industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

Universidad Politécnica de Cartagena

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Control avanzado de un qubit

TRABAJO FIN DE MÁSTER

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

Autor: **Yolanda Villalba Celdrán**
Director: Juan Ignacio Mulero Martínez
Codirector: Javier Molina Vilaplana

Cartagena, Septiembre 2022

En primer lugar, me gustaría agradecer a la Universidad Politécnica de Cartagena, por formarme y acogerme durante todos estos años, en especial al área de Ingeniería de Sistemas y Automática, por haber hecho posible este trabajo.

Agradecer también a Juan Ignacio Mulero Martínez y a Javier Molina Vilaplana, por su atención y ayuda, que ha sido fundamental para la realización de este proyecto.

Y por último a mi familia, pareja y amigos, quienes siempre han estado dispuestos en todo momento a ayudarme y acompañarme en este camino. Sin vosotros nada de esto habría sido posible.

Gracias.

Resumen

Villalba Celdrán, Yolanda. *"Control avanzado de un qubit"*. Cartagena, Septiembre 2022.

El control es una rama interdisciplinaria entre la ingeniería y las matemáticas cuyo objetivo es manipular activamente el comportamiento de cualquier sistema dinámico. Actualmente, con el desarrollo de nuevas tecnologías para dispositivos basados en las leyes de la mecánica cuántica, el crecimiento de técnicas de control aplicadas a estos sistemas se ha convertido en el foco tecnológico actual.

El principal problema es que ciertas propiedades de la Mecánica Cuántica no permiten transferir fácilmente los esquemas de control clásicos al dominio cuántico. Tradicionalmente, toda la información necesaria se puede obtener y emplear de manera efectiva para retroalimentar el sistema y especializar las acciones de control al estado actual. Por el contrario, en un sistema cuántico, no toda la información se puede obtener mediante mediciones y, además, cualquier intento de medir el sistema lo perturbará enormemente. Técnicamente, las ecuaciones dinámicas en tal proceso son ecuaciones diferenciales estocásticas no lineales en lugar de ecuaciones diferenciales deterministas para sistemas clásicos.

En este trabajo se pretende controlar un qubit mediante técnicas de control estocástico. En particular se diseñarán esquemas de control realimentado que serán implementados en MATLAB®. Finalmente, se hará una discusión de los resultados obtenidos.

Abstract

Villalba Celdrán, Yolanda. *"Advanced control of a qubit"*. Cartagena, September 2022.

Control is an interdisciplinary field between engineering and mathematics whose objective is to actively manipulate the behaviour of any dynamic system. Currently, with the development of new technologies for devices based on the laws of Quantum Mechanics, the growth of control techniques applied to these systems has become the current technological focus.

The main problem is that certain properties of Quantum Mechanics do not allow classical control schemes to be easily transferred to the quantum domain. Traditionally, all the necessary information can be obtained and used effectively to feedback the system and specialise the control actions to the current state. In contrast, in a quantum system, not all information can be obtained by measurement and, moreover, any attempt to measure the system will greatly perturb it. Technically, the dynamic equations in such a process are non-linear stochastic differential equations instead of deterministic differential equations for classical systems.

The purpose of this work is to control a qubit by means of stochastic control techniques. Specifically, feedback control schemes will be designed and implemented in MATLAB®. Finally, a discussion of the results obtained will be made.

Índice de contenidos

Índice de figuras

Capítulo 1

Introducción y objetivos

A lo largo de la historia, el ser humano ha desarrollado diversas tecnologías a medida que aprendía sobre el comportamiento de la naturaleza gracias a la ciencia. Entre los años 1900 y 1930, el estudio de algunos fenómenos físicos que aún no se comprendían totalmente dio lugar a una nueva teoría física, la Mecánica Cuántica. Esta teoría describe y explica el funcionamiento del mundo microscópico, hábitat natural de moléculas, átomos y los electrones. No sólo ha sido capaz de explicar estos fenómenos, sino que también ha permitido comprender que la realidad subatómica funciona de forma totalmente contraintuitiva, y que en el mundo microscópico se producen fenómenos que no se dan en el mundo macroscópico.

Gracias a la Mecánica Cuántica y a los diversos estudios realizados, en los próximos años se avanzará de la era digital a la era cuántica, una tecnología revolucionaria que empezó a desarrollarse en la década de los ochenta, que podría impulsar ámbitos tan importantes como la medicina, la inteligencia artificial y la ciberseguridad. Además, el uso de máquinas aún más potentes de las que hay actualmente en el mercado podría permitir una importante reducción de las emisiones, poniendo al alcance el objetivo de limitar el calentamiento global [1].

La computación cuántica es un campo científico interdisciplinar en alza que aprovecha las leyes de la Mecánica Cuántica para producir un rendimiento exponencialmente mayor en ciertos tipos de cálculos, lo que ofrece la posibilidad de grandes avances en varios mercados finales. Cabe destacar que, los ordenadores clásicos se adaptan mejor al procesamiento cotidiano y sencillo, mientras que las máquinas cuánticas permiten operaciones lógicas más complejas.

La principal forma de aumentar la capacidad de cálculo de los ordenadores clásicos es el procesamiento paralelo. Aquellos ordenadores que soportan este esquema de programación poseen cientos o miles de procesadores. La capacidad de almacenamiento de información y la capacidad de cómputo de un ordenador son proporcionales al número de unidades de almacenamiento y al número de procesadores, respectivamente, es decir, proporcional al tamaño. Por lo tanto, la capacidad de un

ordenador clásico, de almacenamiento y cálculo, crece linealmente en relación con su tamaño.

En un ordenador cuántico, la situación cambia totalmente, de modo que su capacidad crece exponencialmente en relación con su tamaño. Este hecho está íntimamente relacionado con el principio de superposición de la Mecánica Cuántica, conocido como paralelismo cuántico. Se denominan qubits o bits cuánticos a sistemas cuánticos elementales, es decir, sistemas cuánticos obtenidos a partir de dos estados. Un sistema cuántico de n qubits se describe mediante un vector de un espacio de Hilbert complejo de dimensión 2^n. Esto permite codificar una cantidad exponencial de información en el estado de un sistema cuántico de n qubits. Además, cualquier transición del estado del sistema da como resultado la modificación simultánea de toda la información almacenada. Por lo tanto, la capacidad de almacenamiento y computación de una computadora cuántica crece exponencialmente en relación con su tamaño.

No obstante, la medición de estados cuánticos es una gran desventaja de la computación cuántica. Esto es debido principalmente a que las medidas cuánticas no son deterministas, por lo que, si medimos dos estados iguales, los resultados no tienen por qué ser iguales. Por lo tanto, el proceso de medición es un experimento aleatorio en el que la probabilidad de cada resultado está determinada por el estado del sistema.

A diferencia de las computadoras clásicas, las cuánticas son dispositivos analógicos. Este hecho plantea mayores dificultades para la construcción de computadoras cuánticas que las que enfrentan las computadoras clásicas. En primer lugar, el entorno cambia el estado cuántico. Esto puede causar que un fenómeno llamado decoherencia sea una fuente de error. En segundo lugar, la imprecisión de las propias computadoras cuánticas al aplicar algoritmos constituye una nueva fuente de error. Estas son las razones fundamentales por las que la computación cuántica necesita un mecanismo para limitar la acumulación de errores durante el cálculo. Para ello, se deben superar importantes dificultades: los errores cuánticos son continuos, los estados no se pueden replicar y, hasta el final, la información codificada en estados cuánticos no se puede medir. Por último y no menos importante, los sistemas cuánticos, por su propia existencia en el universo, son sistemas abiertos [2]. Es por ello por lo que en este trabajo se estudian sistemas cuánticos abiertos.

La realización de este trabajo fin de estudios se ha visto motivada por varios aspectos. En primer lugar, se trata de una continuación de los trabajos iniciados por Sebastián García [3], en donde en su trabajo fin de máster realizó la simulación y control de un qubit perteneciente a un sistema global con entorno. Para ello, resultó fundamental

comprender en qué se basa la Teoría de Sistemas Cuánticos Abiertos, donde engloba el conjunto de herramientas y recursos necesarios que ha posibilitado contextualizar el problema y comprender el modelo del qubit objetivo sometido a diferentes entornos que lo perturban.

Por otro lado, la realización de este proyecto se ha visto influenciada además por la situación actual. Recientemente, dos grupos de investigación han logrado que los qubits trabajen a temperaturas entre 10 y 15 veces superiores a las habituales, que, hasta ahora, se situaban apenas por encima del cero absoluto. Este aumento de temperatura supondría reducir drásticamente el costo de la unidad de refrigeración. La tecnología de enfriamiento es una gran barrera para la construcción computadoras cuánticas de mayor rendimiento y, en el futuro, podría combinarse con dispositivos de control para agregar calor a un sistema de computación cuántica y evitar que opere a las temperaturas que hasta ahora han sido necesarias [4].

Últimamente, las tecnologías cuánticas han recibido mucha atención pública. Son cada vez más gobiernos los que han puesto en marcha grandes programas de investigación sobre tecnologías cuánticas, como el programa chino, que incluye el lanzamiento de un satélite y la instalación de un enlace de distribución de claves cuánticas entre Pekín y Shangai, o la iniciativa europea Quantum Technologies Flagship, que suma varios miles de millones de euros de financiación pública para este campo en todo el mundo. Al mismo tiempo, grandes empresas multinacionales, como Google, IBM, Intel, Microsoft y Toshiba, han empezado a invertir en tecnologías cuánticas, especialmente en computación y comunicación cuánticas (Figura 1). Además, en la última década se han creado varias empresas que ofrecen con éxito las tecnologías cuánticas a mercados especializados [5].

Un factor de éxito para el rápido avance de las Tecnologías Cuánticas es una comunidad de investigación mundial bien alineada y con una comprensión común de los retos y objetivos. En Europa, esta comunidad se ha beneficiado de varios proyectos de coordinación financiados por la CE que, entre otras cosas, han coordinado la creación de una hoja de ruta de las tecnologías cuánticas.

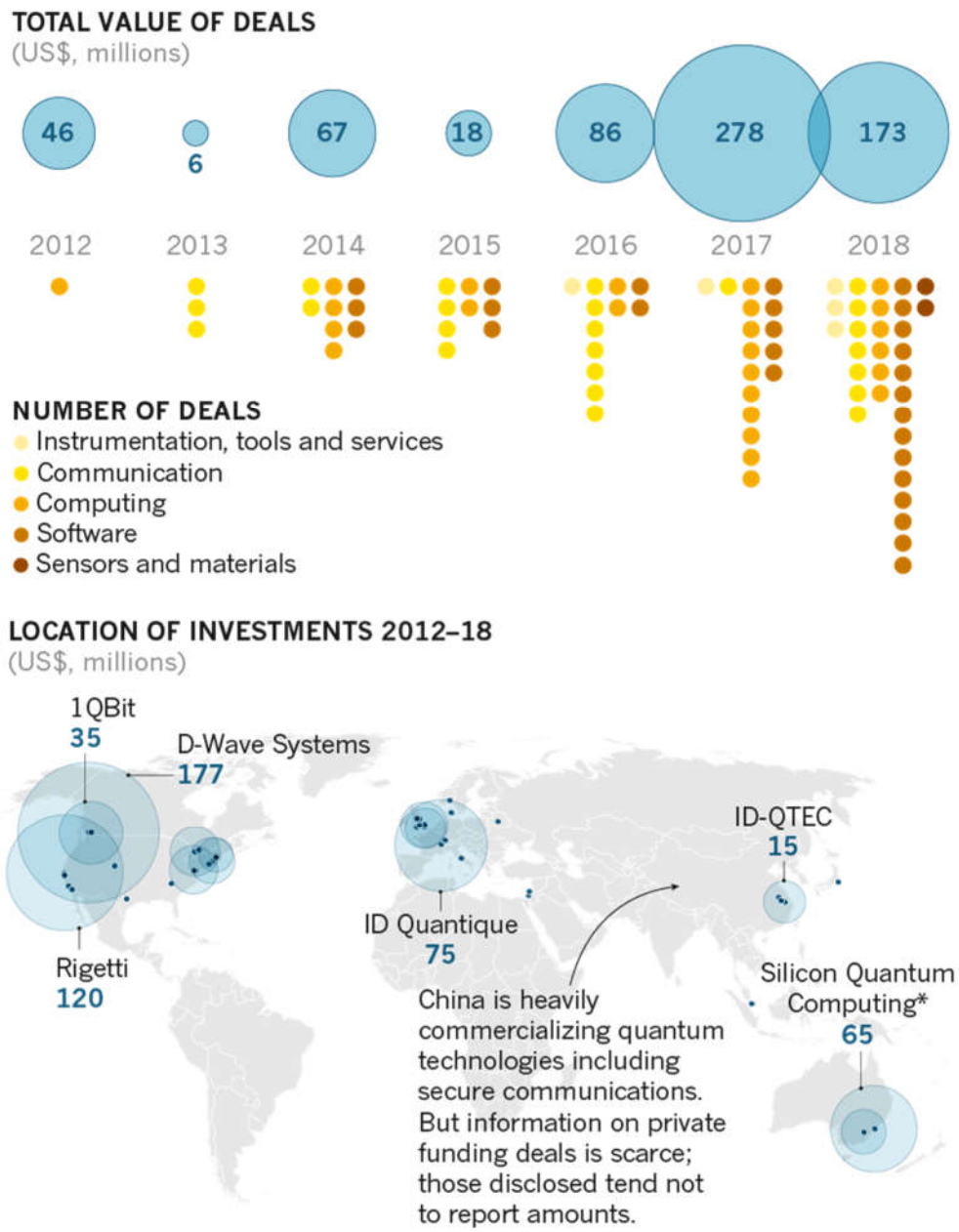

Figura 1. Capital invertido para la investigación de la computación y comunicación cuánticas. Fuente [6].

Considerando todo lo mencionado anteriormente, el objetivo de este trabajo fin de estudios es realizar una investigación, simulación y diseño de los problemas de control en sistemas cuánticos desde una perspectiva teórica, siendo como tal una continuación al trabajo fin de máster de Sebastián García [3]. En tales sistemas, la existencia de disipación e incoherencia hace imposible manipularlos arbitrariamente. Cuando estos efectos no son evitables, resulta fundamental aplicar protocolos de control.

Para llevar a cabo estos objetivos propuestos, la memoria se ha organizado en cinco capítulos. En el primero de ellos se ha realizado una introducción de las distintas

aplicaciones de la Mecánica Cuántica, su repercusión social y económica y los beneficios que se obtendrían si se abordara con éxito su implementación.

En el segundo capítulo, se aborda una introducción a los sistemas cuánticos, una explicación del qubit, ecuación de estado y ecuación diferencial estocástica. Además, se dará a conocer la representación de la esfera de Bloch y algunas propiedades de las matrices de Pauli, que permitirán expresar ecuaciones en el dominio complejo de los sistemas en el espacio tridimensional real.

En el tercer capítulo, se hace una exposición y justificación de la metodología utilizada en el diseño e implementación del qubit, desarrollando y detallando cada aspecto fundamental en la Teoría de Sistemas Cuánticos Abiertos. De esta manera, se expone la ecuación maestra de Lindblad y el modelo de obtención del modelo del qubit con el que se ha trabajado, así como las distintas leyes control a la que hemos sometido el modelo del qubit.

En el cuarto capítulo, se realiza una valoración global de los distintos controladores y métodos de resolución y se presentan y discuten los resultados obtenidos a partir de las figuras que permitan ilustrarlos.

Finalmente, se exponen las conclusiones obtenidas de realización de este trabajo, así como sus posibles vías de extensión.

Capítulo 2

Revisión del Control Cuántico

La ciencia de la información cuántica es una disciplina que aborda cómo pueden explotarse los sistemas cuánticos para mejorar el procesamiento, la transmisión y el almacenamiento de la información. Este campo ha fomentado nuevos experimentos y puntos de vista novedosos sobre los fundamentos conceptuales de la Mecánica Cuántica, y también ha inspirado gran parte de la investigación actual sobre los fenómenos cuánticos coherentes, en la que los sistemas ópticos cuánticos desempeñan un papel destacado.

En la teoría del control cuántico los sistemas a controlar son sistemas cuánticos cuya dinámica se rige por las leyes de la Mecánica Cuántica, ya que proporciona un marco matemático para describir los estados y la evolución de los sistemas cuánticos. En gran parte de la teoría de control cuántico, la controlabilidad de los sistemas cuánticos es una cuestión fundamental. Esta cuestión se refiere a si se puede conducir un sistema cuántico a un estado deseado. Este problema tiene importancia práctica, ya que está estrechamente relacionado con la universalidad de la computación cuántica y la posibilidad de lograr transformaciones a escala atómica o molecular [7].

En este capítulo se han recogido todas aquellas definiciones, teoremas y proposiciones necesarias para la correcta comprensión de los desarrollos teóricos que se han implementado en los capítulos posteriores.

2.1. Introducción a la ecuación de estado y ecuación diferencial estocástica

La ingeniería de control tiene como objetivo gobernar sistemas físicos adhiriéndose a un conjunto de especificaciones o requisitos objetivos mediante el uso de leyes matemáticas. Un sistema regido por estas leyes es lo que usualmente se le denomina sistema controlado, y es capaz de redirigirse a un valor deseado o de referencia para que el sistema dinámico funcione de manera óptima [8].

El comportamiento de un sistema lineal de orden n puede definirse mediante n ecuaciones diferenciales de primer orden. La forma matricial que expresa estas ecuaciones diferenciales es lo que se llama modelo de espacio de estados (2.1), en el que además de la entrada $u(t)$ y la salida $y(t)$ hay un conjunto de variables $x(t)$ que se llaman los estados del sistema.

$$\begin{cases} \dot{x}(t) = A(t)x(t) + B(t)u(t), & \text{ecuación de estado,} \\ y(t) = C(t)x(t) + D(t)u(t), & \text{ecuación de salida.} \end{cases} \tag{2.1}$$

Este modelo es particularmente eficaz utilizando resoluciones numéricas, permitiendo así la obtención de la reacción temporal de un sistema dinámico. De esta manera, podemos imaginar que para el caso de sistemas no lineales puede ser favorable el uso de este tipo de resolución.

En el control clásico se dispone de información del sistema, que es información que permite dirigir el sistema al estado deseado a través de acciones de control. Sin embargo, las peculiaridades de los sistemas cuánticos no permiten obtener toda la información por medición. Además, el hecho de la medición interfiere con el sistema de manera significativa, lo que lleva a la completa decoherencia (o pérdida de información y propiedades cuánticas) del sistema, incluso en el caso de realizar mediciones clásicas en el tiempo. Por lo general, las ecuaciones que modelan los sistemas dinámicos cuánticos observados son ecuaciones diferenciales estocásticas no lineales, por lo que se comportan y se tratan de manera diferente a las ecuaciones diferenciales deterministas asociadas con los sistemas clásicos.

Una ecuación diferencial estocástica (SDEs - *Stochastic Differential Equations*) es una ecuación diferencial en la que uno o más de los elementos y su solución son procesos estocásticos. Estas ecuaciones se utilizan para simular varios fenómenos, como precios de acciones volátiles o sistemas físicos afectados por fluctuaciones térmicas. En física, estas ecuaciones tienen una aplicabilidad aún más amplia, desde la dinámica molecular hasta la neurodinámica y la dinámica de los objetos astrofísicos. Más específicamente, las SDEs describen todos los sistemas dinámicos en los que los sistemas cuánticos son sometidos a medidas continuas, y más en general sistemas sometidos a perturbaciones aleatorias. En general, pueden verse como una generalización de la teoría del sistema dinámico a los modelos de ruido, ya que los sistemas reales no pueden estar completamente aislados de su entorno, por lo que siempre están sujetos a influencias aleatorias externas.

Para comprender mejor el modelo de una SDE primero se hará una breve introducción de cómo se obtienen estas, que se realiza a partir de las ecuaciones diferenciales ordinarias (ODE – *Ordinary Differential Equation*).

$$\frac{dx_t(t)}{dt} = f(t, x). \tag{2.2}$$

Como se puede observar, en una ODE no tenemos en cuenta la aleatoriedad. Podemos escribir (2.2) en la forma diferencial simbólica,

$$dx_t(t) = f(t, x)\, dx. \tag{2.3}$$

También se puede reescribir de una forma más precisa con la forma integral,

$$x(t) = x_0 + \int_0^t f\big(s,x(s)\big)\, ds, \tag{2.4}$$

donde $x(t) = x(t, x_0, t_0)$ es la solución que satisface la condición inicial dada $x(t_0) = x_0$.

Una SDE generalmente contiene una variable que representa el ruido blanco aleatorio, calculado como la derivada del movimiento browniano o proceso de Wiener. La idea de una ODE puede ampliarse a una SDE añadiendo ruido al sistema considerado. Por ejemplo, consideremos el siguiente sistema,

$$\frac{dX(t)}{dt} = f(t)X(t) + h(t)X(t)\xi(t), \tag{2.5}$$

donde ξ(t) es lo que hemos definido anteriormente como ruido blanco del proceso. De otra manera, se puede reescribir la ecuación (2.5), de manera que se sustituye el ruido blanco por $dW(t) = \xi(\mathrm{t})dt$, donde $dW(t)$ representa la forma diferencial de un movimiento Browniano estándar.

$$dX(t) = f(t)X(t) + h(t)X(t)dW(t). \tag{2.6}$$

De esta manera, se puede escribir la forma general de una SDE,

$$\begin{cases} dX(t) = f\big(X(t)\big)dt + g\big(X(t)\big)dW, \quad t > 0, \\ X(0) = X_0, \end{cases} \tag{2.7}$$

donde $f\big(X(t)\big) \in \mathbb{R},\ g\big(X(t)\big) \in \mathbb{R},\ dW \in \mathbb{R}$.

Se puede observar que este sistema guarda similitud con el definido en (2.1) del espacio de estados. La combinación de ambas ecuaciones da un conjunto de ecuaciones que permiten controlar el SDE,

$$\begin{cases} dx(t) = \big(A(t)x(t) + B(t)u(t)\big)dt + g\big(x(t)\big)dW, \\ y(t) = C(t)x(t) + D(t)u(t). \end{cases} \tag{2.8}$$

Existen diferentes métodos para resolver las ecuaciones diferenciales estocásticas, como el método de Euler – Maruyama, el método de Milstein y el método de Runge – Kutta [9].

2.2. Introducción a los sistemas cuánticos

El objetivo de este apartado es entender cómo funcionan los bits cuánticos: cuáles son sus estados, cuáles son las operaciones permitidas sobre un bit cuántico y cómo extraer información de un bit cuántico mediante una medición. Pero antes de hablar de los bits cuánticos, es útil entender primero los bits.

Los bits pueden almacenarse como tal en un dispositivo de almacenamiento digital, utilizando los dígitos binarios 0 y 1 para describir algunas propiedades físicas reales, como la posición y el momento, que son generalmente observables. En cambio, los bits cuánticos, llamados qubits, pueden estar en un estado de $|0\rangle$ y $|1\rangle$, que, corresponden a los estados 0 y 1 de un bit clásico y en un estado de superposición combinación lineal de estos estados básicos, en donde la fase global fase de un estado cuántico no tiene ningún efecto físico observable.

En la Mecánica Cuántica, el estado de un sistema cuántico cerrado puede representarse mediante un vector unitario $|\psi\rangle$, donde la notación "$|\ \rangle$" se conoce como representación de Dirac, y se trata de la notación estándar para los estados en la Mecánica Cuántica [10].

2.2.1. Los postulados de la Mecánica Cuántica

La mecánica cuántica es un marco matemático y conceptual para el desarrollo de teorías físicas, en donde por sí misma, no indica qué leyes debe obedecer un sistema físico. Los postulados de la mecánica cuántica se dedujeron tras un largo proceso de ensayo y error

(en su mayoría), que implicó una cantidad considerable de conjeturas y tanteos por parte de los creadores de la teoría.

Este apartado está dedicado a revisar las herramientas matemáticas de la mecánica cuántica y a presentar una reformulación moderna de los postulados básicos que es adecuada para describir sistemas cuánticos. De esta manera, los postulados de la mecánica cuántica pueden agruparse y resumirse como se muestran a continuación ([10] y [11]).

Postulado 1 (Espacio de estados). A cualquier sistema físico aislado se le asocia un espacio vectorial complejo con producto interior, es decir, un espacio de Hilbert, H, conocido como el espacio de estado del sistema. Cualquier sistema está completamente descrito por su vector de estado, que es un vector unitario en el espacio de estado del sistema.

Supongamos que $|0\rangle$ y $|1\rangle$ forman una base ortonormal para ese espacio de estados. Entonces un vector de estado arbitrario en el espacio de estado espacio de estado se puede escribir como,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2.9}$$

donde $|\psi\rangle \in \mathrm{H}$, mientras que α y β son amplitudes de probabilidad del estado de superposición, que se emplean para conocer cómo se comporta el sistema y satisfacen que $\alpha, \beta \in \mathbb{C}$, $|\alpha|^2 + |\beta|^2 = 1$, ya que las probabilidades deben dar como resultado la unidad, mientras que $|0\rangle$ y $|1\rangle$ son los estados posibles. De esta manera,

- $|\alpha|^2$: Señala la probabilidad, cuando se realiza la medición, de encontrar $|\psi\rangle$ en el estado $|0\rangle$.
- $|\beta|^2$: Señala la probabilidad, cuando se realiza la medición, de encontrar $|\psi\rangle$ en el estado $|1\rangle$.

La condición de que $|\psi\rangle$ sea un vector unitario $\langle\psi|\psi\rangle = 1$, es por tanto equivalente a $|\alpha|^2 + |\beta|^2 = 1$. Esta condición $\langle\psi|\psi\rangle = 1$ suele denominarse la *condición de normalización* de los vectores de estado.

La diferencia entre los bits y los qubits es que un qubit puede estar en un estado distinto de $|0\rangle$ y $|1\rangle$. Un qubit también puede estar en una superposición de estados $|0\rangle$ y $|1\rangle$, dicho de otra manera, como combinación lineal de vectores base,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha\begin{pmatrix}1\\0\end{pmatrix} + \beta\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}\alpha\\\beta\end{pmatrix}. \tag{2.10}$$

La capacidad de un qubit de estar en un estado de superposición va en contra de nuestra comprensión de "sentido común" del mundo físico que nos rodea. Un bit clásico es como una moneda: o cara o cruz. En el caso de las monedas imperfectas, puede haber estados intermedios, como el de estar en equilibrio sobre un borde, pero estos pueden ignorarse en el caso ideal. En cambio, un qubit puede existir en un continuo de estados entre $|0\rangle$ y $|1\rangle$ hasta que es observado. De esta manera, al realizar la medición del qubit sólo podemos obtener como resultado un "0" o un "1", de forma probabilística. Por ejemplo, un qubit puede estar en el estado,

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle, \tag{2.11}$$

que, cuando se mide, la probabilidad de obtener un “0” es del cincuenta por ciento $\left(\left|1/\sqrt{2}\right|^2\right)$ de las veces, y, de la misma forma, la probabilidad de obtener un “1” el cincuenta por ciento de las veces.

En las aplicaciones prácticas, los sistemas cuánticos que hay que controlar no suelen ser simples sistemas cerrados. Pueden ser conjuntos o sistemas cuánticos abiertos y sus estados no pueden escribirse en forma de vectores unitarios. En este caso, es necesario introducir el operador de densidad o la matriz de densidad ρ: H → H para describir estados cuánticos de conjuntos o sistemas cuánticos abiertos [7].

La matriz densidad fue desarrollada por von Neumann y nos permite manejar fácilmente estados que no pueden ser descritos por una sola función de onda. Supongamos que un sistema cuántico se encuentra en un conjunto $\{\rho_j, |\psi_j\rangle\}$ de estados puros, es decir, en una mezcla de un número de estados puros $|\psi_j\rangle$ con probabilidades respectivas ρ_j. La matriz de densidad para el sistema se define como sigue a continuación,

$$\rho \equiv \sum_j \rho_j |\psi_j\rangle\langle\psi_j|, \tag{2.12}$$

donde $\langle\psi_j| = (|\psi_j\rangle)^\dagger$ y $\sum_j \rho_j = 1$, siendo el operador $(.)^\dagger$ el adjunto. Para un estado puro $|\psi\rangle$, $\rho = |\psi\rangle\langle\psi|$ y $Tr(\rho^2) = 1$. Si se da el caso de que el estado ρ de un sistema cuántico satisface que $Tr(\rho^2) < 1$, entonces a esto se le llama estado mixto.

La traza de una matriz es la suma de sus elementos diagonales. Particularizando para la matriz densidad, esta es igual a la unidad, ya que estas componentes son los

coeficientes o magnitudes del estado cuántico normalizado. Por lo tanto, podemos representar el estado de un qubit de la ecuación (2.9) con la matriz de densidad.

$$\rho = |\psi\rangle\langle\psi| = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}(\alpha^*\beta^*) = \begin{pmatrix}|\alpha|^2 & \alpha\beta^*\\ \alpha^*\beta & |\beta|^2\end{pmatrix}. \tag{2.13}$$

Tomando un ejemplo, si el estado $|\psi\rangle$ fuese $|0\rangle$, obtenemos que la matriz densidad resultante es,

$$\rho = |\psi\rangle\langle\psi| = \begin{pmatrix}1\\ 0\end{pmatrix}(1 \;\; 0) = \begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}. \tag{2.14}$$

De esta manera, se comprueba que la matriz densidad nos ofrece la misma información que la forma $|\psi\rangle$, por lo que se suele utilizar debido a su facilidad. Sin embargo, en mecánica cuántica, un estado mezcla es un estado que no está totalmente determinado, a diferencia con un estado puro. Sabiendo esto, suponiendo que para n qubits donde P_1 representa la probabilidad de obtener el estado $|\psi_1\rangle$ y P_2 la probabilidad de obtener $|\psi_2\rangle$, la matriz densidad se puede expresar como,

$$\rho = P_1|\psi_1\rangle\langle\psi_1| + P_2|\psi_2\rangle\langle\psi_2|, \tag{2.15}$$

donde la suma de las probabilidad dan lugar a la unidad, $P_1 + P_2 = 1$. Simplificando (2.15), de forma que $|\psi_1\rangle = |0\rangle$ y que $|\psi_2\rangle = |1\rangle$, se tiene que,

$$\begin{aligned}\rho = P_1|0\rangle\langle 0| + P_2|1\rangle\langle 1| &= P_1\begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix} + P_2\begin{pmatrix}0 & 0\\ 0 & 1\end{pmatrix}\\ &= \begin{pmatrix}P_1 & 0\\ 0 & P_2\end{pmatrix}.\end{aligned} \tag{2.16}$$

La matriz densidad es de gran importancia en el campo de la mecánica cuántica debido a que recoge en un solo operador las probabilidades de los subestados de los estados mezcla.

