industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

Universidad Politécnica de Cartagena

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

CONTROL CUÁNTICO ÓPTIMO REALIMENTADO DE UN QUBIT DE ESTADO SÓLIDO

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

Autor: **Sebastián García Cutillas**
Director: Javier Molina Vilaplana
Codirector: Juan Ignacio Mulero Martínez

Cartagena, 1 de octubre de 2019

Índice

Objetivos y resumen

El control es una rama interdisciplinar entre la ingeniería y las matemáticas, cuyo objetivo es la manipulación activa del comportamiento de cualquier sistema dinámico. Actualmente, con el desarrollo de nuevas tecnologías basadas en dispositivos cuyas leyes vienen dictadas por la mecánica cuántica, el desarrollo de técnicas de control aplicadas a estos sistemas se ha convertido en una prioridad para el desarrollo tecnológico actual.

El principal problema radica en que algunas cualidades de la mecánica cuántica no permiten pasar fácilmente el esquema del control clásico al ámbito cuántico. Clásicamente, es posible adquirir toda la información necesaria y emplearla de manera eficiente para retroalimentar el sistema y especializar la acción de control al estado actual. Por el contrario, en un sistema cuántico no toda la información es asequible mediante una medición y, además, todo intento de medir el sistema lo perturba de manera notable. Técnicamente, las ecuaciones dinámicas en un proceso tal son ecuaciones diferenciales estocásticas (SDE[1]) no lineales en lugar de las ecuaciones diferenciales deterministas de los sistemas clásicos.

El dominio de las tecnologías cuánticas se divide en cuatro campos destacables [1]: *Sensores cuánticos*, donde la alta sensibilidad de los sistemas cuánticos a las perturbaciones se aprovecha para mejorar la precisión de las medidas de magnitudes físicas; *Comunicación cuántica*, donde los fotones se usan para transmitir datos e información de una manera segura; *Simulación cuántica*, donde un sistema cuántico controlado puede simular el comportamiento de otro sistema cuántico menos accesible; y *Computación cuántica*, donde mediante los fenómenos cuánticos se puede conseguir una disminución drástica del tiempo de computación en cálculos numéricos.

Sabiendo los campos de actuación de estas tecnologías y mirando al futuro, del mismo modo que el transistor supone actualmente un elemento imprescindible en la industria, parece difícil no creer que las tecnologías cuánticas supondrán una revolución en sectores como la industria aeroespacial, automotriz o manufacturera, así como en salud, agricultura o finanzas.

El desarrollo de las tecnologías cuánticas ha conseguido captar mucha atención en las últimas décadas. Como aparece explicado en [1], parte de este impulso se debe al "Quantum Manifesto", un documento generado en 2016 por la comunidad científica donde se realiza un llamamiento a gobiernos e industrias para apoyar esta tecnología. El interés generado por este documento fue respondido por la Comisión Europa mediante un proyecto de 10 años de duración (*Flagship*) y con una inversión de más de un billón de euros iniciales, aumentando en volumen cada año que pasa. Por otro lado, potencias mundiales como China, Rusia, Australia, Estados Unidos o Canadá son actualmente líderes en el desarrollo de tecnologías cuánticas. A nivel empresarial, compañías como Intel, HP, Toshiba y Microsoft, con IMB y Google a la cabeza, apuestan por la investigación en los campos de las tecnologías cuánticas como la computación y la comunicación cuánticas.

Con este trabajo se pretende desarrollar un proyecto de investigación enfocado a la simulación numérica de la dinámica y control de un Qubit (bit cuántico): la nueva unidad de información elemental de la teoría de la información cuántica. A diferencia de un bit clásico que solo presenta dos estados posibles, un bit cuántico puede presentar ambos estados al mismo tiempo, es decir, el estado de un Qubit es una combinación lineal de los estados elementales de un bit clásico.

[1] Stochastic Differential Equation.

Esta característica es la que define el potencial de los Qubits como nueva tecnología. Por este mismo motivo, el Qubit, como su propio nombre indica, es un sistema cuántico, estando sometido y regido por las leyes de estos.

En el primer capítulo del trabajo, se realizará una breve introducción a las tecnologías cuánticas, ilustrando el formalismo cuántico a través de la computación cuántica. En esta sección, se estudiará el Qubit, sus estados y las operaciones a las que son sometidos. En el segundo capítulo, se introducirá el modelo de control empleado para sistemas cuánticos en general, concretando para la dinámica y control de un Qubit. En el siguiente capítulo, analizaremos los resultados obtenidos a través de las diferentes simulaciones llevadas a cabo mediante métodos numéricos. En este punto, comprobaremos qué tan bueno es el control propuesto en la bibliografía para un Qubit y si es robusto ante diferentes situaciones. Se cerrará el proyecto con un capítulo dedicado a las conclusiones extraídas y los posibles trabajos de interés que se pueden realizar en un futuro.

Capítulo 1

Qubit, la unidad elemental de información cuántica

Los sistemas cuánticos se caracterizan por su aleatoriedad intrínseca. La ciencia que estudia el control y la manipulación de estos sistemas se llama las ciencias de la información cuántica. Dentro de esta ciencia, las principales tecnologías cuánticas son cuatro [1]:

Los sensores son una de las primeras aplicaciones de las tecnologías cuánticas, puesto que su implementación es relativamente sencilla. Los sensores cuánticos son muy sensibles a variables físicas (gravedad, campo magnético…) y detectan el cambio con una muy alta precisión. Esto es debido a que los sistemas cuánticos en superposición (paralelismo cuántico) producen una gran respuesta al cambio. Ya se han podido identificar diferentes aplicaciones de estos, como son la metrología, el escaneo o la navegación.

Las comunicaciones y su seguridad también sufrirán una revolución cuántica. Los ordenadores cuánticos serán capaces de romper las actuales claves criptográficas clásicas; no habrá manera de mantener la información segura con los métodos actuales de ciberseguridad. Una nueva disciplina, criptografía cuántica, aparece para mejorar la transmisión de información de una forma segura. Como ya se ha comentado, en el momento que se hace una medida sobre un sistema cuántico, se perturba su estado inicial. Por tanto, un receptor sería capaz de ver si un Qubit de información se ha alterado respecto del valor esperado; así, se detectaría la intrusión y se podrían tomar medidas para proteger el mensaje. Las principales aplicaciones se esperan en la transmisión de información a través de canales cuánticos, así como la idea de un futuro internet cuántico.

La simulación cuántica es una de las tecnologías más atractivas, ya que permite simular comportamientos de sistemas complejos, desde moléculas hasta galaxias, de forma natural. A diferencia de los superordenadores, que intentan simular comportamientos de elementos naturales para, por ejemplo, crear nuevos materiales o moléculas complementarias, con los simuladores cuánticos esta representación del comportamiento se realizará de manera natural: las unidades mínimas que constituyen el simulador son las mismas a la que se intenta simular. De esta forma, se podrá simular eficientemente el comportamiento de una molécula porque los elementos que componen el simulador son las mismas partículas elementales de la molécula. Las aplicaciones que esto ofrece ayudarán en la creación de nuevos materiales, nuevos medicamentos, predicción del comportamiento de la naturaleza, etc.

Aunque todas las tecnologías cuánticas mencionadas son de inmensa utilidad, la que se espera verdaderamente importante y revolucionaria es la computación cuántica. Basada en los Qubits antes mencionados, nos abren una capacidad de procesamiento que crece exponencialmente con el número de Qubits que somos capaces de controlar entrelazados entre sí, llevándonos a capacidades que exceden nuestra percepción. Por ejemplo, un ordenador cuántico de cien Qubits tendría una capacidad de procesamiento equivalente al número de átomos que compone nuestro planeta. La posibilidad de computaciones con esta capacidad de cálculo abre un nuevo universo de posibilidades. Hay ya varios tipos de computadores cuánticos, dependiendo de las tecnologías que se utilizan para su construcción y que podríamos clasificar en adiabáticos (ya operativos para determinados usos), orientados a puertas (que es una de las apuestas más fuertes por parte de fabricantes comerciales y donde se tienen puestas muchas expectativas a medio plazo) y topológicos (con un nivel de madurez menor y con revolucionarias posibilidades a medio y largo plazo). Hoy en día no se ha podido ver aplicado el verdadero potencial de la computación cuántica, pues está en sus albores, aunque ya muy cerca de los conceptos de supremacía cuántica (sobre la computación clásica). No hay duda de que estas tecnologías marcarán una auténtica revolución.

A la hora de ilustrar el formalismo de la mecánica cuántica, vamos a considerar el caso de la computación cuántica debido a que es la tecnología cuántica que más importancia está adquiriendo en la actualidad. Por tanto, nos centraremos en este campo, a través de las palabras de Mermin [2], para sintetizar algunas de las ideas de las tecnologías cuánticas y usándolo como contexto para el estudio del Qubit.

Un ordenador cuántico es aquel que opera mediante transformaciones muy especiales de su estado interno, llevadas a cabo gracias a las leyes de la mecánica cuántica y a través de unas condiciones extremadamente controladas a las que serán sometidas todas las interacciones físicas que se produzcan. En este tipo de tecnología cualquier mínima interacción puede provocar el fallo del sistema. A diferencia de lo que ocurre en un ordenador clásico, las interacciones con el medio ambiente, como pueden ser moléculas presentes en el aire que colisionan con el sistema físico o la presencia de mínimos intercambios energéticos radiantes, deben ser evitadas, o en su caso, controladas para asegurar el correcto funcionamiento. Saber qué interacciones son relevantes y cuáles no puede dar lugar a la decoherencia, indeseada en cualquier sistema cuántico pues se pierden las propiedades características del mismo que lo diferencian de un sistema clásico.

No obstante, a pesar de esta característica tan restrictiva, existen dos motivos principales por los que sigue siendo interesante el estudio de estos ordenadores. En primer lugar, porque la separación entre los niveles de energía discretos en los sistemas de escala atómica es mayor que en el caso de los sistemas macroscópicos y, por tanto, es más sencillo aislar dinámicamente el sistema. Por otro lado, si los errores que aparecen en el sistema ocurren con un ratio bajo, es posible controlarlos fácilmente, incluso desconociendo el origen de estos fallos.

La computación cuántica, además de su aplicación práctica, sirve como una fuente de aprendizaje de matemáticas abstractas, ingeniería de control y, sobre todo, mecánica cuántica. Respecto a este último elemento, los Qubits, elementos básicos de los ordenadores cuánticos, resultan ser un caso ilustrativo y relativamente sencillo para introducirse en el estudio de la mecánica cuántica. Primero, porque representan el sistema físico más sencillo; es discreto, es decir, compuesto por un número finito de estados, en este caso dos (*two-state system*). Por otro lado, aplicar el formalismo abstracto al fenómeno real, una de las partes más complicadas de la mecánica cuántica, no es un problema en la teoría de la computación cuántica si únicamente se busca saber qué es capaz de hacer un ordenador cuántico. Al igual que ocurre en los ordenadores clásicos, no es necesario saber qué dispositivos electrónicos lo componen o cómo funcionan para ser un experto practicante en informática.

Por este mismo motivo, se hará a continuación un breve resumen de los estados y operaciones realizadas sobre bits clásicos, que comparten conceptos con los bits cuánticos, pero introduciendo, en un entorno más conocido y sencillo, la nomenclatura cuántica. Tras ello, se explicará esto mismo aplicado a los bits cuánticos, indicando las particularidades de estos.

1.1. Cbits y sus estados

Un ordenador clásico está construido por circuitos eléctricos que alcanzan dos estados estables posibles, que habitualmente se llaman estado cero (0) y estado uno (1). A este estado se le conoce como bit, es la unidad básica de información y puede ser, por tanto, 0 o 1; dos estados que pueden expresar sí o no, verdadero o falso. Estos ordenadores trabajan con cadenas de ceros y unos, a las que se le aplican operaciones que las convierten en cadenas diferentes.

En la teoría de la computación cuántica es frecuente emplear también el término "bit" para referirse al sistema clásico de dos estados que representa el valor abstracto del bit (0 o 1). Llamar al sistema físico igual que al valor abstracto puede llevar a confusión. Por este motivo se suele emplear el término Cbit ("C" de "*classical"*) para describir el sistema clásico de dos estados y Qbit ("Q" de "*quantum"*) para describir su generalización cuántica. Esta terminología fue introducida por Paul Dirac, que empleó *c-number* y *q-number* para describir las cantidades clásicas y sus generalizaciones cuánticas, respectivamente. No obstante, debido a la pronunciación de Qbit, este suele ser escrito como *Qubit.* En esta primera parte de introducción, debido a que emplearemos la notación de Dirac y con el fin de uniformizar términos, utilizaremos el término Qbit.

En esta notación, también conocida como notación *bra-ket*, se emplea el símbolo $|\ \ \rangle$ (llamado *ket*), donde se sitúa 0 o 1 según el estado del sistema; es decir, los estados estables del Cbit se representan con $|0\rangle$ y $|1\rangle$. Es común nombrar Cbit tanto a esta representación del estado como a la condición física del mismo. No ocurre lo mismo con el Qbit, donde el estado se refiere únicamente al símbolo, pues como se verá más adelante, este no está asociado a ninguna propiedad interna del mismo.

Los posibles estados en los que pueden estar dos Cbits son

$$|0\rangle|0\rangle, |0\rangle|1\rangle, |1\rangle|0\rangle, \text{o } |1\rangle|1\rangle, \tag{1.1}$$

y pueden expresarse de un modo más compacto en un único ket, facilitando su lectura, sobre todo cuando se dispone de un número mayor de bits:

$$|00\rangle, |01\rangle, |10\rangle, \text{o } |11\rangle. \tag{1.2}$$

También existe una tercera forma, $|x\rangle_n$, donde x es el número decimal asociado al número binario, y n es el número de Cbits, indicado para evitar ambigüedades. De este modo, los estados anteriores pueden expresarse también como

$$|0\rangle_2, |1\rangle_2, |2\rangle_2, \text{o } |3\rangle_2. \tag{1.3}$$

No obstante, cuando no exista posibilidad de ambigüedad debido al número de Cbits, el subíndice puede emplearse para otros propósitos.

Se utilizarán (1.1), (1.2) o (1.3) indistintamente según el contexto, aunque principalmente emplearemos las dos primeras formas.

Respecto a esta notación, Dirac introdujo $|\ \ \rangle$ con el fin de trabajar con vectores. En su interior, se puede escribir cualquier elemento que sirva para especificar qué representa el vector. Por tanto, si anteriormente hemos indicado el estado en estas especie de cajas, estamos confirmando que estos estados pueden considerarse vectores.

Los dos estados $|0\rangle$ y $|1\rangle$ de un Cbit representan dos vectores ortogonales unitarios en un espacio bidimensional. Esta interpretación es especialmente útil, por no decir necesaria, en el tratamiento con Qbits. Por tanto, estos dos estados se pueden representar como vectores columnas del siguiente modo:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{1.4}$$

motivo por el que, de forma usual, se suele emplear la palabra Cbit indistintamente para referirnos, por un lado, al estado físico del Cbit y, por otro, los vectores que representan estos estados del Cbit.

Volviendo al caso de 2 Cbits, el espacio vectorial es 4-dimensional, con base ortonormal

$$|00\rangle, |01\rangle, |10\rangle, |11\rangle, \tag{1.5}$$

que también se puede expresar como

$$|0\rangle|0\rangle, |0\rangle|1\rangle, |1\rangle|0\rangle, |1\rangle|1\rangle; \tag{1.6}$$

de hecho, no es más que una forma simplificada de expresar productos tensoriales, que, siendo estrictamente correctos matemáticamente, se expresa como

$$|0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle. \tag{1.7}$$

Por tanto, entendiendo los dos posibles estados de un Cbit como componentes ortogonales unitarios, los estados definidos por un número mayor de Cbits se pueden calcular fácilmente mediante los productos tensoriales, obteniendo como resultado otro vector columna. Un ejemplo claro podría ser el siguiente:

$$|4\rangle_3 = |100\rangle = |1\rangle|0\rangle|0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \tag{1.8}$$

donde si contamos en vertical desde la parte superior el vector columna de 8 componentes de 0 hasta 7, comprobamos que el único elemento no nulo, es decir, el 1, ocupa la posición número cuatro, correspondiente con la primera notación empleada en esta última ecuación, siguiendo la generalización empleada en un principio en (1.4).

Por tanto, la estructura del producto tensorial de los estados de varios Cbit permite que el vector columna de 2^n dimensiones represente un estado particular de los 2^n posibles estados de n Cbits, de tal forma que todas sus entradas sean cero, excepto en la posición hacia abajo desde la parte superior especificada por el número que esos n Cbits están representando en binario, en cuyo lugar habrá un 1.

No obstante, a pesar de que esta representación sirve para ver con mayor claridad cómo se pueden interpretar los estados como vectores para múltiples Cbits, por lo general es más sencillo y práctico trabajar directamente con los vectores estado de las formas (1.1)-(1.3).

1.2. Operaciones reversibles en Cbits

Las operaciones reversibles son aquellas que transforman el Bit desde un estado inicial a uno final empleando únicamente acciones que pueden ser invertibles. Estas operaciones son fundamentales en la computación cuántica. La única operación irreversible útil en un Qbit es el proceso de medida. Solo con ella se puede obtener información del Qbit antes de que este adquiera su forma final.

Por otra parte, en los ordenadores clásicos, la medida no es considerada un proceso computacional debido a su sencillez, exceptuando casos específicos de diseño o investigación.

No obstante, todas las operaciones relevantes en un computador clásico se corresponden con operaciones reversibles en un ordenador cuántico.

En una operación reversible cada estado final proviene de un estado inicial único. Una operación irreversible es *ERASE* (*eliminar* en castellano), que convierte el Cbit al estado $|0\rangle$ independientemente del valor del estado inicial. Por tanto, no hay forma de, una vez aplicado, volver al estado inicial.

La única operación reversible no trivial que podemos aplicar sobre un solo Cbit es la operación NOT o *flip* (negación), representada con el símbolo **X**, que intercambia el estado inicial:

$$\mathbf{X}\colon |x\rangle \to |\bar{x}\rangle; \qquad \bar{1} = 0, \quad \bar{0} = 1. \tag{1.9}$$

Es reversible porque tiene inverso: si se aplica NOT una segunda vez al Cbit, vuelve al estado inicial. Si representamos los dos estados ortogonales del Cbit mediante vectores columna (1.4), podemos expresar NOT como un operador lineal en un espacio vectorial bidimensional, cuya acción sobre los vectores viene dada por la matriz

$$\mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{1.10}$$

Por tanto, las dos operaciones que se pueden realizar sobre un único Cbit son cambiar su estado o dejarlo como está, correspondiente con los dos operadores lineales **X** (NOT) y $\mathbf{1}$ (identidad),

$$\mathbf{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{1.11}$$

Es útil introducir un operador número **n** para un único Cbit, definido como

$$\mathbf{n}|x\rangle = x|x\rangle, \ x = 0 \text{ o } 1; \tag{1.12}$$

$|0\rangle$ y $|1\rangle$ son autovectores de **n** con autovalores 0 y 1. Multiplicar el estado de un Cbit por 1 ya hemos visto que significa no modificarlo, no obstante, multiplicarlo por 0 puede generar dudas. Esta operación, solamente se realiza junto con otras que completen el significado físico.

Explicado **n**, es conveniente definir su complementario

$$\bar{\mathbf{n}} = \mathbf{1} - \mathbf{n} \tag{1.13}$$

de tal forma que $|0\rangle$ y $|1\rangle$ son autovectores de $\bar{\mathbf{n}}$ con autovalores 1 y 0. Las representaciones matriciales de estos operadores son

$$\mathbf{n} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \bar{\mathbf{n}} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}. \tag{1.14}$$

De las definiciones y sus representaciones matriciales podemos extraer algunas propiedades

$$\mathbf{n^2} = \mathbf{n}, \bar{\mathbf{n}}^{\mathbf{2}} = \bar{\mathbf{n}}, \mathbf{n}\bar{\mathbf{n}} = \bar{\mathbf{n}}\mathbf{n} = \mathbf{0}, \mathbf{n} + \bar{\mathbf{n}} = \mathbf{1}. \tag{1.15}$$

También, teniendo en cuenta el operador NOT:

$$\mathbf{nX} = \mathbf{X}\bar{\mathbf{n}}, \ \ \bar{\mathbf{n}}\mathbf{X} = \mathbf{Xn}, \tag{1.16}$$

debido a que negar el Cbit y después actuar en él con **n** ($\bar{\mathbf{n}}$) es equivalente a actuar en él con $\bar{\mathbf{n}}$ (**n**) y después negarlo. La última identidad de interés que mencionaremos es que el operador NOT es su propio inverso:

$$\mathbf{X}^2 = \mathbf{1}. \tag{1.17}$$

Hemos comentado los operadores útiles para un solo Cbit, pero las posibilidades aumentan cuando trabajamos con un par de Cbits. Comenzaremos presentando la operación *swap* (o *exchange*), es decir, *intercambio*, que nombraremos con la letra **S**. Este operador simplemente intercambia el estado de dos Cbits y puede ser escrito como:

$$\mathbf{S} = \mathbf{n} \otimes \mathbf{n} + \bar{\mathbf{n}} \otimes \bar{\mathbf{n}} + (\mathbf{X} \otimes \mathbf{X})(\mathbf{n} \otimes \bar{\mathbf{n}}) + (\mathbf{X} \otimes \mathbf{X})(\bar{\mathbf{n}} \otimes \mathbf{n}), \tag{1.18}$$

donde el producto tensorial ⊗ de dos operadores 1-bit es el operador 2-bit que actúa en el Cbit izquierdo con el operador a la izquierda de ⊗ y en el Cbit derecho con el operador a la derecha de ⊗, debido a las propiedades de este producto[2], del modo

$$(\mathbf{a} \otimes \mathbf{b})\,|x\rangle \otimes |y\rangle = \mathbf{a}|x\rangle \otimes \mathbf{b}|y\rangle. \tag{1.19}$$

Una vez dicho esto, podemos explicar detenidamente el operador swap de (1.18): si el estado de ambos Cbits es $|1\rangle$ (es decir, intercambiar los estados significa dejarlos igual), el único término que actúa sobre ellos es el primero, obteniendo 0 al aplicar el resto; si ambos Cbits son $|0\rangle$, el único término que actúa es el segundo; si el Cbit de la izquierda presenta el estado $|1\rangle$ y el de la derecha está en $|0\rangle$, solo el tercer término actúa y el efecto de negar ambos Cbits es intercambiarlos, y si el Cbit de la izquierda es $|0\rangle$ y el de la derecha es $|1\rangle$, el último término es el que actúa y nuevamente los dos operadores NOT permiten cambiar sus estados.

Debido a la notación exigida por el producto tensorial, para un gran número de Cbits puede resultar incómodo y poco práctico trabajar con operadores de 2 bits que afectan a solo un par de los múltiples Cbits. Por este motivo, la forma de simplificar esta notación (el término a la izquierda de la siguiente ecuación) es mediante subíndices que indican sobre qué Cbit actúa el operador de 1-bit, mientras que el resto permanecen inalterados (son multiplicados por la matriz identidad)

$$\mathbf{1} \otimes \mathbf{1} \otimes \mathbf{a} \otimes \mathbf{1} \otimes \mathbf{b} \otimes \mathbf{1} = \mathbf{a}_3\mathbf{b}_1 = \mathbf{b}_1\mathbf{a}_3. \tag{1.20}$$

El subíndice indica la posición desde la derecha comenzando por cero. Además, con esta notación es invariable el orden en que se escriba **a** y **b** mientras vayan acompañados del subíndice.

Siguiendo esta convención, podemos expresar el operador swap **S** (1.18) que actúa sobre los Cbits i y j como

$$\mathbf{S}_{ij} = \mathbf{n}_i\mathbf{n}_j + \bar{\mathbf{n}}_i\bar{\mathbf{n}}_j + (\mathbf{X}_i\mathbf{X}_j)(\mathbf{n}_i\bar{\mathbf{n}}_j + \bar{\mathbf{n}}_i\mathbf{n}_j). \tag{1.21}$$

Bajo esta nomenclatura, finalizaremos la explicación de este operador, al igual que se hizo con el operador NOT, **X**, indicando que aplicarlo dos veces es equivalente a no aplicarlo ninguna vez:

$$\mathbf{S}_{ij}^2 = \mathbf{1}. \tag{1.22}$$

El segundo operador que explicamos, empleando la notación de subíndices ya introducida, es el operador *controlled-not* o cNOT (negación controlada), $\mathbf{C}_{ij}$, de gran importancia en la computación cuántica. El Cbit de la posición i es el Cbit de control y el de la posición j es el Cbit objetivo. Dependiendo del Cbit de control, el Cbit objetivo permanecerá invariable o cambiará de estado. Así, si el Cbit de control es $|0\rangle$, no se produce el cambio, mientras que si es $|1\rangle$, el

[2] Se puede comprobar que $(\mathbf{a} \otimes \mathbf{b})(\mathbf{c} \otimes \mathbf{d}) = (\mathbf{ac}) \otimes (\mathbf{bd})$.

operador $\mathbf{C}_{ij}$ cambiará el estado del bit objetivo (aplicará el operador NOT sobre ese Cbit). Esto puede expresarse como

$$\mathbf{C}_{10}|x\rangle|y\rangle = |x\rangle|y \oplus x\rangle, \qquad \mathbf{C}_{01}|x\rangle|y\rangle = |x \oplus y\rangle|y\rangle, \tag{1.23}$$

donde $\oplus$ denota adición en módulo 2, es decir, la suma binaria o puerta XOR (OR exclusiva):

$$y \oplus 0 = y, \quad y \oplus 1 = \bar{y} = 1 - y. \tag{1.24}$$

Se puede construir una puerta swap a partir de tres operadores cNOT:

$$\mathbf{S}_{ij} = \mathbf{C}_{ij}\mathbf{C}_{ji}\mathbf{C}_{ij}. \tag{1.25}$$

Otra relación útil para trabajar con esta puerta cNOT es expresarla en función de **n** y **X**:

$$\mathbf{C}_{ij} = \bar{\mathbf{n}}_i + \mathbf{X}_j\mathbf{n}_i, \tag{1.26}$$

donde se puede ver fácilmente que cumple su función.

Podemos explicar una curiosa simetría del cNOT si definimos el operador

$$\mathbf{Z} = \bar{\mathbf{n}} - \mathbf{n} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{1.27}$$

Este operador es un poco particular, pues al aplicarlo sobre 1 se obtiene -1, y su interpretación es confusa. No obstante, este operador jugará un papel fundamental en la computación cuántica, al igual que **X**. Por ahora, en computación clásica, solamente se usará junto con otros operadores.

Debido a como se ha definido, se puede comprobar que el operador NOT, **X**, es anticonmutativo con **Z**:

$$\mathbf{ZX} = -\mathbf{XZ}. \tag{1.28}$$

Debido a que $\bar{\mathbf{n}} + \mathbf{n} = \mathbf{1}$, se pueden expresar $\bar{\mathbf{n}}$ y **n** en función de **Z** y $\mathbf{1}$:

$$\mathbf{n} = \frac{1}{2}(\mathbf{1} - \mathbf{Z}), \qquad \bar{\mathbf{n}} = \frac{1}{2}(\mathbf{1} + \mathbf{Z}), \tag{1.29}$$

formas que nos permitirán reescribir el operador cNOT en términos de **X** y **Z** y demostrar la asimetría que comentamos líneas atrás:

$$\mathbf{C}_{ij} = \frac{1}{2}(1 + \mathbf{Z}_i) + \frac{1}{2}\mathbf{X}_j(1 - \mathbf{Z}_i) = \frac{1}{2}\big(1 + \mathbf{X}_j\big) + \frac{1}{2}\mathbf{Z}_i\big(1 - \mathbf{X}_j\big), \tag{1.30}$$

donde la segunda forma se debe a que **X** y **Z** conmutan cuando $i \neq j$. Si a esta forma de escribir cNOT cambiásemos **X** y **Z**, obtendríamos la expresión anterior, pero con los subíndices cambiados. Es decir, cambiaríamos el Cbit de control por el objetivo, y viceversa; convertiríamos $\mathbf{C}_{ij}$ en $\mathbf{C}_{ji}$. Un operador que produce este efecto es la transformación de Hadamard o de Walsh-Hadamard, denotada con una **H** en computación, y que juega un papel importante en computación cuántica.

$$\mathbf{H} = \frac{1}{\sqrt{2}}(\mathbf{X} + \mathbf{Z}) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{1.31}$$

Teniendo en cuenta que $\mathbf{X}^2 = \mathbf{Z}^2 = \mathbf{1}$ y $\mathbf{XZ} = -\mathbf{ZX}$, se puede expresar **H** en términos de **X** y **Z** y obtener

$$\mathbf{H}^2 = \mathbf{1} \tag{1.32}$$

y

$$\mathbf{HXH} = \mathbf{Z}, \qquad \mathbf{HZH} = \mathbf{X}. \tag{1.33}$$

Estas expresiones permiten cambiar **X** y **Z** en la expresión de $\mathbf{C}_{ij}$ dada por (1.30):

$$\mathbf{C}_{ij} = (\mathbf{H}_i\mathbf{H}_j)\mathbf{C}_{ij}(\mathbf{H}_i\mathbf{H}_j). \tag{1.34}$$

A pesar de que la actuación de **H** sobre un solo Cbit empleando (1.31) no significa nada, el uso de este operador nos lleva a expresiones como esta última. En computación cuántica, la acción de la transformación de Hadamard sobre el estado de un Qbit no solo es considerablemente útil, sino que es sencilla su implementación, por lo que emplear la expresión (1.34) tiene importantes consecuencias.

Todas los operadores vistos en esta sección para n Cbits pueden ser representados matricialmente; no obstante, para los operadores reversibles más generales, existen $(2^n)!$ operaciones realizables sobre n Cbits, dadas por todas sus posibles permutaciones **P** de sus 2^n estados. Por lo que para 2 Cbits, por ejemplo, obtendríamos matrices de $4 \otimes 4$, por tanto, en la práctica, es más útil trabajar directamente con los operadores.

Por último, es interesante explicar una estructura alternativa para el operador swap. Empleando (1.29) para reescribir en términos de **n** y $\bar{\mathbf{n}}$ la expresión (1.21) de **S**, obtenemos:

$$\mathbf{S}_{ij} = \frac{1}{2}(\mathbf{1} + \mathbf{Z}_i\mathbf{Z}_j) + \frac{1}{2}(\mathbf{X}_i\mathbf{X}_j)(\mathbf{1} - \mathbf{Z}_i\mathbf{Z}_j). \tag{1.35}$$

Si definimos

$$\mathbf{Y} = \mathbf{ZX} = -\mathbf{XZ} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \tag{1.36}$$

podemos expresar de forma más compacta **S**:

$$\mathbf{S}_{ij} = \frac{1}{2}(\mathbf{1} + \mathbf{X}_i\mathbf{X}_j - \mathbf{Y}_i\mathbf{Y}_j + \mathbf{Z}_i\mathbf{Z}_j). \tag{1.37}$$

Para evitar el único signo negativo, podemos sustituir **Y** por $-i\mathbf{Y}$ (en un Cbit, multiplicar por i no difiere demasiado a multiplicar por -1). Renombramos ahora **X**, **Z** y $-i\mathbf{Y}$ adoptando una notación más cercana a la física:

$$\sigma_x = \mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad \sigma_y = -i\mathbf{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad \sigma_z = \mathbf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{1.38}$$

llegando a la expresión final del operador swap de la forma

$$\mathbf{S}_{ij} = \frac{1}{2}\left(\mathbf{1} + \sigma_x^{(i)}\sigma_x^{(j)} + \sigma_y^{(i)}\sigma_y^{(j)} + \sigma_z^{(i)}\sigma_z^{(j)}\right). \tag{1.39}$$

La introducción del factor i no solo nos otorga esta simetría; además para las tres matrices se cumple que su cuadrado es igual a la unidad

$$\sigma_x^2 = \sigma_y^2 = \sigma_z^2 = \mathbf{1}, \tag{1.40}$$

todas ellas son anticonmutativas a pares, y el producto de cualquiera dos de ellas presenta una relación sencilla con la tercera:

$$\begin{aligned}\sigma_x\sigma_y &= -\sigma_y\sigma_x = i\sigma_z,\\ \sigma_y\sigma_z &= -\sigma_z\sigma_y = i\sigma_x,\\ \sigma_z\sigma_x &= -\sigma_x\sigma_z = i\sigma_y,\end{aligned} \tag{1.41}$$

donde las relaciones difieren únicamente en una permutación cíclica de x, y y z.

Estas relaciones pueden recogerse en una sola identidad. Sean $\mathbf{a}$ y $\mathbf{b}$ dos vectores tridimensionales,

$$\mathbf{a} = (a_x, a_y, a_z), \qquad \mathbf{b} = (b_x, b_y, b_z), \tag{1.42}$$

y hacemos σ un vector formal donde sus tres componentes son los operadores σ_x, σ_y y σ_z,

$$\sigma = (\sigma_x, \sigma_y, \sigma_z), \tag{1.43}$$

de forma que se pueda definir el producto interno o escalar como

$$\mathbf{a} \cdot \sigma = a_x\sigma_x + a_y\sigma_y + a_z\sigma_z, \tag{1.44}$$

podemos escribir una única relación donde empleando (1.40) y (1.42) como

$$(\mathbf{a} \cdot \sigma)(\mathbf{b} \cdot \sigma) = \mathbf{a} \cdot \mathbf{b} + i(\mathbf{a} \times \mathbf{b}) \cdot \sigma, \tag{1.45}$$

donde $\mathbf{a} \times \mathbf{b}$ es el producto vectorial o producto cruz de $\mathbf{a}$ y $\mathbf{b}$.

Introduciendo i en la definición de σ_y, también la hemos hecho hermitiana[3] (o hermítica), al igual que σ_x y σ_z. Y es aquí donde se encuentra la aplicación final de todas estas transformaciones y definiciones: las matrices σ_x, σ_y y σ_z, junto con la matriz identidad $\mathbf{1}$, forman una base para el álgebra 4-dimensional de matrices bidimensionales de números complejos: cualquier matriz de este tipo es una combinación lineal de estas cuatro con coeficientes complejos. Debido a que las cuatro son hermitianas, cualquier matriz A bidimensional de coeficientes complejos será una combinación lineal real de las cuatro de la forma

$$A = a_0\mathbf{1} + \mathbf{a} \cdot \sigma, \tag{1.46}$$

donde a_0 y los componentes del vector $\mathbf{a}$ son todos números reales.

Las matrices σ_x, σ_y y σ_z fueron introducidas en los inicios de la mecánica cuántica por Wolfgang Pauli para describir el momento angular intrínseco asociado con el espín del electrón, y son utilizadas para muchos otros propósitos dentro de esta física, siendo uno de ellos, como hemos comprobado, la computación cuántica.

Estas matrices las emplearemos más adelante para, en cierto modo, realizar el control de nuestro Qbit de estudio, por lo que era de vital importancia familiarizarse con ellas.

1.3. Qbits y sus estados

Como hemos visto en los apartados anteriores, cada Cbit presenta únicamente dos estados posibles: $|0\rangle$ y $|1\rangle$. No obstante, las posibilidades que nos aporta un Qbit son considerablemente mayores. El estado $|\psi\rangle$ asociado a un Qbit puede ser cualquier vector unitario en el espacio

[3] Una matriz hermitiana es una matriz cuadrada de elementos complejos que es igual a su propia traspuesta conjugada o matriz adjunta. En mecánica cuántica, la matriz adjunta se indica con el símbolo † y se define como $(A_{ij})^\dagger = (A_{ji})^*$, donde * indica complejo conjugado. Por tanto, para una matriz A, se dice que es hermitiana si cumple $A_{ij} = (A_{ij})^\dagger$.

bidimensional generado por $|0\rangle$ y $|1\rangle$. Los escalares de este espacio bidimensional vectorial que contiene los estados de los Qbits son números complejos. El estado general de un Qbit es

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix}, \tag{1.47}$$

donde α_0 y α_1 son dos números complejos limitados únicamente por la condición de que $|\psi\rangle$, al igual que $|0\rangle$ y $|1\rangle$, debe ser un vector unitario en el espacio complejo. Por tanto, solo bajo la condición de normalización

$$|\alpha_0|^2 + |\alpha_1|^2 = 1 \tag{1.48}$$

podemos decir que el estado $|\psi\rangle$ es una superposición de los estados $|0\rangle$ y $|1\rangle$ con amplitudes α_0 y α_1. Se puede comprobar fácilmente que si uno de estos coeficientes es 0 y el otro es 1, nos encontramos con que el Qbit está en uno de los dos estados clásicos $|0\rangle$ y $|1\rangle$[4].

Al igual que el estado general de un Qbit (1.47) es una superposición normalizada arbitraria de los dos estados clásicos posibles, si disponemos de dos Qbits, el estado $|\psi\rangle$ que presenta es una superposición normalizada arbitraria de los cuatro estados ortogonales clásicos,

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle = \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix}, \tag{1.49}$$

con amplitudes complejas que deben cumplir la condición de normalización

$$|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1. \tag{1.50}$$

Esta generalización se puede expresar para n Cbits, donde el estado general es una superposición de los 2^n diferentes estados clásicos, donde sus amplitudes al cuadrado deben sumar la unidad:

$$|\psi\rangle = \sum_{0 \le x \le 2^n} \alpha_x |x\rangle_n, \tag{1.51}$$

$$\sum_{0 \le x \le 2^n} |\alpha_x|^2 = 1 \tag{1.52}$$

En el contexto de la computación cuántica, los 2^n estados clásicos que puede alcanzar n Qbits, son llamados *base computacional* o *base clásica*, y como es de esperar, son un subgrupo muy limitado de la computación cuántica.

Debemos realizar un inciso sobre los estados de n Cbits. Poniendo de ejemplo dos Qbits, si el estado de cada uno de ellos es $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ y $|\phi\rangle = \beta_0|0\rangle + \beta_1|1\rangle$, si aplicásemos la generalización realizada para los estados de varios Cbits, el estado de estos dos vendría dado por el producto tensorial de los estados individuales,

[4] Es interesante recordar que, si nos referimos a Cbits, podemos decir que "tienen valores"; sin embargo, cuando tratamos con Qbits, estos no "tienen valores", sino que "tienen estados", o mejor dicho, están descritos por estados. Esto es debido a que en los Qbits, al contrario de lo que ocurre en los Cbits, no existe una propiedad interna relacionada con sus estados.

$$|\psi\rangle = |\psi\rangle \otimes |\phi\rangle = (\alpha_0|0\rangle + \alpha_1|1\rangle) \otimes (\beta_0|0\rangle + \beta_1|1\rangle)$$
$$= \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle$$
$$= \begin{pmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \end{pmatrix}. \quad (1.53)$$

Sin embargo, comparando esta expresión con la de un estado general de 2-Qbit (1.49), esto no es más que un caso especial que se cumple para $\alpha_{00}\alpha_{11} = \alpha_{01}\alpha_{10}$. Pero como las amplitudes de la expresión general deben cumplir la normalización de (1.50), esta relación puede no cumplirse.

Un sistema multi-Qbit, en contraste con lo que ocurre con los Cbits, no siempre puede ser caracterizado por los estados individuales de estos Qbits. Esto de sebe a que cuando disponemos de varios Qbits, estos pueden encontrarse entrelazados. Esta propiedad no es única de este estudio, sino que ya fue introducida por Schrödinger en mecánica cuántica. Este fenómeno podremos comprobar que afecta, por ejemplo, a las mediciones para un sistema multi-Qbit del que deseásemos saber el estado de únicamente uno de los Qbits.

Hasta ahora, nos hemos referido el estado del Qbit a través de $|\psi\rangle$. No obstante, en nuestro caso, como podremos comprobar más adelante, trabajaremos empleando lo que se conoce como matriz densidad, ρ.

La matriz densidad es un formalismo matemático, presentado por von Neumann, que permite tratar fácilmente estados que no pueden ser descritos por una única función de ondas. Dado un estado $|\psi\rangle$, la matriz densidad se define como $\rho = |\psi\rangle\langle\psi|$, donde $\langle\psi|$ es el vector traspuesto conjugado de $|\psi\rangle$, es decir, $\langle\psi| = \left(|\psi\rangle^{\mathrm{T}}\right)^* = (|\psi\rangle)^\dagger$. En nuestro caso basta con que lo entendamos como una forma alternativa de expresar el estado de un sistema cuántico y saber que es una matriz hermítica $\rho = \rho^\dagger$, con traza $\mathrm{Tr}[\rho] = 1$ y cumpliendo que $\rho^2 = \rho$.

