Universidad de las Ciencias Informáticas

Facultad 6

Título: Algoritmos para la estimación de parámetros en sistemas de ecuaciones diferenciales.

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor

Roberto Maldonado Bosque.

Tutores

Msc. Noel Moreno Lemus.

Msc. Gilberto Arias Naranjo.

Ciudad de la Habana

Julio del 2008

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas de los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Roberto Maldonado Bosque

_____

Firma del Autor

Msc. Noel Moreno Lemus

_____

Firma del Tutor

Msc. Gilberto Arias Naranjo

_____

Firma del Tutor

El futuro tiene muchos nombres.

Para los débiles es lo inalcanzable.

Para los temerosos, lo desconocido.

Para los valientes es la oportunidad.

Víctor Hugo.

AGRADECIMIENTOS

Agradezco a mi madre por creer en mí, a mis familiares por todo su apoyo, a mis tutores por los conocimientos que me han enseñado, a las personas que han colaborado con mi formación como profesional y a las que han ayudado en la creación de este trabajo.

Muchas gracias...

DEDICATORIA

A mi madre por todo el apoyo y amor que me ha dado…

A mi abuela por su presencia y sus consejos…

A mi hermana por su cariño y sinceridad…

A mis tíos por ser mis guías.

A mis familiares, que apoyaron cada uno de mis pasos y me guiaron siempre por el sendero correcto…

A aquellos que aunque no están presenten les hubiese gustado disfrutar como yo de este inolvidable momento…

Roberto Maldonado Bosque.

RESUMEN

Para el desarrollo de la Biología de Sistemas se crean algoritmos y aplicaciones que ayudan a la modelación, simulación y análisis de dichos sistemas. Entre los objetivos más relevantes de esa disciplina se encuentra predecir las condiciones y los valores para los que se obtendrían los mejores resultados en un problema dado. Existen diferentes aplicaciones para la estimación de esos parámetros, pero la principal limitación es que ninguna de ellas se puede integrar directamente a la plataforma BioSyS. Esta investigación surge en el marco de trabajo del Proyecto “BioSyS”, que se desarrolla conjuntamente por el Centro de Inmunología Molecular (CIM) y el Grupo de Bioinformática de la Facultad 6 de la Universidad de las Ciencias Informáticas (UCI), para el cual se han implementado un conjunto de módulos que trabajan independientes. Dentro de sus objetivos se encuentra desarrollar algoritmos capaces de realizar la estimación de parámetros en un sistema de ecuaciones diferenciales a partir de un conjunto de datos de laboratorio. En el documento se presenta el prototipo funcional de una aplicación diseñada para realizar la estimación de parámetros en sistemas de ecuaciones diferenciales. En el se encuentran implementados los algoritmos para dichas estimaciones que incluyen algoritmo genético y de búsqueda directa. Se realizó un análisis previo de la eficiencia de diferentes algoritmos. A partir de ese análisis los algoritmos seleccionados e implementados se compararon con el software SBML-PET para corroborar esa eficiencia. Se realizaron transformaciones a los datos de entrada mediante un procedimiento no reportado en la literatura consultada.

ÍNDICE

INTRODUCCIÓN

La Biología de Sistemas (BS) es un área de las ciencias biológicas que integra conceptos e ideas de las ciencias de la vida, las ciencias de la computación y disciplinas ingenieriles. El desarrollo de herramientas para realizar estudios cualitativos y cuantitativos sobre las dinámicas de los sistemas vivos y la utilización de las mismas para resolver variados problemas científicos, es una buena forma de definir lo que se entiende por Biología de Sistemas. Los sistemas biológicos son muy complejos y difíciles de representar, por lo que se usan modelos matemáticos que puedan describirlos. [1]

Un modelo matemático es una descripción matemática de un fenómeno del mundo real, como puede ser el crecimiento de las poblaciones de animales, el funcionamiento de las neuronas y la dinámica intracelular, por citar solo algunos ejemplos de su aplicación en biología. La finalidad de estos modelos radica en entender en profundidad el fenómeno y tal vez realizar alguna predicción sobre su comportamiento futuro. [2]

Existen diferentes tipos de modelos matemáticos que pueden dar solución a una infinidad de problemas biológicos como Autómatas Celulares, Sistemas de Ecuaciones Diferenciales (SED), Sistemas de Ecuaciones Diferenciales en Derivadas Parciales y Ecuaciones en Diferencias o mapas (recursivas). Los sistemas de ecuaciones diferenciales son muy útiles para la representación de los sistemas biológicos. Son muy utilizados para representar los datos de un experimento y conocer los posibles resultados de una investigación.

Muchas situaciones involucran razones de cambio en la biología, por lo cual su modelo matemático estará dado por una ecuación diferencial o un conjunto de ellas. Una ecuación diferencial es una ecuación que involucra a una función desconocida y alguna de sus derivadas. Las ecuaciones diferenciales juegan un papel esencial en el modelado de procesos físicos, químicos, biológicos, económicos, atmosféricos, etc. En los procesos biológicos el uso de ecuaciones diferenciales es muy importante para analizar, representar y estimar el comportamiento de las poblaciones, especies, enfermedades, etc.

Es necesario para los científicos conocer el posible comportamiento y resultado de las investigaciones que realizan, así podrían tener una idea de los posibles resultados que van a obtener en un experimento dado, razón que concede mucho valor a la investigación de la estimación de parámetros. Esta consiste en el cálculo aproximado del valor de un parámetro en la población, utilizando la inferencia estadística, a partir de los valores observados en una muestra determinada.

En el mundo se han desarrollado algunas aplicaciones computacionales o librerías que pretenden facilitar la estimación de parámetros de un sistema de ecuaciones diferenciales como Biospreadsheet, SBML-PET, Cellware y libSRES. Estas presentan muchas ventajas y algunas deficiencias, pero el problema principal es que ninguna de ellas se puede integrar directamente a BioSyS. Por lo que se puede plantear como **problema científico**: ¿Cómo realizar estimación de parámetros en un sistema de ecuaciones diferenciales a partir de un conjunto de datos de laboratorio?

El **objeto de estudio** es la optimización y el **campo de acción** es la estimación de parámetros en sistemas de ecuaciones diferenciales.

Para solucionar esta problemática se ha propuesto como **objetivo general** de la investigación: Desarrollar algoritmos que permitan estimar los parámetros de un sistema de ecuaciones diferenciales a partir de un conjunto de datos de laboratorio.

De este objetivo general se derivan los siguientes **objetivos específicos**:

- ✓ Proponer algoritmos para la estimación de parámetros de un sistema de ecuaciones diferenciales a partir de un conjunto de datos de laboratorio.
- ✓ Desarrollar un prototipo funcional para comprobar las funcionalidades de los algoritmos.
- ✓ Analizar los resultados obtenidos.

Para alcanzar dichos objetivos se planteó desarrollar las siguientes **tareas**:

- ✓ Revisión del estado del arte acerca de algoritmos de optimización.
- ✓ Revisión del estado del arte acerca de los algoritmos de estimación de parámetros.
- ✓ Revisión del estado del arte de los acerca de los algoritmos y aplicaciones de estimación de parámetros de un sistema de ecuaciones diferenciales desarrolladas en el mundo.
- ✓ Diseño de los algoritmos para la estimación de parámetros.
- ✓ Implementación de los algoritmos para la estimación de parámetros de un sistema de ecuaciones diferenciales.
- ✓ Realización de pruebas para comprobar los algoritmo de estimación de parámetros.

ESTRUCTURA DEL DOCUMENTO

Este documento está compuesto por un resumen, introducción, 3 capítulos que constituyen el cuerpo fundamental del documento, conclusiones generales, bibliografía y referencias bibliográficas.

Capítulo 1

“Fundamentación teórica”: En este capítulo se presentará un resumen de la investigación realizada sobre la estimación de parámetros de un sistema de ecuaciones diferenciales. Se señalan las tendencias actuales y el estado del arte a tener en cuenta. Igualmente se describen las aplicaciones informáticas más relevantes desarrolladas hasta el momento sobre este tema.

Capítulo 2

“Materiales y Métodos”: En este capítulo se realiza la descripción de la investigación como base para la continuación del estudio por otros investigadores. Se abordan además los métodos más conocidos sobre la estimación de parámetros de un sistema de ecuaciones diferenciales realizados por otros autores. Se realiza la justificación de los materiales y métodos seleccionados y sus alternativas.

Capítulo 3

“Resultados y Discusión”: En este capítulo se explica brevemente los aspectos esenciales que se tuvieron en cuenta para la construcción del software de prueba, así como la implementación de los algoritmos y la integración con la Plataforma. Además se ilustran y analizan los resultados de la investigación y las pruebas realizadas al algoritmo. Se realiza una evaluación de las implicaciones, trascendencia y beneficios de estos resultados y las posibles aplicaciones del sistema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 INTRODUCCIÓN

En el presente capítulo se brinda una descripción general de la Biología de Sistemas, los modelos matemáticos, los sistemas de ecuaciones diferenciales. Se describen la optimización y la estimación de parámetros, así como el análisis de algunos algoritmos existentes que permiten la estimación de parámetros de un sistema de ecuaciones diferenciales. Finalmente se da una descripción de las herramientas y metodologías a utilizar para desarrollar la solución propuesta.

1.2 BIOLOGÍA DE SISTEMAS.

La Biología de Sistemas (BS) es un área de las ciencias biológicas que integra conceptos e ideas de las ciencias de la vida, las ciencias de la computación y disciplinas ingenieriles. El desarrollo de herramientas para realizar estudios cualitativos y cuantitativos sobre las dinámicas de los sistemas vivos y la utilización de las mismas para resolver variados problemas científicos, es una buena forma formas de definir lo que se entiende por Biología de Sistemas.

En la actualidad la Biología de Sistemas es una de las áreas de las ciencias interdisciplinarias más prometedoras. Gracias a la relación de campos como la Genómica, Proteómica, Bioinformática y la Biología celular con la Computación y las Matemáticas, se han llegado a descubrir principios fundamentales del funcionamiento de la vida celular antes desconocidos. [3]

La Biología de Sistemas trata de entender los sistemas biológicos a diferentes niveles de abstracción, desde el nivel molecular hasta los ecosistemas y haciendo uso de diferentes tipos de modelos matemáticos y técnicas computacionales.

La comunidad de investigadores de la BS ha propuesto un modelo al que llaman “las 4 Ms” para el estudio de los sistemas biológicos, dividiéndolo esencialmente en dos áreas, una experimental y otra computacional. Dentro del área experimental encontramos la manipulación de datos y las mediciones, dentro del área computacional la minería de datos y la modelación. La aplicación que se desarrollará está concebida dentro del área computacional, específicamente en la Modelación.

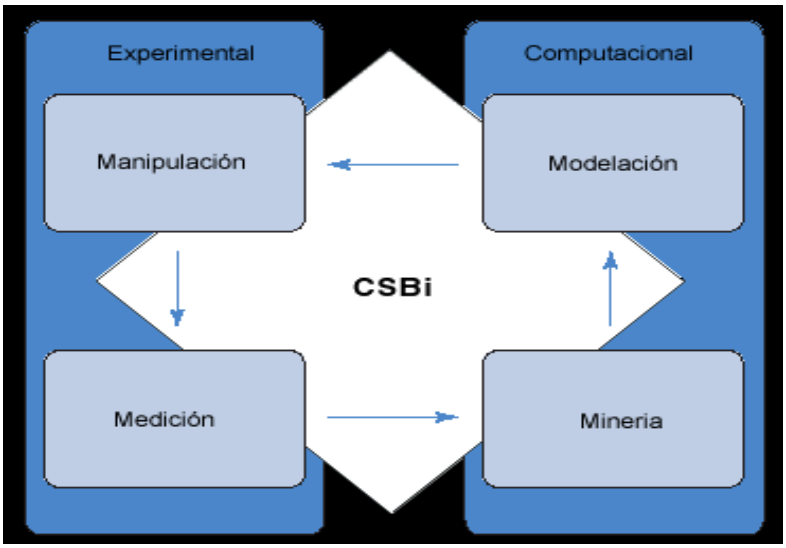

Figura 1.1 Modelo para el estudio de los sistemas biológicos.

Una definición más acertada sobre la Biología de Sistemas es que “Intenta describir los mecanismos que regulan la vida de los seres vivos de forma que, conocida la descripción, se pueda predecir qué va a ocurrir en una célula, tejido o ser vivo. Lo que se pretende es en el futuro poder describir a los seres vivos mediante modelos matemáticos que puedan predecir el mal funcionamiento de una célula o un ser vivo. Para por la que la presente investigación científica estudia la Biología de Sistemas para desarrollar un software que permita utilizando modelos matemáticos y en particular los SED, representar y analizar el posible comportamiento de un sistema biológico determinado. [4]

1.3 MODELOS MATEMÁTICOS.

El modelo matemático nos permite representar un problema médico o biológico de una manera objetiva en que se definen una serie de relaciones matemáticas entre las mediciones cuantitativas (del problema) y sus propiedades. Un modelo matemático es una descripción matemática de un fenómeno del mundo real, como puede ser el crecimiento de las poblaciones de animales, la concentración de un producto en una reacción química, el funcionamiento de las neuronas y la dinámica intracelular, por citar solo algunos ejemplos de su aplicación en biología. La finalidad de estos modelos radica en entender en profundidad el fenómeno y tal vez realizar alguna predicción sobre su comportamiento futuro.

Existen situaciones (mayoritariamente en biología) donde no hay leyes físicas que nos guíen, en estos casos debemos recolectar datos (ya sea mediante libros, bases de datos en Internet o realizando

experimentos) e intentar distinguir patrones. Luego podemos graficar estos patrones para obtener fórmulas matemáticas apropiadas. Muchas situaciones involucran razones de cambio, por lo cual su modelo matemático estará dado por una ecuación diferencial o un conjunto de ellas. Una ecuación diferencial es una ecuación que involucra a una función desconocida y alguna de sus derivadas. [2]

La segunda etapa de este proceso consiste en resolver matemáticamente el modelo que planteamos en la primera etapa para obtener conclusiones matemáticas. Estas conclusiones son analizadas en la tercera etapa e interpretadas como información sobre el fenómeno del mundo real. En la etapa final se comprueban las predicciones realizadas comparándolas con nueva información tomada del mundo real. Si las predicciones difieren en gran medida de la realidad deberemos aumentar la resolución o formular un nuevo modelo y comenzar el ciclo de nuevo. (Figura 1.2)

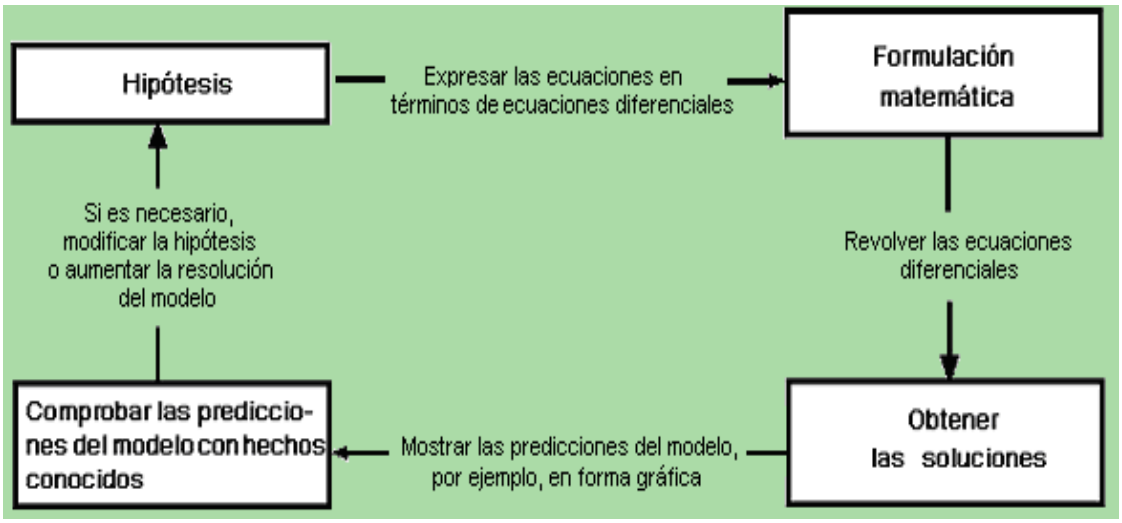

Figura 1.2 Forma de resolver los sistemas de ecuaciones diferenciales.

