Universidad
Rey Juan Carlos

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

Curso Académico 2009/2010

Proyecto de Fin de Carrera

SIMULACIÓN MEDIANTE APPLETS DE JAVA DE SISTEMAS DINÁMICOS EXCITABLES

Autor: Carlos Sánchez Vega

Tutores: Inés Pérez Mariño
Jesús M. Seoane Sepúlveda

Agradecimientos

Me gustaría agradecer la ayuda prestada por los tutores que tuve en mi proyecto, los Profesores Inés Pérez y Jesús M. Seoane, del Departamento de Física de la Universidad Rey Juan Carlos, por el tiempo y ayuda prestados durante las tutorías que necesité para desarrollar el proyecto.

Por otro lado, me gustaría agradecer a mi familia el apoyo mostrado durante el desarrollo del proyecto de final de carrera, pues me ayudaron y animaron a continuar en los momentos en los que me encontré desalentado.

Resumen

El proyecto desarrollado permite la simulación del sistema dinámico de FitzHugh-Nagumo (FHN) que estudia la dinámica de una neurona. Concretamente, el sistema de FHN es un sistema prototipo de sistema excitable en Dinámica neuronal. La representación, implementada mediante un applet de JAVA, muestra el comportamiento de la misma en diferentes situaciones.

En la implementación de este applet se estudia además cómo controlar el comportamiento de sistemas dinámicos excitables mediante la técnica de control por la fase Φ, que consiste en la aplicación de una perturbación armónica que es ajustada en busca de comportamientos diferentes del sistema de FHN. Este método de control por la fase podría ser útil para un mejor entendimiento de los sistemas excitables, así como para comprender diferentes fenómenos relacionados con el comportamiento de las neuronas.

En cuanto a nuestra aplicación gráfica, se desarrolló mediante un applet de Java, que es uno de los lenguajes de programación orientados a objetos más usados en la actualidad. Por otro lado, se optó por realizarlo con la ayuda del entorno de desarrollo NetBeans, que es un proyecto de código abierto creado por la empresa Sun MicroSystems, que hace más cómoda la programación de aplicaciones en Java.

La ventaja de los applets de Java radica en que la simulación del sistema dinámico puede ejecutarse en cualquier sistema operativo (Unix,Mac OS,Windows,etc) y mediante cualquiera de los navegadores web disponibles (Mozilla Firefox, Safari, Opera, Windows Explorer, etc). Esto se traduce en que la aplicación posea una gran portabilidad y usabilidad y, de este modo, que puedan acceder a ella un gran número de usuarios.

El usuario podrá comprobar de una forma visual y rápida el comportamiento de las neuronas en el sistema dinámico de FHN, por lo que el applet podría ser útil tanto para profesores como para alumnos de institutos y universidades, ya sea para la docencia o para la comprensión de los diferentes fenómenos relacionados con el sistema dinámico de FHN.

Índice

1.Introducción

Los **sistemas dinámicos** son sistemas que se utilizan para describir muchos de los fenómenos que se producen en la naturaleza, que presentan un cambio o evolución de su estado en el tiempo y tratan de buscar respuesta a muchos de dichos fenómenos. Los sistemas dinámicos se pueden clasificar en lineales y no lineales.

Los llamados **Sistemas Lineales** permiten un tratamiento algebraico, cuyas soluciones tienen un comportamiento relativamente simple y pueden ser analizadas con cierto tipo de facilidad. Por otro lado, los **Sistemas No Lineales** pueden dar lugar a dinámicas muy complejas y por tanto son muy difíciles de analizar. Por ello, los sistemas dinámicos no lineales se apoyan en métodos numéricos tales como las ecuaciones diferenciales. La mayoría de los fenómenos físicos de interés son no lineales.

Por otro lado, los **sistemas dinámicos excitables** son aquellos en los que su configuración permanece estable en ausencia, o presencia, de pequeñas perturbaciones; en el caso del sistema dinámico de FHN [1] dichas perturbaciones son los impulsos eléctricos. Si las perturbaciones son suficientemente fuertes, el sistema realiza una excursión en el espacio de las fases (en muchos casos independiente de la magnitud de la perturbación), volviendo al estado original de reposo.

La mayor parte de las veces, la resolución de ecuaciones diferenciales no es posible por métodos matemáticos exactos, por lo que, desde un punto de vista geométrico, se procederá sólo a encontrar elementos que ayuden a bocetar las trayectorias fásicas mediante las cuales determinar puntos estacionarios que se aproximen al valor real de las funciones.

En lo que respecta al proyecto de final de carrera presentado, el **Sistema Dinámico de FHN** es un sistema dinámico no lineal y enmarcado dentro de los llamados sistema excitables, que trata de explicar el comportamiento de una neurona cuando es excitada mediante impulsos eléctricos.

El cerebro es un sistema complejo y la comprensión de la actividad cerebral, por su importancia y dificultad, constituye uno de los grandes retos de la ciencia moderna. No es posible prescindir del uso de modelos matemáticos para entender la funcionalidad de la mente en términos de las bases fisicoquímicas de la fisiología del cerebro. La aproximación matemática al estudio del cerebro y del sistema nervioso en general contempla, entre otros aspectos, la construcción y el análisis de modelos de las unidades fundamentales que lo constituyen, es decir, las células nerviosas o neuronas. Los fenómenos eléctricos juegan un papel determinante en la fisiología de las células nerviosas [5] y es un factor que ha sido estudiado en profundidad en el pasado y lo sigue siendo en la actualidad.

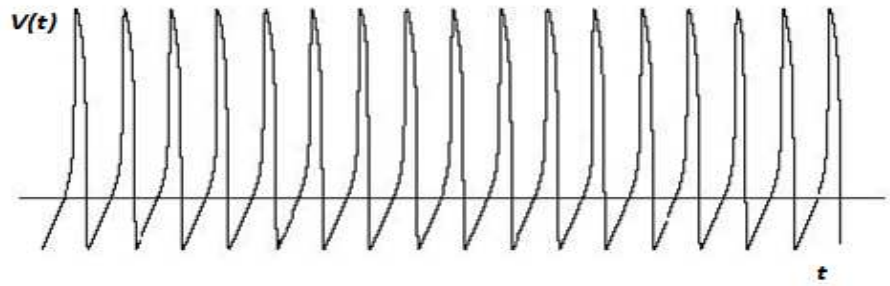


Figura 1.- *Tren de potenciales de acción en el cual el sistema de FHN responde a una dinámica de disparos*

En cuanto al sistema de FHN, es un modelo que permite observar muchos de los efectos descubiertos en las células de las neuronas. Concretamente, el sistema periódico de FHN ha sido utilizado para investigar los efectos de perturbaciones externas en la generación de pulsos eléctricos o potenciales de acción en las neuronas (Figura 1). Lo que es más, el sistema periódico de FHN ha sido usado, recientemente, para investigar el papel del ruido en el proceso de codificación, o la relación entre la respuesta del modelo de la neurona de FHN y las perturbaciones externas.

El control del comportamiento de las neuronas podría resultar de gran utilidad, pero existen interrogantes sobre cómo controlar la relación de entrada-salida mediante una perturbación externa. Con dicho objetivo, se han propuesto varias técnicas para controlar el comportamiento de varios modelos relacionados con el sistema de FHN. Considerando que el sistema de FHN puede presentar un comportamiento caótico, se puede llevar a cabo el control del sistema mediante la perspectiva del control del caos. Por otro lado, el término **caótico** (comportamiento errático o aperiódico) se suele utilizar para describir sistemas no lineales cuya dinámica evoluciona de puntos estables a estados en los cuales no se puede determinar una regularidad u orden, en nuestro caso, puntos en los cuales el sistema de FHN presenta un gran número de picos o valores máximos. Los sistemas caóticos son deterministas (y en este sentido, clásicos) ya que se conoce tan precisamente como se quiera la secuencia que les da origen, es decir, la ley que rige su evolución. Sin embargo, son impredecibles a tiempos suficientemente largos dado su dependencia sensible a las condiciones iniciales. Es decir, valores inicialmente muy próximos unos de otros difieren exponencialmente a lo largo del tiempo dando lugar a situaciones muy distintas. Los métodos de control del caos se pueden clasificar en métodos "realimentados" y "no realimentados".

En cuanto a los métodos no realimentados, son usados con frecuencia para eliminar el caos en sistemas dinámicos con forzamiento periódico. Un grupo importante de sistemas de este tipo son los osciladores no lineales, cuya ecuación puede ser descrita como:

$$\ddot{x} + \mu\dot{x} + \frac{\mathrm{d}V}{\mathrm{d}x} = F\cos(\omega t)$$

donde μ es el coeficiente de amortiguación, *V(x)* la función del potencial *y F cos (ωt)* un forzamiento periódico externo. Dependiendo del valor del potencial *V(x),* nos encontramos con diferentes grupos de osciladores.

2. Objetivos y metodología

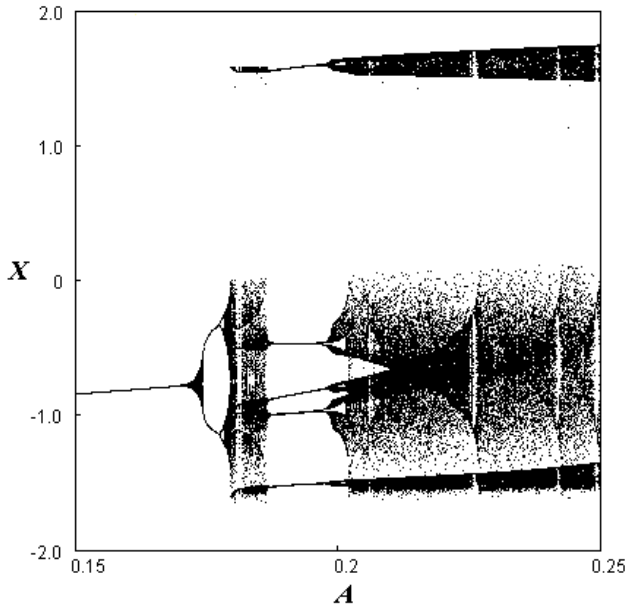


Figura 2.- *Diagrama de bifurcaciones tomando como variable X y respecto de un parámetro de control A.*

El principal objetivo del proyecto de final de carrera desarrollado es la representación del sistema dinámico excitable de FHN. Dicha representación debe ser intuitiva, para que cualquier usuario con unos conocimientos básicos en física pueda usar la aplicación de una forma fácil y cómoda. Como anteriormente explicamos, los sistemas dinámicos son aquellos sistemas complejos en los que su comportamiento o propiedades varían en función del tiempo. Un ejemplo de dicho tipo de sistemas es el sistema dinámico de FHN. El usuario podrá interactuar con la aplicación y estudiar su comportamiento introduciendo en la aplicación unos parámetros de entrada que harán que la representación del sistema varíe. En dicha representación, el usuario podrá elegir el tipo de gráfico a representar: una o varias órbitas del sistema dinámico, diagramas de bifurcaciones (Figura 2), etc. Un diagrama de bifurcaciones se puede definir como un gráfico en el que se representan las soluciones de los sistemas dinámicos en función del rango de valores de uno o más parámetros. En nuestro caso, los diagramas representarán los máximos relativos de las X para el sistema de FHN (eje vertical de coordenadas) en función de un determinado parámetro de control (Figura 2). Los diagramas de bifurcaciones son útiles para ver como varía el comportamiento dinámico del sistema a medida que cambiamos el valor de nuestro parámetro de control.

Figura 3.- *Curso temporal del voltaje eléctrico a través de la membrana celular excitada*

Respecto a la elección del tipo de representación elegido para la aplicación informática, se decidió implementar la representación mediante un applet de Java. Las principales ventajas presentadas por dicho sistema de representación radican en que los applets se pueden ejecutar en cualquier sistema operativo y navegador disponibles en la actualidad, factores que dotan a la aplicación de una gran portabilidad.

Dado el carácter de portabilidad de los applets, cualquier usuario podría ejecutar la representación de la aplicación del sistema dinámico de FHN en cualquier ordenador, ya sea en universidades, colegios, institutos, o en cualquier otro ordenador personal. El applet podría ser útil para la docencia, pues muestra de forma clara el comportamiento del sistema dinámico de FHN facilitando así su comprensión.

2.1.Descripción del problema

El sistema dinámico de FHN se describe mediante las siguientes ecuaciones:

$$\frac{dx}{dt} = c(-y + x - (x^3/3) + S(t)),$$

$$\frac{dy}{dt} = x - by + a,$$

donde $S(t) = A \sin(\frac{2\pi}{T}t)$

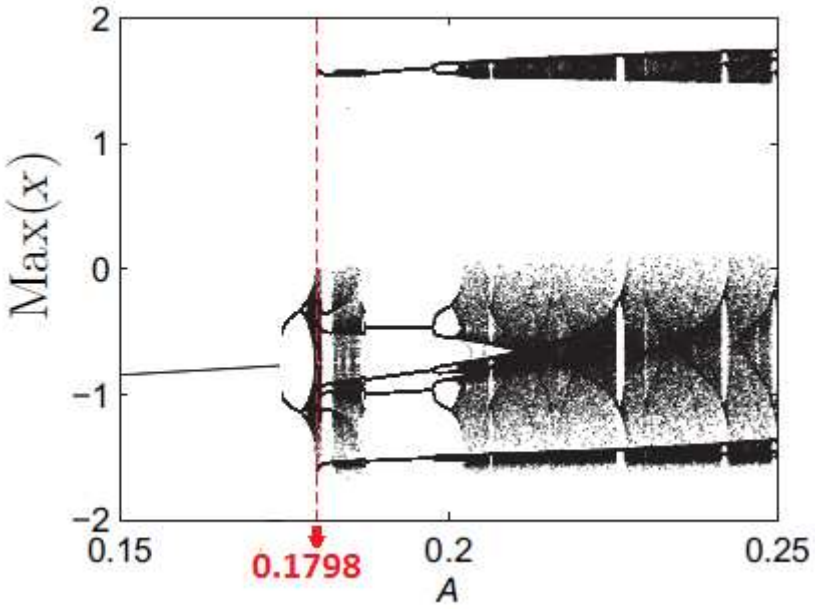


Figura 4.- *Diagrama de bifurcaciones para los máximos de las X tomando como parámetro de control la amplitud A. Se observa una dinámica de disparo y un comportamiento caótico para valores de la amplitud mayores o iguales a 0.1798 . Sin embargo, para valores de la amplitud inferiores a 0.1798, el sistema presenta un comportamiento caótico o no sin dinámica de disparo.*

Con respecto a las ecuaciones anteriormente mostradas, en el contexto de la neurofisiología, x(t) es conocida como la variable del voltaje, y(t) la variable de recuperación, S(t) es el control externo y a, b, c son constantes positivas.

Los sistemas dinámicos no lineales, y en particular el Sistema de FHN, se pueden controlar mediante **técnicas de control** [6]. Dichas técnicas se basan en la estabilización del sistema mediante pequeñas perturbaciones que regulen su comportamiento. No obstante, dichas perturbaciones deben tomar valores que eviten modificaciones sustanciales en la dinámica original del sistema. En nuestro caso, vamos a usar la técnica de control de fase, en la cual se añaden perturbaciones a uno de los parámetros accesibles del sistema. En nuestro caso, se incorpora una pequeña perturbación periódica a la amplitud del parámetro de control externo $S(t)$. El parámetro $S(t)$ es reemplazado por el término $S'(t) = A(1+ \delta(t))\sin(2\pi t/T)$, donde $\delta(t) = \epsilon \sin((2\pi rt/T) + \Phi)$. El parámetro r es la relación existente entre la frecuencia aplicada a la perturbación $\delta(t)$ y la frecuencia propia de control externo $S(t)$. El parámetro con mayor importancia en nuestro sistema de control de fase es Φ, que representa la diferencia de fase la perturbación aplicada y el control $S(t)$. La técnica de control de fase consiste en seleccionar un valor apropiado de la fase Φ, una vez que *r y* ϵ han sido fijadas, con el objetivo de provocar el comportamiento deseado en el sistema.