La traza de una matriz es la suma de los elementos de su diagonal. En el caso de la matriz densidad, esta es igual a uno debido a que estas componentes son los coeficientes o amplitudes de un estado cuántico sometidos a la normalización (1.52).

Por tanto, teniendo en cuenta la definición anterior, podemos expresar el estado (1.47) de un Qbit a través de la matriz densidad. Sabiendo que el estado de un Qbit es

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \alpha_0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \alpha_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix}, \quad (1.54)$$

podemos obtener la matriz densidad como

$$\rho = |\psi\rangle\langle\psi| = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} (\alpha_0^* \quad \alpha_1^*) = \begin{pmatrix} |\alpha_0|^2 & \alpha_0\alpha_1^* \\ \alpha_0^*\alpha_1 & |\alpha_1|^2 \end{pmatrix}, \quad (1.55)$$

donde se ha tenido en cuenta que el producto de un número complejo por su conjugado es igual a su módulo al cuadrado. Una vez calculada, se verifica que la traza de esta ha de ser igual a la unidad.

Si el estado $|\psi\rangle$ fuese uno de los estados de la base clásica, por ejemplo, $|0\rangle$, se obtiene

$$\rho = |\psi\rangle\langle\psi| = \begin{pmatrix} 1 \\ 0 \end{pmatrix} (1 \quad 0) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad (1.56)$$

resultado que se podía intuir a través del último término de (1.55).

Como se ha comprobado, la matriz densidad ofrece la misma información que el estado de la forma $|\psi\rangle$. No obstante, suele emplearse debido a la facilidad para expresar estados mezcla. En mecánica cuántica, un estado mezcla, en contraposición con un estado puro, es aquel que no está máximamente determinado. Supongamos que disponemos de un Qbit de un conjunto de n Qbits, donde la probabilidad de obtener el estado $|\psi_1\rangle$ es P_1 y la de obtener $|\psi_2\rangle$ es P_2. La matriz densidad viene dada por

$$\rho = P_1|\psi_1\rangle\langle\psi_1| + P_2|\psi_2\rangle\langle\psi_2|, \tag{1.57}$$

donde las probabilidades suman la unidad: $P_1 + P_2 = 1$.

Simplifiquemos el caso anterior de tal forma que $|\psi_1\rangle = |0\rangle$ y $|\psi_1\rangle = |1\rangle$:

$$\rho = P_1|0\rangle\langle 0| + P_2|1\rangle\langle 1| = P_1\begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix} + P_2\begin{pmatrix}0 & 0\\ 0 & 1\end{pmatrix} = \begin{pmatrix}P_1 & 0\\ 0 & P_2\end{pmatrix} \tag{1.58}$$

Por tanto, la matriz densidad no solo ofrece la misma información, sino que en estos estados mezcla es capaz de recoger en un solo operador las probabilidades de los diferentes subestados. Es por este motivo por el que adquiere tanta importancia en mecánica cuántica.

1.4. Operaciones reversibles en Qbits

En computación cuántica, las operaciones reversibles aplicadas sobre el estado de un Qbit vienen dadas por cualquier transformación lineal que convierta el vector unitario de ese estado en otro vector unitario. Estas transformaciones **u** son llamadas unitarias y satisfacen la condición

$$\mathbf{u}\mathbf{u}^\dagger = \mathbf{u}^\dagger\mathbf{u} = 1. \tag{1.59}$$

Debido a que toda transformación unitaria tiene su inverso unitario, estas acciones sobre un Qbit son reversibles, característica fundamental en computación cuántica. Estas operaciones sobre un único Qbit son conocidas como puertas 1-Qbit, y su acción aparece representada en la Figura 1.1[5].

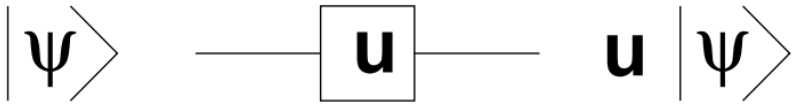

*Figura 1.1 Acción de una puerta 1-Qbit **u** sobre un Qbit*

El caso más general de operación reversible sobre n Cbits en computación clásica es la permutación de cualquiera de sus $(2^n)!$ estados posibles. En computación cuántica, el caso más general de operación reversible sobre n Qbits está representada por cualquier transformación lineal unitaria de unos vectores iniciales a otros finales. Cualquier transformación unitaria **U** de dimensión 2^n cumple

[5] Para las representaciones de los circuitos, se emplean las siguientes convenciones: un Qbit es representado con una línea horizontal fina, mientras que para varios de ellos la línea será gruesa; las puertas se representan por cajas o círculos sobre la línea horizontal; los Qbits de entrada se encuentran en un ket a la izquierda del diagrama, mientras que las salidas a la derecha, leyendo el diagrama de izquierda a derecha en el orden de actuación de las puertas, pero siendo esta la secuencia contraria a como aparecen estas puertas en las ecuaciones.

$$\mathbf{U}\mathbf{U}^{\dagger} = \mathbf{U}^{\dagger}\mathbf{U} = 1. \tag{1.60}$$

Estas transformaciones son llamadas puertas n-Qbits.

La representación para una única puerta n-Qbit es igual que para las puertas 1-Qbit, pero cambiando el operador **u** por **U**, por ese mismo motivo, mostramos el caso para dos transformaciones unitarias sucesivas: primero se aplica **V** sobre los Qbits y después **U**, que se puede expresar físicamente como $\mathbf{U}(\mathbf{V}|\psi\rangle) = \mathbf{U}\mathbf{V}|\psi\rangle$ y se representa como aparece en la Figura 1.2.

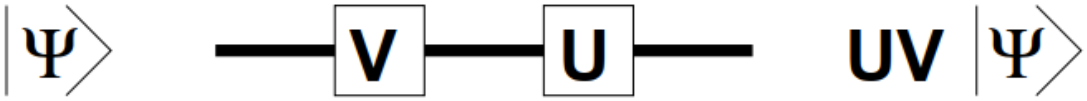

*Figura 1.2 Acción de dos puertas n-Qbit **V** y **U** sobre n Qbits*

Como se ha comentado brevemente con anterioridad, cualquier operación **P** que actúa sobre n Cbits puede asociarse con su correspondiente operación unitaria **U** en n Qbits. Para ello, únicamente se define la acción **U** en la base clásica para ser idéntica a **P** en los correspondientes estados clásicos, y debido a que se ha definido **U** sobre una base clásica, esta puede extenderse para n Qbits por linealidad, lo que implica únicamente la permutación arbitraria de sus amplitudes manteniendo la norma y la linealidad y, por tanto, la unitariedad. Muchas operaciones unitarias sobre Qbits son definidas de este modo, como extensión lineales de las operaciones clásicas sobre Cbits. No obstante, no todas se definirán de este modo tan sencillo.

Como se puede deducir del párrafo anterior, el uso e importancia de estas transformaciones unitarias no se debe tanto a su unitariedad, sino principalmente a su linealidad, que claro está, es una característica de esta.

Por lo general, en el diseño actual de algoritmos cuánticos, las transformaciones unitarias son restringidas a aquellas que actúan únicamente sobre un Qbit o dos en un instante. Esto es debido a que, por un lado, la complejidad aumenta considerablemente cuanto mayor es el orden de las puertas; por otro lado, no supone un grave problema debido a que todas las operaciones lógicas clásicas, que se pueden diseñar con 3 Cbits en computación clásica, se pueden reconstruir con puertas 1-Qbit y 2-Qbits en computación cuántica.

1.5. Medida de Qbits

Para especificar el estado de un Cbit, únicamente se necesita saber si ese estado es $|0\rangle$ o $|1\rangle$. No obstante, para conocer el estado de un Qbit con una cierta precisión, es necesario conocer las amplitudes de (1.47), que son coeficientes complejos sujetos a la condición de normalización (1.48).

No obstante, esto no supone un problema grave. El mayor inconveniente está en cómo adquirir esa información. En la computación clásica, para saber en qué estado se encuentra un Cbit basta con mirar, sin que esta lectura altere su funcionamiento. Sin embargo, en la computación cuántica, debido a su naturaleza, la situación es totalmente diferente.

En esencia, no se puede extraer la información contenida en las amplitudes α_x que definen el estado de un Qbit. No se pueden leer estos valores y, por tanto, no se puede saber qué estado presenta el Qbit. Como se ha comentado anteriormente, el estado de n Qbits no está asociado con ninguna propiedad específica de esos Qbits, al contrario de lo que ocurre con los Cbits. Solo

existe una posibilidad para extraer información de n Qbits en un estado dado: realizar una medición.

Realizar una medición consiste en realizar una prueba al Qbit, obteniendo como salida 0 o 1. Sin embargo, esta salida no está determinada, por lo general, por el estado $|\psi\rangle$. Este estado solamente establece la probabilidad de que se produzca una salida u otra; es decir, los coeficientes representan la probabilidad de que se obtenga ese resultado.

La probabilidad de obtener un resultado en particular viene dada por el cuadrado de la amplitud de ese estado $|x\rangle$ dado en la superposición de $|\psi\rangle$ expresado en la base clásica de 2^n. Es decir, si el estado de n Qbits es

$$|\psi\rangle_n = \sum_{0 \le x \le 2^n} \alpha_x |x\rangle_n, \tag{1.61}$$

entonces la probabilidad de obtener 0's o 1's debido a la medida dependerá de la expresión binaria del entero x:

$$p(x) = |\alpha_x|^2. \tag{1.62}$$

Esta regla básica indica cómo extraer información de un estado cuántico y fue enunciada por Max Born, motivo por el que se conoce como regla de Born. Establece la relación entre los coeficientes y los números que se obtienen en la medición de Qbits. Por tanto, se puede dar sentido físico a la condición de normalización (1.52) a la que están sometidos los Qbits.

Este proceso de medición se realiza mediante un hardware con un monitor digital conocido como puerta de medición n-Qbit, y aparece representado en la Figura 1.3. Al contrario de lo que ocurre con el resto de puertas (puertas unitarias), la puerta de medición no es lineal; es decir, podemos obtener una misma salida para diferentes entradas debido a que se basa en probabilidades. Una vez realizada la medición, no se puede deshacer la acción: no hay modo de reconstruir el estado inicial $|\psi\rangle$ a partir del estado final $|x\rangle$.

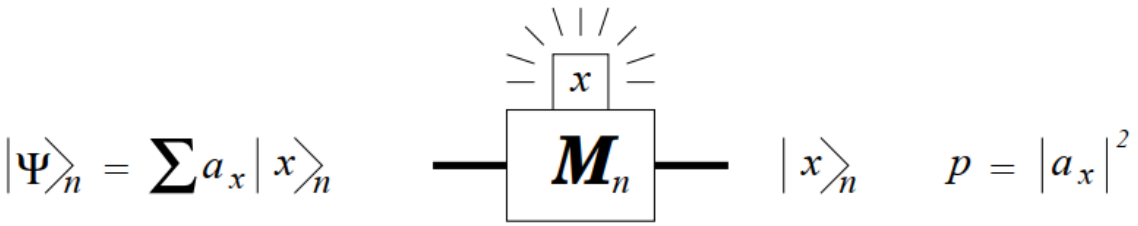

Figura 1.3 Acción de una puerta de medición

La aplicación más sencilla, y la que más nos interesa, de la regla de Born es para un solo Qbit. Si el estado de un Qbit es la superposición (1.47) de los estados $|0\rangle$ y $|1\rangle$ con amplitudes α_0 y α_1, entonces el resultado de la medición es 0 con probabilidad $|\alpha_0|^2$ o 1 con probabilidad $|\alpha_1|^2$.

Hay que tener en cuenta que, una vez obtenido un estado de salida, toda la información de las amplitudes del resto de posibles estados desaparece y únicamente ha servido para obtener ese resultado final, sabiendo que probablemente esas amplitudes eran menores que la que acompañaba al posible estado final en el momento de ser medido. Una vez realizada la medición, el estado inicial $|\psi'\rangle$ del Qbit será $|x\rangle$, con coeficiente unidad.

Toda esa información perdida se denomina a menudo "reducción" o "colapso" del estado; como consecuencia de la medición, el estado previo a esta se reduce o colapsa al estado posterior a la

misma. Tampoco debe entenderse esto como un hecho problemático, teniendo en cuenta que estos coeficientes solo son símbolos abstractos que calculan probabilidades de salida y que no se corresponden con ninguna propiedad interna.

Dicho esto, el objetivo de la computación cuántica es reducir la mayoría de las amplitudes a valores muy próximos o iguales a cero, manteniendo altos los coeficientes de los valores que se desean obtener. Por este motivo, es fundamental realizar operaciones en las que se pueda comprobar fácilmente que el resultado obtenido es el deseado, siendo esta una cuestión de importancia en la computación cuántica.

Existe una versión más específica de la regla de Born, conocida como *regla de Born generalizada*. Esta se aplica cuando se quiere medir solamente un Qbit de un conjunto de n Qbits, empleando una puerta de medición 1-Qbit. Se basa en que cualquier estado $|\psi\rangle$ de n Qbits se puede expresar de la forma

$$|\psi\rangle = a_0|0\rangle|\phi_0\rangle + a_1|1\rangle|\phi_1\rangle, \qquad |a_0|^2 + |a_1|^2 = 1, \tag{1.63}$$

donde el Qbit a medir aparece a la izquierda y donde $|\phi_0\rangle$ y $|\phi_1\rangle$ son estados normalizados (pero no necesariamente ortogonales) de los $n-1$ Qbits no medidos. Esta ecuación viene del hecho de que la forma general (1.61) de $|\psi\rangle$ se puede expresar de la forma (1.63) con $|\phi_0\rangle$ y $|\phi_1\rangle$ dadas por

$$|\phi_0\rangle \propto \sum_{x=0}^{2^{n-1}-1} \alpha_x|x\rangle_{n-1}, \qquad |\phi_1\rangle \propto \sum_{x=0}^{2^{n-1}-1} \alpha_{x+2^{n-1}}|x\rangle_{n-1}. \tag{1.64}$$

La regla generalizada de Born afirma que si solo se mide el estado del 1-Qbit a la izquierda de la expresión (1.63), entonces la puerta de medición 1-Qbit indicará x (0 o 1) con probabilidad $|a_x|^2$, mientras que el estado de los n Qbit será el producto $|x\rangle|\phi_x\rangle$. Esta acción de la puerta de medición 1-Qbit sobre n-Qbit aparece representada en la Figura 1.4.

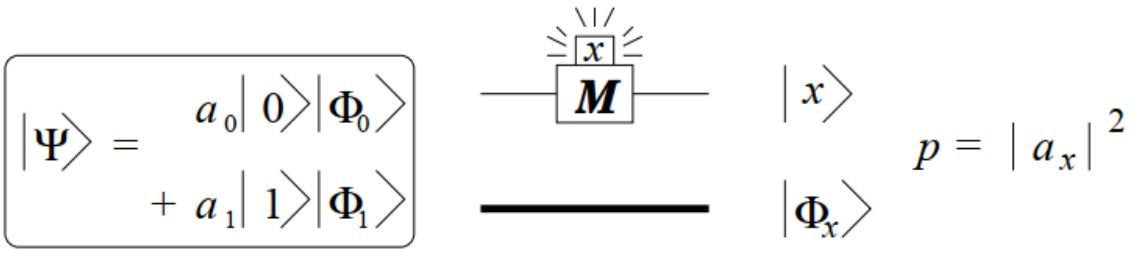

Figura 1.4 Regla de Born generalizada

En la Figura 1.5 se representa el caso en que la puerta 1-Qbit actué sobre un Qbit que inicialmente no esté entrelazado con los $n-1$ Qbits restantes. De este modo, la medición está gobernada por la regla de Born ordinaria, mientras que el resto de Qbits no medidos simplemente se mantienen en el mismo estado que antes de la medición, hecho que se podía intuir también de la regla de Born generalizada, en cuyo caso los dos estados $|\phi_0\rangle$ y $|\phi_1\rangle$ son idénticos.

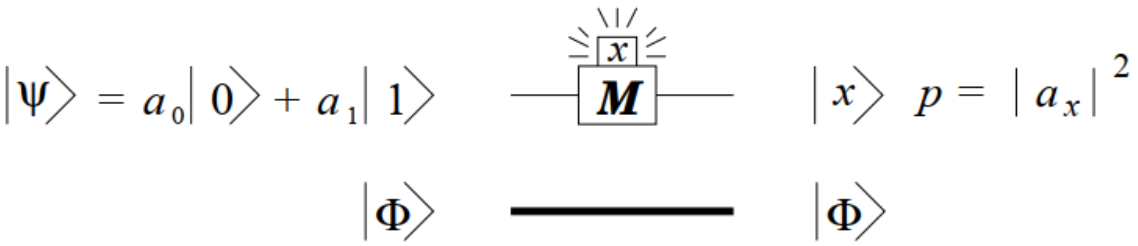

Figura 1.5 Regla de Born generalizada para Qbits desentrelazados

Si se aplica la regla de Born generalizada n veces, una vez a cada uno de los n Qbits, mediante puertas de medición 1-Qbit, se puede comprobar que el estado final de los n Qbits es x con probabilidad $|\alpha_x|^2$, donde x es el entero de n bits cuyos bits son dados por las lecturas de las n puertas de medición 1-Qbit. Esto no es más que la regla de Born ordinaria donde n puertas de medición 1-Qbit juegan el mismo papel que una sola puerta de medición n-Qbit. Es decir, con la puerta de medición 1-Qbit se puede construir una puerta de medición n-Qbit, como se representa en la Figura 1.6.

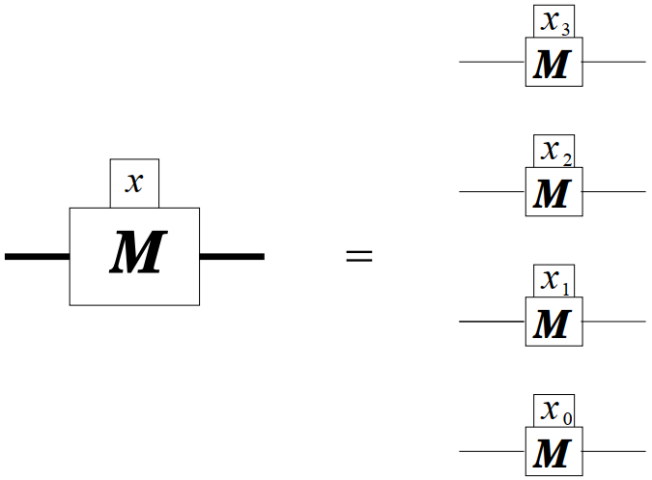

Figura 1.6 Construcción de una puerta de medición n-Qbit a partir de n puertas de medición 1-Qbit

Se puede extender la regla de Born generalizada para el caso de querer medir más de un Qbit. El estado general de $m + n$ Qbits puede escribirse como

$$|\psi\rangle_{m+n} = \sum_x \alpha_x |x\rangle_m |\phi_x\rangle_n \tag{1.65}$$

donde $\sum_x |\alpha_0|^2 = 1$ y los estados $|\phi\rangle_n$ están normalizados, pero no son necesariamente ortogonales. Aplicando la regla de Born generaliza m veces a m Qbits en un estado $m + n$ Qbit, la regla establece que solo los m Qbits a la izquierda de la expresión son medidos, con probabilidad $|\alpha_x|^2$ de que el resultado sea x, mientras que después de la medición el estado de los $m + n$ Qbits vendrá dado por el producto

$$|x\rangle_m |\phi_x\rangle_n \tag{1.66}$$

donde los m Qbits medidos presentan el estado $|x\rangle_m$, y los n Qbits sin medir, el estado $|\phi_x\rangle_n$.

Es importante comprender con claridad la regla de Born. El estado del Qbit antes de ser medido no es $|0\rangle$ o $|1\rangle$ con una cierta probabilidad, sino que es una superposición de ambos: es $|0\rangle$ y $|1\rangle$. Una vez realizada la medición, el Qbit adquiere un valor teniendo en cuenta esas probabilidades.

Si el estado (1.61) es uno de los estados $|x\rangle_n$ de la base computacional 2^n de forma que $\alpha_x = 1$ y $\alpha_y = 0$ cuando $x \neq y$, entonces la regla de Born se reduce a que el resultado de la medición es x con probabilidad 1 y, por tanto, el estado final alcanzado después de la medición es el mismo que el estado previo a esta. Mediante la computación cuántica se pueden restringir los valores de n Qbits a los estados de la base computacional, donde no se producirán cambios en los estados al realizar la medición.

Hemos visto cómo funciona la regla de Born y se ha definido en base a un estado $|\psi\rangle$. No obstante, también explicaremos brevemente el formalismo de la matriz densidad para la medida. Para un estado $|\psi\rangle$, definiendo los operadores $\mathbb{P}_0 = |0\rangle\langle 0|$ y $\mathbb{P}_1 = |1\rangle\langle 1|$, la probabilidad de obtener $|0\rangle$ o $|1\rangle$ vendrá dada por

$$P_x = X^\dagger \rho X = \mathrm{Tr}[X\rho], \tag{1.67}$$

donde X es el operador del observable x. Por tanto, de este modo existe una probabilidad P_0 de obtener $|0\rangle$ y P_1 de obtener $|1\rangle$. Se puede comprobar para el estado (1.55), pues

$$P_0 = \mathrm{Tr}[\mathbb{P}_0\rho] = Tr\left[\begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}\begin{pmatrix}|\alpha_0|^2 & \alpha_0\alpha_1^*\\ \alpha_0^*\alpha_1 & |\alpha_1|^2\end{pmatrix}\right] = \mathrm{Tr}\left[\begin{pmatrix}|\alpha_0|^2 & 0\\ 0 & 0\end{pmatrix}\right] = |\alpha_0|^2, \tag{1.68}$$

$$P_1 = \mathrm{Tr}[\mathbb{P}_1\rho] = Tr\left[\begin{pmatrix}0 & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}|\alpha_0|^2 & \alpha_0\alpha_1^*\\ \alpha_0^*\alpha_1 & |\alpha_1|^2\end{pmatrix}\right] = \mathrm{Tr}\left[\begin{pmatrix}0 & 0\\ 0 & |\alpha_1|^2\end{pmatrix}\right] = |\alpha_1|^2. \tag{1.69}$$

De este modo se puede apreciar la regla de Born para la medida también en la matriz densidad.

Antes de finalizar la sección, debemos comentar que las puertas de medición, además de proporcionar un estado final, también se utilizan al principio del cálculo para determinar y fijar unos valores iniciales en los Qbits; de otro modo, no se podría saber en qué estado se encuentran estos Qbits al principio, determinar si esta colección de Qbits está entrelazada con otros ni, por tanto, saber si presentan un estado propio.

De este modo es como realmente se opera: se hace pasar por puertas de medición a n Qbits, y a aquellos que presentan estado 1 se les aplica el operador NOT, obteniendo como resultado 0. Este es el estado inicial que toman la mayoría de los algoritmos de computación cuántica. Este proceso aparece representado en la Figura 1.7.

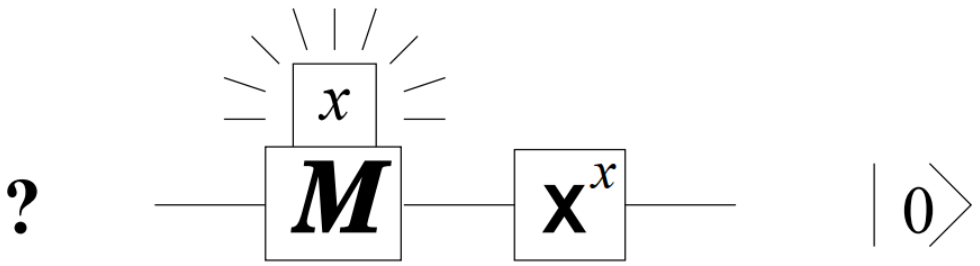

Figura 1.7 Preparación del estado de un Qbit mediante puerta de medición y operador NOT

Por tanto, podemos concluir el apartado diciendo que dos son las funciones que realiza la medición en computación cuántica: preparan los Qbits para los cálculos posteriores en el ordenador, y tras realizar estos, se extrae del Qbit una salida digital. La primera acción inicial se denomina "preparación de estado", pues define el estado inicial del Qbit. De este modo, se pueden aplicar los operadores unitarios correspondientes, derivando en una transformación unitaria del estado inicial que, mediante la regla de Born, determina las probabilidades del estado final.

1.6. Tabla comparativa: Cbits vs. Qbits

Para finalizar con esta introducción teórica, se recogerá en una tabla las características más destacables de los Cbits y Qbits. Emplearemos el término "Bit", con *B* mayúscula, para referirnos al "Qbit o Cbit" según el caso (para diferenciar de "bit", con *b* minúscula, que significa "0 o 1").

Tabla 1. Comparación Cbits y Qbits

Cbits vs Qbits	**Cbits**	**Qbits**
Estados de n Bits	$\|x\rangle_n, \quad 0 \leq x \leq 2^n$	$\sum \alpha_x \|x\rangle_n, \quad \sum \|\alpha_x\|^2 = 1$
Subconjuntos de n Bits	Siempre presentan estados	Generalmente no tienen estados
Operaciones reversibles sobre los estados	Permutaciones	Transformaciones unitarias
¿Relación del estado con alguna propiedad del Bit?	Sí	No
Para saber el estado del Bit	Examinarlo	No se puede saber directamente
Para obtener información de Bits	Basta con mirarlos	Medición
Información obtenida	x	x con probabilidad $\|\alpha_x\|^2$
Estado después de adquirir la información	El mismo: se mantiene en $\|x\rangle$	Diferente: cambia a $\|x\rangle$

Capítulo 2

Modelo para el control de un Qubit

En este capítulo realizaremos una introducción al control de sistemas cuánticos monitorizados, donde continuaremos presentando algunos conceptos de mecánica cuántica. Una vez visto cómo se procede en el control de este tipo de sistemas, explicaremos el modelo de Qubit que se empleará en este proyecto, incluyendo la instalación experimental a través de la que se definen las ecuaciones empleadas. Por último, se presentará el control propuesto en la bibliografía.

2.1. Control cuántico realimentado (QFC)

En el control cuántico realimentado (*Quantum Feedback Control*[6]) [3], un observador monitoriza un sistema cuántico de forma continua, utilizando la información obtenida en esta medida para controlar el sistema mediante la modificación de su Hamiltoniano y/o el observable que se está midiendo.

Brevemente podemos decir que, en mecánica cuántica, el Hamiltoniano H es un operador que cuantifica la energía total del sistema. Es importante no confundir su nomenclatura con el operador Hadamard **H** (en negrita). Por otro lado, se dice que un sistema es *observable* si cualquier estado x_0 puede ser determinado mediante la salida del sistema y durante un tiempo finito. Por tanto, un observable puede ser cualquier variable medida a la salida que nos permita conocer el estado del sistema.

Por último, antes de continuar, debemos diferenciar entre la medición comentada en la sección 1.5 y la monitorización realizada en un QFC. En este último caso, se realiza una medida continua en el tiempo de tal modo que esta se puede entender como una "parte" del sistema, permitiendo conocer las amplitudes de los estados de la superposición: se observa la evolución formando parte de ella. Sin embargo, la puerta de medición es una acción puntual, en un determinado instante, que únicamente nos devuelve un estado del sistema generador, perdiendo toda la información de los coeficientes. Más adelante explicaremos con más detalle que entendemos por *monitorización*.

En consecuencia, al contrario de lo que ocurre en el control clásico, en QFC la medición, que es una parte del lazo de realimentación, afecta a la dinámica del sistema. No obstante, se puede estudiar en el marco de la teoría de control clásica. Esto se debe a que los sistemas cuánticos no dejan de ser ejemplos de sistemas dinámicos no lineales con ruido. Lo mismo ocurre para otras ramas como la economía.

Por tanto, en el caso de sistemas cuánticos monitorizados, se pueden aplicar las técnicas de control clásicas. No obstante, las herramientas más poderosas solamente se pueden aplicar a sistemas lineales, por lo que son poco útiles para los sistemas cuánticos.

Nos centraremos, por tanto, en hacer una breve explicación del tipo de sistema al que vamos a realizar control y sus particularidades.

La dinámica de un sistema cuántico aislado viene dada por la ecuación de Schrödinger. Si describimos el sistema empleando la matriz densidad, ρ, la expresión viene dada por

$$\dot{\rho} = \frac{d\rho}{dt} = -\left(\frac{i}{\hbar}\right)[H, \rho], \tag{2.1}$$

donde $\hbar = h/2\pi$ es la constante de Planck reducida o constante de Dirac y $[H, \rho]$ es el conmutador de H y ρ. Sin entrar en detalle, el conmutador de dos operadores lineales A y B es

[6] QFC.

un nuevo operador definido por la diferencia del producto de operadores: $[A, B] = AB - BA$. Cuando su resultado es igual a 0, los operadores conmutan y el orden de actuación no afecta al estado cuántico al que se la aplica; en caso contrario, no conmutan y el orden sí afecta al estado final.

Volviendo a la ecuación anterior, si tenemos en cuenta el entorno, este introduce ruido, que puede ser añadido a la ecuación como términos de la ecuación maestra de Lindblad[7] del siguiente modo

$$\dot{\rho} = -\left(\frac{i}{\hbar}\right)[H, \rho] + \gamma D[c]\rho \tag{2.2}$$

con

$$D[c]\rho = 2c^{\dagger}\rho c - c^{\dagger}c\rho - \rho c^{\dagger}c, \tag{2.3}$$

γ una constante de velocidad, un ratio, y c un operador que depende del acoplamiento entre el sistema y el entorno.

Podemos añadir también a la ecuación el efecto producido por la monitorización del sistema, aunque conviene explicar en primer lugar qué entendemos por *monitorización*.

Monitorizar un sistema consiste en realizar una lectura continua de alguna propiedad del sistema. Poniendo como ejemplo de propiedad la posición x, en cada paso de tiempo dt, se obtiene una pequeña información de esta.

Para entender por qué se obtiene una pequeña información, simplemente tenemos que referirnos al concepto clásico de medición: todas las medidas reales son inexactas, es decir, obtenemos en ellas el valor verdadero de la propiedad, x, más un error, ϵ, que es aleatorio, y que limita el resultado. El resultado de una medición clásica para un intervalo de tiempo dt se puede expresar como

$$R = x + \epsilon, \tag{2.4}$$

pudiendo expresar su equivalente cuántico como

$$R = \langle X \rangle + \epsilon, \tag{2.5}$$

donde X es el operador para el observable x y $\langle X \rangle = \mathrm{Tr}[X\rho]$ es su valor esperado.

Si el error es Gaussiano, es decir, sigue una distribución normal (que es lo más probable por el Teorema Central del Límite[8], siendo este el que estudiaremos), se puede demostrar que la varianza del error, $Var(\epsilon) = \sigma_{\epsilon}^2$, es proporcional a $1/dt$ y que ϵ lo es a $1/\sqrt{dt}$. Por este motivo, el valor de R se vuelve infinito conforme $dt \to 0$. Para evitarlo, podemos escribir las ecuaciones en términos de $dr = Rdt$, una cantidad finita. Por lo que tenemos

$$dr = \langle X \rangle dt + g dW, \tag{2.6}$$

[7] En mecánica cuántica, la ecuación maestra de Lindblad es simplemente un caso general de un proceso de Markov, que es un modelo estocástico en la que la probabilidad de cada evento depende únicamente del evento anterior.

[8] El Teorema Central del Límite postula que, en condiciones muy generales respecto a las distribuciones de los sumandos, la suma de variables aleatorias independientes tiende a distribuirse normalmente a medida que aumenta el número de sumandos.

donde g es una constante o función que determina la cantidad de ruido y dW es una variable aleatoria gaussiana con media cero y varianza igual a dt.

La introducción del término dW se ha podido realizar debido a que, en un principio, nuestro error ϵ se podía considerar como un ruido blanco ξ ya que no presenta correlación en el tiempo. El término dW, por su parte, hace referencia a un proceso de Wiener [4]. Este es el modelo matemático empleado para describir el movimiento Browniano, por lo que se suelen emplear ambos nombres indistintamente. No obstante, en un sentido estricto, el movimiento Browniano es el proceso físico; el proceso estocástico más básico que, en síntesis, presenta trayectorias continuas, incrementos independientes y distribución normal. Como hemos considerado por el TCL que el error, es decir, el ruido blanco, presenta distribución normal, podemos considerar la relación existente entre la derivada respecto al tiempo del proceso de Wiener y el ruido blanco, dada por

$$\xi(\cdot) = \frac{dW(\cdot)}{dt}. \tag{2.7}$$

Con esta relación podemos comprender el cambio realizado para obtener (2.6).

En cada intervalo dt, el valor dW es tomado de la función de densidad de probabilidad para una distribución normal

$$p(dW) = \frac{1}{\sigma\sqrt{2\pi}} e^{[-(dW-\mu)^2/(2\sigma^2)]} = \frac{1}{\sqrt{2\pi dt}} e^{[-dW^2/(2dt)]}. \tag{2.8}$$

Por tanto, monitorizar no es más que realizar una medición continua de una cantidad física x, que proporcione un flujo continuo de resultados de medición $dr(t)$. Ahora debemos saber cómo varía el estado ρ en cada intervalo de tiempo como resultado de la medición. En un intervalo dt, el estado $\rho(t)$ varía hasta $\rho(t) + \mathrm{d}\rho$, con

$$d\rho = -kD[X]\rho + \sqrt{2k}(X\rho + \rho X - 2\langle X\rangle\rho)dW, \tag{2.9}$$

donde $k = 1/(8g^2)$ es la velocidad a la que se extrae información de x del sistema y, por lo general, $\langle X\rangle = \mathrm{Tr}[X\rho]$. El término dW que aparece en esta ecuación proviene directamente de la medición de dr.

Normalmente, se suele dividir $d\rho$ entre dt para obtener la variación respecto al tiempo $\dot{\rho}$, y escribir la ecuación de movimiento para ρ usando esta velocidad. También se puede expresar en función de los diferenciales. Podemos expresar la ecuación completa de un sistema con ruido y monitorizado como

$$\begin{aligned} d\rho = -\left(\frac{i}{\hbar}\right)[H,\rho]dt + D[c]\rho dt - kD[X]\rho dt \\ + \sqrt{2k}(X\rho + \rho X - 2\langle X\rangle\rho)dW, \end{aligned} \tag{2.10}$$

que es una ecuación estocástica, es decir, contiene un término que fluctúa aleatoriamente.

Las ecuaciones estocásticas presentan ciertas particularidades, por lo que es necesario usar las reglas del cálculo estocástico (ver [5]). A diferencia de lo que ocurre en el cálculo determinista donde $dt^2 = 0$ debido a que un infinitesimal elevado al cuadrado o una potencia mayor es despreciable respecto al resto de términos, en el cálculo estocástico la regla básica es que $dW^2 = dt$, mientras que $dt^2 = dtdW = 0$; es decir, los términos dW^2 se mantendrán en las expansiones de las ecuaciones estocásticas y podrán ser sustituidos por dt, mientras que dt^2 será despreciado. Además de esto, la regla de la cadena usual para derivadas tampoco se puede

aplicar, siendo necesario emplear la fórmula de Itô, entre otras particularidades. Para la resolución de estas ecuaciones en este proyecto, emplearemos métodos numéricos.

Ahora que disponemos de la ecuación sobre la evolución de nuestro sistema monitorizado, debemos incluir el feedback. En QFC, la forma de realizar control es modificando el Hamiltoniano del sistema, H, y también, si se desea, el observable medido X, haciendo que estos sean función de los resultados de la medición. Específicamente, se permite que H y X en el tiempo t sean cualquier función de los resultados de la medición $dr(t)$ obtenidos hasta ese momento.

La matriz densidad en el tiempo t determina completamente el comportamiento futuro del sistema, por lo que la mayoría de los protocolos de control se realizan haciendo $H(t)$ y $X(t)$ funciones de $\rho(t)$. Por lo tanto, el observador monitoriza $\rho(t)$ para resolver (2.10) desde el tiempo inicial hasta el tiempo t y, tras ello, se elige $H = H(t,\rho)$ y $X = X(t,\rho)$. Por lo que la ecuación para la evolución de ρ incluyendo el control es

$$d\rho = -\left(\frac{i}{\hbar}\right)[H(t,\rho),\rho]\mathrm{d}t + D[c]\rho\mathrm{d}t - kD[X(t,\rho)]\rho\mathrm{d}t + \sqrt{2k}[X(t,\rho)\rho + \rho X(t,\rho) - 2\langle X(t,\rho)\rangle\rho]\mathrm{d}W. \quad (2.11)$$

El problema del control es determinar $H(t,\rho)$ y/o $X(t,\rho)$ con el fin de alcanzar la evolución deseada de la forma más precisa posible en presencia de ruido. Este objetivo último puede ser: alcanzar el estado deseado para un tiempo T, producir una evolución lo más parecida posible a la deseada, o llegar a un estado deseado de la forma más rápida posible. En nuestro control, estas tres características serán las que analizaremos, además de otras como su robustez, por ejemplo.

2.2. Modelo físico de un Qubit y control propuesto

En este proyecto, vamos a estudiar la evolución de un solo Qubit de estado sólido, tal y como aparece en [6]. Para este modelo podemos emplear el formalismo Bayesiano[9], que nos permite tener en cuenta el ruido debido a la medición continua y aplicar el control con un conocimiento explícito de esta. Este formalismo muestra que un detector ideal (cuantitativamente limitado) puede monitorizar con precisión la evolución de un Qubit. Esto nos permite controlar la evolución aleatoria del Qubit haciendo que siga una trayectoria deseada. Es importante tener en cuenta que las perturbaciones añadidas por el detector deben poder ser eliminadas eficientemente en la realimentación. El control diseñado en este modelo pretende mantener las oscilaciones del Qubit durante un tiempo infinito, intentando minimizar la diferencia de fases estimada y deseada.

El control que se va a aplicar aparece esquematizado en la Figura 2.1. En ella, se puede comprobar que el control se aplica a la medición del Qubit junto con el ruido ambiente y la perturbación añadida por el detector. El procesador simplemente "traduce" la señal y el controlador produce la señal de control que se aplica sobre el Qubit a través del actuador.

[9] En la teoría clásica de probabilidad, se emplea el Teorema de Bayes para tratar con sucesos en los que se dispone de información incompleta. El teorema de probabilidad total hace inferencia sobre un suceso B a partir de los resultados de los sucesos A; por su parte, el teorema de Bayes calcula la probabilidad de A condicionado a B. Es este el motivo por el que el formalismo Bayesiano toma el nombre de este teorema.

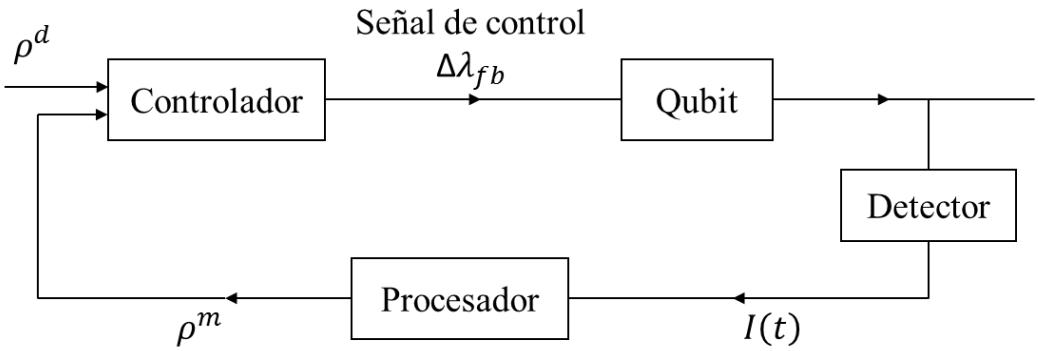

Figura 2.1 Esquema para el control de un Qubit de estado sólido con ruidos procedentes del entorno y la medición

La medida monitorizada que proporciona el procesador aparece con el superíndice m (de "*monitored*" en inglés), mientras que la deseada con la que se compara presenta el superíndice d (de "*desired*" en inglés). Nuestro observable será la matriz densidad ρ y el control lo realizaremos empleando el Hamiltoniano $H = H(t, \rho)$.

