Universidad Zaragoza
1542

TRABAJO FIN DE GRADO

Dinámica y control de sistemas híbridos clásico-cuánticos desde una perspectiva geométrica

Autor:

Fernando Ezquerro Sastre

Director

Dr. Jesús Clemente Gallardo

Facultad de Ciencias
Departamento de Física Teórica

28 de junio de 2019

Resumen

En este trabajo se va a estudiar la dinámica y el control de los sistemas híbridos usando la geometría diferencial. Primero se va a presentar la formulación geométrica de la Mecánica Cuántica. Posteriormente se van a deducir las ecuaciones de Ehrenfest como ejemplo de dinámica de sistema híbrido y se van a geometrizar. Tras ello se va a presentar la condición de controlabilidad y de control óptimo para el formalismo geométrico. Finalmente, se va a aplicar todo lo anterior a un ejemplo sencillo de sistema híbrido en el que se van a realizar varios ejemplos de control óptimo.

Índice

1. Introducción y Objetivos

En este trabajo se van a estudiar los sistemas híbridos con el formalismo geométrico, esto significa que se va a usar la geometría diferencial para tratarlos. Los sistemas híbridos están formados por grados de libertad que se tratan de forma clásica y otros que se tratan de forma cuántica pero que están acoplados. Esto suele usarse cuando la resolución de todo el sistema de forma cuántica es demasiado compleja, por lo que se aproxima que parte del sistema se puede tratar de forma clásica.

Un ejemplo de sistemas híbridos son los sistemas moleculares en los que se separa el bloque formado por la parte nuclear y los electrones internos que se puede tratar de forma clásica y los electrones de valencia que se tratan de forma cuántica. Este tipo de modelos pueden ser particularmente útiles en, por ejemplo, el uso de superficies de energía potencial (PES). Este consiste en resolver la ecuación de Schrödinger de la parte electrónica en función de las posiciones de los núcleos, con lo que se obtiene una superficie de energía potencial de la parte electrónica en función de las posiciones nucleares. Usando esta superficie se puede obtener la dinámica nuclear teniendo en cuenta solamente los distintos grados de libertad nucleares, es decir, que toda la información de la parte electrónica está en la superficie de energía potencial.

Como queremos geometrizar los sistemas híbridos, y estos están formados por una parte clásica y una parte cuántica, hay que saber cómo geometrizar cada una de estas partes. El uso de la geometría diferencial para la mecánica clásica se desarrolló durante la segunda mitad del siglo XX (puede consultarse por ejemplo en [1]). Los elementos más relevantes de este formalismo son: el espacio de fases del sistema se identifica con una variedad diferencial, las magnitudes físicas son funciones diferenciables sobre esta variedad y la dinámica esta dada por las curvas integrales de un campo vectorial que es el campo hamiltoniano, el cual se puede obtener a partir del corchete de Poisson del hamiltoniano. La geometrización de la Mecánica Cuántica se empezó a desarrollar en los años 70 en trabajos como [2] y se ha conseguido tanto en la imagen de Schrödinger como en la de Heisenberg. En la primera sección de este trabajo se desarrollará brevemente el formalismo geométrico en la imagen de Schrödinger basándonos en desarrollos más recientes como [3].

Tras esto, en la segunda sección del trabajo, estudiaremos las ecuaciones de Ehrenfest y su geometrización. Las ecuaciones de Ehrenfest son un caso concreto de dinámica de sistemas híbridos en el que se tratan los complejos moleculares como se ha descrito anteriormente, es decir, la parte nuclear de forma clásica y la parte electrónica de forma cuántica. Tras una deducción de las ecuaciones de Ehrenfest se presentará una geometrización de estas a partir de la geometrización de las partes clásicas y cuánticas.

La geometrización de las ecuaciones de Ehrenfest nos permite aplicar los teoremas de control de la geometría diferencial a un sistema cuya dinámica está definida por estas ecuaciones. En la tercera sección nos centraremos en las condiciones de controlabilidad del sistema desde un punto de vista geométrico y en la aplicación del control óptimo sobre estas ecuaciones usando el principio de máximo de Pontriaguin. Esto nos permitirá analizar la controlabilidad de un sistema híbrido y cómo controlarlo optimizando ciertos parámetros como la energía empleada o el tiempo.

Finalmente usaremos estas herramientos para analizar un ejemplo sencillo de sistema híbrido descrito mediante las ecuaciones de Ehrenfest, del cual obtendremos las ecuaciones de la dinámica usando para la parte cuántica dos enfoques distintos: los estados del espacio de Hilbert y las matrices densidad correspondientes a estados puros. Una vez obtenida la dinámica comprobaremos la controlabilidad del sistema y resolveremos numéricamente un problema de control óptimo sobre el sistema comparando los resultados obtenidos con los dos tratamientos distintos de la parte cuántica.

Los objetivos del trabajo son:

- Obtener las ecuaciones de Ehrenfest formuladas de forma geométrica para un caso general
- Obtener las ecuaciones de la dinámica para los estados y coestados para el principio de máximo de Pontriagin usando las ecuaciones de Ehrenfest para un caso general
- Calcular de forma explícita las ecuaciones de Ehrenfest de forma geométrica para un ejemplo sencillo de sistema híbrido usando para la parte cuántica los estados del espacio de Hilbert y las matrices densidad
- Resolver numéricamente varios casos de control óptimo para el mismo ejemplo centrándonos en controlar la parte clásica

Podemos considerar nuestro ejemplo de referencia el control de reacciones químicas. Este puede estudiarse como un sistema híbrido en el que nos interesa controlar la posición y momento de los núcleos, pero no el estado cuántico de los electrones. Esto se debe a que la condición más importante para que se produzcan estas reacciones es aproximar lo suficiente los átomos y que no tengan una velocidad relativa entre ellos alta, esto nos lleva a tener que controlar la posición y momento de los núcleos. Debido a este argumento en la resolución numérica del control nos centraremos en controlar la parte clásica.

2. Formalismo geométrico de la Mecánica Cuántica

En el formalismo geométrico de la Mecánica Cuántica se busca un formalismo similar al formalismo geométrico de la Mecánica Clásica para la Mecánica Cuántica, centrándonos en este caso en situaciones de dimensión finita. Por sencillez vamos a tratar la Mecánica Cuántica desde la imagen de Schrödinger. Los posibles estados físicos están representados por los rayos en el espacio de Hilbert, es decir, todos los puntos que se diferencian en una fase global o en su norma. Las magnitudes físicas están descritas por operadores autoadjuntos con los que se puede definir el valor medio, que es una función de los estados. La dinámica está dada por la ecuación de Schrödinger que describe la evolución de los estados en el espacio de Hilbert como la solución de una ecuación diferencial. Lo que se busca en esta sección es adaptar estos elementos al formalismo geométrico.

2.1. Estados

Se quiere traducir los estados del espacio de Hilbert $\mathcal{H}$ a una variedad diferencial real M_Q. Para ello separamos la parte real e imaginaria de las coordenadas en una determinada base. Tomando como base del espacio de Hilbert $\{|a_i\rangle\}$ los estados se pueden expresar como:

$$|\psi\rangle = \sum_{i=1}^{n} b_i \, |a_i\rangle = \sum_{i=1}^{n} (q_i + ip_i) \, |a_i\rangle \quad q_i, p_i \in \mathbb{R} \tag{2.1}$$

Con el conjunto de estas coordenadas reales $(q_1, \ldots, q_n, p_1, \ldots, p_n) \equiv (p, q)$ se forma una variedad diferencial real, pero para que esta sea equivalente a $\mathcal{H}$ hay que traducir su estructura algebraica a estructuras tensoriales en M_Q.

Para traducir sus estructuras se usa el fibrado tangente TM_Q. En este al vector correspondiente al estado $|\psi\rangle$, le asociamos el campo vectorial X_ψ que se puede escribir de la siguiente forma en las coordenadas anteriores:

$$X_\psi = \sum_{i=1}^{n} \left(q_i \frac{\partial}{\partial q_i} + p_i \frac{\partial}{\partial p_i} \right) \qquad \frac{\partial}{\partial q_i}, \frac{\partial}{\partial p_i} \in TM_Q \tag{2.2}$$

Primero la estructura compleja se traduce con un tensor $J : TM_Q \to TM_Q$ que cumple que ((p, q) son las coordenadas de los campos):

$$J(p, q) = (-p, q) \;\; \Rightarrow J^2 = -\mathbb{I} \tag{2.3}$$

Para traducir el producto escalar entre dos estados $|\psi\rangle$ y $|\phi\rangle$ lo escribimos en sus coordenadas reales (q_ψ, p_ψ) y (q_ϕ, p_ϕ), se obtiene:

$$\langle\psi|\phi\rangle = \langle(q_\psi, p_\psi)|(q_\phi, p_\phi)\rangle = \langle q_\psi|q_\phi\rangle + \langle p_\psi|p_\phi\rangle + i(\langle q_\psi|p_\phi\rangle - \langle p_\psi|q_\phi\rangle) \tag{2.4}$$

Si escribimos este resultado en la variedad real, es decir,como una operación con tensores sobre los campos vectoriales sobre un determinado punto $|\chi\rangle$ asociados a los estados $|\psi\rangle, |\phi\rangle$ se obtiene:

$$\langle\psi|\phi\rangle = \langle X_\psi|X_\phi\rangle \, (\chi) = g(X_\psi, X_\phi) + i\omega(X_\psi, X_\phi) \tag{2.5}$$

A partir de la expresión en coordenadas reales se puede deducir que el tensor g será simétrico y el tensor ω antisimétrico. Se puede probar de forma sencilla (ver en [6]) que g es una métrica y ω una forma simpléctica. Los tres tensores (J, g, ω) cumplen la condición de compatibilidad $g(X_\psi, X_\phi) = \omega(JX_\psi, X_\phi)$ que traduce la sesquilinealidad del producto escalar a la variedad real. El conjunto (J, g, ω) al cumplir la condición de compatibilidad forma una estructura de Kähler. También podemos escribir estos tensores en su versión contravariante que se usará posteriormente. En la base escrita anteriormente tendrían la siguiente forma:

$$G = \sum_{i=1}^{n} \left(\frac{\partial}{\partial q_i} \otimes \frac{\partial}{\partial q_i} + \frac{\partial}{\partial p_i} \otimes \frac{\partial}{\partial p_i} \right) \;\; \Omega = \sum_{i=1}^{n} \left(\frac{\partial}{\partial q_i} \wedge \frac{\partial}{\partial p_i} \right) \;\; J = \sum_{i=1}^{n} \left(\frac{\partial}{\partial p_i} \otimes dq_i - \frac{\partial}{\partial q_i} \otimes dp_i \right) \tag{2.6}$$

2.2. Magnitudes físicas y observables

Las magnitudes físicas se representan mediante operadores lineales sobre $\mathcal{H}$ que son autoadjuntos para el producto escalar. Para adaptarlos al formalismo se toma el valor medio sobre estados normalizados de la siguiente forma:

$$f_X(\psi) = \frac{1}{2}\langle\psi|X\psi\rangle \qquad \langle\psi|\psi\rangle = 1 \tag{2.7}$$

Si tomamos el estado $|\psi\rangle$ como un punto de M_Q se puede ver f_X como una función cuadrática sobre M_Q, que cumple que si el operador X es un observable (autoadjunto) producirá valores reales con esta función.

Esta función definida anteriormente tiene el problema de no ser constante para los puntos de M_Q que tienen distinta norma pero representan el mismo estado, por lo que se define una nueva función:

$$e_X(\psi) = \frac{\langle\psi|X\psi\rangle}{\langle\psi|\psi\rangle} \tag{2.8}$$

Esta función que define el valor esperado del observables X en el rayo ψ sí que cumple que es constante para los puntos con distinta norma, pero ya no es una función cuadrática. A partir de esta también se pueden obtener los otros valores físicos relevantes: los autovalores y autovectores del operador. Los autovectores corresponden a los puntos críticos de esta función, es decir, que cumplen que $de_X(\psi) = 0 \Leftrightarrow |\psi\rangle$ es un autovector. Para obtener los autovalores correspondientes solo hay que aplicar la función sobre los autovectores.

2.3. Dinámica

La dinámica vendrá dada por las curvas integrales del siguiente campo vectorial:

$$X_H = \hbar^{-1}\Omega(df_H, \cdot) = \hbar^{-1}\sum_{i=1}^{n}\left(\frac{\partial f_H}{\partial p_i}\frac{\partial}{\partial q_i} - \frac{\partial f_H}{\partial q_i}\frac{\partial}{\partial p_i}\right) \tag{2.9}$$

Veamos cómo las curvas integrales coinciden con la ecuación de Schrödinger. Primero hay que calcular $f_H(\psi)$ para lo que hay que escribir el hamiltoniano en las coordenadas reales, por lo que será una matriz de dimensión $2n \times 2n$. El cálculo sería de la siguiente forma:

$$f_H = \frac{1}{2}\begin{pmatrix} q_1, p_1, \cdots, q_n, p_n \end{pmatrix}\begin{pmatrix} H_{q_1q_1} & H_{q_1p_1} & \cdots & H_{q_1q_n} & H_{q_1p_n} \\ H_{p_1q_1} & H_{p_1p_1} & \cdots & H_{p_1q_n} & H_{p_1p_n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ H_{q_nq_1} & H_{q_np_1} & \cdots & H_{q_nq_n} & H_{q_np_n} \\ H_{p_nq_1} & H_{p_np_1} & \cdots & H_{p_nq_n} & H_{p_np_n} \end{pmatrix}\begin{pmatrix} q_1 \\ p_1 \\ \vdots \\ q_n \\ p_n \end{pmatrix} \tag{2.10}$$

A partir de esta expresión, derivando se obtienen las siguientes curvas integrales para las coordenadas reales:

$$\frac{dq_i}{dt} = \hbar^{-1}\left(H_{p_iq_1}q_1 + H_{p_ip_1}p_1 + \ldots + H_{p_iq_n}q_n + H_{p_ip_n}p_n\right) \tag{2.11}$$

$$\frac{dp_i}{dt} = -\hbar^{-1}\left(H_{q_iq_1}q_1 + H_{q_ip_1}p_1 + \ldots + H_{q_iq_n}q_n + H_{q_ip_n}p_n\right) \tag{2.12}$$

Podemos expresar estas ecuaciones de forma compacta usando el tensor J y volviendo a la notación de estados ψ, obteniéndose la siguiente relación:

$$\frac{d\psi}{dt} = -\hbar^{-1} JH\psi \Rightarrow J\hbar\frac{d\psi}{dt} = H\psi \tag{2.13}$$

Dicha expresión es la ecuación de Schrödinger, ya que como se ha mencionado anteriormente la acción de J sobre un vector representa en el espacio real, multiplicar por la unidad imaginaria en el espacio complejo.

Esta ecuación se puede definir usando el corchete de Poisson asociado a la forma simpléctica ω, ya que este actúa de la siguiente forma:

$$\{f, g\} = \sum_{i=1}^{n} \left(\frac{\partial f}{\partial p_i}\frac{\partial g}{\partial q_i} - \frac{\partial f}{\partial q_i}\frac{\partial g}{\partial p_i} \right) \tag{2.14}$$

Por tanto la ecuación de la dinámica se puede escribir de la siguiente forma:

$$X_H = \hbar^{-1}\{f_H, \cdot\} \tag{2.15}$$

Ahora veamos un ejemplo que se usará en la sección 5. Se van usar unidades naturales $\hbar = 1$. Consideremos un sistema cuántico de dos niveles con un hamiltoniano $H = B_x\sigma_x + B_y\sigma_y$ con σ las matrices de Pauli y B variables dependientes del tiempo (este sería el caso de aplicar un campo magnético a un sistema con spín 1/2 en las direcciones x e y). Los estado escritos en las coordenadas reales tendrán cuatro componentes $(\phi_1, \phi_2) = (q_1, p_1, q_2, p_2)$ con $\phi_j = q_j + ip_j$. El hamiltoniano escrito en las coordendas reales es:

$$H = B_x \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} + B_y \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & B_x & B_y \\ 0 & 0 & -B_y & B_x \\ B_x & -B_y & 0 & 0 \\ B_y & B_x & 0 & 0 \end{pmatrix} \tag{2.16}$$

Con este hamiltoniano podemos calcular f_H obteniéndose:

$$f_H = \frac{1}{2}\begin{pmatrix} q_1, p_1, q_2, p_2 \end{pmatrix} \begin{pmatrix} 0 & 0 & B_x & B_y \\ 0 & 0 & -B_y & B_x \\ B_x & -B_y & 0 & 0 \\ B_y & B_x & 0 & 0 \end{pmatrix} \begin{pmatrix} q_1 \\ p_1 \\ q_2 \\ p_2 \end{pmatrix} = B_x(q_1q_2 + p_1p_2) + B_y(q_1p_2 - q_2p_1) \tag{2.17}$$

A partir del cual se pueden obtener las ecuaciones de la dinámica para las coordenadas usando (2.9), con lo que obtenemos:

$$\dot{q}_1 = B_xp_2 - B_yq_2 \quad \dot{p}_1 = -B_xq_2 - B_yp_2 \quad \dot{q}_2 = B_xp_1 + B_yq_1 \quad \dot{p}_2 = -B_xq_1 + B_yp_1 \tag{2.18}$$

Ahora podemos comparar el resultado con el que se obtendría usando las coordenadas complejas:

$$\dot{\phi}_1 = -(iB_x + B_y)\phi_2 \qquad \dot{\phi}_1 = -(iB_x - B_y)\phi_1 \tag{2.19}$$

Que si se escribe en las coordendas reales coincide con (2.18). Por lo que el valor medio y la dinámica coinciden con el valor al calcularlo de la forma convencional.

