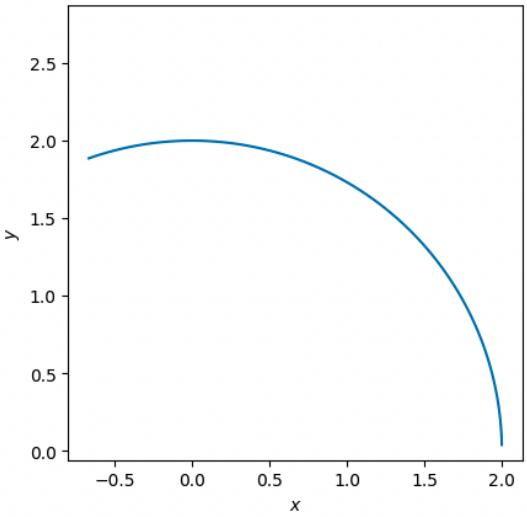
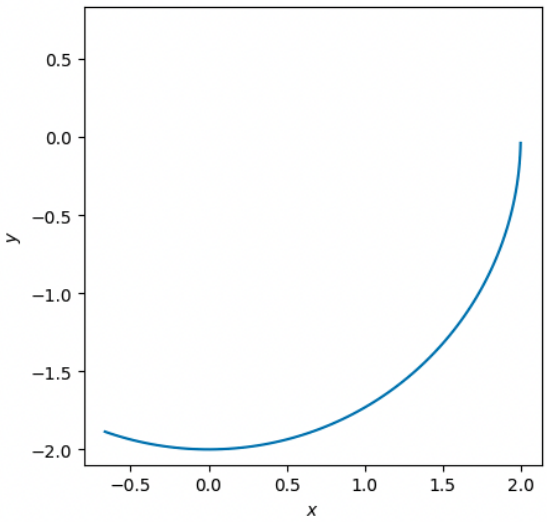
Project Notebook

15/12/22

I didn't do much work yesterday. Today I'm going to try and integrate the motion of the particle backwards in time, just like in Hao+ (2020). Here's what the trajectory looks like when integrating forwards in time. The particle starts at (2., 0.) and moves anticlockwise. Hopefully when integrating backwards, we'll get clockwise motion.





Success!
From now on, we will stick with the backwards integration to make comparison easier with Hao+ (2020).

In Hao+ (2020), they use a starting energy of $E_K = 0.2 - 2$ MeV. I'm assuming that this is the kinetic energy, since the total energy must be greater than the rest mass 0.511 MeV. Using the formula $E_K = (\gamma - 1)m_0c^2$, we get $0.271 < \beta < 0.979$. The electrons are relativistic. The particles end up between $4 < L < 10$. I'm unsure how to convert the speed to $\dot{\phi}_0$. Do I divide by L? or by LR_S? Have sent an email to Emma.

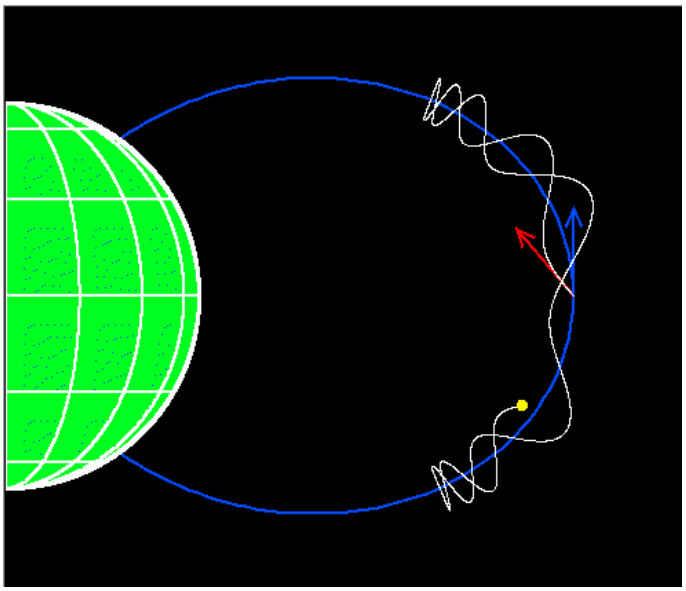
16/12/22

Emma has since replied. She mentioned using the formulae for drift velocity in Van Allen+ (1980), but I'm unsure how that fits in with the initial velocity.

Van Allen

This paper adopts the model of the magnetic field's origin being a point magnetic dipole of moment $M = 0.20$ G located at the geometrical centre of the planet with M parallel to Ω, the rotational angular velocity of the planet. This is a simplification of the observed field in (Smith+ 1980, Acuna and Ness 1980).

Equatorial pitch angle



A charged particle will gyrate around the magnetic field line, clockwise or counterclockwise depending on the sign of its electrical charge, and it will also bounce up and down the field line, "mirroring" and reversing direction at symmetrical points above and below the magnetic equator, as shown in the figure (projecting the particle trajectory into a meridional plane).

The equatorial pitch angle is the angle between the velocity of the particle (red arrow) and the magnetic field (blue arrow). In this plot, $\alpha_0 \approx 40^\circ$. The larger α_0, the smaller the latitude at which it mirrors. At $\alpha_0 = 90^\circ$, where the particle moves perpendicular to the magnetic field there, will mirror at the equator and therefore remains at the magnetic equatorial plane. This is the regime that we are interested in.

Bounce-averaged longitudinal drift

The exact motion of a charged particle is complicated. We can simplify things by averaging over the bouncing motion to get the longitudinal drift (eastward is defined as positive while westward is negative).

The bounce-averaged longitudinal drift of the guiding centre of a particle trapped in the magnetic field is given by

$$\omega_D = \frac{3m_0c^2\beta^2\gamma L}{2qB_0R^2}\frac{F(\lambda_m)}{G(\lambda_m)}$$

where λ_m is the magnetic latitude of the particle's mirror point, and F and G are functions of λ_m. It is important to mention that this is the longitudinal angular velocity in the reference frame rotating with the planet. For Saturn, this is approximately

$$\omega_D = \pm 2.083 \times 10^{-5} L E_K \left(\frac{E_K + m_0c^2}{E_K + m_0c^2}\right)\left(\frac{F}{G}\right)$$

The plus/minus sign applies to protons/electrons.

For $\alpha_0 = 90^\circ$, we have $\lambda_m = 0^\circ$. Both the integrals for F and G are equal, so their ratio is 1. This simplifies the equation to:

$$\omega_D = \pm 2.083 \times 10^{-5} L E_K \left(\frac{E_K + m_0c^2}{E_K + m_0c^2}\right)$$

To convert to the angular velocity in an inertial frame, we simply add on the corotational angular velocity, giving

$$\omega_I = \Omega + \omega_D$$

Resonant energy

Consider a neutral body in orbit around Saturn. The mean angular velocity of this satellite in gravitational orbit around Saturn is given by

$$\omega_k = \left(\frac{GM}{a}\right)^{\frac{1}{2}}\left(1 - \frac{3J_2R^2}{2a^2}\right)^{-\frac{1}{2}}$$

If $r = 1.871\ R_S$, then we have a geostationary orbit and $\omega_k = \Omega$.

