MAS2008 ASSIGNMENT 2:
INVESTIGATING A DYNAMICAL SYSTEM

1. Assignment

In this assignment you will investigate the dynamics of a version of the Hénon-Heiles system which is the following pair of second-order differential equations:

$$\ddot{x} = -x + x^2 - y^2$$
$$\ddot{y} = -y - 2xy.$$

Here dots denote derivatives with respect to time, so $\ddot{x} = \frac{d^2x}{dt^2}$, $\ddot{y} = \frac{d^2y}{dt^2}$. We can recast these equations as a system of first-order equations in four variables (x, y, p, q) as follows:

$$\dot{x} = p$$
$$\dot{y} = q$$
$$\dot{p} = -x + x^2 - y^2$$
$$\dot{q} = -y - 2xy$$

You will need to submit a Jupyter notebook containing text cells with explanations and mathematical arguments, as well as code cells with Python code to complete various tasks. Your notebook should be self-contained, and and designed to be comprehensible to a second-year mathematics student who has not read this briefing document. Further details of the submission format are in Section 2.

1.1. Theorems.

Definition 1.1. We define the potential energy V, the kinetic energy T and the total energy E as follows:

$$V = \tfrac{1}{2}x^2 + \tfrac{1}{2}y^2 + xy^2 - \tfrac{1}{3}x^3$$
$$T = \tfrac{1}{2}\dot{x}^2 + \tfrac{1}{2}\dot{y}^2 = \tfrac{1}{2}p^2 + \tfrac{1}{2}q^2$$
$$E = V + T.$$

Carry out the following tasks. For some of them you will be able to find answers on the internet, but you will need to rewrite them in your own words using the same notation and terminology as in this brief. For some of them it is unlikely that you will find exactly the right thing on the internet (although you may find something similar) so you will need to do your own work.

Task 1.2. Show that the total energy E is conserved, i.e. $\dot{E} = 0$.

Task 1.3. In terms of the complex variable $z = x + iy$, show that $\ddot{z} = -z + \overline{z}^2$ and $V = \frac{1}{2}|z|^2 - \frac{1}{3}\mathrm{Re}(z^3)$ and $T = \frac{1}{2}|\dot{z}|^2$.

Task 1.4. The point $a_1 = (1, 0)$ is stationary, i.e. we have a solution to the differential equation with $x = 1$ and $y = 0$ for all t (so $p = \dot{x} = 0$ and $q = \dot{y} = 0$). In total there are four stationary points, which we will call a_0, a_1, a_2 and a_3. Find these points.

Task 1.5. Show that

$$V - \tfrac{1}{6} = (x + \tfrac{1}{2})(y - (x-1)/\sqrt{3})((y + (x-1)/\sqrt{3})).$$

and thus describe the set of points where $V = 1/6$. For this you just need to do some algebraic manipulation. You can either

(a) Do it by hand, and write the calculation in LaTeX in a text cell in your notebook.
(b) Work out how to get sympy to do it within Python.
(c) Ask Wolfram Alpha or some similar online service to do it.

(d) Use a symbolic mathematics program such as Mathematica, Maple, Sage or Maxima. (Note that Mathematica and Maple are not free but the university has site licenses so you can get them from IT Services. The other two are free.)

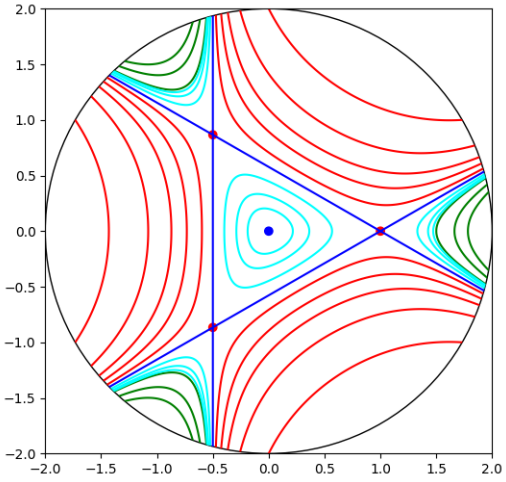
In cases (c) or (d), your notebook should record exactly what you entered and what result you received. If you want to use a symbolic mathematics program to check that $A = B$, you should generally ask it to simplify $A - B$ and check that you get zero.

Task 1.6. Show that at points of the form

$$(x, y) = \frac{3 + s^2}{2\sqrt{3}(1 - s^2)}(\sqrt{3},\ s)$$

we have $V = 0$. Again, this is just a matter of algebraic manipulation, which you can approach by any of the methods in the previous task.

1.2. **Contours.** Consider the following picture, which shows the contours $V = c$ for various values of c.

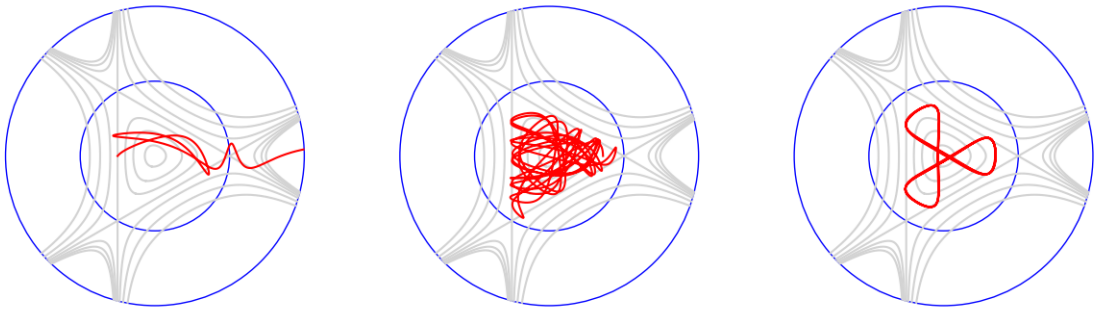


- The values of c are chosen to give a set of curves that are fairly evenly spaced.
- Different colours are used for the cases $c \leq 0$, $0 \leq c < 1/6$, $c = 1/6$ and $c > 1/6$.
- The four stationary points are also shown.
- The initial grid has been made fine enough to ensure that the curves are nice and smooth.
- The picture also shows a circle of radius 2, and the `set_clip_path` has been used to hide everything outside of that circle.

Write a function `draw_contours(R=2, n=500)` which produces a plot as above, with a clipping radius of R and an initial grid of size $n \times n$ for finding the contours. (It is not so obvious how to use `set_clip_path()` correctly, but you can ask an AI assistant for examples, or just search the web.)

The picture is clearly unchanged if we rotate everything through a 1/3-turn. Do some mathematical analysis to explain this. (You may choose to work with complex variables as in Task 1.3.)

1.3. **Trajectories.** Here are some sample solutions for the differential equation, with different initial conditions:



The steps below will build up to generating pictures of this kind.

First, define a function `escape(t,u)`, such that `escape(t,[x,y,p,q])` is positive when (x, y) lies inside the circle of radius 2 centred at the origin, and negative when (x, y) lies outside that circle. (Note that t is unused but needs to be included anyway for the following steps to work correctly.) Then you should enter `escape.terminal=True`.

Now define a function `solve(u0)`. This should accept an argument `u0=[x0,y0,p0,q0]` giving initial conditions for the differential equation, and it should call `scipy.integrate.solve_ivp` with `u0` and some other arguments to solve the differential equation from that starting point. By default you should solve the equation for $0 \leq t \leq 10$ with 1000 time steps, but your function `solve(u0)` should have optional arguments so that these defaults can be overridden. You should also pass the argument `events=[escape]` to `solve_ivp()`. This will cause the solution process to stop if the solution reaches the circle of radius 2. The `solve()` function should just return the result produced by `solve_ivp()`.

Now define a function `show_trajectory(u0)` which calls `solve()` and then plots the result together with some other features. These features should certainly include the circles of radii one and two, and the three straight lines that give the edges of the central triangle. You could also include some contours of V, as in the pictures above. If you want to include contours then ideally you should not recalculate them for each plot, but should work out how to calculate them once and save them so that they can be added to each new plot without extra work. However, this is not so easy to arrange; it is a challenge for enthusiasts.

Now experiment with different initial conditions to generate some example plots.

- If you start with (x, y) in the central triangle then the initial potential energy V will be less than $1/6$. If the initial values of p and q are also small then the total energy E will be less than $1/6$. As E is constant and $T \geq 0$ we see that V can never reach $1/6$ so the trajectory will be confined to the middle triangle.
- If the energy is less than $1/6$ but close to $1/6$ then the trajectory will usually be quite chaotic and will fill out most of the middle triangle.
- If the initial condition is $(x, 0, 0, q)$ with x and q fairly small, then the trajectory will usually be a complicated curve circulating around the origin. However, if you fix x and adjust q carefully then you can make the solution join up with itself to give a nice simple closed curve.
- If the initial speed is higher, so that the total energy is greater than $1/6$, then the trajectory will typically escape from the central triangle and then run out to infinity, either out to the right (which we call direction 1) or top left (direction 2) or bottom left (direction 3). Usually the escape will happen quickly but if the energy is only a little higher than $1/6$ and the initial conditions are adjusted carefully then escape can be delayed.

1.4. **Escape time.** Write a function `escape_time(u0)` that returns the first time that the solution hits the circle of radius two (if we start with initial condition `u0` at $t = 0$). Here you should let the solution run for $0 \leq t \leq 100$ with 10^4 time steps, and you should return `np.inf` if the trajectory has still not hit the circle at $t = 100$. (If you have defined the `solve()` function correctly and understood properly how it works, then the `escape_time()` function can be very short.)

Now fix a speed v, say $v = 0.5$, and consider initial conditions $(x, y, p, q) = (-0.5,\ y,\ v\cos(\theta),\ v\sin(\theta))$ with $|y| \leq \sqrt{3}/2$ (so (x, y) lies on the left hand edge of the middle triangle and $\sqrt{p^2 + q^2} = v$). Try to find

initial conditions of this type that make the escape time as large as possible. One approach is just to loop through possible values of y and θ. If you do this, you should first try it with a small number of values, say 4 different values for y and 4 different values for θ making 16 trajectories in total. Once you are sure that your code works correctly you can loop through a much larger number of values, and leave your computer to work on it for an extended period. Every time you break the record for the escape time you should print the values of y and θ so that progress is not lost if the process crashes. You should put your code in a function `find_long_trajectory(v)`. The version of the notebook that you submit should contain the definition of `find_long_trajectory(v)` and should contain plots of the longest trajectories that you found but it should not actually call `find_long_trajectory()`.

As an alternative to running a loop as above, you could work out how to use `scipy.optimize.minimize()` to achieve the same thing. That function has an argument called `method`; I have found that `method='Nelder-Mead'` gives the best results in this context. Note that maximizing the escape time is the same as minimizing the negative of the escape time.

2. Guidance

- Your notebook should be self-contained. It should be possible for someone to read through your notebook and understand what you are doing and why, without needing to refer to this briefing document.
- You may discuss the assignment with other students, but you must not copy code or text from them. You must write your own notebook in your own words based on your own understanding. You must also mention any collaboration in the acknowledgements section of your notebook, including the names of people with whom you worked.
- You may search the internet for information, but you must mention all sources that you have used in your acknowledgements, with specific URLs.
- You can use code that you find on the internet or that is given to you by an AI assistant, but you must acknowledge it. You must also ensure that the code you submit complies precisely with the notation and terminology used in this briefing document, and that the function names, arguments and return values are exactly as specified. You will probably need to modify code obtained from elsewhere to achieve this.
- The briefing asks you to define various functions. These functions should not print or plot anything unless the briefing specifically says that they should.
- All acknowledgements must appear in a separate markdown cell at the top of your notebook, with heading "Acknowledgements".
- All nontrivial functions should have docstrings.
- For all code that implements a nonobvious algorithm, you should add detailed comments in the code to prove that you understand how the algorithm works.
- When developing your notebook, you will probably move backwards and forwards, inserting things and executing code in different places. However, before submission you should tidy up your code. Remove anything that is not needed and check that the rest can be executed in order from the top to the bottom without errors and that this generates all the required plots and prints all the required messages. If you wish to execute any code that will cause an error, then you should write a markdown cell explaining the situation, and then wrap the offending code in a `try ... except` block like this:

```
try:
    bad_function(666)
except Exception as e:
    print(f"As explained above, we get an error as follows: {e}")
```

- 20% of the mark for this assignment will be for presentation. This will include organisation of the notebook into sections, effective use of LaTeX, spelling and grammar, and similar things.
- Upload your notebook using Blackboard.
- Do not include your own name or registration number in your notebook. (Blackboard will ensure that your work is tagged with your name at the point when that becomes necessary.)