2.4. El espacio proyectivo

Otro aspecto a tener en cuenta es que, como se ha comentado anteriormente, los estados físicos están representados por los rayos de $\mathcal{H}$ y con estos se forma el espacio proyectivo. Dicho espacio también se puede traducir a este formalismo como una subvariedad $\mathcal{P}(\mathcal{H})$ de M_Q y se puede definir una proyección $\pi : M_Q \to \mathcal{P}$. Esta aplicación actúa de tal forma que todos los puntos de M_Q que corresponden al mismo 'rayo' son proyectados el mismo punto en $\mathcal{P}$. Se denota $[\psi]$ el punto de $\mathcal{P}$ que es la imagen por π del punto ψ en M_Q. Para introducir de forma geométrica la relación de equivalencia que define el proyectivo hay que introducir dos campos vectoriales: Δ que tiene en cuenta las variaciones de módulo y Γ que tiene en cuentas las variaciones de fase global. Esta relación se da porque los puntos que se pueden alcanzar a partir de uno dado por curvas integrales de uno o el otro campo representan esta clase de equivalencia. Los campos tienen la siguiente forma en las coordenadas anteriores:

$$\Delta = \sum_{i=1}^{n} \left(q_i \frac{\partial}{\partial q_i} + p_i \frac{\partial}{\partial p_i} \right) \qquad \Gamma = \sum_{i=1}^{n} \left(p_i \frac{\partial}{\partial q_i} - q_i \frac{\partial}{\partial p_i} \right) \tag{2.20}$$

Otra forma equivalente a usar el proyectivo es usar matrices densidad correspondientes a estados puros, ya que existe un difeomorfismo entre ambos espacios. El cual es una aplicación $u : \mathcal{P} \to D^1(\mathcal{H})$, donde $D^1(\mathcal{H})$ es el subespacio de proyectores de rango uno, que actúa de la siguiente forma: $u([\psi]) = \frac{|\psi\rangle\langle\psi|}{\langle\psi|\psi\rangle} = \rho_\psi$. Como las matrices densidad son autoadjuntas se pueden escribir con coordenadas reales en una base del espacio de operadores correspondiente.

También hay que tener en cuenta la evolución temporal de las matrices densidad y cómo calcular valores medios. La ecuación de Schrödinger para matrices densidad es la ecuación de von Neumann y el valor medio de un operador es la traza del producto de la matriz densidad y el operador:

$$i\frac{d\rho(t)}{dt} = [H, \rho(t)] = H\rho(t) - \rho(t)H \qquad \langle X \rangle = Tr(\rho X) \tag{2.21}$$

Para geometrizar las ecuaciones podemos usar el difeomorfismo existente entre el proyectivo y los estados puros. Como u tiene inversa al ser un difeomorfismo se puede usar el *pullback* de la aplicación inversa $(u^{-1})^*$ que traslada la forma simpléctica del proyectivo a las matrices densidad de estados puros. La imagen de la forma simpléctica de $\mathcal{P}$ se corresponde con un corchete de Poisson, que nos permite calcular la dinámica de las matrices densidad como: $\frac{d\rho}{dt} = \hbar^{-1}\{H, \rho\}$. Como los operadores también son autoadjuntos podemos escribirlos con coordenadas reales en la misma base que ρ. Con esto podemos escribir el valor medio y la evolución temporal como:

$$\langle X \rangle = \sum_{i=1}^{n^2} \sum_{j=1}^{n^2} \rho_i X_j Tr(A_i A_j) = \sum_{i=1}^{n^2} \sum_{j=1}^{n^2} \rho_i X_j \delta_j^i = \sum_{i=1}^{n^2} X_i \rho_i \tag{2.22}$$

$$\frac{d\rho_k}{dt} = \hbar^{-1}\{H, \rho_k\} = \hbar^{-1} \sum_{j=1,l=1}^{n^2} C_{jk}^l H_j \rho_l \tag{2.23}$$

donde A_k es la base del espacio de operadores autoadjuntos; X_k, H_k y ρ_k son las coordenadas k del operador X, del hamiltoniano y de la matriz densidad en la base A respectivamente; y C_{jk}^l son las constantes de estructura que vienen definidas por la relación $[A_j, A_k] = C_{jk}^l A_l$. Para más detalles del formalismo geométrico con matrices densidad puede consultarse [3] y [5].

3. Sistemas híbridos. Ecuaciones de Ehrenfest

Como se ha indicado en la introducción, los sistemas híbridos son sistemas en los que se combinan grados de libertad cúanticos y clásicos. Se suelen usar cuando la resolución de todo el sistema de forma cuántica es demasiado compleja por lo que se aproximan algunos grados de libertad cuánticos por clásicos. Uno de los casos más representativos es el tratamiento de moléculas, en los que se puede recurrir a un tratamiento clásico de los núcleos y cuántico de los electrones de valencia que se justifica por la diferencia de masas y las distintas escalas temporales para sus dinámicas. Un método para realizar esta separación son las ecuaciones de Ehrenfest. Veamos cómo se pueden deducir.

3.1. Deducción de las ecuaciones de Ehrenfest

Tomemos como ejemplo el caso de una molécula formada por n átomos y m electrones. Considerando solamente las interacciones electromagnéticas el hamiltoniano se puede escribir como:

$$\begin{aligned} H &= -\hbar^2 \sum_{i=1}^{n} \frac{\nabla_i^2}{2M_i} - \hbar^2 \sum_{i=1}^{m} \frac{\nabla_i^2}{2m} + \frac{1}{4\pi\epsilon_0} \left(\sum_{j<i}^{n} \frac{Z_j Z_i}{|\vec{R}_i - \vec{R}_j|} + \sum_{j<i}^{m} \frac{1}{|\vec{r}_i - \vec{r}_j|} - \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{Z_i}{|\vec{R}_i - \vec{r}_j|} \right) \\ &= T_N + T_e + V_{NN} + V_{ee} + V_{Ne} = T_N + T_e + V = T_N + H_e \end{aligned} \tag{3.1}$$

donde $\vec{R}_i$ y $\vec{r}_i$ son las posiciones de los núcleos y de los electrones respectivamente, M_i es la masa del núcleo i (en unidades de la masa del electrón, ya que se toma esta como masa unidad) y Z_i es la carga eléctrica de núcleo i. La resolución de este hamiltoniano de forma cuántica es demasiado compleja en general, por lo que hay que realizar aproximaciones.

Una primera aproximación es factorizar la función de onda en una parte nuclear y otra parte electrónica. Dicha factorización se puede realizar de forma exacta para un cierto tiempo t_0, pero la evolución de esta no tiene porque conservar esa factorización. Al considerar la función de onda factorizada se esta imposibilitando el posible entrelazamiento entre la parte nuclear y electrónica. Entonces para un cierto tiempo t_0 la función de onda se puede escribir:

$$\Phi(r, R, t_0) \simeq \Phi_e(r, t_0)\Phi_N(R, t_0) \tag{3.2}$$

donde $\Phi_e(R, t)$ es la función de ondas de la parte electrónica y $\Phi_N(R, t)$ es la función de ondas que corresponde a la parte nuclear. Las variables r se refieren al conjunto de posiciones de los electrones ($r \equiv \vec{r}_1, \ldots, \vec{r}_m$) y R al conjunto de posiciones de los núcleos ($R \equiv \vec{R}_1, \ldots, \vec{R}_n$).

Para obtener las ecuaciones de Ehrenfest se hace una modificación a la factorización de la ecuación (3.2) añadiendo un factor multiplicativo, por lo que la relación queda de la siguiente forma:

$$\Phi(r, R, t) \simeq \Phi_e(r, t)\Phi_N(R, t) \exp\left(i \int_{t_0}^{t} E_e(t')dt' \right) \tag{3.3}$$

donde $E_e(t) = \langle \Phi_e(r, t)\Phi_N(R, t)|H_e|\Phi_e(r, t)\Phi_N(R, t)\rangle$. Esta aproximación se basa en que la función de onda del núcleo esta muy localizada en comparación con la función de onda de la parte electrónica.

Se puede comprobar (por ejemplo en [8]) que la diferencia de esta función de onda con respecto a la exacta viene dada por:

$$\|\Phi(r,R,t) - \Phi_e(r,t)\Phi_N(R,t)\exp\left(i\int_{t_0}^{t} E_e(t')dt'\right)\| \sim \mathcal{O}(\epsilon/L) \tag{3.4}$$

donde ϵ^2 es la varianza de la densidad de probabilidad de los núcleos en el tiempo t_0 y L la escala de longitud de la función de onda cuántica, que en este caso sería del orden del tamaño de los orbitales atómicos. A partir de esta factorización de la función de onda, se puede llegar a las siguientes ecuaciones para la evolución temporal de la parte electrónica y nuclear:

$$i\hbar\frac{\partial\Phi_e(r,t)}{\partial t} = T_e\Phi_e(r,t) + \left(\int {\Phi_N}^*(R,t)V(r,R)\Phi_N(R,t)dR\right)\Phi_e(r,t) \tag{3.5}$$

$$i\hbar\frac{\partial\Phi_N(R,t)}{\partial t} = T_N\Phi_N(R,t) + \left(\int {\Phi_e}^*(r,t)H_e(r,R)\Phi_e(r,t)dr\right)\Phi_N(R,t) \tag{3.6}$$

La ecuación (3.5) se obtiene introduciendo la función de ondas (3.3) en la ecuación de Schrödinger, multiplicando por la izquierda por ${\Phi_N}^*(R,t)$ e integrando sobre R. Para la ecuación (3.6) se actúa igual, solo que se multiplica por ${\Phi_e}^*(r,t)$ y se integra sobre r.

La segunda aproximación necesaria es plantear las ecuaciones de la parte nuclear de forma semiclásica. Primero escribimos la función de onda nuclear de forma polar, $\Phi_N = A(R,t)\exp(\frac{i}{\hbar}S(R,t))$ con A y S reales. Si sustituimos la forma polar en la ecuación (3.6) y separamos en parte real e imaginaria, se obtienen las siguientes ecuaciones:

$$\frac{\partial A^2(R,t)}{\partial t} = -\sum_{i=1}^{n}\frac{\nabla_i}{M_i}\left(A^2(R,t)\nabla_i S(R,t)\right) \tag{3.7}$$

$$\frac{\partial S(R,t)}{\partial t} = -\sum_{i=1}^{n}\left(\frac{(\nabla_i S(R,t))^2}{2M_i} - \hbar\frac{\nabla^2 A(R,t)}{2M_i A}\right) - \frac{\langle\Phi_e(r,t)|H_e|\Phi(r,t)\rangle}{\hbar} \tag{3.8}$$

La ecuación (3.7) es una ecuación de continuidad para la densidad de probabilidad de los núcleos, ya que A^2 es la densidad de probabilidad de estos. Ahora, en la ecuación (3.8), si tomamos el límite clásico en el que $\hbar \to 0$ y las funciones de onda de los núcleos están localizadas (como se ha usado anteriormente) se puede despreciar el segundo término del sumatorio (hay que tener en cuenta que para el caso electrónico este término sería bastante mayor al estar dividido por la masa), por lo que nos queda una ecuación del tipo Hamilton-Jacobi donde $S(R,t)$ sería la acción y $\nabla_i S(R,t) = \vec{P_i}$ el momento lineal del núcleo i. También al considerar que los núcleos están localizados, la integral en (3.5) se puede aproximar por $V(r,R)$. Teniendo esto en cuenta, la dinámica de la parte nuclear y electrónica se pueden escribir como:

$$\frac{d\vec{P_i}}{dt} = -\langle\Phi_e|\nabla_i H_e(r,R)|\Phi_e\rangle \qquad\qquad i\hbar\frac{d\Phi_e}{dt} = H_e(r,R)\Phi_e \tag{3.9}$$

Al resolver las ecuaciones (3.7) y (3.8) con esta aproximación del límite clásico puede comprobarse (por ejemplo en [8]) que la diferencia entre la solución exacta y la aproximación es:

$$\|\Phi_N(R,t) - A(R,t)e^{i\frac{S(R,t)}{\hbar}}\| \sim \mathcal{O}(\sqrt{m/M}) \tag{3.10}$$

donde M es la masa de los núcleos y m la de los electrones, por lo que la diferencia será pequeña. Usando que el momento es $\vec{P}_i = M_i \frac{d\vec{R}_i}{dt}$, ya podemos obtener las ecuaciones de Ehrenfest a partir de las ecuaciones (3.9)

$$\frac{d\vec{R}_i}{dt} = \frac{\vec{P}_i}{M_i} \tag{3.11}$$

$$\frac{d\vec{P}_i}{dt} = -\langle\psi|\nabla_i H_e(r,R)|\psi\rangle \tag{3.12}$$

$$i\hbar\frac{d}{dt}|\psi\rangle = H_e(r,R)|\psi\rangle \tag{3.13}$$

Una vez obtenidas las ecuaciones se puede comprobar ([8]) que, debido a las aproximaciones que se han tomado, la diferencia entre la solución exacta y la que se obtiene de las ecuaciones de Ehrenfest es del orden de $\mathcal{O}(\epsilon/L + \sqrt{(m/M)})$. Estas ecuaciones serán válidas mientras M sea lo suficientemente grande con respecto a m y la varianza inicial de la función de ondas nuclear $\epsilon^2(t_0)$ sea lo suficientemente pequeña.

3.2. Geometrización de las ecuaciones de Ehrenfest

Para la geometrización del sistema híbrido primero veamos la geometrización de ambas partes (la clásica y la cuántica) por separado.

En la parte nuclear se ha llegado a una formulación Hamiltoniana clásica por lo que se puede geometrizar como cualquier sistema clásico. En la geometrización la variedad diferencial es el espacio de fase formado por las posiciones y momentos de los núcleos es decir $M_C = (\mathbb{R}^3)^{2n}$. Para representar las posiciones y los momentos se va a usar la siguiente notación: $R_{i,j}$ y $P_{i,j}$ que corresponde a la posición y momento respectivamente del núcleo i en la componente j. Los observables son funciones diferenciables $f : M_C \to \mathbb{R}$ tal que $f \in C^\infty(M_C)$, es decir, que para cada punto de M_C asignan un número real que será el resultado de la medida. Como este espacio de fase es una variedad simpléctica, permite la introducción de un corchete de Poisson sobre las funciones diferenciables, que actúa de la siguiente forma:

$$\{f,g\} = \sum_{i=1}^{n}\sum_{j=1}^{3}\left(\frac{\partial f}{\partial R_{i,j}}\frac{\partial g}{\partial P_{i,j}} - \frac{\partial f}{\partial P_{i,j}}\frac{\partial g}{\partial R_{i,j}}\right) \qquad f,g \in C^\infty(M_C) \tag{3.14}$$

Si el hamiltoniano del sistema fuera H se podría obtener la dinámica como las curvas integrales del siguiente campo vectorial: $X_H = \{H, \cdot\}$.

Ahora veamos la parte electrónica. Esta tiene una formulación cuántica por lo que para geometrizarla podemos aplicar lo visto en la primera sección. Aunque para poder aplicarlo debemos considerar el sistema electrónico de dimensión finita. Esto se puede justificar por dos razones. Una es que en la resolución numérica de las ecuaciones al discretizar el sistema pasará a ser de dimensión finita. Otra razón es que el sistema cuántico se puede aproximar a la situación en la que el sistema no es capaz de alcanzar todos los estados posibles de energía y solo consideramos aquellos que tienen una probabilidad significativa, por lo que el sistema pasaría a ser de dimensión finita. En consecuencia como se ha descrito anteriormente tendremos: una variedad diferencial M_Q que corresponde al espacio de estados (ecuación (2.1)), funciones sobre M_Q que actúan como los observables (ecuación (2.7)) y la ecuación de Schrödinger que se puede describir mediante

las curvas integrales de un campo vectorial obtenido mediante un corchete de Poisson (ecuación (2.15)).

Una vez que tenemos la parte nuclear y electrónica en formulación geométrica por separado ahora tenemos que juntar ambas. Primero el espacio de estados será el producto cartesiano del espacio de estados de cada subsistema:

$$M = M_C \times M_Q \tag{3.15}$$

Por ello M será una variedad diferencial y simpléctica al ser el producto cartesiano de dos variedades diferenciales y simplécticas. La elección del producto cartesiano traduce la factorizabilidad de la función de onda que se ha tomado en la deducción de las ecuaciones de Ehrenfest. Las coordenadas que describirán un estado serán un conjunto de posiciones clásicas, de momentos clásicos y de las coordendas reales en M_Q, es decir que un estado $\phi \in M$ será:

$$\phi = (\vec{R}_1, \ldots, \vec{R}_n, \vec{P}_1, \ldots, \vec{P}_n, q_1, \ldots, q_m, p_1, \ldots, p_m) \tag{3.16}$$

donde habría n núcleos y se considera un sistema de m niveles por lo que corresponderá con un espacio de Hilbert de dimensión m. Por lo cual M tendría dimensión $d = 6n + 2m$.

