Phase space of the hydrogen atom in a circularly polarized microwave field

Pol Navarro Pérez

July 1, 2025

Abstract

Abstract

This study investigates the dynamics of the hydrogen atom subjected to circularly polarized microwave fields, focusing on ionization processes and orbit classification. Starting from the simplest case ($K = 0$), we analyzed the behavior of unbounded regions and their role in leading to ionizing orbits as $K \neq 0$. Through numerical methods, we explored the influence of invariant manifolds and identified conditions that facilitate ionization.

1 Introduction

Dynamical systems are mathematically modeled by equations such as:

$$x_{k+1} = f(x_k), \text{ with } K \in \mathbb{Z} \text{ , for maps} \tag{1}$$

$$\frac{dx}{dt} = f(x), \text{ with } t \in \mathbb{R}, \text{ for vector fields} \tag{2}$$

Sometimes, these equations cannot be solved analytically. In such cases, alternative methods are employed to analyze the system's behavior. These methods not only aid in understanding the nature of the problem but also provide a structured approach to determine what to search for and how to do so.

In our reference paper [1], numerical methods have been applied to a hydrogen atom interacting with a circularly polarized microwave field, this is the *"CP problem"*. The main goal has been to study the possible paths of the electron to escape or ionization, depending on the distance to the nucleus.

In the study of a dynamical system, the natural approach often begins with identifying the simplest solutions—equilibrium points—and progressively exploring more complex invariant structures, such as the stable and unstable manifolds of equilibria, homoclinic connections, periodic orbits (PO), and their associated manifolds. This process reveals how invariant tori and other dynamical barriers, such as bottlenecks near unstable periodic orbits, shape the system's behavior and influence transitions like ionization or escape.

Similar methods have been applied to other fields; for instance, [2] investigates the origin of irregular moons around giant planets using chaos-assisted capture mechanisms. In atomic physics, [3] frames the hydrogen atom as a nonlinear oscillator under perturbation, exploring its interaction with circularly polarized light and linking it to complex dynamics such as periodic orbits and chaos. These studies provide a broader perspective on the applicability of dynamical systems methods across different domains.

In this work, we will closely follow the steps outlined in [1], sharing the same objectives. Our approach will involve exploring methods to reach similar conclusions through numerical techniques. In certain cases, we will discuss the computational challenges encountered and the strategies employed to address them in pursuit of our goals. The aim is not to replicate every graph presented but rather to critically assess the claims made, explore potential strategies, or at least provide some insight into possible approaches.

2 Background of the CP Problem

The starting point of the work is the Hamiltonian for a hydrogen atom under the influence of a microwave field, this is:

$$H = \frac{1}{2}(p_x^2 + p_y^2 + p_z^2) - \frac{1}{r} + F(x\cos(wt) + y\sin(wt)), \tag{3}$$

where (x, y, z) and (p_x, p_y, p_z) are the canonical coordinates and their conjugate momenta, $r^2 = x^2 + y^2 + z^2$, w is the angular frequency of the microwave field, and $F > 0$ is the field strength. We work in the planar case ($z = 0$) and consider $K = \frac{F}{w^{\frac{4}{3}}}$.

To make the Hamiltonian autonomous, we move to a rotating frame with angular velocity w, rescale time by $s = wt$, and apply the symplectic change of variables:

$$(x, y) = a(\bar{x}, \bar{y}), \quad (p_x, p_y) = aw(\bar{p}_x, \bar{p}_y), \quad \text{with } a^3 w^2 = 1. \tag{4}$$

In the new variables (dropping the bar for simplicity), the Hamiltonian transforms into:

$$H = \frac{1}{2}(p_x^2 + p_y^2) - (xp_y - yp_x) - \frac{1}{r} + Kx. \tag{5}$$

This simplification enables us to write the equations of motion in an autonomous form:

$$\begin{aligned} x' &= \frac{\partial H}{\partial p_x} = p_x + y, \\ y' &= \frac{\partial H}{\partial p_y} = p_y - x, \\ p_x' &= -\frac{\partial H}{\partial x} = p_y - \frac{x}{r^3} - K, \\ p_y' &= -\frac{\partial H}{\partial y} = -p_x - \frac{y}{r^3}. \end{aligned} \tag{6}$$

Remark 2.1. *These motion equations satisfy the symmetry*

$$(t, x, y, p_x, p_y) \to (-t, x, -y, -p_x, p_y), \tag{7}$$

which will be key to finding homoclinic connections with respect $y = 0$. This is easily proved if we consider the change of variables:

$$\tilde{t} = -t, \quad \tilde{x} = x, \quad \tilde{y} = -y, \quad \tilde{p}_x = -p_x, \quad \tilde{p}_y = p_y. \tag{8}$$

Under this transformation, the derivatives concerning time become:

$$\frac{d}{dt} = \frac{d}{d\tilde{t}} \cdot \frac{d\tilde{t}}{dt} = -\frac{d}{d\tilde{t}}. \tag{9}$$

Substituting into the equations of motion, we find:

$$\begin{aligned} \tilde{x}' &= -\frac{d\tilde{x}}{d\tilde{t}} = -(p_x + y) = -\tilde{p}_x - \tilde{y}, \\ \tilde{y}' &= -\frac{d\tilde{y}}{d\tilde{t}} = -(p_y - x) = -\tilde{p}_y + \tilde{x}, \\ \tilde{p}_x' &= -\frac{d\tilde{p}_x}{d\tilde{t}} = -\left(p_y - \frac{x}{r^3} - K\right) = -\tilde{p}_y + \frac{\tilde{x}}{r^3} + K, \\ \tilde{p}_y' &= -\frac{d\tilde{p}_y}{d\tilde{t}} = -\left(-p_x - \frac{y}{r^3}\right) = \tilde{p}_x + \frac{\tilde{y}}{r^3}. \end{aligned} \tag{10}$$

Rewriting these equations in terms of the new variables, we recover the original system:

$$\begin{aligned} x' &= \frac{\partial H}{\partial p_x} = p_x + y, \\ y' &= \frac{\partial H}{\partial p_y} = p_y - x, \\ p_x' &= -\frac{\partial H}{\partial x} = p_y - \frac{x}{r^3} - K, \\ p_y' &= -\frac{\partial H}{\partial y} = -p_x - \frac{y}{r^3}. \end{aligned} \tag{11}$$

Thus, the symmetry is satisfied.

2.1 Equilibrium points

Equilibrium points are obtained from the solutions of Eq.6. These are

$$\begin{aligned} p_x + y &= 0, \\ p_y - x &= 0, \\ p_y - \frac{x}{r^3} - K &= 0, \\ -p_x - \frac{y}{r^3} &= 0. \end{aligned} \tag{12}$$

We first get $p_x = -y$ and $p_y = x$ from the first two equations. Then, we can substitute and get

$$x - \frac{x}{r^3} - K = 0$$
$$y - \frac{y}{r^3} = 0$$

It is clear we have two possible cases for the second equation, $y = 0$ or $r = 1$. For the first one

$$x - \frac{x}{|x|^3} - K = 0$$

Multiplying by x^2 we get

$$x^3 - \frac{x^3}{|x|^3} - K = 0$$

This is the same as

$$x^3 - Kx^2 - sign(x) = 0$$

Regarding the second case, $r = 1$, we get $K = 0$, corresponding to the two-body problem. If we focus on $K > 0$ we shall consider the first case then.

To show that there are exactly two equilibrium points, we analyze the polynomial

$$f(x) = x^3 - Kx^2 - \text{sign}(x),$$

considering separately the cases $x > 0$ and $x < 0$. This can be done using Descartes' Rule of Signs and the Bolzano Theorem.
For $x > 0$, the polynomial takes the form

$$f(x) = x^3 - Kx^2 - 1. \tag{13}$$

By Descartes' Rule of Signs, the sequence of coefficients is $+1, -K, -1$, corresponding to the terms $x^3, -Kx^2$, and -1. There is exactly one sign change in this sequence, which implies the existence of exactly one positive root. To confirm this, we apply the Bolzano Theorem. Evaluating $f(x)$ at key points, we have

$$f(0) = -1 < 0, \quad \lim_{x\to\infty} f(x) = \infty > 0.$$

Since $f(x)$ is continuous and changes sign between $x = 0$ and $x \to \infty$, the Bolzano Theorem guarantees the existence of exactly one root for $x > 0$.
For $x < 0$, the polynomial becomes

$$f(x) = x^3 - Kx^2 + 1. \tag{14}$$

By Descartes' Rule of Signs, the sequence of coefficients is $-1, -K, +1$, corresponding to the terms $x^3, -Kx^2$, and $+1$. There is exactly one sign change in this sequence, which implies the existence of exactly one negative root. Again, applying the Bolzano Theorem, we evaluate $f(x)$ at key points:

$$\lim_{x\to-\infty} f(x) = -\infty < 0, \quad f(0) = +1 > 0.$$

Since $f(x)$ is continuous and changes sign between $x \to -\infty$ and $x = 0$, the Bolzano Theorem guarantees the existence of exactly one root for $x < 0$.
Combining these results, we conclude that $f(x)$ has exactly one positive root and exactly one negative root. Therefore, the system has exactly two equilibrium points. We have not considered $x = 0$ since $sign(x)$ is not defined for that point.

We can extract more information about these equilibrium points. We use the derivative $f'(x)$ and the behavior of $f(x)$ for small and large values of K. For Eq.13 we have

$$f'(x) = 3x^2 - 2Kx \tag{15}$$

Since $x > 0$, $f'(x) > 0$ for small and large x, so $f(x)$ is strictly increasing for $x > 0$. Evaluating the limits,

$$f(0) = -1 < 0, \quad f(2K/3) > 0,$$

implies the root lies in the interval $(1, 2K/3)$. As $K \to 0$, the root x_2 approaches 1, and it is bounded as

$$x_2 > \max\left(1, \frac{2K}{3}\right).$$

For Eq.14 we have

$$f'(x) = 3x^2 - 2Kx \tag{16}$$

Since $x < 0$, the term $-2Kx > 0$, so $f'(x) > 0$ and $f(x)$ is strictly increasing for $x < 0$. Evaluating the limits for small K,

$$f(0) = 1 > 0, \quad f(-1) = -K - 2 < 0,$$

implies the root lies in the interval $(-1, -1/\sqrt{K})$. As $K \to 0$, the root x_1 approaches -1, and it is bounded as

$$\max\left(-1, -\frac{1}{\sqrt{K}}\right) < x_1 < 0.$$

To confirm that x_1 and x_2 are increasing functions of K, we differentiate $f(x)$ implicitly with respect to K:

$$3x^2\frac{dx}{dK} - x^2 - 2Kx\frac{dx}{dK} = 0 \implies \frac{dx}{dK} = \frac{x^2}{3x^2 - 2Kx}.$$

For $x < 0$, both numerator and denominator are positive, so $\frac{dx}{dK} > 0$. For $x > 0$, the same holds, ensuring $\frac{dx}{dK} > 0$ in both cases. Thus, x_1 and x_2 are increasing functions of K.
In conclusion:

1. $\max\left(-1, -\frac{1}{\sqrt{K}}\right) < x_1 < 0$, and $x_2 > \max\left(1, \frac{2K}{3}\right)$;
2. As $K \to 0$, $x_1 \to -1$ and $x_2 \to 1$;
3. Both x_1 and x_2 increase with the value of K

K		L_1	L_2
0.0015749	x	-0.99947531	1.00052524
	h	-1.50157449	-1.49842469
	λ_i	$-0.06868279, 0.06868279$	$-0.06868279, 0.06868279$
0.1	x	-0.97611251	1.03446911
	h	-1.42120432	-1.39829568
	λ_i	$-0.53037020, 0.53037020$	$-0.53037020, 0.53037020$

Table 1: Equilibrium points, energies, and eigenvalues (λ_i) for two different values of K.

Table 1 shows some key properties from the two equilibrium points L_1 and L_2 for the two K values. As we stated before, as we decrease K, both equilibrium points tend to $x_i = |1|$.

2.2 Stability

The next step is to analyze the stability of the equilibrium points x_1 and x_2. The Jacobian matrix of our systems at each equilibrium point is given by:

$$Df(x, y, p_x, p_y) = \begin{pmatrix} 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ \frac{2x^2 - y^2}{r^5} & \frac{3xy}{r^5} & 0 & 1 \\ \frac{3xy}{r^5} & \frac{2y^2 - x^2}{r^5} & -1 & 0 \end{pmatrix},$$

where $r = \sqrt{x^2 + y^2}$ and from Eq.12 we know that $y = -p_x$ and $x = p_y$. Therefore, equilibrium points are $(x_1, 0, 0, x_1)$ and $(x_2, 0, 0, x_2)$. Substituting these coordinates into the Jacobian simplifies the matrix at each point to:

$$Df(x_i, 0, 0, x_i) = \begin{pmatrix} 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ \frac{2}{|x_i|^3} & 0 & 0 & 1 \\ 0 & -\frac{1}{|x_i|^3} & -1 & 0 \end{pmatrix}.$$

Definition 2.1. *An equilibrium point p is called **hyperbolic** if all the eigenvalues of $A := Df(p)$ have real parts different from 0.*

Definition 2.2. *An equilibrium point p is called **elliptic** if all the eigenvalues of $A := Df(p)$ have purely imaginary parts, and the associated Jordan matrix is diagonal.*

Theorem 2.1 (Stability of Equilibrium Points)**.** *Consider the continuous dynamical system*

$$\dot{x} = f(x),$$

with an equilibrium point p. Then:

1. *If all the eigenvalues of $A := Df(p)$ have real parts less than 0, then p is **asymptotically stable**.*
2. *If any eigenvalue of A has a real part greater than 0, then p is **unstable**.*

The eigenvalues of $Df(x_i, 0, 0, x_i)$ determine the stability of the equilibrium points. Computing the characteristic polynomial, we obtain:

$$\lambda^4 + \left(-\frac{1}{|x_i|^3} + 2\right)\lambda^2 + \frac{1}{|x_i|^3}\left(1 - \frac{2}{|x_i|^3}\right) = 0.$$

Equilibrium point x_1: For $x_1 < 0$, substituting $|x_1| = -x_1$, the characteristic polynomial simplifies. The eigenvalues are:

$$\pm\sqrt{\frac{1 + 2x_1^3 \pm \sqrt{8x_1^3 + 9}}{2|x_1|^3}}.$$

This yields two real eigenvalues and two purely imaginary eigenvalues, indicating that x_1 is of type *center* $\times$ *saddle*. Consequently, x_1 is unstable.
Equilibrium point x_2: For $x_2 > 0$, substituting $|x_2| = x_2$, the characteristic polynomial simplifies. The eigenvalues are:

$$\pm\sqrt{\frac{1 - 2x_2^3 \pm \sqrt{9 - 8x_2^3}}{2x_2^3}}.$$

The stability of x_2 depends on K:

- For $K \leq 3^{-4/3}/2 \approx 0.1156$, all eigenvalues have real parts equal to 0, making x_2 a *center* $\times$ *center*.
- For $K > 0.1156$, the discriminant becomes negative, and x_2 transitions to a *complex saddle*.

In summary, for any positive K value L_1 is unstable. On the other hand, L_2 transitions from stable to unstable as K increases.

3 Hill's Regions

To complete the characterization of the system, we now turn to the analysis of the Hill's regions. These regions provide valuable insight into the allowed configurations of the system by determining the spatial regions where motion is energetically permitted. This final step helps us frame the system more comprehensively and establish a clear understanding of its constraints and possible behaviors.
This is computed considering the Hamiltonian equation from our system Eq.5 with $H = h$ for a fixed value of energy. With some direct procedure, we get that

$$\frac{1}{2}(x'^2 + y'^2) = \frac{p_x^2}{2} + \frac{p_y^2}{2} + \frac{y^2}{2} + \frac{x^2}{2} + p_x y - p_y x \tag{17}$$

Therefore, using the Hamiltonian function

$$h + (\frac{y^2}{2} + \frac{x^2}{2} + \frac{1}{r} - Kx) \geq 0 \tag{18}$$

Then we just have to evaluate if given h and K the condition is satisfied or not

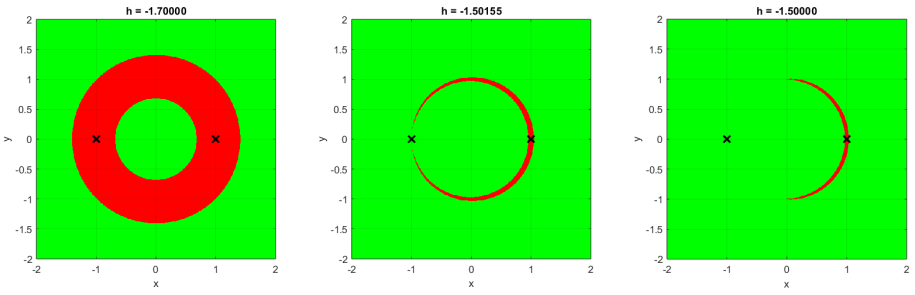

Figure 1: Hills regions in (x,y) coordinates of the CP problem for K = 0.0015749 and different energies: a) h = -1.7 (left), b) h = -1.50155 (center) and h = -1.5 (right). In green the allowed regions and red the forbidden. The equilibrium points are marked with a cross.