El lazo presentado controla el Qubit caracterizado por el Hamiltoniano

$$H_{qb} = \frac{\varepsilon_{fb}}{2}\sigma_z + \lambda_{fb}\sigma_x = \begin{pmatrix} \frac{\varepsilon_{fb}}{2} & \lambda_{fb} \\ \lambda_{fb} & -\frac{\varepsilon_{fb}}{2} \end{pmatrix}, \tag{2.12}$$

donde ε es la asimetría de energía entre dos niveles del Qubit, es decir, entre los dos estados localizados $|0\rangle$ y $|1\rangle$ que presenta, y λ es la amplitud de tunelización o fuerza de tunelización.

ε es un factor intrínseco de la construcción del Qubit y hace referencia a la energía que presenta el Qubit según se encuentre en un estado u otro (como niveles de energía); por su parte, λ nos indica qué probabilidad existe de que el electrón pase de un estado a otro (por ejemplo, en un caso extremo donde $\lambda = 0$, el Qubit no cambiaría de estado).

Aunque no sea objeto de estudio en este proyecto, explicaremos brevemente la instalación física sobre la que se realiza la medida (ver [7]), para así entender algunos de los parámetros que aparecerán en nuestro modelo. Aunque el estudio es aplicable a multitud de tipos de Qubits y detectores, pondremos como ejemplo principal un punto cuántico doble ocupado por un solo electrón.

De modo simplificado, este punto cuántico es un electrón que se encuentra entre los dos niveles de energía (cada uno de ellos representando un estado, $|0\rangle$ o $|1\rangle$), entre una barrera de potencial, presentando una probabilidad dada por λ de estar en un estado u otro. Para realizar la medición, hacemos uso de un detector, cuya corriente es sensible a la ubicación del electrón.

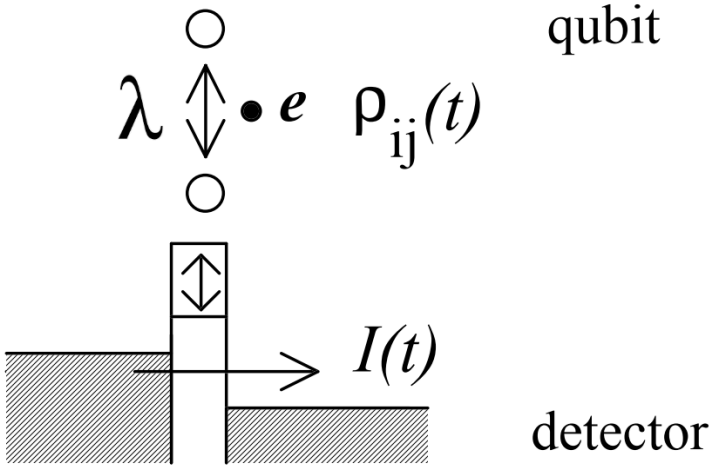

Figura 2.2 Instalación experimental de un punto cuántico doble

Sabiendo que cada una de las posiciones representa un estado de la base clásica y que la corriente de entrada es I_0 y apoyándonos en la Figura 2.2: si el electrón ocupa la posición 1, la corriente promedio a través del detector es I_1, mientras que si está en la posición 2, la corriente promedio es I_2. La diferencia entre ambas, ΔI, determina la respuesta del detector dependiendo de la posición del electrón.

Para realizar esta instalación experimental, es necesario tener en cuenta parámetros como el tiempo de medición, la temperatura, frecuencias del detector, tipo de régimen, etc., pero que no analizaremos en detalle pues no son objeto de este estudio.

Volviendo al modelo analítico y las ecuaciones, la asimetría de energía del Qubit ε_{fb} y la amplitud de tunelización λ_{fb} pueden ser ambas controladas mediante un lazo de control:

$$\lambda_{fb} = g + \Delta\lambda_{fb}, \qquad \varepsilon_{fb} = \varepsilon + \Delta\varepsilon_{fb}; \tag{2.13}$$

No obstante, en nuestro caso asumiremos $\Delta\varepsilon_{fb} = 0$, por lo que únicamente controlaremos la tunelización. El Qubit descrito por este Hamiltoniano presenta, sin interacciones ni control, una frecuencia natural de

$$\Omega = \frac{\sqrt{4g^2 + \varepsilon^2}}{\hbar}, \tag{2.14}$$

que también es llamada frecuencia de Rabi.

La evolución de la matriz densidad ρ del Qubit se puede describir mediante las ecuaciones bayesianas, mostradas a continuación en la forma de Itô:

$$\dot{\rho}_{11} = -\dot{\rho}_{22} = -2\frac{\lambda_{fb}}{\hbar}\mathrm{Im}(\rho_{12}) + \rho_{11}\rho_{22}\frac{2\Delta I}{S_0}\xi(t), \tag{2.15}$$

$$\dot{\rho}_{12} = i\frac{\varepsilon_{fb}}{\hbar}\rho_{12} + i\frac{\lambda_{fb}}{\hbar}(\rho_{11} - \rho_{22}) - (\rho_{11} - \rho_{22})\frac{\Delta I}{S_0}\rho_{12}\xi(t) - \left[\gamma + \frac{(\Delta I)^2}{4S_0}\right]\rho_{12}, \tag{2.16}$$

donde $\xi(t)$ es el ruido blanco de la señal del detector, con $\langle\xi(t)\rangle = 0, \langle\xi(t)\xi(t')\rangle = \sigma^2\delta(t - t')$ y $\sigma = \sqrt{S_0/2}$, siendo S_0 la densidad espectral del ruido de la corriente (nos indica

cómo está distribuido); cuanto menor sea σ, mayor será el ruido. Este ruido $\xi(t)$ proviene de la corriente del detector $I(t)$, la señal de salida, que es

$$I(t) = I_0 + \frac{\Delta I}{2}(2\rho_{11} - 1) + \xi(t). \tag{2.17}$$

Por otro lado, $\Delta I = I_1 - I_2$ es la diferencia entre las corrientes medias I_1 e I_2 correspondientes a los dos estados del Qubit, e $I_0 = (I_1 + I_2)/2$. Estas ecuaciones implican una respuesta débil del detector $\Delta I \ll I_0$, corriente cuasicontinua y un voltaje del detector grande en comparación con la energía del Qubit.

El ratio de desvanecimiento[10] (*dephasing rate*) o velocidad de decoherencia $\gamma = \gamma_d + \gamma_{env}$ presenta la contribución debida al detector no ideal, $\gamma_d = (\eta^{-1} - 1)(\Delta I)^2/4S_0$ (donde $\eta \leq 1$ es la eficiencia cuántica) y la contribución γ_{env} debida a interacción extra con el entorno. Debido a la naturaleza y propiedades de la matriz densidad, que comentamos después de presentar (2.1), debe cumplirse

$$\rho_{11} + \rho_{22} = 1 \text{ y } \rho_{21} = \rho_{12}^*. \tag{2.18}$$

Estas ecuaciones definen la evolución del Qubit a partir de las componentes de su matriz densidad. No obstante, la forma más adecuada para visualizar un estado cuántico de un sistema binario o *two-level system* viene dada por su representación en una esfera de Bloch.

La esfera de Bloch [8] es una representación geométrica de los estados de un sistema binario, como el Qubit, como puntos en la superficie de una esfera unitaria. Muchas operaciones en Qubits individuales pueden describirse visualmente dentro de la esfera de Bloch.

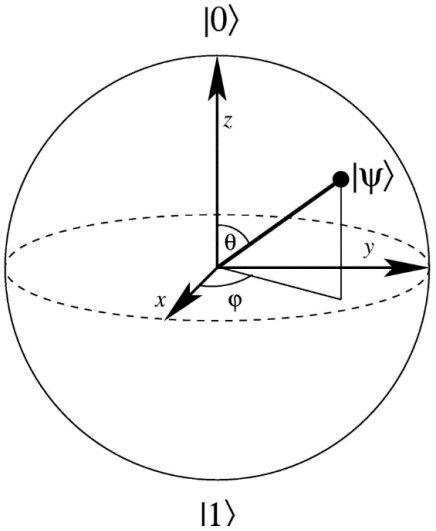

Figura 2.3 Esfera de Bloch [9]

Un estado arbitrario de un Qubit puede escribirse como

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle\right), \tag{2.19}$$

donde θ, ϕ y γ son números reales. Los números $0 \leq \theta \leq \pi$ y $0 \leq \phi \leq 2\pi$ definen un punto en una esfera tridimensional unitaria, que es la esfera de Bloch propiamente dicha. Todos los

[10] Mecanismo indeseado que recupera el comportamiento clásico en un sistema cuántico, midiendo como disminuye la coherencia a lo largo del tiempo.

estados del Qubit con valores arbitrarios de γ están representados por el mismo punto en la esfera de Bloch; es decir, el factor $e^{i\gamma}$ no tiene efectos observables y, por tanto, se puede escribir

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle. \tag{2.20}$$

La esfera de Bloch no es más que una generalización de la representación de números complejos como puntos en el círculo unidad del plano complejo.

Como comentamos al final de la sección 1.2, cualquier matriz A bidimensional de coeficientes complejos es una combinación lineal real de la base que definimos a partir de (1.38) y la matriz identidad:

$$\mathcal{B} = \{\mathbf{1}, \sigma_x, \sigma_y, \sigma_z\}. \tag{2.21}$$

Siendo la combinación lineal la de la ecuación (1.46), que volvemos a mostrar a continuación

$$A = a_0\mathbf{1} + \mathbf{a}\cdot\sigma = a_0 + \mathbf{a}\cdot\sigma, \tag{2.22}$$

donde a_0 y los componentes del vector $\mathbf{a}$ son todos números reales.

Por tanto, sabiendo que la matriz densidad ρ de un sistema de dos niveles es una matriz hermitiana de 2×2, esta puede ser definida como un "vector" en el espacio vectorial $N^2 = 4$ dimensional de $\mathbb{R}$ [10]. Este espacio vectorial $\mathcal{R}_{\mathcal{H}}$ está formado por la base ortogonal que definimos a partir de (2.21) y su ortogonalidad está basada en el producto interno de Hilbert-Schmidt: $\langle A, B\rangle_{HS} = \mathrm{tr}(AB)$, siendo así A y B son dos operadores de Hilbert-Schmidt. Puesto que

$$\langle \mathbf{1}, \mathbf{1}\rangle = \langle \sigma_j, \sigma_j\rangle = 2 \qquad (j = x, y, z), \tag{2.23}$$

nuestra base $\mathcal{B}$ no está normalizada. En vez de realizar la normalización de estos vectores, se suele incluir un factor $1/2$ en la descomposición de ρ:

$$\rho = \frac{1}{2}\left[p_1\mathbf{1} + p_x\sigma_x + p_y\sigma_x + p_z\sigma_x\right] = \frac{1}{2}[p_1\mathbf{1} + \mathbf{p}\cdot\sigma], \tag{2.24}$$

equivalente a (2.22) pero con el factor $1/2$.

La normalización de la matriz densidad nos permite comprobar que

$$1 = \mathrm{tr}(\rho) = \langle \mathbf{1}, \rho\rangle = \frac{1}{2}p_1\langle \mathbf{1}, \mathbf{1}\rangle = \frac{1}{2}p_1 2 = p_1. \tag{2.25}$$

Como resultado, la matriz densidad de un sistema binario puede ser siempre expresado como un vector real de tres componentes $\mathbf{p} = (p_x, p_y, p_z)$. A estos vectores los denominaremos vectores de Bloch y deben presentar un radio menor o igual que el de la esfera de radio unidad mencionada anteriormente, la esfera de Bloch.

Teniendo esto en cuenta, podemos expresar, empleando las ecuaciones del estado del Qubit (2.15) y (2.16), las componentes del vector de Bloch en función de las componentes de la matriz densidad y realizar las simulaciones con estas ecuaciones; es decir, calcularemos la evolución de los vectores de Bloch a lo largo del tiempo.

Podemos desarrollar entonces la expresión (2.24), y obtener las relaciones entre las componentes de ρ y las de $\mathbf{p}$. Centrándonos en el segundo término del paréntesis

$$\mathbf{p}\cdot\sigma = p_x\sigma_x + p_y\sigma_y + p_z\sigma_z = p_x\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} + p_y\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix} + p_z\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$$
$$= \begin{pmatrix}p_z & p_x - ip_y\\ p_x + ip_y & -p_z\end{pmatrix}, \tag{2.26}$$

podemos sustituirlo en (2.24):

$$\rho = \frac{1}{2}[\mathbf{1} + \mathbf{p}\cdot\sigma] = \frac{1}{2}\begin{pmatrix}p_z & p_x - ip_y\\ p_x + ip_y & -p_z\end{pmatrix} = \begin{pmatrix}\rho_{11} & \rho_{12}\\ \rho_{21} & \rho_{22}\end{pmatrix}. \tag{2.27}$$

Igualando los términos de ambas matrices llegamos a las relaciones que nos permiten expresar $\rho = \rho(\mathbf{p})$:

$$\begin{aligned}\rho_{11} &= \frac{1}{2}(1 + p_z), \qquad \rho_{12} = \frac{1}{2}(p_x - ip_y),\\ \rho_{22} &= \frac{1}{2}(1 - p_z), \qquad \rho_{21} = \frac{1}{2}(p_x + ip_y).\end{aligned} \tag{2.28}$$

Para obtener la evolución de los componentes del vector de Bloch, necesitamos expresarlas, en un principio, en función de las componentes de la matriz densidad

$$\begin{aligned}p_x &= \rho_{12} + \rho_{21},\\ p_y &= i(\rho_{12} - \rho_{21}),\\ p_z &= \rho_{11} - \rho_{22},\end{aligned} \tag{2.29}$$

y a continuación derivarlas respecto al tiempo:

$$\begin{aligned}\dot{p}_x &= \dot{\rho}_{12} + \dot{\rho}_{21} = \dot{\rho}_{12} + \dot{\rho}_{12}^* = 2\mathrm{Re}\dot{\rho}_{12},\\ \dot{p}_y &= i(\dot{\rho}_{12} - \dot{\rho}_{21}) = i(\dot{\rho}_{12} - \dot{\rho}_{21}^*) = i(2\mathrm{Im}\dot{\rho}_{12}\, i) = -2\mathrm{Im}\dot{\rho}_{12},\\ \dot{p}_z &= \dot{\rho}_{11} - \dot{\rho}_{22} = 2\dot{\rho}_{11},\end{aligned} \tag{2.30}$$

donde hemos aplicado las relaciones existentes para la suma y diferencia de un número complejo y su conjugado para $\dot{p}_x$ y $\dot{p}_y$. Tras desarrollar las ecuaciones que acabamos de escribir, llegamos finalmente a

$$\dot{p}_x = \frac{\varepsilon_{fb}}{\hbar}p_y - p_z\frac{\Delta I}{S_0}p_x\xi(t) - \left[\gamma + \frac{(\Delta I)^2}{4S_0}\right]p_x, \tag{2.31}$$

$$\dot{p}_y = -\left[\frac{\varepsilon_{fb}}{\hbar}p_x + 2\frac{\lambda_{fb}}{\hbar}p_z + p_z\frac{\Delta I}{S_0}p_y\xi(t) + \left[\gamma + \frac{(\Delta I)^2}{4S_0}\right]p_y\right], \tag{2.32}$$

$$\dot{p}_z = 2\frac{\lambda_{fb}}{\hbar}p_y + (1 - p_z^2)\frac{\Delta I}{S_0}\xi(t). \tag{2.33}$$

Este SDE se puede expresar matricialmente como

$$\begin{cases}d\mathbf{p}(t) = A\big(\mathbf{p}(t)\big)dt + B\big(\mathbf{p}(t)\big)dW(t) \quad (t > 0)\\ \mathbf{p}(0) = \mathbf{p}_0\end{cases}, \tag{2.34}$$

donde se ha empleado la relación (2.7) nuevamente para indicar el ruido a través del término $dW(t)$, siendo las matrices las siguientes:

$$\mathbf{p}(t) = \begin{pmatrix}p_x\\ p_y\\ p_z\end{pmatrix}, \qquad p_0 = \begin{pmatrix}p_{x_0}\\ p_{y_0}\\ p_{z_0}\end{pmatrix}, \tag{2.35}$$

$$A(\mathbf{p}(t)) = \begin{pmatrix} -\left[\gamma + \frac{(\Delta I)^2}{4S_0}\right] p_x & \frac{\varepsilon_{fb}}{\hbar} p_y & 0 \\ -\frac{\varepsilon_{fb}}{\hbar} p_x & -\left[\gamma + \frac{(\Delta I)^2}{4S_0}\right] p_y & -2\frac{\lambda_{fb}}{\hbar} p_z \\ 0 & 2\frac{\lambda_{fb}}{\hbar} p_y & 0 \end{pmatrix}, \qquad (2.36)$$

y

$$B(\mathbf{p}(t)) = \begin{pmatrix} -p_z \frac{\Delta I}{S_0} p_x & 0 & 0 \\ 0 & -p_z \frac{\Delta I}{S_0} p_y & 0 \\ 0 & 0 & (1 - p_z^2)\frac{\Delta I}{S_0} \end{pmatrix}. \qquad (2.37)$$

De este modo podemos diferenciar de un modo más sencillo los términos que introducen ruido. Además, podemos comprobar que este ruido es diagonal, siendo este uno de los requisitos del software de simulación, como se explica en el Apéndice A.

Para describir la evolución del Qubit podemos emplear indistintamente tanto las ecuaciones en función de las componentes de su matriz densidad (2.15)-(2.16), como las ecuaciones en base al componente del vector de Bloch asociado (2.31)-(2.33). Además, gracias a las relaciones (2.28) y (2.29) podemos transformar unas variables en otras de un modo sencillo.

Una vez definida la trayectoria del Qubit, seguimos presentando algunos parámetros y características del sistema con el que estamos trabajando.

Se puede caracterizar el acoplamiento entre el Qubit y el detector mediante el parámetro adimensional $C = \hbar(\Delta I)^2/(4S_0 g)$, donde se considera acoplamiento débil cuando $C \lesssim 1$, que será el que empleemos nosotros para la mayoría de las simulaciones.

También, en un principio, consideraremos el control "Bayesiano", que requiere de un procesador que resuelve las ecuaciones (2.15) y (2.16) en tiempo real. También despreciaremos el efecto del ancho de banda de la línea que transporta la señal del detector así como el retraso de la señal en el lazo de control. Como consecuencia de esto, asumiremos que el valor monitorizado o medido $\rho^m(t)$ de la matriz densidad del Qubit coincide con el valor actual $\rho(t)$.

Para el control de realimentación compararemos, como ya se ha comentado y se aprecia en la Figura 2.1, la evolución monitorizada del Qubit con una evolución deseada. La diferencia entre ambas es usada como control para ir reduciendo el desfase entre estas. El objetivo es mantener las oscilaciones deseadas mediante un largo periodo de tiempo. La evolución deseada es

$$\rho_{11}^d = \frac{1 + \cos \Omega t}{2}, \qquad \rho_{12}^d = i\frac{\sin \Omega t}{2}. \qquad (2.38)$$

En la bibliografía podemos encontrar un control para un Qubit de estado sólido dado por

$$\Delta\lambda_{fb} = -Fg\Delta\phi_m, \qquad (2.39)$$

$$\Delta\phi_m = \phi_m(t)(\mathrm{mod}\ 2\pi) - \Omega t\ (\mathrm{mod}\ 2\pi)^{11}, \tag{2.40}$$

$$\phi_m(t) = \mathrm{atan}\frac{2\mathrm{Im}\rho_{12}^m}{\rho_{11}^m - \rho_{22}^m} + \frac{\pi}{2}[1 - \mathrm{sgn}(\rho_{11}^m - \rho_{22}^m)], \tag{2.41}$$

donde ϕ_m es el valor de la fase medido, la diferencia de fase $\Delta\phi_m$ es definida como $|\Delta\phi_m| \leq \pi$ y F es un factor adimensional de la fuerza del feedback (el segundo término en (2.41) permite una continuidad de fase en 2π). Definido de este modo, el controlador disminuye la diferencia de fase (realimentación negativa): si la fase $\phi_m(t)$ está adelantada al valor deseado, $\Delta\lambda_{fb}$ es negativo, lo que ralentiza las oscilaciones del Qubit; si $\phi_m(t)$ está atrasado respecto al valor deseado, la frecuencia de oscilación aumenta. Por tanto, podemos caracterizar la eficiencia de la retroalimentación mediante la diferencia de fases $\Delta\phi_m$. Podemos expresarlo como porcentaje de error si tenemos en cuenta que el ángulo de máxima diferencia entre ellos es π.

[11] En el documento en el que está basado este proyecto [6], se define $\Delta\phi_m = \phi_m(t) - \Omega t\ (\mathrm{mod}\ 2\pi)$, pero este provoca grandes errores en la simulación. Con el nuevo definido, el control se realiza correctamente para, al menos, un período.

Capítulo 3

Resultados y simulaciones

En esta sección estableceremos los valores numéricos necesarios para realizar la simulación de la dinámica del Qubit, así como el control de este. A través del capítulo, se representarán principalmente las gráficas de las componentes del vector de Bloch y del error existente durante el control. También analizaremos la influencia de diferentes parámetros definitorios del modelo del Qubit. En referencia al control, estudiaremos no solo casos para frecuencia natural con diferentes condiciones iniciales, sino también para velocidades diferentes, definiendo las limitaciones que estas modificaciones presentan.

3.1. Resolución

La dinámica del Qubit, junto con la influencia de la medición, viene dada por un sistema de Ecuaciones Diferenciales Estocásticas no lineal. Es por este motivo por el que para llevar a cabo las simulaciones, tanto con control como sin él, emplearemos métodos numéricos. La información detallada sobre el método empleado, así como el software y el código, aparece recogida en el Apéndice A.

Teniendo en cuenta las ecuaciones a resolver (2.15) y (2.16)[12], tomamos los siguientes valores para los diferentes parámetros (ver [11]): $\hbar/g = 1.6 \cdot 10^2 dt$, $S_0/\Delta I^2 = 2.5 \cdot 10^5 dt$, $\hbar/\varepsilon = 10^3 dt$, $\gamma = 0$ y $\sigma = 1$. Estableceremos el paso $dt = 0.01$ y, por lo general, el número de simulaciones será igual a 300, quedando todo determinado a excepción de elegir las condiciones iniciales, limitadas por las relaciones de (2.18) y que expresaremos en un vector de la forma $\rho_0 = \left(\rho_{11_0}, \rho_{12_0}, \rho_{22_0}\right)$ o $\mathbf{p_0} = \left(p_{x_0}, p_{y_0}, p_{z_0}\right)$ según el código que estemos empleando. No obstante, las gráficas que mostraremos serán de las componentes del vector de Bloch pues resultan más intuitivas para su estudio. El tiempo de simulación lo modificaremos según sea necesario. Cuando queramos observar la influencia de un parámetro determinado o realizar pruebas de cualquier otro tipo, especificaremos qué parámetros cambian.

3.2. Simulación de un Qubit sin control

Comenzamos con la simulación de un Qubit, girando a su frecuencia natural. Estableceremos como condiciones iniciales $\overrightarrow{\rho_0} = (0, 0, 1)$, puesto que para el control viene dado en términos de ρ, equivalente a $\mathbf{p_0} = (0, 0, -1)$. En esta simulación, también hemos añadido las ecuaciones y gráficas del control, centrándonos en $\Delta\phi_m$, para poder comprobar su funcionamiento[13].

[12] En un principio emplearemos las ecuaciones que describen la dinámica del Qubit en términos de ρ.

[13] Sobre las gráficas, aunque se pueda emplear el comando `saveas(fig,filename)` para exportarlas directamente desde MATLAB, la calidad del mismo se ve afectada al emplearlo en comparación con realizar directamente capturas de pantalla, por lo que usaremos esta segunda opción.

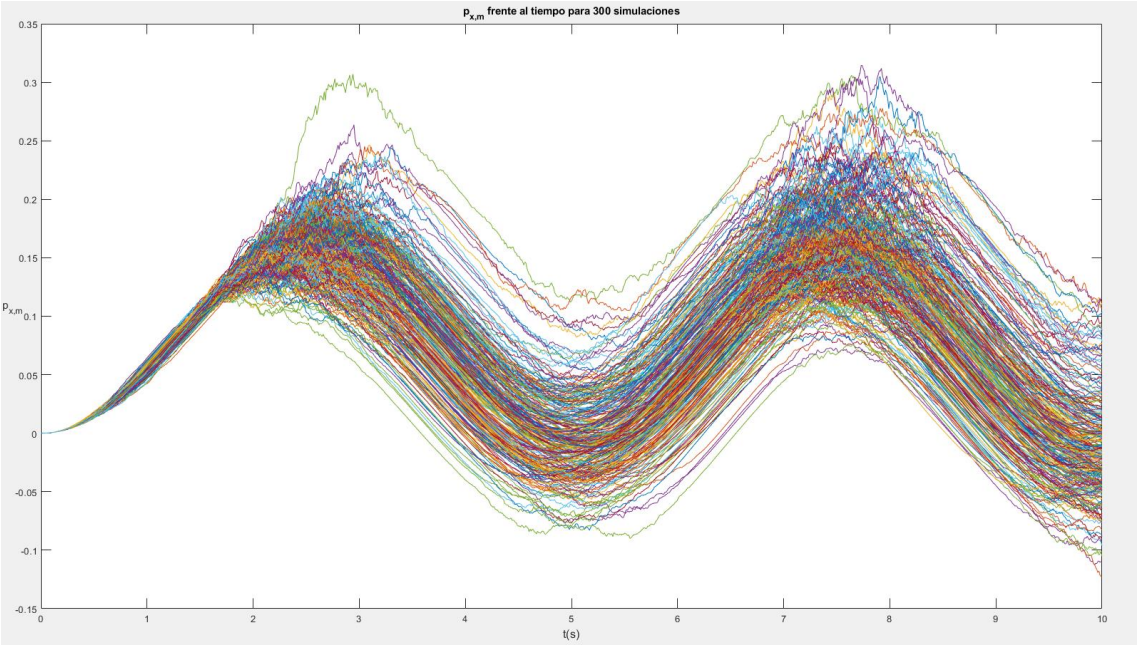

Figura 3.1 Simulación de un Qubit sin control. p_x^m frente al tiempo para 300 simulaciones

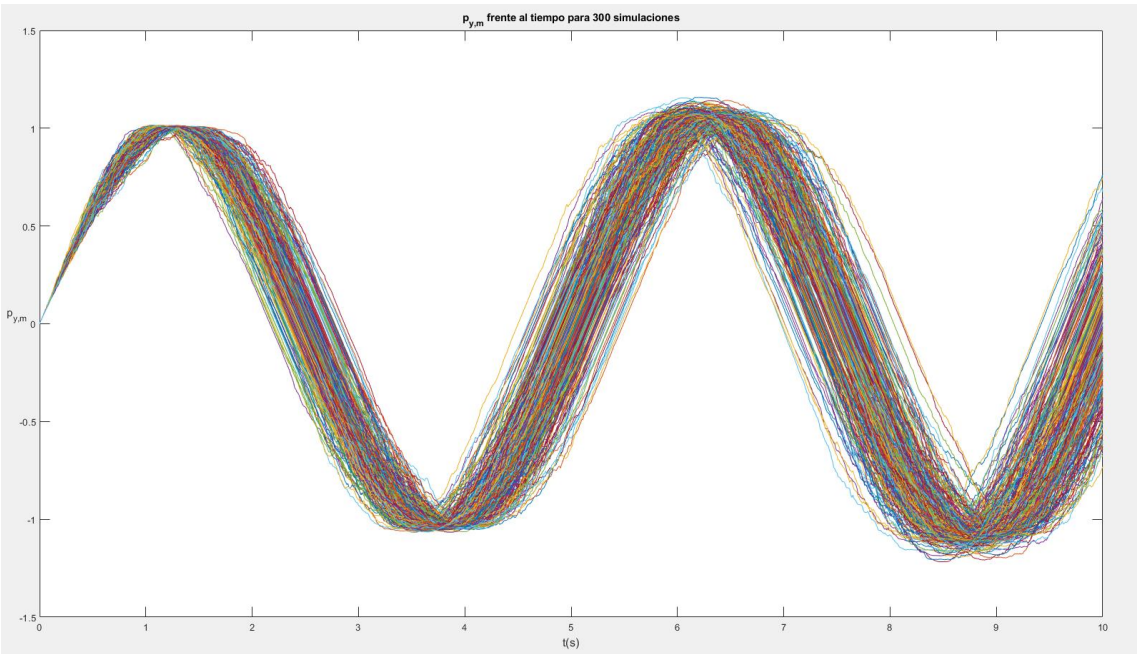

Figura 3.2 Simulación de un Qubit sin control. p_y^m frente al tiempo para 300 simulaciones

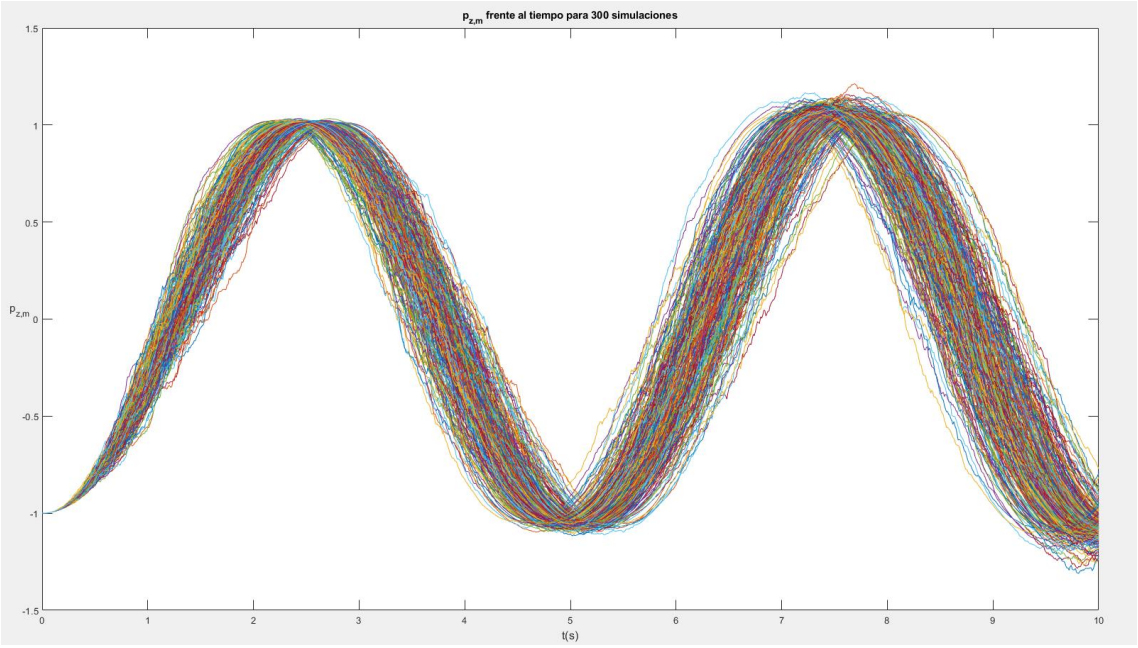

Figura 3.3 Simulación de un Qubit sin control. p_z^m frente al tiempo para 300 simulaciones

Podemos comprobar cómo el código simula perfectamente todas las trayectorias indicadas, siguiendo todas una evolución similar con la única diferencia aportada por el ruido. No obstante, estas tres gráficas simplemente son para demostrar que el algoritmo funciona correctamente. Para poder estudiar con mayor claridad el Qubit, emplearemos valores medios. Además, representaremos, también en valor medio, la intensidad del detector, que es el aporte de ruido.

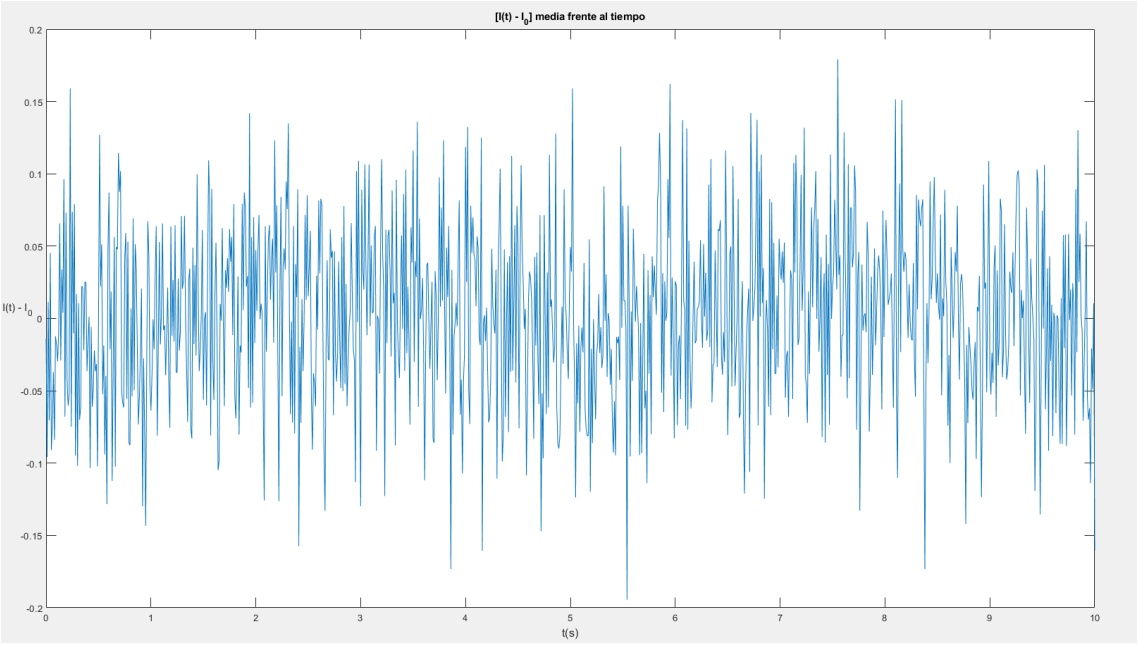

Figura 3.4 Simulación de un Qubit sin control. $I(t) - I_0$ media frente al tiempo

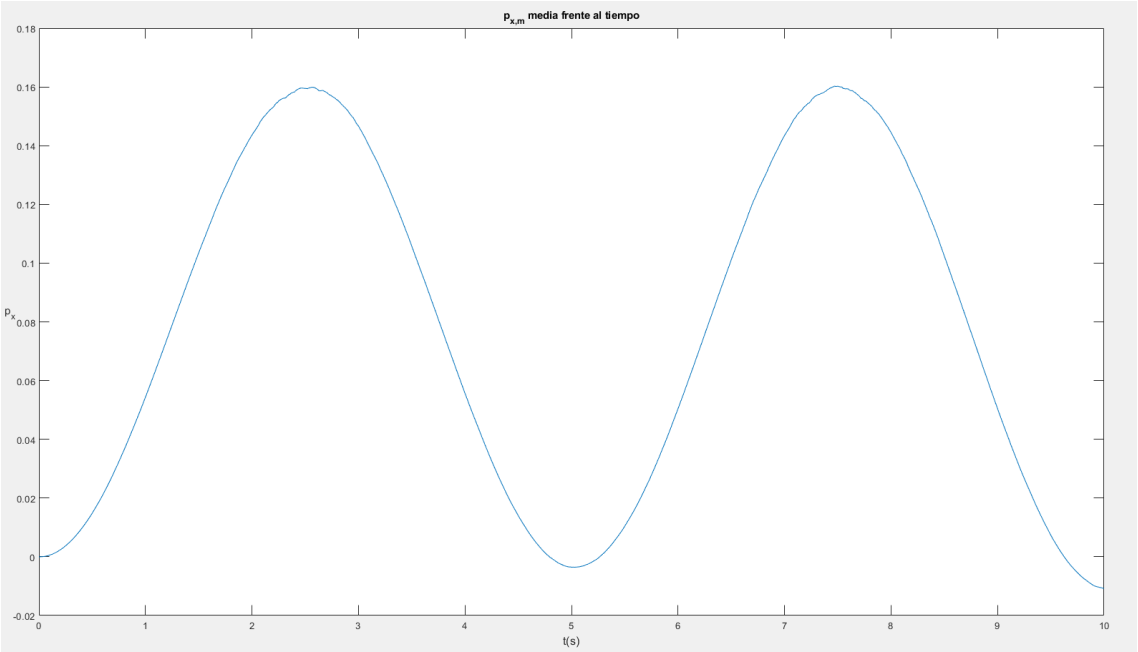

Figura 3.5 Simulación de un Qubit sin control. p_x^m media frente al tiempo

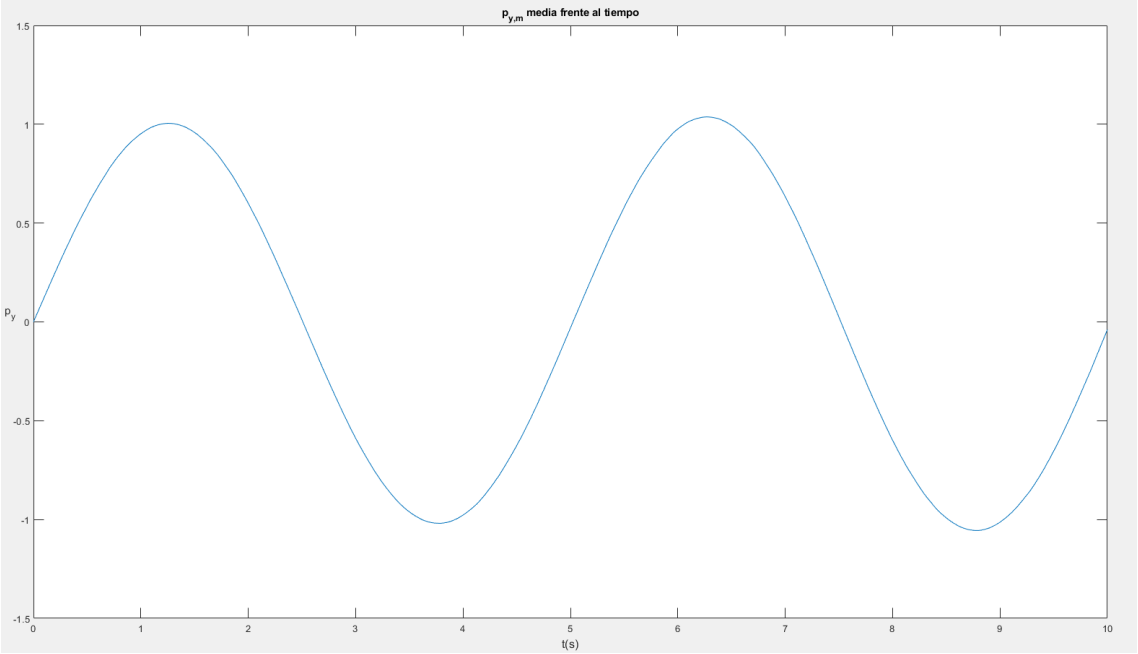

Figura 3.6 Simulación de un Qubit sin control. p_y^m media frente al tiempo

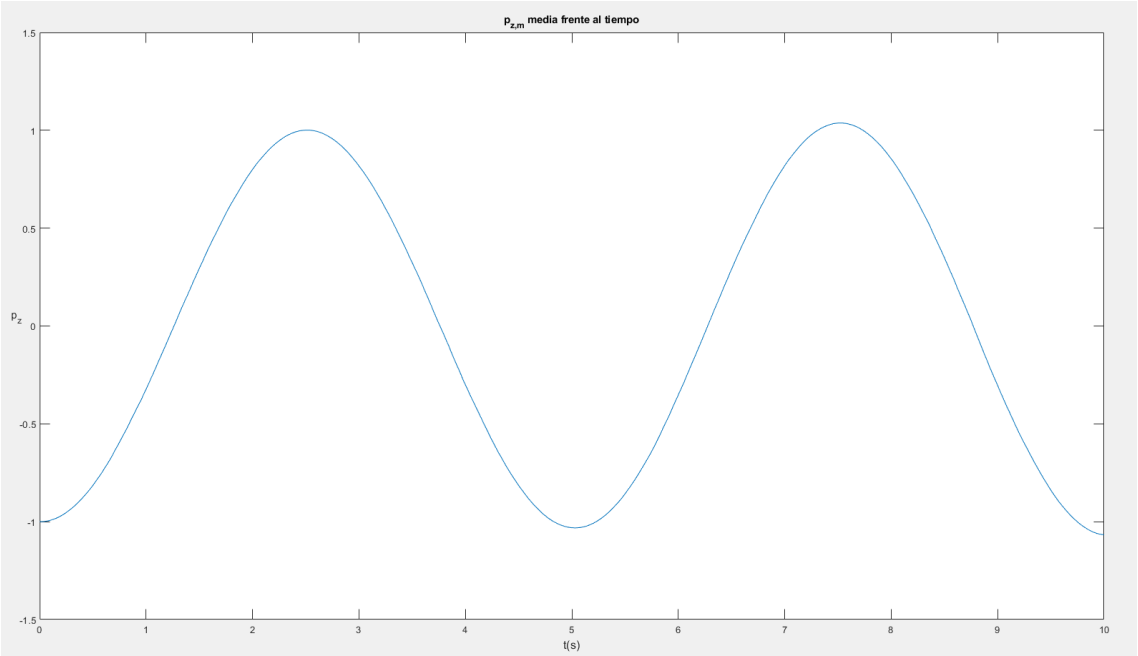

Figura 3.7 Simulación de un Qubit sin control. p_z^m media frente al tiempo

La intensidad, el ruido debido al detector, depende exclusivamente de este y no del control que se realice, por lo que tanto sin control como con él, mostrará un aspecto similar. En estas imágenes se observa claramente la evolución natural del Qubit. Es necesario mencionar la pequeña oscilación que presenta p_x^m, que en un caso ideal debería ser cero. Esto se debe a la asimetría del Qubit, que obliga a este a "desviarse". Más adelante estudiaremos con mayor profundidad este parámetro. Nos centraremos ahora en el funcionamiento de las ecuaciones y el control.

Aunque en este apartado no estamos realizando control, podemos calcular los parámetros de este a modo de comprobación, estableciendo una comparación con los valores que definimos anteriormente como "deseados" y que nombraremos en este caso, para evitar confusiones, como valores "modelo", $\mathbf{p}_{mod}$. Es necesario reiterar que no se está realizando ningún tipo de control, pues no se está aplicando la retroalimentación y, por tanto, no se está modificando ningún parámetro. No obstante, eso no nos impide calcular los parámetros del control y ver si se comportan como es de esperar.

Empleando (2.38) como trayectoria modelo, si calculamos $\Delta\phi_m$ y lo representamos en una gráfica, podremos comprobar que será durante toda la simulación igual a π, puesto que hemos inicializado el Qubit con este desfase inicial y no se está realizando ningún control.

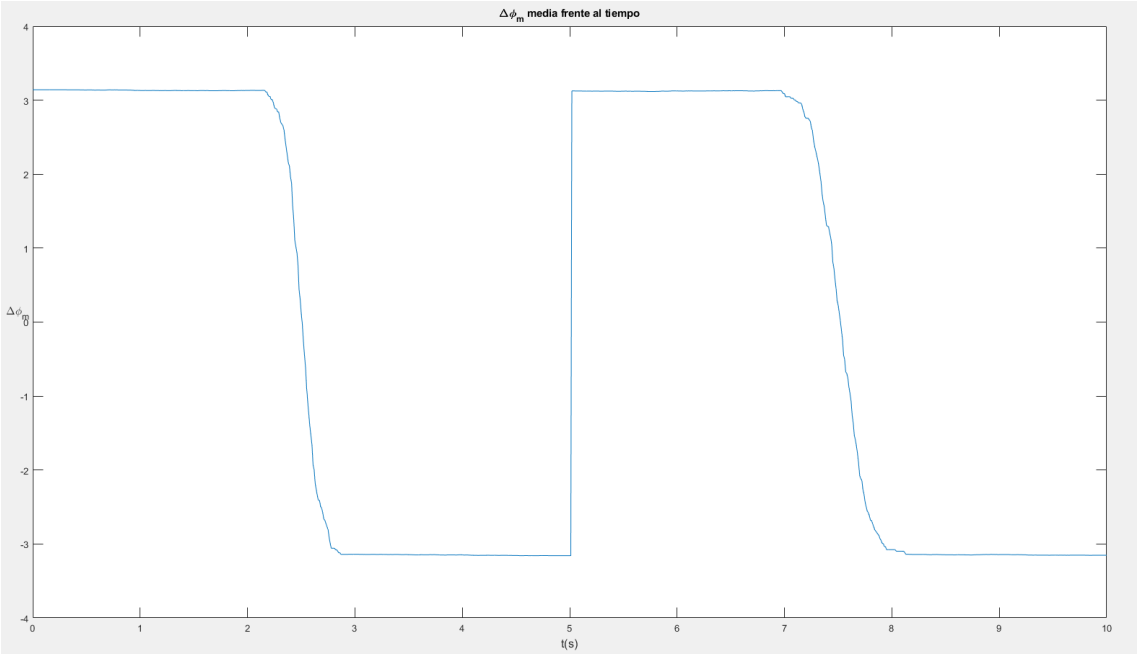

Figura 3.8 Simulación de un Qubit sin control. $\Delta\phi_m$ media frente al tiempo entre dos Qubits con diferentes condiciones iniciales

Efectivamente, siempre presenta una diferencia de fase de modulo π. El cambio de signo simplemente se debe a cómo mide la diferencia de ángulos según el cuadrante en el que se encuentre (recordemos que ϕ_m (2.41) está definido mediante la arcotangente de una división y puede entrar en conflicto con los signos).

Representaremos el mismo caso pero cambiando las condiciones iniciales para que coincidan con la trayectoria modelo, de modo que $\overrightarrow{\rho_0} = (1,0,0)$ o $\mathbf{p_0} = (0,0,1)$. También representaremos los vectores monitorizado y modelo, para realizar de un modo más efectivo las comparaciones de los parámetros del control.

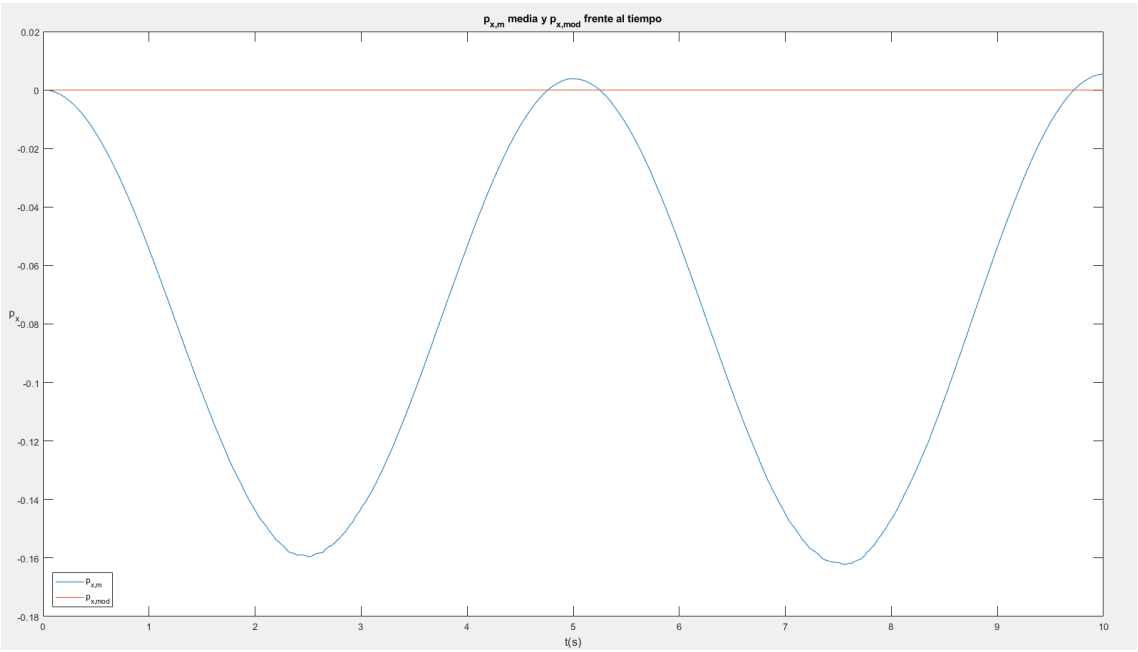

Figura 3.9 Simulación de un Qubit sin control. p_x^m media y p_x^{mod} frente al tiempo, con mismas condiciones iniciales

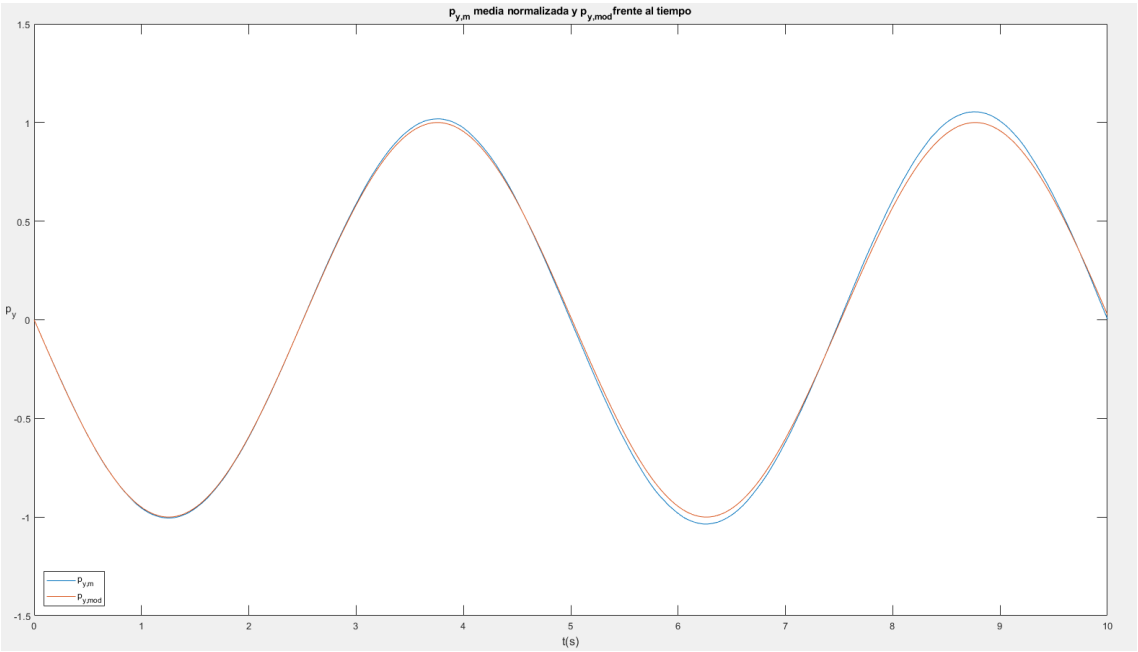

Figura 3.10 Simulación de un Qubit sin control. p_y^m media y p_y^{mod} frente al tiempo, con mismas condiciones iniciales

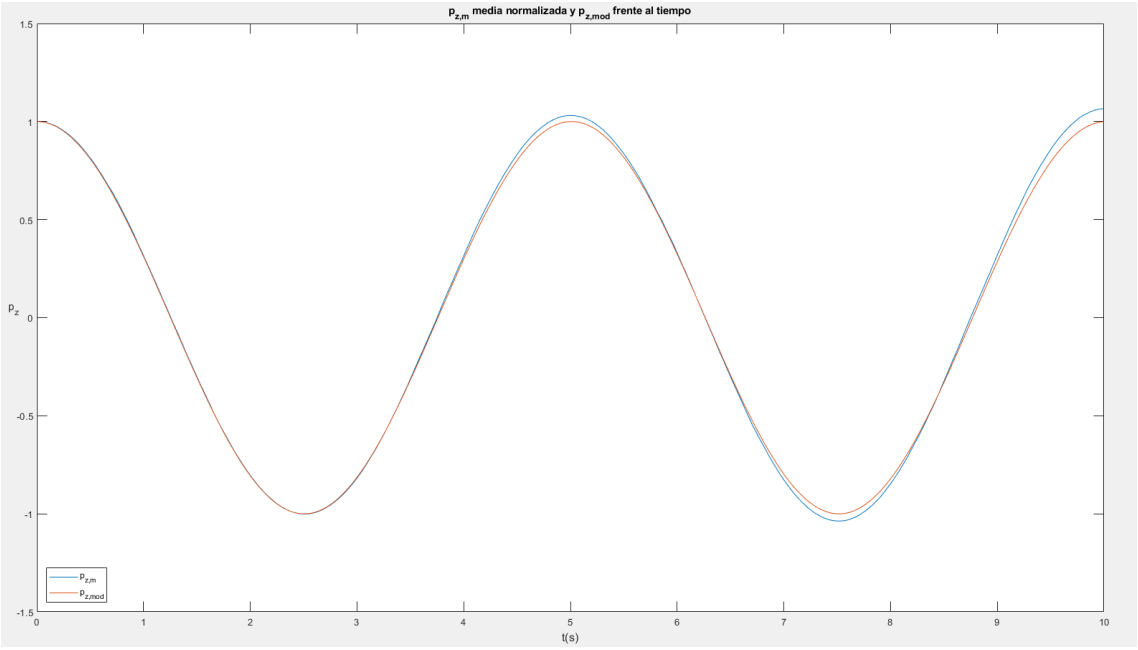

Figura 3.11 Simulación de un Qubit sin control. p_z^m media y p_z^{mod} frente al tiempo, con mismas condiciones iniciales

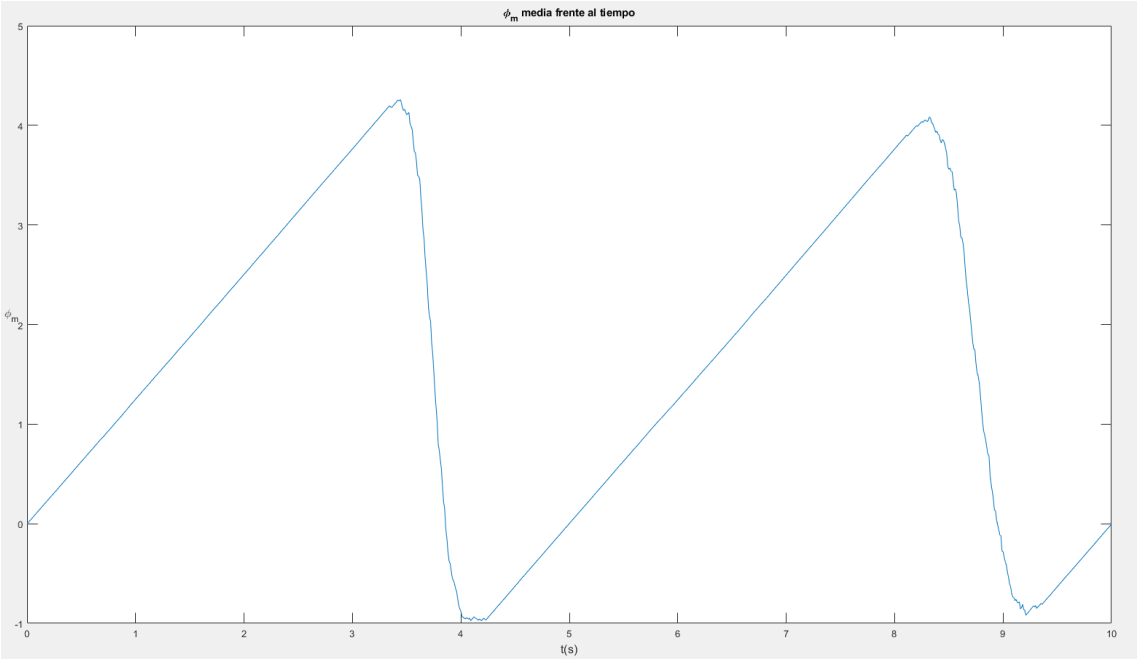

Figura 3.12 Simulación de un Qubit sin control. ϕ_m media frente al tiempo entre dos Qubits con mismas condiciones iniciales

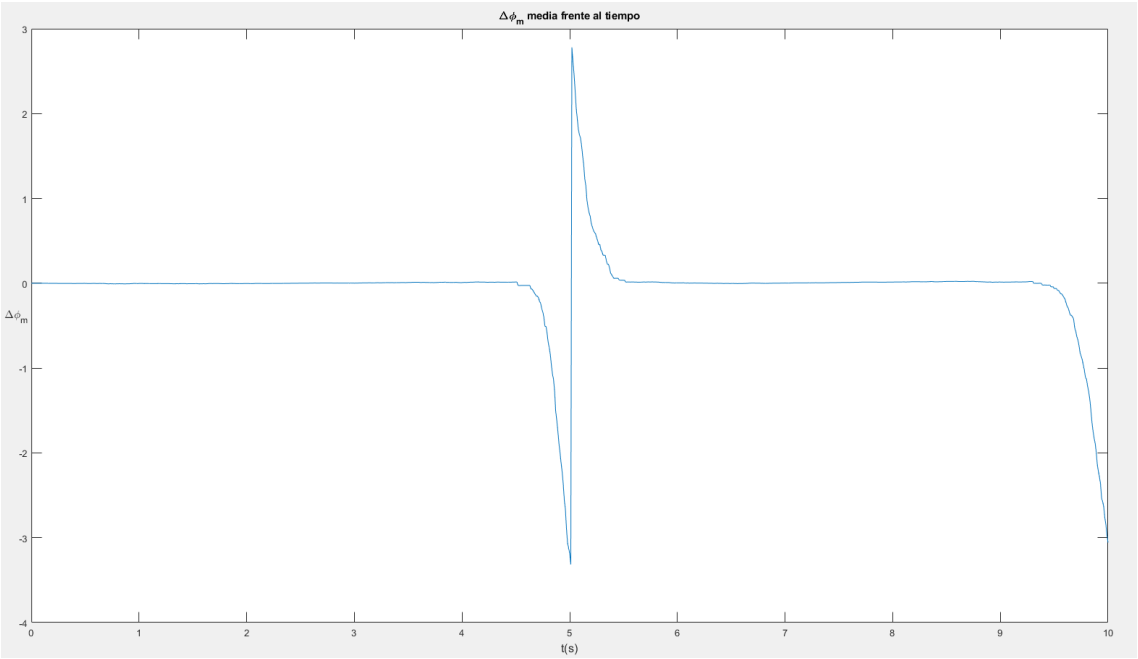

Figura 3.13 Simulación de un Qubit sin control. $\Delta\phi_m$ media frente al tiempo entre dos Qubits con mismas condiciones iniciales

Podemos observar cómo efectivamente el Qubit monitorizado sigue, aunque totalmente desacoplado puesto que no existe control, la misma evolución que la trayectoria modelo. Por otro lado, respecto a los parámetros de control, el valor de ϕ_m abarca el rango de valores desde menos $\pi/2$ hasta $\pi + \pi/2$, por lo que se comporta como es de esperar. Por su parte, $\Delta\phi_m$ se mantiene prácticamente nulo a excepción de cuando acaba el ciclo, que interpreta que existe un desfase de 2π en vez de 0. Habrá que vigilar este parámetro porque tal vez presente problemas de cara al control.

Por último, en las representaciones de p_y y p_z, se puede apreciar que los valores monitorizados a lo largo del tiempo superan los valores modelo en amplitud. Por la propia definición de estos componentes y de la esfera de Bloch es imposible que esto ocurra. Así, normalizaremos los valores para apreciar mejor el control de fases, como se muestra a continuación.

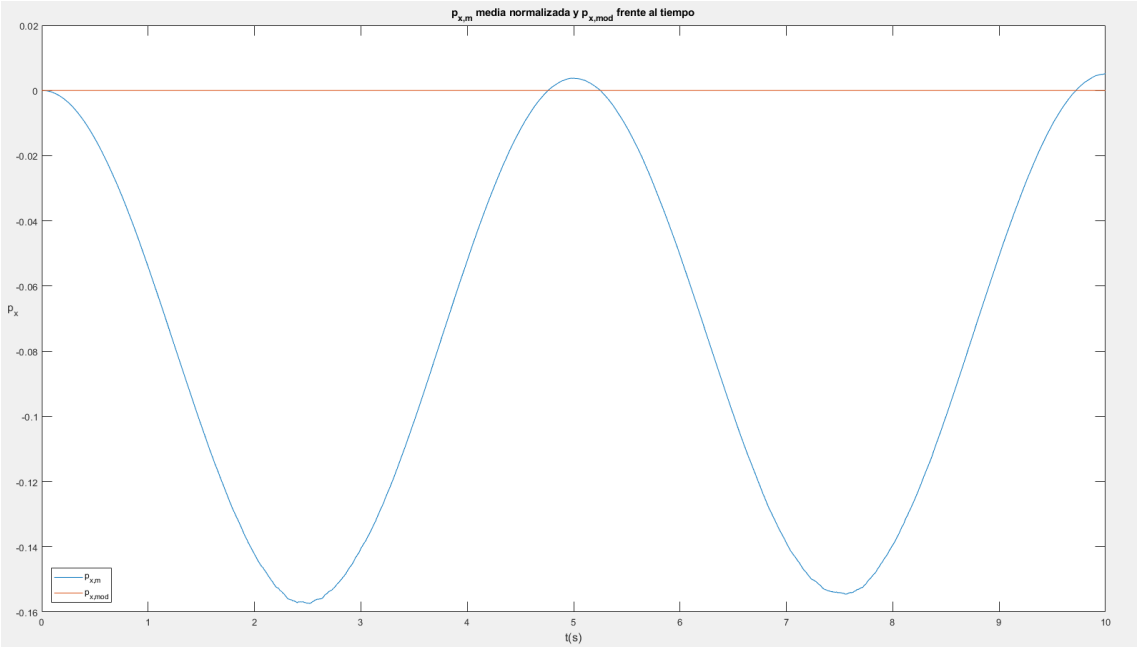

Figura 3.14 Simulación de un Qubit sin control. p_x^m media normalizada y p_x^{mod} frente al tiempo, con mismas condiciones iniciales

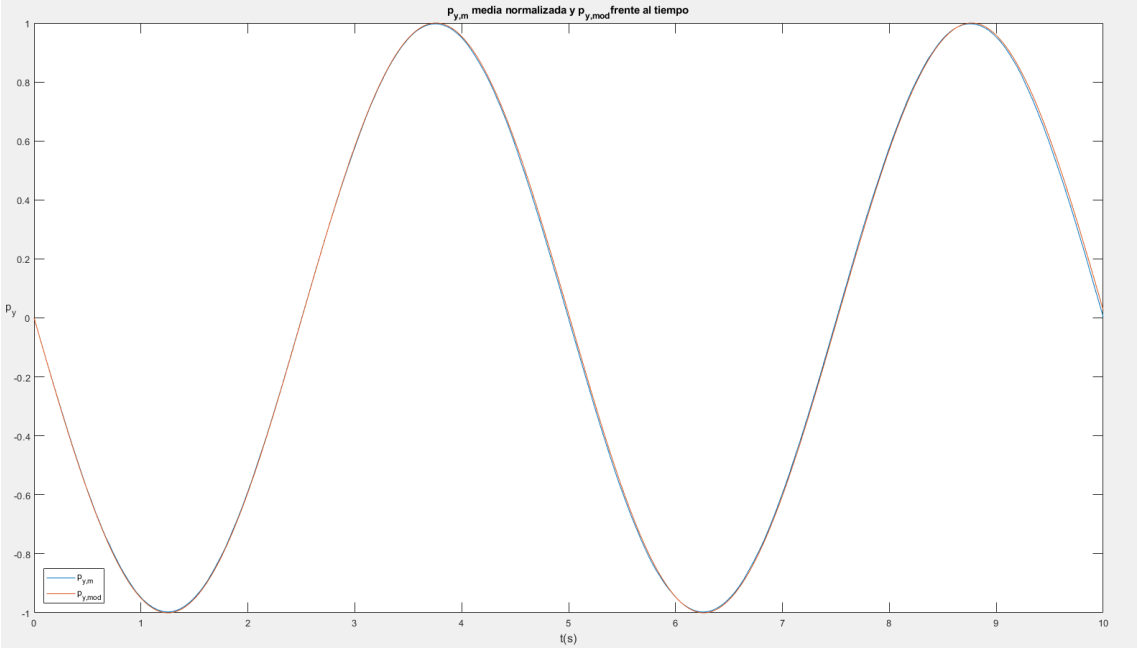

Figura 3.15 Simulación de un Qubit sin control. p_y^m media normalizada y p_y^{mod} frente al tiempo, con mismas condiciones iniciales

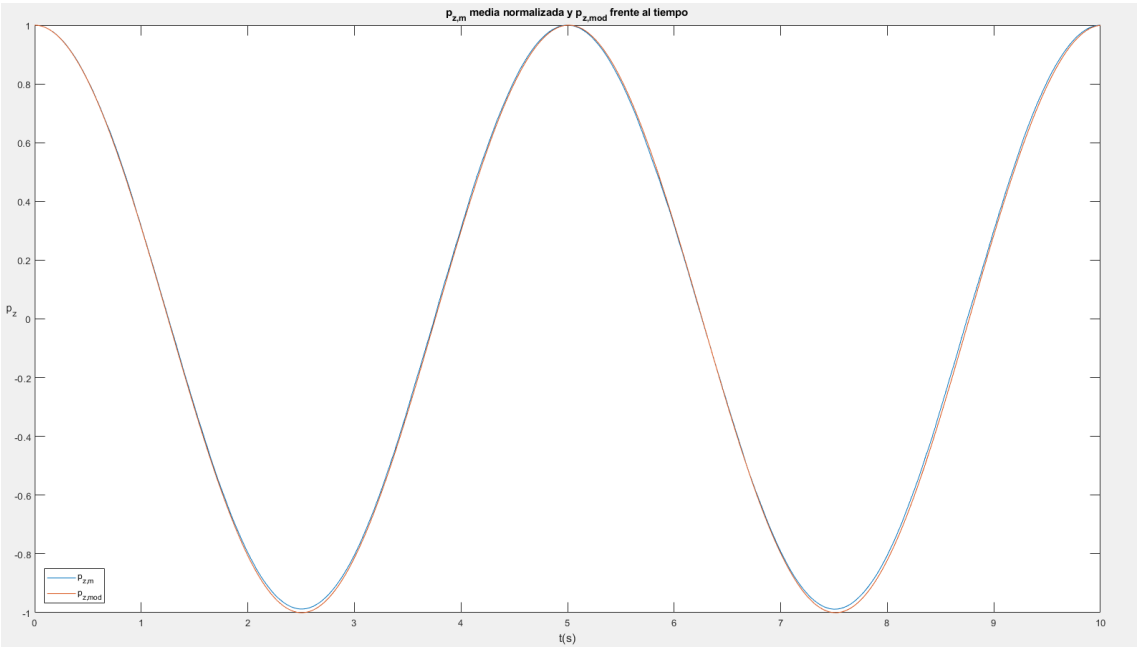

Figura 3.16 Simulación de un Qubit sin control. p_z^m media normalizada y p_z^{mod} frente al tiempo, con mismas condiciones iniciales

Se aprecia que la simulación es como deseamos. Por lo tanto, a no ser que se indique lo contrario, las siguientes simulaciones y representaciones se realizarán empleando valores medios y normalizados.

3.3. Simulación de un Qubit controlado

Una vez analizada la evolución natural de un Qubit, podemos controlarlo para el caso de que se inicie en condiciones iniciales diferentes o variemos la frecuencia de este. Para ello, haremos uso de las ecuaciones de control (2.39)-(2.41). En un principio, estableceremos $F = 3$ y $\overrightarrow{\rho_0} = (0, 0, 1)$.

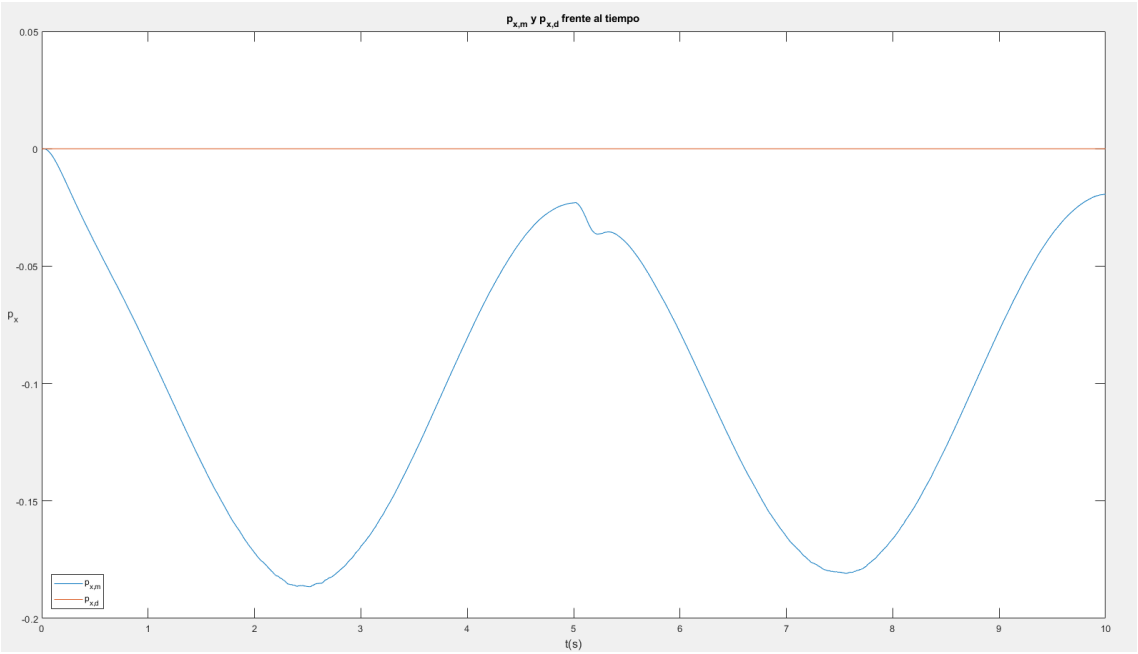

Figura 3.17 Simulación de un Qubit controlado. p_x^m y p_x^d frente al tiempo

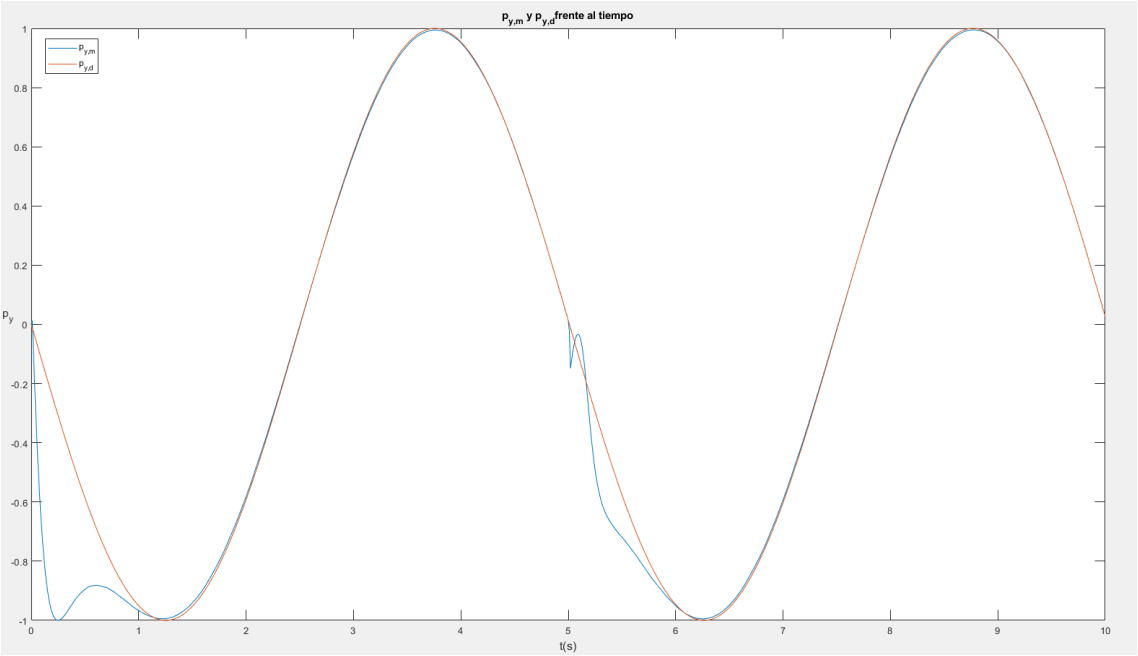

Figura 3.18. Simulación de un Qubit controlado. p_y^m y p_y^d frente al tiempo

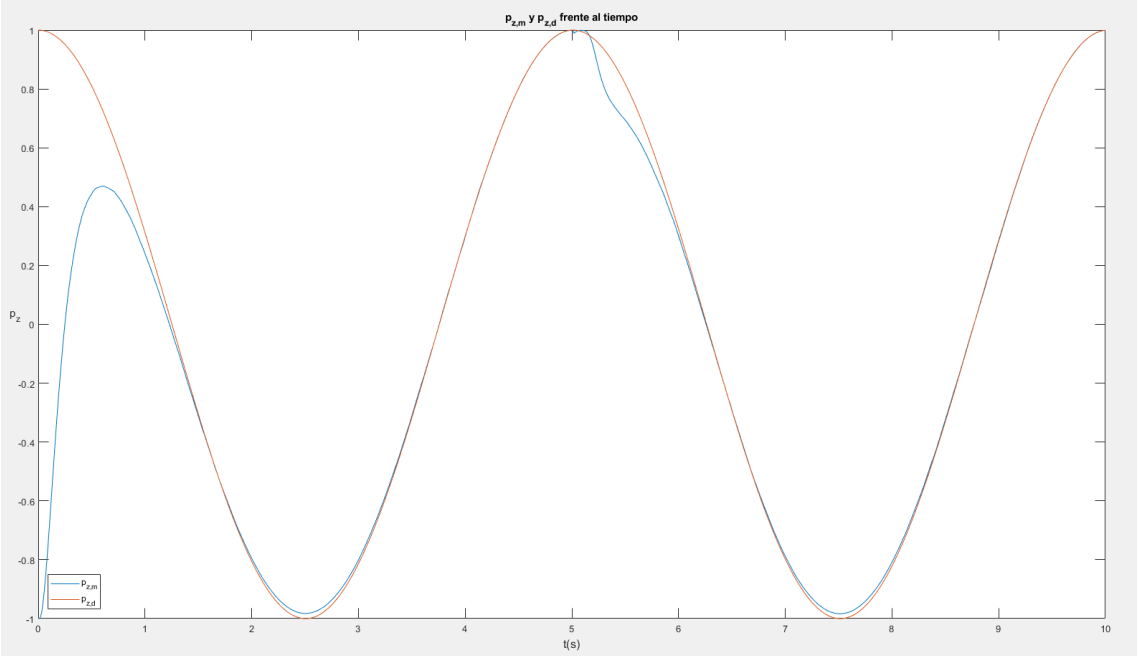

Figura 3.19 Simulación de un Qubit controlado. p_z^m y p_z^d frente al tiempo

Como se aprecia en las gráficas, el control se realiza rápidamente: tanto p_y^m como p_z^m alcanzan rápidamente la trayectoria deseada y la siguen de forma precisa. No obstante, existe una perturbación al final del periodo (en torno a $t = 5$ s), en el mismo instante que en la simulación anterior para el Qubit sin controlar, incluso para la coordenada X, donde a pesar de no ser igual al valor esperado por la existencia de ε no nulo, se produce esa pequeña perturbación.

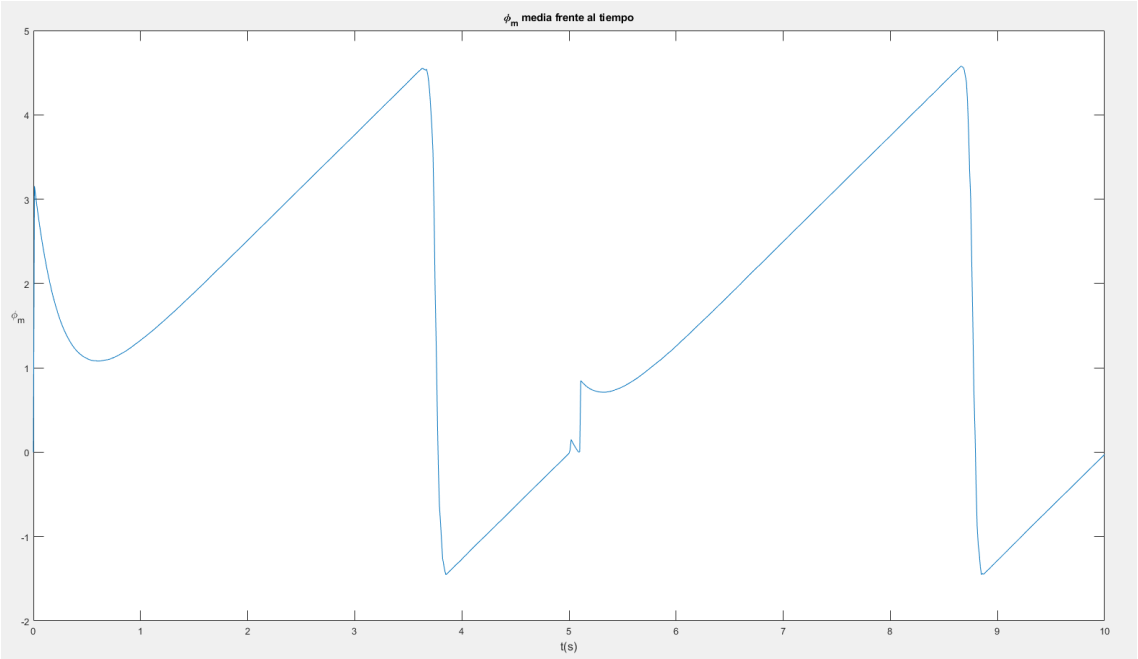

Figura 3.20 Simulación de un Qubit controlado. ϕ_m frente al tiempo

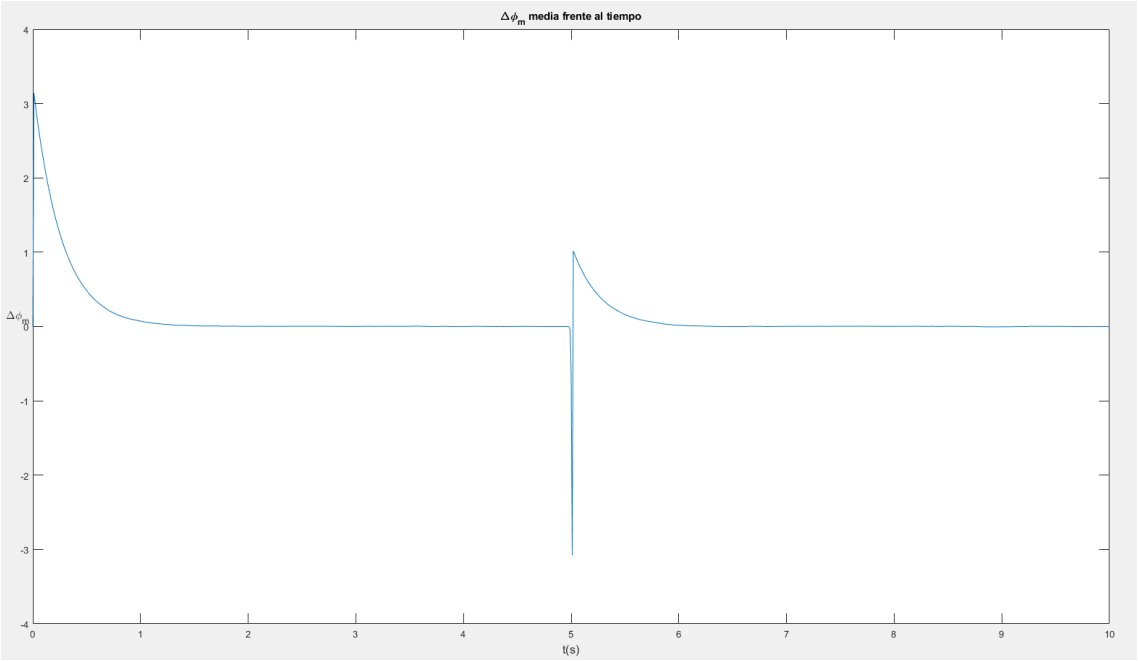

Figura 3.21 Simulación de un Qubit controlado. $\Delta\phi_m$ frente al tiempo

Efectivamente, se refleja en ϕ_m, pero sobre todo en $\Delta\phi_m$, esa discontinuidad que comentábamos. Tal y como está definido actualmente el control, existe un problema cuando se finaliza un ciclo completo. Si observamos estas cuatro gráficas para las 300 trayectorias, podemos apreciar que en algunas de estas trayectorias existen saltos en ese punto crítico.