Como la definición rigurosa de los observables para los sistemas híbridos no es necesaria para el trabajo se muestra en el Anexo A. Antes de describir la dinámica definamos las proyecciones sobre cada uno de los dos subsistemas:

$$\pi_C : M_C \times M_Q \to M_C \qquad\qquad \pi_Q : M_C \times M_Q \to M_Q \tag{3.17}$$

Para definir la dinámica se puede aprovechar que cada uno de los subsistemas tiene definido un corchete de Poisson. Debido a esto la dinámica es como combinar dos sistemas clásicos, por lo que el corchete de Poisson del sistema completo será:

$$\{\cdot,\cdot\} = \{\cdot,\cdot\}_C + \hbar^{-1}\{\cdot,\cdot\}_Q \tag{3.18}$$

donde $\{\cdot,\cdot\}_C$ actúa sobre la parte clásica y $\{\cdot,\cdot\}_Q$ sobre la parte cuántica. También podemos definir el tensor simpléctico equivalente sobre M combinando las estructuras simplécticas de M_C y M_Q, y las proyecciones de la ecuación (3.17) de la siguiente forma:

$$\omega = \pi_C^* \omega_C + \hbar^{-1} \pi_Q^* \omega_Q \tag{3.19}$$

donde π_C^* y π_Q^* son los *pullback* de las proyecciones definidas en (3.17). A partir del corchete de Poisson del sistema completo podemos obtener la dinámica como las curvas integrales del siguiente campo vectorial: $X_H = \{f_H, \cdot\}$, donde f_H es el hamiltoniano del sistema completo que en este caso está dado por:

$$f_H = \sum_{i=1}^{n} \frac{\vec{P}_i^2}{2M_i} + \langle \psi(q,p) | H_e(\vec{R}) | \psi(q,p) \rangle \tag{3.20}$$

Las curvas integrales usando este hamiltoniano resultan en las siguientes ecuaciones:

$$\frac{dR_{i,j}}{dt} = \frac{\partial f_H}{\partial P_{i,j}} = \frac{P_{i,j}}{M_i} \qquad i = 1, \ldots, n \quad j = 1, 2, 3 \tag{3.21}$$

$$\frac{dP_{i,j}}{dt} = -\frac{\partial f_H}{\partial R_{i,j}} = -\nabla_{i,j} \langle \psi(q,p) | H_e(\vec{R}) | \psi(q,p) \rangle \qquad i = 1, \ldots, n \quad j = 1, 2, 3 \tag{3.22}$$

$$\frac{dq_i}{dt} = \hbar^{-1} \frac{\partial f_H}{\partial p_i} \qquad i = 1, \ldots, m \tag{3.23}$$

$$\frac{dp_i}{dt} = -\hbar^{-1} \frac{\partial f_H}{\partial q_i} \qquad i = 1, \ldots, m \tag{3.24}$$

Vemos como estas ecuaciones son equivalentes a las ecuaciones de Ehrenfest obtenidas anteriormente, ya que las ecuaciones (3.21) y (3.22) son iguales a las ecuaciones (3.11) y (3.12) mientrás que, como se ha visto en la primera sección, las ecuaciones (3.23) y (3.24) son equivalentes a la ecuación de Schrödinger (3.13).

Por tanto se ha conseguido geometrizar un sistema híbrido clásico-cuántico con todos los elementos necesarios y se ha conseguido recuperar las ecuaciones de Ehrenfest. También se puede hacer un desarrollo equivalente si en la parte cuántica se trabaja con el espacio proyectivo $\mathcal{P}$ en vez de M_Q al disponer de los mismos elementos, pero sobre la variedad diferencial definida en $\mathcal{P}$. Como se ha descrito en la sección anterior se pueden usar las matrices densidad de forma equivalente al proyectivo.

Usando las ecuaciones (2.22) y (2.23) se pueden escribir las ecuaciones de Ehrenfest con matrices densidad de la siguiente forma:

$$\frac{dR_{i,j}}{dt} = \frac{P_{i,j}}{M_i} \qquad i = 1, \ldots, n \quad j = 1, 2, 3 \tag{3.25}$$

$$\frac{dP_{i,j}}{dt} = -\nabla_{i,j} \sum_{k=1}^{m^2} \rho_k H_{ek} \qquad i = 1, \ldots, n \quad j = 1, 2, 3 \tag{3.26}$$

$$\frac{d\rho_k}{dt} = \{H_e, \rho_k\} = \sum_{j=1,l=1}^{m^2} C_{jk}^l H_{ej} \rho_l \tag{3.27}$$

donde A_k es la base del espacio de operadores autoadjuntos, H_{ek} es la coordenada k en esa base del hamiltoniano electrónico, ρ_k es la coordenadas k de la matriz densidad en la base A y C_{jk}^l son las constantes de estructura. Con esta construcción, se ha conseguido escribir las ecuaciones de Ehrenfest geometrizadas usando matrices densidad.

4. Control

Sea la dinámica de un sistema un conjunto de n ecuaciones diferenciales que dependen de un conjunto de m parámetros α dependientes del tiempo ($\dot{x}_i(t) = f_i(x(t), \alpha_j(t))$ con $i = 1, \ldots, n$ y $j = 1, \ldots, m$). Ese conjunto de m parámetros se denominan los controles del sistema porque permiten modificar la dinámica. Los valores de los controles estarán definidos en un cierto conjunto $A \in \mathbb{R}^m$, denominándose controles admisibles aquellos que pertenecen al conjunto.

Un ejemplo podría ser una partícula cargada sometida a la gravedad y a un campo eléctrico orientado en la misma dirección que la fuerza gravitatoria pero del que solo se puede cambiar la intensidad pero no el sentido. La ecuación de la dinámica sería $\dot{v} = qE(t) - mg$. En función del valor de $E(t)$ la velocidad será positiva o negativa, por lo que este controla el sistema y sería el control. Como no se puede cambiar el sentido del campo eléctrico los controles admisibles serían los que cumplan que $E > 0$.

Una vez que tenemos los controles nos podemos preguntar si existen funciones de control que conecten dos puntos arbitrarios del espacio de estados en un tiempo finito. Este problema se denomina controlabilidad del sistema. Para ver qué condición se debe cumplir para que sea controlable vamos a tratar el control desde el punto de vista geométrico (puede consultarse [9] para un desarrollo más detallado de este tema). De esta forma los estados pertenecen a una variedad diferencial M y las funciones de la ecuación diferencial corresponden a campos vectoriales, por lo que la ecuación diferencial se puede escribir:

$$\dot{x}(t) = X(x(t), \alpha(t)) = X_0(x(t)) + \sum_{i=1}^{m} \alpha_i(t) X_i(x(t)) \tag{4.1}$$

donde X_0 corresponde a la parte de la dinámica en la que no intervienen los controles y recibe el nombre de *drift*, y X_i son los campos de control. La solución de la ecuación diferencial serán las curvas integrales de este campo vectorial. La teoría de control geométrico ([9]) prueba que para que el sistema sea controlable los campos de control tienen que generar mediante el conmutador de campos vectoriales todo el espacio tangente. La forma con la que se conmutan los campos es el corchete de Lie, que actúa de la siguiente forma sobre los campos vectoriales escritos en coordenadas:

$$[X_1, X_2] = \sum_{i=1}^{n} \sum_{j=1}^{n} \left(X_j \frac{\partial Y_i}{\partial x_j} - Y_j \frac{\partial X_i}{\partial x_j} \right) \frac{\partial}{\partial x_i} \quad \text{con } X = \sum_{i=1}^{n} X_i \frac{\partial}{\partial x_i} \text{ y } Y = \sum_{i=1}^{n} Y_i \frac{\partial}{\partial x_i} \tag{4.2}$$

y dota al conjunto de campos de una estructura de álgebra de Lie. Si el álgebra generada por los campos de control cubre todo el espacio tangente en cada punto, el sistema será controlable. Para ver cómo se puede generar el álgebra de Lie a partir de unos campos dados mediante corchetes de Lie veamos un ejemplo con dos campos X_1 y X_2. Primero calculamos $[X_1, X_2] = X_3$. Si este nuevo campo es independiente de los otros dos campos calculamos dos nuevos campos $[X_1, X_3] = X_4$ y $[X_2, X_3] = X_5$ y continuamos calculando nuevos campos mediantes los corchetes hasta que no se generen nuevos campos independientes a los anteriores. A este conjunto lo llamaremos $\text{Lie}(X_1, X_2) = \{X_1, X_2, X_3 \ldots X_p\}$.

4.1. Control óptimo y principio de máximo de Pontriagin

El control óptimo consiste en encontrar las leyes para las funciones de control tales que un cierto criterio de optimización se consigue mediante estas. El criterio de optimización se da con un funcional de coste que depende del estado del sistema y de los controles. La solución al control óptimo da un conjunto de ecuaciones diferenciales para los estados que minimizan este funcional de coste. Hay varios métodos para conseguir el control óptimo, en este caso vamos a tratar el principio de máximo de Pontriagin.

Este principio nos permite obtener las condiciones para los controles dada una cierta función de coste y las ecuaciones diferenciales que seguirán los estados. Para una demostración rigurosa de este puede consultarse [10]. Veamos en qué consiste este método. Primero consideremos un funcional de coste de la siguiente forma:

$$J(u(t)) = \int_{T_0}^{T} L(x(t), u(t))dt \tag{4.3}$$

donde $x(t) \equiv (x_1(t), \ldots, x_n(t))$ son los grados de libertad del sistema y $u(t) \equiv (u_1(t), \ldots, u_m(t))$ son los controles del sistema. Se define el hamiltoniano de Pontriagin de la siguiente forma:

$$\mathcal{H}(x(t), p(t), u(t)) = \sum_{i=1}^{n} p_i(t) f_i(x(t), u(t)) + L(x(t), u(t)) \tag{4.4}$$

con $f_i(x(t), u(t))$ la dinámica de los grados de libertad es decir $\dot{x}_i(t) = f_i(x(t), u(t))$ y los términos $p_1(t), \ldots, p_n(t)$ son unas variables accesorias denominadas coestados (para acortar la notación se usará $p(t) \equiv (p_1(t), \ldots, p_n(t))$. Se observa cómo este hamiltoniano es como el hamiltoniano de la Mecánica Clásica donde $x(t)$ corresponden a las posiciones y $p(t)$ a los momentos. El principio de máximo de Pontriagin nos dice que si escogemos las funciones de control que maximizan el hamiltoniano $\mathcal{H}$ respecto al conjunto de controles admisibles, entonces la solución de las ecuaciones de Hamilton usando estos controles será la que optimice el funcional de coste. La condición de que los controles maximicen el hamiltoniano nos dará unas ecuaciones algebraicas para los controles en función de los estados y coestados. Este principio se puede usar también para minimizar el funcional de coste. En ese caso habrá que encontrar los controles que minimizan el hamiltoniano de Pontriaguin en vez de maximizarlo. Tras obtener las ecuaciones de los controles se puede obtener la dinámica de los estados y coestados aplicando las ecuaciones de Hamilton al hamiltoniano de Pontriagin de la siguiente forma:

$$\dot{x}_i(t) = \frac{\partial \mathcal{H}}{\partial p_i} = f_i(x(t), u'(t)) \qquad \dot{p}_i - = \frac{\partial \mathcal{H}}{\partial x_i} = -\sum_{j=1}^{n} p_j \frac{\partial f_j(x(t), u'(t))}{\partial x_i} - \frac{\partial L(x(t), u'(t))}{\partial x_i} \tag{4.5}$$

donde $u'(t)$ son los controles que maximizan el hamiltoniano, por lo que serán funciones de los estados y momentos es decir $u'(t) \equiv u'(x(t), p(t))$. Las soluciones de estas ecuaciones diferenciales corresponden a la solución óptima del problema de control para el funcional de coste elegido.

Ahora apliquemos este método al sistema híbrido definido en el apartado anterior mediante las ecuaciones de Ehrenfest. Hay que tener en cuenta que los estados serán las posiciones clásicas, momentos clásicos y las coordenadas reales que representan M_Q de los estados cuánticos. Por ello

a los coestados los vamos a denominar Π en vez de p, para no confundirlos con los momentos. Otra consideración es que los controles se van a aplicar solo sobre el hamiltoniano cuántico. Teniendo esto en cuenta el hamiltoniano de Pontriagin que se obtiene para un sistema con n partículas clásicas en tres dimensiones y un espacio de Hilbert para la parte cuántica de dimensión m, es:

$$\mathcal{H}(R,P,q,p,\Pi,u)=\sum_{i=1}^{n}\left(\vec{\Pi}_{Ri}\frac{\vec{P}_i}{M_i}-\vec{\Pi}_{Pi}\nabla_i f_{He}\right)+\sum_{i=1}^{m}\left(\Pi_{qi}\frac{\partial f_{He}}{\partial p_i}-\Pi_{pi}\frac{\partial f_{He}}{\partial q_i}\right)+L(R,P,q,p,u) \tag{4.6}$$

En esta expresión hemos usado unidades naturales ($\hbar=1$) y las ecuaciones de la dinámica (3.21),(3.22),(3.23) y (3.24). La función f_{He} es $f_{He}(R,q,p,u)=\langle\phi(q,p)|H_e(R,u)|\phi(q,p)\rangle$. Para reducir la notación se ha omitido la dependencia temporal de todas las variables. Para los coestados se ha usado la siguiente notación: $\Pi_{Ri,j}$ corresponde a la componente j del coestado correspondiente a la posición de la partícula i, $\Pi_{Pi,j}$ corresponde a la componente j del coestado correspondiente al momento de la partícula i y Π_{xi} corresponde al coestado de la componente x_i de la parte cuántica con x siendo q o p.

Una vez que tenemos el hamiltoniano de Pontriagin primero hay que calcular los valores de los controles que optimizan el funcional de coste. Para esto, en el caso de que el dominio de los controles sea adecuado, los valores de los controles serán los que anulan las derivadas, es decir $\frac{\partial\mathcal{H}}{\partial u_i}=0$. Así en el caso de haber r controles se obtendrían r ecuaciones con r controles a obtener en función de los estados y coestados. Como estas ecuaciones no tienen por qué ser lineales puede que no tengan una solución analítica. Una vez obtenidas las funciones de control que son solución de estas ecuaciones habría que comprobar que son máximos ya que, la condición de que se anule la derivada, solo nos dice que son extremos locales o puntos silla.

Imponiendo la condición de que la derivada de los controles se anule y calculando las ecuaciones de Hamilton se obtienen las siguientes ecuaciones:

$$\frac{\partial\mathcal{H}}{\partial u_i}=0\Rightarrow\sum_{j=1}^{m}\left(\Pi_{qj}\frac{\partial^2 f_{He}}{\partial u_i\partial p_j}-\Pi_{pj}\frac{\partial^2 f_{He}}{\partial u_i\partial q_j}\right)-\sum_{j=1}^{n}\Pi_{Pj}\frac{\partial(\vec{\nabla}_j f_{He})}{\partial u_i}+\frac{\partial L}{\partial u_i}=0\quad i=1,\dots,r \tag{4.7}$$

$$\dot{\vec{R}}_i=\frac{\vec{P}_i}{M_i}\qquad\dot{\vec{P}}_i=-\nabla_i f_{He}\qquad\dot{\vec{\Pi}}_{Pi}=-\frac{\vec{\Pi}_{Ri}}{M_i}-\nabla_{Pi}L\quad i=1,\dots,n \tag{4.8}$$

$$\dot{\vec{\Pi}}_{Ri}=\sum_{j=1}^{n}\vec{\Pi}_{Pj}\nabla_i\nabla_j f_{He}-\sum_{j=1}^{m}\left(\Pi_{qj}\nabla_i\frac{\partial f_{He}}{\partial p_j}-\Pi_{pj}\nabla_i\frac{\partial f_{He}}{\partial q_j}\right)-\nabla_i L\quad i=1,\dots,n \tag{4.9}$$

$$\dot{q}_i=\frac{\partial f_{He}}{\partial p_i}\qquad\dot{p}_i=-\frac{\partial f_{He}}{\partial q_i}\quad i=1,\dots,m \tag{4.10}$$

$$\dot{\Pi}_{qi}=\sum_{j=1}^{n}\vec{\Pi}_{Pj}\frac{\partial(\nabla_j f_{He})}{\partial q_i}-\sum_{j=1}^{m}\left(\Pi_{qj}\frac{\partial^2 f_{He}}{\partial q_i\partial p_j}-\Pi_{pj}\frac{\partial^2 f_{He}}{\partial q_i\partial q_j}\right)-\frac{\partial L}{\partial q_i}\quad i=1,\dots,m \tag{4.11}$$

$$\dot{\Pi}_{pi}=\sum_{j=1}^{n}\vec{\Pi}_{Pj}\frac{\partial(\nabla_j f_{He})}{\partial p_i}-\sum_{j=1}^{m}\left(\Pi_{qj}\frac{\partial^2 f_{He}}{\partial p_i\partial p_j}-\Pi_{pj}\frac{\partial^2 f_{He}}{\partial p_i\partial q_j}\right)-\frac{\partial L}{\partial p_i}\quad i=1,\dots,m \tag{4.12}$$

Con las r ecuaciones de (4.7) se obtienen las funciones de control $u'(t)\equiv u'(R,P,q,p,\Pi_R,\Pi_P,\Pi_q,\Pi_p)$. Resolviendo las demás ecuaciones diferenciales se podría obtener la evolución temporal de R, P, q y p.