Figure 1 shows the allowed regions of motion for each point in a $[-2,2] \times [-2,2]$ windows. It is interesting to compare these h values studied with some reference, in our case the key values are the energy of the equilibrium points L_1 and L_2. These are $h_1 = -1.50157448675964$ and $h_2 = -1.49842469$, with $x_1 = -0.99947530882688$ and $x_2 = 1.00052524235312$. We see that for energies below h_1 there are two possible regions of motion, divided by a forbidden region. Nevertheless, for bigger h values there is only one component and the inner and outer regions are connected by a bottleneck region.

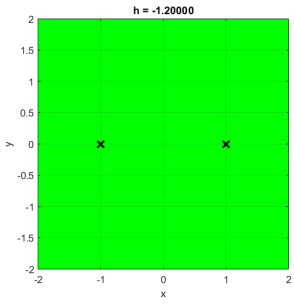

Figure 2: Hill's regions for $h = -1.2 > h_2$.

Figure 2 shows that for $h \geq h_2$ all the region is allowed. There is no kind of restriction.

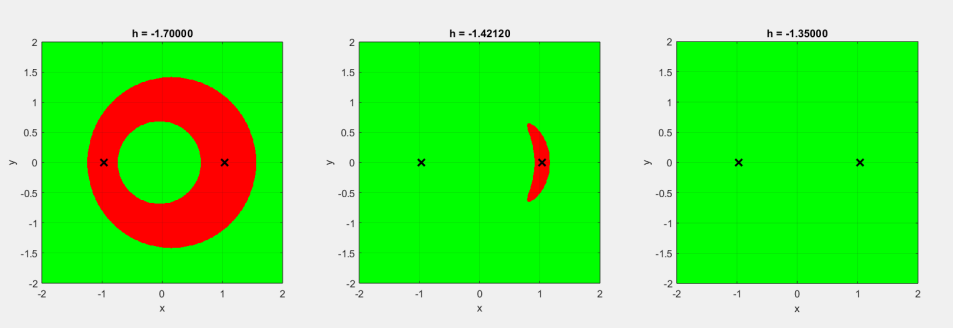

Figure 3: Hills regions for $K = 0.1$ and some h values bigger, lower and between L_1 and L_2 energies.

4 Dynamics around the equilibrium points

4.1 Introduction to homoclinic connections

In this section, we start with a simple study of the dynamics of the electron around the equilibrium points. As we will study later periodic orbits we must introduce some fundamental concepts to study the behavior of the orbits and the corresponding numerical approach.

Definition 4.1 (Local Stable Manifold)**.** *Let $\epsilon > 0$ be given. The* local stable manifold *of a point p is defined as:*

$$W^s_{loc}(\epsilon) = \{x \in \mathbb{R}^n \mid \|\phi(t,x) - p\| < \epsilon, \text{ for } t \geq 0\},$$

where $\phi(t,x)$ represents the flow of the dynamical system. This is the set of points in $\mathbb{R}^n$ that converge to p under forward time evolution, within a neighborhood of radius ϵ around p.

Definition 4.2 (Local Unstable Manifold)**.** *Let $\epsilon > 0$ be given. The* local unstable manifold *of a point p is defined as:*

$$W^u_{loc}(\epsilon) = \{x \in \mathbb{R}^n \mid \|\phi(t,x) - p\| < \epsilon, \text{ for } t \leq 0\},$$

where $\phi(t,x)$ represents the flow of the dynamical system. This is the set of points in $\mathbb{R}^n$ that approach p under backward time evolution, within a neighborhood of radius ϵ around p.

The local stable and unstable manifolds are crucial for understanding the dynamics near equilibrium points. They represent the sets of trajectories that asymptotically converge to or diverge from the equilibrium under the flow of the system.

Theorem: Local Stable/Unstable Manifolds for Flows
Let $x' = f(x)$, where p is an equilibrium point and f is a smooth function. Assume that the Jacobian matrix $A = Df(p)$ satisfies the following:

- A has d eigenvalues with strictly negative real parts.
- A has k eigenvalues with strictly positive real parts.
- E^s and E^u are the corresponding linear subspaces for the stable and unstable eigenspaces, respectively.

Then, for $\epsilon > 0$ small enough, the local stable and unstable manifolds, $W^s_{\text{loc}}(\epsilon)$ and $W^u_{\text{loc}}(\epsilon)$, are smooth manifolds of dimensions d and k, tangent to E^s and E^u, respectively, at p. Specifically:

$$W^s_{\text{loc}}(\epsilon) = \{x \in \mathbb{R}^n \mid \|\phi(t,x) - p\| < \epsilon,\ t \geq 0\},$$

$$W^u_{\text{loc}}(\epsilon) = \{x \in \mathbb{R}^n \mid \|\phi(t,x) - p\| < \epsilon,\ t \leq 0\}.$$

If $x \in W^s_{\text{loc}}(\epsilon)$, then $\phi(t,x) \to p$ as $t \to +\infty$.
If $x \in W^u_{\text{loc}}(\epsilon)$, then $\phi(t,x) \to p$ as $t \to -\infty$.

Definition: Homoclinic Orbit
Given p as an equilibrium point, a *homoclinic orbit* of p is a trajectory $\phi(t,x)$ such that:

$$\lim_{t \to \pm\infty} \phi(t,x) = p,$$

or equivalently:

$$\phi(t,x) \in W^s(p) \cap W^u(p).$$

Definition: Heteroclinic Orbit
Given two equilibrium points p and q, a *heteroclinic orbit* from p to q is a trajectory $\phi(t,x)$ such that:

$$\lim_{t \to +\infty} \phi(t,x) = p, \quad \lim_{t \to -\infty} \phi(t,x) = q,$$

or equivalently:

$$\phi(t,x) \in W^u(p) \cap W^s(q).$$

4.2 Manifolds of the CP problem

In this subsection, we investigate the local stable and unstable manifolds associated with the saddle-type equilibrium point L_1.

We start by computing the eigenvalues and eigenvectors for the equilibrium points of the system, given specific values of the parameters K and h. As an example, consider the case $K = 0.1$ and $h = -1.7$. For these values, the equilibrium points are computed as:

$$x_1 = -0.967753, \quad h_1 = -1.598370, \quad x_2 = 1.034469, \quad h_2 = -1.398296,$$

where x_1 and x_2 represent the coordinates of the equilibrium points, and h_1 and h_2 are their corresponding energies. Next, we compute the eigenvalues and eigenvectors associated with the equilibrium point x_1. The eigenvalues are:

$$\lambda_1 = -0.530370, \quad \lambda_2 = 0.530370.$$

The eigenvector associated with λ_1 corresponds to the stable direction (negative real part), and the eigenvector associated with λ_2 corresponds to the unstable direction (positive real part).
Since one eigenvalue is negative and the other is positive, L_1 is a saddle point. This classification implies that trajectories near L_1 will converge along the stable manifold (aligned with the stable eigenvector) and diverge along the unstable manifold (aligned with the unstable eigenvector).

To compute the local stable and unstable manifolds, we start by selecting initial points near the equilibrium. These initial points are defined as $x_{\text{stable},\pm} = x_1 \pm s \cdot v_{\text{stable}}$ and $x_{\text{unstable},\pm} = x_1 \pm s \cdot v_{\text{unstable}}$, where s is a small perturbation parameter. In this work, we use $s = 10^{-6}$ to ensure the initial points remain close to the equilibrium. This choice provides two branches for each manifold: one corresponding to a positive perturbation along the eigenvector and another to a negative perturbation.

Once explained the set up we integrated it numerically. For the unstable manifold, the integration is performed forward in time, following the direction of divergence from the equilibrium. For the stable manifold, the integration is

performed backward in time, following the direction of convergence toward the equilibrium. This approach allows us to trace the local dynamics accurately and obtain a complete representation of the stable and unstable manifolds of the saddle point L_1.

As shown in the code, the computation is performed using a Poincaré function. In this function, we integrate the orbit until a sign change is detected in the desired function, g. This function defines our Poincaré surface, which, in the following figure, is represented as $y = 0$. Once a sign change is identified, we apply the Newton-Raphson method to the function g to precisely determine the crossing point, iterating until a specified tolerance is achieved.

$$t^{m+1} = t^m - \frac{g(t^m)}{g'(t^m)} \tag{19}$$

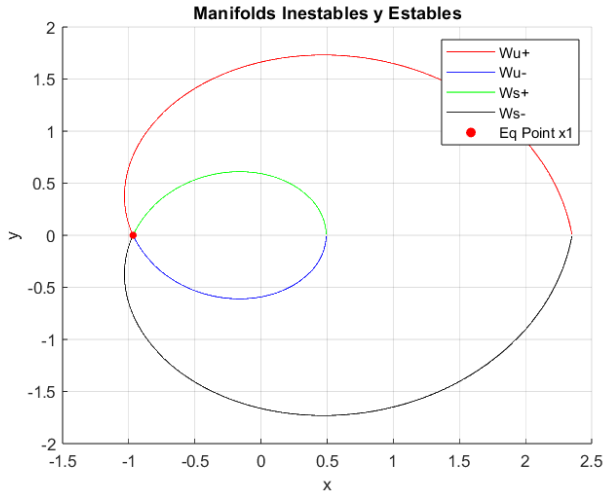

Figure 4: Branches of the manifolds around the equilibrium point L_1 for $K = 0.1$ and $h = -1.7$.

From Figure 4 we can call the intersection of W^u_+ and W^s_- outer connections, and the other two branches' inner connections.

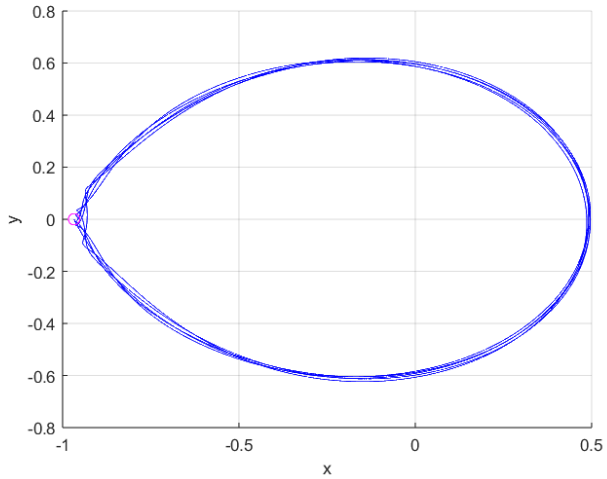

Figure 5: Orbit until 10 crossings with $y = 0$ of W^u_- and $K = 0.1$.

Figure 5 shows the behavior around x_1 if we compute until 10 crossings. It shows a bounded orbit, inside the corresponding inner region, and does not form a periodic orbit.

4.3 Homoclinic connections

Once the manifolds have been introduced, we can proceed to explore the concept of homoclinic connections. Due to the inherent symmetry of our system, a perpendicular intersection between W^u (unstable manifold) and W^s (stable manifold) implies the existence of a symmetric homoclinic orbit. Specifically, if we compute a branch of the manifold up to a Poincaré section and observe that $x' = 0$ at the intersection, a symmetric orbit concerning this Poincaré section exists. This represents a trajectory where the electron departs from the equilibrium point along the unstable

manifold and subsequently returns via the stable manifold, forming a closed trajectory in the phase space.

The presence of these symmetric homoclinic orbits plays a critical role in the structure of the phase space. As we previously demonstrated, the system exhibits symmetry given by

$$(t, x, y, p_x, p_y) \to (-t, x, -y, -p_x, p_y).$$

This symmetry allows us to infer that when W^u and W^s intersect perpendicularly, i.e., $x' = 0$ at $y = 0$, a symmetric homoclinic orbit exists.

Therefore, for multiple values of K, we compute the value of x' at the crossing point for one of the branches of the manifolds to investigate the conditions under which homoclinic orbits and periodic orbits emerge.

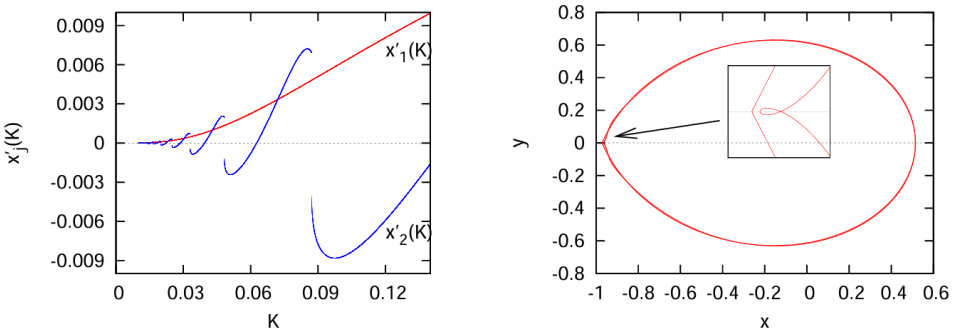

Figure 6: x'_j, with j being the number of crossings with $y = 0$, for multiple K values. Figure taken from [1]

Figure 6 shows the x' value of the j crossing for different K values and a Poincaré section $y = 0$. As we commented before, we are interested in the points such that $x'_j(K) = 0$. We see that this dynamical system will not have symmetric homoclinic connections around L_1, the first equilibrium point, for the first crossing. Nevertheless, there are infinite values of $x'_2(K) = 0$ which will correspond to 2-crossing symmetric homoclinic orbits. We also can observe how as we decrease the value of K the x' values are getting in general closer to the desired value of 0. This means that as we take lower values we find more orbits that lead to the symmetric homoclinic orbits.

For the first crossing, the way the code was programmed resulted in some oscillations around the $x' = 0$ value. I could manage to increase the precision but was not suitable to do the $x'(K)$ such a precision. Ideally, the expected result should be an increasing continuous function that does not cross the x-axis.

Remark 4.1. *The finite discontinuities from Figure 6 are from loops of the orbits that may intersect with the Poincaré section. This means that the crossing point is a completely different point and thus the derivative doesn't need to be similar. This is shown in Figure 7, using 3 crossings with the Poincaré section.*

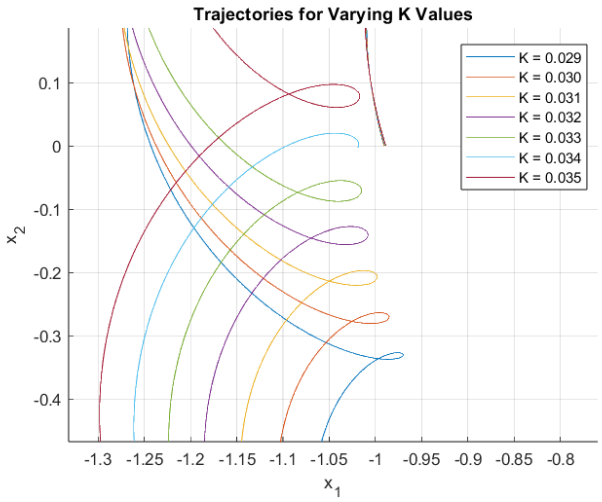

Figure 7: Loops from the unstable negative branch for multiple K values.