Existen diferentes tipos de modelos matemáticos que pueden dar solución a una infinidad de problemas biológicos como Autómatas Celulares, Sistemas de Ecuaciones Diferenciales, Sistemas de Ecuaciones Diferenciales en Derivadas Parciales y Ecuaciones en Diferencias o mapas (recursivas). Pero se utilizara los sistemas de ecuaciones diferenciales porque es una de las formas más fácil de representar sistemas biológicos a partir de datos de laboratorio.

1.4 SISTEMAS DE ECUACIONES DIFERENCIALES.

La aplicación de los modelos matemáticos es útil para comprender la complejidad de los fenómenos

biológicos. Las reacciones se expresan en sistemas de ecuaciones, y la integración numérica de las variables se aplica para simular estas reacciones. Han sido los sistemas de ecuaciones diferenciales los de más amplio uso en la modelación de problemas biológicos, debido fundamentalmente, a las facilidades de modelación y de simulación que brindan. Con SED se pueden describir gran cantidad de procesos biológicos y los métodos numéricos para la resolución de los mismos son muy conocidos y están disponibles tanto en la literatura como en diversos software.

Se dice que una ecuación que contiene las derivadas de una o más variables dependientes, con respecto a una o más variables independientes, es una ecuación diferencial. Razón por la que se considera a un SED como un conjunto de funciones diferenciales que satisfacen todas y cada una de las ecuaciones del sistema. Según el tipo de ecuaciones diferenciales puede tenerse un sistema de ecuaciones diferenciales ordinarias o un sistema de ecuaciones en derivadas parciales.

En los procesos biológicos el uso de ecuaciones diferenciales es muy importante para representar, analizar y estimar el comportamiento de las poblaciones, especies, enfermedades, etc. Por todas las causas antes mencionadas se le concede importancia al estudio de los SED, como modelo matemático a utilizar en la aplicación a implementar.

1.5 OPTIMIZACIÓN.

Es muy importante para los científicos conocer el posible comportamiento y resultado de las investigaciones que realizan, así podrían tener una idea de los posibles resultados que van a obtener en un experimento dado, razón que concede mucho valor al estudio de la estimación de parámetros. La estimación de parámetros consiste en el cálculo aproximado del valor de un parámetro en la población, utilizando la inferencia estadística, a partir de los valores observados en una muestra determinada.

La estimación se hace cuando tenemos un conjunto de datos experimentales y se quiere saber los parámetros del SED que mejor se ajustan a dichos datos, esto se resuelve como un problema de optimización.

En la BS es necesaria la optimización de las funciones objetivos para saber los parámetros que permiten lograr los mejores resultados posibles para un experimento. Un problema de optimización trata entonces de tomar una decisión óptima para maximizar o minimizar un criterio determinado. El término optimización se aplica a aquellos casos en que se maximiza o minimiza la función objetivo.

Los problemas de optimización, por comodidad, se suelen enfocar sólo hacia búsquedas de mínimos.

Cualquier máximo de la función modelo se puede considerar siempre como el opuesto de un mínimo, lo que permite la búsqueda de máximos con los mismos algoritmos que para mínimos: simplemente basta con cambiar el signo de la función objetivo. En los problemas de optimización es frecuente encontrar funciones que contengan varios máximos y mínimos. Se considera óptimo al más favorable de los mínimos relativos (o máximos, si se maximiza la respuesta), de entre todos los encontrados en el espacio de los parámetros. En general, el más favorable suele ser el más profundo de todos los mínimos o el máximo más elevado, aunque en algunas ocasiones hay que tener en cuenta otros factores que consideren la realidad experimental antes de su elección.

1.6 ALGORITMOS DE OPTIMIZACIÓN.

En el mundo existen varios algoritmos de optimización, como el método Levenberg-Marquardt, la Programación Genética, Simulada Recocido, RBFN, Algoritmos Genéticos y Búsqueda Directa. Estos algoritmos se investigarán para determinar en cuales se va ha profundizar el estudio.

EL MÉTODO LEVENBERG-MAQUARDT.

Algoritmo de Levenberg - Marquardt: Este algoritmo es una modificación del método de Newton, el que fue diseñado para minimizar funciones que sean la suma de los cuadrados de otras funciones no lineales; es por ello que el algoritmo de Levenberg - Marquardt, tiene un excelente desempeño en el entrenamiento de redes neuronales donde el rendimiento de la red esté determinado por el error medio cuadrático.

Este algoritmo requiere mucha más computación por iteración, debido a que implica el cálculo de matrices inversas. A pesar de su gran esfuerzo computacional sigue siendo el algoritmo de entrenamiento más rápido para redes neuronales cuando se trabaja con un moderado número de parámetros en la red, si el número de parámetros es muy grande utilizarlo resulta poco práctico. Debido a la gran cantidad de parámetros en la aplicación a desarrollar este algoritmo no es el más indicado.

SIMULADA RECOCIDO.

El método del recocido se utiliza en la industria para obtener materiales más resistentes, o más cristalinos, en general, para mejorar las cualidades de un material.

El proceso consiste en “derretir” el material (calentarlo a muy alta temperatura). En esa situación, los átomos adquieren una distribución “azarosa” dentro de la estructura del material y la energía del sistema es máxima. Luego se hace descender la temperatura muy lentamente por etapas, dejando que en cada

una de esas etapas los átomos queden en equilibrio (es decir, que los átomos alcancen una configuración óptima para esa temperatura). Al final del proceso, los átomos forman una estructura cristalina altamente regular, el material alcanza así una máxima resistencia y la energía del sistema es mínima.

El algoritmo se divide en etapas. A cada etapa le corresponde una temperatura menor que la que tenía la etapa anterior (a esto hace referencia la monotonía: después de cada etapa la temperatura baja, se enfría el sistema). Por lo tanto hace falta un criterio de cambio de la temperatura ("cuanto tiempo" se espera en cada etapa para dar lugar a que el sistema alcance su "equilibrio térmico").

Simulada Recocido: el criterio de aceptación es probabilística

Si c (j) – c (i) < 0 entonces acepto j

Si c (j) – c (i) > 0 entonces acepto j con probabilidad exp [(c (i) – c (j)) / t_k]

(En la interacción k se genera un numero al azar r y se acepta j si r < exp [(c (i) – c (j)) / t_k])

En este caso, cada vecino de una solución tiene una probabilidad positiva de reemplazar a la solución actual. Los t_k se eligen de forma tal que a medida que avanzan las iteraciones, aceptar soluciones con grandes incrementos en el costo es menos probable (pero sigue existiendo una probabilidad positiva de aceptarlos). [5]

FUNCIÓN DE BASE RADIAL DE LA RED (RBFN).

La función de base radial de la red (RBFN), es uno de los métodos de aprendizaje artificiales que describen complejas relaciones no lineales entre entradas y salidas. El RBFN puede resolver parabólica, hiperbólica, elíptica y ecuaciones diferenciales parciales numéricamente y es capaz de aproximación no lineal del tiempo efectivamente. Debido a la creciente acumulación de información biológica, RBFN es ahora aplicado en el campo de la bioquímica.

Se ha probado el algoritmo RBFN en la estimación de parámetros para las búsquedas a nivel local y mundial. RBFN permite reducir el coste computacional omitiendo integraciones numéricas. Se ha aplicado el método RBFN para la estimación de parámetros en un biosistema fijo. Los resultados demuestran que en comparación con el algoritmo genético, este método facilita la adquisición más exacta o equivalente a la mitad de los parámetros de cálculo de tiempo y el rendimiento de un 50% de aumento en la tasa de éxito de optimización. Este algoritmo se propone para una versión posterior del software. [6]

ALGORITMO GENÉTICO.

Los algoritmos genéticos corresponden a la clase de métodos estocásticos de búsqueda. Mientras la mayoría de estos métodos operan sobre una única solución, estos algoritmos operan en una población de soluciones. La idea básica, inspirada en los procesos evolutivos en biología es que el contenido genético de una población contiene potencialmente la solución, o una solución mejor, a un dado problema de adaptación. [7]

Toda la información sobre este algoritmo está en la ayuda de Matlab y se encuentra implementado, con posibilidades de modificarlo en caso necesario.

BÚSQUEDA DIRECTA.

Es un método para resolver problemas de optimización que no requiere ninguna información sobre el gradiente de la función objetivo. A diferencia de más tradicionales de optimización de los métodos que utilizan información sobre el gradiente o superior derivados para buscar un punto óptimo, un algoritmo de búsqueda directa de las búsquedas una serie de puntos alrededor del punto actual, en busca de uno cuando el valor de la función objetivo es menor que el valor en el punto actual.

Toda la información sobre este algoritmo está en la ayuda de Matlab y se encuentra implementado, con posibilidades de modificarlo en caso necesario.

1.7 SOFTWARES EXISTENTES.

En el mundo se han desarrollado algunas aplicaciones computacionales o librerías que pretenden facilitar la estimación de parámetros de un sistema de ecuaciones diferenciales como BioGrid, Cellware y libSRES. Estas presentan muchas ventajas y algunas deficiencias, pero el problema principal es que ninguna de ellas se puede integrar directamente a BioSyS.

A continuación se detallan las características de algunas de estas aplicaciones y se explica el porque no pueden ser utilizadas en BioSyS.

1.7.1 BIOSPREADSHEET

El depósito general biológico para los Datasets de la interacción (BioGrid) es la base de datos biológica de las interacciones de la proteína-proteína. Se esfuerza proporcionar un recurso comprensivo de las interacciones de la Proteína-Proteína para todas las especies importantes. En conjunto con la aplicación Biospreadsheet utilizan un archivo que se llama ESS, desarrollado en el lenguaje Java y realiza la

estimación de parámetros en sistemas de ecuaciones diferenciales. Pero es muy rígido en la entrada de datos.

1.7.2 CELLWARE

Cellware no sólo ha sido diseñado para realizar el modelado y simulación de genes reguladores y de vías metabólicas, sino también ofrecer un entorno integrado de las diversas representaciones matemáticas, la estimación de parámetros y optimización. Además, de fácil visualización gráfica y la capacidad para ejecutar grandes y complejos modelos serían proporcionados por defecto. Una característica muy especial de Cellware es, ser la primera red basada en la herramienta de modelado y simulación en el campo de la Biología de Sistemas pero, al no contar con el acceso a la Grid que utiliza, no se puede integrar a BioSyS.[8]

1.7.3 LIBSRES

Es una librería implementada en C como una estrategia evolutiva de algoritmos estocásticos para la estimación de parámetros en sistemas de ecuaciones diferenciales que no es multiplataforma. En algunos casos la estimación de parámetros puede verse como un caso particular de la optimización de una función objetiva si lo que queremos saber es los mejores resultados calculados con respecto a los datos experimentales. [9]

1.7.4 SBML - PET

SBML-PET (System Biology Markup Language - Parameter Estimation Tool) es una herramienta de estimación de parámetros que utiliza el lenguaje para el modelado de sistemas biológicos. Esta herramienta de trabajo ha sido diseñada para hacer la estimación de parámetros para modelos biológicos incluyendo vías de señalización, redes de regulación genéticas y las vías metabólicas. SBML-PET apoya la importación y exportación de modelos en el formato SBML. Actualmente, puede funcionar en Linux y Cygwin en Windows.

SBML - PET no posee una interfaz apropiada donde se puedan entrar los datos necesarios para las estimaciones y todos las operaciones son solicitadas por consola, lo que resulta muy complicado para el usuario.[10]

1.8 CONCLUSIONES

Después de haber realizado un estudio y análisis del estado del arte de las aplicaciones informáticas que

son utilizadas en el resto del mundo para la estimación de parámetros en un sistema de ecuaciones diferenciales es preciso mencionar que todas las aplicaciones encontradas se consideran buenas, pero no pueden ser integradas al software BioSyS, fundamentalmente porque se desconocen los formatos de entrada de datos o resultan difíciles de trabajar los mismos. Para esta primera versión se escogieron Algoritmo Genético y Búsqueda Directa porque son algoritmos conocidos, implementados en Matlab que es el asistente matemático que usa BioSyS, aunque no se descarta la utilización de otros algoritmos en futuras versiones. Además estos dos algoritmos no requieren de condiciones adicionales para su funcionamiento, lo que facilita notablemente su utilización.

CAPÍTULO 2: MATERIALES Y MÉTODOS

2.1 INTRODUCCIÓN

Como se trabaja en la elaboración de un software, la sección de materiales está centrada en metodología de desarrollo, el lenguaje de programación utilizado, las herramientas de implementación que se utilizaron, el lenguaje de modelado y la herramienta case. La sección dedicada a los métodos se describen los algoritmos matemáticos seleccionados en la implementación de la aplicación, así como aquellos algoritmos novedosos que el equipo de desarrollo ha implementado para resolver los problemas concretos a los que nos enfrentamos.

2.2 MATERIALES

2.2.1 METODOLOGÍAS DE DESARROLLO DE SOFTWARE

PROCESO UNIFICADO ABIERTO (OpenUP)

El OpenUP es un Proceso Unificado que aplica propuestas iterativas e incrementales dentro del ciclo de vida, tratando de ser manejable en relación con el RUP. Debemos estar claro que el uso de OpenUP se debe realizar cuando se tiene un equipo pequeño, cuando se quiere evitar a ser cargado excesivamente con metodologías formales improductivas.

Una manera buena de entender que OpenUP es pensar en él como:

Un proceso ágil, ligero como que promueve el desarrollo del software las prácticas más buenas, asiéndolo un proceso pequeño extensible si es necesario (hibrido, incluir partes de otros modelos).

Sin embargo, OpenUP asume que el equipo del proyecto no es con toda seguridad responsable de las actividades y decisiones que se asignan a otras áreas de la organización.

Se considera que estas nuevas metodologías son extensamente aplicables y deben no ser usadas si no se tiene requisitos estables, ya que no se estaría en la posición de seguir un proceso planeado, formal. En estas situaciones un proceso unificado puede ser menos cómodo.

Después de analizar todas estas metodologías se ha definido a Open UP como la más factible para la aplicación que se está desarrollando.

2.2.2 LENGUAJE DE MODELADO

UML no es un lenguaje de programación sino un lenguaje de propósito general para el modelado orientado a objetos y también puede considerarse como un lenguaje de modelado visual que permite una abstracción del sistema y sus componentes.

El lenguaje UML tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, etc., hasta la implementación y configuración con los diagramas de despliegue.

Otra de las ventajas que presenta el UML es que es independiente del lenguaje de implementación. Además por ser un método formal de modelado aporta las ventajas de mayor rigor en la especificación, verificación y validación del modelo realizado, permite automatizar algunos procesos y posibilita la generación de código a partir de modelos y viceversa.

2.2.3 HERRAMIENTA CASE

Visual Paradigm es una potente herramienta CASE que utiliza como lenguaje de modelación el UML. La herramienta fue desarrollada para una amplia gama de usuarios incluyendo Ingenieros de Software, analista de sistemas, analistas del negocio y arquitectos de sistemas. Proporciona a los desarrolladores una plataforma con interfaz amigable que les permite diseñar un producto con calidad de forma rápida.

Facilita la interoperabilidad con otras herramientas CASE y se integra con las siguientes herramientas Java: Eclipse/IBM WebSphere, JBuilder, NetBeans IDE, Oracle JDeveloper, BEA Weblogic.

Está disponible en varias ediciones: Enterprise, Professional, Community, Standard, Modeler y Personal. Por el equipo de desarrollo trabajar sobre la plataforma GNU\Linux y no contar con la versión de Rational Rose que corre sobre dicha plataforma se usará para el desarrollo de la aplicación la herramienta CASE Visual Paradigm.