5. Ejemplo

Dado que las potenciales aplicaciones a reacciones químicas resultan muy complejas, como ejemplo de un sistema híbrido se va a tomar un oscilador armónico en dos dimensiones para la parte clásica y un sistema de dos niveles para la parte cuántica. Sobre la parte cuántica se va a aplicar un campo magnético en dos direcciones que servirá como control del sistema. El objetivo va a ser conseguir trayectorias con las que se minimice la energía mientras pasan por ciertos puntos del espacio de estados clásicos y cuánticos. Para conseguirlo se van a variar las condiciones iniciales de los coestados que se obtienen usando el método de Pontriaguin para el control óptimo. Se va a usar el formalismo geométrico para sistemas híbridos de la sección 3 y el método de control óptimo de la sección 4.

5.1. Dinámica

Primero veamos las ecuaciones del movimiento. La parte clásica es la de un oscilador armónico habitual es decir

$$H_C = \frac{P_x^2 + P_y^2}{2m} + \frac{1}{2}k(R_x^2 + R_y^2) \tag{5.1}$$

Donde P_x y P_y es el momento, R_x y R_y la posición, m la masa de la partícula clásica y k la constante del oscilador armónico.
El hamiltoniano de la parte cuántica será:

$$H_Q = f(R_x, R_y)\left(B_x(t)\sigma_x + B_y(t)\sigma_y\right) \tag{5.2}$$

donde $f(R_x, R_y)$ es el acoplo entre la parte clásica y cuántica (que para este ejemplo se va a tomar como $f = \epsilon(R_x^2 + R_y^2)$ por simplicidad, siendo ϵ es una variable que controla la intensidad del acoplamiento), $B_x(t)$ y $B_y(t)$ son las coordenadas del campo magnético que se va a aplicar que depende del tiempo, σ_x y σ_y son las matrices de Pauli. Usando las coordenadas reales para este sistema cuántico $\{q_1, q_2, p_1, p_2\}$, el hamiltoniano total del sistema es :

$$f_H = \frac{P_x^2 + P_y^2}{2M} + \frac{1}{2}k(R_x^2 + R_y^2) + \epsilon(R_x^2 + R_y^2)\left(B_x(p_1p_2 + q_1q_2) + B_y(q_1p_2 - p_1q_2)\right) \tag{5.3}$$

El valor medio del hamiltoniano cuántico ya se ha calculado en (2.17) pero sin el acoplo. A partir de este momento no se va a escribir la dependencia temporal de las variables y para la posición y momento se va a usar $R_x^2 + R_y^2 \equiv R^2$ y $P_x^2 + P_y^2 \equiv P^2$ para reducir la notación. Con este hamiltoniano se pueden calcular las ecuaciones del movimiento como se ha visto anteriormente en las ecuaciones (3.11),(3.12) y (3.13). Se obtiene:

$$\dot{R}_x = \frac{P_x}{m} \qquad \dot{R}_y = \frac{P_y}{m} \tag{5.4}$$

$$\dot{P}_x = -kR_x - 2\epsilon R_x\left(B_x(p_1p_2 + q_1q_2) + B_y(q_1p_2 - p_1q_2)\right) \tag{5.5}$$

$$\dot{P}_y = -kR_y - 2\epsilon R_y\left(B_x(p_1p_2 + q_1q_2) + B_y(q_1p_2 - p_1q_2)\right) \tag{5.6}$$

$$\dot{q}_1 = \epsilon R^2(B_x p_2 - B_y q_2) \qquad \dot{q}_2 = \epsilon R^2(B_x p_1 + B_y q_1) \tag{5.7}$$

$$\dot{p}_1 = -\epsilon R^2(B_x q_2 + B_y p_2) \qquad \dot{p}_2 = -\epsilon R^2(B_x q_1 - B_y p_1) \tag{5.8}$$

Una vez que ya tenemos las ecuaciones de la dinámica pasemos a obtener las ecuaciones para

el control óptimo. Como queremos minimizar la energía que se suministra al sistema mediante el campo magnético, el funcional de coste que se define es:

$$J(B_x(t), B_y(t)) = \int_{T_0}^{T} (B_x^2(t) + B_y^2(t))dt \tag{5.9}$$

Con este funcional de coste y las ecuaciones de la dinámica anteriores se obtiene el siguiente hamiltoniano de Pontriaguin:

$$\begin{aligned}\mathcal{H} = &\frac{P_x\Pi_{Rx}}{m} + \frac{P_y\Pi_{Ry}}{m} + \Pi_{Px}\left(-2R_x\epsilon\left(B_x\left(p_1p_2 + q_1q_2\right) + B_y\left(-p_1q_2 + p_2q_1\right)\right) - R_xk\right)\\ &+ \Pi_{Py}\left(-2R_y\epsilon\left(B_x\left(p_1p_2 + q_1q_2\right) + B_y\left(-p_1q_2 + p_2q_1\right)\right) - R_yk\right) - \Pi_{p1}\epsilon R^2\left(B_xq_2 + B_yp_2\right)\\ &- \Pi_{p2}\epsilon R^2\left(B_xq_1 - B_yp_1\right) + \Pi_{q1}\epsilon R^2\left(B_xp_2 - B_yq_2\right) + \Pi_{q2}\epsilon R^2\left(B_xp_1 + B_yq_1\right) - \left(B_x^2 + B_y^2\right)\end{aligned} \tag{5.10}$$

donde Π_i son los coestados. Notar que se ha añadido el signo menos al funcional al añadirlo al hamiltoniano para que haya que maximizar el hamiltoniano para minizar la energía. Las ecuaciones para la evolución de los coestados se pueden hallar a partir de este hamiltoniano con las ecuaciones (4.8),(4.9),(4.11) y (4.12), se obtiene:

$$\begin{aligned}\dot{\Pi}_{Rx} = &2R_x\Pi_{p1}\epsilon\left(B_xq_2 + B_yp_2\right) + 2R_x\Pi_{p2}\epsilon\left(B_xq_1 - B_yp_1\right) - 2R_x\Pi_{q1}\epsilon\left(B_xp_2 - B_yq_2\right)\\ &- 2R_x\Pi_{q2}\epsilon\left(B_xp_1 + B_yq_1\right) - \Pi_{Px}\left(-2\epsilon\left(B_x\left(p_1p_2 + q_1q_2\right) + B_y\left(-p_1q_2 + p_2q_1\right)\right) - k\right)\end{aligned} \tag{5.11}$$

$$\begin{aligned}\dot{\Pi}_{Ry} = &2R_y\Pi_{p1}\epsilon\left(B_xq_2 + B_yp_2\right) + 2R_y\Pi_{p2}\epsilon\left(B_xq_1 - B_yp_1\right) - 2R_y\Pi_{q1}\epsilon\left(B_xp_2 - B_yq_2\right)\\ &- 2R_y\Pi_{q2}\epsilon\left(B_xp_1 + B_yq_1\right) - \Pi_{Py}\left(-2\epsilon\left(B_x\left(p_1p_2 + q_1q_2\right) + B_y\left(-p_1q_2 + p_2q_1\right)\right) - k\right)\end{aligned} \tag{5.12}$$

$$\dot{\Pi}_{Px} = -\frac{\Pi_{Rx}}{m} \qquad \dot{\Pi}_{Py} = -\frac{\Pi_{Ry}}{m} \tag{5.13}$$

$$\dot{\Pi}_{q1} = B_x\Pi_{p2}\epsilon R^2 - B_y\Pi_{q2}\epsilon R^2 + 2R_x\Pi_{Px}\epsilon\left(B_xq_2 + B_yp_2\right) + 2R_y\Pi_{Py}\epsilon\left(B_xq_2 + B_yp_2\right) \tag{5.14}$$

$$\dot{\Pi}_{q2} = B_x\Pi_{p1}\epsilon R^2 + B_y\Pi_{q1}\epsilon R^2 + 2R_x\Pi_{Px}\epsilon\left(B_xq_1 - B_yp_1\right) + 2R_y\Pi_{Py}\epsilon\left(B_xq_1 - B_yp_1\right) \tag{5.15}$$

$$\dot{\Pi}_{p1} = -B_x\Pi_{q2}\epsilon R^2 - B_y\Pi_{p2}\epsilon R^2 + 2R_x\Pi_{Px}\epsilon\left(B_xp_2 - B_yq_2\right) + 2R_y\Pi_{Py}\epsilon\left(B_xp_2 - B_yq_2\right) \tag{5.16}$$

$$\dot{\Pi}_{p2} = -B_x\Pi_{q1}\epsilon R^2 + B_y\Pi_{p1}\epsilon R^2 + 2R_x\Pi_{Px}\epsilon\left(B_xp_1 + B_yq_1\right) + 2R_y\Pi_{Py}\epsilon\left(B_xp_1 + B_yq_1\right) \tag{5.17}$$

Ahora calculemos la condición que tienen que cumplir los campos magnéticos para que se maximice el hamiltoniano. En este caso se puede aplicar la condición de que la derivada sea nula y al ser la dependencia con los campos magnéticos lineal y cuadrática se va a obtener una única solución que será el máximo.

Se obtienen los siguiente valores para los campos magnéticos aplicando la ecuación (4.7):

$$\begin{aligned}B_x = &-R_x\Pi_{Px}\epsilon\left(p_1p_2 + q_1q_2\right) - R_y\Pi_{Py}\epsilon\left(p_1p_2 + q_1q_2\right)\\ &- \frac{\Pi_{p1}\epsilon}{2}q_2R^2 - \frac{\Pi_{p2}\epsilon}{2}q_1R^2 + \frac{\Pi_{q1}\epsilon}{2}p_2R^2 + \frac{\Pi_{q2}\epsilon}{2}p_1R^2\end{aligned} \tag{5.18}$$

$$\begin{aligned}B_y = &-R_x\Pi_{Px}\epsilon\left(-p_1q_2 + p_2q_1\right) - R_y\Pi_{Py}\epsilon\left(-p_1q_2 + p_2q_1\right)\\ &- \frac{\Pi_{p1}\epsilon}{2}p_2R^2 + \frac{\Pi_{p2}\epsilon}{2}p_1R^2 - \frac{\Pi_{q1}\epsilon}{2}q_2R^2 + \frac{\Pi_{q2}\epsilon}{2}q_1R^2\end{aligned} \tag{5.19}$$

Las ecuaciones que van a determinar la dinámica optimizada son las 8 ecuaciones diferenciales para los estados y las 8 para los coestados sustituyendo los campos magnéticos en estas por las relaciones (5.18) y (5.19). Este conjunto de ecuaciones habrá que resolverlo de forma numérica.

5.2. Dinámica con matrices densidad

Ahora se van a realizar los mismo cálculos que en el apartado anterior pero usando matrices densidad para la parte cuántica. Como se ha mencionado anteriormente, si solo se usan estados puros, usar matrices densidad es equivalente al proyectivo (al existir un difeomorfismo entre ambos). Las ecuaciones de Ehrenfest con matrices densidad se han obtenido en (3.25),(3.26) y (3.27).

Vamos a centrarnos en el caso que se está tratando de un sistemas de 2 niveles. Las matrices densidad serán matrices 2×2 hermíticas, por lo que puede escogerse una base de la forma: $\{\mathbb{I}, \sigma_x, \sigma_y, \sigma_z\}$. Como las matrices densidad tienen que tener traza unidad y ser hermíticas se pueden escribir en esta base con coordenadas reales de la siguiente forma:

$$\begin{pmatrix} \frac{1}{2}+y_3 & y_1+iy_2 \\ y_1-iy_2 & \frac{1}{2}-y_3 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + y_1\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + y_2\begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix} + y_3\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{5.20}$$

Por lo que las matrices densidad solo dependen de las coordendas en las matrices de Pauli, ya que la coordenada correspondiente a la identidad esta fijada. Debido a esto se pueden usar solo tres coordenadas. A estas coordenadas las denominaremos de la siguiente forma: $\rho_x \equiv y_1$, $\rho_y \equiv y_2$ y $\rho_z \equiv y_3$. Escribiendo la ecuación de von Neumann en estas coordenadas para el hamiltoniano de la parte cuántica del apartado anterior se obtienen las siguientes ecuaciones:

$$\dot{\rho}_x = \epsilon R^2 B_y \rho_z \qquad \dot{\rho}_y = -\epsilon R^2 B_x \rho_z \qquad \dot{\rho}_z = \epsilon R^2 (B_x \rho_y - B_y \rho_x) \tag{5.21}$$

Estas nos dan la evolución temporal de las coordenadas y coinciden con la expresión (3.27) si se desarrolla para este caso particular. El valor medio del hamiltoniano cuántico en estas coordenadas es:

$$\langle H_e \rangle = Tr(\rho H_e) = \epsilon R^2 (B_x \rho_x + B_y \rho_y) \tag{5.22}$$

Una vez que tenemos estos tres elementos (coordenadas reales, evolución temporal y el valor medio) podemos pasar al sistema híbrido. Con este valor medio se puede calcular la evolución de las coordenadas clásicas usando las ecuaciones (3.21) y (3.22). Se obtiene:

$$\dot{R}_x = \frac{P_x}{m} \qquad \dot{R}_y = \frac{P_y}{m} \tag{5.23}$$

$$\dot{P}_x = -kR_x - 2\epsilon R_x \left(B_x \rho_x + B_y \rho_y\right) \qquad \dot{P}_y = -kR_y - 2\epsilon R_y \left(B_x \rho_x + B_y \rho_y\right) \tag{5.24}$$

Como en el apartado anterior una vez obtenidas las ecuaciones del movimiento del sistema completo, pasamos a calcular el control óptimo. El cálculo es igual pero con las nuevas ecuaciones del movimiento y las nuevas coordenadas cuánticas. El funcional de coste es el mismo por lo que el hamiltoniano de Pontriaguin es:

$$\begin{aligned} \mathcal{H} = {} & \frac{P_x \Pi_{Ry}}{m} + \frac{P_y \Pi_{Ry}}{m} + \Pi_{Px}\left(-2R_x\epsilon\left(B_x\rho_x + B_y\rho_y\right) - R_x k\right) + \Pi_{Py}\left(-2R_y\epsilon\left(B_x\rho_x + B_y\rho_y\right) - R_y k\right) \\ & + \Pi_{\rho x}\epsilon R^2 B_y \rho_z - \Pi_{\rho y}\epsilon R^2 B_x \rho_z + \Pi_{\rho z}\epsilon R^2\left(B_x\rho_y - B_y\rho_x\right) - \left(B_x^2 + B_y^2\right) \end{aligned} \tag{5.25}$$

Como anteriormente, a partir de este hamiltoniano se obtienen las siguientes ecuaciones para la evolución de los coestados Π_i:

$$\dot{\Pi}_{R_x} = 2\Pi_{\rho y}\epsilon R_x B_x \rho_z - 2\Pi_{\rho x}\epsilon R_x B_y \rho_z - 2\Pi_{\rho z}\epsilon R_x \left(B_x\rho_y - B_y\rho_x\right) + \Pi_{Px}\left(2\epsilon\left(B_x\rho_x + B_y\rho_y\right) + k\right) \tag{5.26}$$

$$\dot{\Pi}_{R_y} = 2\Pi_{\rho y}\epsilon R_y B_x \rho_z - 2\Pi_{\rho x}\epsilon R_y B_y \rho_z - 2\Pi_{\rho z}\epsilon R_y \left(B_x\rho_y - B_y\rho_x\right) + \Pi_{Py}\left(2\epsilon\left(B_x\rho_x + B_y\rho_y\right) + k\right) \tag{5.27}$$

$$\dot{\Pi}_{P_x} = -\frac{\Pi_{R_x}}{m} \qquad \dot{\Pi}_{P_y} = -\frac{\Pi_{R_y}}{m} \tag{5.28}$$

$$\dot{\Pi}_{\rho_x} = 2B_x R_x \Pi_{Px}\epsilon + 2B_x R_y \Pi_{Py}\epsilon + B_y \Pi_{\rho z}\epsilon R^2 \tag{5.29}$$

$$\dot{\Pi}_{\rho_y} = -B_x \Pi_{\rho z}\epsilon R^2 + 2B_y R_x \Pi_{Px}\epsilon + 2B_y R_y \Pi_{Py}\epsilon \tag{5.30}$$

$$\dot{\Pi}_{\rho_z} = B_x \Pi_{\rho y}\epsilon R^2 - B_y \Pi_{\rho x}\epsilon R^2 \tag{5.31}$$

Las ecuaciones para que los campos magnéticos optimicen la energía son:

$$B_x = -R_x\Pi_{Px}\epsilon\rho_x - R_y\Pi_{Py}\epsilon\rho_x - \frac{\Pi_{\rho y}\epsilon}{2}\rho_z R^2 + \frac{\Pi_{\rho z}\epsilon}{2}\rho_y R^2 \tag{5.32}$$

$$B_y = -R_x\Pi_{Px}\epsilon\rho_y - R_y\Pi_{Py}\epsilon\rho_y + \frac{\Pi_{\rho x}\epsilon}{2}\rho_z R^2 - \frac{\Pi_{\rho z}\epsilon}{2}\rho_x R^2 \tag{5.33}$$

Así ya tenemos las ecuaciones diferenciales para los estados y coestados (que en este caso son 14 ecuaciones) y los valores de los campos magnéticos con los que hay que resolverlas para el control óptimo. Como en el caso anterior estas ecuaciones habrá que resolverlas numéricamente.