Incrementando la amplitud del control externo, el sistema se somete a una bifurcación de doble periodo, la cual lleva a un comportamiento como se ve en la Figura 4, donde los valores máximos de x están representados en función de la amplitud. El valor de los parámetros usados en este diagrama de bifurcaciones son $a = 0.7$, $b = 0.8$, $c = 12.5$ y $T = 1.125$.

En dicho diagrama se intentan reflejar mediante un punto cada uno de los valores máximos de las X's que tendrá el sistema de FHN para el intervalo de valores de la amplitud (0.15,0.25). Como podremos observar, nos encontramos con dos comportamientos muy diferenciados: si la amplitud es mayor o igual a 0.1798, el sistema presenta un comportamiento caótico y posee una dinámica de disparo (Figura 1), pues el sistema de FHN presenta valores elevados de la variable X. Sin embargo, para valores de la amplitud inferiores a 0.1798 el sistema presenta un comportamiento caótico o no pero sin dinámica de disparo (la variable X no alcanza valores tan elevados) .

2.2 Estudio de alternativas

Las ecuaciones diferenciales aparecen naturalmente al modelar situaciones físicas y de otras ciencias, en nuestro caso el Sistema Dinámico de FHN. Por lo general, la solución exacta de un problema de valor inicial es imposible o difícil de obtener en forma analítica. Por tal razón, se utilizan métodos numéricos para aproximar dichas soluciones. Entre ellos los que explicaremos a continuación:

Método de Euler

El método de Euler [2] de primer orden es un método para la resolución de ecuaciones diferenciales ordinarias y las soluciones que proporciona son aproximaciones numéricas (Figura 5). La idea de dicho método está basada en el significado geométrico de la pendiente de una función en un punto inicial dado para, posteriormente, poder hallar el punto siguiente. Dicha ecuación sería la siguiente:

$y_{n+1} = y_n + hf(x_n, y_n) + O(h^2)$,

donde h es el tamaño del paso temporal. Este método es más preciso cuanto menor sea el tamaño de paso o h, y presenta un margen de error elevado respecto al verdadero valor de las funciones.

No es muy recomendable el uso de este tipo de método para realizar aproximaciones ya que no es muy preciso en comparación con otros métodos con el mismo valor de "h", ni tampoco es muy estable.

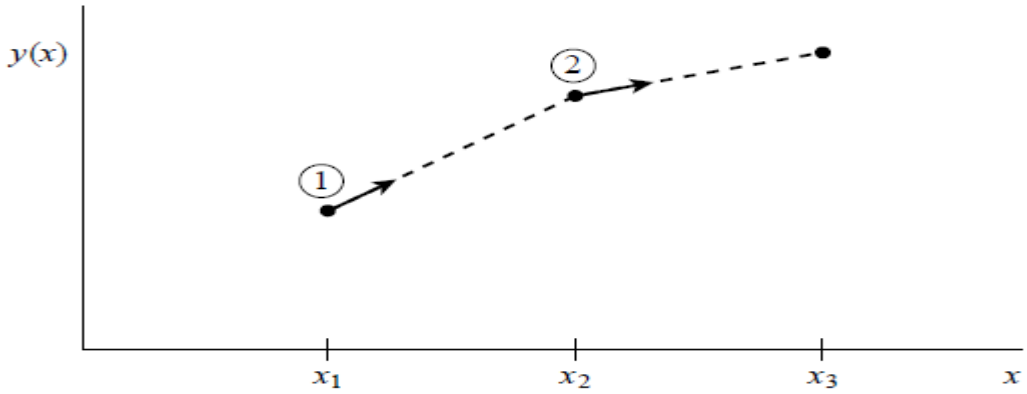


Figura 5.- *Método de Euler: es el más simple y menos preciso de los métodos de aproximación de ecuaciones diferenciales ordinarias. La pendiente o derivada en el punto inicial (①) es extrapolado para obtener el siguiente valor en la función (②). El método tiene precisión de primer orden.*

Con el objetivo de remediar lo errores anteriormente mencionados, se desarrollaron nuevos métodos de aproximación, como el método de Euler mejorado.

Método de Euler mejorado

La diferencia de este método (Figura 6) respecto al método anteriormente mencionado consiste en que el método de Euler usa la pendiente de la función en el punto inicial del intervalo para extrapolar el valor del punto siguiente. Por el contrario, en el método de Euler mejorado, dicho proceso no será suficiente para hallar el valor del siguiente punto y se deberá acompañar de un paso intermedio. Es decir, se debe hallar la derivada de la función en un punto intermedio del intervalo para, a continuación, extrapolar el valor en el punto final del mismo. Por dicho motivo, este método es también conocido como el método del punto medio. Las ecuaciones del método serían las del siguiente conjunto:

$k1 = hf\,(xn,\ yn)$

$k2 = hf\,(xn + \tfrac{1}{2}h,\ yn + \tfrac{1}{2}h\,k1)$

$y_{n+1} = yn + k2 + O(h^3)$

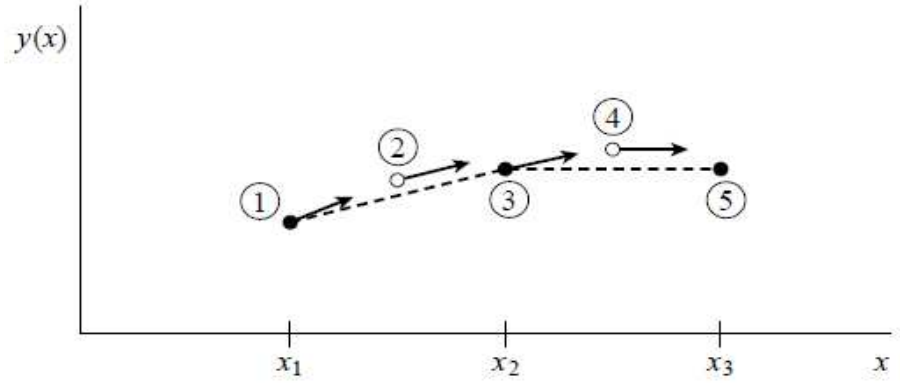


Figura 6.- *Método del punto medio: se aumenta la precisión usando el valor de la pendiente en el punto inicial (①) para encontrar los puntos intermedios de los intervalos (②, ④). Una vez hallados, se usará la pendiente en los puntos intermedios a lo largo de toda la función para poder hallar los siguientes puntos. En la imagen, los puntos coloreados en negro (①. ③, ⑤) representan los valores finales de la función, mientras que los puntos no coloreados (②, ④) representan los puntos de la función que han sido descartados una vez que su derivada o pendiente ha sido calculada y usada.*

Método de Runge-Kutta

Fue creado en 1900 por los matemáticos C. Runge y W. Kutta y es uno de los procedimientos más usados en la actualidad debido a la alta precisión de sus aproximaciones.

Como procedimiento de Runge-Kutta se entiende no sólo un método, sino que realmente son un conjunto de métodos iterativos cuya función es la de aproximar el valor de ecuaciones diferenciales ordinarias.

Como mencionamos anteriormente, el método de Euler se mueve a lo largo de la tangente de una cierta curva que esta "cerca" a la curva desconocida o buscada, es decir, cerca del valor real de la función en un punto. Los métodos Runge-Kutta extienden esta idea geométrica al utilizar varias derivadas o tangentes intermedias, en lugar de sólo una, para aproximar la función desconocida. Para realizar este método se deben calcular cuatro valores en cada intervalo k1, k2, k3 y k4, correspondientes a cuatro pendientes intermedias, y se debe elegir un paso de integración h. Las ecuaciones del método de Runge - Kutta de cuarto orden son las siguientes:

$$
\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\
k_3 &= hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\
k_4 &= hf(x_n + h, y_n + k_3) \\
y_{n+1} &= y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5)
\end{aligned}
$$

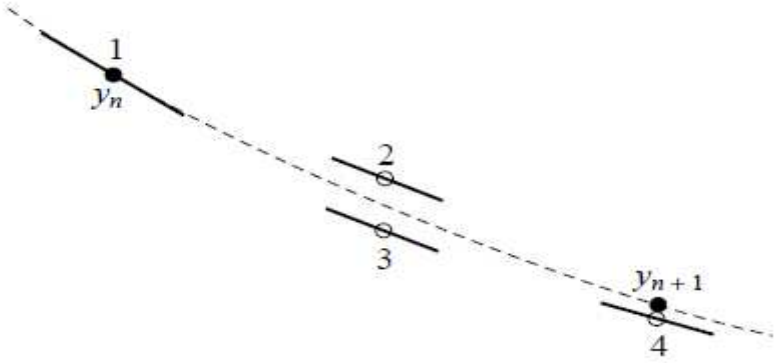


Figura 7.- *Método de Runge-Kutta de cuarto orden. Para cado anchura de paso "h" se debe calcular la pendiente cuatro veces: una en el punto inicial (1) y otras en los puntos intermedios (2, 3) y otro en el punto final del intervalo (4). A Partir de estas derivadas se calcula el valor final de la función (mostrado con un punto negro relleno) .*

Para la implementación del applet, se ha usado el algoritmo de Runge-Kutta de cuarto orden (Figura 4). Las aproximaciones el método de Runge-Kutta de cuatro orden es mucho mayor que la de los métodos de Euler anteriormente mencionados, por lo que fue el método usado para la implementación del applet sobre el sistema dinámico de FHN.

Otras alternativas

Se estuvieron meditando distintas formas de representación para el sistema dinámico de FHN, pero finalmente se optó por desarrollar la aplicación mediante un applet de Java , debido a la portabilidad de este tipo de representación. Respecto a la metodología seguida para la representación, se decidió desarrollar inicialmente las partes básicas del applet para, posteriormente, ir añadiendo sucesivamente nuevas características que completasen la interfaz.

2.3. Metodología empleada

La ingeniería del software es la disciplina o área de la informática que ofrece métodos y técnicas para desarrollar y mantener software de calidad. Las técnicas de desarrollo que ofrece tienen como objetivo mejorar la productividad y la calidad del software estableciendo metodologías sistemáticas o predecibles. En al caso del applet del sistema dinámico de FHN, se ha optado por la metodología incremental [11].

Figura 8.- *Método incremental*

La idea principal de la metodología iterativa de desarrollo de software es implementar las aplicaciones de manera incremental, permitiendo al usuario sacar ventaja de lo aprendido a lo largo del desarrollo anterior incrementando, sucesivamente, versiones de la aplicación. Los pasos claves en el proceso son comenzar con una implementación simple de los requerimientos del sistema, e iterativamente mejorar la secuencia evolutiva de versiones hasta que el sistema completo esté implementado. En cada iteración se realizan cambios en el diseño y se agregan nuevas funcionalidades y capacidades al sistema. En cuanto a las diferentes etapas de la metodología iterativa, son las siguientes:

Análisis de requisitos

En esta etapa se capturan las necesidades o las condiciones a satisfacer de un software nuevo o modificado. Es trascendental que el análisis de requisitos alcance un estado óptimo antes de alcanzar la fase de "Diseño". Los buenos requisitos no deben tener ambigüedades ni contradicciones.

En esta etapa se tuvo que comprender el sistema dinámico excitable de FHN y el método de aproximación establecido para resolverlo y crear un esquema o idea elemental de cómo implementar la interfaz gráfica del applet, idea que fue cambiando progresivamente en las sucesivas iteraciones del desarrollo del applet.

Diseño

En esta etapa se define una solución que convierta los requisitos en la aplicación deseada. El análisis de requisitos se centraba en qué se debía hacer en la aplicación, mientras que la fase de diseño hace hincapié en cómo se debe realizar. En esta etapa se diseñó el diagrama de clases y se decidió la función que tendrían cada una de las clases de la aplicación del sistema dinámico de FHN.

Programación

En esta fase se codifica, es decir, se lleva a código fuente las clases definidas en la etapa anterior de "Diseño". En esta etapa se codificaron las clases [3], diseñadas en la etapa anterior de análisis, en el lenguaje de programación Java con la ayuda del entorno NetBeans.

Pruebas

Consiste en comprobar que el software realice correctamente las tareas indicadas en la fase de análisis de requisitos. En esta fase se comprobaron, entre otras tareas, que la animación de las gráficas se reflejaba correctamente o que se pudieran desplazar las gráficas hacia los lados.

Mantenimiento

El mantenimiento de software es el proceso de control, mejora y optimización del software ya desarrollado. Incluye depuración de errores y defectos que puedan haberse filtrado de la fase de pruebas. Esta fase es la última antes de iterar, según el modelo iterativo incremental, que se aplica al ciclo de vida del desarrollo de software. En esta etapa, entre otras acciones, se mejoró el aspecto de la interfaz de la aplicación y se fueron incorporando las nuevas funcionalidades al applet.

3.Descripción informática

Para la implementación de la interfaz se ha recurrido a un applet de java. Podríamos definir los applets como programas escritos en Java, que se descargan desde un servidor y se ejecutan en el navegador del cliente, es decir, en el navegador de la persona que quiera ejecutar dicho programa. Para añadir el applet en una página web, se usa una página HTML, que será donde se añadirá la aplicación desarrollada en Java. El navegador del usuario es el que se encargará de ejecutar dicho documento HTML.

Para ejecutar un applet se puede recurrir al *AppletViewer*, desarrollado por la compañía Sun MicroSystems, o por medio de un navegador web, como por ejemplo Internet Explorer, Netscape, Opera, Safari o Mozilla Firefox entre otros. Entre las características principales de los applets se pueden citar las siguientes :

- Los applets poseen un **esquema de seguridad** que restringe el acceso a partes sensibles de la máquina en la que se ejecutan. Como ejemplo de este tipo de acciones podemos citar, por ejemplo, el acceso a archivos del usuario para escribir en ellos o modificar información. Por el contrario, si se desea que los applets puedan realizar dicho tipo de acciones, se les deberán entregar los permisos de ejecución correspondientes.

- La gran ventaja de los applets es su **portabilidad**. Son multiplataformas, pues los applets de Java se pueden ejecutar en cualquier navegador disponible y, además, en cualquiera de los sistemas operativos que dispongan de JVM, entre otros, Windows, Linux o Mac OS.

- Como **desventaja** que tienen los applets podríamos decir que los **navegadores** que los vayan a ejecutar **necesitan un plug-in de Java**, que no siempre está disponible en todos los navegadores. Algunas organizaciones sólo permiten la instalación de software a los administradores. Como resultado, muchos usuarios, sin privilegios para instalar el plug-in en su navegador, no pueden ver los applets.

3.1 Especificación

Podríamos definir la especificación de requisitos como el proceso en el que se describe el comportamiento esperado en el software una vez desarrollado. Gran parte del éxito de un proyecto de software radicará en la identificación de los requisitos. Los requisitos se pueden clasificar en funcionales y no funcionales.

Por un lado, los requisitos funcionales son aquellas características que debe incorporar el sistema o aplicación a desarrollar, como acciones que éste deberá ser capaz de desempeñar. Por otro lado, los requisitos no funcionales están más enfocados al diseño o la implementación

de la aplicación. Son aquellos requisitos que ni describen información a guardar ni las funciones que la aplicación debe desarrollar : calidad, eficiencia, disponibilidad...