Remark 4.2. *Given the system's rapid evolution, it was necessary to use smaller time steps after detecting the crossing to ensure accuracy. However, such precision was not optimal at the beginning, particularly when considering all four branches. Additionally, we found that dynamically adapting the precision as K decreased significantly increased computational efficiency.*

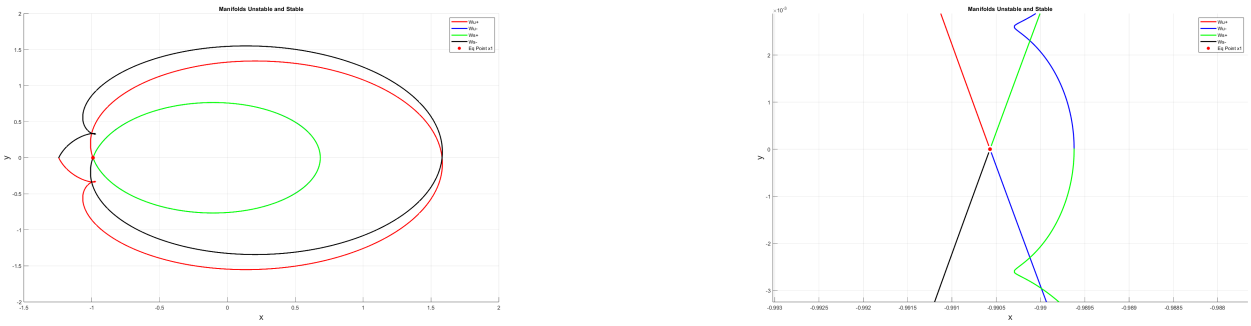

Figure 8: Negative branch of the unstable manifold computer for K = 0.02855986 and up to the second crossing $y = 0$.

Figure 8 shows this kind of asymmetric homoclinic connection between the manifolds for a specific value of K. In this particular case, we computed $x' = -3.7192 \cdot 10^{-6}$ for the crossing value of the unstable negative branch. This kind of study helps study transit and non-transit orbits. Given a fixed energy level, *transit orbits* are trajectories that cross the bottleneck region, determined by the zero-velocity curves. Conversely, *non-transit orbits* remain confined to their respective half-spaces, bouncing back after approaching the bottleneck.

The existence of transit and non-transit orbits is intricately related to the invariant manifolds and homoclinic connections. For small K and energies $h_1 < h < h_2$, specific orbits repeatedly transition between the inner and outer regions without escaping, despite theoretically approaching ionization. This behavior is governed by invariant tori, which act as barriers in phase space, preventing ionization under certain conditions.

4.4 Periodic Orbits

4.4.1 Background

The study of periodic orbits is fundamental for understanding the dynamics of the Circular Planar (CP) problem. These orbits represent trajectories where a system returns to its initial state after a fixed period. The classification of these orbits provides insights into their geometric and dynamic properties, such as stability and bifurcations.

In this section, we introduce key definitions essential for describing periodic orbits. These include the notions of direct and retrograde orbits, which categorize the motion based on the direction of the projection in the (x, y)-plane.

Definition 4.3. *A trajectory $\phi(t, x)$ is T-**periodic** if:*

1. $\phi(T, x) = x$,
2. $\phi(t, x) \neq x$ *for* $0 < t < T$.

Definition 4.4. *A set $A \subset \mathbb{R}^n$ is called **invariant** for a dynamical system if, for any initial condition $x_0 \in A$, the solution of the system passing through x_0 at time $t = 0$ remains in A for all t.*

Definition 4.5. *A periodic orbit (PO) is classified as **direct** if its projection onto the (x, y)-plane moves in a counterclockwise direction. Conversely, a PO is called **retrograde** if the projection moves in a clockwise direction.*

Definition 4.6. *If all the eigenvalues of $A := Df(p)$ have modulus different from 1, then p is called **hyperbolic**.*

Definition 4.7. *If all the eigenvalues of $A := Df(p)$ have modulus equal to 1 and the Jordan associated matrix is diagonal, then p is called **elliptic**.*

Definition 4.8. *Consider a periodic orbit $\phi(t, x)$ of period T for a continuous dynamical system. The **monodromy matrix** M is defined as the state transition matrix evaluated at $t = T$, denoted as:*

$$M = \Phi(T, x_0),$$

where $\Phi(t, x_0)$ satisfies the variational equation:

$$\frac{d}{dt}\Phi(t, x_0) = Df(\phi(t, x_0))\Phi(t, x_0), \quad \Phi(0, x_0) = I.$$

Definition 4.9. *The eigenvalues of the monodromy matrix M are called the **Floquet multipliers** of the periodic orbit. They determine the stability of the orbit.*

Definition 4.10. *A periodic orbit is said to be **linearly stable** if all the Floquet multipliers of its monodromy matrix M lie inside or on the unit circle in the complex plane, except for a simple multiplier at 1 corresponding to time invariance.*

Definition 4.11. *A periodic orbit is called **critical** if one of its Floquet multipliers is exactly equal to 2 or -2. These critical orbits are associated with bifurcations that generate new families of periodic orbits.*

Definition 4.12. *A **bifurcation** occurs in a dynamical system when a small change in a parameter results in a qualitative change in the system's behavior or the structure of its solutions, such as the creation or destruction of periodic orbits.*

Lemma 4.1. *For a periodic orbit of a Hamiltonian system, one of the Floquet multipliers is always equal to 1.*

Lemma 4.2. *Consider a periodic orbit of a continuous dynamical system. The stability of the periodic orbit is determined by the location of the Floquet multipliers in the complex plane:*

1. *If all Floquet multipliers lie inside the unit circle (except for the multiplier at 1), the periodic orbit is asymptotically stable.*
2. *If any Floquet multiplier lies outside the unit circle, the periodic orbit is unstable.*

Theorem 4.1 (Lyapunov's Theorem)**.** *Consider a dynamical system defined by the ODE:*

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \quad \mathbf{x} \in U \subset \mathbb{R}^n,$$

where $H(\mathbf{x})$ is a smooth, nondegenerate first integral of the system (i.e., $\nabla H(\mathbf{r}) \neq 0$ for all regular points $\mathbf{r}$). Assume that the system has an equilibrium point at $\mathbf{r}$ with characteristic exponents $\pm i\omega, \lambda_3, \ldots, \lambda_n$, where:

$$\frac{\lambda_j}{iw} \notin \mathbb{Z}, j = 3, ..., n$$

Then, there exists a one-parameter family of periodic orbits emanating from the equilibrium point. As the periodic orbits approach the equilibrium point, their period tends to $\frac{2\pi}{\omega}$.

4.4.2 Numerical Computation of PO

In this section, we detail the steps followed to find the periodic orbits around an equilibrium point. To understand the framework we are working with, we will introduce the Poincaré Section Plot. This is a (x,y) representation of the points satisfying $x' = 0$ and $y' < 0$. We can intuitively think about this as how the crossing points change depending on the initial conditions taken. Remark that the already introduced function g in our code is now $x' = 0$,

We approach this computation by trying to find initial conditions $(x, 0, 0, y')$ such that cross the Poincaré section with a $x' = 0$, given K and h. By the system's symmetry already proved, this means the existence of symmetric periodic orbits. We start with a point close to the equilibrium points and then compute its x'. Once detected two start points with opposite sign change we use a bisection method to get a good approximation of our point.

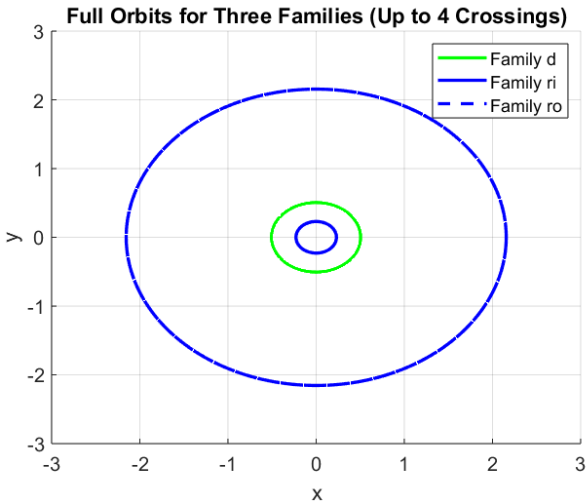

Figure 9: Orbits (x, y) from the initial conditions that lead to symmetric periodic orbits for $K = 0.0015749$ and $h = -1.7$.

The tolerances for the Poincaré computation were set to 10^{-3}, and while this value ensures reasonable accuracy, it introduces minor imprecision in x'. However, as shown in Figure 9, for $n_{\text{crossings}} = 4$, the results align well with the expected periodic orbits for the three families.

Figure 9 shows the trajectories starting from the initial conditions $(x, 0, 0, p_y)$ for the families d, ri, and ro. Family d is shown in green, while families ri and ro are represented in blue.
At this stage, it is essential to understand the dynamics of our system by studying the Poincaré section plot. Using the implemented code, we systematically consider multiple initial conditions to determine the exact points where the trajectory intersects the surface $x' = 0$ and $y < 0$. This condition corresponds to a symmetric homoclinic orbit.

As programmed, the computation time to obtain results similar to the expected ones was very high, and in some cases, the outcomes were slightly different due to large tolerances. Since the idea behind the computation has already been explained, and the functions have been detailed, we have used the reference figures from [1] for a better explanation.

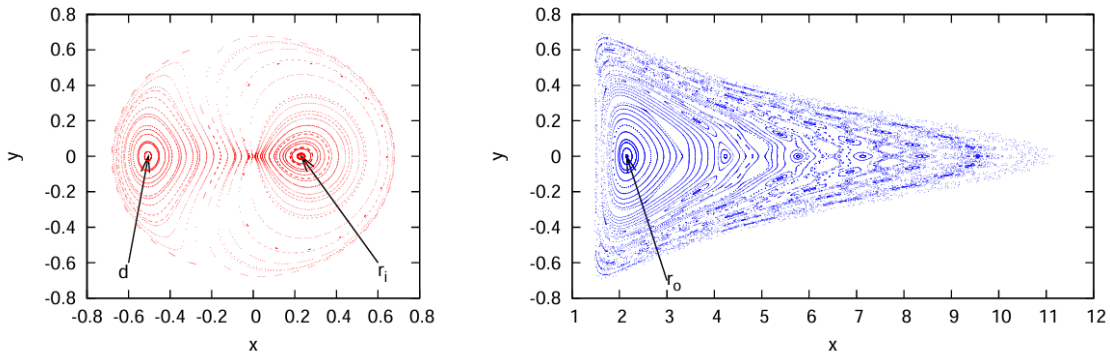

Figure 10: $K = 0.0015749$ and $h = -1.7$. Points (x, y) on the PSP for $x' = 0$ and $y' < 0$: (a) within the bounded component of the Hill region (left) and (b) within the unbounded component (right). In the inner region, the two fixed points correspond to stable periodic orbits from the main families d and ri. In the outer region, a stable periodic orbit from the family ro is displayed.

Figure 10 provides valuable insights into the dynamics of our system for $K \neq 0$ and $h = -1.7$ (see Figure 1). In this plot, three periodic points are observed, each corresponding to a periodic orbit. The left point represents a direct periodic orbit belonging to the family d, while the right point corresponds to a retrograde periodic orbit from the family r_i. Additionally, in the unbounded Hill region on the right side, we observe a third fixed point on the PSP, which corresponds to a retrograde periodic orbit from the family r_o. As it is seen, the fixed points on the PSP correspond to the PO plotted before.

Remark 4.3. *Intuitively, this is clear, but to formalize the reasoning: a fixed point in our Poincaré section plot corresponds to a periodic orbit because it implies that the intersection point with the surface $x' = 0$ remains constant and does not shift. This invariance ensures that the trajectory is closed and repeats itself, satisfying the condition for a periodic orbit in the phase space.*

It is also worth noting that the invariant curves around the fixed points observed in the PSP translate into quasi-periodic orbits in the full space. These results are compatible with the ones from Figure 9.

Family	(x, y)
d	$(-0.50709435, 0.00000000)$
ri	$(0.22969072, 0.00000000)$
ro	$(2.15529215, 0.00000000)$

Table 2: Initial conditions for the main families of symmetric periodic orbits shown in the PSP for $K = 0.0015749$ and $h = -1.7$.

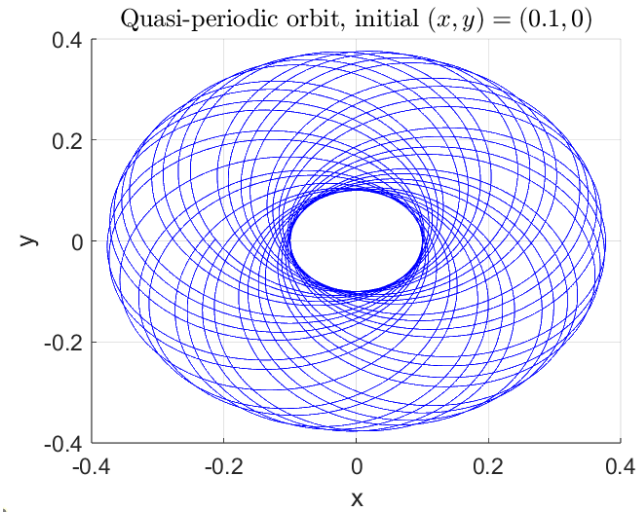

Figure 11: Orbit up to 100 crossing with PS $x' = 0$ for an initial condition from an invariant curve of the PSP.

Figure 11 shows how a particular point from the invariant bounded curve on Figure 10 generates a quasi-periodic orbit.

4.4.3 Periodic Orbits in the CP Problem

Once detailed the whole process we have a sustainable intuition to explain the results expected from the reference work.

Periodic Orbits (PO) for $K = 0$
From the rotating two-body problem with $K = 0$, it is well established that for $h < -\frac{3}{2}$, three families of circular periodic orbits (PO) exist:

- A family of direct periodic orbits, denoted as d.
- Two families of retrograde periodic orbits, denoted as ri and ro.

For a fixed energy level $h < -\frac{3}{2}$:

- The (x, y)-projection of the POs belonging to families d and ri lies within the bounded Hill region.
- The projection of the family ro lies within the unbounded region.

Periodic Orbits (PO) for $K > 0$ Small
When $K > 0$ is small, these families also persist, and we denote them in the same way:

- **Family d:** Exists for $h < h_1$, where the periodic orbits approach the origin as $h \to -\infty$. As $h \to h_1$, the period of the orbit tends to infinity.
- **Family ri:** Exists for all energy values, both positive and negative. For this family:
  - The period varies within $[0, 2\pi]$.
  - There are two limiting cases: an orbit of infinite radius and period 2π as $h \to +\infty$, and the origin as $h \to -\infty$.
- **Family ro:** Exists for $h < h_1$, where the stability parameter alternates between 2 and -2. Numerous bifurcating families emerge along this family.

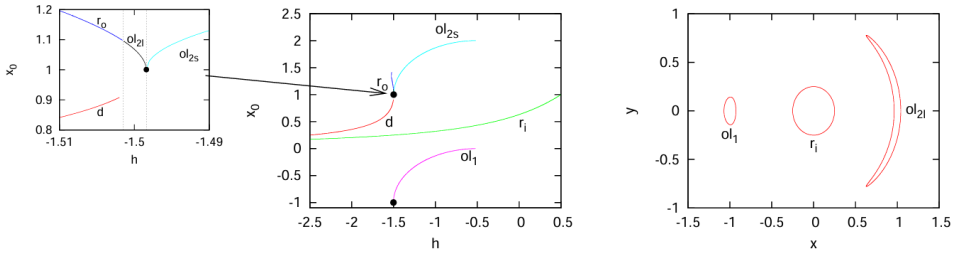

Figure 12: $K = 0.0015749$. (a) Left: Characteristic curves of the six principal families of periodic orbits represented in the (h, x_0) plane. The black dots indicate the positions of the equilibrium points L_1 and L_2. The vertical dashed lines correspond to the energy levels $h = h_1$ and $h = h_2$. A magnified view around the equilibrium point L_2 highlights the existence of the r_0 family for $h < h_1$ and the $ol2l$ family for $h_1 < h < h_2$, as detailed in the text. (b) Right: For $h = -1.499$, examples of periodic orbits are shown in the (x, y)-projection: one from the $ol1$ family (left), one from the r_i family (center), and one from the $ol2l$ family (right). Lyapunov orbit is explained in the following section. Figure from [1]

These families of periodic orbits are analogous to those found around L_4 and L_5 in the Restricted Three-Body Problem (RTBP). Their characteristic curves are typically represented in the (h, x_0) plane, where h is the energy and x_0 is the x-coordinate of the initial condition at the intersection with the x-axis. For example, in the case $K = 0.0015749$, the characteristic curves of these six families are shown in Figure 12. The initial condition is chosen as the intersection point with the largest x-coordinate.

4.5 Lyapunov Orbits

Once presented the main families of PO we are going to introduce the LPO, based on the *Lyapunov's Theorem* already presented.

Following our reference work we present these orbits for a fixed value $K = 0.0015749$. As we know, we must check if $\frac{\lambda_j}{iw} \notin \mathbb{Z}, j = 3, ..., n$ to state that there exists a LPO.