5.3. Controlabilidad

Para comprobar si el sistema es controlable se va a usar un programa en Python (se muestra en el Anexo B) con el se van a calcular los campos que se obtienen a partir de los corchetes de Lie entre los campos vectoriales. Si el número de campos linealmente independientes que se pueden generar coincide con la dimensión del espacio tangente entonces el sistema será controlable.

Los campos vectoriales vienen dados por la dinámica por la siguiente ecuación:

$$\dot{x}(t) = X_0(x(t)) + B_x X_x(x(t)) + B_y X_y(x(t)) \tag{5.34}$$

Ahora veamos que forma tienen esos campos vectoriales en función de si se escoge el formalismo de Schrödinger o el de la matriz densidad. Para el formalismo de Schrödinger el espacio tangente tendrá dimensión 8 y escogemos la siguiente base $(\frac{\partial}{\partial R_x}, \frac{\partial}{\partial R_y}, \frac{\partial}{\partial P_x}, \frac{\partial}{\partial P_y}, \frac{\partial}{\partial q_1}, \frac{\partial}{\partial q_2}, \frac{\partial}{\partial p_1}, \frac{\partial}{\partial p_2})$. Usando las ecuaciones (5.4),(5.5),(5.6),(5.7) y (5.8), y multiplicando por la norma de la parte cuántica $(q_1^2 + q_2^2 + p_1^2 + p_2^2)$ en la parte clásica del hamiltoniano se puede deducir que los campos tendrán la siguiente forma en la base anterior:

$$X_0 = \left(\frac{P_x}{m}, \frac{P_y}{m}, -kR_x, -kR_y, \left(\frac{P^2}{m} + kR^2\right)p_1, \left(\frac{P^2}{m} + kR^2\right)p_2, -\left(\frac{P^2}{m} + kR^2\right)q_1, -\left(\frac{P^2}{m} + kR^2\right)q_2\right) \tag{5.35}$$

$$X_x = \left(0, 0, -2\epsilon R_x(p_1p_2 + q_1q_2), -2\epsilon R_y(p_1p_2 + q_1q_2), \epsilon R^2 p_2, \epsilon R^2 p_1, -\epsilon R^2 q_2, -\epsilon R^2 q_1\right) \tag{5.36}$$

$$X_y = \left(0, 0, -2\epsilon R_x(q_1p_2 - p_1q_2), -2\epsilon R_y(q_1p_2 - p_1q_2), -\epsilon R^2 q_2, \epsilon R^2 q_1, -\epsilon R^2 p_2, \epsilon R^2 p_1\right) \tag{5.37}$$

Al introducir estos campos en el programa se observa que se pueden generar 7 campos independientes por lo que el sistema es accesible. Hay que tener en cuenta que son 7 y no 8 porque al ser una dinámica hamiltoniana conserva la norma de los estados cuánticos, por lo que hay un grado de libertad menos. El término que de la norma cuántica se ha añadido porque es necesario para generar la dirección de movimiento de la identidad en la parte cuántica. En las demás partes del trabajo no se tiene en cuenta porque al trabajar con estados normalizados este término es uno.

Para el caso de usar matrices densidad la base escogida es $(\frac{\partial}{\partial R_x}, \frac{\partial}{\partial R_y}, \frac{\partial}{\partial P_x}, \frac{\partial}{\partial P_y}, \frac{\partial}{\partial \rho_x}, \frac{\partial}{\partial \rho_y}, \frac{\partial}{\partial \rho_z},)$. Como anteriormente usando las ecuaciones (5.21),(5.23) y (5.24) se puede deducir que los campos tendrán la siguiente forma en la base anterior:

$$X_0 = \left(\frac{P_x}{m}, \frac{P_y}{m}, -kR_x, -kR_y, 0, 0, 0\right) \tag{5.38}$$

$$X_x = \left(0, 0, -2\epsilon R_x\rho_x, -2\epsilon R_y\rho_x, 0, -\epsilon R^2\rho_z, \epsilon R^2\rho_y\right) \tag{5.39}$$

$$X_y = \left(0, 0, -2\epsilon R_x\rho_y, -2\epsilon R_y\rho_y, -\epsilon R^2\rho_z, 0, -\epsilon R^2\rho_x\right) \tag{5.40}$$

En este caso al usar estos campos se consiguen generar 6 campos independientes por lo que el sistema es accesible. Por la misma razón que en el caso anterior hay un grado de libertad menos que dimensiones.

Como se ha usado el *drift* para generar todo el espacio tangente se ha comprobado que el sistema es accesible para todos los puntos de la variedad pero no tiene por que ser controlable (para esa condición solo habría que usar los campos de control para generar el espacio tangente). En este caso particular como todas las componentes del *drift* pueden cambiar de signo no se va a dar el caso de que una cierta componente de la velocidad no se pueda cambiar de dirección que es lo que hace que un sistema accesible no sea controlable. Debido a esto nos es suficiente con comprobar la accesiblidad del sistema.

5.4. Resolución numérica del problema

Para la resolución numérica del problema se han usado librerías de código abierto de Python (*Sympy* y *SciPy*). Primero hay que resolver el conjunto de ecuaciones diferenciales de estados y coestados, y posteriormente hay que determinar las condiciones iniciales de los coestados para realizar la trayectoria deseada.

Para integrar el conjunto de ecuaciones diferenciales se ha usado un método de Runge-Kutta explícito de orden 8 disponible en la función *ode* de la librería *Scipy*. Se ha comprobado que durante los tiempos de integración que se han usado para la resolución del problema los estados cuánticos no perdían su normalización (se han obtenido variaciones en la norma menores que 10^{-9}) por lo que se ha considerado un método adecuado.

Para resolver el problema de que la trayectoria pase por un determinado punto en un determinado tiempo variando las condiciones iniciales, lo que se ha hecho es definir una función que será la distancia mínima de la trayectoria (que se integrará un tiempo T) a ese punto y que dependerá de las condiciones iniciales, es decir:

$$f(X_0) = \min\left(\sqrt{\sum_i (X_i(t, X_0) - X_i^{\text{obj}})^2}\right) \tag{5.41}$$

donde la suma en i es a las coordenadas que se quiere aproximar la trayectoria y X^{obj} es el punto a dónde se quiere llegar. Si se quiere pasar por más de un punto se pueden sumar estas funciones, una por cada punto por el que se quiere hacer pasar la trayectoria. El objetivo es minimizar esta función.

Hay que tener en cuenta que para calcular el valor de la función para un determinado valor de las condiciones iniciales de los coestados hay que integrar toda la trayectoria hasta el tiempo T y obtener la distancia mínima al punto en cuestión. Debido a esto no hay expresión análitica para esta función, por lo que se va a realizar una minimización de esta de forma numérica. Para ello se van a usar el método de minimización global *basinhopping* y el método de minimización local *Nelder-Mead*, ambos disponibles en la libreria *Scipy*.

Una vez que se ha obtenido un valor que se considere adecuado para esta función ya se tiene la trayectoria que minimiza la energía que se le suministra (al haber aplicado control óptimo) y que pasa por los puntos deseados al controlar los condiciones iniciales de los coestados. El código que se ha usado para la resolución del problema y la representación gráfica de los resultados se muestra en el Anexo C.

5.5. Resultados

Se van a calcular tres casos distintos usando las dos formas explicadas anteriormente. Se van a escoger los siguientes valores para los parámetros: $k = 1$, $m = 1$ y $\epsilon = 10$. El valor de ϵ regula la intensidad del acoplo entre la parte cuántica y clásica. Se ha escogido un valor para este tal que en la optimización numérica no se necesiten valores muy grandes o pequeños de los coestados para facilitar la convergencia de la integración numérica. Se van a representar las trayectorias obtenidas en el espacio de fases clásico (posición frente al momento), la trayectoria de la parte cuántica en la esfera de Bloch (que consiste en representar los estados cuánticos como matrices densidad en la base que se ha usado en el cálculo de la dinámica) y el valor de la energía del campo magnético que suministramos ($B_x^2 + B_y^2$). En las representaciones para referirse al caso en el que se usa la imagen de Schrödinger se indicara en la leyenda qp y en el caso de la matriz densidad se indicará con ρ. En las trayectorias que se van a mostrar se han obtenidos distancias (a los puntos a los que se quiere llegar) menores que 10^{-6}, siendo la distancia el valor definido en (5.41), que en función de cada caso tiene en cuenta o no las coordenadas cuánticas.

Primero se van a unir dos puntos en las coordenadas clásicas, es decir que solo se va a tener en cuenta R_x, R_y, P_x y P_y. Se va a iniciar la trayectoria en el punto $(0, 0, 1, 1)$ y se quiere llegar al punto $(1, 1, 0, 0)$. Se han obtenido las siguientes trayectorias:

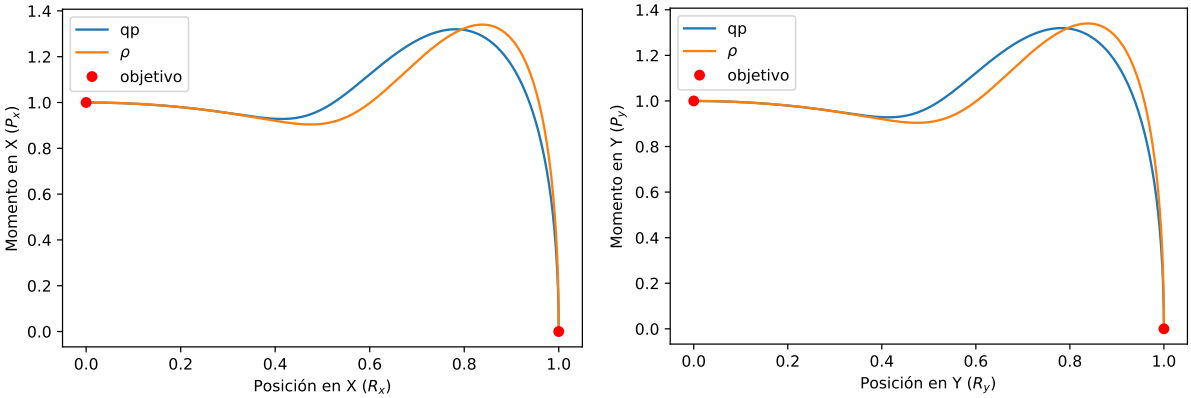

Figura 1: Trayectoria en el espacio de fases clásico uniendo dos puntos clásicos

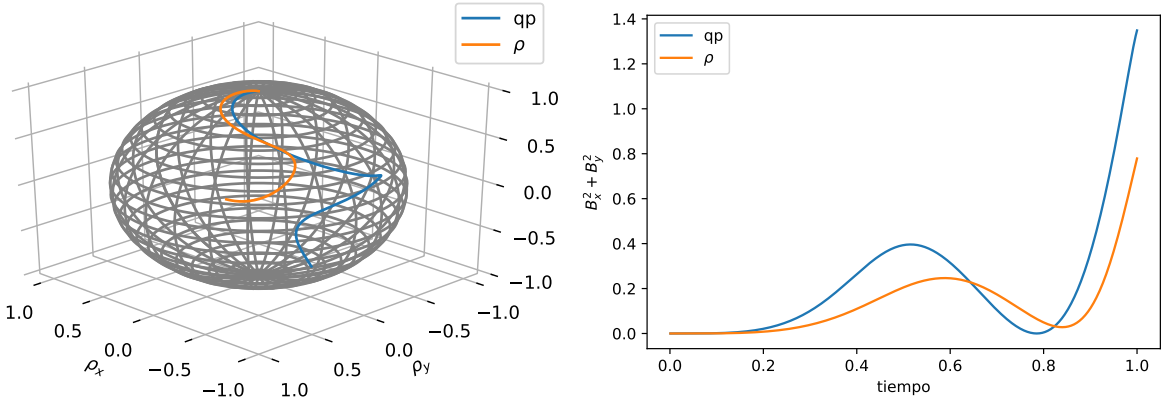

Figura 2: Trayectoria en la esfera de Bloch y energía del campo magnético en cada instante de tiempo uniendo dos puntos clásicos

Se observa como la trayectoria de la parte clásica es similar en los extremos pero distinta en la parte central. También se aprecia como en el caso de qp se necesita una energía mayor.

Ahora se van a unir dos puntos pero en las coordenadas clásicas y cuánticas. Para la parte clásica se va a iniciar en el punto $(0, 0, 1, 1)$ y se va finalizar en el punto $(0{,}75, 0{,}75, 0{,}25, 0{,}25)$. Para la parte cuántica hay que tener en cuenta que hay distintos puntos en las coordenadas qp que equivalen al mismo en las coordenadas ρ. En las coordenadas ρ se va a iniciar en el punto $(0, 0, -1)$ y se va a finalizar en el punto $(0, 0, 1)$, y en las coordenadas qp se va a iniciar $(0, 1, 0, 0) \equiv (0, 0, -1)$ y se va finalizar en el punto $(0, 0, 1, 0) \equiv (0, 0, 1)$, por lo que en la representación en la esfera de Bloch los puntos iniciales y finales son iguales para qp y ρ. Se obtienen las siguiente trayectorias:

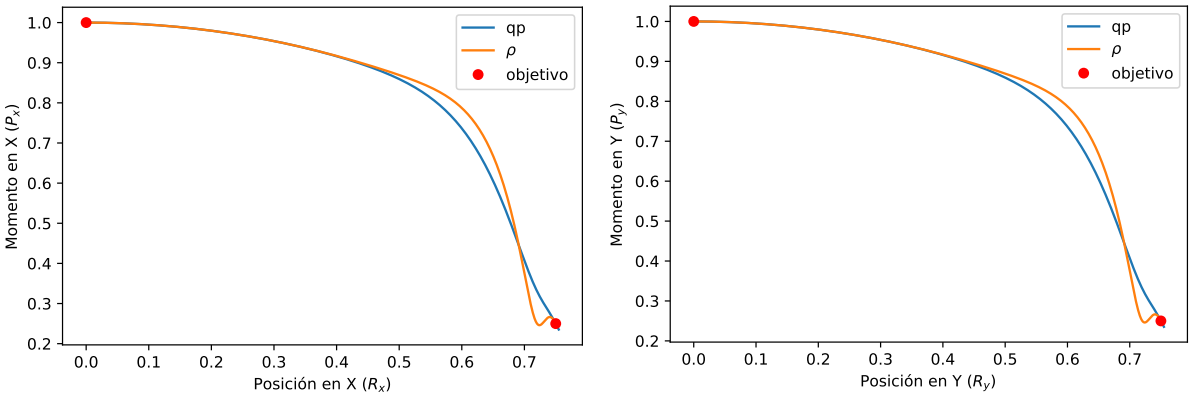

Figura 3: Trayectoria en el espacio de fases clásico uniendo dos puntos híbridos

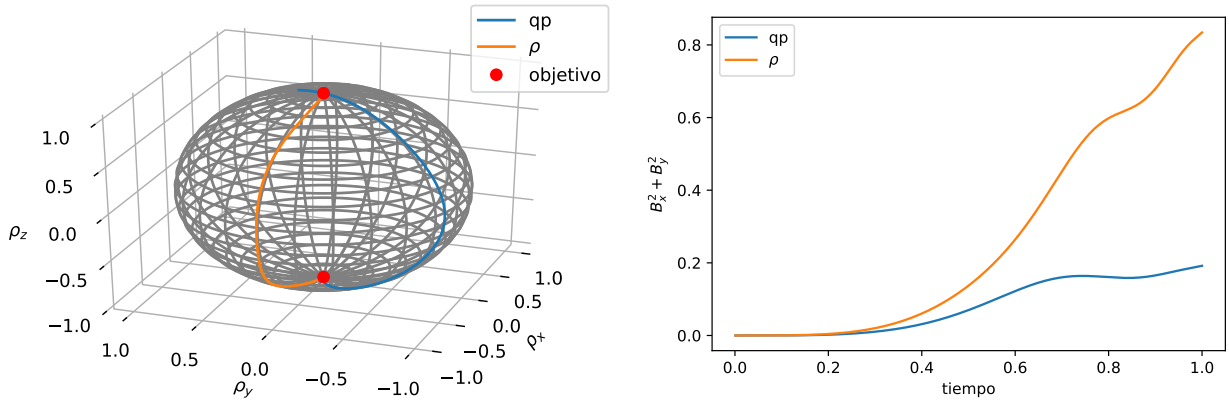

Figura 4: Trayectoria en la esfera de Bloch y energía del campo magnético en cada instante de tiempo uniendo dos puntos híbridos

En este caso observamos como también la trayectoria en la parte clásica es similar pero al final se diferencia claramente. En la parte cuántica, aunque empiecen y acaben en los mismos puntos, las trayectorias son muy distintas. En este caso, a diferencia del anterior se necesita más energía en el caso de ρ. Esto puede que se deba a que la trayectoria en qp es más similar a la del oscilador armónico, por lo que se necesita menos energía para corregirla.

Finalmente se van a unir tres puntos pero solamente en las coordenadas clásicas. Se va a iniciar en el punto $(0, 0, 1, 1)$, se va a continuar por el punto $(0{,}5, 0{,}5, 0{,}5, 0{,}5)$ y se va a finalizar en el punto $(0{,}25, 0{,}25, 0{,}75, 0{,}75)$. Para este caso solo se han conseguido resultados satisfactorios en el caso de qp, por lo que no se van a mostrar resultados para ρ. Se obtiene la siguiente trayectoria:

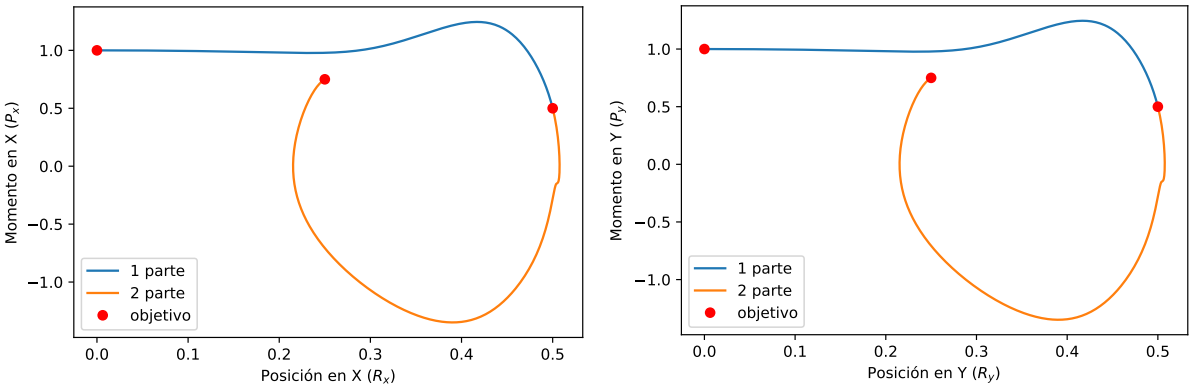

Figura 5: Trayectoria en el espacio de fases clásico uniendo tres puntos clásicos

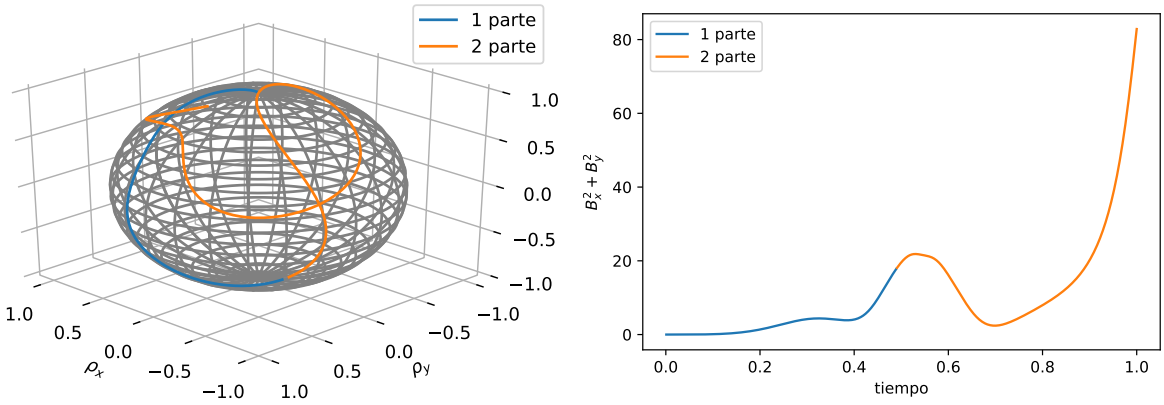

Figura 6: Trayectoria en la esfera de Bloch y energía del campo magnético en cada instante de tiempo uniendo tres puntos clásicos

Se observa cómo en este caso la energía que hay que suministrar es un orden de magnitud mayor que en los casos anteriores. Esto se debe principalmente a que en este caso al principio la trayectoria es opuesta a la del oscilador armónico, por lo que se debe aportar suficiente energía para que la fuerza que ejerce el campo magnético sea mucho mayor que la del oscilador. Que la energía es mayor también se puede observar en la representación de la esfera de Bloch, ya que, en este caso, se recorre mayor distancia en el mismo tiempo. Ello se debe a que el campo magnético es más intenso, por lo que produce variaciones más rápidas entre los dos niveles de energía.

6. Conclusiones

Se ha visto cómo se puede geometrizar con todos los elementos necesarios la Mecánica Cuántica. Esto nos ha permitido usar el formalismo geométrico para sistemas híbridos y, en particular, para las ecuaciones de Ehrenfest. Al geometrizarlas se han podido utilizar los teoremas de controlalibidad y control óptimo de sistemas clásicos en sistemas descritos mediante las ecuaciones de Ehrenfest. Finalmente se ha aplicado todo lo anterior a un ejemplo de sistema híbrido sencillo. En este se ha visto cómo se puede controlar el sistema minimizando la energía necesaria para controlarlo y las diferencias entre tratar la parte cuántica en el espacio de Hilbert o en el proyectivo mediante matrices densidad.

En la resolución numérica se ha conseguido controlar con bastante precisión la parte clásica en tres casos distintos, lo cual en una de las aplicaciones de la ecuaciones de Ehrenfest (la cinética de las reacciones químicas) es lo más relevante ya que el estado de la parte cuántica no es importante para que ocurran las reacciones químicas. También hay que comentar que la resolución numérica del problema de control con el programa que se ha diseñado no es sencilla, ya que las ecuaciones podían no converger o la minimización de la función no ser lo suficientemente precisa. Debido a esto, obtener las trayectorias que se han mostrado ha sido bastante problemático ya que la optimización de la distancia y convergencia dependía mucho de las condiciones iniciales que se introducían en la función de optimización e integración. Por ello no se ha podido obtener una trayectoria que pasara por tres puntos usando la matriz densidad. Tampoco se han conseguido trayectorias que pasarán por más puntos debido a estas mismas causas. En definitiva este aspecto es el principal a mejorar del trabajo realizado.

A partir de este trabajo se pueden plantear otros problemas, como usar una función de acoplo $f(R)$ más realista del tipo coulombiana, es decir, inversa a la distancia o usar potenciales para la parte clásicas que no sean armónicos. También se podría mejorar el programa o usar otros lenguajes no abiertos como Mathematica con lo que se podrían minimizar los problemas que se han comentado anteriormente en la resolución numérica. Otro aspecto a estudiar más a fondo es la diferencia entre usar el espacio de Hilbert completo o el proyectivo en el formalismo híbrido, ya que como se ha visto, se obtienen diferentes trayectorias en función del que se utilice. Finalmente se podrían desarrollar otros problemas de optimización con este formalismo como el problema de tiempo mínimo.

Bibliografía

[1] R. Abraham and J. E. Marsden. «Foundations of Mechanics. Addison-Wesley». Redwood City, 2nd edition, 1987.

[2] T. WB Kibble. «Geometrization of quantum mechanics». Communications in Mathematical Physics 65.2, págs. 189-201(1979).

[3] J. Clemente-Gallardo. «The Geometrical Formulation of Quantum Mechanics». Rev. Real Academia de Ciencias 67 págs 51-103(2012). ISSN: 0370-3207

[4] J. L. Alonso et al. «Statistics and Nosé formalism for Ehrenfest dynamics». J. Phys. A: Math. Theor. 44 395004(2011)

[5] J. F. Cariñena et al.«Tensorial dynamics on the space of quantum states». J. Phys. A: Math. Theor. 50 365301(2017)

[6] J. A. Jover. «Sistemas cuánticos abiertos: descripción geométrica, dinámica y control». Tesis Doctoral. Universidad de Zaragoza. (2017)

[7] I. Tavernelli y B. Curchod. «Ab initio nonadiabatic molecular dynamics TDDFT for ultrafast electronic dynamics». TDDFT School Benasque (2014)

[8] F. A.Bornemann, P. Nettesheim y C. Schütte. «Quantum-classical molecular dynamics as an approximation to full quantum dynamics». J. Chem. Phys. 105 (3). 0021-9606/96/105(3)/1074/10 (1996)

[9] V. Jurdjevic. «Geometric control theory». Cambridge Studies in Advanced Mathematics, 52 (1997)

[10] L. C. Evans. «An Introduction to Mathematical Optimal Control Theory». Control course en Department of Mathematics, Berkeley. https://math.berkeley.edu/ evans/control.course.pdf

[11] B. Bonnard y J. Caillau. «1 Introduction to Nonlinear Optimal Control».En: Advanced Topics in Control Systems Theory. Lecture Notes in Control and Information Science, vol 328. Springer, London (2006)

[12] H. J. Sussmann y J. C. Willems. «300 Years of Optimal Control: From The Brachystochrone to the Maximum Principle». IEEE Control Systems 0272-1708/97 (1997)

A. Observables en sistemas híbridos

Para ver cómo se definen los observables en los sistemas híbridos primero hay que considerar las proyecciones en cada uno de los dos subsistemas:

$$\pi_C : M_C \times M_Q \to M_C \qquad \pi_Q : M_C \times M_Q \to M_Q \tag{A.1}$$

Para los observables vamos a imponer la condición de que sean constantes bajo los cambios de fase global de la parte cuántica. Esto se puede hacer a partir del campo vectorial de la ecuación (2.20) generalizado para este caso de la siguiente forma:

$$\Gamma_Q = \mathbb{I} \otimes \Gamma \qquad \text{tal que} \quad \pi_{C*}\Gamma_Q = 0 \quad \text{y} \quad \pi_{Q*}\Gamma_Q = \Gamma \tag{A.2}$$

donde π_{C*} y π_{Q*} son los *pushforward* y π_C^* y π_Q^* son los *pullback* de las proyecciones definidas en (A.1). Por ello, los observables serán funciones $f \in C^\infty(M_C \times M_Q)$ tales que $\Gamma_Q f = 0$. Dentro de este conjunto de observables hay tres conjuntos importantes. Primero el conjunto de los que solo dependen de los grados de libertad clásicos, por lo que serán las funciones $f_C \in C^\infty(M_C)$ tales que $f = \pi_C^*(f_C)$ por lo que cumplirán que $f(\vec{R}, \vec{P}, q, p) = f_C(\vec{R}, \vec{P})$. Otro conjunto es en el que solo dependen de los grados de libertad cuánticos: serán las funciones $f = \pi_Q^*(f_Q)$,tales que $f(\vec{R}, \vec{P}, p, q) = f_Q(q, p)$ y $\Gamma(f_Q) = 0$. Una propiedad importante es que este conjunto es mayor que el de los observables puramente cuánticos que estarán contenidos en este. Finalmente el conjunto de combinaciones lineales arbitrarias de los dos conjuntos anteriores, serán las funciones que cumplan que $f = \pi_C^*(f_C) + \pi_Q^*(f_Q)$ y $f(\vec{R}, \vec{P}, q, p) = f_C(\vec{R}, \vec{P}) + f_Q(q, p)$. Se ha usado una notación tal que las coordenadas sin índice indican el conjunto de las coordenadas, por ejemplo $q \equiv q_1, \ldots, q_n$.

B. Código para la controlabilidad

Este es el código usado para comprobar la controlabilidad en los dos casos.

```
1
2  from sympy import *
3  from scipy.optimize import minimize
4  from scipy.optimize import NonlinearConstraint
5  from scipy.optimize import basinhopping
6  from scipy.integrate import ode
7  from scipy.integrate import simps
8  from IPython.display import display, Math, Latex
9  from random import random
10 import matplotlib.pyplot as plt
11 from mpl_toolkits.mplot3d import Axes3D
12 import cmath
13 N_plot=1000
14 Px,Py,Rx,Ry,p1,p2,q1,q2,Bx,By=symbols('Px Py Rx Ry p1 p2 q1 q2 Bx By')
15 Xpx,Xpy,XRx, XRy,Xp1,Xp2,Xq1,Xq2=symbols('Xpx Xpy XRx XRy Xp1 Xp2 Xq1 Xq2')
16 ep,m,k,Bx2,By2,t=symbols('ep m k Bx1 By1 t')
17 d=8
18 H1=(Px**2/m/2+Py**2/m/2+k*(Rx**2+Ry**2)/2)*(q1**2+q2**2+p1**2+p2**2) #parte clasis
19 f=(Rx**2+Ry**2)
20 H2=ep*f*(Bx*(p1*p2+q1*q2)+By*(q1*p2-p1*q2)) #parte cuantica
21 H=H1+H2
22 x,y,z=symbols('x y z')
23 #ecuaciones de ehrefenst
24 Rxt=diff(H,Px)
25 Ryt=diff(H,Py)
26 Pxt=-diff(H,Rx)
27 Pyt=-diff(H,Ry)
28 q1t=diff(H,p1)
29 q2t=diff(H,p2)
30 p1t=-diff(H,q1)
31 p2t=-diff(H,q2)
32 #coordenadas
33 X=(Rx, Ry, Px, Py, q1, q2, p1, p2)
34 Xt=[]
35 Xt.append(Rxt)
36 Xt.append(Ryt)
37 Xt.append(Pxt)
38 Xt.append(Pyt)
39 Xt.append(q1t)
40 Xt.append(q2t)
41 Xt.append(p1t)
42 Xt.append(p2t)
43 V=Matrix(Xt) #vector que guarda la evolucion temporal de cada componente
44 X0=V.subs(Bx,0).subs(By,0) #calculo de los 3 campos de la dinamica del sistema
45 Xx=(V-X0).subs(By,0)/Bx
46 Xy=(V-X0).subs(Bx,0)/By
47 def corchete(X,Y,C): #funcion que calcula el corchete de Lie
48     R=[0,0,0,0,0,0,0,0]
49     for i in range (0,8):
50         for j in range (0,8):
51             R[i]=R[i]+X[j]*diff(Y[i],C[j])-Y[j]*diff(X[i],C[j])
52     return Matrix(R);
53 def ind(F,P,a):#funcion que comprueba si al anadir un vector a una familia este es
54     A=F[:,:]
55     A[:,a]=P
```

```
    A=A.rref(simplify=True)[0]
    b=A.rank()
    return b
F=zeros(8,8)
F[:,0]=X0
F[:,1]=Xx
F[:,2]=Xy
c=F.rank()

while c<(d-1):
    z=c
    for i in range(0,z):
       # print(i,z)
        for j in range(0,z):
            if j>i and c<d:
              aux=corchete(F[:,z-1-i],F[:,z-1-j],X)
              b=ind(F,aux,c)
              if b>c:
                 F[:,c]=aux
                 c=b
                 print('cambio')
                 print(c,z-1-i,z-1-j)
                 i=z
                 j=z
              else:
               print(c,z-1-i,z-1-j)
    if z==c:
        c=d
for i in range(0,d):
   print(F[:,i])
print(F.rank())
#Repetimos el proceso con rho
Px,Py,Rx,Ry,px,py,pz,Bx,By=symbols('Px Py Rx Ry px py pz Bx By')
XPx, XPy,XRx,XRy, Xpx,Xpy,Xpz=symbols('XPx XPy XRx XRy Xpx Xpy Xpz')
ep,m,k,Bx2,By2,t=symbols('ep m k Bx1 By1 t')
 d=7
H1=Px**2/m/2+Py**2/m/2+k*(Rx**2+Ry**2)/2 #parte clasica
f=(Rx**2+Ry**2)
#f=1/(1+sqrt(Rx**2+Ry**2))
H2=ep*f*(Bx*px+By*py) #parte cuantica
H=H1+H2
print(H)
#ecuaciones de ehrefenst
Rxt=diff(H,Px)
Ryt=diff(H,Py)
Pxt=-diff(H,Rx)
Pyt=-diff(H,Ry)
pxt=ep*f*By*pz
pyt=-ep*f*Bx*pz
pzt=ep*f*(Bx*py-By*px)
X=(Rx, Ry, Px, Py, px, py, pz)
Xt=[]
Xt.append(Rxt)
Xt.append(Ryt)
Xt.append(Pxt)
Xt.append(Pyt)
Xt.append(pxt)
Xt.append(pyt)
```

```
Xt.append(pzt)
V=Matrix(Xt) #vector que guarda la evolucioon temporal de cada componente
X0=V.subs(Bx,0).subs(By,0) #calculo de los 3 campos de la dinamica del sistema
Xx=(V-X0).subs(By,0)/Bx
Xy=(V-X0).subs(Bx,0)/By
def corchete2(X,Y,C): #funcion que calcula el corchete de Lie
    R=[0,0,0,0,0,0,0]
    for i in range (0,d):
        for j in range (0,d):
            if i<4 or j<4:
             R[i]=R[i]+X[j]*diff(Y[i],C[j])-Y[j]*diff(X[i],C[j])
            if i==4 and j==5:
             R[i]=R[i]+(X[j]*diff(Y[i],C[j])-Y[j]*diff(X[i],C[j]))*pz
            if i==5 and j==4:
             R[i]=R[i]-(X[j]*diff(Y[i],C[j])-Y[j]*diff(X[i],C[j]))*pz
            if i==5 and j==6:
             R[i]=R[i]+(X[j]*diff(Y[i],C[j])-Y[j]*diff(X[i],C[j]))*px
            if i==6 and j==5:
             R[i]=R[i]-(X[j]*diff(Y[i],C[j])-Y[j]*diff(X[i],C[j]))*px
            if i==4 and j==6:
             R[i]=R[i]-(X[j]*diff(Y[i],C[j])-Y[j]*diff(X[i],C[j]))*py
            if i==6 and j==4:
             R[i]=R[i]+(X[j]*diff(Y[i],C[j])-Y[j]*diff(X[i],C[j]))*py
    return Matrix(R);
B=corchete(Xx,Xy,X)[:]
F=zeros(d,d)
F[:,0]=X0
F[:,1]=Xx
F[:,2]=Xy
c=F.rank()
while c<(d-1):
    z=c
    for i in range(0,z):
       # print(i,z)
        for j in range(0,z):
            if j>i and c<d:
              aux=corchete2(F[:,i],F[:,j],X)[:]
              b=ind(F,aux,c)
              if b>c:
                 F[:,c]=aux
                 c=b
                 print('cambio')
              print(c,i,j)
    if z==c:
        c=d
for i in range(0,c):
    print(F[:,i])
    print()
print(F.rank())
```

C. Código para la optimización

Este es el código para la resolución de las trayectorias en el primer caso mostrado:

```
from numpy import kron,dot, array
from sympy import *
from scipy.optimize import minimize
from scipy.optimize import NonlinearConstraint
from scipy.optimize import basinhopping
import numpy as np
from scipy.integrate import ode
from scipy.integrate import simps
from IPython.display import display, Math, Latex
from random import random
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
Px,Py,Rx,Ry,p1,p2,q1,q2,Bx,By=symbols('Px Py Rx Ry p1 p2 q1 q2 Bx By')
Xpx,Xpy,XRx, XRy,Xp1,Xp2,Xq1,Xq2=symbols('Xpx Xpy XRx XRy Xp1 Xp2 Xq1 Xq2')
ep,m,k,Bx2,By2,t=symbols('ep m k Bx1 By1 t')
N_plot=1000
H1=Px**2/m/2+Py**2/m/2+k*(Rx**2+Ry**2)/2 #parte clasisca
f=(Rx**2+Ry**2)#acoplo
H2=ep*f*(Bx*(p1*p2+q1*q2)+By*(q1*p2-p1*q2)) #parte cuantica
H=H1+H2
print(H)
#ecuaciones de ehrefenst
Rxt=diff(H,Px)
Ryt=diff(H,Py)
Pxt=-diff(H,Rx)
Pyt=-diff(H,Ry)
q1t=diff(H,p1)
q2t=diff(H,p2)
p1t=-diff(H,q1)
p2t=-diff(H,q2)
print(q1t)
print(q2t)
print(p1t)
print(p2t)
#Hamiltoniano Pontryagin
H=Xpx*Pxt+Xpy*Pyt+XRx*Rxt+XRy*Ryt+Xq1*q1t+Xq2*q2t+Xp1*p1t+Xp2*p2t-Bx**2-By**2
print(H)
print()
#despejamos los controles
Bx1=diff(H,Bx)+2*Bx
By1=diff(H,By)+2*By
Bx2=Bx1/2
By2=By1/2
print(Bx2)
print(By2)
#ecuaciones para los coestados
XRxt=-diff(H,Rx)
XRyt=-diff(H,Ry)
Xpxt=-diff(H,Px)
Xpyt=-diff(H,Py)
Xq1t=-diff(H,q1)
Xq2t=-diff(H,q2)
Xp1t=-diff(H,p1)
Xp2t=-diff(H,p2)
#sustituimos los valores de Bx y By
```

```
r=[(Bx,Bx2),(By,By2)]
Xt=[]
Xt.append(Rxt.subs(r))
Xt.append(Ryt.subs(r))
Xt.append(Pxt.subs(r))
Xt.append(Pyt.subs(r))
Xt.append(q1t.subs(r))
Xt.append(q2t.subs(r))
Xt.append(p1t.subs(r))
Xt.append(p2t.subs(r))
Xt.append(XRxt.subs(r))
Xt.append(XRyt.subs(r))
Xt.append(Xpxt.subs(r))
Xt.append(Xpyt.subs(r))
Xt.append(Xq1t.subs(r))
Xt.append(Xq2t.subs(r))
Xt.append(Xp1t.subs(r))
Xt.append(Xp2t.subs(r))
L=len(Xt)
#for i in range(0,15):
#simplify(Xt[i])
print(Xt[4])
print(Xt[5])
print(Xt[6])
print(Xt[7])
#sustituimos los valores de m ,k y ep
r=[(k,1),(m,1),(ep,10)]
for i in range (0,L):
    Xt[i]=Xt[i].subs(r)
    #simplify((Xt[i]))
Bx2=Bx2.subs(r)
By2=By2.subs(r)
#creamos un vector con las variables
X=(Rx,Ry,Px,Py,q1,q2,p1,p2,XRx,XRy,Xpx,Xpy,Xq1, Xq2,Xp1,Xp2)
Bx=lambdify([X],Bx2)
By=lambdify([X],By2)
#definimos la funcion para resolver la ODE
f1=[]
for i in range (0,L):
 f1.append(lambdify([X,t],Xt[i]))
def f(t,x,arg):
    re=[]
    for i in range (0,L):
        re.append(f1[i](x,t))
    return re;
#definimos el jacobiano para resolver la ODE
j1=[None]*L
for i in range(0,L):
    j1[i]=[None]*L

for i in range(0,L):
 for j in range(0,L):
        #j1[i][j]=lambdify([X,t],simplify(diff(Xt[i],X[j])))
         j1[i][j]=lambdify([X,t],diff(Xt[i],X[j]))
def jac(t,x,arg):
    re=[None]*L
    for i in range(0,L):
        re[i]=[None]*L
```

```
    for i in range(0,L):
        for j in range(0,L):
         re[i][j]=j1[i][j](x,t)
    return re;
def distancia(X1,X2,n):
  a=0
  for i in range (0,n):
    a=a+(X1[i]-X2[i])**2
  return sqrt(a)
X0,t0=[0,0,1,1,1.0,0.0,0,0,0,0,0,0,0,0,0,0],0.0#condiciones inciales
dt=0.01
t1=1
X_obj=[1,1,0,0]#objetivo
X_Ci=[0,0,0, 0,0,0,0,0]#condiciones de inicio de optiomizacion
def evolucion(X0_C):#funcion que es la resolucion del sistema
  X0[8]=X0_C[0]
  X0[9]=X0_C[1]
  X0[10]=X0_C[2]
  X0[11]=X0_C[3]
  X0[12]=X0_C[4]
  X0[13]=X0_C[5]
  X0[14]=X0_C[6]
  X0[15]=X0_C[7]
  sol=ode(f,jac).set_integrator('dop853')
  sol.set_initial_value(X0,t0).set_f_params(1.0).set_jac_params(2.0)
  a=100000
  b=[0,0,0,0]
  while sol.successful() and sol.t < t1:
     sol.integrate(sol.t+dt)
     for i in range(0,4):
       b[i]=sol.y[i]
     c=distancia(b,X_obj,4)
     if(c<a):
        a=c
  return a;
#optimizamos los coestados iniciales
#res=minimize(evolucion,X_Ci,method='nelder-mead',
#options={'xtol': 0.000001,'ftol':0.000001,'maxiter':300, 'disp':True})
minimizer= {'method': 'Nelder-Mead'}
res=basinhopping(evolucion,X_Ci,minimizer_kwargs=minimizer,niter=100)
print(res)

for i in range(0,8):#ponemos como condiciones iniciales el resultado
    X0[i+8]=res.x[i]
dt=0.001
t1=1
a=100000
b=[0,0,0,0]
sol=ode(f,jac).set_integrator('dop853')
sol.set_initial_value(X0,t0).set_f_params(1.0).set_jac_params(2.0)
j=0
dat=[]
X_fin=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0, 0,  0, 0,  0,0,  0,0,0]
while sol.successful() and sol.t < t1:
     j=j+1
     sol.integrate(sol.t+dt)
     for i in range(0,4):
       b[i]=sol.y[i]
```

```
        c=distancia(b,X_obj,4)
        if(c<a):
            a=c
            X_fin=sol.y
        dat.append(sol.t)
        for ii in range(0,L):
            dat.append(sol.y[ii])
        dat.append(Bx(sol.y))
        dat.append(By(sol.y))

print(j)
print(a)
print(len(dat))
tplot=[]
Eplot=[]
for i in range(0,N_plot):
    tplot.append(dat[i*19])
    Eplot.append(dat[17+i*19]**2+dat[18+i*19]**2)
I=simps(Eplot,tplot)
print(I)
Rxplot=[]
Pxplot=[]
for i in range(0,N_plot):
    Rxplot.append(dat[1+i*19])
    Pxplot.append(dat[3+i*19])

Ryplot=[]
Pyplot=[]
for i in range(0,N_plot):
    Ryplot.append(dat[2+i*19])
    Pyplot.append(dat[4+i*19])

def proyeccion(q1,q2,p1,p2):
    x=2*(q1*q2+p1*p2)
    y=2*(q1*p2-q2*p1)
    z=(q1**2+p1**2-q2**2-p2**2)
    return x,y,z
pxplot=[]
pyplot=[]
pzplot=[]
aux=[0,0,0]
for i in range(0,N_plot):
    aux=proyeccion(dat[5+i*19],dat[6+i*19],dat[7+i*19],dat[8+i*19])
    pxplot.append(aux[0])
    pyplot.append(aux[1])
    pzplot.append(aux[2])
#Repetimos el proceso con rho
Px,Py,Rx,Ry,px,py,pz,Bx,By=symbols('Px Py Rx Ry px py pz Bx By')
XPx, XPy,XRx,XRy, Xpx,Xpy,Xpz=symbols('XPx XPy XRx XRy Xpx Xpy Xpz')
ep,m,k,Bx2,By2,t=symbols('ep m k Bx1 By1 t')

H1=Px**2/m/2+Py**2/m/2+k*(Rx**2+Ry**2)/2 #parte classca
#f=(Rx**2+Ry**2+Px**2+Py**2) #acoplo
#f=0
f=(Rx**2+Ry**2)
#f=1/(1+sqrt(Rx**2+Ry**2))
H2=ep*f*(Bx*px+By*py) #parte cuantica
H=H1+H2
```

```
print(H)
#ecuaciones de ehrefenst
Rxt=diff(H,Px)
Ryt=diff(H,Py)
Pxt=-diff(H,Rx)
Pyt=-diff(H,Ry)
pxt=ep*f*By*pz
pyt=-ep*f*Bx*pz
pzt=ep*f*(Bx*py-By*px)
#Hamiltoniano Pontryagin
H=XPx*Pxt+XPy*Pyt+XRx*Rxt+XRy*Ryt+Xpx*pxt+Xpy*pyt+Xpz*pzt-Bx**2-By**2
print(H)
#despejamos los controles
Bx1=diff(H,Bx)+2*Bx
By1=diff(H,By)+2*By
Bx2=Bx1/2
By2=By1/2
print(Bx2)
print(By2)
#ecuaciones para los coestados
XRxt=-diff(H,Rx)
XRyt=-diff(H,Ry)
XPxt=-diff(H,Px)
XPyt=-diff(H,Py)
Xpxt=-diff(H,px)
Xpyt=-diff(H,py)
Xpzt=-diff(H,pz)
#sustituimos los valores de Bx y By
r=[(Bx,Bx2),(By,By2)]
Xt2=[]
Xt2.append(Rxt.subs(r))
Xt2.append(Ryt.subs(r))
Xt2.append(Pxt.subs(r))
Xt2.append(Pyt.subs(r))
Xt2.append(pxt.subs(r))
Xt2.append(pyt.subs(r))
Xt2.append(pzt.subs(r))
Xt2.append(XRxt.subs(r))
Xt2.append(XRyt.subs(r))
Xt2.append(XPxt.subs(r))
Xt2.append(XPyt.subs(r))
Xt2.append(Xpxt.subs(r))
Xt2.append(Xpyt.subs(r))
Xt2.append(Xpzt.subs(r))
L=len(Xt2)
print(L)
#for i in range(0,15):
# simplify(Xt[i])
#sustituimos los valores de m ,k y ep
r=[(k,1),(m,1),(ep,10)]
for i in range (0,L):
    Xt2[i]=Xt2[i].subs(r)
    simplify((Xt[i]))
Bx2=Bx2.subs(r)
By2=By2.subs(r)
#creamos un vector con las variables
X2=Matrix([Rx,Ry,Px,Py,px,py,pz,XRx,XRy,XPx,XPy,Xpx,Xpy,Xpz])
Bx=lambdify([X2],Bx2)
```

```
By=lambdify([X2],By2)
#definimos la funcion para resolver la ODE
f1=[]
for i in range (0,L):
 f1.append(lambdify([X2,t],Xt2[i]))
def f(t,x,arg):
    re=[]
    for i in range (0,L):
        re.append(f1[i](x,t))
    return re;
#definimos el jacobiano para resolver la ODE
j1=[None]*L
for i in range(0,L):
    j1[i]=[None]*L

for i in range(0,L):
 for j in range(0,L):
        j1[i][j]=lambdify([X2,t],simplify(diff(Xt2[i],X2[j])))
def jac(t,x,arg):
    re=[None]*L
    for i in range(0,L):
        re[i]=[None]*L
    for i in range(0,L):
        for j in range(0,L):
         re[i][j]=j1[i][j](x,t)
    return re;
def distancia(X1,X2,n):
  a=0
  for i in range (0,n):
    a=a+(X1[i]-X2[i])**2
  return sqrt(a)
X0,t0=[0, 0,1,1,0,0.0, 1.0,0,0,0,0,0,0,0],0.0#condiciones inciales
dt=0.01
t1=1
X_obj=[1,1,0,0]#objetivo
X_Ci=[0,  0,  0, 0,  0,  0,0]#condiciones de inicio de optiomizacion
def evolucion2(X0_C):#funcion que es la resolucion del sistema
  X0[7]=X0_C[0]
  X0[8]=X0_C[1]
  X0[9]=X0_C[2]
  X0[10]=X0_C[3]
  X0[11]=X0_C[4]
  X0[12]=X0_C[5]
  X0[13]=X0_C[6]
  sol=ode(f,jac).set_integrator('dop853')
  sol.set_initial_value(X0,t0).set_f_params(1.0).set_jac_params(2.0)
  a=100000
  while sol.successful() and sol.t < t1:
     sol.integrate(sol.t+dt)
     c=distancia(sol.y,X_obj,4)
     if(c<a):
        a=c
  return a;
#optimizamos los coestados iniciales
#res=minimize(evolucion,X_Ci,method='nelder-mead',
#options={'xtol': 0.000001,'ftol':0.000001,'maxiter':300, 'disp':True})
minimizer={'method': 'Nelder-Mead'}
res=basinhopping(evolucion,X_Ci,minimizer_kwargs=minimizer,niter=100)
```

```
print(res)

for i in range(0,7):#ponemos como condiciones iniciales el resultado
    X0[i+8]=res.x[i]
dt=0.001
t1=1
a=100000
b=[0,0,0,0]
sol=ode(f,jac).set_integrator('dop853')
sol.set_initial_value(X0,t0).set_f_params(1.0).set_jac_params(2.0)
j=0
dat2=[]
X_fin=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0, 0,  0, 0,  0,0,  0]
while sol.successful() and sol.t < t1:
     j=j+1
     sol.integrate(sol.t+dt)
     c=distancia(sol.y,X_obj,4)
     if(c<a):
        a=c
        X_fin=sol.y
     dat2.append(sol.t)
     for ii in range(0,L):
        dat2.append(sol.y[ii])
     dat2.append(Bx(sol.y))
     dat2.append(By(sol.y))
print(j)
print(a)
print(len(dat2))
tplot2=[]
Eplot2=[]
for i in range(0,N_plot):
    tplot2.append(dat2[i*17])
    Eplot2.append(dat2[15+i*17]**2+dat2[16+i*17]**2)
I=simps(Eplot2,tplot2)
print(I)
#representamos los resultados
Rxplot2=[]
Pxplot2=[]
for i in range(0,N_plot):
    Rxplot2.append(dat2[1+i*17])
    Pxplot2.append(dat2[3+i*17])
plt.plot(Rxplot,Pxplot,label='qp')
plt.plot(Rxplot2,Pxplot2,label=r'$\rho $')
plt.plot([0,1],[1,0],'ro',label='objetivo')

plt.xlabel('Posicion en X ($R_x$)')
plt.ylabel('Momento en X ($P_x$)')
plt.legend()
plt.savefig('X1.pdf',bbox_inches='tight')
plt.show()
Ryplot2=[]
Pyplot2=[]
for i in range(0,N_plot):
    Ryplot2.append(dat2[2+i*17])
    Pyplot2.append(dat2[4+i*17])
plt.plot(Ryplot,Pyplot,label=r'qp')
plt.plot(Ryplot2,Pxplot2,label=r'$\rho $')
plt.plot([0,1],[1,0],'ro',label='objetivo')
```

```
plt.xlabel('Posicion en Y ($R_y$)')
plt.ylabel('Momento en Y ($P_y$)')
plt.legend()
plt.savefig('Y1.pdf',bbox_inches='tight')
plt.show()
plt.plot(tplot,Eplot,label=r'qp')
plt.plot(tplot2,Eplot2,label=r'$\rho $')
plt.xlabel('tiempo')
plt.ylabel('$B_x^2+B_y^2$')
plt.legend()
plt.savefig('E1.pdf',bbox_inches='tight')
plt.show()
pxplot2=[]
pyplot2=[]
pzplot2=[]
for i in range(0,N_plot):
    pxplot2.append(dat2[5+i*17])
    pyplot2.append(dat2[6+i*17])
    pzplot2.append(dat2[7+i*17])
# dibjamos la esfera
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
u = np.linspace(0, 2 * np.pi, 100)
v = np.linspace(0, np.pi, 100)
x1 = 1 * np.outer(np.cos(u), np.sin(v))
y1 = 1 * np.outer(np.sin(u), np.sin(v))
z1 = 1 * np.outer(np.ones(np.size(u)), np.cos(v))
ax.plot_wireframe(x1, y1, z1, color='grey',rstride=4,cstride=4)
ax.plot(pxplot, pyplot, pzplot,label=r'qp')
ax.plot(pxplot2, pyplot2, pzplot2,label=r'$\rho $')
ax.view_init(elev=30, azim=135)
ax.set_xlabel(r'$\rho_x $')
ax.set_ylabel(r'$\rho_y $')
ax.set_zlabel(r'$\rho_z $')
ax.w_xaxis.set_pane_color((1.0, 1.0, 1.0, 0.0))
ax.w_yaxis.set_pane_color((1.0, 1.0, 1.0, 0.0))
ax.w_zaxis.set_pane_color((1.0, 1.0, 1.0, 0.0))
ax.w_xaxis.line.set_color((1.0, 1.0, 1.0, 0.0))
ax.w_yaxis.line.set_color((1.0, 1.0, 1.0, 0.0))
ax.w_zaxis.line.set_color((1.0, 1.0, 1.0, 0.0))
ax.zaxis.set_rotate_label(False)
ax.set_xticks([-1,-0.5, 0,0.5, 1])
ax.set_yticks([-1,-0.5, 0,0.5, 1])
ax.set_zticks([-1,-0.5, 0,0.5, 1])
plt.legend()
plt.savefig('Bloch1.pdf',bbox_inches='tight')
plt.show
```

Para el segundo caso la única modificación que hay que hacer es calcular la distancia usando también las coordenadas cuánticas por lo que no se muestra el código al ser muy similar.

Para el tercer caso el código usado es el siguiente:

```
from numpy import kron,dot, array
from sympy import *
from scipy.optimize import minimize
from scipy.optimize import NonlinearConstraint
from scipy.optimize import basinhopping
```

```
import numpy as np
from scipy.integrate import ode
from scipy.integrate import simps
from IPython.display import display, Math, Latex
from random import random
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
Px,Py,Rx,Ry,p1,p2,q1,q2,Bx,By=symbols('Px Py Rx Ry p1 p2 q1 q2 Bx By')
Xpx,Xpy,XRx, XRy,Xp1,Xp2,Xq1,Xq2=symbols(' Xpx Xpy XRx XRy Xp1 Xp2 Xq1 Xq2')
ep,m,k,Bx2,By2,t=symbols('ep m k Bx1 By1 t')
N_plot=1000
H1=Px**2/m/2+Py**2/m/2+k*(Rx**2+Ry**2)/2 #parte clasisca
#f=(Rx**2+Ry**2+Px**2+Py**2) #acoplo
#f=0
f=(Rx**2+Ry**2)
#f=1
H2=ep*f*(Bx*(p1*p2+q1*q2)+By*(q1*p2-p1*q2)) #parte cuantica
H=H1+H2
print(H)
#ecuaciones de ehrefenst
Rxt=diff(H,Px)
Ryt=diff(H,Py)
Pxt=-diff(H,Rx)
Pyt=-diff(H,Ry)
q1t=diff(H,p1)
q2t=diff(H,p2)
p1t=-diff(H,q1)
p2t=-diff(H,q2)
print(q1t)
print(q2t)
print(p1t)
print(p2t)
#Hamiltoniano Pontryagin
H=Xpx*Pxt+Xpy*Pyt+XRx*Rxt+XRy*Ryt+Xq1*q1t+Xq2*q2t+Xp1*p1t+Xp2*p2t-Bx**2-By**2
print(H)
print()
#despejamos los controles
Bx1=diff(H,Bx)+2*Bx
By1=diff(H,By)+2*By
Bx2=Bx1/2
By2=By1/2
print(Bx2)
print(By2)
#ecuaciones para los coestados
XRxt=-diff(H,Rx)
XRyt=-diff(H,Ry)
Xpxt=-diff(H,Px)
Xpyt=-diff(H,Py)
Xq1t=-diff(H,q1)
Xq2t=-diff(H,q2)
Xp1t=-diff(H,p1)
Xp2t=-diff(H,p2)
#sustituimos los valores de Bx y By
r=[(Bx,Bx2),(By,By2)]
Xt=[]
Xt.append(Rxt.subs(r))
Xt.append(Ryt.subs(r))
Xt.append(Pxt.subs(r))
```

```
Xt.append(Pyt.subs(r))
Xt.append(q1t.subs(r))
Xt.append(q2t.subs(r))
Xt.append(p1t.subs(r))
Xt.append(p2t.subs(r))
Xt.append(XRxt.subs(r))
Xt.append(XRyt.subs(r))
Xt.append(Xpxt.subs(r))
Xt.append(Xpyt.subs(r))
Xt.append(Xq1t.subs(r))
Xt.append(Xq2t.subs(r))
Xt.append(Xp1t.subs(r))
Xt.append(Xp2t.subs(r))
L=len(Xt)
#for i in range(0,15):
#simplify(Xt[i])
print(Xt[4])
print(Xt[5])
print(Xt[6])
print(Xt[7])
#sustituimos los valores de m ,k y ep
r=[(k,1),(m,1),(ep,10)]
for i in range (0,L):
    Xt[i]=Xt[i].subs(r)
    #simplify((Xt[i]))
Bx2=Bx2.subs(r)
By2=By2.subs(r)
#creamos un vector con las variables
X=(Rx, Ry,Px,Py,q1,q2,p1,p2,XRx,XRy,Xpx,Xpy,Xq1,Xq2,Xp1,Xp2)
Bx=lambdify([X],Bx2)
By=lambdify([X],By2)
#definimos la funcion para resolver la ODE
f1=[]
for i in range (0,L):
 f1.append(lambdify([X,t],Xt[i]))
def f(t,x,arg):
    re=[]
    for i in range (0,L):
        re.append(f1[i](x,t))
    return re;
#definimos el jacobiano para resolver la ODE
j1=[None]*L
for i in range(0,L):
    j1[i]=[None]*L

for i in range(0,L):
 for j in range(0,L):
        #j1[i][j]=lambdify([X,t],simplify(diff(Xt[i],X[j])))
         j1[i][j]=lambdify([X,t],diff(Xt[i],X[j]))
def jac(t,x,arg):
    re=[None]*L
    for i in range(0,L):
        re[i]=[None]*L
    for i in range(0,L):
        for j in range(0,L):
         re[i][j]=j1[i][j](x,t)
    return re;

```

```
122
123 def distancia(X1,X2,n):
124   a=0
125   for i in range (0,n):
126     a=a+(X1[i]-X2[i])**2
127   return sqrt(a)
128 X0,t0=[0,0,1,1,1.0,0.0,0,0,0,0,0,0,0,0,0,0],0.0#condiciones inciales
129 dt=0.01
130 t1=0.5
131 t2=1
132 X_obj=[0.5,0.5,0.5,0.5]
133 X_obj2=[0.25,0.25,0.75,0.75]#objetivo
134 X_Ci=[1,1,0,0,0, 0,0.1,0.1]#condiciones de inicio de optiomizacion
135 def evolucion(X0_C):#defiimos la funcion que es la resolucion del sistema
136   X0[8]=X0_C[0]
137   X0[9]=X0_C[1]
138   X0[10]=X0_C[2]
139   X0[11]=X0_C[3]
140   X0[12]=X0_C[4]
141   X0[13]=X0_C[5]
142   X0[14]=X0_C[6]
143   X0[15]=X0_C[7]
144   sol=ode(f,jac).set_integrator('dop853')
145   sol.set_initial_value(X0,t0).set_f_params(1.0).set_jac_params(2.0)
146   a=100000
147   a2=10000
148   while sol.successful() and sol.t < t1:
149      sol.integrate(sol.t+dt)
150      c=distancia(sol.y,X_obj,4)
151      if(c<a):
152         a=c
153   while sol.successful() and sol.t < t2:
154      sol.integrate(sol.t+dt)
155      c=distancia(sol.y,X_obj2,4)
156      if(c<a2):
157         a2=c
158   return a+a2;
159 #optimizamos los coestados iniciales
160 #res=minimize(evolucion,X_Ci,method='nelder-mead',
161 #options={'xtol': 0.000001,'ftol':0.000001,'maxiter':300, 'disp':True})
162 minimizer={'method': 'Nelder-Mead'}
163 res=basinhopping(evolucion,X_Ci,minimizer_kwargs=minimizer,niter=100)
164 print(res)
165 for i in range(0,8):#ponemos como condiciones iniciales el resultado
166     X0[i+8]=res.x[i]
167 dt=0.001
168 t1=1
169 a=100000
170 a2=10000
171 b=[0,0,0,0]
172 sol=ode(f,jac).set_integrator('dop853')
173 sol.set_initial_value(X0,t0).set_f_params(1.0).set_jac_params(2.0)
174 j=0
175 dat=[]
176 X_fin=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0, 0,  0, 0,  0,0,  0,0,0]
177 X_fin2=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0, 0,  0, 0,  0,0,  0,0,0]
178 while sol.successful() and sol.t < t2:
179      j=j+1
```

```
180       sol.integrate(sol.t+dt)
181       c=distancia(sol.y,X_obj,4)
182       c2=distancia(sol.y,X_obj2,4)
183       if(c<a):
184          a=c
185          X_fin=sol.y
186          taux=sol.t
187       if(c2<a2):
188          a2=c2
189          X_fin2=sol.y
190          taux2=sol.t
191       dat.append(sol.t)
192       for ii in range(0,L):
193          dat.append(sol.y[ii])
194       dat.append(Bx(sol.y))
195       dat.append(By(sol.y))
196 print(a)
197 print(a2)
198 print(taux)
199 print(taux2)
200 print(len(dat))
201 tplot=[]
202 Eplot=[]
203 tplot2=[]
204 Eplot2=[]
205 for i in range(0,N_plot):
206     if(dat[i*19]<taux):
207       tplot.append(dat[i*19])
208       Eplot.append(dat[17+i*19]**2+dat[18+i*19]**2)
209     if(dat[i*19]>taux):
210       tplot2.append(dat[i*19])
211       Eplot2.append(dat[17+i*19]**2+dat[18+i*19]**2)
212 I=simps(Eplot,tplot)
213 print(I)
214 Rxplot=[]
215 Pxplot=[]
216 Rxplot2=[]
217 Pxplot2=[]
218 for i in range(0,N_plot):
219     if(dat[i*19]<taux):
220       Rxplot.append(dat[1+i*19])
221       Pxplot.append(dat[3+i*19])
222     if(dat[i*19]>taux):
223       Rxplot2.append(dat[1+i*19])
224       Pxplot2.append(dat[3+i*19])
225 Ryplot=[]
226 Pyplot=[]
227 Ryplot2=[]
228 Pyplot2=[]
229 for i in range(0,N_plot):
230     if(dat[i*19]<taux):
231       Ryplot.append(dat[2+i*19])
232       Pyplot.append(dat[4+i*19])
233     if(dat[i*19]>taux):
234       Ryplot2.append(dat[2+i*19])
235       Pyplot2.append(dat[4+i*19])
236 def proyeccion(q1,q2,p1,p2):
237     x=2*(q1*q2+p1*p2)
```

```
    y=2*(q1*p2-q2*p1)
    z=(q1**2+p1**2-q2**2-p2**2)
    return x,y,z
pxplot=[]
pyplot=[]
pzplot=[]
pxplot2=[]
pyplot2=[]
pzplot2=[]
aux=[0,0,0]
for i in range(0,N_plot):
    aux=proyeccion(dat[5+i*19],dat[6+i*19],dat[7+i*19],dat[8+i*19])
    if(dat[i*19]<taux):
     pxplot.append(aux[0])
     pyplot.append(aux[1])
     pzplot.append(aux[2])
    if(dat[i*19]>taux):
     pxplot2.append(aux[0])
     pyplot2.append(aux[1])
     pzplot2.append(aux[2])
#representamos los resultados
plt.plot(Rxplot,Pxplot,label='1 parte')
plt.plot(Rxplot2,Pxplot2,label='2 parte')
plt.plot([0,0.5,0.25],[1,0.5,0.75],'ro',label='objetivo')
plt.xlabel('Posicion en X ($R_x$)')
plt.ylabel('Momento en X ($P_x$)')
plt.legend()
plt.savefig('X3.pdf',bbox_inches='tight')
plt.show()
plt.plot(Ryplot,Pyplot,label=r'1 parte')
plt.plot(Ryplot2,Pxplot2,label=r'2 parte')
plt.plot([0,0.5,0.25],[1,0.5,0.75],'ro',label='objetivo')
plt.xlabel('Posicion en Y ($R_y$)')
plt.ylabel('Momento en Y ($P_y$)')
plt.legend()
plt.savefig('Y3.pdf',bbox_inches='tight')
plt.show()
plt.plot(tplot,Eplot,label=r'1 parte')
plt.plot(tplot2,Eplot2,label=r'2 parte')
plt.xlabel('tiempo')
plt.ylabel('$B_x^2+B_y^2$')
plt.legend()
plt.savefig('E3.pdf',bbox_inches='tight')
plt.show()
# draw sphere
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
u = np.linspace(0, 2 * np.pi, 100)
v = np.linspace(0, np.pi, 100)
x1 = 1 * np.outer(np.cos(u), np.sin(v))
y1 = 1 * np.outer(np.sin(u), np.sin(v))
z1 = 1 * np.outer(np.ones(np.size(u)), np.cos(v))
ax.plot_wireframe(x1, y1, z1, color='grey',rstride=4,cstride=4)
ax.plot(pxplot, pyplot, pzplot,label=r'1 parte')
ax.plot(pxplot2, pyplot2, pzplot2,label=r'2 parte')
ax.view_init(elev=30, azim=135)
ax.set_xlabel(r'$\rho_x $')
ax.set_ylabel(r'$\rho_y $')
```

```
296 ax.set_zlabel(r'$\rho_z $')
297 ax.w_xaxis.set_pane_color((1.0, 1.0, 1.0, 0.0))
298 ax.w_yaxis.set_pane_color((1.0, 1.0, 1.0, 0.0))
299 ax.w_zaxis.set_pane_color((1.0, 1.0, 1.0, 0.0))
300 ax.w_xaxis.line.set_color((1.0, 1.0, 1.0, 0.0))
301 ax.w_yaxis.line.set_color((1.0, 1.0, 1.0, 0.0))
302 ax.w_zaxis.line.set_color((1.0, 1.0, 1.0, 0.0))
303 ax.zaxis.set_rotate_label(False)
304 ax.set_xticks([-1,-0.5, 0,0.5, 1])
305 ax.set_yticks([-1,-0.5, 0,0.5, 1])
306 ax.set_zticks([-1,-0.5, 0,0.5, 1])
307 plt.legend()
308 plt.savefig('Bloch3.pdf',bbox_inches='tight')
309 plt.show
```