Requisitos funcionales

- La aplicación debe representar el gráfico correspondiente al sistema dinámico de FHN.
- El programa debe representar el diagrama de bifurcaciones respecto de la fase Φ del sistema dinámico excitable de FHN.
- El applet debe representar el diagrama de bifurcaciones respecto de la amplitud A del sistema dinámico excitable de FHN.
- El programa dispondrá de los elementos necesarios que recojan los parámetros de entrada introducidos por el usuario.
- El programa deberá poder representar los distintos gráficos mediante una animación.
- La aplicación dispondrá de los controles necesarios para que el usuario pueda pausar, interrumpir o iniciar una representación.
- El programa permitirá variar la velocidad de representación de las animaciones.
- El programa poseerá controles que permitan que el usuario pueda mover hacia los lados, acercarse o alejarse de las representaciones.
- La aplicación permitirá cambiar el color de las representaciones.

Requisitos no funcionales

- El programa deber ser desarrollado en un applet de Java.
- La aplicación debe ser fácil de usar e intuitiva.
- La aplicación debe responder a los órdenes de usuario de una forma efectiva y rápida.
- La interfaz de la aplicación debe ser atractiva para el usuario.
- El programa deberá controlar la introducción de datos erróneos que pudiesen provocar errores de ejecución de la aplicación.

3.2 Diseño

Java proporciona las clases AWT y Swing, que son muy útiles para implementar interfaces gráficas de usuario. En la implementación del applet se ha decidido escoger la clase Swing, ya que proporciona una serie de ventajas frente a la AWT: posibilita la inclusión de iconos en los botones; se pueden elegir varios tipos de bordes disponibles para los elementos; los botones pueden tener una forma no rectangular o, por último, se pueden usar los "ToolTips", que son mensajes emergentes de información para el usuario.

Swing proporciona muchas clases prefabricadas para hacer más fácil la implementación de interfaces gráficas. A continuación, explicaremos brevemente los elementos que han sido utilizados en el diseño gráfico:

- JTabbedPane: es un contenedor que se emplea para agrupar otros elementos en pestañas. Este elemento da la posibilidad al usuario de acceder a las variables de cada grupo haciendo click en la pestaña correspondiente a dicho grupo. Este elemento lo hemos usado para poder contener las variables correspondientes a las órbitas 1 y 2 , cada una en una pestaña diferente.

- JTextField: es un componente que permite la entrada de texto por parte del usuario. Lo hemos usado para que el usuario introduzca los valores de las variables de las órbitas y, en el caso de que el usuario elija representar un diagrama de bifurcaciones, para que introduzca el número de puntos para los que debe calcular el diagrama. Cuanto mayor sea el número de puntos introducidos en el JTextField del diagrama de bifurcaciones, mayor resolución tendrá la representación del diagrama.

- JCheckBox: se trata de un componente que se emplea para seleccionar una opción entre un conjunto de opciones predefinidas. Se usa para indicar si queremos que se muestren los ejes de coordenadas en la representación o, en el JTabbedPane de las órbitas, para indicar que si queremos representar el sistema dinámico de FHN con una o dos órbitas.

- JButton: es un componente en el que el usuario hace click para desencadenar una acción específica. Se usa para desplazar los gráficos y los diagramas de bifurcaciones; para iniciar, pausar o detener la reproducción de una gráfica mediante los botones de "Play", "Pause" y "Stop" respectivamente; cambiar el color en el que se representa la gráfica o diagrama; en el caso de que se esté representado dinámicamente el sistema dinámico de FHN y se pulse el botón del icono con la tijera, para borrar la trayectoria anteriormente dibujada y, por último, los botones del panel de zoom se usan para alejar o acercarse a la gráfica.

- JProgressBar: consiste en una barra de progreso que permite ver de una forma gráfica la evolución de una tarea. Nos será útil para poder ver la evolución de la reproducción de las gráficas.

- JComboBox: consiste en una lista desplegable de cajas de texto. Se debe elegir una de dichas opciones de las cajas de texto y estas son excluyentes. Inicialmente, sólo se puede ver el valor de una de las opciones de la caja de texto y, para ver las demás se debe hacer click en la pestaña pequeña situado en su extremo derecho. Dicha acción desplegará una lista con todas las opciones de la caja de texto a elegir. En este applet, se usa JComboBox para poder elegir el tipo de gráfico a representar o para elegir el intervalo de valores de la fase Φ, en el caso de que se elija representar el diagrama de bifurcaciones respecto de Φ, o para elegir el intervalo de valores de A, en el caso de que se elija representar el diagrama de bifurcaciones respecto de la amplitud A.

- JRadioButton: permite la elección entre un conjunto de elementos. Se usa junto a la clase "JButtonGroup", que asociada a un conjunto de JRadioButton, posibilita que sólo sea posible la elección de uno de los JRadioButton. Lo usaremos para poder elegir entre la representación dinámica o estática de la gráfica.

- JSlider: permite la selección de una forma gráfica de un valor, que debe oscilar entre un máximo y un mínimo, mediante el deslizamiento de un barra. Este elemento lo emplearemos para aumentar o disminuir la velocidad de representación de la gráfica en caso de que se haya elegido la representación dinámica.

- JLabel: es una etiqueta de texto. Lo usamos para informar al usuario a qué variable corresponden cada uno de los JTextFields; para indicar el tipo de gráfico seleccionado o para saber para qué se usan cada una de las componentes si pasamos el puntero del ratón sobre ellas, en el JLabel de la parte superior del panel de dibujo de las gráficas.

- JPanel: es un contenedor que sirve para agrupar otros elementos. Facilita el diseño y la organización de los componentes en la interfaz de usuario. En el applet hemos usado varios JPanel, entre ellos las grandes superficies rectangulares que están a la izquierda y derecha de la zona de representación de las gráficas, o el que está en su parte superior, que contiene la etiqueta de texto con información para el usuario.

A continuación, nos centraremos en las partes significativas de la interfaz en grupo, que usarán los elementos anteriormente descritos.

3.2.1.Panel del tipo de gráfico

Figura 3.1.- Panel Tipo Gráfico

Este panel está formado por un JComboBox. Dicha componente contiene tres posibles gráficas para representar: el "sistema dinámico de FHN"; "diagrama de

bifurcaciones Φ", que es el diagrama de bifurcaciones formado tomando valores de la variable Φ para el eje horizontal y "diagrama de bifurcaciones A", que toma para el eje horizontal valores de la amplitud "A".

3.2.2. Panel de los intervalos de cálculo de los diagramas de bifurcaciones

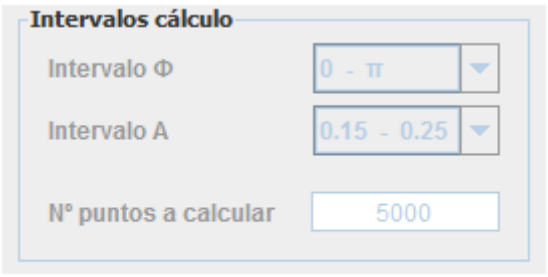


Figura 3.2.- Panel de los intervalos de cálculo de los diagramas de bifurcación

Este panel está compuesto por dos JComboBox que corresponden a los rangos de intervalos para los diagramas de bifurcaciones respecto de Φ y respecto de la amplitud "A". Estas componentes anteriormente mencionadas están acompañadas, cada una, de una etiqueta de texto o JLabel que indicará si el intervalo se corresponde al intervalo respecto de Φ o respecto de la amplitud.

Por último, el panel de intervalos cuenta con un JTextField, que servirá para poder indicar el número de puntos para los que se dibujará el diagrama de bifurcaciones. Cuanto mayor sea el valor indicado para los puntos, mayor será la resolución de los diagramas de bifurcaciones dibujados. Este JTextField de los puntos está acompañado también por un JLabel, que indicará para qué sirve dicho JTextField.

3.2.3. Panel de las características de representación

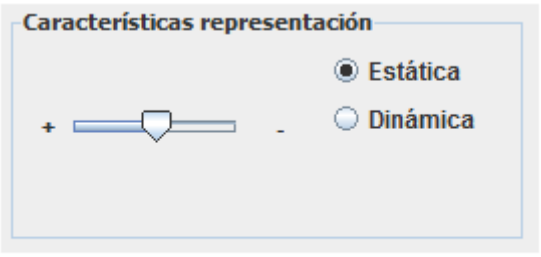


Figura 3.3.- Panel de las caraterísticas de representación

En este contenedor se han incluido los siguientes elementos: dos JRadioButton excluyentes para indicar si se prefiere que la representación se realice con una animación, JRadioButton "Dinámica", o si queremos que se represente toda la gráfica de golpe, JRadioButton "Estática". Estos dos JRadioButton están acompañados por un JRadioButtonGroup, cuya única función en el panel es que la elección de estas dos opciones anteriormente mencionadas sean excluyentes.

Por otro lado, el panel también tiene un JSlider que servirá para indicar la velocidad a la que se mostrará la reproducción si la opción seleccionada es la "Dinámica". Este JSlider tiene al lado de cada uno de sus extremos dos JLabel con los signos "+" y "-", que indicarán hacia qué lado hay que mover el JSlider para conseguir una mayor o menor velocidad de reproducción respectivamente.

3.2.4. Panel para mover la gráfica

Figura 3.4.-Panel mover gráfica

Este panel está compuesto por cinco botones. Los cuatro botones con imágenes de flechas servirán para poder desplazar o mover la gráfica una vez que se ha representado o, también, para poder desplazarnos durante la reproducción en caso de que el tipo de gráfica seleccionada sea el "Sistema dinámico FHN".

Por último, el botón central formado por un icono con un signo de "x" sirve para que la gráfica vuelva a su posición inicial. Por consiguiente, esto será útil para poder volver a situar la representación en el centro del panel de dibujo si hemos pulsado los botones de las flechas para desplazar la gráfica.

3.2.5. Panel del zoom

Este panel está formado por tres botones. Los botones de la parte superior e inferior del panel sirve para poder ampliar o alejarse de la gráfica respectivamente. Por otro lado, el botón que está entre los dos, en el medio, sirve para volver a representar la gráfica en su posición inicial. Por tanto, este botón nos será útil en el caso de que se hayan pulsado los botones de acercarse o alejarse y se quiera que la gráfica se represente centrada y en su posición original.

Figura 3.5.- Panel del zoom

3.2.6 Panel contenedor de la derecha del panel de dibujo

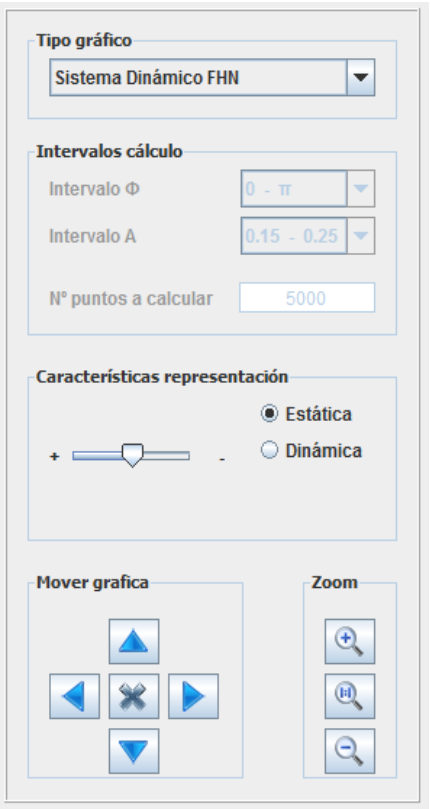


Figura 3.6- Panel contenedor de la derecha del panel de dibujo

Este panel tendrá como única función contener los paneles anteriormente citados.

Cada uno de estos paneles tendrá un "TitledBorder" con un texto de título de panel de color azul para indicar el uso del panel.

A su vez, este panel contenedor tendrá un "EtchedBorder" de tipo "Raised".

3.2.7. Panel del Play/Pause/Stop

Figura 3.7.- Panel Play/Pause/Stop

Este panel contiene tres botones: el botón con la imagen del Play sirve para poder iniciar la representación de la gráfica o para poder reanudar la reproducción en caso de que se haya pausado anteriormente; el botón con la imagen de "Pause" servirá para poder pausar la animación de la gráfica en caso de que el tipo de reproducción elegido sea la reproducción dinámica y, por último, tenemos el botón del "Stop", cuya función es interrumpir la reproducción de la gráfica y limpiar la pantalla de dibujo.

El botón de "Pause" permanecerá inactivo en el caso de que el tipo de reproducción elegido sea el "estático". Por otro lado, el botón de "Play" estará desactivado durante la reproducción dinámica de una gráfica y volverá a quedar activo una vez que haya finalizado la reproducción de la misma.

3.2.8. Panel "Mostrar"

Figura 3.8.- Panel Mostrar

Este panel estará formado por dos botones y un JCheckBox: El botón con la imagen de una tijera servirá, durante la reproducción dinámica del gráfico "Sistema Dinámico FHN", para poder borrar la trayectoria de la gráfica anteriormente dibujada en el panel de dibujo. Esto puede ser muy útil, por ejemplo, si no tenemos interés en saber la trayectoria de una gráfica hasta un momento determinado. En el momento que queramos, pulsaríamos el botón de la tijera y, automáticamente, se borraría lo anteriormente dibujado a ese instante. En nuestro caso, el sistema dinámico excitable de FHN, tiene un comportamiento periódico en ciertas ocasiones y, por tanto, hay secciones en las que el dibujo de la trayectoria de las órbitas es muy enrevesado y poco claro. Esta funcionalidad es muy útil en dichos casos, pues se ayudaría a simplificar y ver de una forma más clara la trayectoria dibujada a partir del instante deseado.

El otro botón del panel es un botón con la imagen de una paleta de dibujo. Este botón sirve para poder cambiar el color de la gráfica durante la reproducción, en el caso de de que se haya elegido la representación dinámica del "Sistema Dinámico FHN", o tras finalizar la representación de cualquiera de las gráficas. Este botón permanecerá inactivo hasta que no se pulse el botón del "Play" .

Finalmente, tenemos el JCheckBox de los Ejes. Este elemento servirá para decidir si la representación de la gráfica será mostrada con o sin ejes. Se podrá seleccionar o deseleccionar dicho elemento durante la reproducción dinámica, si el tipo de gráfico elegido es el "Sistema Dinámico FHN", o tras la finalización de la representación de la gráfica en cualquiera de los demás casos.

3.2.9.Panel de información del JLabel

La única utilidad de este panel es la de alojar una etiqueta de texto de información. Esta etiqueta de texto informará al usuario sobre el tipo de gráfica elegida o, en caso de que situemos el cursor por alguna de las componentes significativas del applet, nos indicará el uso de dicha componente y si está desactivada.

Este panel tendrá un "EtchedBorder" de tipo "Raised".

Figura 3.9.- Panel de información del JLabel

3.2.10.Panel de las órbitas

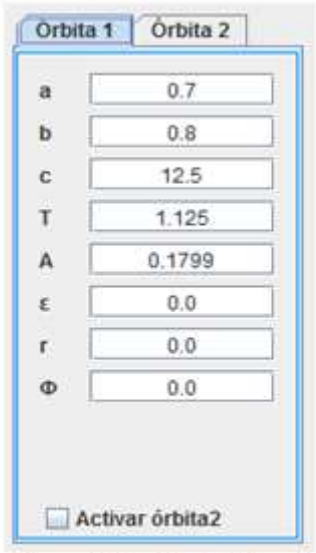


Figura 3.10.- Panel de las órbitas

Este panel está formado por dos pestañas. La función de la primera de las pestañas será la de contener todas las posibles variables que tengan que ver con la primera de las órbitas.

Este panel contendrá varios JTextField que servirán para recoger los valores que el usuario quiere dar a cada una de las variables. Estos JTextField estarán acompañados, cada uno, por una etiqueta de texto o JLabel que indicará a qué variable corresponden cada uno de los JTextField. En el caso de que el tipo de gráfico elegido a representar sea el "Sistema Dinámico FHN", podremos seleccionar o deseleccionar el JCheckBox que hay en la parte inferior de la primera de las pestañas. Este JCheckBox servirá para habilitar la segunda de las pestañas del JTabbedPane, que contendrá las variables de la segunda órbita.

Esta segunda pestaña contará con varios JLabel a su vez, que acompañarán a cada uno de los JTextField de las variables de la segunda órbita.

3.2.11.Panel contenedor de la izquierda

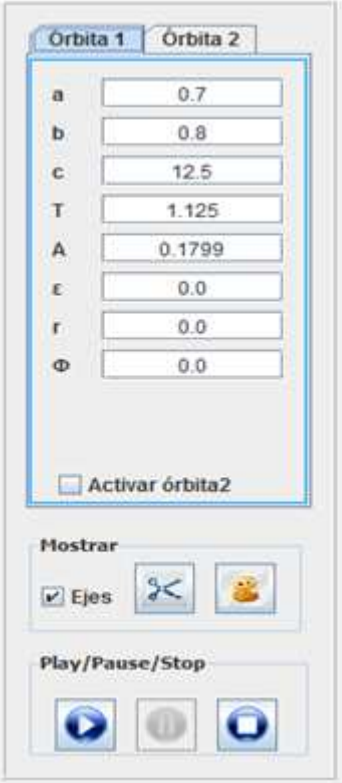


Figura 3.10.- Panel contenedor de la izquierda

Este panel alberga los paneles, anteriormente mencionados, de la parte izquierda del panel de dibujo. Todos los paneles alojados en este contenedor tendrán un "TitledBorder" con un título de panel de color azul.

Por otro lado, este panel contenedor de la izquierda del panel de dibujo, tendrá un "EtchedBorder" de clase "Raised".

3.2.13.Panel de dibujo

En la parte central del applet tendremos un panel, inicialmente en color blanco, que servirá para poder mostrar la representación de la gráfica. Este panel tendrá un borde de tipo "EtchedBorder" de la clase "Raised"

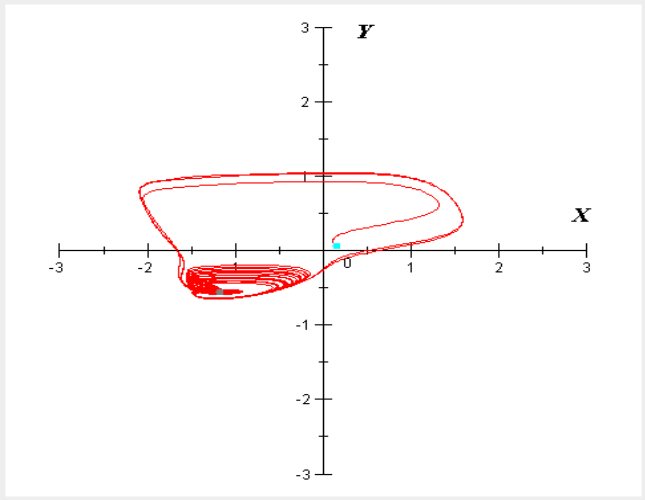


Figura 3.12.- Panel de dibujo con gráfica sistema FHN dibujada

3.2.14. Interfaz de la aplicación

Para finalizar mostraremos el aspecto completo de la interfaz con todos los componentes mostrados en los puntos anteriores.

Este panel tendrá un borde de tipo "EtchedBorder" de la clase "Raised"

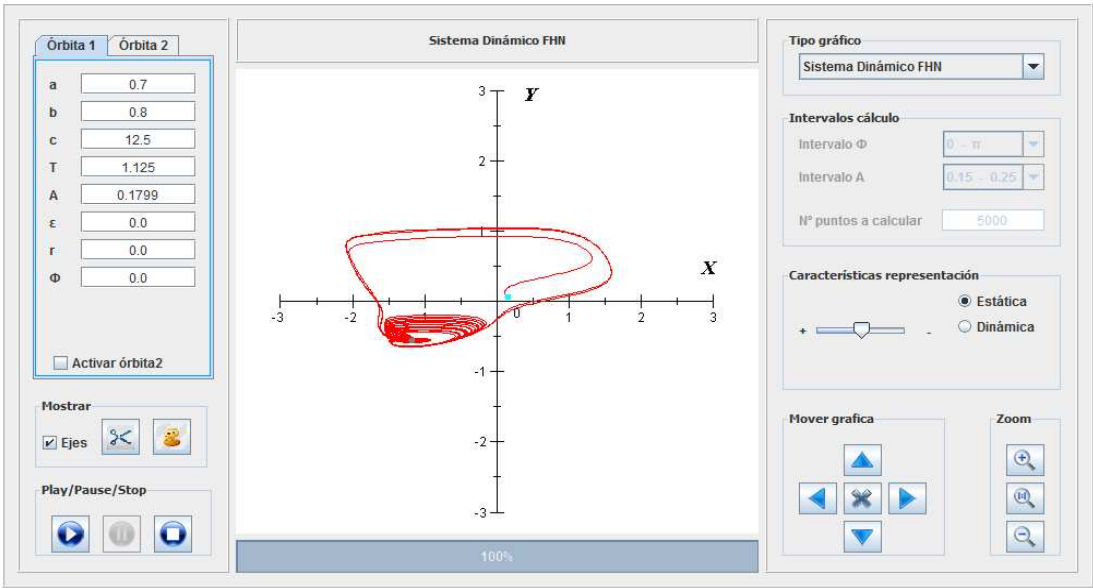
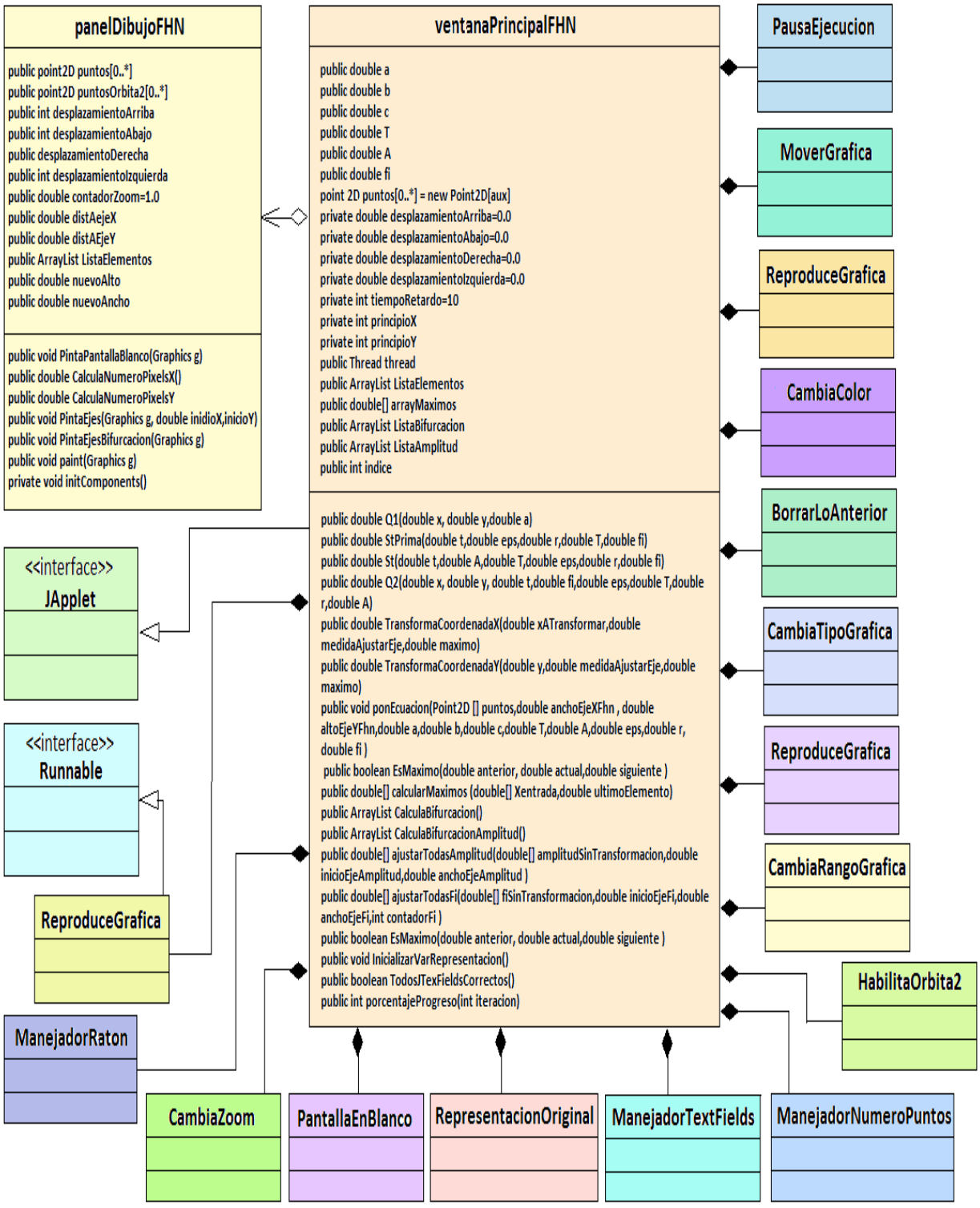


Figura 3.13.- Aspecto de la interfaz de la aplicación

3.2.14. Diseño de clases

A continuación mostraremos el esquema seguido para las clases del proyecto de final de carrera. Con el fin de que el diagrama de clases [7] sea lo más entendible posible, se ha optado por simplificarlo.



Clase ventanaPrincipalFHN

La clase ventanaPrincipalFHN, pintada en color carne en el diagrama de clases, es la clase principal del proyecto. Esta clase hereda de la interfaz JApplet, necesaria para todas las clases que quieran implementar un applet.

Una de las funciones de la clase ventanaPrincipalFHN es la de establecer la disposición de todos los elementos del applet en la interfaz gráfica. Además, deberá definir qué acciones se desencadenarán cuando el usuario interactúa con los distintos elementos de la interfaz, acciones reflejadas en el diagrama de clases mediante las distintas clases de color amarillo. Estas funcionalidades son asociadas a los elementos en el método init() y sirven, por ejemplo, para que cuando se pulse el botón con el símbolo de "Play", se dibuje una gráfica. Por otro lado, la clase ventanaPrincipalFHN calcula los puntos necesarios para pintar las distintas gráficas del applet. Estos puntos se calculan mediante el método de aproximación de Runge-Kutta, definido en el método *ponEcuacion* y, posteriormente, ajustan los puntos para que se representen en el plano de dibujo.

Un aspecto a destacar por su importancia en el applet son los threads: Los *threads* o *hilos de ejecución* [8] son segmentos de código de un programa que se ejecutan secuencialmente de modo independiente de las otras partes del programa. Esto contribuye decisivamente en el rendimiento de las aplicaciones. En nuestro caso los hilos nos son de gran ayuda, ya que la *Java Virtual Machine* (Máquina Virtual de Java) es un sistema multitarea, es decir, permite la realización de varias tareas al mismo tiempo.

La clase ReproduceGrafica, pintada en amarillo, usa "Runnable, en verde en el diagrama. Dicha interfaz Runnable es necesaria para el uso de threads y sin ella no se podrían representar las animaciones de las gráficas [10], pausarlas, o cambiar la velocidad de reproducción de las mismas mediante una barra de velocidad.

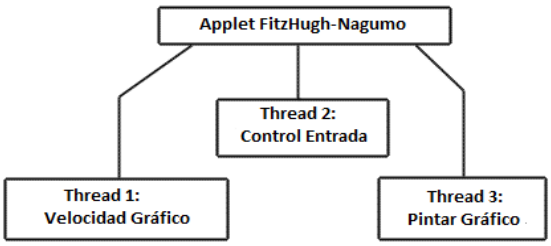


Figura 3.15.- Esquema ejemplo uso threads

Clase panelDibujoFHN

La clase panelDibujoFHN, en amarillo claro en el diagrama de clases, se encargará de pintar los puntos de la gráfica. Para ello, recibirá los puntos de la clase "ventanaPrincipalFHN ".

3.3. Modo de uso

Una vez que se ha cargado y se ha iniciado el applet, podremos comenzar con la reproducción de alguna de las gráficas. El gráfico seleccionado por defecto en el JComboBox del panel del tipo de gráfica es el “Sistema Dinámico FHN”. Para poder cambiar el tipo de gráfica a representar simplemente deberíamos pulsar la pestaña de dicho JComboBox y hacer click en la gráfica deseada. Otro aspecto a destacar es que cada vez que se cambie el tipo de gráfica a representar, el panel de la pantalla de dibujo se limpiará.

Para iniciar la representación de cualquiera de las gráficas se deberá pulsar el botón con el icono de “Play”. Para pausar la reproducción o animación de una gráfica deberemos pulsar el botón de “Pause” y para detener o limpiar la pantalla deberemos hacerlo en el botón de “Stop”. Todos los botones anteriormente mencionados se encuentran en el panel de "Play/Pause/Stop”.

Tipos de representación

La representación de las gráfica puede ser dinámica o estática: la representación estática hace que se dibuje toda la gráfica entera, es decir, como quedaría reflejada en su estado final. Por otro lado, la representación dinámica representa las gráficas mediante una animación. No obstante, si durante la representación de una animación se pulsa el JRadioButton de "Estática", se mostrará la totalidad de la gráfica y, por tanto, finalizará la representación de la misma.

Para elegir entre uno de los dos tipos de reproducción se deberá hacer click en el JRadioButton "Estática, opción preseleccionada al iniciar el applet, o en el JRadioButton "Dinámica" del panel "Características representación". Dichas opciones son excluyentes y siempre habrá un tipo de representación seleccionado.

Por otro lado, la representación dinámica representa las gráficas mediante una animación. Dicha representación se puede pausar, pulsando el botón de "Pause" del panel "Play/Pause/Stop" y para reanudar la reproducción el usuario debería volver a pulsar el botón de "Play".

"Sistema Dinámico FHN".

Como hemos mencionado anteriormente, el tipo de gráfica seleccionada por defecto en el JComboBox del tipo de gráfica es el "Sistema Dinámico FHN". Por tanto, si queremos iniciar la representación sin cambiar las variables, deberemos pulsar el botón del "Play". Por otro lado, si hay otra gráfica diferente que permanece seleccionada en el JComboBox del tipo de gráfica y queremos elegir el gráfico "Sistema Dinámico FHN", lo que deberemos hacer es pinchar en la pestaña y hacer click en la opción con dicho nombre.

Si nos fijamos en la parte superior izquierda del applet, vemos un panel con dos pestañas:

Si queremos dibujar sólo una órbita, nos deberemos fijar únicamente en la titulada como "Órbita1, la primera de ellas. En dicha pestaña nos podemos encontrar varios JTextField, que servirán para que el usuario introduzca los valores de las variables que se recogerán para representar la gráfica del sistema dinámico de FHN perteneciente. Si el usuario quisiera cambiar el valor por defecto de las variables de los JTextFied, lo que hará será hacer click con el ratón en el JTextFied de la variable correspondiente y teclear el valor deseado. Una vez que se han modificado los parámetros deseados para la representación de la gráfica, se deberá pulsar el botón de "Play" y, unos instantes después, aparecerá la gráfica deseada.