2.2.4 LENGUAJE DE PROGRAMACIÓN

Se utilizará Java como lenguaje de programación para desarrollar la herramienta por ser un lenguaje cuya portabilidad está verdaderamente probada y no requerir largos períodos de tiempo para el desarrollo de las aplicaciones.

Otras características de dicho lenguaje son:

Orientado a Objetos: Java trabaja con sus datos como objetos y con interfaces a esos objetos, soporta las características propias del paradigma orientado a objetos: abstracción, encapsulamiento, herencia y polimorfismo.

Simple: Posee una curva de aprendizaje muy rápida. Ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos.

Robusto: Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores lo antes posible en el ciclo de desarrollo. Java obliga a la declaración explícita de los tipos de los ítems de información, reduciendo así las posibilidades de error. Maneja la memoria para eliminar las preocupaciones por parte del programador de la liberación o corrupción de la misma.

La razón principal por la que se utiliza Java es que el software BioSyS está desarrollado en este lenguaje de programación.

2.2.5 HERRAMIENTAS DE IMPLEMENTACIÓN

NETBEANS

Dentro de los entornos de desarrollo integrado (IDE del inglés Integrated Development Environment) para el desarrollo de aplicaciones usando como lenguaje de programación Java se pueden encontrar NetBeans, JBuilder y Eclipse. De todas estas, la herramienta que se ha seleccionado para implementar la interfaz visual de la aplicación es NetBeans.

El NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans, es un producto libre y gratuito sin restricciones de uso.

MATLAB

Es un entorno de computación y desarrollo de aplicaciones, totalmente integrado orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos. Matlab integra análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en que se escribirían tradicionalmente, sin necesidad de hacer uso de la programación tradicional. Matlab

permitirá a la desarrollar una implementación sencilla de algunos algoritmos matemáticos necesarios para su funcionamiento.

El Matlab cuenta con paquetes de funciones que permiten resolver múltiples problemas, además de que brindar la posibilidad de implementar funciones propias, haciendo uso del lenguaje de programación antes mencionado. Para resolver SED existen en el Matlab ocho métodos numéricos. Usando estos métodos e implementando funciones propias es posible realizar estudios de sistemas dinámicos, pero, para ellos hay que tener un dominio profundo del Matlab y de su lenguaje de programación, requisitos estos, muy alejados de los conocimientos que posee un investigador en ciencias biológicas.

2.3 MÉTODOS

A continuación se comienzan a detallar los principales algoritmos existentes que serán utilizados durante la presente investigación.

2.3.1 ALGORITMOS DE INTEGRACIÓN NUMÉRICA

Un sistema de ecuaciones diferenciales tiene la forma general:

$$dx/dt = f(x, \{\mu\}, t)$$

Donde x es un vector en un espacio n-dimensional, f (.) es un vector de funciones, {μ} es un conjunto de parámetros y t es el tiempo de integración. Dependiendo del contexto las variables x = (x1, x2,..., xn) pueden ser tanto las posiciones y velocidades de los planetas como las poblaciones de un conjunto de especies en un ecosistema, por mencionar solo dos posibilidades. La evolución dependerá también de los valores del conjunto de parámetros {μ}.

La mayoría de los SED que se obtienen como resultado de la modelación de problemas reales son altamente no lineales y no tienen solución analítica, por lo que se hace necesario recurrir al uso de métodos numéricos. Muchos métodos han sido desarrollados en los últimos años, buscando las mejores aproximaciones y algoritmos cada vez más rápidos.

Teniendo en cuenta que la aplicación estaría acoplada al asistente matemático Matlab, podemos utilizar todos aquellos métodos de resolución de SED que el mismo implementa. Existen distintos métodos numéricos para la resolución de sistemas de ecuaciones diferenciales, pero el método seleccionado es el Runge-Kutta porque los resultados obtenidos tienen menor error. Toda la información referente a los métodos numéricos se puede encontrar en la ayuda del Matlab.

MÉTODO DE RUNGE-KUTTA

“Basados en la Serie de Taylor, donde la cantidad de términos que se decidan tomar de la serie determinará la precisión del método. En estos métodos se le denomina paso (h) a ()n+nXX−1 este paso puede ser fijo obteniéndose puntos equidistantes; y también se puede variar haciendo aumentar el paso al doble en caso de detectar errores pequeños o reducirlo a la mitad en presencia de errores grandes, esto complica el programa de cálculo, pero aumenta la eficiencia y en la mayoría de los casos son más rápidos.

El método de segundo orden requiere evaluar cada función 2 veces en cada paso de integración y el error local es del orden h3, de la misma forma el método de cuarto orden necesita 4 evaluaciones de cada función y posee un error local del orden h5; esto posibilita que se pueda integrar con un paso mayor para lograr un error similar al que obtendríamos si utilizásemos el de orden dos, mejorando así el tiempo de ejecución del algoritmo.

Los métodos de Runge-Kutta son métodos de paso simple, que sólo requieren de los resultados del paso anterior, esto les brinda la posibilidad de auto iniciarse ya que parten de las condiciones iniciales. Al igual que todos los métodos de paso simple no presentan inestabilidad numérica para paso h suficientemente pequeño, esto implica que pequeños cambios en las condiciones iniciales del sistema sólo originan cambios acotados en la solución. Una desventaja de los métodos Runge-Kutta es que requieren múltiples evaluaciones de la funciones en cada paso…“[11]

El objetivo de los métodos de Runge-Kutta es evitar el cálculo de derivadas de f(x, y), a base de calcular F (t, y) en más puntos, manteniendo el orden del error y la complejidad de cálculo. Los métodos de orden 2, 3 y 4 requieren el cálculo de F (t, y) en 2, 3 y 4 puntos respectivamente. Los métodos de orden m > 4 requieren el cálculo de F (t, y) en más de m puntos (no más de m+2). Se emplea una función de incremento φ (ti, Yi, h), para la aproximación de F (t, y) en intervalo [ti, ti+1]. [12]
El método de Runge-Kutta se utiliza en este caso para resolver numéricamente el conocido problema de Cauchy.

Problema de Cauchy

Definición: Se llama problema de Cauchy o problema de valores iniciales, asociado a una E.D.O. de orden uno, al problema de resolver:

$$(P) \equiv \begin{cases} y' = f(x, y) & \text{ecuación diferencial} \\ y(x0) = y0 & \text{condición inicial} \end{cases}$$

Intuitivamente de lo que se trata es de encontrar una solución de y´ = f (x, y) pero con la condición de que pase por el punto (x0, y0). [13]

Este algoritmo se utiliza para resolver las ecuaciones diferenciales que hacen falta para evaluar la función objetivo.

2.3.2 FUNCIÓN OBJETIVO

La función objetivo que se emplea para la optimización es la siguiente:

$$Fobj = \sum_{n}^{N} \sum_{c}^{P} \sum_{i}^{O} (Ypred(n, c, i) - Yexp(n, c, i))^2$$

Donde:

- N es el número de experimentos.
- P es el número de tiempos medidos por cada experimento.
- O es el número de variables del modelo.
- Ypred es la predicción de datos del modelo matemático.
- Yexp es el resultado de la transformación de los datos medidos en el experimento.

Esta función se utiliza con el fin de estimar la distancia entre las series temporales obtenidas en las simulaciones de las medidas experimentalmente.

2.3.3 ALGORITMO GENÉTICO

El algoritmo genético es un método para resolver tanto limitada como ilimitadamente problemas de optimización, que se basa en la selección natural, el proceso que impulsa la evolución biológica. El algoritmo genético en repetidas ocasiones modifica una población de soluciones individuales. En cada paso, el algoritmo genético selecciona personas al azar de la población actual para ser padres y utiliza a

los hijos para producir la próxima generación. Durante las sucesivas generaciones, la población "evoluciona" hacia una solución óptima. Puede aplicar el algoritmo genético para resolver una variedad de problemas de optimización que no son muy adecuados para la optimización de los algoritmos estándar, incluidos los problemas en que la función objetivo es discontinuo, no diferenciables, estocásticos, o altamente no lineales. Todo lo referente al algoritmo genético se encuentra en la ayuda de Matlab.

Terminología de los Algoritmos Genéticos:

Funciones de Adecuación

La función de adecuación (la función objetivo) es la función que desea optimizar.

Los individuos

Un individuo es cualquier punto al que se puede aplicar la función de idoneidad. El valor de la función de idoneidad para un individuo es su puntuación. Un individuo es a veces lo que se refiere como el genoma y el vector de entradas de un individuo como los genes.

Poblaciones y Generaciones

Una población es un conjunto de individuos. Por ejemplo, si el tamaño de la población es 100 y el número de variables en la función de adecuación es de 3, representan una matriz de población de 100*3. En cada iteración, el algoritmo genético realiza una serie de cálculos sobre la población actual para producir una nueva población. Cada una de las sucesivas poblaciones llama una nueva generación.

Diversidad

La diversidad se refiere a la distancia media entre individuos de una población. Una población tiene una alta diversidad, si la media distancia es grande; de lo contrario es de baja diversidad. La diversidad es esencial para el algoritmo genético, ya que permite al algoritmo de búsqueda una mayor región del espacio.

Valores de adecuación y mejores valores de adecuación.

El valor de adecuación de un individuo es el valor de la función de idoneidad para ese individuo. Debido a que la caja de herramientas encuentra el mínimo de la función de idoneidad, el mejor valor para una población es el valor más pequeño para cualquier individuo en la población.

Padres e hijos

Para crear la próxima generación, el algoritmo genético elige a determinados individuos en la población actual, llamado los padres, y los usa para crear los individuos de la próxima generación, llamados hijos. Normalmente, el algoritmo es más probable que seleccione los padres que tienen mejores valores de adecuación.

¿Como funcionan los Algoritmos Genéticos?

1) El algoritmo comienza por la creación de una población inicial aleatoria.
2) Genera una secuencia de nuevas poblaciones, o usa las generaciones los individuos de la generación actual para crear la próxima generación. Para esto, el algoritmo realiza los siguientes pasos:
   - Analiza los valores de cada uno de los miembros de la población actual para calcular el valor de adecuación.
   - Recorre los resultados de adecuación en bruto para convertirlos en un rango de valores más usados.
   - Selecciona los padres sobre la base de su adecuación.
   - Produce los hijos de los padres. Los hijos son producidos ya sea por hacer cambios al azar a un solo progenitor - mutación - o mediante la combinación de los vectores entradas de un par de padres - cruzamiento.
   - Sustituye a la población actual con los hijos para formar la próxima generación.
3) El algoritmo se detiene cuando uno de los criterios de parada se cumple.

Población inicial

El algoritmo comienza por la creación de una población inicial aleatoria, como se muestra en la siguiente:

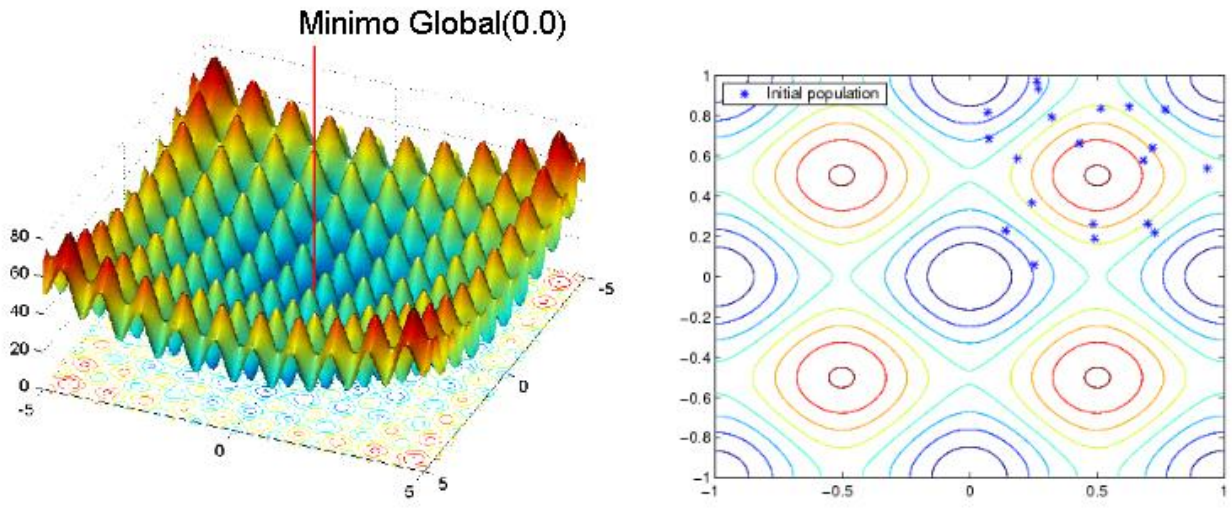

Figura 2.3.1 Condiciones iniciales del Algoritmo Genético.

La población inicial contiene 20 individuos por defecto. Todas las personas en la población inicial tienen genes que se encuentran entre 0 y 1, ya que el valor por defecto del rango inicial de las opciones de la población es [0, 1].

Normas de creación:

El algoritmo genético utiliza tres tipos principales de las normas en cada paso para crear la próxima generación de la actual población:

-Normas de selección para seleccionar a los individuos, llamando los padres, que contribuyen a la población en la próxima generación.

-Normas de creación para combinar dos padres para formar hijos para la próxima generación.

-Normas de mutación para que se aplican los cambios al azar a cada uno los padres para formar hijos.

Características del algoritmo genético:

-Genera una población de puntos en cada iteración. El mejor punto a la población acerca a una solución óptima.

-Selecciona la próxima población de cómputo que utiliza generadores de números aleatorios.

Parcelas de las generaciones futuras

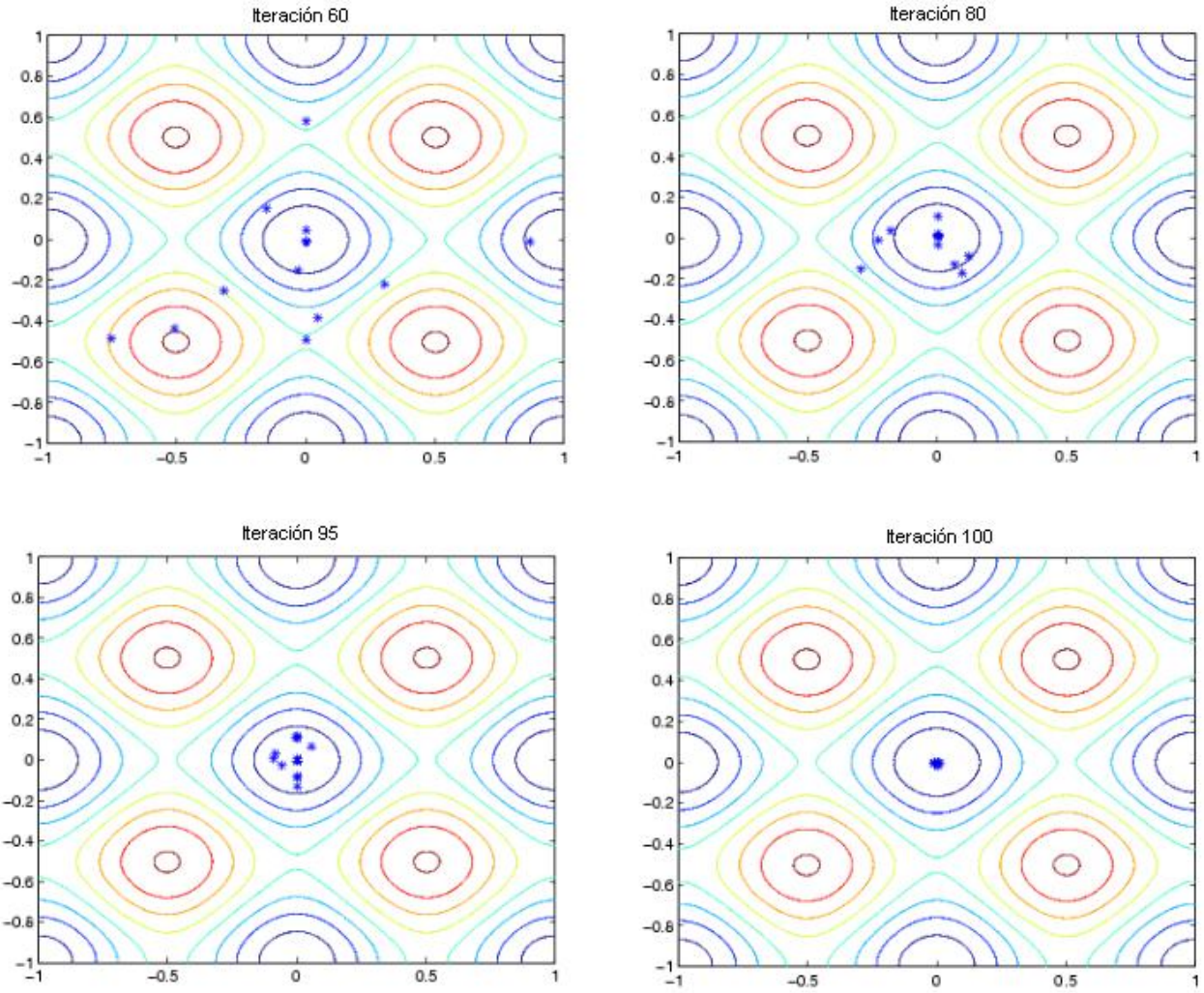

Figura 2.3.2 Parcelas de las generaciones futuras.