The relative angular velocity between a charged particle and a satellite in a circular prograde motion at the same radial distance is given by

$$\omega_I - \omega_k = \Omega + \omega_D - \omega_k$$

and thus,

$$T_E = \frac{2\pi}{|\omega_I - \omega_k|}$$

The value of E such that T_E is infinite is called the resonant energy, ie, when the angular velocity of a charged particle is equal to the angular velocity due to gravitational motion. This condition is satisfied when $\Omega + \omega_D - \omega_k = 0$, a quadratic equation for E_R, whose solution is

$$E_R = 0.5\left[E' - 1.022 + (E'^2 + 1.0445)^{\frac{1}{2}}\right]$$

where

$$E' = 4.800 \times 10^4 \frac{(\Omega - \omega_k)}{L}$$

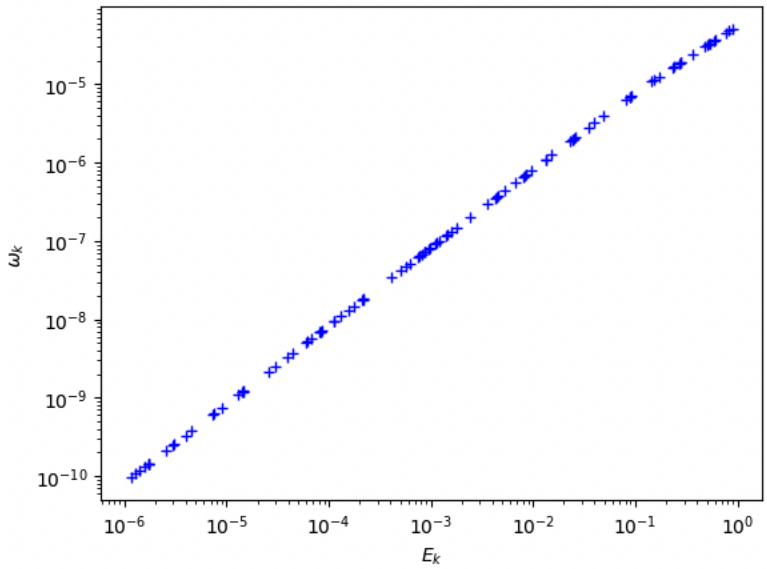
This equation is only for $\lambda_m = 0°$.

19/12/22

In Hao+ (2020), specifically section *B.2. Simulating the Trajectory of Electrons*, the motion of equatorial mirroring electrons is obtained. Just to clarify that at E_{CDR}, it is the azimuthal motion that cancels out, and we have radial motion. All the motion that is studied here is in the equatorial plane.

Charged particles in the magnetosphere drift under the influence of electric ($E \times B$) and the energy-dependent magnetic drift. This electric drift includes contributions from corotation and convection. Hao and others adopt a simple uniform, noon-to-midnight electric field, effectively corresponding to a flow from dusk to dawn.

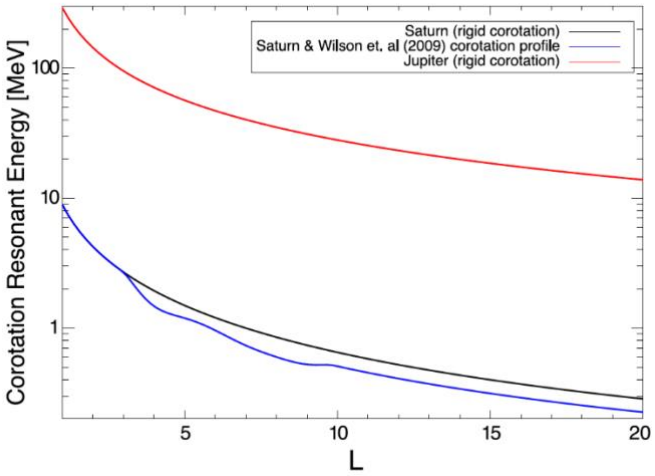
This electric field distorts the circular, bounce-averaged drift orbits of charged particles in Saturn's magnetic dipole. For electrons, for whom their magnetic drift opposes corotation, there are energies (E_{CDR}) where the two terms cancel, leaving the non-corotational electric fields to dominate.
Here's ω_D as a function of E_k.



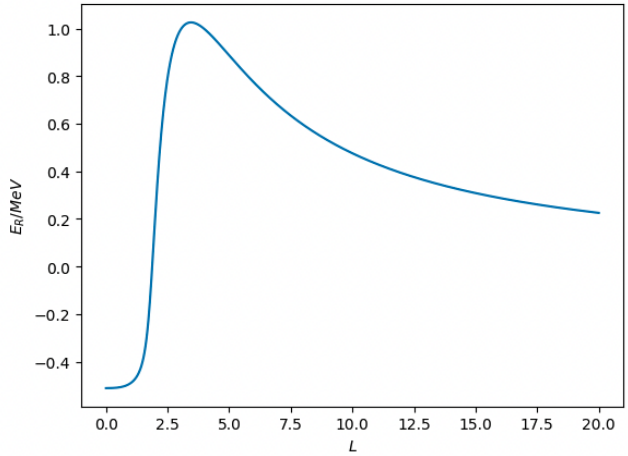
This formula only works if E_k is given in MeV.

21/12/22

The goal today will be to recreate the following figure from Roussos+ (2018b)

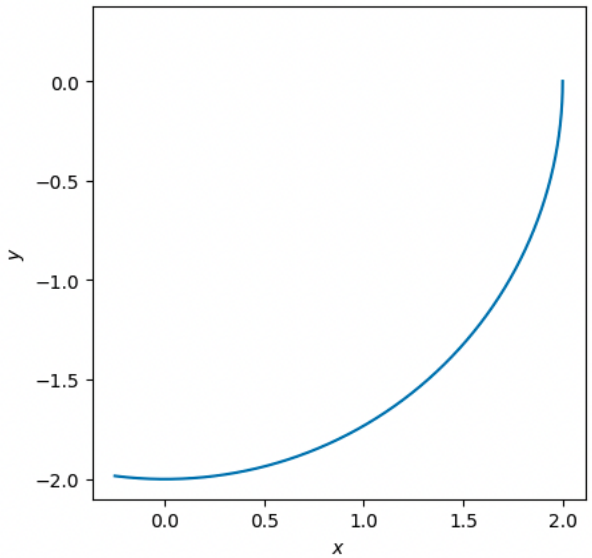


Managed to get this plot. The negative energies are concerning.



23/12/22

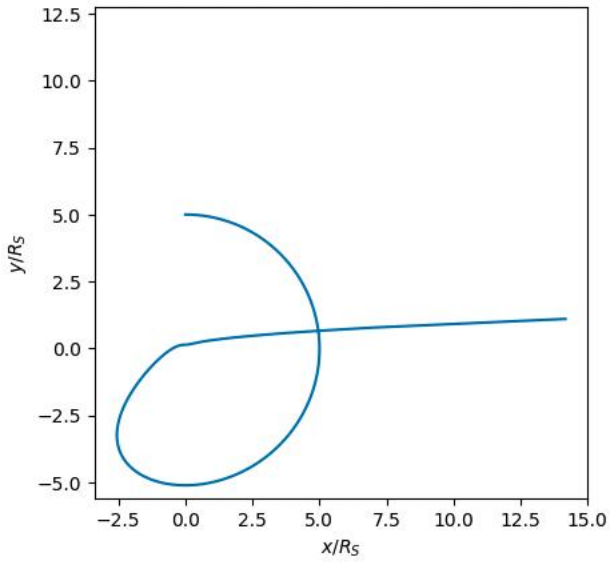
Perhaps some progress has been made? I've used realistic values in my plots. I was wondering why the x and y values weren't changing much, and I realised that maybe it's because the simulation was only running for 1 second. I've cranked up the time to 100 seconds I got the following plot:



Promising.

Let's run it backwards for 10 minutes.

Less good news :(The integrator returned a warning, saying:
UserWarning: dop853: step size becomes too small
Unsure what this means, but I'll do some reseach.



28/12/22

It's been a few days! I'm still unsure what to do with the code, so I'll do some reading today. Maybe *Dynamics of Geomagnetically Trapped Radiation*.

We say that cyclotron motion exists if at any instant of time we can define a moving frame of reference in which an observer sees the particle in a period orbit perpendicular to the magnetic field. There must be a *single* periodicity and it must last at least *one* full cycle. There are cases in which this orbit in that moving frame is circular. If this frame can be found, then we say that the *guiding centre approximation* holds. The guiding centre is the *guiding centre*.