Equilibrium Point (Lyapunov Theorem)	**Energy**	**Eigenvalues**	**Type**
$x_1 = -0.99947531$ Lyapunov Theorem: Yes	$h_1 = -1.50157449$	$0.06868279 + 0.00000000i$ $-0.06868279 + 0.00000000i$ $0.00000000 + 1.00156957i$ $0.00000000 - 1.00156957i$	Saddle × Center
$x_2 = 1.00052524$ Lyapunov Theorem: Yes	$h_2 = -1.49842469$	$0.00000000 + 0.06879105i$ $0.00000000 - 0.06879105i$ $0.00000000 + 0.99841968i$ $0.00000000 - 0.99841968i$	Center × Center

Table 3: Analysis of Equilibrium Points and Lyapunov Theorem application for $K = 0.0015749$.

Table 3 has been computed to analyze the LPO around the two periodic points. It demonstrates the existence of a one-parameter family of LPO around x_1, referred to as the previously introduced ol_1, with a period asymptotically approaching $T = \frac{2\pi}{1.00156957}$.

In this section, we provide further insights into the origins of these LPO, complementing the discussion on periodic orbits (PO). Figure 12 shows the (x, y) projections of ol_1, ol_{2l}, and ol_{2s}, already introduced earlier. Below, we summarize and provide additional detailed information.

- **Family of LPO around L_1 (ol_1):** This family exists for $h \geq h_1$, with the computed orbits being unstable. For values of h slightly above and close to h_1, these orbits exhibit the hyperbolic characteristics of the L_1 equilibrium point, classified as a center-saddle type. The continuation of this family has been halted near a collision with the origin. However, it could be extended using regularized coordinates, leading to periodic orbits with loops at energy levels higher than those associated with the collision.
- **Family of LPO of long period around L_2 (ol_{2l}):** This family exists for $h_1 < h \leq h_2$. The computed orbits are linearly stable and feature numerous critical orbits. These orbits trace a trajectory in configuration space that encircles the zero-velocity curve (ZVC).

- **Family of LPO of short period around L_2 (ol_{2s}):** This family exists for $h \geq h_2$. The computed orbits are also linearly stable. Similar to the ol_1 family, the continuation of this family is stopped when a collision with the origin occurs.

For L_1 we have one eigenvalue that leads to the family ol_1. This family will tend to a period of $T = \frac{2\pi}{1.0015697} \approx 6.27$.

For L_2, two eigenvalues satisfy the *Lyapunov Theorem*, leading to the existence of two one-parameter families of *LPO* around L_2, denoted as ol_{2s} and ol_{2l}. These have respective periods of $T_s = \frac{2\pi}{0.06879105} = 91,33$ and $T_l = \frac{2\pi}{0.99841968} \approx 6.29$.

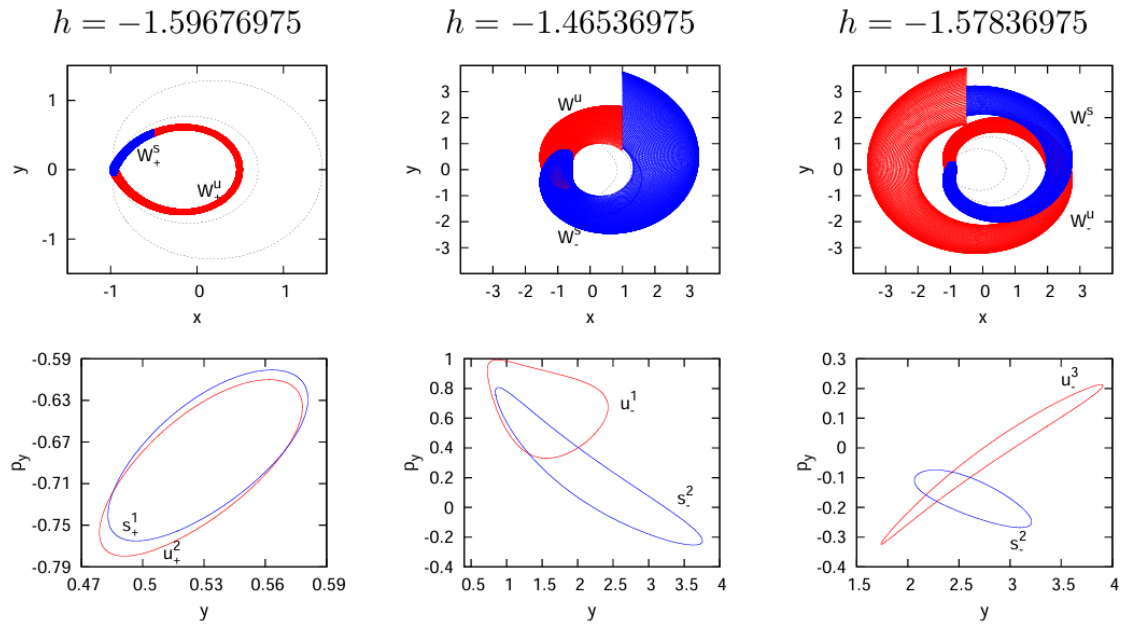

Figure 13: Examples of homoclinic connections in configuration space for $K = 0.1$. (a) An inner homoclinic connection ($h = -1.59676975$), and (b, c) two outer homoclinic connections ($h = -1.46536975$ and $h = -1.57836975$). The dotted lines represent the ZVC at each energy level. Figure from [1].

The results in Figure 13 highlight the dependence of homoclinic connections on the energy h and parameter K. For $K = 0.1$, the intersections of the stable (W^s) and unstable (W^u) manifolds in the (y, p_y)-plane demonstrate key differences:

- **Inner Homoclinics:** Corresponding to intersections of W^s_+ and W^u_+, these connections are more localized, indicating close interactions near the equilibrium points.
- **Outer Homoclinics:** Formed by W^s_- and W^u_-, these intersections extend further in the (y, p_y)-plane, reflecting trajectories that loop farther from the central region.
- **Energy Dependence:** For higher h, the manifold intersections broaden, while for lower h, they become more compact, reflecting the confinement of trajectories at reduced energy levels.

These results confirm the existence of families of inner and outer homoclinic connections for $h > h_1$, as expected

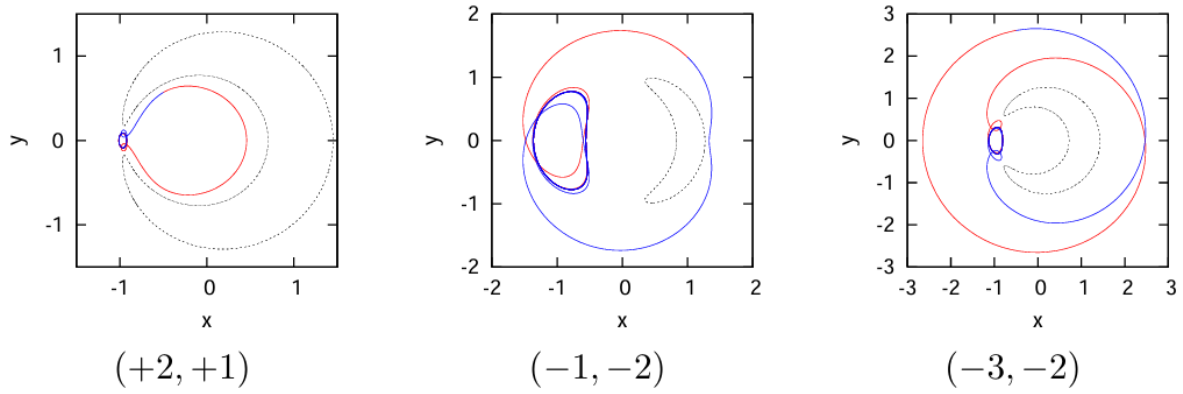

Figure 14: Examples of homoclinic connections in configuration space for $K = 0.1$. (a) An inner homoclinic connection ($h = -1.59676975$), and (b, c) two outer homoclinic connections ($h = -1.46536975$ and $h = -1.57836975$). The dotted lines represent the ZVC at each energy level. Figure from [1].

The bottleneck formed by the zero-velocity curve (ZVC) acts as a critical gateway, enabling transitions between the inner and outer regions of phase space, Figure 14. These transitions are facilitated by the interaction of stable and unstable manifolds, with their geometry determining the flow through the bottleneck. The structure of the ZVC strongly depends on the energy level.

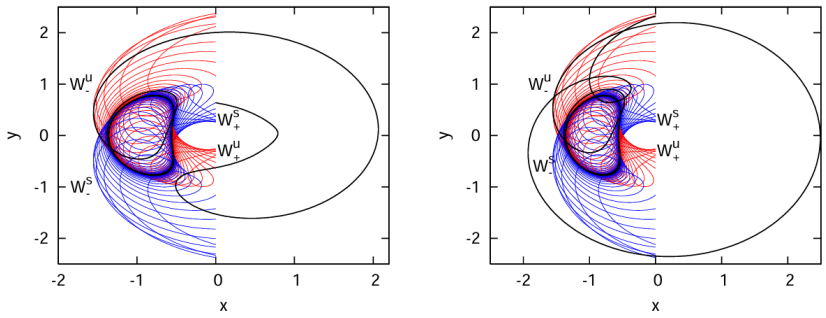

Figure 15: Visualization of transit and non-transit passages for orbits in phase space. (a) The transit orbit crosses into a neighborhood of the origin at its approach to the Lyapunov orbit, demonstrating an exchange between the inner and outer regions. (b) The non-transit orbit remains confined to the outer region, failing to cross the bottleneck formed by the zero-velocity curve. Both orbits (black curves) originate from the unstable branch W^u_- and intersect the stable branch W^s_-. Figure from [1].

Figure 15 illustrates the distinction between transit and non-transit orbits. Transit orbits, like the one shown in (a), cross the bottleneck region and transition between the inner and outer regions of the phase space. This behavior highlights the role of invariant manifolds in enabling such exchanges. In contrast, the non-transit orbit shown in (b) does not cross the bottleneck and remains confined to its initial region, reflecting the constraints imposed by the system's dynamics.

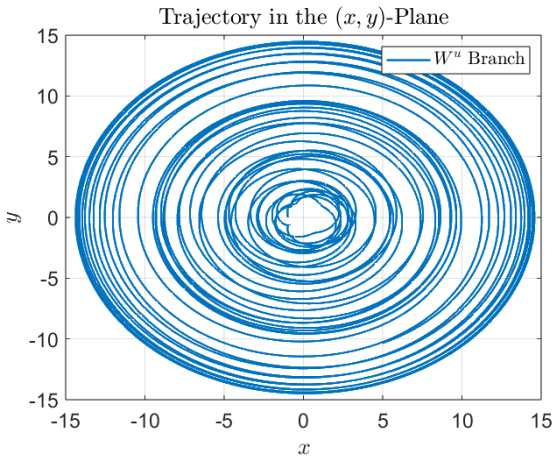

Figure 16: Unstable negative branch around L_1.

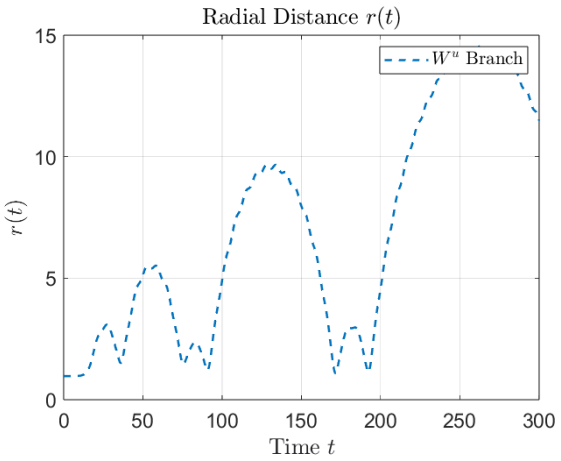

Figure 17: Radial distance for W^u_- branch.

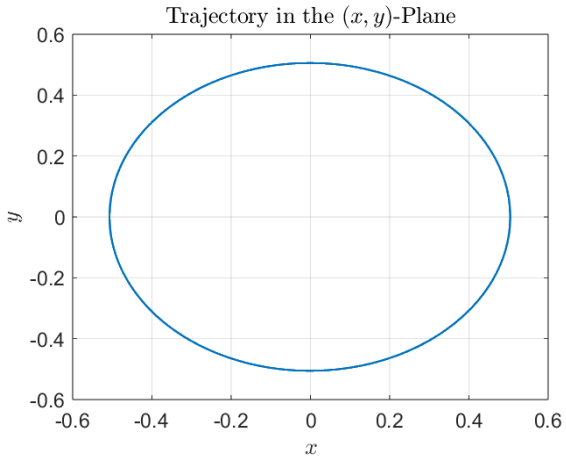

Figure 18: Periodic r_d orbit.

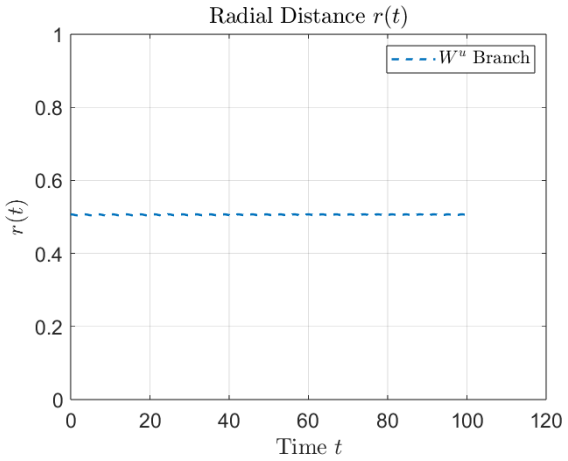

Figure 19: Radial distance for r_d orbit.

Figure 20: Visualization of unstable branches and periodic orbits, along with their corresponding radial distances for W^u and r_d.

To conclude this section, we present several orbits to introduce a key variable: the radial distance to the nucleus. Figure 20 shows how, for periodic orbits (PO), the radial distance appears constant. It is important to emphasize that this does not necessarily have to be the case. By the definition of periodicity, the orbit returns to its starting point, and therefore, the radial distance will return to the same value after a period T. Additionally, it is shown that the unstable manifold does not diverge monotonically. Instead, it oscillates progressively, moving away and returning, but it gradually escapes further. This behavior will be crucial in the next section.

5 Ionization

In this section we are going to use already-introduced concepts to focus on "slow ionization". This concept means that the electron goes to infinity after some complex behavior. We briefly first detail the framework of our system from a global point of view.

5.1 Global Dynamics

5.1.1 $K = 0$ Dynamics

When $K = 0$, the system corresponds to the rotating two-body problem, which is integrable. The dynamics consist of simple, well-understood types of orbits: rotating ellipses, parabolas, and hyperbolas. In the Poincaré Surface of Section (PSP) for $K = 0$, the bounded orbits are represented by invariant curves corresponding to rotating ellipses. These appear around the origin for orbits in the bounded component of Hill's region and farther out for orbits in the unbounded component.

The transition between bounded and unbounded orbits is marked by parabolic trajectories, which act as the boundary. The PSP, shown in Figure 21, clearly illustrates these dynamics: periodic or quasiperiodic orbits form closed or

invariant curves, while unbounded sets correspond to parabolic and hyperbolic trajectories. For instance, 22 shows how taking as initial condition a point from the blue unbounded region, this is a point from an invariant curve, we get a quas-periodic orbit.

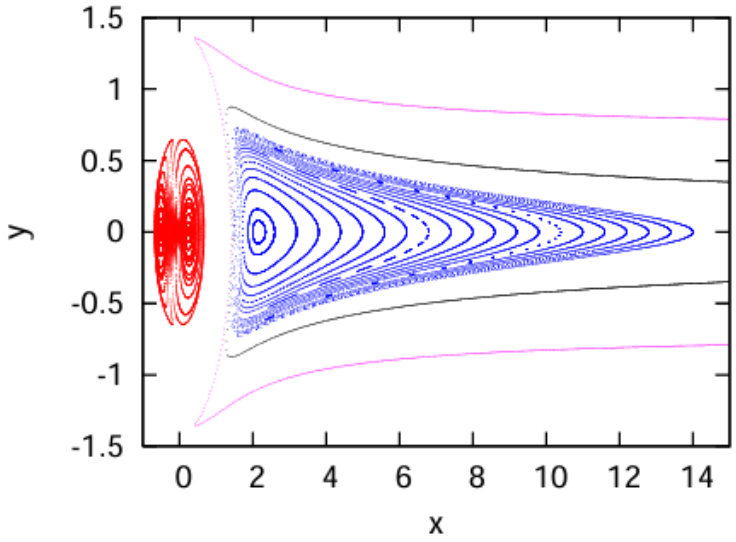

Figure 21: PSP for $K = 0$. The invariant curves correspond to rotating ellipses in the bounded and unbounded components of Hill's region. Parabolic orbits separate bounded (elliptic) from unbounded (hyperbolic) dynamics. Figure from [1].

