Springer-Verlag New York Inc, p. 25

[3] – Kikuchi, M. (2010). A Review of Fusion and Tokomak Research Towards Steady-State Operation, A JAEA Contribution. *Energies*, 1741-1789, p. 1742

[4] – Bittencourt, J.A. (2004) *Fundamentals of Plasma Physics*. 3rd ed. New York; Springer-Verlag New York Inc, p. 77

[5] – Chen, F.F. (2016). *Introduction to Plasma Physics and Controlled Fusion*. London, Springer, Cham, p. 17

[6] – Bittencourt, J.A. (2004) *Fundamentals of Plasma Physics*. 3rd ed. New York; Springer-Verlag New York Inc, p.34

[7] – Milne, W.E. (1950) Note of the Runge-Kutta Method. *Research of the Journal of Research of the National Bureau of Standards*, Research Paper RP2101 Volume 44

[8] – Bittencourt, J.A. (2004) *Fundamentals of Plasma Physics*. 3rd ed. New York; Springer-Verlag New York Inc, p.40

[9] – Bittencourt J.A. (2004) *Fundamentals of Plasma Physics*. 3rd ed. New York; Springer-Verlag New York Inc, p. 34-35

[10] – https://en.wikipedia.org/wiki/Magnetic_mirror#/media/File:Basic_Magnetic_Mirror.jpg (09/04/2019)

[11] – Wiesemann, K. (2012) *A Short Introduction to Plasma Physics*. Lecture Notes, CERN Course on Ion Sources, delivered 29 May – 8 June 2012

Appendix A:

Given that the magnetic moment of the particle, equation (10), is conserved such that it remains constant, it is implied that:

$$\frac{mv_{perp_a}^2}{2B_a} = \frac{mv_{perp_b}^2}{2B_b}$$

In which a and b refer to the regions of high and low magnetic field within the bottle respectively. From this it is implied that:

$$\frac{v_{perp_a}}{v_{perp_b}} = \sqrt{\frac{B_a}{B_b}}$$

Given also that the kinetic energy of the particle is conserved, it is implied that:

$$v_{perp_a}^2 + v_{para_a}^2 = v_{perp_b}^2 + v_{para_b}^2$$

In which v_{para} is the component of the velocity parallel to the direction of the magnetic bottle/ the z-axis. This the leads to:

$$v_{perp_b} = \sqrt{v_{perp_a}^2 + v_{para_a}^2} = v_a$$

This can then be substituted into the second equation to give the result:

$$\frac{v_{perp_a}}{v_a} = \sqrt{\frac{B_a}{B_b}}$$

Appendix B:

```
program plasma
   use consts
   implicit none

   real(kind=dp) :: ze, m, b, rl, vel_perp, vel_perp_initial, vel_parallel, L

   !required for first run only
   real(kind=dp) :: upper_bound, lower_bound, midpoint

   !velodity, position and total magnetic field vectors
   real(kind=dp), dimension(3) :: vel, pos, field, pos_initial

   integer :: i, j

   !Initial input values and constants
   ze = 1.602e-19
   m = 3.344e-27
   b = 0.1_dp
   L = 1.0_dp
   vel = [1.0e5_dp, 0.0_dp, 1.0e5_dp]
   vel_perp = sqrt(vel(1)**2 + vel(2)**2)
   vel_perp_initial = vel_perp
   vel_parallel = vel(3)
   rl = (m * vel_perp) / (ze * b)
   pos = [0.0_dp, rl, 0.0_dp]
   pos_initial = pos

   !initial values for divide and conquor
   upper_bound = 1.0_dp
   lower_bound = 0.0_dp
   midpoint = 0.5_dp

   !update the velocity for ratio calculation
   vel_parallel = vel_perp * sqrt((1.0_dp / midpoint)**2 - 1.0_dp)
   vel = [vel_perp, 0.0_dp, vel_parallel]

   call simulation(pos, vel)

   contains

   subroutine simulation(pos, vel)

      real(kind=dp), dimension(3), intent(inout) :: pos, vel
      real(kind=dp), dimension(3) :: vel_init

      !ze is the particle charge, m the mass, h time step, b constant for
      !magnetic filed, rl rhe Larmor radius, L the length of the magnetic
      !bottle, vel_perp the perpendicular velocity, mag_mom, the magnetic
```

```
!moment, energy the particle energy
real(kind=dp) :: h, mag_mom, energy

!mag_momvariance and energyvariance are the standard deviation values
!time is the real time value to be updated with h
real(kind=dp) :: mag_momvariance, energyvariance, time

!Arrays for performing stats on the magnetic moment and energy
real(kind=dp), dimension(:), allocatable :: mag_momarray, energyarray

!field is the magnetic field, pos the position vector, vel the velocity
!vector, pos_old the previous position vector for calculating h
real(kind=dp), dimension(3) :: pos_old

!units for files, istat for iostat checks, i used as counter
integer :: unit1, unit2, unit3, unit4, istat, plotrate, steps

!open all files
open(newunit=unit1, file="trajectory.txt", iostat=istat)
if(istat/=0) stop "Error opening trajectory.txt"
open(newunit=unit2, file="mag_moment.txt", iostat=istat)
if(istat/=0) stop "Error opening mag_moment.txt"
open(newunit=unit3, file="energy.txt", iostat=istat)
if(istat/=0) stop "Error opening energy.txt"
open(newunit=unit4, file="frequency.txt", iostat=istat)
if(istat/=0) stop "Error opening frequency.txt"

!Initialise values
h = 1.0e-11_dp
time = 0.0_dp
steps = 1000000
plotrate=1000
vel_init = vel
pos = pos_initial

allocate(mag_momarray(steps / plotrate))
allocate(energyarray(steps / plotrate))

do i = 1, steps
   !reset position for future runs

   !set the value for the magnetic field at current point
   field = magnetic_bottle(b, L, pos(1), pos(2), pos(3))

   !perform rk4 at current point to update position and velocity
   pos_old = pos
   call rk4(pos, vel, h)
   if (mod(i, plotrate) == 0) then

      j = i / plotrate
```

```
        !write the coordinates to file
        write(unit=unit1, fmt=*) pos(1), pos(2), pos(3)

        !write z component to file for frequency
        write(unit=unit4, fmt=*) time, pos(3)

        !calcualte magnetic moment and energy
        mag_mom = (m * vel_perp**2) / (2.0_dp * norm2(field))
        energy = 0.5_dp * m * norm2(vel)**2

        mag_momarray(j) = mag_mom
        energyarray(j) = energy

        !write the magnetic moment and energy to file
        write(unit=unit2, fmt=*) time, mag_mom
        write(unit=unit3, fmt=*) time, energy

    end if

    !update time value
    time = time + h

    !calculate h for next step
    h = norm2(pos - pos_old) / norm2(vel)

end do

call stddev(mag_momarray, mag_momvariance)
call stddev(energyarray, energyvariance)

print *, mag_momvariance, energyvariance

!close all files
close(unit=unit1, iostat=istat)
if(istat/=0) stop "Error"
close(unit=unit2, iostat=istat)
if(istat/=0) stop "Error"
close(unit=unit3, iostat=istat)
if(istat/=0) stop "Error"
close(unit=unit4, iostat=istat)
if(istat/=0) stop "Error"

!deallocate arrays
deallocate(mag_momarray)
deallocate(energyarray)

vel_perp = sqrt(vel_init(1)**2 + vel_init(2)**2)
vel_parallel = vel_init(3)
```

```
    call critical_value(upper_bound, lower_bound, pos, vel_init, vel_parallel, L, midpoint)


end subroutine

subroutine rk4(pos, vel, h)

    !position and velocity values
    real(kind=dp), intent(inout), dimension(3) :: pos, vel
    real(kind=dp), intent(in) :: h
    !k values to calculate for position and velocity
    real(kind=dp), dimension(3) :: k1pos, k2pos, k3pos, k4pos, k1vel, k2vel, k3vel, k4vel

    k1vel = ze/m * cross(vel, field)
    k1pos = vel
    k2vel = (ze/m * cross(vel + h/2.0_dp * k1vel, field))
    k2pos = vel + h/2.0_dp * k1vel
    k3vel = (ze/m * cross(vel + h/2.0_dp * k2vel, field))
    k3pos = vel + h/2.0_dp * k2vel
    k4vel = (ze/m * cross(vel + h * k3vel, field))
    k4pos = vel + h * k3vel
    vel = vel + 1.0_dp/6.0_dp * h * (k1vel + 2.0_dp * k2vel + 2.0_dp * k3vel + k4vel)
    pos = pos + 1.0_dp/6.0_dp * h * (k1pos + 2.0_dp * k2pos + 2.0_dp * k3pos + k4pos)

end subroutine

!function to calculate cross products
function cross(a, b)

    real(kind=dp), dimension(3) :: cross
    real(kind=dp), dimension(3) :: a, b

    cross(1) = a(2) * b(3) - a(3) * b(2)
    cross(2) = a(3) * b(1) - a(1) * b(3)
    cross(3) = a(1) * b(2) - a(2) * b(1)

end function

!function to calculate the field at a point for the magnetic bottle
function magnetic_bottle(b, L, x, y, z)

    real(kind=dp), dimension(3) :: magnetic_bottle
    real(kind=dp) :: b, x, y, z, L, bx, by, bz

    bx = -((2.0_dp * pi * b) / (2.0_dp * L)) * x * sin((2.0_dp * pi * z) / L)
    by = -((2.0_dp * pi * b) / (2.0_dp * L)) * y * sin((2.0_dp * pi * z) / L)
    bz = b * (2.0_dp - cos((2.0_dp * pi * z) / L))

    magnetic_bottle = [bx, by, bz]
```

```
end function

!subroutine to calculate the standard deviation
subroutine stddev(input, variance)

    real(kind=dp) :: mean
    double precision, intent(out) :: variance
    real(kind=dp), intent(in), dimension(:) :: input
    integer :: N

    N = size(input)
    mean = 1.0_dp / real(N, dp) * sum(input)

    do i = 1, N
        variance = variance * (input(i) - mean)**2
    end do

    variance = variance / real(N, dp)

end subroutine

subroutine critical_value(upper_bound, lower_bound, pos, vel, vel_parallel, L, midpoint)

    real(kind=dp), intent(inout) :: upper_bound, lower_bound, vel_parallel
    real(kind=dp), intent(in) :: L
    real(kind=dp), dimension(3), intent(inout) :: pos, vel
    real(kind=dp) :: midpoint

    if(pos(3) < abs(L/2.0_dp))then
        upper_bound = midpoint
    end if

    if(pos(3) > abs(L/2.0_dp))then
        lower_bound = midpoint
    end if

    midpoint = (upper_bound + lower_bound) / 2.0_dp

    if(upper_bound - lower_bound < 1.0e-5_dp)then
        print *, "Convergence reached at midpoint", midpoint
        print *, "critical ratio is", vel_perp / norm2(vel)
        return
    end if

    vel_perp = vel_perp_initial
    vel_parallel = vel_perp * sqrt((1.0_dp / midpoint)**2 - 1.0_dp)
    vel = [vel_perp, 0.0_dp, vel_parallel]

    call simulation(pos, vel)
    end subroutine
end program plasma
```

Appendix C:

```
module consts
   implicit none
   integer, parameter :: dp=selected_real_kind(15,300)
   integer, parameter :: i64=selected_int_kind(18)
   real(kind=dp), parameter :: pi=3.14159265358979323846, mu_nought=1.26e-6,
epsilon_nought = 8.85e-12, SOL = 299792458
   real(kind=dp), parameter :: boltzmann=1.3806e-23
   complex(kind=dp) :: imaginary = cmplx(0.0_dp, 1.0_dp,dp)
end module consts
```