Si se quisieran dibujar dos órbitas, además de fijarse en la primera pestaña, deberíamos prestar atención a la segunda de las pestañas, la correspondiente a la "Órbita 2". Dicha pestaña está deshabilitada y para habilitarla y poder dibujar dos órbitas, se debe seleccionar el JCheckBox etiquetado con "Activar órbita2" de la pestaña "Órbita1". Una vez seleccionado, se podrá acceder a los elementos de la segunda órbita pulsando en la pestaña correspondiente. Si nos fijamos en el panel de las variables de la "Órbita2", se puede ver que contiene las mismas variables que la primera de las órbitas y el comportamiento de las cajas de texto que recogen las variables es idéntico a las de la primera de las órbitas. Para iniciar la representación en el caso de las dos órbitas, al igual que en cualquier tipo de representación, deberemos pulsar el botón de "Play".

Un aspecto importante a destacar es el formato de las variables introducidas en los campos de texto: los números decimales separarán la parte entera de la decimal mediante un punto. Para evitar la introducción de valores erróneos para las variables, no se permitirán otros caracteres que no sean números y puntos. No obstante, si el usuario pulsa alguna otra tecla incorrecta, se podrá ver que las teclas pulsadas se marcaran pero automáticamente se borrarán del campo de texto. Esto se ha conseguido mediante el control de las teclas que el usuario pulsa en los campos de texto.

Si nos fijamos en el color en que se dibujan las gráficas del "Sistema Dinámico FHN", vemos que se dibuja por defecto en rojo, en el caso de que sólo se dibuje una órbita; o rojo y azul para la primera y segunda órbita, respectivamente, si se quieren representar las dos órbitas. Estos colores pueden ser cambiados durante la reproducción de una gráfica o una vez que se haya terminado la representación. Para ello, se tendrá que pulsar el botón de la paleta que hay en el panel titulado "Mostrar". Al hacerlo, aparecerá una ventana con numerosos colores en diferentes tonalidades. Si decidimos cambiar el color, tendremos que pulsar en el color deseado y, a continuación, el botón "Aceptar" de la ventana de los colores. En el caso de que una vez aparecido la ventana de los colores decidamos no cambiar el color de representación de la gráfica, cerraremos la ventana y no se producirá ningún cambio en el color en el que se representa. Por otra parte, si el usuario decidiera cambiar el color de la órbita 1 y la 2, en caso de que haya decidido dibujar dos órbitas, deberá pulsar igualmente el botón de la paleta. Si lo hace, automáticamente se le abrirá una ventana para cambiar el color de la primera de las órbitas y después de que pulse el botón "Aceptar" o cierre la ventana del color de dicha órbita, se abrirá la venta del color de la órbita 2. Si el usuario cambia el color de una de las órbitas y

decide no cambiar el color de la otra, sólo se cambiará el color de la órbita deseada. De esta forma se puede elegir cambiar el color de la primera, segunda o ambas órbitas.

Por otra parte, otro botón que es importante citar es el de la tijera, que se encuentra en el panel de "Mostrar". Dicho botón sirve para que, en caso que se decida animar la gráfica, borre el recorrido anterior dibujado de la gráfica durante la reproducción dinámica de la misma. Esto nos será útil en el caso de que queramos ver la trayectoria dibujada de la gráfica a partir de un determinado momento. Para ello, en el momento justo en el que empiece a interesarnos el recorrido, pulsaremos el botón de la tijera. Esto borrará todo el recorrido anterior de la gráfica y seguirá dibujando sólo a partir del momento en el que este botón se pulsó. Al igual que en el caso del botón del color, este botón se podrá usar tanto para una única órbita como también para dos.

La interfaz incorpora además otras funcionalidades. Entre ellas se encuentran las de ampliar o alejarse del gráfico y la opción de desplazarlo hacia los lados. Para ampliar el gráfico, el usuario deberá pulsar los botones de la lupa con la imagen y un signo positivo o si quiere alejarse pulsará la lupa con el signo negativo. Por otro lado, para desplazar la gráfica se deberán pulsar los botones que hay en el panel titulado "Mover Grafica". Hacia arriba en el botón con la imagen de la flecha hacia arriba; hacia abajo, pulsando el botón con la imagen de la flecha para abajo; hacia la derecha, en botón con la flecha apuntado hacia dicho lado y por último, también hacia la izquierda. Todas estas acciones de desplazamiento y zoom se pueden ejecutar durante, o una vez que se ha terminado la reproducción, y se pueden combinar. Es decir, podemos mover la gráfica hacia un lado y acercarnos o alejarnos de dicho punto si quisiéramos. Para finalizar, hay dos botones que tienen la misma función: volver a representar la gráfica en su posición y tamaño original. Estos botones son los de la "X", que hay entre los botones de desplazamiento, o el botón de la lupa con una "H" en medio, que está entre los botones de zoom de acercarse y alejarse de la gráfica.

Diagramas de bifurcaciones

Para cambiar el tipo de gráfico a exponer se deberá seguir el proceso anteriormente mencionado y seleccionar entre " Diagrama de bifurcaciones A" y " Diagrama de bifurcaciones Φ". Por otro lado, si se ha cambiado el de tipo de gráfica a representar y anteriormente ya había algún tipo de gráfico representado, se limpiará automáticamente la pantalla de dibujo.

Una vez que uno de los diagramas de bifurcaciones ha sido o permanece seleccionado, se verá que el JComboBox correspondiente al intervalo de representación respecto de la "A", en el caso de que el tipo de gráfico elegido sea el " Diagrama de bifurcaciones A", o el intervalo de la Φ, en caso de que el elegido sea "Diagrama de bifurcaciones Φ" del panel de "Intervalos de cálculo", se activarán. Posteriormente, se podrá cambiar, si así se desea, el intervalo de los valores del eje horizontal para el que se dibujará el diagrama. Para ello, se deberá pinchar en la pestaña de la caja del intervalo correspondiente y hacer click en el rango deseado. Cada vez que se seleccione un intervalo diferente, se limpiará el panel de dibujo y quedará en blanco.

Si nos centramos en la parte inferior del panel en el que están los intervalos, se verá que hay un JTextField. Este elemento sirve para indicar el número de puntos para el que se calculará el diagrama de bifurcaciones seleccionado. Dicha caja de texto no permite la introducción de ningún carácter no numérico. De la misma forma que en los JTextField anteriormente mencionados, si se pulsa cualquier otra tecla, el carácter correspondiente a dicha tecla se borrará. Por otro lado, es importante destacar que si se introdujese un número incoherente o erróneo para este campo, como puede ser el "0", una vez que se pulse el botón de "Play" se abrirá una ventana en la que se mostrará que el dato introducido fue no era correcto, pues no tiene sentido calcular, en este caso, el diagrama de bifurcaciones de 0 puntos.

Una vez seguidos los pasos anteriores, se podrá cambiar el valor de las variables para la representación del diagrama de representación en caso de que se requiera. Para ello, se deberán rellenar los JTextField de la órbita1 del panel de las pestañas, ya que el panel correspondiente a la segunda de las órbitas permanecerá inactivo si el tipo de gráfico elegido es un diagrama de bifurcaciones. En el caso de que el tipo de gráfico seleccionado sea el "Diagrama de Bifurcaciones A", el campo de texto correspondiente a la variable "A" del panel de la órbita1 permanecerá inactivo. Esto es debido a que el valor de la variable "A" se recogerá del campo de intervalo del JComboBox de "Intervalo A". De forma análoga, si el diagrama escogido hubiese sido el "Diagrama de bifurcaciones Φ", el campo de la variable "Φ" del panel de la órbita1 permanecerá inactivo, ya que los valores de dicha variable se recogerán del campo de JComboBox de "Intervalo Φ".

Al igual que en el "Sistema Dinámico FHN", se puede cambiar el color de representación de los diagramas de bifurcaciones en reproducción o una vez haya terminado y, para hacerlo, se deberá seguir el mismo proceso que el explicado con anterioridad. Por último, también se podrá desplazar o hacer zoom en la gráfica. La diferencia respecto al funcionamiento de la gráfica de "Sistema Dinámico FHN" radica en que estas acciones se podrán realizar únicamente cuando la representación de los diagramas de bifurcaciones haya terminado.

Aspectos generales de la representación de todos los gráficos

Si nos fijamos en el panel "Mostrar" hay un JCheckBox titulado "Ejes". Dicho elemento sirve para dibujar o borrar los ejes de coordenadas de los gráficos. Esta acción podrá llevarse a cabo tanto durante la animación de las gráficas como una vez terminadas.

En el caso de que el tipo de reproducción elegida sea la dinámica, se podrá también acelerar y disminuir la velocidad de la animación. Esto se conseguirá desplazando hacia la izquierda, la barra del JSlider del panel de "Características de representación" si se desea acelerar, o hacia la derecha, si se desea frenar la velocidad de reproducción.

Por último, el panel que hay encima del panel de dibujo alojará un campo de texto. Dicho campo de texto dará información al usuario del tipo de gráfica seleccionada; o si se posa el cursor del ratón sobre algunas de las componentes del applet, mostrará información sobre su uso.

3.4. Implementación

En este apartado trataremos de explicarán los aspectos más importantes y relevantes de las clases de las que consta la implementación del applet [4].

Respecto al formato elegido para explicar las partes más importantes del código de la implementación, se mostrarán todos los fragmentos y, posteriormente, en los casos en los que se crea necesario, se incluirá una pequeña sección que explicará el funcionamiento de dicho código.

Clase ventanaPrincipalFHN

La cabecera de la clase es la siguiente:

```
public class ventanaPrincipalFHN extends javax.swing.JApplet {...}
```

Dicha clase heredará de la clase JApplet, ya que lo que se querrá implementar será un applet de Java. Para indicar que la clase hereda de una clase se debe poner la palabra "extends" y a continuación el nombre de la clase heredada.

Esta clase, llevará a cabo varias funciones importantes para el applet, entre ellas destacan las siguientes:

- Realizar los cálculos necesarios para mostrar los diferentes gráficos. Para ello, se deberán calcular las aproximaciones de Runge-Kutta del tipo de gráfico elegido y ajustar los valores de los puntos obtenidos al panel de dibujo para que se dibujen en él de una manera limpia y correcta.

- Controlar los eventos o el comportamiento de las distintas funcionalidades de la interfaz del applet. Es decir, la clase "ventanaPrincipalFHN" describirá cuáles son las acciones que se deberán desencadenar cuando se utilicen las diferentes elementos de la interfaz [9]. Ejemplos de dichas acciones serían la acción que se producirá si un usuario pulsa el botón del "Play", o lo que pasará si un usuario desplaza la barra de velocidad durante la reproducción dinámica de una gráfica.

En cuanto a los elementos de la clase "ventanaPrincipalFHN", podemos destacar varios. El primer de los métodos encontrados en la clase es el método "init", cuyo esqueleto es el que mostramos:

Explicación 1

```
public void init() {
        try {
              java.awt.EventQueue.invokeAndWait(new Runnable() {
         public void run() {
                    initComponents();
```

```
                    ...
                    En esta parte se asociarían las clases, en las que se describen las
                    acciones provocadas  cuando el usuario interactúa con los diferen -
                    tés elementos de la interfaz , con los elementos de la interfaz
                    que lo provocan. Un ejemplo: en esta parte se asignará la clase en la
                    que se describe cómo pintar un gráfico con el botón del "Play", para
                    que cuando se pulse el botón de "Play" se dibuje un gráfico
                                        ...
                     }
            });
        } catch (Exception ex) {
                    ex.printStackTrace();}}
```

Funcionamiento: el método initComponents(), que está a su vez dentro del método init anterior, sirve para establecer la disposición de todos los elementos de la interfaz gráfica en el applet, es decir, donde estarán situados.

Explicación 2

Ahora procederemos a explicar el método ponEcuacion, que servirá para calcular los puntos de la gráfica del sistema dinámico de FHN. Para ello se deberá usar el método de aproximación de Runge-Kutta.

```
//_________METODOS ECUACIÖN DE FHN_______

public double Q1(double x, double y,double a){
        return (y - b*x +a);
}

public double StPrima(double t,double eps,double r,double T,double fi){
        return (eps*Math.sin((2* Math.PI*r*t)/T)+((fi*Math.PI)/180));
}

public double St(double t,double A,double T,double eps,double r,double fi){
        return ((A* (1+StPrima(t,eps,r,T,fi)))*Math.sin((2* Math.PI /T)* t));
}

public double Q2(double x, double y, double t,double fi,double eps,double T,double r ,double A){
        return (-x *c +c*y -(c*y*y*y)/3 + c * St(t,A,T,eps,r,fi));
}

public void ponEcuacion(Point2D [] puntos,double anchoEjeXFhn , double altoEjeYFhn,
        double a, double b,double c,double T,double A,double eps,double r, double fi ){
```

```
 double kx1,kx2,kx3,kx4,ky1,ky2,ky3,ky4;
//condiciones iniciales para calcular la aproximación de Runge-Kutta
 double x=0.1;
 double y=0.1;

//como es un sistema dinámico, la función variará en el tiempo. Deberemos iniciar el   tiempo
double tiempo=0.0;
double xTransformado;
double yTransformado;
int i;
double maximoYFhn=Double.MIN_VALUE;
double minimoYFhn=Double.MAX_VALUE;

double maximoXFhn=Double.MIN_VALUE;
double minimoXFhn=Double.MAX_VALUE;
 for (i=0;i<aux;i++){
            kx1=Q1(x,y,a);
             ky1=Q2(x,y,tiempo,fi,eps,T,r,A);
            kx2=Q1(x+H2*kx1,y+H2*ky1,a);
            ky2=Q2(x+H2*kx1,y+H2*ky1,tiempo+H2,fi,eps,T,r,A);
            kx3=Q1(x+H2*kx2,y+H2*ky2,a);
            ky3=Q2(x+H2*kx2,y+H2*ky2,tiempo+H2,fi,eps,T,r,A);
             kx4=Q1(x+h*kx3,y+h*ky3,a);k
            y4=Q2(x+h*kx3,y+h*ky3,tiempo+h,fi,eps,T,r,A);
            x=x+ H6*(kx1+2.0*kx2+2.0*kx3+kx4);
            y=y+ H6*(ky1+2.0*ky2+2.0*ky3+ky4);
            maximoYFhn=Math.max(maximoYFhn,y);
            minimoYFhn=Math.min(minimoYFhn,y);
            maximoXFhn=Math.max(maximoXFhn,x);
            minimoXFhn=Math.min(minimoXFhn,x);
            puntosSinAjustar[i] = new Point2D.Double (y,x);
             tiempo=tiempo+h;
}
 maximoYFhn=Math.max(maximoYFhn,Math.abs(minimoYFhn));
maximoXFhn=Math.max(maximoXFhn,Math.abs(minimoXFhn));

 //Math.ceil() nos devolverá el número entero inmediatamente superior al pasado como
parámetro
double maximoEjeXFhn=Math.ceil(maximoYFhn);
double maximoEjeYFhn=Math.ceil(maximoXFhn);

/*Estas dos medida será necesarias para que las unidades de los ejes x's e y's sean
proporcionales la cota de los ejes de de coordenadas vendrán dadas por el máximo de  los
valores inferiores*/
panelDibujoFHN.maximoTotal=Math.max(maximoEjeXFhn,maximoEjeYFhn);

/*cojo la menor de las medidas para que el alto y el ancho quepa en nuestro panel y
que nuestras unidades del eje de las X's y el eje de las Y's sean iguales, al igual que el     tamaño
```

```
de nuestros ejes de coordenadas*/
double medidaAjustarEje=Math.min(anchoEjeXFhn,altoEjeYFhn);

for(int j=0;j<aux;j++){

xTransformado=TransformaCoordenadaX(puntosSinAjustar[j].getX(),medidaAjustarEje,
panelDibujoFHN.maximoTotal);

yTransformado=TransformaCoordenadaY(puntosSinAjustar[j].getY(),medidaAjustarEje,
panelDibujoFHN.maximoTotal);
puntos[j] = new Point2D.Double (xTransformado,yTransformado);}}
```

Funcionamiento: Este método calcula los puntos para pintar una gráfica del "Sistema Dinámico FHN". Durante el proceso de cálculo de los puntos, hay varios métodos que son usados: Q2, St, StPrima y Q1. Dichos métodos sirven para descomponer la ecuación del sistema de FHN en métodos más pequeños y así hacer el código más comprensible y manejable.

