The Tango of Two Atoms

In my project I'll be looking at the 1D analysis of two atoms pushing and pulling on each other. This system has just two degrees of freedom: position and velocity. For my system we will be assuming one atom is always at the origin, with the other atom being our main focus.

First I'll derive the equations of motion by using the Euler-Lagrange equation, then we'll look at that equation to try and anwser what it is, what it does, and what assumptions & limitations it holds.

Then I'll throw it in our code and look at a phase diagram and try to predict certain families of solutions if there are multiple. From there I'll plot some trajectories just compare to my predicitons, then we'll play around with any parameters discussed and see what they do.

After all that, I'll turn this into D2L but this time WITH the juypter notebook, unlike the previous worked problems which in which I forgot...Sorry...

The Potential Equation

So for this system i initially thought about using the standard ocsillator equation as my potential, but i came accross a cooling looking potential called the Morse Potential:

$$U(r) = D_e(1 - e^{-a(r-r_e)})^2$$

Why use this crazy non occilating looking thing? Because the internet tells me it's better; thats why. In particular, the morse potential is better because it is a better approximation for the vibrational structure of a diatomic molecule. Wait, diatomic molecule? Thats right, I can't seem to find an anwser for if this potential works on two unbonded atoms, so we are saying our system of two atoms are bonded together as a molecule. Does this really change how we are looking at our system? Well, in my uneducated opinion, it does not. We are just looking at how close or far apart our two atoms want to be from each other, just like two unbonded atoms, just a bit closer already.

Lets go over what this different, but simple looking equation holds for us:

- r is the distance between the two atoms.
- r_e is the equilibrium bond legnth.
- D_c represents the dissociation energy of the bond between the atoms.
- a represents the depth of the potential energy well, and according to wikipedia, $a = \sqrt{k_e/2D_e}$ where k_e is the force constant, not too disimilar to our regular harmonic ocillator constant k

Assumptions/Limitations in the Potential

Chat_GBT was quite helpful in explaining the assumptions and limitations of this potential, unlike a certain search engine. Some assumptions, as I understand them, are as follows:

- The system is only between two atoms.
- Both of these atoms are not effected by neighboring atoms and their pull/pushes upon them.
- The movement derived from this potential is near the equilibrium position, as the equation becomes less accurate at greater distances. What is that distance? Couldnt find a good anwser.

Some limitations of the potential are:

- Assumes the bond legnth/equilibrium position are constant, unlike real life.
- Doesnt consider electron-electron repulsion or electron correlation which can incluence the potential energy landscape.
- Not only does it not provide a good approximation at great distances, but apperently it also doesnt do too well at very small distances either. (Really starting to question how good this potential is)
- The real world isnt perfect and atoms may not behave exactly like this potenital. (Wow I never wouldve guessed. Thanks internet.)

The Lagrangian

Alright. Its big boy latex time, here we go. Im going to leave $\dot{r}$ as v since we are in one direction.

$$\begin{aligned}\mathcal{L} = T - U = Kinetic - Potential \\ = \frac{1}{2}mv^2 - D_e(1 - e^{-a(r-r_e)})^2\end{aligned}$$

Check my math in case I'm crazy:

$$(1 - e^{-a(r-r_e)})(1 - e^{-a(r-r_e)}) = 1 - 2e^{-a(r-r_e)} + e^{-2a(r-r_e)}$$

Plug back in...

$$\mathcal{L} = \frac{1}{2}mv^2 - D_e + D_e 2e^{-a(r-r_e)} - D_e e^{-2a(r-r_e)}$$

and so here we have our lagrangian (seperated a little more to make derivatives easier)

$$\mathcal{L} = \frac{1}{2}mv^2 - D_e + D_e 2e^{-ar}e^{ar_e} - D_e e^{-2ar}e^{ar_e}$$

Lets start on the Euler-Lagrange equation:

$$\frac{\partial \mathcal{L}}{\partial q} = \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}}$$

$$-aD_e 2e^{-ar}e^{ar_e} + 2aD_e e^{-2ar}e^{2ar_e} = \frac{d}{dt}(mv)$$

$$2aD_e(-e^{-ar}e^{ar_e} + e^{-2ar}e^{2ar_e}) = ma$$

$$2aD_e(-e^{a(r_e-r)} + e^{2a(r_e-r)}) = ma$$

And so we have our equation for acceleration as:

$$a = \frac{2aD_e}{m}(e^{2a(r_e-r)} - e^{a(r_e-r)})$$

Predictions from equations

I might be crazy, but i think the only fixed point we have here is when $r = r_e$ as that will make a nice 0 appear that multiplies everything else into 0, so the fixed point in phase space would be (r_e,0). To me, the equation looks like it will occilate when r is much greater than r_e. In our exponential terms, the only difference is that the first term has an extra multiple of two in the exponent. So i'm thinking that those exponents become negative when $r > r_e$ and the exponentials essentially flip, with the first term having a smaller number overall due to the larger value in the denominator. That means the sign of the acceleration overall will stay negative as well until $r < r_e$ in which case the value of a will become positive again and try to force the atom away. Bam, oscillations!

I dont think i'm going to have many families of solutions for this one once I make the phase diagram. I think it might just end up being a bunch of circles floating around, but I honestly have no idea since I'm terrible at predicting behaviors with exponentials. I'm hoping it doesnt, because I already made a Beyonce "if you like it then you shoulda put a ring on it reference" which means i'm out of jokes to distract you from any mistakes I make.

The Code

It is time. I'm out of predictions (and out of requirments from rubric) which only leaves us with making rocks that can talk to each other do all the work.

For the parameters in our equation, I'm basing the values from the hydrogen molecule H2, but the resulting values and attempts at phase portraits arent really great so I had to scale them up.

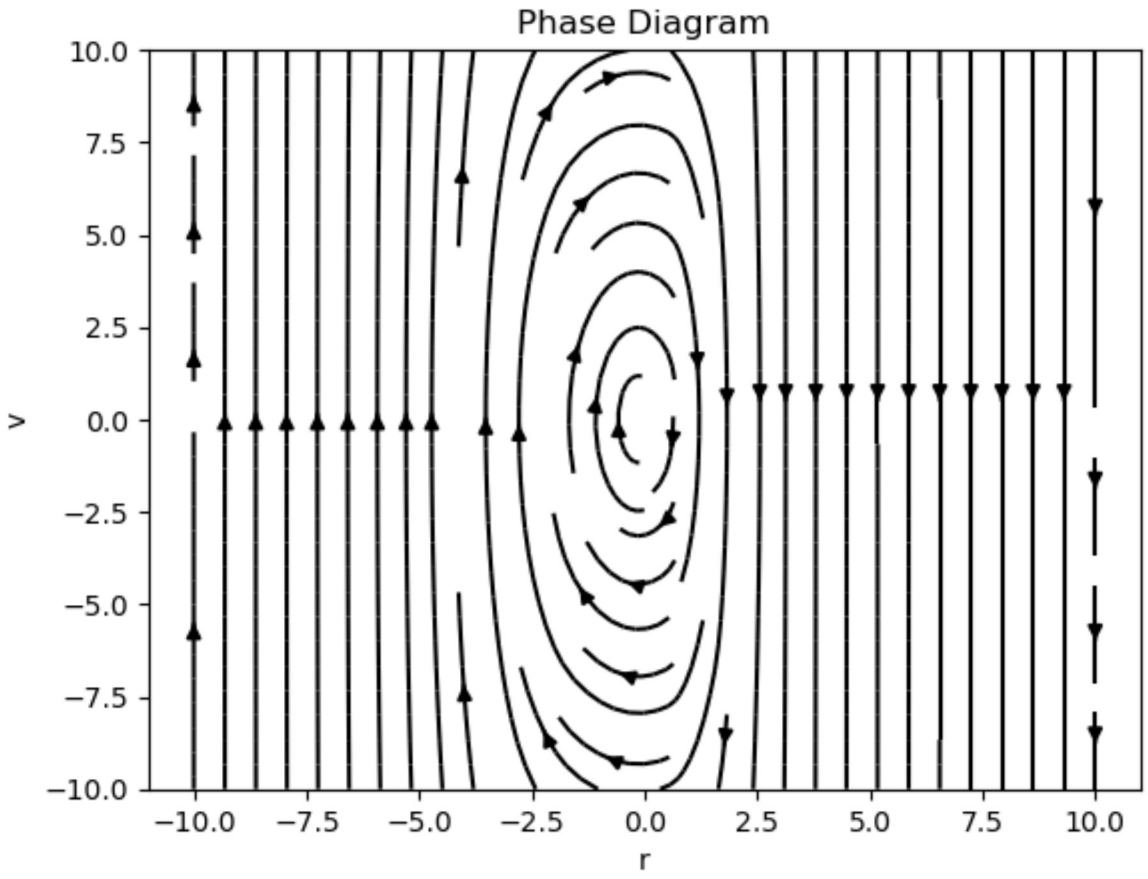
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
```

In [83]:

```
# Alot of this code is based from our class activity code
def diffyq_ivp(t,curr_vals,a,d,re):
    '''Equation for solve_ivp to use'''

    r,v = curr_vals

    vdot = (2*a*d/m)*(np.exp(2*a*(r-re))-np.exp(a*(re-r)))

    return v, vdot

def diffyq(r,v,a,d,re):
    '''Equation for the phase plotting code to use'''

    vdot = (2*a*d/m)*(np.exp(2*a*(r-re))-np.exp(a*(re-r)))

    return v, vdot

def ComputeSHOPhase(x,v,a,d,re):
    ## Prep the arrays with zeros at the right size
    xdot, vdot = np.zeros(x.shape), np.zeros(v.shape)

    ## Set the limits of the loop based on how
    ## many points in the arrays we have
    Xlim, Ylim = x.shape

    ## Calculate the changes at each location and add them to the arrays
    for i in range(Xlim):
        for j in range(Ylim):
            xloc = x[i, j]
            yloc = v[i, j]
            xdot[i,j], vdot[i,j] = diffyq(xloc, yloc, a,d,re)

    return xdot, vdot

### Variables
# original values i tried to use, but the values are so small that getting the phas
#a= 1.0  # m^(-1) ranges from around .1 to 2
#d= 1*(1.60219e-19) #values range from a few, to tens of eV.
#re= 1e-10 #m which ranges from 1 to 2
#m = 1.673e-27  #kg, the mass of the hydrogen ATOM. molecule is just x2
#ic = (2e-10,0) #(r,v)

a=1
d=-1
re=0 #just make the equilibrium point 0 basicly to hopefully run around
m=1

### Phase Diagram setup
fig, ax = plt.subplots(2, figsize = (14, 40))
#In this portion, X is r, but i didnt want to risk screwing it up by changing it
X = np.linspace(-10.0, 10.0, 20)
VX = np.linspace(-10.0, 10.0, 20)
# Get back pairs of coordinates for every point in the space
X, VX = np.meshgrid(X, VX)
# Run our calculations
Xdot, VXdot = ComputeSHOPhase(X, VX, a,d,re)
# Ploting the phase plot
plt.streamplot(X, VX, Xdot, VXdot, color='k')
plt.title('Phase Diagram')
```

```
plt.title( Phase Diagram )
plt.ylabel('v')
plt.xlabel('r')
```

Out[83]: Text(0.5, 0, 'r')



First Phase Plot Impressions

Look at our wonderful eliptical-shaped phase diagram. Those curves continue up the V axis in case your curious, steadily increasing the size of our eliptical shape, which makes me think that it doesnt matter how fast the particle might be moving, the bond will still hold and the particle will continue to osscilate around the equilibrium point. This is an example of the limitations of this potential, as we know that if you excite an atom enough, the bond between the two atoms will break and allow the atoms to be on their way.

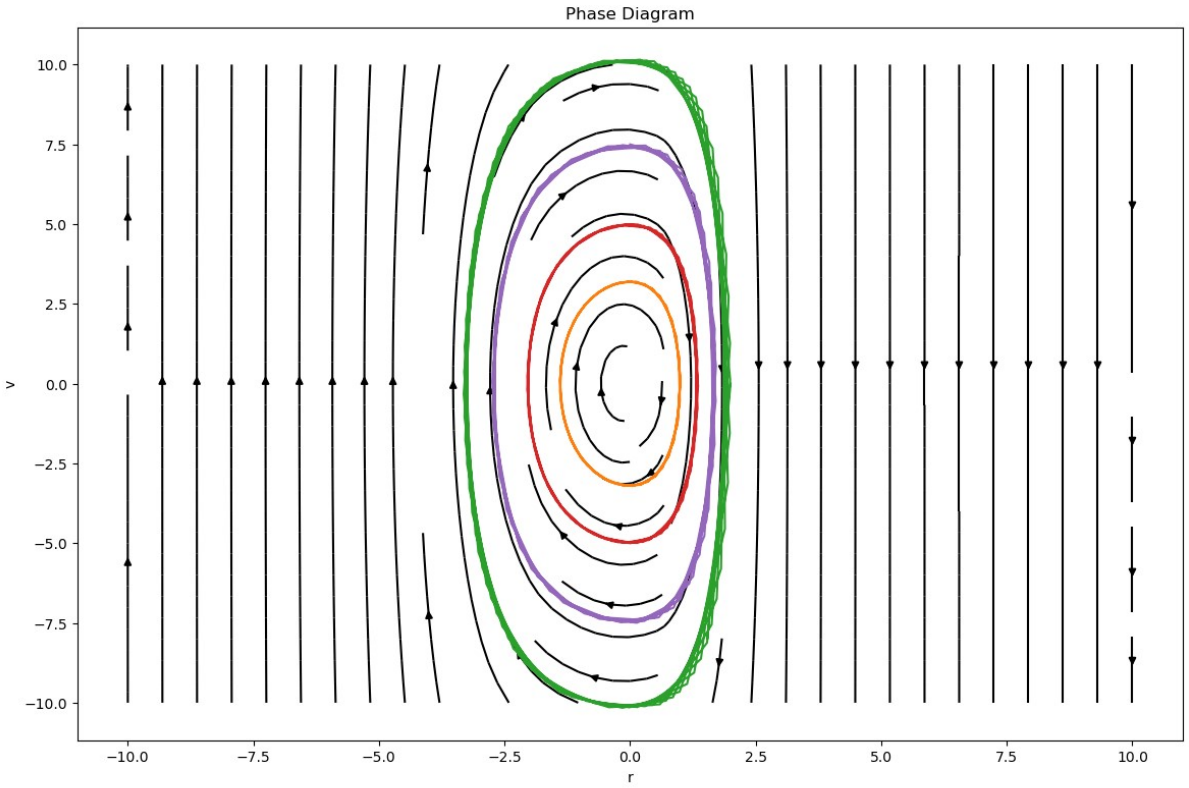
I expect the trajectories to look like the diagram and just be a bunch of rings. If I plotted the position against time, I'm sure it will show an occilating motion. Lets try it.

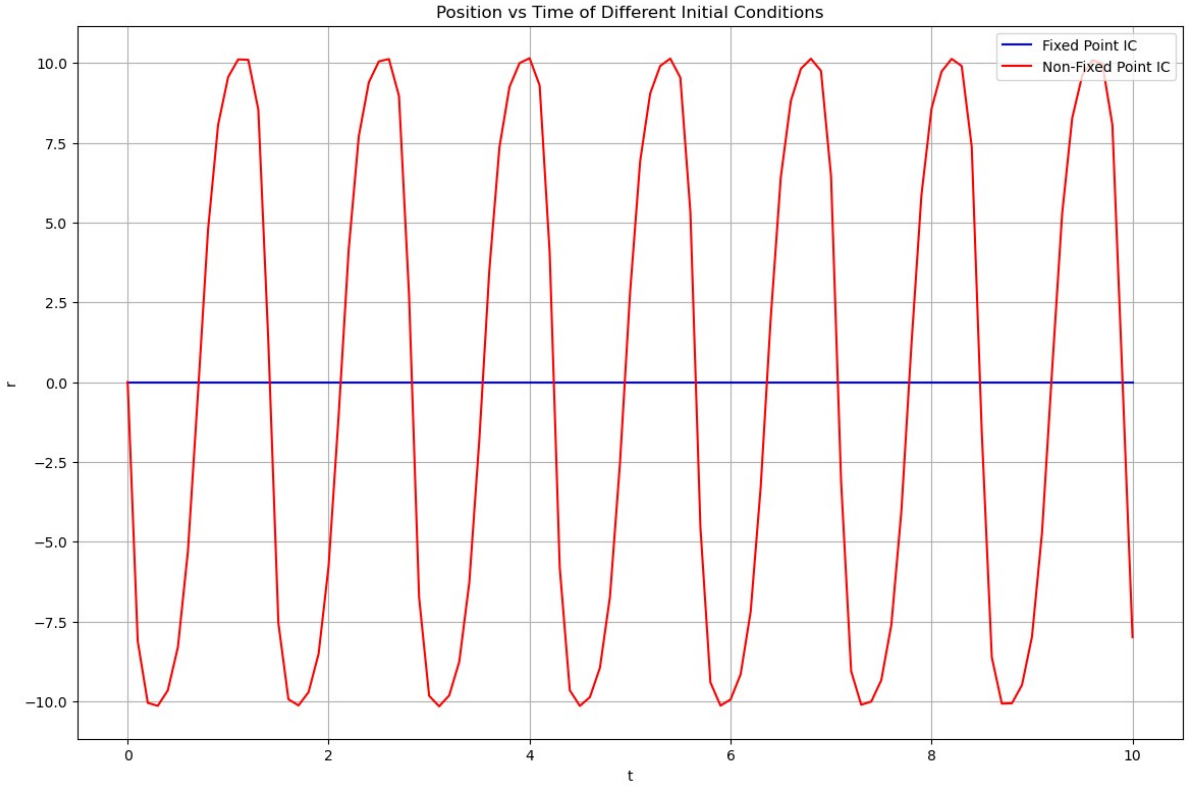
In [122…

```
a=1
d=-1
re=0 #just make the equilibrium point 0 basicly to hopefully run around
m=1

### Phase Diagram setup
fig, ax = plt.subplots(2, figsize = (14, 20))
#In this portion, X is r, but i didnt want to risk screwing it up by changing it
X = np.linspace(-10.0, 10.0, 20)
VX = np.linspace(-10.0, 10.0, 20)
# Get back pairs of coordinates for every point in the space
X, VX = np.meshgrid(X, VX)
# Run our calculations
Xdot, VXdot = ComputeSHOPhase(X, VX, a,d,re)
# Ploting the phase plot
ax[0].streamplot(X, VX, Xdot, VXdot, color='k')
ax[0].set_title('Phase Diagram')
ax[0].set_ylabel('v')
ax[0].set_xlabel('r')

### Solve IVP stuff
tmax = 10
dt = .1
tspan = (0,tmax)
t = np.arange(0,tmax+dt, dt)

ic0 = (0,0)
ic1 = (1,0)
ic2 = (2,0)
ic3 = (0,-5)
ic4 = (0,7.5)


sol0 = solve_ivp(diffyq_ivp, tspan, ic0, t_eval = t, args = (a,d,re), method = 'LSO
sol1 = solve_ivp(diffyq_ivp, tspan, ic1, t_eval = t, args = (a,d,re), method = 'LSO
sol2 = solve_ivp(diffyq_ivp, tspan, ic2, t_eval = t, args = (a,d,re), method = 'LSO
sol3 = solve_ivp(diffyq_ivp, tspan, ic3, t_eval = t, args = (a,d,re), method = 'LSO
sol4 = solve_ivp(diffyq_ivp, tspan, ic4, t_eval = t, args = (a,d,re), method = 'LSO


ax[0].plot(sol0.y[0], sol0.y[1])
ax[0].plot(sol1.y[0], sol1.y[1])
ax[0].plot(sol2.y[0], sol2.y[1])
ax[0].plot(sol3.y[0], sol3.y[1])
ax[0].plot(sol4.y[0], sol4.y[1])
ax[1].plot(sol0.t, sol0.y[1], label = 'Fixed Point IC', color = 'blue')
ax[1].plot(sol2.t, sol2.y[1], label = 'Non-Fixed Point IC', color = 'Red')
ax[1].grid()
ax[1].legend()
ax[1].set_title('Position vs Time of Different Initial Conditions')
ax[1].set_ylabel('r')
ax[1].set_xlabel('t')
```

Out[122]:

```
Text(0.5, 0, 't')
```


Phase Diagram
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
v
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
r


Position vs Time of Different Initial Conditions
Fixed Point IC
Non-Fixed Point IC
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
r
0
2
4
6
8
10
t

Parameter Play time

And thats basicly it? Its just a bunch of eiliptical/symetric rings around the equilibrium point that produce trajectories looking the same, but is this the way the atoms inside the H2 molecule act? Maybe not, as my parameters are not close to their real value since those values make it very hard to see the phase diagram itself. So, lets look at what changing the various parameters might do. I'll plot a single trajectory on them to help us better understand the changes.

I will not be messing with the equilibrium point as all it does is shift the center of the circle left or right as I'm sure you can imagine.

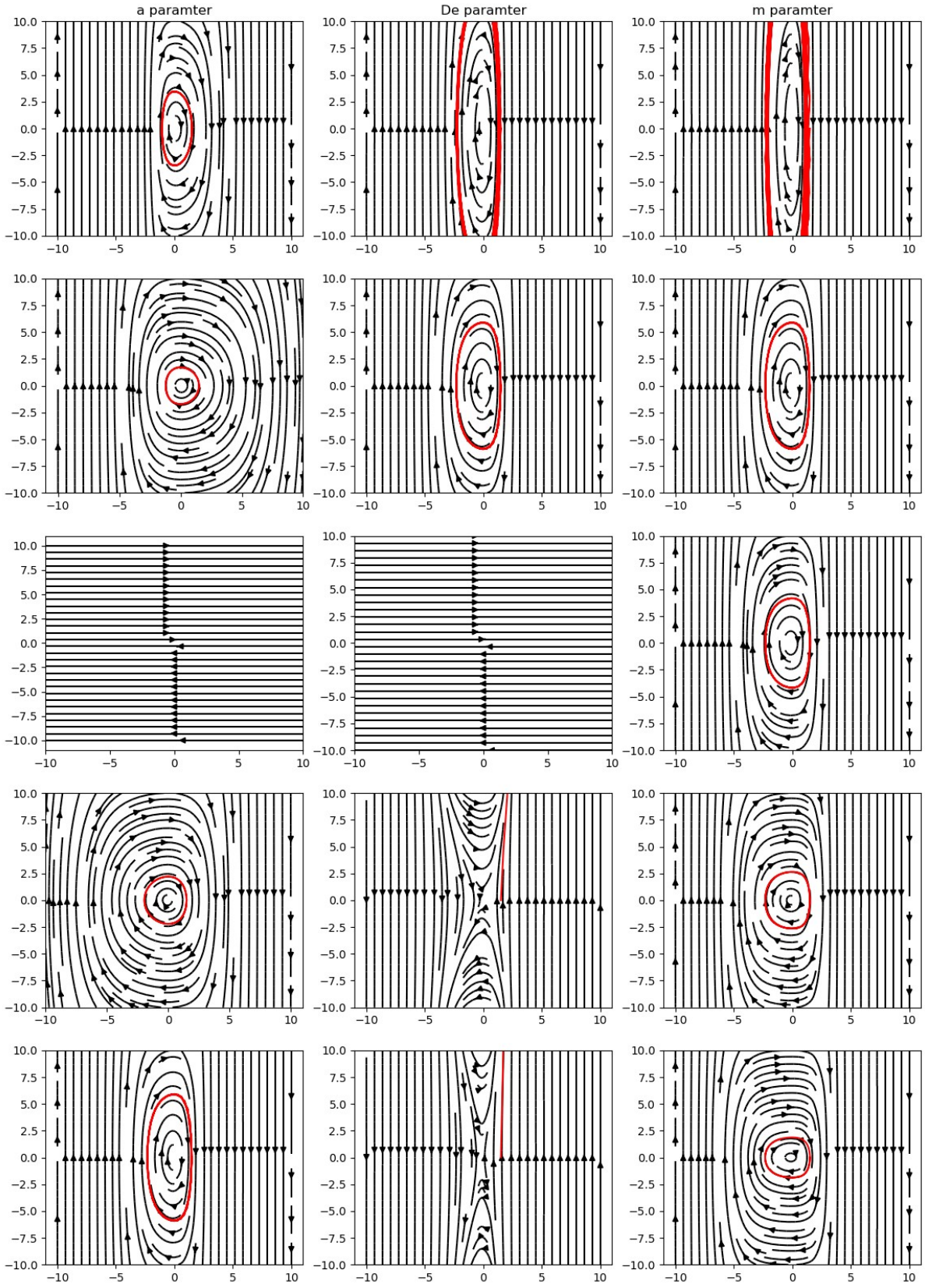
In [117…

```
### Solve IVP stuff + inital conditions for trajectory
tmax = 10
dt = .1
tspan = (0,tmax)
t = np.arange(0,tmax+dt, dt)
ic = (1.5,0)


a_list=[-1, -.5, 0, .5, 1]
d_list=[-5, -1, 0, 1, 5]
m_list=[.1,1,2,5,10]

fig, ax = plt.subplots(5,3, figsize = (14, 20))


for index in range(len(a_list)):
    d = -1
    m = 1
    re = 0
    a = a_list[index]
    ### Phase Diagram setup
    #In this portion, X is r, but i didnt want to risk screwing it up by changing i
    X = np.linspace(-10.0, 10.0, 20)
    VX = np.linspace(-10.0, 10.0, 20)
    # Get back pairs of coordinates for every point in the space
    X, VX = np.meshgrid(X, VX)
    # Run our calculations
    Xdot, VXdot = ComputeSHOPhase(X, VX, a,d,re)
    # Ploting the phase plot
    ax[index,0].streamplot(X, VX, Xdot, VXdot, color='k')
    sol = solve_ivp(diffyq_ivp, tspan, ic, t_eval = t, args = (a,d,re), method = 'L
    ax[index,0].plot(sol.y[0], sol.y[1], color = 'red')
for index in range(len(d_list)):
    d = d_list[index]
    m = 1
    re = 0
    a = 1
    ### Phase Diagram setup
    #In this portion, X is r, but i didnt want to risk screwing it up by changing i
    X = np.linspace(-10.0, 10.0, 20)
    VX = np.linspace(-10.0, 10.0, 20)
    # Get back pairs of coordinates for every point in the space
    X, VX = np.meshgrid(X, VX)
    # Run our calculations
    Xdot, VXdot = ComputeSHOPhase(X, VX, a,d,re)
    # Ploting the phase plot
    ax[index,1].streamplot(X, VX, Xdot, VXdot, color='k')
    sol = solve_ivp(diffyq_ivp, tspan, ic, t_eval = t, args = (a,d,re), method = 'L
    ax[index,1].plot(sol.y[0], sol.y[1], color = 'red')
    ax[index,1].set_ylim(-10,10)
for index in range(len(m_list)):
    d = -1
    m = m_list[index]
    re = 0
    a = 1
    ### Phase Diagram setup
    #In this portion, X is r, but i didnt want to risk screwing it up by changing i
    X = np.linspace(-10.0, 10.0, 20)
    VX = np.linspace(-10.0, 10.0, 20)
```

```
    VX = np.linspace(-10.0, 10.0, 20)
    # Get back pairs of coordinates for every point in the space
    X, VX = np.meshgrid(X, VX)
    # Run our calculations
    Xdot, VXdot = ComputeSHOPhase(X, VX, a,d,re)
    # Ploting the phase plot
    ax[index,2].streamplot(X, VX, Xdot, VXdot, color='k')
    sol = solve_ivp(diffyq_ivp, tspan, ic, t_eval = t, args = (a,d,re), method = 'L
    ax[index,2].plot(sol.y[0], sol.y[1], color = 'red')
    ax[index,2].set_ylim(-10,10)

ax[0,0].set_title('a paramter')
ax[0,1].set_title('De paramter')
ax[0,2].set_title('m paramter')
```

```
C:\Users\jwill\AppData\Local\Temp\ipykernel_13008\4068161443.py:7: RuntimeWarning:
overflow encountered in double_scalars
  vdot = (2*a*d/m)*(np.exp(2*a*(r-re))-np.exp(a*(re-r)))
```

Out[117]:
```
Text(0.5, 1.0, 'm paramter')
```


a paramter
De paramter
m paramter

Each column represents 1 of the 3 parameters we are playing with, with 5 values of each representing each row. The values I used for each parameter are different, but so you dont have to look through the code to find it, the values from top to bottom are as follows:

- 1st Column: a = -1, -.5, 0, .5, 1
- 2st Column: d = -5, -1, 0, 1, 5
- 3st Column: m = .1, 1, 2, 5, 10

So lets look at what effects we have. We'll start from least interesting to most interesting.

Leading the way as the most boring parameter we have *drumroll* MASS (m)! so obviously mass wasnt going to be very impactful as it is essential just a scaling constant of how much of an effect the force has on our atom. We see that a high mass makes the atom harder to move, and as such never gets to higher speeds unlike our low mass values which are shown to move much faster with the same initial conditions.

Up next we have the dissociation energy (D_e) which in the equation works just like mass does as a scaling constant. The interesting part is when you forget that D_e should be negative when you plug it into your equation as i found out when first trying to produce the phase diagram. I started with a positive value which made our system repulsionary (is that a word?) instead of attractive and really just threw me for a loop. But assuming you've got that negative sign there it does indeed just act like mass does.

Our winner for most interesting is of course, our lovely... *checks notes*... oh its just called a, but its cool. Now, I'm not entirely certain I can justify a ever being negative, since its supposed to just represent the depth of the potential well, but it does produce an interesting pattern when you compare them. You can see that when the value is zero, nothing cool happens and the atom will just continue on its path. But when you watch the phase diagram approach zero from the negative side, you see the eliptical orbit, for lack of a better analogy, grow a beer belly towards the positive r direction. Of course when we approach from the other side, the same beer belly grows towards the negative r direction. The fact that the possible elliptical paths get smooshed together as $a \to \pm\infty$ makes sense though since as the potential well gets deeper, it should be harder to escape it. The weird assymentrical growing of the phase diagram is what I cant explain though, but thats why there are smarter people in the world; to figure that out instead of me.

BONUS if you make a negative, and D_e positive, the lines of the phase diagram look like the positive D_e examples i gave you before, but rotated onto its side. Its a cool thing that I cant reproduce when D_e is negative.

THE END

Sorry if I missed a bunch of spelling errors when finalizing this, the only spelling I can do is the magical kind.

References

Wiki page - https://en.wikipedia.org/wiki/Morse_potential

Most of the code is shamelessly stolen from class activities.

ChatGBT - I've heard people want us to cite prompts used to generate the responces, but i literally just begged the ai to give me ideas with 2 degrees of freedom, then it came up with the morse potential, and then i just asked how it worked and what the limitations are. I can pull the conversation up on my computers at anytime so if you need them just let me know.

- Not sure if you care about ChatGBT, but i'm just covering my bases.

In [ ]: