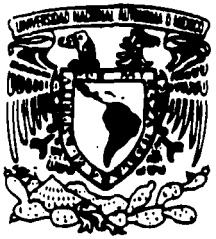
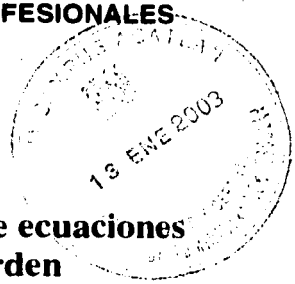


UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ACATLÁN



Sistema JSERO para la solución de ecuaciones diferenciales de segundo orden

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

L I C. E N M A T E M Á T I C A S A P L I C A D A S Y C O M P U T A C I Ó N

P R E S E N T A :

C É S A R SANDOVAL HERNÁNDEZ

ASESOR DE TESIS:
DR. SERGIO VÍCTOR CHAPA VERGARA

NAUCALPAN, EDO. DE MEX., ENERO DE 2003.

A

Agradecimientos

La tesis que actualmente presento es el fruto de un trabajo que empezó desde mi primer contacto con el sistema escolar. Viéndolo en retrospectiva, es ahí donde realmente comenzó mi carrera que ahora termina con la conclusión de esta tesis.

En primer lugar, quiero agradecer a todos los profesores que contribuyeron a mi formación. En especial quisiera agradecer al Dr. Sergio Víctor Chapa Vergara, asesor de la presente tesis por haberme guiado en su planteamiento y desarrollo.

Pero, sobre todo, deseo agradecer a la honorable Universidad Nacional Autónoma de México, institución en la que viví durante prácticamente cinco años y en la cual espero seguir participando por el resto de mi vida. En ella encontré todo lo que necesitaba para tener un desarrollo personal completo. Disfrute de las actividades culturales, recreativas y de una gran cantidad de servicios como bibliotecas, centros de idiomas, centros de cómputo, entre otros.

Además de que tuve la posibilidad de tener profesores que me transmitieron sus conocimientos y contribuyeron enormemente a mi desarrollo intelectual y aprendizaje. En la Universidad aprendí a aprender, hicieron de mi una persona con la capacidad de obtener mis propios conocimientos y aplicarlos de la forma más adecuada. Por todo esto, me siento privilegiado de entre muchos jóvenes que están en condiciones de estudiar una carrera universitaria y por muchas razones no pueden hacerlo. Privilegio que he buscado toda mi vida y ahora tengo.

Por otra parte, mi forma de agradecer a esta institución será siendo parte del desarrollo de mi país. Ser joven es símbolo de energía, pasión, sueños, ideales, dinamismo y una gran cantidad de anhelos que nos mueven. Sin embargo, el ser un joven universitario engloba esto y mucho más. Implica la gran responsabilidad de poder cambiar el rumbo de nuestro país el cual esta sufriendo grandes transformaciones.

También quiero ofrecer todo mi reconocimiento y más profundo agradecimiento a mis padres, quienes me han enseñado valores fundamentales que me han servido en mi vida. Me han enseñado a buscar siempre la felicidad, la cual no es un estado o un instante en la vida, sino una actitud ante ella. También me han enseñado a buscar el éxito sin jamás corromper el alma, es decir, sin sentimientos tales como la envidia, el egoismo, la pereza, la desconfianza y el odio ya que sólo causan problemas. Éxito que cada uno puede concebir como quiera, pero que estimo en el ser humano deberían de

ser el amor, la amistad y la integridad, la cual abarca el éxito profesional, académico, social y humano. Todo esto me ha hecho ser una mejor persona. A buscar formar una familia armoniosa y con principios y por ello contribuir a formar una mejor nación.

Por otro lado, deseo agradecer a Dios quien me ha enseñado que la vida es como cuando realizamos un viaje en tren, si éste va muy lento, aun el paisaje más hermoso terminará por aburrirnos; y si es muy rápido no tendremos la posibilidad de admirarlo; lo que me ha llevado a la conclusión de buscar un equilibrio en mi vida.

Finalmente, deseo agradecer a todas aquellas personas que de alguna manera han contribuido a que haya llegado hasta este punto tan importante de mi vida con la terminación de esta tesis. Aquellos que me apoyaron económica y moralmente. Aquellos que en muchas ocasiones se desvelaron conmigo. Aquellos que con solo una mirada o una palabra me transmitieron el orgullo tan inmenso que sentían por verme cursando una carrera universitaria. Orgullo que llegaba a lo más profundo de mi ser motivándome a seguir adelante. Aquellos que sufrieron con mis derrotas y se regocijaron con mis triunfos. Aquellos que con sus palabras fueron capaces de recordarme en momentos clave de mi vida por qué deseaba tener el título de licenciado en Matemáticas Aplicadas y Computación. A todos ...

Gracias

Dedicatoria

A mis amados padres *Angela Hernández Lindero* y *Pascual Sandoval Peralta* quienes han sido el motor de mi vida. Y de quienes me siento tremendamente orgulloso por su tenacidad ante ella.

A mis hermanos *Martha Sandoval Hernández* y *Ulises Sandoval Hernández* quienes en todo momento me apoyaron y motivaron para concluir mi tesis. Y quienes son parte de lo más valioso que la vida me ha dado.

A la honorable *Universidad Nacional Autónoma de México* que me brindó todas las posibilidades para cursar en sus extraordinarias instalaciones la carrera de Matemáticas Aplicadas y Computación.

A la *Fundación Telmex* que me apoyó enormemente no solo económicamente sino también en mi desarrollo personal. Y a todos los amigos que encontré en esta institución.

Al *Liceo Franco Mexicano* que indudablemente me dio sólidas bases para tener un alto desempeño en la universidad y para la vida.

A *aquel* ser que en los momentos diíciles y hermosos de mi vida estuvo a mi lado y a quien recordaré siempre.

A *Dios*, divino ser a quien siempre le pedí llegar hasta este momento y amorosamente me concedió ese deseo.

D

Contenido

Introducción

Las ecuaciones diferenciales de segundo orden han sido históricamente muy importantes dentro de disciplinas como la física, la electrónica o la mecánica. Esto se debe a que una gran cantidad de fenómenos físicos pueden ser representados matemáticamente por ellas. Pero existe dentro de dichas ecuaciones una que ha resultado de suma importancia e incluso fundamental para el desarrollo de la ciencia moderna. Se trata de la ecuación diferencial de Schrödinger, la cual se encuentra en la base de la teoría de la mecánica cuántica y ha brindado a los físicos, entre otras cosas, la posibilidad de estudiar la dinámica de las partículas del micromundo, tales como el fotón o el electrón.

El objetivo de la tesis es ofrecer un sistema de software que permita la visualización de las funciones solución de la ecuación diferencial de Schrödinger. Además, el sistema debe obtener no sólo valores cuantitativos sino también cualitativos de estas funciones, a través de la visualización del espacio fase, el mapa de estabilidad potencial periódico y los valores asintóticos. En el capítulo 1 se hace una descripción general de la interfaz gráfica y el modo de uso de dicho sistema.

La tesis no pretende ofrecer un sistema novedoso de solución de este tipo de ecuaciones. De hecho, existe un sistema llamado **SERO**, desarrollado por el Doctor Harold V. McIntosh, investigador de la Universidad Autónoma de Puebla, en el lenguaje *C-Objetivo* que corre en *Open Step* y *Next Step*, que realiza esto. Sin embargo, dicho sistema tiene algunos problemas que se verán más adelante y que han motivado a dar una alternativa que los corrija. Pero antes de comenzar con el desarrollo del sistema, una pregunta fundamental debe ser respondida: ¿cuáles son las herramientas computacionales que deben ser utilizadas para la elaboración de la nueva versión de **SERO**, llamada **JSERO**?

Grandes cambios se dan día a día en el mundo de la computación. Esto hace

verdaderamente difícil desarrollar sistemas que sean fáciles de mantener, no solo en el sentido de poder agregarles o quitarles funcionalidad, sino que además puedan funcionar en las cada vez más diversas plataformas. El lenguaje de programación *Java* ofrece, entre otras cosas, poder desarrollar sistemas que funcionen en una notable cantidad de plataformas. De esta manera se elimina la necesidad de hacer cambios sustanciales en el código para cada una de ellas. Existen más razones por las cuales se podría justificar el uso de *Java* para codificar a **JSERO**. Éstas se explicarán con más detalle también en el capítulo 1.

Por otra parte, los sistemas en los que se trabaja actualmente se han vuelto cada vez más complejos debido a que se requiere que realicen una mayor cantidad de actividades y con una mayor semejanza a la realidad. El programador debe, por esta razón, considerar la creación de diagramas que permitan, por ejemplo, tener una idea clara de todos los objetos que el sistema contiene, así como las relaciones que existen entre cada uno de ellos. Existen varios lenguajes de modelado orientados a objetos, a través de los cuales se puede llevar a cabo esto. El lenguaje de modelado que esta tomando mucha fuerza entre los desarrolladores de software orientado a objetos y del cual se hace uso en la presente tesis para la presentación de la estructura y comportamiento del sistema es el Lenguaje de Modelado Unificado (UML: *Unified Modeling Language*) del cual se hablará más profundamente también en el capítulo 1. Éste lenguaje de modelado es una herramienta muy útil para el programador que no pretende solamente realizar un programa a *ciegas*, sino que desea ir más lejos, buscando tener un mayor control de las características estructurales y dinámicas del sistema que desea codificar.

Aunque encontrar la solución analítica de la ecuación general de Schrödinger es muy complejo, existen casos que simplifican enormemente este problema. De hecho, en la presente tesis se trabaja con un caso particular de la ecuación que, bajo ciertas condiciones, puede ser reducida a una ecuación de la forma de Sturm Liouville. En el capítulo 2, se hace un breve análisis de esta ecuación y se obtienen propiedades importantes de sus soluciones. También se introduce el concepto de espacio fase que es muy importante para observar el comportamiento de las soluciones. Por otro lado, es importante señalar que el presente trabajo se encuentra directamente vinculado a una de las líneas de investigación que realiza el Doctor Sergio Chapa Vergara sobre ecuaciones diferenciales. Es por ello que en el capítulo 2 se hace uso de conceptos teóricos que son ampliamente tratados en la documentación que el Doctor Sergio Chapa ha realizado, siendo éstas de gran utilidad para comprender y verificar la validez de los resultados experimentales que se ven al final de la presente tesis.

Por otro lado, en los capítulos 3 y 4 se muestran las principales características

estructurales y dinámicas de **JSERO** en el UML. En el capítulo 3 se plantean las clases que implementan la integración numérica de la ecuación diferencial real homogénea. El método usado es el de Runge Kutta, el cual resulta, como se verá en el transcurso de dicho capítulo, muy fácil de codificar. Mientras que en el capítulo 4 se muestra el resto de las clases que hacen uso de las presentadas en el capítulo 3 y que además generan la interfaz gráfica de usuario, entre otras cosas. Paralelamente a los diagramas, se presenta el código en *Java*, lo que permite ejemplificar la forma tan clara de pasar del diseño de un sistema en UML a su implementación. Estos dos capítulos permiten al lector tener una idea amplia de la forma en que trabaja internamente **JSERO**, así como de su estructura general.

Finalmente, en el capítulo 5, se ofrece una serie de casos de estudio. Éste capítulo permite constatar el correcto funcionamiento de **JSERO** al comparar los resultados experimentales para ciertos potenciales con sus soluciones analíticas y los teoremas relativos a las soluciones de una ecuación de la forma de Sturm Liouville planteados en el capítulo 2.

Capítulo 1

El sistema JSERO

La elaboración de sistemas computacionales tuvo inicialmente como objetivo la solución de problemas que, sin la ayuda de la computadora, serían prácticamente imposibles de resolver. Sin embargo, hasta hace algunos años, crear un sistema de cómputo resultaba realmente difícil y engorroso, debido a que los lenguajes de programación estaban a bajo nivel; es decir, lejos del dominio del problema y cerca del dominio de la solución.

Afortunadamente para los programadores, los lenguajes de programación han evolucionado enormemente y permiten dedicar más tiempo al análisis y diseño del sistema y preocuparse menos por la forma en que se resolverá, es decir, la implementación. No obstante, el problema al que ahora se enfrenta el desarrollador de software es que cada vez existe una mayor cantidad de tecnologías y de formas de enfrentar un mismo problema. La pregunta que se desprende de lo anterior y cuya respuesta resulta en muchos casos fundamental es ¿qué lenguaje de programación permite resolver mejor este problema?

A lo largo de este capítulo se trata de responder a esta pregunta dando al lector, en forma general, los conceptos computacionales básicos que han sido utilizados para desarrollar la nueva versión de **SERO** y explicando las razones por las cuales se ha hecho uso de estas herramientas computacionales. Esta nueva versión llamada **JSERO** ha sido implementada específicamente en el lenguaje de programación orientado a objetos *Java*. Aunque gracias a que las estructuras estáticas y dinámicas del sistema se presentan en el lenguaje de modelado unificado (UML) es posible implementarlo en cualquier otro leguaje de programación orientada a objetos (POO),

como *Small Talk* o *C++*.

Asimismo, se presenta al lector las características visuales que posee el sistema (interfaz gráfica) y se da una explicación detallada de los componentes con los que cuenta **JSERO** y la forma en que puede proceder para trabajar con él.

1.1 Ambiente de la Programación Orientada a Objetos

Como se mencionó anteriormente, **JSERO** está basado en tres herramientas computacionales, que son *Java*, la *POO* y el *UML*. Estas herramientas se exponen a continuación.

1.1.1 La Programación Orientada a Objetos

Los lenguajes de programación estructurados han sido muy usados desde su aparición. Han tenido una amplia penetración en diversos sectores del mercado, pero los resultados no siempre han sido muy buenos. Por lo que algunas compañías se dieron a la tarea de encontrar otras alternativas. Una de estas alternativas es la Programación Orientada a Objetos (POO).

Aun cuando el nacimiento de la POO se remonta al año de 1967, cuando fue desarrollado *Simula-67*, no es hasta la década de los 80 cuando tiene su auge. Es en esta época cuando sistemas como *Small-Talk* y más tarde *C-Objetivo*, *C++*, *Eiffel* y *CLOS* hacen su aparición y permiten el crecimiento de la POO.

La POO consiste en ubicar el dominio del problema y su solución dentro del punto de vista de los objetos. Estos objetos poseen métodos y atributos que los caracterizan. Los atributos marcan el estado en el que se encuentra el objeto. Mientras que los métodos sirven esencialmente para modificarlos, obtenerlos o para hacer uso de ellos. Esto da al programador ciertas ventajas que no ofrece la programación estructurada y algunas de las cuales se presentan a continuación.

1.1.1.1 Menor complejidad

Un problema al que se enfrentan los programadores, analistas, diseñadores de software y gente dedicada a esta área del conocimiento, es que los sistemas de cómputo son cada vez más grandes y complejos. Esto se debe en parte a que esperamos que los sistemas hagan cada vez más cosas de manera más eficiente y con una mayor semejanza con la realidad. Esto, aunado al hecho de que cada vez contamos con menos tiempo para encontrar una solución a nuestro problema y debemos de explotar más y mejor los recursos de la computadora. Así, se deben buscar alternativas que permitan eliminar en lo posible estos problemas, y una de ellas es la POO. Evidentemente no existe una solución que haga mágicamente que el desarrollo de software sea algo extremadamente sencillo. Pero ataca muchos de los problemas antes mencionados.

La visualización de un problema a través de la POO tiene muchas ventajas. Una de ellas es que la complejidad de los sistemas no aumenta en la misma proporción que el crecimiento de los sistemas. De hecho, la generación de aplicaciones implica muchas veces muy pocas líneas de código el cual a su vez es extremadamente comprensible e intuitivo.

En los sistemas basados en la POO, los objetos y las clases que se han identificado en el análisis del modelo tienen representaciones directas en el código, lo que hace muy sencilla la identificación de las relaciones entre la definición del problema (análisis) y su solución (código). Éste es un punto ventajoso sobre la programación estructurada, ya que a diferencia de ésta resulta sumamente sencillo llevar la parte del análisis del sistema al diseño y viceversa.

Por otra parte, la POO mantiene una fuerte cohesión entre los datos y las operaciones que los manipulan, lo que permite que, tal como en la realidad, sea posible manipular las características de los objetos. Si volteamos por un segundo a nuestro alrededor, nos daremos cuenta de que todo lo que podemos ver son objetos. Estos poseen atributos y funciones que realizan cada uno de manera distinta o semejante a otros. Es precisamente en esto en lo que se basa la POO. En visualizar cada uno de los componentes que deseamos utilizar, para solucionar nuestro problema como objetos que a su vez pueden estar compuestos de otros objetos. Esta semejanza con la realidad hace que las abstracciones de nuestro problema sean más intuitivas y poderosas. Aún más, los usuarios y compradores del software pueden comprender las implicaciones de sus requerimientos más claramente, porque el sistema es construido usando sus propios conceptos. La perspectiva del objeto está a alto nivel, cerca del dominio del problema y lejos del dominio de la implementación.

1.1.1.2 Mejor estabilidad en la presencia de cambios

Debido a que las abstracciones de la POO están basadas en el mundo real, tienden a ser más estables. Construir objetos con responsabilidades adecuadamente delegadas y una comunicación correcta entre ellos hace que el sistema cumpla correctamente con todas sus tareas asignadas. Lo que hace a su vez que cuando se necesite agregar o quitar funcionalidad al sistema, o modificar su comportamiento, sólo sea necesario hacer uso de conceptos como polimorfismo, herencia, etc. Esto se traduce en un fácil mantenimiento de los sistemas de cómputo, por parte no sólo de los desarrolladores del software sino también de otros desarrolladores que hacen uso de él dentro de sus aplicaciones.

Esta es una de las características más valiosas de la POO, ya que mantener un sistema estructurado implica normalmente un consumo excesivo de tiempo y de cambios sustanciales en su estructura, lo que no sucede con los sistemas orientados a objetos que han sido bien diseñados.

1.1.1.3 Reuso de código

Esta es otra parte importante de la POO. El reuso de código en los sistemas estructurados es generalmente un problema de modificación del código fuente para adaptarlo a nuestros nuevos requerimientos. Pero en los sistemas de POO este problema es muy fácilmente resuelto. Existen dos formas de hacer reuso de código: a través de la composición y de la herencia. La composición permite el uso completo del código tal y como ha sido creado. Mientras que la herencia permite extender las características del código, agregándole o modificándole funcionalidad.

Por todas estas razones, se puede decir que la POO es una vía excelente para modelar de manera eficiente el sistema que se está planteando en este trabajo, debido a que, entre otras cosas, requiere que en el futuro se le pueda agregar funcionalidad.

1.1.2 El Lenguaje de Modelado Unificado (UML)

Para crear un sistema computacional orientado a objetos no basta con conocer y manejar correctamente un lenguaje orientado a objetos, como *Java* o *C++*. Esto sólo es un primer paso para desarrollar un buen sistema orientado a objetos, ya que

no garantiza que se elabore un sistema de calidad. Es fundamental que se analice y diseñe dicho sistema desde el punto de vista de objetos, a través de alguna metodología orientada a objetos como los patrones GRASP [1]. Sin embargo, el presente trabajo no aplica ninguna metodología, dado que ya se tiene elaborado un sistema y lo que se busca es replantearlo en un lenguaje y plataforma diferentes.

Dicho esto, el primer paso es llevar a un metamodelo la parte estructural y dinámica del sistema. Una vez teniendo estas características visibles en diagramas, se procede a su implementación en un lenguaje de POO, que en el caso de la presente tesis es *Java*.

El Lenguaje de Modelado Unificado (Unified Modeling Language), es un lenguaje de modelado visual de propósito general, que es usado para especificar, visualizar, construir y documentar los componentes de un sistema computacional orientado a objetos. Lo que permite comprender, diseñar, configurar, mantener y controlar información acerca de estos sistemas [2].

UML no promueve un método estándar de elaboración de sistemas; sin embargo permite, gracias al metamodelo y notación que posee, el uso de métodos que lleven a la creación de un sistema robusto y de fácil mantenimiento.

El propio nombre de UML muestra la razón por la que fue creado. Se trata de un lenguaje de modelación que unifica la experiencia pasada acerca de las técnicas de modelación, e incorpora al software actual las mejores prácticas en una propuesta estándar.

Antes de la aparición de UML, muchos libros de la metodología orientada a objetos promovían sus propios conceptos, definiciones, notación y terminología; aunque la esencia de todos estos conceptos no variaba mucho.

Entonces, los metodólogos Rumbaugh, Booch y Jacobson formaron la Corporación de Software Rational [3], la cual tenía como objetivo obtener un lenguaje de modelización que combinara los conceptos comúnmente aceptados de muchos métodos orientados a objetos, seleccionando claras definiciones para cada concepto, así como una notación y terminología. Su trabajo conjunto resultó en lo que se conoce actualmente como Lenguaje de Modelado Unificado (UML).

En 1996, la OMG (Object Management Group), compuesta por grandes empresas de software, buscó propuestas para la estandarización del modelado orientado a objetos. Entonces, los autores de UML (Jacobson, Booch y Rumbaugh) comien-

zan a trabajar con metodólogos y desarrolladores de otras compañías, para dar una propuesta atractiva a los miembros de la OMG que además fuera aceptada por los creadores de herramientas, metodólogos y desarrolladores, quienes serían los usuarios. Finalmente, en 1997, UML fue aceptado por la OMG.

La aceptación de UML por la OMG ha propiciado que su popularidad crezca entre los vendedores de herramientas CASE, y los metodólogos, quienes anunciaron que usarían su notación para trabajos posteriores. Por lo que, aunque existen varios métodos de análisis y diseño orientados a objetos, como Coad/Yourdan, Shlaer y Mellor, Booch, OMT, Fusion y Jacobson; el que indiscutiblemente ha tomado más fuerza es UML.

UML se caracteriza por ofrecer información acerca de la estructura estática y el comportamiento dinámico de un sistema. La estructura estática define los tipos de objetos importantes para un sistema y para su implementación, así como las relaciones entre los objetos. El comportamiento dinámico define la historia de los objetos en el tiempo y las comunicaciones que tienen los objetos para alcanzar los objetivos.

El UML contiene dos partes fundamentales: la notación y el metamodelo. La notación es la sintaxis del lenguaje de modelado. Esta define, entre otras cosas, cómo se representan elementos y conceptos, tales como clases, asociaciones y multiplicidad. Mientras que un metamodelo es un diagrama que representa los artefactos contenidos dentro del sistema.

Una de las ventajas de realizar un modelado orientado a objetos es poder construir un modelo visual que permita la fácil comprensión de sistemas complejos, aun para personas que no tengan conocimientos computacionales. Y una vez que se haya hecho un correcto modelado, la implementación se pueda llevar a cabo en cualquier lenguaje de POO (*C++*, *Java*, *C - Objetivo*, etc.), de una manera casi natural.

UML permite entonces que el programador se enfoque esencialmente al modelado del sistema en lugar de ir directamente a la programación. De hecho, esta última se realiza con más facilidad a partir del diagrama. Nos permite también explorar varias arquitecturas y diseños de soluciones fácilmente, antes de escribir una sola línea de código.

De hecho, se puede usar en sistemas implementados en varios lenguajes de programación y plataformas. El *Front-end* sería idéntico o similar en todas las clases, mientras que el *back-end* diferiría un poco para cada plataforma.

Uno de los objetivos de UML es hacerlo tan simple como sea posible y que sea capaz de modelar los sistemas prácticos que se necesita construir. Se pretende también que sea un lenguaje universal, como cualquier lenguaje de propósito general.

En el caso concreto de nuestro sistema **JSERO**, realizar un modelado en UML permitiría tener una mejor comprensión de todas las clases que usa y la interacción que existe entre cada una de ellas; es decir, la comunicación que hay entre los objetos. Además de que representaría la base para poder agregarle nuevas funcionalidades de una manera más sencilla, al concebir las modificaciones en los diagramas y luego plasmarlos en el código, o incluso poder migrarlo a cualquier otro lenguaje de POO. Si el lector está interesado en obtener mayor información acerca de UML, se recomienda consultar [3] donde puede encontrar la documentación completa de este lenguaje de modelización.

1.1.3 El lenguaje Java

Java es un lenguaje completamente orientado a objetos. Esto quiere decir que todo es visto como un objeto en *Java*. Desde la creación de un tipo de datos, hasta los componentes que integran la interfaz gráfica; por lo que cuenta con todas las cualidades antes mencionadas de la POO.

Pero ¿qué más podemos decir acerca de *Java*? *Java* ha sido pensado para hacer aplicaciones que se ejecuten en la internet (applets). Aunque también es posible ejecutarlas como aplicaciones en las computadoras (standalone). Esta caracterísica de *Java* brinda muchas ventajas sobre otros lenguajes de programación. Una de ellas es que el problema que históricamente había sido el realizar una aplicación en internet (programación distribuida, bases de datos, etc.), se ha vuelto un asunto mucho más sencillo.

Por otro lado, *Java* trabaja en *cross platform*; lo que significa que una aplicación hecha en este lenguaje de programación corre eficientemente en cualquier plataforma, ya sea Windows, Linux, Solaris o Mac OS. Esta característica de *Java* elimina la necesidad de realizar cambios en el código para ejecutar el programa en una plataforma distinta a la que originalmente se utilizó. Una mayor cantidad de usuarios, de una gran variedad de plataformas, puede ser alcanzado si se hace uso de este lenguaje de programación.

Sin embargo, existen desventajas en este último punto. Una de las más im-

portantes es que la aplicación es lenta. *Java* contiene un interprete llamado Virtual Machine (VM) que permite el *cross platform*. La lentitud radica en que la VM tiene que procesar los datos en función de los recursos del sistema que se están usando.

Pero esta lentitud se aprecia más cuando los sistemas se vuelven muy grandes. En el caso de nuestro sistema, éste no sería un problema, ya que es relativamente pequeño.

Por otro lado, *Java* posee una sintaxis muy similar a la de *C/C++*. Considerando que este último es un lenguaje de programación muy popular entre los programadores, esto se vuelve una ventaja enorme, ya que no se tiene el problema de aprender nuevas sintaxis que harían más lento el proceso de traslado del lenguaje *C/C++* a *Java*.

Otro punto que es importante señalar es que *Java* no maneja apuntadores, como en el caso de *C/C++*; sin embargo, usa eficientemente las referencias a objetos de manera casi natural. Por esta razón, la programación en *Java* es mucho más fácil que en otros lenguajes que hacen uso de los apuntadores. Además de que el hecho de que no se usen los apuntadores también impide que se haga un mal uso de los recursos de la memoria de las computadoras, donde se ejecutan la aplicaciones. Sin embargo, hay que decir que esto limita en gran medida la libertad de los programadores para explotar dichos recursos.

Java cuenta además con una librería muy poderosa llamada *Swing* que contiene una gran cantidad de componentes que permiten generar una interfaz de usuario muy amigable en muy pocas líneas de código. De hecho, el sistema trabaja en su mayoría con esta librería.

El sistema operativo *MAC OS X* fue usado para realizar las pruebas de la nueva versión en *Java* del sistema **SERO**, aunque no se usan sus recursos gráficos. Esto es, se hace uso exclusivamente de la parte gráfica estándar de *Java*. Este sistema operativo cuenta con la versión 1.3.0 del compilador de *Java*. Aunque el sistema puede ser usado sin problemas con versiones más recientes.

Las características antes mencionadas de *Java* son entonces ideales para codificar en este lenguaje el sistema **JSERO**. Para mayor información acerca de *Java* se recomienda ampliamente revisar el libro de Bruce Eckel [10] y visitar la página oficial de *Java* [11].

1.2 Interfaz de usuario del sistema

En esta sección se hace la presentación de la interfaz de usuario del sistema. Es importante señalar que, para llevar a cabo la construcción de dicha interfaz, se ha hecho uso de la potente librería de *Java*, llamada *Swing*, que contiene una serie de objetos, tales como botones, páneles, menús, etc., que permiten al desarrollador la creación de una interfaz visual amigable para el usuario.

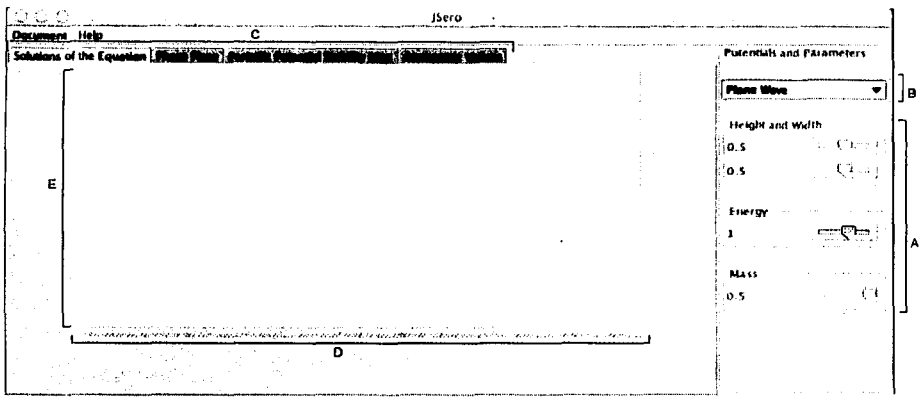


Figura 1.1: Interfaz inicial del sistema **JSERO**.

En la figura 1.1 se presenta el sistema, tal y como aparece cuando inicia. Los objetos de **A** son esencialmente instancias de **JSliders** que permiten la manipulación de los parámetros de la ecuación de Schrödinger que se trata en capítulos posteriores. Estos parámetros son: el alto y ancho del potencial (no se aplica a todos los potenciales), la energía total de la partícula y su masa. Mientras que el objeto **B** es una instancia de la clase **JComboBox** de *Java*, que contiene todos los potenciales de los que se puede servir el usuario.

Por otro lado, el objeto de **C** es una instancia del **JTabbedPane** a través del cual el usuario puede pasar de un espacio a otro (espacio solución, espacio fase, mapa de estabilidad periódica potencial y los valores asintóticos).

Finalmente, **D** es la barra de herramientas que es usada esencialmente para manipular las gráficas que se visualizan en **E**. Todos los espacios de solución poseen

su propia barra de herramientas por lo que **D** varía de un espacio a otro.

1.2.1 Espacio solución

Al seleccionar la barra de herramientas **D** cuando se está en el espacio de soluciones, la interfaz de usuario tiene la forma de la figura 1.2.

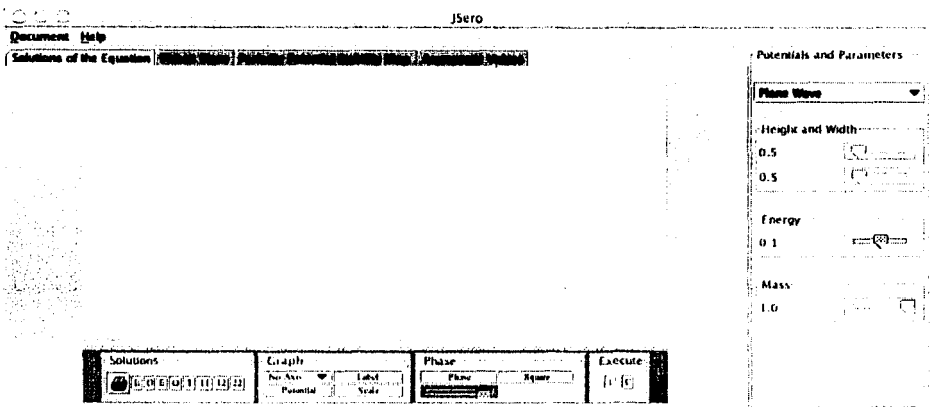


Figura 1.2: Interfaz de usuario de **JSERO** cuando se ha seleccionado la pestaña *Solutions of the equation* y se ha expandido su barra de herramientas

Se puede observar que la barra de herramientas contiene una serie de *sliders*, páneles, botones y combo box que sirven para modificar el contenido del visualizador de las gráficas **E**. Pero veamos más detalladamente el funcionamiento de cada uno de los páneles que se encuentran dentro de **D**.

Primeramente, el panel llamado *Solutions* (Soluciones), contiene nueve botones. El primero de ellos sirve para imprimir el contenido de **E**. Mientras que los otros ocho botones permiten la graficación de:

- Solución par
- Solución impar

- Derivada de la solución par
- Derivada de la solución impar
- La traza
- Cuadrado de la función de onda correspondiente a la solución par (probabilidad normalizada)
- Producto de las dos soluciones
- Cuadrado de la función de onda correspondiente a la solución impar (probabilidad normalizada)

Por otro lado, el panel *Graph* (Gráfica) contiene un *combo box* y tres botones. Este panel sirve para decorar las gráficas que estan siendo visualizadas. El *combo box* permite al usuario poner o quitar los ejes coordenados de la gráfica. Mientras que el botón *Label* decora el contenedor de las gráficas **E** poniéndole o quitándole en la parte inferior izquierda el potencial que está siendo graficado. Por último, el botón *Potential* grafica el potencial correspondiente que ha sido seleccionado en **B**.

Ahora bien, el panel *Phase* (Fase) posee dos botones y un *slider*. El primer botón grafica, como su nombre lo indica, la fase; y el segundo su cuadrado. El *slider* que se encuentra abajo de estos dos botones permite hacer variar la fase para distintos valores.

En cuanto al útimo panel, sólo se encuentran en él tres botones. El primero de ellos sirve para actualizar los valores no normalizados de las soluciones. Mientras que el segundo botón normaliza dichos valores. El último, sólo sirve para hacer un limpiado de **E** y **D**.

La última característica que posee esta barra de herramientas es que al oprimir sobre la parte gris obscura del panel, éste desaparece, y en su lugar aparece otro panel que contiene solamente dos *sliders*. El primero de ellos cambia la escala de las gráficas, mientras que el segundo hace variar los valores del potencial. En la figura 1.3 se aprecia este segundo panel.

1.2.2 Espacio fase

En la figura 1.4 se puede observar que el espacio fase es similar al espacio de soluciones; lo único que cambia es la barra de herramientas. En ella solo existe un panel

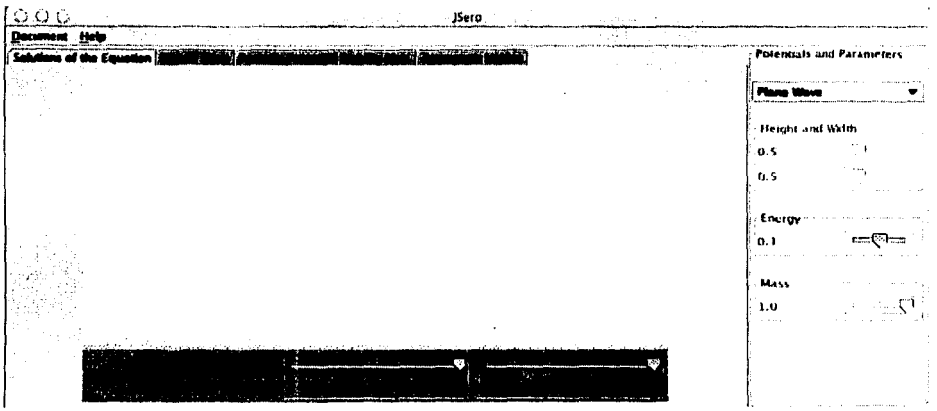


Figura 1.3: Panel utilizado para modificar la escala y el potencial

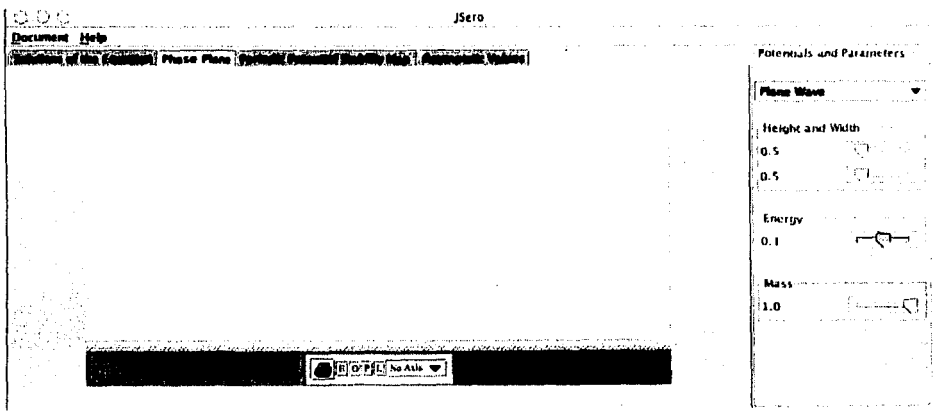


Figura 1.4: Barra de herramientas del espacio fase

que contiene cinco botones y un *combo box*. El primero de ellos, como en el caso anterior, sirve para imprimir el contenido del visualizador **E**. Los tres siguientes botones grafican la solución par, impar y la fase, respectivamente, en este espacio. Mientras que el botón *Label* y el *combo box* sirven, como en el caso del espacio solución, para

etiquetar el visualizador **E** con el potencial con el cual se está actualmente trabajando y poner los ejes coordenados respectivamente.

También, como en el caso de la barra de herramientas del espacio solución, es posible cambiar la escala de las gráficas y los valores del potencial al oprimir sobre la parte gris obscura del panel y hacer variar los *sliders* que aparecen en el nuevo panel.

1.2.3 Mapa de estabilidad potencial periódico

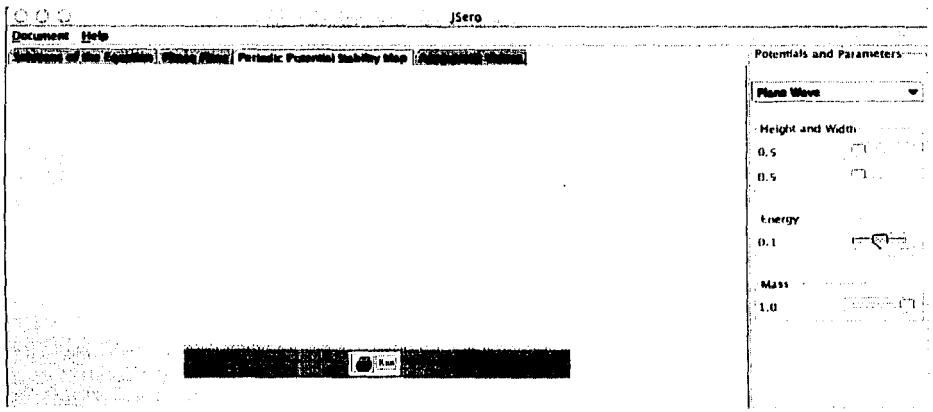


Figura 1.5: Barra de herramientas del mapa de estabilidad potencial periódico

En la barra de herramientas de este espacio sólo existe la posibilidad de graficar los valores del mapa de estabilidad potencial periódica para el potencial sinusoidal e imprimirlo. La barra de herramientas correspondiente a este potencial se muestra en la figura 1.5.

1.2.4 Valores asintóticos

Finalmente, en este último espacio (figura 1.6), se pueden visulizar los valores asintóticos, únicamente para los potenciales: pozo cuadrado, oscilador armónico de Dirac y

Coulomb. También tiene la habilidad de imprimir la gráfica de los valores asintóticos para este potencial, al presionar sobre el botón de impresión. La barra de herramientas tiene la misma forma que para el caso del mapa de estabilidad potencial periódico.

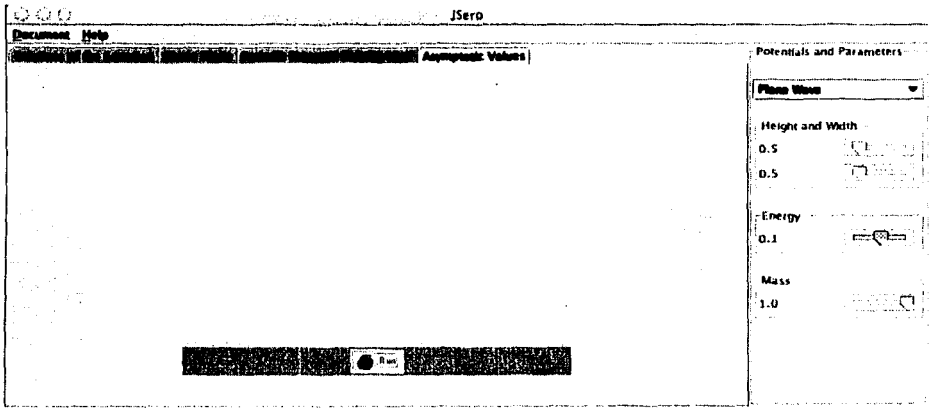


Figura 1.6: Barra de herramientas de los valores asintóticos

1.3 Casos reales de uso

Una manera excelente que tiene UML para representar, de forma escrita o gráfica, la secuencia de eventos generados por los actores externos al sistema, que tienen como fin realizar un proceso, son los llamados *casos de uso*. El caso de uso general de la *selección de una gráfica* se muestra a continuación.

Acción del actor	Respuesta del sistema
1 Este caso de uso comienza cuando un estudiante empieza a utilizar el sistema	
2 Introduce los valores de los parámetros de la ecuación, manipulando los *sliders* **A**	
3 Escoge el potencial en **B**	
4 Escoge el espacio de soluciones que desea ver en **C**	**5** Muestra el espacio de solución
6 Escoge la gráfica seleccionando el botón deseado en **D**	**7** Muestra las gráficas correspondientes en **E**
8 Opcionalmente pide la impresión de las gráficas seleccionando el botón de impresión en **D**	**9** Imprime las gráficas que se visualizan en **E** en la impresora configurada
10 El estudiante recibe la información que necesita y termina	

La secuencia de pasos que sigue una persona (en este caso un estudiante) que desea hacer uso del sistema para graficar las soluciones de la ecuación de Schrödinger y opcionalmente imprimirlas, es en realidad muy sencilla. Simplemente tiene que seguir los pasos que se muestran en *selección de una gráfica*

En la figura 1.7 se muestra un escenario del caso de uso *selección de una gráfica*. En ella se puede apreciar, de la misma forma que en el caso de la tabla, pero esta vez de una manera gráfica, los pasos que tiene que seguir el usuario externo para lograr visualizar una solución de la ecuación diferencial y opcionalmente imprimirla.

En realidad resulta sumamente sencillo deducir, a partir de los casos de uso, la forma en que se puede llegar a graficar una solución en **JSERO**. Se puede ver claramente que solamente son necesarios cuatro pasos para obtener la información que se necesita. Sin embargo, se debe mencionar que esto resulta así, en parte, gracias a que los diagramas en UML son muy explícitos.

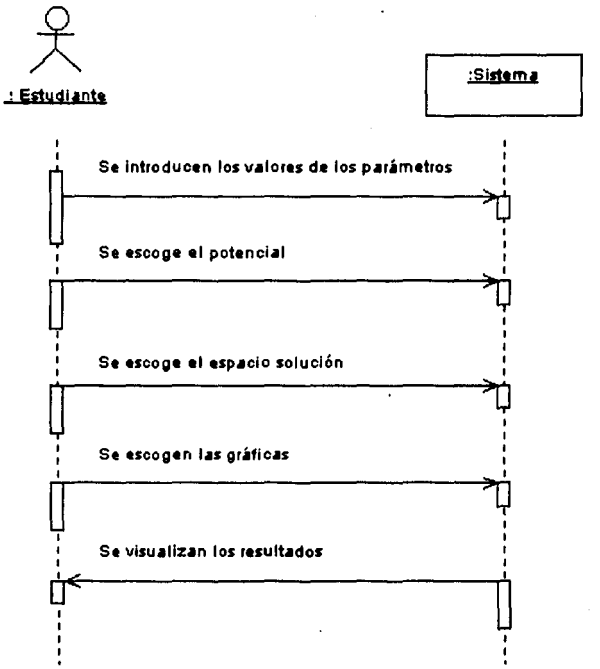


Figura 1.7: Caso real de uso de la *selección de una gráfica*

Capítulo 2

Ecuaciones diferenciales de segundo orden

Históricamente, las ecuaciones diferenciales de segundo orden (EDSO) han estado sujetas a muchos estudios; esto se debe a que la dinámica de una gran cantidad de fenómenos físicos puede ser descrita con la ayuda de este tipo de ecuaciones. Su uso se da muy frecuentemente en áreas como la física, la elecrónica y la mecánica.

El interés de esta tesis por las EDSO se debe a que la ecuación que se desea tratar (la ecuación diferencial de Schrödinger en una sola dimensión e independiente del tiempo), es una EDSO de la forma de Sturm Liouville.

El objetivo de este capítulo es abordar algunas de las propiedades que poseen las EDSO de la forma de Sturm Liouville. De estas propiedades se deducen algunos aspectos importantes de las gráficas de las soluciones de la ecuación de Schrödinger que se plantean en el último capítulo. Como ya se dijo anteriormente, no se pretende hacer un análisis riguroso de estas ecuaciones. En realidad sólo se dan algunos teoremas que nos permiten obtener información sobre el comportamiento de las soluciones sin resolver propiamente la ecuación.

Además, cabe señalar que existe una amplia literatura en torno al tratamiento de las EDSO y específicamente al problema de Sturm Liouville. De hecho la presente tesis se encuentra sumergida dentro de la línea de investigación de Ecuaciones Diferenciales que realiza el doctor Sergio V. Chapa Vergara; motivo por el cual el presente capítulo hace amplio uso de sus documentos [8] y [9]. De estas últimas dos referencias

se toman muchos conceptos que se utilizan a lo largo de este capítulo y que sirven para explicar los resultados experimentales que ofrece **JSERO**.

Por otro lado, la información que se puede obtener acerca de las soluciones de una ecuación diferencial no solo es cuantitativa sino también cualitativa. Este tipo de análisis se puede llevar a cabo en el llamado espacio fase que se verá hacia el final de este capítulo.

2.1 La ecuación diferencial real homogénea de segundo orden

Primeramente, tratemos la ecuación diferencial de segundo orden real homogénea, que en su forma general es la siguiente:

$$p_0(x)\frac{d^2y}{dx^2} + p_1(x)\frac{dy}{dx} + p_2(x)y = 0 \tag{2.1}$$

Esta ecuación contiene las funciones $p_0(x)$, $p_1(x)$ y $p_2(x)$, las cuales deben ser continuas en el intervalo $a \leq x \leq b$. Además, $p_0(x)$ no puede ser cero en ningún punto de este intervalo.

2.1.1 El Wronskiano

Es importante en este punto definir al wronskiano de la ecuación general de segundo orden. Éste nos será de utilidad para presentar la solución general de la ecuación diferencial (2.1), que veremos más adelante.

Si $f(x)$ y $g(x)$ son dos soluciones de (2.1), podemos definir entonces el wronskiano como sigue:

$$W(f,g;x) = \begin{vmatrix} f(x) & f'(x) \\ g(x) & g'(x) \end{vmatrix} \tag{2.2}$$

de donde se deduce la siguiente expresión

$$W(f,g;x) = f(x)g'(x) - g(x)f'(x). \tag{2.3}$$

Si $f(x)$ y $g(x)$ son linealmente independientes entonces se tiene:

$$W(f,g;x) \neq 0; \qquad a \leq x \leq b$$

La identidad de Abel se puede expresar como:

$$W(f,g;x) = W(f,g;x_0)e^{-\int_{x_0}^{x} \frac{p_1(\sigma)}{p_0(\sigma)}d\sigma} \tag{2.4}$$

donde x_0 es un punto dado, dentro del intervalo (a,b).

De esta última igualdad se puede apreciar que el wronskiano asociado a (2.1) conserva su signo para todo valor x en (a,b). Esta propiedad es de gran utilidad para la demostración de los teoremas de separación y comparación.

2.1.2 Principio de superposición

Propongamos ahora una forma de escribir la solución general a la ecuación (2.1), en términos del wronskiano.

Si $f(x)$ y $g(x)$ son soluciones linealmente independientes de la ecuación (2.1), entonces la solución más general tiene la forma

$$y(x) = c_1 f(x) + c_2 g(x) \tag{2.5}$$

donde c_1 y c_2 son constantes arbitrarias.

Poniendo las siguientes condiciones iniciales:

$$\begin{aligned} y(x_0) &= y_0 \\ y'(x_0) &= y'_0 \end{aligned} \tag{2.6}$$

se obtiene, con base en la solución general (2.5), el siguiente par de ecuaciones con incógnitas en c_1 y c_2

$$\begin{aligned} c_1 f(x_0) &+ c_2 g(x_0) &= y_0 \\ c_1 f'(x_0) &+ c_2 g'(x_0) &= y'_0 \end{aligned} \tag{2.7}$$

las cuales pueden ser resueltas en términos de los wronskianos de la siguiente manera:

$$\begin{aligned} c_1 &= \frac{W_1(y,g;x_0)}{W(f,g;x_0)} \\ c_2 &= \frac{W_2(f,y;x_0)}{W(f,g;x_0)} \end{aligned} \tag{2.8}$$

El wronskiano $W(f,g;x_0)$ se define en el punto x_0 y los wronskianos W_1 y W_2 son expresados como:

$$W_1(y,g;x_0) = \begin{vmatrix} y(x_0) & g(x_0) \\ y'(x_0) & g'(x_0) \end{vmatrix} \tag{2.9}$$

$$W_2(f,y;x_0) = \begin{vmatrix} f(x_0) & y(x_0) \\ f'(x_0) & y'(x_0) \end{vmatrix} \tag{2.10}$$

Finalmente, tenemos que a partir del principio de superposición, de las condiciones iniciales y del Wronskiano, la solución a la ecuación diferencial homogénea de segundo orden queda expresada como sigue:

$$y(x) = \frac{W_1(y,g;x_0)}{W(f,g;x_0)} f(x) + \frac{W_2(f,y;x_0)}{W(f,g;x_0)} g(x) \tag{2.11}$$

2.2 Ecuación diferencial autoadjunta

Es importante reiterar que la ecuación diferencial de Schrödinger unidimensional e independiente del tiempo, tiene la forma normal de Sturm Liouville. Motivo por el cual esta sección se encarga de mostrar en qué consiste esta particular forma deEDSO. Además, se ofrece una serie de teoremas a partir de los cuales se desprende información muy relevante acerca del comportamiento de las soluciones.

2.2.1 Forma normal de Liouville

La ecuación (2.1) puede escribirse en una forma de Liouville bajo ciertas condiciones. Esto resulta muy útil si se quiere estudiar las principales características de sus funciones solución.

Primeramente, dividamos nuestra ecuación por $p_0(x)$, con lo que obtenemos:

$$y''(x) + \rho_1(x)y'(x) + \rho_2(x)y(x) = 0 \tag{2.12}$$

donde $\rho_1(x) = \frac{p_1(x)}{p_0(x)}$ y $\rho_2(x) = \frac{p_2(x)}{p_0(x)}$

Hagámos la siguiente igualdad $y(x) = u(x)v(x)$. Derivando dos veces la función y obtenemos:

$$y' = u'v + uv' \tag{2.13}$$

$$y'' = uv'' + 2u'v' + u''v \tag{2.14}$$

Sustituyendo (2.13) y (2.14) en (2.12) obtenemos una ecuación para v

$$uv'' + (2u' + \rho_1 u)v' + (u'' + \rho_1 u' + \rho_2 u)v = 0 \tag{2.15}$$

Si no se conoce una solución particular u de (2.13), se puede escoger u, tal que cumpla con: $2u' + \rho_1 u = 0$. Entonces, la solución a esta última ecuación sería $u = exp(-1/2 \int \rho_1(x)dx)$.

De donde podemos concluir que

$$u' = -\frac{1}{2}\rho_1 u \tag{2.16}$$

y

$$u'' = -\frac{1}{2}\rho_1 u + \frac{1}{4}\rho_1^2 u \tag{2.17}$$

De donde se obtiene finalmente que la ecuación (2.14) se puede reducir a:

$$y''(x) + g(x)y(x) = 0 \tag{2.18}$$

donde

$$g(x) = \rho_2(x) - \frac{1}{4}\rho_1^2(x) - \frac{1}{2}\rho_1'(x) \tag{2.19}$$

La ecuación (2.18), está en la *Forma Normal de Liouville*, la cual nos permitirá deducir propiedades importantes sobre las soluciones de la ecuación diferencial de Schrödinger para la onda plana que, como veremos más adelante, tiene la misma forma.

2.2.2 Teoremas de oscilación y comparación

En esta sección se describen los teoremas de oscilación y comparación, los cuales son de gran ayuda para realizar un análisis de las gráficas de las soluciones, aun sin resolverlas.

El teorema de separación de Sturm, que se enuncia a continuación, nos ofrece información acerca de las posiciones de las soluciones, una con respecto a la otra.

Teorema 2.1 *Si f y g son soluciones linealmente independientes de (2.1), entonces f debe anularse entre dos ceros sucesivos de g y recíprocamente. Es decir, los ceros de f y g alternan.*

Según el teorema anterior, las soluciones de la ecuación diferencial se comportan como las funciones mostradas en la figura 2.1.

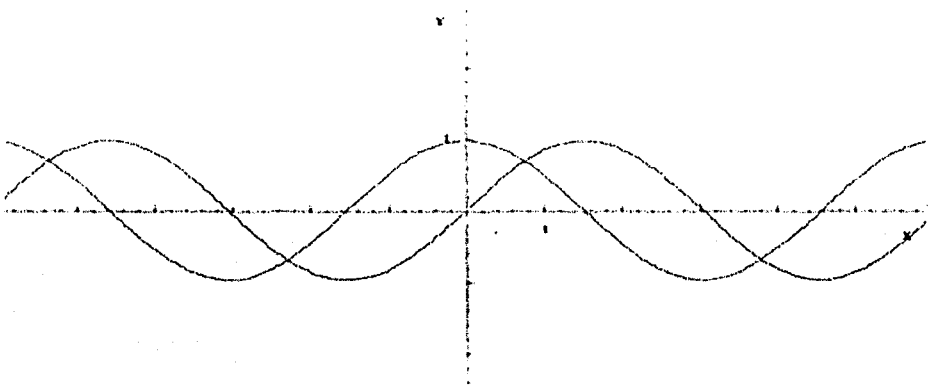


Figura 2.1: Comportamiento de las soluciones descritas en el teorema 2.1

Los dos teoremas que a continuación se enuncian dan información significativa acerca de la concavidad de las soluciones.

Teorema 2.2 *Sea la ecuación diferencial $y''(x) + g(x)y(x) = 0$, si $g(x) < 0$ en el intervalo (a, b), entonces cualquier solución $u(x)$ (no idénticamente cero) de la ecuación diferencial no tiene más de un cero en (a, b).*

Tomemos ahora dos EDSO que se encuentran en su forma normal de Liouville.

$$y''(x) + g(x)y(x) = 0 \tag{2.20}$$

$$y''(x) + g(x)y(x) = 0 \tag{2.20}$$

$$z''(x) + h(x)z(x) = 0 \tag{2.21}$$

El siguiente teorema nos ofrece información relativa a las soluciones de las dos ecuaciones diferenciales.

Teorema 2.3 (teorema de comparación) *Sea $g(x) < h(x)$ para $x \geq x_0$. Si $y(x)$ es la solución de (2.20) con las condiciones iniciales $y(x_0) = y_0$, $y'(x_0) = y'_0$, tales que $y(x) > 0$, para algún intervalo a la derecha de x_0. Y si $z(x)$ es la solución de (2.21), satisfaciendo las condiciones $z(x_0) = y_0$, $z'(x_0) = y'_0$. Entonces $y(x) > z(x)$ para $x > x_0$, mientras se tenga $y(x) > 0$.*

Teorema 2.4 *Si $0 < m < g(x) < M$, para $a \leq x \leq b$, y si x_0 y x_1 son ceros consecutivos (entre a y b) de una solución de (2.20), entonces:*

$$\frac{\pi}{\sqrt{M}} < x_1 - x_0 < \frac{\pi}{\sqrt{m}} \tag{2.22}$$

De los teoremas anteriores hemos aprendido mucho acerca del comportamiento de las soluciones de la EDSO que, bajo ciertas cir-cunstancias, se puede reducir a la forma de Liouville. También pudimos comprobar que, aunque encontrar una solución explícita a una EDSO no siempre es posible, existen teoremas que nos ofrecen información muy importante acerca de ellas, sin tener que resolverlas.

2.2.3 Forma autoadjunta

Propongámos a **L** como el operador diferencial de segundo orden. Entonces

$$\mathbf{L} \equiv p_0(x)\frac{d^2}{dx^2} + p_1(x)\frac{d}{dx} + p_2(x) \tag{2.23}$$

El operador **L** es un operador lineal.

Para obtener el operador formalmente adjunto $\mathbf{L}^*$ asociado a $\mathbf{L}$, se forma el producto $v\mathbf{L}[u]$ y se integra sobre el intervalo (a, b). Integrando repetidamente por partes, el resultado puede expresarse en la forma:

$$\int_a^b vL[u]dx = |\ldots|_a^b + \int_a^b uL^*[v]dx \tag{2.24}$$

En este momento sería conveniente realizar algunas observaciones sobre las funciones u y v. Estas son completamente arbitrarias, excepto por ser suficientemente diferenciables para que existan $\mathbf{L}[u]$ y $\mathbf{L}^*[v]$; pero no son necesariamente soluciones de ninguna ecuación diferencial particular. Integrémos por partes el producto $\mathbf{L}^*[v]$:

$$\int_a^b vL[u]dx = \int_a^b (vp_0u'' + vp_1u' + vp_2u)dx \tag{2.25}$$

Analicemos el primer término de la integral. Esta se puede escribir de la siguiente forma al integrar por partes:

$$\int_a^b vp_0u''dx = vp_0u'|_a^b - \int_a^b u'd(vp_0) \tag{2.26}$$

Y entonces se puede escribir:

$$\int_a^b vp_0u''dx = vp_0u'|_a^b - \int_a^b (u'vp_0' + u'v'p_0)dx \tag{2.27}$$

Por otro lado, el segundo término se puede también integrar por partes de la siguiente manera, obteniendo:

$$\int_a^b vp_1u'dx = vp_1u|_a^b - \int_a^b (uv'p_1 + uvp_1')dx \tag{2.28}$$

Por lo que, al combinar esos dos resultados, se obtiene:

$$\int_a^b v\mathbf{L}(u) = (vp_0u' + vp_1u)|_a^b + \int_a^b (-(vp_0)'u' - (vp_1)'u + vp_2u)dx \tag{2.29}$$

$$\int_a^b v\mathbf{L}(u) = (vp_0u + vp_1u - (vp_0)'u)|_a^b + \int_a^b ((vp_0)''u - (vp_1)'u + vp_2u)dx \tag{2.30}$$

y entonces:

$$\int_a^b v\mathbf{L}[u]dx = (vp_0u' + vp_1u - (vp_0)'u)|_a^b + \int_a^b u((p_0v)'' - (p_1v)' + p_2v)dx \tag{2.31}$$

Al comparar(2.24) con (2.31) se deduce que

$$\begin{aligned} \mathbf{L}^*[v] &= (p_0v)'' - (p_1v)' + p_2v \\ \mathbf{L}^*[v] &= p_0v'' + (2p_0' - p_1)v' + (p_0'' - p_1' + p_2)v \end{aligned} \tag{2.32}$$

Por lo que el operador L^* tiene la forma:

$$\mathbf{L}^* \equiv p_0\frac{d^2}{dx^2} + (2p_0' - p_1)\frac{d}{dx} + (p_0'' - p_1' + p_2) \tag{2.33}$$

La ecuación $\mathbf{L}^*[v] = 0$ es conocida como la *Ecuación Formalmente Adjunta* de $\mathbf{L}[u] = 0$.

Cuando $\mathbf{L} = \mathbf{L}^*$, se dice que $\mathbf{L}$ es formalmente autoadjunto. Para que el operador diferencial de segundo orden sea formalmente autoadjunto se puede ver, de las ecuaciones (2.23) y (2.32), que se debe cumplir con las siguientes relaciones:

$$p_1 = 2p_0' - p_1$$

$$p_2 = p_0'' - p_1' + p_2$$

Estas relaciones se cumplen solamente si $p_1 = p_0'$. Lo que implica que un operador diferencial lineal autoadjunto de segundo orden pueda expresarse en la forma

$$\mathbf{L} \equiv p_0 \frac{d^2}{dx^2} + p_0' \frac{d}{dx} + p_2 = \frac{d}{dx}(p_0 \frac{d}{dx}) + p_2 \tag{2.34}$$

Llamemos $p(x)$ a p_0 y $q(x)$ a p_2 respectivamente. Se dice que una ecuación diferencial lineal de segundo orden es autoadjunta, si se puede escribir en la forma

$$(p(x)u')' + q(x)u = 0 \tag{2.35}$$

A esta última ecuación se le conoce como la *ecuación de Sturm-Liouville.*

2.3 Métodos de solución

2.3.1 Método del matrizante

Se sabe que una ecuación diferencial de orden n, puede ser puesta en la forma de un sistema de n ecuaciones diferenciales de primer orden. Y esta a su vez puede ser escrita en forma matricial, como se muestra a continuación:

$$\frac{d\mathbf{Z}}{dz} = \mathbf{A}(z)\mathbf{Z} \tag{2.36}$$

donde $\mathbf{A}(z)$ es una matriz $n \times n$ de funciones complejas de variable real.

Se puede demostrar que esta ecuación tiene la solución:

$$\mathbf{Z}(z) = \mathbf{\Omega}(z, z_0)\mathbf{Z}(z_0) \tag{2.37}$$

Donde el operador $\mathbf{\Omega}(z, z_0)$ se expresa de la siguiente forma

$$\begin{aligned}\mathbf{\Omega}(z, z_0) &= \left\{\mathbf{I} + \int_{z_0}^{z} \mathbf{A}(\sigma)d\sigma + \int_{z_0}^{z}\int_{z_0}^{\sigma} \mathbf{A}(\sigma)\mathbf{A}(\sigma_1)d\sigma_1 d\sigma \right. \\ &+ \int_{z_0}^{z}\int_{z_0}^{\sigma}\int_{z_0}^{\sigma_1} \mathbf{A}(\sigma)\mathbf{A}(\sigma_1)\mathbf{A}(\sigma_2)d\sigma_2 d\sigma_1 d\sigma + \cdots \\ &+ \left. \int_{z_0}^{z}\int_{z_0}^{\sigma}\int_{z_0}^{\sigma_1}\cdots\int_{z_0}^{\sigma_{n-1}} \mathbf{A}(\sigma)\ldots\mathbf{A}(\sigma_n)d\sigma_n\ldots d\sigma \right\}. \end{aligned} \tag{2.38}$$

Al operador $\mathbf{\Omega}$ se le conoce como el **matrizante**. Para mayor información acerca de lo anterior, el lector puede referirse a [9].

2.3.2 La representación exponencial

Bajo ciertas condiciones, el matrizante puede ser expresado en términos de la función exponencial.

$$\mathbf{\Omega}(z, z_0) = e^{\int_{z_0}^{z} \mathbf{A}(\sigma)d\sigma} \tag{2.39}$$

Por lo que la matriz $\mathbf{Z}(z)$ puede representarse como:

$$\mathbf{Z}(z) = exp\left(\int_{z_0}^{z} \mathbf{A}(\sigma)d\sigma\right)\mathbf{Z}(z_0) \tag{2.40}$$

Un resultado interesante es el de encontrar la solución con condición inicial. Si consideramos una matriz $\mathbf{A}(z)$ integrable en un intervalo $I = [z_0, z]$, con $z_0 = 0$ y $\left[\mathbf{A}(z),\ \int_{z_0}^{z} \mathbf{A}(u)d(u)\right] = 0$, donde el paréntesis anterior representa el paréntesis de Lie,

$$\left[\mathbf{A}(z), \int_{z_0}^{z} \mathbf{A}(u)du\right] = \mathbf{A}(z)\int_{z_0}^{z} \mathbf{A}(u)du - \left(\int_{z_0}^{z} \mathbf{A}(u)du\right)\mathbf{A}(z)$$

Entonces el sistema $\frac{d\mathbf{Z}(z)}{dz} = \mathbf{A}(z)\mathbf{Z}(z)$, con condición inicial $\mathbf{Z}(0) = 1$, tiene una solución matricial:

$$\mathbf{Z}(z) = exp\left(\int_{z_0}^{z} \mathbf{A}(u)du\right)$$

Veamos cómo en el caso en el que la matriz de coeficientes es constante, el matrizante puede ser expresado en la forma matricial (2.39). Es fácil ver cómo en el caso en que la matriz A es constante, la expresión (2.38) puede escribirse de la siguiente forma:

$$\mathbf{\Omega}(z, z_0) = \left\{\mathbf{I} + \mathbf{A}(z - z_0) + \mathbf{A}^2\frac{(z - z_o)^2}{2!} + \mathbf{A}^3\frac{(z - z_o)^3}{3!} + \dots\right\} \tag{2.41}$$

Reconociendo la igualdad $\sum_{n=0}^{\infty}\frac{x^n}{n!} = e^x$ se obtiene:

$$\mathbf{\Omega}(z, z_0) = e^{A(z-z_0)} \tag{2.42}$$

De hecho, esto se esperaba; ya que la matriz constante $\mathbf{A}$ es integrable en I y

$\left[\mathbf{A}, \int_{z_0}^{z} \mathbf{A}d(u)\right] = 0$

La igualdad (2.42) será de mucha utilidad en las siguientes secciones.

2.3.3 Solución en el espacio fase

El *espacio fase* sirve para analizar el comportamiento global que tienen las soluciones de un sistema de ecuaciones diferenciales de primer orden; también se le conoce como *plano fase*, cuando se trata de un sistema de dos ecuaciones. El espacio es definido por las coordenadas de cada vector o columna independiente de la matriz solución, correspondiente a nuestro sistema de ecuaciones diferenciales.

El *plano fase* nos ofrece entonces información cualitativa acerca de las soluciones de nuestra EDSO.

Consideremos el operador diferencial de segundo orden $\mathbf{L}$, aplicado a la función u. Este operador, siendo la forma normal de Sturm-Liouville, tiene la forma $\mathbf{L}[u] \equiv -(pu')' + qu = \lambda u$.

Al transformarla en un par de ecuaciones diferenciales homogéneas lineales de primer orden, su representación matricial es la siguiente:

$$\frac{d}{dt}\begin{bmatrix} \varphi & \psi \\ p\varphi' & p\psi' \end{bmatrix} = \begin{bmatrix} 0 & 1/p \\ q-\lambda & 0 \end{bmatrix}\begin{bmatrix} \varphi & \psi \\ p\varphi' & p\psi' \end{bmatrix} \tag{2.43}$$

Las columnas de la matriz solución son los vectores solución del sistema (2.43).

Ahora bien, si ponemos

$$Z = \begin{bmatrix} \varphi & \psi \\ p\varphi' & p\psi' \end{bmatrix}$$

y

$$M = \begin{bmatrix} 0 & 1/p \\ q-\lambda & 0 \end{bmatrix}$$

la ecuación (2.43) se puede escribir:

$$\frac{d}{dt}\mathbf{Z} = \mathbf{MZ}$$

con condiciones iniciales:

$$\mathbf{Z}(t_0) = \mathbf{1}$$

2.3.4 Campo de movimiento circular

Dada la igualdad (2.43), supongamos que

$$\mathbf{M} = \begin{bmatrix} 0 & 1 \\ -E & 0 \end{bmatrix}$$

y que $E > 0$

Según lo expuesto en la sección anterior y lo que se describe en [9] acerca de que cuando la matriz $\mathbf{M}$ es constante, se obtiene una solución exponencial de la forma:

$$\mathbf{Z}(x) = e^{M(x-x_0)}\mathbf{Z}(x_0),$$

donde $(x_0, \mathbf{Z}(x_0))$ es la condición inicial de la ecuación diferencial que queremos resolver. Ahora bien, como la serie escalar dada por $\sum_{n=0}^{\infty} \frac{x^n}{n!} = e^x$ se extiende al caso de las matrices, la solución puede ser expresada de la siguiente forma:

$$\mathbf{Z}(x) = \sum_{n=0}^{\infty} \frac{(M(x-x_0))^n}{n!}\mathbf{Z}(x_0) = \sum_{n=0}^{\infty} \frac{M^n(x-x_0)^n}{n!}\mathbf{Z}(x_0)$$

$$\mathbf{Z}(x) = (I + M(x-x_0) + \ldots + \frac{M^n(x-x_0)^n}{n!})\mathbf{Z}(x_0)$$

Observemos el comportamiento de los exponentes de M:

$$\begin{aligned} M &= \begin{bmatrix} 0 & 1 \\ -E & 0 \end{bmatrix} \\ M^2 &= \begin{bmatrix} -E & 0 \\ 0 & -E \end{bmatrix} = -E\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ M^3 &= -E\begin{bmatrix} 0 & 1 \\ -E & 0 \end{bmatrix} \end{aligned} \tag{2.44}$$

Entonces, al reemplazar estas matrices en la ecuación, se tiene:

$$\mathbf{Z}(x) = \left\{ I + (x - x_0)M - E\frac{(x-x_0)^2}{2!}I - E\frac{(x-x_0)^3}{3!}M + E^2\frac{(x-x_0)^4}{4!}I + \ldots \right\} \mathbf{Z}(x_0)$$

Agrupando del lado izquierdo las potencias pares y del lado derecho las impares, se obtiene lo siguiente:

$$\begin{aligned} \mathbf{Z}(x) &= \left\{ I \left(1 - E\frac{(x-x_0)^2}{2!} + E^2\frac{(x-x_0)^4}{4!} - E^3\frac{(x-x_0)^6}{6!} + \ldots \right) \right. \\ &+ \left. \frac{M}{\sqrt{E}} \left(E^{\frac{1}{2}}\frac{(x-x_0)^1}{1!} - E^{\frac{3}{2}}\frac{(x-x_0)^3}{3!} + \ldots \right) \right\} \mathbf{Z}(x_0) \end{aligned} \tag{2.45}$$

En forma compacta tendríamos:

$$\mathbf{Z}(x) = \left\{ I \sum_{i=0}^{\infty} (-1)^i \frac{\left(E^{\frac{1}{2}}(x-x_0)\right)^{2i}}{(2i)!} + \frac{M}{\sqrt{E}} \sum_{i=0}^{\infty} (-1)^i \frac{\left(\sqrt{E}(x-x_0)\right)^{2i+1}}{(2i+1)!} \right\} \mathbf{Z}(x_0)$$

Reconociendo las igualdades:

$$\cos\sqrt{E}(x-x_0) = \sum_{i=0}^{\infty} (-1)^i \frac{\left(E^{\frac{1}{2}}(x-x_0)\right)^{2i}}{(2i)!}$$

$$\sin\sqrt{E}(x-x_0) = \sum_{i=0}^{\infty} (-1)^i \frac{\left(\sqrt{E}(x-x_0)\right)^{2i+1}}{(2i+1)!}$$

De aquí se obtiene que:

$$\mathbf{Z}(x) = \left\{ I\cos\sqrt{E}(x-x_0) + \frac{M}{\sqrt{E}}\sin\sqrt{E}(x-x_0) \right\} \mathbf{Z}(x_0)$$

Por lo tanto, si ponemos:

$$\mathbf{Z}(x_0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

en la cual se encuentran los dos vectores linealmente independientes

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

y

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

los cuales representan las dos condiciones iniciales y $x_0 = 0$, entonces se obtiene:

$$\mathbf{Z}(x) = \begin{bmatrix} \cos\sqrt{E}(x) & \frac{\sin\sqrt{E}(x)}{\sqrt{E}} \\ -\sqrt{E}\sin\sqrt{E}(x) & \cos\sqrt{E}(x) \end{bmatrix} \tag{2.46}$$

Al graficar las coordenadas de los vectores obtendríamos una serie de elipses, como las presentadas en la figura 2.2, cuyas dimensiones dependen de los valores de E.

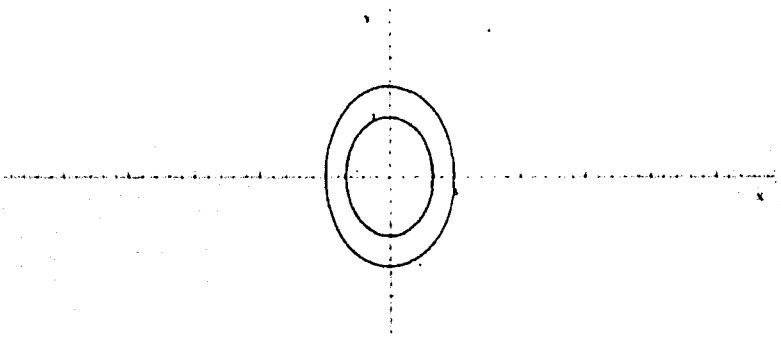


Figura 2.2: Las soluciones (2.46) representadas en el espacio fase

2.3.5 Campo de movimiento hiperbólico 45^0

Ahora bien, si tomamos la misma matriz que en la sección anterior, pero ahora con $E < 0$, entonces tendríamos:

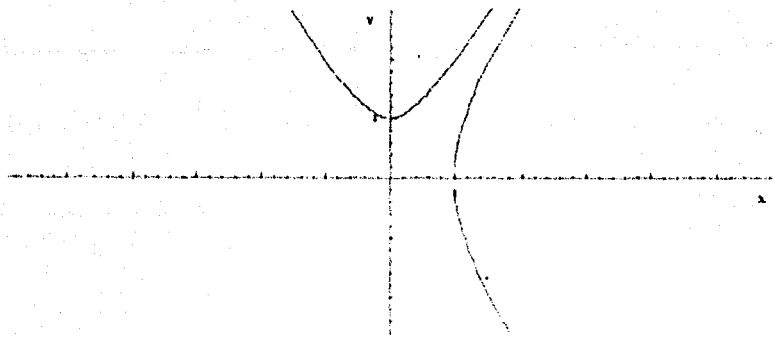


Figura 2.3: Las soluciones (2.49) representadas en el espacio fase

$$\mathbf{Z}(x) = \left\{ I\left(1 - E\frac{(x-x_0)^2}{2!} + E^2\frac{(x-x_0)^4}{4!} - E^3\frac{(x-x_0)^6}{6!} + \ldots\right) + M\left(\frac{(x-x_0)^1}{1!} - E\frac{(x-x_0)^3}{3!} + E^2\frac{(x-x_0)^5}{5!}\ldots\right)\right\}\mathbf{Z}(x_0) \quad (2.47)$$

$$\mathbf{Z}(x) = \left\{ I\left(1 + \frac{[\sqrt{|E|}(x-x_0)]^2}{2!} + \frac{[\sqrt{|E|}(x-x_0)]^4}{4!} + \frac{[\sqrt{|E|}(x-x_0)]^6}{6!} + \ldots\right) + \frac{M}{\sqrt{-E}}\left(\frac{[\sqrt{|E|}(x-x_0)]^1}{1!} + \frac{[\sqrt{|E|}(x-x_0)]^3}{3!} + \ldots\right)\right\}\mathbf{Z}(x_0) \quad (2.48)$$

De donde se obtiene finalmente:

$$\mathbf{Z}(x) = \left\{\mathbf{I}\cosh\left(\sqrt{|E|}(x-x_0)\right) + \frac{\mathbf{M}}{\sqrt{|E|}}\sinh\left(\sqrt{|E|}(x-x_0)\right)\right\}\mathbf{Z}(x0)$$

Y entonces, si tomamos $\mathbf{Z}(x_0) = \mathbf{I}$ y $x_0 = 0$, la forma matricial de la solución queda:

$$\mathbf{Z}(x) = \left\{ \mathbf{I} \cosh\left(\sqrt{|E|}(x - x_0)\right) + \frac{\mathbf{M}}{\sqrt{|E|}} \sinh\left(\sqrt{|E|}(x - x_0)\right) \right\} \mathbf{Z}(x0)$$

Y entonces, si tomamos $\mathbf{Z}(x_0) = \mathbf{I}$ y $x_0 = 0$, la forma matricial de la solución queda:

$$\mathbf{Z}(x) = \begin{bmatrix} \cosh\left(\sqrt{|E|}x\right) & \frac{\sinh\left(\sqrt{|E|}x\right)}{\sqrt{|E|}} \\ \sqrt{|E|}\sinh\left(\sqrt{|E|}x\right) & \cosh\left(\sqrt{|E|}x\right) \end{bmatrix} \tag{2.49}$$

Al graficar en el espacio fase las soluciones, se obtiene un comportamiento hiperbólico como se muestra en la figura 2.3.

2.3.6 Diagrama de estabilidades

Existen más casos que resultan dependiendo de los valores de la matriz de coeficientes constante **M**. Consideremos el siguiente sistema lineal:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} \tag{2.50}$$

donde a, b, c y d son constantes reales. Y sean $p = a + d$, $q = ad - bc$ y $\Delta = p^2 - 4q$. Entonces, el nodo crítico $(0, 0)$ es un:

- nodo, si $q > 0$ y $\Delta \geq 0$
- punto silla, si $q < 0$
- punto espiral, si $p \neq 0$ y $\Delta < 0$
- centro, si $p = 0$ y $q > 0$

Por otro lado, también se tiene que el punto crítico $(0, 0)$ es:

- asintóticamente estable, si $q > 0$ y $p < 0$
- estable, si $q > 0$ y $p = 0$
- inestable, si $q < 0$ o $p > 0$

Las figuras correspondientes a estos casos, se muestran en el diagrama de estabilidades de la figura 2.4.

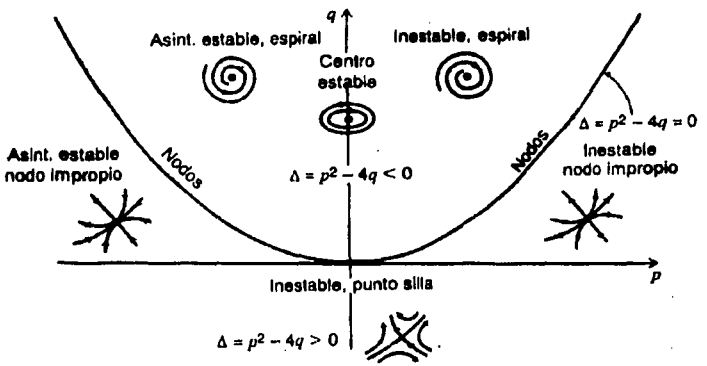


Figura 2.4: Diagrama de estabilidades

Para ejemplificar lo anterior, tomemos el caso visto en la sección (2.3.4), donde

$$M = \begin{bmatrix} 0 & 1 \\ -E & 0 \end{bmatrix}$$

Poniendo $a = 0$, $b = 1$, $c = -E$ y $d = 0$, tenemos: $p = 0$, $q = E$ y $\Delta = -4E$ con $E > 0$. Por tanto, según lo visto, el punto $(0,0)$ es un centro estable.

Capítulo 3

Núcleo numérico de JSERO

3.1 Antecedentes de JSERO

SERO (Second Order Real Ordinary Equation) es un paquete desarrollado por el doctor Harold V. McIntosh. Como su nombre lo indica, ha sido creado para hacer una integración numérica de las ecuaciones diferenciales ordinarias reales de segundo orden; además de tener el propósito específico de desplegar visualmente las soluciones de las ecuaciones diferenciales. Sin embargo, el sistema no solamente arroja resultados cuantitativos representados en el espacio solución, también permite acceder a información de tipo cualitativa acerca de las funciones propias de la ecuación diferencial de Schrödinger. Esto se logra a través de la graficación de las soluciones en el espacio fase, el mapa de estabilidad potencial periódico y los valores asintóticos.

Inicialmente, este sistema fue creado en *Fortran*; pero con el advenimiento de la programación orientada a objetos se decidió hacer una versión en el lenguaje de programación *C-Objetivo*, que corre en *Open Step* (OS) y *Next Step* (NS). *C-objetivo* es un lenguaje de programación netamente orientado a objetos; esto hace que el código que se escribe para realizar un sistema sea sumamente elegante, además de que ofrece una extraordinaria interfaz visual.

Sin embargo, el problema que enfrenta la versión de *C-Objetivo* que corre en OS y NS es que estas dos plataformas están a punto de desaparecer, debido a que no tuvieron el impacto que se deseaba en el mercado.

De ahí la necesidad por migrarlo a un lenguaje distinto de POO que sea más comercial y permita dejar intactas sus propiedades. Además, se busca que este nuevo lenguaje pueda correr en varias plataformas, para que no quede *"atrapado"* en una sola y de esta forma una mayor cantidad de usuarios pueda hacer uso de él.

Por otro lado, **JSERO**, al igual que **SERO**, realiza la integración numérica de la ecuación diferencial de Schrödinger aplicando el método de Runge Kutta. Aunque existen diversos métodos para la solución numérica de las EDSO, quizás este método sea el más usado; debido a que permite obtener muy buenas aproximaciones de las soluciones deseadas y puede ser fácilmente codificado. De hecho ofrece mejores aproximaciones que los métodos de Euler, Euler mejorado o el de Taylor; motivo por el cual se ha utilizado como base numérica para obtener las funciones propias de la ecuación de Schrödinger.

JSERO es un sistema que implementa básicamente el método numérico de Runge Kutta de cuarto orden; aunque también hace uso del método de sexto orden para ciertos potenciales y del método de cuarto orden para la ecuación inhomogénea.

De hecho, lo que realmente hace **JSERO** internamente es trabajar con una reducción de la ecuación diferencial lineal de segundo orden en un par de ecuaciones de primer orden escritas de una forma matricial. Esto es, **JSERO** puede trabajar con ecuaciones de la forma:

$$\frac{d\mathbf{Z}(z)}{dz} = M\mathbf{Z}(z) \tag{3.1}$$

Donde M es una matriz de coeficientes de 2×2, cuyos términos pueden ser o no funciones de la variable independiente z. Mientras que $\mathbf{Z}$ es la matriz de soluciones del sistema de ecuaciones.

Como ya se ha dicho, el núcleo numérico de **SERO** está esencialmente basado en el método de integración numérica de Runge Kutta y el sistema que actualmente se plantea, llamado **JSERO**, también está basado en este método de integración numérica. En el presente capítulo se presentan las clases computacionales que implementan dicho método. Asimismo, se da una breve descripción de la forma en que otros componentes de **JSERO** interactuán con el paquete, donde se encuentran las clases que definen el método numérico.

3.2 Método de Runge Kutta

El método más usado de Runge Kutta en la práctica es el de cuarto orden. Matemáticamente se expresa de la siguiente manera:

$$y_{i+1} = y_i + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

Donde:

$$\begin{aligned} K_1 &= hf(x_i, y_i) \\ K_2 &= hf(x_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}K_1) \\ K_3 &= hf(x_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}K_2) \\ K_4 &= hf(x_i + h, y_i + K_3) \end{aligned}$$

y donde h es el llamado paso de integración, cuyo valor, en el caso de **JSERO**, es de 0.05, y

$$\frac{dy}{dx} = f(x, y)$$

con condición inicial $y(x_0) = y_0$

Se puede ver claramente que el método sólo es aplicable a una ecuación diferencial de primer orden. Sin embargo, dado que una ecuación diferencial de orden superior puede ser escrita como un sistema de ecuaciones diferenciales de primer orden de la forma (3.1), el método se puede extender al caso de una ecuación diferencial de orden superior a 1. Por lo que resulta muy sencillo, computacionalmente hablando, plantear el método de Runge Kutta concretamente para la ecuación de Schrödinger.

3.3 El comportamiento dinámico y la estructura estática

UML dispone de una serie de vistas que permiten la descripción amplia de las características de un sistema computacional. Una vista es simplemente un subconjunto

de constructores del modelado UML, que representa un aspecto de un sistema. Cada tipo de diagrama provee una notación visual para los conceptos en cada vista.

Las vistas pueden dividirse en tres áreas: clasificación estructural, comportamiento dinámico y administración del modelo.

La clasificación estructural describe las partes del sistema y su relación con otros componentes de una manera estática; mientras que el comportamiento dinámico describe el comportamiento de un sistema en el tiempo. El comportamiento puede ser descrito como una serie de cambios en el sistema.

De esta manera, UML permite describir completamente las características más importantes de **JSERO** en forma de metamodelo, haciendo muy comprensible para el lector y desarrollador su constitución y la interacción de los componentes.

Por otro lado, paralélamente a los diagramas, se plantea en esta sección parte del código en *Java* correspondiente a dichos diagramas. Esto ayuda a tener una idea clara de la forma casi transparente con la que se puede plasmar un metamodelo en código de un lenguaje orientado a objetos.

3.3.1 La clase OpMat

OpMat es la clase principal del paquete. En ella se agregan todos los objetos que representan a los potenciales (barrera, pozo cuadrado, etc.) y los objetos que representan las diferentes versiones del método numérico de Runge Kutta (cuarto orden para la ecuación homogénea e inhomogénea o sexto orden).

Ante un cambio en el estado de los componentes de la barra de herramientas, realizado por uno de los actores externos, las clases derivadas de **Tool** (de la cual se hablará más detalladamente en el siguiente capítulo), envían el potencial seleccionado por el usuario y los valores de los parámetros de la ecuación diferencial de Schrödinger, a la clase **OpMat**, la cual recalcula los valores de las funciones solución de dicha ecuación. Y entonces, esta clase hace uso de otras dos clases: **POT** y **RungeKutta**; para aplicar los métodos numéricos de Runge Kutta. Gráficamente, esto se puede con más detalle en el diagrama de colaboración que se encuentra en la figura 3.1.

Se puede observar que la figura sólo muestra el caso en el que una instancia de **Tools** actualiza los valores del vector solución para el caso específico del potencial

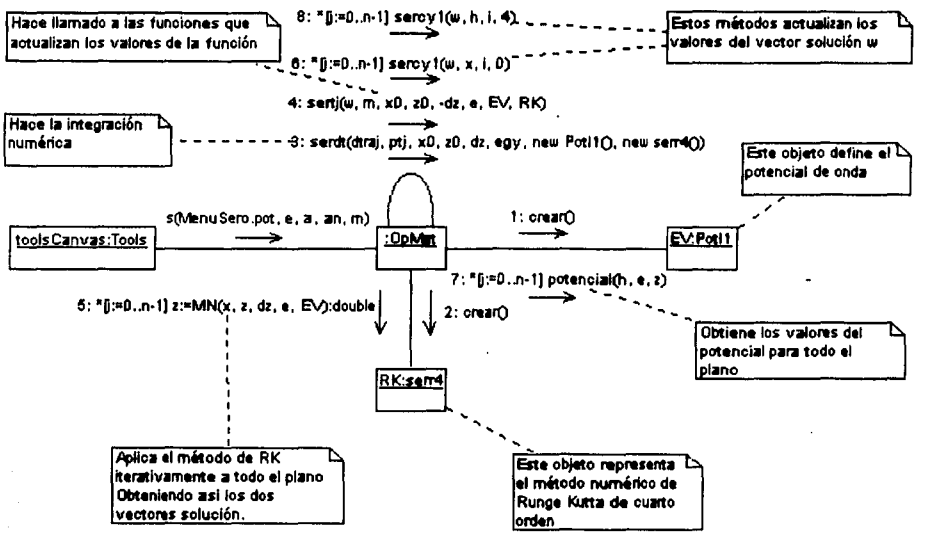


Figura 3.1: Desencadenamiento de eventos cuando se efectúa una acción sobre alguno de los objetos de la barra de herramientas del espacio solución

de onda plana. El proceso que se sigue cuando **Tools** actualiza los valores del vector solución para los otros potenciales es el mismo, por lo que se omite la presentación de los diagramas para el resto de los potenciales.

Mientras que las clases derivadas de **ContainerGraph** (**Contenedor**, **ContPSM**, **ContPhasePlane** y **ContAValues**), de las cuales se hablará también en el próximo capítulo, únicamente hacen uso de la clase **OpMat** para obtener los valores de las funciones solución. Esto lo logra haciendo un llamado a la clase **OpMat** mediante el método **getSolVal()**. Lo anterior se ilustra gráficamente en la figura 3.2.

De lo anterior se deduce que la representación en la vista de clases de **OpMat** debe ser la mostrada en la figura 3.3. Como se puede apreciar en dicha figura, existen más métodos que deben ser implementados, sin embargo estos métodos realizan una tarea más específica; por lo que no se habló de ellos durante la elaboración de los diagramas de colaboración.

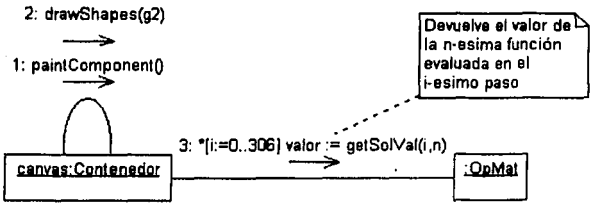


Figura 3.2: Eventos que se desencadenan cuando el contenedor de las gráficas (en este caso del espacio solución) se tiene que volver a dibujar

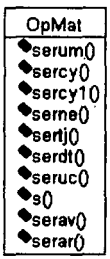


Figura 3.3: Estructura de la clase **OpMat**

3.3.2 La clase Potencial

Una de las principales funcionalidades que ofrece el sistema es la posibilidad de escoger de entre una serie de potenciales a introducir dentro de la ecuación diferencial de Schrödinger. De hecho, el usuario puede escoger quince potenciales diferentes. Estos potenciales son:

- Onda plana, Schrödinger
- Pozo finito, Schrödinger
- Barrera finita, Schrödinger
- Escalón, Schrödinger
- Oscilador armónico, Schrödinger

- Oscilador Cuártico, Schrödinger
- Oscilador armónico factorizado,Dirac
- Gausiano, Dirac
- Sinusoidal
- Lineal
- Coulomb
- Cuadrado Inverso
- Damped classical oscillator with forcing
- Oscilador armónico de Dirac
- Cuadrado inverso factorizado

Se ha decidido construir una clase abstracta, llamada **Pot**, a partir de la cual se derivan todos los potenciales. Esta clase define los métodos que sus clases derivadas deben poseer. Como se puede ver en la figura 3.4, sólo define un método llamado **potencial()**, el cual como su nombre lo indica implementa el potencial. Existen quince clases derivadas, una para cada potencial que se implementa en el sistema.

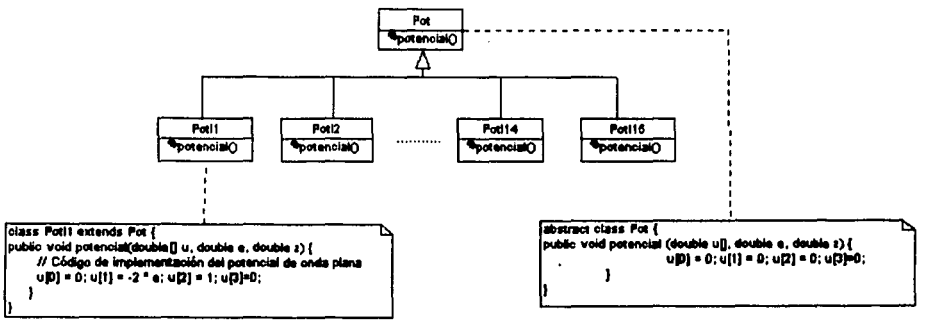


Figura 3.4: Representación estructural de la clase **Pot** y de sus clases derivadas

Esta figura muestra además el código en *Java* de uno de los potenciales para ilustrar la forma en que se realiza esta implementación. Se puede observar, del código en *Java* que se muestra en la figura anterior, la enorme sencillez con la que se puede

hacer una clase abstracta (anteponiedo la palabra reservada **abstract**) y una clase derivada (posponiendo la palabra reservada **extends** y el nombre de la clase padre). Por otra parte, como se había mencionado anteriormente, una de las principales ventajas de la POO es la facilidad con la que se puede dar mantenimiento al sistema. Del diagrama anterior, por ejemplo, se puede inferir que agregar o quitar un potencial al sistema se traduce en agregar o quitar una clase en el diagrama sin que esto repercuta en el resto del código. Evidentemente, sólo se tiene que cuidar que no se hagan llamadas a esta clase desde el resto del código; esto da un gran control al desarrollador sobre el sistema, ya que a través de los diagramas se sabe lo que va a suceder ante un cambio en éste, incluso sin tener que hacer modificaciones en el código.

3.3.3 La clase Runge Kutta

Esta clase es quizá la más importante de todas las que utiliza este paquete, debido a que en ella se implementa el método numérico de Runge Kutta; es decir, en ella se realiza la integración numérica de la ecuación diferencial de Schrödinger. Su representación en UML se puede observar en la figura 3.5.

Se trata de una clase abstracta llamada **RungeKutta**, de la cual se derivan las clases **serr4**, **serri4** y **serr6**, que representan a los métodos de Runge Kutta de cuarto orden para la ecuación homogénea, cuarto orden para la ecuación inhomoénea y sexto orden respectivamente.

De nuevo, se aclara que sólo se muestra en la figura el código para uno de los métodos (cuarto orden), con el fin de aclarar el diagrama.

Se puede ver claramente que la clase **RungeKutta** contiene los métodos que las clases derivadas utilizan en común. Esto se logra mediante el concepto computacional llamado "herencia".

Se ha omitido también el código que implementa cada uno de los métodos, porque además de ser muy largo no es de interés ver la parte técnica. En realidad, el interés radica en cómo se pasa del modelo en UML al código en *Java*.

Mientras que, por otro lado, haciendo uso de otro concepto de la POO, llamado "polimorfismo", se implementa dentro de cada clase derivada el método **MN()**. Esto se debe a que cada clase implementa de una manera distinta este método; es decir,

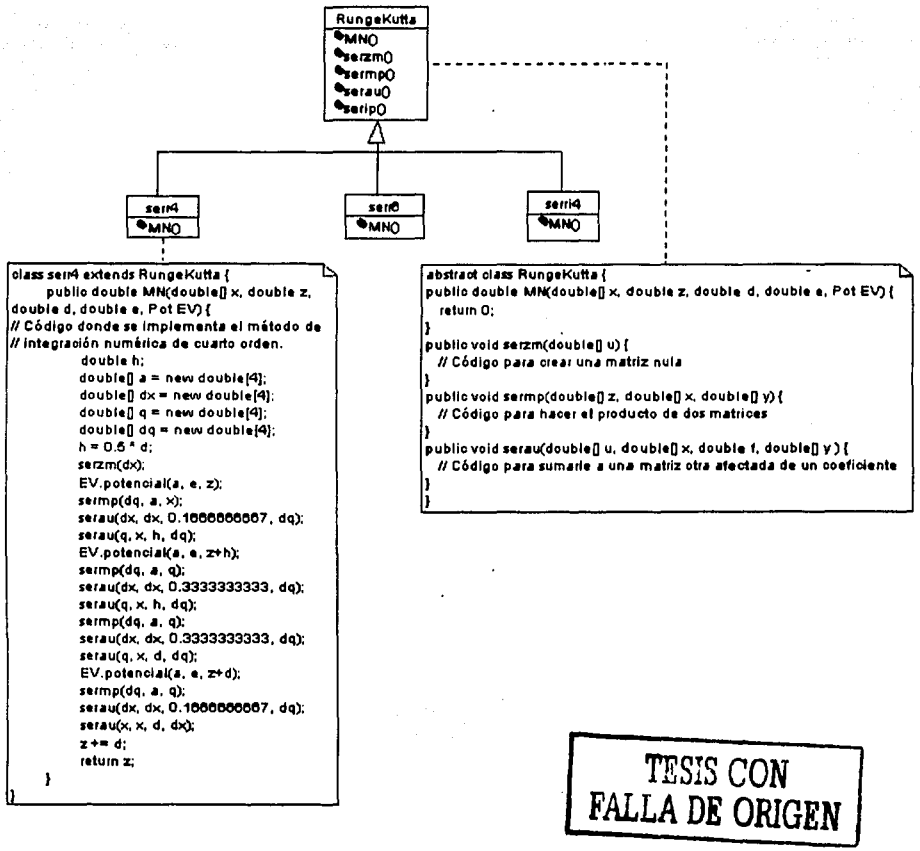


Figura 3.5: Representación estructural de la clase **RungeKutta** y de sus clases derivadas (arriba). Se muestra también parte del código que implementa el método numérico de Runge Kutta de cuarto orden (abajo)

su método numérico.

3.3.4 Diagrama general

De los anteriores diagramas se obtiene la estructura completa del núcleo numérico de **JSERO** (figura 3.6). Este diagrama permite al desarrollador tener un control enorme sobre el sistema; ya que, en caso de querer agregar, quitar o modificar funcionalidad, no necesita ver el código completo de **OpMat**, sino que simplemente se puede recurrir a los diagramas y determinar las consecuencias que tiene efectuar algún cambio en su estructura o funcionamiento. En el caso de grandes sistemas, esto se traduciría en un ahorro enorme de tiempo y dinero.

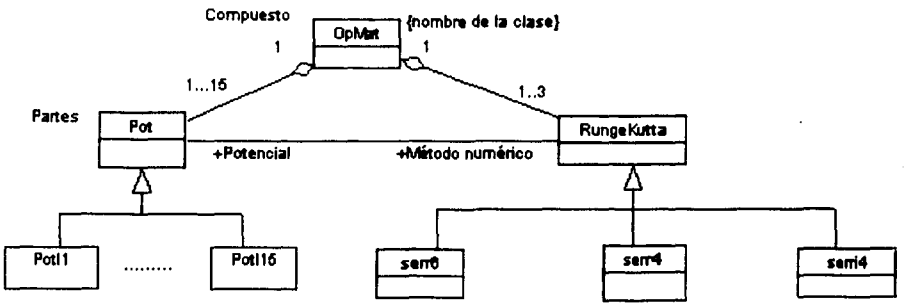


Figura 3.6: Estructura general de la clase **OpMat**, que es responsable de efectuar los cálculos de integración numérica de la ecuación diferencial

Capítulo 4

Diseño e implementación del sistema

La realización de un sistema de cómputo requiere de una metodología de desarrollo que sirva de guía para lograr un sistema robusto y bien diseñado. Normalmente los pasos que el desarrollador sigue son los de análisis, diseño e implementación. En el caso de un sistema basado en la POO existen varios métodos que pueden y deben ser usados. Es preciso hacer uso de alguno de estos métodos si se desea no sólo hacer un sistema basado en nuestra experiencia sino en la experiencia de muchos programadores experimentados. Uno de estos métodos podría ser los *patrones GRASP* [1] (General Responsability Asignment Software Patterns: patrones de los principios generales para asignar responsabilidades). Este método es muy recomendable y podría ser utilizado por todo aquel que pretenda hacer un *"buen"* sistema de software.

Sin embargo, el actual trabajo no aplica ningún método, debido a que se cuenta ya con un sistema que funciona correctamente en C-Objetivo; por lo que se puede prescindir del uso de alguna metodología de desarrollo, así como de la fase de análisis, puesto que se sabe perfectamente lo que el sistema debe realizar. El trabajo se enfoca entonces en el diseño en UML del sistema, con lo que se intenta especificar la estructura y el funcionamiento de cada uno de sus componentes de software.

En el diseño se busca dar solución a la necesidad de satisfacer los requerimientos y necesidades del sistema. En esta fase se lleva a cabo un proceso de reemplazar, complementar y mejorar el sistema existente. El diseño también muestra las actividades que son realizadas por los usuarios y por los objetos que constituyen el sistema

en conjunto.

Por otro lado, el presente capítulo muestra parte de la implementación del sistema en el lenguaje *Java*; esto permite mostrar al lector la manera en que los artefactos de UML y el código de un lenguaje de POO se relacionan.

4.1 Comportamiento del sistema

Este es un punto muy importante dentro del diseño del sistema; aquí se asignan las responsabilidades que cada objeto debe cumplir para lograr que el sistema funcione como se espera. También muestra la forma en que dichos objetos deben enviar los mensajes entre ellos, para que se realicen las tareas asignadas a cada uno.

4.1.1 Inicialización del sistema

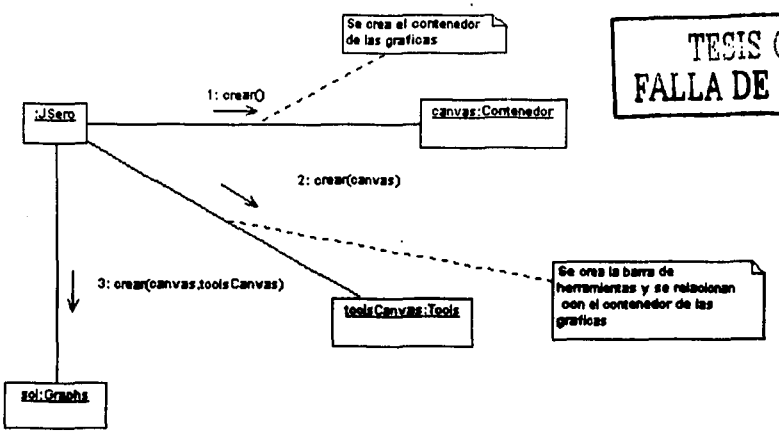


Figura 4.1: Inicialización de los objetos **canvas**, **toolsCanvas** y **sol**; necesarios para la creación del espacio solución

Al poner en marcha el sistema, se desencadena la creación de una serie de objetos que son agregados al sistema general. En la figura 4.1 se presenta la forma en que se lleva a cabo la creación de dos de estos objetos que son necesarios para la presentación del espacio solución; estos objetos son la barra de herramientas (**toolsCanvas**) y el contenedor de las gráficas (**Canvas**). En las siguientes secciones se explicará más detalladamente la función que tienen estos dos objetos dentro del sistema y su relación. Es importante señalar que se omite en la figura la forma en que se crean los contenedores y herramientas del resto de los espacios, porque esencialmente este proceso se realiza de la misma forma que en el caso del espacio solución.

4.1.2 Graficación de las funciones solución

Para poder graficar las funciones solución de la ecuación diferencial de Schrödinger, lo único que debe hacerse es presionar sobre alguno de los botones del panel *Solutions*. En la figura 4.2 se presenta la acción que se produce al presionar específicamente sobre el botón par. No se presentan las acciones que se producen al presionar sobre alguno de los otros botones, porque básicamente se realiza de la misma forma. Esto nos ayuda a evitar la repetición engorrosa de los diagramas.

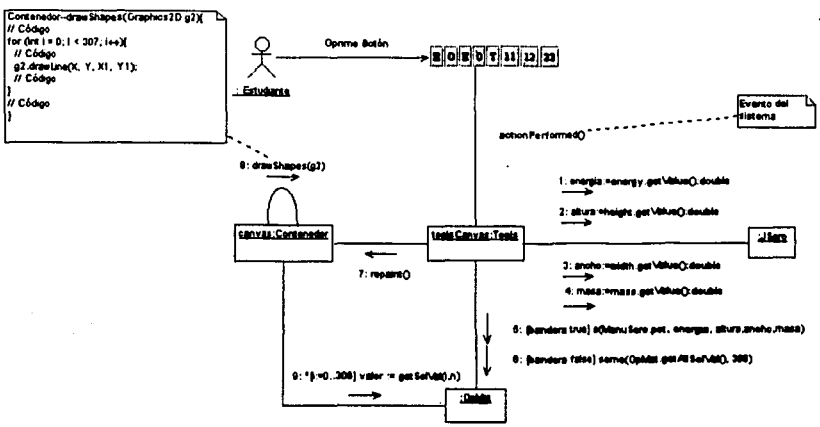


Figura 4.2: Eventos que se producen internamente cuando el usuario del sistema (estudiante) presiona sobre alguno de los botones del panel de soluciones

Posiblemente sea necesaria una explicación breve del diagrama. Primeramente, el sistema capta la acción de presionar sobre el botón de la barra de herramientas. Entonces, se obtienen los valores de los parámetros y se envian a la clase **OpMat**, para que realice los cálculos en función de estos valores; posteriormente, se le ordena al **Contenedor** que se vuelva a dibujar con los nuevos valores del vector solución que son obtenidos al hacer un llamado al método **getSolVal()**.

4.1.3 Impresión de las gráficas

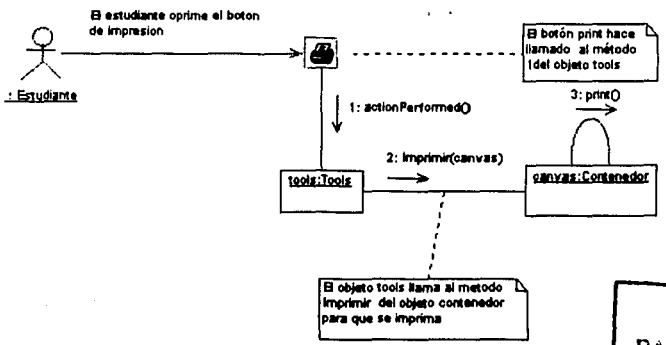


Figura 4.3: Eventos que se producen al interior del sistema cuando el usuario oprime el botón de impresión

Una de las funcionalidades más importantes que posee el sistema es la capacidad que tiene de imprimir las funciones que grafica. Esto lo realiza de una manera sumamente sencilla para el usuario; lo único que el usuario debe hacer es presionar el botón de impresión y escoger el formato de la impresión (número de páginas, impresora, etc.). Lo que sucede a continuación en el interior del sistema, y que por supuesto es invisible para el usuario, se muestra en la figura 4.3.

4.1.4 Limpiado de la pantalla

En efecto, una vez que se han graficado las funciones requeridas por el usuario, tal vez éste tenga la necesidad de volver a graficar otras funciones con otros parámetros. En lugar de quitar manualmente estas gráficas, el usuario únicamente debe presionar el botón **Clear** y se borran todas las gráficas que se tengan en pantalla.

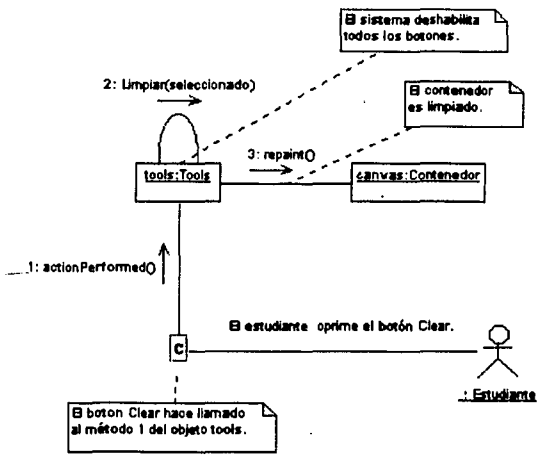


Figura 4.4: Comunicación entre objetos para realizar la tarea que el usuario solicita al sistema de limpiar la pantalla y desactivar los botones, cuando éste presiona el botón *Clear* para el contenedor y barra de herramientas del espacio solución

El procedimiento que se requiere para lograr esto se ilustra en la figura 4.4. En realidad éste es muy claro: una vez que se presiona el botón **Clear**, se llama al método **Limpiar()**, que pertenece a **Tools**; éste procede a desactivar todos los botones de la instancia de esta clase y posteriormente se actualiza la pantalla.

4.1.5 Selección del potencial

Se crea una instancia del **JComboBox**, que contiene todos los potenciales de los que dispone el sistema. Entonces se selecciona uno de estos y lo que sucede a continuación es lo que se muestra en la figura 4.5.

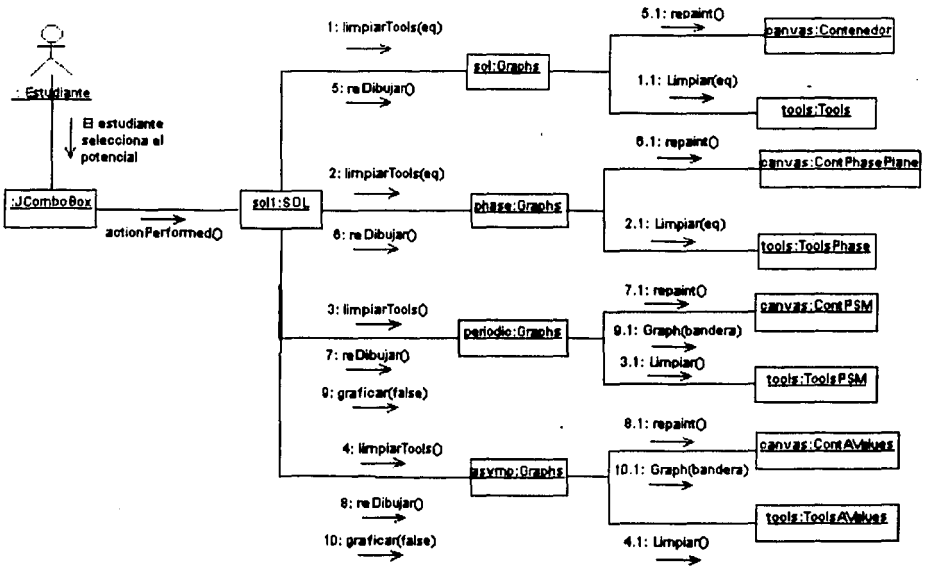


Figura 4.5: Eventos que se producen al interior del sistema cuando el usuario selecciona alguno de los potenciales del **JComboBox**

Primeramente, el método **actionPerformed()** de la instancia de la clase **SOL** es activado, lo que desencadena una serie de activaciones de los métodos de otras clases. En esencia, lo que sucede es que hace llamar a los métodos **limpiarTools()** de cada uno de los objetos que representan los diferentes espacios de graficación; lo que se traduce en una desactivación de todos los botones. Posteriormente, una vez que se desactivan estos botones, se procede al limpiado de las pantallas. Por otro lado, también se actualiza el potencial con el que se está trabajando, lo que nos permite hacer los cálculos de las funciones solución con el potencial que ha sido seleccionado.

4.2 Estructura estática del sistema

El diseño del diagrama de clases se deriva de los diagramas de colaboración. Una vez que se sabe cuáles son las conexiones entre los objetos y los métodos que estos deben de contener para realizar las tareas que tienen encomendadas, se procede a la elaboración del diagrama de clases. En éste último se plasman, de manera estática, las relaciones entre las clases, sus métodos y atributos.

4.2.1 Las clases Contenedor y ContainerGraph

La clase **Contenedor** (figura 4.6) ha sido diseñada para contener las gráficas de las funciones del espacio solución. Dentro de los métodos más importantes que implementa, destacan: **paintComponent()**, **drawShapes()** y **print()**. Los dos primeros métodos se encargan de dibujar en la pantalla las funciones que el usuario requiere; mientras que el último método se dedica a la impresión del contenido del objeto **Contenedor**. En él se designa el formato de la impresión, la escala y el contenido de la impresión.

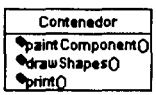


Figura 4.6: Representación estática de la clase **Contenedor**

Es preciso insistir en el hecho de que las otras clases **ContPhasePlane**, **ContPSM** y **ContAValues** funcionan básicamente de la misma manera que la clase **Contenedor**; por lo que no es necesario que se dé una explicación de su funcionamiento. Todas estas clases derivan de la clase padre **ContainerGraph**, que se muestra en la figura 4.7.

Figura 4.7: Representación estática de la clase **ContGraph**

Esta última implementa una serie de métodos necesarios para la correcta graficación de las funciones solución; aunque destaca el método **Imprimir()**, el cual crea un objeto de *Java* encargado de las impresiones y llama al método **print()**, que finalmente imprime el contenido del **Contenedor**. El diagrama general de clases se muestra en la figura 4.8.

En el código presentado en esta figura se observa que, para graficar las soluciones, se hace uso de la librería *Graphics2D* de *Java*; ésta es una herramienta excelente para graficar una gran variedad de figuras de una manera relativamente sencilla. Esta librería permite también manipular los colores, fondos y formas de las gráficas. El lector puede obtener mayor información acerca de su funcionamiento en [11]

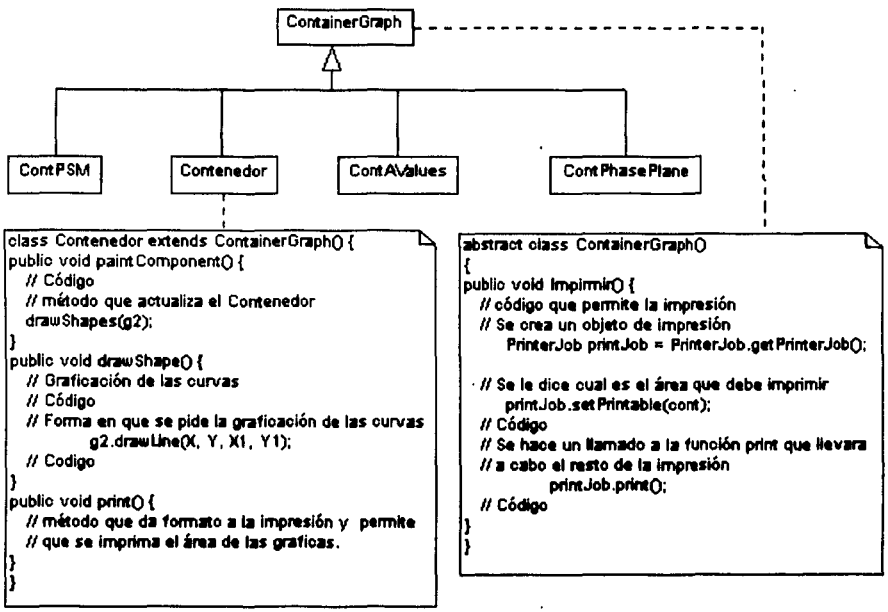


Figura 4.8: Representación de las clases diseñadas para contener las gráficas

4.2.2 Las clases Tools y Tool

Es importante tener un objeto en el que se grafican las funciones; pero no se pueden graficar si no se tiene un objeto que permita al usuario interactuar con el sistema y le diga cuáles son las funciones que desea visualizar y cuáles son los parámetros de éstas. Las clases **Tools** y **Tool** se encargan precisamente de resolver este problema.

La clase **Tool** (figura 4.9) es la clase padre de todas las clases (**Tools**, **ToolsPSM**, **ToolsAValues** y **ToolsPhase**) correspondientes a las herramientas de los diferentes espacios. En ella se implementa una serie de métodos que las clases derivadas heredan; estos métodos sirven esencialmente para introducir en la barra de herramientas todos sus componentes: el método **addToolPanel()** agrega a la barra de herramientas los paneles; los métodos **addToolButton()** y **addTool()** sirven para agregar los botones al panel y a la barra respectivamente; y finalmente el método **addToolSlider()** agrega los *sliders*.

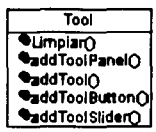


Figura 4.9: Estructura de la clase **Tool**

Mientras que la clase **Tools** implemeta una serie de métodos dentro de los que destacan **Limpiar()**, **graficando()**, **actionPerformed()** y **stateChanged()**. Su representación se muestra en la figura 4.10.

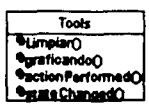


Figura 4.10: Estructura de la clase **Tools**, que corresponde a las herramientas disponibles en el espacio solución

El método **Limpiar()** es el encargado, como su nombre lo indica, de limpiar

la pantalla y desactivar todos los botones de nuestro menú de herramientas. Cada barra de herramientas implementa su propia función **Limpiar()**.

Por otro lado, el método **graficando()** es de gran importancia ya que se encarga de cumplir con la responsabilidad de actualizar los valores de los parámetros. Esto lo realiza al mandar a la instancia de la clase **OpMat**, encargada de la aplicación de los métodos muméricos de Runge Kutta, los parámetros potencial seleccionado, energía y masa de la partícula, y altura y ancho del potencial al cual está sometida dicha partícula.

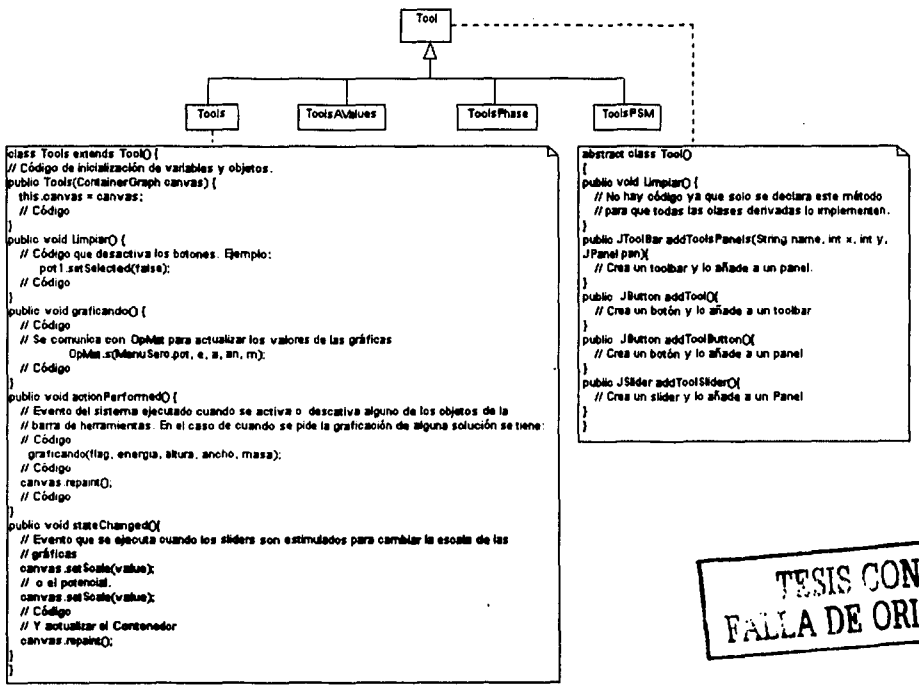


Figura 4.11: Representación de las barras de herramientas de los distintos espacios solución (arriba). Mapeo correspondiente al código en *Java* (abajo).

Finalmente, los dos últimos métodos (**actionPerformed()** y **stateChanged()**) se encargan de captar los estímulos exteriores (provenientes del usuario, que podría ser un estudiante). Estos estímulos serían, por ejemplo, los de presionar un botón o desplazar el *slider* que representa el valor de los parámetros. El diagrama general correspondiente a estas dos clases se visualiza en la figura 4.11.

Se ha mencionado ya cómo operan los objetos de la clase **Tools**. Ahora vamos a aclarar cómo está constituida dicha clase; es decir, vamos a determinar cómo son agregados los botones, *sliders*, páneles, etc., en la barra de herramientas. En la figura 4.12 se muestra el diagrama que permite visualizar esto.

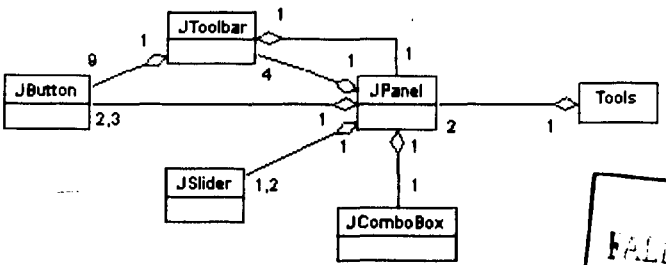


Figura 4.12: Composición de objetos que forman la barra de herramientas del espacio solución

Es importante señalar que todas estas clases pertenecen a la librería *Swing* de *Java*, excepto la clase **Tools**.

4.2.3 La clase Graphs

Esta clase (mostrada en la figura 4.13) es básicamente una extensión de la clase **JPanel** (perteneciente a la librería Swing). Ha sido diseñada para contener los objetos de la barra de herramientas y los contenedores, y posteriormente se agrega a la interfaz general del programa.

Los métodos que implementa son **limpiarTools()**, **reDibujar()** y **graficar()**. Donde **limpiarTools()** sólo se encarga de pedir al objeto **Tools** que limpie sus objetos; mientras que **reDibujar()** pide al objeto **canvas** que limpie su contenido.

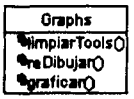


Figura 4.13: Estructura de la clase **Graphs**

Por otro lado, **graficar()** es usada únicamente por las instancias de **ContPSM** y **ContAValues**, avisándoles si hay que graficar o no sus funciones.

4.2.4 Diagrama general de JSERO

La clase **Contenedor** está estrechamente ligada a la clase **Tools**, debido a que un cambio en alguno de los objetos de esta última repercute en un cambio en la primera. Esto significa que frente a una selección de alguno de los elementos de las herramientas, las gráficas del **Contenedor** sufren un cambio (se agregan, se eliminan o se modifican las gráficas). Lo que se traduce en una asociación entre las dos clases.

De todo lo anterior se obtiene el diagrama general que muestra la estructura estática de todo el sistema **JSERO**. Este diagrama se encuentra en la Figura 4.14.

Por otro lado, se puede observar que resulta extremadamente fácil para cualquier persona tener una idea clara de todas las partes que componen al paquete.

Se puede apreciar en el diagrama que se introdujeron algunas clases que pertenecen a *Java*. El nombre de estas clases comienzan con **J**. Todas estas clases pertenecen a una poderosa librería de *Java* llamada *Swing*, la cual ha sido diseñada para que el desarrollador cree una interfaz de usuario amigable en pocas líneas de código. Los objetos que son instancias de estas clases son agregados a la clase principal **JSERO** creando así la interfaz de nuestro sistema constituida por botones, barras de desplazamiento, paneles, etc.

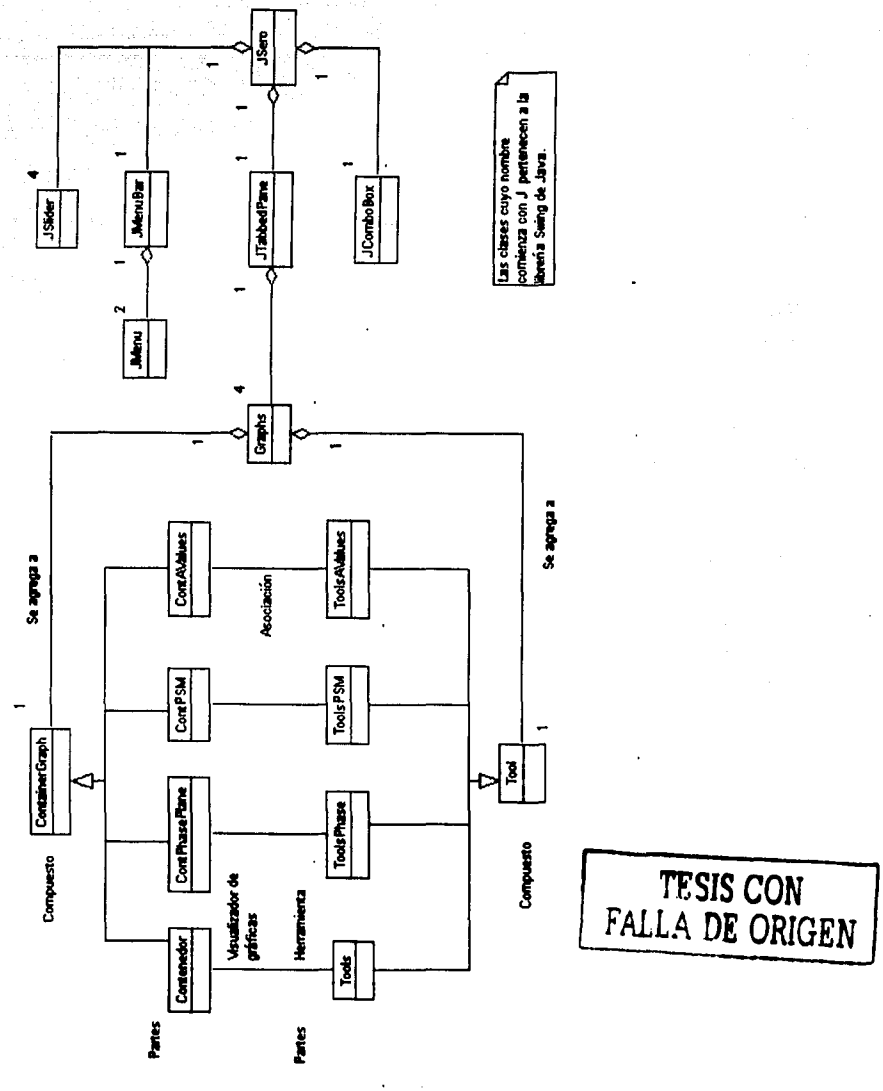


Figura 4.14: Diagrama general de clases del sistema **JSERO**

Capítulo 5

Casos de estudio

En este capítulo se consideran algunos casos de estudio que ponen a prueba al sistema **JSERO**. Se podrá observar que los resultados obtenidos coinciden con los teoremas estudiados en capítulos anteriores.

De la misma forma, se obtienen las soluciones analíticas de la ecuación diferencial de Schrödinger para los potenciales de onda plana, barrera y pozo cuadrado. Estas soluciones deben coincidir con las graficadas por **JSERO**. Se verá que esto resulta así, corroborando la validez de los resultados experimentales.

5.1 La ecuación diferencial de Schrödinger

La ecuación diferencial de Schrödinger es, quizá, la ecuación más famosa y una de las más importantes dentro de la teoría de la mecánica cuántica. No está por demás mencionar que dicha teoría revolucionó la física hace aproximadamente 100 años, cuando en 1900 Planck descubrió el quantum de energía. A este suceso siguieron una serie de descubrimientos entre 1900 y 1926 (considerado este periódo como el más esplendoroso y revolucionario de la ciencia moderna), tales como el quantum de luz de Einstein (1905), el átomo de hidrógeno de Bohr (1913), la mecánica matricial de Heisenberg (1925), la mecánica ondulatoria de Schrödinger (1926), y la interpretación estadística de Born (1926) [8]. Estos descubrimientos constituirían la base de la teoría cuántica. Por otro lado, la mecánica cuántica no solamente ha mejorado profunda-

mente nuestra comprensión de la naturaleza del comportamiento de las partículas del micromundo, sino que también ha sentado las bases para el desarrollo de nuevas tecnologías como los semiconductores, lásers, imágenes de resonancia magnética, entre otros; además de que nos ha llevado a replantear las concepciones de nuestra propia existencia desde un punto de vista filosófico. La importancia de la ecuación diferencial de Schrödinger radica en que esencialmente describe la dinámica de la mecánica cuántica. A continuación se exponen brevemente algunas de las características de la ecuación diferencial de Schrödinger.

5.1.1 Características de la ecuación

En 1926, Heisenberg concibe la idea de representar las cantidades físicas por conjuntos de números complejos dependientes del tiempo, proponiendo lo que se conoce actualmente como la Mecánica Matricial. Por otro lado, independientemente de Heisenberg y casi al mismo tiempo, Erwin Schrödinger, desde el punto de vista de las ecuaciones diferenciales parciales, da una versión distinta del problema. Sin embargo, a pesar de que la forma en que abordan ambos el problema es completamente diferente, llegan al mismo resultado (según lo demostrado por el propio Schrödinger), dado que el sentido físico es el mismo. Se obtuvieron entonces dos formas diferentes de describir la ecuación de onda o la dinámica de la mecánica cuántica.

En lo que concierne a la ecuación de Schrödinger, se trata el problema de la cuantización como un problema de *eigenvalores*, seleccionando adecuadamente una ecuación diferencial y la imposición de condiciones a la frontera satisfactorias.

Sin embargo, a pesar de que ya se tenía la ecuación de onda, todavía no se llegaba a un concenso en cuanto al significado de la solución de la ecuación (la función ondulatoria); de hecho, éste es aún el tema central de muchas controversias.

A pesar de que su sentido exacto no es claro, aún en nuestros días, para los científicos; se tienen tres principales interpretaciones que fueron dadas por Schrödinger, Max Born y Louis de Broglie, respectivamente.

La interpretación de Max Born sugiere que la ecuación de onda debería ser interpretada en términos de probabilidad según Gasiorowicz [6]. Su dependencia de las coordenadas y del tiempo da la probabilidad de encontrar una partícula en un lugar y a un momento dado. Esto se puede traducir matemáticamente como:

$$\int_{-\infty}^{\infty} \|\Psi(x)\|^2 dx = 1$$

Más precisamente, se obtiene la probabilidad de que una partícula se detecte en un lugar y momento, dada la acción que se produce ahí, como puede ser la interacción con los instrumentos de medición como se explica en el libro de Rydnik [4]. La ecuación de Schrödinger se vislumbra entonces como la posible salida al problema que plantea el principio de incertidumbre, acerca de la imposibilidad que se tiene de encontrar la posición y la velocidad de una partícula a la vez y a un instante dados. Esto significa que a pesar de que no se puede encontrar con exactitud los valores de éstas dos al mismo tiempo, sí es posible encontrar la probabilidad de que una partícula se encuentre en una posición en un momento dado y a una velocidad dada.

5.1.2 Derivación de la ecuación de Schrödinger en una dimensión

La ecuación de Schrödinger, en su forma general, contiene tres variables que representan las tres dimensiones espaciales y una más representando al tiempo; además contiene números complejos. Sin embargo, el presente trabajo sólo aborda la ecuación independiente del tiempo y en una sola dimensión. Comencemos entonces la derivación de la ecuación diferencial en una sola dimensión dependiente del tiempo. El lector puede obtener mayor información acerca de la ecuación en el libro de Eisberg [5].

En su teoría no relativista, Schrödinger tomó para E la definición clásica de la energía total:

$$E = \frac{p^2}{2m} + V \tag{5.1}$$

Donde el primer término del lado derecho representa la energía cinética y el segundo la energía potencial.

Además de que también tomó las dos ecuaciones de De Broglie:

$$\lambda = \frac{h}{p} \text{ y } \nu = \frac{E}{h}$$

Donde λ y ν representan la longitud de onda y la frecuencia de las ondas piloto asociadas con la partícula que posee un momento p y una energía total no relativista E. Mientras que h es la llamada constante de Planck, llamada así en honor al físico Max Planck quien fue el primero en descubrirla, escribiendo estas dos ecuaciones en términos de los parámetros:

$$K = 2\pi k \text{ y } \omega = 2\pi\nu$$

donde $k = \frac{1}{\lambda} = \frac{p}{h}$, tenemos:

$$p = \hbar K, E = \hbar\omega \tag{5.2}$$

Combinando las ecuaciones (5.1) y (5.2), obtenemos:

$$\frac{\hbar^2 K^2}{2m} + V(x,t) = \hbar\omega \tag{5.3}$$

Por supuesto, en este capítulo únicamente se plantea la ecuación para una sola partícula y en una dimensión, y no para tres dimensiones, como se hace en la representación general de la fórmula. Ahora bien, para encontrar la forma de la ecuación de Schrödin-ger, tomamos una de la funciones de onda $\Psi_f(x,t)$ para la partícula libre y cons-truimos una ecuación consistente con (5.3), que sólo tenga términos de esta función y sus derivadas. Esta igualdad nos da una ecuación diferencial, que tiene como solución la función de onda para la partícula libre, cuando $V(x,t) = V_0$ es una constante.

Construyamos la ecuación diferencial, partiendo de una combinación de dos funciones de la forma:

$$\Psi_f(x,t) = \cos(Kx - \omega t) + \gamma \sin(Kx - \omega t)$$

donde γ es una constante que se debe determinar.

Calculemos algunas derivadas:

$$\begin{aligned}
\tfrac{d\Psi_f(x,t)}{dx} &= -K\sin(Kx-\omega t) + K\gamma\cos(Kx-\omega t) \\
\frac{d^2\Psi_f(x,t)}{dx^2} &= -K^2\cos(Kx-\omega t) - K^2\gamma\sin(Kx-\omega t) \qquad (5.4) \\
\tfrac{d\Psi_f(x,t)}{dt} &= \omega\sin(Kx-\omega t) - \omega\gamma\cos(Kx-\omega t)
\end{aligned}$$

Al derivar dos veces con respecto a x, se obtiene el factor K^2, y al derivar una vez con respecto a t se obtiene el factor ω. Esto sugiere la ecuación:

$$\alpha\frac{d^2\Psi_f(x,t)}{dx^2} + V_0\Psi_f(x,t) = \beta\frac{d\Psi_f(x,t)}{dt} \qquad (5.5)$$

como posible ecuación diferencial que satisface las condiciones; donde α y β son contantes que se deben determinar.

Sustituyendo las derivadas (5.4) en la ecuación diferencial (5.5), obtenemos:

$$[-\alpha K^2 + V_0 + \beta\omega\gamma]\cos(Kx-\omega t) + [-\alpha K^2\gamma + V_0\gamma - \beta\omega]\sin(Kx-\omega t) = 0$$

Entonces, para que esta ecuación se satisfaga para todos los valores de x y t, se debe tener:

$$-\alpha K^2 + V_0 = -\beta\gamma\omega \qquad (5.6)$$

$$-\alpha K^2 + V_0 = \frac{\beta}{\gamma}\omega \qquad (5.7)$$

Resolviendo las tres ecuaciones (5.3), (5.6) y (5.7) con las tres constantes α, β y γ obtenemos:

$$\gamma = \pm i, \alpha = -\frac{\hbar^2}{2m} \text{ y } \beta = \pm i\hbar$$

Al elegir $\gamma = i$ se obtiene:

$$-\frac{\hbar^2}{2m}\frac{d^2\Psi_f(x,t)}{dx^2} + V_0\Psi_f(x,t) = i\hbar\frac{d\Psi_f(x,t)}{dt}$$

Se ha obtenido la ecuación de Schrödinger para el caso especial, cuando el potencial es contante $V(x,t) = V_0$, partiendo de una función de partícula libre Ψ_f.

Supondremos que, cuando $V(x,t)$ no es constante, la ecuación diferencial que controla la propagación de la función de onda es de esta forma. La ecuación básica de la mecánica cuántica no relativista de Schrödinger, para una partícula sometida a un potencial V, está dada entonces por:

$$i\hbar\frac{d}{dt}\Psi(x,t) = -\frac{\hbar^2}{2m}\frac{d^2}{d^2x}\Psi(x,t) + V(x,t)\Psi(x,t) \tag{5.8}$$

Esta ecuación puede escribirse también de la siguiente forma:

$$ih\frac{d}{dt}\Psi(x,t) = H\Psi(x,t)$$

donde H es tal que $H = -\frac{\hbar^2}{2m}\frac{d^2}{d^2x} + V$; esto es, H es la suma de la energía cinética y la potencial. A H se le conoce como el operador energía o el Hamiltoniano.

Esta ecuación es realmente hermosa; en ella se puede observar la notable diferencia de las ecuaciones de la física clásica y la cuántica, a través de la aparición del número imaginario i en la ecuación.

Es necesario hacer algunas observaciones acerca de la ecuación de Schrödinger. La ecuación contiene un número complejo, mismo que nos obliga a tener como soluciones funciones complejas. La consecuencia inmediata que tiene este resultado (que no es sorprendente sino todo lo contrario: es algo que se esperaba), es que no existe ninguna forma de medir el valor de una función de onda, ya que es imposible medir con un instrumento físico y real el valor de una cantidad compleja; esto es, no podemos atribuir a las funciones de onda una existencia física.

Resolver la ecuación general de Schrödinger es una tarea que puede ser compleja; sin embargo, existe una gran variedad de fenómenos que facilitan su solución. Tal es el caso de los llamados problemas estacionarios: podemos suponer la estacionareidad de la energía de una partícula en el tiempo; esto es, que la energía no varía con el tiempo, lo que significa que si aplicamos la relación de Heisenberg:

$$\Delta E \times \Delta t \geq h$$

entonces, como el tiempo no juega ningún papel en el valor de la energía, se puede establecer sin ningún problema la incertidumbre para el valor de t de $\Delta t = \infty$; entonces la incertidumbre de la energía es $\Delta E = 0$; lo que significa que podríamos conocer la energía de la partícula de forma precisa [4].

5.1.3 La independencia del tiempo

Supongamos que la energía potencial sólo es una función de x; la ecuación de Schrödinger se reduce entonces a:

$$i\hbar\frac{d}{dt}\Psi(x,t) = -\frac{\hbar^2}{2m}\frac{d^2}{d^2x}\Psi(x,t) + V(x)\Psi(x,t) \tag{5.9}$$

Propongamos la solución a esta ecuación de la siguiente forma:

$$\Psi(x,t) = \psi(x)\phi(t) \tag{5.10}$$

Sustituyendo la ecuación (5.10) en (5.9), tenemos:

$$-\frac{h^2}{2m}\frac{d^2}{dx^2}\psi(x)\phi(t) + V(x)\psi(x)\phi(t) = i\hbar\frac{d}{dt}\psi(x)\phi(t)$$

Entonces:

$$-\frac{\hbar}{2m}\phi(t)\frac{d^2\psi(x)}{dx^2} + V(x)\psi(x)\phi(t) = i\hbar\psi(x)\frac{d\phi(t)}{dt}$$

Dividiendo ambos miembros entre $\psi(x)\phi(t)$, obtenemos:

$$\frac{1}{\psi(x)}\left[-\frac{\hbar^2}{2m}\frac{d^2\psi(x)}{dx^2} + V(x)\psi(x)\right] = \frac{i\hbar}{\phi(t)}\frac{d\phi(t)}{dt}$$

Tomando en cuenta que la parte izquierda de la ecuación depende solamente de x y y, en tanto que la derecha sólo de t (las cuales son dos variables independientes la una de la otra), se deduce que ambos miembros deben ser iguales a una cantidad que no dependa ni de x ni de t, de manera que sea cierta la igualdad para todos los valores de las variables. Así, ambos miembros deben ser iguales a la misma constante C de separación; lo que implica que:

$$\frac{1}{\psi(x)}\left[-\frac{h^2}{2m}\frac{d^2\psi(x)}{dx^2} + V(x)\psi(x)\right] = C \tag{5.11}$$

$$\frac{i\hbar}{\phi(t)}\frac{d\phi(t)}{dt} = C \tag{5.12}$$

Debido a que esta última ecuación diferencial es de primer orden, su solución es una exponencial. Es fácil demostrar que su solución es:

$$\phi(t) = e^{-\frac{iCt}{\hbar}} \tag{5.13}$$

La función ϕ es una función compleja oscilatoria del tiempo:

$$\phi(t) = e^{-\frac{iCt}{\hbar}} = \cos(Ct/\hbar) - i\sin(Ct/\hbar)$$

con la frecuencia ν dada por $2\pi\nu = \frac{C}{\hbar}$. O sea, $\nu = \frac{C}{2\pi\hbar} = \frac{C}{h}$. Entonces, según lo visto anteriormente, $\nu = \frac{E}{h}$, donde E es la energía total de la partícula; ya que $\phi(t)$ contiene la dependencia respecto a t de Ψ. Por lo tanto, C debe ser igual a la energía total E; de donde se deduce que la ecuación (5.11) tiene la forma:

$$-\frac{h^2}{2m}\frac{d^2\psi(x)}{dx^2} + V(x)\psi(x) = E\psi(x) \tag{5.14}$$

Mientras que la ecuación (5.13) tiene la forma:

$$\phi(t) = e^{-\frac{iEt}{\hbar}} \tag{5.15}$$

Por lo que la solución a la ecuación de Schrödinger, que tiene la forma (5.9), tiene como solución general:

$$\Psi(x,t) = \psi(x)e^{-\frac{iEt}{\hbar}} \tag{5.16}$$

Donde ψ se conoce como la ecuación de Schrödinger para la onda plana o función propia de la ecuación diferencial de Schrödinger. Esta función es entonces la solución de la ecuación diferencial:

$$-\frac{h^2}{2m}\frac{d^2\psi(x)}{dx^2} + (V(x) - E)\psi(x) = 0 \tag{5.17}$$

Esta ecuación es precisamente la que resuelve numéricamente el sistema **JSERO** y cuyas soluciones pueden ser estudiadas cuantitativa y cualitativamente a través del sistema. Ella representa a la ecuación de Schrödinger independiente del tiempo y en una sola dimensión.

Parte importante de la elaboración de esta tesis es la aplicación que pueda tener el sistema. Este último capítulo está dedicado a la prueba del sistema para varios potenciales, lo que nos permitirá ejemplificar su uso y correcto funcionamiento.

Se visualizan además las diferentes funciones solución de la ecuación de Schrödinger. graficadas en los distintos espacios (espacio solución, espacio fase, mapa de estabilidad potencial periódico y valores asintóticos)

5.2 La ecuación diferencial de onda plana

Recordemos que la ecuación diferencial de Schrödinger se caracteriza por tener la forma de Sturm-Liouville; por lo que las propiedades de las soluciones de la ecuación diferencial que tiene la forma de Sturm-Liouville, mencionadas en el segundo capítulo de la presente tesis, se aplican a las soluciones de la ecuación de Schrödinger. A continuación, se obtiene su solución analítica para un potencial constante, para ejemplificar los resultados anteriores.

5.2.1 Solución analítica de las funciones propias de la ecuación de Schrödinger para la onda plana

La ecuación de onda plana se caracteriza por tener un potencial constante $V = 0$, por lo que la ecuación (5.17) se convierte en:

$$\frac{h^2}{2m}\frac{\partial^2}{\partial^2 x}\psi + E\psi = 0$$

Ahora bien, si hacemos variar m de tal forma que obtengamos $\frac{h^2}{2m} = 1$, se obtiene:

$$\frac{\partial^2}{\partial^2 x}\psi + E\psi = 0 \tag{5.18}$$

Dado que se trata de una ecuación diferencial lineal homogénea de segundo orden, podemos construir un sistema de dos ecuaciones diferenciales de primer orden, a partir de esta ecuación diferencial de segundo orden. Hagamos un cambio de variable de la siguiente forma:

$$Z_1 = \psi$$

$$Z_2 = \dot{\psi} = \dot{Z}_1$$

La ecuación (5.18) se transforma entonces en:

$$\dot{Z}_2 + EZ_1 = 0$$

De donde se obtiene el siguiente sistema de ecuaciones:

$$\dot{Z}_1 = Z_2$$

$$\dot{Z}_2 = -EZ_1$$

Pongamos este sistema de ecuaciones en su forma matricial:

$$\frac{d}{dx}\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -E & 0 \end{bmatrix}\begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} \tag{5.19}$$

Se tiene entonces la ecuación diferencial matricial:

$$\frac{d\mathbf{Z}(x)}{dx} = M\mathbf{Z}(x) \tag{5.20}$$

donde

$$M = \begin{bmatrix} 0 & 1 \\ -E & 0 \end{bmatrix} \tag{5.21}$$

Esta ecuación es la misma que vimos en la sección (2.3.4); por lo que, si $E > 0$, las soluciones son:

$$Z(x) = \begin{bmatrix} \cos\sqrt{E}(x - x_0) & \frac{\sin\sqrt{E}(x-x_0)}{\sqrt{E}} \\ -\sqrt{E}\sin\sqrt{E}(x - x_0) & \cos\sqrt{E}(x - x_0) \end{bmatrix} \tag{5.22}$$

Mientras que si $E < 0$, entonces, según lo visto en la sección (2.3.5), las soluciones son:

$$Z(x) = \begin{bmatrix} \cosh(\sqrt{-E}(x - x_0)) & \frac{\sinh(\sqrt{-E}(x-x_0))}{\sqrt{-E}} \\ \sqrt{-E}\sinh(\sqrt{-E}(x - x_0)) & \cosh(\sqrt{-E}(x - x_0)) \end{bmatrix} \tag{5.23}$$

5.2.2 Solución computacional

A la columna del lado izquierdo de la matriz (5.22) se le conoce como solución par y a la del lado derecho como solución impar. Con base en los resultados anteriores, podemos corroborar el teorema 2.1 visto anteriormente, el cual indica que dadas dos soluciones linelmente independientes de una ecuación diferencial de la forma (2.1), estas dos soluciones deben anularse entre dos ceros sucesivos; lo cual se puede ver claramente en la figura 5.1, donde se obtienen dos funciones, una de ellas dada por un seno y la otra por un coseno, las cuales es bien sabido alternan sus ceros. Por otra parte, estas soluciones corresponden también a las que se encuentran en la página web Hyper Physics [12]; de hecho, en esta página el lector puede encontrar más información acerca de las funciones de onda para la onda plana y el oscilador armónico de Dirac e información general de la mecánica cuántica.

Las soluciones en el espacio fase se dan en la figura 5.2 siguiente, donde -como se esperaba según lo visto en la sección (2.3.4)-, aparecen una serie de elipses centradas con respecto al origen. Estas dos elipses corresponden a las soluciones par e impar de la ecuación diferencial de Schrödinger para la onda plana.

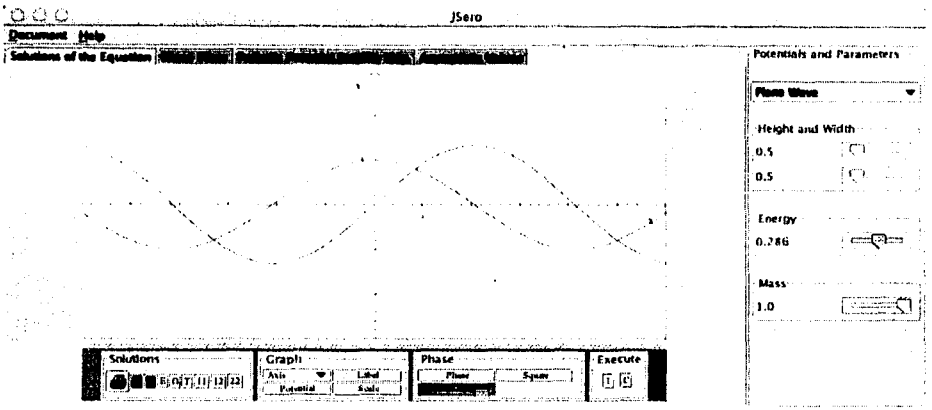


Figura 5.1: Soluciones de la ecuación de onda plana

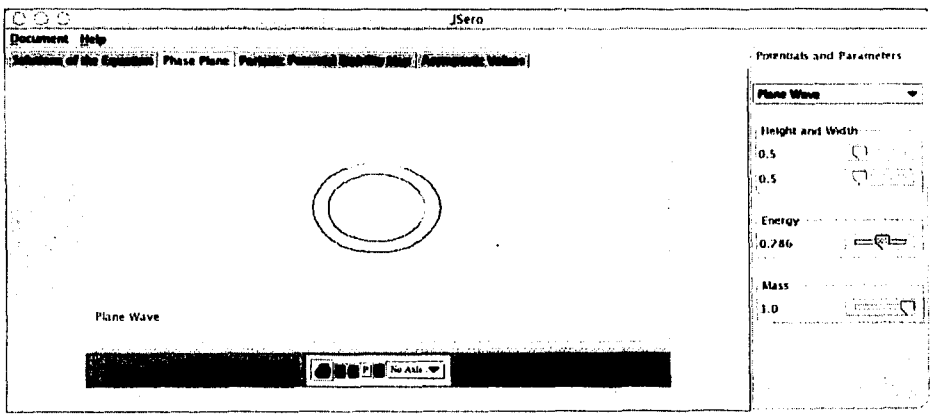


Figura 5.2: Soluciones 5.22 graficadas en el espacio fase

5.3 El potencial barrera

Como su nombre lo indica, se trata de un potencial que tiene la forma de una barrera, y su uso se da muy frecuentemente en el área de la física y la electrónica. La elaboración

del transistor pudo ser posible gracias al conocimiento que se desprendió del estudio de partículas como el electrón sometidas a este potencial.

5.3.1 Solución analítica

La ecuación para el potencial barrera está dada por:

$$\frac{d^2\Psi(x)}{dx^2} - (f(x) - E)\Psi(x) = 0 \tag{5.24}$$

Donde:

$$f(x) = \begin{cases} 0, & \text{para } x < a \text{ y } x > b \\ \alpha, & \text{para } a < x < b \end{cases} \tag{5.25}$$

Y donde $\alpha > 0$. Evidentemente la ecuación diferencial tiene que ser resuelta dependiendo de los valores del potencial. Dividamos el eje de las x en tres regiones: $x < -a$, $-a < x < a$ y $x > a$. Para la primera y tercera región la ecuación diferencial se reduce a:

$$\frac{d^2\Psi(x)}{dx^2} + E\Psi(x) = 0 \tag{5.26}$$

Se trata de la ecuación de una partícula libre de energía E. Como se pudo apreciar en la sección anterior, las soluciones dependen de los valores de E. Si los valores de E toman los valores típicos positivos, las soluciones son ondas viajeras; es decir, se esperan funciones de tipo seno y coseno.

En la segunda región, la ecuación toma la forma:

$$\frac{d^2\Psi(x)}{dx^2} + (E - \alpha)\Psi(x) = 0 \tag{5.27}$$

Otros subcasos se presentan. Si $E - \alpha > 0$, entonces la solución de la ecuación es:

$$\mathbf{Z}(x) = \begin{bmatrix} \cos\sqrt{E-\alpha}(x-x_0) & \frac{\sin\sqrt{E-\alpha}(x-x_0)}{\sqrt{E-\alpha}} \\ -\sqrt{E-\alpha}\sin\sqrt{E-\alpha}(x-x_0) & \cos\sqrt{E-\alpha}(x-x_0) \end{bmatrix} \tag{5.28}$$

En este caso se tienen ondas viajeras en las tres regiones en las que se divide el eje de las x. Entonces, lo que se tiene es una deformación en cuanto a las gráficas, debido a que tanto la frecuencia como la amplitud cambian en el intervalo. De hecho, la frecuencia disminuye en las dos soluciones, lo que resulta en una apariencia de contracción de las gráficas en la segunda sección, con respecto a las otras dos.

Pero si $E - \alpha < 0$, se tiene una solución de la forma:

$$\mathbf{Z}(x) = \begin{bmatrix} \cosh(\sqrt{\alpha-E}(x-x_0)) & \frac{\sinh(\sqrt{\alpha-E}(x-x_0))}{\sqrt{\alpha-E}} \\ \sqrt{\alpha-E}\sinh(\sqrt{\alpha-E}(x-x_0)) & \cosh(\sqrt{\alpha-E}(x-x_0)) \end{bmatrix} \tag{5.29}$$

Cabe señalar en este punto que estas soluciones coinciden perfectamente con lo que se esperaba del teorema (2.2) de oscilación y comparación, del que se deduce que como $E - \alpha < 0$, en el intervalo $-a < x < a$, entonces las soluciones de la ecuación diferencial no tienen más de un cero en dicho intervalo. Esto se puede ver claramente del resultado anterior, ya que el seno hiperbólico sólo podría cortar una vez el eje de las x y el coseno hiperbólico ninguna.

5.3.2 Solución computacional

La gráfica de las soluciones de la ecuación de Schrödinger para un potencial barrera cuando $E - \alpha < 0$, tiene la forma que se plantea en la figura 5.3.

La función de onda, al entrar en el potencial barrera, sufre una deformación; esto es, la solución par toma la forma del coseno hiperbólico, mientras que la solución impar toma la forma de un seno hiperbólico.

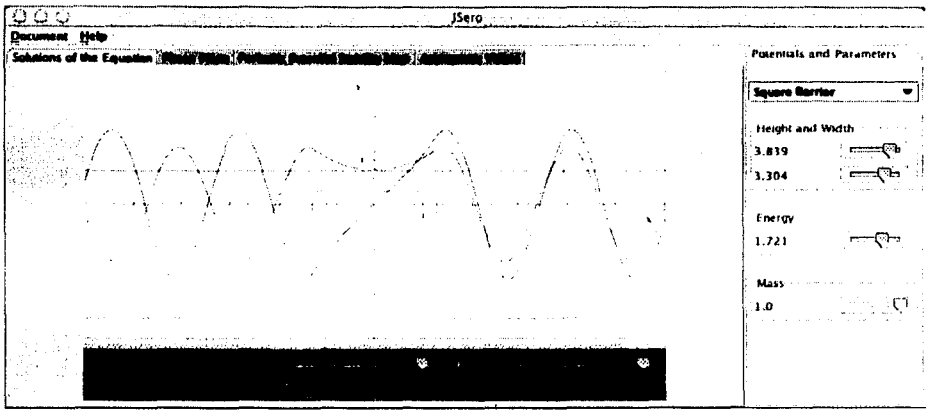


Figura 5.3: Comportamiento de las soluciones al entrar en el potencial barrera

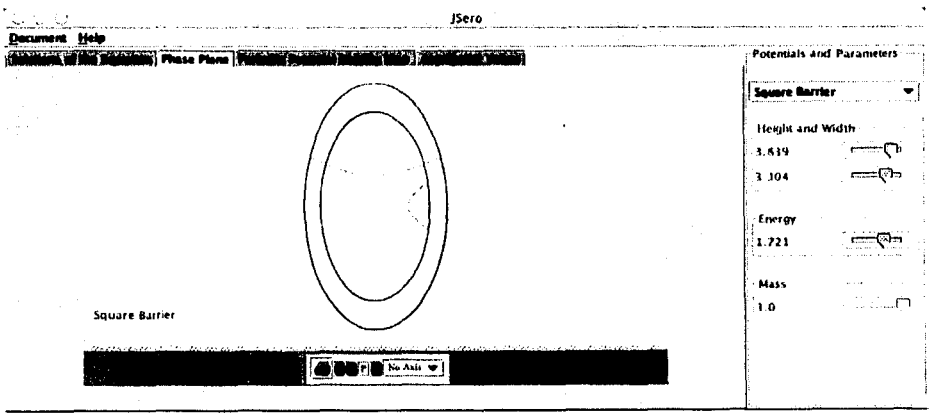


Figura 5.4: Graficación en el espacio fase de las soluciones de la ecuación de Schrödinger sometida a un potencial barrera

En la figura 5.4 se muestran las soluciones par e impar de la ecuación en el espacio fase. Se observa cómo las dos soluciones forman elipses centradas con

respecto al origen. Esto corresponde a la parte que se encuentra fuera del potencial; sin embargo, hay una parte en el que ambas salen de la elipse y después la retoman. Esta parte es precisamente donde el potencial crece y se vuelve positivo.

5.4 El potencial pozo cuadrado

Éste es el potencial más simple que puede confinar una partícula en una región finita del espacio. Frecuentemente se encuentra este potencial en la mecánica cuántica; se usa para representar una situación en el que una partícula se mueve en una región restringida del espacio bajo la influencia de las fuerzas que la confinan a esa región [5].

5.4.1 Solución analítica

La ecuación correspondiente al pozo cuadrado es similar a la del potencial barrera:

$$\frac{d^2\Psi(x)}{dx^2} - (f(x) - E)\Psi(x) = 0 \tag{5.30}$$

donde:

$$f(x) = \begin{cases} 0, & \text{para } x < a \text{ y } x > b \\ -\alpha, & \text{para } a < x < b \end{cases} \tag{5.31}$$

Y donde $\alpha > 0$. De la misma manera que en la ecuación 5.24, existen varios casos que se tienen que tomar para resolver la ecuación diferencial de Schrödinger cuando tiene un potencial pozo cuadrado; de hecho se tienen las mismas regiones de estudio.

Primeramente, tomemos la primera y tercera región. De nuevo la ecuación toma la forma de una partícula libre y sus soluciones son trigonométricas; pero si tomamos la segunda región entonces la ecuación es de la forma:

$$\frac{d^2\Psi(x)}{dx^2} + (E+\alpha)\Psi(x) = 0 \tag{5.32}$$

De la misma manera que antes, se tienen dos subcasos: si $E+\alpha > 0$, entonces la solución es:

$$\mathbf{Z}(x) = \begin{bmatrix} \cos\sqrt{E+\alpha}(x-x_0) & \frac{\sin\sqrt{E+\alpha}(x-x_0)}{\sqrt{E+\alpha}} \\ -\sqrt{E+\alpha}\sin\sqrt{E+\alpha}(x-x_0) & \cos\sqrt{E+\alpha}(x-x_0) \end{bmatrix} . \tag{5.33}$$

De la ecuación anterior se esperaría que las funciones solución tengan una frecuencia mayor a la que tenían antes de entrar al potencial o a la que tendrían al salir de él.

Pero si $E+\alpha < 0$, entonces de la misma forma que anteriormente tendríamos una solución con senos hiperbólicos y cosenos hiperbólicos:

$$\mathbf{Z}(x) = \begin{bmatrix} \cosh(\sqrt{-(\alpha+E)}(x-x_0)) & \frac{\sinh(\sqrt{-(\alpha+E)}(x-x_0))}{\sqrt{-(\alpha+E)}} \\ \sqrt{-(\alpha+E)}\sinh(\sqrt{-(\alpha+E)}(x-x_0)) & \cosh(\sqrt{-(\alpha+E)}(x-x_0)) \end{bmatrix} \tag{5.34}$$

5.4.2 Solución computacional

Éste es un potencial que es muy parecido al anterior, sólo que simétricamente opuesto con respecto al eje horizontal. En la figura 5.5 se muestra el comportamiento de las funciones de onda en el caso en el que $E+\alpha > 0$. Se puede apreciar que cuando la función entra en el potencial, estas funciones sufren una deformación; dicha deformación corresponde a una frecuencia mayor en el potencial a la que tienen antes o después del potencial. En [5] se puede encontrar que las soluciones son muy parecidas a las que se grafican en la figura 5.5. Mientras que para otros valores del potencial y ciertos valores de energía (niveles de energía) las gráficas de **JSERO** son similares a las que se encuentran en [12]. Por otro lado, en la figura 5.6 se ven las mismas funciones de onda representadas en el espacio fase. El aumento en la frecuencia provoca que las elipses se cierren más y cambien de forma.

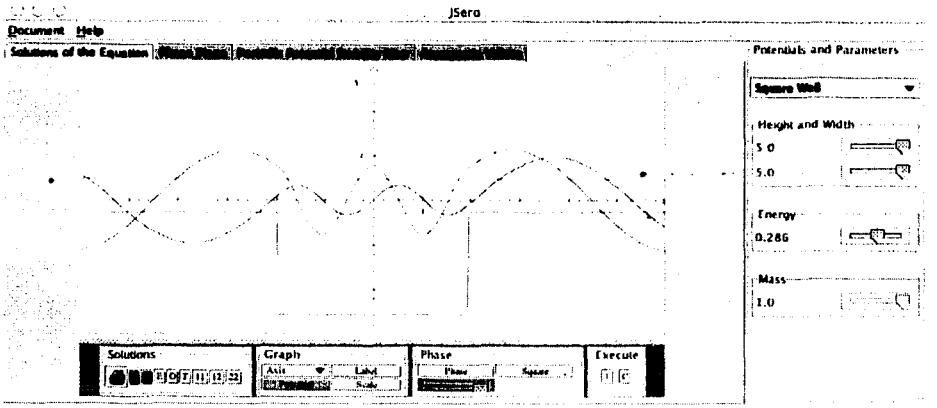


Figura 5.5: Gráfica de las soluciones de la ecuación de Schrödinger para un potencial pozo cuadrado

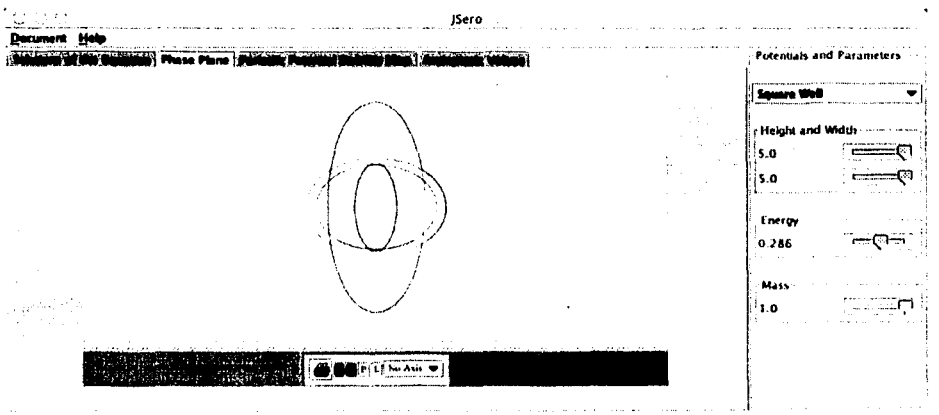


Figura 5.6: Gráfica en el espacio fase de las soluciones de la ecuación de Schrödinger para un potencial pozo cuadrado

5.5 Otros potenciales

5.5.1 El potencial lineal

La ecuación de Schrödinger para un potencial lineal tiene la forma:

$$\frac{d^2\Psi(x)}{dx^2} - (\alpha x - E)\Psi(x) = 0 \tag{5.35}$$

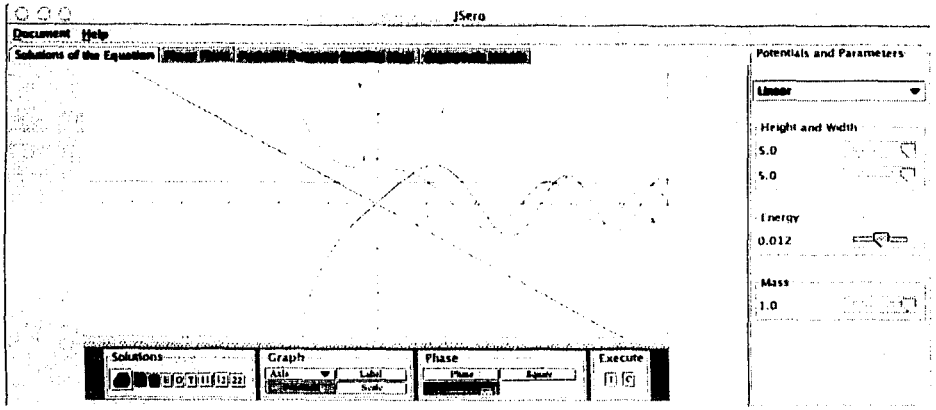


Figura 5.7: Gráfica de las soluciones de la ecuación diferencial de Schrödinger para un potencial lineal

En la figura 5.7 se muestra el resultado de una integración numérica de la ecuación. Se escogieron las dos soluciones básicas; la primera de ellas similar a la función seno (valor inicial 0, derivada 1). La segunda similar al coseno (valor inicial 1, derivada 0). Nótese cómo las funciones trigonométricas de un lado del eje de las ordenadas se conectan suavemente con las funciones hiperbólicas que se encuentran del otro lado de dicho eje.

La figura 5.8 muestra el mismo resultado en el espacio fase; probablemente se necesite aquí una explicación de lo que se ve en este espacio. Como se dijo anteriormente, las funciones de onda tienen un comportamiento trigonométrico del lado derecho de la singularidad, mientras que del lado izquiero se tiene un comportamiento casi exponencial. Esto hace que en el espacio fase se obtenga una curva con movimiento circular en el centro y después se dispara hacia fuera.

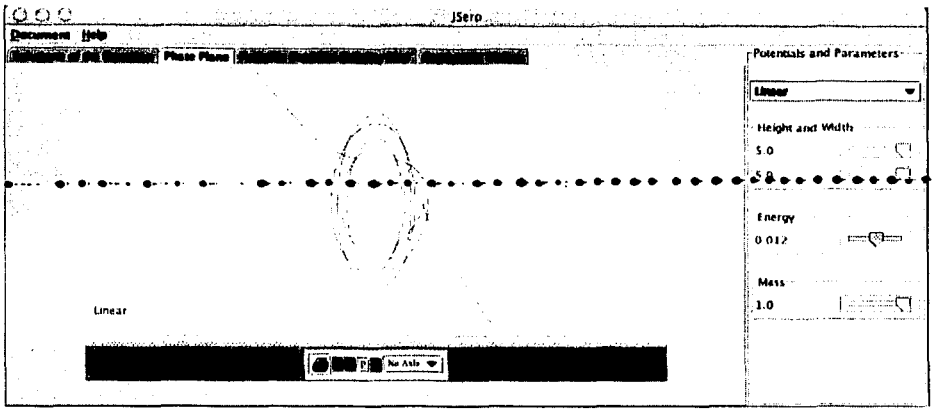


Figura 5.8: Gráfica de las soluciones para un potencial lineal en el espacio fase

5.5.2 Oscilador armónico de Dirac

El sistema de ecuaciones diferenciales de primer orden correspondiente al oscilador armónico cuyo potencial es $\frac{1}{6}x^2$ y masa m tiene la forma:

$$\frac{d}{dx}\begin{bmatrix}\phi(x)\\ \psi(x)\end{bmatrix} = \begin{bmatrix}0 & m-E+\frac{1}{6}x^2\\ m+E-\frac{1}{6}x^2 & 0\end{bmatrix}\begin{bmatrix}\phi(x)\\ \psi(x)\end{bmatrix} \tag{5.36}$$

Si la masa domina, las soluciones son funciones hiperbólicas, si no son completamente trigonométricas cuando la masa es cero; esto se puede constatar en la figura 5.9.

Para una masa pequeña, la región que se encuentra dentro de la barrera potencial da soluciones oscilatorias, de una forma similar a los polinomios de Hermite encontrados en la ecuación de Schrödinger. En el campo de la masa las soluciones son exponenciales, como se esperaba; sin embargo, hay una nueva característica en la ecuación de Dirac: en la región que se encuentra fuera, las soluciones son trigonométricas otra vez. La figura 5.10 muestra las funciones de onda para el oscilador armónico de Dirac en el espacio fase. Una explicación amplia de lo anterior se puede encontrar en el documento de McIntosh [7].

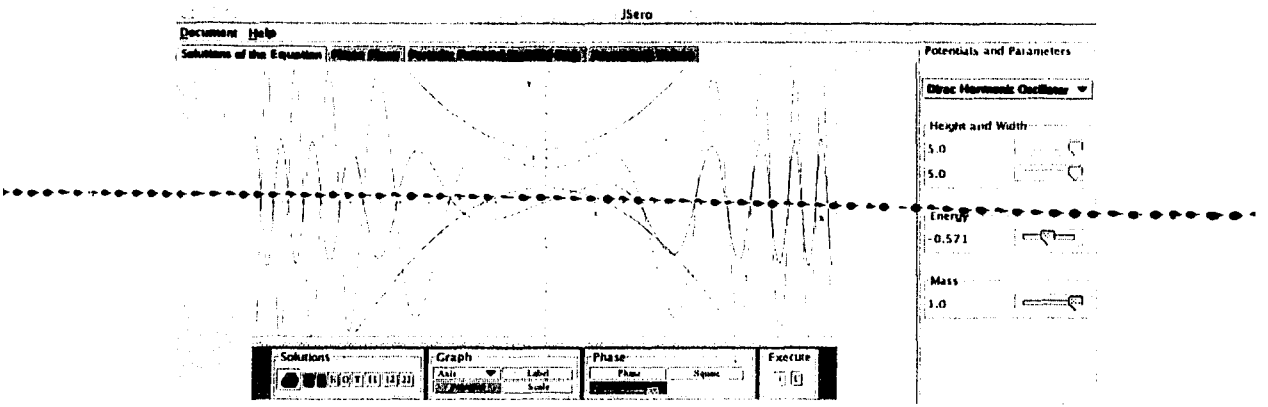


Figura 5.9: Soluciones de la ecuación diferencial de Schrödinger para el potencial oscilador armónico de Dirac

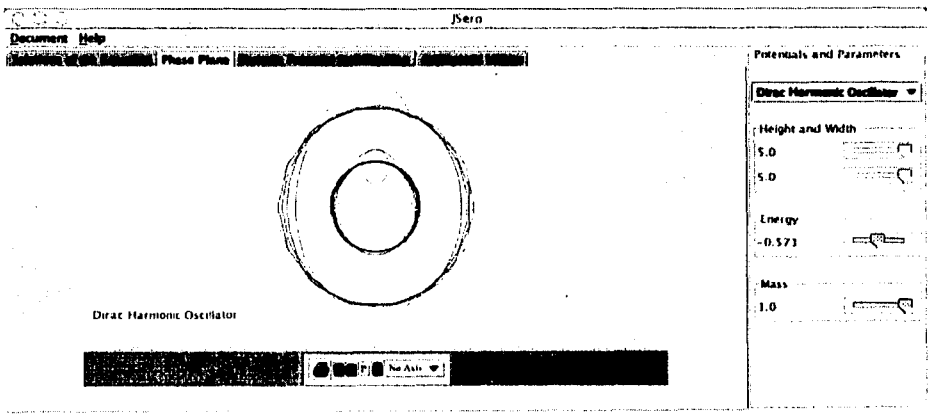


Figura 5.10: Soluciones en el espacio fase

Por otro lado, los valores asintóticos pueden ser vistos en la figura 5.11. No se dará una explicación de este espacio porque se encuentra fuera de los objetivos de esta tesis; si el lector desea saber más acerca de él puede referirse a [7].

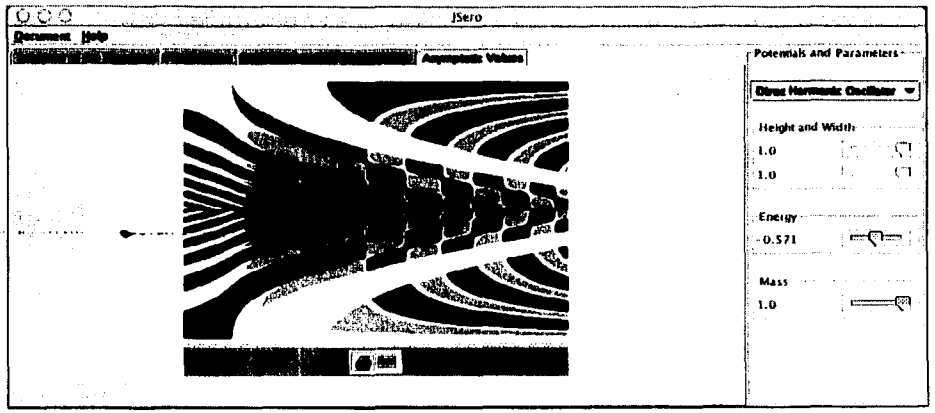


Figura 5.11: Valores asintóticos de la ecuación de Schrödinger para el potencial oscilador armónico de Dirac

5.5.3 El potencial sinusoidal

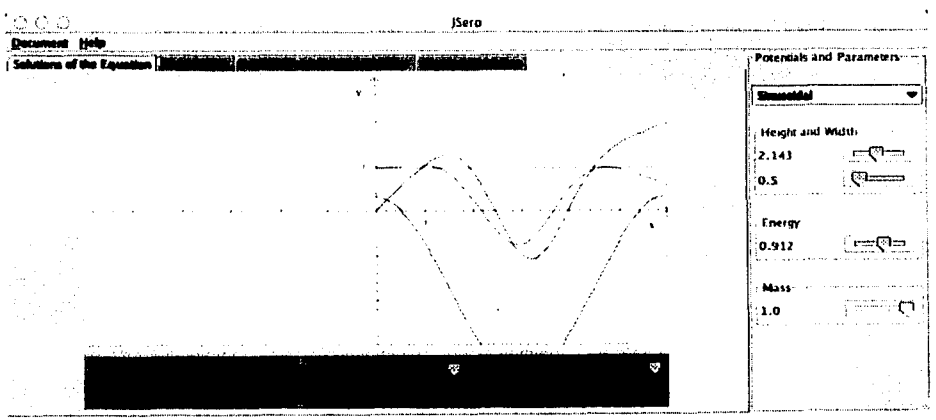


Figura 5.12: Gráfica de las soluciones para un potencial sinusoidal

La forma matricial de la ecuación de Schrödinger correspondiente a este po-

tencial es:

$$\frac{d\mathbf{Z}(x)}{dx} = \begin{bmatrix} 0 & 1 \\ h\cos x - E & 0 \end{bmatrix} \mathbf{Z}(x_0) \tag{5.37}$$

Este potencial tiene la forma de una función trinométrica, como se puede apreciar en la figura 5.12; mientras que sus soluciones son también de tipo trigonométrico.

El hecho de que las funciones solución sean casi trigonométricas se ve reflejado en el espacio fase (figura 5.13) al visualizar en éste una serie de funciones que tienen un comportamiento similar al de unas elipses.

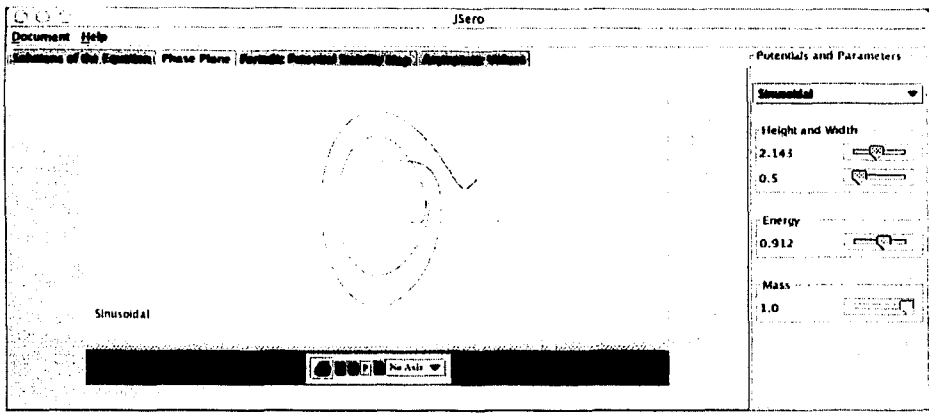


Figura 5.13: Gráfica de las soluciones para un potencial sinusoidal en el espacio fase

Por otra parte, en la figura 5.14 se muestra el mapa de estabilidad potencial periódico, cuyo análisis no será tratado tampoco en esta tesis; baste decir que este mapa también ofrece información cualitativa de las soluciones de la ecuación diferencial de Schrödinger para un potencial sinusoidal. Sin embargo, el lector puede encontrar una explicación más detallada acerca del tratamiento de la ecuación para los potenciales vistos en esta sección en [7], así como información acerca de la interpretación del mapa de estabilidad potencial periódica y de los valores asintóticos.

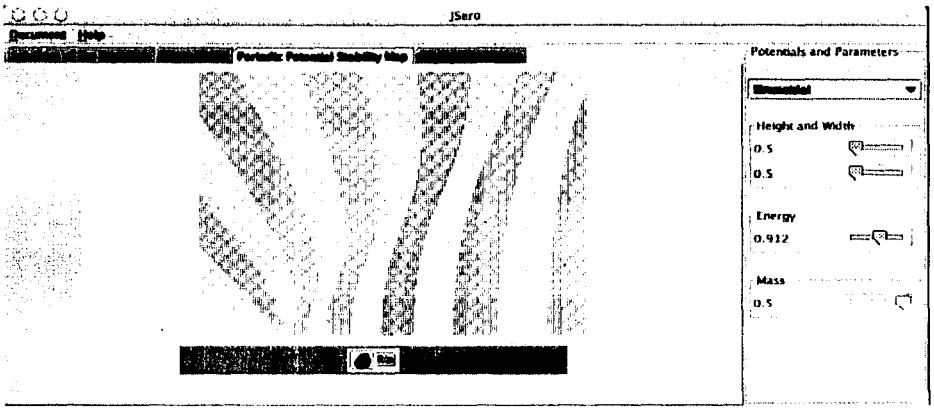


Figura 5.14: Mapa de estabilidad potencial periódica

5.5.4 Tabla de otros potenciales

El resto de los potenciales que el sistema **JSERO** es capaz de utilizar se muestra en la tabla 5.1. Si el lector lo desea puede encontrar más información acerca de estos potenciales en [5] y [12].

Finalmente, es importante destacar que los resultados que el sistema **JSERO** ofrece son similares a los que se encuentran en [5] y [12], para los diferentes niveles de energía permitidos para los potenciales pozo cuadrado, onda plana y oscilador armónico de Dirac. Esto permite al lector verificar la validez de los resultados presentados por **JSERO**.

Matriz de coeficientes	Nombre
$\begin{bmatrix} 0 & 1 \\ \begin{cases} h-E, & \text{para } x \geq 0 \\ -E, & \text{c.o.c} \end{cases} & 0 \end{bmatrix}$	Escalón
$\begin{bmatrix} 0 & 1 \\ x^2-E & 0 \end{bmatrix}$	Oscilador armónico de Schrödinger
$\begin{bmatrix} 0 & 1 \\ x^4-E & 0 \end{bmatrix}$	Oscilador cuártico de Schrödinger
$\begin{bmatrix} -m\sin(2x(\frac{x^2}{6}-E)) & m\cos(2x(\frac{x^2}{6}-E)) \\ m\cos(2x(\frac{x^2}{6}-E)) & m\sin(2x(\frac{x^2}{6}-E)) \end{bmatrix}$	Oscilador de Dirac factorizado
$\begin{bmatrix} 0 & m-he^{-\frac{x^2}{w}}+E \\ m+he^{-\frac{x^2}{w}}-E & 0 \end{bmatrix}$	Potencial Gaussiano
$\begin{bmatrix} 0 & 1 \\ \begin{cases} \frac{-1}{x+7.5}-\frac{E}{2}, & \text{para } x > -7.4 \\ -\frac{E}{2}, & \text{c.o.c} \end{cases} & 0 \end{bmatrix}$	Coulomb

Tabla 5.1: Lista de potenciales que no fueron tratados en la tesis, pero que también pueden ser resueltos por el sistema **JSERO**

Conclusión

La abundante producción de tecnología en los útimos años, para cumplir con los cada vez mayores requerimientos del mercado y su constante evolución, han obligado a los desarrolladores de software a actualizarse continuamente; esto con el fin de valerse de las cada vez más poderosas herramientas de esta nueva tecnología. Pero esto conlleva un trabajo cada vez más arduo para no quedarse rezagado. El presente trabajo ha hecho uso de dos de estas heramientas, que son UML y *Java* para lograr el diseño e implementación respectivamente del sistema **JSERO**.

UML ha permitido obtener un metamodelo, a partir del cual se puede obtener de manera gráfica, información muy completa de la estructura estática y dinámica de **JSERO**. La realización de los diagramas en UML tiene varias ventajas. Una de estas ventajas es que el sistema no está sujeto a ningún lenguaje de programación. En el presente trabajo se ha hecho la implementación en *Java*, pero es posible hacerla en cualquier otro lenguaje de programación orientado a objetos, como *C++*, *Small Talk*, etc. Si se necesita escribir el sistema en algún otro lenguaje de POO, únicamente se tiene que recurrir a los diagramas, eliminando así la tediosa necesidad de tener que revisar el código para poder comprender su funcionamiento. También se ha podido ver a lo largo del trabajo que pasar los diagramas en UML a código en *Java* se efectúa de una manera prácticamente natural. De hecho, sólo se hace un mapeo de los artefactos en UML a las palabras reservadas correspondientes a código en *Java*. Es importante también señalar que UML ayuda al desarrollador a efectuar modificaciones en el sistema a alto nivel, controlando completamente las repercusiones que esto puede ocasionar. Esto significa que, aun sin haber efectuado cambio alguno en el código, se sabe perfectamente lo que sucederá.

Por otro lado, el uso del lenguaje de programación *Java* ha dotado al sistema de valiosas propiedades. El sistema tiene ahora la extraordinaria característica de poderse ejecutar en una notable cantidad de plataformas, como *Mac OS X*, *Windows*,

Linux, etc. Esto permite que una mayor cantidad de usuarios pueda hacer uso de él. Por otra parte, el uso de las potentes librerías de *Java* ha permitido realizar una interfaz gráfica muy amigable para el usuario, y ha ayudado a brindar al software excelentes funcionalidades como las de impresión, carga de gráficas excesivamente pesadas, la graficación de las funciones solución, entre otras. *Java* también le ha dado la posibilidad al sistema de poderse ejecutar desde la red en forma de *applet* o simplemente como una aplicación en forma de *standalone*.

Asimismo, se ha logrado la construcción de un sistema que permite no solo tener un análisis cuantitativo de las funciones propias de una de las ecuaciones más importantes de la física, como lo es la ecuación de Schrödinger; sino que también permite que se obtenga un análisis cualitativo de dichas funciones, a través de su graficación en el espacio fase, y la graficación del mapa de estabilidad potencial y de los valores asintóticos para algunos potenciales.

Finalmente, es importante señalar que resta aún trabajo por hacer; a continuación se mencionan los posibles trabajos que se podrían continuar a partir del que ya se ha hecho en esta tesis:

- Realizar un analizador sintáctico. Esto con el fin de escribir la ecuación que se desea resolver; es decir, que no se tenga que escoger de entre una serie de ecuaciones ya definidas. Se deberían aplicar los diagramas de UML en la elaboración de dicho sistema.
- Pasar el sistema SECO, desarrollado en *C-Objetivo* por el doctor Harold V. McIntosh para resolver EDSO complejas, a UML; e implementarlo en un lenguaje de POO como *Java*.

Las anteriores sugerencias ayudarían a integrar los diagramas propuestos en el presente trabajo con los nuevos diagramas, para obtener así un sistema general que permita visualizar las soluciones de las EDSO reales o complejas, no solo para ciertos potenciales, sino para los que el usuario desee ingresar.

Lista de figuras

Bibliografía

[1] Larman, Craig, *UML y Patrones Introducción al análisis y diseño orientado a objetos*, Prentice Hall, México, 1999.

[2] Booch, G., Jacobson, I. y Rumbaugh, J. *The Unified Modeling Language Reference Manual*, Addison-Wesley, USA, 1999.

[3] *Compañía de software Rational* http://www.rational.com, 2001.

[4] Rydnik, Vitali, *Qué es la mecánica cuántica*, Ediciones Quinto Sol, México, 1990.

[5] Eisberg, Robert M., *Fundamentals of modern physics*, Wiley, USA, 1961.

[6] Gasiorowicz, Stephen, *Quantum Physics*, Wiley, USA, 1995.

[7] McIntosh, Harold V., *Complex Variable Theory*, Documento, México, 1999.

[8] Chapa, Sergio V., *Ecuaciones diferenciales y computación científica*, http://delta.cs.cinvestav.mx/~schapa/proyectos/ecdif/Main.html, Documento, 2001.

[9] Chapa, Sergio V., *Ecuaciones diferenciales de orden n y sistema de ecuaciones diferenciales*, http://delta.cs.cinvestav.mx/~schapa/proyectos/ecdif/Main.html, Documento, 2001.

[10] Eckel, Bruce, *Thinking in Java*, Prentice Hall, USA, 2000.

[11] *Página oficial de Java*, http://java.sun.com, 2001.

[12] *Hyper Physics*, http://hyperphysics.phy-astr.gsu.edu/hbase/hframe.html, Georgia State University, 2001.