Condiciones de Parada

El algoritmo genético utiliza los siguientes cinco condiciones para determinar cuándo parar:

Generaciones: El algoritmo se detiene cuando el número de generaciones alcanza el valor de las Generaciones.

Límite de Tiempo: El algoritmo se detiene después de ejecutar por un importe de tiempo en segundos igual al Límite de Tiempo.

Límite de Adecuación: El algoritmo se detiene cuando el valor de la función de idoneidad para el mejor punto en la población actual es inferior o igual al Límite de Adecuación.

Sustentación de Generaciones: El algoritmo se detiene si no hay mejora en la función objetivo para una secuencia de generaciones consecutivas de longitud Sustentación de Generaciones.
Sustentación de Limite de Tiempo: El algoritmo se detiene si no hay mejora en la función objetivo durante un intervalo de tiempo en segundos igual a Sustentación de Limite de Tiempo.

2.3.4 BÚSQUEDA DIRECTA

Búsqueda directa es un método para resolver problemas de optimización que no requieren ninguna información sobre el gradiente de la función objetivo. El algoritmo de búsqueda directa busca una serie de puntos alrededor del punto actual, en buscando un punto donde el valor de la función objetivo es menor que el valor en el punto actual. Puede utilizar métodos de búsqueda directa para resolver una variedad de problemas de optimización que no son muy adecuados para la optimización con algoritmos estándar, incluidos los problemas en que la función objetivo es discontinuo, no diferenciables, estocásticos, o altamente no lineales. Nos centraremos en una clase especial de algoritmos de búsqueda directa llamado patrón de búsqueda.

Patrón de búsqueda.

El algoritmo patrón de búsqueda calcula una secuencia de puntos que se acerque al punto óptimo.

1. En cada paso, el algoritmo busca un conjunto de puntos, llamado malla, en torno al punto actual, el punto calculado en el paso anterior del algoritmo.

2. El algoritmo constituye una malla añadiendo al punto actual un escalar múltiple de un conjunto fijo de vectores llamado patrón.

3. Si el algoritmo encuentra un punto en la malla que mejora la función objetivo en el punto actual, el nuevo punto se convierte en el punto actual en el siguiente paso del algoritmo.

Patrones

Un patrón es una colección de vectores que utiliza el algoritmo para determinar cuales puntos buscar en cada iteración. Por ejemplo, si hay dos variables independientes en el problema de optimización, el valor por defecto consiste en los siguientes vectores.

v1 = [1 0], v2 = [0 1], v3 = [-1 0], v4 = [0 -1].

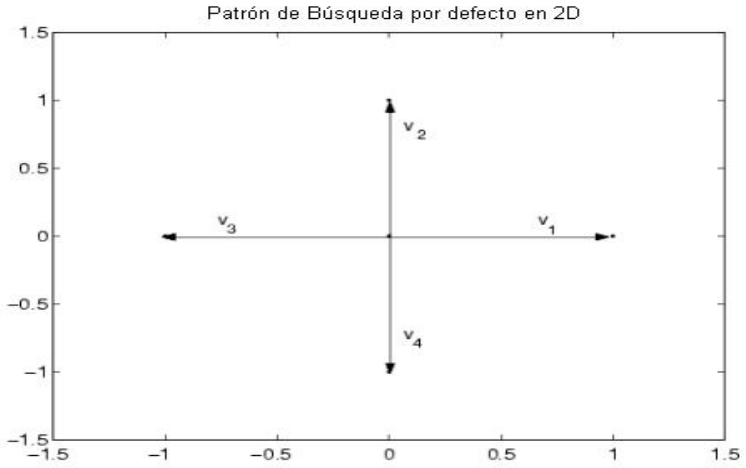

Figura 2.3.3 Patrón de búsqueda.

Mallas

En cada paso, el algoritmo patrón de búsqueda busca un conjunto de puntos, llamado malla, para mejorar la función objetivo. Como el algoritmo constituye la malla:

1. Multiplicando el patrón de vectores por un escalar.

2. Agregando los vectores resultantes para el actual punto (el punto con el mejor valor de la función objetivo encontrada en el paso anterior). Por ejemplo, supongamos que el punto actual es [1.6 3.4] y el tamaño de la malla 4.

El algoritmo multiplica el patrón de vectores por 4 y los añade al actual punto para obtener la siguiente malla.

[1.6 3.4] + 4 * [1 0] = [5.6 3.4]

[1.6 3.4] + 4 * [0 1] = [1.6 7.4]

[1.6 3.4] + 4 * [-1 0] = [-2.4 3.4]

[1.6 3.4] + 4 * [0 -1] = [1.6 -0.6]

Plan del patrón de búsqueda

En cada paso, el algoritmo sondea los puntos en la actual malla para calcular los valores de la función objetivo. Por defecto, el algoritmo detiene los sondeos de la malla tan pronto como se encuentre un punto

cuyo valor de la función objetivo es menor que el del actual punto. El sondeo es entonces llamado exitoso y el punto se convierte en el punto actual en la próxima iteración. Si usted cambia el sondeo a una completo, el algoritmo calcula los valores de la función objetivo en todos los puntos de malla.

Después de un exitoso sondeo, el algoritmo multiplica la actual malla por 2, el valor por defecto del factor de expansión de la malla. Debido a que el tamaño de la malla inicial es 1, a la segunda iteración el tamaño de la malla es de 2. Si el algoritmo no puede encontrar un punto que mejora la función objetivo, el sondeo no tiene éxito y el punto actual sigue siendo el mismo en la próxima iteración. Después de un infructuoso sondeo, el algoritmo multiplica el actual tamaño de la malla por 0.5, el valor por defecto de factor de contracción de malla. El algoritmo realiza entonces el sondeo con un menor tamaño de la malla.

2.4 CONCLUSIONES

A modo de conclusiones se hace un resumen de los materiales y métodos utilizados en el desarrollo de la aplicación.

- Materiales
  - ✓ Metodología de desarrollo
    - ◆ OpenUP
  - ✓ Lenguaje de modelado
    - ◆ UML
  - ✓ Herramienta Case
    - ◆ Visual Paradigm
  - ✓ Lenguaje de programación
    - ◆ Java
  - ✓ Herramientas de implementación
    - ◆ NetBeans
    - ◆ MatLab
- Métodos

- ✓ Algoritmos de integraciones numéricas
  - ◆ Método de Runge-Kutta
- ✓ Algoritmos de estimación de parámetros
  - ◆ Algoritmo Genético
  - ◆ Búsqueda Directa

CAPÍTULO 3: RESULTADOS Y DISCUSIÓN

3.1 INTRODUCCIÓN

En el presente capítulo se exponen los principales resultados obtenidos a lo largo del trabajo, a la vez que se discute sobre la importancia y relevancia de los mismos.

3.2 ESQUEMA GENERAL DE LOS ALGORITMOS

Este esquema describe los pasos del proceso de estimación que se quiere desarrollar con los algoritmos de manera general. Muestra una etapa inicial donde los datos de los experimentos se transforman en función del modelo matemático. Posteriormente se realiza la inicialización con la asignación y selección al azar de los valores de los parámetros. Se resuelve el sistema de ecuaciones diferenciales usando el método de Runge-Kutta de Matlab y con la función objetivo se calcula la distancia entre estas soluciones y los datos experimentales transformados. La evolución se realiza al clasificar, seleccionar y producir la descendencia, garantizada la próxima generación se vuelve a resolver el sistema de ecuaciones diferenciales para la población actual, se calcula la distancia y se vuelve a realizar la evolución, el proceso culmina al cumplirse alguno de los criterios de parada implementados en el método y descritos en materiales y métodos.

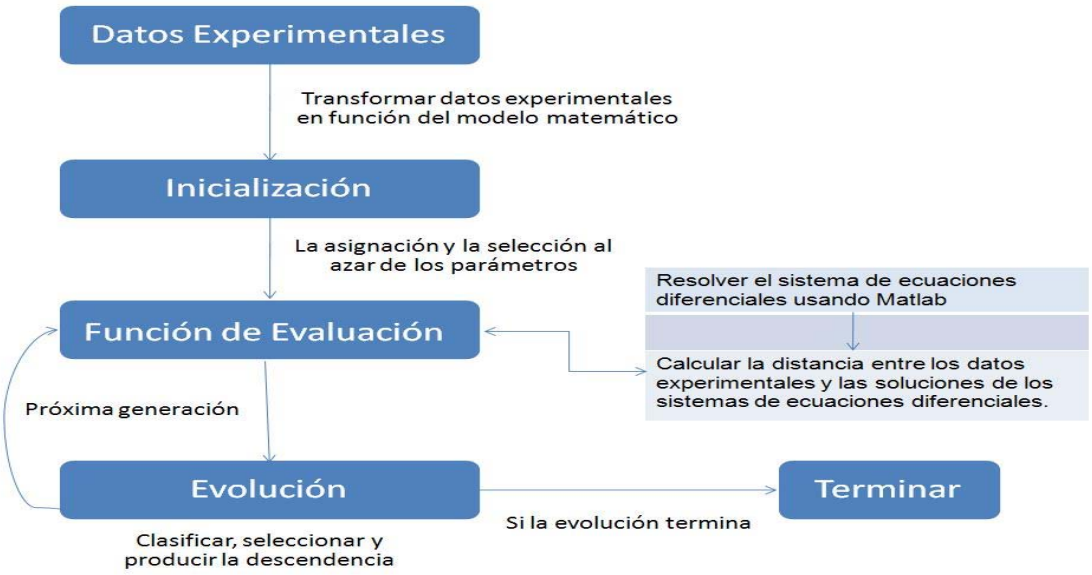

Figura 3.2 Esquema general de algoritmos de estimación.

3.3 ALGORITMOS DE ESTIMACIÓN DE PARÁMETROS

Ya se mostró el esquema general de funcionamiento de los algoritmos de estimación de parámetros, explicándose de manera general las diferentes etapas de los mismos. Pero cada algoritmo realiza la evolución de manera específica para cada uno de ellos como se explica a continuación.

3.3.1 ALGORITMO GENÉTICO

El algoritmo genético para la evolución crea la próxima generación con las normas mencionadas en la sesión 2.3.3, como se muestra en la siguiente figura.

Figura 3.3.1 Esquema del Algoritmo Genético.

3.3.2 BUSQUEDA DIRECTA

El algoritmo de búsqueda directa para la evolución selecciona la próxima generación siguiendo el patrón de búsqueda mencionadas en la sesión 2.3.4, como se muestra en la siguiente figura.

Figura 3.3.1 Esquema del algoritmo de Búsqueda Directa.

3.4 DISEÑO DE CLASE Y PATRONES DEL DISEÑO

3.4.1 DIAGRAMAS DE CLASES

Un subsistema es un sistema cuya operación es independiente de los servicios provistos por otros subsistemas, por lo que se definió un solo subsistema para modelar el diagrama de clases del diseño.

Subsistema Estimación: Este subsistema se encarga de la gestión del proceso de estimación de parámetros de ecuaciones diferenciales utilizando datos obtenidos en el laboratorio. Dado una serie de

datos experimentales, la selección de un modelo matemático y una serie de condiciones, se devuelven los valores de los parámetros donde se obtienen los mejores resultados. Existen varias interfaces porque el prototipo funcional que se implemento se comporta como un wizard.

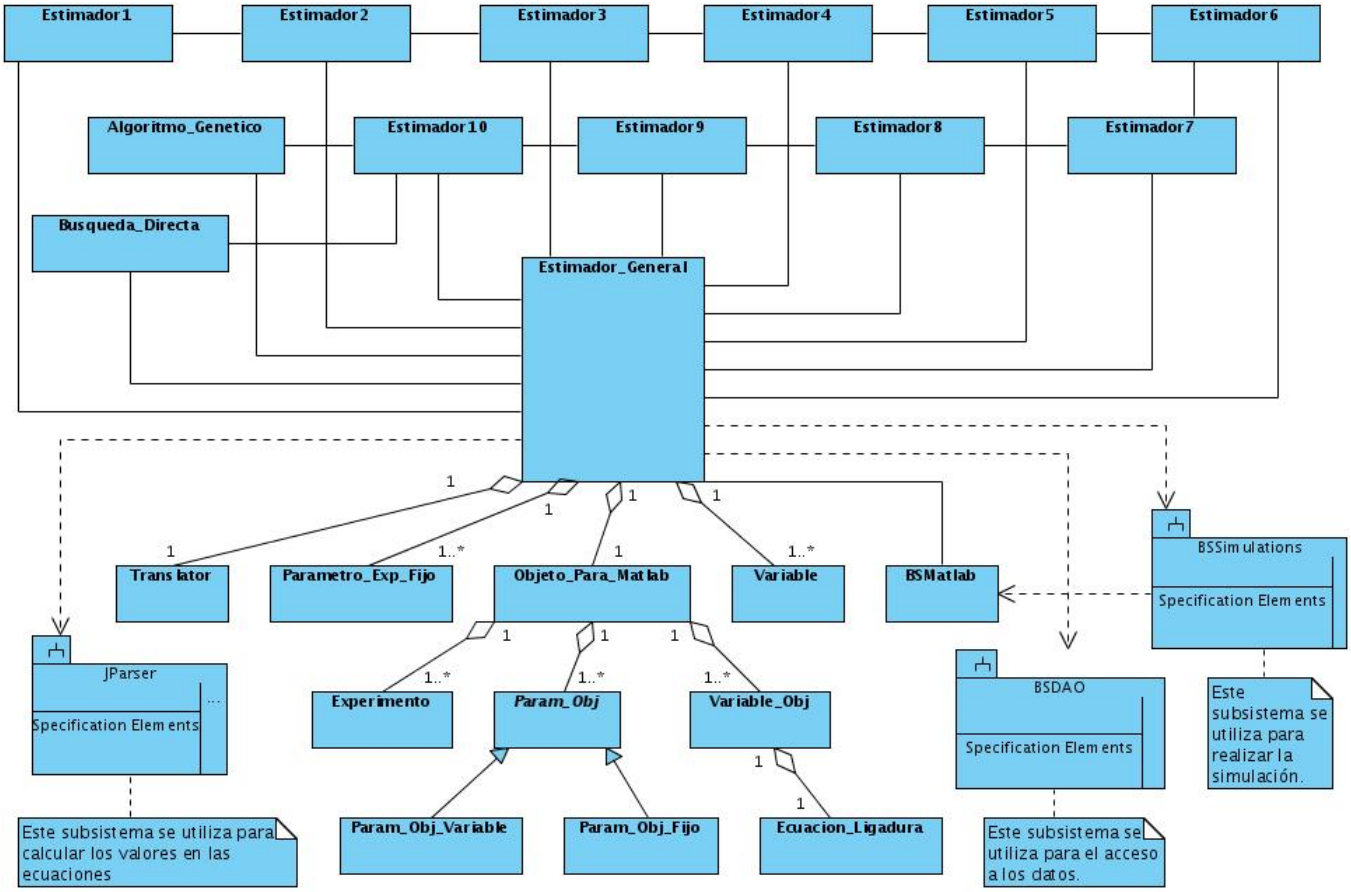

Figura 3.4 Diagrama de clases.