Postulado 2 (Medidas observables). Todas las mediciones posibles en un sistema cuántico se describen mediante un operador hermítico u observable. Sabiendo que cualquier observable O tiene la siguiente forma,

$$O = \sum_i a_i|a_i\rangle\langle a_i|, \tag{2.17}$$

donde $a_i \in \mathbb{R}$ son los valores propios del observable y $|a_i\rangle$ sus correspondientes vectores propios. La probabilidad de obtener el resultado a_i al medir la propiedad descrita por el observable O en un estado $|\psi\rangle$ viene dada por,

$$P(a_i) \;=\; |\langle\psi|a_i\rangle|^2. \tag{2.18}$$

Después de la medición, obtenemos el estado $|a_i\rangle$ si el resultado a_i fue medido. Esto nos permite calcular las posibles salidas de un sistema, la probabilidad de estos resultados, así como el estado después de la medición. Cabe destacar que, tras realizar una medición, el estado suele cambiar, ya que este solo puede permanecer inalterado si ya se encontraba en un estado propio del observable.

Es posible calcular el valor deseado de una medición definida por el operador O en un estado $|\psi\rangle$ con sólo aplicar la fórmula simple,

$$\langle O\rangle = \langle\psi|O|\psi\rangle. \tag{2.19}$$

Para el caso de los estados mixtos, la probabilidad de obtener una salida que corresponda a un vector propio $|a_i\rangle$ es,

$$P(a_i) = Tr\big[|a_i\rangle\langle a_i|\rho\big], \tag{2.20}$$

siendo el valor deseado del operador O,

$$\langle O\rangle = Tr[O\rho]. \tag{2.21}$$

Postulado 3 (Evolución en el tiempo). La evolución de un sistema cuántico cerrado se describe mediante una transformación unitaria. Es decir, el estado $|\psi\rangle$ del sistema en el tiempo t_1 está relacionado con el estado $|\psi\rangle$ del sistema en el tiempo t_2 por un operador unitario U que depende sólo de los tiempos t_1 y t_2,

$$\tfrac{d}{dt}|\psi\rangle \;= U|\psi\rangle. \tag{2.22}$$

Al igual que la mecánica cuántica no nos dice el espacio de estados o el estado cuántico de un sistema cuántico concreto, tampoco nos dice qué operadores unitarios U describen la dinámica cuántica del mundo real. La Mecánica Cuántica se limita a asegurar que la evolución de cualquier sistema cuántico cerrado puede describirse de esa manera.

Otro operador unitario interesante es la puerta de Hadamard, que denotamos como H. Dada una base ortonormal, cualquier estado puro $|\psi\rangle$ de los vectores $|0\rangle$ y $|1\rangle$ de un sistema cuántico de dos niveles puede escribirse como una superposición de vectores base, cada uno de los cuales tiene un coeficiente o magnitud complejo. No obstante, debido a que solo la fase relativa entre los coeficientes de los dos vectores base tiene algún

significado físico, podemos tomar los coeficientes de $|0\rangle$ como números reales en lugar de números negativos.

$$\begin{aligned} |+\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = H|0\rangle, \\ |-\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = H|1\rangle. \end{aligned} \tag{2.23}$$

Siendo la matriz correspondiente expresada como,

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{2.24}$$

Este postulado requiere que el sistema descrito sea cerrado. Es decir, que no interactúe de ninguna manera con otros sistemas. En la realidad, por supuesto, todos los sistemas, excepto el Universo en su conjunto, interactúan al menos de alguna manera con otros sistemas. Sin embargo, hay sistemas interesantes que pueden describirse con una buena aproximación como cerrados, y que se describen mediante la evolución unitaria con alguna buena aproximación. Además, todo sistema abierto puede describirse como parte de un sistema cerrado más grande, el Universo, que experimenta una evolución unitaria.

El postulado 3 describe cómo se relacionan los estados cuánticos de un sistema cuántico cerrado en dos momentos diferentes. Se puede dar una versión más refinada de este postulado que describe la evolución de un sistema cuántico en *tiempo continuo*. Antes de enunciar el postulado revisado, conviene que el operador H que aparece a continuación no es el mismo que el operador de Hadamard, que acabamos de introducir.

Postulado 3'. La evolución temporal del estado de un sistema cuántico cerrado se describe mediante la ecuación de Schrödinger,

$$i\hbar \frac{\partial}{\partial t}|\psi(t)\rangle = H_0|\psi(t)\rangle, \quad |\psi(t = 0)\rangle = |\psi_0\rangle, \tag{2.25}$$

donde H_0 es el Hamiltoniano libre del sistema, es decir, un operador hermítico sobre H, $\hbar$ es la constante de Planck reducida, y el estado inicial tiene norma unitaria $\|\psi_0\|^2 = \langle\psi_0|\psi_0\rangle = 1$. Para simplificar, consideramos sistemas cuánticos de dimensión finita, que son aproximaciones apropiadas en muchas situaciones prácticas.

La evolución temporal de la matriz de densidad, conocida como ecuación de von Neumann se puede obtener aplicando la ecuación de Schrödinger,

$$\begin{aligned} i\hbar\frac{\partial}{\partial t} &= i\hbar\left(\frac{\partial}{\partial t}|\psi\rangle\right)\langle\psi| + i\hbar|\psi\rangle\left(\frac{\partial}{\partial t}|\psi\rangle\langle\psi|\right) \\ &\iff H|\psi\rangle\langle\psi| - |\psi\rangle\langle\psi|H = [H,\rho]. \end{aligned} \tag{2.26}$$

Postulado 4 (Sistemas compuestos). El espacio de estado de un sistema físico compuesto, formado por N subsistemas, es el producto tensorial del espacio de estado de cada componente $\mathrm{H} = \mathrm{H}_1 \otimes \mathrm{H}_2 \ldots \otimes \mathrm{H}_N$. El estado del sistema físico compuesto viene dado por un vector unitario de H.

Un sistema cuántico compuesto formado por dos subsistemas A y B se define en un espacio de Hilbert $\mathrm{H} = \mathrm{H}_A \otimes \mathrm{H}_B$, que es el producto tensorial de los espacios de Hilbert H_A y H_B en los que se definen los subsistemas A y B. Para el sistema cuántico compuesto, su estado ρ_{AB} puede describirse mediante el producto tensorial de los estados de sus subsistemas; es decir, $\rho_{AB} = \rho_A \otimes \rho_B$.

Considerando cualquier estado puro bipartito $|\psi\rangle_{AB}$, este se puede escribir como un producto tensorial de estados puros $|\varphi\rangle_A \in \mathrm{H}_A$ y $|\chi\rangle_B \in \mathrm{H}_B$,

$$|\psi\rangle_{AB} = |\varphi\rangle_A \otimes |\chi\rangle_B \tag{2.27}$$

A esto lo llamamos estado separable; en caso contrario, lo llamamos estado entrelazado. El entrelazamiento cuántico es un fenómeno exclusivamente mecánico cuántico que desempeña un papel clave en muchas aplicaciones interesantes de la comunicación y la computación cuánticas, y se puede encontrar una discusión detallada en.

Todo esto lleva al concepto de definir una matriz de densidad reducida para estudiar propiedades. Si tenemos el sistema compuesto anterior de A y B, nos puede interesar estudiar algunas propiedades del subsistema correspondiente a uno de los subespacios, para ello, definimos la matriz de densidad reducida. Si el estado de nuestro sistema está descrito por una matriz de densidad ρ, el operador de densidad reducida del subsistema está definido por el operador $\rho_A = Tr_B[\rho]$, donde Tr_B es la traza parcial sobre el subespacio B y es definido como,

$$\mathrm{Tr}_B\left[\sum_{i,j,k,l}|A_i\rangle\langle A_j|\otimes|B_k\rangle\langle B_l|\right]=\sum_{i,j}|A_i\rangle\langle A_j|\,\mathrm{Tr}\left[\sum_{k,l}|B_k\rangle\langle B_l|\right] \tag{2.28}$$

Los conceptos de matriz de densidad reducida y trazado parcial son esenciales en el estudio de los sistemas cuánticos abiertos. Si queremos calcular la ecuación de movimientos de un sistema afectado por un entorno, debemos trazar este entorno y tratar sólo la matriz de densidad reducida del sistema. Esta es la idea principal de la teoría de los sistemas cuánticos abiertos.

2.2.2. Modelo de control en la esfera de Bloch

Anteriormente, se ha explicado lo que representa el operador densidad, conociéndose que ρ es un operador hermítico. La matriz de densidad ρ para un qubit es una matriz compleja hermitiana de dimensión 2x2, por lo que puede expandirse sobre la base $\mathscr{B}=\{I,\sigma_x,\sigma_y,\sigma_z\}$, siendo σ_i las matrices de Pauli. Esta base es ortogonal debido al producto interno de Hilbert – Schmidt, $\langle A,B\rangle_{HS}=Tr(AB)$, siendo A y B operadores de Hilbert – Schmidt. De esta manera, la matriz densidad se expresa como,

$$\rho = aI + c\sigma_x + d\sigma_y + b\sigma_z, \tag{2.29}$$

donde $a,b,c,d\in\mathbb{R}$. Usaremos las matrices de Pauli más adelante para controlar de alguna manera los qubits que han sido objetos de estudio, por lo que, a continuación, se realizará un inciso sobre estas.

En Mecánica Cuántica, cada matriz de Pauli (σ_x, σ_y y σ_z) está relacionada con un operador de momento angular que corresponde a un observable que describe el espín de una partícula en cada una de las tres direcciones espaciales, además, se utilizan en muchas otras áreas de la física, incluida la computación cuántica.

$$\sigma_x=\begin{pmatrix}0&1\\1&0\end{pmatrix},\qquad \sigma_y=\begin{pmatrix}0&-i\\i&0\end{pmatrix},\qquad \sigma_z=\begin{pmatrix}1&0\\0&-1\end{pmatrix}. \tag{2.30}$$

Estos operadores satisfacen que ${\sigma_x}^2={\sigma_y}^2={\sigma_z}^2=I$, y además son operadores que no conmutativos, esto quiere decir que,

$$\begin{aligned}\sigma_x\sigma_y &= -\sigma_y\sigma_x = i\sigma_z,\\ \sigma_y\sigma_z &= -\sigma_z\sigma_y = i\sigma_x,\\ \sigma_z\sigma_x &= -\sigma_x\sigma_z = i\sigma_y.\end{aligned} \tag{2.31}$$

Estas relaciones pueden expresarse en términos de vectores tridimensionales, $a = (a_x, a_y, a_z)$, y $b = (b_x, b_y, b_z)$, de esta forma, podemos realizar un único vector σ, en el que incluye las tres componentes, σ_x, σ_y y σ_z,

$$\sigma = (\sigma_x, \sigma_y, \sigma_z). \tag{2.32}$$

El producto escalar se puede escribir como $a \cdot \sigma = a_x\sigma_x + a_y\sigma_y + a_z\sigma_z$, de forma que utilizando ${\sigma_x}^2 = {\sigma_y}^2 = {\sigma_z}^2 = I$ y los vectores a y b definidos anteriormente, podemos escribir una única relación,

$$(a \cdot \sigma)(b \cdot \sigma) = a \cdot b + i(a \times b) \cdot \sigma, \tag{2.33}$$

donde $a \times b$ representa el producto vectorial. En definitiva, los operadores σ_x, σ_y y σ_z junto con la matriz identidad I dan lugar a un pilar para el álgebra de matrices bidimensionales de números complejos, en donde cualquier matriz de este tipo resulta ser una combinación lineal de las cuatro matrices,

$$A = a_0 I + a \cdot \sigma, \tag{2.34}$$

donde $a_0, a \in \mathbb{R}$.

Considerando esta transformación, podemos redefinir la ecuación (2.29), de forma que cualquier operador de densidad bidimensional ρ puede expandirse utilizando la identidad I y las matrices de Pauli,

$$\rho = \frac{I + x\sigma_x + y\sigma_y + z\sigma_z}{2},\ x^2 + y^2 + z^2 \leq 1, \tag{2.35}$$

$(x, y, z) \in \mathbb{R}^3$ son las coordenadas del vector ortogonal $(\vec{\imath}, \vec{\jmath}, \vec{k})$ correspondientes al vector de Bloch $\vec{a} \in \mathbb{R}^3, \vec{a} = x\vec{\imath} + y\vec{\jmath} + z\vec{k}$.

Sin embargo, la matriz densidad como se ha explicado anteriormente, debe satisfacer que $Tr(\rho^2) = 1$, por lo tanto, $2a = 1 \Rightarrow a = 1/2$. Entonces, podemos obtener la forma usual de la matriz densidad como,

$$\begin{aligned} \rho &= \frac{1}{2}(I + \vec{a}\,\sigma) \\ &= \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{x}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + \frac{y}{2}\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} + \frac{z}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \\ &= \frac{1}{2}\begin{pmatrix} 1 + z & x - iy \\ x + iy & 1 - z \end{pmatrix}, \end{aligned} \tag{2.36}$$

donde $x = 2c, y = 2d, z = 2b$.

En general, considerando el caso de un sistema cuántico abierto que sufre pérdidas al exterior, este vector se encuentra en o dentro de la esfera unitaria, denominada como *esfera de Bloch*, que representa un medio útil para visualizar el estado de un solo qubit, y a menudo sirve como un excelente banco de pruebas para las ideas sobre la computación y la información cuánticas. Sin embargo, considerando el caso de un estado cuántico puro, este vector se encuentra en la esfera unitaria.

La esfera de Bloch es una esfera unitaria, con puntos antipodales que corresponden a un par de vectores de estado mutuamente ortogonales. Los polos norte y sur de la esfera de Bloch se suelen elegir para que correspondan a los vectores base estándar $|0\rangle$ y $|1\rangle$, respectivamente, que a su vez podrían corresponder, por ejemplo, a los estados de espín arriba y espín abajo de un electrón. En consecuencia, la superficie de la esfera de Bloch representa todos los estados puros de un sistema cuántico bidimensional, mientras que el interior corresponde a todos los estados mixtos.

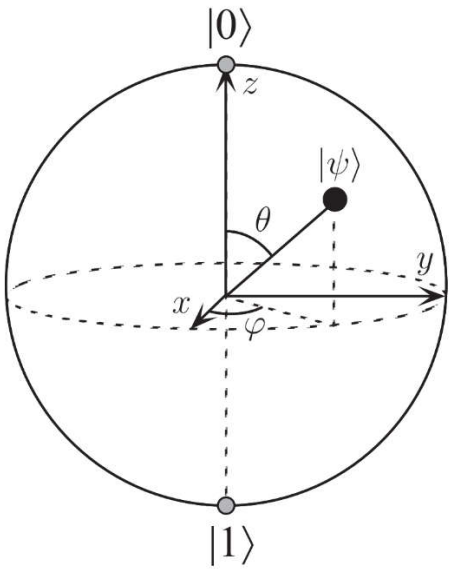

Figura 2. Representación de un qubit mediante la esfera de Bloch. Fuente: [10]

Los parámetros θ y φ reinterpretados en coordenadas esféricas vienen determinados a partir de la ecuación que define el estado del qubit en coordenadas esféricas,

$$|\psi\rangle = cos\frac{\theta}{2}|0\rangle + e^{i\varphi}sin\frac{\theta}{2}|1\rangle, \qquad 0 \leq \theta \leq \pi, \qquad 0 \leq \varphi \leq 2\pi, \quad (2.37)$$

donde θ y φ definen la colatitud con respecto al eje z y la longitud con respecto al eje x, especificando de esta manera, un punto en la esfera unidad en $\mathbb{R}^3$.

2.2.3. Estabilidad y estabilización de sistemas cuánticos

La ecuación diferencial estocástica cuántica describe la evolución de sistemas cuánticos interactuados con ruidos cuánticos fundamentales,

$$dx(t) = f(x(t), u(t), t)dt + g(x(t), u(t), t)dW, \tag{2.38}$$

donde $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^n$, $f, g: \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \Rightarrow \mathbb{R}^n$, siendo $u(t)$ el vector de control [12].

Se puede demostrar que, bajo condiciones que se especificarán más adelante, el sistema anterior, que no es necesariamente estabilizable mediante una realimentación de estado ordinaria, es localmente estabilizable mediante una ley de realimentación dinámica ordinaria de la forma,

$$u(t) = u(0) + \int_0^t k\big(x(\tau), u(\tau)\big)\, d\tau, \tag{2.39}$$

de forma que,

$$\begin{cases} dx = f(x, y)dt + g(\dot{x})d\omega, \\ dy = vdt, \end{cases} \tag{2.40}$$

donde $u = y$ y $v = k(x,y)$ es una retroalimentación de estado. Sontag y Sussman demostraron que, en general, los niveles deterministas controlables no son estabilizables por leyes de retroalimentación continuas. La teoría de la estabilidad determinista en primer lugar fue introducida por Lyapunov, y estudia el comportamiento de los sistemas físicos y dinámicos, siendo una parte esencial de los sistemas de control. Las condiciones suficientes para leyes de retroalimentación, excepto para el caso en el equilibrio, se derivaron en [13], donde también se introdujo la noción apropiada de estabilización dinámica para sistemas estocásticos.

2.2.3.1. Estabilidad estocástica

Un concepto fundamental de la ingeniería de control es la estabilidad. En los sistemas lineales, a menudo se pueden distinguir dos tipos de estabilidad: la estabilidad interna, que se basa en el comportamiento de respuesta a la entrada cero; y la estabilidad externa, que se caracteriza por la entrada; en particular, la estabilidad BIBO (BIBO – *Bounded - Input, Bounded - Output*) se define como una respuesta finita a una entrada igualmente finita ([14] y [3]).

Sin embargo, para sistemas no lineales, como los que nos ocupan en este proyecto, no siempre se aplican los criterios propuestos para sistemas lineales. Para tales sistemas más complejos, se puede realizar un análisis de estabilidad interna de acuerdo con el criterio de energía, una de las herramientas más poderosas de las cuales es la teoría de estabilidad de Lyapunov.

Si ampliamos la ecuación (2.38) en forma integral, el sistema queda como,

$$x_t = x_0 + \int_0^t f(x_s)\, ds + \int_0^t g(x_s)\, dw_s, \tag{2.41}$$

donde x_t representa la solución de la ecuación diferencial estocástica de Itô. Podemos decir que existe $K \in \mathbb{R}$, de forma que para cada $x \in \mathbb{R}$,

$$|f(x)| + |g(x)| \leq K(1 + |x|). \tag{2.42}$$

Para cada $s \geq 0$ y $x \in \mathbb{R}^n$, denotaremos por $x_t^{s,x}$, $s \leq t$, la solución en el tiempo t de la ecuación diferencial estocástica que parte del estado x en el momento s.

A continuación, se hace necesario mencionar la noción de estabilidad asintótica para la solución de equilibrio de la ecuación estocástica diferencial definida en (2.41).

Definición 1. Se dice que la solución de equilibrio $x_t \equiv 0$ de la ecuación diferencial estocástica (2.41) es *localmente estable* en términos probabilísticos si y sólo si, para cualquier $s \geq 0$ *y* $\epsilon > 0$,

$$\begin{aligned} &\lim_{x \to 0} P\left(\sup_{s \leq t} \left|x_t^{s,x}\right| > \epsilon\right) = 0, \\ &\lim_{x \to 0} P\left(\sup_{t \to \infty} \left|x_t^{s,x}\right| = 0\right) = 1. \end{aligned} \tag{2.43}$$

Denotando por L el generador infinitesimal de la solución proceso estocástico x_t de la ecuación diferencial estocástica (2.41), es decir, L es el operador diferencial de segundo orden definido para cualquier función ψ,

$$L\psi(x) = \sum_{i=1}^{n} f_i(x)\frac{\partial \psi}{\partial x_i}(x) + \frac{1}{2}\sum_{i,j=1}^{n} a_{ij}(x)\frac{\partial^2 \psi}{\partial x_i \partial x_j}(x) \tag{2.44}$$

donde $a_{ij}(x) = \sum_{k=1}^{m} g_{ik}(x)g_{kj}(x)$, $1 \leq i, j \leq n$, en donde se puede demostrar el siguiente teorema estocástico de Lyapunov.

Teorema 1. Sea D una región abierta acotada del origen en $\mathbb{R}^n$, y supongamos que existe una función de Lyapunov V definida en D, es decir, una función C^2 donde V mapea D en $\mathbb{R}$ y que es positiva definida, tal que $LV(x) < 0$, para cada $x \in D$, $x \neq 0$. Entonces la solución de equilibrio $x_t \equiv 0$ de la ecuación diferencial estocástica (2.41) es *localmente estable asintóticamente* en términos de probabilidad.

Suponiendo que la parte *drift* en (2.41) está parametrizada por un controlador u, se tiene el sistema diferencial estocástico controlado en la forma Itô,

$$dx_t = f(x_t, u)dt + g(x_t)dw_t, \tag{2.45}$$

donde,

1. $x_0 \in \mathbb{R}^n$.
2. u es la ley de control $\mathbb{R}^p$.
3. f y g son funciones de Lipschits en donde $f, g: \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \Rightarrow \mathbb{R}^n$, de forma que,
   - $f(0,0) = 0$ y $g(0) = 0$,
   - existe una constante no negativa K tal que para cualquier $x \in \mathbb{R}^n$ y $u \in \mathbb{R}^p$, $|f(x,u)| + |g(x)| \leq K(1 + |x| + |u|)$.

Entonces, la ecuación (2.45) se dice que es *local y asintóticamente estable en el origen* si el sistema diferencial estocástico mostrado a continuación,

$$d\begin{pmatrix} x_t \\ y_t \end{pmatrix} = \begin{pmatrix} f(x_t, u) \\ v \end{pmatrix}dt + \begin{pmatrix} g(x_t) \\ 0 \end{pmatrix} dw_t, \tag{2.46}$$

es localmente estabilizable por retroalimentación en el origen.

La versión estocástica del teorema de Artstein demostrada en [13], garantiza la existencia de leyes de retroalimentación de estado que, estabilizan localmente de forma asintótica la solución de equilibrio de sistemas diferenciales estocásticos de control afines en la entrada en la forma,

$$dx_t = \big(f(x_t) + uh(x_t)\big)dt + g(x_t)dw_t, \tag{2.47}$$

donde f y g son funciones ya definidas, h es otra función $h \in \mathbb{R}^n$, y u es la ley de control.

Nótese que L dada en la ecuación (2.44) es el *generador infinitesimal* asociado al sistema diferencial estocástico autónomo deducido de (2.47). Entonces, la función de control de Lyapunoy para el sistema diferencial estocástico (2.47) puede ser introducida como se define a continuación.

Definición 2. Una función V uniforme y definida positivamente que mapea $\mathbb{R}^n$ en $\mathbb{R}$ se dice que es una función de control de Lyapunov para el sistema diferencial estocástico (2.47), si y sólo si,

$$\inf_{u \in \mathbb{R}} \{LV(x) + u\nabla V(x)h(x)\} < 0, \quad \forall\, x \in \mathbb{R}^n, x \neq 0, \tag{2.48}$$

siendo $V(x)$ la función de Lyapunov asociada al sistema estocástico diferencial (2.47). Podemos definir ahora dos funciones a y b como,

$$a(x) = LV(x), \qquad b(x) = \nabla V(x)h(x), \qquad b(x) = 0 \Rightarrow a(x) < 0, \tag{2.49}$$

para todo $x \neq 0$.

De esta forma, la versión estocástica del teorema de Artstein confirma el siguiente resultado.

Teorema 2. Supongamos que V es una función de Lyapunov de control para el sistema diferencial estocástico (2.47). Entonces, la ley de retroalimentación de estado u definida para cualquier x en $\mathbb{R}^n$, se define como,

$$u(x) = \begin{cases} 0, & \text{si } x = x_d, \\ -\rho\big(a(x), b(x)\big), & \text{en otro caso,} \end{cases} \tag{2.50}$$

en donde,

$$\rho(x) = \begin{cases} 0, & \text{si } b = 0 \text{ y } a < 0, \\ \dfrac{a + \sqrt{a^2 + b^2}}{2}, & \text{en otro caso,} \end{cases} \tag{2.51}$$

siendo a y b son las funciones dadas en (2.49), y hace que el sistema diferencial estocástico (2.47) sea *localmente estable* en términos de probabilidad [13].

2.2.3.2. Condiciones de estabilización

Consideremos el sistema diferencial estocástico controlado en forma Itô, con las mismas condiciones efectuadas en el apartado anterior,

$$dx_t = f(x_t, u)dt + g(x_t)dw_t. \tag{2.52}$$

Para demostrar la condición de estabilización, se utiliza el teorema de Artstein [13].

Un sistema se dice que estabilizable por realimentación si existe una ley de control $u = k(x)$ tal que el sistema realimentado es estable en algún sentido.

Función de Lyapunov de control

Definición 1. El sistema en (3.5) se dice que satisface una condición estocástica de Lyapunov en el origen si existe un entorno W del origen en $\mathbb{R}^n$ y una función definida positiva $\Phi: W \to \mathbb{R}^+$ y una función definida positiva ρ tal que para todo $x \in W \backslash \{0\}$ se verifica la condición (3.*13*).

Definición 2. La función de control de Lyapunov (CLF) Φ se dice que satisface la propiedad de control acotado si existe una función real positiva $c: W \to \mathbb{R}$ tal que c está acotada en W y para cada $x \in W \backslash \{0\}$ existe un control $u \in \mathbb{R}^m$ que satisface las siguientes desigualdades,

$$||u|| < c(x), \qquad Y\Phi(x) < 0.$$

Si además $\lim_{x \to 0} c(x) = 0$, entonces se dice que Φ satisface la propiedad de control pequeño.

Teorema de Artstein – Sontag estocástico

Teorema 3 (i) El sistema (3.5) satisface una condición estocástica de Lyapunov en el origen si y sólo si es un sistema estabilizable asintóticamente por medio de una ley de realimentación $u = k\,(x)$, que es suave en un entorno del origen excepto posiblemente en $x = 0$.

(ii) La correspondiente CLF, satisface la propiedad de control acotado, si y sólo si existe un estabilizador $u = k\,(x)$ que es suave en un entorno W del origen, excepto posiblemente en $x = 0$ y satisface,

$$||k(x)|| < c(x),$$

donde c se define en (1). Esto implica que la función V está acotada en un entorno del origen y si, además Φ satisface la propiedad de control pequeño entonces $k(x) \to 0$ cuando $x \to 0$.

2.3. Descripción de la biblioteca QUANTUM

Para este trabajo se ha utilizado una biblioteca gratuita de comandos de Wolfram Mathematica realizada por José Luis Gómez – Muñoz para la obtención de las coordenadas en la hiperesfera de Bloch, llamada QUAMTUM [15].

QUAMTUM es un complemento de Mathematica que puede utilizarse para realizar calculas con notación de Dirac y notación de operadores, permitiendo al usuario introducir los cálculos directamente en la notación habitual de la Mecánica Cuántica y obtener resultados en el mismo formato. El desarrollo de esta biblioteca comenzó hace varios años, con el fin de estudiar los sistemas cuánticos aleatorios. Posteriormente, se incluyeron muchas otras funcionalidades, como álgebra de operadores y conmutadores, permitiendo además la simulación y graficación de circuitos de computación cuántica, generación y solución de ecuaciones de movimiento de Heisenberg, entre otras. Hasta la fecha, QUANTUM sigue siendo una herramienta única en su uso de la notación de Dirac, es por ello por lo que se utilice tanto en la entrada como en la salida de los cálculos.

Como ya se ha explicado, QUANTUM es un complemento de Mathematica que implementa cálculos en notación de Dirac, y permite al usuario final introducir dicha

notación utilizando el teclado y las barras de herramientas de las paletas. Como ejemplo sencillo, el usuario puede escribir y obtener fácilmente el siguiente resultado,

$$\begin{aligned}&\text{Entrada: } |\psi\rangle = \alpha|\phi_1\rangle + \beta|\phi_2\rangle,\\&\text{Salida: } \alpha\alpha^*|\phi_1\rangle\langle\phi_1| + \alpha\beta^*|\phi_1\rangle\langle\phi_2| + \beta\alpha^*|\phi_2\rangle\langle\phi_1| + \beta\beta^*|\phi_2\rangle\langle\phi_2|.\end{aligned} \tag{2.53}$$

El núcleo de QUANTUM se construyó basándose en la información cuántica y las aplicaciones de la computación cuántica, por lo que varios de los comandos avanzados de la versión actual están restringidos a los espacios discretos de Hilbert. Este se puede dividir en tres módulos principales; la primera, *Quantum'Notation'*, que implementa la notación bra-ket, el álgebra no conmutativa de operadores y conmutadores, y las medidas cuánticas. El segundo módulo, *Quantum'Computing'*, incluye qubits y puertas cuánticas para la simulación de algoritmos y el dibujo automatizado de circuitos de computación cuántica. Por último, el tercer módulo, *Quantum'QHD'*, implementa la aproximación *Quantized Hamilton Dynamics* (QHD) a las ecuaciones de movimiento de Heisenberg. Para el desarrollo de este trabajo fin de estudios, se ha utilizado el primer módulo, por lo que a continuación se realizará una descripción de este.

El módulo *Quantum'Notation'* implementa el álgebra no conmutativa de los operadores, los conmutadores y la notación bra-ket de Dirac. Además, incluye comandos para la simulación de medidas cuánticas, el cálculo de la traza parcial, la transposición parcial, entre otros.

La Figura 3 muestra una jerarquía de algunos de los comandos de este módulo, así como grupos de comandos desarrollados principalmente en base a Kets, operadores y operaciones. Así, se incluyen las principales operaciones básicas utilizadas en la notación de Dirac para bases discretas como qubits, qudits, operadores, productos adjunto, escalar, interno y tensorial, así como otros objetos específicos en forma de estados de Bell, matrices de Pauli y operaciones de trazado parcial. Una de las paletas de este módulo facilita la introducción de la notación que contiene los principales elementos del módulo (Figura 4).