We can split the instantaneous velocity of the GCS into two components: parallel and perpendicular to the magnetic field. $V_{||}$ is equal to the parallel velocity of the particle and $V_{\perp}$ is called the particle's *drift velocity* V_D because it represents the velocity with which we see the particle 'drift away' from the initial field line in the inertial frame.

Quasi-trapped particles are particles that remain trapped in the magnetosphere for only a limited time, running into the boundary at the flanks of the magnetosphere.

The curvature drift is given by

$$\boldsymbol{V_C} = \frac{mv_{||}^2}{qR_cB}\boldsymbol{m} \times \boldsymbol{e}$$

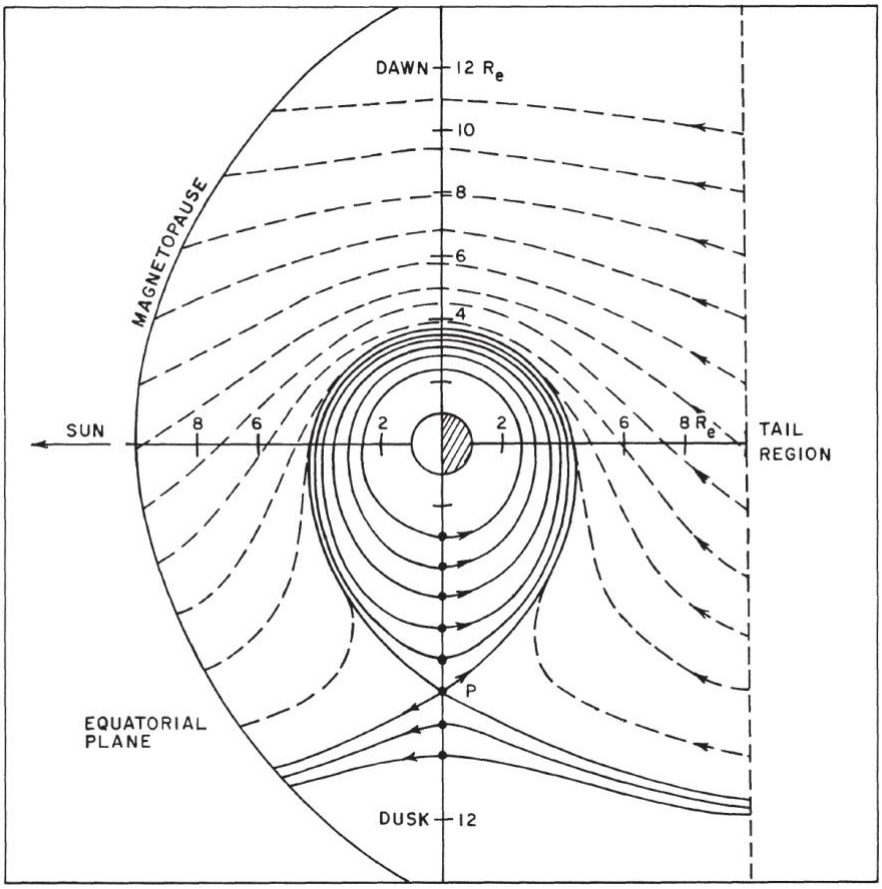
At each point, this drift is perpendicular to the field line's osculating plane. Notice that there is no curvature drift if $\alpha = 90°$.

The curvature and gradient drift always point in the same direction. We can combine these two phenomena into V_{CG}:

$$\boldsymbol{V}_{CG} = \frac{m}{2qB^2}\left(v_{\perp} + 2v_{\parallel}^2\right)\boldsymbol{e} \times \nabla_{\perp}B$$
$$= -\frac{mv^2}{2qB^2}(2 - \sin^2\alpha)\boldsymbol{e} \times \boldsymbol{n}$$

For equatorial motion, this simplifies to $\boldsymbol{V}_{CG} = -\frac{mv^2}{2qB^2}\boldsymbol{e} \times \boldsymbol{n}$.

The drift due to an electric field is $\boldsymbol{V}_E = \frac{\boldsymbol{E}\times\boldsymbol{e}}{B}$. Positive and negative electrons drift in the *same* direction. $\boldsymbol{V}_E$ is in the reference frame in which the *induced* electric field exactly cancels out the external electrostatic field. For high particle energies (≥ 100 keV), then the electric field drift can be neglected.
ß
Might be worth trying to recreate Fig 27.



ß

29/12/22

One thing that is key to consider is the drift due to the electric field. A possible model for the electrostatic potential is

$$U(r,\varphi) = q\phi(r,\varphi) = \pm\left[-\frac{C_1}{r} + C_2 r\sin\varphi\right]$$

The + (-) corresponds to protons (electrons).

The first term is responsible for a corotational radial electric field, i.e., one which in combination with a magnetic dipole field produces angular drift equal to that of the earth's rotation. The second term leads to a uniform dawn-to-dusk electric field.

In the presence of an electric field, therefore, an equatorial particle is exposed to both the gradient drift, and the electric field drift. Note that there is no curvature drift for equatorial particles.

The general recipe for computing equatorial particle drift paths is:

i) Give the initial point, initial T_i, and compute μ and initial total drift velocity $\boldsymbol{V}_E + \boldsymbol{V}_G$.

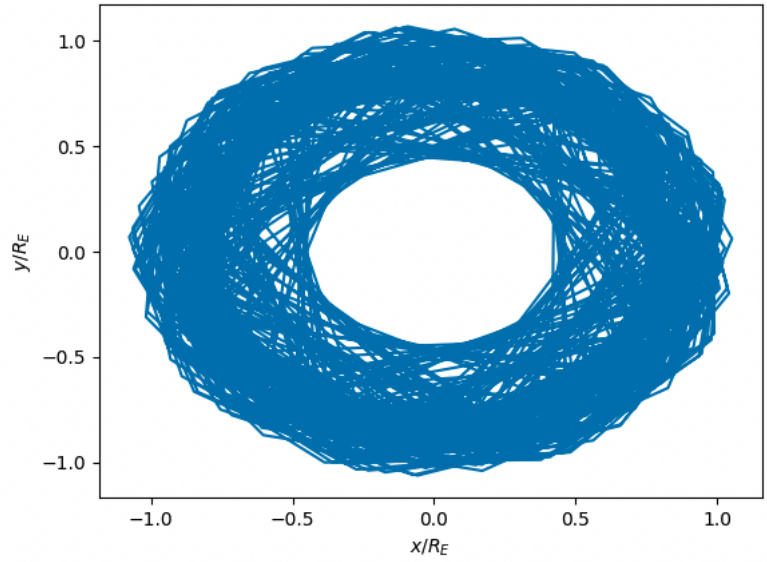
ii) Integrate the drift velocity vector. For each time step, compute the particle's kinetic energy from $T = MB$.

I'll look at this if nothing else works.

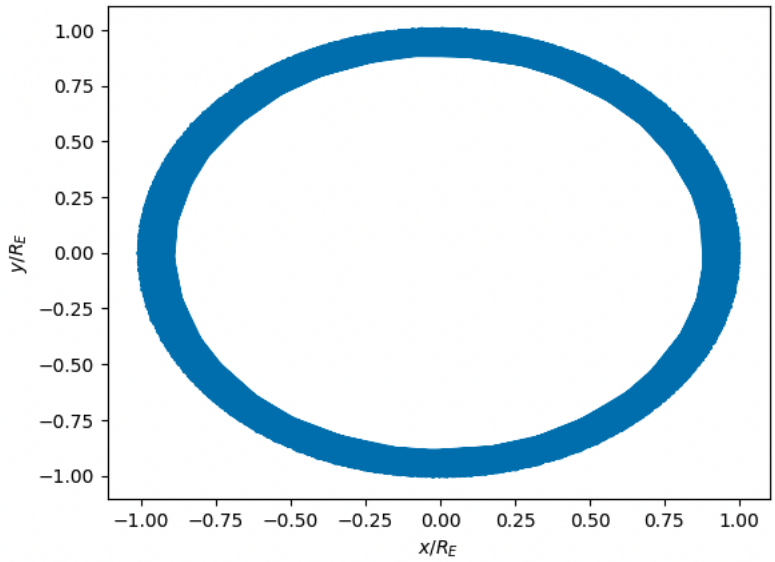
While planetary rotation is considered important for particle acceleration in Jupiter and Saturn[1], the electric field produced in the inner magnetosphere by Earth's rotation ca change the velocity of trapped particles by only 1-2 km/s. This is much less than the 100,000 km/s speeds seen for radiation belt electrons. Tomorrow, I'll take a look at Ukhorskiy+ (2014) and try and plot Eq. (1).

2/1/22

Happy late new year! I did do some work last Friday but forgot to write it up. I implemented Eq. 1.

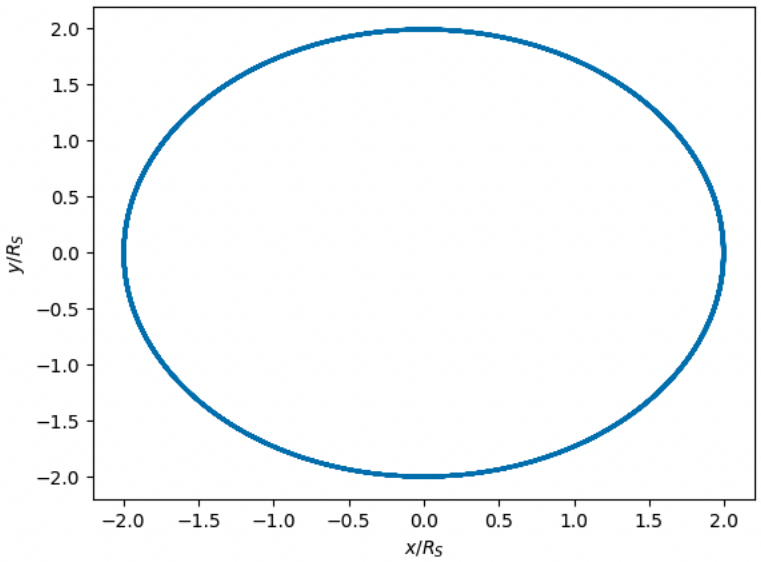


The integration method is not the best. I used the RK45 method in solve_ivp. After reading the documentation, it seems that this method should not be used for 'stiff' problems. Still unsure what this means, but I used LDSOA, which is an Adams/BDF method with automatic stiffness detection and switching and got the plot on the right. It looks much better. I guess the system of equations (Eq. 1) is stiff. It's still concerning how many loops the particle is taking; it seems to take the particle around 0.1 ms to orbit. This is way too fast. Time to do a quick estimate of how long it *should* take. Using an energy of 1 MeV, the speed is around 2.82×10^8 m/s. This gives an orbit time of approximately 0.141 s. We're around a factor of 1000 out.



[1] Brice, N. M. & Ioannidis, G. A. The magnetospheres of Jupiter and Earth. Icarus 13, 173–183 (1970), Hill, T. W. Inertial limit on corotation. J. Geophys. Res. 84, 6554–6558 (1979), Vasyliu¯nas, V. M. in Physics of the Jovian Magnetosphere (ed. Dessler, A. J.) 395–453 (Cambridge Univ. Press, 1983), Mauk, B. H. et al. in Saturn from Cassini-Huygens (eds Dougherty, M. K., Esposito, L. W. & Krimigis, S. M.) 281–331 (Springer, 2009).

I've tried using the LSODA method for the Hao+ paper. It's producing an orbit at least, but I'm unsure whether it *should*. This simulation was run for 500 hours.



4/1/23

So before I move on, the code I have for converting polar to cartesian coordinates, defined here as:

```
def pol2cart(positions):
    '''
    input: An array containing (L, phi) at each time point. Size: 2 x N,
where N is the number of time steps.
    output: An array containing (x, y) at each time point. Size: 2 x N,
where N is the number of time steps.
    '''
    N = len(positions.T)
    # initialising the array. Size: 2 x N.
    cart_pos = np.zeros((2, N))
    # go through every time step
    for i in range(N):
        # get the L and phi values at these time steps
        L = positions[0, i]
        phi = positions[1,i]
        # convert (L, phi) to (x, y)
        cart_pos[0, i] = L * np.cos(phi)
        cart_pos[1, i] = L * np.sin(phi)
    return(np.array(cart_pos))
```

Success! The following code is much faster:

```
def pol2cart_maybe_better(pos):
    return np.array([pos[0, :] * np.cos(pos[1, :]), pos[0, :] * np.sin(pos[1, :])])
```

Way faster. Running the same array of size (2, 5 000 000) through both functions gives the following output: First and second method took 6.6 and 0.2 respectively.

I'm interested in the magnitude of $\dot{L}$ and $\dot{\phi}$. Using initial values of $L, \phi = 1, 0$ gives $\dot{\phi}/\dot{L} = 240$. That's huge. Why is this so large?

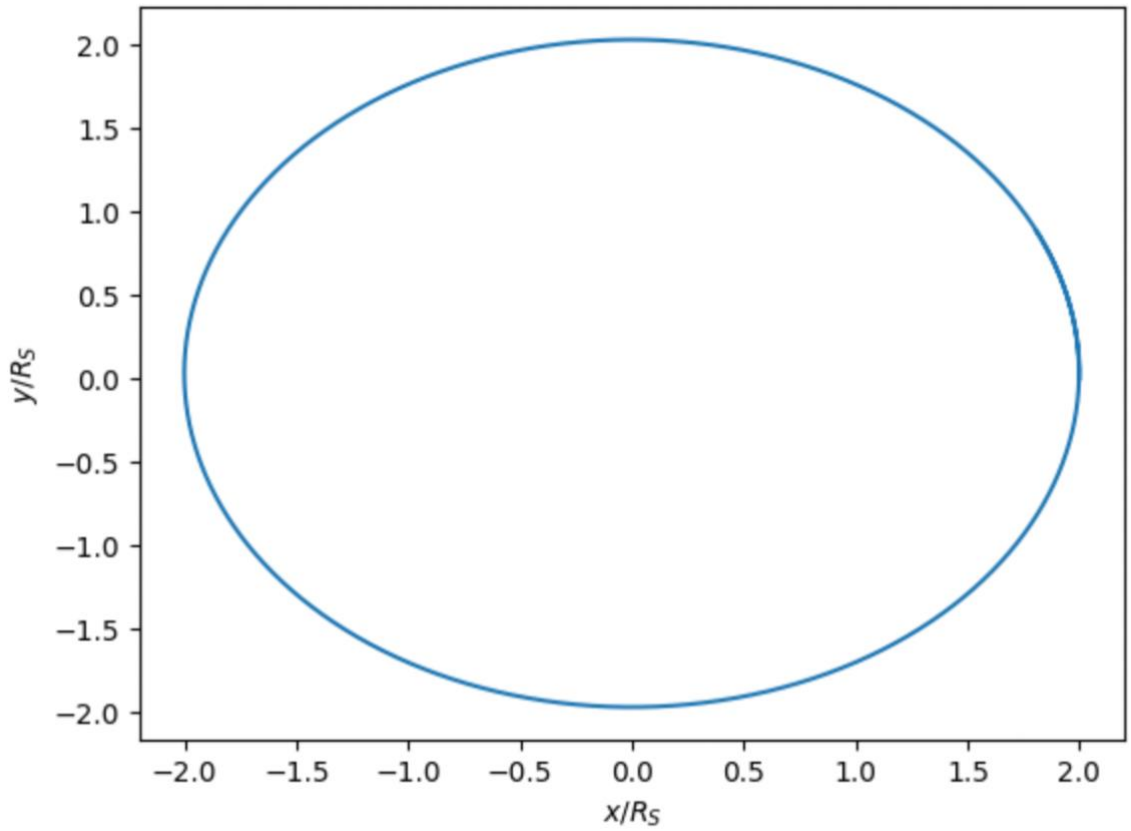
5/1/23

I have just read a paper titled *Hamiltonian theory of adiabatic motion of relativistic charged particles*. They have the same expression for $\mu = p_\perp^2/2mB$. However, the discrepancy is with γ which they've defined as $\gamma = \sqrt{1 + \frac{2\mu_0 B}{mc^2} + \left(\frac{p_{\parallel 0}}{mc^2}\right)^2}$. Funky. Let's try and implement this. Running the simulation for 60 hours is starting to take some time. It

took 28 seconds for 200 billion points. Even after this length of time, the L coordinate only changed by 0.007. This is tinnnnny. :(Need to have a think about why.

9/2/23

I have a suspicion that $\Omega(L)$ from Wilson has been implemented wrong. Looking at magnitudes, the second and third term in $\dot{\phi}$ is around 10^{-6}, while the first is 10^{-2}. For now, I've replaced $\Omega(L)$ as a constant and set it to 10^{-7}.



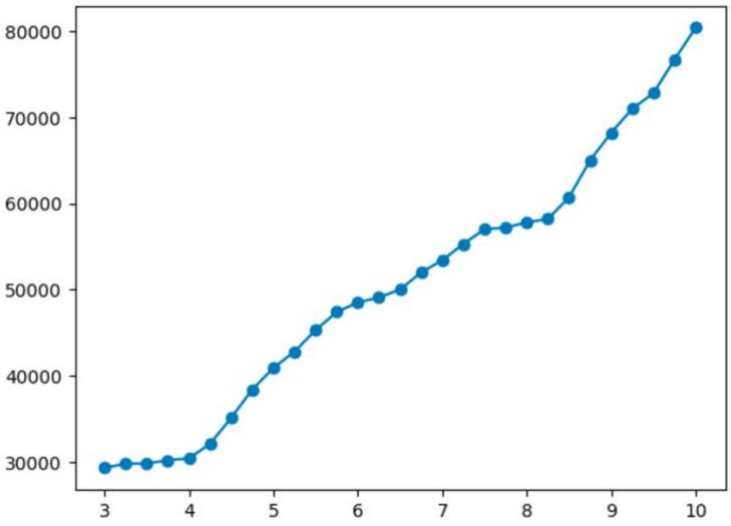
Still a circle D:

2/1/23

I had a meeting with Emma on 31/1/23 and she said that the value of α in the Hao+ paper was wrong. It is more likely that the authors used $\alpha = 90°$. She also suggested I go back to the old, proper formula for γ.

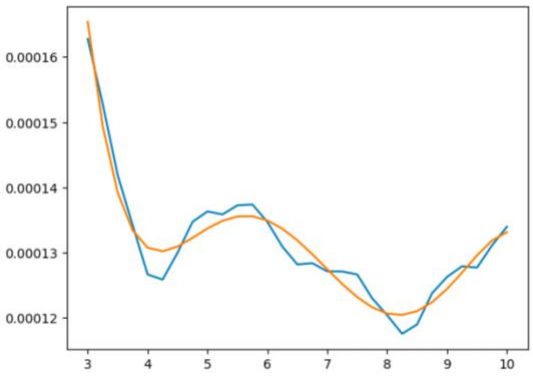
6/1/23

Turns out I've been a factor of 10 out for the v_ϕ. Here's v_ϕ against L:

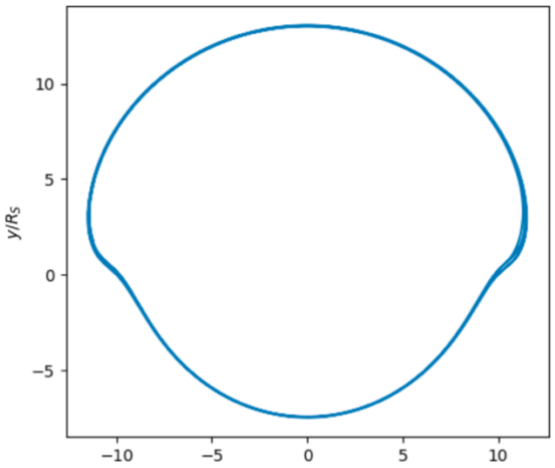


Time to implement.

Pictured below is Ω as a function of L for a fifth-order polynomial, as done in Wilson+. I might up it to a higher order polynomial later.



I have something that *isn't* a circle! No idea what this is though.



Now to figure out how to loop through energies and output the results as an image.

I have managed to save the image with the energy in MeV as the file name.

Now to put everything in a loop and try different energies.

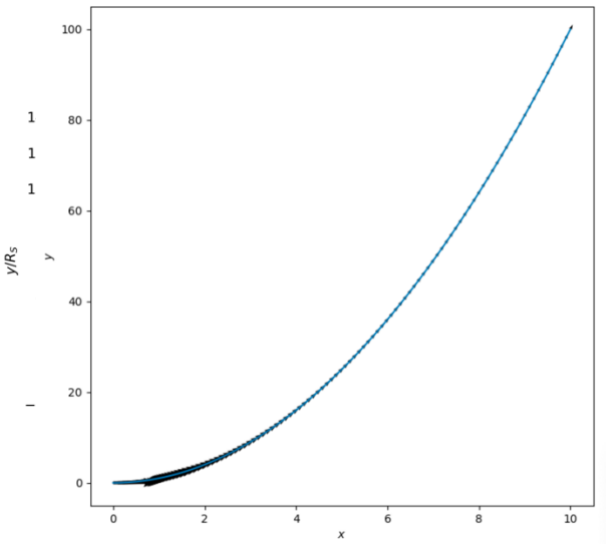
Fingers crossed D:

7/2/23

Meeting with Emma was okay. She is suspicious about $E = 0.3$ MeV. I don't blame her. To do some diagnosing, I will plot

$$\boldsymbol{v}_D = \frac{\boldsymbol{B} \times \boldsymbol{\nabla}\phi_{eff}}{B^2}$$

where $\phi_{eff} = -E_0 r \sin\phi + \frac{\mu B_0 R_S^2}{qr^3} - \frac{\Omega(L) B_S R_S^3}{r}$. The first term is convection, the second is the gradient drift, and the third is corotation.



I have managed to plot the gradient at each x value on this quadratic graph.

(2 hours)

9/2/23

I want to have the arrows at the points that I want. So I would like to pass in an array of points I want the arrows to be drawn at. Initially I thought that it would be good to be able to pass in any kind of point and an arrow drawn at the closest possible location, but this is a bit more work than I'd like right now.