Los otros subsistemas que se emplean son JParser, BSDAO y BSSimulations. El JParser se utiliza para calcular los valores con las ecuaciones de ligadura. El BSDAO se emplea para acceder a los datos, cuando se requiere el acceso a los modelos guardados en la base de datos. El subsistema BSSimulations se utiliza para realizar las simulaciones.

3.4.2 PATRONES DE DISEÑO

Patrones GoF:

Mediador: Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que

funcionan como un conjunto. En este caso tenemos la clase Estimador_General que nos permite que todos los objetos estén coordinados.

Recuerdo: Permite volver a estados anteriores del sistema.

Observador: Se utiliza este patrón en el proceso de la actualización del proceso de estimación. Cuando se modifica el estado de un objeto automáticamente se modifican todos los objetos que dependen de este.

Estrategia: Permite disponer de varios métodos para resolver un problema y elegir cuál utilizar en tiempo de ejecución como Algoritmo Genético y Búsqueda Directa.

Patrones GRASP:

Experto: Como la información está encapsulada en clases, cada una es responsable controlar el acceso a esa información.

Creador: Se crean objetos de una clase para acceder a la información de esta y poder manipularla.

Bajo acoplamiento: Hay poca dependencia entre las clases del sistema.

Alta cohesión: Para que ninguna clase maneje demasiada información se ha distribuido la información.

Controlador: Con la clase Estimador_General se controla la gestión entre las clases interfaces y las clases manejadoras de datos.

3.5 IMPLEMENTACIÓN DE LOS ALGORITMOS

Los algoritmos de estimación fueron implementados haciendo uso del Matlab, como ya se explicaba en sesiones anteriores. A continuación se describen los principales algoritmos y se muestran fragmentos de código que resultan significativos dentro de los mismos.

3.5.1 GESTIÓN DEL PROCESO DE ESTIMACIÓN,

Como antes mencionamos el prototipo funcional desarrollado funciona como un wizard así que las interfaces se van mostrando consecutivamente y en el orden necesario para realizar correctamente la estimación de parámetros.

Las primeras 3 interfaces se encargan de solicitarle al cliente todos los datos experimentales, es decir: número de experimento, variables y parámetros fijos y variables, así como los valores de cada uno de ellos.

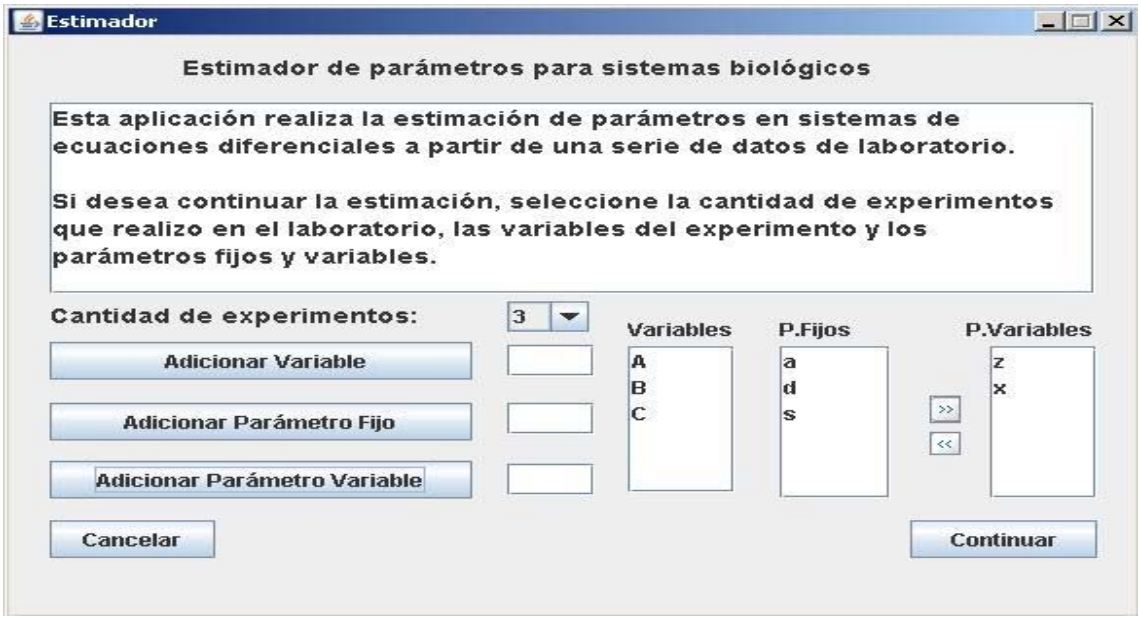

Figura 3.5.1 Entrada de los parámetros, variables y de la cantidad de experimentos.

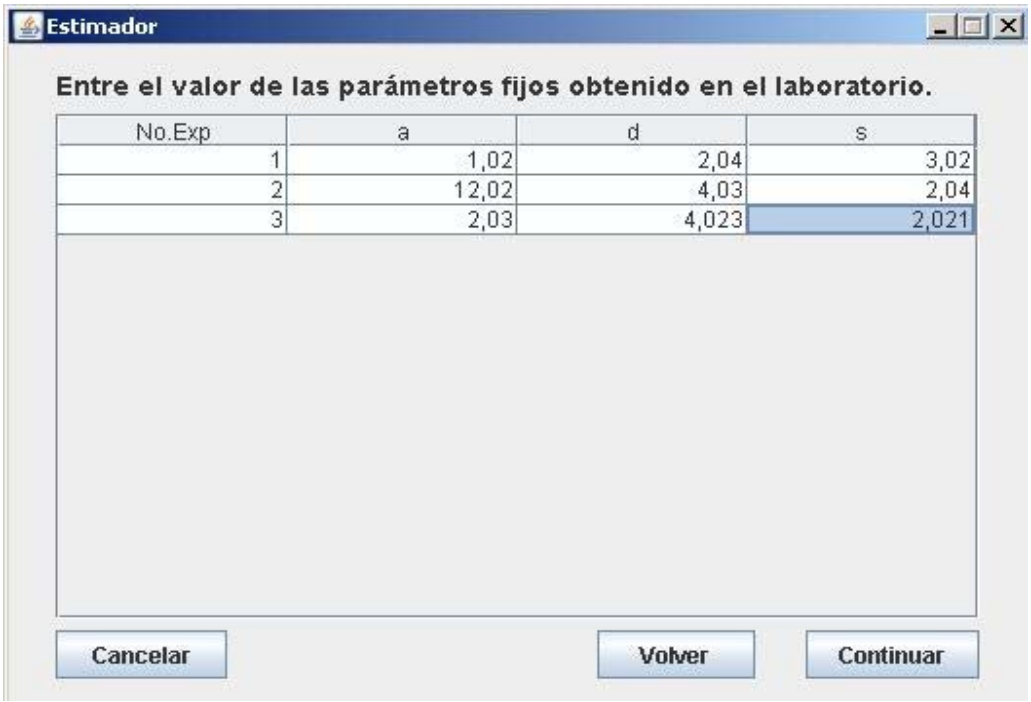

Figura 3.5.2 Entrada de los valores de los parámetros fijos medidos experimentalmente.

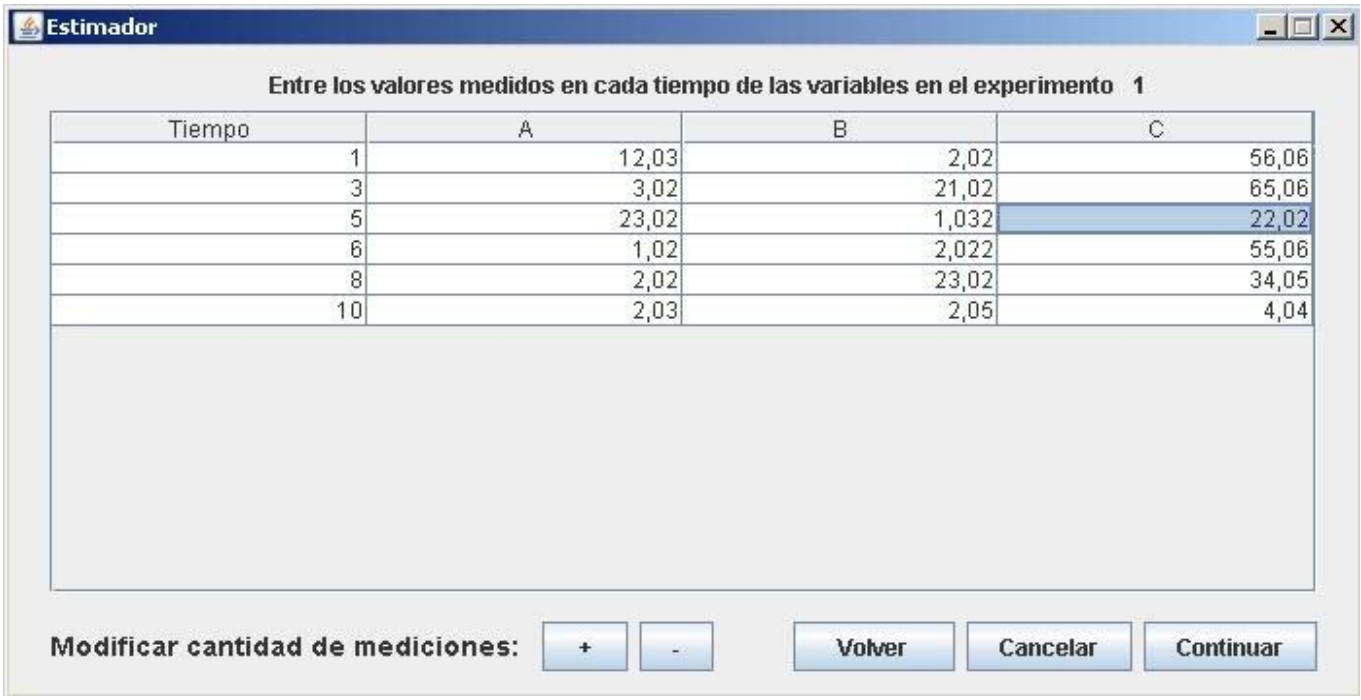

Figura 3.5.3 Entrada de los valores de las variables para cada simulación en cada experimento.

En las siguiente seis interfaces se selecciona un modelo matemático, se definen los parámetros variables de este modelo y se entran las ecuaciones de ligadura para calcular los datos del modelo utilizando los datos experimentales, como muestran las figuras siguientes:

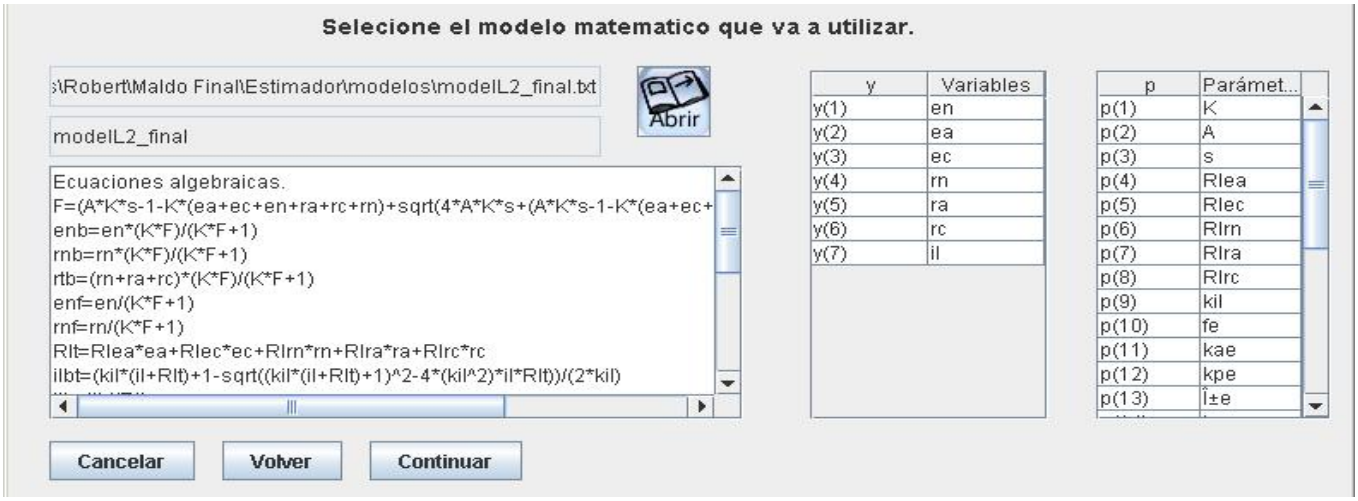

Figura 3.5.4 Selección del modelo matemático.

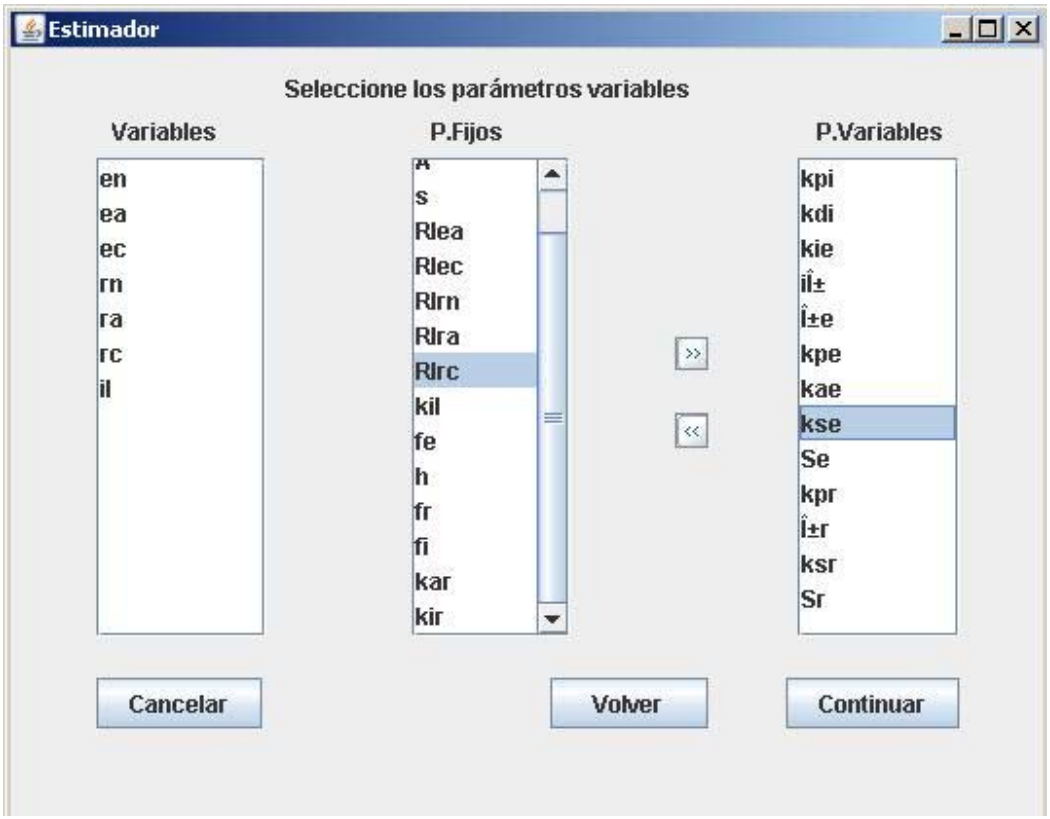

Figura 3.5.5 Selección de los parámetros variables del modelo.

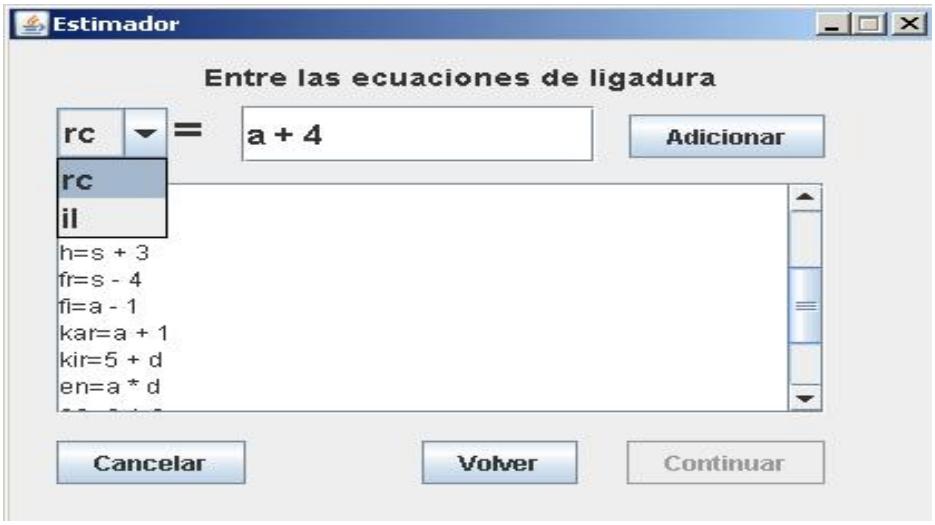

Figura 3.5.6 Entrada de ecuaciones de ligadura para transformar los valores fijos del modelo.

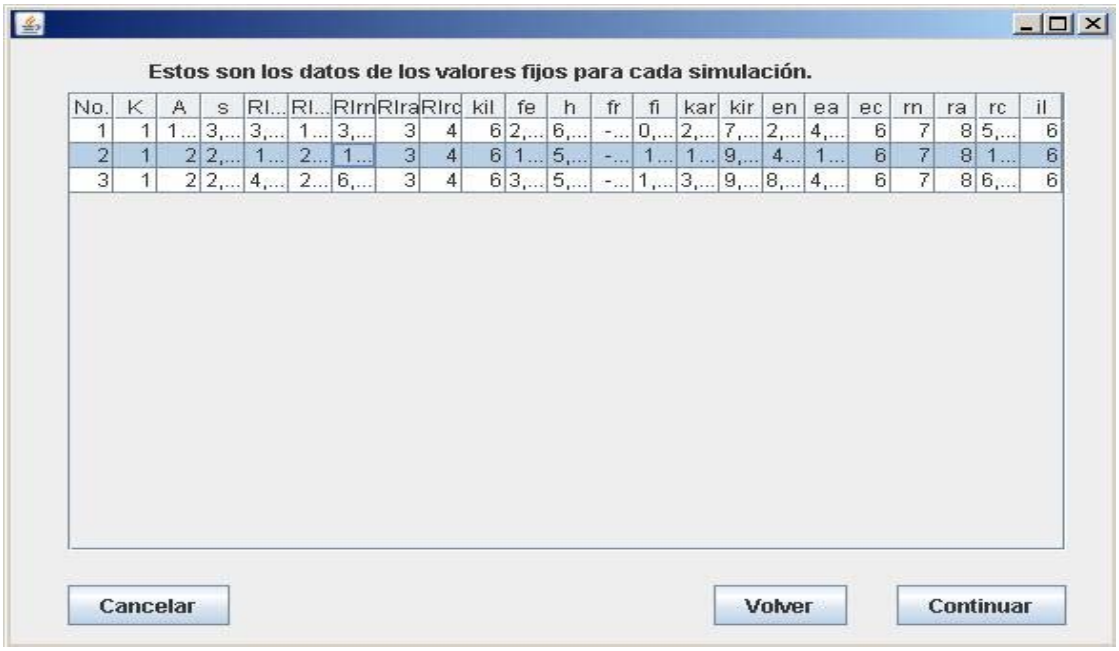

Figura 3.5.7 Muestra de los datos de los valores fijos transformados por las ecuaciones.

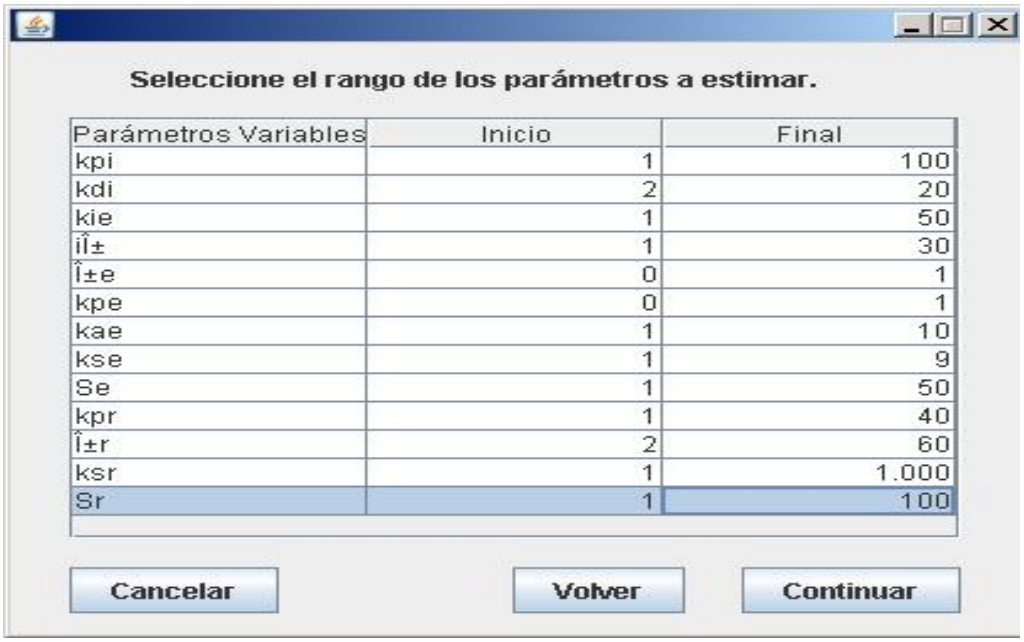

Figura 3.5.8 Entrada de los rangos de los parámetros variables.

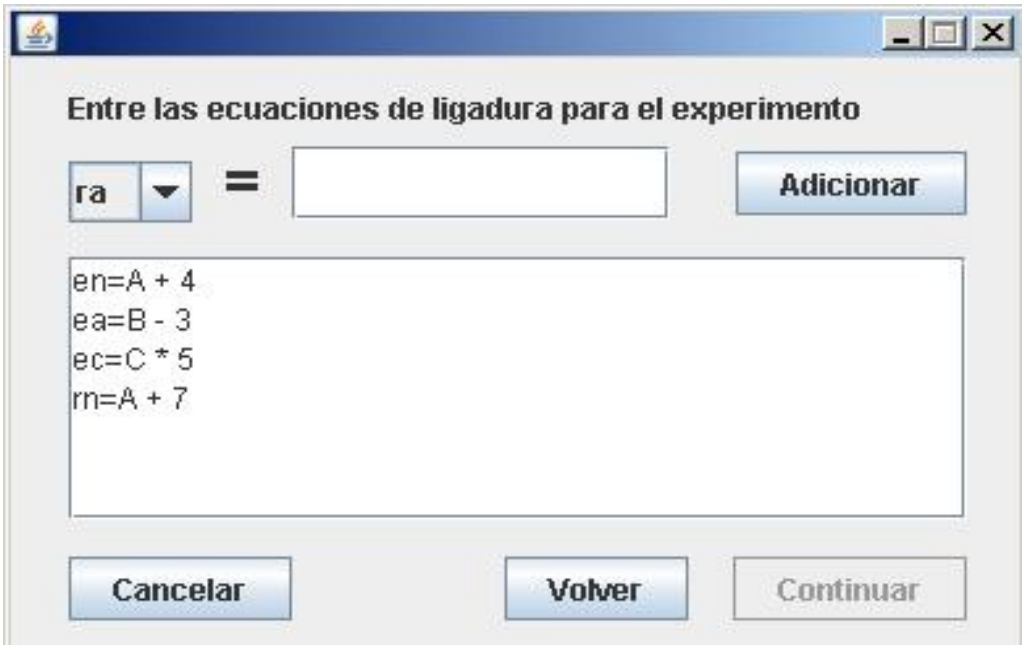

Figura 3.5.9 Entrada de las ecuaciones de ligadura para transformar los valores variables.

Las siguientes interfaces se emplean para seleccionar que algoritmo se desea utilizar y definir las opciones para el algoritmo seleccionado.

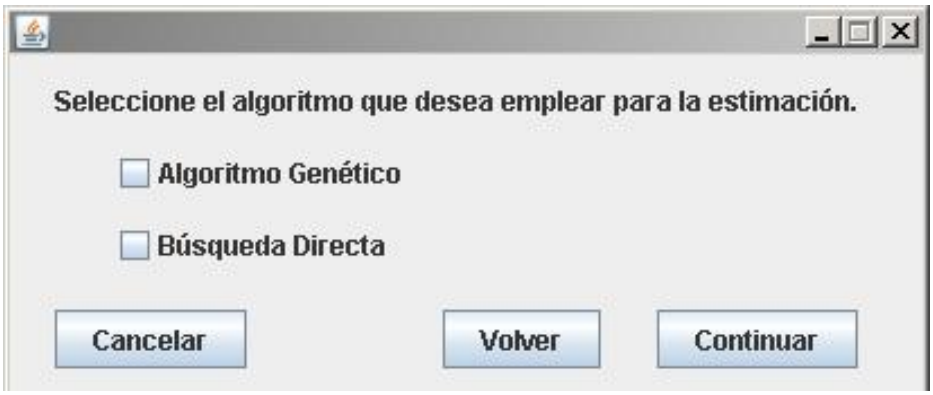

Figura 3.5.Selección del algoritmo de estimación que se va a utilizar.

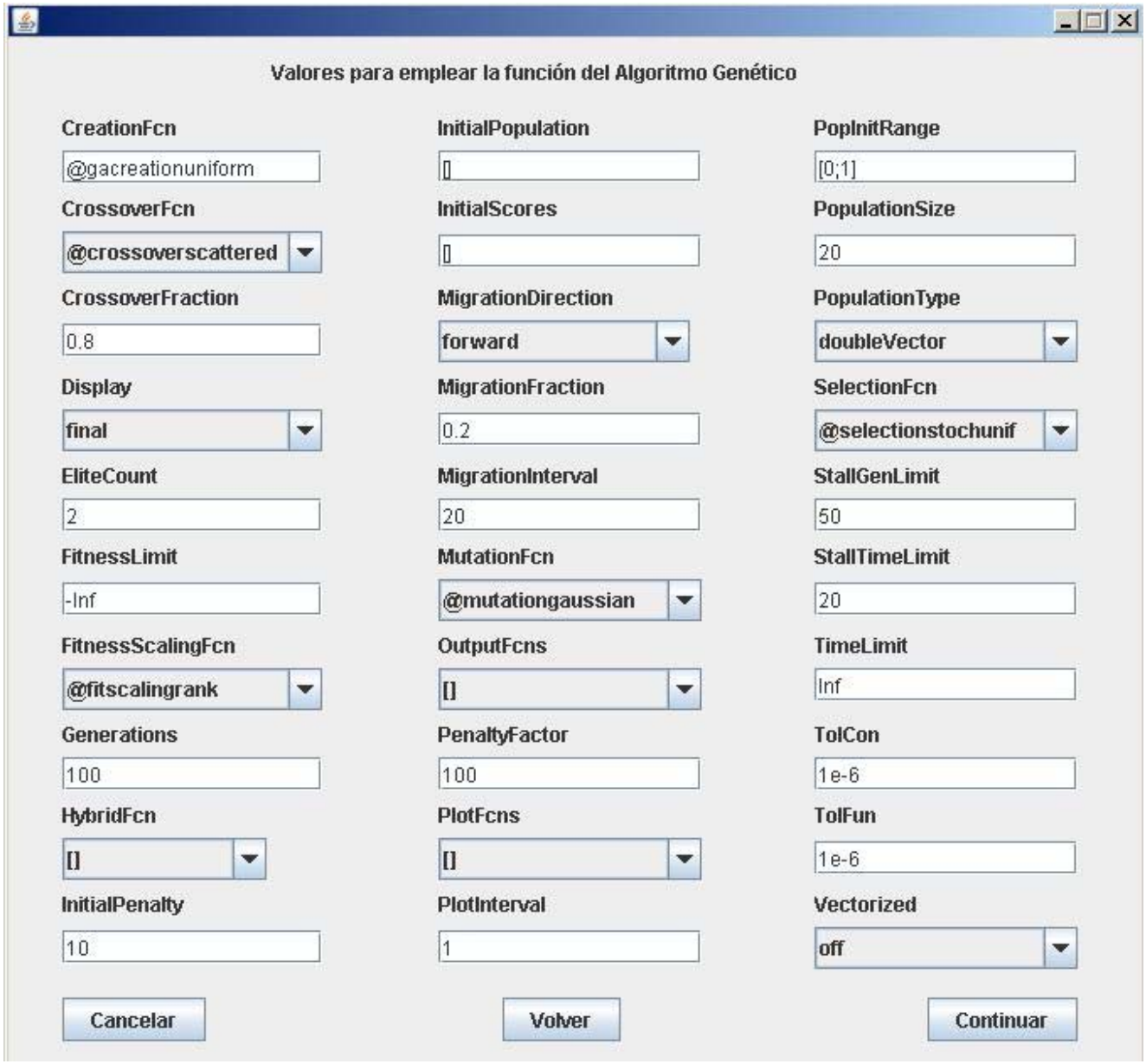

Figura 3.5.11 Definición de las opciones del algoritmo genético.

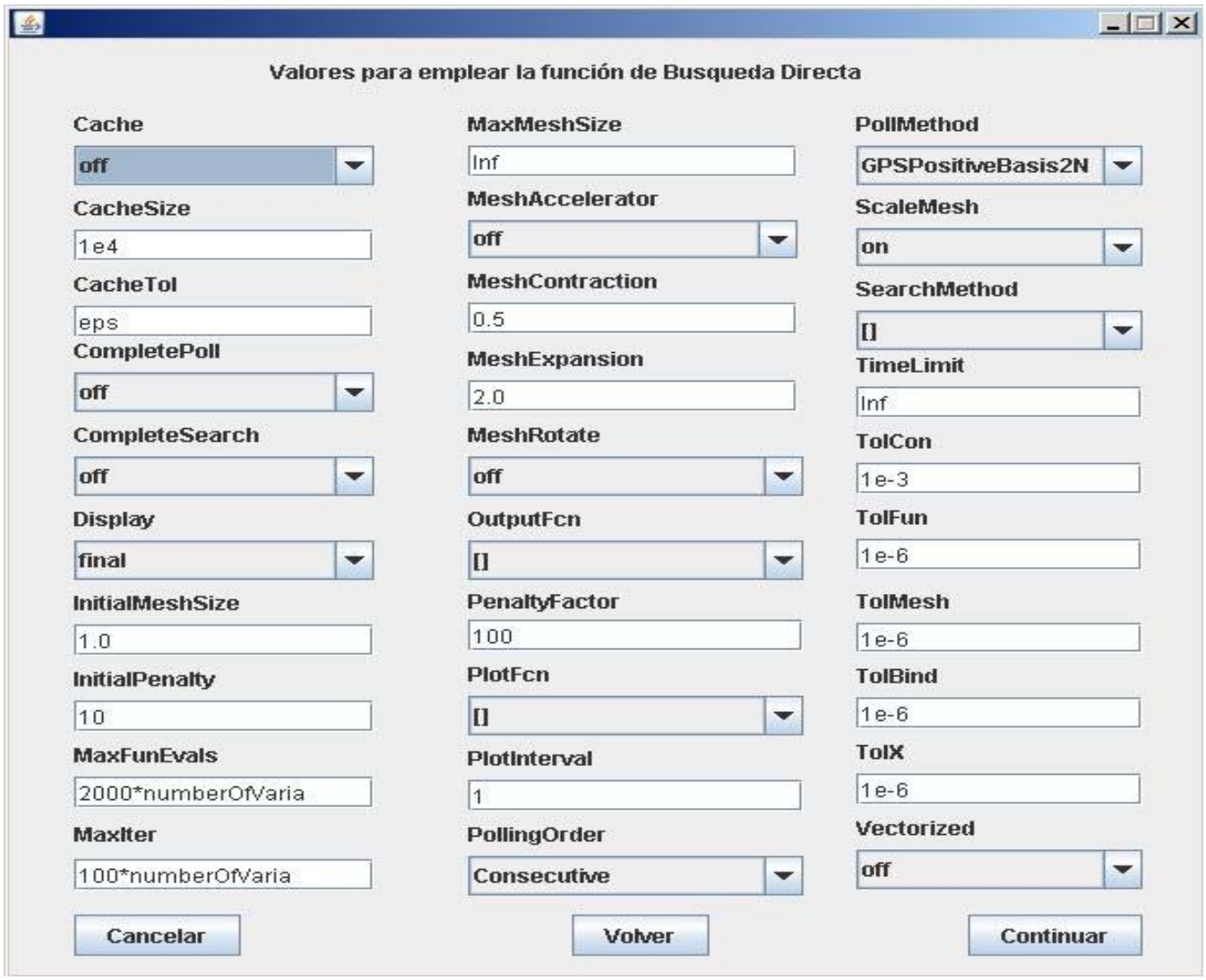

Figura 3.5.12 Definición de las opciones de búsqueda directa.

3.5.2 CONEXIÓN DE JAVA Y MATLAB

Para realizar la conexión entre Java y Matlab se hace uso de una clase denominada Engine. Para eso esta clase cuenta con cinco funciones principales, que son:

- Abrir el Matlab.
- Cerrar el Matlab.
- Enviar datos al Matlab.

- Recibir datos de Matlab.
- Ejecutar comandos del Matlab

3.5.3 FUNCIÓN OBJETIVO

Como se explicaba anteriormente la función objetivo para el caso de los dos algoritmos seleccionados obedece a la función mostrada en la sesión 2.3.2. La implementación de la misma en Matlab se muestra a continuación:

```
%% Funcion encargada de calcular la distancia entre los datos experimentales y las simulaciones correspondientes
function [fitness] = objetivefun (x, n, method, sistema, options)
%% Inicializacion de variables
fitness = 0;
exp_ic = csvread('exp_ci.csv'); % Condiciones iniciales de todos los experimentos
exp_param = csvread('exp_param.csv'); % Parametros de todos los experimentos.
% En el fichero de condiciones inciales y en el de parametros, aquellos que se quieran estimar tienen valor NaN
for i = 1 : n % Ciclo que recorre el numero de experimentos
    txt = ['exp_data' , int2str(i), '.csv']; % Fichero de datos del experimento i
    exp_data = csvread(txt);
    t_data = exp_data(:, 1);
    % Se sustituyen los NaN por los valores generados por el algoritmo genetico en cada iteracion
    count = 1;
    a = size (exp_ic);
    for j = 1 : a(2)
        if (isnan(exp_ic(i, j)))
            exp_ic(i, j) = x(count);
            count = count + 1;
        end
    end
    a = size (exp_param);
    for k = 1 : a(2)
        if (isnan(exp_param(i, k)))
            exp_param(i, k) = x(count);
            count = count + 1;
        end
    end   % Se realiza la integracion para cada experimento con los valores de ci y parametros ya modificados
    try
        [t, x] = eval([method,'(@',sistema,',[',mat2str(t_data),'],[',num2str(exp_ic(i, :)),'],options,[',num2str(exp_param(i, :)),'])']);
        sim_data = [t x];
        fitness = fitness + distancia(exp_data, sim_data); %Se calcula el fitness haciendo uso de la funcion distancia
    catch
        fitness = fitness + 1e8;
    end
end
```

Figura 3.5.1 Código de la función objetivo de Matlab.

```
%% Funcion de distancia entre dos series temporales

function [distan] = distancia (exp_data, sim_data)
a = size(exp_data);
distan = 0;

for l = 2 : a (2)
    for m = 1 : a (1)
        distan = distan + ((exp_data(m, l) - sim_data(m, l))^2);
    end
end
```

Figura 3.5.2 Código de la función distancia

Dentro de la definición de esta función existen realmente dos funciones, la función objetivefun y la función distancia. Dentro de objetivefun se ejecutan, entre otras operaciones, un ciclo que recorre el número de experimentos y dentro de este ciclo se llama a la función distancia que es la encargada de evaluar la distancia entre la serie temporal experimental y simulada.

3.5.4 ALGORITMOS GENÉTICOS

Aunque el Matlab tiene una función llamada ga() que es la que permite ejecutar algoritmos genéticos se ha implementado una función nombrada myga() que utiliza ga() pero está enfocada al caso particular de estimación de parámetros en sistemas de ecuaciones diferenciales, que es el problema que se quiere resolver en el presente trabajo. El código de dicha función es el que se muestra a continuación:

```
function [record] = myga (n, method, sistema, NVARS, odeoptions, gaoptions)

% Se crean las variables que recibe la funcion objetivefun

n
method
sistema
options = odeoptions;
options1 = gaoptions;

% Se crea in line la funcion fitfun1

fitfun1 = @(x) objetivefun(x, n, method, sistema, options);
record = [];

% Se inicia el algoritmo genetico

[x fval] = ga(fitfun1, NVARS,[],[],[],[],[],[],[],options1);
record = [record; fval];
```

Figura 3.5.3 Código de la función myga.

Esta función inicializa las variables necesarias para resolver el problema de estimación mediante algoritmos genéticos, crea los ficheros vacios donde se almacenarán todos los resultados, con el objetivo de poder brindarle al usuario un grupo de resultados y no solo el mejor valor obtenido, inicializa *in line* (en tiempo de ejecución) la función objetivo, pues la misma se sale del marco que normalmente define el Matlab para problemas de optimización, y se ejecuta el algoritmo genético.

3.5.5 BÚSQUEDA DIRECTA

La implementación de la función búsqueda directa es exactamente igual que la función myga(), salvo que esta recibe un punto inicial (x0) y myga() recibe NVARS (numero de variables a estimar), por lo que no se hará una detallada descripción de la misma.

```
function [record] = mypatternsearch(n, method, sistema, x0, odeoptions, gaoptions)

% Se crean las variables que recibe la funcion objetivefun

n
method
sistema
options = odeoptions;
options1 = gaoptions;

% Se crea in line la funcion fitfun1

fitfun1 = @(x) objetivefun(x, n, method, sistema, options);
record = [];

% Se inicia el algoritmo genetico

[x fval] = patternsearch(fitfun1, x0,[],[],[],[],[],[],[],options1);
record = [record; fval];
```

Figura 3.5.4 Código de la función mypatternsearch.

3.5.6 BÚSQUEDA HIBRIDA

Adicionalmente a los dos algoritmos propuestos se realizaron estimaciones aprovechando las facilidades que brinda el Matlab de definir dentro de los algoritmos genéticos que se usen los algoritmos de búsqueda directa.

Esta estrategia resulta interesante pues los algoritmos genéticos permiten buscar mínimos globales y los de búsqueda directa mínimos locales, por lo que la estrategia hibrida debe permitir una convergencia más rápida.

No se presenta ningún código de esta estrategia porque para hacer la búsqueda hibrida basta con definir la función búsqueda directa en la interfaz que se muestra en la figura 3.5.11, donde se definen los parámetros del algoritmo genético.

3.5.7 BÚSQUEDA DE LOS MEJORES RESULTADOS

Esta función tiene como objetivo extraer de los ficheros que almacenan todos los resultados n mejores valores, donde n es un número especificado por el usuario. A continuación el código de la misma:

```
function [best_param, best_fitness] = buscar_mejores (n)

% Esta función busca las mejores n estimaciones
param = csvread('param.csv');
fitness = csvread('fitness.csv');
param(1, :) = []; % Elimino la primera fila que fue la de la creacion
fitness(1, :) = []; % Lo mismo
a = size(param)
best_param = zeros(n, a(2));
best_fitness = zeros(n, 1);
count = 1;

for i = 1 : n
    b = find(fitness == min(fitness));
    best_param(count, :) = param(b(1), :);
    best_fitness(count) = min(fitness);
    count = count + 1;
    param(find(fitness == min(fitness)), :) = [];
    fitness(find(fitness == min(fitness)), :) = [];
end
```

Figura 3.5.5 Código de la función buscar_mejores.

3.6 RESULTADOS EXPERIMENTALES

Con el objetivo de demostrar el correcto funcionamiento de los algoritmos implementados se decidió comparar los mismos con los resultados que se obtendrían con la librería SBML-PET que es una de las más utilizadas para realizar estimación de parámetros en este tipo de problemas y de la cual se tienen varios ejemplos. Además se ha escogido como ejemplo para la comparación el modelo matemático de Michaelis-Menten, que describe la cinética enzimática.

El modelo Michaelis-Menten comprende cuatro especies moleculares, es decir, la enzima, E, el sustrato, S, el producto, P y el intermedio heterodimer, ES. Las reacciones se listan a continuación:

$$E + S \underset{k_2}{\overset{k_1}{\Leftrightarrow}} ES \xrightarrow{k_3} E + P$$

El SED consiste en estas 4 ecuaciones diferenciales:

$$\frac{d[E]}{dt} = k_2[ES] + k_3[ES] - k_1[E][S]$$

$$\frac{d[S]}{dt} = k_2[ES] - k_1[E][S]$$

$$\frac{d[ES]}{dt} = k_1[E][S] - k_2[ES] - k_3[ES]$$

$$\frac{d[P]}{dt} = k_3[ES]$$

3.6.1 MODELO MATEMÁTICO EN MATLAB

A continuación el código fuente en Matlab del modelo matemático anteriormente mencionado.

```
function [r] = mme(varargin)
%Crear un vector de ceros con tantas componentes como poblaciones existan
r = zeros(4,1);
%Inicialización de valores iniciales de las poblaciones y parametros
y = varargin{2};
p = varargin{3};
%Ecuaciones algebraicas que tienen alguna implicación sobre el sistema de ecuaciones diferenciales
%Sistema de ecuaciones diferenciales en formato Matlab
r(1) = p(2)*y(3) + p(3)*y(3) - p(1)*y(1)*y(2);
r(2) = p(2)*y(3) - p(1)*y(1)*y(2);
r(3) = p(1)*y(1)*y(2) - p(2)*y(3) - p(3)*y(3);
r(4) = p(3)*y(3);
%Validación para que la corrida se detenga si alguna población tiende a infinito
if isequal(isfinite(r),true(size(r))) == false
error(lastwarn)
end
```

Figura 3.6 Código del modelo Michaelis-Menten en Matlab.

Esta función inicializa los valores de condiciones iniciales y parámetros y devuelve para cada iteración del método de Runge-Kutta el valor del sistema de ecuaciones diferenciales descrito por Michaelis-Menten.

3.6.2 RESULTADOS DE LA ESTIMACIÓN

Para realizar la comparación con lo reportado en la literatura se corrió el mismo problema, estimando los mismos parámetros y trabajando con los mismos juegos de datos experimentales. Estos datos se relacionan a continuación:

ID Parámetros	Valor Mínimo	Valor Máximo
E	0.1	1000
S	0.1	1000
k1	0.001	1000
k2	0.001	1000
k3	0.001	1000

Figura 3.6.1 Rango de los parámetros a estimar.

Tiempo	S	SD	P	SD
1	0.924055	0.04620275	4.94519	0.2472595
2	0.268851	0.01344255	7.84506	0.392253
3	0.0963395	0.004816975	9.11061	0.4555305
4	0.0372971	0.001864855	9.63658	0.481829
5	0.0148637	0.000743185	9.85204	0.492602
6	0.00599125	0.000299563	9.93985	0.4969925
7	0.00242596	0.000121298	9.97556	0.498778
8	0.00098412	0.000049206	9.99007	0.4995035
9	0.000399518	1.99759E-05	9.99597	0.4997985
10	0.00016224	0.000008112	9.99836	0.499918

Figura 3.6.2 Datos experimentales para las ecuaciones de Michaelis-Menten.

En las gráficas siguientes se pueden observar los resultados para el modelo Michaelis-Menten de los algoritmos seleccionados y del SBML-PET. Se puede apreciar la efectividad de estos algoritmos y establecer una comparación con el SBML-PET.

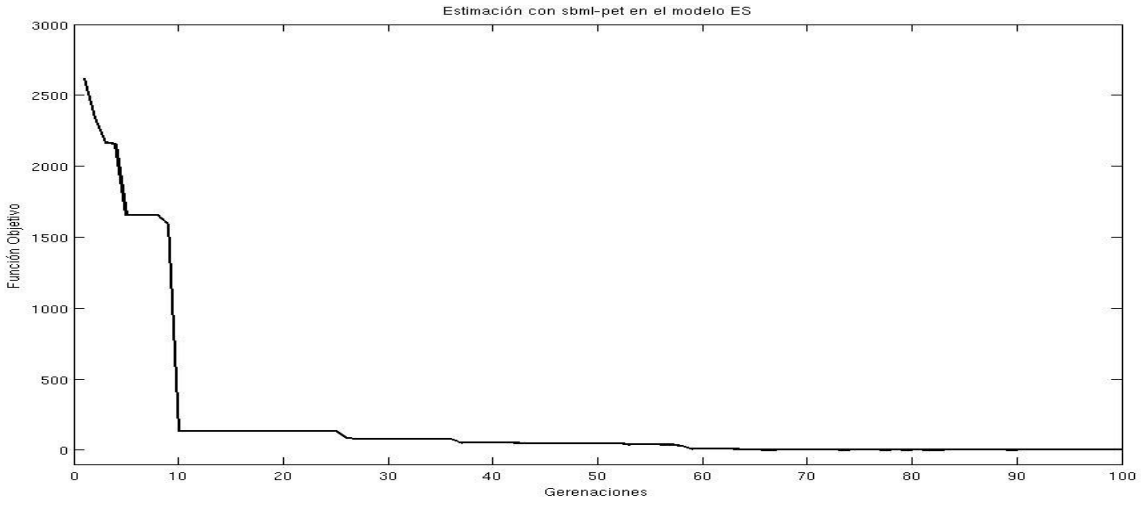

Figura 3.6.3 Estimación con SBML-PET en el modelo ES.

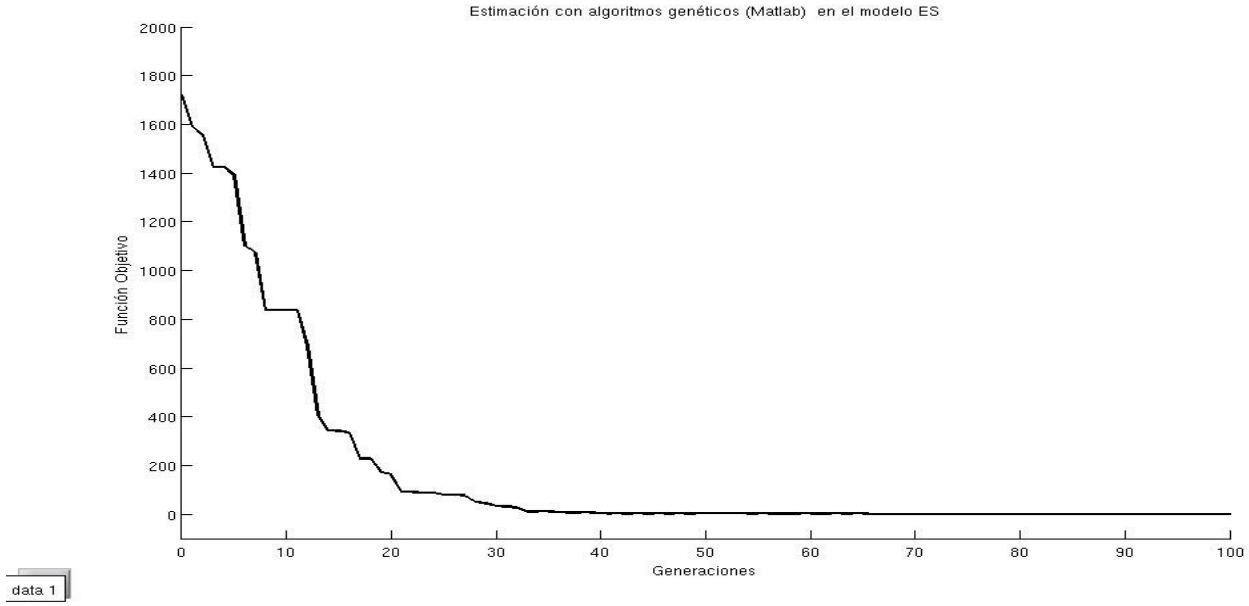

Figura 3.6.4 Estimación con algoritmo genético (Matlab) en el modelo ES.

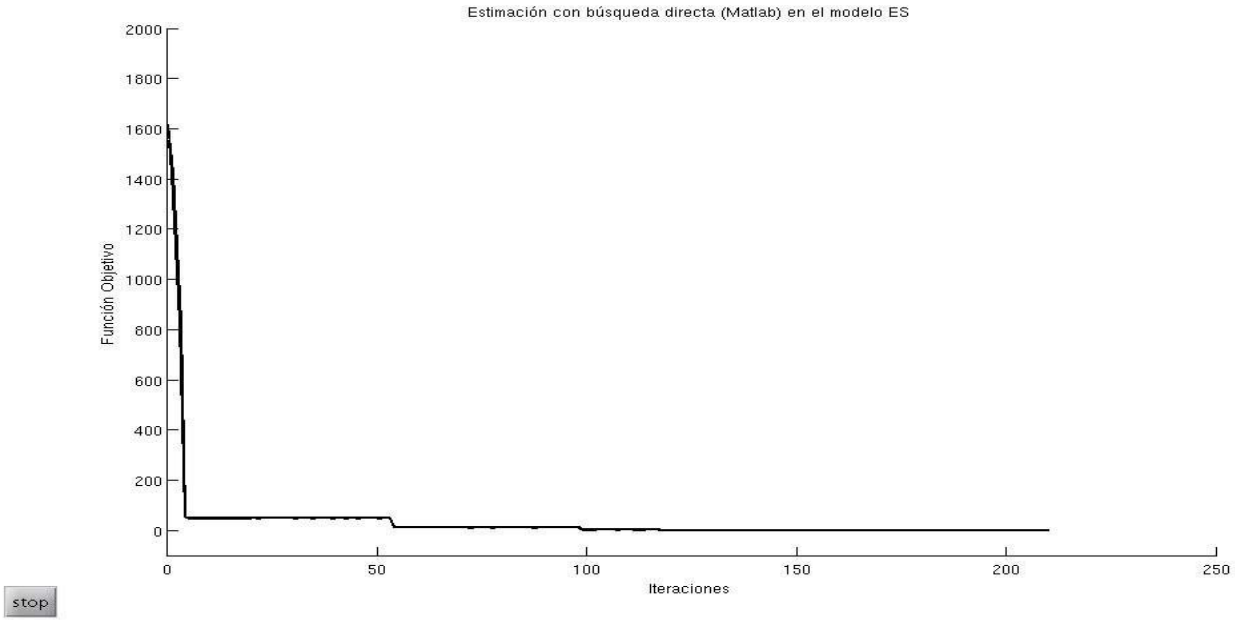

Figura 3.6.5 Estimación con búsqueda directa (Matlab) en el modelo ES.

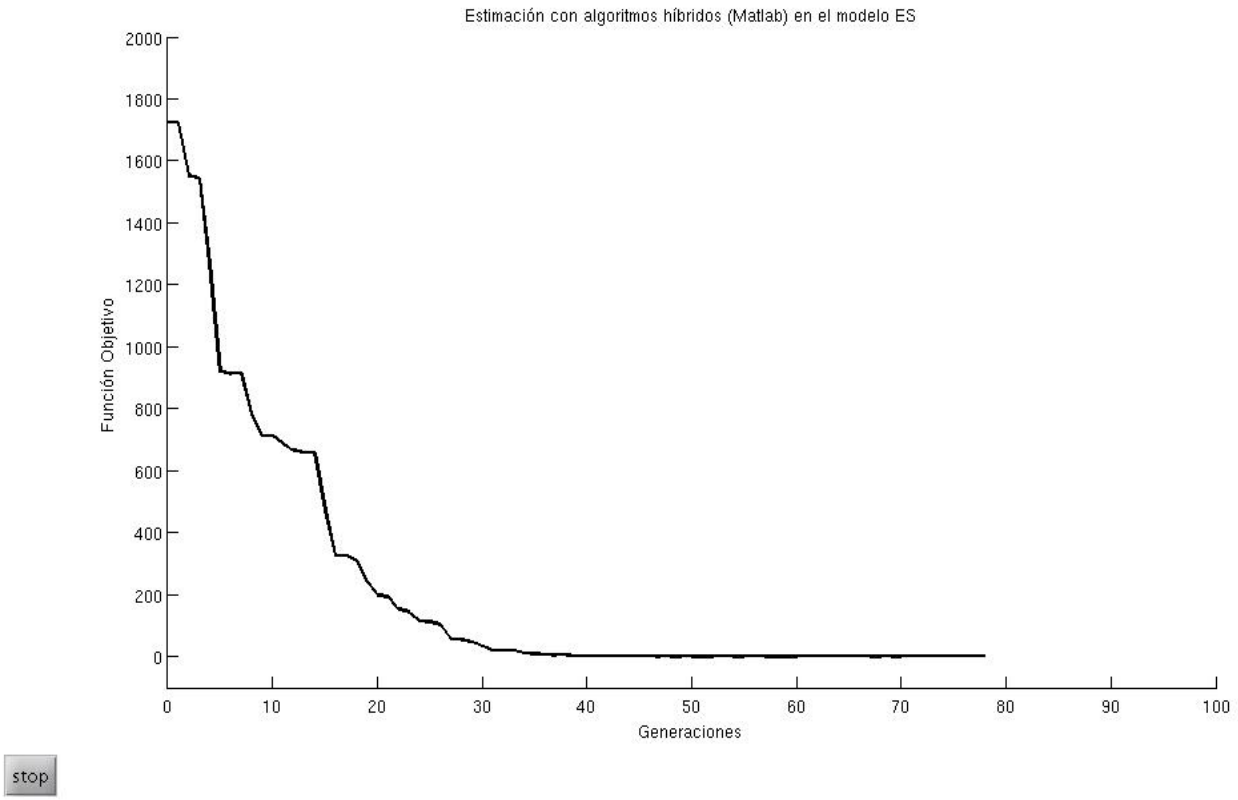

Figura 3.6.6 Estimación con búsqueda directa (Matlab) en el modelo ES.

En las cuatro gráficas se aprecia una buena convergencia de los métodos de optimización utilizados. Todos llegan a un mínimo cercano a cero. De este primer análisis se puede concluir que los algoritmos propuestos son efectivos.

En la figura 3.6.3 se observa que la convergencia ocurre alrededor de las 60 generaciones, en el caso de los algoritmos genéticos, figura 3.6.4, la convergencia es sobre las 65 generaciones y la figura 3.6.5, búsqueda directa, converge alrededor de las 120 iteraciones.

Por último la gráfica 3.6.6 muestra la convergencia del algoritmo híbrido, que ocurre sobre las 40 generaciones. En este caso no se llega a las 100 generaciones porque el algoritmo converge mucho antes, apreciándose así las ventajas de usas esta estrategia.

Teniendo en cuenta que los algoritmos genéticos en ambos casos fueron configurados para que en cada generación tuvieran 100 individuos en la población se puede concluir que el algoritmo que más rápido converge es el de búsqueda directa porque en los casos del sbml-pet y algoritmos genéticos en Matlab se hicieron 6000 y 6500 iteraciones respectivamente, mientras en la búsqueda directa solo se hicieron 120 iteraciones.

3.7 CONCLUSIONES

En este capítulo se presentó un esquema que permitió entender los procesos de los algoritmos de estimación que se investigaron a profundidad. Se explicaron los algoritmos diseñados a nivel de pseudocódigos, los diagramas de clases y patrones de diseño utilizados para desarrollar el prototipo funcional. Los resultados de los experimentos fueron discutidos y analizados desde el punto de vista de los investigadores y se demostró la calidad de los algoritmos seleccionados para la estimación de parámetros de un sistema de ecuaciones diferenciales.

CONCLUSIONES GENERALES

- Se diseñaron y se implementaron algoritmos de estimación de parámetros en un sistema de ecuaciones diferenciales.
- Se implementó un prototipo funcional, con una interfaz gráfica agradable para comprobar los algoritmos.
- Se realizó un análisis previo de la eficiencia de diferentes algoritmos y a partir de ese análisis los algoritmos seleccionados e implementados se compararon con el software SBML-PET, ratificando así el buen funcionamiento y rapidez de los mismos.

RECOMENDACIONES

- ✓ Realizar el levantamiento formal de los requerimientos de software para una distribución del sistema que permita desarrollarlo a través del ciclo de desarrollo y con las pautas que propone la metodología OpenUp.
- ✓ Incorporarle a la herramienta ha desarrollar una base de datos para los resultados de las estimaciones.
- ✓ Incorporarle a la herramienta ha desarrollar las funcionalidades de realizar la estimación de forma distribuida.
- ✓ Incorporar otros métodos para resolver las integraciones numéricas.
- ✓ Desarrollar los algoritmos Búsqueda Directa y Algoritmo Genético en Java para garantizar la independencia del producto.

REFERENCIAS BIBLIOGRÁFICAS

1. *A graphical notation for biochimical networks. Kitano, H. 5, 2003, Biosilico, Vol. 1.*
2. *Bioinformáticos. Modelos Matemáticos. [Citado el: 10 de 02 de 2008.] http://www.bioinformaticos.com.ar/articulos/biomatematica01.htm*
3. *Bioinformatic@.es Newsletter. [En línea] 06 de 03 de 2002. [Citado el: 20 de 01 de 2008.]* http://www.webzinemaker.com/digitalbiology/
4. TERRA. *La Biología de Sistemas mejorará el tratamiento de enfermedades* [*Citado* el: 26 de enero de 2008]. Disponible en: http://www.imbiomed.com/Innsz/Nnv46n4/espanol/Wnn44-07.html.
5. *Eduardo Morales Manzanares.* Recocido Simulado, *2004-11-02.* [Consultado el: 12 de marzo del 2008]. Disponible en: http://ccc.inaoep.mx/%7Eemorales/Cursos/Busqueda04/node76.html
6. Yoshiya Matsubara, Shinichi Kikuchi, Masahiro Sugimoto and Masaru Tomita. Parameter estimation for stiff equations of biosystems using radial basis function networks. 27 Abril 2006. [Citado el: 25 de marzo del 2008]. Disponible en: http://www.biomedcentral.com/1471-2105/7/230
7. Matías Ison, Jacobo Sitt, Marcos Trevisan. Algoritmos genéticos: aplicación en MATLAB, Noviembre 25, 2005. Disponible en: http://www.df.uba.ar/users/mison/genetico.tar.gz
8. Zhike Zi. SBML-PET A Systems Biology Markup Language (SBML) based Parameter Estimation Tool. [Citado el: 8 de febrero del 2008]. Disponible en: http://sysbio.molgen.mpg.de/SBML-PET/
9. Xinglai Ji. LibSRES: A C Library of Stochastic Ranking Evolution Strategy. September 6, 2005. [Citado el: 13 de febrero del 2008]. Disponible en: http://csbl.bmb.uga.edu/~jix/science/libSRES/.
10. Systems Biology Group. Grid Cellware: the first grid-enabled tool for modeling and simulating cellular processes. Noviembre 16, 2004. [Citado el: 20 de febrero del 2008].
11. MANUEL ÁLVAREZ BLANCO; ARNALDO GÓMEZ MONTENEGRO*, et al. Matemática numérica.* 2003. [Citado el: 26 de febrero del 2008].
12. Francisco M. González-Longatt. Métodos Numéricos para la Solución Ecuaciones Diferenciales: Un Ejemplo Comparativo. [*Citado* el: 10 de febrero del 2008].
13. Jorge Baier Aranda. Programación Genética. [Citado el: 9 de enero del 2008].

BIBLIOGRAFÍA

1. Noel Moreno Lemus. Tesis de Maestría. [Consultado el: 5 de febrero del 2008].
2. Zhike Zi. SBML-PET A Systems Biology Markup Language (SBML) based Parameter Estimation Tool. [Consultado el: 8 de febrero del 2008]. Disponible en:

   http://sysbio.molgen.mpg.de/SBML-PET/
3. Xinglai Ji. LibSRES: A C Library of Stochastic Ranking Evolution Strategy. September 6, 2005. [Consultado el: 13 de febrero del 2008]. Disponible en: http://csbl.bmb.uga.edu/~jix/science/libSRES/.
4. Systems Biology Group. Grid Cellware: the first grid-enabled tool for modeling and simulating cellular processes. Noviembre 16, 2004. [Consultado el: 20 de febrero del 2008]. Disponible en: https://www.bii-sg.org/sbg/cellware
5. Yoshiya Matsubara, Shinichi Kikuchi, Masahiro Sugimoto and Masaru Tomita. Parameter estimation for stiff equations of biosystems using radial basis function networks. 27 Abril 2006. [Consultado el: 25 de febrero del 2008]. Disponible en: http://www.biomedcentral.com/1471-2105/7/230
6. Daniel T. Gillespie. Exact Stochastic Simulation of Coupled Chemical Reactions. 1977. [Consultado el: 28 de febrero del 2008].
7. Jorge Baier Aranda. Programación Genética. [Consultado el: 9 de enero del 2008].
8. Jacobson, Ivar y Booch, Grady y Rumbaugh, James. El proceso unificado de software. Primera edición. Pearson Educación, S.A. 2000. [Consultado el: 5 de marzo del 2008].
9. Sitio de Metodología de Desarrollo. Ayuda extendida de OpenUp. [Consultado el: 15 de marzo del 2008]. Disponible en: http://10.34.20.5:5800/OpenUP.
10. Craig Larman. UML y Patrones. [Consultado el: 10 de abril del 2008]. Disponible en: http://biblioteca.uci.cu/bives/titdigitales.htm
11. Monografías. Matlab. [Consultado el: 19 de abril del 2008]. Disponible en: http://www.monografias.com/trabajos5/matlab/matlab.shtml.
12. NETBEANS. NetBeans [Consultado el: 16 de marzo del 2008]. Disponible en:

http://www.netbeans.org/index_es.html.

13. Juan Julián Merelo Guervós. Informática evolutiva: Algoritmos genéticos. [Consultado el: 10 de abril del 2008]. Disponible en: http://geneura.ugr.es/~jmerelo

14. *Eduardo Morales Manzanares.* Recocido Simulado, *2004-11-02.* [Consultado el: 12 de marzo del 2008]. Disponible en: http://ccc.inaoep.mx/%7Eemorales/Cursos/Busqueda04/node76.html

15. Leah Edelstein-Keshet. Mathematical models in biology, 1988.

16. PARADIGM, V. *Documentation* [Consultado el: 18 de febrero del 2008]. Disponible en: http://www.visual-paradigm.com/.

17. RO, C. *Modelo matemático en medicina y biología. Bases teóricas y fundamentos* [Consultado el: 16 de enero del 2008]. (Revista Invest Clin). Disponible en: http://www.imbiomed.com/Innsz/Nnv46n4/espanol/Wnn44-07.html.

18. TEJADA, D. H. Disponible en: http://teleprogramadores.com.

19. Carlos E. Cuesta. Patrones de Diseño. Consultado el: 26 de marzo de 2008].

20. Francisco M. González-Longatt. Métodos Numéricos para la Solución Ecuaciones Diferenciales: Un Ejemplo Comparativo. [Consultado el: 15 de abril del 2008].

21. Pedro Hernandez. Solución numérica de las ecuaciones de movimento, varios métodos Runge-Kutta, 2006-02-20. [Consultado el: 22 de abril del 2008].

ANEXOS

GLOSARIO DE TÉRMINOS

- BS: Biología de Sistemas.
- SED: Sistemas de Ecuaciones Diferenciales.