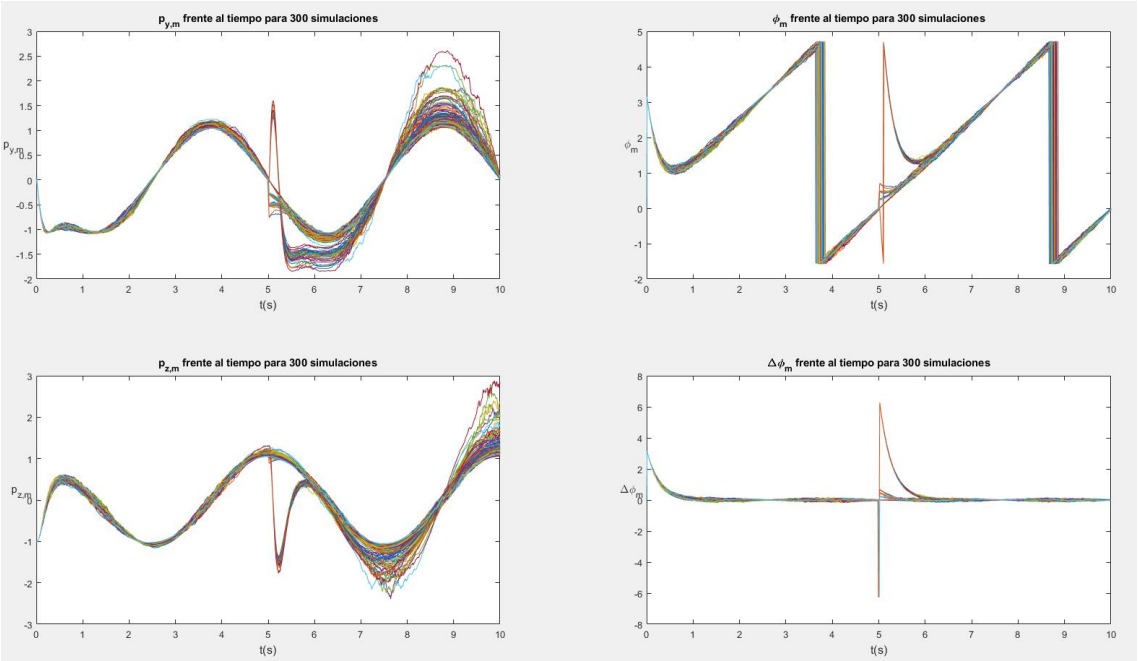

Figura 3.22 Simulación de un Qubit controlado. p_y^m, p_z^m, ϕ_m y $\Delta\phi_m$ frente al tiempo para 300 simulaciones

El problema no es solo que funcione para un único ciclo, aunque se haya especificado que la finalidad de este proyecto es permitir el control de un Qubit durante un tiempo indefinido, sino que estos saltos de $\Delta\phi_m$ imposibilitan totalmente el control en otras condiciones. Así, si duplicamos la frecuencia natural Ω, la manera de realizar control de forma eficiente es, en teoría, aumentando la fuerza. No obstante, debido a estas discontinuidades, existe una fuerza máxima para la que el control deja de funcionar. Así, con $F = 5$ existe un desfase constante (a excepción de los saltos), es decir, no se llega a alcanzar la trayectoria deseada, mientras que si aumentamos $F = 10$, el sistema se inestabiliza completamente debido a que estos saltos son más pronunciados conforme aumentamos este parámetro.

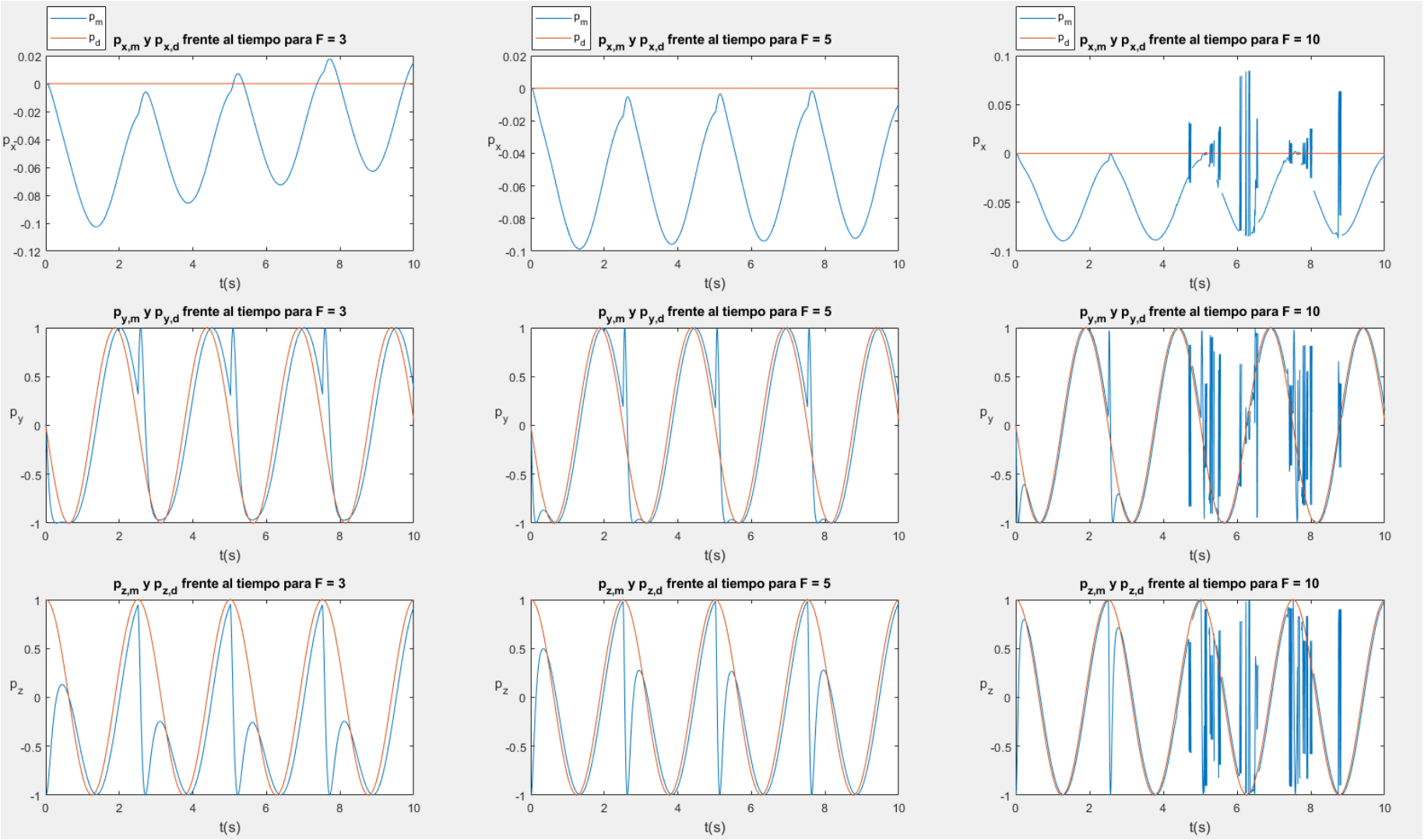

Figura 3.23 Simulación de un Qubit controlado. p_x, p_y y p_z frente al tiempo para fuerzas $F = \{3, 4, 5\}$ y $\Omega = 2\Omega_0$

Como se indicó en la nota al pie de página cuando definimos los parámetros de control, el control propuesto en la bibliografía presentaba errores y problemas a lo largo del ciclo. Siendo más específicos ahora, presentaba problemas para la frecuencia natural de oscilación, posiblemente debido al uso de la arcotangente. Al modificarlo como escribimos en las ecuaciones, aseguramos la continuidad de un periodo, pero seguimos usando las mismas herramientas matemáticas para medir este desfase: la arcotangente.

Por otro lado, aunque el control funcionase, es poco intuitivo descubrir de dónde sale y en base a qué se ha elegido ese control; más adelante podremos razonar en cierto modo qué significaba, pero en un principio, y más teniendo en cuentas los términos que incluyen la función signo o los módulos, no está del todo claro por qué se usa precisamente ese control y no otro.

Por tanto, desde el punto de vista del control, no parece viable emplear módulos, arcotangentes e incluso funciones signo para establecer un feedback, pues son parámetros que en la práctica, como hemos comprobado, no son robustos ni intuitivos. Es por estos motivos por los que modificaremos el control propuesto inicialmente, obtenido de la bibliografía.

3.4. Definición de un nuevo control

El control aplicado al Qubit se basa en las componentes de la matriz densidad para reducir la diferencia de fases entre la trayectoria estimada y deseada. El control que diseñemos debe cumplir esta misma función. Para ello, emplearemos la evolución del Qubit en función del vector de Bloch. Por tanto, empleando las ecuaciones (2.31)-(2.33), definiremos un nuevo control.

Según la bibliografía, el control del Qubit viene determinado por (2.39)-(2.41), basándose en la diferencia de fases. Una vez realizado el cambio, podemos comprobar que para el cálculo de ϕ_m, se estaban empleando en cierto modo los vectores de Bloch; así, $\phi_m = |p_y|/|p_z|$, sin tener en cuenta el segundo término de ϕ_m. No obstante, obtenidas las evoluciones de las componentes de este, es mucho más sencillo determinar esta diferencia de fases. En nuestro caso, como estamos tratando con dos vectores, el monitorizado $\mathbf{p}^m$ y el deseado $\mathbf{p}^d$, haremos uso del producto escalar o interno para determinar el ángulo. El producto escalar para un espacio euclídeo se define como

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos\theta. \tag{3.1}$$

Por tanto, podemos despejar el coseno y obtener fácilmente el ángulo entre ambos vectores. A partir de $\cos\theta$ podríamos calcular $\theta = \text{acos}(\cos\theta)$ [14]. Sin embargo, solamente con esto no es suficiente.

En primer lugar, el producto escalar, por su propia definición, simplemente nos permite calcular el ángulo entre dos vectores, no la diferencia de ángulos; esto es, no nos indica si un vector está adelantado o retrasado respecto al otro[15]. Por otro lado, el coseno es una función par, es decir, cumple $f(x) = f(-x)$. Por este motivo, obtendríamos el mismo resultado para $\pm\theta$, que se

[14] Llegados a este punto podríamos pensar en establecer un parámetro de control de la forma $\Delta = \cos 0 - \cos\theta = 1 - \cos\theta$, y obtener directamente el desfase entre vectores. No obstante, el máximo error para este caso sería equivalente a $\Delta = 2$, mientras que en el caso de emplear directamente θ, el máximo error es equivalente a $\Delta = \pi$, presentando esta segunda opción una mayor precisión.

[15] En caso de que sí se indicase respecto a qué vector se está calculando el ángulo, el software que estamos empleando, MATLAB, tampoco nos permitiría definirlo, pues limita la imagen o recorrido de la función $\cos x$ a $\{[0, \pi]\}$; es decir, deberíamos tener en cuenta que para diferencias de ángulos mayores de π, calcularía el "camino más corto" entre los vectores. Lo mismo ocurre para $\sin x$.

traduce en que da igual que el vector monitorizado se encuentre adelantado o atrasado respecto al deseado, lo interpretará siempre del mismo modo; al realizar el arcocoseno se obtiene un ángulo positivo puesto que $\cos\theta = \cos(-\theta)$. Por tanto, no está totalmente definida la posición de los vectores. Para solventar este problema, haremos uso del producto vectorial o producto cruz.

El producto vectorial nos permite también obtener el ángulo entre los dos vectores, de la forma

$$\mathbf{a} \times \mathbf{b} = |\mathbf{a}||\mathbf{b}| \sin\theta. \tag{3.2}$$

A diferencia de lo que ocurre con el producto escalar, el producto vectorial es anticonmutativo, es decir, $\mathbf{a} \times \mathbf{b} = -(\mathbf{b} \times \mathbf{a})$, lo que quiere decir que existe una diferencia en el signo de $\sin\theta$ si se calcula de una forma u otra, si el primer vector va adelantado o retrasado respecto al segundo. No obstante, sigue presentando otro problema debido a la naturaleza de la función seno; esta es una función impar, es decir, $f(-x) = -f(x)$, por lo que presentará inconvenientes similares a los explicados para el coseno. En el caso de este, $\sin\theta = \sin(\pi - \theta)$.

Nuestro control, inicialmente podemos plantearlo como

$$\Delta\lambda_{fb} = -Fg\Delta\theta \tag{3.3}$$

$$\Delta\theta = \operatorname{acos}(\cos\theta) \tag{3.4}$$

$$\cos\theta = \frac{\mathbf{p^m} \cdot \mathbf{p^d}}{|\mathbf{p^m}||\mathbf{p^d}|} \tag{3.5}$$

Una vez definido completamente el ángulo, existe un último par de cosas a tener en cuenta en nuestro control. Primero, una vez que sabemos si el vector medido está adelantado o retrasado respecto al deseado, debemos establecer una condición que cambie el "sentido" del feedback; es decir, que nos permite modificar $\Delta\lambda_{fb}$ según necesitemos frenar el vector controlado o acelerarlo. Esto lo podemos realizar mediante una condición lógica para el caso en que el vector medido esté adelantado al vector deseado

$$\Delta\theta = -\Delta\theta \text{ si } \cos\theta > 0 \text{ y } \sin\theta < 0 \tag{3.6}$$

De este modo, cambiaremos el sentido de la fuerza del feedback. No obstante, para poder observar con mayor facilidad el error absoluto, sin necesidad de saber la posición relativa de los vectores, estableceremos una variable signo s inicializada a 1 multiplicando a $\Delta\theta$ y que cambiará su valor a -1 de forma puntual cuando se cumpla la condición, por lo que tras establecer $\Delta\lambda_{fb}$ en ese paso, recuperará su valor positivo.

El segundo apunte que tomaremos tiene que ver con los parámetros intrínsecos del Qubit. El hamiltoniano viene determinado por λ y ε. En nuestro control, modificamos los parámetros de la tunelización λ, pero no los de la asimetría del Qubit ε. Tiene sentido puesto que ε es un parámetro intrínseco a la construcción del Qubit. Si el Qubit fuese ideal, este valor sería igual a cero y presentaría una trayectoria perfecta a lo largo de un plano meridional, tal y como está definida nuestra trayectoria deseada. No obstante, en la práctica no todo es ideal. Por ese motivo, estamos trabajando con un valor distinto de cero para ε. Esto provoca que, cuando calculamos el ángulo entre vectores, exista siempre un desfase que no se puede eliminar puesto que depende de ε. Por este motivo, nuestro control no lo realizaremos con los vectores monitorizados "completos", sino que eliminaremos de nuestro control la aportación del p_x, puesto que para la trayectoria ideal esta componente es nula.

Por tanto, definiremos unos vectores de control (subíndice c), tanto para los vectores medidos como los deseados, de la forma

$$\begin{aligned} \mathbf{p}_c &= (p_y, p_z), \\ \mathbf{p}_{cross} &= (0, p_y, p_z), \end{aligned} \tag{3.7}$$

donde $\mathbf{p}_{cross}$ se ha definido de este modo para poder operar adecuadamente con el producto vectorial. Nuestro control, es decir, tanto el producto escalar como el vectorial, lo realizaremos con estos vectores. Así, aseguramos que exista siempre la menor diferencia de fases entre los valores medidos y los deseados.

Finalmente, nuestro control queda recogido en las ecuaciones (3.8)-(3.12).

$$\Delta\lambda_{fb} = -sFg\Delta\theta, \tag{3.8}$$

$$\Delta\theta = \operatorname{acos}(\cos\theta) \tag{3.9}$$

$$\cos\theta = \frac{\mathbf{p_c^m} \cdot \mathbf{p_c^d}}{|\mathbf{p_c^m}||\mathbf{p_c^d}|}, \tag{3.10}$$

donde el valor de s viene dado por la condición

$$s = \begin{cases} -1, & \cos\theta > 0 \text{ y } \sin\theta < 0 \\ 1, & \text{en otro caso} \end{cases} \tag{3.11}$$

con

$$\sin\theta = \frac{\mathbf{p_c^m} \times \mathbf{p_c^d}}{|\mathbf{p_c^m}||\mathbf{p_c^d}|}. \tag{3.12}$$

El código para las simulaciones de la evolución del Qubit en la forma de los vectores de Bloch y el nuevo control se encuentran recogidos también en el Apéndice A.

Por último, es interesante mencionar un parámetro para medir la eficiencia de control que emplearemos en alguno de los apartados, definido como

$$D = \cos\Delta\theta \tag{3.13}$$

de este modo, cuando el desfase sea cercano 0, D presentará un valor próximo a 1, mientras que para desfases cercanos a π, D será prácticamente nulo.

Por tanto, a partir de ahora simularemos empleando las ecuaciones de evolución (2.31)-(2.33) del vector de Bloch, y aplicando el control definido en (3.8)-(3.12). Como ya comentamos anteriormente, realizaremos las gráficas a partir de los valores medios y normalizados, empleando los parámetros indicados en la sección 3.1. Por otro lado, ciertas gráficas como la de la intensidad en el detector, las omitiremos pues no muestran nada interesante más allá de lo que ya hemos visto. Recordemos que, a excepción de lo que dependa del control, las simulaciones y cálculos son equivalentes puesto que se pueden obtener mediante una transformación directa.

3.5. Simulación de un Qubit sin control mediante su vector de Bloch asociado

En primer lugar, realizaremos una simulación sin control para garantizar que el Qubit se comporta como lo hacía anteriormente. Al igual que para la simulación sin control mediante las ecuaciones de la matriz densidad, estableceremos unas condiciones iniciales iguales a unos valores modelo; de esta manera, el Qubit debería seguir la misma trayectoria y podremos comprobar cómo se comporta el parámetro de control $\Delta\theta$.

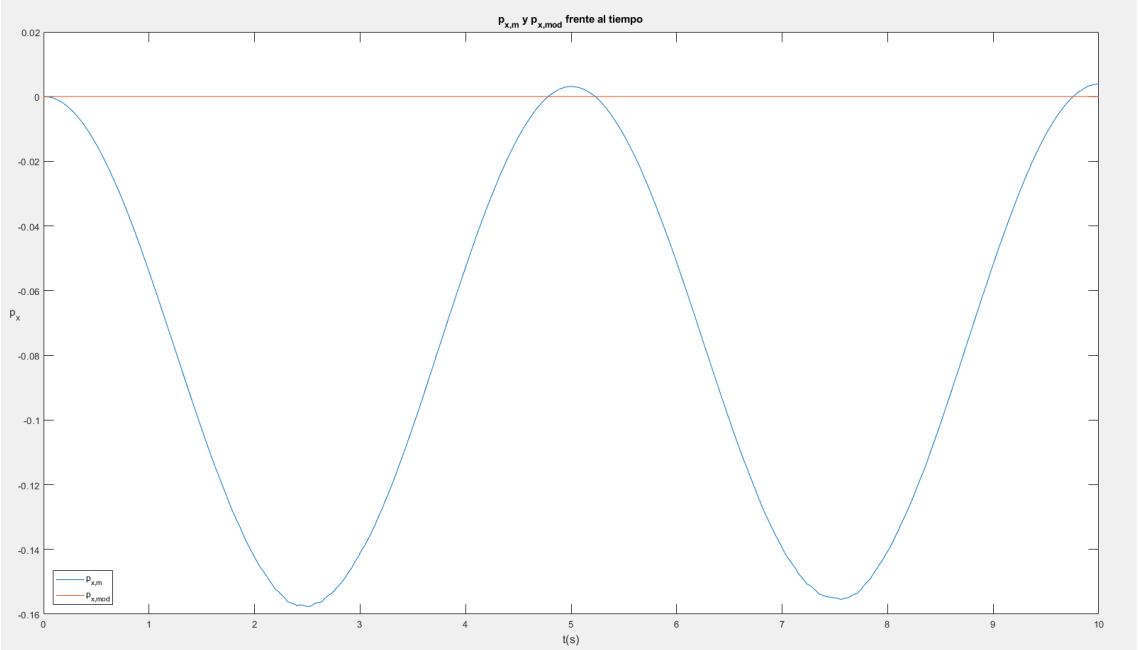

Figura 3.24 Simulación de un Qubit sin control. p_x^m y p_x^{mod} frente al tiempo

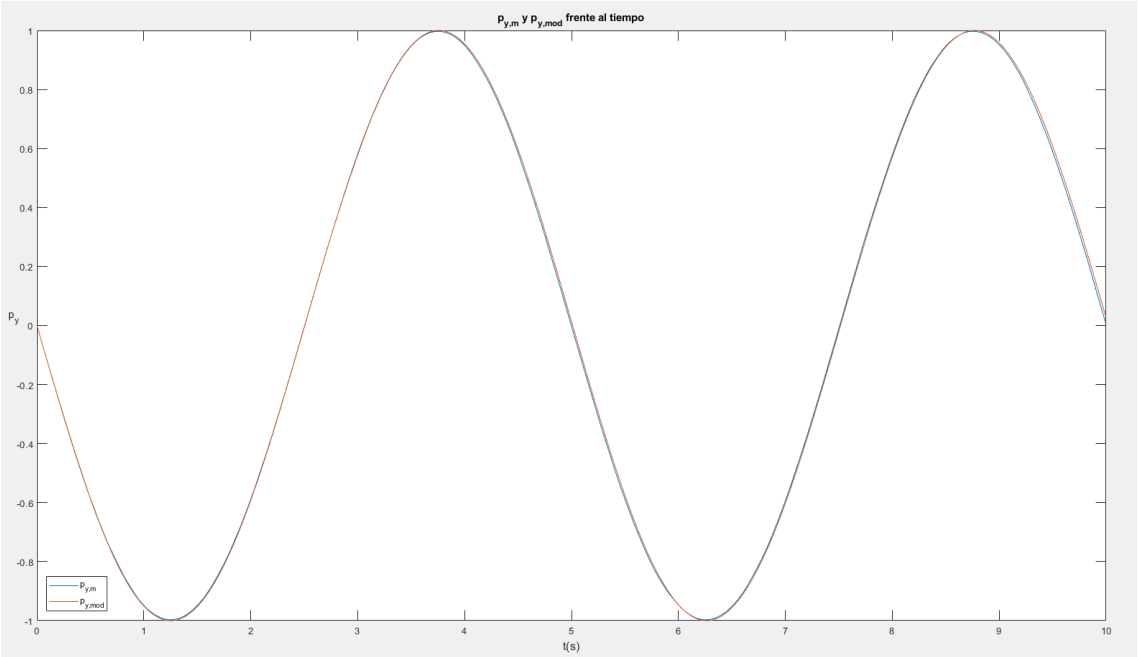

Figura 3.25 Simulación de un Qubit sin control. p_y^m y p_y^{mod} frente al tiempo

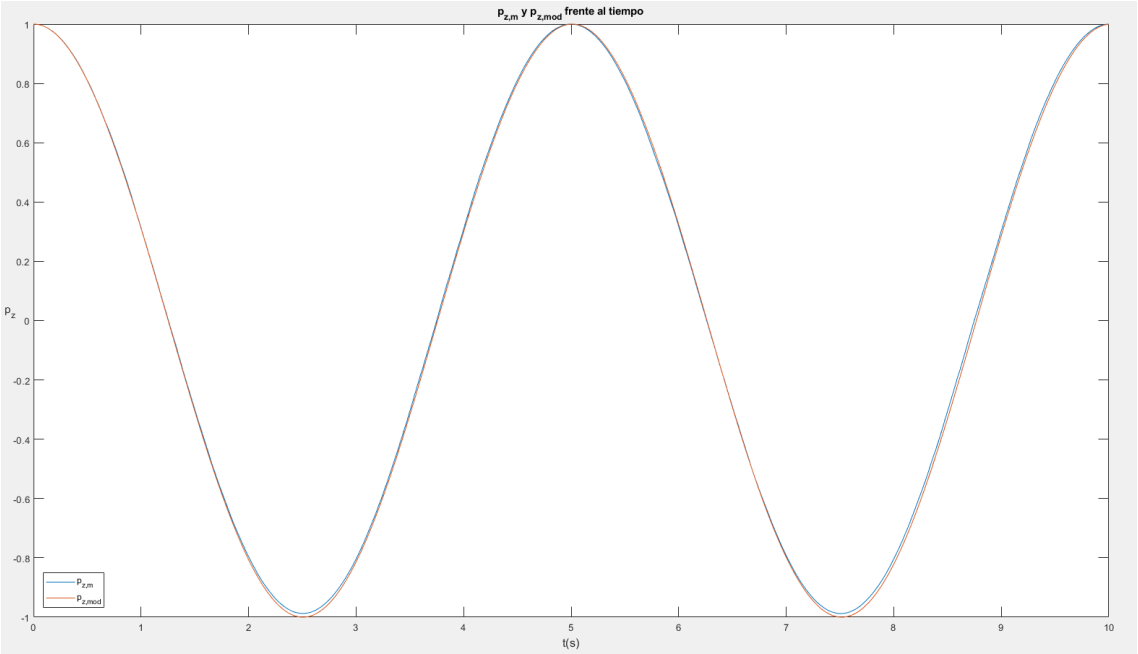

Figura 3.26 Simulación de un Qubit sin control. p_z^m y p_z^{mod} frente al tiempo

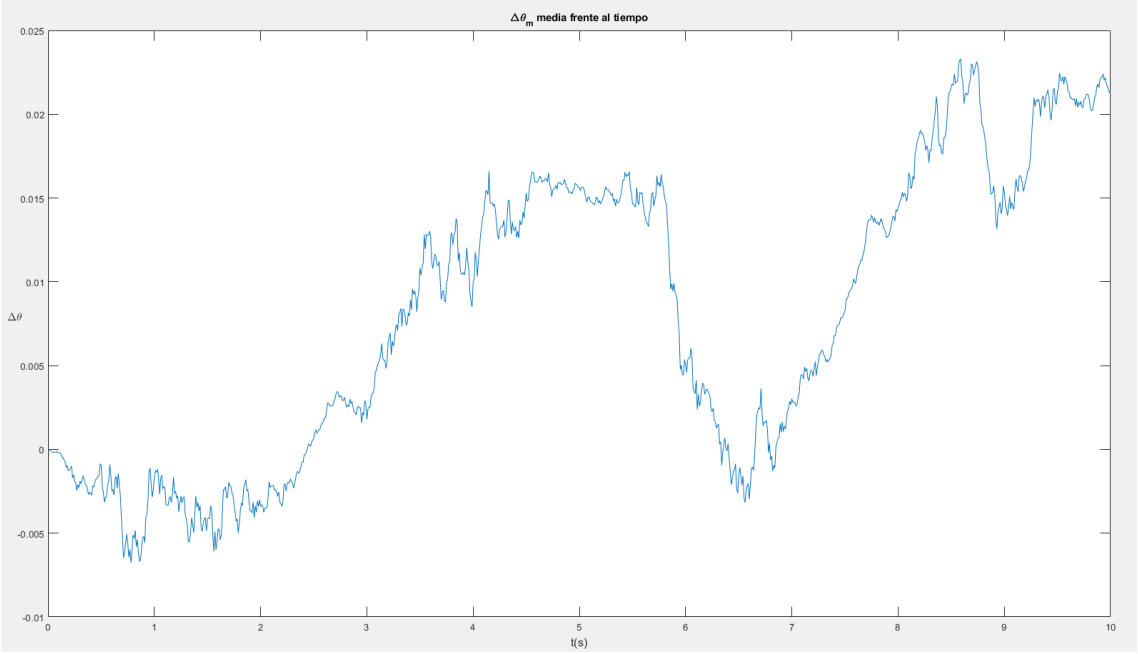

Figura 3.27 Simulación de un Qubit sin control. $\Delta\theta$ frente al tiempo

Se observa que las gráficas de la evolución del Qubit son exactamente iguales que para las simulaciones realizadas con el otro grupo de ecuaciones; el Qubit girando a su frecuencia natural con las condiciones adecuadas y sin perturbaciones, se comporta como los valores modelo. En la coordenada X del vector se puede apreciar esa pequeña variación en torno al valor nulo debido a la influencia de ε, como ya se ha comentado. Más adelante, durante el control, explicaremos con mayor exactitud cómo influye este parámetro. Por su parte, en la gráfica de $\Delta\theta$ se aprecia que presenta un valor muy cercano a cero.

3.6. Control de un Qubit para condiciones iniciales diferentes a las deseadas

En primer lugar, analizaremos cómo se comporta el Qubit y su control para el caso de que se inicie con unas condiciones iniciales diferentes a las deseadas. Las equivalentes para el caso $\overrightarrow{\rho_0} = (0, 0, 1)$ son $\mathbf{p}_0 = (0, 0, -1)$. Al igual que antes, emplearemos $F = 3$.

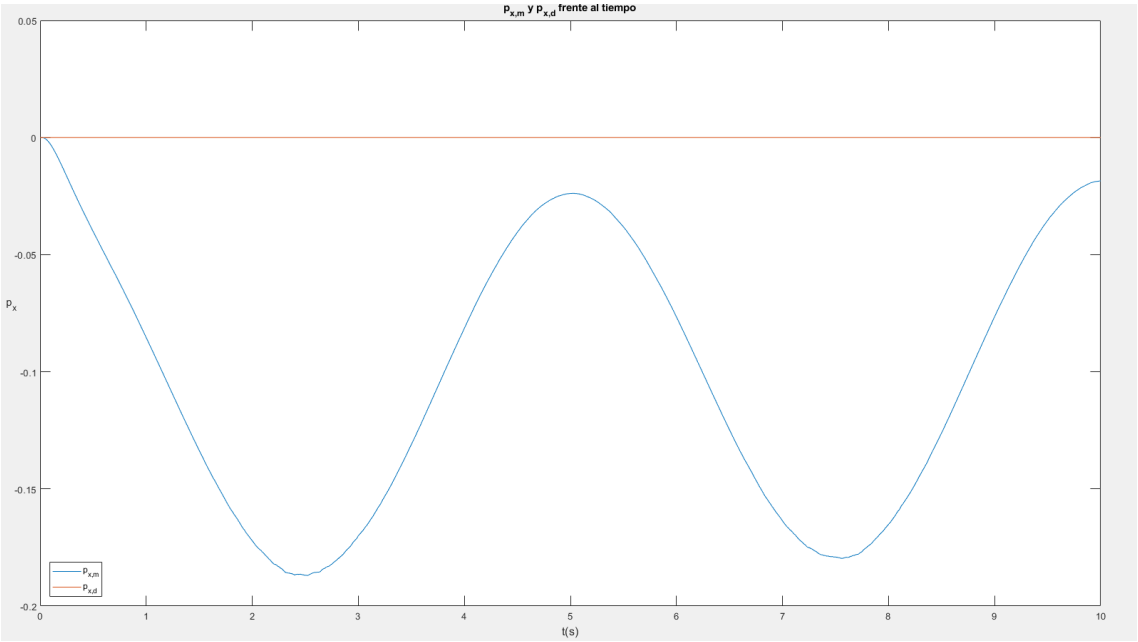

Figura 3.28 Simulación de un Qubit controlado. p_x frente al tiempo

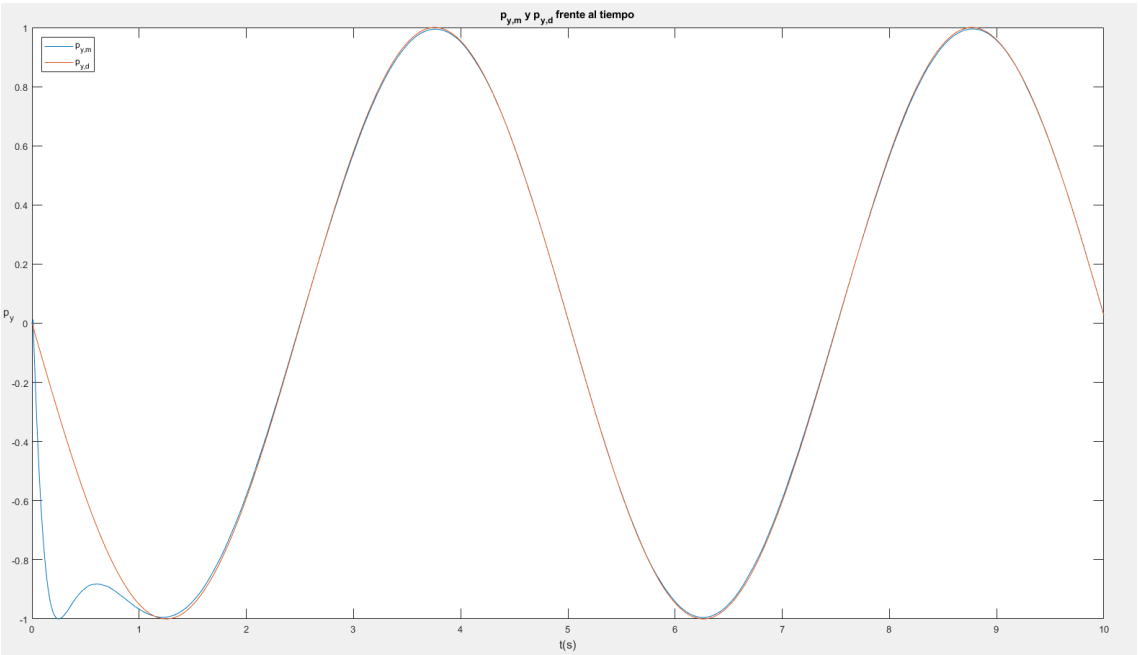

Figura 3.29 Simulación de un Qubit controlado. p_y frente al tiempo

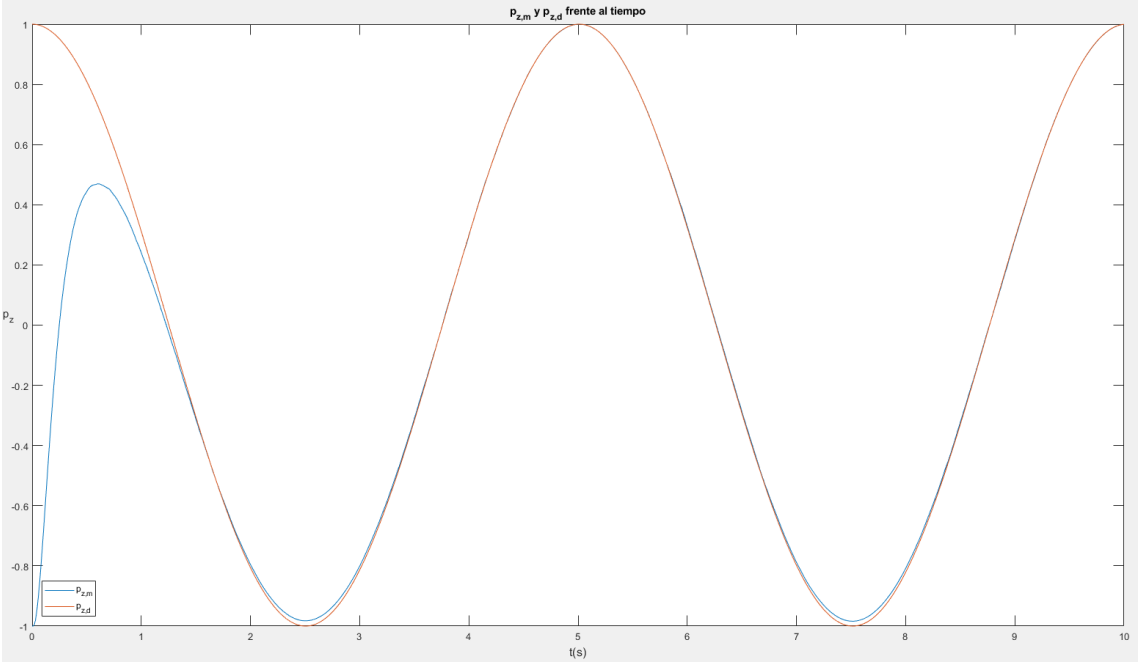

Figura 3.30 Simulación de un Qubit controlado. p_z frente al tiempo

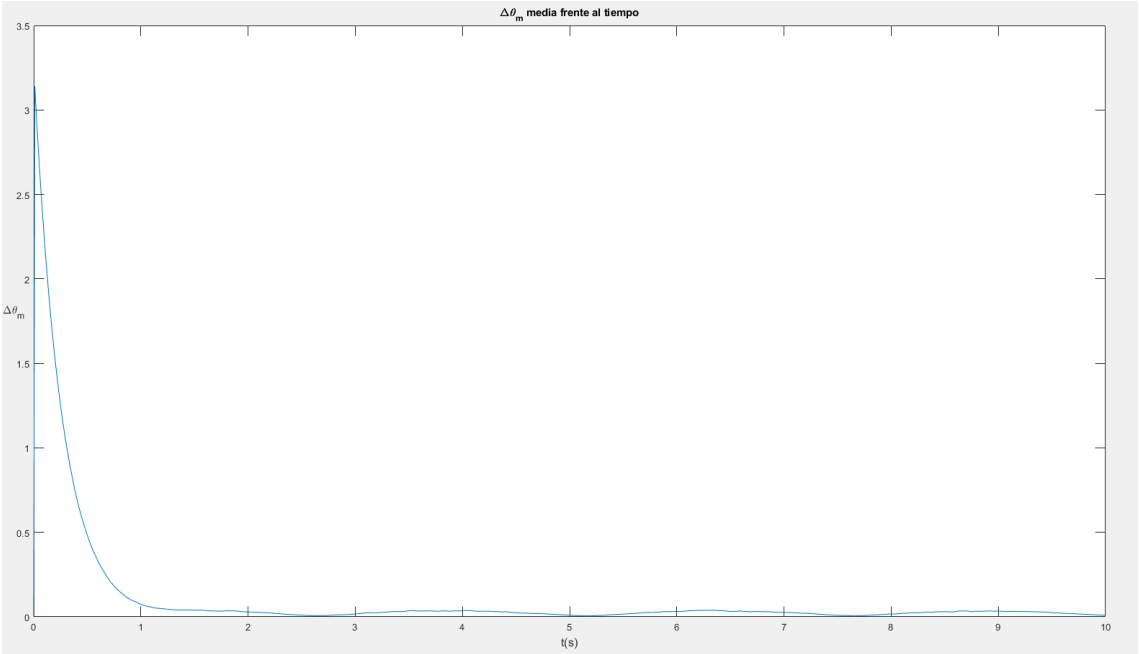

Figura 3.31 Simulación 3.6.A. $\Delta\theta$ frente al tiempo

Se aprecia claramente que las componentes del vector monitorizado cambian rápidamente su trayectoria natural y alcanzan de una forma rápida y precisa los valores deseados. En la gráfica de p_x se comprueba lo que comentamos a la hora de definir el control: el valor de ε impide que se pueda seguir la trayectoria deseada completamente. A continuación, mostraremos

brevemente cómo se verían las gráficas para un caso ideal, esto es, un Qubit simétrico con $\varepsilon = 0$.

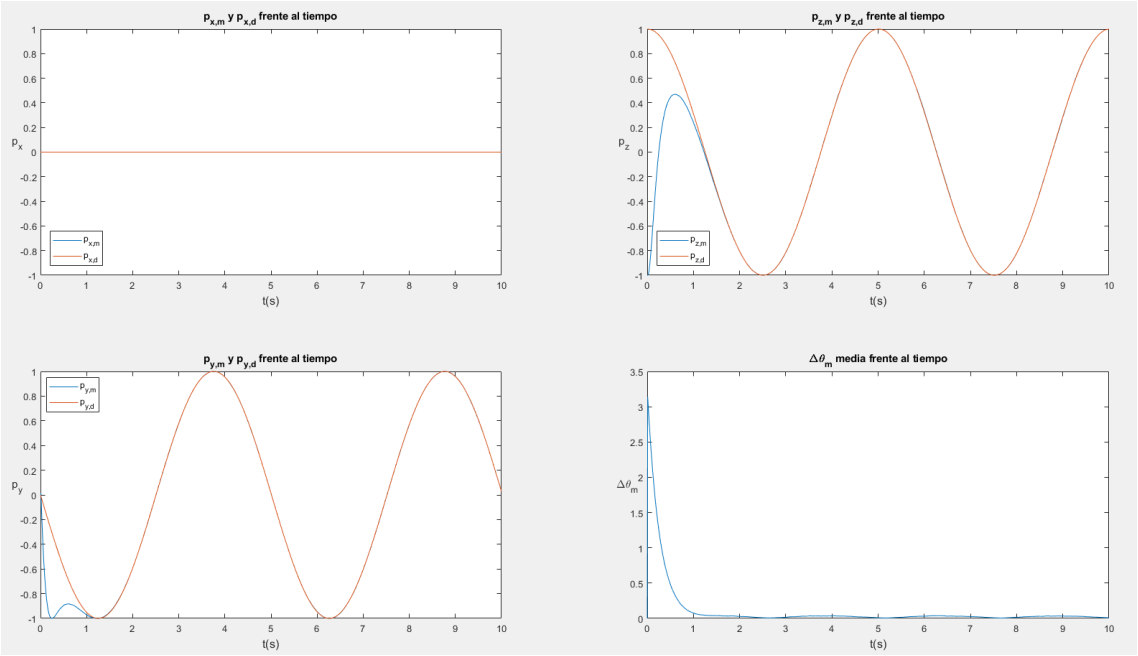

Figura 3.32 Simulación de un Qubit controlado. p_x, p_y, p_z y $\Delta\theta$ frente al tiempo con $\varepsilon = 0$

Se puede observar que para este caso ideal, el valor de $p_x^m = p_x^d = 0$, y que el resto de las gráficas se comportan del mismo modo. Como hemos explicado durante el control, la ángulo entre los vectores que pueda provocar ε no aparece recogida en la gráfica de $\Delta\theta$, que presenta valores muy cercanos a 0 una vez finalizado el período de transición inicial para los dos casos.

Por otro lado, se ha conseguido solventar el problema que producía el control propuesto inicialmente: este Qubit podría ser controlado de un modo eficiente durante un tiempo largo indefinido, que es uno de los objetivos del proyecto.

Por último, para apreciar el control, así como lo explicado sobre el factor de asimetría del Qubit, representaremos algunos estados en la esfera de Bloch a lo largo del tiempo. Una vez más, podemos apreciar la evolución del Qubit. En ambos casos, el Qubit se inicia en condiciones totalmente opuestas a las deseadas, avanza en sentido contrario hasta una cierta diferencia de fases, momento en el que se frena hasta que lo alcanza completamente y evolucionan del mismo modo. Si lo que buscamos son diferencias entre ambas simulaciones, para el Qubit ideal existe un desfase prácticamente nulo en todas las dimensiones, mientras que para el caso del Qubit asimétrico, la diferencia es nula simplemente respecto al plano YZ.

En las siguientes imágenes se puede visualizar en una esfera de Bloch la evolución de los vectores monitorizados (rojo) y deseados (azul) para ambos casos, “leyéndose” de izquierda a derecha. Se puede comprender por qué se afirmó anteriormente que la esfera de Bloch es la representación más sencilla para un sistema cuántico binario.

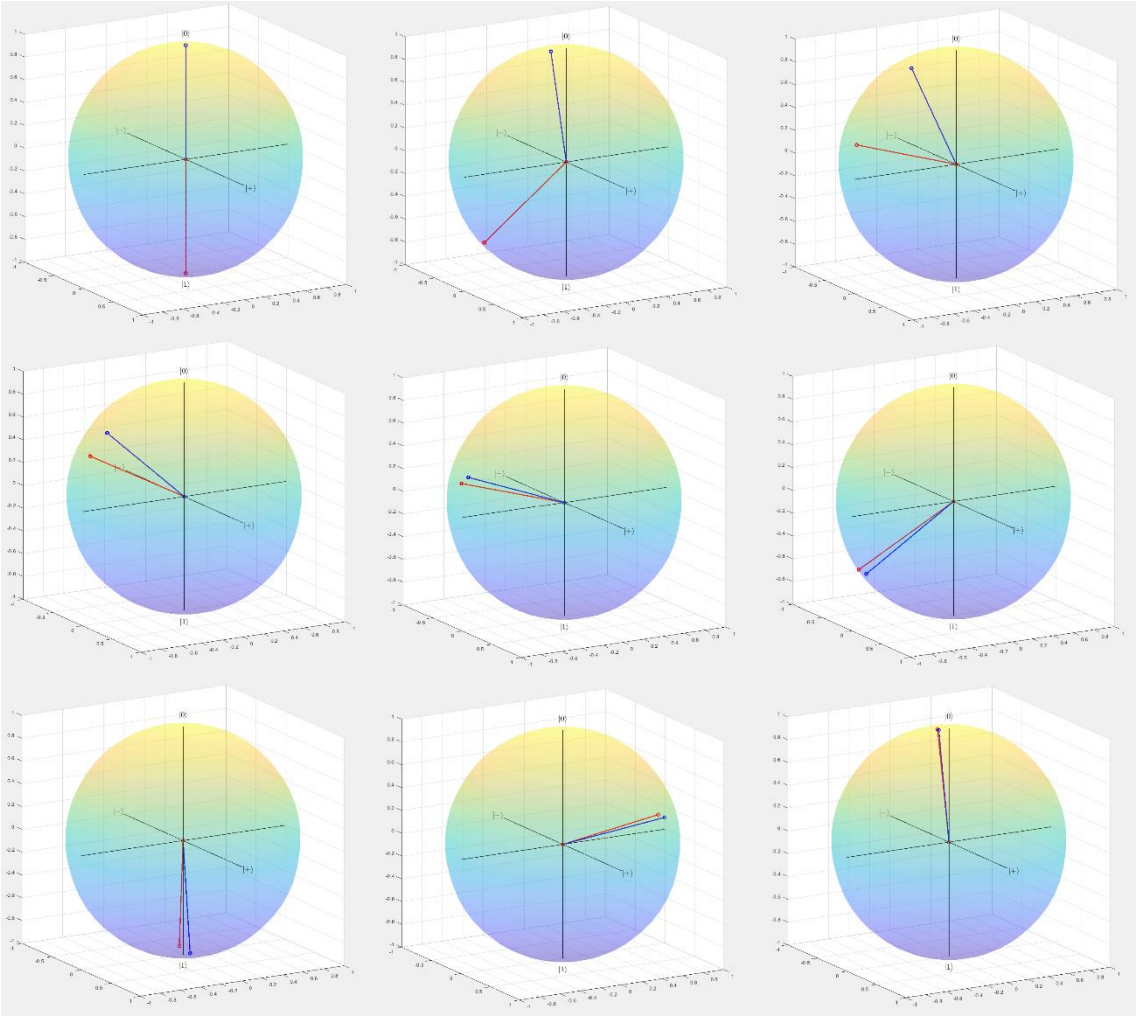

Figura 3.33 Simulación de un Qubit controlado. Representación de un Qubit en la esfera de Bloch con $\varepsilon \neq 0$

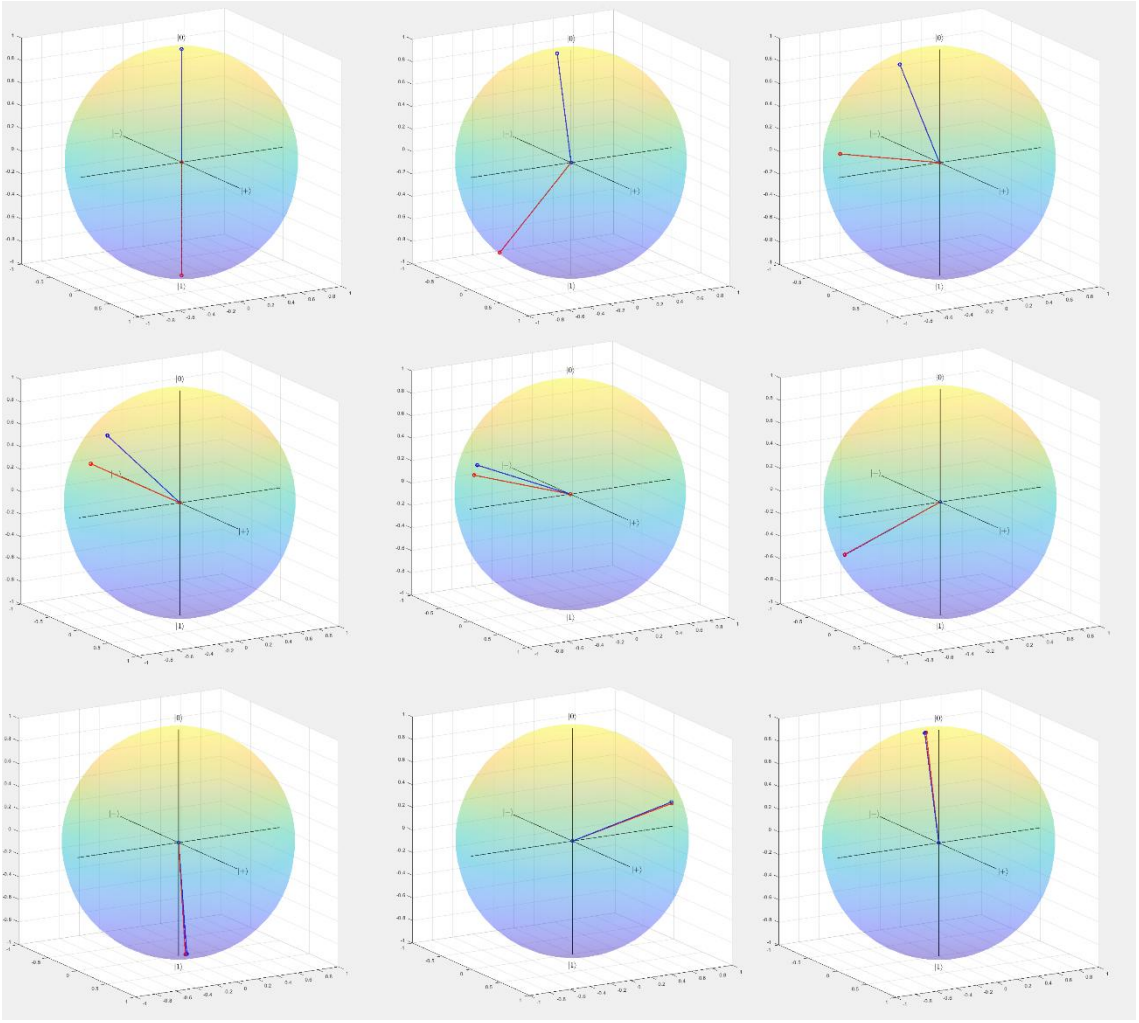

Figura 3.34 Simulación de un Qubit controlado. Representación de un Qubit en la esfera de Bloch con $\varepsilon = 0$

3.7. Control de un Qubit para velocidades diferentes a la natural

El Qubit que estamos estudiando presenta una frecuencia natural (2.14). Con el control obtenido de la bibliografía, pudimos comprobar que este parámetro limitaba el funcionamiento de este. Procederemos ahora a comprobar cómo funciona el nuevo feedback. Para ello, realizaremos pruebas para velocidades $\Omega_2 = 2\Omega$, $\Omega_3 = 3\Omega$, $\Omega_4 = 4\Omega$, $\Omega_{0.5} = \Omega/2$ y $\Omega_0 = 0$, modificando las fuerzas cuando sea necesario. Las condiciones iniciales las mantendremos, por lo general, $\mathbf{p}_0 = (0, 0, -1)$.

El primer caso de estudio es para una frecuencia igual al doble de la natural, y para fuerzas $F = \{3, 5, 10\}$ se obtienen los siguientes resultados.

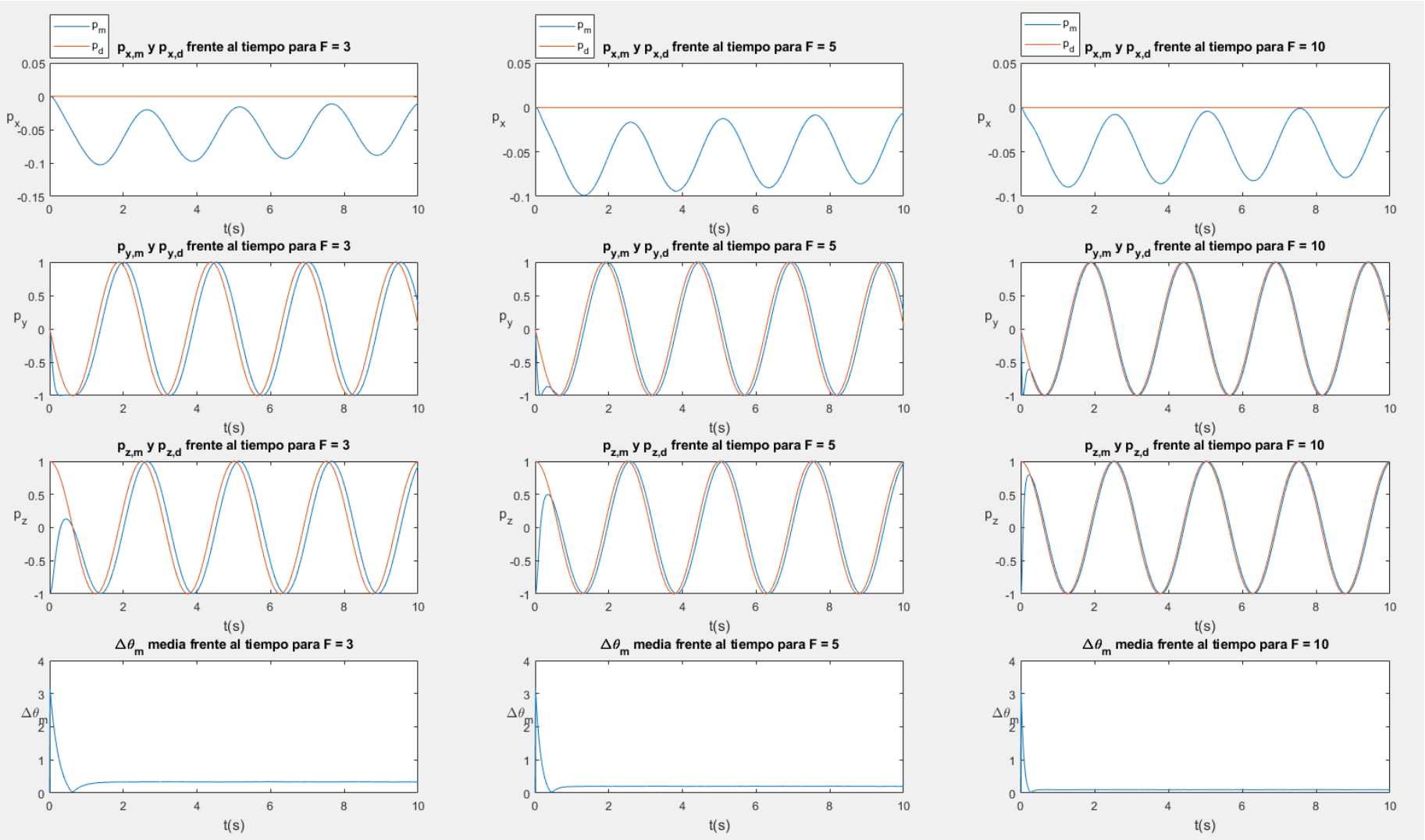

Figura 3.35 Simulación de un Qubit controlado. p_x, p_y, p_z y $\Delta\theta$ para frecuencia $\Omega_2 = 2\Omega$

Podemos sacar algunas conclusiones de las gráficas. A medida que aumenta la fuerza, el Qubit monitorizado tarda menos en alcanzar el valor deseado; esto se traduce en que el error alcanza su valor mínimo antes conforme mayor es la fuerza. Por otro lado, también a medida que aumenta la fuerza, el desfase en el tiempo estacionario disminuye, es decir, el error en estacionario es menor cuanto mayor es la fuerza.

Hay que tener en cuenta que los valores que se muestran son los medios y, en el caso de las componentes del vector, normalizados también. Si no normalizásemos lo valores, conforme aumentase la fuerza, mayor serían los valores que alcanzan estas componentes. No obstante, el control que realizamos es sobre la fase, por eso esto no significa un gran problema.

El segundo caso de estudio es para una frecuencia igual a tres veces la natural. Por lo que se ha apreciado en el caso anterior, cabe esperar que una fuerza pequeña no sea suficiente para controlarlo; emplearemos fuerzas $F = \{5, 10, 20\}$.

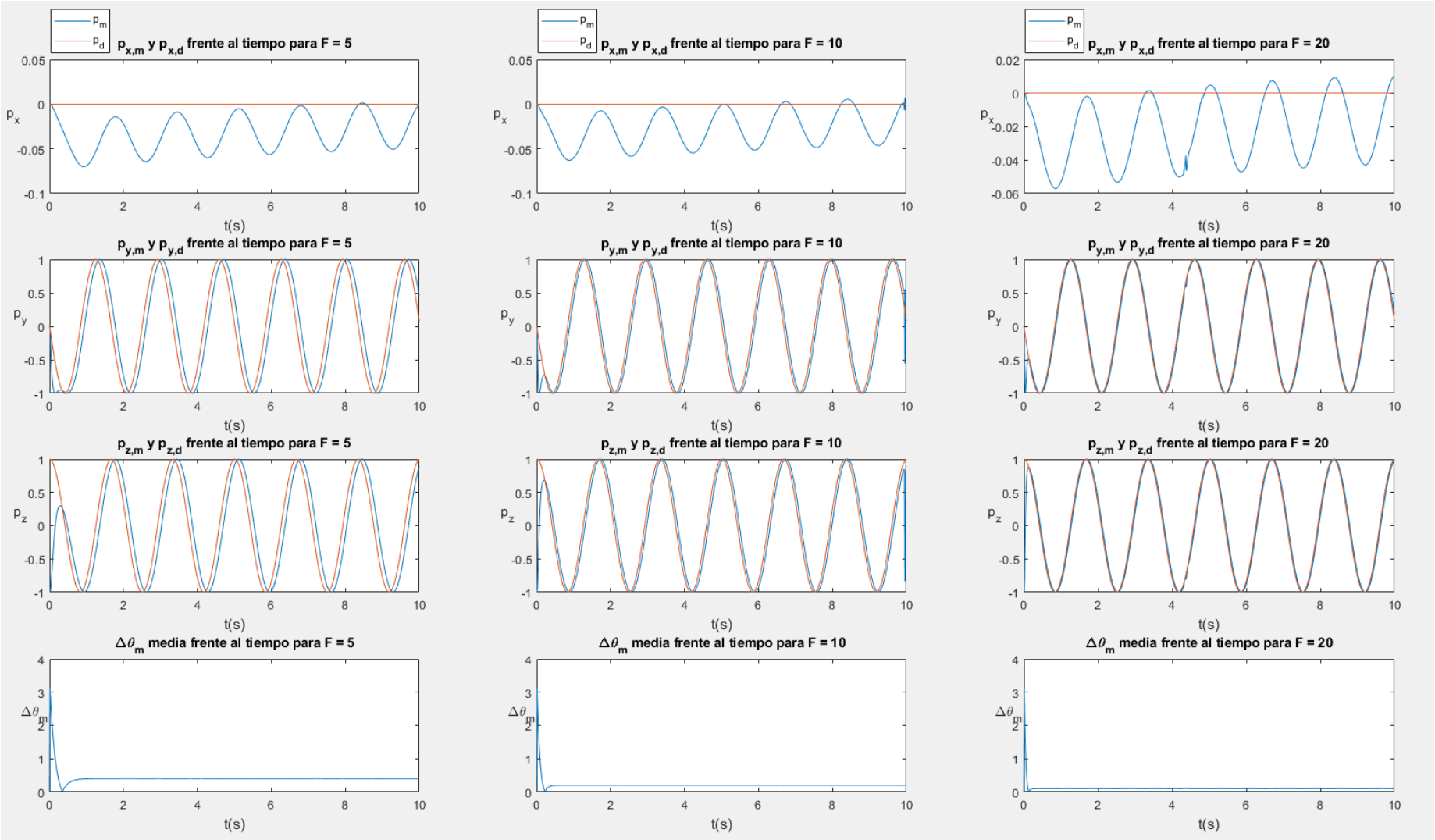

Figura 3.36 Simulación de un Qubit controlado. p_x, p_y, p_z y $\Delta\theta$ para frecuencia $\Omega_3 = 3\Omega$

En la Figura anterior podemos comprobar que para la frecuencia señalada, el control funciona correctamente. Se cumple la misma relación respecto a los desfases y nuestro parámetro de control $\Delta\theta$ con respecto a la fuerza que antes. No obstante, respecto a la componente p_x, se puede observar que, aunque para $F = 10$ su valor se acerca más a cero que en el caso de $F = 5$, sí que es cierto que para $F = 20$ esta presenta una mayor oscilación e incluso un pequeño pico (en torno al cuarto segundo). Si probamos a realizar simulaciones para $F = 30$, se puede comprobar que el sistema se descontrola; es decir, existe una fuerza límite para cada frecuencia a partir de la cual el sistema se descontrola.

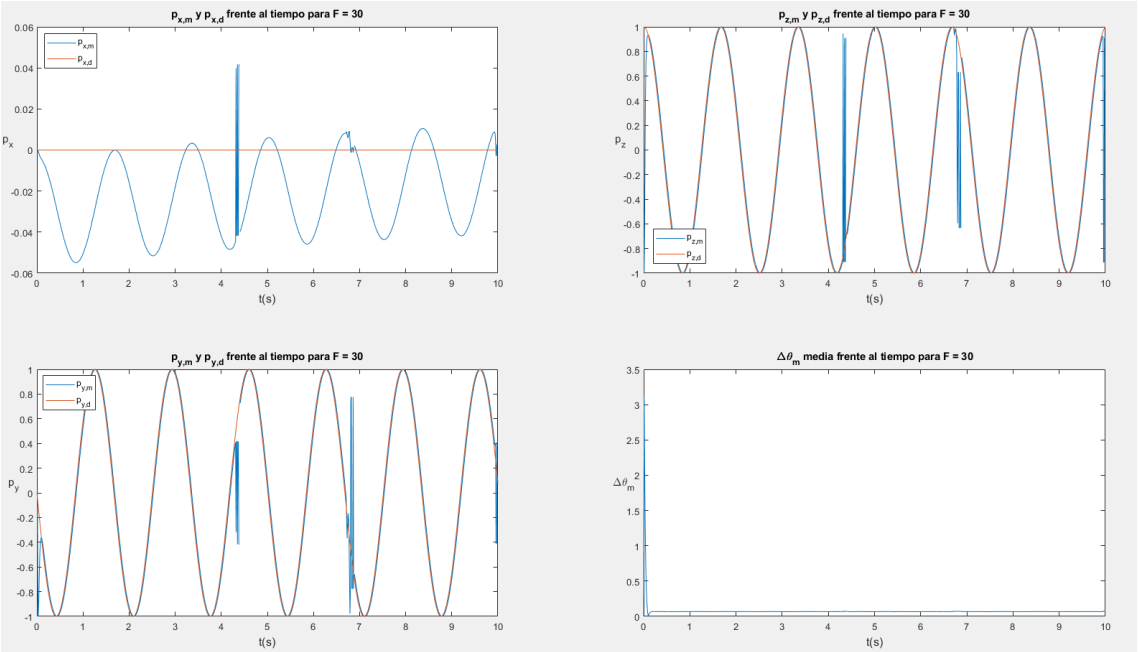

Figura 3.37 Simulación de un Qubit controlado. p_x, p_y, p_z y $\Delta\theta$ para frecuencia $\Omega_3 = 3\Omega$ y $F = 30$

La última frecuencia más rápida de la natural que probaremos es $\Omega_4 = 4\Omega$.

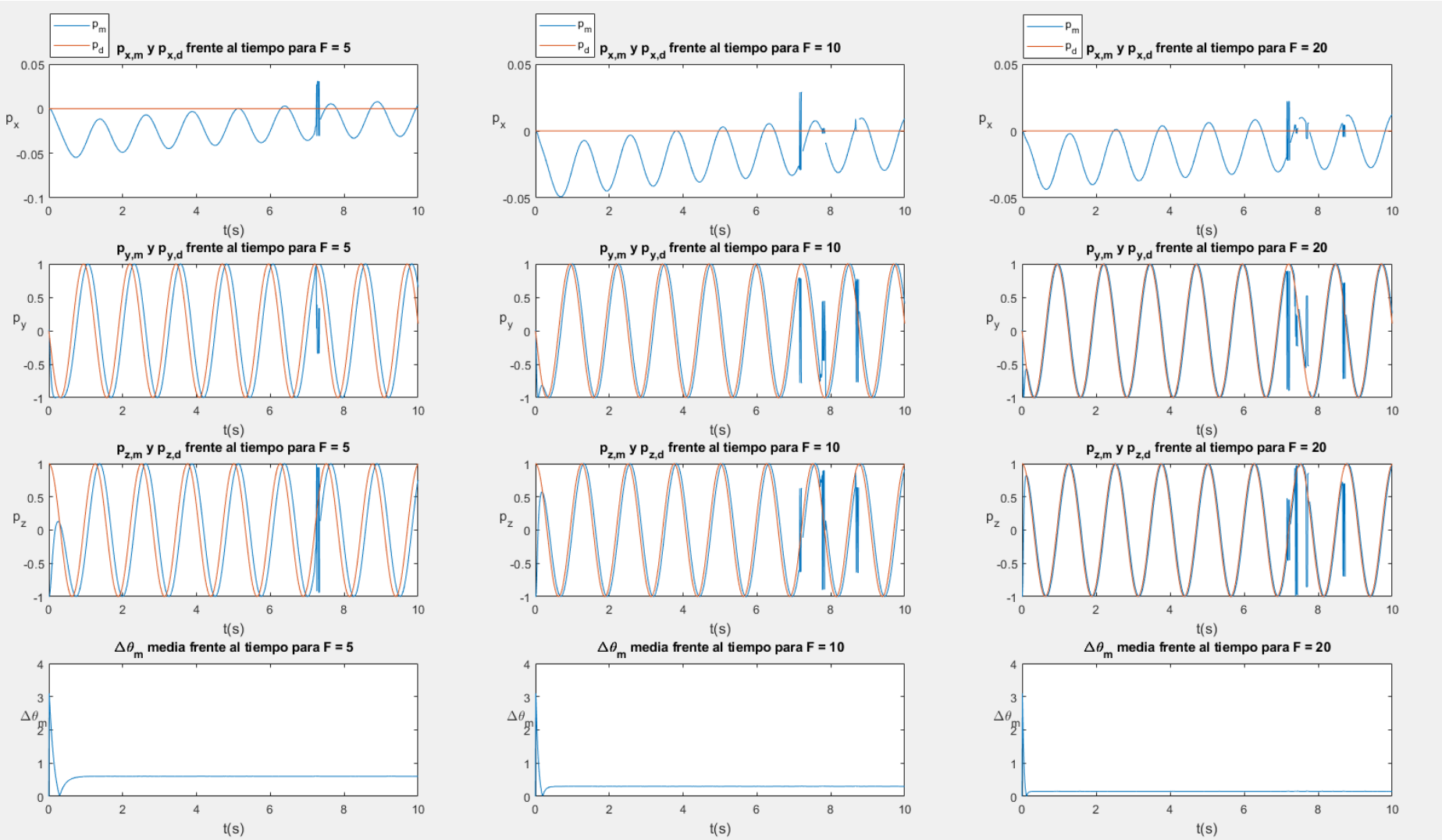

Figura 3.38 Simulación de un Qubit controlado. p_x, p_y, p_z y $\Delta\theta$ para frecuencia $\Omega_4 = 4\Omega$

Existen inestabilidades, a pesar de aumentar la fuerza. Hay que tener en cuenta que el control inicial de la bibliografía estaba diseñado para una única frecuencia; por tanto, parece que no es posible controlar cuando se alcanza esta velocidad. A pesar de que el error parezca cercano a cero, existen discontinuidades que perjudican considerablemente la evolución del Qubit.

No obstante, vamos a estudiar con mayor detenimiento el último caso, es decir, para $F = 20$. Para ello, analizaremos los datos numéricos en bruto. Observando los datos de p_x, podemos ver que existen algunas columnas en las que se alcanzan valores indeterminados[16], es decir, trayectorias que se han inestabilizado completamente. Al eliminar manualmente estas trayectorias (pues no está implementada esta opción en el software de simulación), se puede comprobar que simplemente cuatro trayectorias de las 300 se inestabilizaban. Es decir, solo un $1.34\ \%$ de las trayectorias presentan error, un porcentaje bastante bajo.

De este modo, podemos realizar las medias y normalizar los valores, y observar cómo se comporta la simulación.

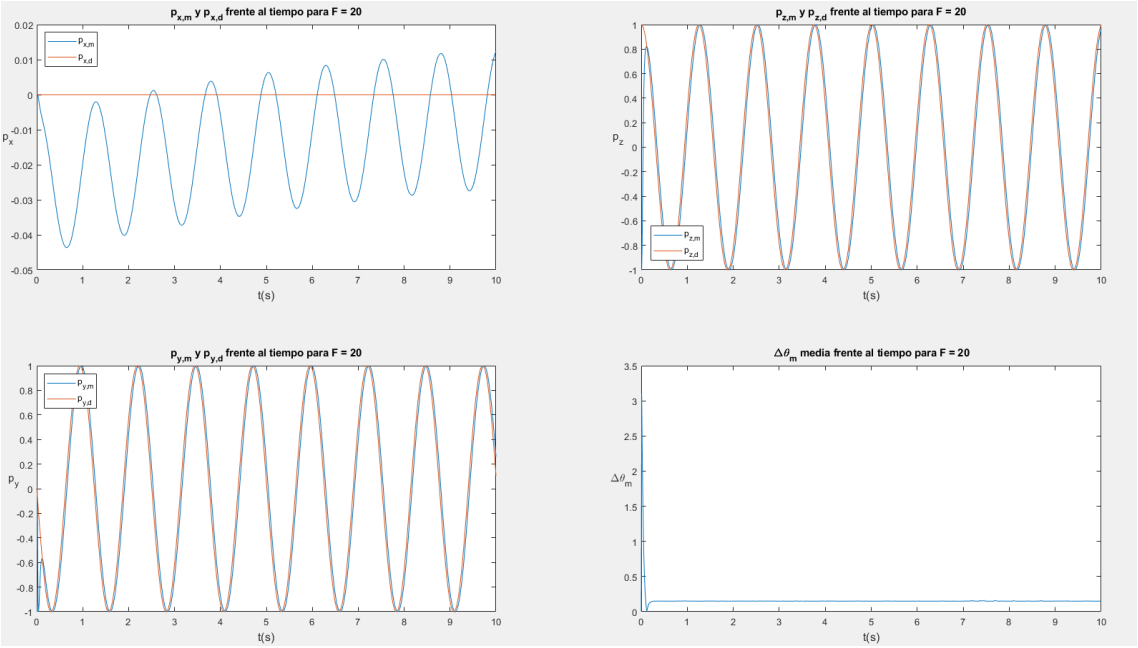

Figura 3.39 Simulación de un Qubit controlado. p_x, p_y, p_z y $\Delta\theta$ para frecuencia $\Omega_4 = 4\Omega$ y $F = 20$, eliminando trayectorias inestables

Como era de esperar al eliminar las trayectorias indeseadas, el control se realiza correctamente. Ya no existen las discontinuidades de la primera prueba, pues se debía únicamente a esas trayectorias. Por tanto, podemos concluir diciendo que para una frecuencia cuatro veces la natural, se puede realizar un control aceptable, aunque con una posibilidad de error existente; cuanto mayor sea la frecuencia, mayor número de trayectorias se inestabilizarán.

A continuación, analizaremos una frecuencia más lenta que la natural, exactamente a la mitad de velocidad. En este caso, emplearemos las fuerzas $F = \{3, 5, 10\}$.

[16] En MATLAB, los valores indeterminados tales como $0/0$ son señalados como `NaN`. Para eliminar las columnas de una matriz A en los que aparezcan estos valores basta con ejecutar la siguiente instrucción: `out = A(:,all(~isnan(A)))`.

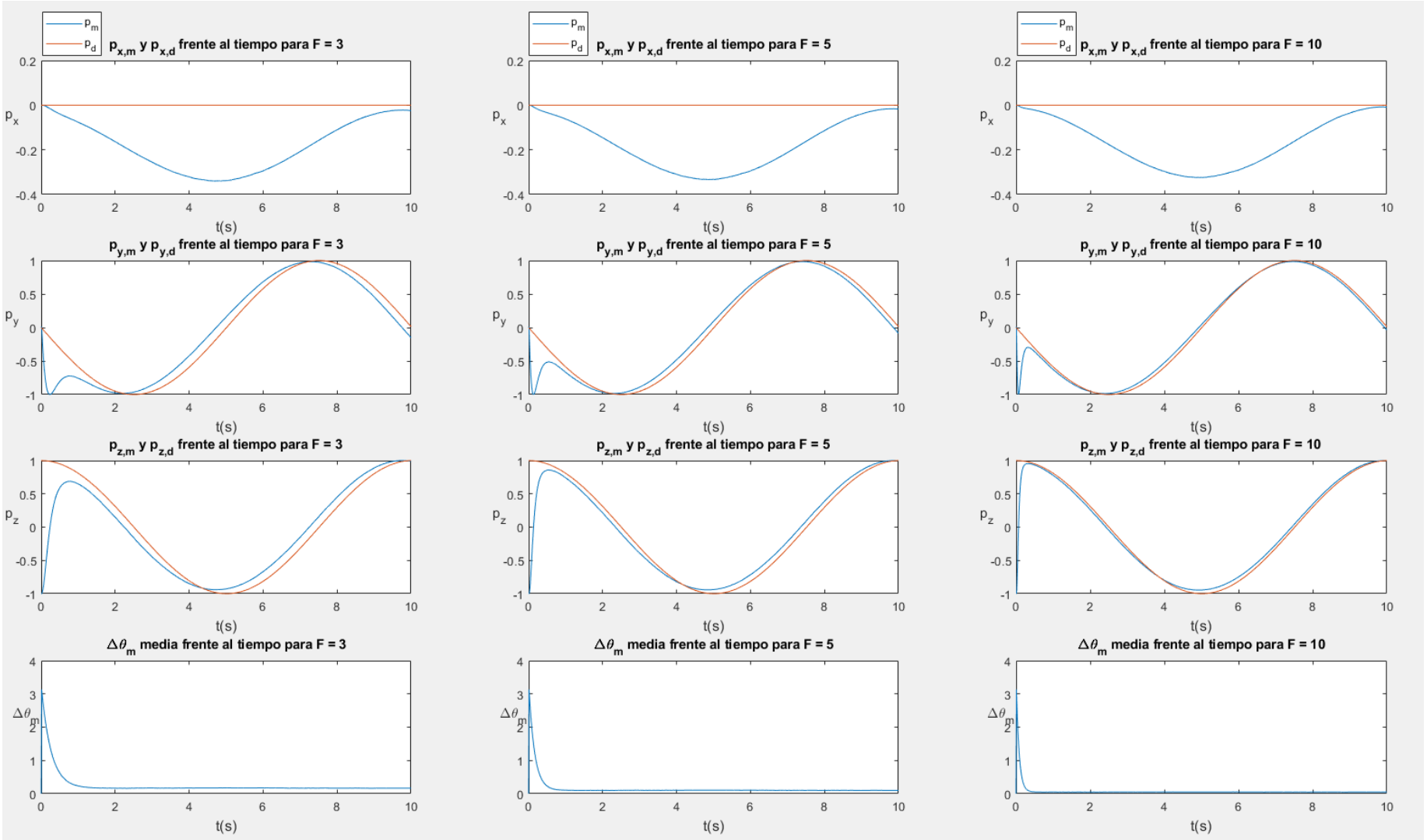

Figura 3.40 Simulación de un Qubit controlado. p_x, p_y, p_z y $\Delta\theta$ para frecuencia $\Omega_{0.5} = \Omega/2$

Para frecuencias lentas, el control actúa como en los casos anteriores. Se puede alcanzar un buen control incluso para $F = 3$. No obstante, existe una diferencia respecto a los casos en los que la frecuencia es más rápida de la natural: podemos observar cómo el valor de p_x alcanza valores muy altos, superiores a $|0.35|$. No obstante, parece que conforme aumenta la fuerza se reduce este valor, pero de un modo muy leve. Este hecho no se produce si estuviésemos ante un Qubit simétrico con $\varepsilon = 0$. Para apreciar cómo influye en frecuencias lentas la asimetría, realizaremos una comparación para $F = 10$.

En la Figura 3.41, se aprecia que la componente p_x se comporta como la deseada, pues su valor ahora es nulo. Además de eso, no se aprecian variaciones significativas, más allá de una leve mejora en las otras dos componentes.

Por último, fijaremos el valor deseado del Qubit a 0 o 1. De este modo, podemos ver si es capaz de frenar el valor monitorizado hasta el punto de presentarse estático en torno a estos puntos. Simularemos para el caso de que deseemos 0, y probaremos con diferentes fuerzas hasta alcanzar un buen control. Los resultados se muestran en la Figura 3.42.

Para los valores de fuerza indicados, se puede comprobar que también se realiza un buen control. En este caso, la coordenada Y parece que es más difícil de controlar que para casos anteriores. No obstante, la diferencia entre el valor deseado y el monitorizado para p_y se reduce considerablemente con el aumento de la fuerza. La coordenada p_x sigue siendo prácticamente imposible controlarla. Por su parte, la coordenada p_z presenta un muy buen control incluso para valores bajos de fuerza.

Como conclusión, podemos decir que, cuanta más diferencia existe respecto a la frecuencia natural, más fuerza necesitaremos para controlar con precisión la fase del Qubit. No obstante, una vez que se alcanza el estado deseado, con un mayor o menor desfase de error, este se mantiene constante, y debido a que nos encontramos trabajando en el campo de la mecánica cuántica, se puede solucionar de un modo relativamente sencillo. Por otro lado, las coordenadas p_y y p_z se controlan siempre correctamente; sin embargo, tal y como está definido el control así como las ecuaciones, la coordenada p_x viene fuertemente determinada por el valor de la asimetría del Qubit ε, y aunque aumentando el valor de la fuerza esta se va aproximando a cero, no es una solución viable.

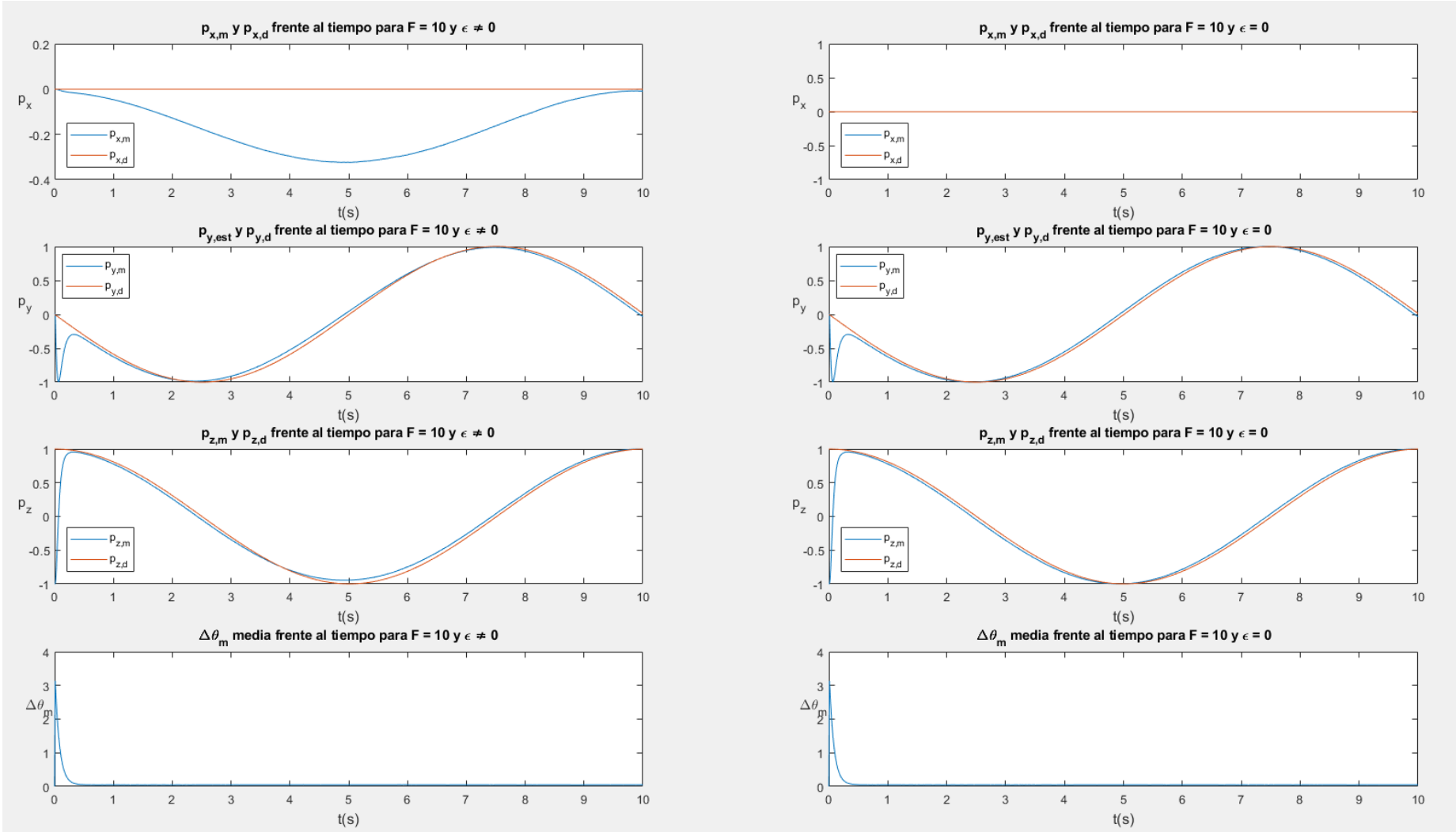

Figura 3.41 Simulación de un Qubit controlado. p_x, p_y, p_z y $\Delta\theta$ para frecuencia $\Omega_{0.5} = \Omega/2$ e influencia de ε

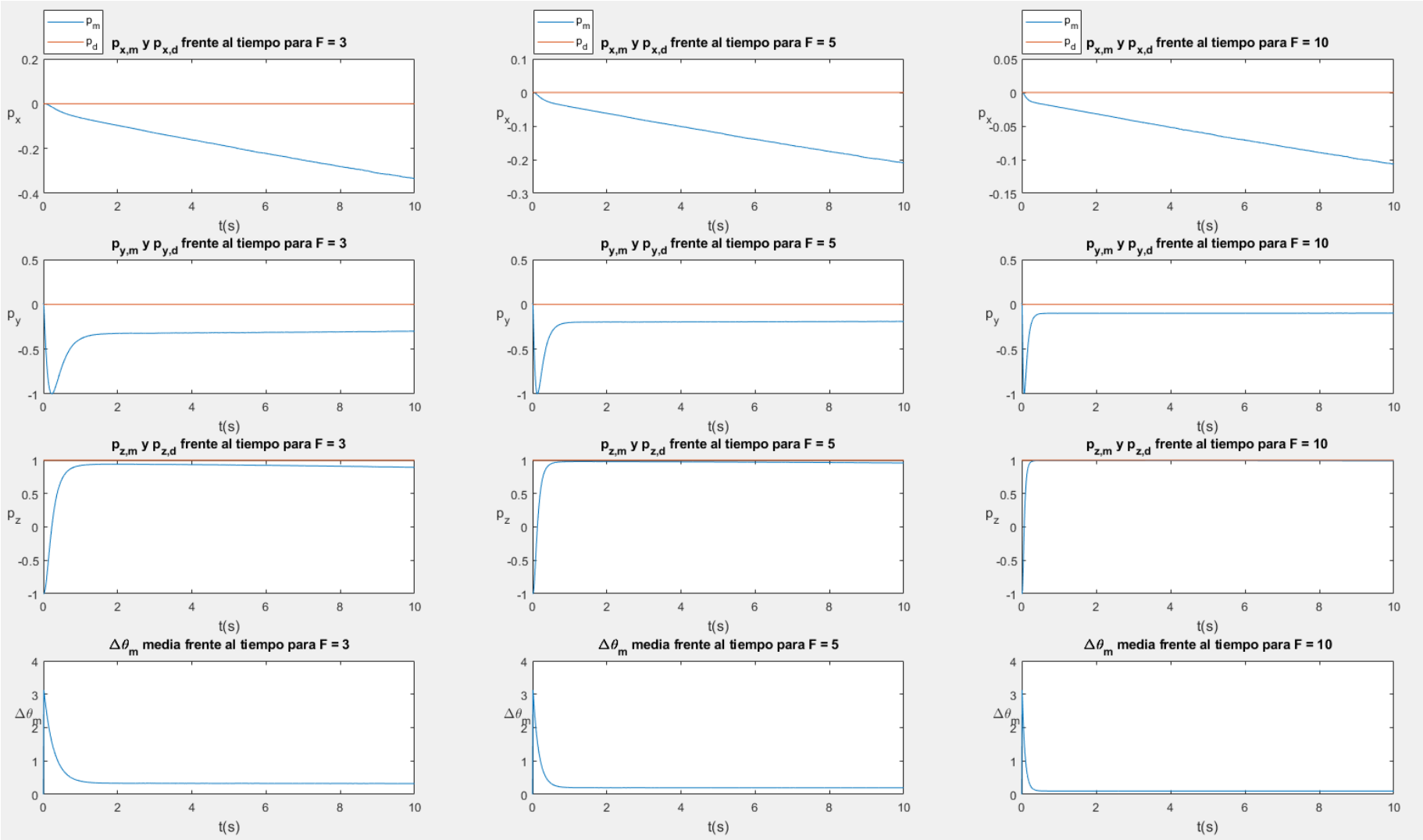

Figura 3.42 Simulación de un Qubit controlado. p_x, p_y, p_z y $\Delta\theta$ para frecuencia $\Omega_0 = 0$ (estado deseado constante $|0\rangle$)

3.8. Control de un Qubit para una única trayectoria

Hemos estado realizando hasta ahora representaciones promediadas. La evolución del Qubit viene dada por una ecuación diferencial estocástica, por lo que presenta un ruido asociado, en este caso procedente del detector. Simularemos ahora la evolución de un único Qubit, viendo cómo se comporta. Lo realizaremos únicamente para la frecuencia natural, condiciones iniciales $\mathbf{p}_0 = (0, 0, -1)$ y $F = 3$.

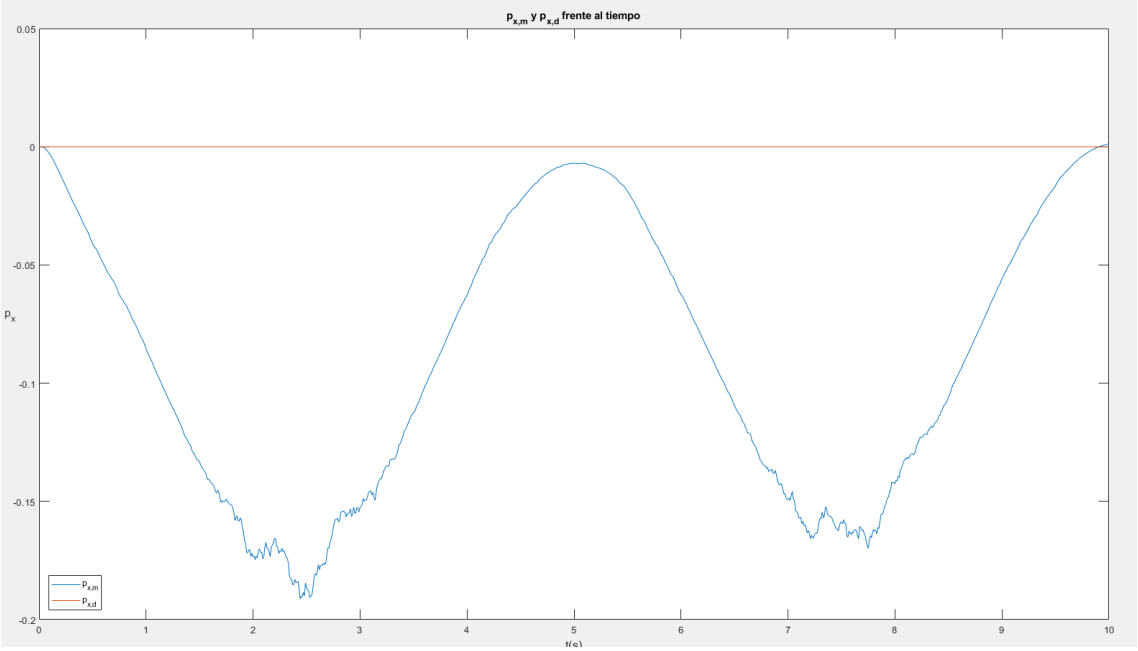

Figura 3.43 Simulación de un Qubit controlado. p_x frente al tiempo para una sola trayectoria

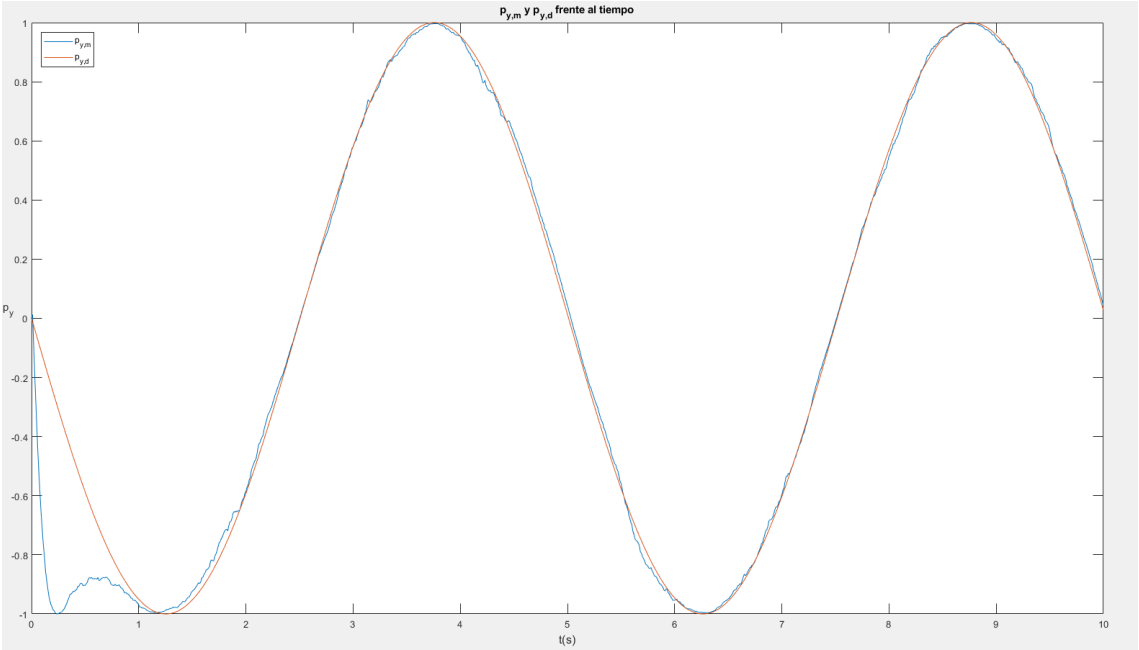

Figura 3.44 Simulación de un Qubit controlado. p_y frente al tiempo para una sola trayectoria

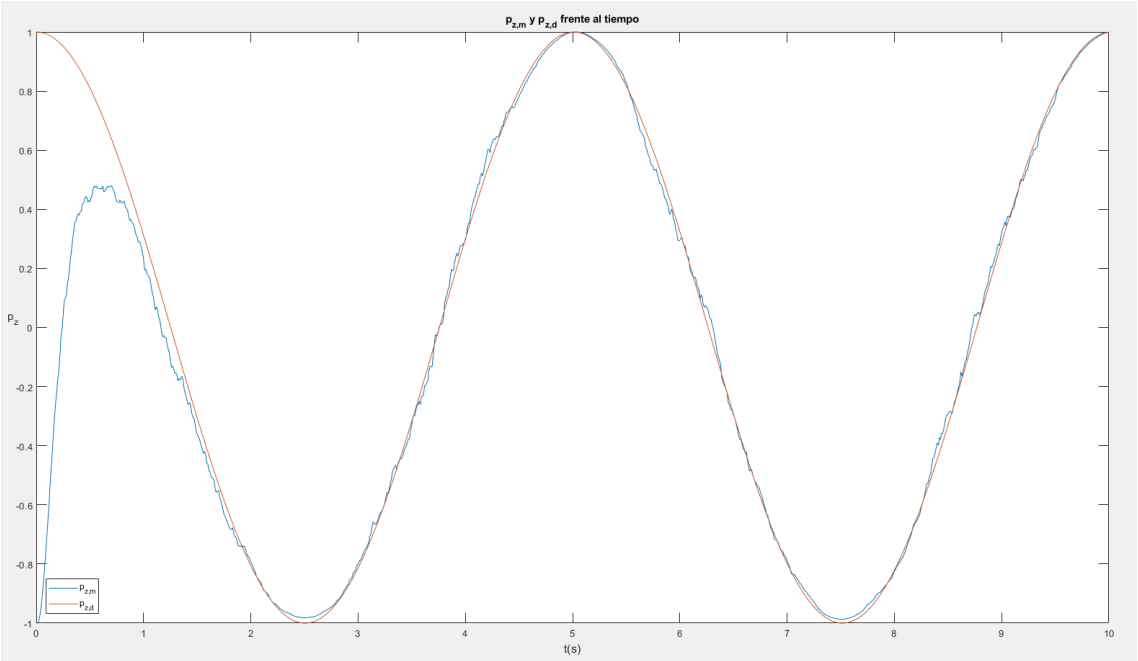

Figura 3.45 Simulación de un Qubit controlado. p_z frente al tiempo para una sola trayectoria

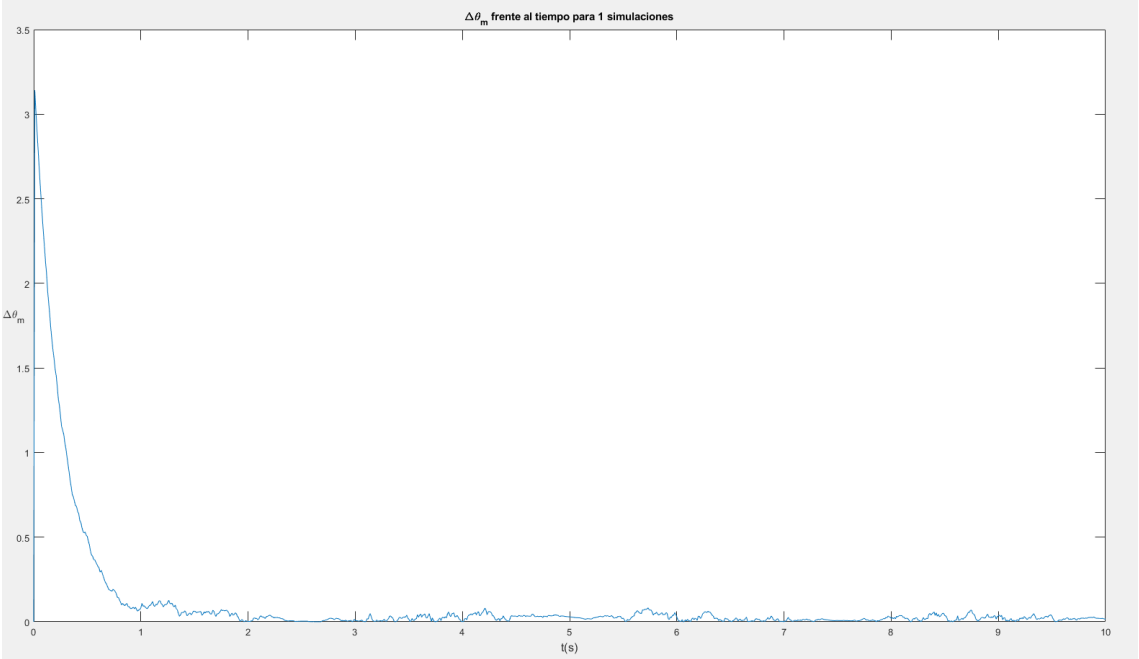

Figura 3.46 Simulación de un Qubit controlado. $\Delta\theta$ frente al tiempo para una sola trayectoria

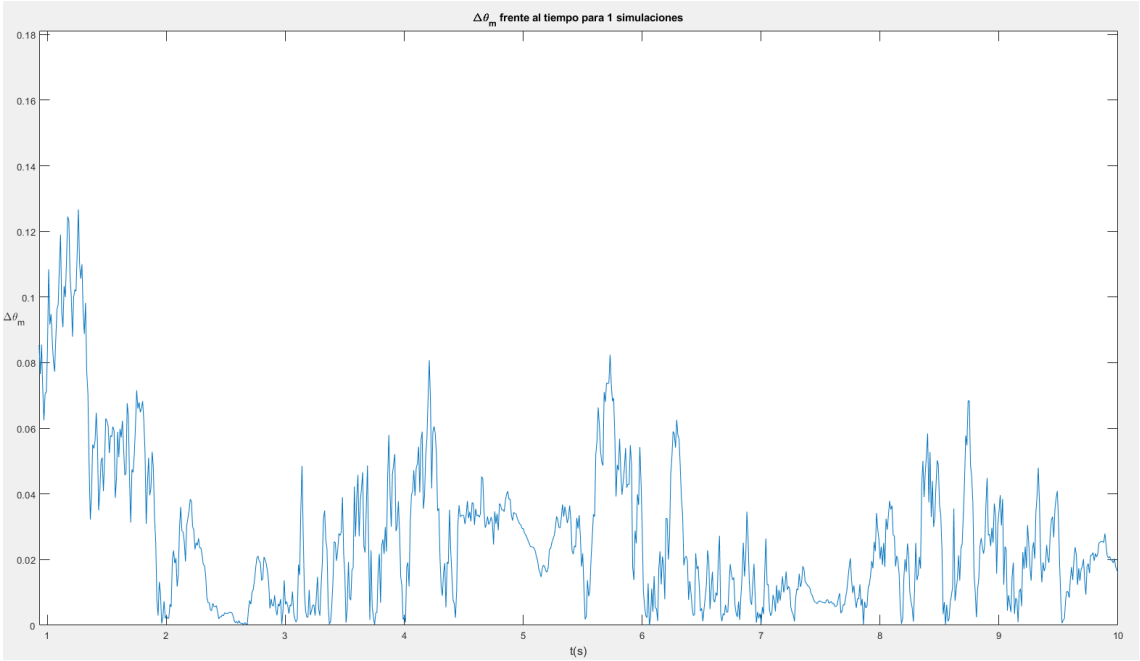

Figura 3.47 Simulación de un Qubit controlado. $\Delta\theta$ frente al tiempo para una sola trayectoria (detalle)

En estas gráficas podemos apreciar claramente ese ruido inherente a la medición. Por lo demás, el Qubit se comporta siguiendo la misma evolución que para los valores promedio, presentando una valor de $\Delta\theta$ que no supera, una vez alcanzado el estado estacionario, el valor de 0.1.

Es necesario mencionar que según la bibliografía, la forma de Itô nos permite promediar con facilidad las ecuaciones del Qubit sobre el proceso del ruido y observar con claridad la evolución del Qubit como si este ruido no existiese. No obstante, para el análisis de una realización individual, las ecuaciones en la forma de Itô son menos transparentes en su interpretación física que las de la forma de Stratonovich. Podemos considerar que hemos obtenido un buen resultado, pero debemos tener esto en cuenta.

3.9. Relación entre acoplamiento y fuerza

El último factor de estudio que analizaremos es la relación entre el acoplamiento y la fuerza. Ya definimos estas dos variables anteriormente. No obstante, hemos estado empleando un valor constante para el acoplamiento; más concretamente, un valor $C = 6.4 \cdot 10^{-4}$. En este apartado estudiaremos la evolución del Qubit y su control para una serie de valores de acoplamientos y fuerzas, y representaremos en una superficie el valor de eficiencia D presentado en (3.13), para el tiempo final de simulación.

Los valores que emplearemos son $C = \{0.01, 0.02, \dots, 0.09, 0.1, 0.2, \dots, 1\}$ y $F = \{0.5, 0.6, \dots, 7\}$ debido a que estamos trabajando con la frecuencia natural y no se precisa de más fuerza. Debido al coste computacional de esta tarea, realizaremos las simulaciones para 100 trayectorias.

Una vez realizadas las simulaciones y calculados todos los valores de $D(T)$, podemos obtener la superficie de la Figura 3.48, donde los valores del acoplamiento se han representado en escala logarítmica para facilitar su lectura. Con ella podemos estudiar algunas características del modelo.

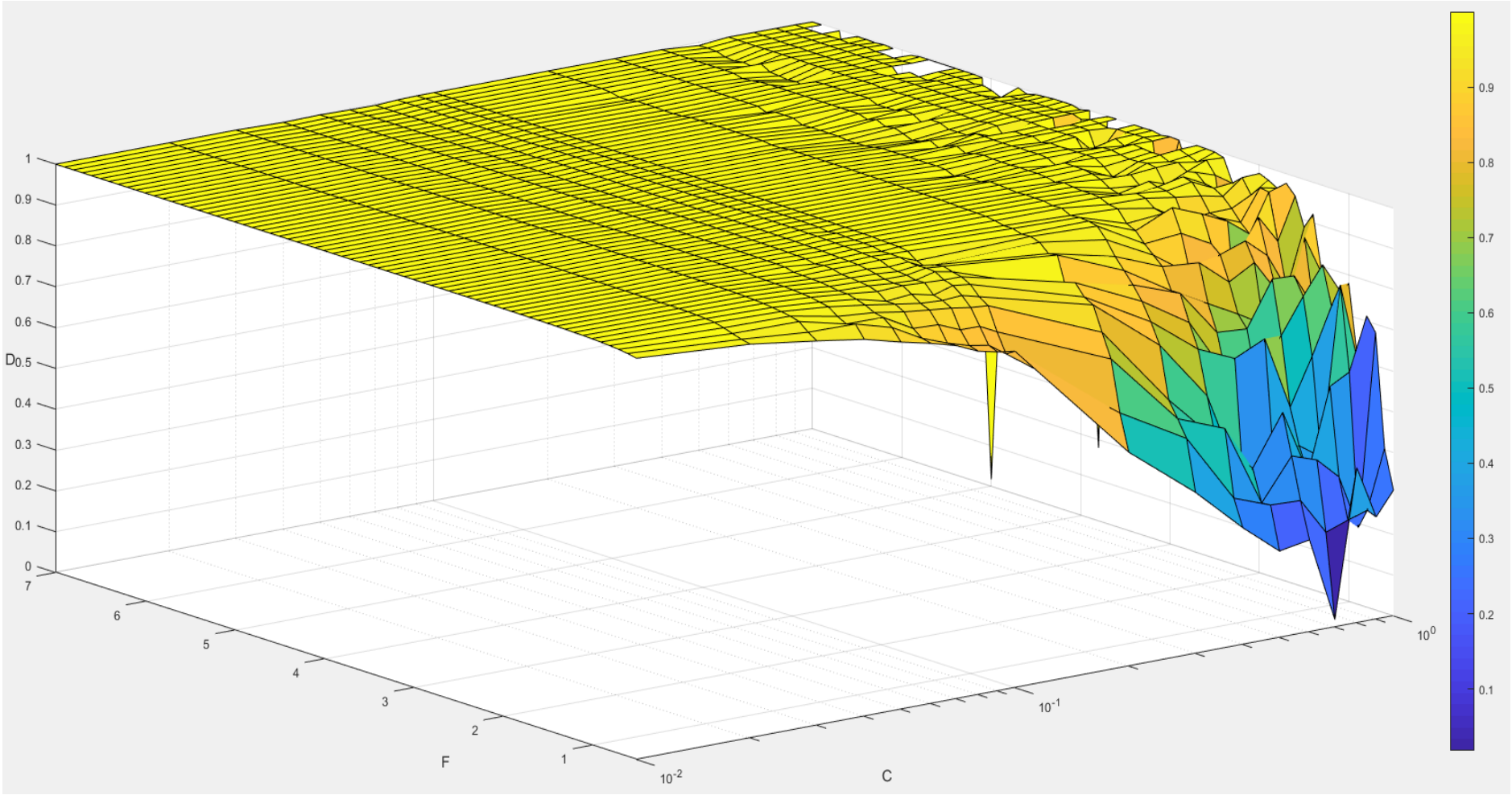

Figura 3.48 Superficie que relaciona acoplamiento C, fuerza F y factor de eficiencia $D(T)$ para un Qubit controlado

En primer lugar, podemos afirmar que cuanto menor es el acoplamiento, mejor es el control. Concretamente, para valores $C < 0.1$, a excepción de los casos de fuerzas pequeñas menores que la unidad, el comportamiento es prácticamente igual en todos los pares de valores fuerza-acoplamiento. Este resultado es esperado puesto que el acoplamiento está definido principalmente por la diferencia de corriente del detector ΔI y la densidad espectral del ruido de la corriente S_0 y, por tanto, cuanto menor sea la relación entre estos dos, menor será el ruido introducido en el sistema.

Estrechamente relacionado con lo anterior: cuanto mayor es la fuerza, mejor es el control. No obstante, es necesario realizar algunos incisos. La fuerza, para valores bajos del acoplamiento, no mejora necesariamente el control. Es decir, para valores bajos de C, se obtiene $D \approx 1$ para prácticamente todo el rango de fuerzas, como ya hemos señalado. No obstante, para valores relativamente altos, la fuerza sí permite realizar un buen control. Como se observa en la gráfica, los valores de alto acoplamiento y poca fuerza derivan en un mal control, con valores de D por debajo de 0.5.

No obstante, debemos tener en cuenta un dato importante. A pesar de que el control sea "mejor", entendiendo como mejor que la diferencia o error entre los estados monitorizado y deseado sea menor, existen multitud de trayectorias que se inestabilizan según los valores de C y F. No obstante, el software de simulación no presenta la característica para indicar automáticamente el número de trayectorias eliminadas por inestabilidad.

Tanto los valores de fuerza como los de acoplamiento influyen en estas inestabilidades. Por un lado, para acoplamientos elevados, el sistema se descontrola completamente. Poniendo de ejemplo el caso $C = 1$ y $F = 3$, podemos comprobar que el número de trayectorias se reduce desde las cien iniciales hasta solamente una. Si representamos las gráficas se puede comprobar que el ruido es tan grande que nos hace imposible decir que se está realizando un buen control. Además, como consecuencia de este elevado ruido, no es posible emplear el valor $D(T)$ como factor de eficiencia representativo puesto que en ese instante final el error está completamente descontrolado, convirtiéndose el azar en el factor más determinante para obtener el valor de D.

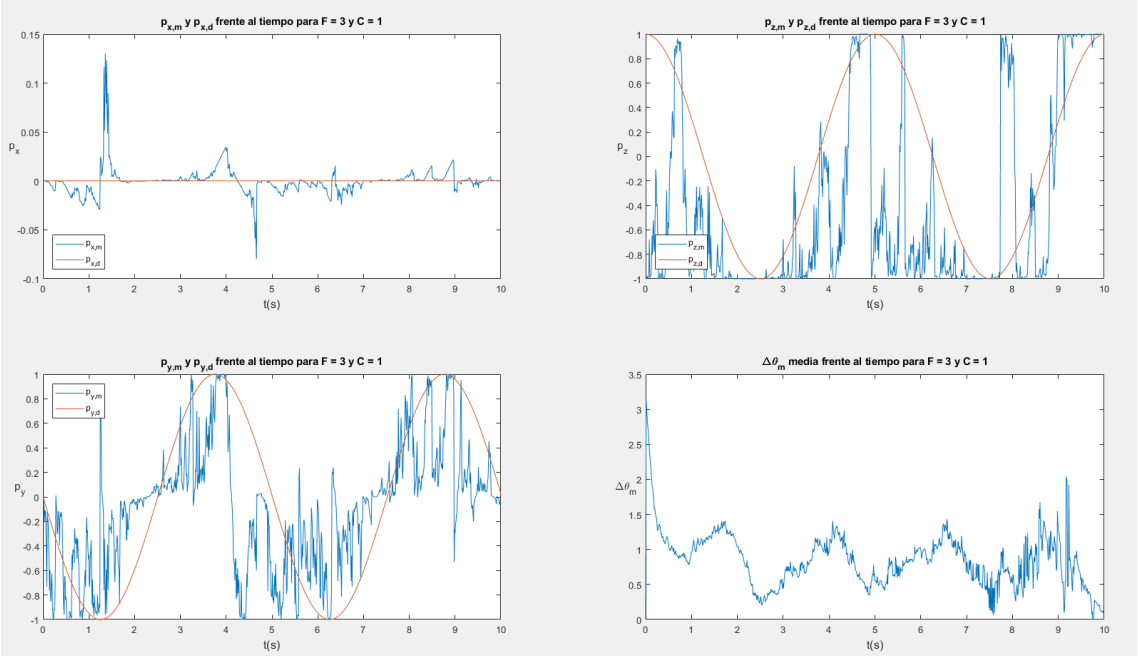

Figura 3.49 Simulación de un Qubit controlado. p_x, p_y, p_z y $\Delta\theta$ con $C = 1$, eliminando trayectorias inestables

En las gráficas se puede observar claramente lo que hemos comentado. Por tanto, esto simplemente nos permite cerciorar que la superficie que se ha realizado no es del todo representativa y confirmar que, al aumentar el acoplamiento, aumenta, de forma general, el número de trayectorias inestables así como el error entre los valores monitorizados y deseados.

En cuanto a los valores de la fuerza, ocurre algo parecido a lo comentado con el acoplamiento. En este caso, para valores de acoplamiento que no inestabilicen completamente la simulación, el aumento de la fuerza provoca un mejor control, pero aumenta también la inestabilidad. Pondremos como ejemplo un valor de acoplamiento medio, como puede ser $C = 0.1$, y dos fuerzas $F = 2$ y $F = 6$.

En la Figura 3.50 se puede observar lo comentado. Cuanto mayor es la fuerza, menor es el valor de $\Delta\theta$ y más cercanas son las trayectorias simuladas a las deseadas. No obstante, cuando miramos el número de trayectorias eliminadas para cada simulación, podemos comprobar que para $F = 2$ se han eliminado, de un total de 100, 25 trayectorias, mientras que para $F = 6$ se han tenido que descartar 33. Una vez más, al igual que ocurrió para el caso del control con frecuencias mayores a la natural, corresponde a quien realiza el control decidir a qué dar prioridad: a la estabilidad del sistema o a la máxima disminución posible del error. No obstante, parece evidente que la primera opción, desde el punto de vista de la ingeniería del control, debe ser prioritaria.

Para finalizar el apartado, indicaremos que, tal y como está definido el modelo, habría que acotar el estudio a $C \leq 0.1$ y $F \leq 5$, siendo la superficie generada la mostrada en la Figura 3.51.

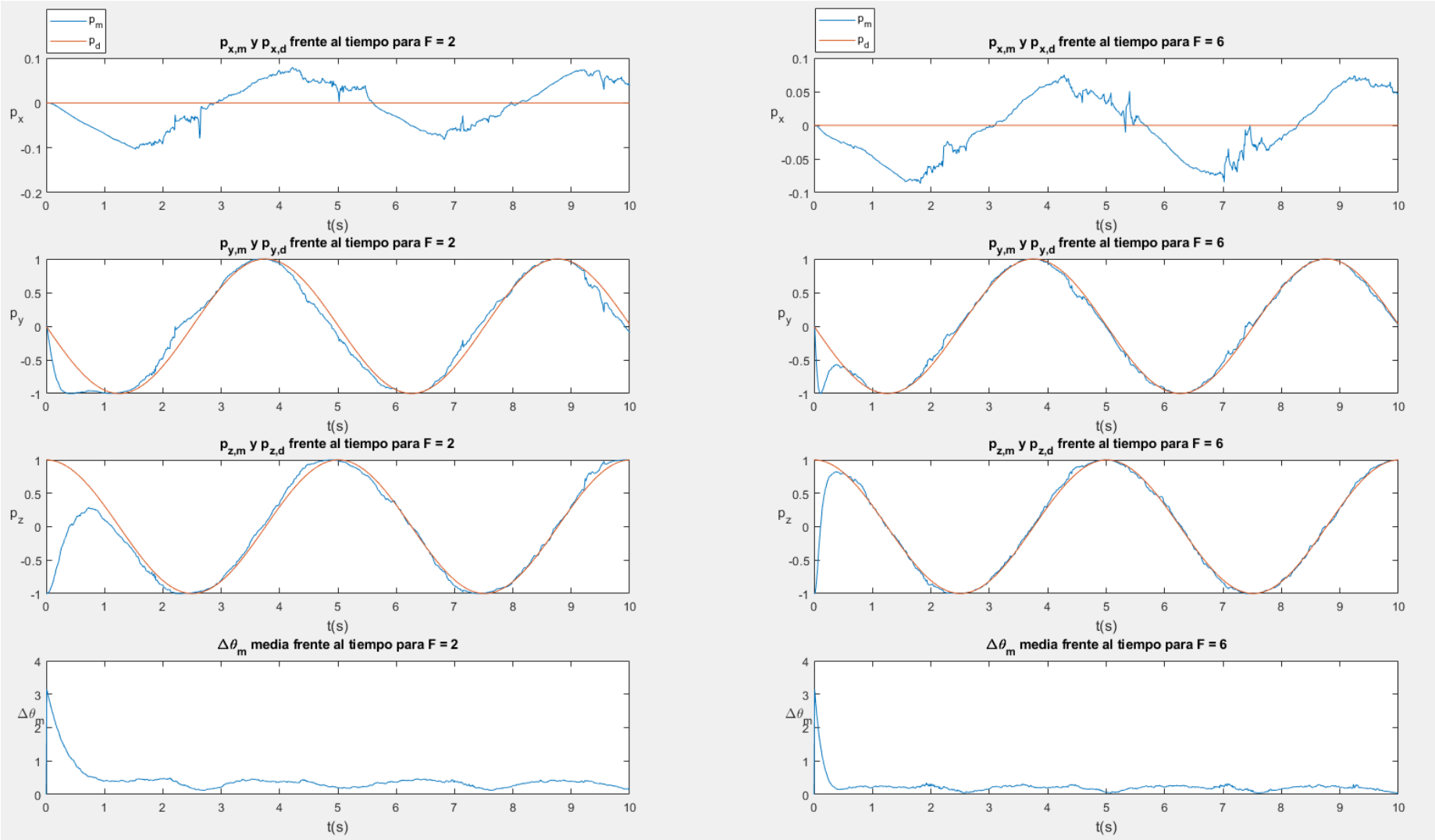

Figura 3.50 Simulación de un Qubit controlado. p_x, p_y, p_z y $\Delta\theta$ con $C = 0.1$, eliminando trayectorias inestable

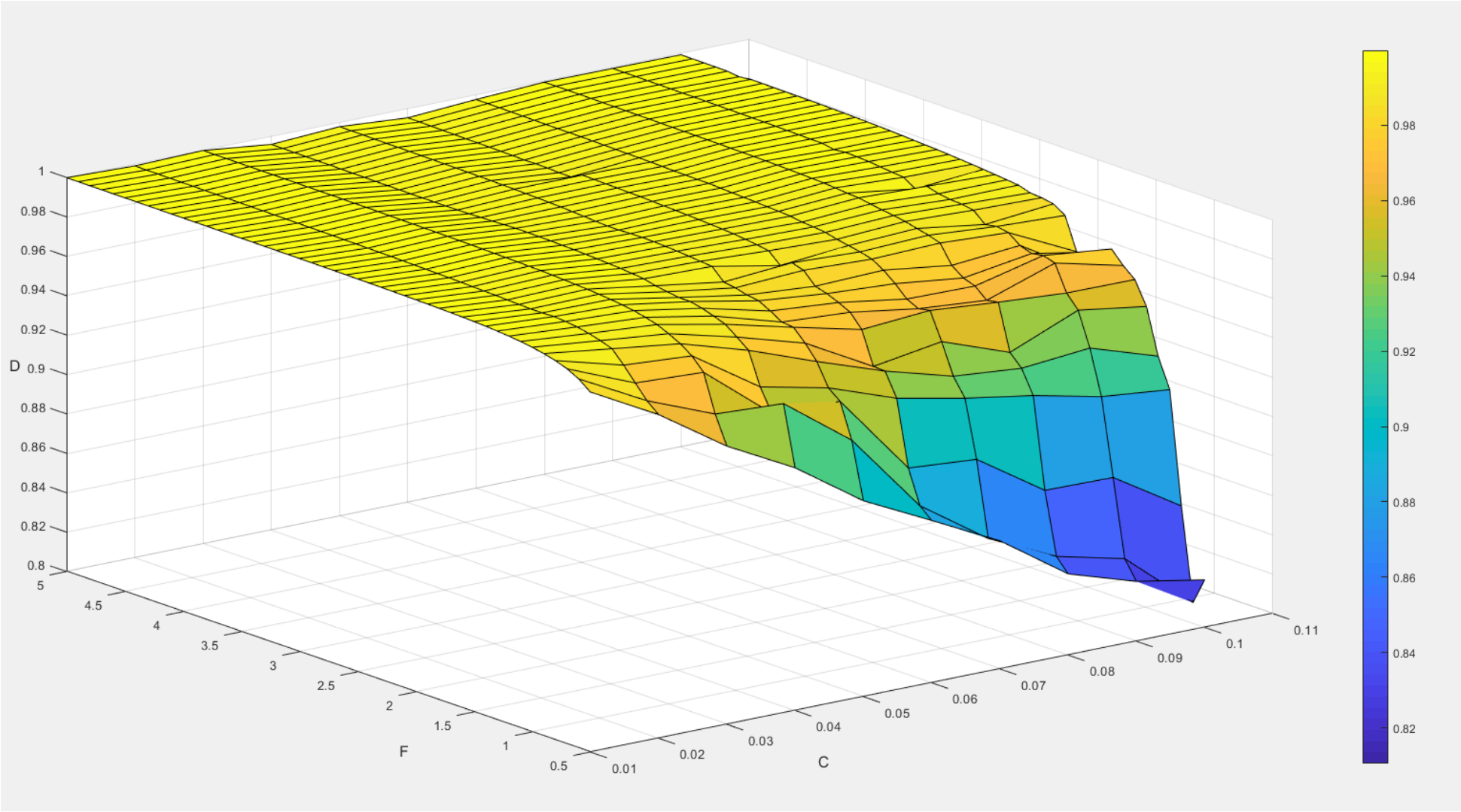

Figura 3.51 Superficie que relaciona acoplamiento C, fuerza F y factor de eficiencia $D(T)$ para un Qubit controlado (detalle)

Conclusiones y trabajos futuros

El objetivo inicial de este trabajo era desarrollar un proyecto de investigación enfocado a la simulación numérica de la dinámica y control de un Qubit. No obstante, a lo largo del mismo han surgido diferentes contratiempos que nos han permitido poder estudiar el sistema con una mayor profundidad.

Antes de comenzar el proyecto, debido al carácter multidisciplinar del control, sumado al campo de trabajo en el que está enmarcado, la mecánica cuántica, fue necesario realizar un estudio inicial de diferentes áreas, entre las que se encuentran el control en espacio de estados [12], una introducción a SDE [4] o el manejo del software de simulación, MATLAB [13, 14].

Una vez adquiridos estos conocimientos, se ha realizado una introducción a las tecnologías cuánticas, donde hemos podido conocer la importancia que están adquiriendo actualmente y la revolución global que van a suponer a corto y largo plazo. En el contexto de una de estas tecnologías, la computación cuántica, se ha introducido el formalismo cuántico, así como las características y operaciones principales del Qubit, comprendiendo algunas de las peculiaridades fundamentales de la mecánica cuántica como son la superposición o la medición de estos sistemas.

En el segundo capítulo hemos estudiado en qué consiste el control cuántico realimentado y cuáles son los parámetros sobre los que se puede trabajar. Después de esto, se ha explicado el modelo de un Qubit mediante el formalismo Bayesiano, que tiene en cuenta la influencia del ruido del detector. Se han introducido las ecuaciones que permiten estudiar su dinámica, así como el control inicial que es utilizado en la bibliografía. En esta sección también se ha introducido el concepto de la esfera de Bloch, una poderosa herramienta que permite visualizar sistemas binarios complejos en un espacio real tridimensional.

Por último, se ha procedido a realizar las simulaciones de la evolución del Qubit. Mediante el software de simulación, se ha podido visualizar esta dinámica y aplicar el control de la bibliografía. No obstante, hemos podido comprobar que no funcionaba correctamente, lo que nos ha llevado a definir un nuevo control. A partir de estas nuevas ecuaciones, se han podido estudiar diferentes situaciones. Se ha comprobado que el control es positivamente funcional ante condiciones iniciales adversas o velocidades frecuencias a la natural.

No obstante, durante el proyecto, se han advertido algunas limitaciones, no solo relacionadas con el software de simulación como ya hemos ido mencionando. El principal trabajo que habría que realizar en un futuro es conseguir implantar un control óptimo que nos permita modificar la trayectoria del Qubit. Es decir, tal y como están definidas actualmente las ecuaciones de la dinámica del Qubit, así como el control, solamente es posible realizar un control a lo largo del plano medio vertical, pero no del horizontal. Este control óptimo concedería una libertad total en torno a la evolución deseada, sin limitarla a un único plano, ya sea sin o con una ligera desviación. Esto consistiría en añadir unos términos u_x, u_y y u_z a las ecuaciones de las componentes del vector de Bloch, es decir, a p_x, p_y y p_z respectivamente. Este era uno de los objetivos iniciales, pero finalmente se ha decidido trabajar en él en un futuro.

Otro trabajo interesante, aunque no prioritario, sería realizar el estudio de dos Qubits que funcionen simultáneamente. En primer lugar, se podría estudiar el caso de que estuviesen desentrelazados, donde se debería comprobar fácilmente que no influye uno sobre el otro. Finalmente, el punto de mayor interés sería ver cómo el entrelazamiento cuántico afecta a las resultados de uno en función de otro.

Apéndice A

Software de simulación y algoritmos empleados

A.1. Método numérico: Euler-Maruyama

La aproximación numérica más sencilla para resolver SDE y que emplearemos en este proyecto, es el método de Euler-Maruyama. Un sistema SDE d-dimensional se puede expresar de un modo formal como:

$$\begin{cases} \dot{X}(t) = f\big(X(t)\big) + g\big(X(t)\big)\xi(t) & (t < 0) \\ X(0) = x_0 \end{cases}, \tag{A.1}$$

donde $f : \mathbb{R}^d \times \Theta \to \mathbb{R}^d$, $g : \mathbb{R}^d \times \Theta \to \mathbb{R}^{d\times m}$ y $\xi(t)$ es m-dimensional. Teniendo en cuenta la relación entre el ruido blando y el proceso de Wiener $\dot{W}(\cdot) = \xi(\cdot)$, y multiplicando (A.1) por dt podemos expresar el sistema anterior de la forma

$$\begin{cases} dX(t) = f\big(X(t)\big)dt + g\big(X(t)\big)dW(t) & (t < 0) \\ X(0) = x_0 \end{cases}. \tag{A.2}$$

Para un tiempo $[t_0, T]$ y una discretización dada $t_0 < t_1 < \cdots < t_n < \cdots < t_N = T$, la aproximación de Euler-Maruyama [14], para un sistema SDE, presenta el siguiente algoritmo (solo disponible para la forma de Itô):

$$y_{n+1}^k = y_n^k + h_n f^k + g^{k,k}\Delta W_n^k, \qquad y_0^k = x_0^k, \qquad n = 0,1,\dots,N-1 \tag{A.3}$$

donde $y_n^k = y^k(t_n), h_n = t_{n+1} - t_n$ es el paso de iteración, $\Delta W_n = W(t_{n+1}) - W(t_n) \sim \mathcal{N}(0, h_n)$ con $W(t_0) = 0$, y $\mathcal{N}$ representa la distribución normal.

Este método presenta un orden de convergencia fuerte de $1/2$ (y orden de convergencia débil de 1), y a diferencia del otro método disponible en el software de simulación (Milstein), no es necesario el término derivativo, que viene dado por $dg(X)/dX$, y dado la naturaleza de nuestras ecuaciones, resulta complicado obtenerlo. Por otro lado, en la versión utilizada del software, se asume que $g(\cdot,\cdot)$ es una matriz diagonal; es decir, se trabaja con un ruido diagonal y, por tanto, $m = d$.

A.2. Software de simulación: SDE Toolbox

SDE Toolbox [14] es un software de libre distribución destinado a la simulación y estimación de SDE con MATLAB. En lo referente a su uso y manejo, basta con definir el sistema a resolver en un archivo de la forma *nombre_sdefile.m*, y realizar la llamada a otra función que resuelva este SDE. El paquete de códigos dispone de herramientas interactivas adaptadas a varios modelos típicos de SDE.

No obstante, para nuestro caso el software presenta algunas limitaciones:

- No posibilidad de control. Este fue el problema principal del empleo del simulador. SDE Toolbox está diseñado para la estimación y simulación de SDE, pero no específicamente para el control de estos sistemas. Por tanto, en las herramientas interactivas mencionadas, no existe una opción para modificar los valores a cada paso: se pide introducir parámetros constantes para todas las trayectorias y pasos de simulación. Es por este motivo, por el que, como se puede comprobar más adelante, insertamos directamente los códigos de las funciones que emplea el programa para nuestro control, modificándolos adecuadamente.
  Además, provoca que para el caso de control, se ejecute el programa tantas veces como número de trayectorias y no una sola vez teniendo en cuenta este parámetro, elevando el tiempo computacional de cálculo.

- Generación de ruidos aleatorios. El software genera ruidos aleatorios mediante la función *randn* para cada fila de la matriz resultado (donde las columnas son las trayectorias y las filas indican el instante de tiempo). Por tanto, crea los ruidos para el paso t_0, a continuación para el paso t_1 y así hasta el paso t_N. Para acelerar el proceso, genera ruido, dentro de un paso t_n, para la primera mitad de trayectorias y, para el resto, empleaba valores antisimétricos a partir de las recién generadas. Sin embargo, nosotros precisamos de un mismo ruido aleatorio para las tres incógnitas, puesto que se debe al detector, el aparato de medida, y por tanto el ruido que introduce en las ecuaciones debe ser el mismo para todas ellas. Esta modificación se resolvió como se explica en el código.

Por tanto, para la simulación y control de los diferentes casos desarrollados a lo largo del proyecto, se emplean códigos distintos. No obstante, mostraremos las partes más importantes, pues de uno a otro no existen modificaciones completas.

A.3. Códigos empleados

Todos los códigos serán adjuntados al final. A continuación únicamente realizaremos una breve explicación de qué realiza cada uno de ellos. También debemos mencionar que no mostraremos todo el código, sino simplemente lo necesario para comprender el funcionamiento del control; omitiremos cosas repetidas, códigos de gráficas, etc.

A.3.1 Definición de SDE en *nombre_sdefile.m*

En estos códigos, como ya se ha comentado, simplemente se definía el sistema a resolver, con los parámetros a introducir y las incógnitas. Debido a que hemos simulado tanto el sistema en función de la matriz densidad como en función del vector de Bloch, disponemos de dos códigos: *Qubit_sdefile.m* y *QubitBloch_sdefile.m*, respectivamente. No obstante, mostraremos únicamente el segundo, pues son análogos.

A.3.2 Simulación de un Qubit sin controlar

Entre la simulación sin y con control, varía una parte del código que ralentiza la ejecución de la segunda respecto a la primera, pero que es necesaria para que, con $F = 0$, se comporten ambos del mismo modo. Este cálculo se puede realizar mediante el código *Qubit_sim.m*, cuando las ecuaciones empleadas son las de la matriz densidad, o con *Qubit_Bloch_sim.m*, cuando se emplean las ecuaciones en base al vector de Bloch. Se recoge en el apéndice el primer código de los mencionados.

A.3.3 Simulación de un Qubit controlado

Este código corresponde al Qubit controlado. Empleamos dos programas para este apartado: *Qubit_sim_control.m* y *Qubit_Bloch_sim_control.m*, según se llame a la función *Qubit_sdefile.m* o *QubitBloch_sdefile.m*, respectivamente. Puesto que el control se realizó finalmente mediante la representación en la esfera de Bloch, escribimos en las siguientes páginas el código *Qubit_Bloch_sim_control.m*. Mostraremos únicamente la sección de código modificada, pues a partir de la sección *Cálculos posteriores* todo se realiza del mismo modo que para el Qubit sin controlar.

A.3.4 Relación entre fuerza y acoplamiento

Simplemente es un código adaptado para el estudio de los parámetros de fuerza y acoplamiento, para que se sucedan los datos contenidos en diferentes vectores y automatizar así los cálculos. Además, se añade el cálculo de D, así como su representación.

A.3.5 Esfera de Bloch

Para representar los componentes en la esfera de Bloch, únicamente hay que ejecutar los códigos que se presenta junto al resto de códigos bajo el nombre *visualizeBlochVect.m.*

QubitBloch_sdefile.m

```
function [out1,out2,out3] = QubitBloch_sdefile(t,x,flag,bigtheta, ...
    SDETYPE,NUMDEPVARS,NUMSIM)

% SDE model definition: drift, diffusion, derivatives and initial
% conditions
%
% [out1,out2,out3] = Qubit_sdefile(t,x,flag,bigtheta,SDETYPE, ...
%    NUMDEPVARS,NUMSIM)
%
% IN:     t; working value of independent variable (time)
%         x; working value of dependent variable
%         flag; a switch, with values 'init' or otherwise
%         bigtheta; complete structural parameter vector
%         SDETYPE; the SDE definition: can be 'Ito' or 'Strat'
% (Stratonovich)
%         NUMDEPVARS; the number of dependent variables, i.e. the SDE
% dimension
%         NUMSIM; the number of desired simulations for the SDE
numerical
% integration
% OUT:    out1; in case of flag='init' is just the initial time,
otherwise
% it is the (vector of) SDE drift(s)
%         out2; in case of flag='init' is the initial value of the
% dependent variables. Otherwise it is the SDE diffusion(s)
%         out3; in case of flag='init' it is nothing. Otherwise it is
the
% SDE's partial derivative(s) of the diffusion term

% Parameters
pxzero       = bigtheta(1);
pyzero       = bigtheta(2);
pzzero       = bigtheta(3);
lambda_h     = bigtheta(4);
deltaI       = bigtheta(5);
S0           = bigtheta(6);
epsilon_h    = bigtheta(7);

if nargin < 3 || isempty(flag)

   xsplitted  =  SDE_split_sdeinput(x,NUMDEPVARS);


%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::
  %::::::::::::::::::::::::  DEFINE HERE THE SDE
::::::::::::::::::::::::::
  %:::::: (define the initial conditions at the bottom of the page)
:::::::

%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::

  % e.g. for a thre-dimensional SDE write X1 = xsplitted{1}; X2 =
  % xsplitted{2}; X3 = xsplitted{3};
  px = xsplitted{1};
```

```
  py = xsplitted{2};
  pz = xsplitted{3};

   switch upper(SDETYPE)
   case 'ITO'
       % the Ito SDE drift w.r.t. X1
       driftpx = epsilon_h * py - ((deltaI^2)/(4 * S0)) * px;
       % the Ito SDE diffusion w.r.t. X1
       diffusionpx = -(deltaI / S0) * px .* pz;
       % the diffusion derivative w.r.t. X1
       derivativepx = [];
       % the Ito SDE drift w.r.t. X2
       driftpy = -(epsilon_h * px + 2 * lambda_h * pz + ((deltaI^2)/(4
...
           * S0)) * py);
       % the Ito SDE diffusion w.r.t. X2
       diffusionpy = -(deltaI / S0) * py .* pz;
       % the diffusion derivative w.r.t. X2
       derivativepy = [];
       % the Ito SDE drift w.r.t. X3
       driftpz = 2 * lambda_h * py;
       % the Ito SDE diffusion w.r.t. X3
       diffusionpz = (deltaI / S0) * (1 - pz.^2);
       % the diffusion derivative w.r.t. X3
       derivativepz = [];
   end

    out1 = zeros(1,NUMDEPVARS*NUMSIM);
    out1(1:NUMDEPVARS:end) = driftpx;
    out1(2:NUMDEPVARS:end) = driftpy;
    out1(3:NUMDEPVARS:end) = driftpz;
    out2 = zeros(1,NUMDEPVARS*NUMSIM);
    out2(1:NUMDEPVARS:end) = diffusionpx;
    out2(2:NUMDEPVARS:end) = diffusionpy;
    out2(3:NUMDEPVARS:end) = diffusionpz;
    out3 = [];
    out3(1:NUMDEPVARS:end) = derivativepx;
    out3(2:NUMDEPVARS:end) = derivativepy;
    out3(3:NUMDEPVARS:end) = derivativepz;


%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::

%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::

%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::

else

    switch(flag)
    case 'init'
        out1 = t;

%::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::::
%:::::::::::::::::  DEFINE HERE THE SDE INITAL CONDITIONS
:::::::::::::::::
```

```
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::::

        % write here the SDE initial condition(s)
        out2 = [pxzero pyzero pzzero];

%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::::
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::::::::::
%:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:::::

        out3 = [];


    otherwise
        error(['Unknown flag ''' flag '''.']);
    end
end
```

Qubit_sim.m

```
%Simulación de un Qubit de estado sólido
clc;clear all;close all;
tamano=get(0,'ScreenSize');     % Establecer tamaño gráficas
%% Parámetros de SDE y paso dt = h

h = 0.01;
var = 1;
S0 = 2 * var^2;
S0_deltaI2 = 2.5e5*h;
deltaI = (S0/(S0_deltaI2))^(1/2);
hp_epsilon = 1e3*h;
hp_g = 1.6e2*h;
omega = (4*(1/hp_g)^2 + (1/hp_epsilon)^2)^(1/2);
rho11zero=1+0j;
rho12zero=0+0j;

%% Definición de datos para el método de resolución numérico
(algoritmo)

% parameters = [rho11zero, rho12zero, rho22zero, lambdafb_hp, deltaI,
S0,
% epsilonfb_hp]
parameters = [rho11zero, rho12zero, 1 - rho11zero, 1/hp_g, deltaI, S0,
...
    1/hp_epsilon];
problem = 'Qubit';
t0 = 0;
T = 10;
numsim = 300;
sdetype = 'Ito';
randseed = 0;
rng(randseed,'v5normal'); % regenerar semilla de números aleatorios
integrator = 'EM';
model = 'Qubit';
numdepvars = 3;
yesdata = 0;
tiempo=t0:h:T;

%% Relación de parámetros definidos con los que trabaja SDE Toolbox

bigtheta=parameters;
PROBLEM=problem;
OWNTIME=tiempo;
NUMDEPVARS=numdepvars;
NUMSIM=numsim;
SDETYPE=sdetype;
SEED=randseed;

%% Ejecución del código SDE_euler.m, con modificación del ruido

N = length(OWNTIME);
handle = waitbar(0,'Computing trajectories...');

[t, xstart] = feval([PROBLEM, '_sdefile'],OWNTIME(1),[],'init', ...
    bigtheta,SDETYPE,NUMDEPVARS,NUMSIM);  % condiciones iniciales

XVARS = zeros(N,NUMSIM*NUMDEPVARS);  % generación de la matriz
resultado
```

```
xstart = xstart';
XVARS(1,:) = xstart([1:size(xstart,1)]' * ones(1,NUMSIM), :)';
% equivalente a XVARS(1,:) = repmat(xstart,1,NUMSIM), pero más rápido;

% Creación de la matriz dW
dW = randn(length(OWNTIME),NUMSIM);
dW(1,:)=0; % En la primera fila se ubican las condiciones iniciales

for j=2:N
    waitbar((j-1)/(N-1));
    % t se define como el tiempo de inicio de este intervalo
    x = XVARS(j-1, :);           % valores XVARS al inicio del
intervalo
    h = OWNTIME(j)- t;           % dt (final - inicial) -> paso

    simulacion=1;
    for indice=1:NUMDEPVARS:NUMSIM*NUMDEPVARS
        Winc(1,indice:indice+(NUMDEPVARS-1))=dW(j,simulacion);
        simulacion=simulacion+1;
    end

    [f,g] = feval([PROBLEM, '_sdefile'], t, x, [], bigtheta,SDETYPE,
...
        NUMDEPVARS,NUMSIM);     % la salida de sdefile

    XVARS(j , :) = x + f * h + g .* Winc ;  % el esquema de Euler-
Maruyama
    % para SDE en la forma de Ito con ruido diagonal

    t = OWNTIME(j);    % t y j referidos al final del intervalo
end

xhat = XVARS;
close(handle);

%% Cálculos posteriores

% Componentes de la matriz densidad
% Para las numsim simulaciones

r11_m=xhat(:,1:NUMDEPVARS:end);
r12_m=xhat(:,2:NUMDEPVARS:end);
r21_m=conj(r12_m);
r22_m=xhat(:,3:NUMDEPVARS:end);

% Valores medios para cada simulación

r11_m_mean=mean(r11_m,2);
r12_m_mean=mean(r12_m,2);
r21_m_mean=mean(r21_m,2);
r22_m_mean=mean(r22_m,2);

%% Valores normalizados

% Para las numsim simulaciones
px_m = r12_m + r21_m;
py_m = 1i*(r12_m - r21_m);
pz_m = r11_m - r22_m;
px_m_norm = randn(length(tiempo),numsim);
```

```
py_m_norm = randn(length(tiempo),numsim);
pz_m_norm = randn(length(tiempo),numsim);

for a=1:numsim
    for b=1:length(tiempo)
        p_m_norm = [px_m(b,a) py_m(b,a) pz_m(b,a)]/norm([px_m(b,a) ...
            py_m(b,a) pz_m(b,a)]);
        px_m_norm(b,a) = p_m_norm(1,1);
        py_m_norm(b,a) = p_m_norm(1,2);
        pz_m_norm(b,a) = p_m_norm(1,3);
    end
end

r11_m_norm = (pz_m_norm + 1) / 2;
r12_m_norm = (px_m_norm - 1i*py_m_norm)/2;
r21_m_norm = conj(r12_m_norm);
r22_m_norm = 1 - r11_m_norm;

% Valores medios para cada simulación
px_m_mean = r12_m_mean + r21_m_mean;
py_m_mean = 1i*(r12_m_mean - r21_m_mean);
pz_m_mean = r11_m_mean - r22_m_mean;

p_m_mean = [px_m_mean py_m_mean pz_m_mean];
p_m_mean_norm = randn(length(tiempo),numdepvars);

for i=1:length(tiempo)
    p_m_mean_norm(i,:) = p_m_mean(i,:)/norm(p_m_mean(i,:));
end

px_m_mean_norm = p_m_mean_norm(:,1);
py_m_mean_norm = p_m_mean_norm(:,2);
pz_m_mean_norm = p_m_mean_norm(:,3);

r11_m_mean_norm = (pz_m_mean_norm + 1) / 2;
r12_m_mean_norm = (px_m_mean_norm - 1i*py_m_mean_norm)/2;
r21_m_mean_norm = conj(r12_m_mean_norm);
r22_m_mean_norm = 1 - r11_m_mean_norm;

%% Corriente detector I(t) - I0
I_I0_simulaciones = (deltaI / 2) * (2 * r11_m - 1) + dW;
I_I0_mean=mean(I_I0_simulaciones,2);

%% Control
% Aunque no se controle el Qubit en este código, puede resultar
interesante
% calcular ciertos parámetros
% lambda_fb = (1 - f * delta_phi_t) * g

% Para las numsim simulaciones
t_numsim = repmat(transpose(tiempo),1,numsim);
% t_numsim solo es necesario para el cálculo
phi_m = atan((2 * imag(r12_m))./(r11_m - r22_m)) + (pi / 2) ...
    * (1 - sign(r11_m - r22_m));
delta_phi = mod(phi_m,2*pi) - mod(omega * t_numsim, 2*pi);
% t es un vector, mientras que modulo_phi es una matriz porque
presenta
% todas las trayectorias, por eso se creó t_numsim
```

```
% Valores medios para cada simulación

phi_m_mean     = mean(phi_m,2);
delta_phi_mean = mean(delta_phi,2);

%% Componentes de la matriz densidad deseada
r11_d = (1 + cos(omega * tiempo)) / 2;
r12_d = (1i * sin(omega * tiempo)) / 2;
r21_d = conj(r12_d);
r22_d = 1/2 - (1/2)*cos(omega*tiempo);

% Trayectoria deseada
px_d = r12_d + r21_d;
py_d = 1i*(r12_d - r21_d);
pz_d = r11_d - r22_d;

p_d = [transpose(px_d) transpose(py_d) transpose(pz_d)];
```

Qubit_Bloch_sim_control.m

```
%Simulación de un Qubit de estado sólido controlado
clc;clear all;close all;
tamano=get(0,'ScreenSize');     % Establecer tamaño gráficas
%% Parámetros de SDE y paso dt = h

h = 0.01;
var = 1;
S0 = 2 * var^2;
S0_deltaI2 = 2.5e5*h;
deltaI = (S0/(S0_deltaI2))^(1/2);
hp_epsilon = 1e3*h;
hp_g = 1.6e2*h;
g_hp = 1/hp_g;
%{
C = 0.0005;
deltaI2_S0 = C / hp_g;
deltaI = (deltaI2_S0 * S0)^(1/2);
%}
omega = (4*(1/hp_g)^2 + (1/hp_epsilon)^2)^(1/2);
pxzero=0+0j;
pyzero=0+0j;
pzzero=-1+0j;
fuerza = 3;
signo = 1;

%% Definición de datos para el método de resolución numérico
(algoritmo)

% parameters = [pxzero, pyzero, pzzero, lambdafb_hp, deltaI, S0,
% epsilonfb_hp]
parameters = [pxzero, pyzero, pzzero, g_hp, deltaI, S0, 1/hp_epsilon];
problem = 'QubitBloch';
t0 = 0;
T = 10;
numsim = 300;
sdetype = 'Ito';
randseed = 0;
rng(randseed,'v5normal'); % regenerar semilla de números aleatorios
integrator = 'EM';
model = 'QubitBloch';
numdepvars = 3;
yesdata = 0;
tiempo=t0:h:T;

%% Trayectoria deseada

r11_d = (1 + cos(omega * tiempo)) / 2;
r12_d = (1i * sin(omega * tiempo)) / 2;
r21_d = conj(r12_d);
r22_d = 1/2 - (1/2)*cos(omega*tiempo);

px_d = r12_d + r21_d;
py_d = 1i*(r12_d - r21_d);
pz_d = r11_d - r22_d;

p_d = [transpose(px_d) transpose(py_d) transpose(pz_d)];
p_d_control = [transpose(py_d) transpose(pz_d)];
p_d_cross = [transpose(px_d)*0 transpose(py_d) transpose(pz_d)];
```

```
%% Relación de parámetros definidos con los que trabaja SDE Toolbox

bigtheta=parameters;
PROBLEM=problem;
OWNTIME=tiempo;
NUMDEPVARS=numdepvars;
NUMSIM=1;
SDETYPE=sdetype;
SEED=randseed;
columna_px=1;
columna_py=columna_px+1;
columna_pz=NUMDEPVARS;

%% Ejecución del código SDE_euler.m, con modificación del ruido

% Para realizar control, es necesario realizar las trayectorias una a
una
% para poder modificar lambda

% Definición de los tamaños de las matrices para reducir tiempos
PROD_ESC_VALORES = zeros(length(OWNTIME),numsim);
COSENO_VALORES = zeros(length(OWNTIME),numsim);
PROD_CROSS_VALORES = zeros(length(OWNTIME),numsim);
SENO_VALORES = zeros(length(OWNTIME),numsim);
DELTA_VALORES = zeros(length(OWNTIME),numsim);
LAMBDAFB_HP_VALORES = zeros(length(OWNTIME),numsim);
Winc=zeros(1,NUMDEPVARS);

% Creación de la matriz dW
dW = randn(length(OWNTIME),numsim);
dW(1,:)=0; % En la primera fila se ubican las condiciones iniciales

for simulacion=1:numsim

    N = length(OWNTIME);
    handle = waitbar(0,'Computing trajectories...');
    [t, xstart] = feval([PROBLEM, '_sdefile'],OWNTIME(1),[],'init',
...
        bigtheta,SDETYPE,NUMDEPVARS,NUMSIM);  % condiciones iniciales
    XVARS = zeros(N,NUMSIM*NUMDEPVARS);  % generación de matriz
resultado
    xstart = xstart';
    XVARS(1,:) = xstart(1:size(xstart,1)' * ones(1,NUMSIM), :)';
    % equivalente a XVARS(1,:) = repmat(xstart,1,NUMSIM), pero más
rápido;

    for j=2:N

        waitbar((j-1)/(N-1));
        % t se define como el tiempo de inicio de este intervalo
        x = XVARS(j-1, :);      % valores XVARS al inicio del
intervalo
        h = OWNTIME(j)- t;      % dt (final - inicial) -> paso

        [f,g] = feval([PROBLEM, '_sdefile'], t, x, [], bigtheta, ...
            SDETYPE,NUMDEPVARS,NUMSIM);     % la salida de sdefile

        Winc(1,:)=dW(j,simulacion);
```

```
        XVARS(j , :) = x + f * h + g .* Winc ;  % el esquema de
        % Euler-Maruyama para SDE en la forma de Ito con ruido
diagonal

        t = OWNTIME(j);    % t y j referidos al final del intervalo

        % Cálculos del control

        % Valores del paso actual

        px_m_t = XVARS(j,1);
        py_m_t = XVARS(j,2);
        pz_m_t = XVARS(j,3);

        p_m_control = [py_m_t pz_m_t];
        p_m_cross = [0 py_m_t pz_m_t];

        % Operaciones

        prod_esc = dot(p_d_control(j,:),p_m_control);
        coseno = prod_esc/(norm(p_d_control(j,:))*norm(p_m_control));

        delta_theta = acos(coseno);

        prod_cross = cross(p_d_cross(j,:),p_m_cross);
        seno = prod_cross(1)/(norm(p_d_cross(j,:))*norm(p_m_cross));
        % seno calculado con primer elemento de prod_cross para
mantener la
        % información aportada por el signo

        if coseno>0 && seno<0
            signo=-1;
        end

        lambdafb_hp = (1 - signo*fuerza*delta_theta)*g_hp;
        signo=1;

        PROD_ESC_VALORES(j,simulacion)=prod_esc;
        COSENO_VALORES(j,simulacion)=coseno;
        PROD_CROSS_VALORES(j,simulacion)=prod_cross(1);
        SENO_VALORES(j,simulacion)=seno;
        DELTA_VALORES(j,simulacion)=delta_theta;
        LAMBDAFB_HP_VALORES(j,simulacion)=lambdafb_hp;
        bigtheta(4) = lambdafb_hp;

    end

    resultados = XVARS;
    xhat(:,columna_px:columna_pz)=repmat(resultados,1,1);
    columna_px=columna_px+NUMDEPVARS;
    columna_py=columna_py+NUMDEPVARS;
    columna_pz=columna_pz+NUMDEPVARS;
    bigtheta(4)=g_hp;
    h=0.01;
    close(handle);

end
```

Qubit_Bloch_sim_control_coupling.m

```
%Simulación de un Qubit de estado sólido controlado
clc;clear all;close all;
tamano=get(0,'ScreenSize');     % Establecer tamaño gráficas
%% Parámetros de SDE y paso dt = h

h = 0.01;
var = 1;
S0 = 2 * var^2;
hp_epsilon = 1e3*h;
hp_g = 1.6e2*h;
g_hp = 1/hp_g;

C_values = 0.005:0.005:0.01;
deltaI2_S0_values = C_values / hp_g;
deltaI_values = (deltaI2_S0_values * S0).^(1/2);

omega = ((4*(1/hp_g)^2 + (1/hp_epsilon)^2)^(1/2));
pxzero=0+0j;
pyzero=0+0j;
pzzero=-1+0j;

fuerza_values = 0.5:0.1:0.6;

signo = 1;

%% Definición de datos para el método de resolución numérico
(algoritmo)

% parameters = [pxzero, pyzero, pzzero, lambdafb_hp, deltaI, S0,
% epsilonfb_hp]
parameters = [pxzero, pyzero, pzzero, g_hp, 0, S0, 1/hp_epsilon];
problem = 'QubitBloch';
t0 = 0;
T = 10;
numsim = 300;
sdetype = 'Ito';
randseed = 0;
rng(randseed,'v5normal'); % regenerar semilla de números aleatorios
integrator = 'EM';
model = 'QubitBloch';
numdepvars = 3;
yesdata = 0;
tiempo=t0:h:T;

%% Definición de los tamaños de las matrices (4-dimensional)

% Defino los tamaños de las matrices para optimizar el código
px_m =
zeros(length(tiempo),numsim,length(C_values),length(fuerza_values));
py_m =
zeros(length(tiempo),numsim,length(C_values),length(fuerza_values));
pz_m =
zeros(length(tiempo),numsim,length(C_values),length(fuerza_values));

PROD_ESC_VALORES = zeros(length(tiempo),numsim,length(C_values), ...
    length(fuerza_values));
COSENO_VALORES = zeros(length(tiempo),numsim,length(C_values), ...
```

```
    length(fuerza_values));
PROD_CROSS_VALORES = zeros(length(tiempo),numsim,length(C_values), ...
    length(fuerza_values));
SENO_VALORES = zeros(length(tiempo),numsim,length(C_values), ...
    length(fuerza_values));
DELTA_VALORES = zeros(length(tiempo),numsim,length(C_values), ...
    length(fuerza_values));
LAMBDAFB_HP_VALORES = zeros(length(tiempo),numsim,length(C_values),
...
    length(fuerza_values));
%Winc=zeros(1,NUMDEPVARS);

% Creación de la matriz dW
dW = randn(length(tiempo),numsim);
dW(1,:)=0; % En la primera fila se ubican las condiciones iniciales

%% Trayectoria deseada

r11_d = (1 + cos(omega * tiempo)) / 2;
r12_d = (1i * sin(omega * tiempo)) / 2;
r21_d = conj(r12_d);
r22_d = 1/2 - (1/2)*cos(omega*tiempo);

px_d = r12_d + r21_d;
py_d = 1i*(r12_d - r21_d);
pz_d = r11_d - r22_d;

p_d = [transpose(px_d) transpose(py_d) transpose(pz_d)];
p_d_control = [transpose(py_d) transpose(pz_d)];
p_d_cross = [transpose(px_d)*0 transpose(py_d) transpose(pz_d)];

%% Loop 4D

for indice_C=1:length(C_values)
    parameters(5)=deltaI_values(indice_C);

    for indice_fuerza=1:length(fuerza_values)
        fuerza=fuerza_values(indice_fuerza);

        % Relación de parámetros definidos con los que trabaja SDE
Toolbox
        bigtheta=parameters;
        PROBLEM=problem;
        OWNTIME=tiempo;
        NUMDEPVARS=numdepvars;
        NUMSIM=1;
        SDETYPE=sdetype;
        SEED=randseed;
        columna_px=1;
        columna_py=columna_px+1;
        columna_pz=NUMDEPVARS;

        Winc=zeros(1,NUMDEPVARS);

        % Ejecución del código SDE_euler.m, con modificación del ruido

        % Para realizar control, es necesario realizar las
trayectorias una
        % a una para poder modificar lambda
```

```
        for simulacion=1:numsim

            N = length(OWNTIME);
            handle = waitbar(0,'Computing trajectories...');
            [t, xstart] = feval([PROBLEM, '_sdefile'],OWNTIME(1),[],
...
                'init',bigtheta,SDETYPE,NUMDEPVARS,NUMSIM);  % CI
            XVARS = zeros(N,NUMSIM*NUMDEPVARS);  % generar matriz
resultado
            xstart = xstart';
            XVARS(1,:) = xstart(1:size(xstart,1)' * ones(1,NUMSIM),
:)';
            % equivalente a XVARS(1,:)=repmat(xstart,1,NUMSIM), más
rápido

            for j=2:N

                waitbar((j-1)/(N-1));
                % t se define como el tiempo de inicio de este
intervalo
                x = XVARS(j-1, :);      % valores XVARS inicio de
intervalo
                h = OWNTIME(j)- t;      % dt (final - inicial) -> paso

                [f,g] = feval([PROBLEM, '_sdefile'], t, x, [],
bigtheta,...
                    SDETYPE,NUMDEPVARS,NUMSIM);     % la salida de
sdefile

                Winc(1,:)=dW(j,simulacion);

                XVARS(j , :) = x + f * h + g .* Winc ;  % el esquema
de
                % Euler-Maruyama para SDE en la forma de Ito ruido
diagonal

                t = OWNTIME(j);    % t y j referidos al final del
intervalo

                % Cálculos del control

                % Valores del paso actual

                px_m_t = XVARS(j,1);
                py_m_t = XVARS(j,2);
                pz_m_t = XVARS(j,3);

                p_m_control = [py_m_t pz_m_t];
                p_m_cross = [0 py_m_t pz_m_t];

                % Operaciones

                prod_esc = dot(p_d_control(j,:),p_m_control);
                coseno = prod_esc/(norm(p_d_control(j,:))* ...
                    norm(p_m_control));

                delta = acos(coseno);
```

```
                prod_cross = cross(p_d_cross(j,:),p_m_cross);
                seno = prod_cross(1)/(norm(p_d_cross(j,:))* ...
                    norm(p_m_cross));

                if coseno>0 && seno<0
                    signo=-1;
                end

                lambdafb_hp = (1 - signo*fuerza*delta)*g_hp;
                signo=1;

                PROD_ESC_VALORES(j,simulacion,indice_C,indice_fuerza)
= ...
                    prod_esc;
                COSENO_VALORES(j,simulacion,indice_C,indice_fuerza) =
...
                    coseno;

PROD_CROSS_VALORES(j,simulacion,indice_C,indice_fuerza)=...
                    prod_cross(1);

SENO_VALORES(j,simulacion,indice_C,indice_fuerza)=seno;

DELTA_VALORES(j,simulacion,indice_C,indice_fuerza)=delta;

LAMBDAFB_HP_VALORES(j,simulacion,indice_C,indice_fuerza)...
                    =lambdafb_hp;

                bigtheta(4) = lambdafb_hp;

            end

            resultados = XVARS;
            xhat(:,columna_px:columna_pz)=repmat(resultados,1,1);
            columna_px=columna_px+NUMDEPVARS;
            columna_py=columna_py+NUMDEPVARS;
            columna_pz=columna_pz+NUMDEPVARS;
            bigtheta(4)=g_hp;
            h=0.01;
            close(handle);

        end

    % Cálculos posteriores

    % Componentes del vector de Bloch
    % Para las numsim simulaciones

    px_m(:,:,indice_C,indice_fuerza)=xhat(:,1:NUMDEPVARS:end);
    py_m(:,:,indice_C,indice_fuerza)=xhat(:,2:NUMDEPVARS:end);
    pz_m(:,:,indice_C,indice_fuerza)=xhat(:,3:NUMDEPVARS:end);

    % Valores medios para cada simulación

    px_m_mean=mean(px_m,2,'omitnan');
    py_m_mean=mean(py_m,2,'omitnan');
    pz_m_mean=mean(pz_m,2,'omitnan');
```

```
    end
end

%% Cálculo de D y conversiones necesarias

delta_mean = mean(DELTA_VALORES,2,'omitnan');

D_4D = cos(delta_mean);
D = zeros(length(C_values),length(fuerza_values));

for indice_C=1:length(C_values)
    for indice_fuerza=1:length(fuerza_values)
        D(indice_C,indice_fuerza)=D_4D(length(D_4D),1,indice_C, ...
            indice_fuerza);
    end
end

C_matriz = repmat(transpose(C_values),1,length(fuerza_values));
fuerza_matriz = repmat(fuerza_values,length(C_values),1);

figure
surf(C_matriz,fuerza_matriz,D)
```

visualizeBlochVect.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', 15, 'HorizontalAlignment', 'Center')
text(0, 0, -1.1, '$\left| 1 \right>$', 'Interpreter', 'latex',...
    'FontSize', 15, 'HorizontalAlignment', 'Center')
text(1.1, 0, 0, '$\left| + \right>$', 'Interpreter', 'latex',...
    'FontSize', 15, 'HorizontalAlignment', 'Center')
text(-1.1, 0, 0, '$\left| - \right>$', 'Interpreter', 'latex',...
    'FontSize', 15, 'HorizontalAlignment', 'Center')
view([60 15])
%view([90 0])    % ángulo del plano medio

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')

for i=1:length(tiempo)

    bloch_des = line( [0 p_d(i,1)], [0 p_d(i,2)], [0 p_d(i,3)], ...
        'LineWidth', 2, 'Marker', 'o', 'Color', 'b')
    drawnow

    %F(i) = getframe(gcf);

    delete(bloch_est);

    bloch_est = line( [0 p_m_mean_norm(i,1)], [0 p_m_mean_norm(i,2)],
...
        [0 p_m_mean_norm(i,3)],'LineWidth', 2, 'Marker', 'o', 'Color',
'r')
    drawnow

    delete(bloch_des);
end

%{
video = VideoWriter('Bloch');
open(video)
writeVideo(video, F);
close(video)
%}

delete(bloch_est);
```

Bibliografía

[1] Rubio-Manzanares, A., Arranz, U., Gaspar, V., & Romero, J. T. (2019). *La España cuántica: Una aproximación empresarial*.

[2] Mermin, N. David (2006). Lecture notes on Quantum Computation. *Fundamental Properties of Cbits and Qbits*. Ithaca (New York): Cornell University. Disponible en: http://www.lassp.cornell.edu/mermin/qcomp/chap1.pdf

[3] Jacobs, K., & Shabani, A. (2008). Quantum feedback control: How to use verification theorems and viscosity solutions to find optimal protocols. *Contemporary Physics*, *49*(6), 435–448.

[4] Evans, L. (2014). An Introduction to Stochastic Differential Equations. In American Mathematical Society (Ed.), *An Introduction to Stochastic Differential Equations.*

[5] Jacobs, K., & Steck, D. A. (2006). A straightforward introduction to continuous quantum measurement. *Contemporary Physics*, *47*(5), 279–303.

[6] Zhang, Q., Ruskov, R., & Korotkov, A. N. (2005). Continuous quantum feedback of coherent oscillations in a solid-state qubit. *Physical Review B - Condensed Matter and Materials Physics*, *72*(24).

[7] Korotkov, A. N. (2000). Selective quantum evolution of a qubit state due to continuous measurement. *Physical Review B - Condensed Matter and Materials Physics*, *63*(11).

[8] Glendinning, I. (2005). *European Centre for Parallel Computing at Vienna. The Bloch Sphere*. Disponible en: http://www.vcpc.univie.ac.at/~ian/hotlist/qc/talks/bloch-sphere.pdf

[9] Nölle, Michael & Omer, Bayan. (2006). Representation of Cyclic Colour Spaces within Quantum Space.

[10] Korch Research Group. (2013). Theory of Open Quantum Systems. Northwestern University. Disponible en: https://sites.northwestern.edu/koch/teaching/

[11] Alonso, J. J., Lutz, E., & Romito, A. (2016). Thermodynamics of Weakly Measured Quantum Systems. *Physical Review Letters*, *116*(8), 1–6.

[12] Gil Nobajas, J. J., & Rubio Díaz-Cordovés, Á. (2010). *Fundamentos de Control Automáticos de Sistemas Continuos y Muestreados*. Unicopia, C.B, 127–135.

[13] MathWorks. "MATLAB - El lenguaje del cálculo técnico", e*s.mathworks.com*, 2019. [Online]. Disponible en: https://es.mathworks.com/products/matlab.html?s_tid=hp_products_matlab.

[14] Picchini, U. (2007). "SDE Toolbox: Stochastic Differential Equations with MATLAB", *sdetoolbox.sourceforge.net*, 2007. [Online]. Disponible en: http://sdetoolbox.sourceforge.net.