Por otro lado, para cada uno de los puntos calculados se deberá calcular la aproximación Runge-Kutta de cuarto orden. Por ello, se usarán varias variables intermedias correspondientes a las pendientes intermedias de cada intervalo. Dichas variables son: kx1, kx2, kx3, kx4, ky1, ky2, ky3 y ky4, para las X's y las Y's, respectivamente.

Hay varias variables, como maximoYFhn o maximoXFhn, que sirven para calcular el máximo valor de las variables X's y de las variables Y's durante todo el proceso de cálculo de puntos. Posteriormente, una vez calculado el valor de todos los puntos, se averiguará qué variable entre "maximoYFhn" y "maximoXFhn" es mayor. Con dicho valor se podrán ajustar los tamaños de los ejes de coordenadas horizontal y vertical y, además, que ambos tengan dibujados el mismo número de ticks y unidades.

En Java, el inicio del eje de coordenadas está en la esquina superior izquierda del panel de dibujo y los valores de las Y's aumentan según descendemos en la pantalla. Sin embargo, nuestro sistema de coordenadas, el humano, tiene como origen de coordenadas en el centro del panel de dibujo y el valor de las Y's aumenta según vamos hacia arriba en el panel. Para adaptar el "sistema de coordenadas de Java" al sistema de "coordenadas humano" y que la gráfica esté centrada en el panel de dibujo, se usan los métodos "TransformaCoordenadaX" y "TransformaCoordenadaY".

El contenido de dichos métodos es el siguiente:

```
public double TransformaCoordenadaX(double xATransformar,double
                                     medidaAjustarEje,double maximo){
```

```
                double auxX= (Math.abs(xATransformar) * (medidaAjustarEje/2.0)/maximo);
                    if (xATransformar>0){
                            return ((panelDibujoFHN.getWidth()/2.0) + auxX);
                    }else if(xATransformar<0){
                             return ((panelDibujoFHN.getWidth()/2.0) - auxX);
                    }else{
                            return (panelDibujoFHN.getWidth()/2.0);
                     }
}


//Ajusta la coordenada de entrada "y" al panel de dibujo panelDibujoFHN"
public double TransformaCoordenadaY(double y,double medidaAjustarEje,double maximo){
        //Establezco una regla de tres para saber en qué punto se debería establecer nuestro punto de
        entrada
        double auxY=((Math.abs(y)* medidaAjustarEje/2.0)/maximo);
        if (y<0){
                 return panelDibujoFHN.getHeight()/2.0+auxY;
        }else if (y>0){
                 return panelDibujoFHN.getHeight()/2.0-auxY;
        }else{
                return panelDibujoFHN.getHeight()/2.0;
        }
}
```

Funcionamiento: Nos centraremos en el funcionamiento del método del método "TransformaCoordenadaX": si el valor de la x es positivo, se deberá sumar el valor del punto a la mitad del valor de la dimensión del eje de las X's. Esto servirá para adaptar el punto al sistema de coordenadas humano, cuyo origen está en el centro del panel. No obstante, si el valor del punto es negativo, se deberá restar a la mitad del eje de coordenadas de las X's, que es el centro de la pantalla, el valor de la variable x. Finalmente, si la variable tiene el valor 0 nos deberemos quedar en el centro de la pantalla.

El valor calculado como "auxX" corresponde a una regla de tres. Dicha regla de tres sería la siguiente: si el valor máximo de las x (maximo) le corresponde la mitad del eje de las X's (medidaAjustarEje), entonces a mi valor de la x de entrada le corresponde el valor...

El método para ajustar las Y's sería análogo al explicado. Para que ajuste los puntos al sistema de coordenadas humano deberé sumar la mitad del alto de la pantalla a cada uno de los puntos de entrada hallados.

Explicación 4

Ahora explicaremos la clase que describe lo que ocurrirá cuando el usuario pulse el botón del "Play":

```
public class ReproduceGrafica implements ActionListener,Runnable{
public void actionPerformed(ActionEvent e) {
        if (TodosJTexFieldsCorrectos()){
                if (!pausado){
                        ...
                    if ((!dibujaBifurcacionFi)&&(!dibujaBifurcacionAmplitud)){
                      if (!quiereOrbita2){
                          panelDibujoFHN.quiereOrbita2=false;
                              //dejaré un margen de 20 pixels a cada lado del ancho del
                               diagrama de bifurcación
                              anchoEjeXFhn=panelDibujoFHN.getWidth()-40;
                              altoEjeYFhn=panelDibujoFHN.getHeight()-40;
                              ...
                              //calculo los puntos que se representarán y ajusto los
                                        puntos al panel para que se dibujen correctamente
                              ponEcuacion
                                        (puntos,anchoEjeXFhn,altoEjeYFhn,a,b,c,T,A,eps,r,fi);

                                        ...

                              //Esto me servirá para poder hacer el zoom de la ecuación con las
                               proporciones adecuadas (el zoom lo haremos según las
                               coordenadas del punto inicial)
                               principioX=(int)puntos[0].getX();

                               //Averiguamos la distancia del punto inicial al eje de las y's, que
                               será en la mitad del ancho del panel
                               distAejeY=principioX-(panelDibujoFHN.getWidth()/2);
                               principioY=(int)puntos[0].getY();

                               //Averiguamos la distancia del punto inicial al eje de las x's, que
                               será en la mitad del alto del panel
                               distAejeX=panelDibujoFHN.getHeight()/2.0 -principioY;

                               //guardo las distancia del punto inicial al eje horizontal y vertical
                              de coordenadas respectivamente
                               panelDibujoFHN.distAejeX=distAejeX;
                               panelDibujoFHN.distAejeY=distAejeY;

                               //Copio los datos en el array de los puntos del array del panel
                               panelDibujoFHN.puntos= new Point2D[aux];

                              System.arraycopy(puntos,0,panelDibujoFHN.puntos, 0,
                              puntos.length);
                }else{
                                        ...
```

```
            }
                                    ...
            //la línea inferior sirve para iniciar un hilo. Automáticamente
            pasará a ejecutarse el método "Run" de la parte inferior
            thread.start();
    }}}
```

Funcionamiento: si nos fijamos en el método, se puede ver un fragmento resaltado en negrita. Dicho segmento tiene utilidad únicamente para el tipo de gráfico "sistema dinámico FHN". Como comentamos anteriormente, los puntos aproximados mediante el método de Runge-Kutta se ajustan al panel de dibujo según el tamaño del panel.

Para ajustar la coordenada X de todos los puntos hallados, debemos calcular el tamaño inicial del eje de coordenadas horizontal. Hemos querido dejar un margen en el lado derecho e izquierdo del panel, por lo que el extremos izquierdo del eje horizontal de coordenadas comenzará 20 pixels más a la derecha del comienzo de la parte izquierda del panel y terminará 20 pixels antes del extremo derecho del panel .

Además, para ajustar las coordenadas Y's de los puntos hallados, dejaremos 20 pixels de margen tanto en la parte superior del panel como en la inferior. Lo planteado se resume en las siguientes líneas:

```
anchoEjeXFhn=panelDibujoFHN.getWidth()-40;
altoEjeYFhn=panelDibujoFHN.getHeight()-40;
        ...
//calculo los puntos que se representarán y ajusto los puntos al
panel para que se dibujen correctamente
ponEcuacion(puntos,anchoEjeXFhn,altoEjeYFhn,a,b,c,T,A,eps,r,fi);
```

Si nos fijamos en la parte inferior del método "actionPerformed" anterior, hay una línea cuyo contenido es "thread.start()". Dicho comando hará que se comience a ejecutar el método "Run" de la clase, que mostramos en la parte inferior.

```
public void run() {

    if ((!dibujaBifurcacionFi)&&(!dibujaBifurcacionAmplitud)) {
        if (!quiereOrbita2){
                //servirá para indicar el número de puntos que se dibujarán en el
                panel
```

```
indice=0;
...
while ((indice<puntos.length)&&(noTerminada)){
 //miraré si el usuario ha pulsado el "tick" de representación
de los ejes de coordenadas

                panelDibujoFHN.quiereOrbita2=false;

                panelDibujoFHN.conEjes=jCheckBoxEjes.isSelected();
                        if (jRadioButtonEstatica.isSelected()){
                                //Si la representación es estática se deberán
                                representar todos los puntos de golpe
                                panelDibujoFHN.indicePuntos=puntos.length;
                                barraProgreso.setStringPainted(true);
                                barraProgreso.setValue(100);
                                panelDibujoFHN.repaint();
                                botonPause.setEnabled(false);
                        }else{
                                /aumento el valor de indice para que cada
                                vez se vayan dibujando más puntos en la
                                gráfica
                                botonPause.setEnabled(true);
                                panelDibujoFHN.indicePuntos=indice;
                                barraProgreso.setStringPainted(true);
                                barraProgreso.
                                setValue(porcentajeProgreso(indice));
                                panelDibujoFHN.repaint();
                        try {
                          /*recojo el tiempo de retardo indicado por la
                          barra de velocidad de representación y así
                          hacer que el tiempo entre un dibujo sea más o
                            menos grande */
                                Thread.sleep(tiempoRetardo);
                        }
                        catch (InterruptedException e) {
                                return;
                        }
                    }
                    indice++;
                }

 noTerminada=false;
...
//como ya hemos terminado de dibujar no nos hace falta el hilo y por
lo tanto detenemos o "matamos" al hilo
thread.stop();
}else{
```

```
                //Para la órbita 2 el proceso sería análogo, por lo que omitiremos esta parte
                ...
        }
            }else if (dibujaBifurcacionFi){
                ...
            }else{
        ...
        }}}
```

Funcionamiento: Si nos fijamos en la parte en negrita del método Run veremos un bucle. Dicho bucle está estructurado en dos partes: la primera es de la opción de reproducción estática y la segunda es cuando la reproducción elegida fue la dinámica.

Para dibujar las gráficas se debe indicar al panel de dibujo, la clase encargada de pintar, el número de puntos de la gráfica que se deberán mostrar. En la reproducción dinámica, en cada iteración del bucle se irá incrementando en uno el índice de números de puntos que deberá pintar el panel de dibujo. Dicha orden, la de pintar, será llevada a cabo mediante la línea "panelDibujoFHN.repaint()". No obstante, cuando la opción seleccionada para el tipo de representación es la estática, el número de puntos a pintar será la totalidad de la gráfica y, para que el bucle deje de iterar, en nuestro caso incluimos la orden "noTerminado=True". El fragmento del bucle mencionado es el siguiente:

```
while ((indice<puntos.length)&&(noTerminada)){
        ...
        if (jRadioButtonEstatica.isSelected()){
                ...
                panelDibujoFHN.indicePuntos=puntos.length;
                panelDibujoFHN.repaint();
                noTerminada=True;
        else{
                panelDibujoFHN.indicePuntos=indice;
                panelDibujoFHN.repaint();

                ...
        }
        indice++;
}
```

Explicación 5

El siguiente método goza de gran importancia para la representación del diagrama de bifurcaciones respecto de Φ.

```
 //este método me servirá para poder calcular los datos de la bifurcación respecto de Fi    public
ArrayList CalculaBifurcacion(){
        double numeroACalcular=aux+numeroPuntosACalcular;
        ...
        backupLargoEjeX=(panelDibujoFHN.getHeight()-80);
         largoEjeX=backupLargoEjeX;
         //hemos situado el inicio del eje X (eje vertical) en la parte superior del panel
        inicioEjeX=(panelDibujoFHN.getHeight()/2.0)-(largoEjeX/2.0);
         double finalEjeX=panelDibujoFHN.getHeight()/2.0+largoEjeX/2.0;
        alturaEjeHorizontal=finalEjeX;

        //Pongo un margen de 60 pixels a la derecha e izquierda
        backupAnchoEjeFi=panelDibujoFHN.getWidth()-120;
        anchoEjeFi=backupAnchoEjeFi;
        inicioEjeFi=(panelDibujoFHN.getWidth()/2.0)-anchoEjeFi/2.0;
        panelDibujoFHN.inicioFi=inicioFi;
        panelDibujoFHN.finFi=finFi;

                ...
        //declaración de variables
                ...
         //inicializamos las variables que vamos a usar para hacer los cálculos
        //nos servirá para saber por qué iteración vamos en el array donde guardamos los datos

        contadorFi=0;

         //En esta lista auxiliar vendrán los elementos sin estar transformados
         maximosSinTranformar=new ArrayList();
        maximo=Float.MIN_VALUE;
         minimo=Float.MAX_VALUE;

                ...

         //este bucle se ejecutara para cada valor de fi
         for (fi=inicioFi;fi<finFi;fi=fi+incrementoFi){
                x=0.1;
                y=0.1;
                tiempo=0.0;
                  ...
                //se calcula la aproximación mediante Runge-Kutta
          ...

                //hallo cuáles son los puntos máximos de la iteración anteriormente
                almacenada
                arrayDeMaximos=calcularMaximos(xSinTransformacion,ultimoElemento);
                maximosSinTranformar.add(arrayDeMaximos);
                fiSinTransformacion[contadorFi]=fi;
                contadorFi++;
        }
```

```
//Ajusto los valores de las Fi en el eje horizontal
fiYaAjustadas=ajustarTodasFi(fiSinTransformacion,inicioEjeFi,anchoEjeFi,contadorFi);

//Ajusto los valores de las X's en el eje vertical y se lo paso a la variable del panel de  dibujo
panelDibujoFHN.ListaElementos=AjustarTodasX(maximosSinTranformar,largoEjeX,
                                                    finalEjeX,Math.ceil(maximo));

 //Esta variable servirá para decir cual es entero inmediatamente superior al máximo
panelDibujoFHN.maximaXDiagramas=Math.ceil(maximo);

panelDibujoFHN.fiTransformados=new double[fiYaAjustadas.length];
//Copio los valores de las fi ajustadas a una variable en el panel de dibujo
System.arraycopy(fiYaAjustadas,0,panelDibujoFHN.fiTransformados,0,
                                                    fiYaAjustadas.length);

return panelDibujoFHN.ListaElementos; }
```

Funcionamiento: este método sirve para calcular los puntos que se dibujarán en el diagrama de bifurcaciones respecto de Φ. Hemos decidido que el eje vertical de coordenadas, el eje de las X para este tipo de gráfico, tenga un margen en la parte superior e inferior del panel. Dicho margen será de 40 pixeles en ambos extremos, por lo que el eje vertical tendrá las longitudes del alto del panel menos los dos márgenes, es decir, 80 pixeles. Por tanto, el eje vertical de coordenadas del diagrama de bifurcaciones empezará 40 pixeles más abajo de la parte superior del panel y terminará 40 pixeles antes del extremos inferior del panel. Esto es:

```
backupLargoEjeX=(panelDibujoFHN.getHeight()-80);
largoEjeX=backupLargoEjeX;
```