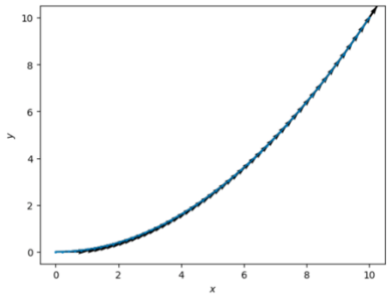


Figure 3 50 points

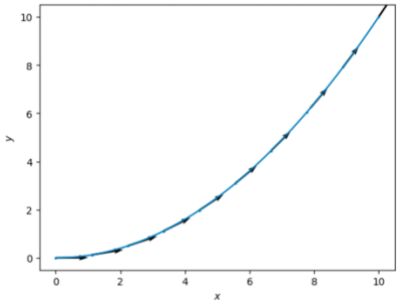


Figure 2 10 points

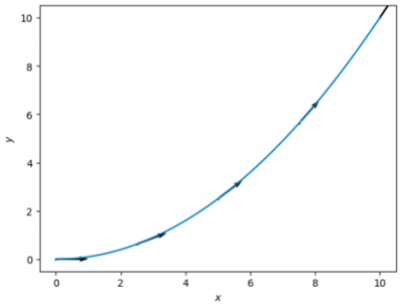


Figure 1 5 points

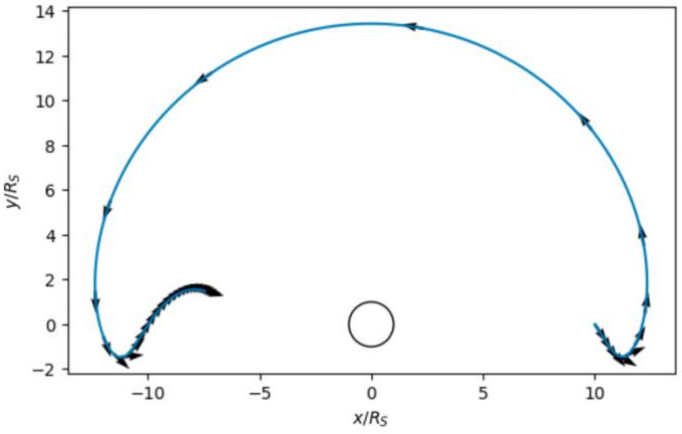
It works! Great.

Now onto finding the velocity vector at each point.

The plot_vector function is *so* slow. I need to figure out a way to make it faster. I've also realised that I assumed x would be a sorted list, which it necessarily doesn't have to be. Some new code:

```
def find_idx(arr, wanted_arr):
    arrsorted = np.argsort(arr)
    ypos = np.searchsorted(arr[arrsorted], wanted_arr)
    idx = arrsorted[ypos]
    return(idx)
```

Seems to work! There's some bunching towards the curvy bits, but not much I can do about that, I guess. I mean I *could*, but I don't want to. The point of the arrows is to plot it at several points anyway, so it's fine.



10/2/23

Today is the day I will get the vectors! (Hopefully.) $\boldsymbol{v}_D = \frac{\boldsymbol{B}\times\boldsymbol{\nabla}\phi_{eff}}{B^2}$ where

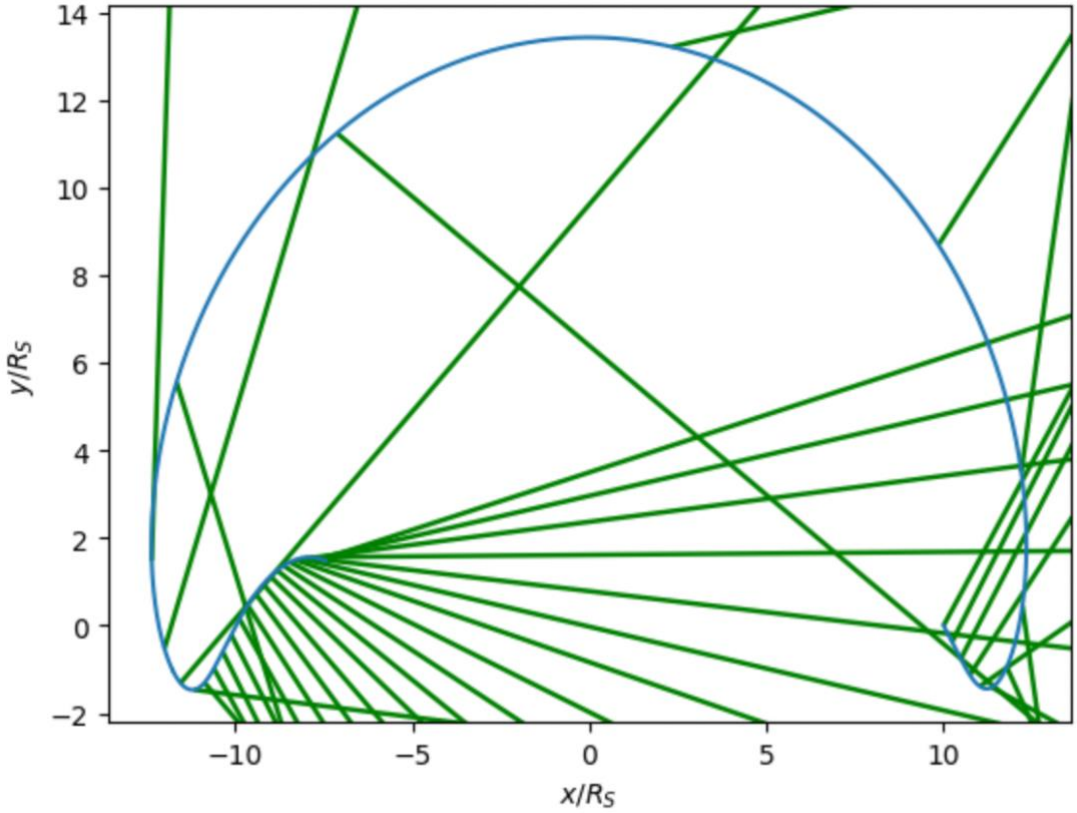
$$\phi_{eff} = -E_0 r \sin\phi + \frac{\mu B_0 R_S^3}{q r^3} - \frac{\Omega(L) B_S R_S^3}{r}$$

$\phi_{eff} = -E_0 R_S L \sin\phi + \frac{\mu B_0}{q L^3} - \frac{\Omega(L) B_S R_S^2}{L}$.

Here are the three terms in order:

```
[-0.00000000e+00 1.06523079e-10 2.13046163e-10 ... -4.44888925e-04 -4.44888915e-04
-4.44888905e-04]
[-0.00646359 -0.00646359 -0.00646359 ... -0.01475099 -0.01475099 -0.01475099]
[-1002.25659658 -1002.2565051 -1002.25641363 ... -2771.0017466 -2771.00191103 -
2771.00207546]
```

The corotation term is by far the largest.



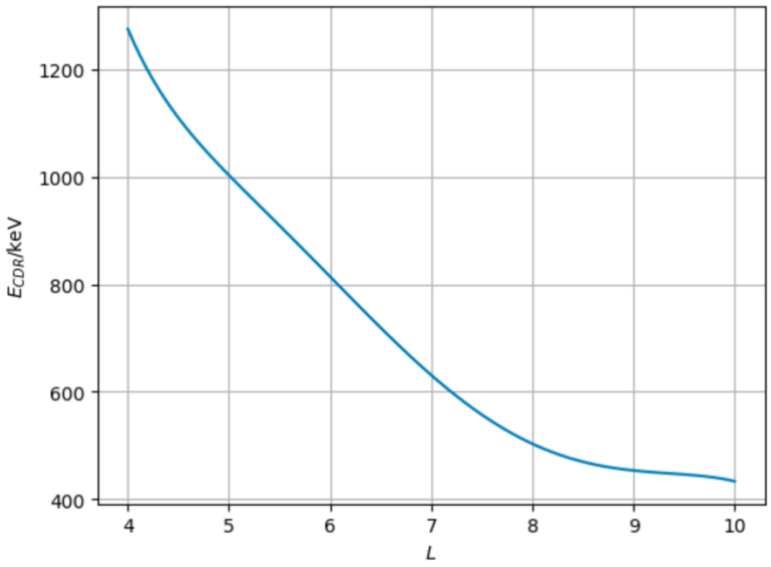
Woah.

13/2/23

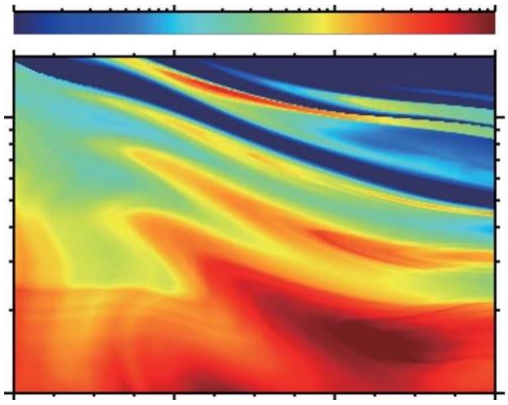
Today I will plot E_{CDR} as a function of L.

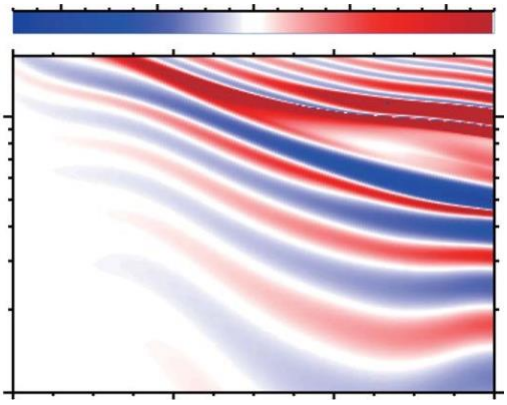
Solving for $v(L)$ first. Setting $f_{drift} = 0$ gives $v^4 + \frac{\alpha^2\Omega^2}{L^2c^2}v^2 - \frac{\alpha^2\Omega^2}{L^2} = 0$, which we can solve analytically. $\alpha = \frac{2eB_S R_S^2}{3m}$.

Here is E_CDR against L:



It has the same general shape that we'd expect





20/2/23

Haven't worked on this for a while because I've been busy with PhD applications! Steps for today are to:

1. Differentiate ϕ algebraically.
2. Find $\boldsymbol{v}$.
3. Plot these vectors.

Since we want a velocity below c, $\nabla\phi$ must be of order 10^3 or less.

8/3/23

I promise I've been working. It's been a busy couple of weeks, with the PhD interview happening. I have got plots of all the orbits from L= 4 – 10 and E=0.1 – 1.1 MeV. I need to categorise them. Today I'm trying to port my code over to Kaggle, because I can run it for much longer on their servers.

Long day. I've managed to cut the run time by half! Success.

It has been made even faster. Shaved off a couple of seconds per orbit.

11/3/23

Made the discover that changing N to 1 works perfectly well and cannot notice a difference between N = 1 and N = 1000. Saves so much time.

E = 0 gives circular orbits. So would expect for exponential decay that after a while, the orbit corresponds to a circle (although this depends on the momenta?).

17/03/23

A lot has happened since I last wrote anything down. I had a meeting with Emma, and she was happy with the code and progress.

Now, however, it seems that there's some sort of bug in the code. Changing N seems to change how far along we integrate. It's not a problem with the solve_for_L_E_k function since that works fine by itself. It must be the implementation of the orbits function.

Nevermind – I was changing t_f accidentally, instead of N. Whoops!

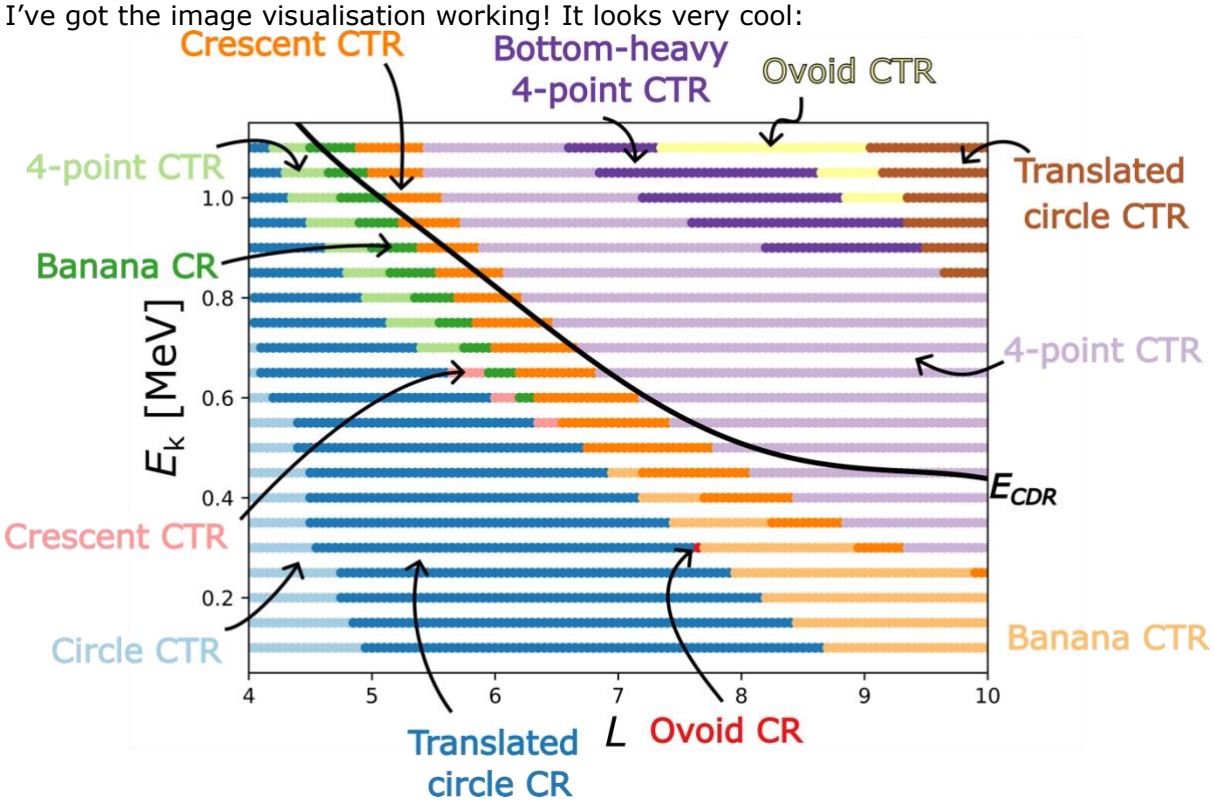
I now am plotting the orbit from blue (start) to red (end) with fewer points to make it more visible.

I've started categorising the data in an excel sheet.

18/03/23
Some of the orbits require a relative tolerance of 1e-6.

21/03/23
I have been working on the image visualisation! In order that the image is properly scaled, I have scaled up the rows (which correspond to L) so that the values are repeated the correct amount. Of course, this only works properly if the number of L values is a factor of the total length we need. This may not be the case, so I have randomly distributed the difference across each row, so that over all, it should make a smaller impact.

30/03/23
I've got the image visualisation working! It looks very cool:



I need to figure out why we don't see banana orbits that much and why they're not around the CDR energy.

05/04/23
I think that it's fine. The banana orbits don't have to be directly around the CDR energy. But the electric field is way more important than I thought. The orbits affected by radial transport occur over a much bigger range.

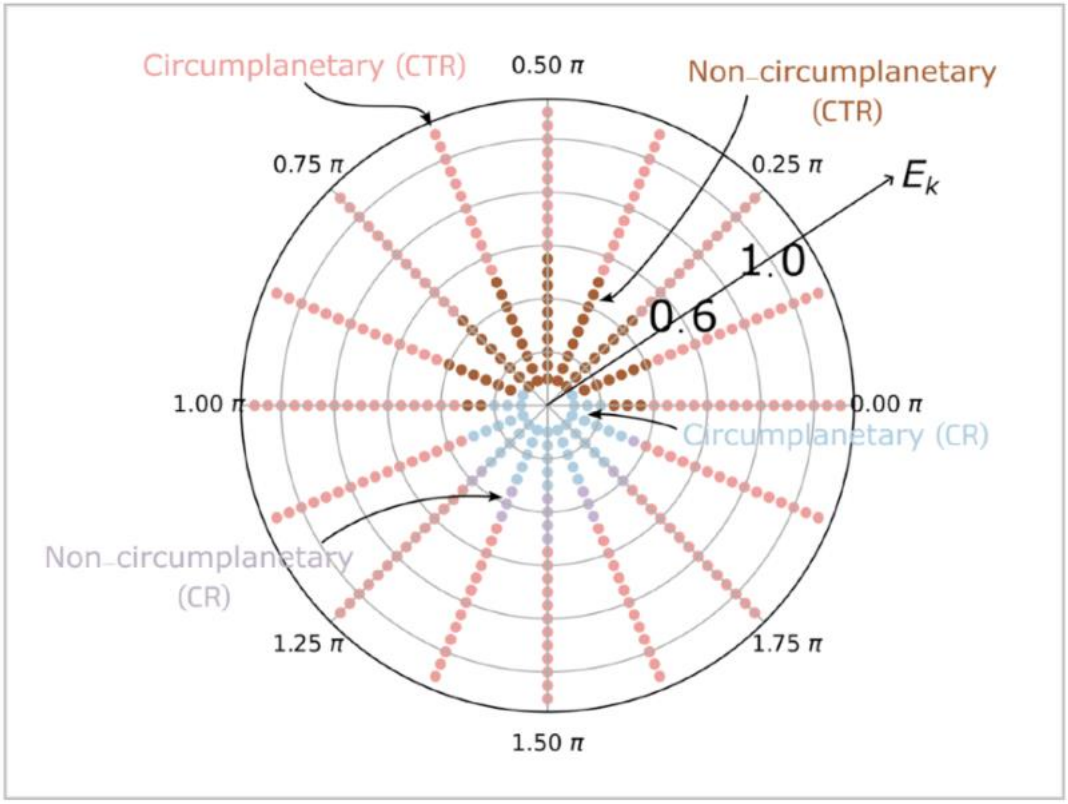
09/04/23
More lab report writing!

I'm curious about what effect the starting angle has on the orbits, so I'll test this out today.

We get way more banana orbits around 90 degrees. Not sure why this is yet.

10/04/23

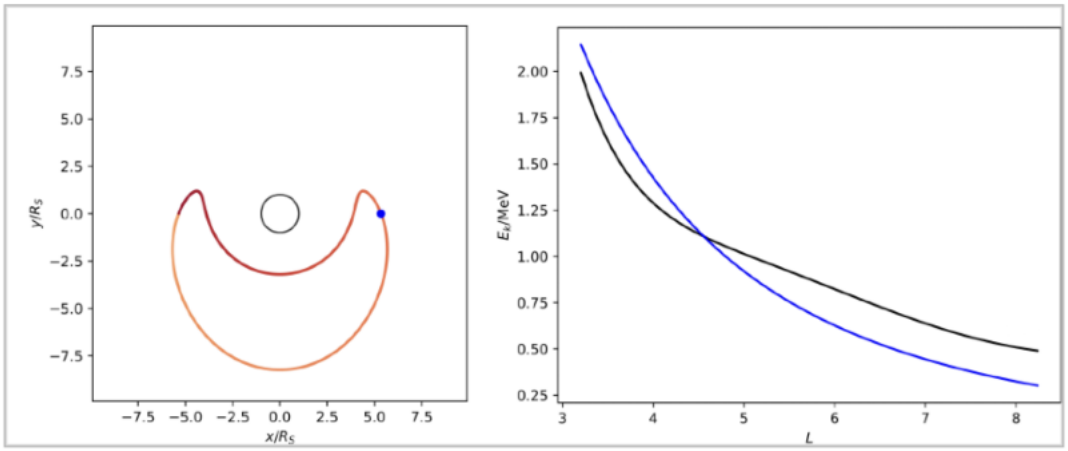
Gonna plot whether orbits are circumplanetary or CR as a function of kinetic energy and L.



There is some asymmetry in this, but also symmetry. So cool. I'll explain this in my report. Or at least try to.

16/04/23

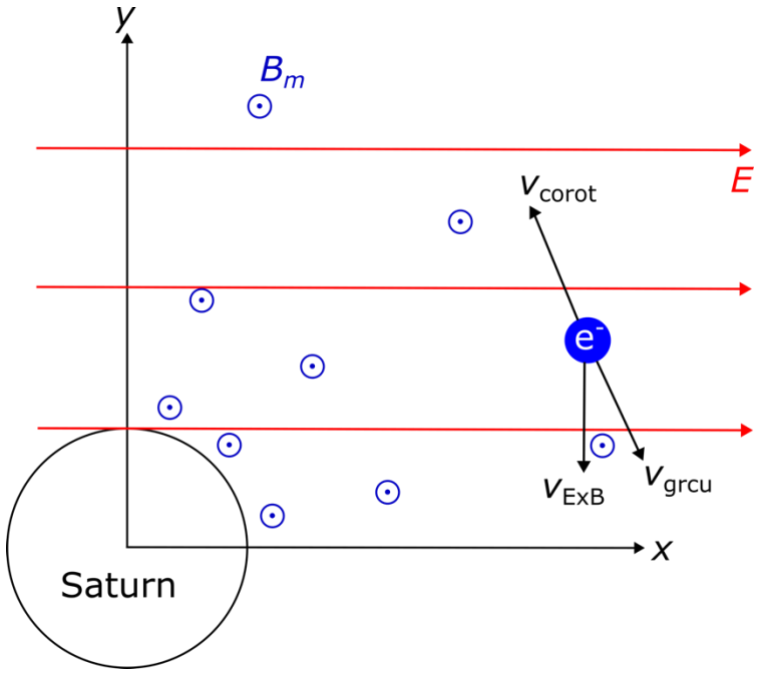
I've been trying to figure out why banana orbits are a thing – or I guess why radial diffusion happens more specifically. Lemme plot the particles energy with the theoretical CDR energy and see if there's anything.



There's a cross over around 4.5, which makes sense. At this point, the corotation and gradient drifts cancels and we get radial diffusion.

21/04/23

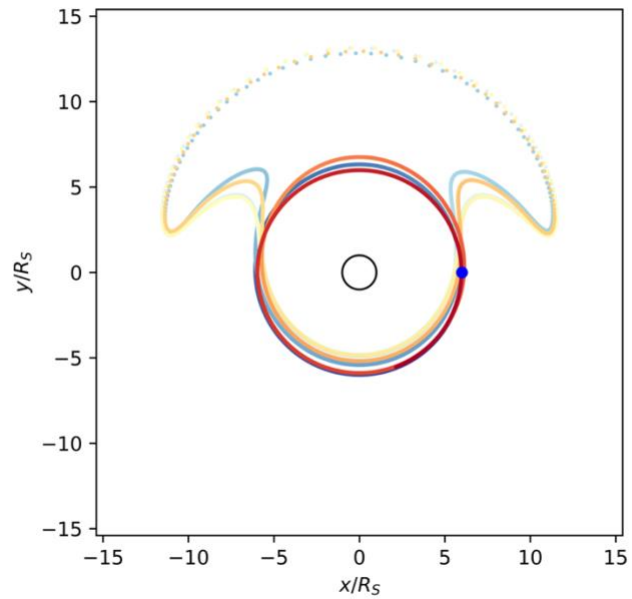
This is just more report writing. Not fun. Need to explain what the forces are on the electron. I've drawn this diagram.



24/04/23

Realised that I've accidentally implemented a dawn-to-dusk electric field, but this is fine because I can just rotate the orbits by 90 degrees.

I've also looked at a sinusoidal electric field for a more realistic field I got some funky behaviour:



It starts out as a circle, then becomes a mushroom, then back to a circle. Although where it starts and ends depends on the electric field. There is also an elongation slightly in the radial direction at noon.

1/5/23

Having read Walt, especially chapter 6 and 8, I *think* I understand the radial diffusion theory. We can parametrise the PSF with any set of independent coordinates, and transforming between them requires the Jacobian to not vanish. I think if the azimuthal angle corresponds to more than one L value, then the Jacobian vanishes. Need to explore this more though.

12/05/23

First draft is done, I think. :)))))))
Time to go through the figures and rotate them.
Also need to get an abstract done, whoops.