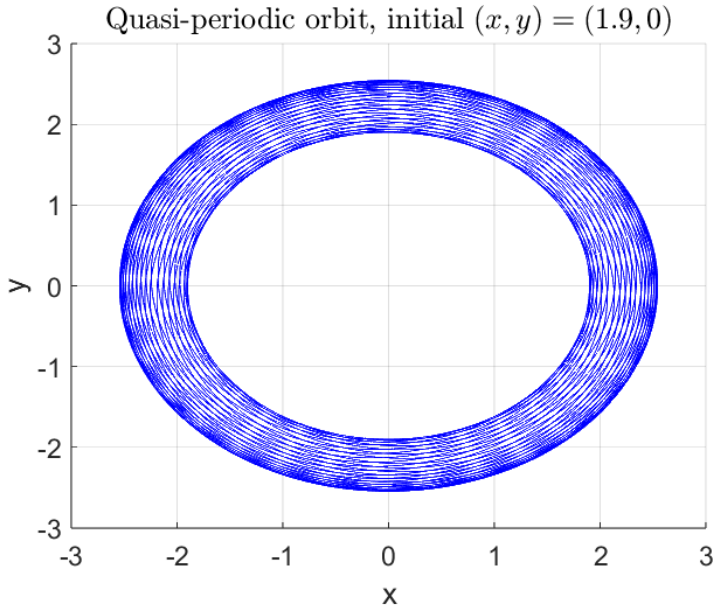

Figure 22: Orbit (x,y) taken initial condition from the inverted bell from the unbounded region in the PSP.

5.1.2 $K > 0$ Dynamics

When $K \neq 0$, the dynamics of the system become more complex due to the non-integrability of the Hamiltonian. As the parameter K increases from zero, several invariant structures emerge, significantly altering the behavior of the system. The motion is no longer confined to regular periodic orbits, and the system exhibits both regular and chaotic behavior. The specific dynamics depend on the energy levels considered and the value of K.

For small values of K, the system retains some regular features, such as invariant curves and periodic orbits. However, as K increases, the stability of these structures is compromised. According to the KAM theorem, many invariant curves are destroyed, and the system exhibits the phenomenon of overlapping resonances, leading to the interlacing of regular and chaotic regions, see [4]. The regular motion remains confined within certain invariant curves, while the chaotic regions exhibit erratic behavior.

At higher values of K, particularly for $K > 0$, the system transitions to a regime where the dynamics are dominated by a mix of stable periodic orbits (such as the LPOs around equilibrium points like L_1) and chaotic trajectories. These erratic orbits, which appear to ionize, can have long escape times, with the distance from the origin oscillating in a non-monotonic fashion.

As shown in [1], and with the help of the PSP for different energy values, interesting dynamics depend on this energy level. For instance, let us take $K = 0.0015749$. Our energy levels are the ones shown in Table 1.

- For energy levels $h < h_1$, the configuration space is divided into bounded and unbounded regions by the Hill's region boundaries. At $h = -1.7$, the bounded region resembles the rotating two-body problem, exhibiting stable periodic orbits surrounded by invariant curves. In the unbounded region, the dynamics become more irregular as $K > 0$, with many invariant curves destroyed, leading to chaotic layers, periodic orbits, and regions of stochasticity. Farther from the origin, invariant curves break, forming chains of islands and hyperbolic orbits. Erratic trajectories transition between islands and may slowly escape, showing structured stochasticity influenced by hyperbolic orbits and heteroclinic connections.

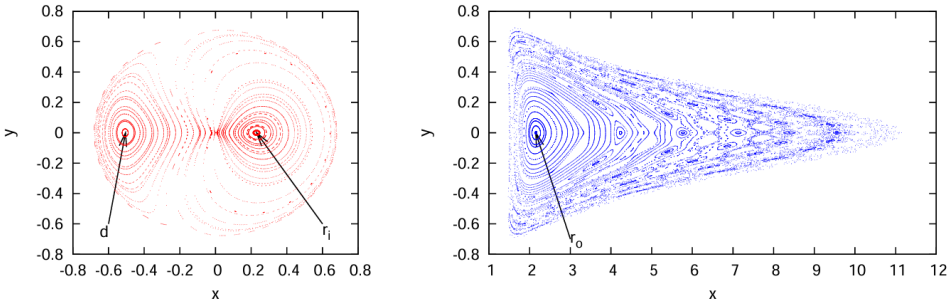

Figure 23: PSP points (x, y) for $K = 0.0015749$ and $h = -1.7$, evaluated at $x' = 0$ and $y' < 0$. Panel (a) depicts the bounded region of the Hill's surface (left), while panel (b) illustrates the unbounded region (right). In the inner bounded region, two fixed points correspond to stable periodic orbits of the families d and ri. In the outer unbounded region, a stable periodic orbit associated with the family ro is observed. Figure from [1].

- For $h \in (h_1, h_2)$, with $h = -1.5$, the zero-velocity curve (ZVC) forms a bounded, right-moon-shaped curve intersecting the x-axis at two points, except for $h = h_1$, where it intersects only at L_1. In this energy range, the forbidden region of motion separates the iterates of the Poincaré map in the inner and outer regions. A neck connects these regions, allowing trajectories to travel between them, eliminating motion barriers around the origin.

  For $h = h_1$, the equilibrium point L_1 emerges, and the orbits near it collapse into the unstable and stable manifolds of L_1. For $h > h_1$, the periodic orbit around L_1 and it's nearly coinciding stable and unstable manifolds persist, along with the internal retrograde orbit and invariant curves around it. In addition, periodic orbits of the main families $ol1$, ri, and $ol2l$ are observed.

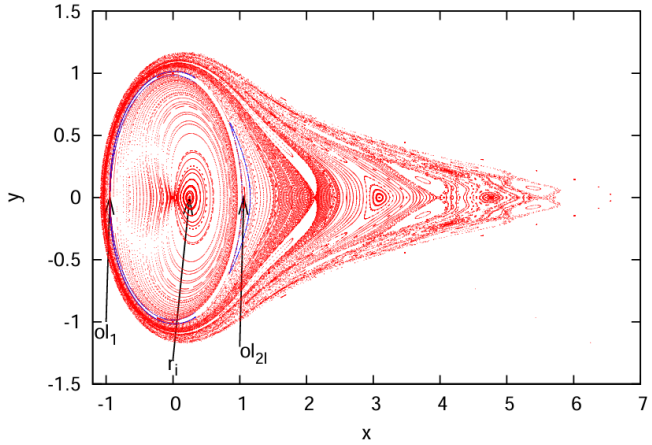

Figure 24: PSP plot for $K = 0.0015749$ and $h = -1.5$, showing the orbits of the main families. The one-dimensional curves, representing the intersection of the two-dimensional invariant manifolds of the LPO around L_1 with the section, are displayed in blue. These invariant manifolds are confined within the outermost invariant curve surrounding the origin. Additionally, the forbidden moon-shaped region of motion is visible. Figure from [1]

- For $h > h_2$, the dynamics exhibit no zero-velocity curve (zvc). The system includes a stable retrograde orbit of the family ri, a stable short-period LPO around L_2, and a hyperbolic LPO around L_1 with its stable and unstable invariant manifolds, W^s and W^u. At these energy levels, a last invariant curve surrounds the origin, confining the invariant manifolds of the family $ol1$. Outside this curve, chaotic behavior becomes prevalent, with erratic and ionizing orbits appearing.

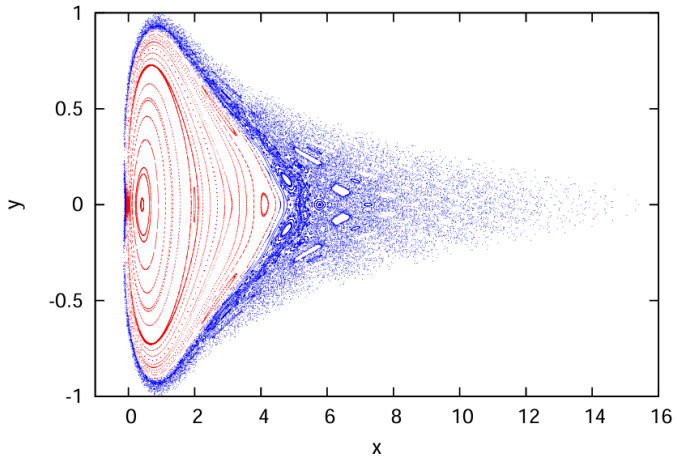

Figure 25: PSP points (x, y) for $K = 0.0015749$ and $h = -0.557$, evaluated at $x' = 0$ and $y' < 0$. Blue and red points represent orbits outside and inside the outermost invariant curve around the origin, respectively, although this invariant set has not been explicitly computed. Figure from [1].

These results correspond to a small K. For larger values of K, the dynamics remain similar but exhibit some nuances. For example, the manifolds of certain LPO are no longer confined.

By experimenting with the code, we can observe these differences directly. The following image illustrates the radial distances for the nucleus, comparing cases with $h > h_2$ for both a small and a large K, following the unstable negative branch of the manifold around L_1.

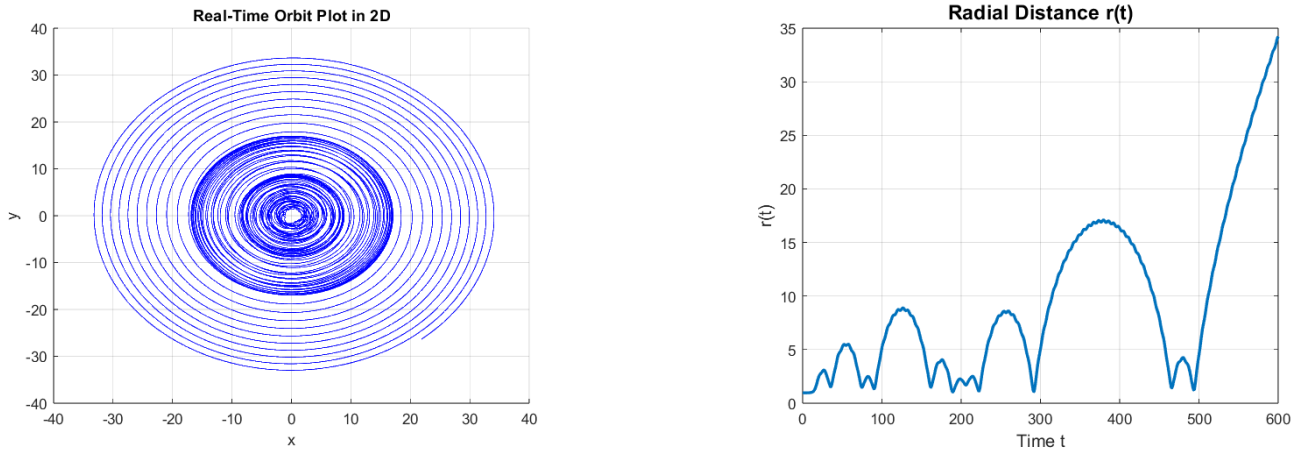

Figure 26: W^u_- orbit and radial distance around L_1 for a high value $K = 0.1$ and a fixed high value $h = -1$.

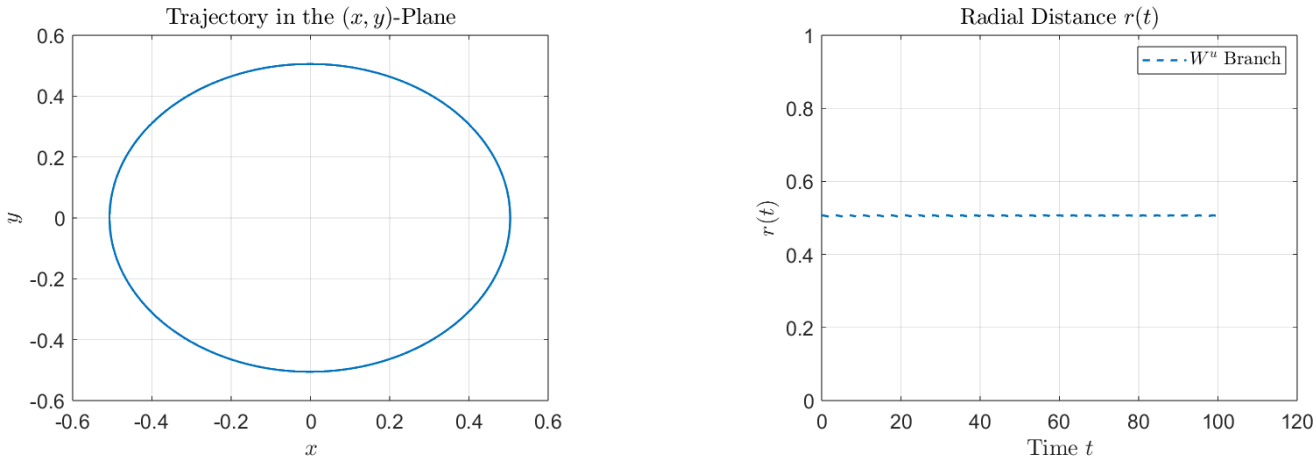

Figure 27: W^u_- orbit and radial distance around L_1 for a high value $K = 0.0015749$ and a fixed high value $h = -1$.

The previous figures (26 and 27) illustrate how, for the same energy level, larger values of K enable the manifolds to facilitate the ionization of the electron. As shown in the $x'(K)$ plot, it becomes more likely for the manifolds to intersect in such a way that make a homoclinic connection. This prevents escape to the outer region, even when the energy level would allow it.

6 Comments on Escape Rates

In this section, we analyze the phase space dynamics to provide quantitative predictions for escape rates. The system exhibits different types of trajectories, including fast and slow ionizing orbits, each characterized by their specific behaviors and the role of invariant manifolds.

6.1 Fast Ionization (FI)

Fast ionizing orbits are characterized by trajectories that spiral outward without revisiting the vicinity of the nucleus. In rotating coordinates, these orbits are nearly radial, while in inertial coordinates, they resemble hyperbolas. This type of escape occurs rapidly, with minimal oscillations in the radial distance $r(t)$ over time.

6.2 Slow Ionization (SI)

In contrast, slow ionizing orbits display erratic behavior, involving successive approaches to and receding from the nucleus before eventual escape. These orbits alternate between maxima and minima of the radial distance $r(t)$, with no clear monotonic trend. The dynamics are dominated by the interaction of invariant manifolds and the surrounding chaotic region, which facilitates complex, non-linear trajectories.

6.3 Ionization pathways

To identify whether an orbit exhibits hyperbolic (fast escape) behavior, the osculating sidereal energy $E_s(t)$ is used as a criterion. This energy, derived from the two-body problem approximation, is defined as

$$E_s = \frac{1}{2}(X'^2 + Y'^2) - \frac{1}{\sqrt{X^2 + Y^2}},$$

where X, Y are positions and X', Y' are velocities in the non-rotating frame. Typically oscillatory over time, an orbit is considered hyperbolic if $E_s(t) > \delta > 0$ for a small threshold δ. Orbits satisfying this condition exhibit fast ionization (FI) behavior, characterized by continuous growth in distance from the origin.

To classify orbits that do not exhibit immediate escape, the parameters T and D are introduced. Here, T represents a large observation time, and D denotes a critical distance from the origin beyond which escape is considered to occur. Commonly used values in numerical explorations are $T = 5 \times 10^4$ or $T = 10^5$ and $D = 100$, though they can be adjusted depending on the specific system under study. Using these parameters, Effective Bounded Erratic (EBE) orbits are defined as trajectories that, within the interval $t \leq T$, remain confined within the distance D and satisfy $E_s(t) \leq 0$. While periodic orbits always meet these criteria, other trajectories may transition from EBE to FI behavior as T or D increases.

The erratic region R in the (x_0, θ) plane offers a useful framework for identifying initial conditions associated with EBE or FI behavior. For small K, the region R contains initial conditions for which $E_s < 0$, corresponding to EBE orbits, while points outside R are often associated with FI behavior. However, R does not act as an absolute barrier; rather, it marks a transition zone where the dynamics shift between erratic and hyperbolic escape. As x_0 increases, R narrows and eventually disappears for sufficiently large x_0 (approximately $x_0 \sim 2/K^2$ for small K), at which point all initial conditions correspond to FI orbits.

6.4 Osculating sidereal energy

For fixed energy h, the initial conditions are considered in the region $(x_0, \theta) \in [x_m, \infty) \times [0, 2\pi)$, where $\theta = 0$ and $\theta = 2\pi$ are identified, and x_m is a positive value sufficiently far from the origin. This region avoids the initial conditions near the nucleus and focuses on the right-hand side of the zero-velocity curve, defined by $x_0 > x_c$, where x_c is the farthest intersection of the zero-velocity curve with $y = 0$ in the configuration space.
The dynamics of the CP problem in this region are approximated using the two-body problem, where the osculating sidereal energy is given by:

$$E_s^0 = \frac{v^2}{2} + \frac{x_0^2}{2} + x_0 v \sin\theta - \frac{1}{x_0},$$

with v determined from the problem's constraints. Initial conditions where $E_s^0 < 0$ define the **erratic region R**, which serves as a set of candidates for erratic or escaping orbits. The equation:

$$\sin\theta = \frac{-h - x_0^2 + K x_0}{v x_0}$$

describes the boundary of R within the (x_0, θ) plane.
The properties of R are as follows:

- For $h < -\frac{1}{2K}$, R is empty because $f(x_0) = \frac{-h - x_0^2 + K x_0}{v x_0}$ satisfies $f(x_0) < -1$.
- For $h > -\frac{1}{2K}$, the function $f(x_0)^2 = 1$ has two solutions:

  $$\tau_{1,2} = \frac{1 + Kh \pm \sqrt{1 + 2Kh}}{K^2},$$

  and R is bounded within $x_0 \in (\tau_1, \tau_2)$. As x_0 increases, R takes on a "spear-like" shape symmetric around $\theta = 3\pi/2$, as $f(x_0)$ approaches -1 rapidly.

For small K, the extent of R can grow significantly, up to a value proportional to $2/K^2$. This allows for the existence of erratic orbits with initial conditions far from the nucleus, despite having negative osculating energy. However, the boundary of R does not act as a strict separator between erratic and periodic behavior but instead marks a transition region between these dynamics.

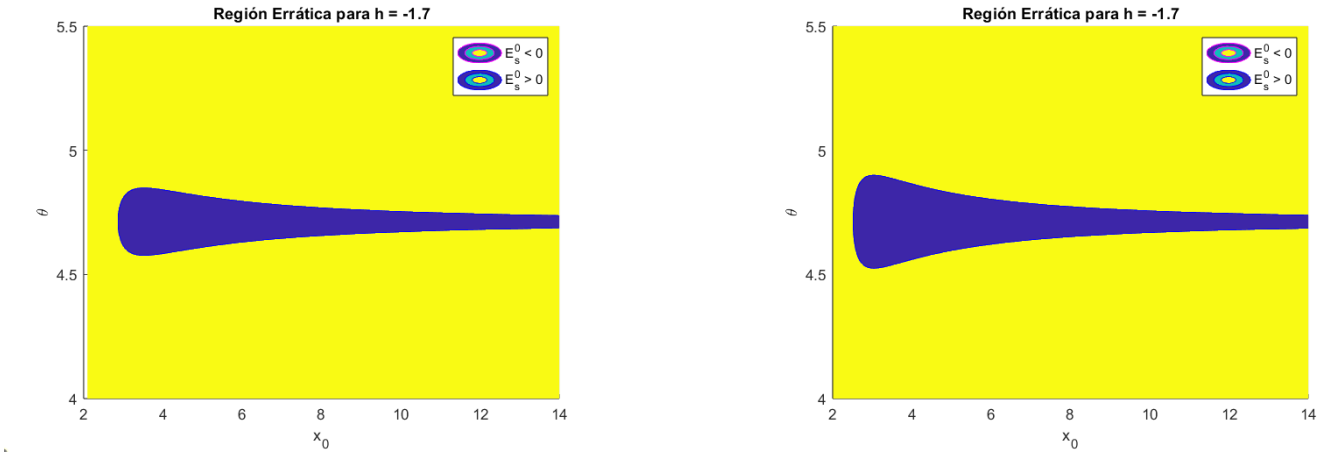

Figure 28: Regions depending on E_s value for $h = -1.7$ and a) $K = 0.0015749$ b) $K = 0.1$.

As Figure 28 shows, as K increases, the region R, defined by $E_s^0 < 0$, expands and includes trajectories farther from the nucleus. Larger K values disrupt invariant curves, leading to chaotic layers and increasing the likelihood of erratic behavior or orbital escape. This effect reflects the weakening confinement of R as K grows.

7 Conclusions

In this work, we have studied the hydrogen atom under circularly polarized microwave fields. We have progressively developed strategies to understand the dynamics of the system, which led us to analyze the possibility of ionization and classify different types of orbits.

Starting from the simplest case of $K = 0$, we observed how the unbounded regions contain points that would lead to ionizing orbits for $K \neq 0$. Additionally, as K increases, the manifolds play a fundamental role in the ionization process. We have studied how even with very high values of h the introduced value of K makes sometimes impossible to ionize the electron. Nevertheless, the contrary happened when K is big. In this case, even the manifold can move the electron from a very close position close to the nucleus to an ionization path.

In the final sections, focusing on ionization, we highlighted how the unconfined region is rich in potentially ionizing orbits. Within this framework, we identified specific initial conditions that, based on their osculating sidereal energy, emerge as natural candidates for either ionizing or EBE (Extended Bounded Escape) orbits.

In this work, we aimed to follow a reference study with a clear objective. Our goal was twofold: first, to expand on certain foundational concepts, providing a deeper explanation of their origin; and second, to numerically replicate the key results presented in the reference paper.

Regarding the computational implementation, several challenges arose. Initially, we sought an alternative to the Hill's regions by proposing the pseudo-arc method. However, for certain values of h, the function G did not converge properly, resulting in only a partially computed region. Additionally, we were unable to achieve precise convergence results for specific cases, such as obtaining the $x'(K)$ curve. Furthermore, the computational time required to generate the Poincaré Sectio Plot was not optimal given my own code implementation. Consequently, as the goal was not to replicate every plot from the reference study exactly, some graphs were directly taken from the reference paper.