Por otro lado, se dejarán 60 pixeles en los extremos derecho e izquierdo del panel, por lo que el eje horizontal de coordenadas, el de la Φ, empezará 60 pixeles a la derecha del extremos izquierdo del panel de dibujo y terminará 60 pixeles antes del extremo derecho del panel de dibujo. Es decir:

```
backupAnchoEjeFi=panelDibujoFHN.getWidth()-120;
anchoEjeFi=backupAnchoEjeFi;
```

Para ajustar las coordenadas de los puntos a los ejes de coordenadas del diagrama de bifurcaciones tenemos los métodos "AjustarTodasFi" y "AjustarTodasX, que a su vez se descomponen en otros métodos más simples. Estos métodos son los siguientes:

```
 //Este método ajusta el valor de "valorPuntoEnX" para poder dibujarlo en el diagrama de bifurcaciones
en el eje vertical
public double AjustaUnaSolaX(double valorPuntoEnX,double largoEje,double maximo){
```

```
        double  altoEje =largoEje;
        double  mitadAltoEje=(altoEje/2.0);


        /*En este caso "maximo" será el valor superior entero más próximo al máximo
        valor de las X's halladas durante las iteraciones del método
        "calculaBifurcación". Hacemos una regla de tres para saber dónde colocar el
        nuevo punto:"Si el punto máximo ocuparía el final del la mitad del eje vertical
        entonces mi punto ocuparía ...*/
        double alturaAbsoluta=Math.abs(valorPuntoEnX)*(mitadAltoEje/maximo);

        /*El punto 0 está en el medio del eje de las X's (eje vertical. Por tanto, si "valorPuntoEnX" es
        positivo, deberé situar el valor del punto en la parte superior del punto 0. En cambio, si
        "valorPuntoEnX" es negativo, deberé situar el punto en la parte inferior del eje del punto 0*/
        if (valorPuntoEnX>0){
                return (mitadAltoEje-alturaAbsoluta+inicioEjeX);
        }else if (valorPuntoEnX==0){
                return (inicioEjeX+(largoEjeX/2.0));
        }else{
               return ((mitadAltoEje+alturaAbsoluta)+inicioEjeX);
        }
}

 /*Este método ajusta los puntos de las X's para un vector; es decir, calcula el valor de todas las x para
un único valor de Φ o de la Amplitud, dependiendo del tipo de diagrama de bifurcaciones*/
public double[] AjustarIteracionX(double[] xSinTransformacion,double largoEje,double maximo){

        double[] arraySalida=new double[xSinTransformacion.length];
        for ( int i=0;i<xSinTransformacion.length;i++){
                arraySalida[i]= AjustaUnaSolaX(xSinTransformacion[i],largoEje,maximo);
        }
        return arraySalida;
}

 //servirá para ajustar todas las X's en el eje vertical de los diagramas de bifurcaciones.
public ArrayList AjustarTodasX(ArrayList ListaXSinTransformacion,double largoEje,double
        alturaEjeHorizontal,double maximo){
        ArrayList ListaXajustadas=new ArrayList();
        double[] xSinTransformacion;
        //Le solicito a la lista que me devuelva un iterador con todos los  elementos contenidos en ella

        for ( Iterator iterador = ListaXSinTransformacion.listIterator();iterador.hasNext(); ) {
                //cojo el elemento apuntado por "iterador"
                xSinTransformacion=(double[])iterador.next();
                double[] arraySalida=AjustarIteracionX(xSinTransformacion,largoEje,maximo);
                ListaXajustadas.add(arraySalida);
        }
        return ListaXajustadas;
}
```

```
//ajusta el valor de una sola  "fi" (eje horizontal) para poder representarlas en un diagrama de bifurcaciones en el eje horizontal. Es una regla de tres: si el valor máximo de fi ocupa el ancho del eje Fi, entonces el valor de entrada le corresponde la posición en el eje de las fi...)
public double AjustaUnaSolaFi(double fi, double inicioEjeFi,double anchoEjeFi,int contadorFi){
        return ((double)fi*(double)(anchoEjeFi/contadorFi) +(double)inicioEjeFi);                    }

//ajusta todas las "fi" para situarlas en el diagrama de bifurcaciones
        public double[] ajustarTodasFi(double[] fiSinTransformacion,double inicioEjeFi,double
                                                                anchoEjeFi,int contadorFi ){

                for(int i=0;i<fiSinTransformacion.length;i++){
                        fiTransformado[i]=AjustaUnaSolaFi(i,inicioEjeFi,anchoEjeFi,contadorFi);
                }
                return fiTransformado;     }
```

panelDibujoFHN

Esta clase, llevará a cabo varias funciones, cuya función más importante es :

- Dibujar los puntos recibidos de la clase "ventanaPrincipalFHN". Es decir, dibujar los distintos gráficos. Para dibujar cada uno de los gráficos elegidos la clase calcula en qué puntos irán dibujados los ticks que indican las unidades de los ejes.

Es necesario, para el entendimiento de la clase, que nos centremos en las siguientes secciones :

```
public void PintaEjes ( Graphics g,double inicioX,double inicioY){

        //Como dijimos anteriormente, el ancho del eje de coordenadas horizontal tendrá una longitud igual al ancho del panel menos los márgenes, es decir, 80 pixeles correspondientes al total de pixeles para los márgenes.
        double altoPanel=getHeight();
        double anchoPanel=getWidth();
        double antiguoAncho=anchoPanel-40;
        double antiguoAlto=altoPanel-40;

       /*hayo la dimensión menor de los ejes de coordenadas para escoger la más pequeña  y  así saber que los ejes, cuando se representen, lo harán con el mismo tamaño y las unidades */
        antiguoAncho=Math.min(antiguoAncho, antiguoAlto);
        antiguoAlto=antiguoAncho;

         //Si se ha pulsado el botón de zoom cambiará el tamaño de los ejes
        nuevoAncho= antiguoAncho*contadorZoom;
        nuevoAlto= antiguoAlto*contadorZoom;
```

```
double mitadNuevoAncho= nuevoAncho/2.0;
double mitadNuevoAlto=nuevoAlto/2.0;

//ahora estableceremos la distancia del punto inicial al eje de las X's y al eje de las
Y's respectivamente por si se ha pulsado el zoom
double nuevoDistAejeX=distAejeX*contadorZoom;
double nuevoDistAejeY=distAejeY*contadorZoom;

//hallo en qué punto se encuentra la mitad del eje X nuevo, es decir, la coordenada
horizontal donde se sitúa el eje vertical de coordenadas
double posXEjey=inicioX-nuevoDistAejeY;

 //pixels en los que empieza nuestro eje horizontal de coordenadas
double inicioNuevoX=posXEjey-mitadNuevoAncho;
double finalNuevoX=posXEjey+mitadNuevoAncho;

//hallo en qué punto se encuentra la mitad del eje Y nuevo, es decir, la coordenada
vertical donde se sitúa el eje horizontal de coordenadas
double posYEjex=inicioY+nuevoDistAejeX;

//la parte de arriba del eje vertical de coordenadas
double inicioNuevoY=posYEjex-mitadNuevoAlto;

//la parte de inferior del eje vertical de coordenadas
 double finalNuevoY=posYEjex+mitadNuevoAlto;

//Ahora determinare la altura a la que se dibujará el eje de las X's y la anchura a la
//se dibujará el eje de las Y`s
 double alturaEjeX=posYEjex;
 double anchuraEjeY=posXEjey;
 g.setColor(Color.BLACK);

 //Dibujo el eje de la X
g.drawLine((int)inicioNuevoX,(int)alturaEjeX,(int)finalNuevoX,(int)alturaEjeX);

//Dibujo el eje de la Y
 g.drawLine((int)anchuraEjeY,(int)inicioNuevoY,(int)anchuraEjeY,(int)finalNuevoY);

//averiguo el numero de pixels necesarios para dividir los ejes de coordenadas en
unidades
 int numeroPixelsX=Math.round((int)CalculaNumeroPixelsX());
 int mitadNumeroPixelsX=Math.round(numeroPixelsX/2);
 int numeroPixelsY=Math.round((int)CalculaNumeroPixelsY());
 int mitadNumeroPixelsY= Math.round(numeroPixelsY/2);
 int i =0;
 int contador=0;

//Dibujamos los TICKS GRANDES correspondientes a las unidades SEMIEJE NEGATIVO  DE LAS
X'S
for (int j =(int)posXEjey;j>inicioNuevoX; j--){
        if (i > 0){
```

```
                        if ((((i%numeroPixelsX)==0))&&(contador>(-maximoTotal+1))){
                                contador--;
                                g.drawLine(j,(int)(alturaEjeX-6),j,(int)(alturaEjeX+6));
                                 g.drawString(""+contador,j-7,((int)(alturaEjeX+20)));
                        }
                    }
                    i++;
                }

                //Dibujamos los ticks  PEQUEÑOS DEL SEMIEJE NEGATIVO DE LAS X'S
                int k=(int)posXEjey-(mitadNumeroPixelsX);
                while (k>inicioNuevoX){
                        g.drawLine(k,(int)(alturaEjeX-3),k,(int)(alturaEjeX+3));
                        k=k-numeroPixelsX;
                }

                        ...
}
```

Funcionamiento: si nos fijamos en el código anteriormente expuesto podemos ver un fragmento resaltado en negrita. La función de esa parte es determinar los puntos en los cuales comenzarán los ejes de coordenadas para el tipo de gráfico "Sistema Dinámico FHN".

Para que los ejes de coordenadas tengan el mismo tamaño, se debe escoger la menor de las dimensiones entre los ejes de coordenadas de las X's y el de las Y's, pues si cogemos el mayor de ambos ejes, puede que uno de los ejes no se pueda dibujar en su totalidad en el plano de dibujo porque no quepa. Esto se muestra en la siguiente línea:

```
antiguoAncho=Math.min(antiguoAncho, antiguoAlto);
```

En la interfaz de la aplicación nos podemos acercar o alejar de la gráfica haciendo zoom. Para llevar la cuenta del número de veces que se han pulsado dichos botones, tendremos un contador llamado "contadorZoom". Dicho contador nos será muy útil para ajustar el tamaño de los ejes de coordenadas dependiendo del número de veces que se hayan pulsado los botones de zoom. En el código de Java del método anterior son las siguientes líneas:

```
nuevoAncho= antiguoAncho*contadorZoom;
nuevoAlto= antiguoAlto*contadorZoom;
double mitadNuevoAncho= nuevoAncho/2.0;
double mitadNuevoAlto=nuevoAlto/2.0;
```

En el dibujo de la gráfica guardaremos la distancia del punto inicial, el primero calculado, al eje vertical y horizontal de coordenadas en distAejeY y distAejeX, respectivamente. Los tamaños de esas distancias, después de pulsar los botones de zoom, no son los mismos, ya que las dimensiones de los ejes de coordenadas y los puntos dibujados en el plano difieren en tamaño

y posición. Por ello, multiplicaremos la anterior distancia a los ejes por el contador "contadorZoom". Estos cálculos nos serán útiles para que los ejes de coordenadas estén siempre cuadrados en el panel. En el código es lo siguiente:

```
//ahora estableceremos la distancia del punto inicial al eje de las X's y al eje de las Y's respectivamente por si se ha pulsado el zoom
double nuevoDistAejeX=distAejeX*contadorZoom;
double nuevoDistAejeY=distAejeY*contadorZoom;
```

Ahora prestaremos atención a las siguientes dos líneas del método:

```
int numeroPixelsX=Math.round((int)CalculaNumeroPixelsX());
int numeroPixelsY=Math.round((int)CalculaNumeroPixelsY());
```

Estas dos líneas usan los métodos "CalculaNumeroPixelsX" y "CalculaNumeroPixelsY". Estos métodos mostrarán en qué posición se deberán dibujar cada unos de los ticks y unidades en los ejes de coordenadas.

```
public double CalculaNumeroPixelsX(){
        //Ahora hallaremos el numero de pixels necesarios para una unidad en el eje de las X's con una regla de tres
        return ((1.0*nuevoAncho/2.0)/maximoTotal);
}

public double CalculaNumeroPixelsY(){
        //Ahora hallaremos el numero de pixels necesarios para una unidad en el eje de las X's con una regla de tres
        return ((1.0*nuevoAlto/2.0)/maximoTotal);
}
```

El valor devuelto por ambos método corresponde a el cálculo de una regla de tres. Por ejemplo, en el caso de " CalculaNumeroPixelsX", el razonamiento sería: si el punto máximo calculado ocuparía la mitad del ancho del panel, entonces 1 unidad estaría situada en ...

El razonamiento para el método CalculaNumeroPixelsY sería muy similar al explicado de CalculaNumeroPixelsX.

Los ticks que hemos comentados se dibujarán mediante varios bucles, que recorrerán los ejes dibujando en las posiciones indicadas por CalculaNumeroPixelsX y CalculaNumeroPixelsY. Un ejemplo de dicho bucle sería:

```
//Dibujamos los TICKS GRANDES correspondientes a las unidades SEMIEJE NEGATIVO DE LAS X'S
        for (int j =(int)posXEjey;j>inicioNuevoX; j--){
                if (i > 0){
                        if ((((i%numeroPixelsX)==0))&&(contador>(-maximoTotal+1))){
                                contador--;
```

```
                    g.drawLine(j,(int)(alturaEjeX-6),j,(int)(alturaEjeX+6));
                    g.drawString(""+contador,j-7,((int)(alturaEjeX+20)));
                }
            }
            i++;
        }
```

3.5. Movimientos representativos del sistema

En esta sección se mostrarán algunas imágenes de los movimientos más importantes representados en el applet del sistema dinámico de FHN y los diagramas de bifurcaciones.