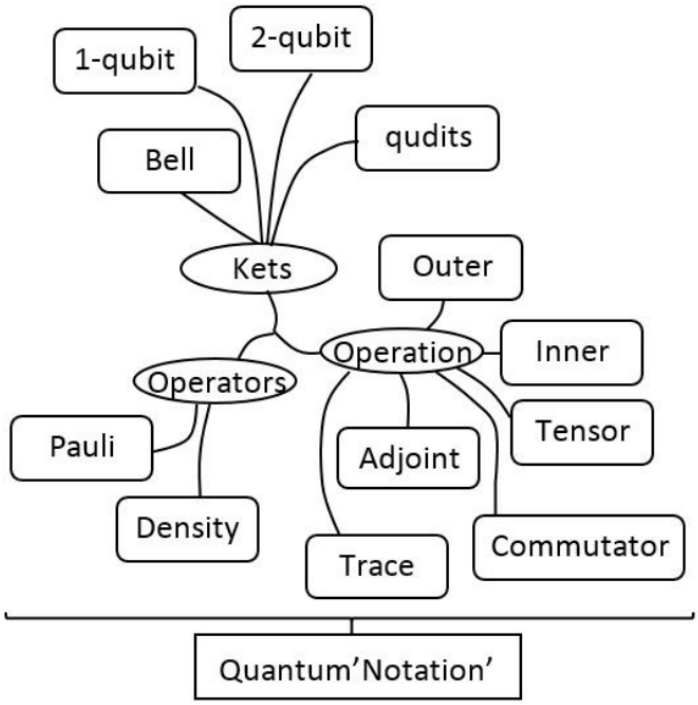

Figura 3. Principales secciones desarrolladas en el módulo Quantum'Notation' que integran los aspectos básicos en la notación de Dirac: bra-ket's, operadores y operaciones.

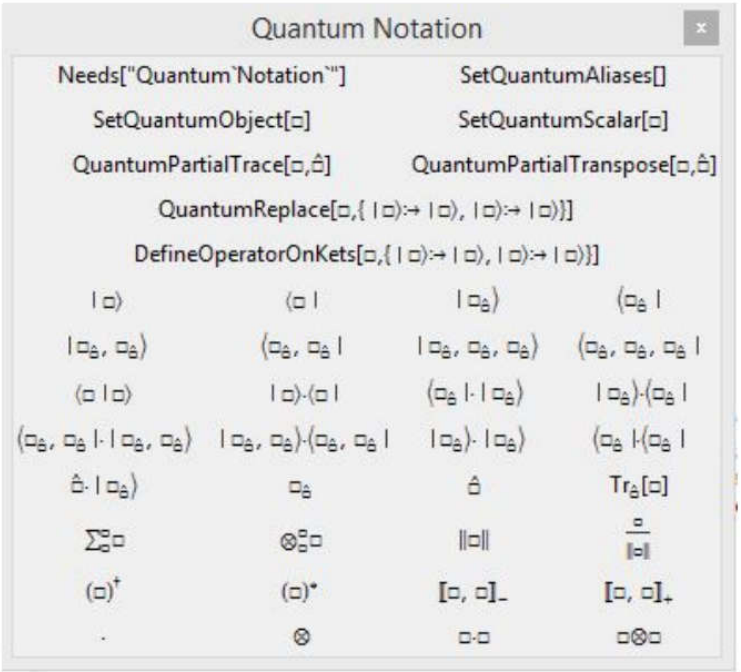

Figura 4. Paleta Quantum'Notation' que muestra los principales comandos implementados en el módulo: bra-ket's y operaciones básicas. Fuente: [15].

De esta forma, con estas construcciones básicas, es posible establecer y calcular estados, operadores, proyecciones, mediciones, probabilidades y trazas de objetos

cuánticos que operan sobre una base de producto tensorial de espacios discretos de Hilbert, cubriendo muchas posibilidades para los sistemas cuánticos [15].

Capítulo 3

Modelo del qubit

Los sistemas cuánticos son muy susceptibles a los cambios externos, por lo que generalmente se ha idealizado en un sistema cuántico cerrado, es decir, presentando un aislamiento completo. En la naturaleza, sin embargo, esto rara vez sucede porque existen interacciones inevitables entre el sistema de interés y el medio ambiente en su conjunto. Por tanto, es necesario utilizar métodos que permitan la eliminación del entorno de las ecuaciones de la dinámica del sistema de estudio. Este es un problema fundamental en la teoría de los sistemas cuánticos abiertos [11].

Este trabajo fin de estudios se ha realizado a partir de un sistema no lineal abierto, por lo que, a continuación, se ha recogido la explicación de los sistemas cuánticos abiertos, la ecuación generalizada de Lindblad, la ecuación del qubit y los controladores empleados.

3.1. Ecuación maestra del qubit

3.1.1. Sistemas cuánticos abiertos

Un sistema cuántico abierto es un sistema no aislado que interactúa con su entorno, en el que a veces se denomina *batch* o *reservoir*, en donde el resultado final es el estado de todo el universo descrito por una función de onda ψ. En realidad, todo sistema es abierto y un sistema cuántico aislado es una idealización y, en consecuencia, la ecuación de Schrödinger también es una idealización para sistemas cuánticos cerrados.

En la Figura 5 se aprecia que existe un sistema total que consiste en el sistema de interés y el entorno. Por lo tanto, el objetivo de la teoría de sistemas cuánticos abiertos es inferir del sistema completo las ecuaciones dinámicas del sistema de interés, que suelen ser más prácticas y fáciles de resolver.

En los sistemas cuánticos abiertos, hay un caso de especial interés, el de los sistemas conectados a diferentes entornos modelados por interacciones de Markov. Este tipo de interacción consiste básicamente en el concepto físico de procesos aleatorios, en

donde, para este caso, la dinámica del sistema suele caracterizarse por la ecuación de Lindblad, también conocida como ecuación de Gorini – Kossakowski – Sudarshan – Lindblad [11].

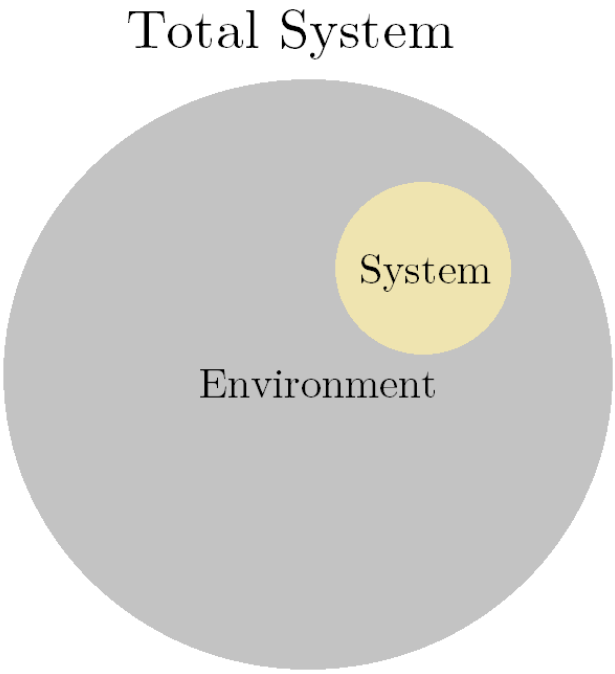

Figura 5. Esquema general de un sistema cuántico abierto. Fuente: [11].

3.1.2. Ecuación maestra de Lindblad

La ecuación maestra de Lindblad define los sistemas cuánticos abiertos, que podemos expresarla como una ecuación diferencial que evoluciona con el tiempo con el operador densidad $t \longmapsto \rho(t)$,

$$\frac{d}{dt}\rho = -i[H,\rho] + \sum_{v} L_v \rho L_v^{\dagger} - \frac{1}{2}\left(L_v^{\dagger}L_v\rho \; + \; \rho L_v^{\dagger}L_v\right), \tag{3.1}$$

donde H representa en esta ecuación el Hamiltoniano que puede depender del tiempo (operador hermítico sobre el espacio de Hilbert) y L_v son los operadores de Hilbert que pueden ser no necesariamente hermíticos [9].

En la ecuación diferencial (3.1), si la condición inicial ρ_0 es Hermitiana de traza uno y no negativa, entonces su solución $\rho(t)$ para $t \geq 0$ es también Hermitiana, no negativa y de traza uno. Para evitar tecnicismos matemáticos, se ha considerado en el teorema siguiente que el espacio de Hilbert es de dimensión finita.

Teorema 1. *Supongamos que el espacio de Hilbert es de dimensión finita. Entonces, para cualquier operador hermítico* $t \longmapsto H(t)$ *y cualesquiera operadores* $L_v(t)$*, la solución de (3.1) con una condición inicial* ρ_0 *hermitiana, no negativa y de traza uno, está definida para todo* $t > 0$*, que sigue siendo hermitiana, no negativa y de traza uno.*

La existencia de la solución para $t > 0$ es consecuencia de un resultado estándar sobre sistemas diferenciales lineales de dimensión finita y con coeficientes acotados y medibles en el tiempo. La conservación de la hermiticidad y de la traza se deduce directamente del hecho de que el lado derecho de (3.1) es hermítico en cuanto ρ es hermítico, y admite una traza nula. La conservación de la positividad es menos sencilla,

$$\frac{d}{dt}\rho = A\rho + \rho A^\dagger + \sum_v L_v \rho L_v^\dagger, \tag{3.2}$$

siendo $A = -iH - \frac{1}{2}\sum_v L_v^\dagger L_v$. Considerando que la solución para la ecuación $\frac{d}{dt}E = AE$ con $E_0 = I$, donde E es invertible y define el cambio de variables $\rho = E\xi E^\dagger$, entonces se tiene que,

$$\frac{d}{dt}\xi = \sum_v M_v \xi M_v^\dagger, \tag{3.3}$$

donde $M_v = E^{-1} L_v E^{-1}$. El hecho de que $\xi_0 = \rho_0$ sea hermítica no negativa y que $\frac{d}{dt}\xi$ sea también hermítica y no negativa, implica que ξ permanece no negativa para todo $t > 0$.

Teorema 2. *Consideremos dos soluciones de (3.1),* ρ *y* ρ'*, a partir de dos operadores hermíticos no negativos de traza uno* ρ_0 *y* ρ_0'*. Supongamos que el espacio de Hilbert es de dimensión finita y que el operador hermítico H(t) y los operadores* $L_v(t)$ *son funciones acotadas y medibles de tiempo. Entonces para cualquier* $0 \leq t_1 \leq t_2$*,*

$$\begin{aligned} &Tr(|\rho(t_2), \rho'(t_2)|) \leq Tr(|\rho(t_1), \rho'(t_1)|), \\ &F\left(\rho(t_2), \rho'(t_2)\right) \geq F\left(\rho(t_1), \rho'(t_1)\right). \end{aligned} \tag{3.4}$$

3.1.3. Representación de la ecuación de Lindlad en la esfera de Bloch

Anteriormente, en el capítulo 2, se explicó qué es la esfera de Bloch y el por qué su importancia en la representación de estados cuánticos. En este apartado, se ha explicado cómo se ha obtenido la ecuación estocástica que define nuestro sistema.

Para poder trabajar en MATLAB®, se hace necesario obtener las expresiones estocásticas de la siguiente forma,

$$dX_t = F(t, X_t)\, dt \; + \; G(t, X_t), \tag{3.5}$$

donde $F(t, X_t)$ define el término de arrastre o drift, este es el término que gobierna la ecuación diferencial, es decir, el determinista, mientras que $G(t, X_t)$ es el de difusión o *diffusion*, que es la parte que representa al ruido y es de naturaleza Browniana. Para la obtención de estos, se ha utilizado el paquete QUANTUM, en concreto, el módulo *Quantum'Notation'*.

Para ello, primero se han definido las matrices de Pauli, representadas como σ_i, la matriz densidad, ρ, explicada en el capítulo anterior; y el Hamiltoniano, H, definido como $H = \frac{\omega_{eg}}{2}\sigma_3$, donde ω_{eg} es la frecuencia del qubit. Posteriormente, en la Figura 7, se han obtenido las expresiones de los términos de *drift* y *diffusion.*

De esta forma, la ecuación que define el sistema controlado se puede expresar como,

$$\begin{pmatrix} dp_x \\ dp_y \\ dp_z \end{pmatrix} = \begin{pmatrix} -\frac{\Gamma_m}{2} & -\frac{1}{\hbar}\omega_{eg} & 0 \\ \frac{1}{\hbar}\omega_{eg} & -\frac{\Gamma_m}{2} & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} dt + \begin{pmatrix} 0 \\ -\frac{x_3}{\hbar} \\ \frac{x_2}{\hbar} \end{pmatrix} u dt + \begin{pmatrix} -\sqrt{\eta\Gamma_m}x_1x_3 \\ -\sqrt{\eta\Gamma_m}x_2x_3 \\ -\sqrt{\eta\Gamma_m}\left(x_3^2 - 1\right) \end{pmatrix} dW, \tag{3.6}$$

donde $p = \left(p_x, p_y, p_z\right)$ es el vector de Bloch, vector real de tres componentes, Γ_m representa la tasa de disipación, $\hbar$ la constante de Planck, ω_{eg} la frecuencia del qubit, η la eficiencia del detector, siendo $\eta \in [0,1]$ y u es la ley de control.

Finalmente, la función de Lyapunov asociada al sistema es,

$$V(\rho) = \; 1 \; - \; p_z. \tag{3.7}$$

```
SetQuantumObject[σ];
(σ_{a:0|1|2|3})^2 := σ_0;
(σ_{a:0|1|2|3})^† := σ_a;
σ_{a:0|1|2|3} · σ_0 := σ_a;
σ_0 · σ_{b:0|1|2|3} := σ_b;
σ_{a:1|2|3} · σ_{b:1|2|3} := KroneckerDelta[a, b] * σ_0 + i * Sum_{c=1}^{3} Signature[{a, b, c}] * σ_c;
                             delta de Kronecker                 signatura
[[σ_0, σ_{b:0|1|2|3}]]_- := 0;
[[σ_{a:1|2|3}, σ_{b:1|2|3}]]_- := 2 * i * Sum_{c=1}^{3} Signature[{a, b, c}] * σ_c;
                                            signatura
[[σ_0, σ_{b:0|1|2|3}]]_+ := 2 * σ_b;
[[σ_{a:1|2|3}, σ_{b:1|2|3}]]_+ := 2 * KroneckerDelta[a, b] * σ_0;
                                   delta de Kronecker
ρ = (σ_0 + x[t] σ_1 + y[t] σ_2 + z[t] σ_3) / 2
1/2 (σ_0 + σ_1 x[t] + σ_2 y[t] + σ_3 z[t])
H = ω_eg/2 σ_3
σ_3 ω_eg / 2
[[σ_3, σ_1]]_-
2 i σ_2
```

Figura 6. Definiciones de las matrices de Pauli, matriz densidad y Hamiltoniano

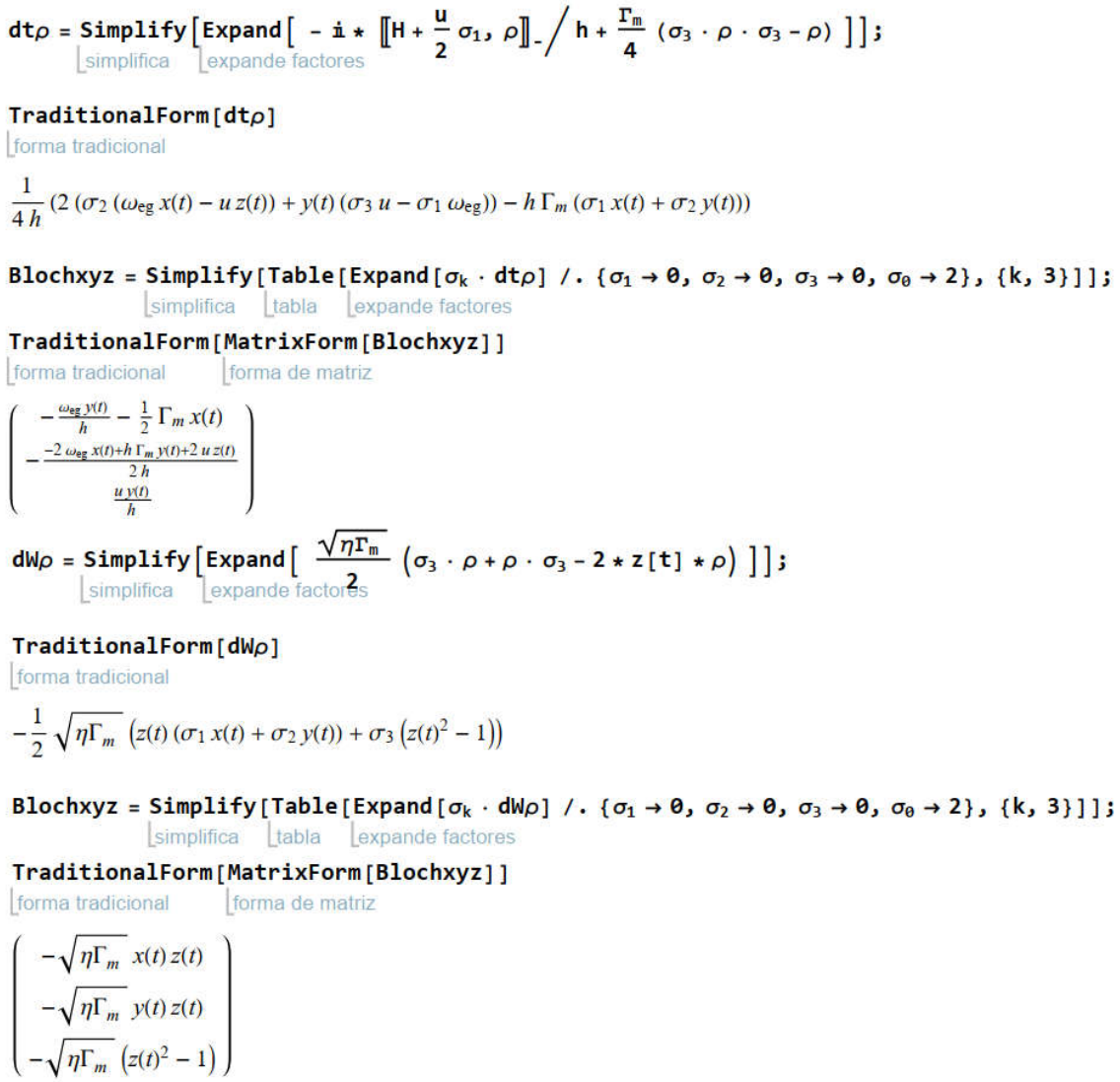

Figura 7. Obtención de los términos *drift* y *diffusion* en Wolfram Mathematica

3.2. Descripción de los controladores

En este apartado, se ha introducido los controladores empleados para el control del qubit. Para ello, se han utilizado dos controladores, el primero, propuesto por Florchinger y el segundo controlador, propuesto Juan Ignacio Mulero Martínez.

3.2.1. Controlador de Florchinger

En general una ecuación diferencial estocástica afín en el control se puede escribir de la siguiente forma,

$$dx = f(x)\ dt + g(x)u\ dt + \sigma(x)\ dW, \tag{3.8}$$

donde $f\colon \mathbb{R}^n \rightarrow \mathbb{R}^n, g\colon \mathbb{R}^n \rightarrow \mathbb{R}^m, \sigma\colon \mathbb{R}^n \rightarrow \mathbb{R}^p$.

El objetivo de control consiste en trasladar el estado x al estado deseado x_d. Consideraremos funciones de Lyapunov que midan la distancia entre estos estados. Una elección sencilla sería la siguiente,

$$\Phi(x) = (x - x_d)^T (x - x_d). \tag{3.9}$$

El generador infinitesimal $\mathcal{D}$ de la ecuación diferencial estocástica no controlada ($u = 0$) actúa como el operador *d/dt* para el caso determinista. La aplicación de $\mathcal{D}$ sobre una función $\Phi\colon \mathbb{R}^n \rightarrow \mathbb{R}$ viene dada por,

$$\mathcal{D}\Phi(x) = \nabla\Phi(x)^T f(x) + \sigma(x)^T (H\Phi(x))\, \sigma(x), \tag{3.10}$$

donde $\nabla\Phi(x)$ es el gradiente de la función Φ con respecto a x y $H\Phi(x)$ es el Hessiano de Φ.

Para la parte controlada, es decir, $u \neq 0$, también habrá un generador infinitesimal, denotado como Y,

$$Y\Phi(x) = \mathcal{D}\Phi(x) + \mathcal{D}_u\Phi(x)u, \tag{3.11}$$

donde $\mathcal{D}_u\Phi(x) = \nabla\Phi(x)^T g(x)$.

Para diseñar una ley de control que estabilice el sistema necesitamos recurrir a la noción de Lyapunov de control [14]:

Definición 1. *Una función* $\Phi\colon \mathbb{R}^n \rightarrow \mathbb{R}$ *definida positiva se dice que es una función de Lyapunov de control para un sistema estocástico si y sólo si*

$$\alpha_1(\|x\|) \leq \Phi(x) \leq \alpha_2(\|x\|) \tag{3.12}$$

y para $x \neq x_d$,

$$D_u\Phi(x) = 0 \Rightarrow D\Phi(x) < 0 \tag{3.13}$$

Llamamos $a(x) = D\Phi(x), b(x) = D_u\phi(x)$.

Utilizaremos la ley de control por realimentación de estado $u(x)$ propuesta por Florchinger, expresada como,

$$u(x) = \begin{cases} 0, & \text{si } x = x_d, \\ -\rho\big(a(x), b(x)\big), & \text{en otro caso,} \end{cases} \tag{3.14}$$

en donde,

$$\rho(x) = \begin{cases} 0, & \text{si } b = 0 \text{ y } a < 0, \\ \dfrac{a + \sqrt{a^2 + b^2}}{2}, & \text{en otro caso.} \end{cases} \tag{3.15}$$

3.2.2. Controlador de J. Ignacio Mulero

Para la realización del segundo controlador, se van a establecer dos regiones, una primera región C_1 definida como,

$$C_1 = \{x \in B_\varepsilon(x_F) \cap S^{n^2-1} \setminus \{x_F\} : \mathcal{L}_g V(x) = 0\}, \tag{3.16}$$

donde $\mathcal{L}_g V(x) = \nabla V(x) \cdot g(x) = (\nabla V(x)^T \cdot g(x))$. Por otro lado, tenemos el segundo conjunto C_2 definido como,

$$C_2 = \{x \in B_\varepsilon(x_F) \cap S^{n^2-1} \setminus \{x_F\} : \mathcal{D}_{\omega_0} V(x) \geq 0\}, \tag{3.17}$$

donde $\omega_0(x) = f(x)$ y $\mathcal{D}_{\omega_0}$ es el generador infinitesimal, de segundo orden,

$$\mathcal{D}_{\omega_0} V(x) = \mathcal{L}_{\omega_0} V(x) + \frac{1}{2} \mathcal{L}_\sigma^2 V(x), \tag{3.18}$$

donde $\mathcal{L}_{\omega_0} V(x) = \nabla V(x) \cdot f(x) = (\nabla V(x)^T \cdot f(x))$. Por otro lado, tenemos que $\mathcal{L}_\sigma^2 V(x) = \sigma(x)^T (HV(x)) \sigma(x)$ siendo $HV(x)$ el Hessiano de $V(x)$.

Con esto, podemos definir el controlador como,

$$u(x) = -v(x)\, \mathcal{L}_g V(x), \tag{3.19}$$

donde

$$v(x) = \begin{cases} 0, & \text{si } x \in C_1 \cup \{x_F\}, \\ 1 + \dfrac{\varphi(x) \mathcal{D}_{\omega_0} V(x)}{\mathcal{L}_g V(x)}, & x \neq x_F \text{ y } \mathcal{L}_g V(x) \neq 0, \end{cases} \tag{3.20}$$

siendo $\varphi(x)$ una función continua definida como,

$$\varphi(x) = \frac{d(x, C_1)}{d(x, C_1) + d(x, C_2)}, \tag{3.21}$$

donde $d(x, C_1)$ es la distancia del punto x al conjunto C_1, mientras que $d(x, C_2)$ representa la distancia del punto x al conjunto C_2. En resumen, la distancia se puede calcular de la siguiente forma,

$$d(x, C_1) = \inf\{d(x, y) : y \in C_1\}. \tag{3.22}$$

Si $x \in C_1 \Rightarrow d(x, C_1) = 0 \Rightarrow \varphi(x) = 0$. Por otro lado, si $x \in C_2 \Rightarrow d(x, C_2) = 0 \Rightarrow \varphi(x) = 1$.

A continuación, se ha explicado paso por paso cómo se han obtenido las diferentes ecuaciones necesarias para su correcta aplicación en MATLAB®.

Considerando que la función de Lyapunov es $V(x) = 1 - x_3$, entonces,

$$\mathcal{L}_g V(x) = (\nabla V(x)^T \cdot g(x)) = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}^T \begin{pmatrix} 0 \\ -\frac{x_3}{\hbar} \\ \frac{x_2}{\hbar} \end{pmatrix} = -\frac{1}{\hbar} x_2. \tag{3.23}$$

Para comprobar qué valores formarán parte del conjunto C_1 hemos definido anteriormente en (3.16) que se debía cumplir que $\mathcal{L}_g V(x) = 0$. Esto se cumplirá cuando $x_2 = 0$, de forma que recorremos el plano $x_1 x_3$ con vector normal que define el plano en (0,1,0).

Por lo tanto, tomando el punto (x_1, x_2, x_3) y los puntos del círculo máximo, que es la intersección de la esfera con el plano $x_1 x_3$, es decir, $(cos\ \alpha, 0, sin\ \alpha)$, se tiene que,

$$d_1 = min_{\alpha \in [-\pi, \pi]}\ arccos\ (x_1\ cos\ \alpha + x_3\ sin\ \alpha). \tag{3.24}$$

Si se pone en cuadratura $x_1\ cos\ \alpha + x_3\ sin\ \alpha$, entonces,

$$x_1\ cos\ \alpha + x_3\ sin\ \alpha = \sqrt{x_1^2 + x_3^2}\ cos\ (\alpha + \theta). \tag{3.25}$$

De esta manera, el arccos es estrictamente decreciente, por lo que el mínimo se alcanza cuando $cos\ (\alpha + \theta) = 1$. Por tanto,

$$d_1 \;= arccos\,\sqrt{x_1^2 \,+\, x_3^2} \;=\; arccos\,\sqrt{1+x_2^2}\,. \tag{3.26}$$

Entonces, la distancia del conjunto C_1 queda como,

$$d(x, C_1) \;=\; min\,\{d_1\} \;=\; arccos\,\sqrt{1+x_2^2}\,. \tag{3.27}$$

Esto nos indica que, $x \,\in\, C_1 \;\Rightarrow\; d(x, C_1) \,=\, 0$ cuando $x_2 \,=\, \pm\,1$, dado que el $arccos\;[-1,1] \;\rightarrow\; [0,\pi]$ es monótona decreciente y la función distancia es no negativa, la distancia mínima se alcanza cuando el argumento es 0.

El conjunto C_2 definido en (3.17), debe cumplir la condición de $\mathcal{D}_{\omega_0}V(x) \,\geq\, 0$. El generador infinitesimal de segundo orden tendrá la siguiente forma,

$$\mathcal{D}_{\omega_0}V(x) = \mathcal{L}_{\omega_0}V(x) + \frac{1}{2}\mathcal{L}_{\sigma}^{2}V(x) = (\nabla V(x)^T \cdot f(x)) + \frac{1}{2}\sigma(x)^T(HV(x))\sigma(x). \tag{3.28}$$

Para el caso particular del qubit,

$$\begin{aligned}\mathcal{D}_{\omega_0}V(x) \;=\; &\begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}^T \begin{pmatrix} -\frac{\Gamma_m}{2} & -\frac{1}{\hbar}\omega_{eg} & 0 \\ \frac{1}{\hbar}\omega_{eg} & -\frac{\Gamma_m}{2} & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \;+ \\ &\frac{1}{2}\begin{pmatrix} -\sqrt{\eta\Gamma_m}x_1x_3 \\ -\sqrt{\eta\Gamma_m}x_2x_3 \\ -\sqrt{\eta\Gamma_m}\left(x_3^2-1\right) \end{pmatrix}^T \left(\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} -\sqrt{\eta\Gamma_m}x_1x_3 \\ -\sqrt{\eta\Gamma_m}x_2x_3 \\ -\sqrt{\eta\Gamma_m}\left(x_3^2-1\right) \end{pmatrix} \right) \;=\; 0.\end{aligned} \tag{3.29}$$

Resolviendo (3.29), se obtiene que $\mathcal{D}_{\omega_0}V(x) \,=\, 0$. De esta forma, la función distancia del conjunto C_2 queda entonces limitada por 2π,

$$d(x, y) \;=\; min\,\{arccos\,(x^T y), 2\pi\}. \tag{3.30}$$

Sabiendo que la distancia entre x y $C_2 \,=\, \varnothing$ debería ser la máxima, al trabajar en una esfera, entonces, se tiene que,

$$d(x, C_2) \;=\; d(x, \varnothing) \;=\; 2\pi. \tag{3.31}$$

Finalmente, sabiendo las distancias, se puede resolver la ecuación (3.21),

$$\varphi(x) = \frac{d(x, C_1)}{d(x, C_1) + d(x, C_2)} = \frac{arccos\sqrt{1 + x_2^2}}{arccos\sqrt{1 + x_2^2} + 2\pi}. \qquad (3.32)$$

Todo esto, se ha definido en MATLAB® con el objetivo de obtener un sistema controlado.

3.3. Métodos numéricos para la resolución de las ecuaciones diferenciales estocásticas

Existen diferentes métodos para resolver las ecuaciones diferenciales estocásticas, como el método de Euler – Maruyama, el método de Milstein y el método de Runge – Kutta. En este trabajo, se ha empleado el método numérico de solución de las SDE de MATLAB®, el método de Euler – Maruyama y el método de Milstein, por lo que, en esta sección se realizará una explicación de estos métodos.

3.3.1. Método numérico de MATLAB®

El paquete Financial Toolbox™ de MATLAB® ofrece diversas funciones para el modelado y análisis estadístico de datos. Las diversas herramientas incorporadas en esta toolbox de las SDEs hacen posible la modelación y simulación de una variedad de procesos estocásticos. Anteriormente, se ha señalado que para poder trabajar en MATLAB®, las expresiones estocásticas deben estar expresadas por su término de arrastre y de difusión como,

$$dX_t = F(t, X_t)\,dt + G(t, X_t). \qquad (3.33)$$

La sintaxis empleada para la creación de una ecuación diferencial estocástica en MATLAB® es la que se muestra en la Figura 8.

Figura 8. Sintaxis empleada para la creación de SDEs. Fuente: [16].

En este trabajo fin de estudios se ha implementado de dos formas distintas el método de Euler – Maruyama. En primer lugar, se ha realizado a través del algoritmo que usa MATLAB® por defecto para la simulación de las ecuaciones diferenciales estocásticas. En segundo lugar, se ha realizado mediante implementación propia, por lo que, en el siguiente apartado se explicará la teoría de este método.

3.3.2. Método de Euler – Maruyama

El método de Euler – Maruyama es uno de los métodos más simples para la aproximación de una SDE. Se trata de un análogo del método de Euler para ecuaciones diferenciales ordinarias. Este método es el más simple debido a que parte de una aproximación de primer orden,

$$X(t_k) = X(t_{k-1}) + f\left(t_{k-1}, X(t_{k-1})\right)\Delta t + g\left(t_{k-1}, X(t_{k-1})\right)\Delta W(t_k). \quad (3.34)$$

En donde el término Browniano puede aproximarse como,

$$\Delta W(t_k) \; = \; \epsilon(t_k)\sqrt{\Delta t}, \quad (3.35)$$

donde $\epsilon(.)$ representa un proceso discreto de ruido blanco Gaussiano, en donde la media es 0 y la desviación típica es 1 [9].

3.3.3. Método de Milstein

Para obtener una aproximación aún más precisa que el de Euler – Maruyama se suele utilizar el propuesto por Milstein, un método para la solución numérica aproximada de una SDE. Este modelo se ha implementado a través del paquete SDETools ofrecida por Andrew D. Horchler [17]. El método de Milstein es más complejo ya que es una aproximación de segundo orden,

$$X(t_k) = X(t_{k-1}) + f\left(t_{k-1}, X(t_{k-1})\right)\Delta t + g\left(t_{k-1}, X(t_{k-1})\right)\Delta W(t_k) + \frac{1}{2}\frac{\partial g}{\partial X}\left(t_{k-1}, X(t_{k-1})\right)[(\Delta W(t_k))^2 - \Delta t] \tag{3.36}$$

Estos métodos convergen correctamente si el tamaño de paso Δt es lo suficientemente pequeño. Esto es importante ya que, en la práctica, los errores de redondeo se interponen cuando el tamaño del paso es demasiado pequeño. Es por ello por lo que, para obtener una solución aproximada para un SDE dado, necesitamos simular un número significativo de trayectorias, ya que una única trayectoria simplemente sería la realización de una distribución desconocida variante en el tiempo [9].

Capítulo 4

Resultados

Una vez descritos en capítulos anteriores la teoría correspondiente del qubit, los métodos numéricos de resolución de las ecuaciones diferenciales estocásticas y los controladores empleados, se realiza en este capítulo una simulación de los modelos no controlado y controlado, presentando los resultados obtenidos.

En primer lugar, se han desarrollado pruebas para observar el comportamiento del qubit sin hacer uso de controladores. Posteriormente, se han aplicado los diferentes controladores propuestos en el capítulo anterior, donde además se ha comprobado los parámetros de los que depende el sistema.

Todo esto se ha llevado a cabo utilizando MATLAB®, en donde para la resolución del sistema del qubit no controlado y controlado, con diferentes parámetros de simulación. Estos parámetros son, T_{mes} constante de tiempo de la medida, $T_{mes} = 1$; Γ_m, ratio de emisión espontánea, $\Gamma_m = 1/T_{mes}$; ω_{eg} , frecuencia natural, $\omega_{eg} = 2\pi/T_{mes}$; η , la eficiencia, $\eta = 1$, ya que consideramos que es un caso ideal; $\hbar$, la constante de Planck reducida, $\hbar = 1$; dt, el paso, $dt = 0{,}01\ s$ y X_{eq}, el punto deseado al que queremos alcanzar, $X_{eq} = (0,0,1)$. Se ha tomado como punto de partida inicial $\left(0.1, \sqrt{35}/10\,, -0.8\right)$.

En el Anexo A se encuentra el código que se ha desarrollado para este trabajo fin de estudios.

4.1. Sistema no controlado

Para los sistemas no controlados, se ha empleado el método numérico de solución de las SDE de MATLAB®, el método de Euler – Maruyama y el método de Milstein, cuyos resultados se exponen a continuación.

4.1.1. Método de Euler – Maruyama

Para la implementación del método de Euler – Maruyama se han empleado métodos diferentes, la que ofrece el paquete Financial Toolbox™ de MATLAB® y mediante implementación propia.

En las Figuras 1 y 2 se ilustran los valores medios de las componentes p_x, p_y y p_z del vector de Bloch. En la Figura 9, se muestra la resolución del sistema a partir de la herramienta proporcionada por el programa, mientras que, en la Figura 10, se muestra la proporcionada por la implementación propia, para 100 trayectorias. En ambas, se puede observar que las componentes p_x y p_y se amortiguan con el paso del tiempo, esto se debe a la disipación a la que está sujeto el propio qubit. Respecto a p_z, esta componente no llega a estabilizarse en el punto deseado, lo cual era de esperar ya que estamos en un sistema al que no le hemos aplicado ley de control para su correcta estabilización.

Si observamos la función de Lyapunov para el sistema no controlado en los distintos métodos de resolución, observamos grandes diferencias. Por un lado, en la Figura 9, se observa que los valores medios aumentan, mientras que en la Figura 10, los valores disminuyen. Además, se puede comprobar que, conforme los valores de las componentes del vector de Bloch se acercan al estado objetivo, disminuyen los valores de la función de Lyapunov, en caso contrario, los valores aumentarían al alejarse del estado objetivo. Con esto se comprueba que la función de Lyapunov marca la distancia que hay entre el estado que se mide y el estado deseado, siendo el objetivo al utilizar una ley de control que esta función evolucione a tiempo infinito a valor cero.

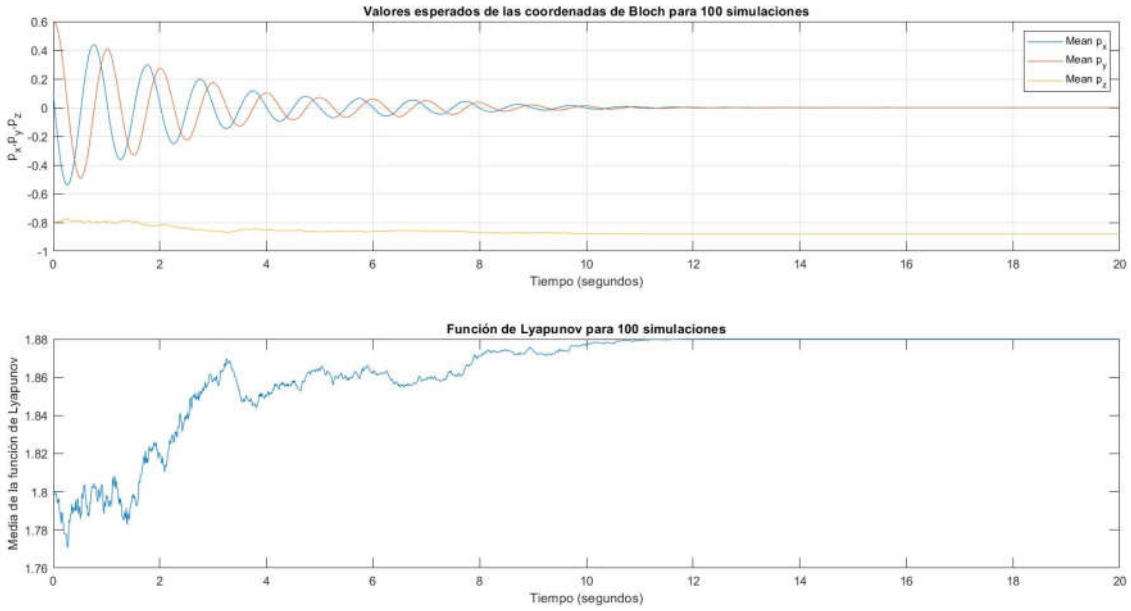

Figura 9. Simulación del qubit no controlado. Componentes promedio del vector de Bloch y de la función de Lyapunov, a partir del método de Euler - Maruyama, mediante Financial Toolbox™ MATLAB®, para 100 trayectorias.

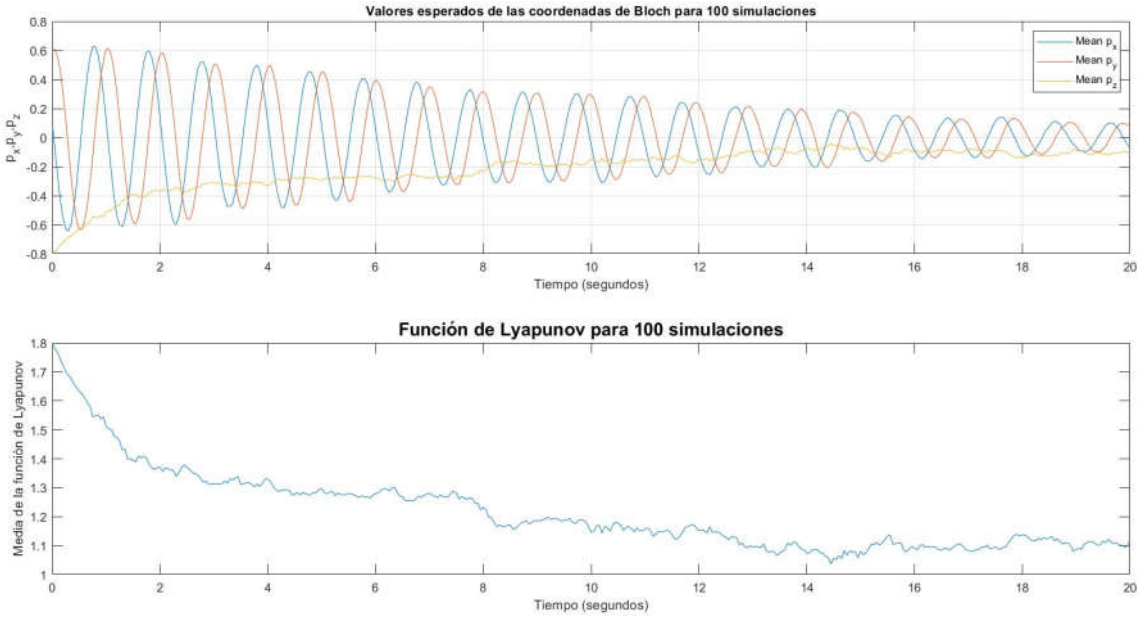

Figura 10. Simulación del qubit no controlado. Componentes promedio del vector de Bloch y de la función de Lyapunov, a partir del método de Euler - Maruyama, mediante implementación propia, para 100 trayectorias.

4.1.2. Método de Milstein

Para la implementación del método de Milstein se ha empleado la que ofrece Andrew D. Horchler en el paquete SDETools [17].

En la Figura 11 en donde se muestran los valores promedio de las componentes p_x, p_y y p_z del vector de Bloch para 100 trayectorias, se puede observar que las componentes p_x y p_y tienden a cero con el paso del tiempo, mientras que p_z, esta componente no llega a estabilizarse en el punto deseado, tendiendo al valor -0,9.

Asimismo, se han representado los valores promedios de la función de Lyapunov para 100 trayectorias. Se puede comprobar que, como ocurría en el caso de Euler – Maruyama, conforme los valores de las componentes se alejan del polo norte, los valores de la función de Lyapunov aumentan.

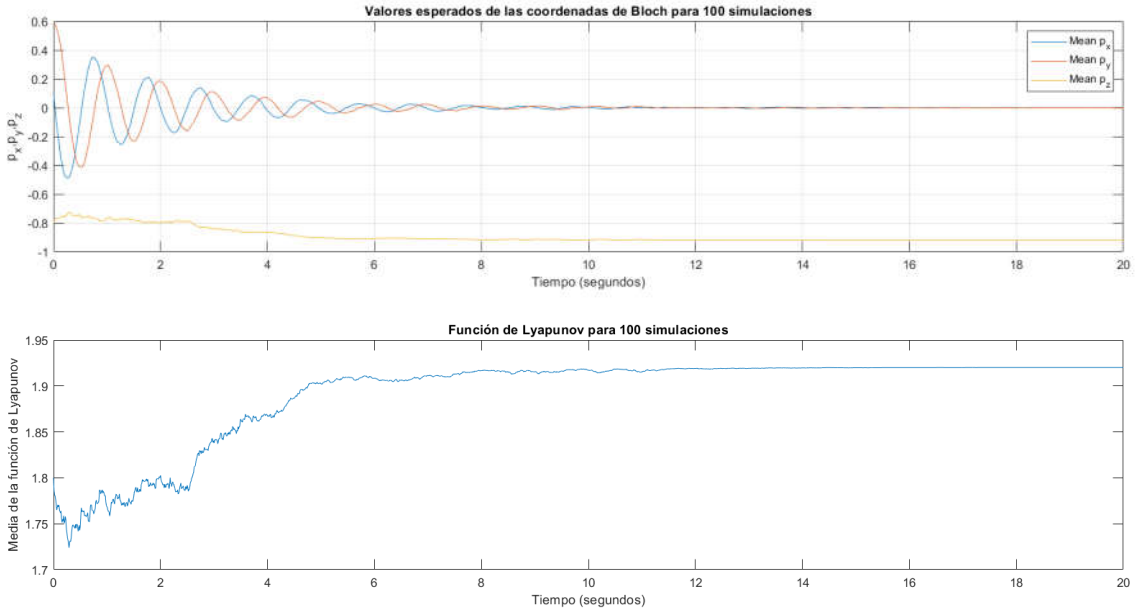

Figura 11. Simulación del qubit no controlado. Componentes promedio del vector de Bloch y de la función de Lyapunov, a partir del método de Milstein, para 100 trayectorias.

4.2. Sistema controlado

En este apartado se recogen los resultados obtenidos de la simulación del modelo del qubit controlado. Para ello, como se ha explicado anteriormente, se han empleado dos leyes de control diferentes, por un lado, el controlador de Florchinger y, por otro lado, el controlador de J. Ignacio Mulero.

Adicionalmente, se seguirá empleando los mismos parámetros iniciales indicados al inicio de este capítulo, de manera que se busca que el sistema controlado alcance a tiempo infinito el estado deseado (0,0,1).

4.2.1. Controlador de Florchinger

Para la resolución del modelo, se aplicará el controlador de Florchinger, cuya base teórica se ha explicado en el capítulo 3, apartado 3.2.1.

4.2.1.1. Método de Euler – Maruyama

Para la realización del método de Euler – Maruyama se han empleado los mismos métodos que para el sistema no controlado, por un lado, la que incluye el paquete Financial Toolbox™ de MATLAB® y, por otro lado, mediante implementación propia.

Resolución a través de Financial Toolbox™ de MATLAB®

En la Figura 12 se representa las componentes del vector de Bloch para las 100 simulaciones realizadas. Conforme avanza el tiempo, las componentes p_x y p_y realizan trayectorias sinusoidales, es decir, oscilando entre [-1,1], estabilizándose ambas en 0. A diferencia de estas componentes, p_z se estabiliza en torno a los 14 segundos a 1. Por lo que, se puede comprobar que se llega al estado objetivo marcado inicialmente, cumpliéndose la condición de unitariedad del sistema.

Si comparamos los valores medios esperados de las coordenadas de Bloch junto con la mediade la acción de control y de la función de Lyapunov (Figura 13), observamos que, conforme nos acercamos al estado deseado, la distancia que nos marca la función de Lyapunov decrece exponencialmente, llegando a 0 al alcanzarlo. Por otro lado, la acción de control, una vez alcanzado el estado deseado en $p_z = 1$, es cero, pero sigue activa, ya que el sistema posee pequeñas perturbaciones que hace que no se comporte de forma ideal.

Esto se puede apreciar en la Figura 14, donde se ilustran los valores de la señal de control para todas las trayectorias y la señal de control promediada.

Para observar cómo se comportan los valores de la función de Lyapunov en las 100 simulaciones, en la Figura 15, se muestra una representación del comportamiento de esta. Se cumple que no hay ninguna perturbación en ninguna de las simulaciones realizadas, finalizando todas en cero.

Finalmente, se ha representado en la Figura 16 algunas de las simulaciones en la hiperesfera de Bloch, seleccionadas aleatoriamente. Estas trayectorias se encuentran en la superficie de la esfera, comprobándose, además, que siguen el comportamiento de punto inicial $(0.1, \sqrt{35}/10, -0.8)$ a punto deseado (0,0,1).

Todo esto da lugar a un buen comportamiento del controlador de Florchinger, con el método de resolución de Euler – Maruyama que ofrece el paquete Financial Toolbox™ de MATLAB®, obteniéndose los resultados esperados, por lo que se puede concluir que se trata de una simulación satisfactoria.

Resolución a través de implementación propia

A diferencia de los métodos de resolución ofrecidos por los paquetes Financial Toolbox™ de MATLAB® y SDETools de Andrew Horchler, se ha diseñado un modelo de implementación propia para el método de Euler – Maruyama. Este modo de resolución permite una mayor precisión a la hora de ajustar parámetros de la simulación, a cambio de una mayor complejidad de uso.

En la Figura 17 se observa el mismo comportamiento sinusoidal por parte de las componentes p_x y p_y que se comentaba anteriormente, sin embargo, algunas simulaciones no llegan a estabilizarse a tiempo final como pasaba con el paquete Financial Toolbox™, sino que poseen pequeñas perturbaciones. No obstante, la componente p_z, para el caso de todas las simulaciones, puede apreciarse que se estabiliza en el punto deseado.

Representando en la Figura 18 los valores promediados de las coordenadas de Bloch, acción de control y función de Lyapunov, se puede analizar que las perturbaciones causadas por las componentes p_x y p_y, a tiempo aproximado de 10 segundos, provocan que la acción de control actúe con mayor frecuencia que a tiempo inicial, esto también se puede observar en la Figura 19. No obstante, de las Figuras 11 y 12, se observa que todas

las simulaciones terminan aproximadamente en cero, ya que la energía del sistema se disipa.

En la Figura 21, se encuentran cuatro trayectorias representadas en la esfera de Bloch. Se puede concluir que se cumple la condición de unitariedad del sistema.

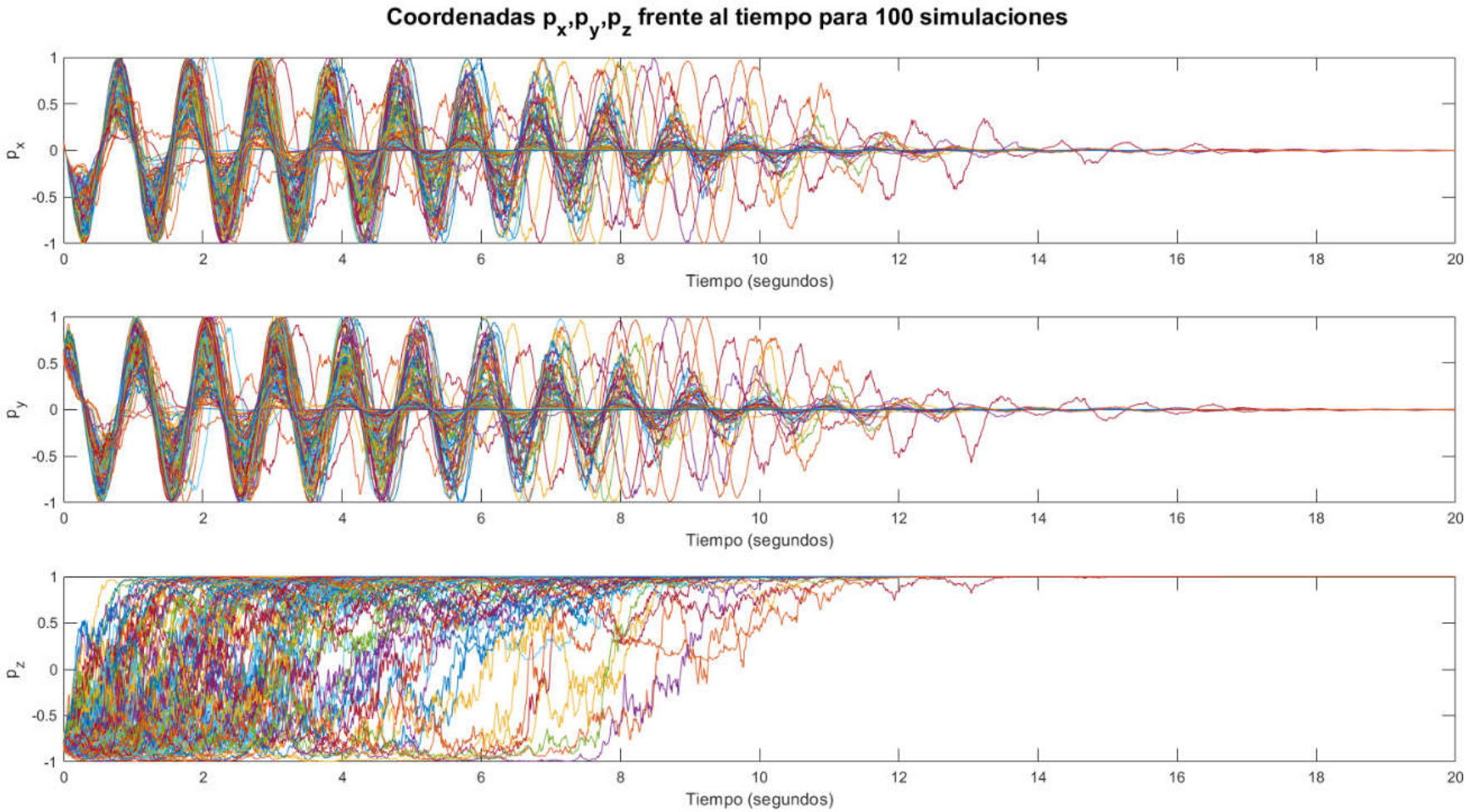

Figura 12. Simulación del qubit controlado. Componentes del vector de Bloch, a partir de Financial Toolbox™ de MATLAB, mediante Euler - Maruyama, para 100 simulaciones.

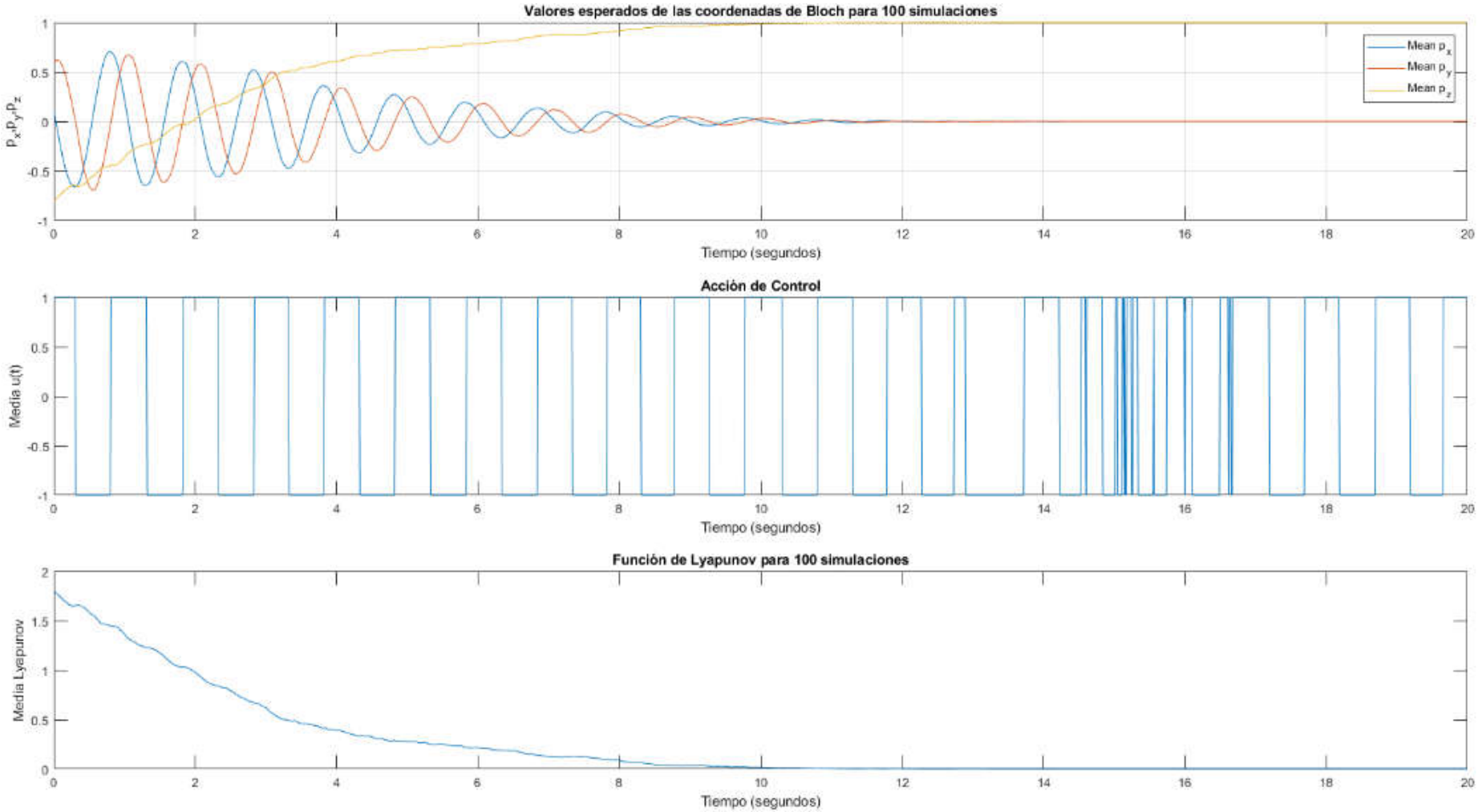

Figura 13. Simulación del qubit controlado. Coordenadas del vector de Bloch, acción de control y función de Lyapunov a partir del método de Euler - Maruyama, mediante Financial Toolbox™ MATLAB®, para 100 trayectorias.

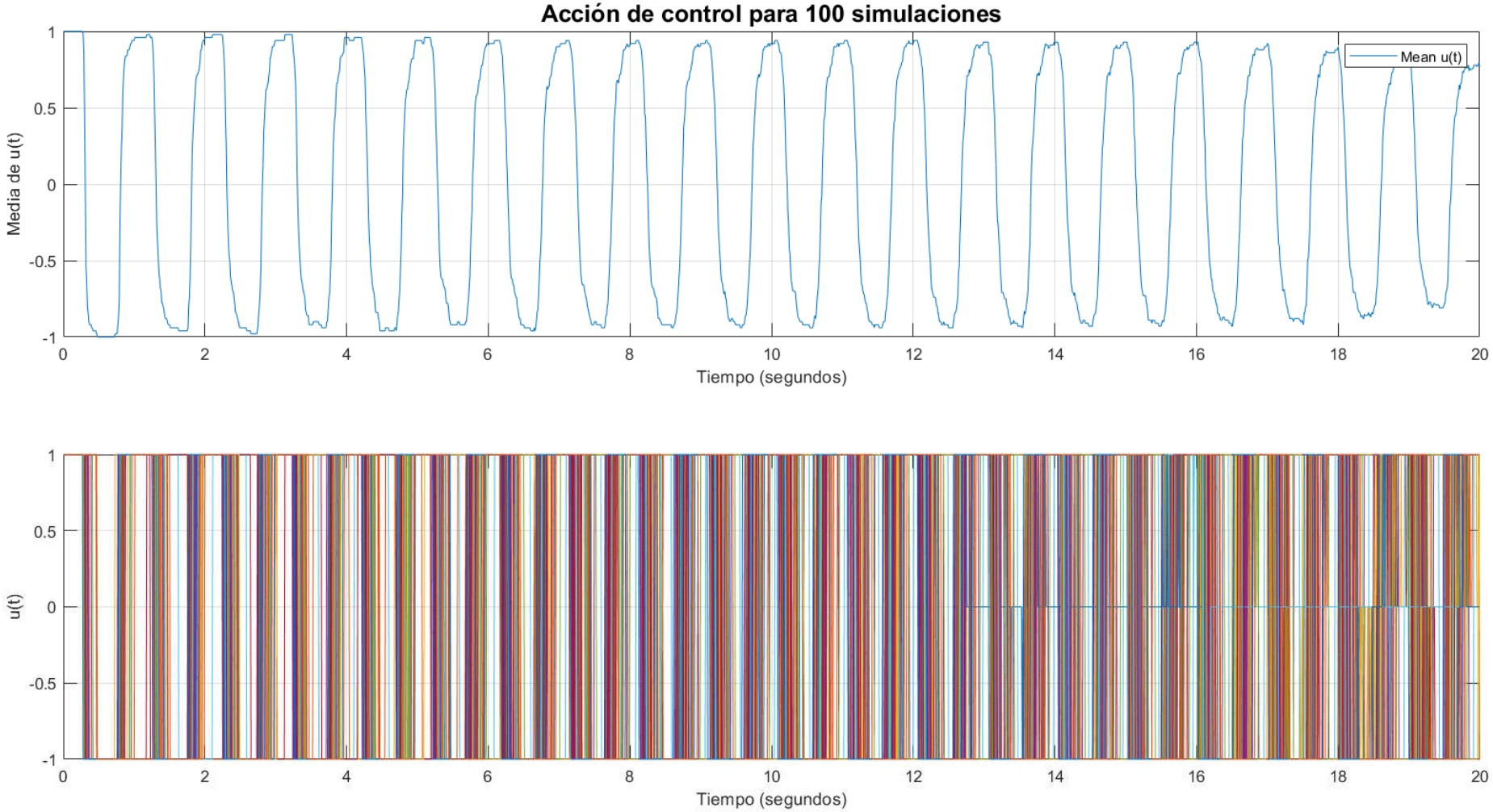

Figura 14. Simulación del qubit controlado. Valores medios y reales de la acción de control a partir del método de Euler - Maruyama, mediante Financial Toolbox™ MATLAB®, para 100 trayectorias.

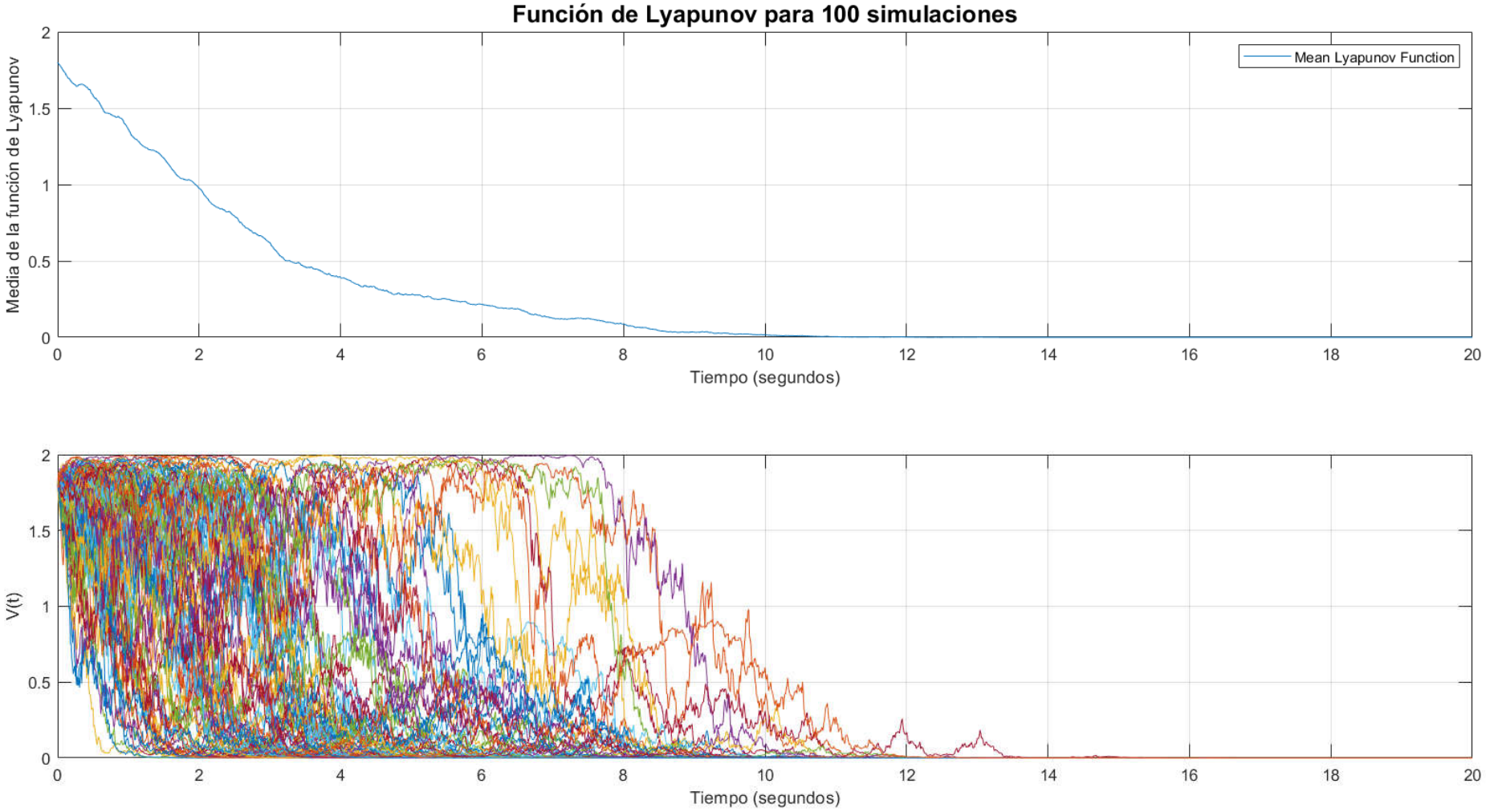

Figura 15. Simulación del qubit controlado. Valores medios y reales de la función de Lyapunov a partir del método de Euler - Maruyama, mediante Financial Toolbox™ MATLAB®, para 100 trayectorias.

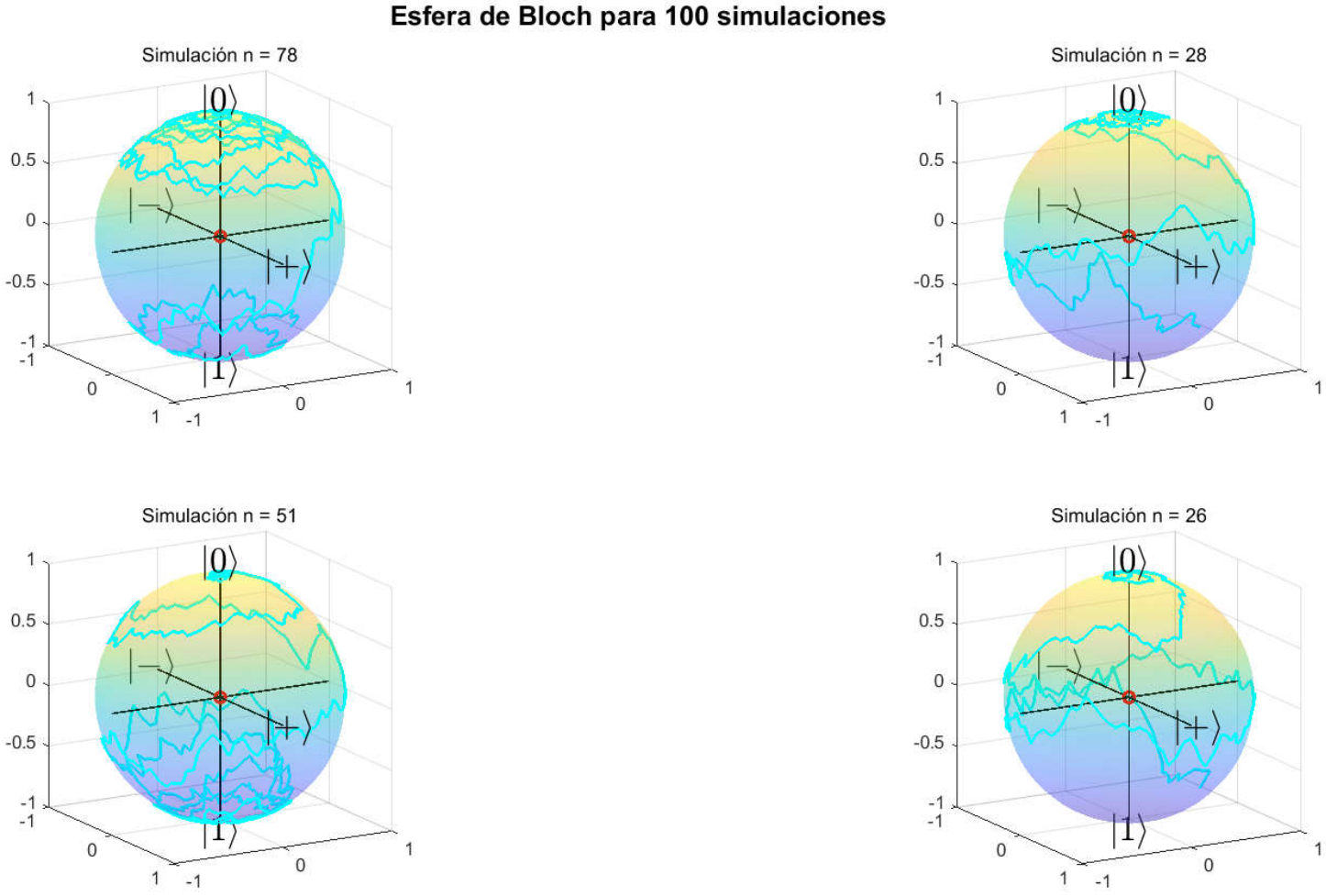

Figura 16. Simulación del qubit controlado. Representación en la esfera de Bloch a partir del método de Euler - Maruyama, mediante Financial Toolbox™ MATLAB®, para 100 trayectorias.

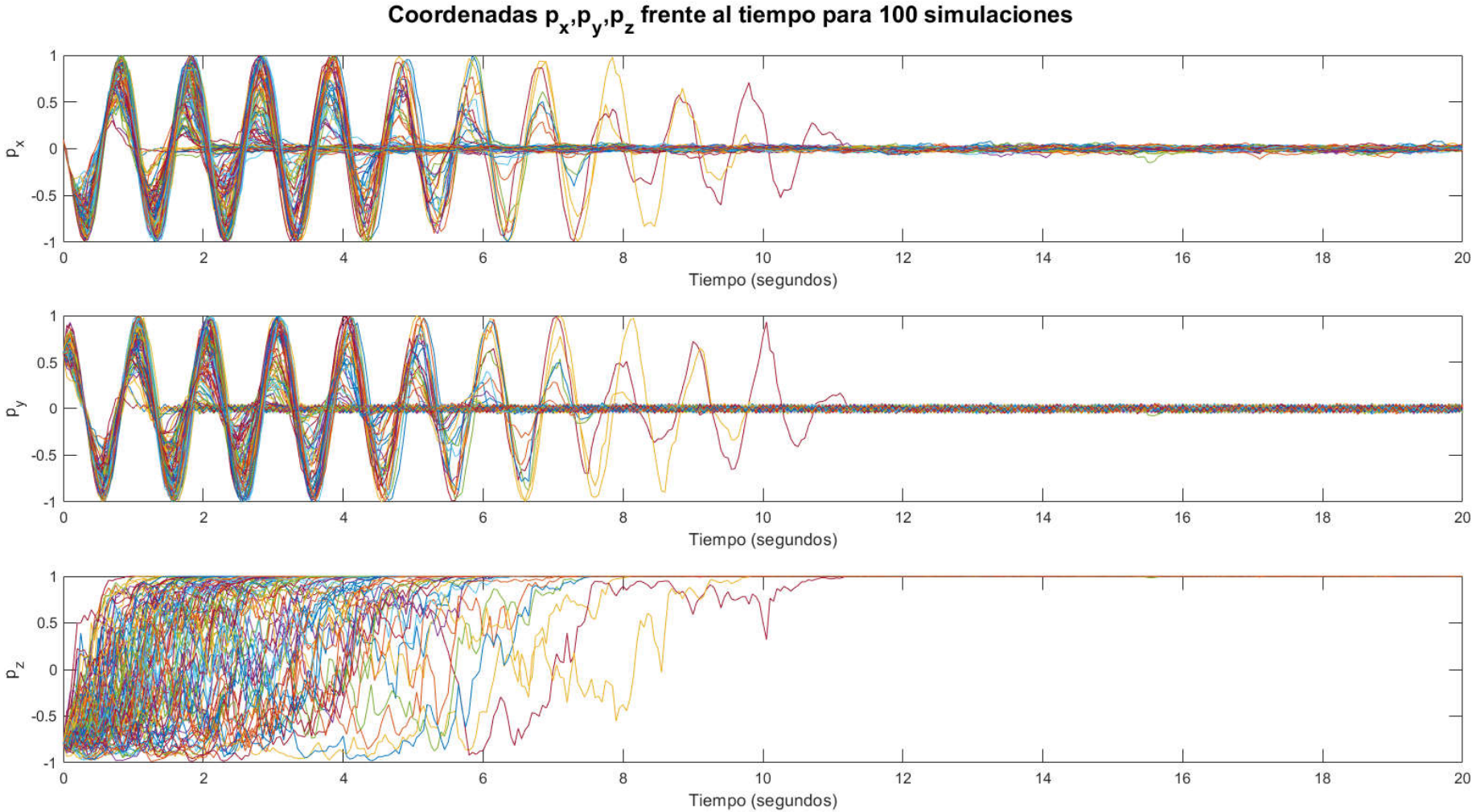

Figura 17. Simulación del qubit controlado. Componentes del vector de Bloch, a partir de implementación propia, mediante Euler - Maruyama, para 100 simulaciones.

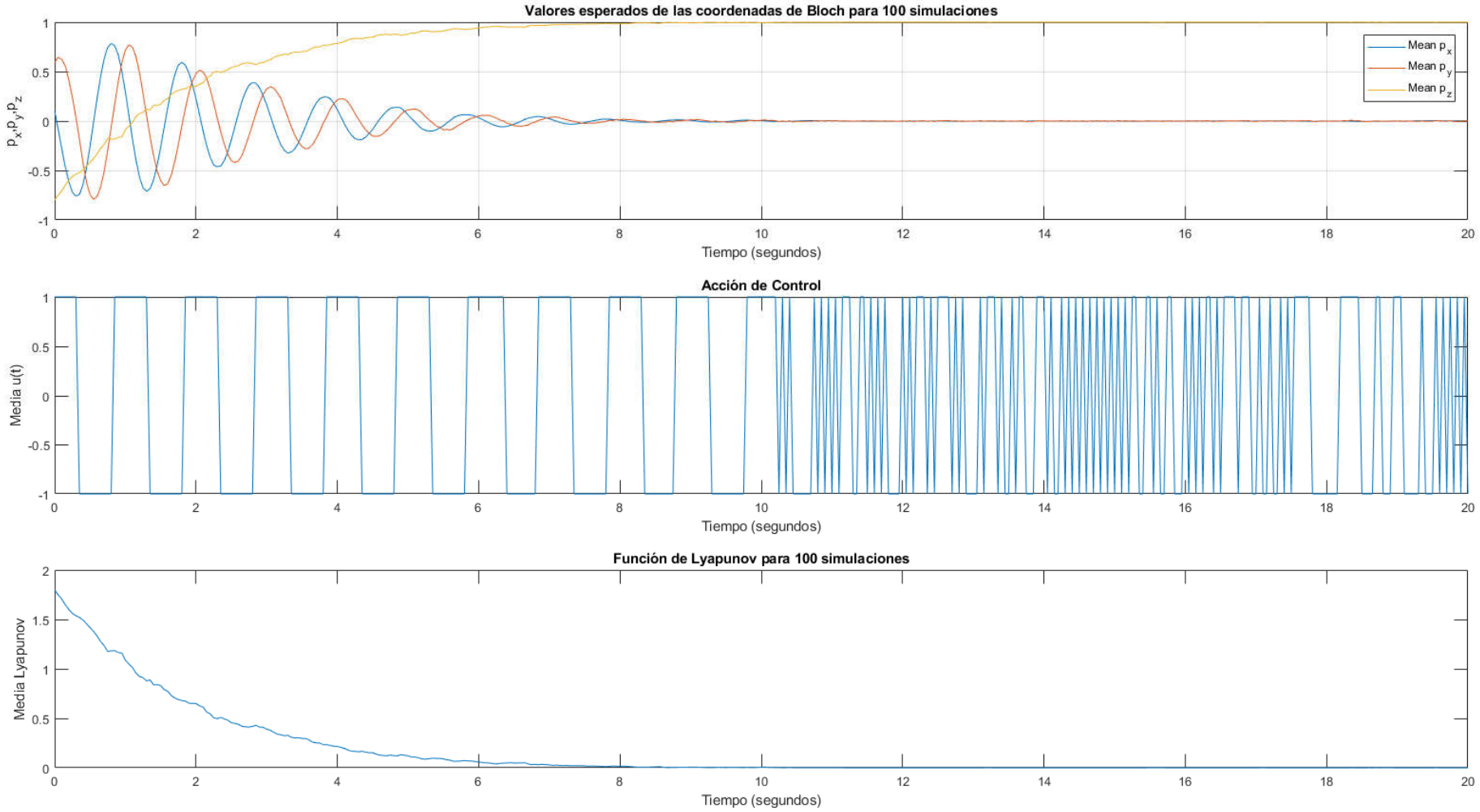

Figura 18. Simulación del qubit controlado. Coordenadas del vector de Bloch, acción de control y función de Lyapunov, a partir de implementación propia, mediante Euler - Maruyama, para 100 simulaciones.

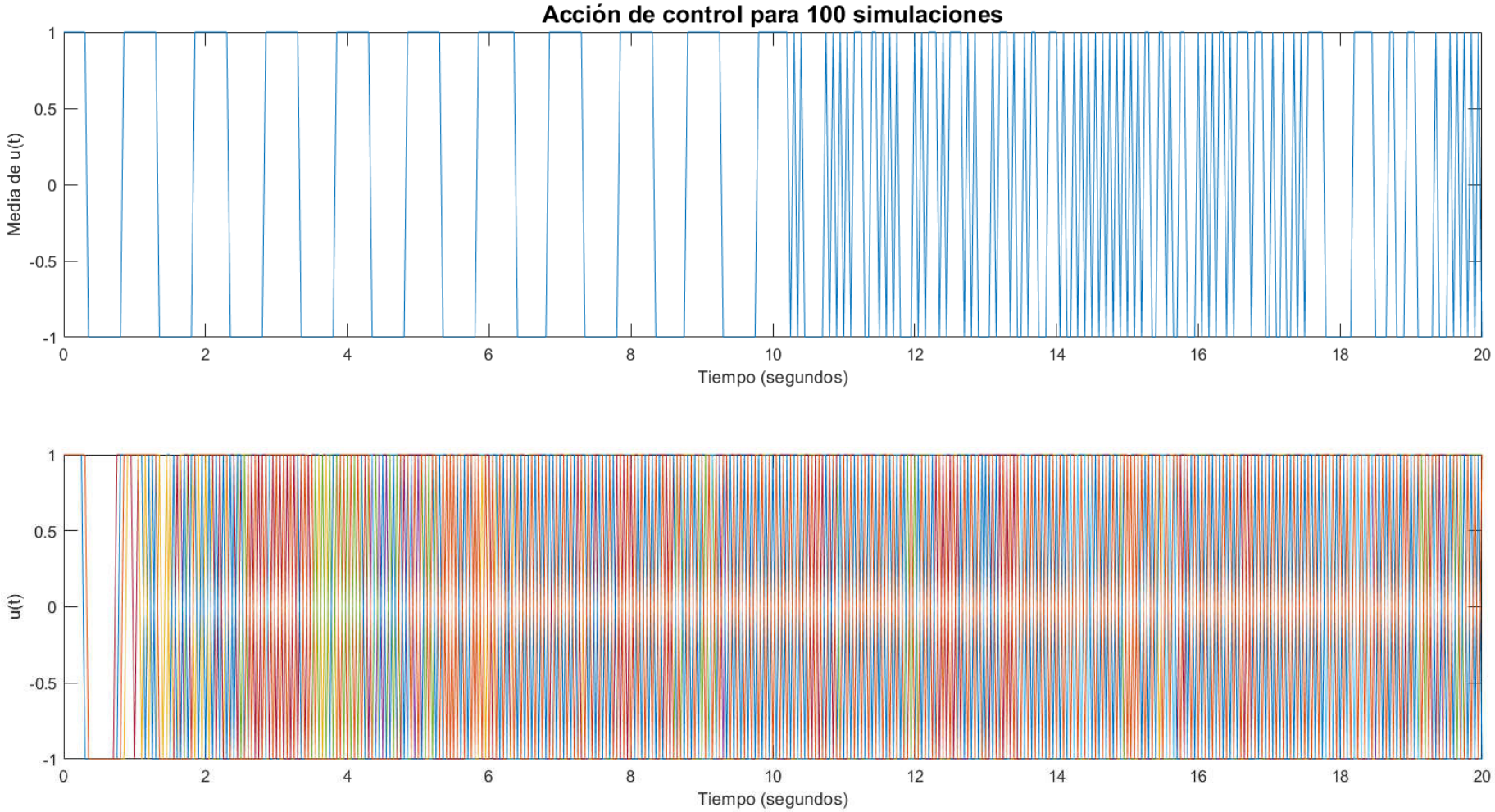

Figura 19. Simulación del qubit controlado. Valores medios y reales de la acción de control, a partir de implementación propia, mediante Euler - Maruyama, para 100 simulaciones.

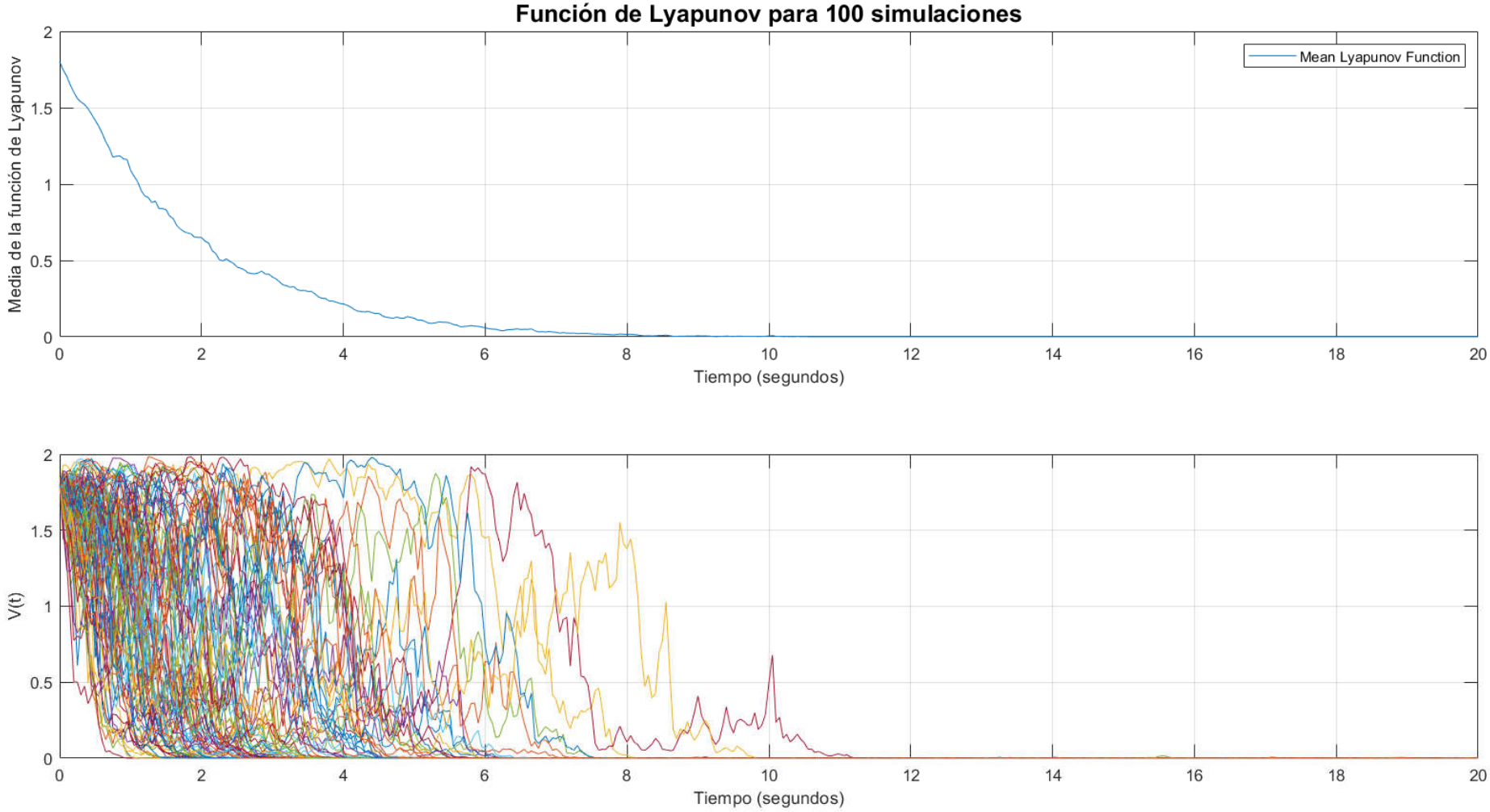

Figura 20. Simulación del qubit controlado. Valores medios y reales de la función de Lyapunov, a partir de implementación propia, mediante Euler - Maruyama, para 100 simulaciones.

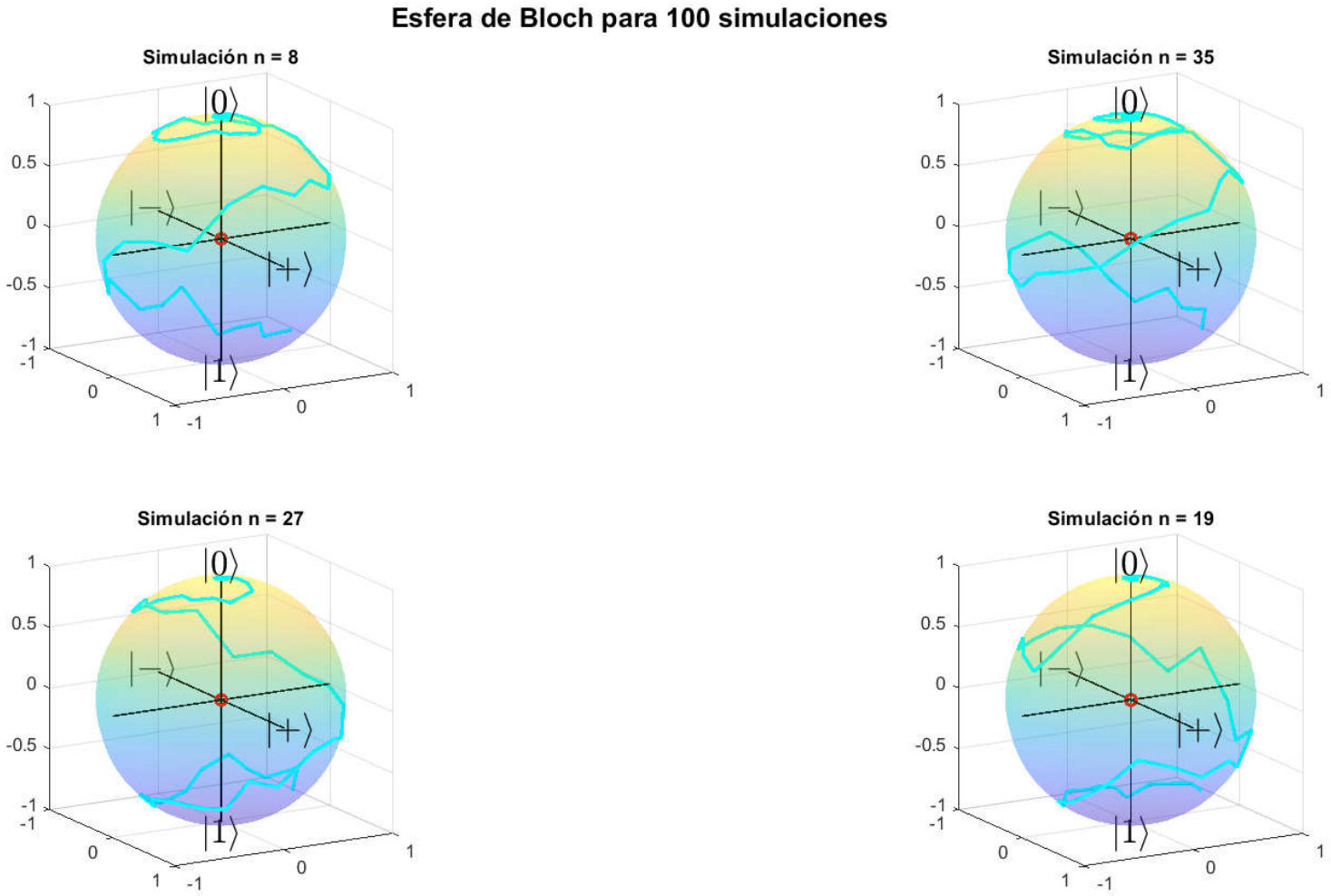

Figura 21. Simulación del qubit controlado. Representación en la esfera de Bloch, a partir de implementación propia, mediante Euler - Maruyama, para 100 simulaciones.

4.2.1.2. Método de Milstein

En la Figura 22 se puede observar las distintas componentes del vector de Bloch a partir del método de Milstein. A diferencia de lo que ocurría con el método de Euler – Maruyama, estas componentes sufren grandes dispersiones, llegando en algunas trayectorias a no alcanzar el estado deseado.

En la Figura 23 se representan los valores medios esperados de las coordenadas de Bloch junto con la media de la acción de control y de la función de Lyapunov. Donde se puede apreciar que, al aproximarse al estado objetivo, la función de Lyapunov decrece exponencialmente, hasta que la distancia al estado deseado es nula. La acción de control, una vez alcanzado el estado objetivo, como ocurría en el caso de Euler – Maruyama, sigue activa, ya que el sistema posee pequeñas perturbaciones. Para una visualización de todas las simulaciones realizadas, la Figura 24 muestra los valores obtenidos de la acciónde control, mientras que la Figura 25 muestra los de la función de Lyapunov.

Sobre la representación de algunas simulaciones en la esfera de Bloch (Figura 26), podemos observar que a algunas les supone un esfuerzo alcanzar el estado objetivo si comparamos con las obtenidas con el método de Euler – Maruyama.

Aunque con el método de Milstein se ha obtenido una disipación de energía en el sistema y el promediado de las componentes del vector de Bloch alcanzan el polo norte o estado deseado, para el controlador de Florchinger obteníamos mejores resultados con el método de resolución de Euler – Maruyama que ofrece el paquete Financial Toolbox™ de MATLAB®.

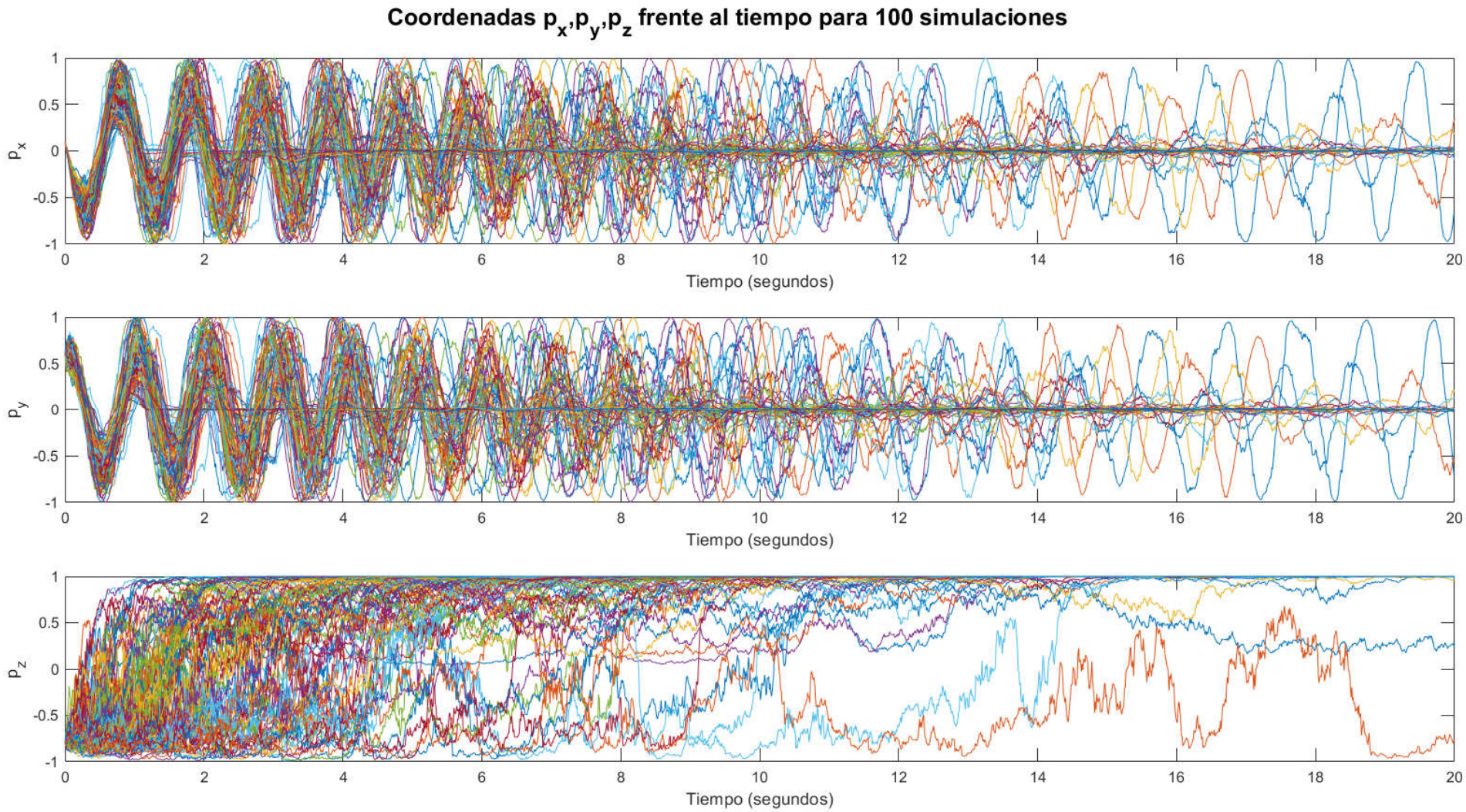

Figura 22. Simulación del qubit controlado. Componentes del vector de Bloch, mediante Milstein, para 100 simulaciones.

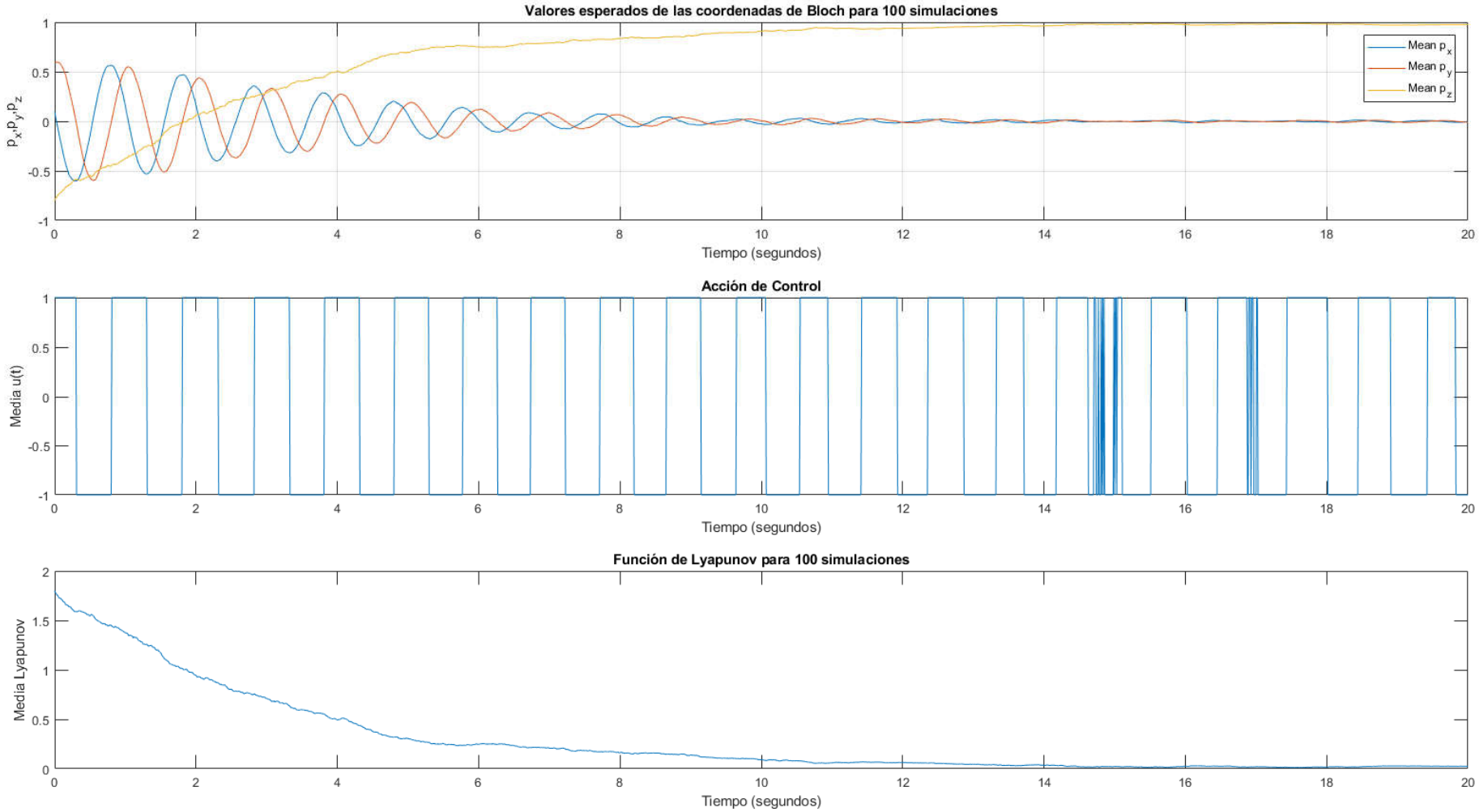

Figura 23. Simulación del qubit controlado. Coordenadas del vector de Bloch, acción de control y función de Lyapunov, mediante Milstein, para 100 simulaciones.

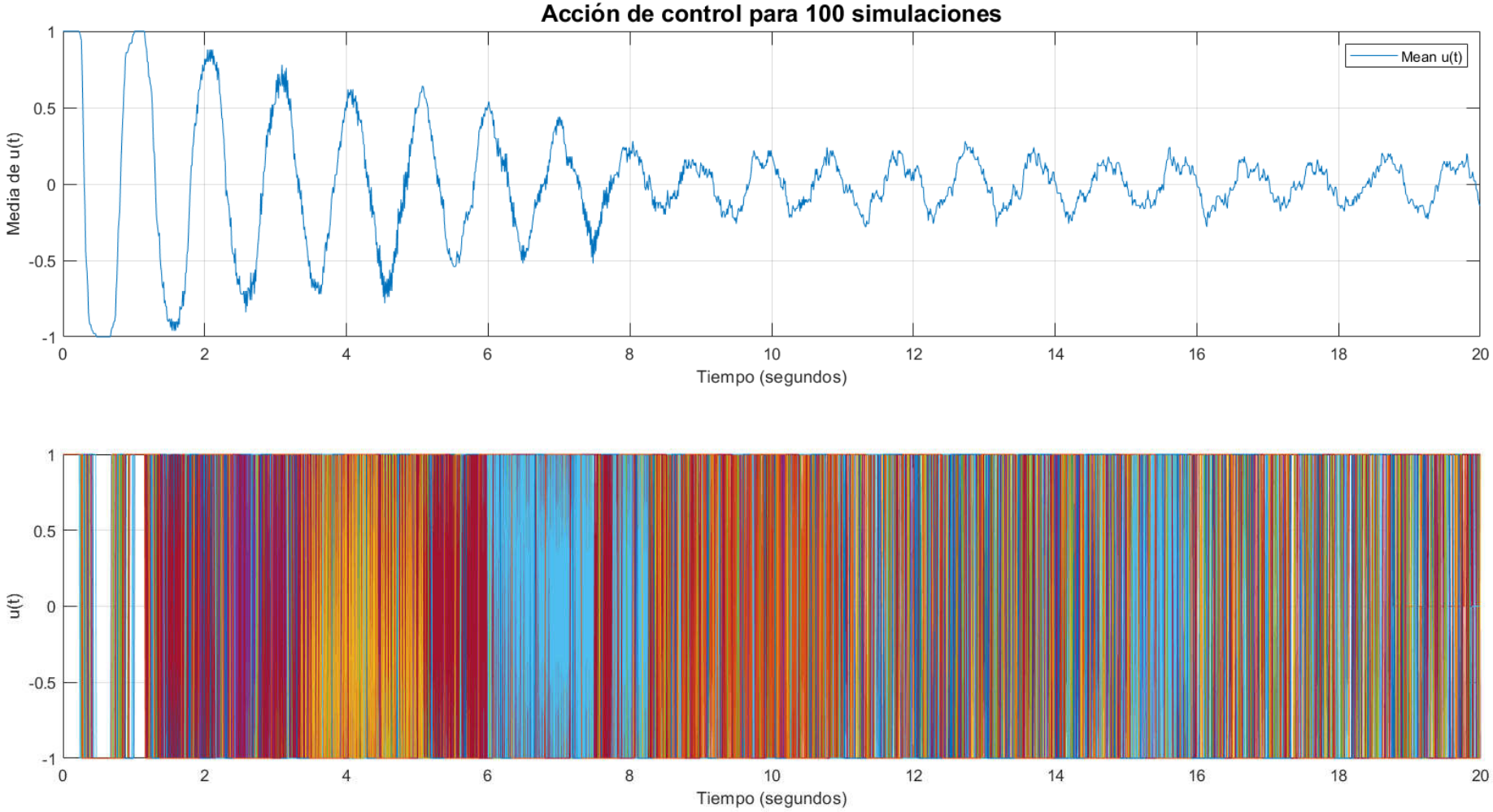

Figura 24. Simulación del qubit controlado. Valores medios y reales de la acción de control, mediante Milstein, para 100 simulaciones.

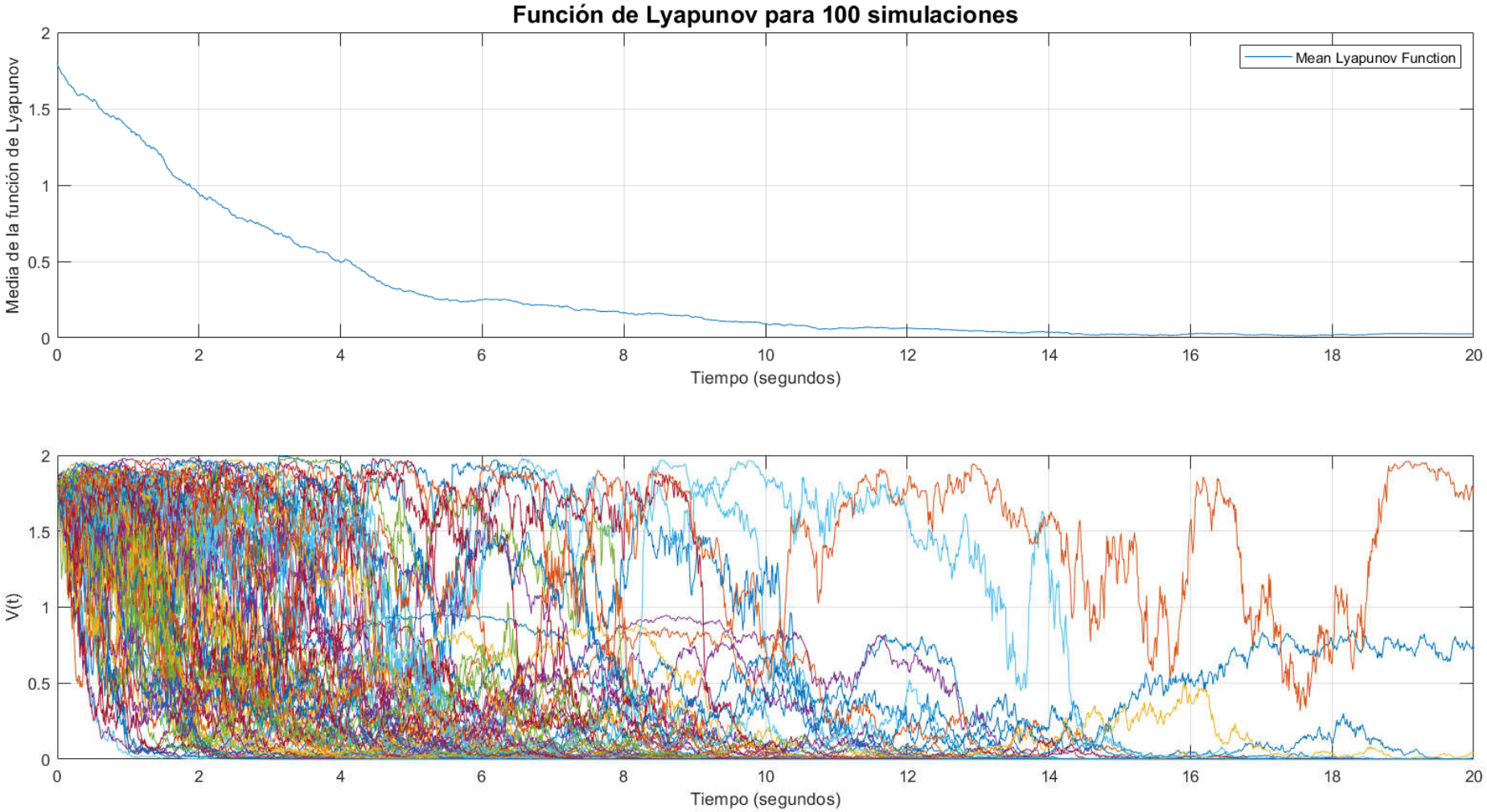

Figura 25. Simulación del qubit controlado. Valores medios y reales de la función de Lyapunov, mediante Milstein, para 100 simulaciones.

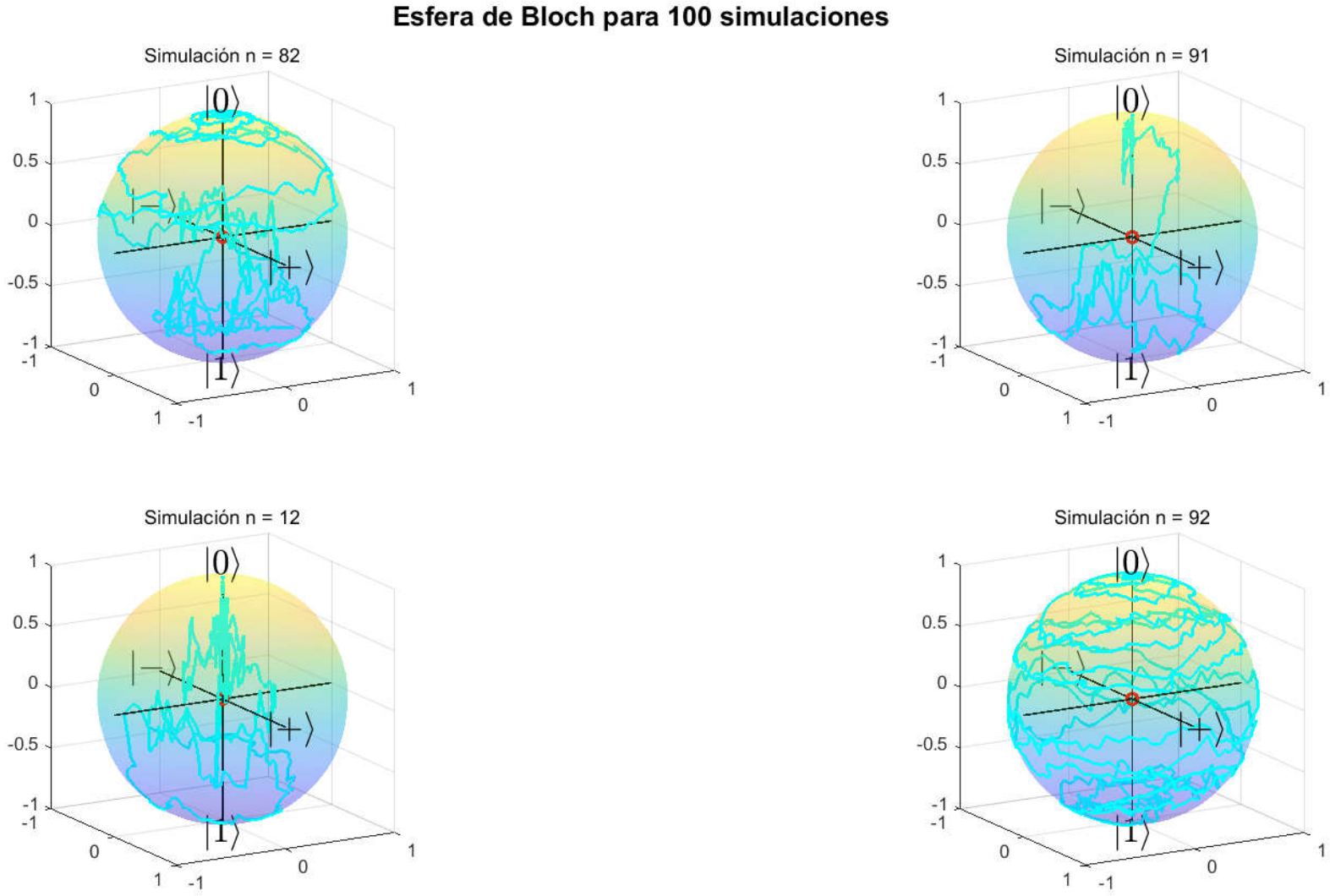

Figura 26. Simulación del qubit controlado. Representación en la esfera de Bloch, mediante Milstein, para 100 simulaciones.

4.2.2. Controlador de J. Ignacio Mulero

En este apartado se exponen los distintos resultados obtenidos a partir del segundo controlador empleado, propuesto por J. Ignacio Mulero, cuya explicación teórica se muestra en el capítulo 3, apartado 3.2.2.

Para este controlador, se han usado los mismos parámetros explicados al inicio del capítulo, usando como métodos de resolución los de Euler – Maruyama y los de Milstein.

4.2.2.1. Método de Euler – Maruyama

Para este método se han empleado los mismos métodos que se han utilizado anteriormente, estos son la que incorpora el paquete Financial Toolbox™ de MATLAB® y mediante implementación propia.

Resolución a través de Financial Toolbox™ de MATLAB®

Si tomamos como punto inicial el punto marcado al inicio del capítulo, es decir, un punto alejado del polo norte, nos encontramos con el problema mostrado en la Figura 27, en donde la simulación deja de muestrear a partir de un cierto tiempo. Esto es debido a que la estabilidad del sistema es local, es decir, la convergencia se produce para puntos cercanos al equilibrio. Por otro lado, el paquete Financial Toolbox™ no permite la normalización de los valores para este tipo de controlador.

No obstante, este método de resolución no deja de ser válido, ya que cabe destacar que sí se obtenían resultados al situar el punto inicial próximo al polo norte. Debido a esto, para puntos cercanos del polo norte objetivo, el método de Euler – Maruyama a partir de Financial Toolbox™ resultaba ser exitoso para este tipo de controlador.

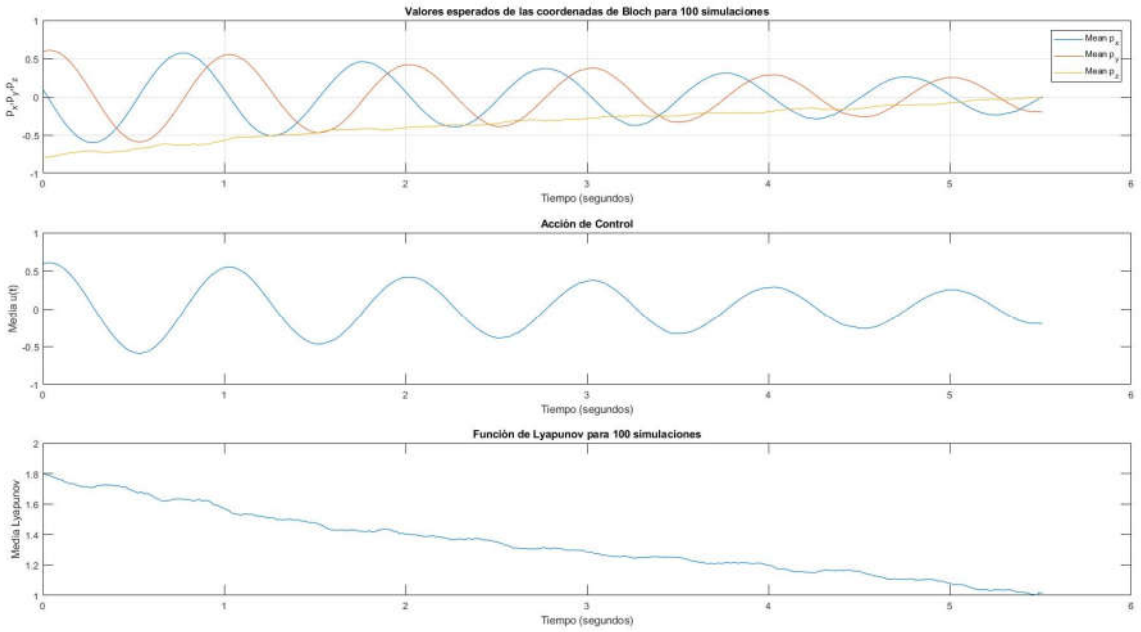

Figura 27. Simulación del qubit controlado. Coordenadas del vector de Bloch, acción de control y función de Lyapunov, mediante Milstein, para 100 simulaciones.

Resolución a través de implementación propia

A continuación, se muestran los resultados obtenidos del método de Euler – Maruyama mediante implementación propia situando el punto inicial alejado del polo norte.

Las componentes del vector de Bloch para las 100 simulaciones realizadas se muestran en la Figura 28. En esta se puede apreciar que todas las componentes presentan algunas trayectorias que no se llegan a estabilizar a tiempo final, aun así, la tendencia del sistema parece indicar que, con el paso del tiempo, el sistema concurre en el estado objetivo, ya que el número de trayectorias que no alcanzan ese estado es menor.

Comparando los valores medios esperados de las coordenadas de Bloch junto con la media de la acción de control y de la función de Lyapunov (Figura 29), observamos que, la media de las distintas componentes sí que alcanzan el estado objetivo, haciendo de esta manera, que la señal promediada de control y de Lyapunov se aproximen lentamente en torno al valor cero. Estos comportamientos se pueden observar también a nivel global, es decir, para todas las simulaciones realizadas, por un lado, la Figura 30 para la acción de control empleada, y, por otro lado, la Figura 31, para la función de Lyapunov. La acción de control, a tiempos pequeños presenta grandes oscilaciones, disminuyendo conforme avanza el tiempo, ya que las componentes del vector de Bloch se van acercando al polo norte. Asimismo, con respecto a la función de Lyapunov, la

distancia que nos marca decrece exponencialmente. No obstante, si tomamos en cuenta la acción de control y de Lyapunov para las 100 trayectorias, se puede apreciar que existen algunos casos en donde estas no decrecen, mostrando algunas veces picos espurios.

Finalmente, en la Figura 32 se ilustran algunas de las simulaciones en la hiperesfera de Bloch, seleccionadas aleatoriamente. De esta, se comprueba que todas ellas alcanzan el polo norte.

Realizando la simulación a través el método de Euler – Maruyama mediante implementación propia se concluye que, a partir de esta implementación, los comportamientos obtenidos del modelo del qubit son satisfactorios tanto para puntos alejados como cercanos al polo objetivo.

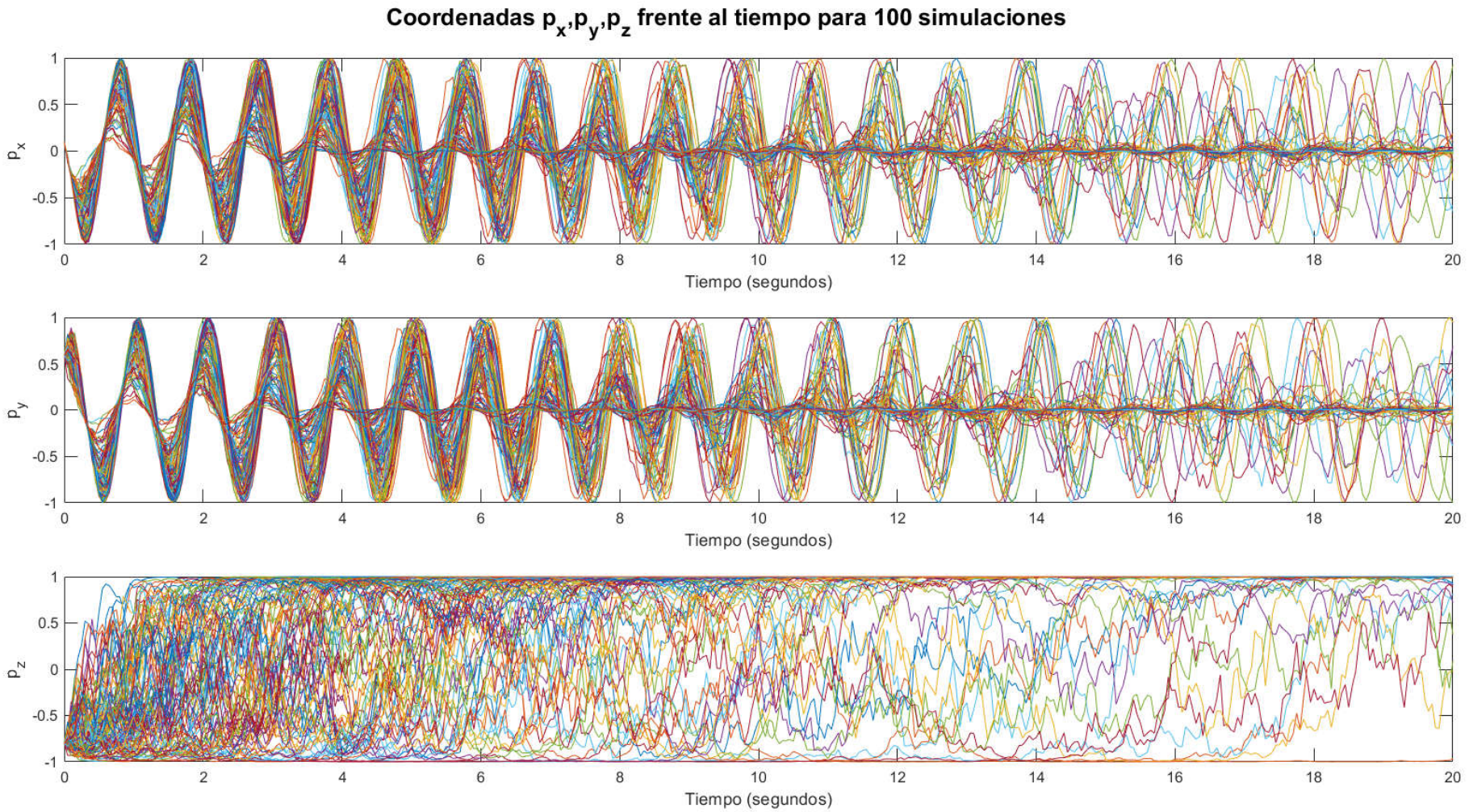

Figura 28. Simulación del qubit controlado. Componentes del vector de Bloch, a partir de implementación propia, mediante Euler - Maruyama, para 100 simulaciones.

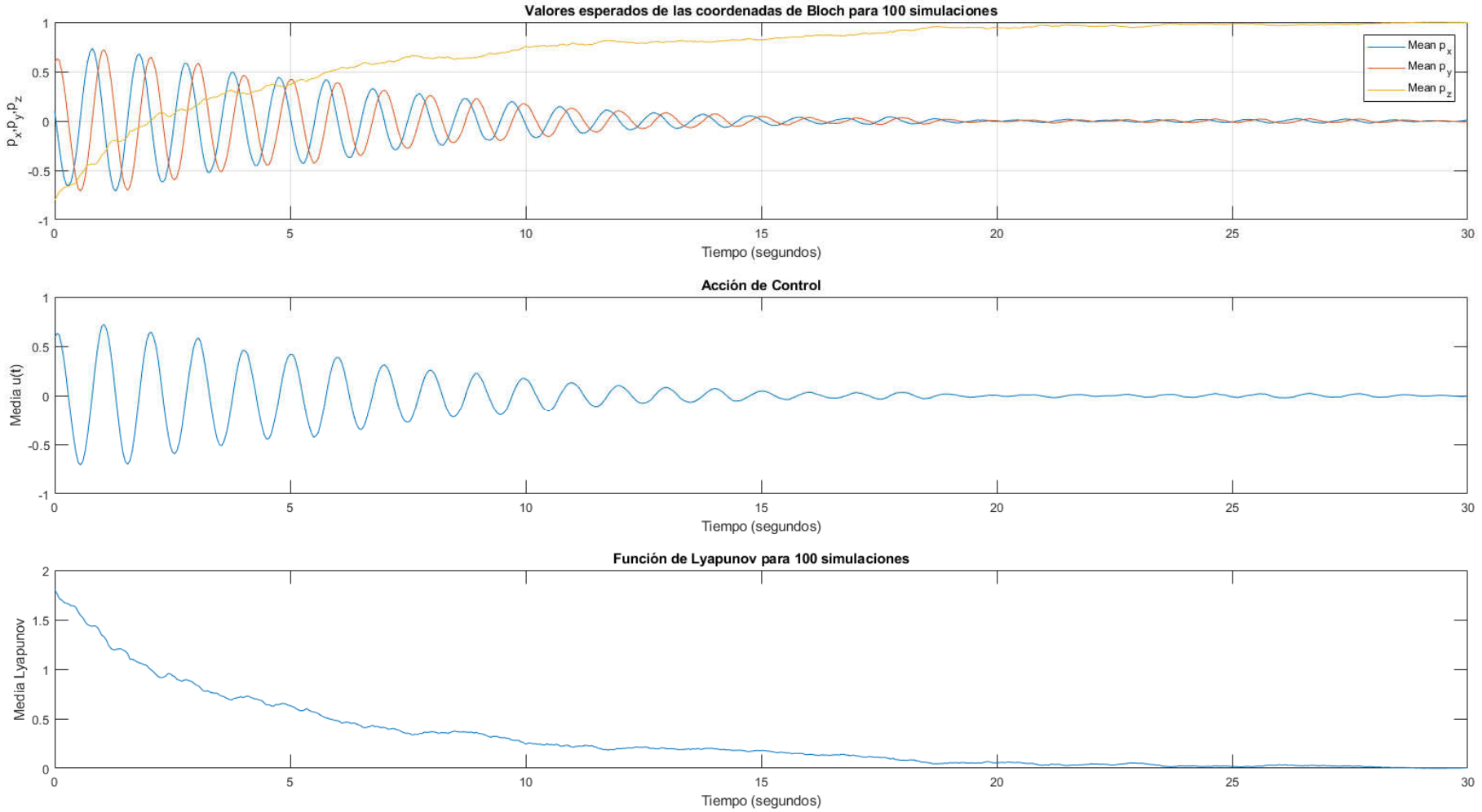

Figura 29. Simulación del qubit controlado. Coordenadas del vector de Bloch, acción de control y función de Lyapunov, a partir de implementación propia, mediante Euler - Maruyama, para 100 simulaciones.

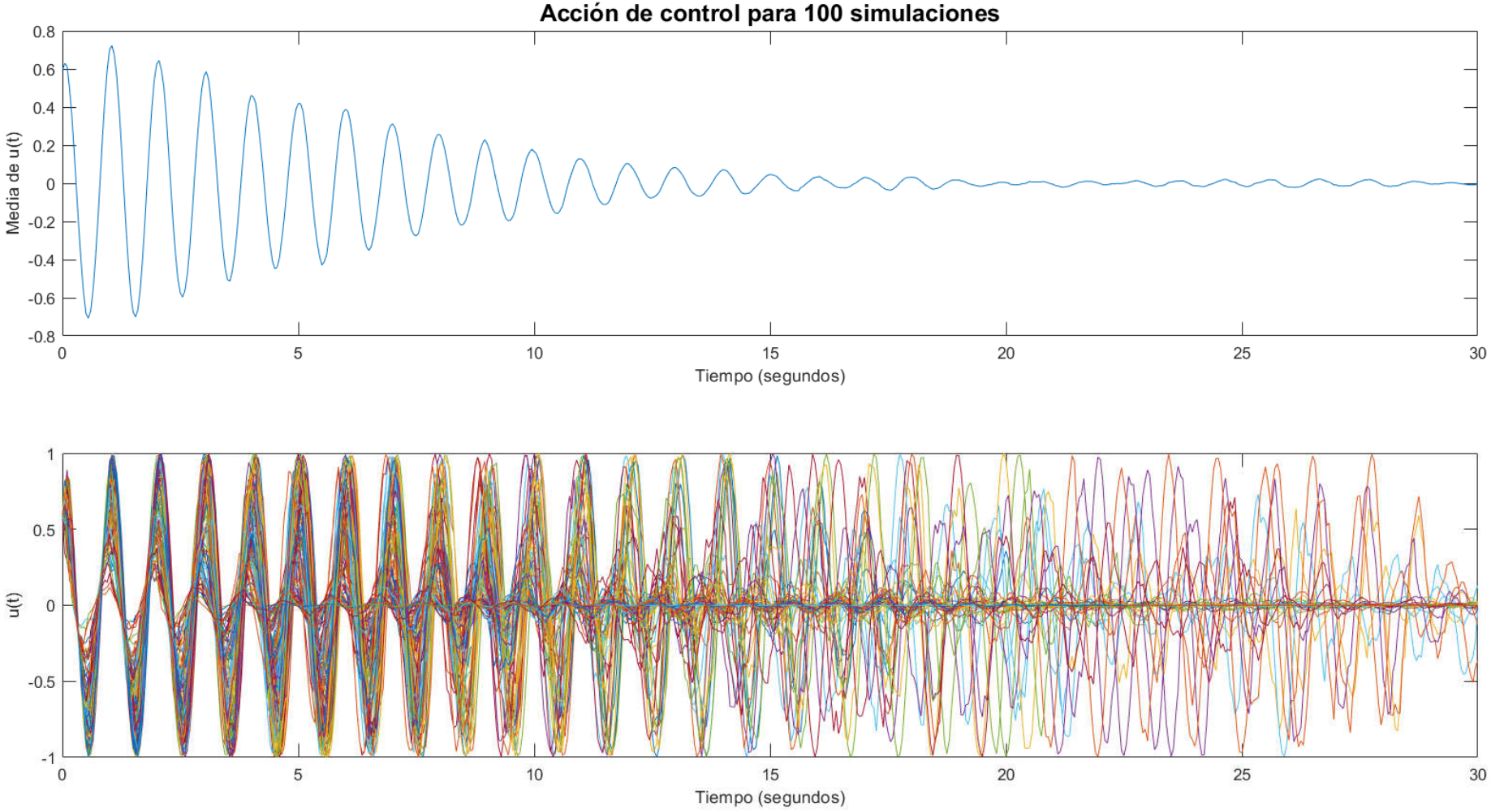

Figura 30. Simulación del qubit controlado. Valores medios y reales de la acción de control, a partir de implementación propia, mediante Euler - Maruyama, para 100 simulaciones.

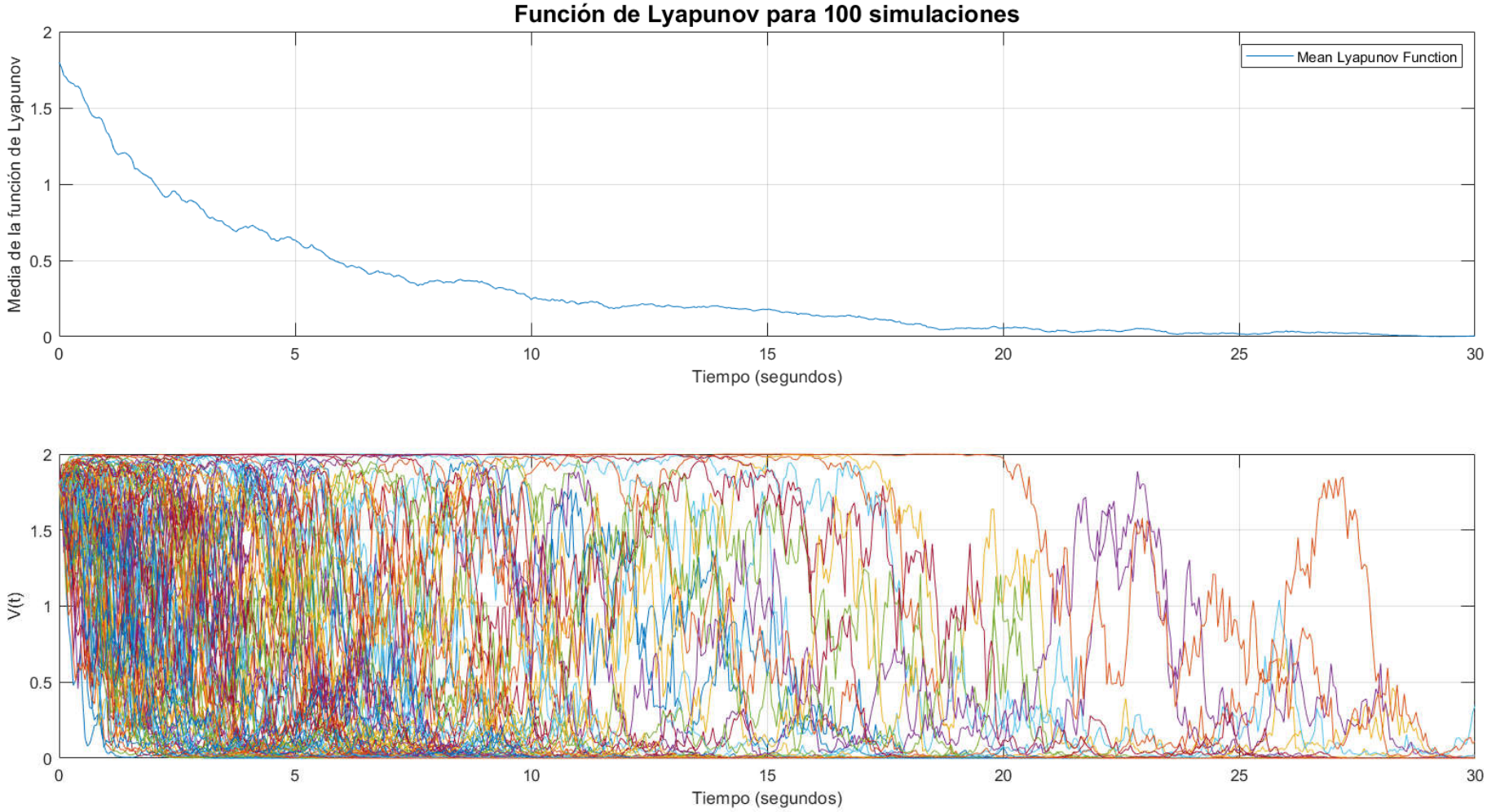

Figura 31. Simulación del qubit controlado. Valores medios y reales de la función de Lyapunov, a partir de implementación propia, mediante Euler - Maruyama, para 100 simulaciones.

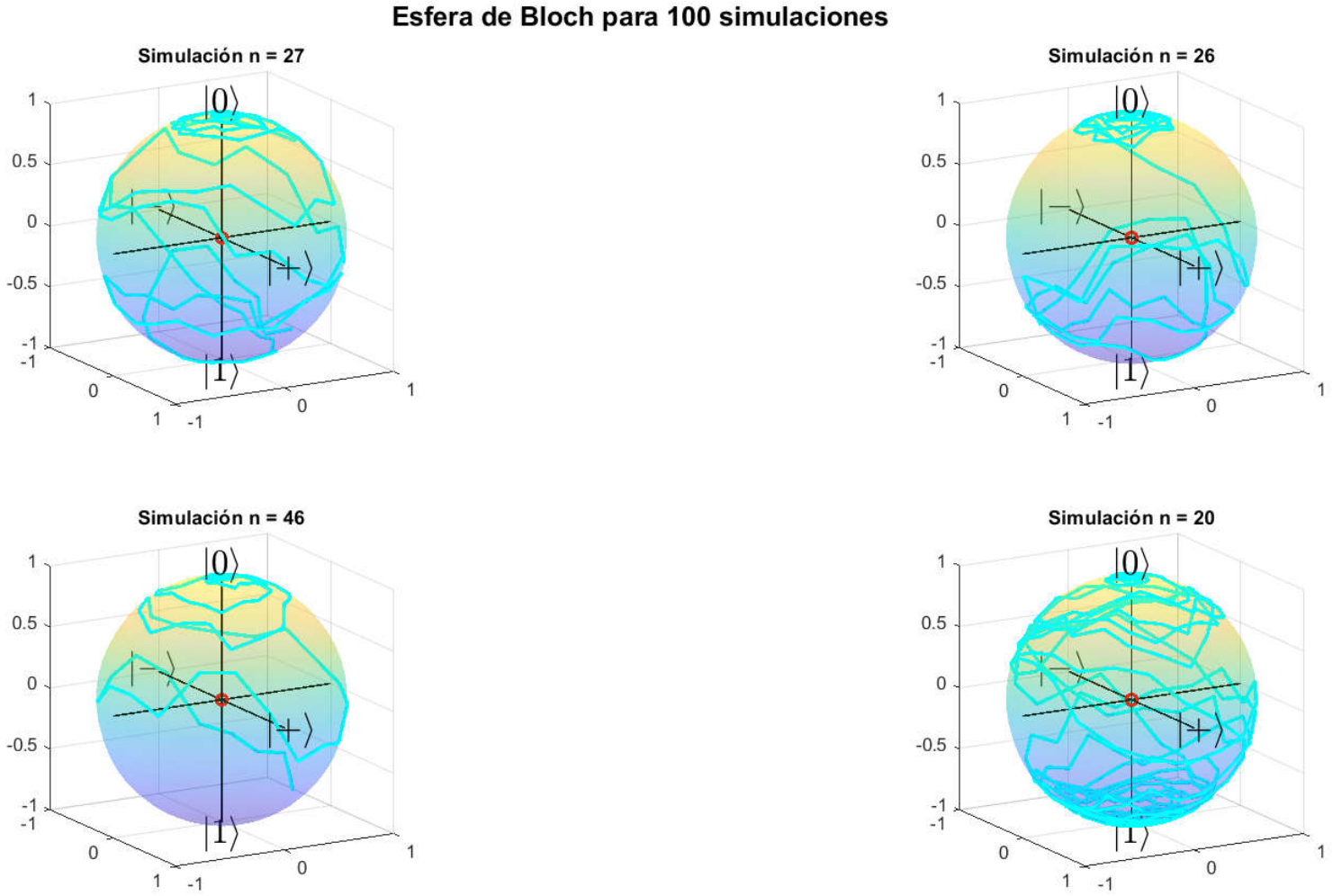

Figura 32. Simulación del qubit controlado. Representación en la esfera de Bloch, a partir de implementación propia, mediante Euler - Maruyama, para 100 simulaciones.

4.2.2.2. Método de Milstein

Con la resolución del segundo controlador con el método de Milstein a través de la herramienta SDETools con el punto inicial lejano al punto deseado, ocurría algo similar al obtenido anteriormente con el Euler – Maruyama mediante el paquete Financial Toolbox™. Este comportamiento se puede observar en la Figura 33.

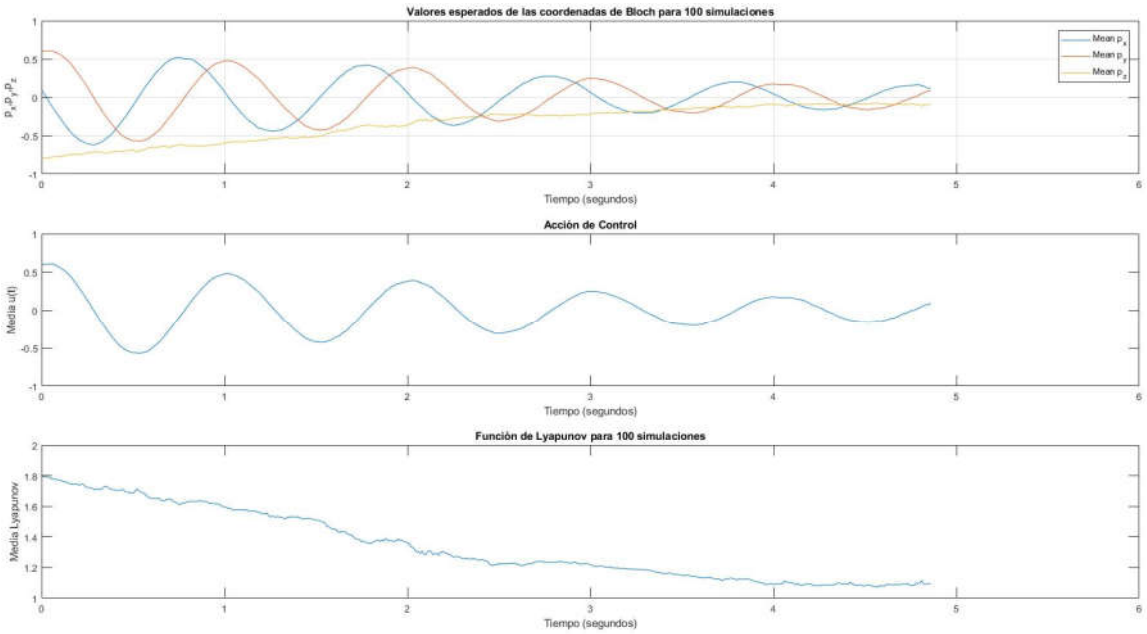

Figura 33. Coordenadas del vector de Bloch, acción de control y función de Lyapunov, mediante Milstein, para 100 simulaciones.

Dado que los métodos ofrecidos por los paquetes Financial Toolbox™ y SDETools no permiten la correcta normalización en los valores, la estabilidad del sistema es solamente local, a diferencia del método de resolución por implementación propia. Este último método se muestra superior a la hora de simular el sistema dado que permite utilizar puntos iniciales situados en cualquier zona de la esfera, independientemente de su distancia al estado objetivo.

Capítulo 5

Conclusiones

Una vez descritos en capítulos anteriores la teoría necesaria para el entendimiento del qubit y de los distintos modelos desarrollados e implementados para el diseño, simulación y control del qubit, se realiza en este capítulo una valoración de los distintos controladores y métodos de resolución empleados para su control. El objetivo de este trabajo fin de estudios consistía en realizar una continuación del trabajo fin de máster realizado por Sebastián García [3], en donde llevó a cabo el estudio del control realimentado estocástico de qubits cuánticos.

1. Como toda actividad técnica, ésta se ha iniciado con la identificación y definición del problema y con la necesaria revisión de las bases del control cuántico. Como consecuencia de esta revisión se puede concluir que los sistemas cuánticos son muy susceptibles a los cambios externos, por lo que generalmente se idealizan en sistemas cuánticos cerrados, presentando un aislamiento completo. En la naturaleza, sin embargo, esto rara vez sucede porque existen interacciones inevitables entre el sistema de interés y el medio. Por tanto, es necesario utilizar métodos que permitan la eliminación del entorno de las ecuaciones de la dinámica del sistema de estudio. Es por ello por lo que este trabajo fin de estudios se ha realizado a partir de un sistema no lineal abierto.
2. Para el control del modelo del qubit se propusieron inicialmente dos controladores, el primero el propuesto por Florchinger, y el segundo propuesto por J. Ignacio Mulero. El estudio del modelo del qubit controlado se llevó a cabo a partir de dos métodos de resolución de ecuaciones diferenciales estocásticas diferentes, el método de Euler – Maruyama y el método de Milstein. Estos métodos se han programado a partir de tres formas diferentes, por un lado, el método de Euler – Maruyama se resolvió a partir del paquete Financial Toolbox de MATLAB® y por un método de implementación propia, mientras que para el método de Milstein se hizo uso del paquete SDETools proporcionado por Andrew Horchler.
3. Para el caso del sistema no controlado, se ha comprobado que el sistema no llega a estabilizarse en el punto deseado.

4. Respecto al controlador propuesto por Florchiner, para el método de Euler – Maruyama a partir del paquete Financial Toolbox de MATLAB®, se puede concluir que la simulación es satisfactoria. El comportamiento del sistema es el deseado, ya que se estabiliza a tiempo infinito al punto norte objetivo. Por otro lado, la función de Lyapunov tiende al valor cero, sin presentar ninguna perturbación. Además, de la ley de control se observa que esta sigue un buen comportamiento.
5. Respecto al controlador propuesto por Florchiner, para el método de Euler – Maruyama a partir del método de implementación propia también se alcanzan los valores objetivos, notándose algunas pequeñas diferencias en las formas de resolución mediante Financial Toolbox.
6. Respecto al controlador propuesto por Florchiner, para el método de Milstein las perturbaciones suelen acentuarse un poco más, en donde también se puede observar que la función de Lyapunov no llega a 0. Esto nos indica que realmente no se alcanzó el estado objetivo, aunque sí que se aproxima a él.
7. Se puede concluir que el controlador de Florchinger tiene un comportamiento adecuado si se aplica el método de resolución de Financial Toolbox de MATLAB®. Sin embargo, también podemos considerar como válido el método de implementación propia, ya que, aunque el sistema presentaba de esta forma más perturbaciones, seguía alcanzándose el polo norte. El método de Milstein resultó ser el que peor resultados arrojó.
8. Respecto al controlador propuesto por J. Ignacio Mulero, para el método de Euler – Maruyama ofrecida por el paquete Financial Toolbox de MATLAB® y mediante Milstein a partir de SDETools no se obtuvieron resultados válidos para puntos iniciales alejados del polo norte objetivo. La convergencia sólo se produce para puntos cercanos al equilibrio, como consecuencia de la estabilidad local del sistema.
9. Respecto al controlador propuesto por J. Ignacio Mulero, el método de Euler – Maruyama a partir de la implementación propia resulta ser válido para puntos alejados. Aunque algunas trayectorias no llegan a estabilizarse a tiempo final, el sistema concurre en el estado objetivo. Por otro lado, la señal de control y la función de Lyapunov se aproximan lentamente en torno al valor cero.
10. El análisis llevado a cabo da lugar a un buen comportamiento del controlador de J. Ignacio Mulero con el método de resolución de Euler – Maruyama mediante implementación propia, tanto para puntos alejados como cercanos al polo norte

objetivo. No obstante, cabe destacar también que los métodos de resolución que ofrece el paquete Financial Toolbox de MATLAB® y SDETools ofrecen resultados satisfactorios para puntos próximos al objetivo.

Tras las conclusiones marcadas anteriormente, este trabajo fin de máster podría abrir las puertas a otro tipo de trabajos futuros de ingeniería de control cuántico que despiertan cierto interés. Por ejemplo, la introducción de perturbaciones sobre el sistema, especialmente en aquellos casos en donde los resultados eran exitosos, para poder comprobar si el sistema se comporta adecuadamente ante ciertos imprevistos. Por otro lado, también resultaría interesante la implementación de controladores que garanticen la estabilidad global o la ampliación a sistemas de más de un qubit.

Apéndices

Apéndice A

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% Simulation of the first controller %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%% Financial Toolbox & SDETools %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% Yolanda Villalba Celdran %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc; clear; close all; clear all;
%% Initial parameters
Tmes = 1; % measurement time constant
gamma_m = 1/Tmes; % Spontaneous emission rate
h = 1; % Planck constant
w_eg = 2*pi/Tmes; % Natural frequency
eta = 1; % Efficiency of the detector
dt = 0.01; % step
T = 20*Tmes; % Terminal time
rng('default'); % regenerate random seed
Xeq = [0;0;1]; % X objective in equilibrium

%% Advanced control of a qubit
% f(x) & g(x)function
% px = X(1); py = X(2); pz = X(3);
f = @(t,X) [-gamma_m/2,-w_eg/h,0;w_eg/h,-gamma_m/2,0;0,0,0]*X;
g = @(t,X) [0;-X(3)/h;X(2)/h];

% Diffusion Term
sigma = @(t,X) [-sqrt(eta*gamma_m)*X(1)*X(3);
    -sqrt(eta*gamma_m)*X(2)*X(3);
   - sqrt(eta*gamma_m)*((X(3)^2)-1)];

% Drift Term
%a = @(t,X) (X-Xeq)'*f(t,X)+sigma(t,X)'*sigma(t,X)/2;
%b = @(t,X) (X-Xeq)'*g(t,X);

H = diag([0,0,0]);
grad = @(t,X) -[0;0;1];
a = @(t,X) grad(t,X)'*f(t,X)+sigma(t,X)'*H*sigma(t,X)/2;
b = @(t,X) grad(t,X)'*g(t,X);

ro = @(t,X) (b(t,X)~=0 ||
a(t,X)>=0)*(a(t,X)+sqrt(a(t,X)^2+b(t,X)^2))/b(t,X);
u = @(t,X) -(X(3)<1)*ro(t,X); % Controlled system
%u = @(t,X) 0; % Uncontrolled system

drift = @(t,X) f(t,X)+g(t,X)*u(t,X);

%% Simulation of the system
% Building of the model
%x0=[0.1;sqrt(1-0.1^2-0.9^2);0.9];
x0 = [0.1;sqrt(1-0.1^2-(-0.8)^2);-0.8]; % Initial point
nPaths = 100; % nTrials or nPaths

% Simulation methods
```

```
% From Financial Toolbox MATLAB: Euler - Maruyama Method
obj = sde(drift,sigma,'StartTime',0,'StartState',x0);
[X2,time,dW] = obj.simulate(T/dt,'DeltaTime',dt,'nTrials',
nPaths,'Antithetic', true);

% From SDETools: Milstein Method
% time=0:dt:T;
% opts = sdeset('SDEType','Ito','RandSeed',3);
%
% for i = 1:nPaths
%     X = sde_milstein(drift,sigma,time,x0,opts);
%
%     opts = sdeset(opts, 'RandSeed', 3+i);
%     X2(:,:,i) = X;
% end

% Normalizing the vector
for i = 1:nPaths
    for j = 1:length(time)
        if norm(X2(j,:,i)) > 1
        X2(j,:,i) = X2(j,:,i)/norm(X2(j,:,i));
        end
    end
end

% Resizing vectors
px = reshape(X2(:,1,:),[length(time),nPaths]);
py = reshape(X2(:,2,:),[length(time),nPaths]);
pz = reshape(X2(:,3,:),[length(time),nPaths]);

% Computing mean values for the stochastic state paths
m_norm_X = mean(X2,3);
m_px = mean(px,2);
m_py = mean(py,2);
m_pz = mean(pz,2);

% Computing Lyapunov function
lyap = 1-pz;
m_lyap = mean(lyap,2);

u_values = zeros(length(time),nPaths);

for i = 1:nPaths % Depth of X2
    for j = 1:length(time) % Rows of X2
        f_values = [-gamma_m/2,-w_eg/h,0;w_eg/h,-
gamma_m/2,0;0,0,0]*[px(j,i);py(j,i);pz(j,i)];
        g_values = [0;-pz(j,i)/h;py(j,i)/h];
        sigma_values = [-sqrt(eta*gamma_m)*px(j,i)*pz(j,i);
            -sqrt(eta*gamma_m)*py(j,i)*pz(j,i);
            - sqrt(eta*gamma_m)*((pz(j,i)^2)-1)];

        a_values = -[0;0;1]'*f_values+sigma_values'*H*sigma_values/2;
        b_values = -[0;0;1]'*g_values;

        ro_values(j,i) = (b_values~=0 ||
a_values>=0)*(a_values+sqrt(a_values^2+b_values^2))/b_values;
        u_values(j,i) = -(pz(j,i)<1)*ro_values(j,i);
        %u_values(j,i) = 0;
    end
end
```

```
m_u = mean(u_values,2);

%% Displaying output results
% p_x, p_y and p_z stabilization
figure;
plot(time,m_norm_X);
xlim([0 20]);
title(['Valores esperados de las Coordenadas de Bloch para
',num2str(nPaths),' simulaciones'], 'FontSize', 15);
xlabel('Tiempo (segundos)', 'FontSize', 11);
legend('Mean p_{x}','Mean p_{y}','Mean p_{z}');
grid;

% Stabilization & control signal
figure;
subplot(2,1,1);
plot(time,m_norm_X);
xlim([0 20]);
title(['Valores esperados de las coordenadas de Bloch para
',num2str(nPaths),' simulaciones'], 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_x,p_y,p_z');
grid on;
legend('Mean p_{x}','Mean p_{y}','Mean p_{z}');
subplot(2,1,2);
v=0*time;
for i=1:length(time)
    v(i)=u(i,m_norm_X(i,:)');
end
plot(time,v);
xlim([0 20]);
title('Acción de Control', 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('u(t)', 'FontSize', 11);

% Stabilization & Lyapunov function
figure;
subplot(2,1,1);
plot(time,m_norm_X);
xlim([0 20]);
title(['Valores esperados de las coordenadas de Bloch para
',num2str(nPaths),' simulaciones'], 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_x,p_y,p_z');
grid on;
legend('Mean p_{x}','Mean p_{y}','Mean p_{z}');
subplot(2,1,2);
plot(time,m_lyap);
xlim([0 20]);
title(['Función de Lyapunov para ', num2str(nPaths),' simulaciones'],
'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media de la función de Lyapunov', 'FontSize', 11);

% Stabilization & control signal & Lyapunov
figure;
subplot(3,1,1);
plot(time,m_norm_X);
xlim([0 20]);
title(['Valores esperados de las coordenadas de Bloch para
',num2str(nPaths),' simulaciones'], 'FontSize', 11);
```

```
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_x,p_y,p_z');
grid on;
legend('Mean p_{x}','Mean p_{y}','Mean p_{z}');
subplot(3,1,2);
v=0*time;
for i=1:length(time)
    v(i)=u(i,m_norm_X(i,:)');
end
plot(time,v);
xlim([0 20]);
title('Acción de Control', 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media u(t)', 'FontSize', 11);
subplot(3,1,3);
plot(time,m_lyap);
xlim([0 20]);
title(['Función de Lyapunov para ', num2str(nPaths),' simulaciones'],
'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media Lyapunov', 'FontSize', 11);

% p Bloch paths
figure;
subplot(3,1,1);
plot(time,px)
xlim([0 20]);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_{x}', 'FontSize', 11)
subplot(3,1,2);
plot(time,py)
xlim([0 20]);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_{y}', 'FontSize', 11)
subplot(3,1,3);
plot(time,pz)
xlim([0 20]);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_{z}', 'FontSize', 11)
sgtitle(['Coordenadas p_{x},p_{y},p_{z} frente al tiempo para ',
num2str(nPaths), ' simulaciones'],...
    'FontSize', 15, 'FontWeight','Bold');


% Lyapunov function
figure;
subplot(2,1,1);
plot(time,m_lyap);
xlim([0 20]);
title(['Función de Lyapunov para ', num2str(nPaths),' simulaciones'],
'FontSize', 15);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media de la función de Lyapunov', 'FontSize', 11);
grid on;
legend('Mean Lyapunov Function');
subplot(2,1,2);
plot(time,lyap);
xlim([0 20]);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('V(t)', 'FontSize', 11);
grid on;
```

```
% Control signal u(t,X)
figure
subplot(2,1,1);
plot(time,m_u);
xlim([0 20]);
title(['Acción de control para ', num2str(nPaths), ' simulaciones'],
'FontSize', 15);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media de u(t)', 'FontSize', 11);
grid on;
legend('Mean u(t)');
subplot(2,1,2);
plot(time,u_values);
xlim([0 20]);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('u(t)', 'FontSize', 11);
grid on;

% Bloch sphere
number_npath = randi([0, nPaths],1,4);

figure
subplot(2,2,1);
plotBlochSphere
hold on
plot3(X2(:,1,number_npath(1)),X2(:,2,number_npath(1)),X2(:,3,number_np
ath(1)), 'LineWidth', 2, 'Color', 'c')
title(['Simulación n = ',num2str(number_npath(1))], 'FontSize', 11,
'FontWeight','Normal')

subplot(2,2,2);
plotBlochSphere
hold on
plot3(X2(:,1,number_npath(2)),X2(:,2,number_npath(2)),X2(:,3,number_np
ath(2)), 'LineWidth', 2, 'Color', 'c')
title(['Simulación n = ',num2str(number_npath(2))], 'FontSize', 11,
'FontWeight','Normal')

subplot(2,2,3);
plotBlochSphere
hold on
plot3(X2(:,1,number_npath(3)),X2(:,2,number_npath(3)),X2(:,3,number_np
ath(3)), 'LineWidth', 2, 'Color', 'c')
title(['Simulación n = ',num2str(number_npath(3))], 'FontSize', 11,
'FontWeight','Normal')

subplot(2,2,4);
plotBlochSphere
hold on
plot3(X2(:,1,number_npath(4)),X2(:,2,number_npath(4)),X2(:,3,number_np
ath(4)), 'LineWidth', 2, 'Color', 'c')
title(['Simulación n = ',num2str(number_npath(4))], 'FontSize', 11,
'FontWeight','Normal')
sgtitle(['Esfera de Bloch para ', num2str(nPaths), ' simulaciones'],
'FontSize', 15, 'FontWeight','Bold');
```

Apéndice B

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% Simulation of the first controller %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% Self-implementation method %%%%%%%%%%%%%%%%%%%%
%%%%%%%% Preservation of Unitarity in the Density Operator %%%%%%%%%
%%%%%%%%%%%%%%%%%%%% Yolanda Villalba Celdran %%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc; clear; close all;
%% Initial parameters
Tmes = 1; % measurement time constant
gamma_m = 1/Tmes; % Spontaneous emission rate
h = 1; % Planck constant
w_eg = 2*pi/Tmes; % Natural frequency
eta = 1; % Efficiency of the detector
dt = 0.01; % step
rng('default'); % regenerate random seed
Xeq = [0;0;1]; % X objective in equilibrium

%% Pauli operators
i = sqrt(-1);
Sm = [0 0; 1 0]; Sp = Sm';
Sx = Sm+Sp; Sz = Sp*Sm-Sm*Sp; Sy = -i*Sz*Sx; Id = Sx*Sx;

%% Initialization
Tsim = 20*Tmes; dt = Tmes/20; t = [0:dt:Tsim];
X = zeros(3,length(t));

z0 = -0.8;
y0 = 0.1;
Y0 = [0.1;sqrt(1-y0^2-z0^2);z0];

%% Model
f = @(t,X) [-gamma_m/2,-w_eg/h,0;w_eg/h,-gamma_m/2,0;0,0,0]*X;
g = @(t,X) [0;-X(3)/h;X(2)/h];

sigma = @(t,X) [-sqrt(eta*gamma_m)*X(1)*X(3);
    - sqrt(eta*gamma_m)*X(2)*X(3);
    - sqrt(eta*gamma_m)*((X(3)^2)-1)];

H=diag([0,0,0]);
grad = @(t,X) -[0;0;1];
a = @(t,X) grad(t,X)'*f(t,X)+sigma(t,X)'*H*sigma(t,X)/2;
b = @(t,X) grad(t,X)'*g(t,X);

ro = @(t,X) (b(t,X)~=0 ||
a(t,X)>=0)*(a(t,X)+sqrt(a(t,X)^2+b(t,X)^2))/b(t,X);
u = @(t,X) -(X(3)<1)*ro(t,X);
%u = @(t,X) 0;
drift = @(t,X) f(t,X)+g(t,X)*u(t,X);

nPaths = 100;
for i = 1:nPaths
    Y = Y0;
    for k = 1:1:length(t)
        dW = randn*sqrt(dt);
        X(:,k,i) = Y;
```

```
        Y = X(:,k,i)+drift(k,X(:,k,i))*dt+sigma(k,X(:,k,i))*dW;
        % Normalization when we move out of the sphere
        if norm(Y) > 1
            Y = Y/norm(Y);
        end
        % Normalization to pure states
        % Y=Y/norm(Y);
    end

end

% Resizing vectors
px = reshape(X(1,:,:),[length(t),nPaths]);
py = reshape(X(2,:,:),[length(t),nPaths]);
pz = reshape(X(3,:,:),[length(t),nPaths]);

% Computing Lyapunov function
lyap = 1-X(3,:,:);
m_lyap = mean(lyap,3);

%% Displaying output results
% p_x, p_y and p_z stabilization
figure;
media=mean(X,3);
plot(t,media);
title(['Valores esperados de las Coordenadas de Bloch para
',num2str(nPaths),' simulaciones'], 'FontSize', 15);
xlabel('Tiempo (segundos)', 'FontSize', 11);
legend('Mean p_{x}','Mean p_{y}','Mean p_{z}');
grid;

% Stabilization & control signal
figure;
subplot(2,1,1);
media=mean(X,3);
plot(t,media);
title(['Valores esperados de las coordenadas de Bloch para
',num2str(nPaths),' simulaciones'], 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_x,p_y,p_z');
legend('Mean p_{x}','Mean p_{y}','Mean p_{z}');
grid;
subplot(2,1,2);
v=0*t;
for i=1:length(t)
    v(i)=u(i,media(:,i));
end
plot(t,v);
title('Acción de Control', 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('u(t)', 'FontSize', 11);

% Stabilization & Lyapunov function
figure;
subplot(2,1,1);
media=mean(X,3);
plot(t,media);
title(['Valores esperados de las coordenadas de Bloch para
',num2str(nPaths),' simulaciones'], 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
```

```
ylabel('p_x,p_y,p_z');
legend('Mean p_{x}','Mean p_{y}','Mean p_{z}');
grid;
subplot(2,1,2);
plot(t,m_lyap);
title(['Función de Lyapunov para ', num2str(nPaths),' simulaciones'],
'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media de la función de Lyapunov', 'FontSize', 11);

% Stabilization & control & Lyapunov function
figure;
subplot(3,1,1);
media=mean(X,3);
plot(t,media);
title(['Valores esperados de las coordenadas de Bloch para
',num2str(nPaths),' simulaciones'], 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_x,p_y,p_z');
legend('Mean p_{x}','Mean p_{y}','Mean p_{z}');
grid;
subplot(3,1,2);
v=0*t;
for i=1:length(t)
    v(i)=u(i,media(:,i));
end
plot(t,v);
title('Acción de Control', 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media u(t)', 'FontSize', 11);
subplot(3,1,3);
plot(t,m_lyap);
title(['Función de Lyapunov para ', num2str(nPaths),' simulaciones'],
'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media Lyapunov', 'FontSize', 11);

% p Bloch paths
figure;
subplot(3,1,1);
plot(t,px)
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_{x}', 'FontSize', 11)
subplot(3,1,2);
plot(t,py)
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_{y}', 'FontSize', 11)
subplot(3,1,3);
plot(t,pz)
xlim([0 20]);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_{z}', 'FontSize', 11)
sgtitle(['Coordenadas p_{x},p_{y},p_{z} frente al tiempo para ',
num2str(nPaths), ' simulaciones'],...
    'FontSize', 15, 'FontWeight','Bold');

% Lyapunov function
figure;
subplot(2,1,1);
plot(t,m_lyap);
```

```
title(['Función de Lyapunov para ', num2str(nPaths),' simulaciones'],
'FontSize', 15);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media de la función de Lyapunov', 'FontSize', 11);
grid on;
legend('Mean Lyapunov Function');
subplot(2,1,2);
plot(t,permute(lyap, [2 3 1]));
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('V(t)', 'FontSize', 11);
grid on;

% Control signal u(t,X)
v=0*t;
for i=1:length(t)
    v(i)=u(i,media(:,i));
end
u_values = zeros(401, 100);
for i=1:length(t)
    for j = 1:nPaths
    u_values(i,j)=u(i,X(:,i,j));
    end
end
figure;
subplot(2,1,1);
plot(t,v);
title(['Acción de control para ', num2str(nPaths), ' simulaciones'],
'FontSize', 15);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media de u(t)', 'FontSize', 11);
subplot(2,1,2);
plot(t,u_values)
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('u(t)', 'FontSize', 11);

% Bloch sphere
number_npath = randi([0, nPaths],1,4);

figure
subplot(2,2,1);
plotBlochSphere
hold on
plot3(X(1,:,number_npath(1)),X(2,:,number_npath(1)),X(3,:,number_npath
(1)), 'LineWidth', 2, 'Color', 'c')
title(['Simulación n = ',num2str(number_npath(1))])


subplot(2,2,2);
plotBlochSphere
hold on
plot3(X(1,:,number_npath(2)),X(2,:,number_npath(2)),X(3,:,number_npath
(2)), 'LineWidth', 2, 'Color', 'c')
title(['Simulación n = ',num2str(number_npath(2))])

subplot(2,2,3);
plotBlochSphere
hold on
plot3(X(1,:,number_npath(3)),X(2,:,number_npath(3)),X(3,:,number_npath
(3)), 'LineWidth', 2, 'Color', 'c')
title(['Simulación n = ',num2str(number_npath(3))])
```

```
subplot(2,2,4);
plotBlochSphere
hold on
plot3(X(1,:,number_npath(4)),X(2,:,number_npath(4)),X(3,:,number_npath
(4)), 'LineWidth', 2, 'Color', 'c')
title(['Simulación n = ',num2str(number_npath(4))])
sgtitle(['Esfera de Bloch para ', num2str(nPaths), ' simulaciones'],
'FontSize', 15, 'FontWeight','Bold');
```

Apéndice C

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% Simulation of the second controller %%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%% Financial Toolbox & SDETools %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% Yolanda Villalba Celdran %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc; clear; close all; clear all;
%% Initial parameters
Tmes = 1; % measurement time constant
gamma_m = 1/Tmes; % Spontaneous emission rate
h = 1; % Planck constant
w_eg = 2*pi/Tmes; % Natural frequency
eta = 1; % Efficiency of the detector
dt = 0.01; % step
T = 20*Tmes; % Terminal time
rng('default'); % regenerate random seed
Xeq = [0;0;1]; % X objective in equilibrium

%% Advanced control of a qubit
% f(x) & g(x)function
% px = X(1); py = X(2); pz = X(3);
f = @(t,X) [-gamma_m/2,-w_eg/h,0;w_eg/h,-gamma_m/2,0;0,0,0]*X;
g = @(t,X) [0;-X(3)/h;X(2)/h];

% Diffusion Term
sigma = @(t,X) [-sqrt(eta*gamma_m)*X(1)*X(3);
    -sqrt(eta*gamma_m)*X(2)*X(3);
   - sqrt(eta*gamma_m)*((X(3)^2)-1)];

% Drift Term
H=diag([0,0,0]);
grad=@(t,X) -[0;0;1];
C_1 = @(t,X) grad(t,X)'*g(t,X);
C_2 = @(t,X) grad(t,X)'*f(t,X)+sigma(t,X)'*H*sigma(t,X)/2;
phi = @(t,X) acos(sqrt(1-X(2)^2))/(acos(sqrt(1-X(2)^2))+2*pi);
v = @(t,X) (X(3)~= abs(1) && X(2)~=
abs(1))*(1+(phi(t,X)*C_2(t,X)/C_1(t,X)));
u = @(t,X) -v(t,X)*C_1(t,X); % Controlled system
%u = @(t,X) 0; % Uncontrolled system

drift = @(t,X) f(t,X)+g(t,X)*u(t,X);

%% Simulation of the system
% Building of the model
%x0=[0.1;sqrt(1-0.1^2-0.9^2);0.9];
%x0=[0.1;sqrt(1-0.1^2-0.85^2);0.85];
x0 = [0.1;sqrt(1-0.1^2-(-0.8)^2);-0.8];
nPaths = 100; % nTrials or nPaths

% Simulation methods
% From Financial Toolbox MATLAB: Euler - Maruyama Method
% obj = sde(drift,sigma,'StartTime',0,'StartState',x0);
% [X2,time,dW] = obj.simulate(T/dt,'DeltaTime',dt,'nTrials',
nPaths,'Antithetic', true);

% From SDETools: Milstein Method
```

```
time=0:dt:T;
opts = sdeset('SDEType','Ito','RandSeed',3);
for i = 1:nPaths
    X = sde_milstein(drift,sigma,time,x0,opts);

    opts = sdeset(opts, 'RandSeed', 3+i);
    X2(:,:,i) = X;
end

% Normalizing the vector
for i = 1:nPaths
    for j = 1:length(time)
        X2(j,:,i) = X2(j,:,i)/norm(X2(j,:,i));
    end
end

%% Resizing vectors
px = reshape(X2(:,1,:),[length(time),nPaths]);
py = reshape(X2(:,2,:),[length(time),nPaths]);
pz = reshape(X2(:,3,:),[length(time),nPaths]);

% Computing mean values for the stochastic state paths
m_norm_X = mean(X2,3);
m_px = mean(px,2);
m_py = mean(py,2);
m_pz = mean(pz,2);

% Computing Lyapunov function
lyap = 1-pz;
m_lyap = mean(lyap,2);

for i = 1:nPaths % Depth of X2
    for j = 1:length(time) % Rows of X2
        f_values = [-gamma_m/2,-w_eg/h,0;w_eg/h,-
gamma_m/2,0;0,0,0]*[px(j,i);py(j,i);pz(j,i)];
        g_values = [0;-pz(j,i)/h;py(j,i)/h];
        sigma_values = [-sqrt(eta*gamma_m)*px(j,i)*pz(j,i);
            -sqrt(eta*gamma_m)*py(j,i)*pz(j,i);
            - sqrt(eta*gamma_m)*((pz(j,i)^2)-1)];

        dist_C1 = acos(sqrt(1-(py(j,i))^2));
        dist_C2 = 2*pi;
        phi_values = dist_C1/(dist_C1+dist_C2);

        D_w0 = -eta*gamma_m*(1-pz(j,i)^2)^2;
        L_gV = -2/h*py(j,i)*pz(j,i);
        v_values =  (pz(j,i)~= abs(1) && py(j,i)~=
abs(1))*(1+(phi_values*D_w0/L_gV));
        u_values(j,i) = v_values*L_gV;
        %u_values(j,i) = 0;
    end
end

m_u = mean(u_values,2);

%% Displaying output results
% p_x, p_y and p_z stabilization
figure;
plot(time,m_norm_X);
```

```
xlim([0 20]);
title(['Valores esperados de las Coordenadas de Bloch para
',num2str(nPaths),' simulaciones'], 'FontSize', 15);
xlabel('Tiempo (segundos)', 'FontSize', 11);
legend('Mean p_{x}','Mean p_{y}','Mean p_{z}');
grid;

% Stabilization & control signal
figure;
subplot(2,1,1);
plot(time,m_norm_X);
xlim([0 20]);
title(['Valores esperados de las coordenadas de Bloch para
',num2str(nPaths),' simulaciones'], 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_x,p_y,p_z');
grid on;
legend('Mean p_{x}','Mean p_{y}','Mean p_{z}');
subplot(2,1,2);
v=0*time;
for i=1:length(time)
    v(i)=u(i,m_norm_X(i,:)');
end
plot(time,v);
xlim([0 20]);
title('Acción de Control', 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('u(t)', 'FontSize', 11);

% Stabilization & Lyapunov function
figure;
subplot(2,1,1);
plot(time,m_norm_X);
xlim([0 20]);
title(['Valores esperados de las coordenadas de Bloch para
',num2str(nPaths),' simulaciones'], 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_x,p_y,p_z');
grid on;
legend('Mean p_{x}','Mean p_{y}','Mean p_{z}');
subplot(2,1,2);
plot(time,m_lyap);
xlim([0 20]);
title(['Función de Lyapunov para ', num2str(nPaths),' simulaciones'],
'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media de la función de Lyapunov', 'FontSize', 11);


% Stabilization & control signal & Lyapunov
figure;
subplot(3,1,1);
plot(time,m_norm_X);
xlim([0 20]);
title(['Valores esperados de las coordenadas de Bloch para
',num2str(nPaths),' simulaciones'], 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_x,p_y,p_z');
grid on;
legend('Mean p_{x}','Mean p_{y}','Mean p_{z}');
subplot(3,1,2);
```

```
v=0*time;
for i=1:length(time)
    v(i)=u(i,m_norm_X(i,:)');
end
plot(time,v);
xlim([0 20]);
title('Acción de Control', 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media u(t)', 'FontSize', 11);
subplot(3,1,3);
plot(time,m_lyap);
xlim([0 20]);
title(['Función de Lyapunov para ', num2str(nPaths),' simulaciones'],
'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media Lyapunov', 'FontSize', 11);

% p Bloch paths
figure;
subplot(3,1,1);
plot(time,px)
xlim([0 20]);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_{x}', 'FontSize', 11)
subplot(3,1,2);
plot(time,py)
xlim([0 20]);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_{y}', 'FontSize', 11)
subplot(3,1,3);
plot(time,pz)
xlim([0 20]);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_{z}', 'FontSize', 11)
sgtitle(['Coordenadas p_{x},p_{y},p_{z} frente al tiempo para ',
num2str(nPaths), ' simulaciones'],...
    'FontSize', 15, 'FontWeight','Bold');


% Lyapunov function
figure;
subplot(2,1,1);
plot(time,m_lyap);
xlim([0 20]);
title(['Función de Lyapunov para ', num2str(nPaths),' simulaciones'],
'FontSize', 15);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media de la función de Lyapunov', 'FontSize', 11);
grid on;
legend('Mean Lyapunov Function');
subplot(2,1,2);
plot(time,lyap);
xlim([0 20]);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('V(t)', 'FontSize', 11);
grid on;

% Control signal u(t,X)
figure
subplot(2,1,1);
plot(time,m_u);
```

```
xlim([0 20]);
title(['Acción de control para ', num2str(nPaths), ' simulaciones'],
'FontSize', 15);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media de u(t)', 'FontSize', 11);
grid on;
legend('Mean u(t)');
subplot(2,1,2);
plot(time,u_values);
xlim([0 20]);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('u(t)', 'FontSize', 11);
grid on;

% Bloch sphere
number_npath = randi([0, nPaths],1,4);

figure
subplot(2,2,1);
plotBlochSphere
hold on
plot3(X2(:,1,number_npath(1)),X2(:,2,number_npath(1)),X2(:,3,number_np
ath(1)), 'LineWidth', 2, 'Color', 'c')
title(['Simulación n = ',num2str(number_npath(1))], 'FontSize', 11,
'FontWeight','Normal')

subplot(2,2,2);
plotBlochSphere
hold on
plot3(X2(:,1,number_npath(2)),X2(:,2,number_npath(2)),X2(:,3,number_np
ath(2)), 'LineWidth', 2, 'Color', 'c')
title(['Simulación n = ',num2str(number_npath(2))], 'FontSize', 11,
'FontWeight','Normal')

subplot(2,2,3);
plotBlochSphere
hold on
plot3(X2(:,1,number_npath(3)),X2(:,2,number_npath(3)),X2(:,3,number_np
ath(3)), 'LineWidth', 2, 'Color', 'c')
title(['Simulación n = ',num2str(number_npath(3))], 'FontSize', 11,
'FontWeight','Normal')

subplot(2,2,4);
plotBlochSphere
hold on
plot3(X2(:,1,number_npath(4)),X2(:,2,number_npath(4)),X2(:,3,number_np
ath(4)), 'LineWidth', 2, 'Color', 'c')
title(['Simulación n = ',num2str(number_npath(4))], 'FontSize', 11,
'FontWeight','Normal')
sgtitle(['Esfera de Bloch para ', num2str(nPaths), ' simulaciones'],
'FontSize', 15, 'FontWeight','Bold');
```

Apéndice D

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% Simulation of the second controller %%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% Self-implementation method %%%%%%%%%%%%%%%%%%%%
%%%%%%%% Preservation of Unitarity in the Density Operator %%%%%%%%%
%%%%%%%%%%%%%%%%%%%% Yolanda Villalba Celdran %%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc; clear; close all;
%% Initial parameters
Tmes = 1; % measurement time constant
gamma_m = 1/Tmes; % Spontaneous emission rate
h = 1; % Planck constant
w_eg = 2*pi/Tmes; % Natural frequency
eta = 1; % Efficiency of the detector
dt = 0.01; % step
rng('default'); % regenerate random seed
Xeq=[0;0;1]; % X objective in equilibrium

%% Pauli operators
i=sqrt(-1);
Sm=[0 0; 1 0];Sp=Sm';
Sx=Sm+Sp;Sz=Sp*Sm-Sm*Sp;Sy=-i*Sz*Sx;Id=Sx*Sx;

%% Inicialization
Tsim=30*Tmes; dt=Tmes/20; t=[0:dt:Tsim];
X=zeros(3,length(t));

z0=-0.8;
y0=0.1;
Y0=[0.1;sqrt(1-y0^2-z0^2);z0];

%% Model
f=@(t,X) [-gamma_m/2,-w_eg/h,0;w_eg/h,-gamma_m/2,0;0,0,0]*X;
g=@(t,X) [0;-X(3)/h;X(2)/h];

sigma = @(t,X) [-sqrt(eta*gamma_m)*X(1)*X(3);
    -sqrt(eta*gamma_m)*X(2)*X(3);
   - sqrt(eta*gamma_m)*((X(3)^2)-1)];


H=diag([0,0,0]);
grad=@(t,X) -[0;0;1];
C_1 = @(t,X) grad(t,X)'*g(t,X);
C_2 = @(t,X) grad(t,X)'*f(t,X)+sigma(t,X)'*H*sigma(t,X)/2;
phi = @(t,X) acos(sqrt(1-X(2)^2))/(acos(sqrt(1-X(2)^2))+2*pi);
v = @(t,X) (abs(X(2))~= 1)*(1+(phi(t,X)*C_2(t,X)/C_1(t,X)));
u = @(t,X) -v(t,X)*C_1(t,X);
% u =@(t,X) 0;


drift=@(t,X) f(t,X)+g(t,X)*u(t,X);
nPaths=100;
for i=1:nPaths
    Y=Y0;
    for k=1:1:length(t)
        dW=randn*sqrt(dt);
```

```
            X(:,k,i)=Y;
            Y=X(:,k,i)+drift(k,X(:,k,i))*dt+sigma(k,X(:,k,i))*dW;
            % Normalization when we move out of the sphere
            % if norm(Y) > 1
                % Y = Y/norm(Y);
            % end
            % Normalization to pure states
            % Y=Y/norm(Y);
           Y=Y/norm(Y);
        end

end

% Resizing vectors
px = reshape(X(1,:,:),[length(t),nPaths]);
py = reshape(X(2,:,:),[length(t),nPaths]);
pz = reshape(X(3,:,:),[length(t),nPaths]);

% Computing Lyapunov function
lyap = 1-X(3,:,:);
m_lyap = mean(lyap,3);

%% Displaying output results
% p_x, p_y and p_z stabilization
figure;
media=mean(X,3);
plot(t,media);
title(['Valores esperados de las Coordenadas de Bloch para
',num2str(nPaths),' simulaciones'], 'FontSize', 15);
xlabel('Tiempo (segundos)', 'FontSize', 11);
legend('Mean p_{x}','Mean p_{y}','Mean p_{z}');
grid;

% Stabilization & control signal
figure;
subplot(2,1,1);
media=mean(X,3);
plot(t,media);
title(['Valores esperados de las coordenadas de Bloch para
',num2str(nPaths),' simulaciones'], 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_x,p_y,p_z');
legend('Mean p_{x}','Mean p_{y}','Mean p_{z}');
grid;
subplot(2,1,2);
v=0*t;
for i=1:length(t)
    v(i)=u(i,media(:,i));
end
plot(t,v);
title('Acción de Control', 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('u(t)', 'FontSize', 11);

% Stabilization & Lyapunov function
figure;
subplot(2,1,1);
media=mean(X,3);
plot(t,media);
```

```
title(['Valores esperados de las coordenadas de Bloch para
',num2str(nPaths),' simulaciones'], 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_x,p_y,p_z');
legend('Mean p_{x}','Mean p_{y}','Mean p_{z}');
grid;
subplot(2,1,2);
plot(t,m_lyap);
title(['Función de Lyapunov para ', num2str(nPaths),' simulaciones'],
'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media de la función de Lyapunov', 'FontSize', 11);

% Stabilization & control & Lyapunov function
figure;
subplot(3,1,1);
media=mean(X,3);
plot(t,media);
title(['Valores esperados de las coordenadas de Bloch para
',num2str(nPaths),' simulaciones'], 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_x,p_y,p_z');
legend('Mean p_{x}','Mean p_{y}','Mean p_{z}');
grid;
subplot(3,1,2);
v=0*t;
for i=1:length(t)
    v(i)=u(i,media(:,i));
end
plot(t,v);
title('Acción de Control', 'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media u(t)', 'FontSize', 11);
subplot(3,1,3);
plot(t,m_lyap);
title(['Función de Lyapunov para ', num2str(nPaths),' simulaciones'],
'FontSize', 11);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media Lyapunov', 'FontSize', 11);

% p Bloch paths
figure;
subplot(3,1,1);
plot(t,px)
xlim([0 20]);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_{x}', 'FontSize', 11)
subplot(3,1,2);
plot(t,py)
xlim([0 20]);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_{y}', 'FontSize', 11)
subplot(3,1,3);
plot(t,pz)
xlim([0 20]);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('p_{z}', 'FontSize', 11)
sgtitle(['Coordenadas p_{x},p_{y},p_{z} frente al tiempo para ',
num2str(nPaths), ' simulaciones'],...
    'FontSize', 15, 'FontWeight','Bold');
```

```
% Lyapunov function
figure;
subplot(2,1,1);
plot(t,m_lyap);
title(['Función de Lyapunov para ', num2str(nPaths),' simulaciones'],
'FontSize', 15);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media de la función de Lyapunov', 'FontSize', 11);
grid on;
legend('Mean Lyapunov Function');
subplot(2,1,2);
plot(t,permute(lyap, [2 3 1]));
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('V(t)', 'FontSize', 11);
grid on;

% Control signal u(t,X)
v=0*t;
for i=1:length(t)
    v(i)=u(i,media(:,i));
end
u_values = zeros(401, 100);
for i=1:length(t)
    for j = 1:nPaths
    u_values(i,j)=u(i,X(:,i,j));
    end
end
figure;
subplot(2,1,1);
plot(t,v);
title(['Acción de control para ', num2str(nPaths), ' simulaciones'],
'FontSize', 15);
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('Media de u(t)', 'FontSize', 11);
subplot(2,1,2);
plot(t,u_values)
xlabel('Tiempo (segundos)', 'FontSize', 11);
ylabel('u(t)', 'FontSize', 11);

% Bloch sphere
number_npath = randi([0, nPaths],1,4);

figure
subplot(2,2,1);
plotBlochSphere
hold on
plot3(X(1,:,number_npath(1)),X(2,:,number_npath(1)),X(3,:,number_npath
(1)), 'LineWidth', 2, 'Color', 'c')
title(['Simulación n = ',num2str(number_npath(1))])


subplot(2,2,2);
plotBlochSphere
hold on
plot3(X(1,:,number_npath(2)),X(2,:,number_npath(2)),X(3,:,number_npath
(2)), 'LineWidth', 2, 'Color', 'c')
title(['Simulación n = ',num2str(number_npath(2))])

subplot(2,2,3);
plotBlochSphere
```

```
hold on
plot3(X(1,:,number_npath(3)),X(2,:,number_npath(3)),X(3,:,number_npath
(3)), 'LineWidth', 2, 'Color', 'c')
title(['Simulación n = ',num2str(number_npath(3))])

subplot(2,2,4);
plotBlochSphere
hold on
plot3(X(1,:,number_npath(4)),X(2,:,number_npath(4)),X(3,:,number_npath
(4)), 'LineWidth', 2, 'Color', 'c')
title(['Simulación n = ',num2str(number_npath(4))])
sgtitle(['Esfera de Bloch para ', num2str(nPaths), ' simulaciones'],
'FontSize', 15, 'FontWeight','Bold');
```

Apéndice E

```
%plotBlochSphere.m
[Xs, Yx, Zx] = sphere(25);
mySphere = surf(Xs, Yx, Zx);
axis equal
shading interp
mySphere.FaceAlpha = 0.25

line([-1 1], [0 0], [0 0], 'LineWidth', 1, 'Color', [0 0 0])
line([0 0], [-1 1], [0 0], 'LineWidth', 1, 'Color', [0 0 0])
line([0 0], [0 0], [-1 1], 'LineWidth', 1, 'Color', [0 0 0])

text(0, 0, 1.1, '$\left| 0 \right>$', 'Interpreter', 'latex',...
 'FontSize', 20, 'HorizontalAlignment', 'Center')
text(1.1, 0, 0, '$\left| + \right>$', 'Interpreter', 'latex',...
 'FontSize', 20, 'HorizontalAlignment', 'Center')
text(-1.1, 0, 0, '$\left| - \right>$', 'Interpreter', 'latex',...
 'FontSize', 20, 'HorizontalAlignment', 'Center')
text(0, 0, -1.1, '$\left| 1 \right>$', 'Interpreter', 'latex',...
 'FontSize', 20, 'HorizontalAlignment', 'Center')
view([60 15])
bloch_des = line( [0 0], [0 0], [0 0], ...
 'LineWidth', 2, 'Marker', 'o', 'Color', 'b');

bloch_est = line( [0 0], [0 0], [0 0], ...
 'LineWidth', 2, 'Marker', 'o', 'Color', 'r');
```

Bibliografía

[1] "Quantum computing just might save the planet | McKinsey." https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/quantum-computing-just-might-save-the-planet.

[2] J. García López De Lacalle, A. García, L. Jesús, G. López, L. Francisco, and G. Mazarío, "Computación Cuántica".

[3] Sebastián García Cutillas, "CONTROL REALIMENTADO ESTOCÁSTICO DE QUBITS CUÁNTICOS," Cartagena, 2021.

[4] "Qubits a un kelvin | Actualidad | Investigación y Ciencia." https://www.investigacionyciencia.es/noticias/lgica-cuntica-caliente-la-computacin-cuntica-encara-el-obstculo-de-la-temperatura-18532.

[5] "The European Quantum Technologies Roadmap," 2017. [Online]. Available: http://qurope.eu/h2020/qtflagship/roadmap2016

[6] E. Gibney, "Quantum gold rush: the private funding pouring into quantum start-ups," *Nature*, vol. 574, no. 7776, pp. 22–24, Oct. 2019, doi: 10.1038/D41586-019-02935-4.

[7] D. Dong and I. R. Petersen, "Quantum control theory and applications: A survey," *IET Control Theory and Applications*, vol. 4, no. 12, pp. 2651–2671, Dec. 2010, doi: 10.1049/IET-CTA.2009.0508.

[8] J. Juan, G. Nobajas, Ángel, and R. Díaz-Cordovés, "FUNDAMENTOS DE CONTROL AUTOMÁTICO DE SISTEMAS CONTINUOS Y MUESTREADOS" 2010.

[9] G. H. P, D. G, and S. K. F. Herzog, "Stochastic Systems," 2011. https://ethz.ch/content/dam/ethz/special-interest/mavt/dynamic-systems-n-control/idsc-dam/Lectures/Stochastic-Systems/Script_Stochastic_Sytems.pdf.

[10] N. Michael A and C. Isaac L, "Quantum Computation and Quantum Information," 2010. http://mmrc.amss.cas.cn/tlb/201702/W020170224608149940643.pdf.

[11] D. Manzano, "A short introduction to the Lindblad master equation," *AIP Adv*, vol. 10, no. 2, Feb. 2020, doi: 10.1063/1.5115323.

[12] J. Sharifi and H. Momeni, "Quantum Stochastic Stability".

[13] P. Florchinger and E. I. Verriest, “Application of stochastic Artstein’s theorem to feedback stabilization,” *Stoch Anal Appl*, vol. 18, no. 3, pp. 361–373, 2000, doi: 10.1080/07362990008809675.

[14] F. Abedi, M. Abu Hassan, and M. Suleiman, “Feedback stabilization and adaptive stabilization of stochastic nonlinear systems by the control Lyapunov function,” *Stochastics*, vol. 83, no. 2, pp. 179–201, 2011, doi: 10.1080/17442508.2011.552723.

[15] J. L. Gómez Muñoz and F. Delgado, “QUANTUM: A Wolfram Mathematica add-on for Dirac Bra-Ket Notation, Non-Commutative Algebra, and Simulation of Quantum Computing Circuits”, doi: 10.1088/1742-6596/698/1/012019.

[16] “Stochastic Differential Equation (SDE) model - MATLAB - MathWorks España.” https://es.mathworks.com/help/finance/sde.html.

[17] “GitHub - horchler/SDETools: Matlab Toolbox for the Numerical Solution of Stochastic Differential Equations.” https://github.com/horchler/SDETools.