CODE

```
format long;
close all;
clear all;
clc

%% HILLS REGIONS
% Parameters
% K = 0.1; % Parameter for the system
% h_values = [-1.7, -1.50155, -1.5]; % Energy levels to analyze

K = 0.0015749;
h_values = [-1.7, -1.421203, -1.35];
tol = 1e-13; % Convergence tolerance
max_iter = 1000; % Maximum iterations for Newton's method
delta_s = 1e-2; % Step size for pseudo-arcs
n = 2000; % Number of points per orbit
delta_y = 1e-4; % Increment for detecting sign changes in G
number = 50000;

% Compute equilibrium points
[x1, x2, h1, h2] = eq_and_energies(K);


x_range = linspace(-2, 2, 500);
y_range = linspace(-2, 2, 500);

tol = 1e-13; % Convergence tolerance
max_iter = 1000; % Maximum iterations for Newton's method
delta_s = 1e-2; % Step size for pseudo-arcs
n = 2000; % Number of points per orbit
delta_y = 1e-4; % Increment for detecting sign changes in G
number = 50000;



% Configuraci n del plot
figure;
for i = 1:length(h_values)
    h = h_values(i); % Energ a actual

    % Inicializar la matriz de colores
    ColorMatrix = zeros(length(y_range), length(x_range));

    % Calcular el Hamiltoniano punto a punto
    for ix = 1:length(x_range)
        for iy = 1:length(y_range)
            energy = h + hamiltonian(x_range(ix), y_range(iy), K);
            if energy < 0
                ColorMatrix(iy, ix) = 1; % Color para valores permitidos
            else
                ColorMatrix(iy, ix) = -1; % Color para valores prohibidos
            end
        end
    end

    % Graficar las regiones
    subplot(1, 3, i);

    imagesc(x_range, y_range, ColorMatrix, [-1, 1]); % Limitar el rango de datos
    set(gca, 'YDir', 'normal'); % Invertir el eje Y para que sea correcto
    colormap([0 1 0; 1 0 0]); % Verde para permitido, rojo para prohibido
    title(sprintf('h = %.5f', h));
    xlabel('x');
    ylabel('y');
    axis equal;
    axis([-2 2 -2 2]); % Ajustar l mites de los ejes
```

```
    grid on;
    hold on;
    plot(x1, 0, 'kx', 'MarkerSize', 10, 'LineWidth', 2);
    plot(x2, 0, 'kx', 'MarkerSize', 10, 'LineWidth', 2);
    hold off;
end

h = -1.2;

% Inicializar la matriz de colores
ColorMatrix = zeros(length(y_range), length(x_range));

% Calcular el Hamiltoniano punto a punto
for ix = 1:length(x_range)
    for iy = 1:length(y_range)
        energy = h + hamiltonian(x_range(ix), y_range(iy), K);
        if energy < 0
            ColorMatrix(iy, ix) = 1; % Color para valores permitidos
        else
            ColorMatrix(iy, ix) = -1; % Color para valores prohibidos
        end
    end
end

% Crear la figura
figure;

% Graficar las regiones
imagesc(x_range, y_range, ColorMatrix, [-1, 1]); % Limitar el rango de datos
set(gca, 'YDir', 'normal'); % Invertir el eje Y para que sea correcto
colormap([0 1 0; 1 0 0]); % Verde para permitido, rojo para prohibido
title(sprintf('h = %.5f', h));
xlabel('x');
ylabel('y');
axis equal;
axis([-2 2 -2 2]); % Ajustar l mites de los ejes
grid on;

% A adir los puntos de equilibrio como crucetas negras
hold on;
plot(x1, 0, 'kx', 'MarkerSize', 10, 'LineWidth', 2); % Puntos en (x1, 0)
plot(x2, 0, 'kx', 'MarkerSize', 10, 'LineWidth', 2); % Puntos en (x2, 0)
hold off;

%% EQ POINTS, EIGENVALUES AND EIGENVECTORS
% Define the parameter K
% K = 0.1; % Example value, you can modify it as needed
K =0.0015749;
% K = 0.028559865;

% Compute equilibrium points and energies
[x1, x2, h1, h2] = eq_and_energies(K);

% Display equilibrium points and their corresponding energies
disp('Equilibrium point x1:');
disp(x1);
disp('Energy at x1 (h1):');
disp(h1);

disp('Equilibrium point x2:');
disp(x2);
disp('Energy at x2 (h2):');
disp(h2);

% Calculate eigenvalues and eigenvectors for x1
[eigenvalues, eigenvectors] = equilibriumEigen(x1);
[eigenvalues2, eigenvectors2] = equilibriumEigen(x2);
% Display the results for x1
disp('Eigenvalues for x1:');
```

```
disp(eigenvalues);

disp('Eigenvectors for x1 (columns correspond to eigenvalues):');
disp(eigenvectors);

disp('Eigenvalues for x2:');
disp(eigenvalues2);

disp('Eigenvectors for x2 (columns correspond to eigenvalues):');
disp(eigenvectors2);

% Display the stable eigenvalue and eigenvector
disp('Stable eigenvalue (negative real part):');
disp(eigenvalues(1)); % First element corresponds to stable eigenvalue
disp('Stable eigenvector:');
disp(eigenvectors(:, 1)); % First column corresponds to stable eigenvector

% Display the unstable eigenvalue and eigenvector
disp('Unstable eigenvalue (positive real part):');
disp(eigenvalues(2)); % Second element corresponds to unstable eigenvalue
disp('Unstable eigenvector:');
disp(eigenvectors(:, 2)); % Second column corresponds to unstable eigenvector




v_unstable = eigenvectors(:, 2);
v_stable = eigenvectors(:, 1);
% Ensure eigenvectors have positive y-coordinate
if v_stable(2) > 0
    v_stable = -v_stable; % Flip stable eigenvector orientation
end

if v_unstable(2) > 0
    v_unstable = -v_unstable; % Flip unstable eigenvector orientation
end

%% MANIFOLDS
% Par metros iniciales
s = 10e-4; % Escala para mover a lo largo del eigenvector inestable
i1 = [x1, 0, 0, x1] + s * v_unstable'; % Primer punto hacia el eigenvector inestable
i2 = [x1, 0, 0, x1] - s * v_unstable'; % Segundo punto en direcci n opuesta
i3 = [x1, 0, 0, x1] +s*v_stable';
i4 = [x1, 0, 0, x1]-s*v_stable';
disp('Point along +eigenvector unstable:');
disp(i1);
disp('Point along -eigenvector unstable:');
disp(i2);

% Plotear los puntos
figure;
hold on;
scatter(i1(1), i1(2), 50, 'red', 'filled', 'DisplayName', 'Point +Eigenvector');
scatter(i2(1), i2(2), 50, 'yellow', 'filled', 'DisplayName', 'Point -Eigenvector');

% Etiquetas y formato
xlabel('x_1');
ylabel('x_2');
title('Initial Points Along the Unstable Eigenvector');
legend('show');
grid on;
hold off;


tol = 10e-2;
n_crossing = 2;
idir = 1;
[times1, sols1, orbit1] = poincare(@g, tol, K, i1, n_crossing, idir);
[times2, sols2, orbit2] = poincare(@g, tol, K, i2, n_crossing, idir);
```

```
[times3, sols3, orbit3] = poincare(@g, tol, K, i3, n_crossing, -idir);
[times4, sols4, orbit4] = poincare(@g, tol, K, i4, n_crossing, -idir);
% Plotear las  rbitas

figure;
hold on;
plot(orbit1(:,1), orbit1(:,2), 'r-', 'DisplayName', 'Unstable branch (+)');
plot(orbit2(:,1), orbit2(:,2), 'magenta', 'DisplayName', 'Unstable branch (-)');
plot(orbit3(:,1), orbit3(:,2), 'b--', 'DisplayName', 'Stable branch (+)');
plot(orbit4(:,1), orbit4(:,2), 'g--', 'DisplayName', 'Stable branch (-)');

% Etiquetas y formato
xlabel('x_1');
ylabel('x_2');
title('Trajectories Along Stable and Unstable Eigenvectors');
legend('show');
grid on;
hold off;


disp(sols2);
%% LOOPS
% Par metros iniciales
K_start = 0.028559865; % Valor inicial de K
K_step = 0.001;        % Incremento de K
K_end = 0.035;         % Valor final de K
tol = 10e-2;           % Tolerancia
n_crossing = 3;        % N mero de cruces
idir = 1;              % Direcci n del flujo
s = 10e-4;             % Escala para mover a lo largo del eigenvector inestable

% Inicializar figura
figure;
hold on;

% Iterar sobre valores de K
for K = K_start:K_step:K_end
    % Calcular puntos de equilibrio y energ as
    [x1, x2, h1, h2] = eq_and_energies(K);

    % Calcular eigenvalores y eigenvectores para x1
    [eigenvalues, eigenvectors] = equilibriumEigen(x1);
    v_unstable = eigenvectors(:, 2); % Eigenvector inestable
    v_stable = eigenvectors(:, 1);   % Eigenvector estable

    % Asegurarse de que los eigenvectores tengan la orientaci n correcta
    if v_stable(2) > 0
        v_stable = -v_stable; % Cambiar orientaci n del estable
    end

    if v_unstable(2) > 0
        v_unstable = -v_unstable; % Cambiar orientaci n del inestable
    end

    % Punto inicial en la rama inestable (-)
    i2 = [x1, 0, 0, x1] - s * v_unstable';

    % Calcular  rbita  con Poincar
    [~, ~, orbit2] = poincare(@g, tol, K, i2, n_crossing, idir);

    % Plotear la  rbita
    plot(orbit2(:, 1), orbit2(:, 2), 'DisplayName', sprintf('K = %.3f', K));
end

% Etiquetas y formato
xlabel('x_1');
ylabel('x_2');
title('Trajectories for Varying K Values');
legend('show', 'Location', 'best');
```

```
grid on;
hold off;

%% PERIODIC ORBITS
xd = -0.5070094351999249; % Example x value
K = 0.0015749; % Given K
H = -1.7; % Given H
xri = 0.229690715908056;
xro = 2.155292149997760;

pyd = dy(xd, K, H);
pyri = dy(xri, K, H);
pyro = dy(xro, K, H);

disp(['p_y(1): ', num2str(pyd(1))]);
disp(['p_y(2): ', num2str(pyd(2))]);
disp(['p_y(1): ', num2str(pyri(1))]);
disp(['p_y(2): ', num2str(pyri(2))]);
disp(['p_y(1): ', num2str(pyro(1))]);
disp(['p_y(2): ', num2str(pyro(2))]);


%%
% Compute Poincare maps and orbits for each family
[PoincareMapTimes_d, sols_d, full_orbit_d] = poincare(@g, tol, K, [xd, 0, 0, pyd(2)],
    n_crossing, idir);
[PoincareMapTimes_ri, sols_ri, full_orbit_ri] = poincare(@g, tol, K, [xri, 0, 0,
    pyri(2)], n_crossing, idir);
[PoincareMapTimes_ro, sols_ro, full_orbit_ro] = poincare(@g, tol, K, [xro, 0, 0,
    pyro(2)], n_crossing, idir);
%%
% Plot full orbits for all families
figure;
hold on;
plot(full_orbit_d(:, 1), full_orbit_d(:, 2), 'g-', 'LineWidth', 2, 'DisplayName', 'Family
    d');
plot(full_orbit_ri(:, 1), full_orbit_ri(:, 2), 'b-', 'LineWidth', 2, 'DisplayName',
    'Family ri');
plot(full_orbit_ro(:, 1), full_orbit_ro(:, 2), 'b--', 'LineWidth', 2, 'DisplayName',
    'Family ro');
hold off;

% Add plot details
title('Full Orbits for Three Families (Up to 4 Crossings)', 'FontSize', 14);
xlabel('x', 'FontSize', 12);
ylabel('y', 'FontSize', 12);
grid on;
legend show;
set(gca, 'FontSize', 12); % Make axis ticks larger
%% QUASIPERIODIC POINT
K = 0.0015749;
h = -1.7;
x = 0.1;
py = dy(x, K, H);
py_i = py(2);
tol = 10e-2;
[PoincareMapTimes_d, sols, full_orbit] = poincare(@g, tol, K, [x, 0, 0, py_i], 80, idir);
%%
figure;
hold on;
sols_filtered = sols(sols(:, 4) < 0, :);

plot(full_orbit(:, 1), full_orbit(:, 2), 'blue-', 'LineWidth', 0.5, 'DisplayName',
    'Quasiperiodic orbit');
% scatter(sols_filtered(:, 1), sols_filtered(:, 2), 50, 'magenta', 'filled',
    'DisplayName', 'Filtered crossings');
xlabel('x', 'FontSize', 16);
ylabel('y', 'FontSize', 16);
```

```
title('Quasi-periodic orbit, initial $(x,y) = (0.1,0)$', 'FontSize', 14, 'Interpreter', 'latex');
grid on;

set(gca, 'FontSize', 14); % Make axis ticks larger
hold off;

%%
[t, sol] = ode45(@(t, state) f(t, state, K, h), tspan, init_cond);

%% LPO
h = -1.499;
K = 0.0015749;
% Compute equilibrium points and energies
[x1, x2, h1, h2] = eq_and_energies(K);

% Display equilibrium points and their corresponding energies
disp('Equilibrium point x1:');
disp(x1);
disp('Energy at x1 (h1):');
disp(h1);

disp('Equilibrium point x2:');
disp(x2);
disp('Energy at x2 (h2):');
disp(h2);

% Calculate eigenvalues and eigenvectors for x1
[eigenvalues, eigenvectors] = equilibriumEigen_imaginary(x1);
[eigenvalues2, eigenvectors2] = equilibriumEigen_imaginary(x2);

% Display the results for x1
disp('Eigenvalues for x1:');
disp(eigenvalues);

disp('Eigenvectors for x1 (columns correspond to eigenvalues):');
disp(eigenvectors);

disp('Eigenvalues for x2:');
disp(eigenvalues2);

disp('Eigenvectors for x2 (columns correspond to eigenvalues):');
disp(eigenvectors2);
%%
eps = 10^-3;
xinitial1 = x1+eps;
prime1 = dy(xinitial1, K, h);
ysign = +1;
tol = 10^-4;
max_counter = 10000;
der1 = prime1(2);
c = bisection_method(xinitial1, xinitial1, K, h, ysign, tol, max_counter);
%%
disp(c);
pyc = dy(c, K, h);
pyc = pyc(2);
[PoincareMapTimes_d, sols, full_orbit] = poincare(@g, tol, K, [x, 0, 0, pyc], 4, idir);
%% DISTANCES
K = 0.1;
h = -1.46536975;
[x1, x2, h1, h2] = eq_and_energies(K);
[eigenvalues, eigenvectors] = equilibriumEigen(x1);
[eigenvalues2, eigenvectors2] = equilibriumEigen(x2);
tol = 10e-2;
n_crossing = 2;
idir = 1;
s = 10e-4;
i2 = [x1, 0, 0, x1] - s * v_unstable';
[PoincareMapTimes_d, sols, full_orbit] = poincare_times(@g, tol, K, i2, 20, idir);
```

```
%%
[PoincareMapTimes, full_orbit] = poincare_times(@g, tol, K, i2, 300, idir);

% Plot r(t)
times = full_orbit(:, 1); % Extract time column
x_vals = full_orbit(:, 2); % Extract x column
y_vals = full_orbit(:, 3); % Extract y column
r_t = sqrt(x_vals.^2 + y_vals.^2);

figure;
plot(times, r_t, 'LineWidth', 2);
title('Radial Distance r(t)', 'FontSize', 14);
xlabel('Time t', 'FontSize', 12);
ylabel('r(t)', 'FontSize', 12);
grid on;

%%
% Compute r(t) from full_orbit
times = full_orbit(:, 1); % Extract time column
x_vals = full_orbit(:, 2); % Extract x column
y_vals = full_orbit(:, 3); % Extract y column

% Compute radial distance r(t)
r_t = sqrt(x_vals.^2 + y_vals.^2);
%%
% Plot r(t) with dashed lines
figure;
plot(times, r_t, '--', 'LineWidth', 1.5, 'DisplayName', '$W^{u}$ Branch');
title('Radial Distance $r(t)$', 'FontSize', 16, 'Interpreter', 'latex');
xlabel('Time $t$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$r(t)$', 'FontSize', 16, 'Interpreter', 'latex');
grid on;
legend('show', 'Interpreter', 'latex');% Adjust tick sizes
ax = gca; % Get current axes
ax.FontSize = 14; % Set font size for tick labels

% Plot the calculated orbit in the (x, y)-plane
figure;
plot(x_vals, y_vals, '-', 'LineWidth', 1.5, 'DisplayName', '$W^{u}$ Branch');
title('Trajectory in the $(x, y)$-Plane', 'FontSize', 14, 'Interpreter', 'latex');
xlabel('$x$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$y$', 'FontSize', 16, 'Interpreter', 'latex');
grid on;
legend('show', 'Interpreter', 'latex');
% Adjust tick sizes
ax = gca; % Get current axes
ax.FontSize = 14; % Set font size for tick labels
%%
K = 0.0015749;
per_radius = [xd, 0, 0, pyd(2)];
[PoincareMapTimes, full_orbit] = poincare_times(@g, tol, K, per_radius, 100, idir);


% Compute r(t) from full_orbit
times = full_orbit(:, 1); % Extract time column
x_vals = full_orbit(:, 2); % Extract x column
y_vals = full_orbit(:, 3); % Extract y column

% Compute radial distance r(t)
r_t = sqrt(x_vals.^2 + y_vals.^2);
%%
% Plot r(t) with dashed lines
figure;
plot(times, r_t, '--', 'LineWidth', 1.5, 'DisplayName', '$W^{u}$ Branch');
title('Radial Distance $r(t)$', 'FontSize', 16, 'Interpreter', 'latex');
xlabel('Time $t$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$r(t)$', 'FontSize', 16, 'Interpreter', 'latex');
grid on;
ylim([0,1]);
```

```
legend('show', 'Interpreter', 'latex');% Adjust tick sizes
ax = gca; % Get current axes
ax.FontSize = 14; % Set font size for tick labels

% Plot the calculated orbit in the (x, y)-plane
figure;
plot(x_vals, y_vals, '-', 'LineWidth', 1.5, 'DisplayName', '$W^{u}$ Branch');
title('Trajectory in the $(x, y)$-Plane', 'FontSize', 14, 'Interpreter', 'latex');
xlabel('$x$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$y$', 'FontSize', 16, 'Interpreter', 'latex');
grid on;
% Adjust tick sizes
ax = gca; % Get current axes
ax.FontSize = 14; % Set font size for tick labels

%%
% Initial condition
% Initial condition
init_cond = i2; % Replace with specific values
tspan = [0, 300]; % Extend the integration time range
K = 0.1; % Example parameter, adjust as needed
h = -1.7; % Energy level

% Options for ode45 with smaller time steps
tol = 1e-9; % Tighter tolerances for higher precision
options = odeset('RelTol', tol, 'AbsTol', tol, 'MaxStep', 1e-3); % Maximum step size for
    finer time resolution

% Integrate using ode45
[t, sol] = ode45(@(t, state) f(t, state, K, h), tspan, init_cond, options);

% Compute r(t) = sqrt(x(t)^2 + y(t)^2)
r_t = sqrt(sol(:, 1).^2 + sol(:, 2).^2);

% Plot r(t)
figure;
plot(t, r_t, 'LineWidth', 2);
title('Radial Distance r(t) with Fine Time Steps', 'FontSize', 14);
xlabel('Time t', 'FontSize', 12);
ylabel('r(t)', 'FontSize', 12);
grid on;
%% TWO BODY PROBLEM
K = 0;

h_values = [-1.7, -1.421203, -1.35];
tol = 1e-13; % Convergence tolerance
max_iter = 1000; % Maximum iterations for Newton's method
delta_s = 1e-2; % Step size for pseudo-arcs
n = 2000; % Number of points per orbit
delta_y = 1e-4; % Increment for detecting sign changes in G
number = 50000;

% Compute equilibrium points
[x1, x2, h1, h2] = eq_and_energies(K);
disp(h1);
disp(h2);

x_range = linspace(-2, 2, 500);
y_range = linspace(-2, 2, 500);

tol = 1e-13; % Convergence tolerance
max_iter = 1000; % Maximum iterations for Newton's method
delta_s = 1e-2; % Step size for pseudo-arcs
n = 2000; % Number of points per orbit
delta_y = 1e-4; % Increment for detecting sign changes in G
number = 50000;
```

```
% Configuraci n del plot
figure;
for i = 1:length(h_values)
    h = h_values(i); % Energ a actual

    % Inicializar la matriz de colores
    ColorMatrix = zeros(length(y_range), length(x_range));

    % Calcular el Hamiltoniano punto a punto
    for ix = 1:length(x_range)
        for iy = 1:length(y_range)
            energy = h + hamiltonian(x_range(ix), y_range(iy), K);
            if energy < 0
                ColorMatrix(iy, ix) = 1; % Color para valores permitidos
            else
                ColorMatrix(iy, ix) = -1; % Color para valores prohibidos
            end
        end
    end

    % Graficar las regiones
    subplot(1, 3, i);

    imagesc(x_range, y_range, ColorMatrix, [-1, 1]); % Limitar el rango de datos
    set(gca, 'YDir', 'normal'); % Invertir el eje Y para que sea correcto
    colormap([0 1 0; 1 0 0]); % Verde para permitido, rojo para prohibido
    title(sprintf('h = %.5f', h));
    xlabel('x');
    ylabel('y');
    axis equal;
    axis([-2 2 -2 2]); % Ajustar l mites de los ejes
    grid on;
    hold on;
    plot(x1, 0, 'kx', 'MarkerSize', 10, 'LineWidth', 2);
    plot(x2, 0, 'kx', 'MarkerSize', 10, 'LineWidth', 2);
    hold off;
end

%%
K = 0;
idir = +1;
x = 1.9;
py = dy(x, K, H);
py_i = py(2);
tol = 10e-2;
[PoincareMapTimes_d,full_orbit] = poincare(@g, tol, K, [x, 0, 0, py_i], 80, idir);


%%
figure;
hold on;
sols_filtered = sols(sols(:, 4) < 0, :);

plot(full_orbit(:, 1), full_orbit(:, 2), 'blue-', 'LineWidth', 0.5, 'DisplayName',
    'Quasiperiodic orbit');
% scatter(sols_filtered(:, 1), sols_filtered(:, 2), 50, 'magenta', 'filled',
    'DisplayName', 'Filtered crossings');
xlabel('x', 'FontSize', 16);
ylabel('y', 'FontSize', 16);
title('Quasi-periodic orbit, initial $(x,y) = (1.9,0)$', 'FontSize', 14, 'Interpreter',
    'latex');
grid on;

set(gca, 'FontSize', 14); % Make axis ticks larger
hold off;

%% RADIAL DISTANCE DIF K
% K = 0.1;
K = 0.0015749;
```

```
% h = -1.46536975;
h = -1.4;
[x1, x2, h1, h2] = eq_and_energies(K);
disp(h1);
disp(h2);
[eigenvalues, eigenvectors] = equilibriumEigen(x1);
[eigenvalues2, eigenvectors2] = equilibriumEigen(x2);
tol = 10e-2;
n_crossing = 2;
idir = 1;
s = 10e-4;
% i2 = [x1, 0, 0, x1] - s * v_unstable';
x0 = 3.8;
py = dy(x0, K, h);
i2 = [x0, 0, 0, -py(2)];


[PoincareMapTimes, full_orbit] = poincare_times(@g, tol, K, i2, 1000, idir);
%%
% Plot r(t)
times = full_orbit(:, 1); % Extract time column
x_vals = full_orbit(:, 2); % Extract x column
y_vals = full_orbit(:, 3); % Extract y column
r_t = sqrt(x_vals.^2 + y_vals.^2);

figure;
plot(times, r_t, 'LineWidth', 2);
title('Radial Distance r(t)', 'FontSize', 14);
xlabel('Time t', 'FontSize', 12);
ylabel('r(t)', 'FontSize', 12);
grid on;

%%
% Compute r(t) from full_orbit
times = full_orbit(:, 1); % Extract time column
x_vals = full_orbit(:, 2); % Extract x column
y_vals = full_orbit(:, 3); % Extract y column

% Compute radial distance r(t)
r_t = sqrt(x_vals.^2 + y_vals.^2);
%%
% Plot r(t) with dashed lines
figure;
plot(times, r_t, '--', 'LineWidth', 1.5, 'DisplayName', '$W^{u}$ Branch');
title('Radial Distance $r(t)$', 'FontSize', 16, 'Interpreter', 'latex');
xlabel('Time $t$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$r(t)$', 'FontSize', 16, 'Interpreter', 'latex');
grid on;
legend('show', 'Interpreter', 'latex');% Adjust tick sizes
ax = gca; % Get current axes
ax.FontSize = 14; % Set font size for tick labels

% Plot the calculated orbit in the (x, y)-plane
figure;
plot(x_vals, y_vals, '-', 'LineWidth', 1.5, 'DisplayName', '$W^{u}$ Branch');
title('Trajectory in the $(x, y)$-Plane', 'FontSize', 14, 'Interpreter', 'latex');
xlabel('$x$', 'FontSize', 16, 'Interpreter', 'latex');
ylabel('$y$', 'FontSize', 16, 'Interpreter', 'latex');
grid on;
legend('show', 'Interpreter', 'latex');
% Adjust tick sizes
ax = gca; % Get current axes
ax.FontSize = 14; % Set font size for tick labels
%%
% Par metros iniciales
% K = 0.0015749; % Valor dado de K
K = 0.1;
h = -1.7; % Valor de h para este plot
x0_range = linspace(1, 15, 500); % Valores de x0
```

```
theta_range = linspace(0, 2*pi, 500); % Valores de theta
[X0, Theta] = meshgrid(x0_range, theta_range); % Mallado para (x0, theta)

% Calcular velocidad inicial v seg n la relaci n
v_squared = 2 * (h + X0.^2 / 2 - 1 ./ X0 + K .* X0);
v_squared(v_squared < 0) = NaN; % Filtrar valores negativos para evitar complejos
V = sqrt(v_squared); % Velocidad inicial real

% Calcular energ a osculante E_s^0
E_s = (V.^2 / 2) + (X0.^2 / 2) + X0 .* V .* sin(Theta) - (1 ./ X0);

% Figura para el plot
figure;
hold on;

% Regi n donde E_s^0 < 0 (magenta)
contourf(X0, Theta, real(E_s), [-Inf 0], 'm', 'DisplayName', 'E_s^0 < 0');

% Regi n donde E_s^0 > 0 (azul)
contourf(X0, Theta, real(E_s), [0 Inf], 'b', 'DisplayName', 'E_s^0 > 0');

% Formato del gr fico
title('Regi n Err tica para h = -1.7');
xlabel('x_0');
ylabel('\theta');
legend show;
xlim([2,14]);
ylim([4, 5.5]);
grid on;
hold off;

%%
% Par metros iniciales
h = -1.7; % Valor de h para este plot
x0_range = linspace(1, 15, 500); % Valores de x0
theta_range = linspace(0, 2*pi, 500); % Valores de theta
[X0, Theta] = meshgrid(x0_range, theta_range); % Mallado para (x0, theta)

% Valores de K para comparar
K_values = [0.001, 0.002, 0.005];
colors = [0.8 0 0.8; 0 0.8 0.8; 0.2 0.5 0.2]; % Colores lisos para las regiones de cada K

% Figura para el plot
figure;
hold on;

% Loop sobre los valores de K
for i = 1:length(K_values)
    K = K_values(i);

    % Calcular velocidad inicial v seg n la relaci n
    v_squared = 2 * (h + X0.^2 / 2 - 1 ./ X0 + K .* X0);
    v_squared(v_squared < 0) = NaN; % Filtrar valores negativos para evitar complejos
    V = sqrt(v_squared); % Velocidad inicial real

    % Calcular energ a osculante E_s^0
    E_s = (V.^2 / 2) + (X0.^2 / 2) + X0 .* V .* sin(Theta) - (1 ./ X0);

    % Regi n donde E_s^0 < 0
    contourf(X0, Theta, real(E_s), [-Inf 0], 'LineStyle', 'none', 'FaceColor', colors(i,
        :), 'FaceAlpha', 0.7);
end

% Formato del gr fico
title('Regi n Err tica para Diferentes Valores de K');
xlabel('x_0');
ylabel('\theta');
legend({'K = 0.001', 'K = 0.002', 'K = 0.005'}, 'Location', 'best');
xlim([2,14]);
```

```
ylim([4, 5.5]);
grid on;
hold off;
%%
function [x1, x2, h1, h2] = eq_and_energies(K)
    % Parameters
    tol = 1e-14; % Tolerance for Newton's method
    max_iter = 100; % Maximum number of iterations

    % Define f(x) and its derivative
    f1 = @(x) x^3 - K*x^2 + 1;
    df1 = @(x) 3*x^2 - 2*K*x;

    f2 = @(x) x^3 - K*x^2 - 1;
    df2 = @(x) 3*x^2 - 2*K*x;

    % Define Hamiltonian (energy function)
    H = @(x) 0.5 * x^2 + 1 / abs(x) - K * x;

    % Initial guesses based on the max expressions
    x1 = max(-1, -1/sqrt(K)); % Seed for x1 (L1)
    x2 = max(1, 2*K/3);       % Seed for x2 (L2)

    % Newton's method for x1
    for iter = 1:max_iter
        x1_new = x1 - f1(x1) / df1(x1);
        if abs(f1(x1_new)) < tol
            x1 = x1_new;
            break;
        end
        x1 = x1_new;
    end

    % Newton's method for x2
    for iter = 1:max_iter
        x2_new = x2 - f2(x2) / df2(x2);
        if abs(f2(x2_new)) < tol
            x2 = x2_new;
            break;
        end
        x2 = x2_new;
    end

    % Calculate energies
    h1 =-H(x1); % Energy at x1
    h2 = -H(x2); % Energy at x2

    % % Debugging Output
    % fprintf('Debugging Information:\n');
    % fprintf('x1 = %.14f, f1(x1) = %.14f\n', x1, f1(x1));
    % fprintf('x2 = %.14f, f2(x2) = %.14f\n', x2, f2(x2));
    % fprintf('h1 = %.14f (should be negative)\n', h1);
    % fprintf('h2 = %.14f (should be negative)\n', h2);
    %
    % % Output results
    % fprintf('Equilibrium points and energies:\n');
    % fprintf('x1 = %.14f, h1 = %.14f\n', x1, h1);
    % fprintf('x2 = %.14f, h2 = %.14f\n', x2, h2);
end




function val = g(x)
    val = x(2);  % g(x) = velocity (second component of state vector)
end

function h = hamiltonian(x,y,K)
```

```
    r = (x^2+y^2)^(1/2);
    h = y^2/2+x^2/2+1/r-K*x;

end

function [eigenvalues, eigenvectors] = equilibriumEigen(xi)
    % equilibriumEigen - Computes and normalizes eigenvectors of the Jacobian
    % at the equilibrium point (xi, 0, 0, xi), and orders the eigenvalues and
    % eigenvectors such that:
    % - First is the stable eigenvalue and eigenvector (negative real part)
    % - Second is the unstable eigenvalue and eigenvector (positive real part)
    %
    % Input:
    %   xi - x-coordinate of the equilibrium point
    %
    % Output:
    %   eigenvalues - Vector with eigenvalues, first stable, second unstable
    %   eigenvectors - Matrix whose columns are the corresponding eigenvectors

    % Compute r = |xi|
    r = abs(xi);

    % Construct the Jacobian matrix
    J = [ 0,  1,          1,          0;
         -1,  0,          0,          1;
         (2 * xi^2) / r^5, 0,  0,          1;
         0, -1 / abs(xi)^3, -1,          0 ];

    % Compute eigenvalues and eigenvectors
    [raw_eigenvectors, D] = eig(J);
    raw_eigenvalues = diag(D); % Extract eigenvalues as a vector

    % Identify the indices for stable and unstable eigenvalues
    stable_index = find(real(raw_eigenvalues) < 0, 1); % First negative eigenvalue
    unstable_index = find(real(raw_eigenvalues) > 0, 1); % First positive eigenvalue

    % Extract and normalize the stable eigenvector
    stable_eigenvalue = raw_eigenvalues(stable_index);
    stable_eigenvector = raw_eigenvectors(:, stable_index) / norm(raw_eigenvectors(:,
        stable_index));

    % Extract and normalize the unstable eigenvector
    unstable_eigenvalue = raw_eigenvalues(unstable_index);
    unstable_eigenvector = raw_eigenvectors(:, unstable_index) / norm(raw_eigenvectors(:,
        unstable_index));

    % Construct the output lists
    eigenvalues = [stable_eigenvalue; unstable_eigenvalue];
    eigenvectors = [stable_eigenvector, unstable_eigenvector];
end


function [PoincareMapTimes, PoincareMapSols, full_orbit] = poincare(g, tol, K, x0,
    n_crossing, idir)
    % Maximum en uiterations allowed for refinement
    nmax = 100000;

    % Initialize outputs
    PoincareMapSols = zeros(n_crossing, 4);
    PoincareMapTimes = zeros(n_crossing, 1);
    full_orbit = []; % Full trajectory storage

    % ODE solver options
    options = odeset('RelTol', 1e-10, 'AbsTol', 1e-10);

    % Time and step settings
    tau = 0;
    h = 0.01; % Initial step size
    tspan_fast = 0:1e-3:h;  % Larger steps for faster integration
```

```
    tspan_slow = 0:1e-4:0.001;  % Smaller steps for precise refinement

    % Plot initialization
    figure;
    hold on;
    grid on;
    xlabel('x');
    ylabel('y');
    title('Real-Time Orbit Plot in 2D');

    for ncross = 1:n_crossing
        if ncross ~= 1
            x0 = xk; % Update initial condition
        end

        % Initialize trajectory segment
        found = 0;
        orbit_segment = [];

        % Coarse integration until crossing is detected
        while ~found
            % Use larger steps before crossing
            [~, x_k1] = ode45(@(t, x) f(t, x, K, idir), [0, tspan_fast], x0, options);
            tau = tau + h;
            orbit_segment = [orbit_segment; x_k1]; % Append trajectory segment

            % Plot the segment
            plot(x_k1(:, 1), x_k1(:, 2), 'b');
            drawnow;

            % Check for crossing
            if g(x0) * g(x_k1(end, :)) < 0
                found = 1;

                % Exclude crossing point to avoid redundancy
                x_k1 = x_k1(1:end-1, :);
            end
            x0 = x_k1(end, :); % Update starting point for next step
        end

        % Refinement with smaller steps
        xk = x0; % Start from the detected point
        n = 0;
        while abs(g(xk)) > tol && n < 300
            % Compute time correction step
            delta_tau = -g(xk) / xk(4);
            tau = tau + idir * delta_tau;

            % Refine trajectory using smaller steps
            [~, x_k1] = ode45(@(t, x) f(t, x, K, idir), [0 abs(delta_tau)], xk, options);
            xk = x_k1(end, :);

            % Append refined points excluding duplicates
            orbit_segment = [orbit_segment; x_k1(1:end-1, :)];

            % Plot refined segment
            plot(x_k1(:, 1), x_k1(:, 2), 'b');
            drawnow;

            n = n + 1;
        end

        % Handle maximum iteration case
        if n >= nmax
            disp('Iteration surpassed');
            PoincareMapTimes(ncross, 1) = NaN;
            PoincareMapSols(ncross, :) = NaN;
        else
            % Store crossing information
```

```
            PoincareMapTimes(ncross, 1) = tau;
            PoincareMapSols(ncross, :) = xk;
        end

        % Append the trajectory segment to the full orbit
        full_orbit = [full_orbit; orbit_segment];
    end

    hold off;
end
function df = f(~, x, K, idir)
    df = zeros(4, 1);
    r = sqrt(x(1)^2 + x(2)^2);

    df(1) = x(3) + x(2);
    df(2) = x(4) - x(1);
    df(3) = x(4) - x(1) / r^3 - K;
    df(4) = -x(3) - x(2) / r^3;

    if idir == -1
        df = -df;
    end
end
% function py = dy(x, K, H)
%     py = -(2*H + x^2 + 2/abs(x) - 2*K*x)^(1/2);
%
%     py = [py,py];
% end
function py = dy(x, K, H)
    % compute_py_branches - Computes the two possible values of p_y
    % based on the Hamiltonian equation.
    %
    % Inputs:
    %   x - Initial x value
    %   K - System parameter
    %   H - Energy level (Hamiltonian)
    %
    % Output:
    %   py - A vector containing the two possible p_y values:
    %        py(1): Positive root branch
    %        py(2): Negative root branch

    % Compute the discriminant
    discriminant = x^2 - 2 * (K * x - 1 / abs(x) - H);

    % Check if the discriminant is non-negative
    if discriminant < 0
        error('The discriminant is negative. No real solution exists for p_y.');
    end

    % Compute the two possible p_y values
    py_pos = x + sqrt(discriminant);  % Positive root branch
    py_neg = x - sqrt(discriminant);  % Negative root branch

    % Return the results as a vector
    py = [py_pos, py_neg];
end


% Bisection Method for Finding Periodic Orbits
function c = bisection_method(x1, x2, K, h, ysign, tol, max_counter)
    H = h;
    p_y1 = dy(x1, K, H);
    p_y1 = p_y1(2);
    p_y2 = dy(x2, K, H);
    p_y2 = p_y2(2);
    bool = 0;
    state1 = [x1, 0, 0, p_y1];
    state2 = [x2, 0, 0, p_y2];
```

```
    if F(x1, p_y1, K, H, tol) == 0
        c = x1;
        bool = 1;
    end
    if F(x2, p_y2, K, H, tol) == 0
        c = x2;
        bool = 1;
    end

    counter = 0;
    while abs(x2 - x1) > tol && counter < max_counter && bool == 0
        c = (x2 + x1) / 2;
        p_yc = dy(c, K, H);
        p_yc = p_yc(2);
        if F(c, p_yc, K, H, tol) == 0
            bool = 1;
        else
            if F(x1, p_y1, K, H, tol) * F(c, p_yc, K, H, tol) < 0
                x2 = c;
                p_y2 = p_yc;
            else
                x1 = c;
                p_y1 = p_yc;
            end
        end
        counter = counter + 1;
    end
    c = (x2 + x1) / 2;
end

function res = F(x, p_y, K, h, tol)
    x0 = [x, 0, 0, p_y];
    idir = +1;
    n_crossing = 1;
    [~, sols, ~] = poincare(@g, tol, K, x0, n_crossing, idir);
    res = sols(3); % Check third component (y') at Poincare crossing
end


% IMAGINARY PART ALSO EVEC, EVAL
function [eigenvalues, eigenvectors] = equilibriumEigen_imaginary(xi)
    % equilibriumEigen - Computes and returns all eigenvalues and eigenvectors
    % of the Jacobian at the equilibrium point (xi, 0, 0, xi).
    %
    % Input:
    %   xi - x-coordinate of the equilibrium point
    %
    % Output:
    %   eigenvalues - Vector with all eigenvalues
    %   eigenvectors - Matrix whose columns are the corresponding eigenvectors

    % Compute r = |xi|
    r = abs(xi);

    % Construct the Jacobian matrix
    J = [ 0,  1,         1,         0;
         -1,  0,         0,         1;
         (2 * xi^2) / r^5, 0,  0,         1;
         0, -1 / abs(xi)^3, -1,         0 ];

    % Compute eigenvalues and eigenvectors
    [eigenvectors, D] = eig(J);
    eigenvalues = diag(D); % Extract eigenvalues as a vector

    % Normalize eigenvectors
    for i = 1:size(eigenvectors, 2)
        eigenvectors(:, i) = eigenvectors(:, i) / norm(eigenvectors(:, i));
    end
```

```
end


function [PoincareMapTimes, full_orbit] = poincare_times(g, tol, K, x0, total_time, idir)
    % Computes the trajectory up to a specified total time without refinement.
    % Inputs:
    %   g          - Function defining the Poincar  section.
    %   tol        - Tolerance for checking crossings (still used for output).
    %   K          - System parameter.
    %   x0         - Initial condition (4D vector: [x, y, px, py]).
    %   total_time - Total simulation time.
    %   idir       - Direction of integration (+1 for forward, -1 for backward).
    %
    % Outputs:
    %   PoincareMapTimes - Times when the trajectory crosses the Poincar  section.
    %   full_orbit       - Complete trajectory data with time.

    % Initialize outputs
    PoincareMapTimes = [];
    full_orbit = [];

    % ODE solver options
    options = odeset('RelTol', 1e-10, 'AbsTol', 1e-10);

    % Time and step settings
    tau = 0; % Accumulated time
    h = 10; % Initial step size
    tspan = [0, h]; % Time span for integration steps

    % Plot initialization
    figure;
    hold on;
    grid on;
    xlabel('x');
    ylabel('y');
    title('Real-Time Orbit Plot in 2D');

    % Simulate until total time is reached
    while tau < total_time
        % Integrate using ode45
        [t_segment, x_segment] = ode45(@(t, x) f(t, x, K, idir), tspan + tau, x0,
            options);

        % Update accumulated time and initial condition
        tau = t_segment(end);
        x0 = x_segment(end, :);

        % Store the trajectory segment
        full_orbit = [full_orbit; [t_segment, x_segment]];

        % Check for crossings with Poincar  section
        for i = 1:size(x_segment, 1) - 1
            if g(x_segment(i, :)) * g(x_segment(i + 1, :)) < 0
                % Linear interpolation for crossing time
                t_cross = t_segment(i) + (t_segment(i + 1) - t_segment(i)) * ...
                          (-g(x_segment(i, :))) / (g(x_segment(i + 1, :)) -
                             g(x_segment(i, :)));
                PoincareMapTimes = [PoincareMapTimes; t_cross];
            end
        end

        % Plot the segment
        plot(x_segment(:, 1), x_segment(:, 2), 'b');
        drawnow;

        % Stop if total time is reached
        if tau >= total_time
            break;
        end
```

```
1151        end
1152
1153        hold off;
1154    end
```

References

[1] Barrabés, E., Ollé, M., Borondo, F., Farrelly, D., & Mondelo, J. M. (2012). Phase space structure of the hydrogen atom in a circularly polarized microwave field. Physica D: Nonlinear Phenomena, 241(4), 333-349.

[2] S.A. Astakhov, A.D. Burbanks, S. Wiggins, and D. Farrelly. Chaos assisted in the capture of irregular moons. Nature, 423(6937):264–267 2003.

[3] Sugon Jr, Q., Bennett, C. D. G., & McNamara, D. J. (2024). Hydrogen atom as a nonlinear oscillator under circularly polarized light: epicyclical electron orbits. arXiv preprint arXiv:2410.00056.

[4] Fejoz, J., & Guardia, M. (2023). A remark on the onset of resonance overlap. Regular and Chaotic Dynamics, 28(4), 578-584.