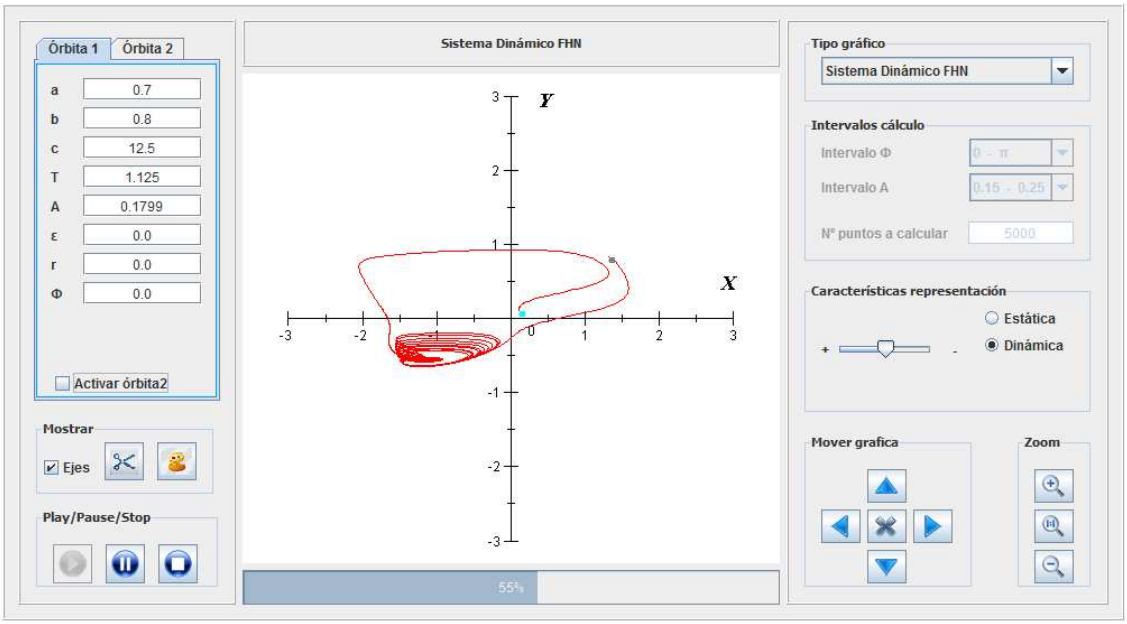


Figura 3.16.- Representación dinámica de una trayectoria para el sistema de FHN

Figura 3.16: esta imagen muestra la interfaz de la aplicación durante la reproducción dinámica de una sola trayectoria del sistema dinámico de FHN

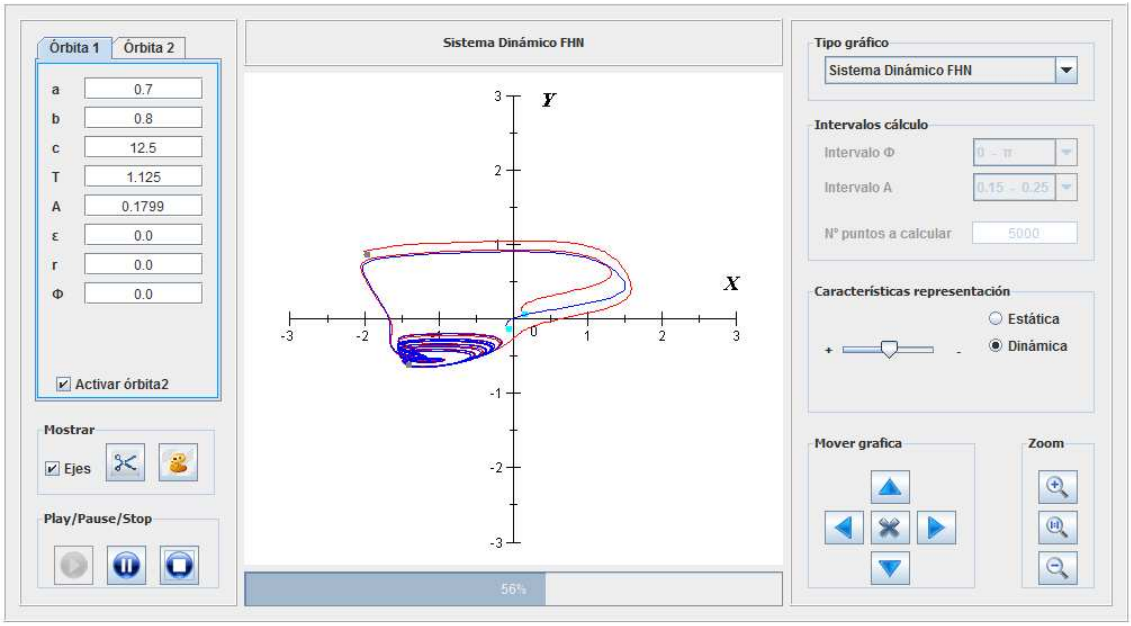


Figura 3.17.- Representación dinámica de dos trayectorias para el sistema de FHN

Figura 3.17: Esta imagen corresponde a la representación dinámica del sistema dinámico de FHN con dos órbitas: la primera, la pintada en rojo, posee los mismos parámetros de entrada que los mostrados en la imagen anterior (Figura 3.16.). Por otro lado, la segunda órbita dibujada, en azul, tiene los mismos parámetros de entrada que los de la primera, exceptuando el valor de la amplitud, que en el caso de la segunda órbita su valor es de 0.1780.

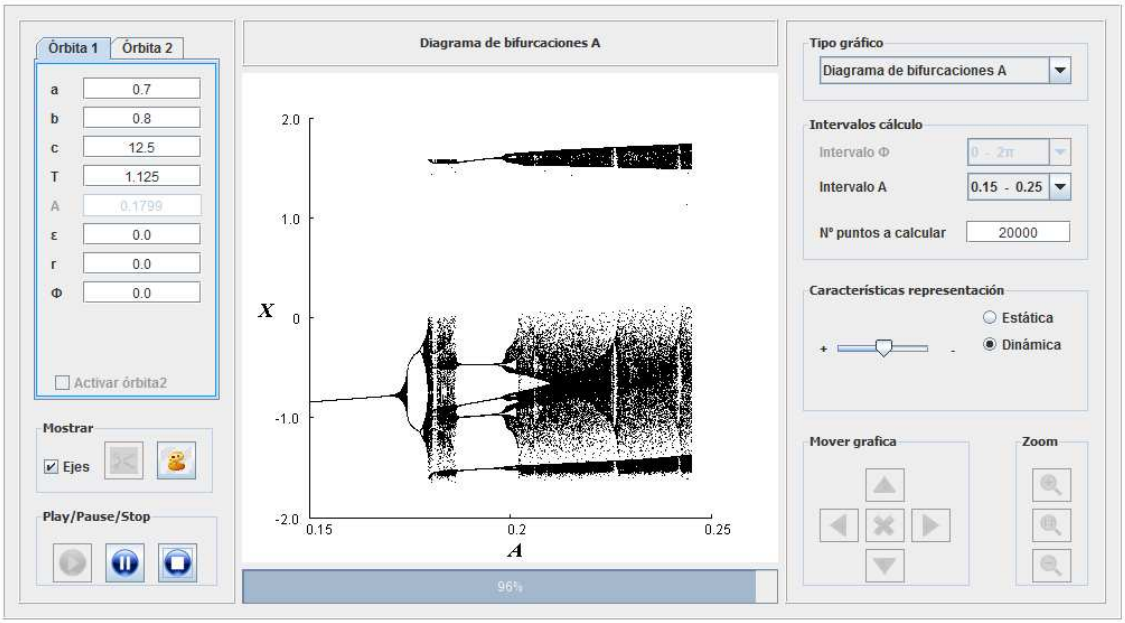


Figura 3.18.- Representación dinámica del diagrama de bifurcaciones respecto de A (0.15, 0.25)

Figura 3.18: Esta imagen corresponde a la representación dinámica del diagrama de bifurcaciones respecto de la amplitud para el intervalo de valores de la amplitud (0.15, 0.25). Si hubiéramos querido aumentar la resolución de la gráfica deberíamos introducir un número entero superior al de la caja de texto de los puntos, que contiene el valor 5000.

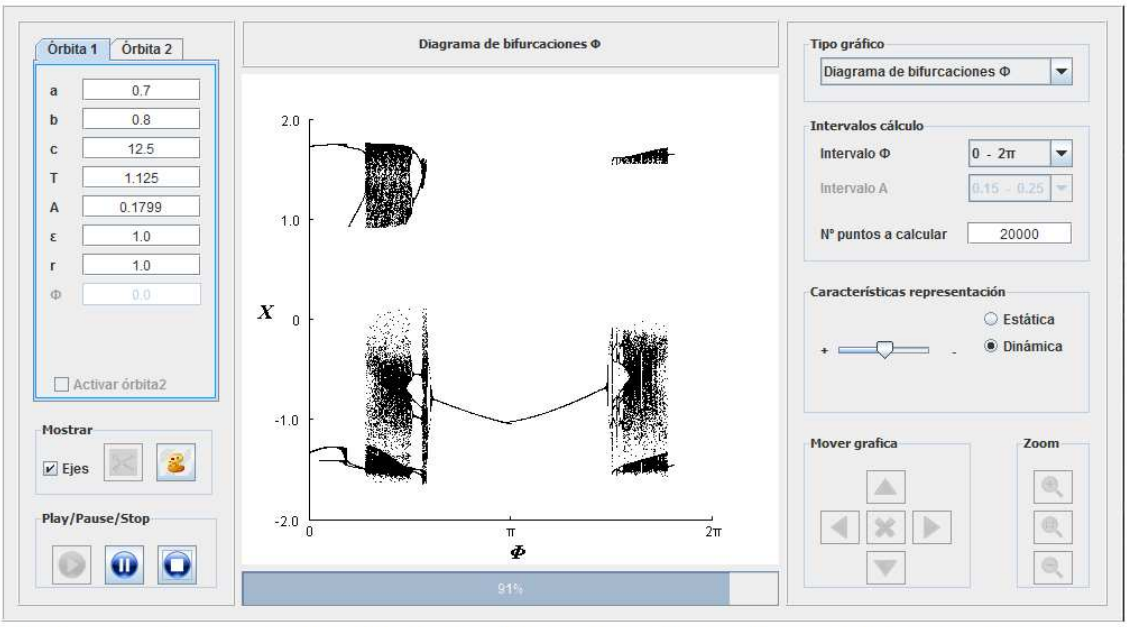


Figura 3.19.- Representación dinámica del diagrama de bifurcaciones respecto de Φ (0 , 2π)

Figura 3.19: Esta imagen corresponde a la representación dinámica del diagrama de bifurcaciones respecto de la fase Φ para el intervalo de valores de Φ de (0,2π).

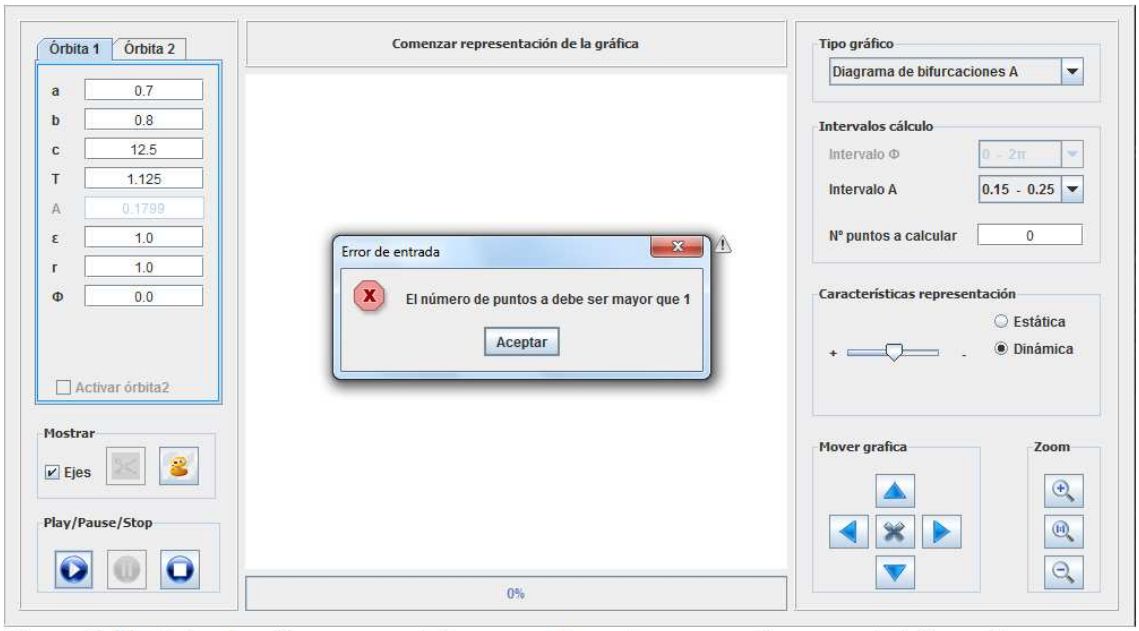


Figura 3.20.- Introducción erronea del numero de puntos en un diagrama de bifurcaciones

Figura 3.20: Esta imagen muestra uno de los mensajes de error mostrados en el caso de que se haya introducido un parámetro erróneo en la caja de texto del número de puntos del diagrama de bifurcaciones.

4. Resultados y conclusiones

El principal objetivo del proyecto de final de carrera ha sido la representación del sistema dinámico de FHN. Por otro lado, la finalidad de la memoria ha sido comentar los pasos más importantes que se han seguido para la implementación de la interfaz del applet y explicar su modo de uso. En resumen, los resultados obtenidos más destacados son los siguientes:

- Se ha conseguido la representación del Sistema Dinámico de FHN mediante un applet de Java. conseguir
- En dicho applet se consigue representar la trayectoria de dicho sistema tanto de una como de dos órbitas, así como los Diagramas de Bifurcaciones en los que se muestran los diferentes comportamientos y la dinámica de nuestro sistema.
- Por último es importante citar que se ha conseguido la implementación de un Control en el Sistema de FHN, que nos permite inducir a nuestro Sistema Dinámico en el régimen deseado.

4.1. Logros principales conseguidos

El logro personal que considero más importante sería la implementación de la representación del Sistema Dinámico de FHN, ya que debía realizar una simulación compleja. La implementación del applet supuso un reto personal porque inicialmente poseía unos conocimientos muy básicos de programación en Java. En un principio, el desarrollo del applet de Java resultó ser una tarea ardua pero, a medida que iba incluyendo nuevas funcionalidades en la interfaz y al ver la mejora que iba adquiriendo progresivamente, aumentó mi motivación o autoestima. Finalmente, otro logro a considerar es el hecho de que la aplicación cumpla con todos los requisitos, tanto los funcionales como los no funcionales.

4.2. Aplicaciones futuras

Los applets de Java tienen la ventaja de que se pueden añadir a páginas web. Por tanto, el applet desarrollado se podría incluir en una página web de forma que la página quedase mucho más completa.

Otra posible funcionalidad que se le podría añadir al applet sería que todos las etiquetas de texto de información del applet se pudiesen cargar en varios idiomas diferentes, dotando así al applet un caracter más internacional y que fuera accesible por un número mayor aún de personas.

Por último, otra funcionalidad que se podría incorporar en un futuro sería que se pudiesen guardar en un fichero los puntos de los distintos gráficos, así como poder también guardar las gráficas.

5. Experiencia personal

La experiencia una vez implementado el proyecto final de carrera del sistema dinámico excitable de FHN ha sido muy positiva.

En lo personal, tras la finalización del proyecto de final de carrera sentí una gran sensación de satisfacción, porque pude salvar correctamente todos los problemas que se me plantearon durante el desarrollo del applet. En un principio, no sabía por dónde comenzar la interfaz y tuve que esforzarme para poder encontrar solución a todos los inconvenientes que surgían, pero con trabajo conseguí superarlos.

Por otro lado, logré profundizar en el lenguaje de programación Java, lenguaje que es muy utilizado en la actualidad y que podrá ser de gran utilidad en mi carrera profesional.

6. Bibliografía

[1] S. Zambrano, J. M. Seoane, I. P. Mariño, M. A. F. Sanjuán, S. Euzzor, R. Meucci y F. T. Arecchi; 2008 *New Journal of Physics 10*

[2] William H. Press, Saul A. Teukolsky, William T. Vettering, Brian P.Flannery, Cambridge University Press; 1992 *Numerical Recipes in C: The art of science computing*

[3] Bruce Eckel;2002 *Piensa en Java* , Cambridge Educación (2ª Edición)

[4] F. Javier Moldes, Anaya Multimedia; 2008 *Guía Práctica Java SE 6*

[5] *"Biophysics of Computation: Information Processing in Simple Neurons "*(Oxford: Oxford University Press)

[6] Heinz-otto Peitgen, Hartmut; 2004 *Chaos And Fractals: New frontiers of Science.* Juergens. Springer Verlag.

[7] José F. Vélez Serrano, Ángel Sánchez Calle, Alfredo Casado. Bernárdez, Santiago Doblas Álvarez. *Técnicas avanzadas de diseño de software: Orientación a objetos , UML, patrones de diseño y Java.* Universidad Rey Juan Carlos.

[8] Creación y control de un thread: http://sunsite.dcc.uchile.cl/java/docs/JavaTut/Cap7/creath.html

[9] Asignar código a la pulsación de botones y otros eventos en Java con Netbeans: http://losremediosinformaticos.blogspot.com/search/label/NetBeans

[10] Animación en Java: http://www.chuidiang.com/chuwiki/index.php?title=Animaci%C3%B3n_en_java

[11] Desarrollo iterativo y creciente http://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente