UNIVERSIDAD DE EL SALVADOR

FACULTAD DE CIENCIAS NATURALES Y MATEMÁTICA

ESCUELA DE MATEMÁTICA

Trabajo de graduación:

"Diseño de un software para la solución numérica de sistemas de ecuaciones lineales, integración y aproximación"

Presentado por:

Br. Porfirio Armando Rodríguez Rodríguez

Para optar al título de:

Licenciado en Matemática

Opción Estadística-Computación

Asesores:

Lic. Manuel de Jesús Cortez Álvarez

Ing. Oscar Hernán Lemus Gómez

Ciudad Universitaria, 17 de septiembre de 2003.

UNIVERSIDAD DE EL SALVADOR

AUTORIDADES UNIVERSITARIAS.

Rectora : Dra. María Isabel Rodríguez.

Secretaria general : Licda. Lidia Margarita Muñoz.

Fiscal General : Lic. Pedro Rosalío Escobar.

FACULTAD DE CIENCIAS NATURALES Y MATEMÁTICAS.

Decana : Licda. Leticia Noemí Paúl de Flores.

ESCUELA DE MATEMÁTICA.

Director : Lic. Alfredo Aguilar González.

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE CIENCIAS NATURALES Y MATEMÁTICA

ESCUELA DE MATEMÁTICA

Trabajo de graduación:

"Diseño de un software para la solución numérica de sistemas de ecuaciones lineales, integración y aproximación"

Presentado por:

Br. Porfirio Armando Rodríguez Rodríguez

Para optar al título de:

Licenciado en Matemática

Opción Estadística-Computación

Asesores:

Lic. Manuel de Jesús Cortez Álvarez : _____

Ing. Oscar Hernán Lemus Gómez : _____

Ciudad Universitaria, 17 de septiembre de 2003.

Agradecimientos

A la Suprema Personalidad de Dios :por su misericordia al sancionar mi destino.

A mis asesores :por aceptar esta tarea y delinear su espectro.

A mi familia :a mis padres por cumplir sus deberes con voluntad, a mis hermanos por ayudar en la medida de lo posible.

A mi familia :a mi esposa por ser el faro que me indica el camino y la razón de este esfuerzo, a mis hijos por que ésto es, en gran parte, para ustedes, aúnque todavía no están aquí.

A todos aquellos anónimos :que de alguna manera contribuyeron para concretar este proyecto con sus aportes desinteresados.

Índice de contenidos

Índice de tablas

Índice de ilustraciones

Resumen

Al desarrollar el software nos enfocamos en el método del ciclo de vida para el desarrollo de sistemas (software) que consta de seis actividades que desarrollaremos en un mismo número de capítulos

Capítulo I. Investigación preliminar

En esta etapa del diseño del software se desarrolló una investigación de los métodos numéricos que interesan implementarse. Se evaluó la el tamaño del proyecto, duración y alcance y el apoyo técnico disponible. Este proceso culminó con la aprobación de la planeación del proyecto del presente trabajo de graduación.

Capítulo II. Determinación de los requerimientos

Se efectuó una investigación detallada de cómo, porqué y cuando se espera que los métodos numéricos seleccionados funcionen. Se especifican algoritmos, fundamentos matemáticos y condiciones para su aplicación. Toda esta investigación profunda junto con las recomendaciones de profesores y las necesidades de estudiantes se analizaron para que al final se tenga una lista de características que el software a desarrollar debe tener.

Capítulo III. Diseño lógico del software

Se especifican las características de diseño que serán retomadas en la etapa de desarrollo del software como diseño de pantallas, diseño del menú principal y los algoritmos que especifican cómo funciona cada uno de los métodos, los datos que utilizan como entradas y la salida que producirán, esto con la finalidad de satisfacer los requerimientos establecidos en la etapa anterior.

Capitulo IV. Desarrollo del software (diseño físico)

Se pone de relieve las técnicas y herramientas al construir la aplicación. Se ejemplifica el desarrollo tomando como base un método numérico completo. Con el propósito de observar la fiabilidad de la aplicación se dispone de todo el código en un proyecto construido en Visual Basic 6.0 y distribuido gratuitamente en un CD (ver anexo). El usuario entendido en

programación puede acceder a este CD y estudiar el código. El código esta documentado con comentarios para esclarecer ciertas partes del mismo.

Capitulo V. Prueba del software

Se pone a prueba el software con ciertos problemas cuyas respuestas conocemos con cierto grado de exactitud para comprobar su desempeño. Se han seleccionado problemas para todos los métodos desarrollados.

Capítulo VI. Implantación del software

La implementación o utilización del software diseñado, se propone a través de un manual de usuario que incluye los siguientes aspectos:

- Requisitos mínimos del software
- Instrucciones para la Instalación del software
- Descripción de las características del software con ejemplos típicos sobre su uso.

Introducción

Los métodos numéricos son algoritmos de los cuales se obtienen uno o varios valores numéricos. Los valores que se obtienen por un método numérico se llaman soluciones numéricas y se dice que se obtienen numéricamente. Las soluciones numéricas, a diferencia de las soluciones analíticas que se obtienen usando fórmulas matemáticas o científicas, generalmente no son exactas. Desdichadamente no todos los fenómenos físicos pueden representarse a través de una fórmula cuya solución analítica esté disponible. En este caso se hace evidente la implementación de un método numérico para obtener una solución con una precisión deseada. En esta era de las computadoras la implementación de los métodos numéricos en estas máquinas significa grandes beneficios para un científico o ingeniero ya que obtiene sus respuestas de una manera eficiente y con una rapidez impresionante.

El presente trabajo de graduación pretende poner a disposición de docentes, estudiantes, científicos, investigadores interesados, un software que le sirva de herramienta para implementar métodos numéricos en problemas que caen en las áreas de sistemas de ecuaciones lineales, ecuaciones diferenciales ordinarias, aproximación e integración con una base teórica sólida y didáctica que sustenta su aplicación y con un análisis del error cometido en los cálculos. Los métodos numéricos que se desarrollarán en el área de ecuaciones lineales son los siguientes: Iteraciones de Jacobi, Iteraciones de Gauss-Seidel, métodos de sobrerrelajación sucesiva (SOR); en el área de ecuaciones diferenciales ordinarias: el método de Runge Kutta-Fehldberg y el método de diferencias finitas; en el área de integración el algoritmo de Romberg, Cuadratura de Gauss-Legendre; y en el área de aproximación de funciones el método de la transformada rápida de Fourier.

Este software está orientado para el usuario de nivel principiante, intermedio y experto. El principiante puede aprender sobre las bases teóricas de ciertos métodos de interés, el intermedio puede aplicar los métodos mencionados y obtener resultados, y el experto lo utiliza como herramienta de análisis numérico en diversos problemas aplicados de su área.

Para su uso fácil y ameno se ha desarrollado el software en un ambiente completamente Visual (íconos, barras de herramientas y otras herramientas visuales) que le es familiar a la mayoría de usuarios. Es un software que tiene las características que poseen la mayoría de

aplicaciones para el sistema operativo Windows. Además se ha utilizado un lenguaje de programación con herramientas de desarrollo rápido como las que proporciona Visual Basic. De este modo se tiene un software robusto y de calidad.

I. Investigación preliminar

1.1. Introducción

En la etapa de investigación preliminar se determinó, basándose en consultas a docentes en esta especialidad, los métodos numéricos a desarrollar. Además, se tomó en cuenta el tiempo y los recursos disponibles para llevar a cabo el proyecto. Se definieron los objetivos a alcanzar y al final de esta etapa se presentó un documento que constituye la planeación del trabajo de graduación.

1.2 Aclaración y comprensión del proyecto

1.2.1. Planteamiento del Problema

Generalidades

Hoy en día el software sobre métodos numéricos puede ser abundante y de muy variada calidad. De este hecho puede surgir la pregunta ¿Es necesario otro software más?.

La respuesta es sí. La existencia de software en el mercado no implica la aversión a desarrollar uno más con características propias orientado al auxilio de investigadores en el área de las ingenierías y las ciencias, de docentes como herramienta didáctica, y estudiantes interesados en los fundamentos teóricos de distintos métodos numéricos que en un futuro tendrán que aplicar en sus áreas de estudio. Lo anterior es cierto, puesto que el software existente de este tipo no se encuentra a disposición inmediata, no son muy especializados o tienen un costo considerable.

Especificidad

Desarrollar un software sobre métodos numéricos para su aplicación en problemas de ingeniería y ciencias con fundamentos teóricos a disposición del ingeniero o científico que los aplica, además se presentará un análisis del error cometido en los cálculos realizados por computadora.

- En problemas que impliquen sistemas de ecuaciones lineales se desarrollarán los siguientes métodos:
  - Iteraciones de Jacobi

- Iteraciones de Gauss-Seidel
- Método de sobrerrelajación sucesiva (SOR)

❖ En problemas que impliquen ecuaciones diferenciales ordinarias se tratarán los siguientes:

- El método de Runge-Kutta-Fehldberg
- Método de diferencias finitas.

❖ En los que impliquen integración se abordaran los métodos:

- Algoritmo de Romberg
- Cuadratura de Gauss-Legendre

❖ Y en problemas de aproximación:

- Transformada rápida de Fourier

Utilizaremos la interfaz gráfica del sistema operativo Windows como plataforma de desarrollo y herramientas de desarrollo rápido, específicamente, Visual Basic. Con lo anterior proponemos desarrollar un sistema de soluciones con herramientas visuales que son familiares para la mayoría de los usuarios como lo son barra de herramientas, íconos, y un subsistema de ayuda y otros elementos característicos de cualquier aplicación en Windows.

Además de ser un software que se utilice como una herramienta para la investigación científica en problemas que no pueden resolverse de manera analítica sino que implican análisis numérico para su resolución, servirá además como una fuente de información teórica didáctica acerca de los métodos numéricos que antes mencionamos. El principiante consulta y el experto analiza y aplica, es la filosofía del software a diseñar.

1.3. Antecedentes y justificación

En la actualidad se venden muchos paquetes de computación para aproximar las soluciones numéricas de los problemas. Por un lado encontramos programas de propósito especifico, elaborados especialmente por estudiantes de ingeniería o carreras afines o que vienen en libros de análisis numérico, que darán resultados satisfactorios en la mayoría de problemas que deba resolver. Por otro lado están los programas de propósito general que encontramos en las bibliotecas de subrutinas matemáticas comunes. Los programas de propósito general ofrecen

medios para reducir los errores atribuibles al redondeo de las computadoras, el subflujo (underflow) y al sobreflujo o desbordamiento (overflow)

En 1971, se desarrollaron los procedimientos de ALGOL en los cálculos de matrices. Se desarrolló un paquete de subrutinas FORTRAN basado principalmente en los procedimientos de ALGOL y se incorporó en las rutinas de EISPACK. Las subrutinas de FORTRAN sirven para calcular los valores y vectores característicos de varias clases de matrices. El proyecto EISPACK fue el primer programa numérico a gran escala que se puso al servicio del dominio público, siendo imitado después por muchos otros.

LINPACK es un paquete de subrutinas de FORTRAN que sirve para analizar y resolver sistemas de ecuaciones lineales y para resolver los problemas lineales de mínimos cuadrados.

El paquete LAPACK, que se puso en venta en 1999, es una biblioteca de subrutinas de FORTRAN que sustituyen a LINPACK y EISPACK, al integrar estos dos conjuntos de algoritmos en un paquete unificado y actualizado.

IMSL (International Mathematical Software Library) contiene las bibliotecas MATH, STAT y SFUN para matemáticas, estadísticas y funciones especiales del análisis numérico, respectivamente. IMSL contiene métodos para los sistemas lineales, análisis de sistemas característicos, interpolación y la aproximación, integración y la diferenciación, ecuaciones diferenciales, transformaciones, ecuaciones no lineales, optimización y las operaciones básicas de matrices / vectores.

El Numerical Algorithms Group (NAG) se fundó en 1971 en el Reino Unido. Ofrece más de 1100 subrutinas en una biblioteca de FORTRAN 77 para más de 90 computadoras diferentes. NAG C ofrece muchas de las rutinas que vienen en la biblioteca de FORTRAN. La biblioteca ofrece rutinas para efectuar la mayor parte de las tareas comunes del análisis numérico en una forma similar a las rutinas de IMSL.

Los paquetes de IMSL Y NAG están destinados a matemáticos, científicos e ingenieros que desean llamar desde un programa las subrutinas de FORTRAN de alta calidad.

Existen otros paquetes como MATLAB que es como su nombre lo sugiere un programa de laboratorio de matrices y GAUSS que es un sistema matemático y estadístico. Ambos ofrecen

integración / diferenciación, sistemas no lineales, transformadas rápidas de Fourier y gráficas y se basan en EISPACK y en LINPACK. Aunque el lenguaje incorporado se asemeja a Pascal, no es posible llamar las rutinas externas que contienen subprogramas compilados en FORTRAN o en C.

Es necesario distinguir los paquetes de cómputo simbólico (como Derive, Mathematica, Maple) de los paquetes de análisis numérico (como los que mencionamos aquí)

Sin embargo, pese a que existe software en esta materia, también existen buenas razones para desarrollar uno que tenga características propias. Entre muchas razones podemos plantear las siguientes, las cuales creemos tienen mayor peso:

La importancia del proyecto

En la actualidad existe software sobre este tema, como se puede apreciar con la información que arriba se presento, empero la mayor parte son comerciales lo que ocasionan que no sean muy accesibles (por el costo, la forma de adquisición, idioma, aplicabilidad, además de otras razones) y no son adecuados para propósitos como los planteados en nuestros objetivos. Son de buena calidad y con un conocimiento sólido matemático y computacional se puede sacar provecho de ellos. Sin embargo teniendo en cuenta que el desarrollo científico y tecnológico de nuestra facultad es su misión primordial se ha optado desarrollar un software más, pero con propósitos distintos, con una buena base teórica que fundamente los métodos numéricos a utilizar para ponerlo a disposición de investigadores, docentes y estudiantes con la esperanza que le sean idóneos para resolver los problemas que ellos enfrentan en este mundo físico. Sí es uno más pero está enfocado en tres grandes tareas:

La Investigación: en este caso, se está realizando un proyecto basado en las ciencias de la computación y la matemática, para apoyar aquellas ciencias e ingenierías en las cuales tiene aplicación este trabajo.

La Proyección Social: esto debido a que tenemos la esperanza que en nuestro trabajo, científicos, ingenieros, docentes, estudiantes encuentren una herramienta de análisis numérico a su alcance (sólo tienen que solicitarlo a las instancias respectivas) No implica ningún costo monetario. Con ello contribuimos al desarrollo científico y tecnológico.

Y la Docencia: porque en cualquier curso de análisis numérico puede servir como herramienta didáctica.

La utilidad del software

Los métodos numéricos son una herramienta básica para un investigador o científico porque le permite solucionar problemas prácticos o aplicados que no podrían resolverse de manera analítica. En ocasiones se hecha mano de los métodos numéricos básicos y con poca precisión para llevar a cabo dicha tarea. Por esa razón el investigador necesita de recursos que le faciliten solucionar los problemas a los cuales se enfrenta y a la vez que le arrojen resultados confiables; el docente y estudiante necesitan herramientas en el desarrollo de cursos de análisis numérico más comprometidos con la enseñanza-aprendizaje con aplicación en un mundo físico. El software que se desarrollará servirá como herramienta de apoyo al científico, investigador, docente o estudiante.

La Factibilidad de su elaboración

Dado que se cuenta con recursos técnicos tales como equipo adecuado, software disponible para programar y desarrollar aplicaciones en un ambiente Windows, docentes idóneos como asesores, bibliografía, etc. ; los recursos económicos que implica la realización del proyecto son mínimos debido a la existencia de recurso técnico apropiado a disposición. Todo esto junto con nuestra visión de trabajo nos permite asegurar que es factible llevar a cabo a feliz termino este proyecto según lo planificado.

1.4. Objetivos del proyecto

Objetivos generales

- ✓ Desarrollar un software por medio del cual se les facilite a los estudiantes la comprensión y utilización de métodos numéricos para la resolución de problemas prácticos (incluso teóricos)
- ✓ Presentar los métodos numéricos con un fundamento teórico detallado y didáctico, permitiendo a los estudiantes y docentes que cursan e imparten respectivamente la cátedra de análisis numérico tener una herramienta de enseñanza-aprendizaje.
- ✓ Desarrollar un software sobre métodos numéricos precisos y eficientes que se puedan aplicar en las áreas de sistemas de ecuaciones lineales, ecuaciones diferenciales ordinarias e integración.
- ✓ Diseñar un software sobre métodos numéricos en las áreas mencionadas arriba en un ambiente gráfico muy amigable y fácil para el usuario.
- ✓ Llevar a la practica en el presente trabajo de graduación los conocimientos adquiridos durante los años de estudio en la carrera de Licenciatura en Matemáticas, opción Estadística-Computación mediante una combinación de la matemática y la computación.

Objetivos específicos

- ✓ Crear un software de análisis numérico en el cual se desarrollen los siguientes métodos:
  - Iteraciones de Jacobi
  - Iteraciones de Gauss-Seidel
  - De Sobrerrelajación Sucesiva (SOR)
  - De Runge-Kutta-Fehlberg
  - Algoritmo de Romberg
  - Cuadratura de Gauss-Legendre
  - Transformada rápida de Fourier

- Diferencias Finitas

- ✓ Que el software sirva como una herramienta visual interactiva para el aprendizaje y la enseñanza de los métodos anteriormente mencionados.
- ✓ Diseñar el software con el lenguaje de programación Visual Basic orientado al ambiente del sistema operativo Windows con el objetivo de aprovechar las ventajas que este tiene sobre los demás sistemas: su fácil uso, su popularidad, su estandarización de aplicaciones, etc.
- ✓ Incluir en el diseño herramientas de ayuda (textos de ayuda desplegables, opción de menú de ayuda con índice y contenido), íconos para la ejecución rápida de comandos incluidas en barras de herramientas y menús contextuales.
- ✓ Poner el software a disposición de la comunidad universitaria en especial de nuestra facultad y que responda a sus necesidades de aplicación de métodos numéricos en las áreas de interés.
- ✓ Fomentar en los estudiantes de la carrera en Licenciatura Estadística y Computación el desarrollo de software destinados al apoyo de la investigación y educación en nuestro país.

II. Determinación de requerimientos

2.1. Introducción

En la etapa de investigación preliminar con la cual se delimitó el marco de trabajo se determinaron que métodos numéricos desarrollar. Esto gracias a las necesidades planteadas por estudiantes de ingeniería y de ciencia, a la experiencia de los profesores de esta área y atendiendo también la dimensión del proyecto a desarrollar (tiempo, tecnología, personal idóneo y costos) En la etapa de determinación de requerimientos se hizo una investigación sobre los métodos determinados en la etapa anterior de manera detenida y detallada para conocer sus requerimientos específicos, tales como tipos de entrada, salidas y procesos. Para descubrir aquellos requerimientos no muy fáciles de detectar se construyo un prototipo de la aplicación.

A continuación se presenta el marco teórico que constituye la investigación teórica que se recopiló acerca de los métodos a desarrollar y posteriormente la lista de requerimientos que deberá satisfacer la aplicación.

2.2. Marco teórico de los métodos a desarrollar

2.2.1. ¿Por qué utilizar métodos numéricos?

Aunque las fórmulas matemáticas y científicas pueden usarse para obtener respuestas numéricas a una gran diversidad de problemas reales, se debe estar preparado para enfrentar situaciones, como lo muestran los siguientes problemas cuando la fórmula adecuada no está disponible.

Problema 1. Encontrar el valor mínimo y el valor máximo que adquiere en el intervalo [0, 1] la función

$$F(x) = x^6 + 5x^4 - 9x + 1$$

Puesto que F(x) es diferenciable, sus extremos en el intervalo cerrado [0, 1] pueden ocurrir, ya sea en un punto extremo o en un punto interior del intervalo, donde $F'(x) = 0$; para encontrar estos últimos valores de x uno debe

Resolver $6x^5 + 20x^3 - 9 = 0$, para todas las x que satisfagan $0 < x < 1$.

Problema 2. Dado un valor de b > 0, evalúe la integral definida $\int_0^b \sqrt{1 + x^3}\,dx$.

Problema 3. Encuentre todos los puntos (*x, y*) en el cuadrado $0 \le x, y \le 1$ que minimizan y maximizan

$$f(x,y) = \left(x^2 + y^2\right)^3 + x^2 y^4 - 2x - 4y$$

Como en el problema 1, sabemos del cálculo que para los puntos interiores (*x, y*) del cuadrado, los extremos de $f(x, y)$ se presentan cuando $\partial f / \partial x = 0$ y (simultáneamente) $\partial f / \partial y = 0$. Para encontrar estos puntos, se debe

$$\text{resolver } \left\{ \begin{aligned} 3x^5 + 6x^3 y^2 + 4x\ y^4 - 1 &= 0 \\ 3y^5 + 8x^2 y^3 + 3x^4 y - 2 &= 0 \end{aligned} \right\} \text{ para que } (x, y) \text{ satisfagan a } 0 < x, y < 1.$$

Problema 4. Encontrar la curva $y = y(x)$ que pase por el punto (0, 1) cuya pendiente de la línea tangente en cualquier punto P(*x, y*) iguala al cuadrado de la distancia de P(*x, y*) al origen, esto es,

$$\text{resolver la ecuación diferencial } \frac{dy}{dx} = y^2 + x^2,\ y = 1 \text{ cuando } x = 0.$$

Problemas similares se resuelven con éxito en cursos de álgebra y cálculo. Sin embargo, vale la pena hacer una advertencia: que nadie ha encontrado todavía una fórmula que, con funciones elementales (algebraicas y trascendentales), proporcione la solución de cualquiera una de ellos; y para algunos de ellos se sabe que no existe tal fórmula.

Afortunadamente, rara vez se necesitan respuestas numéricas exactas. De hecho, en el mundo real, los problemas mismos son, por lo general, inexactos, porque se plantean en términos que se miden y, por lo tanto, son sólo aproximaciones. Lo que suele ser necesario es, no una respuesta exacta, sino una que tenga una *precisión prescrita.*

Aún cuando una fórmula esté disponible, se debe tener cuidado porque hay cálculos matemáticos legítimos que dan respuestas exactas cuando se llevan a cabo usando números reales exactos, pero dan respuestas erróneas cuando se realizan con precisión finita en una calculadora o computadora debido a su error inherente en la representación de los números reales.

2.2.2. ¿Cómo implementar métodos numéricos en dispositivos digitales?

Desde las primeras computadoras se han utilizado algoritmos para poder resolver un problema específico. El **algoritmo** es un procedimiento paso a paso que produce un resultado deseado en un número finito de pasos. El **método numérico** es un algoritmo para encontrar uno o más valores numéricos. Los valores obtenidos por un método numérico se llaman **soluciones numéricas** y se dice que se obtienen numéricamente. A diferencia de las **soluciones analíticas**, que pueden obtenerse aplicando una fórmula, generalmente no puede esperarse que las soluciones numéricas sean exactas. En consecuencia el siguiente paso es un paso muy importante que se debe tomar en cuenta antes de usar cualquier método numérico:

Paso cero. Decidir qué tan exacta quiere que sea la solución numérica deseada.

Los métodos numéricos generalmente se realizan mediante un **dispositivo digital,** esto es, una calculadora o computadora. Los dispositivos digitales difieren en *velocidad* (las computadoras son más rápidas en general), *costo* (las calculadoras son en general más baratas tanto al comprarlas como en su uso), *capacidad de almacenamiento y de entrada / salida* (generalmente mejores en las computadoras), y *programabilidad* (generalmente mejores en las computadoras)

También difieren en su capacidad para distinguir los números reales, lo que a su vez determina la exactitud que puede alcanzar el dispositivo.

2.2.2.1. Los conceptos de error y exactitud

Si X es una aproximación de un valor exacto x, entonces

$\in_X$ es el **error de *X*** definido por $\in_X = x - X$, de modo que $x = X + \in_X$.

Así, $\in_X$ es lo que debe sumarse a la aproximación X para obtener la x exacta.

Un $\in_X$ positivo significa que X es una estimación menor a x; un $\in_X$ negativo significa que X es una estimación mayor a x. Igual, si $x \neq 0$, entonces

ρ_X es el **error relativo de *X*** definido por $\rho_X = \dfrac{x - X}{|x|} = \dfrac{\in_X}{|x|}$.

El error relativo expresa al error como una fracción de $|x|$ y por ello está relacionado al error porcentual. Por ejemplo, si $|\rho_X| < 0.02$, entonces $\in_X < 0.02|x|$, es decir, *X* se aproxima a *x* dentro del 2% de exactitud. De manera más general,

El error porcentual de aproximar *x* por *X* se define como $100\,\rho_X$.

Así, si *X* se conoce con p por ciento de exactitud, entonces $|\rho_X| < 0.01p$.

La exactitud de cifras decimales de *X* se define en términos de su **error absoluto** $|x - X|$**;** así,

X se aproxima a *x* a k cifras decimales, abreviando $x = X(kd)$, si $|x - X| < \frac{1}{2}10^{-k}$**. (Prueba de diferencia absoluta)**

Esta condición no siempre asegura que *x* y *X* sean el mismo número al redondearse a kd. Lo más que puede decirse es:

$x = X(kd) \Rightarrow x$ y *X* redondeadas podrían diferir en *una unidad* en el dígito redondeado.

La idea de dígito significativo es de importancia central cuando se discute la exactitud. El **dígito principal** (o **más significativo**) de un número real x no nulo, es el dígito no nulo más a la izquierda de su expansión decimal. Todos los dígitos, incluyendo los ceros a la derecha del dígito significativo principal, son significativos y el último desplegado se llama **dígito menos significativo**. Sin embargo, los ceros a la izquierda del dígito principal no son significativos.

La siguiente terminología y notación se emplea para determinar la exactitud del dígito significativo:

X se aproxima a *x* a k dígitos significativos, abreviando $x = X(ks)$, si $|x - X| < \frac{1}{2}10^{-k}|x|$**.**

(Prueba de diferencia relativa)

Lo más que puede decirse cuando se redondea *x* y *X* al k-ésimo dígito significativo es:

$x = X(kd) \Rightarrow x$ y *X* redondeadas podrían diferir en *una unidad* en el dígito redondeado.

Así, dividiendo, ya sea $\in_k$ o ρ_X entre 10^p o entre 10^{-p} agrega o quita, respectivamente, p dígitos exactos a la aproximación $x \approx X$. Veamos por qué.

Si se tiene que $|x - X| < \frac{1}{2}10^{-k}$, es decir, que *x* aproxima a *X* con *k* dígitos decimales.

Ahora bien, si dividimos $\in_k$ entre 10^p tenemos $|x - X| < \frac{1}{2}10^{-k} \cdot 10^p$, lo que significa que se quitan *p* cifras decimales exactas a la aproximación, según la definición dada arriba. La misma situación pasa con las cifras significativas. Con la multiplicación sucede lo contrario: es decir se quitan *p* cifras decimales o *p* cifras significativas, según sea el caso.

Es importante la distinción notacional entre $\cong$ y $\approx$.

$x \cong X$ significa que *X* aproxima a *x* [en el sentido que se ha detallado arriba] para todas las cifras desplegadas de *X*.

$x \approx X$ significa que *X* aproxima a *x*, pero no se pretende exigir nada en cuanto a su exactitud.

Como nota final en consideración a la exactitud, veamos los métodos numéricos esquemáticamente como sigue:

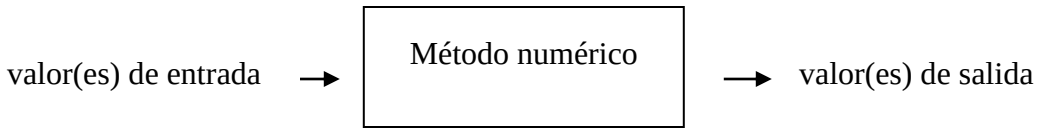


Ilustración 1: Cálculos realizados en un dispositivo digital

Cada dispositivo digital tiene una *precisión fija*, que es el número de dígitos significativos que se espere almacene con exactitud el dispositivo. Así, un dispositivo a ks almacena números reales con cuanto más un pequeño error en el k-ésimo dígito significativo. Para la mayoría de los métodos numéricos, la exactitud alcanzable de un valor de salida es, *cuando más*, la menor exactitud del dispositivo y la exactitud mínima del valor de entrada.

Existen varias razones de por qué los dispositivos digitales programables están especialmente adaptables para implementar métodos numéricos.

- **Facilidad de implementación:** los lenguajes de programación poseen varias estructuras que hacen posible la implementación de cualquier método numérico haciéndolos mucho más fáciles de crear, depurar y modificar.
- **Flexibilidad:** una vez almacenado un programa, podemos utilizarlo para resolver problemas similares. Para ello cambiamos los parámetros que caracterizan al problema desde un dispositivo de entrada.
- **Velocidad y confiabilidad:** una vez que ha sido revisado y se le han proporcionado los datos apropiados al programa para un algoritmo, éste puede ejecutarse más rápidamente y con mucha más confiabilidad que ningún ser humano podría lograr.

2.2.2.2 ¿Cometen errores los dispositivos digitales?

La computadora digital puede fallar operativamente, pero no lógicamente, esto es, la lógica computacional implementada en ellas suministrada por los fabricantes y programadores. Sin embargo, debido a su naturaleza discreta arrojan resultados numéricos reales con un error de redondeo que puede ser significativo después de muchos cálculos. De aquí que los dispositivos digitales hagan, casi sin falla o defecto, aquello para lo que fueron diseñados. De hecho, las computadoras modernas normalmente realizan billones de operaciones sin cometer un solo "error". Sin embargo, *los dispositivos digitales a menudo dan respuestas erróneas*. Con mucha frecuencia, son el resultado de **errores humanos** cometidos por *programador* (fórmula incorrecta o error de lógica en el programa), la *persona en el teclado* (errores tipográficos ya sea en el programa o en los datos de entrada), o el *usuario* (datos de entrada o método incorrectos). Hay sólo un remedio para los errores humanos: encontrarlos y hacer las correcciones apropiadas. Se debe estar alerta contra la posibilidad de que el error humano sea la causa de una respuesta irrazonable de una computadora o de una calculadora.

Una segunda fuente de error de un dispositivo digital es más sutil. Hay cálculos numéricos legítimos que dan respuestas correctas cuando se realizan con el uso de números

reales exactos, pero dan respuestas erróneas cuando se llevan a cabo en una computadora o calculadora.

La mayoría de los cálculos realizados en un dispositivo digital ocurren sin un error serio. Sin embargo, se debe de aceptar el hecho de que ambas, las computadoras y las calculadoras, son capaces de producir respuestas altamente erróneas; y cuando lo hacen, ¡las despliegan tan clara y limpiamente como las correctas. Los usuarios inteligentes de los métodos numéricos siempre examinan la salida de un dispositivo digital para asegurarse de que es razonable (para el problema que se está resolviendo) antes de aceptarla.

Para evitar (o por lo menos minimizar) los errores serios de los dispositivos digitales se deben saber porque ocurrieron. Esto, a su vez, exige comprender cómo los dispositivos digitales almacenan los números y cómo realizan aritmética con ellos.

2.2.3. Precisión fija de un dispositivo digital y el problema de mal condicionamiento

Aquí se describe cómo los dispositivos digitales usan una variante de la notación científica para almacenar números reales eficientemente y se examina los errores introducidos al hacerlo. Las calculadoras almacenan números usando **celdas con decimales codificados en binario (BCD)** que pueden almacenar los conocidos dígitos decimales 0, 1, ..., 9. Las computadoras almacenan números usando lo que puede pensarse como: *"celdas de base b",* cada una capaz de almacenar los dígitos de base-b, 0, 1, ..., b-1, donde la **aritmética de base b** puede ser 2 **(binario)**, 8 **(octal)** o 16 **(hexadecimal)**. Una celda de base-2 se llama **bit**, porque puede almacenar los dos dígitos binarios 0 y 1.

2.2.3.1. Representación de punto flotante normalizada

Los dispositivos digitales utilizan una variante de la notación científica para representar o almacenar un número real. Las computadoras almacenan los números reales usando el hecho de que cada x no nula tiene una **representación de punto flotante binaria normalizada:**

$x = \pm M \cdot 2^E$**,** la **mantisa normalizada** $\boldsymbol{M}$ satisface $\frac{1}{2} \le M < 1$.

Las computadoras tienen cierta flexibilidad en cuanto a la cantidad de memoria que puede asignarse para almacenar números reales. Sin embargo, para una aplicación numérica dada, el número de bits asignados para almacenar un número real generalmente es fijo. En esta situación, un *X* no nulo es un **número de máquina** si y solo si tiene una forma de punto flotante *binaria* normalizada $X = \pm M \cdot 2^E$ que puede almacenarse exactamente usando N + p + 1 bits como sigue:

- **Un bit para el signo:** un bit (llamado **bit de signo**) almacena el signo $(\pm)$ de *X*.
- **Un exponente corrido de p-bits:** p bits almacenan la representación binaria del entero sin signo

  $E + E_0 = (b_{p-1}..b_1 b_0)_2 = b_{p-1}2^{p-1} + \cdots + b_1 2^1 + b_0 2^0$, donde cada b_i es 0 o 1.

- **Una mantisa normalizada de N-bits:** N bits almacenan la representación binaria de la fracción sin signo

  $$M = (.b_1 b_2..b_N)_2 = \frac{b_1}{2^1} + \frac{b_2}{2^2} + \cdots + \frac{b_N}{2^N},$$

  donde cada b_i es 0 o 1, pero $b_1 = 1$.

De esta manera la mantisa normalizada puede estar entre ½ (por la restricción $b_1 = 1$) y $1 - 2^{-N}$ (se puede demostrar tomando como premisa que $\frac{1}{2^n} = \sum_{k=N+1}^{\infty} \frac{1}{2^k}, N = 0,1,..n$); además hay $2^{-(N-1)}$ de estas mantisas. Hay 2^p (por combinatoria de las N posiciones de orden cada una con dos posibilidades) exponentes corridos $E + E_0$ siendo el menor y el mayor respectivamente, 0 y $2^p - 1$. El **corrimiento** E_0 usualmente se toma como 2^{p-1}. En consecuencia, los números de máquina $X = \pm M \cdot 2^E$ se almacenan con

$$-2^{p-1} \le E \le (2^p - 1) - 2^{p-1} = 2^{p-1} - 1,$$

donde *E* es el exponente sin corrimiento de *X*.

Así, los números de máquina *X* positivos están en el intervalo [s, L], donde s el número de máquina positivo más pequeño ($\frac{1}{2} \cdot 2^{-2^{p-1}}$) y S es el número de máquina positivo más grande

$(1-2^{-N})\cdot 2^{2^{p-1}-1}$. También [s, L] se divide en 2^p subintervalos, uno por cada exponente E. Cada uno dos veces mayor que su precedente y conteniendo cada uno $2^{-(N-1)}$ números de máquina *X* equidistantes.

Todo dispositivo digital sólo puede almacenar un número finito de números de máquina X. Los no nulos están distribuidos sobre los intervalos [-L, -s] y [s, L], más densamente cerca de $\pm s$ *y más escasamente cerca de* $\pm L$

Si se aumenta el almacenamiento asignado a *M*, se aumenta la *densidad* de los *X*'s almacenables, mientras que si se aumenta el almacenamiento asignado para E, se aumenta el *tamaño* del intervalo [s, L]. En cualquier caso la mayoría de los números reales no pueden almacenarse exactamente en cualquier dispositivo digital. Si se intenta almacenar una *X* que satisfaga $0<|X|<s$ (respectivamente, $|X|>L$) se produce un **underflow (subflujo)** (respectivamente, **overflow (sobreflujo)).**

2.2.3.2. Error inherente y precisión fija asociado a los dispositivos digitales

Para la mayoría de los números reales *x*, un dispositivo digital no almacena *x*, sino el número de máquina **fl(x)** que *aproxima* a *x*. El error de fl(*x*) se llama **error de redondeo** o **error inherente** al almacenar *x*. Así,

Error inherente = *x* - fl(*x*).

Si el dispositivo es una calculadora que almacena mantisas a *k* dígitos decimales, se obtiene el fl(*x*) más exacto redondeando *x* a *ks.*

La normalización minimiza el error inherente al almacenar tantos dígitos significativos como sea posible.

El error inherente de fl(*x*) depende del número y clase de "celdas" usadas para *M* y de si el dispositivo usado redondea o trunca al almacenar.

Para una computadora que almacena $fl(x)=\pm M\cdot 2^E$ usando una mantisa normalizada de N bits, el incremento más pequeño en *M* es 2^{-N} y por ello

Si la computadora *trunca x* para obtener fl(*x*)

$$|x - fl(x)| \text{ es } \begin{cases} < 2^{-N} \cdot 2^{E} \\ \leq \frac{1}{2} 2^{-N} \cdot 2^{E} \end{cases}$$ Si la computadora *redondea x* para obtener fl(x)

En uno u otro casos el error inherente crece al incrementarse el exponente E.

La **épsilon de máquina (o unidad de máquina)** de un dispositivo digital se define como *el número de maquina positivo* más pequeño $\in$ tal que $1+\in$ no se almacena como 1. Se define como $\in_M$. Para una computadora que almacena mantisas normalizadas de N bits,

$$\in_M = \begin{cases} 2^{1-N} & \text{Si la computadora } \textit{redondea} \text{ para almacenar} \\ 2^{-N} & \text{Si la computadora } \textit{trunca} \text{ para almacenar} \end{cases}$$

Esto se puede visualizar recordando que

$$M = (.b_1 b_2 .. b_N)_2 = \frac{b_1}{2^1} + \frac{b_2}{2^2} + \cdots + \frac{b_N}{2^N},$$

donde cada b_i es 0 o 1, pero $b_1 = 1$.

Puesto que las mantisas normalizadas satisfacen $M \geq \frac{1}{2}$, $fl(x) = \pm M \cdot 2^{E}$ debe satisfacer $|fl(x)| \geq \frac{1}{2} \cdot 2^{E} = 2^{E-1}$. Así, vemos que el error inherente *relativo* satisface

$\rho_{fl(x)} \approx \frac{|x - fl(x)|}{|fl(x)|} \leq \in_M$ que se obtiene sustituyendo $|fl(x)|$ y $|x - fl(x)|$ por sus cotas superiores encontradas arriba, ya sea que se redondee o se trunque al almacenar.

Así para cualquier dispositivo la *épsilon de máquina es una cota superior eficaz para la magnitud del error relativo de fl(x), ya sea que* $|x|$ *sea grande o pequeño.* Esta uniformidad del error inherente relativo hace que ka exactitud del dígito significativo sea más conveniente que la exactitud de la cifra decimal cuando se discuten errores en dispositivos digitales.

Una calculadora o computadora será llamada **dispositivo *ks***, si su épsilon satisface

$$10^{-k} < \in_M \leq 10^{-(k-1)}.$$

Como "regla del pulgar" fl(*x*) almacenada en un dispositivo *ks* aproxima *x* con un pequeño error en el peor de los casos en el *k*-ésimo dígito significativo. De este modo, todas las calculadoras y computadoras son **dispositivos de precisión fija** donde el entero *k* en la condición anterior es la **precisión** del dispositivo.

2.2.3.3. Mal condicionamiento o inestabilidad

Recuerde que si X aproxima a un número *no nulo x*, entonces

el error relativo en X es $\rho_X = \dfrac{x - X}{|x|} = \dfrac{\in_X}{|x|}$.

La prueba de diferencia relativa afirma que X aproxima a *x* a *k* dígitos significativos (*ks)* siempre que $|\rho_X| < 0.5 \cdot 10^{-k}$, es decir, siempre que $|x - X| < 0.5 \cdot 10^{-k}|x|$. Por consiguiente, multiplicando $|\rho_X|$ por 10^p (respectivamente, 10^{-p}) se reduce (respectivamente, se incrementa) la exactitud de X en *p* dígitos significativos.

Teniendo esto en cuenta, veamos cómo un pequeño error en *x* afecta la evaluación de una expresión matemática *f(x).* Si $X \approx x$, la *f(x)* exacta y el valor de *f(X)* pueden verse así,

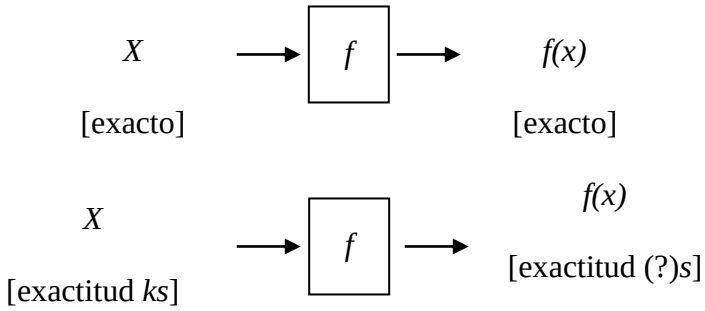


Ilustración 2: Evaluación de una expresión matemática en un dispositivo digital

Planteamos la pregunta, si *X* aproxima a *x* a *ks*, ¿*f(X)* aproximará a *f(x)* alrededor de *ks*? Aunque parezca sorprendente , la respuesta puede ser no. En vista del análisis acerca de la reducción y el incremento de la exactitud enfrentamos este problema buscando un **número de condición C** tal que $|\rho_{f(x)}| \approx C|\rho_X|$.

La constante de proporcionalidad deseada C es un *factor de amplificación del error relativo* en el sentido de que el valor *exacto* de *f(x)* cambia en aproximadamente en C % por cada 1 % de cambio (o error) en x.

El cálculo de *f(x)* está **bien condicionado (o numéricamente estable)** si la exactitud del dígito significativo asegurada del valor calculado de *f(x)* es casi la misma que la de *x*, y **mal condicionado (o numéricamente inestable)** en caso contrario. El que un cálculo mal condicionado plantee un problema serio depende de la precisión del dispositivo usado y de la exactitud deseada de *f(x)*.

2.2.4. Errores en aritmética de precisión fija

Ahora se describirá cómo se propaga el error de redondeo cuando se realizan operaciones aritméticas en calculadoras y computadoras. fl(*x*) representa la aproximación almacenada de un número real *x*, así que $\in_{fl(x)} = x - fl(x)$ es el error inherente al almacenar *x*. Sabemos que puede esperarse que fl(*x*) almacenado en un dispositivo *ks*, sea exacto a *k* dígitos significativos. Sin embargo, una vez almacenados, no hay manera de saber si fl(*x*) es mayor que *x*, menor que *x*, o igual a *x*.

2.2.4.1. Propagación del error en los cálculos aritméticos

Hagamos que $\circ$ represente una de las cuatro operaciones +, -, *, y /. Puesto que un dispositivo almacena fl(*x*) y fl(*y*), y no almacena los valores exactos de *x* y *y*, no puede esperarse que calcule $x \circ y$ exactamente. Lo más que puede esperarse que un dispositivo a *ks* haga, es que calcule $fl(x) \circledcirc fl(y)$, donde $fl(x) \circledcirc fl(y)$ significa:

encuentre $fl(x) \circ fl(y)$ hasta por lo menos *(k+1)s*, luego almacénelo redondeado o truncado a *k* dígitos significativos.

A este procedimiento se le llama **aritmética de precisión fija** o más específicamente **aritmética *ks*** y puede conducir a una variedad de errores.

Tabla 1: Propagación del error de redondeo

Nombre del error	**Descripción del error**
Adición insignificante	La adición o (sustracción) de dos números cuyas magnitudes son tan diferentes que la suma (o diferencia) se redondea al número mayor.
Redondeo escondido	El error en el *k*-ésimo dígito significativo de $X \circledcirc Y$ que puede ocurrir aún si ***X*** y ***Y*** se redondean correctamente a *ks.*
Amplificación del error	La multiplicación de un número erróneo por un número grande (en magnitud) o su división entre un número cercano a cero.
Cancelación sustractiva	La resta de dos números casi iguales(o, de manera equivalente, la suma de un número con otro casi su negativo)

Resumen de las fuentes de error en los dispositivos digitales

- **Error humano:** error cometido por el programador, operador o usuario.
- **Error de redondeo:** nombre general dado a los errores producidos al realizar aritmética en un dispositivo de precisión fija. Principia con el error inherente y luego se propaga en virtud del error escondido, la adición insignificante, la amplificación del error y/o la cancelación sustractiva.
- **Error por truncamiento (o discretización):** El error que ocurre al usar una fórmula que es sólo aproximada.

Existe una diferencia entre el cálculo y el análisis numérico. Para el análisis numérico no es suficiente saber que algo converge a un valor limite. También se necesita una

estimación del error *de la cantidad convergente o algún conocimiento de su* ***velocidad de convergencia****, de tal modo que su exactitud puede evaluarse durante la convergencia.*

2.2.4.2. Análisis cuantitativo del redondeo propagado

He aquí algunas técnicas cuantitativas para monitorear la propagación del error de redondeo. Supongamos que $x \approx X$ y que $y \approx Y$ siendo *x* y *y exactas*. Entonces,

$$\in_{X \pm Y} = (x \pm y) - (X \pm Y) = (x - Y) \pm (y - Y) = \in_X \pm \in_Y \qquad (1)$$

Así, *el error de una suma o resta es la suma o resta de los errores en los términos.* Esto explica cómo es que errores se "esconden" en el dígito menos significativo de la suma o resta de dos números correctamente redondeados. Puesto que en general no sabemos si $\in_X$ y $\in_Y$ tienen signos iguales u opuestos, lo más que puede uno decir acerca del *error absoluto* es

$$\left|\in_{x+y}\right| \le \left|\in_X\right| + \left|\in_Y\right| \text{ y } \left|\in_{x-y}\right| \le \left|\in_X\right| + \left|\in_Y\right| .$$

Así, si *X* y *Y* tienen errores en diferentes cifras decimales , el error absoluto de $X \pm Y$ será comparable al del término que tenga la menor exactitud de cifra decimal. Dividiendo (1) entre $\left|x \pm y\right|$ se obtiene

$$\rho_{X \pm Y} = \frac{\in_{X \pm Y}}{\left|x \pm y\right|} = \frac{\left|x\right|}{\left|x \pm y\right|}\frac{\in_X}{\left|x\right|} + \frac{\left|y\right|}{\left|x \pm y\right|}\frac{\in_X}{\left|y\right|} = \frac{\left|x\right|}{\left|x \pm y\right|}\rho_X \pm \frac{\left|y\right|}{\left|x \pm y\right|}\rho_Y .$$

Cuando $x \pm y$ es mucho más pequeño que *x* o *y*, entonces los factores $\left|x/(x \pm y)\right|$ y $\left|y/(x \pm y)\right|$ amplifican ρ_X o ρ_Y, respectivamente. Este caso es una *cancelación sustractiva.*

Para ver cómo la multiplicación y la división propagan errores, veamos $\in_X$ y $\in_Y$ como incrementos $dX = x - X$ y $dY = y - Y$. Si los errores relativos de *X* y *Y* son pequeños, entonces los errores resultantes en *XY* y *X/Y* pueden ser aproximados con exactitud usando diferenciales:

$$\in_{XY} \approx d(XY) = YdX + XdY = Y \in_X + X \in_Y . \qquad (2)$$

$$\in_{X/Y} \approx d(X/Y) = \frac{YdX - XdY}{Y^2} = \frac{1}{Y}\in_X - \frac{X}{Y^2}\in_Y . \qquad (3)$$

Estas describen la amplificación del error que ocurre cuando se multiplica por un número grande o se divide entre uno pequeño. Si *xy* y *XY* son nulos, entonces dividiendo (2) entre $|xy|$ y (3) entre $|x/y|$ se obtiene

$$\rho_{XY} = \frac{\in_{XY}}{|xy|} \approx \frac{Y}{|y|}\frac{\in_X}{|x|} + \frac{X}{|x|}\frac{\in_Y}{|y|} \approx (\text{signo de } y)\,\rho_X + (\text{signo de } x)\,\rho_Y .$$

$$\rho_{XY} = \frac{\in_{X/Y}}{|x/y|} \approx \frac{|y|}{Y}\frac{\in_X}{|x|} + \frac{X}{|x|}\frac{\in_Y}{|y|}\left(\frac{y}{Y}\right)^2 \approx (\text{signo de } y)\,\rho_X - (\text{signo de } x)\,\rho_Y .$$

Así, para *x* y *y* positivas , el error relativo del producto *XY* (respectivamente, el cociente *X/Y*) es en forma aproximada igual a la suma (respectivamente, a la resta) de los errores relativos de *X* y *Y*. También si *X* y *Y* tienen exactitud a *ks*, entonces su producto y su cociente tiene también exactitud cercana a *ks*, aunque sus *errores absolutos* pueden ser grandes debido a la amplificación del error.

2.2.5. Integración numérica

2.2.5.1. Elementos de integración numérica

En problemas prácticos usualmente es necesario evaluar la integral definida de una función cuya antiderivada no es una función elemental, o no es fácil de obtener. El método básico con que se aproxima $\int_a^b f(x)dx$ recibe el nombre de **cuadratura numérica** y emplea una suma del tipo

$$\sum_{i=0}^{n} a_i f(x_i)$$

para aproximar $\int_a^b f(x)dx$.

En la cuadratura numérica tomamos como base los polinomios interpolantes únicos que coinciden con el integrando en n puntos del intervalo de integración, llamados **nodos**. El primer paso es seleccionar un conjunto de nodos distintos $\{x_0,\ldots,x_n\}$ del intervalo de integración [a, b]. Luego integramos el polinomio interpolante de Lagrange

$$P_n(x) = \sum_{i=0}^{n} f(x_i) L_i(x).$$

Y su término de error de truncamiento en [a, b] para obtener

$$\int_a^b f(x)dx = \int_a^b \sum_{i=0}^{n} f(x_i) L_i(x)dx + \int_a^b \prod_{i=0}^{n} (x - x_i) \frac{f^{(n+1)}(\xi(x))}{(n+1)!} dx$$

$$= \sum_{i=0}^{n} a_i f(x_i) + \frac{1}{(n+1)!} \int_a^b \prod_{i=0}^{n} (x - x_i) f^{(n+1)}(\xi(x))dx$$

donde $\xi(x)$ se encuentra en [a, b] para cada *x* y

$$a_i = \int_a^b L_i(x)dx \text{ para cada } i = 0, 1, \ldots, n.$$

$$L_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)}$$

Por lo tanto la fórmula de cuadratura es

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} a_i f(x_i),$$

con un error dado por

$$E(f) = \frac{1}{(n+1)!}\int_a^b \prod_{i=0}^{n}(x - x_i)\, f^{(n+1)}(\xi(x))dx.$$

Utilizando el primer y segundo polinomio de Lagrange con nodos igualmente espaciados se obtiene la **regla del trapecio** y la **regla de Simpson.**

Regla del trapecio:

$$\int_a^b f(x)dx = \frac{h}{2}[f(x_0) + f(x_1)] - \frac{h^3}{12} f''(\xi).$$

Regla de Simpson:

$$\int_a^b f(x)dx = \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_1)] - \frac{h^5}{90} f^{(4)}(\xi)$$

La derivación normal de las fórmulas del error de cuadratura se base en determinar el conjunto de polinomios con los cuales estas fórmulas producen resultados exactos.

Definición:

El grado de exactitud o precisión de una fórmula de cuadratura es el entero positivo más grande *n*, tal que la fórmula sea exacta para x^k, cuando *k = 0, 1, . . . ,n.*

Las reglas del trapecio y de Simpson tienen, respectivamente, un grado de precisión de uno y tres (esto se deduce a partir de su término de error que tienen segunda y cuarta derivada).

La integración y la suma son operaciones lineales, esto es,

$$\int_a^b (\alpha f(x) + \beta g(x))dx = \alpha\int_a^b f(x)dx + \beta\int_a^a g(x)dx$$

y

$$\sum_{i=0}^{n}(\alpha f(x_i)+\beta g(x_i))=\alpha\sum_{i=0}^{n}f(x_i)+\beta\sum_{i=0}^{n}g(x_i)$$

Para cada par de funciones integrables *f* y *g*, y para cada par de constantes reales α y β.

Lo que significa que el grado de precisión de una fórmula de cuadratura será *n* si y solo si el error *E(P(x)) = 0* para todos los polinomios *P(x) k = 0, 1, ..., n*, pero $E(P(x))\neq 0$ para cualquier polinomio *P(x)* de grado *n +1*.

Las reglas del trapecio y de Simpson son ejemplos de una clase de métodos denominados fórmulas de Newton-Cotes. Existen dos categorías de fórmulas de Newton-Cotes: abiertas y cerradas.

La *fórmula cerrada de (n + 1) puntos de Newton-Cotes* utiliza los nodos $x_i=x_0+ih$, para *i = 0, 1, . . . , n,* donde $x_0=a$, $x_n=b$ y *h = (b - a)/n.* A esta fórmula se le llama cerrada, porque los extremos del intervalo cerrado [a, b] se incluyen como nodos. La formula adopta la forma

$\int_a^b f(x)dx\approx\sum_{i=0}^{n}a_i f(x_i)$, fórmula que se dedujo arriba.

En la *fórmula abierta de Newton-Cotes* se utiliza los nodos $x_i=x_0+ih$, para *i = 0, 1, . . . , n,* donde $x_0=a+h$, $x_n=b-h$ y *h = (b - a)/(n + 2),* puesto que hay dos puntos mas tomando en cuenta los extremos, por lo cual marcamos $x_{-1}=a$ y $x_{n+1}=b$. La formula adopta la forma

$$\int_a^b f(x)dx=\int_{x_{-1}}^{x_{n+1}}f(x)dx\approx\sum_{i=0}^{n}a_i f(x_i).$$

2.2.5.2. Integración numérica compuesta

Una desventaja que por lo general se observa al utilizar las fórmulas de Newton-Cotes es que no son adecuadas para utilizarse en intervalos de integración grandes. Para estos casos se requieren fórmulas de grado superior, y los valores de sus coeficientes son difíciles de obtener. Además las fórmulas de Newton-Cotes se basan en los polinomios interpolantes que emplean nodos con espacios iguales, procedimiento que resulta inexacto en intervalos grandes a causa

de la naturaleza oscilatoria de los polinomios de grado superior. Para solventar este inconveniente existen métodos fragmentarios para realizar la integración numérica, en los cuales se aplican las formulas de Newton-Cotes de bajo orden (regla del trapecio, regla de Simpson, etc.) y de esta manera alcanzar una exactitud tolerable.

Teorema:

Sean $f \in C^4[a,b]$, *n* par, $h = (b - a)/n$ y $x_j = a + jh$ para $j = 0, 1, \ldots, n$. Existe $\mu \in (a,b)$ tal que la **regla compuesta de Simpson** para *n* subintervalos puede escribirse con su término de error como

$$\int_a^b f(x)dx = \frac{h}{3}\left[f(a) + 2\sum_{j=1}^{n/2-1} f(x_{2j}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(b) \right] - \frac{b-a}{180} h^4 f^{(4)}(\mu)$$

Teorema:

Sean $f \in C^2[a,b]$, $h = (b - a)/n$ y $x_j = a + jh$ para $j = 0, 1, \ldots, n$. Existe $\mu \in (a,b)$ tal que la **regla compuesta del trapecio** para *n* subintervalos puede escribirse con su término de error como

$$\int_a^b f(x)dx = \frac{h}{2}\left[f(a) + 2\sum_{j=1}^{n-1} f(x_j) + f(b) \right] - \frac{b-a}{12} h^2 f''(\mu).$$

2.2.5.3. Extrapolación de Richardson

Con la extrapolación de Richardson podemos acelerar la convergencia y así generar resultados de gran exactitud cuando se usan fórmulas de bajo orden (o con un término de error con lenta velocidad de convergencia al limite 0 cuando el tamaño del paso *h* se acerca a 0)

La extrapolación puede aplicarse siempre que se conozca que el método de aproximación tiene un término de error de una forma o patrón predecible. La forma se basa en un parámetro, que generalmente es el tamaño del paso *h* (distancia entre dos nodos consecutivos). Supongamos que, para cada número $h \neq 0$ tenemos una fórmula *N(h)* que aproxima un valor desconocido *M* y que el error de truncamiento (o discretización) que supone presenta la forma

$$M - N(h) = K_1 h + K_2 h^2 + K_3 h^3 + \cdots$$

para un conjunto de constantes desconocidas, pero no cero $K_1, K_2, K_3, \ldots$.

La extrapolación tiene como fin encontrar una forma fácil de combinar las aproximaciones bastante imprecisas *O(h)* (la notación "*O*" indica velocidad de convergencia, en este caso la velocidad de convergencia de la aproximación es comparable a la del error de truncamiento con el parámetro *h* con potencia predominante 1) en forma apropiada para producir fórmulas con un error de truncamiento de orden superior. Supongamos, por ejemplo, que pudiéramos combinar las *N(h)* fórmulas y generar con ellas una aproximación cuya velocidad de convergencia es $O(h^2)$ y fórmula de aproximación $\hat{N}(h)$, para *M* con

$M - \hat{N}(h) = \hat{K}_2 h^2 + \hat{K}_3 h^3 + \cdots$, para un conjunto una vez más desconocido, de constantes $\hat{K}_2, \hat{K}_3, \ldots$.

Si las constantes K_1 y $\hat{K}_2$ son aproximaciones de la misma magnitud y si no hay razón para suponer que no lo sean, entonces las aproximaciones $\hat{N}(h)$ serán mucho mejores que las aproximaciones *N(h)* correspondientes. La extrapolación continúa al combinar las aproximaciones $\hat{N}(h)$ en forma tal que produzcan un error de truncamiento con velocidad de convergencia $O(h^3)$ y así sucesivamente.

La extrapolación de Richardson puede aplicarse siempre que el error de truncamiento de una fórmula presente la forma

$$\sum_{j=1}^{m-1} K_j h^{\alpha_j} + O(h^{\alpha_m})$$

para un conjunto de constantes K_j y cuando $\alpha_1 < \alpha_2 < \alpha_3 < \cdots < \alpha_m$.

Así se tienen los siguientes casos particulares:

Si *M* puede rescribirse en la forma

$$M = N(h) + \sum_{j=1}^{m-1} K_j h^j + O(h^m),$$

entonces para cada *j = 2, 3, . . ., m*, tendremos una aproximación $O(h^j)$ de la forma

$$N_j(h) = N_{j-1}\left(\frac{h}{2}\right) + \frac{N_{j-1}(h/2) - N_{j-1}(h)}{2^{j-1}-1}.$$

2.2.5.4. Integración de Romberg

En el método de la integración de Romberg se combina la regla compuesta del trapecio para obtener aproximaciones preliminares, y luego el proceso de extrapolación de Richardson para mejorar las aproximaciones. En la extrapolación de Richardson se usa una técnica de prorrateo (obtención de las aproximaciones de orden superior a partir de proporciones de aproximaciones de orden inmediato inferior) para producir fórmulas con un error de truncamiento de orden superior.

Recordemos que la regla compuesta del trapecio para aproximar la integral de una función *f* en un intervalo [a, b] por medio de *m* subintervalos es

$$\int_a^b f(x)dx = \frac{h}{2}\left[f(a) + 2\sum_{j=1}^{m-1} f(x_j) + f(b) \right] - \frac{b-a}{12} h^2 f''(\mu),$$

donde $f \in c^2[a,b]$, $\mu \in (a,b)$, $h = (b-a)/m$ y $x_j = a + jh$ para $j = 0, 1, \ldots, m$.

En la integración de Romberg primero obtenemos las aproximaciones mediante la regla compuesta del trapecio, con $m_1 = 1$, $m_2 = 2$, $m_3 = 4$, ..., y $m_n = 2^{n-1}$, donde *n* es un entero positivo. Los valores del tamaño del paso h_k correspondientes a m_k son $h_k = (b-a)/m_k = (b-a)/2^{k-1}$. Con esta notación, la regla del trapecio se expresa como:

$$\int_a^b f(x)dx = \frac{h_k}{2}\left[f(a) + 2\sum_{j=1}^{2^{k-1}-1} f(a + ih_k) + f(b) \right] - \frac{b-a}{12} h_k^2 f''(\mu_k),$$

donde $\mu_k \in (a,b)$.

Si introducimos la notación $R_{k,1}$ para denotar la parte de la integración de Romberg en la que se utiliza la ecuación anterior para realizar la aproximación por trapecios con $m_k = 2^{k-1}$ intervalos, tenemos que

$$R_{1,1} = \frac{h_1}{2}\left[f(a) + f(b)\right] = \frac{(b-a)}{12}\left[f(a) + f(b)\right];$$

$$R_{2,1} = \frac{h_2}{2}\left[f(a) + f(b) + 2f(a + h_2)\right]$$

$$= \frac{(b-a)}{4}\left[f(a) + f(b) + 2f\left(a + \frac{(b-a)}{2}\right)\right]$$

$$= \frac{1}{2}[R_{1,1} + h_1 f(a + h_2)];$$

$$R_{3,1} = \frac{1}{2}\left[R_{2,1} + h_2[f(a + h_3) + f(a + 3h_3)]\right] \quad ,$$

.

.

.

$$R_{k,1} = \frac{1}{2}\left[R_{k-1,1} + h_{k-1}\sum_{i=1}^{2^{k-2}} f(a + (2i-1)h_k)\right]$$

para cada $k = 1, 3, \ldots, n$.

La convergencia de la aproximación por trapecios es lenta aunque los cálculos necesarios no son difíciles, cuando se hacen de la forma que se dedujo arriba. La extrapolación de Richardson servirá para agilizar la convergencia como lo hemos descrito antes.

Se puede demostrar que si $f \in C^{\infty}[a,b]$, entonces podemos escribir la regla compuesta del trapecio con un término de error alterno de la forma

$$\int_a^b f(x)dx - R_{k,1} = \sum_{i=1}^{\infty} K_i h_k^{2i} = K_1 h_k^2 + \sum_{i=2}^{\infty} K_i h_k^{2i},$$

donde K_i para cada i es independiente de h_k sólo en $f^{(2i-1)}(a)$ y $f^{(2i-1)}(b)$. El término de error tiene la forma que precisa la extrapolación de Richardson.

De esta manera, la notación que se usa para la parte de la extrapolación de Richardson en la integración de Romberg, para una fórmula de aproximación $O(h_k^{2j})$ esta definida por

$$R_{k,j} = R_{k,j-1} + \frac{R_{k,j-1} - R_{k-1,j-1}}{4^{j-1} - 1}.$$

Los resultados generados con estas fórmulas se incluyen en la siguiente tabla

Tabla 2: Renglones generados por la integración de Romberg

$R_{1,1}$					
$R_{2,1}$	$R_{2,2}$				
$R_{3,1}$	$R_{3,2}$	$R_{3,3}$			
$R_{4,1}$	$R_{4,2}$	$R_{4,3}$	$R_{4,4}$		
⋮	⋮	⋮	⋮	⋱	
$R_{n,1}$	$R_{n,2}$	$R_{n,3}$	$R_{n,4}$	.....	$R_{n,n}$

Es importante mencionar que el método de Romberg tiene la característica adicional de que: permite calcular un nuevo renglón de la tabla con sólo hacer una aplicación más de la regla del trapecio, y luego usar los valores previamente calculados en un proceso de prorrateo, como se menciono antes, para obtener los elementos sucesivos del renglón. El método con que se construye una tabla de este tipo calcula los elementos o datos renglón por renglón. El algoritmo siguiente lo describe en forma detallada.

Refiérase al capítulo de diseño lógico del software.

Cabe mencionar algunas consideraciones acerca de este algoritmo. Primero, requiere un entero *n* previamente establecido, para determinar el número de renglones a generar. A menudo conviene más fijar una tolerancia de error de la aproximación y generar *n*, dentro de una cota superior, hasta que las entradas diagonales consecutivas $R_{n-1,n-1}$ y $R_{n,n}$ concuerden en el margen de tolerancia. Para evitar la posibilidad de que dos elementos de renglón consecutivos concuerden ente sí, pero no con el valor de la integral a aproximar , generamos aproximaciones hasta que no sólo $|R_{n-1,n-1} - R_{n,n}|$ esté dentro de la tolerancia sino también

$|R_{n-2,n-2} - R_{n-1,n-1}|$. Aunque esta no es una medida aplicable a todos los casos, nos garantizará que dos conjuntos de aproximaciones generados concuerden dentro del límite de la tolerancia especificada, antes de que aceptemos $R_{n,n}$ como suficientemente exacto.

2.2.6. Cuadratura de Gauss-Legendre

Las fórmulas de Newton-Cotes se derivan a partir de polinomios interpolantes. El grado de precisión de estas fórmulas es mayor en uno o dos al grado del polinomio dependiendo de si son abiertas o cerradas. Estas fórmulas, además, toman valores de la función en puntos igualmente espaciados. Para aminorar esta desventaja con respecto a funciones que tienen una gran variación en intervalos pequeños se utilizan estas fórmulas de Newton-Cotes específicas de forma compuesta en secciones del intervalo de integración. Sin embargo este inconveniente puede afectar la aproximación en gran medida debido a la naturaleza oscilatoria de las funciones.

Una alternativa a este problema es la cuadratura de Gauss-Legendre que selecciona los puntos de evaluación de manera óptima y no en una forma igualmente espaciada. Se escogen los nodos o puntos de evaluación $x_1, x_2, \ldots, x_n$ en el intervalo de integración [*a, b*] y los coeficientes $c_1, c_2, \ldots, c_n$, para reducir en lo posible el error esperado que se obtiene al efectuar la aproximación

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} c_i f(x_i)$$

para una función arbitraria *f*. La selección óptima de estos valores, obviamente, es la que dé el resultado exacto de la clase más numerosa de polinomios, es decir, la selección que ofrezca el máximo grado de precisión.

Puesto que los coeficientes $c_1, c_2, \ldots, c_n$ son arbitrarios, y los nodos $x_1, x_2, \ldots, x_n$ están restringidos en el intervalo de integración [*a, b*] tienen *2n* parámetros para elegir. El tipo de polinomios más amplio en que es posible esperar que la fórmula sea exacta es el conjunto de polinomios de grado máximo (*2n - 1*), pues se tienen *2n* condiciones que satisfarán los *2n* parámetros de la fórmula de Gauss-Legendre a raíz de la linealidad de la operación de integración. Se puede lograr la exactitud cuando los valores y constantes se seleccionan de acuerdo a estas condiciones.

Veamos un ejemplo del procedimiento con que se escogen los parámetros apropiados. Se mostrará como seleccionar los coeficientes y los nodos cuando *n = 2* y cuando el intervalo de integración es [-1, 1], veremos que es conveniente que el intervalo de integración sea éste

debido a que las raíces de un polinomio de grado *n* de Legendre que caen en el intervalo *(-1, 1)* nos darán los nodos que deseamos seleccionar. Más adelante se establece la transformación lineal utilizada para transformar un intervalo arbitrario [*a, b*] a [-1, 1]

Queremos determinar c_1, c_2, x_1, x_2, los coeficientes y nodos de la fórmula de cuadratura

$$\int_a^b f(x)dx \approx c_1 f(x_1) + c_2 f(x_2)$$

de forma que dé el resultado exacto siempre que *f(x)* sea el polinomio de grado 2(2) – 1 = 3 o menor, es decir, cuando

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3,$$

para algún conjunto de constantes a_0, a_1, a_2, a_3. Dado que

$$\int (a_0 + a_1 x + a_2 x^2 + a_3 x^3)dx = a_0 \int 1dx + a_1 \int x\,dx + a_2 \int x^2 dx + a_3 \int x^3\,dx$$

se tiene que

$$\begin{aligned} a_0 \int 1dx + a_1 \int x\,dx + a_2 \int x^2 dx + a_3 \int x^3\,dx &= c_1 f(x_1) + c_2 f(x_2) \\ &= c_1\left(\sum_{i=0}^{3} a_0 x_1^i\right) + c_2\left(\sum_{i=0}^{3} a_0 x_2^i\right) \\ &= a_0(c_1 + c_2) + a_1(c_1 x_1 + c_2 x_2) + a_2(c_1 x_1^2 + c_2 x_2^2) + a_3(c_1 x_1^3 + c_2 x_2^3) \end{aligned}$$

Para que se alcance la igualdad se tienen que cumplir de forma simulatanea cuatro condiciones en las que se involucran los parámetros de la fórmula de Gauss-Legendre:

$$c_1 \cdot 1 + c_2 \cdot 1 = \int_{-1}^{1} 1dx = 2, \qquad c_1 \cdot x_1 + c_2 \cdot x_2 = \int_{-1}^{1} x dx = 0,$$

$$c_1 \cdot x_1^2 + c_2 \cdot x_2^2 = \int_{-1}^{1} x^2\,dx = \frac{2}{3} \quad \text{y} \qquad c_1 \cdot x_1^3 + c_2 \cdot x_1^3 = \int_{-1}^{1} x^3\,dx = 0$$

Es muy fácil demostrar, con algunas operaciones algebraicas, que el sistema de ecuaciones tiene la solución única

$c_1 = 1$, $c_2 = 1$, $x_1 = -\frac{\sqrt{3}}{3}$, $x_2 = \frac{\sqrt{3}}{3}$, con los que se obtiene la fórmula de aproximación

$$\int_{-1}^{1} f(x)dx \approx f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right).$$

Como se aseveró al principio esta fórmula produce el resultado exacto con cada polinomio de grado tres o menor, por la manera analítica en que se han elegido los coeficientes y los nodos.

Con obtener fórmulas que sean exactas para polinomios de grado superior se puede aplicar la misma técnica, sim embargo es mejor aplicar un método altenativo.

Definición

Un conjunto de polinomios $\{P_0(x), P_1(x), \ldots, P_n(x), \ldots\}$ con las siguientes propiedades:

1. Para cada *n*, $P_n(x)$ es un polinomio de grado *n*.

2. $\int_{-1}^{1} P(x)\,P_n(x)dx = 0$ siempre que *P(x)* es un polinomio de grado menor que *n*.

Se llaman polinomios ortogonales de Legendre (éstos no son los mismos polinomios interpolantes de Legendre)

Los primeros polinomios de Legendre son

$$P_0(x) = 1\ , \qquad P_1(x) = x\ , \qquad P_2(x) = x^2 - \frac{1}{3},$$

$$P_3(x) = x^3 - \frac{3}{5}x \quad \text{y} \quad P_4(x) = x^4 - \frac{6}{7}x + \frac{3}{5}.$$

Las raíces de estos polinomios tienen características idóneas para producir una fórmula de la aproximación a la integral, que proporcione resultados exactos para cualquier polinomio de un grado menor que *2n*: son diferentes, se encuentran en el intervalo (-1, 1) y tienen simetría con el origen. Por estas características los nodos $x_1, x_2, \ldots, x_n$ necesarios son las raíces del polinomio de Legendre de grado *n* y los coeficientes están definidos por

$$c_i = \int_{-1}^{1} \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)} dx \quad \text{para cada } i = 1, 2, \ldots n.$$

Las constantes c_i necesarias para que la cuadratura funcione , pueden generarse a partir de la ecuación anterior; pero ambas, las constantes y las raíces de los polinomios se tabulan ampliamente.

Para aquellos problemas en los que el intervalo de integración sea diferente de [-1, 1] se realiza una transformación para que podamos utilizar las raíces del polinomio de Legendre correspondiente. Una integral $\int_a^b f(x)dx$ en un intervalo cerrado [*a*, *b*] se puede transformar, en otra en [-1, 1] usando el cambio de variables

$$t = \frac{2x-a-b}{b-a} \Leftrightarrow x = \frac{1}{2}[(b-a)t + a + b],\ dx = \frac{b-a}{2}dt\ .$$

Esto nos permite aplicar la cuadratura de Gauss-Legendre a

$$\begin{aligned}\int_a^b f(x)dx &= \frac{b-a}{2}\int_{-1}^{1} f\left(a + \frac{b-a}{2}(t+1)\right)dt \\ &= \frac{b-a}{2}\sum_{i=1}^{n} c_i\, f\left(a + \frac{b-a}{2}(t_i+1)\right)dt.\end{aligned}$$

donde los puntos $t_1, t_2, ..., t_n$ son las raíces de el polinomio de grado *n* de Legendre.

Veamos el algoritmo de Gauss-Legendre:

Refiérase al capítulo de diseño lógico del software.

2.2.7. Ecuaciones diferenciales ordinarias

2.2.7.1. Problemas de valor inicial

2.2.7.1.1 Teoría elemental de los problemas de valor inicial

Los problemas de ciencias e ingeniería que requieren el cambio de una variable con respecto a otra se modelan con ecuaciones diferenciales. En la mayor parte de ellos hay que resolver un problema de valor inicial, esto es, resolver una ecuación diferencial que satisface una ecuación dada.

Por lo general se recurren a dos métodos para aproximar la solución de una ecuación que modela una situación real, puesto que la ecuación diferencial que modela el problema resulta demasiada complicada para resolverla con exactitud. Uno de ellos consiste en simplificar la ecuación diferencial (haciendo suposiciones simplificadoras con respecto a los términos) de modo que podamos resolverla exactamente con métodos analíticos y utilizar después la solución de la ecuación simplificada para aproximar la solución de la ecuación original. El segundo, que es el que estamos tratando aquí, se vale de métodos numéricos para aproximar la solución del problema original. Este procedimiento es el que se emplea por lo regular, pues los métodos de aproximación dan resultados confiables con mayor exactitud y una información realista sobre el error.

Es importante hacer hincapié de que los métodos de aproximación no producen una aproximación continua (es decir una relación matemática definida en un intervalo continuo) a la solución del problema de valor inicial, sino que se obtienen las aproximaciones en algunos puntos específicos, llamados **puntos de red o maya**, y, a menudo, igualmente espaciados (este espacio recibe el nombre de **tamaño del paso**). Si se requieren valores intermedios se utiliza un método de interpolación, que por lo general es el de Hermite (polinomio de interpolación de Hermite)

Una situación importante que se da con los problemas de valor inicial que se plantean al observar los fenómenos físicos es que sólo pueden aproximar la situación general, por lo cual se necesita saber si cambios pequeños en el enunciado del problema introducen cambios igualmente pequeños en la solución. Esto también es importante por la aparición del error de

redondeo cuando se recurre a métodos numéricos. Nótese que el problema a resolver no esta exactamente planteado debido a la naturaleza inexacta de las mediciones.

Establezcamos la información básica que nos permitirá saber si un problema esta bien planteado o no.

Definición 1

Se dice que una función *f(t, y)* satisface una **condición de Lipschitz** en la variable *y* en un conjunto $D \subset R^2$ si existe una constante $L > 0$ con la propiedad de que

$$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2|$$

siempre que $(t, y_1), (t, y_2) \in D$. A la constante *L* se le llama **constante de Lipschitz** para *f*.

Definición 2

Se dice que un conjunto $D \subset R^2$ es **convexo,** si siempre que $(t, y_1), (t, y_2) \in D$, el punto $((1-\lambda)t_1 + \lambda t_2, (1-\lambda)y_1 + \lambda y_2)$ también pertenece a *D* para cada λ en [1, 0].

Teorema 1

Supongamos que *f(t, y)* esta definida en un conjunto convexo $D \subset R^2$. Si existe una constante $L > 0$ con $\left|\frac{\partial f}{\partial y}(t, y)\right| \leq L$, para todo $(t, y) \in D$,

Entonces *f* satisface una condición de Lipschitz en *D* en la variable *y* con la constante *L* de Lipschitz.

Teorema 2

Supongamos que $D = \{(t, y) / a \leq t \leq b, -\infty < y < \infty\}$ y que *f(t, y)* es continua en *D*. Si *f* satisface una condición de Lipschitz en *D* en la variable *y*, entonces el problema de valor inicial

$y'(t) = f(t, y), a \le t \le b, y(a) = \alpha$, tiene una solución única *y(t)* para $a \le t \le b$.

Volvamos a abordar la cuestión planteada con anterioridad acerca de la inestabilidad:

¿Cómo saber si un problema de valor inicial tiene la propiedad de que pequeños cambios o perturbaciones del planteamiento de este ocasionan cambios igualmente pequeños en su solución?

Formulemos, antes, una definición operacional que exprese este concepto.

Definición 3

Se dice que el problema de valor inicial

$$\frac{dy}{dt} = f(t, y), a \le t \le b, y(a) = \alpha,$$

es un **problema bien planteado** si:

1. El problema tiene una solución única, *y(t);*
2. Para cualquier $\varepsilon > 0$, existe una constante positiva $k(\varepsilon)$ con la propiedad de que siempre que $|\varepsilon_0| < \varepsilon$ y $\delta(t)$ es continua con $|\delta(t)| < \varepsilon$ en [*a*, *b*], el **problema perturbado** que sigue tiene una solución única, *z(t),*

$$\frac{dz}{dt} = f(t, z) + \delta(t), a \le t \le t, z(a) = \alpha + \varepsilon_0,$$

con

$|z(t) - y(t)| < k(\varepsilon)\varepsilon$, para toda $a \le t \le b$.

Los métodos numéricos para la solución de problema con valor inicial de ecuaciones diferenciales siempre se ocuparán de resolver un problema perturbado, porque cualquier error de redondeo introducido en la representación altera el problema original.

Se especificarán las condiciones que garantizan el buen planteamiento de un problema de valor inicial. Esto se enuncia en el siguiente teorema.

Teorema 3

Suponga que $D = \{(t, y) / a \le t \le b, -\infty < y < \infty\}$. Si f es continua y satisface la condición de Lipschitz en la variable *y* en el conjunto *D*, entonces el problema de valor inicial

$$\frac{dy}{dt} = f(t, y), a \le t \le b, y(a) = \alpha$$

estará bien planteado.

2.2.7.1.2. Métodos de Taylor

Puesto que necesitamos aproximaciones suficientemente exactas con un mínimo de esfuerzo al implementar métodos numéricos, para obtener aproximaciones de problema de valor inicial se cuenta con una herramienta útil que nos permite comparar la eficiencia de diversos métodos de aproximación. Esta herramienta se llama *error local de truncamiento* del método. En un paso especifico (cuando se avanza un tamaño de paso en los puntos de red), este error mide la cantidad en que la solución exacta *y(t)* de la ecuación diferencial no satisface la ecuación de diferencia con que se obtiene la aproximación.

Veamos la siguiente definición:

Definición

El método que utiliza la ecuación de diferencia

$$w_0 = \alpha$$
$$w_{i+1} = w_i + h\phi(t_i, w_i),$$

para cada *i = 0, 1, . . ., N-1,*

llamado método de diferencia, tiene el **error local de truncamiento** dado por

$$\tau_{i+1}(h) = \frac{y_{i+1} - (y_i + h\phi(t_i, y_i))}{h} = \frac{y_{i+1} - y_i}{h} - \phi(t_i, y_i).$$

para cada *i = 0, 1, . . .,N-1,* donde, $y_i = y(t_i)$ denota el valor exacto de la solución en t_i.

Al seleccionar los métodos de la ecuación de diferencia para resolver ecuaciones diferenciales ordinarias, se toma en cuenta que sus errores locales de truncamiento sean $O(h^p)$ con el valor de *p* más grande posible, sin permitir que el número y la complejidad de los cálculos de los métodos rebasen una cota razonable.

El método de Euler se deriva aplicando el método de Taylor con *n = 1*, para aproximar la solución de la ecuación diferencial; los métodos que mejoran las propiedades de convergencia de los métodos de diferencia consisten en ampliar esta técnica de derivación para valores mayores de *n*.

Supongamos que la solución *y(t)* del problema de valor inicial

$$\frac{dy}{dt} = f(t, y), a \le t \le b, y(a) = \alpha$$

tiene (*n + 1*) derivadas continuas. Al expandir *y(t)* en función del *n*-ésimo polinomio de Taylor alrededor de t_i, se obtiene

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \ldots + \frac{h^n}{n!}y^{(n)}(t_i) + \frac{h^{n+1}}{(n+1)!}y^{(n+1)}(\xi_i).$$

para alguna $\xi_i \in (t_i, t_{i+1})$.

La diferenciación sucesiva de la sucesión *y(t)* nos da

$$\begin{aligned} y'(t) &= f(t, y(t)), \\ y''(t) &= f'(t, y(t)), \\ &\vdots \\ y^{(k)} &= f^{(k-1)}(t, y(t)). \end{aligned}$$

para *k = 1, 2, . . .,n.*

Sustituyendo estos resultados en la expansión de Taylor obtenemos

$$\begin{aligned} y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2}f'(t_i, y(t_i)) + \ldots \\ + \frac{h^n}{n!}f^{(n-1)}(t_i, y(t_i)) + \frac{h^{n+1}}{(n+1)!}f^{(n)}(\xi_i, y(\xi_i)). \end{aligned}$$

A partir de esta expansión podemos obtener el método de la ecuación de diferencia correspondiente que recibe el nombre de *método de Taylor de orden n* y se obtiene suprimiendo el término residual que contiene a ξ_i y tomando $w_i \approx y(t_i)$.

Método de Taylor de orden *n*:

$$w_0 = \alpha$$
$$w_{i+1} = w_i + hT^{(n)}(t_i, w_i),$$

para cada *i = 0, 1, . . ., N-1*,

donde

$$T^{(n)}(t_i, w_i) = f(t_i, w_i)) + \frac{h}{2} f'(t_i, w_i)) + \ldots + \frac{h^{n-1}}{n!} f^{(n-1)}(t_i, w_i)).$$

El método de Taylor de orden uno es el método de Euler como mencionamos al principio de este planteamiento.

2.2.7.1.3. Métodos de Runge-Kutta

Con los métodos de Taylor se requiere el cálculo y la evaluación de las derivadas de *f(t, y)*. Una desventaja significativa aunque poseen la propiedad de un error local de truncamiento de orden superior al método de Euler. La obtención de las derivadas de *f(t, y)* es un procedimiento lento y complicado en la mayoría de los problemas, por lo cual los método de Taylor rara vez los emplearemos en la práctica.

Los métodos que describimos aquí tienen el error local de truncamiento de orden superior a los métodos de Taylor; además permiten prescindir del cálculo y evaluación de las derivadas de *f* (*t , y)*. Estos métodos se llaman: **métodos de Runge-Kutta.**

Debemos, antes, enunciar el teorema de Taylor para dos variables.

Teorema

Supóngase que *f(t, y)* y todas sus derivadas parciales de orden menor o igual que *n + 1* son continuas en $D = \{(t, y) / a \le t \le b, -\infty < y < \infty\}$, y sea $(t_0, y_0) \in D$. Para toda $(t\ , y\ ) \in D$, existe ξ entre t y t_0 y μ entre y y y_0 con

$$f(t,y) = P_n(t,y) + R_n(t,y),$$

donde

$$\begin{aligned} P_n(t,y) = {} & f(t_0,y_0) + \left[(t-t_0)\frac{\partial f}{\partial t}(t_0,y_0) + (y-y_0)\frac{\partial f}{\partial t}(t_0,y_0) \right] \\ & + \left[\frac{(t-t_0)^2}{2}\frac{\partial^2 f}{\partial t^2}(t_0,y_0) + (t-t_0)(y-y_0)\frac{\partial^2 f}{\partial t \partial y}(t_0,y_0) \right. \\ & \left. + \frac{(y-y_0)^2}{2}\frac{\partial^2 f}{\partial y^2}(t_0,y_0) \right] + \ldots \\ & \left[\frac{1}{n!}\sum_{j=0}^{n}\binom{n}{j}(t-t_0)^{n-j}(y-y_0)^j \frac{\partial^n f}{\partial t^{n-j}\partial y^j}(t_0,y_0) \right] \end{aligned}$$

y

$$R_n(t,y) = \frac{1}{n!}\sum_{j=0}^{n+1}\binom{n+1}{j}(t-t_0)^{n+1-j}(y-y_0)^j \frac{\partial^{n+1} f}{\partial t^{n+1-j}\partial y^j}(\xi,\mu).$$

A la función $P_n(t,y)$ se le llama **polinomio de Taylor de grado n en dos variables** para la función *f* alrededor de (t_0,y_0), y $R_n(t,y)$ es el término residual asociado a $P_n(t,y)$.

El primer paso al derivar el método de Runge-Kutta, es determinar los valores de a_1, α_1 y β_1 con la propiedad de que $a_1 f(t+\alpha_1, y+\beta_1)$ aproxima

$$T^2(t,y) = f(t,y) + \frac{h}{2}f'(t,y)$$

en el método de la ecuación de diferencia

$$w_0 = \alpha$$
$$w_{i+1} = w_i + hT^{(2)}(t_i, w_i),$$

para cada *i = 0, 1, . . ., N-1*,

con error no mayor que $O(h^2)$, o sea el error local de truncamiento del método de Taylor de orden dos.

Debido a que sólo tres parámetros se encuentran en $a_1 f(t+\alpha_1, y+\beta_1)$ y los tres se requieren en el igualamiento de $T^{(2)}$, necesitamos una forma más compleja para cumplir las condiciones que requiere cualquiera de los métodos de Taylor de orden superior.

2.2.7.1.4. El método de Runge-Kutta-Fehlberg

En la integración numérica se analiza el uso apropiado del tamaño del paso para producir métodos de aproximación de la integral con la eficiencia requerida en la cantidad de cálculos. Por sí mismo, ello tal vez no sería suficiente para preferirlos, dado la mayor complejidad de su uso. Pero presentan otra características que los hace sumamente útiles. En el procedimiento del tamaño del paso incorporan una estimación del error de truncamiento que no requiere la estimación de las derivadas superiores de la función. A estos métodos se les llama adaptativos, porque adaptan el número y la posición de los nodos con que se efectúa la aproximación, para garantizar que el error de truncamiento no rebase la cota especificada.

Existe una estrecha relación entre el problema de aproximar el valor de una integral definida y el aproximar la solución de un problema de valor inicial. De este modo, no debe ser sorprendente que haya métodos adaptativos que aproximan las soluciones de los problemas de valor inicial y que no sólo sean eficientes, sino que además incluyan el control del error.

Un método ideal de la ecuación de diferencia

$$w_{i+1} = w_i + h_i \phi(t_i, w_i, h_i), i = 0,1,\ldots,N-1,$$

con que se aproxima la solución *y(t)* al problema de valor inicial

$$y'(t) = f(t,y), a \le t \le b, y(a) = \alpha$$

deberá tener la propiedad de que, con una tolerancia $\varepsilon > 0$, la cantidad mínima de puntos de red servirá para asegurarse de que el error global $|y(t_i) - w_i|$, no rebasará ε con cualquier *i =0, 1, . . ., N.* No debe sorprendernos que tener una cantidad mínima de puntos de red y el control del error global de un método de diferencia , sea incompatible con el espaciamiento uniforme de los puntos en los intervalos. El método de Runge-Kutta-Fehlberg controla eficientemente el error de un método de ecuación de diferencia mediante la selección apropiada de los puntos de red.

Existe una estrecha relación entre el error local de truncamiento y el error global. Mediante métodos de orden distinto podemos predecir el error local de truncamiento y seleccionar con esta predicción un tamaño del paso que controle el error global.

Supongamos que tenemos dos métodos de aproximación. El primero es un método de *n*-ésimo orden obtenido de un método de Taylor de *n*-ésimo orden de la forma

$$y(t_{i+1}) = y(t_i) + h\phi(t_i, y(t_i), h) + O(h^{n+1}),$$

que produce las aproximaciones

$$w_0 = \alpha$$
$$w_{i+1} = w_i + h\phi(t_i, w_i, h),$$

para $i > 0$, con el error local de truncamiento $\tau_{i+1}(h) = O(h^n)$. En términos generales, el método se obtiene aplicando una modificación de Runge-Kutta al método de Taylor.

El segundo método es semejante, pero de orden superior. Por ejemplo, supongamos que proviene de un método de $(n + 1)$-ésimo orden de la forma

$$y(t_{i+1}) = y(t_i) + h\tilde{\phi}(t_i, y(t_i), h) + O(h^{n+2}),$$

y produce las aproximaciones

$$\tilde{w}_0 = \alpha$$
$$\tilde{w}_{i+1} = \tilde{w}_i + h\tilde{\phi}(t_i, \tilde{w}_i, h),$$

para $i > 0$, con error de truncamiento $\tilde{\tau}_{i+1}(h) = O(h^{n+1})$.

Supongamos primero que $w_i \approx y(t_i) \approx \tilde{w}_i$ y seleccionamos un tamaño de paso fijo h para generar las aproximaciones w_{i+1} y $\tilde{w}_{i+1}$ a $y(t_{i+1})$. Entonces

$$\begin{aligned}\tau_{i+1}(h) &= \frac{y(t_{i+1}) - y(t_i)}{h} - \phi(t_i, y(t_i), h) \\ &= \frac{y(t_{i+1}) - w_i}{h} - \phi(t_i, w_i, h) \\ &= \frac{y(t_{i+1}) - [w_i + h\phi(t_i, w_i, h)]}{h} \\ &= \frac{1}{h}(y(t_{i+1}) - w_{i+1}).\end{aligned}$$

De manera análoga,

$$\tilde{\tau}_{i+1}(h) = \frac{1}{h}(y(t_{i+1}) - \tilde{w}_{i+1}).$$

En consecuencia

$$\begin{aligned}\tau_{i+1}(h) &= \frac{1}{h}(y(t_{i+1}) - w_{i+1}) \\ &= \frac{1}{h}\left[(y(t_{i+1}) - \tilde{w}_{i+1}) + (\tilde{w}_{i+1} - w_{i+1})\right] \\ &= \tilde{\tau}_{i+1}(h) + \frac{1}{h}(\tilde{w}_{i+1} - w_{i+1}).\end{aligned}$$

Pero $\tau_{i+1}(h)$ es $O(h^n)$ y $\tilde{\tau}_{i+1}(h)$ es $O(h^{n+1})$, por lo cual la parte significativa de $\tau_{i+1}(h)$ debe provenir de $(1/n)(\tilde{w}_{i+1} - w_{i+1})$. Esto nos da una aproximación calculada fácilmente del error local de truncamiento del método $O(h^n)$:

$$\tau_{i+1}(h) \approx \frac{1}{h}[\tilde{w}_{i+1} - w_{i+1}].$$

Pero, el objetivo no es sólo estimar el error local de truncamiento, sino ajustar además el tamaño del paso para mantenerlo dentro de una cota especificada. Para hacerlo, ahora se supone que como $\tau_{i+1}(h)$ es $O(h^n)$, existe un número *K* independiente de *h* tal que

$$\tau_{i+1}(h) \approx Kh^n.$$

Luego podemos estimar el error local del truncamiento producido al aplicar el método de *n*-ésimo orden con un nuevo tamaño de paso *qh*, usando las aproximaciones originales w_{i+1} y $\tilde{w}_{i+1}$:

$$\tau_{i+1}(qh) \approx K(qh)^n = q^n(Kh^n) \approx q^n \tau_{i+1}(h) \approx \frac{q^n}{h}(\tilde{w}_{i+1} - w_{i+1}).$$

Para establecer la cota de $\tau_{i+1}(qh)$ por ε, escogemos q tal que

$$\frac{q^n}{h}\left|\tilde{w}_{i+1} - w_{i+1}\right| \approx \left|\tau_{i+1}(qh)\right| \leq \varepsilon,$$

es decir, tal que

$$q \leq \left(\frac{\varepsilon h}{\left|\tilde{w}_{i+1} - w_{i+1}\right|}\right)^{1/n}$$

El **Método de Runge-Kutta-Fehlberg** utiliza esta desigualdad para controlar el error local de truncamiento. Este método consiste en emplear el método de Runge-Kutta con el error local de truncamiento de quinto orden,

$$\tilde{w}_{i+1} = \tilde{w}_i + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6,$$

para estimar el error local en un método de Runge-Kutta de cuarto orden dado por

$$w_{i+1} = w_i + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5,$$

donde

$$\begin{aligned}
k_1 &= hf(t_i, w_i),\\
k_2 &= hf\left(t_i + \frac{h}{4}, w_i + \frac{1}{4}k_1\right),\\
k_3 &= hf\left(t_i + \frac{3h}{4}, w_i + \frac{3}{32}k_1 + \frac{9}{32}k_2\right),\\
k_4 &= hf\left(t_i + \frac{12h}{13}, w_i + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3\right),\\
k_5 &= hf\left(t_i + h, w_i + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4\right),\\
k_6 &= hf\left(t_i + \frac{h}{2}, w_i - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5\right).
\end{aligned}$$

Existe una ventaja importante de este método y consiste en que sólo se requieren seis evaluaciones de *f* por paso. Los métodos arbitrarios de Runge-Kutta de cuarto y quinto orden usados de manera conjunta requieren al menos cuatro evaluaciones de *f* con el método de cuarto orden y seis más con el de quinto orden, lo cual nos da un total de, por lo menos, diez evaluaciones de funciones.

Un valor inicial *h* en el *i*-ésimo paso se usó para obtener los primeros valores de w_{i+1} y $\tilde{w}_{i+1}$, que nos permitieron determinar *q* en ese paso, y luego se repitieron los cálculos. Este procedimiento requiere el doble de evaluaciones de funciones por paso, sin control. El valor de *q* determinado en el *i*-ésimo paso cumple dos propósitos:

1. Para rechazar, de ser necesario, la elección inicial de *h* en el paso *i*-ésimo y repetir los cálculos por medio de *qh*, y

2. Para predecir una elección adecuada de *h* para el (*i + 1*)-ésimo paso.

Debido a la sanción que debe pagarse en términos de evaluaciones si se repiten los pasos, *q* tiende a ser elegida de manera conservadora; de hecho, en el método de Runge-Kutta-Fehlberg con $n = 4$, la elección común es

$$q = \left(\frac{\varepsilon h}{2|\tilde{w}_{i+1} - w_{i+1}|} \right)^{1/4} = 0.84 \left(\frac{\varepsilon h}{|\tilde{w}_{i+1} - w_{i+1}|} \right)^{1/4}.$$

Es de hacer ver que en el algoritmo que sigue para el método de Runge-Kutta-Fehlberg, se agrega el paso 9 para suprimir grandes modificaciones al tamaño del paso. Esto se hace para no tener que dedicar mucho tiempo a los tamaños pequeños de paso en las regiones donde hay irregularidades de las derivadas de *y,* y para evitar los grandes tamaños de paso, que pueden llevar a omitir las regiones sensibles entre los pasos. En algunos casos, en el algoritmo se omite totalmente el procedimiento que aumenta el tamaño del paso, y el procedimiento con que se disminuye el tamaño se modifica para que se incorpore sólo cuando es necesario controlar el error.

Veamos el método de Runge-Kutta-Fehlberg:

Refiérase al capítulo de diseño lógico del software.

2.2.7.2. Problema con valor en la frontera

2.2.7.2.1. Teoría elemental de los problemas con valor en la frontera

Hacen falta técnicas nuevas para resolver los problemas cuando las condiciones impuestas son de un valor de frontera y no del tipo de valor inicial. Los métodos para obtener soluciones aproximadas a las ecuaciones diferenciales con valor inicial, requieren que todas las condiciones impuestas a la ecuación diferencial ocurran en un punto inicial. En el caso de una ecuación inicial de segundo orden, necesitamos conocer tanto *w(0)* como $w'(0)$, lo cual no es el caso para problemas con valor en la frontera.

Aquellos problemas físicos que dependen más de la posición que del tiempo, usualmente se describen en función de ecuaciones diferenciales, con las condiciones impuestas en más de punto. Trataremos aquí los problemas con valor en la frontera de dos puntos incluyen una ecuación diferencial de segundo orden de la forma

$$y'' = f(x, y, y'), a \le x \le b,$$

junto con las condiciones de frontera

$$y(a) = \alpha \text{ y } y(b) = \beta.$$

Veamos el siguiente teorema que establece las condiciones generales que garantizan que exista la solución a un problema con valor en la frontera de segundo orden y que dicha solución sea única.

Teorema

Supongamos que la función *f* en el problema con valor en la frontera

$$y'' = f(x, y, y'), a \le x \le b, y(a) = \alpha, y(b) = \beta,$$

es continua en el conjunto

$$D = \{(x, y, y') \mid a \le x \le b, -\infty < y < \infty, -\infty < y' < \infty\},$$

y que $\frac{\partial f}{\partial y}$ y $\frac{\partial f}{\partial y'}$ también son continuas en D. Si

$\frac{\partial f}{\partial y}(x, y, y') > 0$ para toda $(x, y, y') \in D$ y

Existe una constante *M*, con

$\left|\frac{\partial f}{\partial y}(x, y, y')\right| \leq M$, para toda $(x, y, y') \in D$,

entonces el problema con valor en la frontera tiene solución única.

Cuando $f(x, y, y')$ tiene la forma

$$f(x, y, y') = p(x)y' + q(x)y + r(x),$$

la ecuación diferencial

$$y'' = f(x, y, y')$$

es **lineal.** Este tipo de problemas ocurren frecuentemente, y en este caso el teorema puede simplificarse:

Corolario

Si el problema lineal con valor en la frontera

$$y'' = f(x, y, y') = p(x)y' + q(x)y + r(x), a \leq x \leq b, y(a) = \alpha, y(b) = \beta$$

satisface

p(x), q(x) y *r(x)* son continuas en [a, b],

q(x) > 0 en [*a, b*],

entonces el problema tiene una solución única.

2.2.7.2.2. Métodos de diferencias finitas para los problemas lineales

Este método tiene mejores características de estabilidad, pero generalmente hay que trabajar más para obtener la exactitud especificada.

Los métodos para la resolución de problemas con valor en la frontera que contienen diferencias finitas sustituyen las derivadas en la ecuación diferencial mediante una aproximación de cociente de diferencias adecuada. Se selecciona el cociente de diferencias para mantener un orden especificado del error de truncamiento. Pero, por la inestabilidad de

las aproximaciones de diferencias finitas a las derivadas, no podemos escoger un parámetro *h* demasiado pequeño.

El problema de valor de frontera de segundo orden

$$y'' = f(x, y, y') = p(x)y' + q(x)y + r(x), a \le x \le b, y(a) = \alpha, y(b) = \beta.$$

Requiere utilizar las aproximaciones del cociente de diferencias para aproximar tanto a y' como a y''. Primero se selecciona un entero $N > 0$ y dividimos el intervalo [*a, b*] en ($N + 1$) subintervalos iguales cuyos extremos son los puntos de red $x_i = a + ih$, para $i = 0, 1, 2, \ldots N + 1$, donde $h = (b - a)/(N + 1)$. Escogiendo *h* de este modo , se facilita la aplicación de un algoritmo matricial, con el cual se resuelve un sistema lineal que contenga una matriz de *N* x *N*.

En el interior de los puntos de red, x_i, para $i = 1, 2, \ldots, N$, la ecuación diferencial a aproximar es

$$y''(x_i) = p(x_i)y'(x_i) + q(x_i)y(x_i) + r(x_i).$$

Expandiendo *y* en el tercer polinomio de Taylor alrededor de x_i evaluada en x_{i+1} y x_{i-1}, tenemos

$$y(x_{i+1}) = y(x_i + h) = y(x_i) + hy'(x_i) + \frac{h^2}{2}y''(x_i) + \frac{h^3}{6}y'''(x_i) + \frac{h^4}{24}y^{(4)}(\xi_i^+),$$

para alguna $\xi_i^+ \in (x_i, x_{i+1})$, y

$$y(x_{i-1}) = y(x_i - h) = y(x_i) - hy'(x_i) + \frac{h^2}{2}y''(x_i) - \frac{h^3}{6}y'''(x_i) + \frac{h^4}{24}y^{(4)}(\xi_i^-),$$

para alguna $\xi_i^- \in (x_{i-1}, x_i)$, suponiendo $y \in C^4[x_{i-1}, x_{i+1}]$. Si se agregan estas ecuaciones, se eliminan los términos que contienen a $y'(x_i)$ y $y'''(x_i)$ y un simple manejo algebraico da

$$y''(x_i) = \frac{1}{h^2}\left[y(x_{i+1}) - 2y(x_i) + y(x_{i-1})\right] - \frac{h^2}{24}\left[y^{(4)}(\xi_i^+) + y^{(4)}(\xi_i^-)\right].$$

Aplicando el teorema del valor intermedio para simplificar aún más esta expresión obtenemos

$$y''(x_i) = \frac{1}{h^2}\left[y(x_{i+1}) - 2y(x_i) + y(x_{i-1})\right] - \frac{h^2}{12}\left(\frac{y^{(4)}(\xi_i^+) + y^{(4)}(\xi_i^-)}{2}\right),$$

puesto que $\dfrac{y^{(4)}(\xi_i^+) + y^{(4)}(\xi_i^-)}{2}$ se encuentra entre $y^{(4)}(\xi_i^+)$ y $y^{(4)}(\xi_i^-)$ existe un $\xi_i \in (x_{i-1}, x_i)$ tal que $y^{(4)}(\xi_i) = \dfrac{y^{(4)}(\xi_i^+) + y^{(4)}(\xi_i^-)}{2}$. De aquí la expresión

$$y''(x_i) = \frac{1}{h^2}\left[y(x_{i+1}) - 2y(x_i) + y(x_{i-1})\right] - \frac{h^2}{12}\left(y^{(4)}(\xi_i)\right)$$

para alguna $\xi_i \in (x_{i-1}, x_i)$. A esto se le llama **fórmula de las diferencias centradas** para $y''(x_i)$.

De la misma manera se obtiene una fórmula de este tipo para $y'(x_i)$, que da como resultado

$$y'(x_i) = \frac{1}{2h}\left[y(x_{i+1}) - y(x_{i-1})\right] - \frac{h^2}{6}y'''(\eta_i).$$

para alguna $\eta_i \in (x_{i-1}, x_i)$.

La utilización de las fórmulas de diferencias centradas en la ecuación $y''(x_i) = p(x_i)y'(x_i) + q(x_i)y(x_i) + r(x_i)$ genera la ecuación

$$y''(x_i) = \frac{1}{h^2}\left[y(x_{i+1}) - 2y(x_i) + y(x_{i-1})\right] = p(x_i)\left[\frac{1}{2h}\left[y(x_{i+1}) - y(x_{i-1})\right]\right] + q(x_i)y(x_i)$$
$$r(x_i) - \frac{h^2}{12}\left[2p(x_i)y''(\eta_i) - y^{(4)}(\xi_i)\right]$$

Un método de diferencias finitas con error de truncamiento de orden $O(h^2)$ se obtiene empleando esta ecuación junto con las condiciones de frontera $y(a) = \alpha$, $y(b) = \beta$ para definir

$$w_0 = \alpha, w_{N+1} = \beta$$

$$\frac{1}{h^2}\left[2w_i - w_{i+1} - w_{i-1}\right] + p(x_i)\left(\frac{1}{2h}\left[w_{i+1} - w_{i-1}\right]\right) + q(x_i)w_i = -r(x_i),$$

para toda $i = 1, 2, \ldots, N$.

En la forma que consideramos, la ultima ecuación se rescribe como

$$-\left(1+\frac{h}{2}p(x_i)\right)w_{i-1}+\left(2+h^2q(x_i)\right)w_i-\left(1-\frac{h}{2}p(x_i)\right)w_{i+1}=-h^2r(x_i),$$

y el sistema de ecuaciones resultante se expresa en forma de la matriz tridiagonal de N x N

$\mathbf{Aw}=\mathbf{b}$, donde

$$\mathbf{A}=\begin{bmatrix} 2+h^2q(x_i) & -1+\frac{h}{2}p(x_1) & 0 & \cdots & 0 \\ -1-\frac{h}{2}p(x_2) & 2+h^2q(x_2) & -1+\frac{h}{2}p(x_2) & \cdots & \vdots \\ 0 & -1-\frac{h}{2}p(x_3) & 2+h^2q(x_3) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & -1+\frac{h}{2}p(x_{N-1}) \\ 0 & \cdots & 0 & -1-\frac{h}{2}p(x_N) & 2+h^2q(x_N) \end{bmatrix}$$

$$\mathbf{w}=\begin{bmatrix} w_1 \\ w_2 \\ . \\ . \\ . \\ w_{N-1} \\ w_N \end{bmatrix}, \text{ y } \mathbf{b}=\begin{bmatrix} -h^2r(x_i)+\left(1+\frac{h}{2}p(x_1)\right)w_0 \\ -h^2r(x_2) \\ \vdots \\ -h^2r(x_{N-1}) \\ -h^2r(x_N)+\left(1-\frac{h}{2}p(x_N)\right)w_{N+1} \end{bmatrix}.$$

El siguiente teorema establece las condiciones bajo las cuales este sistema lineal tridiagonal tiene una solución única.

Teorema

Supongamos que p, q y r son continuas en $[a, b]$. Si $q(x)\geq 0$ en $[a, b]$, entonces el sistema lineal tridiagonal anterior tiene una solución única siempre y cuando $h < 2/L$, donde $L=máx_{a\leq x\leq b}\left|p(x)\right|$.

Es conveniente señalar que las hipótesis del teorema garantizan una solución única al problema del valor de frontera, pero no que $y \in C^4[a,b]$. Para asegurarnos que el error de truncamiento tiene el orden $O(h^2)$, debemos establecer que $y^{(4)}$ es continua en [a, b].

Veamos el algoritmo lineal de diferencias finitas:

Refiérase al capítulo de diseño lógico del software.

2.2.8. Sistemas de ecuaciones lineales

2.2.8.1. Teoría elemental

Los sistemas de ecuaciones lineales se utilizan en muchos problemas de ingeniería y de las ciencias, así como en aplicaciones de las matemáticas a las ciencias sociales y al estudio cuantitativo de los problemas de administración y economía.

El sistema lineal

$$\begin{aligned} E_1 &: a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ E_2 &: a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_1, \\ &\vdots \\ E_n &: a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_1, \end{aligned}$$

para $x_1,\ldots,x_n$, dadas las a_{ij} con *i, j = 1, 2, . . ., n* y b_i, para *i = 1, 2, . . ., n* tiene su representación lineal, a saber

$\boldsymbol{A}\mathbf{x} = \mathbf{b}$, donde

$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$ es la matriz de coeficientes,

$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \ldots & x_n \end{bmatrix}^t$ es el vector columna de incógnitas y

$\mathbf{b} = \begin{bmatrix} b_1 & b_2 & \ldots & b_n \end{bmatrix}^t$ es el vector columna de términos independientes.

2.2.8.1.1. Norma de vectores y matrices

Antes de introducirnos en materia de los métodos iterativos con que se resuelven los sistemas lineales, necesitamos contar primero con un medio que nos permita medir la distancia entre los vectores columnas *n*-dimensionales, para determinar si una serie de esos vectores convergen a una solución del sistema. Denotemos con R^n el conjunto de todos los vectores columna *n*-dimensionales con componentes de números reales. Si queremos definir una distancia en R^n utilizaremos la noción de una norma.

Definición.

Una **norma vectorial** en R^n es una función, $\|\cdot\|$, de R^n a R con las siguientes propiedades:

(i) $\|\mathbf{x}\| \geq 0$ para todo $\mathbf{x} \in R^{\mathbf{n}}$,

(ii) $\|\mathbf{x}\| = 0$ si y sólo si $\mathbf{x} = (0,0,\ldots,0)^t \equiv \mathbf{0}$,

(iii) $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|$ para todo $\alpha \in R$, $\mathbf{x} \in R^{\mathbf{n}}$,

(iv) $\|\mathbf{x}+\mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ para todo $\mathbf{x}, \mathbf{y} \in R^{\mathbf{n}}$.

Como los vectores de R^n son vectores columna, conviene utilizar la notación de la transpuesta cuando representamos un vector en función de sus componentes.

Veamos la siguiente definición.

Definición.

Las normas l_2 y l_∞ del vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$ están definidas por $\|\mathbf{x}\|_2 = \left\{ \sum_{i=0}^{n} x_i^2 \right\}^{1/2}$

y $\|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$.

La norma l_2 se llama **norma euclidiana** del vector **x,** dado que representa la noción común del distancia respecto al origen en caso de que **x** esté en $R^1 \equiv R, R^2$, o bien R^3.

La norma de un vector proporciona una medida de la distancia entre un vector arbitrario y el vector cero; por ello podemos definir la **distancia entre dos vectores** como la norma de la diferencia de los vectores.

Definición

Una **norma matricial** sobre el conjunto de todas las matrices de *n* x *n* es una función real, $\|\cdot\|$, definida en este conjunto y que satisface para todas las matrices ***A*** y ***B*** de *n* x *n* y todos los números reales α:

(i) $\|\mathbf{A}\| \geq 0$.

(ii) $\|\mathbf{A}\| = 0$ si y sólo si ***A*** es ***0,*** la matriz con todas las entradas cero.

(iii) $\|\alpha\mathbf{A}\| = |\alpha|\|\mathbf{A}\|$

(iv) $\|\mathbf{A}+\mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$

(v) $\|\mathbf{AB}\| \leq \|\mathbf{A}\|\|\mathbf{B}\|$.

Una distancia entre dos matrices A y B de *n* x *n* respecto a esta norma matricial es $\|\mathbf{A}-\mathbf{B}\|$

Teorema.

Si $\|\cdot\|$ es una norma vectorial de R^n, entonces

$$\|\mathbf{A}\| = \max_{\|\mathbf{x}\|=1} \|\mathbf{Ax}\|$$

es una norma matricial.

A esta norma se le llama **norma matricial natural o inducida** asociada con una norma vectorial.

Teorema.

Si $\mathbf{A} = (a_{ii})$ es una matriz de *n* x *n*, entonces

$$\|\mathbf{A}\|_\infty = \max_{1\leq i\leq n} \sum_{j=1}^{n} |a_{ij}|.$$

2.2.8.1.2. Vectores y valores característicos

El teorema anterior nos ofrece un método o una fórmula para determinar la norma l_∞ de una matriz. Aquí veremos que la norma l_2 de una matriz también puede determinarse a través de una fórmula o un método.

Definición.

Si ***A*** es una matriz cuadrada, el polinomio definido por

$$p(\lambda) = \det(\mathbf{A} - \lambda I)$$

recibe el nombre de **polinomio característico** de ***A.***

Definición

Si *p* es el polinomio característico de la matriz **A,** los ceros de *p* se llaman **valores característicos** o **propios (eigenvalores)** de esa matriz. Si λ es un valor característico de ***A*** y si $\mathbf{x} \neq \mathbf{0}$ tiene la propiedad de que $(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}$ entonces a **x** se le llama **vector característico** o propio de la matriz ***A*** correspondiente al valor característico λ .

Definición.

El **radio espectral** $\rho(\mathbf{A})$ de una matriz ***A*** está definido por

$\rho(\mathbf{A}) = máx|\lambda|$, donde λ es un valor característico de ***A***.

Como lo habíamos anunciado y se puede apreciar en el siguiente teorema, el radio espectral guarda estrecha relación con la norma de una matriz.

Teorema.

Si ***A*** es una matriz de *n* x *n* ,entonces

(i) $\|\mathbf{A}\|_2 = \left[\rho(\mathbf{A}^\mathbf{t}\mathbf{A})\right]^{1/2}$,

(ii) $\rho(\mathbf{A}) \leq \|\mathbf{A}\|$, para toda norma natural $\|\cdot\|$.

2.2.8.2. Métodos iterativos para resolver sistemas lineales

Los métodos iterativos comienzan con una aproximación $\mathbf{x}^{(0)}$ a la solución **x** y genera una sucesión de vectores $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ que converge en **x.** Los métodos iterativos conllevan un proceso que convierte el sistema ***A*x = b** en otro equivalente **x = *T*x +c** para alguna matriz fija **T** y un

vector **c**. Luego de seleccionar el vector inicial $\mathbf{x}^{(0)}$ la sucesión de los vectores de la solución aproximada se genera calculando $\mathbf{x}^{(k)} = \mathbf{T}\mathbf{x}^{(k-1)} + \mathbf{c}$ para cada k = 1, 2, 3, ...

Cuando se tienen matrices no densas, en especial las matrices llamadas bandeadas de ancho de banda *p* y en general aquellas con una matriz de coeficientes con un alto porcentaje de elementos cero convienen utilizar los métodos iterativos tratados aquí. L*os* métodos iterativos rara vez se usan para resolver sistemas lineales de pequeña magnitud, ya que el tiempo necesario para conseguir una exactitud satisfactoria rebasa el que requieren los métodos directos, como el de la eliminación gaussiana. Sin embargo, en el caso de sistemas grandes con un alto porcentaje de elementos cero, son eficientes tanto en almacenamiento de computadora como en tiempo de cómputo. Este tipo de sistemas se encuentra con frecuencia en los análisis de circuitos y en la solución numérica de los problemas con valor en la frontera y de ecuaciones diferenciales parciales.

2.2.8.2.1. Método iterativo de Jacobi

Este método consiste en resolver las i-ésima ecuación en ***A*x = b** para x_i a fin de obtener (a condición de que $a_{ii} \neq 0$)

$$x_i = \sum_{\substack{j=i \\ j \neq i}}^{n} \left(\frac{-a_{ij}\, x_j}{a_{ii}} \right) + \frac{b_i}{a_{ii}} \text{ , para } i = 1, 2, \ldots, n.$$

y genera cada $x_i^{(k)}$ a partir de los componentes de $\mathbf{x}^{(k-1)}$ cuando $k \geq 1$ por medio de

$$x_i^{(k)} = \sum_{\substack{j=i \\ j \neq i}}^{n} \left(\frac{-a_{ij}\, x_j^{(k-1)}}{a_{ii}} \right) + \frac{b_i}{a_{ii}}, \text{ para } i = 1, 2, \ldots, n.$$

El método se describe en la forma $\mathbf{x}^{(k)} = \mathbf{T}\mathbf{x}^{(k-1)} + \mathbf{c}$ dividiendo ***A*** en sus partes diagonales y fuera de la diagonal. Para comprobar esto sea ***D*** la matriz diagonal cuya diagonal es la misma

que ***A***, sea ***-L*** la parte estrictamente triangular inferior de la parte ***A*** y sea ***-U*** la parte estrictamente triangular superior de ***A***. Con esta notación, ***A*** se divide en

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & & a_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{bmatrix} - \begin{bmatrix} 0 & 0 & \cdots & 0 \\ -a_{21} & 0 & \cdots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ -a_{n1} & \cdots & -a_{n,n-1} & 0 \end{bmatrix} - \begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ 0 & 0 & \cdots & \vdots \\ \vdots & \ddots & \ddots & -a_{n-1,n} \\ 0 & \cdots & 0 & 0 \end{bmatrix}$$

$$= \mathbf{D} \qquad - \mathbf{L} \qquad - \mathbf{U}$$

Y entonces transformamos la ecuación ***Ax* = b, o (*D* - *L* - *U*)x = b**, en

$$\boldsymbol{Dx = (L + U)x + b}$$

Y, finalmente, en

$$\mathbf{x} = \boldsymbol{D^{-1}(L + U)x} + \boldsymbol{D^{-1}}\mathbf{b}.$$

Esto da origen a la forma matricial del método iterativo de Jacobi:

$$\mathbf{x^{(k)}} = \boldsymbol{D^{-1}(L + U)}\mathbf{x^{(k-1)}} + \boldsymbol{D^{-1}}\mathbf{b}, \; k = 1, 2, \ldots$$

Al introducir la notación $\boldsymbol{T_j = D^{-1}(L + U)}$ y $\mathbf{c_j} = \boldsymbol{D^{-1}}\mathbf{b}$, esta ecuación toma la forma:

$$\mathbf{x^{(k)}} = \boldsymbol{T_j}\mathbf{x^{(k-1)}} + \mathbf{c_j}.$$

Esta ecuación se emplea, solamente, con fines teóricos. En la practica se emplea la primera ecuación que se dedujo.

Veamos el algoritmo de Jacobi:

Refiérase al capítulo de diseño lógico del software.

Algunas consideraciones con este algoritmo son: el paso 3 del algoritmo requiere $a_{ii} \neq 0$ para cada *i =1, 2, . . ., n.* Si uno de los elementos a_{ii} es cero y si el sistema es no singular, podemos reordenar las ecuaciones de modo que ningún $a_{ii} = 0$. Si queremos acelerar la convergencia, debemos arreglar las ecuaciones de modo que a_{ii} sea lo más grande posible.

Otro posible criterio de interrupción en el paso 4 consiste en iterarlo hasta que

$$\frac{\left\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right\|}{\left\|\mathbf{x}^{(k)}\right\|} < \varepsilon .$$

2.2.8.2.2. Método iterativo de Gauss-Seidel

Haciendo un análisis de la ecuación en el planteamiento de el método de Jacobi sugiere una mejora: Si queremos calcular $x_i^{(k)}$, utilizamos los componentes de $\mathbf{x}^{(k-1)}$**.** Como para *i > 1*, ya se calcularon $x_1^{(k)},\ldots,x_{i-1}^{(k)}$ y probablemente sean mejores aproximaciones de las soluciones reales $x_1,\ldots,x_{i-1}$ que $x_1^{(k-1)},\ldots,x_{i-1}^{(k-1)}$, parece más razonable calcular $x_i^{(k)}$ por medio de los valores calculados más recientemente, o sea

$$x_i^{(k)} = \frac{-\sum_{j=1}^{i-1}(a_{ij}\, x_j^{(k)} - \sum_{j=i+1}^{n}(a_{ij}\, x_j^{(k-1)}) + b_i}{a_{ii}} \text{ , para cada } i = 1, 2, \ldots, n.$$

Si queremos expresar el método de Gauss-Seidel en forma matricial, multiplicaremos ambos lados de la ecuación por a_{ii} y reunimos todos los *k*-ésimos términos de iteración, lo que nos da

$$a_{i1}\, x_1^{(k)} + a_{i2}\, x_2^{(k)} + \cdots + a_{ii}\, x_i^{(k)} = -a_{i,i+1}\, x_{i+1}^{(k-1)} - \cdots - a_{in}\, x_n^{(k-1)} + b_i ,$$

para cada *i = 1, 2, ...,n.* Al escribir todas las *n* ecuaciones obtenemos

$$\begin{aligned}
a_{11}\, x_1^{(k)} &= -a_{12}\, x_2^{(k-1)} - a_{13}\, x_3^{(k-1)} - \cdots - a_{1n}\, x_n^{(k-1)} + b_1 ,\\
a_{21}\, x_1^{(k)} + a_{22}\, x_2^{(k)} &= -a_{23}\, x_3^{(k-1)} - \cdots - a_{2n}\, x_n^{(k-1)} + b_2 ,\\
&\vdots\\
a_{n1}\, x_1^{(k)} + a_{n2}\, x_2^{(k)} + \cdots + a_{nn}\, x_n^{(k)} &= b_n ;
\end{aligned}$$

y se deduce que la forma matricial del método de Gauss-Seidel es

$$(\boldsymbol{D} - \boldsymbol{L})\mathbf{x}^{(k)} = \boldsymbol{U}\mathbf{x}^{(k-1)} + \mathbf{b}$$

o bien

$$\mathbf{x}^{(k)} = (\boldsymbol{D} - \boldsymbol{L})^{-1}\boldsymbol{U}\mathbf{x}^{(k-1)} + (\boldsymbol{D} - \boldsymbol{L})^{-1}\mathbf{b}, \text{ para } k = 1, 2, \ldots$$

Si usamos $\boldsymbol{T_g} = (\boldsymbol{D} - \boldsymbol{U})^{-1}\,\boldsymbol{U}$ y $\mathbf{c_g} = (\boldsymbol{D} - \boldsymbol{U})^{-1}\mathbf{b}$, esta técnica tiene la forma:

$$\mathbf{x}^{(k)} = \boldsymbol{T}_g\mathbf{x}^{(k-1)} + \mathbf{c_g}.$$

Para que la matriz triangular inferior $\boldsymbol{D} - \boldsymbol{L}$ sea no singular, es necesario y suficiente que $a_{ii} \neq 0$ para cada $i = 1, 2, \ldots, n$.

Veamos el algoritmo de Gauss-Seidel:

Refiérase al capítulo de diseño lógico del software.

Los comentarios que acompañan al algoritmo de Jacobi respecto a los criterios de ordenación e interrupción, también se aplican al algoritmo Gauss-Seidel. Este método, generalmente, es superior al de Jacobi. Sin embargo, hay sistemas lineales donde el método de Jacobi converge y el de Gauss-Seidel no, y hay otros éste converge y aquél no.

2.2.8.3. Criterios de convergencia de los métodos generales de iteración

Consideremos la siguiente fórmula

$$\mathbf{x}^{(k)} = \boldsymbol{T}\mathbf{x}^{(k-1)} + \mathbf{c}, \text{ para cada } k = 1, 2, \ldots,$$

donde $\mathbf{x}^{(o)}$ es arbitrario.

Veamos el siguiente teorema:

Teorema.

Para cualquier $\mathbf{x}^{(0)} \in R^n$, la sucesión $\left\{\mathbf{x}^{(k)}\right\}_{k=0}^{\infty}$ definida por

$$\mathbf{x}^{(k)} = \boldsymbol{T}\mathbf{x}^{(k-1)} + \mathbf{c}, \text{ para cada } k = 1, 2, \ldots,$$

converge en la solución única de $\mathbf{x} = \boldsymbol{T}\mathbf{x} + \mathbf{c}$ si y sólo si el radio espectral $\rho(\mathbf{T}) < 1$.

Corolario.

Si $\|\mathbf{T}\|<1$, para toda norma matricial normal y si **c** es un vector cualquiera, entonces la sucesión $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ definida por $\mathbf{x^{(k)} = Tx^{(k-1)} + c}$ converge, para cualquier $\mathbf{x}^{(0)} \in R^n$, y las siguientes cotas de error son válidas:

(i) $$\|\mathbf{x} - \mathbf{x}^{(k)}\| \leq \|\mathbf{T}\|^k \|\mathbf{x}^{(0)} - \mathbf{x}\|$$

(ii) $$\|\mathbf{x} - \mathbf{x}^{(k)}\| \leq \frac{\|\mathbf{T}\|^k}{1-\|\mathbf{T}\|} \|\mathbf{x}^{(0)} - \mathbf{x}\|$$

Ahora podemos proporcionar condiciones de suficiencia para la convergencia de los métodos de Jacobi y de Gauss-Seidel.

Teorema.

Si ***A*** es estrictamente dominante en sentido diagonal, entonces con cualquier elección de $\mathbf{x^{(0)}}$, tanto el método de Jacobi como el de Gauss-Seidel dan las soluciones $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ que convergen a la solución única $\boldsymbol{Ax = b}$.

No existen resultados generales que nos digan cuál de los dos métodos, si el de Jacobi o el de Gauss-Seidel, será más eficaz en un sistema lineal arbitrario. Sin embargo, en casos especiales sí conocemos la respuesta, como se demuestra en el siguiente teorema.

Teorema (Stein-Rosenberg)

Si $a_{ij} \leq 0$ para cada $i \neq j$ y si $a_{ij} > 0$ para cada $i = 1, 2, \ldots, n$, donde será, entonces será válida una y sólo una de las siguientes afirmaciones:

a) $0 < \rho(\mathbf{T}_g) < \rho(\mathbf{T}_j) < 1.$

b) $1 < \rho(\mathbf{T}_j) < \rho(\mathbf{T}_g).$

c) $\rho(\mathbf{T}_g) = \rho(\mathbf{T}_j).$

d) $\rho(\mathbf{T}_g) = \rho(\mathbf{T}_j) = 1.$

En el caso especial que se describe en este teorema, vemos en la parte (a) que cuando un método da convergencia, entonces ambos la dan, y el método de Gauss-Seidel converge más rápidamente que el de Jacobi. La parte (b) indica que, cuando un método diverge, entonces ambos divergen, y la divergencia es más pronunciada en el de Gauss-Seidel.

2.2.8.3.1. Método de SOR (Succesive Over-Relaxation)

Una forma de seleccionar un procedimiento que acelere la convergencia consiste en seleccionar un método cuya matriz asociada tenga un radio espectral mínimo, ya que la rapidez de convergencia de un procedimiento depende del radio espectral de la matriz relacionada con el método; por ello. Antes de explicar el método de SOR se debe de explicar un medio para medir el grado en que una aproximación de la solución de un sistema lineal difiere de la verdadera solución. Este método usa el vector que se describe en la siguiente definición.

Definición.

Supongamos que $\tilde{\mathbf{x}} \in R^n$ es una aproximación a la solución del sistema lineal definido por $\boldsymbol{A}\mathbf{x} = \mathbf{b}$. El **vector residual** de $\tilde{\mathbf{x}}$ respecto a este sistema es $\mathbf{r} = \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}$.

En procedimientos como el método de Jacobi o de Gauss-Seidel, un vector residual se asocia a cada cálculo de una aproximación al vector solución. El método tiene por objeto generar una sucesión de aproximaciones que harán que los vectores residuales asociados converjan rápidamente a cero. Supóngase que mediante el vector

$$\mathbf{r}_i^{(k)} = (r_{1i}^{(k)}, \ldots, r_{ni}^{(k)})^t$$

denotamos al vector residual del método de Gauss-Seidel correspondiente al vector solución $\mathbf{x}_i^{(k)}$ aproximado definido por

$$\mathbf{x}_i^{(k)} = (x_1^{(k)}, x_2^{(k)}, \ldots, x_{i-1i}^{(k)}, x_I^{(k-1)}, \ldots, x_n^{(k-1)})^t .$$

La *m*-ésima componente de $\mathbf{r}_i^{(k)}$ es

$$r_{mi}^{(k)} = b_m - \sum_{j=1}^{i-1} a_{mj}\, x_j^{(k)} - \sum_{j=i}^{n} a_{mj}\, x_j^{(k-1)},$$

o, en forma equivalente,

$$r_{mi}^{(k)} = b_m - \sum_{j=1}^{i-1} a_{mj}\, x_j^{(k)} - \sum_{j=i+1}^{n} a_{mj}\, x_j^{(k-1)} - a_{mi}\, x_i^{(k-1)},$$

para toda $m = 1, 2, \ldots, n$.

En particular, la i-ésima componente de $\mathbf{r}_i^{(k)}$ es

$$r_{ii}^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}\, x_j^{(k-1)} - a_{ii}\, x_i^{(k-1)};$$

así que

$$a_{ii}\, x_i^{(k-1)} + r_{ii}^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}\, x_j^{(k-1)}$$

Pero no olvidemos que, en el método de Gauss-Seidel, se decide que $x_i^{(k)}$ sea

$x_i^{(k)} = \dfrac{-\sum_{j=1}^{i-1}(a_{ij}\, x_j^{(k)} - \sum_{j=i+1}^{n}(a_{ij} x_j^{(k-1)}) + b_i}{a_{ii}}$, de modo que la ecuación puede rescribirse así

$$a_{ii}\, x_i^{(k-1)} + r_{ii}^{(k)} = a_{ii}\, x_i^{(k)}$$

De este modo, el método de Gauss-Seidel puede caracterizarse escogiendo $x_i^{(k)}$ que satisfaga

$$x_i^{(k)} = x_i^{(k-1)} + \frac{r_{ii}^{(k)}}{a_{ii}}.$$

Veamos otra conexión entre los vectores residuales y el método de Gauss-Seidel. Consideremos el vector residual $\mathbf{r}_{i+1}^{(k)}$, asociado al vector $\mathbf{x}_{i+1}^{(k)} = (x_1^{(k)}, x_2^{(k)}, \ldots, x_{ii}^{(k)}, x_{i+1}^{(k-1)}, x_i^{(k-1)}, \ldots, x_n^{(k-1)})^t$. De lo anterior se tiene que i-ésima componente de $\mathbf{r}_{i+1}^{(k)}$ es

$$\begin{aligned} r_{i,i+1}^{(k)} &= b_i - \sum_{j=1}^{i} a_{ij}\, x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}\, x_j^{(k-1)} \\ &= b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}\, x_j^{(k-1)} - a_{ii}\, x_i^{(k)}. \end{aligned}$$

Esta ecuación, como podemos observar, implica que $r_{i+1}^{(k)} = 0$. Así pues, en cierto modo el método de Gauss-Seidel se caracteriza también por seleccionar $x_i^{(k)}$ de manera que la i-ésima componente de $\mathbf{r}_{i+1}^{(k)}$ sea cero.

Pero, el reducir a cero una coordenada del vector residual no suele ser la forma más eficiente de disminuir el tamaño global del vector $\mathbf{r}_{i+1}^{(k)}$. Sino que, debemos seleccionar $x_i^{(k)}$ de manera que $\left\|\mathbf{r}_{i+1}^{(k)}\right\|$ sea pequeño.

Al modificar la ecuación de Gauss Seidel con componentes residuales como sigue

$$x_i^{(k)} = x_i^{(k-1)} + \omega \frac{r_{ii}^{(k)}}{a_{ii}} \text{ , que puede reformularse como}$$

$$x_i^{(k)} = (1-\omega)\, x_i^{(k-1)} + \frac{\omega}{a_{ii}}\left[b_i + \sum_{j=1}^{i-1} a_{ij}\, x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}\, x_j^{(k-1)} \right]$$

para ciertas opciones de ω positivo la norma del vector residual se reduce, y se logra una convergencia que es significativamente más acelerada. A los métodos que contienen esta ecuación se les llama **métodos de relajación**. En aquellas selecciones de ω con $0 < \omega < 1$, reciben el nombre de **métodos de subrelajación**. Para las selecciones de ω con $1 < \omega$, a los procedimientos se les denomina **métodos de sobrerrelajación**. Estos procedimientos se designan con la abreviatura **SOR (Succesive Over-Relaxation**, sobrerrelajación sucesiva) y son de gran utilidad en la resolución de sistemas lineales que se presentan en la solución numérica de algunas ecuaciones diferenciales parciales.

Para determinar la forma matricial del método SOR, rescribimos lo anterior como

$$a_{ii}\, x_i^{(k)} + \omega \sum_{j=1}^{i-1} a_{ij}\, x_j^{(k)} = (1-\omega)\, a_{ii}\, x_i^{(k-1)} - \omega \sum_{j=i+1}^{n} a_{ij}\, x_j^{(k-1)} + \omega\, b_i$$

así que

$$(\mathbf{D} - \omega\mathbf{L})\,\mathbf{x}^{(k)} = \left[(1-\omega)\mathbf{D} + \omega\mathbf{U}\right]\mathbf{x}^{(k-1)} + \omega\mathbf{b}$$

o de otra manera

$$\mathbf{x}^{(k)} = (\mathbf{D} - \omega\mathbf{L})^{-1}\left[(1-\omega)\mathbf{D} + \omega\mathbf{U}\right]\mathbf{x}^{(k-1)} + \omega\,(\mathbf{D} - \omega\mathbf{L})^{-1}\mathbf{b}\,.$$

Si utilizamos $\mathbf{T}_\omega = (\mathbf{D} - \omega\mathbf{L})^{-1}\left[(1-\omega)\mathbf{D} + \omega\mathbf{U}\right]$ y $\mathbf{c}_\omega = \omega\,(\mathbf{D} - \omega\mathbf{L})^{-1}\mathbf{b}$ podemos expresar el método SOR en la forma

$$\mathbf{x}^{(k)} = \mathbf{T}_\omega\,\mathbf{x}^{(k-1)} + \mathbf{c}_\omega\,.$$

Veamos el algoritmo de SOR:

Refiérase al capítulo de diseño lógico del software.

2.2.9. Área de aproximación de funciones

2.2.9.1. Teoría elemental

En esta área de la teoría de la aproximación se dan dos tipos generales de problemas: Uno se presenta cuando una función se da de manera explicita, pero queremos encontrar un tipo más simple de ella. Un polinomio, por ejemplo que nos sirva para determinar los valores aproximados de una función dada. El segundo problema de la teoría se refiere a la adaptación o ajuste de la función a ciertos datos y a la búsqueda de la función optima en una clase que podamos emplear para representar los datos de la mejor manera posible.

2.2.9.1.1. Aproximación polinomial trigonométrica

El uso de las series de las funciones seno y coseno para representar funciones arbitrarias comenzó En la década de 1750 comenzó el uso o aplicación de las series de las funciones seno y coseno para representar funciones arbitrarias, con el estudio del movimiento de un resorte en vibración. Este problema fue estudiado por Jean d'Alembert y luego también por el matemático más destacado de la época, Leonhard Euler, sin embargo fue Daniel Bernoulli el primero en proponer el uso de sumas infinitas de senos y cósenos como solución al problema, sumas que ahora conocemos con el nombre de series de Fourier. A principios del siglo XIX, Jean Baptiste Joseph Fourier las utilizó para estudiar el flujo de calor y formuló una teoría muy completa sobre el tema.

En el desarrollo de las series de Fourier, primero debemos observar que, para todo entero positivo *n*, el conjunto de las funciones $\{\phi_0, \phi_1, ..., \phi_{2n-1}\}$ donde

$$\phi_o(x) = \frac{1}{2},$$

$$\phi_k(x) = \cos kx, \text{ para cada } k = 1, 2, \ldots, n$$

y

$$\phi_{n+k}(x) = sen(kx), \text{ para cada } k = 1, 2, \ldots, n\text{-}1$$

es un conjunto ortogonal en $[-\pi,\pi]$ con respecto a $w(x)\equiv 1$. Esta ortogonalidad se deduce del hecho de que, para todo entero *j*, las integrales de sen *jx* y cos *jx* en $[-\pi,\pi]$ son 0, y podemos rescribir los productos de las funciones seno y coseno como sumas utilizando las siguientes tres identidades trigonométricas

$$sen(t_i)sen(t_2) = \frac{1}{2}[\cos(t_1 - t_2) - \cos(t_1 + t_2)],$$
$$\cos(t_i)\cos(t_2) = \frac{1}{2}[\cos(t_1 - t_2) + \cos(t_1 + t_2)],$$
$$sen(t_i)\cos(t_2) = \frac{1}{2}[sen(t_1 - t_2) + sen(t_1 + t_2)].$$

Sea τ_n el conjunto de todas las combinaciones lineales de las funciones $\phi_0, \phi_1, ..., \phi_{2n-1}$. A este conjunto se les denomina conjunto de **polinomios trigonométricos** de grado menor o igual que *n*. (Algunos autores incluyen una adicional en el conjunto, $\phi_{2n}(x) = sen(nx)$. Aquí no se seguirá esa convención)

Queremos obtener la aproximación de *mínimos cuadrados continuos* mediante las funciones de τ_n, para una función $f \in \mathbf{C}[-\pi,\pi]$, en la forma

$$S_n = \frac{a_0}{2} + a_n \cos nx + \sum_{k=1}^{n-1}(a_k \cos kx + b_k \sin kx).$$

Puesto que el conjunto de funciones $\{\phi_0, \phi_1, ..., \phi_{2n-1}\}$ es ortogonal en $[-\pi,\pi]$ respecto a $w(x)\equiv 1$, se deduce (por teorema) que la elección apropiada de coeficientes es

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(kx)dx, \text{ para cada } k = 1, 2, \ldots, n$$

y

$$b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)sen(kx)dx, \text{ para cada } k = 1, 2, \ldots, n - 1.$$

El límite de $S_n(x)$ cuando $n \to \infty$ recibe el nombre de **serie de Fourier** para la función *f*.

Existe un análogo discreto de mucha utilidad para la aproximación de *mínimos cuadrados discretos* y para la interpolación de grandes cantidades de datos.

Vamos a suponer que tenemos u conjunto de $2m$ puntos de parejas de datos $\{(x_j, y_j)\}_{j=0}^{2m-1}$ y además, que los primeros elementos de los pares dividen uniformemente un intervalo cerrado.

Supondremos que el intervalo es $[-\pi, \pi]$, para efectos pedagógicos así que

$x_j = -\pi + \left(\frac{j}{m}\right)\pi$, para cada $j = 0, 1, \ldots, 2m - 1$.

Si esto no fuera el caso, podríamos emplear una transformación lineal simple para traducir los datos en esta forma. $t_i = \frac{2\pi(x_i - a)}{b - a} - \pi$ donde $x_i \in [a, b]$.

Para el caso discreto se tiene como objetivo determinar el polinomio trigonométrico $S_n(x)$ en τ_n que reduzca al mínimo

$$E(S_n) = \sum_{j=0}^{2m-1} [y_j - S_n(x_j)]^2.$$

Para este fin necesitamos seleccionar las constantes $a_0, a_1, \ldots, a_n, b_1, b_2, \ldots, b_{n-1}$ de modo que

$$E(S_n) = \sum_{j=0}^{2m-1} \left[y_j - \left[\frac{a_0}{2} + a_n \cos n x_j + \sum_{k=1}^{n-1} (a_k \cos k x_j + b_k \sin k x_j) \right] \right]^2.$$

La determinación de las constantes se simplifica por le hecho de que, también se tiene un conjunto $\{\phi_0, \phi_1, \ldots, \phi_{2n-1}\}$ que es ortogonal respecto a la suma en puntos uniformemente espaciados $\{(x_j)\}_{j=0}^{2m-1}$ en $[-\pi, \pi]$. Con ello se quiere decir que para cada $k \neq l$,

$$\sum_{j=0}^{2m-1} \phi_k(x_j)\, \phi_l(x_j) = 0.$$

Teorema.

Las constantes de la sumatoria

$$S_n = \frac{a_0}{2} + a_n \cos nx + \sum_{k=1}^{n-1} (a_k \cos kx + b_k \sin kx).$$

que reducen al mínimo la suma de mínimos cuadrados

$$E(a_0, a_1,\ldots,a_n, b_1, b_2,\ldots,b_{n-1}) = \sum_{j=0}^{2m-1}\left[y_j - S_n(x_j)\right]^2.$$

son

$$a_k = \frac{1}{m}\sum_{j=0}^{2m-1} y_j \cos(k\,x_j), \text{ para cada } k = 1, 2, \ldots, n$$

y

$$b_k = \frac{1}{m}\sum_{j=0}^{2m-1} y_j \sin(k\,x_j), \text{ para cada } k = 1, 2, \ldots, n - 1.$$

2.2.9.2. Método de la transformada rápida de Fourier (TRF)

Acabamos de presentar la forma del polinomio trigonométrico de mínimos cuadrados discretos de *n*-ésimo grado en los 2*m* puntos $\{(x_j, y_j)\}_{j=0}^{2m-1}$ donde $x_j = -\pi + \left(\frac{j}{m}\right)\pi$, para cada *j* = 0, 1, …, 2*m* -1.

El polinomio trigonométrico *interpolante* de τ_m en estos dos 2*m* puntos de datos es casi el mismo que el polinomio de mínimos cuadrados. Esto se debe a que el polinomio trigonométrico de mínimos cuadrados reduce al mínimo el término de error

$$E(S_m) = \sum_{j=0}^{2m-1}\left[y_j - S_m(x_j)\right]^2$$

y este error es cero con respecto al polinomio interpolante; de ahí que se disminuya al mínimo cuando $S_m(x_j) = y_i$ para cada *j* = 0, 1, …, 2*m* -1. Pero, se necesita modificar la forma del polinomio, si queremos que los coeficientes asuman la misma forma que en el caso de los mínimos cuadrados. Se tiene que, si *r* no es múltiplo de *m*, entonces

$$\sum_{j=0}^{2m-1}\left(\cos r\,x_j\right)^2 = m.$$

La interpolación requiere calcular

$$\sum_{j=0}^{2m-1}\left(\cos m\,x_j\right)^2,$$

lo cual tiene el valor 2*m*. Para esto es necesario escribir el polinomio interpolante así

$$S_m = \frac{a_0 + a_m \cos mx}{2} + \sum_{k=1}^{m-1} (a_k \cos kx + b_k \sin kx)$$

si queremos que la forma de las constantes a_k y b_k concuerde con las del polinomio de mínimo cuadrados discretos. Por tanto, queremos que las constantes sean

$$a_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \cos(k\, x_j), \text{ para cada } k = 1, 2, \ldots, m$$

y

$$b_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \sin(k\, x_j), \text{ para cada } k = 1, 2, \ldots, m - 1.$$

Cuando se interpolan una gran cantidad de datos uniformemente espaciados por medio de polinomios trigonométricos, puede dar resultados muy exactos. Es el método adecuado de aproximación que tiene aplicación en áreas como las de filtros digitales, patrones de campo de antena, la mecánica cuántica y la óptica, así como muchas áreas relacionadas con los problemas de simulación. Pero, hasta mediados de los años 60 el método no tenía gran aplicación debido a que se requerían muchos cálculos aritméticos en la determinación de las constantes de aproximación. La interpolación de *2m* puntos de datos mediante la técnica de cálculo directo requiere cerca de $(2m)^2$ multiplicaciones y $(2m)^2$ sumas. El error de redondeo asociado a tal cantidad de cálculos generalmente domina la aproximación.

Sin embargo en 1965 un trabajo de J. W. Cooley y J. W. Tukey, publicado en la revista *Mathematics for Computation,* describió otro método para calcular las constantes en el polinomio trigonométrico interpolante que computacionalmente es más eficiente . El método requiere apenas $O(\log_2 m)$ multiplicaciones y $O(\log_2 m)$ sumas, siempre y cuando *m* se elija en forma correcta. En el caso de un problema con miles de puntos de datos, esto disminuye en miles la cantidad de cálculos, en comparación con los millones que se emplean en el método directo. Éste realmente fue descubierto varios años antes de publicarse el trabajo de Cooley y Tukey, pero había pasado inadvertido.

El método de computación rápida presentado por Cooley y Tukey se conoce con el nombre de **algoritmo de Cooley-Tukey** o bien **algoritmo de la transformada rápida de Fourier (TRF)** y ha provocado un verdadera cambio en la utilización de los polinomios trigonométricos interpolantes. Consiste en organizar el problema de manera que el número de

puntos de datos a usar pueda factorizarse fácilmente, sobre todo en potencias de dos, como veremos más adelante.

En vez de evaluar directamente las constantes a_k y b_k, la transformada rápida de Fourier calcula los coeficientes complejos c_k en

$$\frac{1}{m}\sum_{k=0}^{2m-1} c_k e^{ikx},$$

donde

$$c_k = \sum_{j=0}^{2m-1} y_j e^{ik\pi j/m}, \text{ para cada } k = 0, 1, \ldots, 2m - 1.$$

Una vez que se han determinado las constantes c_k, a_k y b_k pueden recuperarse (Esto es, se realiza una antitransformada) usando el hecho de que, para cada $k = 0, 1, \ldots, m$,

$$\begin{aligned}\frac{1}{m}c_k(-1)^k &= \frac{1}{m}c_k e^{-i\pi k} = \frac{1}{m}\sum_{j=0}^{2m-1} y_j e^{ik\pi j/m} e^{-i\pi k} = \frac{1}{m}\sum_{j=0}^{2m-1} y_j e^{ik(-\pi+(\pi j/m))} \\ &= \frac{1}{m}\sum_{j=0}^{2m-1} y_j\left(\cos k\left(-\pi+(\pi j/m)\right)+i\sin k\left(-\pi+(\pi j/m)\right)\right) \\ &= \frac{1}{m}\sum_{j=0}^{2m-1} y_j\left(\cos k\, x_j + i\sin k\, x_j\right).\end{aligned}$$

Por consiguiente,

$$a_k + i\,b_k = \frac{(-1)^k}{m}c_k.$$

Con el fin de simplificar la notación, se agregan b_o y b_m al conjunto, pero ambos son cero y por ello no influyen en la suma resultante.

Veamos por qué la característica de reducción de operaciones de la transformada rápida de Fourier. Se debe al hecho de que para todo entero n,

$$e^{n\pi i} = \cos n\pi + i\sin n\pi = (-1)^n.$$

Supongamos $m = 2^p$ para algún entero positivo p. Para cada $k = 0, 1, \ldots, m\text{-}1$,

$$c_k + c_{m+k} = \sum_{j=0}^{2m-1} y_j e^{ik\pi j/m} + \sum_{j=0}^{2m-1} y_j e^{i(m+k)\pi j/m} = \sum_{j=0}^{2m-1} y_j e^{ik\pi j/m}\left(1 + e^{\pi ij}\right).$$

Sin embargo

$$1+e^{i\pi j}=\begin{cases}2, si\ \ j\ es\ par,\\ 0, si\ \ j\ es\ impar.\end{cases}$$

De esto, hay sólo *m* términos n ceros que deben sumarse. Si *j* se reemplaza con 2*j* en el índice de la suma, podemos escribir la suma como

$$c_k + c_{m+k} = \sum_{j=0}^{2m-1} y_j e^{ik\pi(2j)/m};\ (\alpha)$$

es decir,

$$c_k + c_{m+k} = \sum_{j=0}^{2m-1} y_{2j} e^{ik\pi j/(m/2)}.$$

De la misma manera,

$$c_k - c_{m+k} = 2e^{ik\pi/m} \sum_{j=0}^{2m-1} y_{2j+1} e^{ik\pi j/(m/2)}.$$

Puesto que c_k y c_{m+k} puede recuperarse de las ecuaciones precedentes, estas relaciones determinan todos los coeficientes c_k. Nótese que las sumas de las ecuaciones tienen la misma forma que la suma de la ecuación (α), salvo que el índice ha sido sustituido por *m*/2 lo cual significa que podemos aplicar el proceso de reducción a estos nuevos términos de manera recursiva.

Veamos el algoritmo de la transformada de Fourier:

Refiérase al capítulo de diseño lógico del software.

2.3. Determinación de requerimientos

Posterior a la investigación, a las sugerencias basadas en la experiencia de profesores y de las condiciones propias de un proyecto como éste, como el tiempo, recurso humano y financiero se procede a enlistar los siguientes requerimientos:

- Desarrollar una aplicación que implemente los siguientes métodos numéricos:
  - En problemas que impliquen sistemas de ecuaciones lineales se desarrollarán los siguientes métodos:
    - Iteraciones de Jacobi
    - Iteraciones de Gauss-Seidel
    - Método de sobre relajación sucesiva (SOR)
  - En problemas que impliquen ecuaciones diferenciales ordinarias se tratarán los siguientes:
    - El método de Runge Kutta-Fehldberg
    - Método de diferencias finitas.
  - Y en los que impliquen integración se abordaran los métodos:
    - Algoritmo de Romberg
    - Cuadratura de Gauss-Legendre
  - Y en problemas de aproximación:
    - Transformada rápida de Fourier
- El software debe de poder guardar el trabajo realizado con los diferentes métodos numéricos, en un archivo para propósitos pedagógicos e históricos.
- El software debe de imprimir el trabajo que se haya realizado, para propósitos pedagógicos.
- Debe de poder abrir un archivo previo para análisis posterior y seguimiento de trabajo.

- Proporcionar ayuda sobre cómo, por qué y cuándo se espera que funcionen los métodos en un sistema centralizado y ayuda sensible al contexto.
- Con todo lo anterior, se hace necesario que la aplicación posea una barra de menús con las opciones de archivo y otras opciones estándar para una aplicación en Windows. Además la barra de menú tendrá un menú especial para las opciones de los diferentes métodos a implementarse.
- Una barra de herramientas que proporcione un acceso directo a las opciones de menú más comunes. Esta deberá poder ocultarse.
- Una barra de estado que también podrá ocultarse.
- Como toda aplicación profesional basada en Windows y en otros sistemas deberá tener una pantalla de presentación al cargarse la aplicación en memoria.
- Proporcionar cálculos intermedios relevantes que nos proporcionen información sobre el éxito o fracaso de la aproximación que ayuden a comprender la implantación del método numérico.
- Requerimientos de los métodos iterativos para resolver sistemas de ecuaciones lineales:
    - Interfaces para poder capturar los datos de la matriz de coeficientes, el vector independiente y el vector de aproximación.
    - Mecanismos para poder almacenar y recuperar los datos de entrada.
    - Un analizador de expresiones matemáticas para poder introducir expresiones como la raíz cuadrada de un número, el numero e, y así por el estilo.
    - Mecanismo para proporcionar ayuda al usuario acerca de los parámetros de entrada y los resultados que se obtienen.
- Requerimientos del método de Runge-Kutta-Fehlberg
    - Un analizador de expresiones matemáticas para poder introducir el miembro derecho de una ecuación diferencial de primer orden que constituye una función de dos variables y expresiones matemáticas en los controles que requieren datos numéricos.

- Interfaces para poder capturar los datos de entrada del usuario y presentar los resultados del método numérico.
- Mecanismo para proporcionar ayuda al usuario acerca de los parámetros de entrada y los resultados que se obtienen.

- Requerimientos del método de diferencias finitas.
  - Un analizador de expresiones matemáticas para poder introducir los coeficientes (funciones en términos de la variable x) de la ecuación lineal diferencial de segundo orden y expresiones matemáticas en los controles que requieren datos numéricos.
  - Interfaces para poder capturar los datos de entrada del usuario y presentar los resultados del método numérico.
  - Mecanismo para proporcionar ayuda al usuario acerca de los parámetros de entrada y los resultados que se obtienen.
- Requerimientos del Algoritmo de Romberg
  - Un analizador de expresiones matemáticas para poder introducir la función a integrar y expresiones matemáticas en los controles que requieren datos numéricos.
  - Interfaces para poder capturar los datos de entrada del usuario y presentar los resultados del método numérico.
  - Mecanismo para proporcionar ayuda al usuario acerca de los parámetros de entrada y los resultados que se obtienen.
- Requerimientos de la cuadratura de Gauss-Legendre
  - Un analizador de expresiones matemáticas para poder introducir la función a integrar y expresiones matemáticas en los controles que requieren datos numéricos.
  - Interfaces para poder capturar los datos de entrada del usuario y presentar los resultados del método numérico.

  - Mecanismo para proporcionar ayuda al usuario acerca de los parámetros de entrada y los resultados que se obtienen.

- Requerimientos del método de la transformada rápida de Fourier
  - Interfaces para poder capturar los datos de entrada y para visualizar los resultados del método.
  - Mecanismos para poder almacenar y recuperar los datos de entrada.
  - Un analizador de expresiones matemáticas para poder introducir expresiones matemáticas en los controles que aceptan datos numéricos y la función a aproximar si este es el caso.
  - Mecanismo para proporcionar ayuda al usuario acerca de los parámetros de entrada y los resultados que se obtienen.

III. Diseño lógico del software

3.1. Introducción

En esta etapa se retoman las especificaciones sobre requerimientos que se listaron al final de la etapa anterior para traducirlos a especificaciones de diseño. En primer lugar se especificará la estructura lógica de la aplicación, detallando cada proceso con sus respectivos métodos o clases que pudiera utilizar. Se utilizará un proceso deductivo indicado por niveles para detallara los procesos en la estructura lógica de la aplicación. Enseguida se especifica un menú que será la vía de comunicación entre el usuario y la aplicación junto con todos sus procesos. Y finalmente se diseñan las pantallas que serán las interfaces de comunicación interactiva entre el usuario y los distintos métodos numéricos que se desarrollarán.

3.2. Estructura lógica de la aplicación

Primero analizaremos el nivel 1, que se muestra en la siguiente figura:

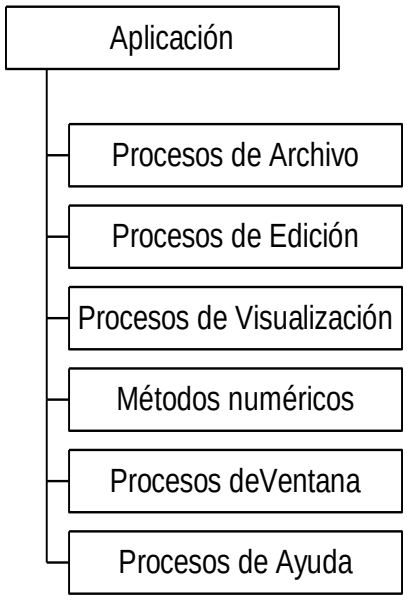


Ilustración 3:Nivel 1 de la estructura lógica de la aplicación

En este nivel se puede apreciar que muchos de los procesos son familiares. Esto se debe que las aplicaciones de Windows tienen procesos de manipulación estándares que son

indispensables en muchas aplicaciones, como por ejemplo: abrir archivo, guardar como, ver barra de herramientas, abrir ventana nueva, etc.

A continuación profundizaremos en la jerarquía que presenta cada grupo de procesos del nivel 1:

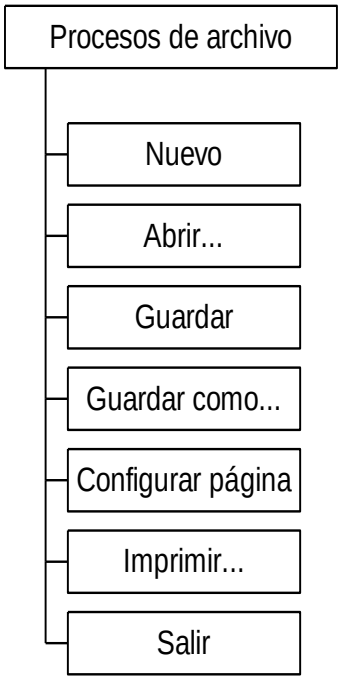


Ilustración 4: Nivel 2 de la estructura lógica de la aplicación. Procesos de archivo.

Estos procesos tienen las mismas funcionalidades que en muchas aplicaciones en Windows y disponen de las mismas interfaces que proporciona el sistema, por lo que le resultará familiar al usuario y de fácil utilización. Con estos procesos podemos manipular todo aquello especificado en la etapa de requerimientos en lo referente a archivos históricos y además nos proporciona una manera de salirnos de la aplicación.

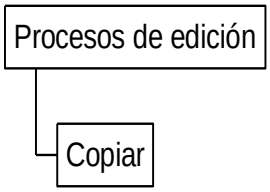


Ilustración 5:Nivel 2 de la estructura lógica de la aplicación. Procesos de edición.

Este elemento de la jerarquía del nivel 2 solo tiene un proceso debido a que el archivo que se va creando no es manipulable por parte del usuario, sino que la aplicación va generándolo de acuerdo a las acciones que el usuario realiza sobre los distintos métodos numéricos que el software implanta. La orden copiar utiliza el portapapeles para almacenar información que podrá compartir con otras aplicaciones, como usted bien sabe si es un usuario de Windows.

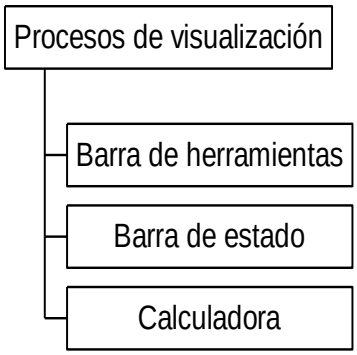


Ilustración 6: Nivel 2 de la estructura lógica de la aplicación. Procesos de visualización.

Pues como su nombre lo indica son procesos que permiten visualizar la barra de herramientas de la aplicación, la barra de estado y poder ejecutar la herramienta Calculadora en donde se pueden realizar operaciones simples y científicas siguiendo las reglas que aparecen en el manual del usuario.

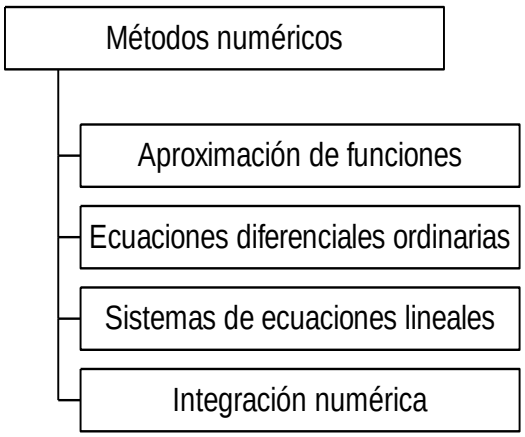


Ilustración 7: Nivel 2 de la estructura lógica de la aplicación. Métodos numéricos.

Como podrá haber adivinado estos son los procesos que conforman la razón de ser de la aplicación. Son, además, los procesos que definiremos más en detalle en el tercer nivel y sobre los cuales se enfoco la mayor parte del trabajo.

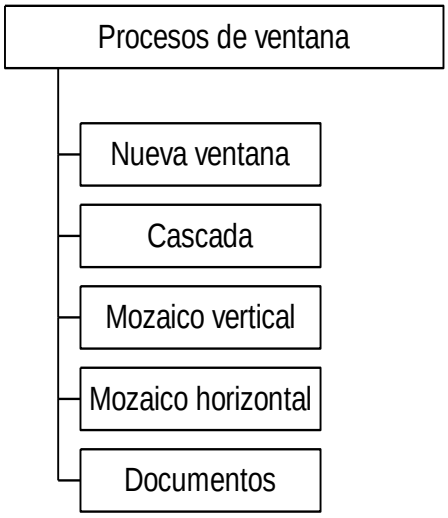


Ilustración 8: Nivel 2 de la estructura lógica de la aplicación. Procesos de ventana.

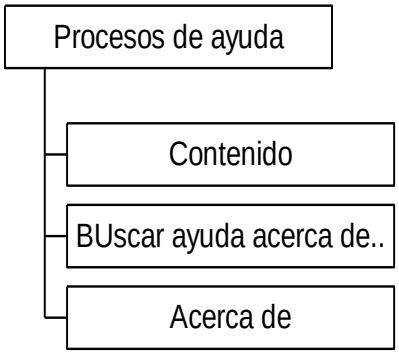


Ilustración 9: Nivel 2 de la estructura lógica de la aplicación. Procesos de ayuda.

Los procesos de ventana controlan las ventanas hijas de la aplicación MDI; y los procesos de ayuda proporcionan auxilio, llamando al sistema de ayuda de la aplicación, sobre teoría y práctica del mismo. Usted ya estará familiarizado con estos procesos si es un usuario de Windows.

A continuación detallaremos el nivel 3. En este nivel se presentan únicamente procesos de los métodos numéricos a implementar, ya que los los demás son especificados por el marco de la aplicación que gerera Visual Bacic, y es poco el tiempo, relativamente, dedicado para personalizarlos.

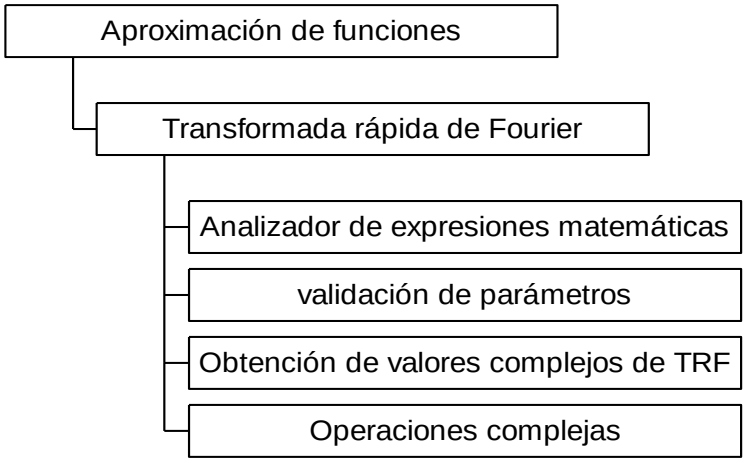


Ilustración 10: Nivel 3 de la estructura lógica de la aplicación. Métodos de Aproximación de funciones.

Veamos el algoritmo de la transformada rápida de Fourier:

3.2.1. Algoritmo de la transformada rápida de Fourier

Para calcular los coeficientes de la suma

$\frac{1}{m}\sum_{k=0}^{2m-1} c_k e^{ikx} = \frac{1}{m}\sum_{k=0}^{2m-1} c_k(\cos(kx) + isen(kx))$, donde $i = \sqrt{-1}$, para los datos $\{(x_j, y_j)\}_{j=0}^{2m-1}$

donde $m = 2^p$ y $x_j = -\pi + j\pi / m$ para *j = 0, 1, . . ., 2m-1*:

ENTRADA : *m, p*; $y_0, y_1, ..., y_{2m-1}$.

SALIDA : números complejos $c_0, c_1, ..., c_{2m-1}$; números reales $a_0, ..., a_m; b_0, ..., b_{m-1}$.

Paso 1 Tome

$M = m;$
$q = p;$
$\zeta = e^{\pi i / m}.$

Paso 2 Para $j = 0, 1, \ldots, 2m - 1$ tome $c_j = y_j$.

Paso 3 Para $j = 1, \ldots, M$ tome

$$\xi_j = \xi^j;$$
$$\xi_{j+M} = -\xi_j.$$

Paso 4 Tome

$$K = 0;$$
$$\xi_0 = 1.$$

Paso 5 Para $L = 1, 2, \ldots, p + 1$ haga los pasos 6-12.

Paso 6 Mientras $K < 2m - 1$ haga los pasos 7-11.

Paso 7 Para $j = 1, 2, \ldots, M$ haga los pasos 8-10.

Paso 8 Sea $K = k_p \cdot 2^p + k_{p-1} \cdot 2^{p-1} + \ldots + k_1 \cdot 2 + k_0;$

(Descomponga k)

Tome

$$K_1 = \frac{K}{2^q} = k_q \cdot 2^{p-q} + \ldots + k_{q+1} \cdot 2 + k_q;$$
$$K_2 = k_q \cdot 2^p + k_{q+1} \cdot 2^{p-1} + \ldots + k_p \cdot 2^q.$$

Paso 9 Tome

$$\eta = c_{K+M} + \xi_{K_2};$$
$$c_{K+M} = c_K - \eta;$$
$$c_K = c_K + \eta.$$

Paso 10 Tome $K = K + 1$.

Paso 11 Tome $K = K + M$.

Paso 12 Tome *K = 0;*

M = M/2 ;

q = q-1.

Paso 13 Mientras $K < 2m - 1$ haga los pasos 14-16.

Paso 14 Sea

$$K = k_p \cdot 2^p + k_{p-1} \cdot 2^{p-1} + \ldots + k_1 \cdot 2 + k_0;$$

(Descomponga K)

$$j = k_0 \cdot 2^p + k_1 \cdot 2^{p-1} + \ldots + k_{p-1} \cdot 2 + k_p.$$

Paso 15 Si $j > K$ entonces intercambie c_j y c_k.

Paso 16 Tome $K = K + 1$.

Paso 17 Tome

$$a_0 = c_0 / m;$$
$$a_m = \mathrm{Re}(e^{-i\pi j} c_j / m).$$

Paso 18 Para $j = 1, \ldots, m - 1$ tome $\begin{array}{l} a_j = \mathrm{Re}(e^{-i\pi j} c_j / m); \\ b_j = \mathrm{Im}(e^{-i\pi j} c_j / m). \end{array}$

Paso 19 SALIDA($c_0, \ldots, c_{2m-1}; a_0, \ldots, a_m; b_0, \ldots, b_{m-1}$);

PARE.

3.2.1.1. Algoritmo alternativo TRF recursiva

Este algoritmo se tomo como alternativa al que se presenta arriba

Propósito: Formar la transformada rápida de Fourier $\{\tilde{F}_0, \tilde{F}_1, \ldots, \tilde{F}_{N-1}\} = \Im\{f_0, f_1, \ldots, f_{N-1}\}$ a partir de un conjunto de N valores muestreados $\{f_0, f_1, \ldots, f_{N-1}\}$ en $[-\pi, \pi]$ cuando $N = N^p$.

ENTRADA : N; $f_0, f_1, \ldots, f_{N-1}$

SALIDA : $\{\tilde{F}_0, \tilde{F}_1, \ldots, \tilde{F}_{N-1}\} = \Im\{f_0, f_1, \ldots, f_{N-1}\}$

Si $N = 2$ entonces

Haga $\begin{array}{l} \tilde{F}_0 = f_0 + f_1; \\ \tilde{F}_1 = f_0 - f_1. \end{array}$

De lo contrario Haga

$$M = N / 2; \zeta = e^{i2\pi / N}$$

$$\{\tilde{E}_0, \tilde{E}_1, \ldots, \tilde{E}_{M-1}\} = \Im\{f_0, f_2, \ldots, f_{N-2}\}$$

$$\{\tilde{O}_0, \tilde{O}_1, \ldots, \tilde{O}_{M-1}\} = \Im\{f_1, f_3, \ldots, f_{N-1}\}$$

Para $k = 0$ hasta $M - 1$ *Hacer*

$$\tilde{F}_k = \tilde{E}_k + \zeta^k \tilde{O}_k ; \tilde{F}_{k+M} = \tilde{E}_k - \zeta^k \tilde{O}_k .$$

Retornar.

Analicemos, ahora, los métodos de ecuaciones diferenciales ordinarias:

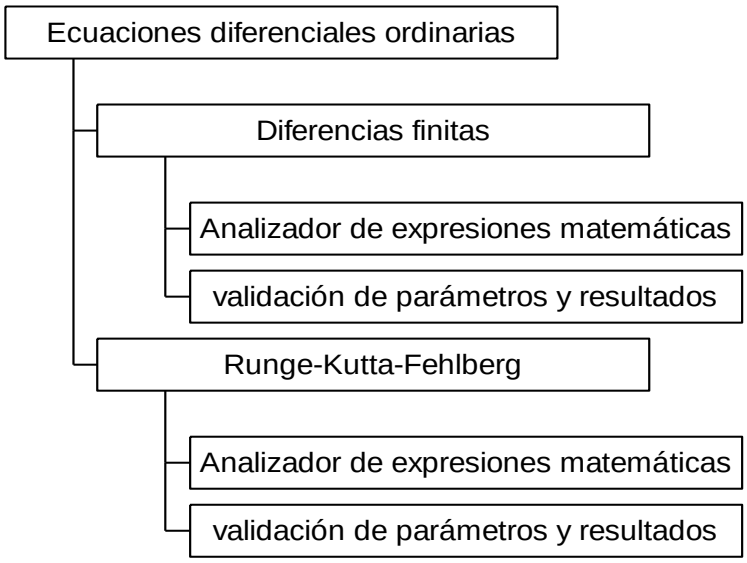


Ilustración 11: Nivel 3 de la estructura lógica de la aplicación. Métodos de ecuaciones diferenciales ordinarias.

Veamos el método de Runge-Kutta-Fehlberg:

3.2.1. Algoritmo de Runge-Kutta-Fehlberg

Para aproximar la solución del problema de valor inicial

$$y' = f(t, y), a \le t \le b, y(a) = \alpha ,$$

con un error local de truncamiento que no rebase la tolerancia especificada:

ENTRADA : extremos *a, b*; condición inicial α; tolerancia *TOL*; tamaño máximo de paso *hmáx*; tamaño mínimo de paso *hmín.*

SALIDA : *t, w, h* donde *w* aproxima a *y(t)* y se uso el tamaño de paso *h* o un mensaje de que se rebasó el tamaño mínimo de paso.

Paso 1 Tome $t = a$;

$w = \alpha$;

$h = hmáx$;

$BAND = 1$;

SALIDA(t, w).

Paso 2 Mientras ($BAND = 1$) haga pasos 3-11.

Paso 3 Tome

$$K_1 = hf(t, w);$$
$$K_2 = hf(t + \frac{1}{4}h, w + \frac{1}{4}K_1);$$
$$K_3 = hf(t + \frac{3}{8}, w + \frac{3}{32}K_1 + \frac{9}{32}K_2);$$
$$k_4 = hf(t + \frac{12}{13}h, w + \frac{1932}{2197}K_1 - \frac{7200}{2197}K_2 + \frac{7296}{2197}K_3);$$
$$K_5 = hf(t + h, w + \frac{439}{216}K_1 - 8K_2 + \frac{3680}{513}K_3 - \frac{845}{4104}K_4);$$
$$K_6 = hf(t + \frac{1}{2}h, w - \frac{8}{27}K_1 + 2K_2 - \frac{3544}{2565}K_3 + \frac{1859}{4104}K_4 - \frac{11}{40}K_6).$$

Paso 4 Tome $R = \frac{1}{h}\left|\frac{1}{360}K_1 - \frac{128}{4275}K_3 - \frac{2197}{75240}K_4 + \frac{1}{50}K_5 + \frac{2}{55}K_6\right|$.

(*Nota:* $R = \frac{1}{h}\left|\tilde{w}_{i+1} - w_{i+1}\right|$.)

Paso 5 Si $R \leq TOL$ entonces haga pasos 6 y 7.

Paso 6 Tome $t = t + h$; (*Aproximadamente aceptada)*

$$w = w + \frac{25}{216}K_1 + \frac{1408}{2565}K_3 + \frac{2197}{4104}K_4 - \frac{1}{5}K_5.$$

Paso 7 SALIDA (t, w, h).

Paso 8 Tome $\delta = 0.84(TOL / R)^{\frac{1}{4}}$.

Paso 9 Si $\delta \leq 0.1$ entonces tome *h = 0.1h*

o si $\delta \geq 4$ entonces tome *h = 4h*

o tome $h = \delta h$. (*calcule nuevo h.)*

Paso 10 Si *h > hmáx* entonces tome *h = hmáx*.

Paso 11 Si $t \geq b$ entonces tome *BAND = 0*

o si *t + h > b* entonces tome *h = b – t*

o si *h < hmín* entonces tome *BAND = 0;*

SALIDA (‘*rebasado h mínimo*’).

(Procedimiento terminado de manera no satisfactoria)

Paso 12 (El procedimiento se completó.)

PARE.

Y el método de diferencias finitas:

3.2.3. Algoritmo del Método de diferencias finitas

Para aproximar la solución del problema con valor en la frontera

$y'' = p(x)y' + q(x)y + r(x), a \leq t \leq b, y(a) = \alpha, y(b) = \beta$:

ENTRADA : extremos a, b; condición de frontera α, β; entero $N \geq 2$.

SALIDA : aproximaciones w_i a $y(x_i)$ para toda i = 0, 1, ...,N+1.

Paso 1 Tome

$$h = (b-a)/(N+1);$$
$$x = a + h$$
$$a_1 = 2 + h^2 q(x);$$
$$b_1 = -1 + (h/2)p(x);$$
$$d_1 = -h^2 r(x) + (1 + (h/2)p(x))a$$

Paso 2 Para i = 2, ..., N-1

Tome

$$x = a + ih$$
$$a_i = 2 + h^2 q(x);$$
$$b_i = -1 + (h/2)p(x);$$
$$c_i = -1 - (h/2)p(x);$$
$$d_i = -h^2 r(x)$$

Paso 3 Tome

$$x = b - h$$
$$a_N = 2 + h^2 q(x);$$
$$c_N = -1 - (h/2)p(x);$$
$$d_1 = -h^2 r(x) + (1 - (h/2)p(x))\beta$$

Paso 4 Tome

$$l_1 = a_1;$$
$$u_1 = b_1/a_1;$$
$$z_1 = d_1/l_1$$

(Pasos 4-8 resuelven un sistema lineal tridiagonal.)

Paso 5 Para i = 2, ..., N-1; Tome

$$l_i = a_i - c_i u_{i-1};$$
$$u_i = b_i/l_i;$$
$$z_1 = (d_1 - c_i z_{i-1}/l_1$$

Paso 6 Tome

$$l_N = a_N - c_N u_{N-1};$$
$$z_N = (d_N - c_N z_{N-1}) / l_N$$

Paso 7 Tome

$$w_0 = \alpha;$$
$$w_{N+1} = \beta;$$
$$w_N = z_N$$

Paso 8 Para i = N-1, ... , 1 tome $w_i = z_i - u_i w_{i+1}$

Paso 9 Para i = 0, ... , N+1 tome $x = a + ih$;

SALIDA (x, w_i)

Paso 10 PARE. *(Procedimiento terminado.)*

Pasemos, enseguida, a los procesos de sistemas de ecuaciones lineales:

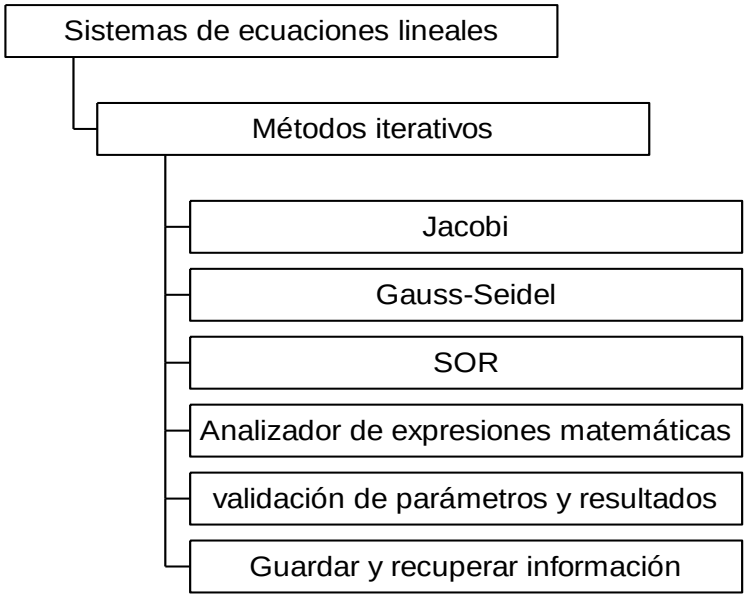


Ilustración 12: Nivel 3 de la estructura lógica de la aplicación. Métodos de Sistemas de ecuaciones lineales.

Veamos los algoritmos que corresponden el método de Jacobi, Gauss-Seidel y SOR:

3.2.4. Algoritmo de Jacobi

Para resolver ***A*x = b** dada una aproximación inicial $\mathbf{x}^{(0)}$**:**

ENTRADA: el número de ecuaciones y de incógnitas n; los elementos $a_{ij}, 1 \le i, j \le n$ de la matriz ***A***; los elementos $b_i, 1 \le i \le n$ de **b**; los elementos $XO_i, 1 \le i \le n$ de **XO = $\mathbf{x}^{(0)}$;** la tolerancia *TOL*; el número máximo de iteraciones *N*.

SALIDA: la solución aproximada $x_1,...,x_n$, o el mensaje de que se rebasó el número de iteraciones.

Paso 1 Tome *k = 1*.

Paso 2 Mientras $(k \le N)$ haga los pasos 3-6.

Paso 3 Para $i = 1, \ldots, n$

tome $x_i = \sum_{\substack{j=i \\ j \neq i}}^{n} \left(\frac{-a_{ij} XO_j}{a_{ii}} \right) + \frac{b_i}{a_{ii}}$.

Paso 4 Si la norma de los vectores $\|\mathbf{x} - \mathbf{XO}\| < TOL$ entonces

SALIDA ($x_1,...,x_n$); *(Procedimiento terminado exitosamente.)*

PARE.

Paso 5 Tome $k = k + 1$.

Paso 6 Para $i = 1, \ldots, n$ tome $XO_i = x_i$.

Paso 7 SALIDA ('número máximo de iteraciones excedido');

(Procedimiento terminado sin éxito.)

PARE.

3.2.5. Algoritmo de Gauss-Seidel

Para resolver ***A*x = b** dada una aproximación inicial $\mathbf{x}^{(0)}$**:**

ENTRADA: el número de ecuaciones e incógnitas n; los elementos $a_{ij}, 1 \le i, j \le n$ de la matriz **A**; los elementos $b_i, 1 \le i \le n$ de **b**; los elementos $XO_i, 1 \le i \le n$ de **XO** = $\mathbf{x}^{(0)}$**;** la tolerancia ***TOL***; el número máximo de iteraciones *N*.

SALIDA: la solución aproximada $x_1,...,x_n$, o el mensaje de que se rebasó el número de iteraciones.

Paso 1 Tome *k = 1*.

Paso 2 Mientras $(k \le N)$ haga los pasos 3-6.

Paso 3 Para *i = 1, . . ., n*

$$x_i = \frac{-\sum_{j=1}^{i-1}(a_{ij}\, x_j - \sum_{j=i+1}^{n}(a_{ij}\, XO_j) + b_i}{a_{ii}}.$$

Paso 4 Si la norma de los vectores $\|\mathbf{x} - \mathbf{XO}\| < TOL$ entonces

SALIDA ($x_1,...,x_n$); *(Procedimiento terminado exitosamente.)*

PARE.

Paso 5 Tome *k = k + 1*.

Paso 6 Para i = 1, . . ., n tome $XO_i = x_i$

Paso 7 SALIDA (‘número máximo de iteraciones excedido’);

(Procedimiento terminado sin éxito.)

PARE.

3.2.6. Algoritmo de SOR

Para resolver ***A*x = b** dada una aproximación inicial $\mathbf{x}^{(0)}$**:**

ENTRADA: el número de ecuaciones e incógnitas *n*; los elementos $a_{ij}, 1 \le i, j \le n$ de la matriz ***A***; los elementos $b_i, 1 \le i \le n$ de **b**; los elementos $XO_i, 1 \le i \le n$ de **XO** = $\mathbf{x}^{(0)}$**;** el parámetro ω; la tolerancia *TOL*; el número máximo de iteraciones *N*.

SALIDA: la solución aproximada $x_1, \ldots, x_n$ o el mensaje de que se rebasó el número de iteraciones.

Paso 1 Tome *k = 1*.

Paso 2 Mientras $(k \le N)$ haga los pasos 3-6.

Paso 3 Para *i = 1, . . ., n*

Tome $x_i = (1-\omega)\, XO_i + \dfrac{\omega}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j - \sum_{j=i+1}^{n} a_{ij}\, XO_j \right].$

Paso 4 Si la norma de los vectores $\|\mathbf{x} - \mathbf{XO}\| < TOL$ entonces
SALIDA ($x_1, \ldots, x_n$); *(Procedimiento terminado exitosamente.)*
PARE.

Paso 5 Tome *k = k + 1*.

Paso 6 Para *i = 1, . . ., n* tome $XO_i = x_i$.

Paso 7 SALIDA (‘número máximo de iteraciones excedido’);

(Procedimiento terminado sin éxito.)

PARE.

Los procesos que corresponden a la integración se sintetizan en la siguiente figura:

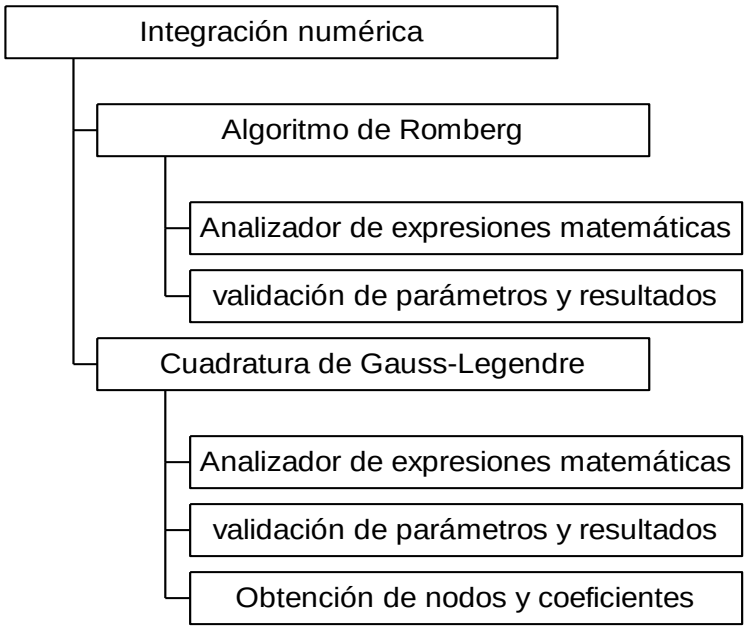


Ilustración 13: Nivel 3 de la estructura lógica de la aplicación. Métodos de integración numérica..

Veamos el algoritmo de Romberg que define las variables de entrada, el propósito y los resultados obtenidos:

3.2.7. Algoritmo de Romberg

Para aproximar la integral $I = \int_a^b f(x)dx$, seleccione un entero n > 0.

ENTRADA : extremos *a, b*; entero *n*.

SALIDA : un arreglo R. *(Calcule R por renglones; sólo los 2 últimos renglones se guardaran en almacenamiento*.)

Paso 1 Tome

$$h = b - a;$$
$$R_{1,1} = \frac{h}{2}(f(a) + f(b)).$$

Paso 2 SALIDA ($R_{1,1}$).

Paso 3 Para $i = 2, \ldots, n$ haga pasos 4-8.

Paso 4 Tome $R_{2,1} = \frac{1}{2}\left[R_{1,1} + h\sum_{k=1}^{2^{i-2}} f(a + (k - 0.5)h) \right]$.

(Aproximación con el método del trapecio)

Paso 5 Para $j = 2, \ldots, i$

Tome $R_{2,j} = R_{2,j-1} + \frac{R_{2,j-1} - R_{1,j-1}}{4^{j-1} - 1}$. (*Extrapolación*.)

Paso 6 SALIDA ($R_{2,j}$ por $j = 1, 2, \ldots, i$).

Paso 7 Tome $h = h/2$.

Paso 8 Para $j = 1, 2, \ldots, i$ tome $R_{1,j} = R_{2,j}$.

(*Actualice el renglón 1 de R.*.)

Paso 9 PARE.

Y el método de Gauss-Legendre:

3.2.8. Algoritmo de Gauss-Legendre

Para aproximar la integral $I = \int_a^b f(x)dx$, seleccione un entero n > 0.

ENTRADA : extremos *a*, *b*; entero *n*.

(*Las raíces* $r_{n,i}$ *y los coeficientes* $c_{n,i}$ *deben estar disponibles para i = 1, 2, ..., n.)*

SALIDA : Aproximación *J* de *I*.

Paso 1 Tome

$h_1 = (b - a)/2;$
$h_2 = (b + a)/2;$
$J = 0.$

Paso 2 Para $i = 0, 1, \ldots, n$ haga

$$x = h_1 r_{n,i} + h_2;$$
$$Q = f(x);$$
$$J = J + c_{n,i}.$$

Paso 3Tome $J = h_1 J$.

Paso 5 *SALIDA(J).*

PARE.

Diseño del menú principal

Como consecuencia de este análisis se diseño un menú que corresponde a los procesos identificados previamente. Así el software dispondrá de un menú principal que se presenta a continuación:

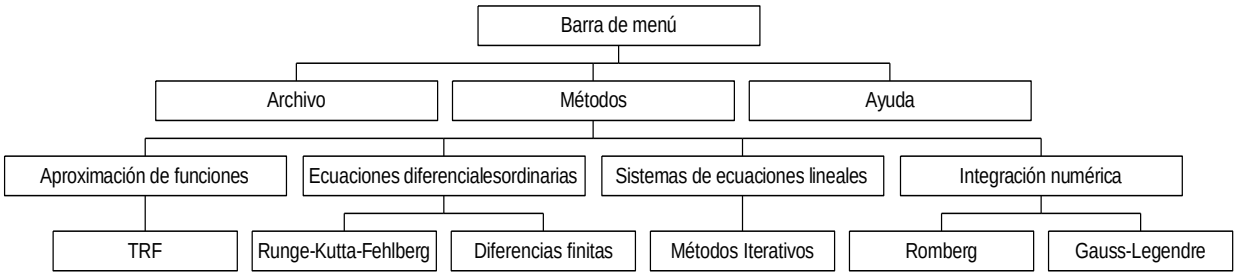


Ilustración 14: Diagrama del menú principal.

El menú anterior contiene las opciones básicas de todo menú en el sistema operativo Windows: Archivo, edición, ver, ventanas, ayuda(algunas opciones no aparecen por motivos de espacio) y el menú especifico llamado métodos, el cual contiene la especificación arriba trazada que nos permite ejecutar los diferentes métodos implementados. La interfaz del software dispone de una barra de herramientas equivalentes a las opciones más comunes en el menú. Este menú esta en armonía con la estructura lógica de la aplicación

3.3. Diseño de ventanas o interfaces de usuario

Atendiendo a las especificaciones de requerimientos que se encontraron en la investigación detallada que se llevo a cabo, se diseñan las interfaces para los distintos métodos que se desarrollarán.

3.3.1. Método de la transformada rápida de Fourier

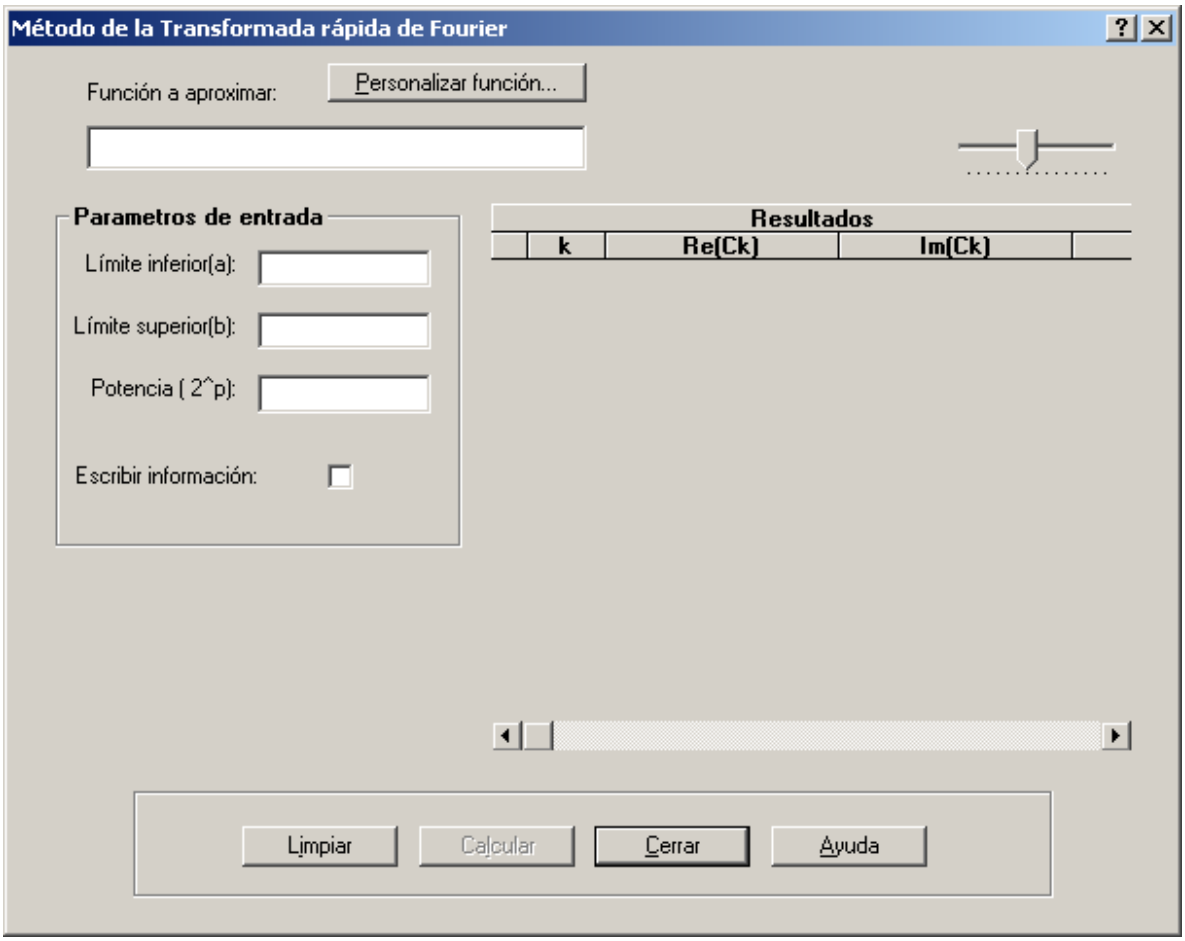


Ilustración 15: Interfaz del método de la transformada de Fourier.

Cuadro de texto Función a aproximar (txtFormula): en este control se digita la función que se aproximará por el método de la transformada de Fourier. En la etapa de implementación se especificarán las reglas sintácticas y las palabras claves para las funciones elementales que podemos introducir.

Grupo de controles Parámetros de entrada

Cuadro de texto Límite inferior(a) (txtLinferior): se especifica el extremo inferior del intervalo del dominio de la función en análisis.

Cuadro de texto Límite superior(b) (txtLsuperior): se especifica el extremo superior del intervalo del dominio de la función en análisis.

Cuadro de texto Potencia(2^p) (txtPotencia): se escribe la potencia de dos que especifica el numéro de nodos(muestras) del intervalo que se tomarán.

Casilla de verificación Escribir información (chkInformacion): si se activa esta casilla se escriben los resultados en el documento que esta activo.

Grupo de botones de orden

Botón Limpiar (cmdLimpiar): Con este boton podemos limpiar los datos sumistrados de la intefaz y los resultados obtenidos.

Botón Calcular (cmdTRF): Ejecutamos el método de la transformada rápida de Fourier. Se activa cuando hayamos suministrado toda la información fundamental.

Botón Cerrar (cmdCancelar): Cerramos la interfaz de usuario.

Botón Ayuda (cmdAyuda): se abre el archivo de ayuda que presenta la información especifica para este método.

Botón Personalizar función (Command1): Abre el cuadro de díalogo en el cual pódemos personalizar una función cualquiera en x. Lea la sección del manual del usuario.

Control deslizante (Slider1): en este control especificamos el número de cifras significativas que se presentarán en los resultados. Oscila entre 1 y 15.

Control cuadricula (DBGridResultados): En este control se presentarán los resultados que arroja el método.

3.3.2. Método de diferencias finitas

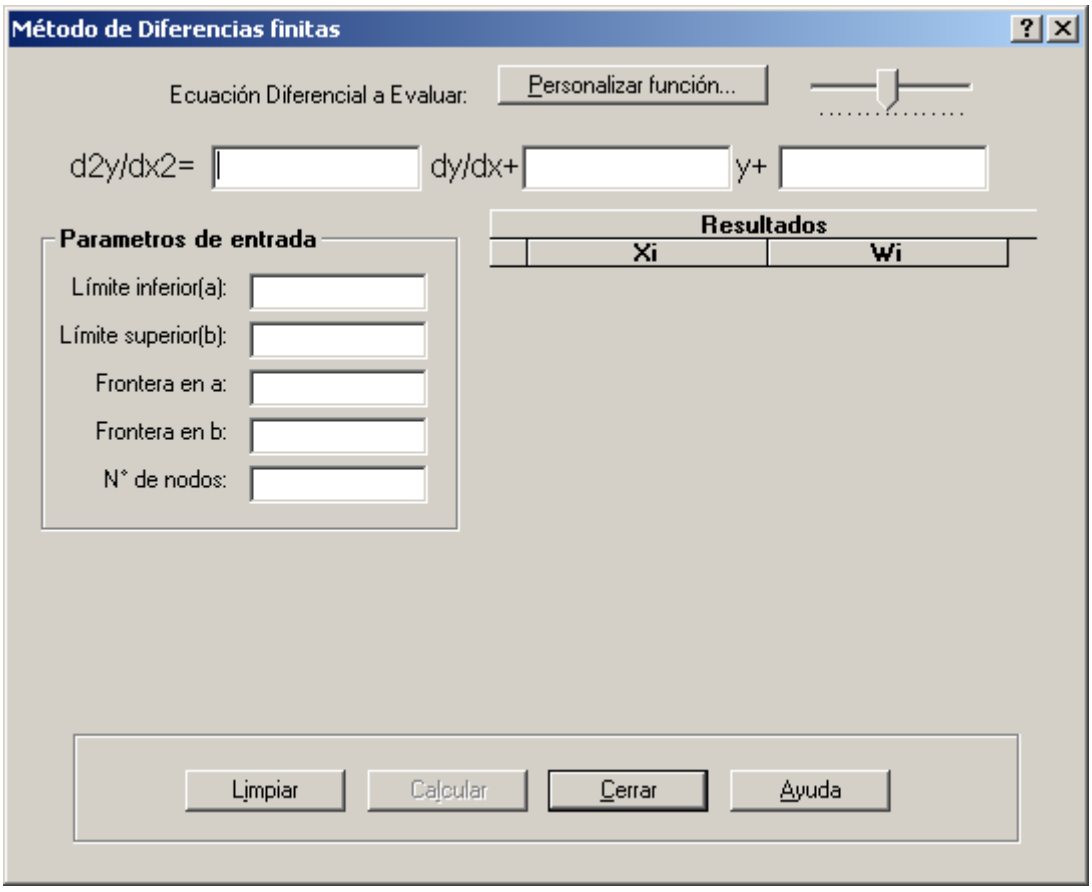


Ilustración 16: Interfaz del método de diferencias finitas.

Cuadros de texto p(x), q(x), r(X) (txtFormula1, txtFormula2, txtFormula3): en estos controles se digitan las funciones que determinan los coeficientes de la ecuación diferencial ordinaria lineal de segundo orden. En la etapa de implementación se especificarán las reglas sintácticas y las palabras claves para las funciones elementales que podemos introducir.

Grupo de controles Parámetros de entrada

Cuadro de texto Límite inferior(a) (txtLinferior): se especifica el extremo inferior del intervalo del dominio de la solución.

Cuadro de texto Límite superior(b) (txtLsuperior): se especifica el extremo superior del intervalo del dominio de la solución.

Cuadro de texto Frontera en a (txtFronteraa): especifica la condición de frontera para el extremo inferior del intervalo.

Cuadro de texto Frontera en b (txtFronterab): especifica la condición de frontera para el extremo superior del intervalo.

Cuadro de texto N° de nodos (txtNodos): se escribe el numéro de nodos(muestras) del intervalo que se tomarán en el método. Este número menos dos es la dimensión del sistema de ecuaciones resultante en el método de diferencias finitas.

Grupo de botones de orden

Botón Limpiar (cmdLimpiar): Con este boton podemos limpiar los datos sumistrados de la intefaz y los resultados obtenidos.

Botón Calcular (cmdDiffinitas): Ejecutamos el método de la Diferencias finitas. Se activa cuando hayamos suministrado toda la información fundamental.

Botón Cerrar (cmdCancelar): Cerramos la interfaz de usuario.

Botón Ayuda (cmdAyuda): se abre el archivo de ayuda que presenta la información especifica para este método.

Botón Personalizar función (Command1): Abre el cuadro de díalogo en el cual pódemos personalizar una función cualquiera en x. Lea la sección del manual del usuario.

Control deslizante (Slider1): en este control especificamos el número de cifras significativas que se presentarán en los resultados. Oscila entre 1 y 15.

Control cuadricula (DBGridResultados): En este control se presentarán los resultados que arroja el método.

3.3.3. Método de Runge-Kutta-Fehlberg

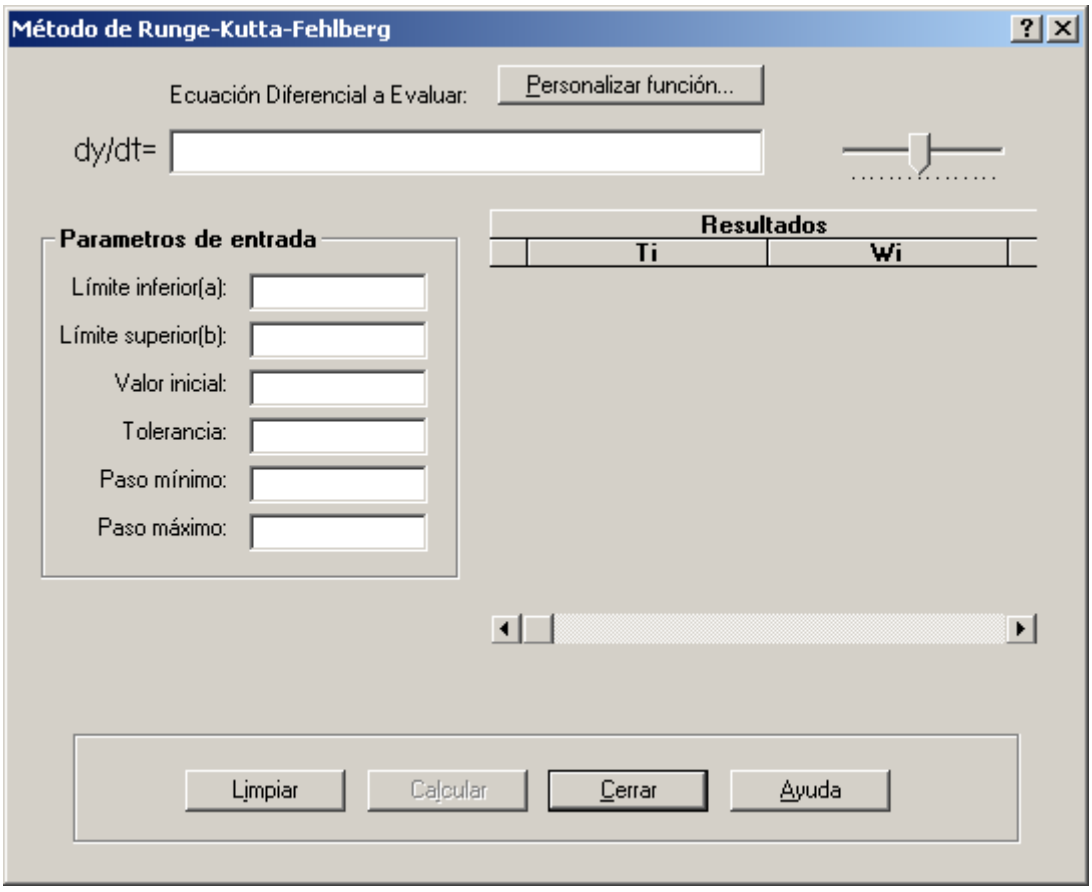


Ilustración 17: Interfaz del método de Runge-Kutta-Fehlberg.

Cuadro de texto Ecuación Diferencial a Evaluar (txtFormula): en este control se digita el mienro derecho de la ecuación general diferencial ordinaria de primer orden. Este miembro es función de las variables *t* y *y*. En la etapa de implementación se especificarán las reglas sintácticas y las palabras claves para las funciones elementales que podemos introducir.

Grupo de controles Parámetros de entrada

Cuadro de texto Límite inferior(a) (txtLinferior): se especifica el extremo inferior del intervalo del dominio de la solución.

Cuadro de texto Límite superior(b) (txtLsuperior): se especifica el extremo superior del intervalo del dominio de la solución.

Cuadro de texto Valor inicial (txtValorInicial): especifica la condición inicial que cumpirá la solución en el extremo inferior del intervalo.

Cuadro de texto tolerancia (txtTolerancia): especifica la tolerancia o el error máximo tolerado para las aproximaciones de los valores de la función..

Cuadro de texto Paso mínimo(txtPasoMinimo): Se escribe el tamaño del paso mínimo permitido.

Cuadro de texto Paso máximo(txtPasoMaximo): Se escribe el tamaño del paso máximo permitido.

Grupo de botones de orden

Botón Limpiar (cmdLimpiar): Con este boton podemos limpiar los datos sumistrados de la intefaz y los resultados obtenidos.

Botón Calcular (cmdRKFehlberg): Ejecutamos el método de Runge-Kutta-Fehlberg. Se activa cuando hayamos suministrado toda la información fundamental.

Botón Cerrar (cmdCancelar): Cerramos la interfaz de usuario.

Botón Ayuda (cmdAyuda): se abre el archivo de ayuda que presenta la información especifica para este método.

Botón Personalizar función (Command1): Abre el cuadro de díalogo en el cual pódemos personalizar una función cualquiera en x. Lea la sección del manual del usuario.

Control deslizante (Slider1): en este control especificamos el número de cifras significativas que se presentarán en los resultados. Oscila entre 1 y 15.

Control cuadricula (DBGridResultados): En este control se presentarán los resultados que arroja el método.

3.3.4. Métodos iterativos para la solución de sistemas de ecuaciones lineales

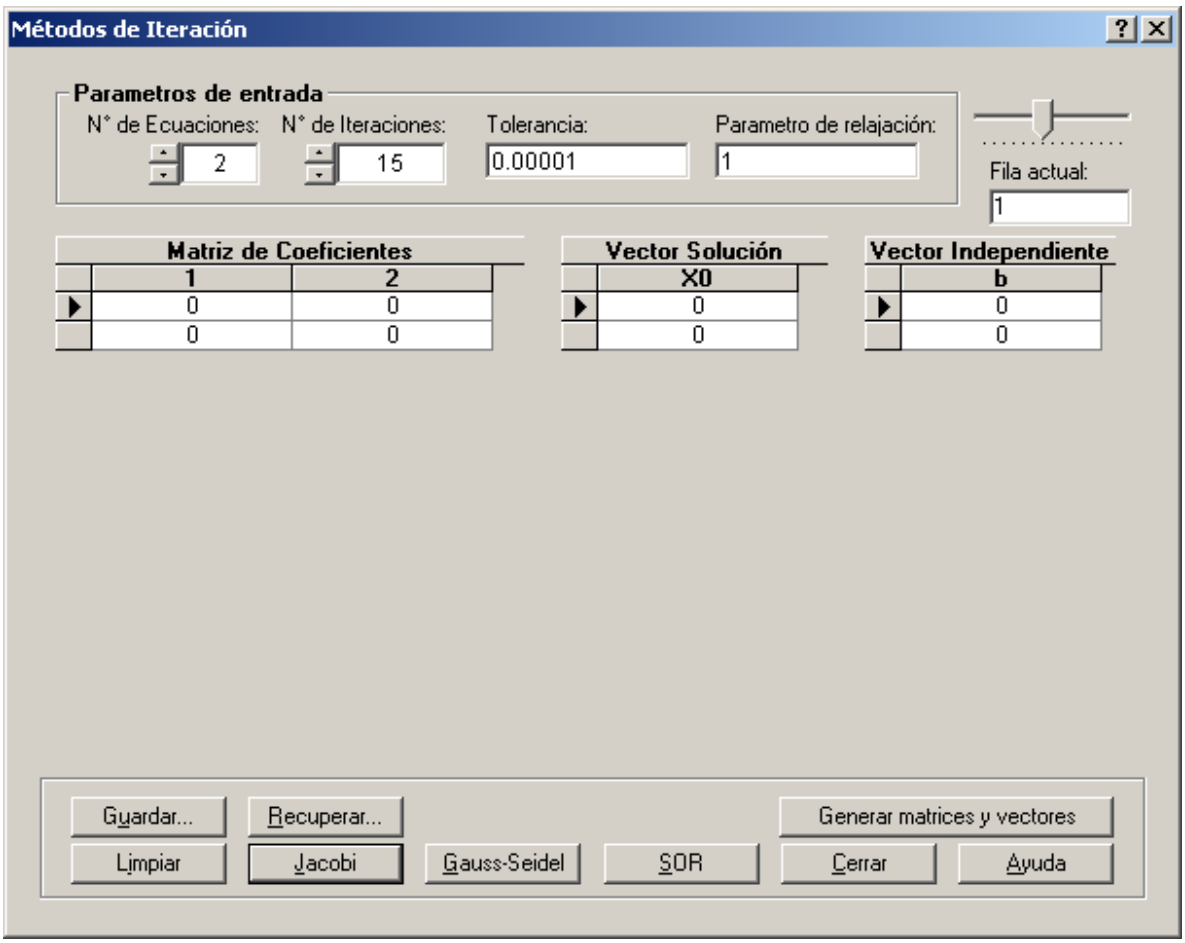


Ilustración 18: Interfaz de los métodos iterativos.

Grupo de controles Parámetros de entrada

Cuadro de texto N° de Ecuaciones (txtNoEcuaciones): Se escribe el número de ecuaciones del sistema y por consiguiente el número de incognitas. Esta asociado con un control Arriba-Abajo, que aumenta y disminuye en 1 el número de ecuaciones. Su valor por defecto es 2.

Cuadro de texto N° de Iteraciones (txtNoIteraciones): Se escribe el número de Iteraciones tolerable para que el método aproxime la solución. Esta asociado con un control

Arriba-Abajo, que aumenta y disminuye en 1 el número de iteraciones. Su valor por defecto es 15.

Cuadro de texto tolerancia (txtTolerancia): especifica la tolerancia o el error máximo tolerado para los componentes individuales del vector solución. Su valor por defecto es 0.00001.

Cuadro de texto Parámetro de relajación(txtParametro): Especifica eñ parámetro de relajación que oscila entre 0 y 2 sin incluir los extremos que se utiliza especialmente en el método de SOR. Su valor por defecto es 1 que equivale al método de Gauss-Seidel.

Grupo de botones de orden

Botón Limpiar (cmdLimpiar): Con este boton podemos limpiar los datos sumistrados de la intefaz y los resultados obtenidos.

Botón Jacobi (cmdJacobi): Ejecutamos el método de Jacobi.

Botón Gauss-Seidel (cmdGaussSeidel): Ejecutamos el método de Gauss-Seidel.

Botón SOR (cmdSOR): Ejecutamos el método de SOR.

Botón Cerrar (cmdCancelar): Cerramos la interfaz de usuario.

Botón Ayuda (cmdAyuda): se abre el archivo de ayuda que presenta la información especifica para este método.

Botón Guardar (cmdGuardar): Se guardan los datos que especifican la matriz de coeficientes, el vector de aproximación inicial y el vector independiente.

Botón Recuperar (cmdRecuperar): Se recuperan los datos que especifican la matriz de coeficientes, el vector de aproximación inicial y el vector independiente previamente guardados con el boton cmdGuardar.

Botón Generar matrices y vectores (cmdGenerar): Actualiza los controles cuadricula para que sean consistentes con el valor especificado en el cuadro de texto txtNoEcuaciones.

Control deslizante (Slider1): en este control especificamos el número de cifras significativas que se presentarán en los resultados. Oscila entre 1 y 15.

Control cuadricula Matriz de Coeficientes (DBGridMatriz): En este control se capturan los coeficientes que acompañan a las incógnitas del sistema.

Control cuadricula Vector Solución (DBGridVectorSol): En este control se captura el vector de aproximación inicial y se presentan los vectores de aproximación sucesivos generados por el método.

Control cuadricula Vector Independiente (DBGridVectorInd): En este control se captura el vector independiente que aparece en el miembro derecho de la ecuación matricial.

3.3.5. Cuadratura de Gauss-Legendre

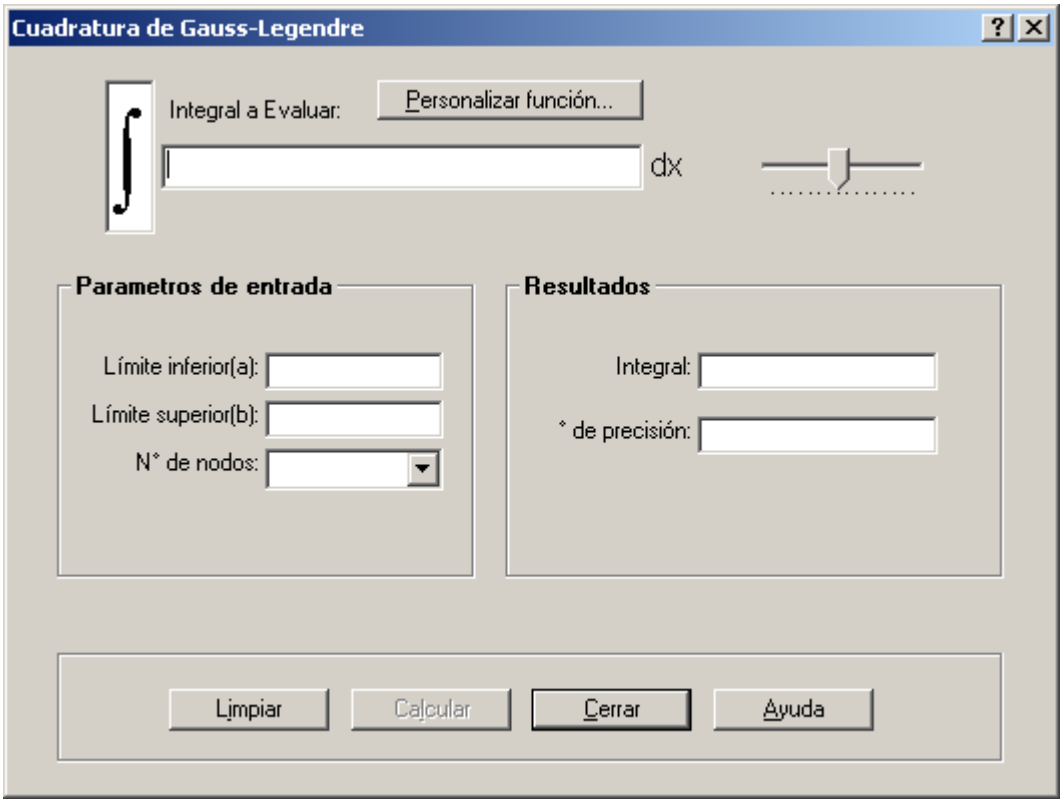


Ilustración 19: Interfaz del método de la cuadratura de Gauss-Legendre.

Cuadro de texto Integral a Evaluar (txtFormula): en este control se digita la función que se integrará por el método de Gauss-Legendre. En la etapa de implementación se especificarán las reglas sintácticas y las palabras claves para las funciones elementales que podemos introducir.

Grupo de controles Parámetros de entrada

Cuadro de texto Límite inferior(a) (txtLinferior): se especifica el extremo inferior del intervalo del dominio de la función en análisis.

Cuadro de texto Límite superior(b) (txtLsuperior): se especifica el extremo superior del intervalo del dominio de la función en análisis.

Cuadro combinado N° de Nodos (cmbNodos): se escribe el numéro de nodos(muestras) del intervalo que se tomarán. Los valores validos son de 1 a 20, 32 y 64.

Grupo de botones de orden

Botón Limpiar (cmdLimpiar): Con este boton podemos limpiar los datos sumistrados de la intefaz y los resultados obtenidos.

Botón Calcular (cmdGauss): Ejecutamos el método de Gauss-Legendre. Se activa cuando hayamos suministrado toda la información fundamental.

Botón Cerrar (cmdCancelar): Cerramos la interfaz de usuario.

Botón Ayuda (cmdAyuda): se abre el archivo de ayuda que presenta la información especifica para este método.

Botón Personalizar función (Command1): Abre el cuadro de díalogo en el cual pódemos personalizar una función cualquiera en x. Lea la sección del manual del usuario.

Control deslizante (Slider1): en este control especificamos el número de cifras significativas que se presentarán en los resultados. Oscila entre 1 y 15.

Grupo de controles Resultados

Cuadro de texto Integral (txtIntegral): aparece el resultado de la evaluación de la integral especificada.

Cuadro de texto ° de precisión (txtPrecision): Presenta el grado de precisión que se alcanza utilizando esta método con el numéro de nodos especificados en el cuadro combinado cmbNodos.

3.3.6. Integración de Romberg

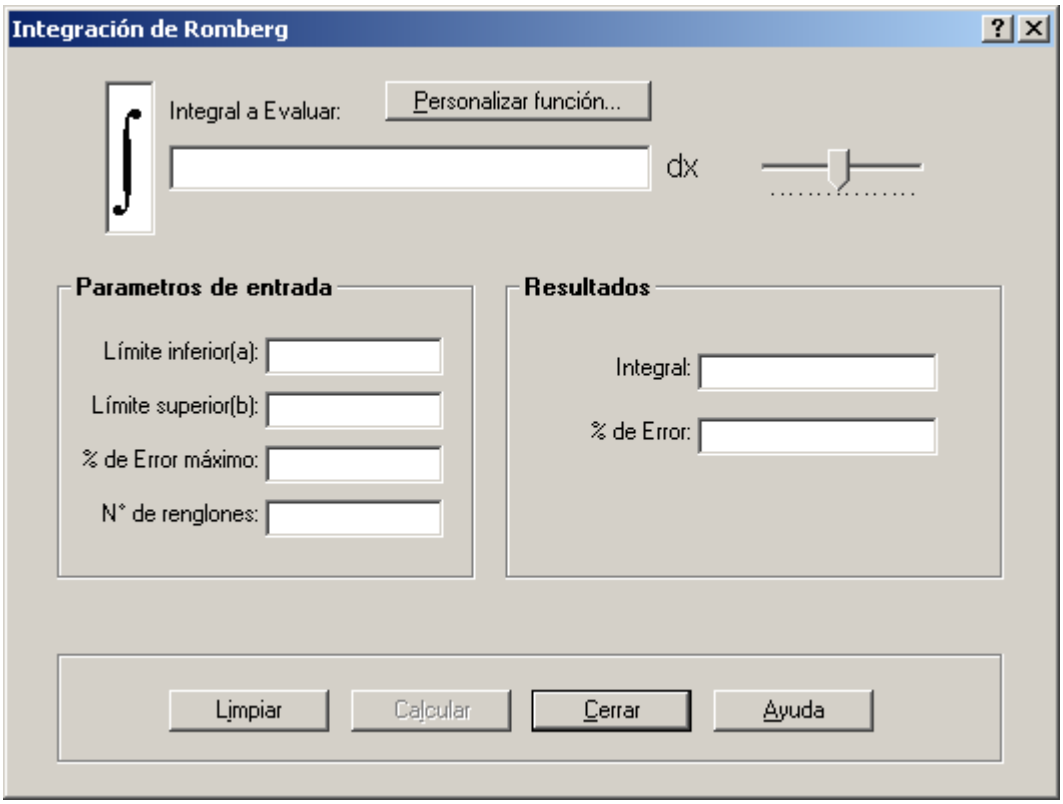


Ilustración 20: Interfaz del método de Romberg.

Cuadro de texto Integral a Evaluar (txtFormula): en este control se digita la función que se integrará por el método de Romberg. En la etapa de implementación se especificarán las reglas sintácticas y las palabras claves para las funciones elementales que podemos introducir.

Grupo de controles Parámetros de entrada

Cuadro de texto Límite inferior(a) (txtLinferior): se especifica el extremo inferior del intervalo del dominio de la función en análisis.

Cuadro de texto Límite superior(b) (txtLsuperior): se especifica el extremo superior del intervalo del dominio de la función en análisis.

Cuadro combinado % de Error Máximo (cmbNodos): se escribe el porcentaje de error máximo tolerado para la aprimación.

Cuadro de texto N° de renglones (txtRenglones): se escribe el número de renglones máximo a generar .

Grupo de botones de orden

Botón Limpiar (cmdLimpiar): Con este boton podemos limpiar los datos sumistrados de la intefaz y los resultados obtenidos.

Botón Calcular (cmdRomberg): Ejecutamos el método de Romberg. Se activa cuando hayamos suministrado toda la información fundamental.

Botón Cerrar (cmdCancelar): Cerramos la interfaz de usuario.

Botón Ayuda (cmdAyuda): se abre el archivo de ayuda que presenta la información especifica para este método.

Botón Personalizar función (Command1): Abre el cuadro de díalogo en el cual pódemos personalizar una función cualquiera en x. Lea la sección del manual del usuario.

Control deslizante (Slider1): en este control especificamos el número de cifras significativas que se presentarán en los resultados. Oscila entre 1 y 15.

Grupo de controles Resultados

Cuadro de texto Integral (txtIntegral): aparece el resultado de la evaluación de la integral especificada.

Cuadro de texto % de Error (txtPorError): Presenta el porcentaje de error que se alcanzó con la aproximación que se despliega en el cuadro de texto Integral.

IV. Desarrollo del software

4.1. Introducción

Ahora se presentará el código que implementa las especificaciones de diseño. Para cada método se escribió un modulo correspondiente en Visual Basic. Veremos como se programa o construye un proyecto en Visual Basic tomando como ejemplo el modulo del método de diferencias finitas. Se presentará la codificación de un solo modulo, puesto que el proceso que se lleva a cabo se repite para los demás módulos. De esta manera se decidió poner todo el código en un CD junto al proyecto completo el cual se podrá adquirir gratuitamente para poder apreciar completamente el proceso de codificación. Vea el anexo para más información.

Para crear una aplicación en Visual Basic se siguen tres etapas. La interfaz de usuario se facilita gracias a las herramientas de desarrollo rápido que proporciona, utilizando para ello el ratón, "dibujando" controles, como cuadros de texto y botones de comando, en un formulario. A continuación, se establecen las propiedades del formulario y los controles para especificar valores como el título, el color y el tamaño. Finalmente, se escribe el código para dar vida a la aplicación. Estos pasos básicos que daremos para el modulo de diferencias finitas mostrarán los principios que se usaron en los demás módulos que se desarrollaron.

4.2. Elementos básicos de Visual Basic

4.2.1. Algunos conceptos de Visual Basic

Es necesario que se entienda el proceso de desarrollo de una aplicación. Para ese fin es útil comprender algunos de los conceptos clave alrededor de los cuales está construido Visual Basic. Visual Basic es un lenguaje de desarrollo para Windows. De aquí que es necesario familiarizarse con el entorno Windows, y conocer algunas diferencias fundamentales entre la programación para Windows frente a otros entornos, si no se tiene experiencia en la programación para Windows

4.2.1.1. ¿Cómo funciona Windows?

No especificaremos detalles técnicos acerca de cómo funciona Windows No es necesario para comprender su funcionamiento general. El funcionamiento de Windows incluye tres conceptos claves: ventanas, eventos y mensajes.

Una ventana es simplemente una región rectangular con sus propios límites. Existen varios tipos de ventanas: una ventana Explorador en Windows, una ventana de documento dentro de su programa de proceso de textos o un cuadro de diálogo que emerge para recordarle una cita, un botón de comando es una ventana. Los iconos, cuadros de texto, botones de opción y barras de menús son todos ventanas.

El sistema operativo controla continuamente cada una de estas ventanas para ver si existen signos de actividad o eventos. Los eventos pueden producirse por tres fuentes: mediante acciones del usuario, como hacer clic con el ratón o presionar una tecla, mediante programación o como resultado de acciones de otras ventanas.

Cada vez que se produce un evento se envía un mensaje al sistema operativo. El sistema procesa el mensaje y lo transmite a las demás ventanas. Entonces, cada ventana puede realizar la acción apropiada, basándose en sus propias instrucciones para tratar ese mensaje en particular (por ejemplo, volverse a dibujar cuando otra ventana la ha dejado descubierta).

Muchos de los mensajes los controla automáticamente Visual Basic, mientras que otros se tratan como procedimientos de evento para nuestra comodidad. Esto le permite crear rápidamente eficaces aplicaciones sin tener que tratar detalles innecesarios.

4.2.1.2. La programación conducida por eventos

En las programación tradicional o "por procedimientos", la aplicación es la que controla qué partes de código y en qué secuencia se ejecutan. En estas aplicaciones la ejecución comienza con la primera línea de código y continúa con una ruta predefinida a través de la aplicación, llamando a los procedimientos según se necesiten.

En la programación conducida por eventos, el código no sigue una ruta predeterminada; ejecuta distintas secciones de código como respuesta a los eventos. Estos eventos se activan mediante acciones del usuario, por mensajes del sistema o de otras aplicaciones, o incluso por la propia aplicación. La secuencia de estos eventos determina la secuencia en la que se ejecuta

el código, por lo que la ruta a través del código de la aplicación es diferente cada vez que se ejecuta el programa o aplicación.

No se puede predecir la secuencia de los eventos, por lo que el código debe establecer ciertos supuestos acerca del "estado del mundo real" cuando se ejecute.

4.2.2. Elementos del entorno integrado de desarrollo (IDE)

El entorno de trabajo de Visual Basic integra funciones diferentes como el diseño, modificación, compilación y depuración en un entorno común llamado Entorno de desarrollo integrado (IDE). Entornos similares se encuentran en otros lenguajes de programación, incluso, aquellos conducidos por procedimientos. En las herramientas de desarrollo más tradicionales, cada una de esas funciones funcionaría como un programa diferente, cada una con su propia interfaz.

¿Cómo iniciar el IDE de Visual Basic?

Para iniciar Visual Basic desde Windows

Haga clic en **Inicio** en la barra de tareas.

Seleccione **Programas** y luego el ícono de **Microsoft Visual Basic**.

–o bien–

Haga clic en **Inicio** en la barra de tareas.

Seleccione **Programas**.

El sistema mostrará la siguiente ventana del IDE:

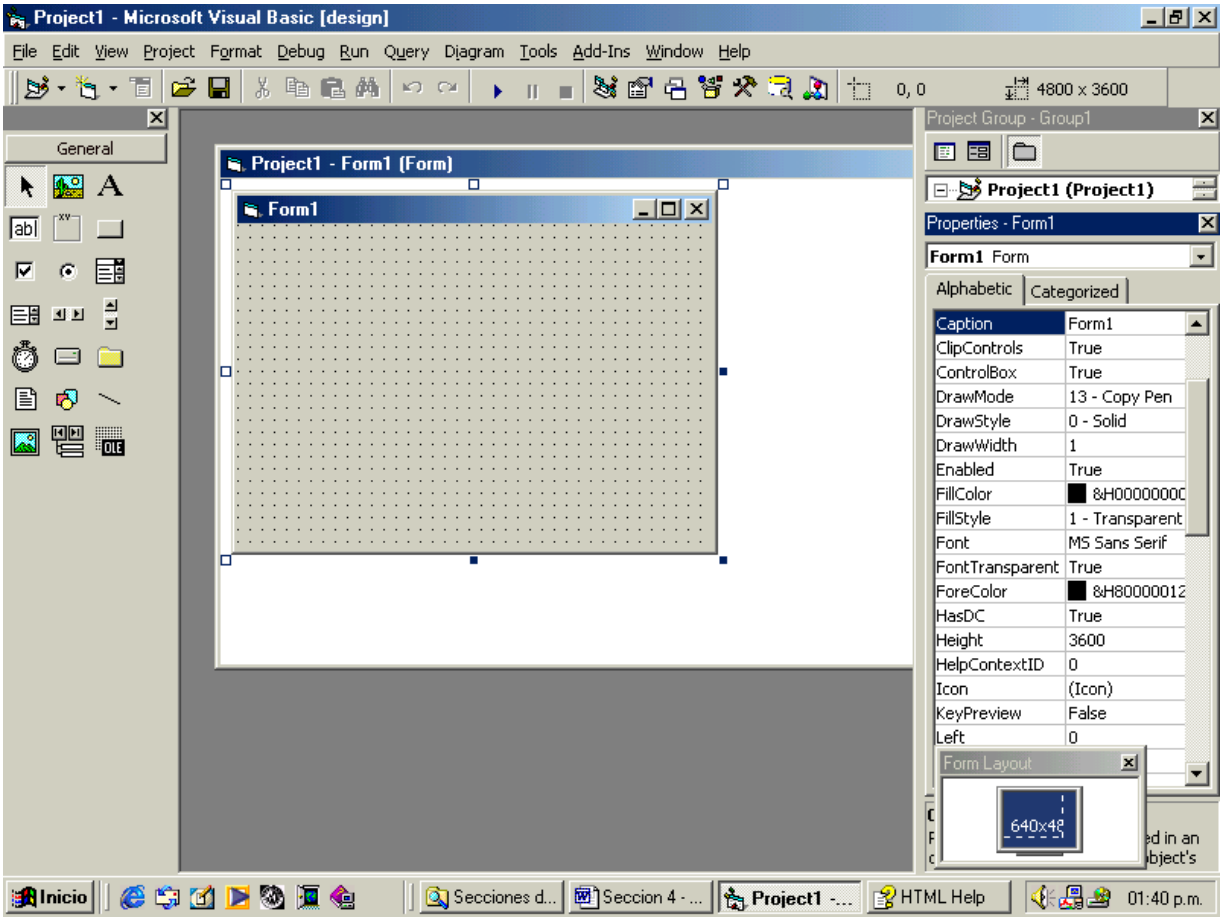


Ilustración 21: El entorno integrado de desarrollo de Visual Basic.

Elementos del entorno integrado de desarrollo

El entorno integrado de desarrollo de Visual Basic (IDE) consta de los siguientes elementos.

Barra de menús

Presenta los comandos que se usan para trabajar con Visual Basic. Además de los menús estándar **Archivo**, **Edición**, **Ver**, **Ventana** y **Ayuda**, se proporcionan otros menús para tener acceso a funciones específicas de programación como **Proyecto**, **Formato** o **Depuración**.

Menús contextuales

Contienen accesos directos a acciones que se realizan con frecuencia. Para abrir un menú contextual, haga clic con el botón secundario del *mouse* en el objeto que está usando. La lista específica de opciones disponibles en el menú contextual depende de la parte del entorno en la

que se hace clic con el botón secundario del *mouse*. Por ejemplo, el menú contextual que aparece cuando hace clic con el botón secundario del *mouse* en el cuadro de herramientas le permite mostrar el cuadro de diálogo **Componentes**, ocultar el cuadro de herramientas, acoplar o desacoplar el cuadro de herramientas, o agregar una ficha personalizada al cuadro de herramientas.

Barras de herramientas

Proporcionan un rápido acceso a los comandos usados normalmente en el entorno de programación. Haga clic en un botón de la barra de herramientas para llevar a cabo la acción que representa ese botón. De forma predeterminada, al iniciar Visual Basic se presenta la barra de herramientas Estándar. Es posible activar o desactivar otras barras de herramientas adicionales para modificar, diseñar formularios desde el comando **Barras de herramientas** del menú **Ver**.

Las barras de herramientas se pueden acoplar debajo de la barra de menús o pueden "flotar" si selecciona la barra vertical del borde izquierdo y la arrastra fuera de la barra de menús.

Cuadro de herramientas

Proporciona un conjunto de herramientas que puede usar durante el diseño para colocar controles en un formulario. Además del diseño del cuadro de herramientas predeterminado, puede crear su propio diseño personalizado si selecciona **Agregar ficha** en el menú contextual y agrega controles a la ficha resultante.

Ventana Explorador de proyectos

Enumera los formularios y módulos del proyecto actual. Un *proyecto* es la colección de archivos que usa para generar una aplicación.

Ventana Propiedades

Enumera los valores de las propiedades del control o formulario seleccionado. Una *propiedad* es una característica de un objeto, como su tamaño, título o color.

Examinador de objetos

Enumera los objetos disponibles que puede usar en su proyecto y le proporciona una manera rápida de desplazarse a través del código. Puede usar el Examinador de objetos para explorar

objetos en Visual Basic y otras aplicaciones, ver qué métodos y propiedades están disponibles para esos objetos, y pegar código de procedimientos en su aplicación.

Diseñador de formularios

Funciona como una ventana en la que se personaliza el diseño de la interfaz de su aplicación. Agregue controles, gráficos e imágenes a un formulario para crear la apariencia que desee. Cada formulario de la aplicación tiene su propia ventana diseñador de formulario.

Ventana Editor de código

Funciona como un editor para escribir el código de la aplicación. Se crea una ventana editor de código diferente para cada formulario o módulo del código de la aplicación.

Ventana Posición del formulario

La ventana Posición del formulario (figura 2.2) le permite colocar los formularios de su aplicación utilizando una pequeña representación gráfica de la pantalla.

Ventanas Inmediato, Locales e Inspección

Estas ventanas adicionales se proporcionan para la depuración de la aplicación. Sólo están disponibles cuando ejecuta la aplicación dentro del IDE.

4.3. Método de diferencias finitas para ecuaciones diferenciales ordinarias lineales de segundo orden con condiciones en las fronteras.

Como lo mencionamos en la introducción hay tres pasos principales para crear una aplicación en Visual Basic:

1. Crear la interfaz.
2. Establecer propiedades.
3. Escribir el código.

Veamos como funciona esto, desarrollando el método de diferencias finitas, a continuación

4.3.1. Crear la interfaz

Para la interfaz completa de la aplicación se usará un asistente para aplicaciones que a partir de unos cuantos datos acerca de la aplicación, que se le suministren nos construye un esquema o estructura de la aplicación, tal como debería lucir en Windows. Esta herramienta ahorra tiempo de desarrollo de software y nos permite ser más productivos al enfocarnos en las especificaciones de nuestra aplicación; además, de construir nuestras interfaces propias para la clase de aplicación que estamos desarrollando.

Con los formularios podemos desarrollar cualquier interfaz para diversos fines. Utilizamos el cuadro de herramientas para agregar controles funcionales a la interfaz. La interfaz para el método de diferencias finitas es la siguiente:

Para dibujar un control mediante el cuadro de herramientas

1. Haga clic en la herramienta del control que ha elegido dibujar
2. Mueva el puntero dentro del formulario. El puntero adoptará la forma de cruz, como se muestra en la figura 2.3.
3. Coloque la cruz donde quiera que aparezca la esquina superior izquierda del control.
4. Arrastre la cruz hasta que el control tenga el tamaño deseado. (*Arrastrar* significa mantener presionado el botón primario del *mouse* mientras mueve un objeto con el *mouse*.)

5. Suelte el botón del *mouse*.

Con esta técnica de arrastrar y soltar colocamos todos los controles necesarios al formulario. De esta manera se construyó la siguiente interfaz:

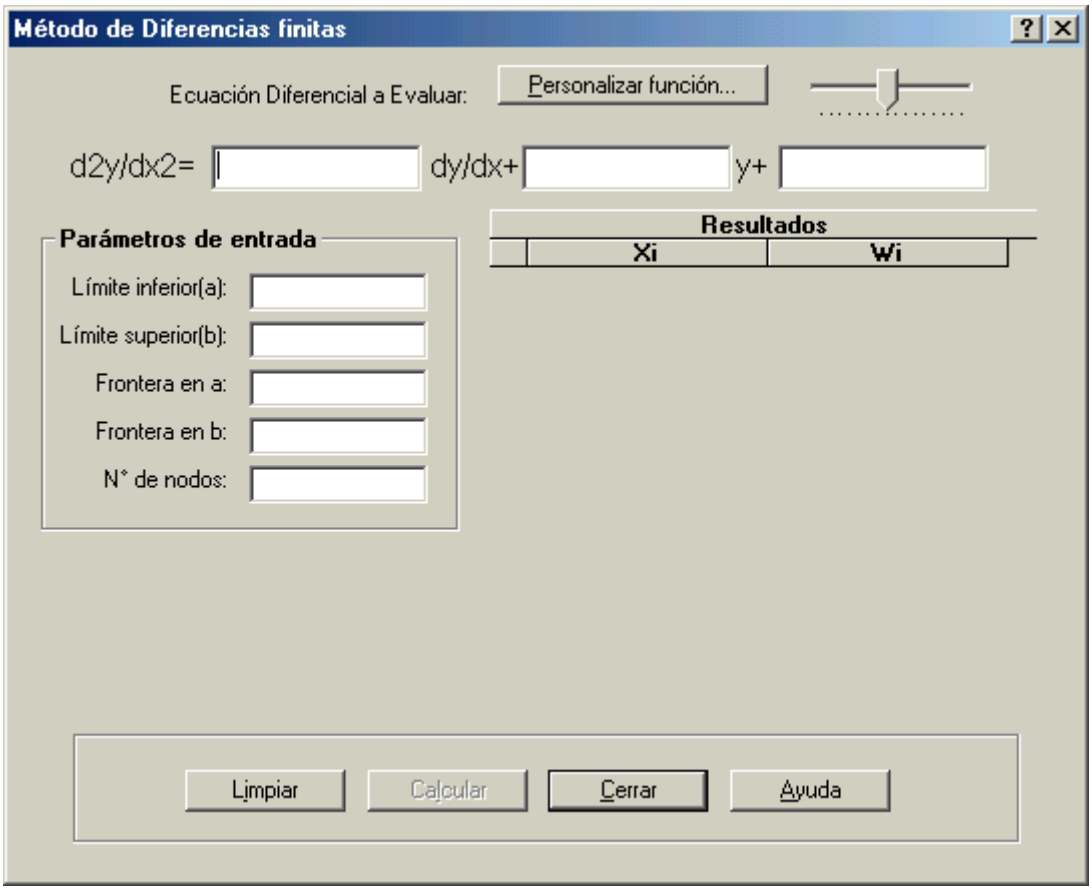


Ilustración 22: Interfaz del método de diferencias finitas 2.

4.3.2. Establecer propiedades

El siguiente paso consiste en establecer las propiedades que nos interesan de los objetos o controles que se crean. La ventana Propiedades proporciona una manera fácil de establecer las propiedades de todos los objetos de un formulario. Para abrir la ventana Propiedades, elija el comando **Ventana Propiedades** del menú **Ver**, haga clic en el botón **Ventana Propiedades** de la barra de herramientas o utilice el menú contextual del control.

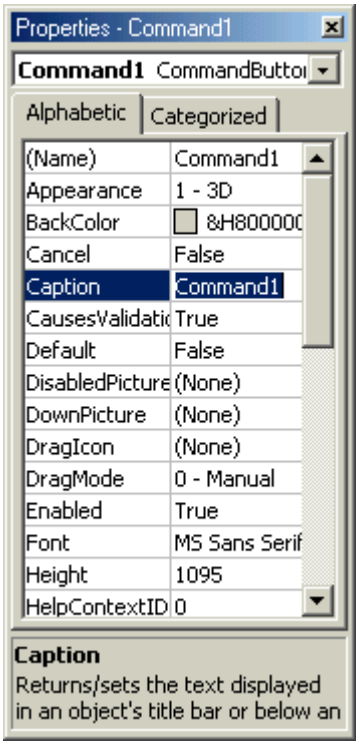


Ilustración 23: La ventana propiedades.

La ventana Propiedades consta de los siguientes elementos:

- Cuadro del objeto: presenta el nombre del objeto para el que puede establecer propiedades. Haga clic en la flecha situada a la derecha del cuadro Objeto para presentar la lista de objetos del formulario actual.

- Fichas de orden: elija entre una lista alfabética de propiedades o una vista jerárquica dividida en categorías lógicas, como las que tratan de la apariencia, fuentes o posición.

- Lista de propiedades: la columna de la izquierda presenta todas las propiedades del objeto seleccionado. Puede modificar y ver los valores en la columna de la derecha.

Para establecer propiedades desde la ventana Propiedades

1. En el menú **Ver**, elija **Propiedades** o haga clic en el botón **Propiedades** de la barra de herramientas.

   La ventana Propiedades presenta los valores del formulario o control seleccionado.

2. En la lista **Propiedades**, seleccione el nombre de una propiedad.

3. En la columna de la derecha, escriba o seleccione el nuevo valor de la propiedad.

En el modulo de diferencias finitas tenemos, por ejemplo, las siguientes propiedades que se establecieron para algunos objetos:

Tabla 3: Propiedades de los controles de la interfaz del método de diferencias finitas.

Objeto	**Propiedad**	**Valor**
Form	Caption	Método de diferencias finitas
Form	Name	FrmDifFinitas
CommandButton	Caption	Calcular
CommandButton	Name	CmdDifFinitas
DBGrid	Name	DBDridResultados
DBGrid	Caption	Resultados
TextBox	Name	TxtFormula1
TextBox	Name	TxtFormula2
TextBox	Name	TxtFormula3
TextBox	Name	TxtLinferior

TextBox	Name	TxtLsuperior
TextBox	Name	TxtFronteraa
TextBox	Name	TxtFronterab
CommandButton	Name	CmdLimpiar
CommandButton	Caption	Limpiar
Slider	Name	Slirder1

4.3.3. Escritura de código

La escritura de código se realiza en la *ventana Editor de código* de Visual Basic (Ilustración 21) El código consta de instrucciones del lenguaje, constantes y declaraciones. Mediante la ventana Editor de código puede ver y modificar rápidamente el código de su aplicación.

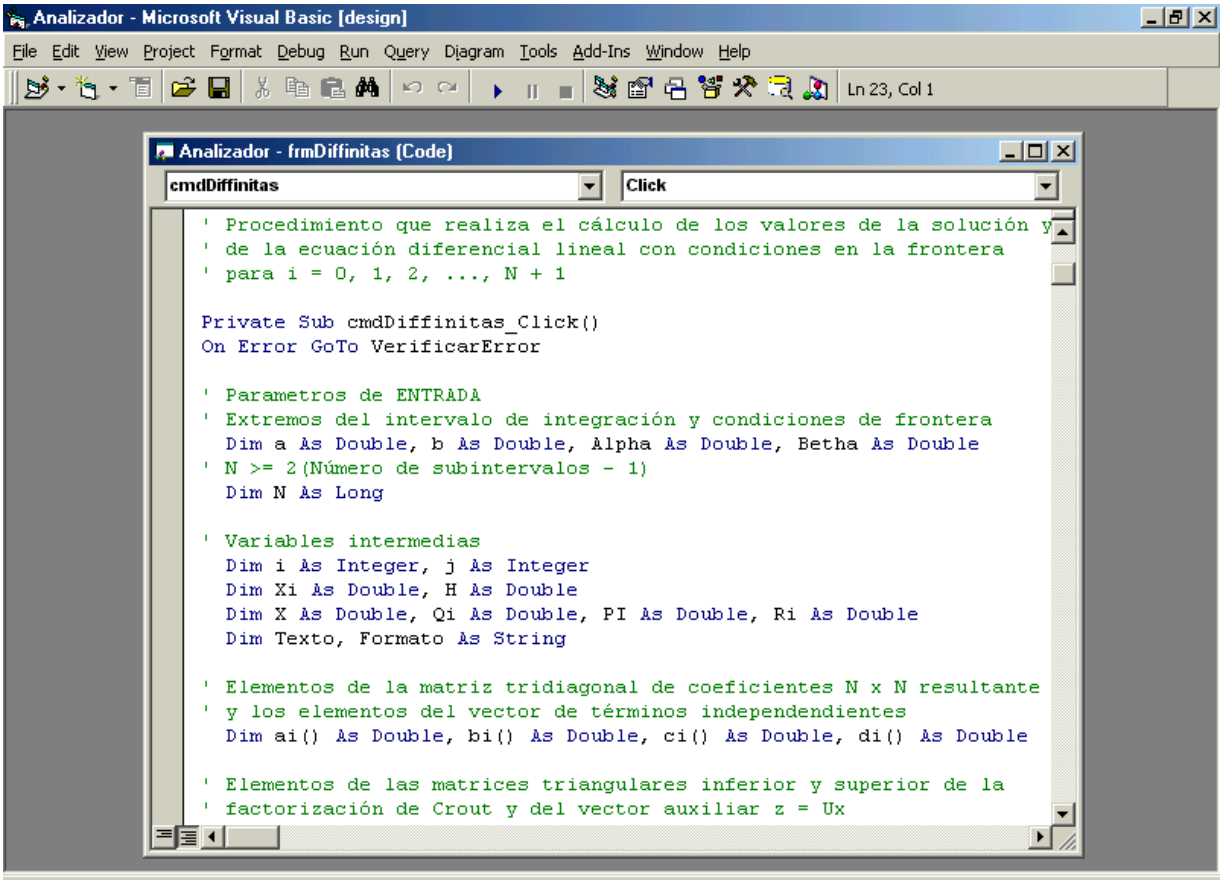


Ilustración 24: Ventana de editor de código.

Para abrir la ventana Código

- Haga doble clic en el formulario o el control para el que desea escribir código.

–o bien–

- En la ventana Explorador de proyectos, seleccione el nombre de un formulario o un módulo y elija **Ver código**.

La ventana Código incluye los siguientes elementos:

- Cuadro de lista **Objeto**: presenta el nombre del objeto seleccionado. Haga clic en la flecha de la derecha del cuadro de lista para presentar una lista de todos los objetos asociados con el formulario.

- Cuadro de lista **Procedimiento**: enumera los procedimientos o eventos de un objeto. El cuadro presenta el nombre del procedimiento seleccionado, en este caso Click. Elija la flecha que hay a la derecha del cuadro para presentar todos los procedimientos del objeto.

Crear procedimientos de evento

El código de Visual Basic se divide en módulos y estos a su vez se dividen en procedimientos. Los procedimientos se activan de acuerdo a eventos y es por eso que se llaman *procedimientos de evento.* Crearemos el procedimiento de evento para el botón cmdLimpiar.

Para crear un procedimiento de evento

1. En el cuadro de lista **Objeto**, seleccione el nombre de un objeto del formulario activo. (El formulario *activo* es el formulario que actualmente tiene el enfoque.)

   Para este ejemplo, elija el botón de comando, cmdLimpiar.

2. En el cuadro de lista **Procedimiento**, seleccione el nombre de un evento del objeto seleccionado.

   Aquí, el procedimiento **Click** ya está seleccionado puesto que es el procedimiento predeterminado para un botón de comando. Se presenta en la ventana Código una *plantilla* para el procedimiento de evento.

3. Escriba el código entre las instrucciones **Sub** y **End Sub**:

   El procedimiento de evento debería parecerse a éste:

```
Private Sub cmdLimpiar_Click()
  If (MsgBox("¿Realmente desea limpiar los datos de todos
  los controles.", vbYesNo, "Confirmación") = vbYes) Then
      txtFormula1.Text = ""
      txtFormula2.Text = ""
      txtFormula3.Text = ""
      txtLsuperior.Text = ""
      txtLinferior.Text = ""
      txtFronteraa.Text = ""
      txtFronterab.Text = ""
      txtNodos.Text = ""
      Erase UserData
      mTotalRows = 0
      DBGridResultados.Refresh
      cmdDiffinitas.Enabled = False
      cmdCancelar.Default = True
  End If
End Sub
```

De esta manera se escriben todos los procedimientos para los demás controles. El código completo se presenta a continuación.

```
Option Explicit
Option Base 1

' Se debe declarar tres objetos de la clase clsEvalua para
introducir
' las funciones p(x), q(x), r(x) de la ecuación (Con el nombre
que se desée),
' para poder utilizarlo(s) en el programa.

Dim objEvalua1 As clsEvalua
Dim objEvalua2 As clsEvalua
```

```
Dim objEvalua3 As clsEvalua

' Matriz que contiene los datos que se presentan en el control GRID
' y el total de filas que hay
Dim UserData() As Double
Dim mTotalRows As Long
Public bytFormula As Byte



' Procedimiento que realiza el cálculo de los valores de la solución y(xi),
' de la ecuación diferencial lineal con condiciones en la frontera
' para i = 0, 1, 2, ..., N + 1

Private Sub cmdDiffinitas_Click()
On Error GoTo VerificarError

' Parametros de ENTRADA
' Extremos del intervalo de integración y condiciones de frontera
  Dim a As Double, b As Double, Alpha As Double, Betha As Double
' N >= 2(Número de subintervalos - 1)
  Dim N As Long

' Variables intermedias
  Dim i As Integer, j As Integer
  Dim Xi As Double, H As Double
  Dim X As Double, Qi As Double, PI As Double, Ri As Double
  Dim Texto, Formato As String

' Elementos de la matriz tridiagonal de coeficientes N x N resultante
```

```
' y los elementos del vector de términos independendientes
  Dim ai() As Double, bi() As Double, ci() As Double, di() As
Double

' Elementos de las matrices triangulares inferior y superior de
la
' factorización de Crout y del vector auxiliar z = Ux
  Dim li() As Double, ui() As Double, Zi() As Double

' Variables de SALIDA
' Valores aproximados de la solución y(xi)
  Dim Wi() As Double

' Captura de parametros y funciones
  a = txtLinferior
  b = txtLsuperior
  Alpha = txtFronteraa
  Betha = txtFronterab
  N = txtNodos - 2

' Verificar sintaxis de las funciones introducidas
  objEvalua1.Expresion = Trim(txtFormula1.Text)
  If objEvalua1.CErr > 0 Then
      MsgBox   objEvalua1.DErr,   vbCritical,   "Error   "   &
objEvalua1.CErr
      Exit Sub
  Else
      txtFormula1.Text = objEvalua1.Expresion
  End If

  objEvalua2.Expresion = Trim(txtFormula2.Text)
  If objEvalua2.CErr > 0 Then
      MsgBox   objEvalua2.DErr,   vbCritical,   "Error   "   &
objEvalua2.CErr
```

```
      Exit Sub
  Else
      txtFormula2.Text = objEvalua2.Expresion
  End If

  objEvalua3.Expresion = Trim(txtFormula3.Text)
  If objEvalua3.CErr > 0 Then
      MsgBox objEvalua3.DErr, vbCritical, "Error " & objEvalua3.CErr
      Exit Sub
  Else
      txtFormula3.Text = objEvalua3.Expresion
  End If

' Validar las funciones en términos de X para que se pueda continuar
  If (objEvalua1.ValidarVar("X") = True Or objEvalua1.ValidarVar() = True) _
     And (objEvalua2.ValidarVar("X") = True Or objEvalua2.ValidarVar() = True) _
     And (objEvalua3.ValidarVar("X") = True Or objEvalua3.ValidarVar() = True) Then
      ReDim ai(N): ReDim bi(N)
      ReDim ci(N): ReDim di(N)
      H = (b - a) / (N + 1)

    ' Obtener los términos de la matriz tridiagonal y del vector independiente
      X = a + H
      Qi = objEvalua2.EvaluarExpr(X)
      If objEvalua2.CErr > 0 Then
          MsgBox objEvalua2.DErr, vbCritical, "Error " & objEvalua2.CErr
          Exit Sub
      End If
```

```
      ai(1) = 2 + (H ^ 2) * Qi

      PI = objEvalua1.EvaluarExpr(X)
      If objEvalua1.CErr > 0 Then
          MsgBox objEvalua1.DErr, vbCritical, "Error " &
objEvalua1.CErr
          Exit Sub
      End If
      bi(1) = -1 + (H / 2) * PI

      Ri = objEvalua3.EvaluarExpr(X)
      If objEvalua3.CErr > 0 Then
          MsgBox objEvalua3.DErr, vbCritical, "Error " &
objEvalua3.CErr
          Exit Sub
      End If
      di(1) = -(H ^ 2) * Ri + (1 + (H / 2) * PI) * Alpha

      For i = 2 To N - 1
          X = a + i * H
          Qi = objEvalua2.EvaluarExpr(X)
          If objEvalua2.CErr > 0 Then
              MsgBox objEvalua2.DErr, vbCritical, "Error " &
objEvalua2.CErr
              Exit Sub
          End If
          ai(i) = 2 + (H ^ 2) * Qi

          PI = objEvalua1.EvaluarExpr(X)
          If objEvalua1.CErr > 0 Then
              MsgBox objEvalua1.DErr, vbCritical, "Error " &
objEvalua1.CErr
              Exit Sub
          End If
```

```
            bi(i) = -1 + (H / 2) * PI
            ci(i) = -1 - (H / 2) * PI

            Ri = objEvalua3.EvaluarExpr(X)
            If objEvalua3.CErr > 0 Then
                MsgBox objEvalua3.DErr, vbCritical, "Error " &
objEvalua3.CErr
                Exit Sub
            End If
            di(i) = -(H ^ 2) * Ri
        Next i

        X = b - H
        Qi = objEvalua2.EvaluarExpr(X)
        If objEvalua2.CErr > 0 Then
            MsgBox objEvalua2.DErr, vbCritical, "Error " &
objEvalua2.CErr
            Exit Sub
        End If
        ai(N) = 2 + (H ^ 2) * Qi

        PI = objEvalua1.EvaluarExpr(X)
        If objEvalua1.CErr > 0 Then
            MsgBox objEvalua1.DErr, vbCritical, "Error " &
objEvalua1.CErr
            Exit Sub
        End If
        ci(N) = -1 - (H / 2) * PI

        Ri = objEvalua3.EvaluarExpr(X)
        If objEvalua3.CErr > 0 Then
            MsgBox objEvalua3.DErr, vbCritical, "Error " &
objEvalua3.CErr
            Exit Sub
```

```
      End If
      di(N) = -(H ^ 2) * Ri + (1 - (H / 2) * PI) * Betha

    ' Ahora que se tienen los términos de la ecuación matricial
Ax = b
    ' Resolver por el método de factorización de Crout
      ReDim li(N)
      ReDim ui(N)
      ReDim Zi(N)

    ' Resolver Lz = b por sustitución progresiva
      li(1) = ai(1)
      ui(1) = bi(1) / ai(1)
      Zi(1) = di(1) / li(1)

      For i = 2 To N - 1
          li(i) = ai(i) - ci(i) * ui(i - 1)
          ui(i) = bi(i) / li(i)
          Zi(i) = (di(i) - ci(i) * Zi(i - 1)) / li(i)
      Next i

      li(N) = ai(N) - ci(N) * ui(N - 1)
      Zi(N) = (di(N) - ci(N) * Zi(N - 1)) / li(N)

    ' Resolver Ux = z por sustición regresiva
      ReDim Wi(N + 2)

      Wi(1) = Alpha
      Wi(N + 2) = Betha
      Wi(N + 1) = Zi(N)

      For i = N - 1 To 1 Step -1
          Wi(i + 1) = Zi(i) - ui(i) * Wi(i + 2)
```

```
      Next i


      Texto = "Método de Diferencias finitas:" & vbCrLf & _
              "Aproximar la solución de la ecuación diferencial
d2y/dt2 = (" & LCase(objEvalua1.Expresion) & _
              ")*dy/dx+("   &   LCase(objEvalua2.Expresion)   &
")*y+" & LCase(objEvalua3.Expresion) & _
              " en el intervalo [" & Str(a) & ", " & Str(b) &
"]"
      Texto = vbCrLf & Texto & vbCrLf & vbCrLf
      Texto = Texto & vbCrLf & "Xi" & vbTab & vbTab & vbTab &
"Wi" & vbCrLf


      ReDim UserData(2, N + 2)
      mTotalRows = 0
      DBGridResultados.AllowAddNew = True
      Formato = "#."
      For j = 2 To Slider1.Value
         Formato = Formato & "0"
      Next j
      Formato = Formato & "E-###"
      DBGridResultados.Columns(0).NumberFormat = Formato
      DBGridResultados.Columns(1).NumberFormat = Formato
      For i = 0 To N + 1
          X = a + i * H
          mTotalRows = mTotalRows + 1
          UserData(1, i + 1) = X
          UserData(2, i + 1) = Wi(i + 1)
          Texto = Texto & Format(X, Formato) & vbTab & vbTab &
Format(Wi(i + 1), Formato) & vbCrLf
      Next i
      Forms(0).ActiveForm.rtfText                              =
Forms(0).ActiveForm.rtfText.Text & Texto
      DBGridResultados.Refresh
      DBGridResultados.AllowAddNew = False
```

```
  End If

Salir:
    Exit Sub

VerificarError:
    If Err.Number = 6 Then
        MsgBox "Número fuera de donimio" & vbCrLf & _
              "Verifique los parametros de entrada.", vbExclamation, "Error"
        Resume Salir
    ElseIf Err.Number = 91 Then
        Resume Next
    Else
        MsgBox Err.Description, vbExclamation, "Error " & Err.Number
        Resume Salir
    End If
End Sub



Private Sub cmdCancelar_Click()
    Unload Me
End Sub

Private Sub cmdLimpiar_Click()
  If (MsgBox("¿Realmente desea limpiar los datos de todos los controles.", vbYesNo, "Confirmación") = vbYes) Then
      txtFormula1.Text = ""
      txtFormula2.Text = ""
      txtFormula3.Text = ""
      txtLsuperior.Text = ""
      txtLinferior.Text = ""
      txtFronteraa.Text = ""
```

```
      txtFronterab.Text = ""
      txtNodos.Text = ""
      Erase UserData
      mTotalRows = 0
      DBGridResultados.Refresh
      cmdDiffinitas.Enabled = False
      cmdCancelar.Default = True
  End If
End Sub

Private Sub Command1_Click()
    frmFunPers.Show vbModeless, Me
End Sub


Private Sub DBGridResultados_UnboundReadData(ByVal RowBuf As
RowBuffer, StartLocation As Variant, ByVal ReadPriorRows As
Boolean)
Dim CurRow&, Row%, Col%, RowsFetched%, Incr%

If ReadPriorRows Then
    Incr% = -1
Else
    Incr% = 1
End If

' Si StartLocation es Null se empieza a leer al
' principio o al final del conjunto de datos.
If IsNull(StartLocation) Then
    If ReadPriorRows Then
        CurRow& = RowBuf.RowCount - 1
    Else

        CurRow& = 0
```

```
    End If
Else
    ' Busca la posición desde la que comenzar a
    ' leer según el marcador StartLocation
    ' y la variable lngIncr%
    CurRow& = CLng(StartLocation) + Incr%
End If

' Se transfieren los datos desde la matriz de
' datos al objeto RowBuf que DBGrid usa para
' presentar los datos
For Row% = 0 To RowBuf.RowCount - 1
    If CurRow& < 0 Or CurRow& >= mTotalRows& Then Exit For
    For Col% = 0 To UBound(UserData, 1) - 1
        RowBuf.Value(Row%, Col%) = UserData(Col% + 1, CurRow& +
1)

    Next Col%
    ' Establece el marcador usando CurRow&,
     ' que es también nuestro índice de matriz
    RowBuf.Bookmark(Row%) = CStr(CurRow&)
    CurRow& = CurRow& + Incr%
    RowsFetched% = RowsFetched% + 1
Next Row%
RowBuf.RowCount = RowsFetched%
End Sub

Private Sub Form_Initialize()
  bytFormula = 2
End Sub

Private Sub Form_Load()
  bytFormula = 2
```

```
  Set objEvalua1 = New clsEvalua
  Set objEvalua2 = New clsEvalua
  Set objEvalua3 = New clsEvalua
End Sub


Private Sub ValidarParametros()
    If (Not IsEmpty(txtLinferior) And Not txtLinferior = "" And
IsNumeric(txtLinferior)) And _
        (Not IsEmpty(txtLsuperior) And Not txtLsuperior = ""
And IsNumeric(txtLsuperior)) And _
        (Not IsEmpty(txtFronteraa) And Not txtFronteraa = ""
And IsNumeric(txtFronteraa)) And _
        (Not IsEmpty(txtFronterab) And Not txtFronterab = ""
And IsNumeric(txtFronterab)) And _
        (Not IsEmpty(txtNodos) And Not txtNodos = "" And
IsNumeric(txtNodos)) And _
        (Not IsEmpty(Me.txtFormula1) And Not Me.txtFormula1 =
"") And _
        (Not IsEmpty(Me.txtFormula2) And Not Me.txtFormula2 =
"") And _
        (Not IsEmpty(Me.txtFormula3) And Not Me.txtFormula3 =
"") Then
            Me.cmdDiffinitas.Enabled = True
            Me.cmdDiffinitas.Default = True
            Exit Sub
    Else
        Me.cmdDiffinitas.Enabled = False
        Me.cmdCancelar.Default = True
    End If
End Sub


Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As
Integer)
  If (MsgBox("¿Realmente desea salir de la interfaz del
método?.", vbYesNo, "Confirmación") = vbYes) Then
```

```
      Cancel = False
  Else
      Cancel = True
  End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
  mTotalRows = 0
  Erase UserData
  Set objEvalua1 = Nothing
  Set objEvalua2 = Nothing
  Set objEvalua3 = Nothing
End Sub

Private Sub txtFormula1_Change()
    Call ValidarParametros
End Sub

Private Sub txtFormula1_GotFocus()
  bytFormula = 2
End Sub

Private Sub txtFormula1_Validate(Cancel As Boolean)
    Call ValidarParametros
End Sub

Private Sub txtFormula2_Change()
    Call ValidarParametros
End Sub

Private Sub txtFormula2_GotFocus()
  bytFormula = 3
End Sub
```

```
Private Sub txtFormula2_Validate(Cancel As Boolean)
    Call ValidarParametros
End Sub

Private Sub txtFormula3_Change()
    Call ValidarParametros
End Sub

Private Sub txtFormula3_GotFocus()
  bytFormula = 4
End Sub

Private Sub txtFormula3_Validate(Cancel As Boolean)
    Call ValidarParametros
End Sub

Private Sub txtFronteraa_Change()
    Call ValidarParametros
End Sub

Private Sub txtFronteraa_Validate(Cancel As Boolean)
Dim objEvalua As New clsEvalua

  objEvalua.Expresion = Trim(txtFronteraa.Text)
  If objEvalua.CErr > 0 Then
      MsgBox   objEvalua.DErr,   vbCritical,   "Error   "   &
objEvalua.CErr
      Cancel = False
      Exit Sub
  Else
      If objEvalua.ValidarVar() = True Then
          txtFronteraa.Text = objEvalua.EvaluarExpr
```

```
      Else
          MsgBox "La fórmula no admite variables." & vbCrLf & _
              "Modifiquela utilizando solo constantes.",
vbExclamation, "Error"
      End If
  End If

  If IsEmpty(txtFronteraa) Or txtFronteraa = "" Or
IsNumeric(txtFronteraa) Then
      Cancel = False
  Else
      MsgBox "El valor debe ser numérico" & vbCrLf & _
              "Cambie el dato a un valor numérico.",
vbExclamation, "Error"
      Cancel = True
      Exit Sub
  End If
  Call ValidarParametros
End Sub

Private Sub txtFronterab_Change()
    Call ValidarParametros
End Sub

Private Sub txtFronterab_Validate(Cancel As Boolean)
Dim objEvalua As New clsEvalua

  objEvalua.Expresion = Trim(txtFronterab.Text)
  If objEvalua.CErr > 0 Then
      MsgBox objEvalua.DErr, vbCritical, "Error " &
objEvalua.CErr
      Cancel = False
      Exit Sub
  Else
```

```
      If objEvalua.ValidarVar() = True Then
          txtFronterab.Text = objEvalua.EvaluarExpr
      Else
          MsgBox "La fórmula no admite variables." & vbCrLf & _
              "Modifiquela utilizando solo constantes.",
vbExclamation, "Error"
      End If
  End If

  If IsEmpty(txtFronterab) Or txtFronterab = "" Or
IsNumeric(txtFronterab) Then
      Cancel = False
  Else
      MsgBox "El valor debe ser numérico" & vbCrLf & _
              "Cambie el dato a un valor numérico.",
vbExclamation, "Error"
      Cancel = True
      Exit Sub
  End If
  Call ValidarParametros
End Sub

Private Sub txtLinferior_Change()
    Call ValidarParametros
End Sub

Private Sub txtLinferior_Validate(Cancel As Boolean)
Dim objEvalua As New clsEvalua

  objEvalua.Expresion = Trim(txtLinferior.Text)
  If objEvalua.CErr > 0 Then
      MsgBox objEvalua.DErr, vbCritical, "Error " &
objEvalua.CErr
      Cancel = False
```

```
      Exit Sub
  Else
      If objEvalua.ValidarVar() = True Then
          txtLinferior.Text = objEvalua.EvaluarExpr
      Else
          MsgBox "La fórmula no admite variables." & vbCrLf & _
              "Modifiquela utilizando solo constantes.",
vbExclamation, "Error"
      End If
  End If

  If IsNumeric(txtLsuperior) And IsNumeric(txtLinferior) Then
      If txtLsuperior > txtLinferior Then
          Cancel = False
      Else
          Cancel = True
          MsgBox "El límite inferior debe ser menor que el
límite superior" & vbCrLf & _
              "Cambie a un valor que cumpla esta condición.",
vbExclamation, "Error"
          Exit Sub
      End If
  ElseIf IsEmpty(txtLinferior) Or txtLinferior = "" Or
IsNumeric(txtLinferior) Then
      Cancel = False
  Else
      MsgBox "El valor debe ser numérico" & vbCrLf & _
              "Cambie el dato a un valor numérico.",
vbExclamation, "Error"
      Cancel = True
      Exit Sub
  End If
  Call ValidarParametros
End Sub
```

```
Private Sub txtLsuperior_Change()
    Call ValidarParametros
End Sub

Private Sub txtLsuperior_Validate(Cancel As Boolean)
Dim objEvalua As New clsEvalua

  objEvalua.Expresion = Trim(txtLsuperior.Text)
  If objEvalua.CErr > 0 Then
      MsgBox   objEvalua.DErr,   vbCritical,   "Error   "   &
objEvalua.CErr
      Cancel = False
      Exit Sub
  Else
      If objEvalua.ValidarVar() = True Then
          txtLsuperior.Text = objEvalua.EvaluarExpr
      Else
          MsgBox "La fórmula no admite variables." & vbCrLf & _
              "Modifiquela   utilizando   solo   constantes.",
vbExclamation, "Error"
      End If
  End If

  If IsNumeric(txtLsuperior) And IsNumeric(txtLinferior) Then
      If txtLsuperior > txtLinferior Then
          Cancel = False
      Else
          Cancel = True
          MsgBox "El inferior debe ser menor que el limite
superior" & vbCrLf & _
              "Cambie a un valor que cumpla esta condición.",
vbExclamation, "Error"
          Exit Sub
      End If
```

```
  ElseIf IsEmpty(txtLsuperior) Or txtLsuperior = "" Or
IsNumeric(txtLsuperior) Then
      Cancel = False
  Else
      MsgBox "El valor debe ser numérico" & vbCrLf & _
             "Cambie el dato a un valor numérico.",
vbExclamation, "Error"
      Cancel = True
      Exit Sub
  End If
  Call ValidarParametros
End Sub



Private Sub txtNodos_Change()
    Call ValidarParametros
End Sub

Private Sub txtNodos_Validate(Cancel As Boolean)
Dim objEvalua As New clsEvalua

  objEvalua.Expresion = Trim(txtNodos.Text)
  If objEvalua.CErr > 0 Then
      MsgBox objEvalua.DErr, vbCritical, "Error " &
objEvalua.CErr
      Cancel = False
      Exit Sub
  Else
      If objEvalua.ValidarVar() = True Then
          txtNodos.Text = objEvalua.EvaluarExpr
      Else
          MsgBox "La fórmula no admite variables." & vbCrLf & _
              "Modifiquela utilizando solo constantes.",
vbExclamation, "Error"
```

```
      End If
  End If

  If IsNumeric(txtNodos) Then
      If txtNodos >= 4 And Not (Round(txtNodos) - txtNodos <>
0) Then
          Cancel = False
      Else
          MsgBox "El valor debe ser un número entero mayor que
3" & vbCrLf & _
              "Cambie el dato a un valor entero mayor que 3.",
vbExclamation, "Error"
          Cancel = True
          Exit Sub
      End If
  ElseIf IsEmpty(txtNodos) Or txtNodos = "" Then
      Cancel = False
  Else
      MsgBox "El valor debe ser numérico" & vbCrLf & _
              "Cambie el dato a un valor numérico.",
vbExclamation, "Error"
      Cancel = True
      Exit Sub
  End If
  Call ValidarParametros
End Sub
```

Se recuerda que este módulo y los demás los podrá encontrar en el CD que se anexa a este documento.

V. Prueba del software

5.1. Introducción

En Visual Basic la prueba del software es interactivo ya que se realiza mientras se escribe. Además se puede ejecutar el programa en cualquier momento del desarrollo del software, por lo cual no se tiene que esperar a que el software este terminado en su totalidad. Con una parte funcional que se disponga, basta para ejecutar la aplicación. El depurador interrumpe la ejecución para acceder al modo de interrupción y podamos realizar los cambios pertinentes. Sin embargo, terminado el software en su totalidad se proceden a ser pruebas más completas y dirigidas al objetivo de su desarrollo, pruebas como las que a continuación se presentan.

5.2. Integración de Romberg

Problema 1

Aproximar $\int_0^{\pi} \sin x dx$. El resultado exacto es 2. Con el algoritmo de Romberg implementado se obtiene la siguiente tabla con 9 cifras significativas y un error porcentual de 10^(-7).

Tabla 4: Resultados del problema 1 de la integración de Romberg

5.07537661E-15					
1.57079633E0	2.09439510E0				
1.89611890E0	2.00455975E0	1.99857073E0			
1.97423160E0	2.00026917E0	1.99998313E0	2.00000555E0		
1.99357034E0	2.00001659E0	1.99999975E0	2.00000002E0		
1.99839336E0	2.00000103E0	2.00000000E0	2.00000000E0	2.00000000E0	2.00000000E0

La primera columna son los resultados de la regla del trapecio compuesta, los demás resultados son refinamientos sucesivos de la extrapolación de Richardson. El último valor es la integral buscada.

Problema 2

Aproximar a $\int_1^{1.5} e^{-x^2} dx$. El valor exacto de la integral con siete decimales es 0.1093643. Se obtiene la siguiente tabla con 7 cifras significativas y un error porcentual de 10^(-5).

Tabla 5: Resultados del problema 2 de la integración de Romberg

1.183197E-1			
1.115627E-1	1.093104E-1		
1.099114E-1	1.093610E-1	1.093643E-1	
1.095009E-1	1.093641E-1	1.093643E-1	1.093643E-1

La integral buscada es el valor de la intersección de la fila cuatro y la columna cuatro. Los restantes valores son aproximaciones que no alcanzan la precisión requerida y fueron explicados en el cuadro anterior.

5.3. Cuadratura de Gauss-Legendre

Problema 1

Considérese el problema de aproximar a $\int_1^{1.5} e^{-x^2} dx$. El valor exacto de la integral con siete decimales es 0.1093643.

Con ***n = 2*** obtenemos 1.094003E-1

Con ***n = 3*** obtenemos 1.093642E-1

Con ***n = 4*** obtenemos 1.093643E-1

Compare los resultados con el problema 2 de la integración de Romberg.

Problema 2

Aproximar la integral $\int_0^{\pi/2} \sin x dx$. El valor exacto de esta integral es 1.

Con ***n = 2*** obtenemos 9.984726E-1

Con ***n = 3*** obtenemos 1.000008E0

Con ***n = 4*** obtenemos 1.000000E0

5.4. Método de Runge-Kutta-Fehlberg

Problema 1

Aproximar la solución del problema de valor inicial

$$y' = y - t^2 + 1, 0 \le t \le 2, y(0) = 0.5,$$

que tiene la solución $y(t) = (t+1)^2 + 0.5e^t$. Los parámetros de entrada son la tolerancia 10^(-5), un tamaño máximo de paso 0.25 y un tamaño mínimo de paso 0.01. Los resultados se muestran en la tabla siguiente:

Tabla 6: Resultados del problema 1 del método de Runge-Kutta-Fehlberg.

t_i	$y_i = y(t_i)$	w_i	R_i
0.000000E0	0.5	5.000000E-1	0.000000E0
2.500000E-1	0.9204873	9.204886E-1	6.211110E-6
4.865522E-1	1.3964884	1.396491E0	4.487106E-6
7.293332E-1	1.9537446	1.953749E0	4.272161E-6
9.793332E-1	2.5864198	2.586426E0	3.774635E-6
1.229333E0	3.2604520	3.260461E0	2.438091E-6
1.479333E0	3.9520844	3.952096E0	7.219341E-7
1.729333E0	4.6308127	4.630827E0	1.481657E-6
1.979333E0	5.2574687	5.257486E0	4.311125E-6
2.000000E0	5.3054720	5.305490E0	4.049330E-10

Donde:

w_i : es la aproximación del valor de $y_i = y(t_i)$

R_i : es el error cometido en la aproximación.

5.5. Métodos iterativos para la solución de sistemas de ecuaciones lineales

Problema 1

Resolver el sistema lineal $\mathbf{Ax} = \mathbf{b}$ dado por

$$
\begin{aligned}
E_1 &: 10x_1 - x_2 + 2x_3 = 6,\\
E_2 &: -x_1 + 11x_2 - x_3 + 3x_4 = 25,\\
E_3 &: 2x_1 - x_2 + 10x_3 - x_4 = -11,\\
E_4 &: 3x_2 - x_3 + 8x_4 = 15
\end{aligned}
$$

La solución única exacta es $\mathbf{x} = (1,2,-1,1)^t$. Las iteraciones que se generan con el método de Jacobi, con una tolerancia de 10^(-3) y un vector de aproximación inicial $\mathbf{x^0} = (0,0,0,0)^t$ se presentan en la siguiente tabla:

Tabla 7: Resultados del problema 1 del método de Jacobi.

K	0	1	2	3	4	5	6	7	8	9
$X_1^{(k)}$	0	6.0000E-1	1.0473E0	9.3264E-1	1.0152E0	9.8899E-1	1.0032E0	9.9813E-1	1.0006E0	9.9967E-1
$X_2^{(k)}$	0	2.2727E0	1.7159E0	2.0533E0	1.9537E0	2.0114E0	1.9922E0	2.0023E0	1.9987E0	2.0004E0
$X_3^{(k)}$	0	-1.1000E0	-8.0523E-1	-1.0493E0	-9.6811E-1	-1.0103E0	-9.9452E-1	-1.0020E0	-9.9904E-1	-1.0004E0
$X_4^{(k)}$	0	1.8750E0	8.8523E-1	1.1309E0	9.7384E-1	1.0214E0	9.9443E-1	1.0036E0	9.9889E-1	1.0006E0

Con la misma aproximación inicial y la misma tolerancia, generamos las siguientes iteraciones con el método de Gauss-Seidel:

Tabla 8: Resultados del problema 1 del método de Gauss-Seidel.

k	0	1	2	3	4	5
$x_1^{(k)}$	0	6.0000E-1	1.0302E0	1.0066E0	1.0009E0	1.0001E0
$x_2^{(k)}$	0	-9.8727E-1	2.0369E0	2.0036E0	2.0003E0	2.0000E0
$x_3^{(k)}$	0	-1.1000E0	-1.0145E0	-1.0025E0	-1.0003E0	-1.0000E0
$x_4^{(k)}$	0	8.7886E-1	9.8434E-1	9.9835E-1	9.9985E-1	9.9999E-1

Problema 2

Dado el sistema lineal $\mathbf{Ax} = \mathbf{b}$

$$\begin{aligned} 4x_1 + 3x_2 &= 24, \\ 3x_1 + 4x_2 - x_3 &= 30, \\ -x_2 + 4x_3 &= -24, \end{aligned}$$

resolverlo aplicando los métodos de Gauss-Seidel y el SOR con $\omega = 1.25$ y una aproximación inicial $\mathbf{x}^{(0)} = (3,4,-5)^t$. La solución exacta es $(3,4,-5)^t$.

En las tablas siguientes se incluyen las primeras siete iteraciones para cada método. Para que las iteraciones tengan una exactitud de 8 cifras significativas, el método de Gauss-Seidel requiere 28 iteraciones, en contraste con las 14 iteraciones que exige el método de sobrerrelajación con $\omega = 1.25$.

Gauss-Seidel

Tabla 9: Resultados del problema 2 del método de Gauss-Seidel.

K	0	1	2	3	4	5	6	7
$x_1^{(k)}$	1	5.250000E0	3.140625E0	3.087891E0	3.054932E0	3.034332E0	3.021458E0	3.013411E0
$x_2^{(k)}$	1	3.812500E0	3.882813E0	3.926758E0	3.954224E0	3.971390E0	3.982119E0	3.988824E0
$x_3^{(k)}$	1	-5.046875E0	-5.029297E0	-5.018311E0	-5.011444E0	-5.007153E0	-5.004470E0	-5.002794E0

SOR con $\omega = 1.25$.

Tabla 10: Resultados del problema 2 del método de SOR..

K	0	1	2	3	4	5	6	7
$x_1^{(k)}$	1	6.312500E0	2.622314E0	3.133303E0	2.957051E0	3.003721E0	2.996328E0	3.000050E0
$x_2^{(k)}$	1	3.519531E0	3.958527E0	4.010265E0	4.007484E0	4.002925E0	4.000926E0	4.000259E0
$x_3^{(k)}$	1	-6.650146E0	-4.600424E0	-5.096686E0	-4.973490E0	-5.005714E0	-4.998282E0	-5.000349E0

5.6. Método de la transformada rápida de Fourier

Problema 1

Sea $f(x) = x^4 - 3x^3 + 2x^2 - \tan x(x-2)$. Para determinar el polinomio trigonométrico interpolante de cuarto orden para los datos $\{x_j, y_j\}_{j=0}^{7}$ donde $x_j = j/4$ y $y_j = f(x_j)$ es necesario transformar el intervalo [0, 2] a $[-\pi, \pi]$. La transformación esta dada por

$$z_j = \frac{2\pi(x_i - x_0)}{x_n - x_0} - \pi ,$$

así que el método de la Transformada de Fourier toma los datos atendiendo a este variable formando el conjunto de pares, pero introducimos el intervalo original [0, 2] (el algoritmo se encarga de la interpretación o viéndolo de otra manera usted).

Así tenemos

$$\left\{ z_j, f\left(\frac{(x_n - x_0)}{2\pi}(z_i + \pi) + z_0 \right) \right\}_{j=0}^{7} .$$

El polinomio interpolante en *z* es

$$S_4(z) = \frac{a_0 + a_4 \cos 4z}{2} + \sum_{k=1}^{3} \left(a_k \cos kz + b_k \sin kz \right)$$

Los coeficientes discretos de Fourier obtenidos con la TRF se presentan en la siguiente tabla, con 2^3 = 8 nodos muestreados:

Tabla 11: Resultados del problema 1 del método de la transformada rápida de Fourier.

k	a_k	b_k
0	1.523957E0	0.000000E0
1	7.718408E-1	-3.863738E-1
2	1.730370E-2	4.687500E-2
4	6.863041E-3	-1.137378E-2

El polinomio trigonométrico $S_4(x)$ en [0, 2] se obtiene sustituyendo $z = \frac{2\pi(x - x_0)}{x_n - x_0} - \pi$. En la tabla siguiente se muestran los valores de *f(x)* y $S_4(x)$.

Tabla 12: Resultados 2 del problema 1 del método de la transformada rápida de Fourier.

X	*f(x)*	$S_4(x)$	$\lvert f(x) - S_4(x) \rvert$
0.125	0.26440	0.25001	1.44×10^{-2}
0.375	0.84647	0.84647	5.66×10^{-3}
0.625	1.35824	1.35824	3.27×10^{-3}
0.875	1.61515	1.61515	2.33×10^{-3}
1.125	1.36471	1.36471	2.02×10^{-3}
1.375	0.71931	0.71931	2.33×10^{-3}
1.625	0.07496	0.07496	4.14×10^{-3}
1.875	-0.12301	-0.13301	1.27×10^{-2}

Los datos de la tabla anterior son resultados que se obtienen al retomar los datos que son arrojados por el método de la transformada rápida de Fourier que se ha implementado. Se pone de relieve, un caso particular y la utilidad del método. Se ha hecho el reemplazo de *z* por *x,* puesto que el intervalo no era $[-\pi, \pi]$.

5.7. Método de Diferencias finitas

Problema 1

El problema con valor de frontera

$$y'' = -\frac{2}{x}y' + \frac{2}{x^2}y + \frac{\sin(\ln x)}{x^2},\ 1 \le x \le 2,\ y(1) = 1,\ y(2) = 2$$ tiene la solución exacta

$$y = c_1 x + \frac{c_2}{x^2} - \frac{3}{10}\sin(\ln x) - \frac{1}{10}\cos(\ln x),$$

donde

$$c_2 = \frac{1}{70}\left[8 - 12\sin(\ln 2) - 4\cos(\ln 2)\right] \approx -0.03920701320$$

$$c_1 = \frac{11}{10} - c_2 \approx 1.1392070132\,.$$

Utilicemos el método de Diferencias finitas para aproximar la solución con un número de nodos igual a 11.

Los resultados se proporcionan en la siguiente tabla:

Tabla 13: Resultados del problema 1 del método de diferencias finitas..

x_i	w_i	$y(x_1)$	$\lvert w_i - y(x_i)\rvert$
1.000000E0	1.00000000E0	1.00000000	
1.100000E0	1.09260052E0	1.09262930	$2.88\text{x}10^{-5}$
1.200000E0	1.18704313E0	1.18708484	$4.17\text{x}10^{-5}$
1.300000E0	1.28333687E0	1.28338236	$4.55\text{x}10^{-5}$
1.400000E0	1.38140205E0	1.38144595	$4.39\text{x}10^{-5}$
1.500000E0	1.48112026E0	1.48115942	$3.92\text{x}10^{-5}$
1.600000E0	1.58235990E0	1.58239246	$3.26\text{x}10^{-5}$
1.700000E0	1.68498902E0	1.68501396	$2.49\text{x}10^{-5}$
1.800000E0	1.78888175E0	1.78889853	$1.68\text{x}10^{-5}$
1.900000E0	1.89392110E0	1.89392951	$8.41\text{x}10^{-6}$
2.000000E0	2.00000000E0	2.00000000	

Donde, w_i es la aproximación, que se ha obtenido, de $y(x_1)$.

VI. Implantación del software

6.1. Introducción

Como resultado o producto del proyecto se ha obtenido un software que se denomina SAN ALFA; SAN por ser un software de análisis numéricos y ALFA porque se espera que se enriquezca con otros proyectos similares que incluyan otras áreas que no se abarcaron aquí.

En esta etapa se realizó un manual del usuario que complementa el software desarrollado y que describe las capacidades del software, cómo instalarlo, especificaciones del hardware y otras características necesarias para poder utilizar la aplicación. El manual del usuario, junto con la teoría matemática están disponibles desde el menú de ayuda de la aplicación y desde las interfaces de los métodos numéricos. Por lo que el usuario dispondrá de la ayuda teórica y práctica al instante.

6.2. Manual del usuario de la aplicación

El Manual del usuario, es un manual acerca de cómo trabajar con SAN ALFA. Para ordenar las características y posibilidades de SAN ALFA, el Manual del usuario comprende temas como la instalación de la aplicación, requisitos mínimos para su instalación y la exploración de sus capacidades.

6.2.1. Instalación de la aplicación.

SAN ALFA se instala en su equipo mediante el programa de instalación. El programa de instalación instala SAN ALFA y otros componentes del producto, desde el CD-ROM al disco duro. También instala los archivos necesarios para mostrar la documentación del CD.

Importante: No puede copiar simplemente los archivos del CD-ROM al disco duro y ejecutar SAN ALFA. Debe usar el programa de instalación, que descomprime e instala los archivos en los directorios apropiados.

6.2.1.1. Requisitos mínimos del sistema

Antes de instalar SAN ALFA, asegúrese de que el equipo cumple los siguientes requisitos mínimos de hardware y software:

- Microprocesador Pentium® 90MHz o superior.
- Pantalla VGA de 640x480 o de resolución superior compatible con Microsoft Windows.
- 24 MB de RAM para Windows 95, 32 MB para Windows NT, 64 MB o más para sistemas operativos superiores como Windows ME, Windows 2000, etc.
- Microsoft Windows NT 3.51 o posterior, o Microsoft Windows 95 o posterior.
- Una unidad de CD-ROM.
- Un *mouse* (ratón) u otro dispositivo de puntero.
- Requisitos de espacio en disco duro: aproximadamente 10 MB para la aplicación y su documentación.

6.2.1.2. Pasos para instalar la aplicación SAN ALFA

Cuando ejecuta el programa de instalación, se crea un directorio para SAN ALFA.

Para realizar la instalación desde el CD-ROM

1. Inserte el disco en la unidad de CD-ROM.
2. Utilice el comando apropiado del entorno operativo para ejecutar el Programa de instalación, que se encuentra en el directorio raíz del CD-ROM. Si está activado AutoPlay en el sistema, el Programa de instalación se cargará automáticamente cuando inserte el disco.
3. Seleccione **Instalar SAN ALFA**.
4. Siga las instrucciones de instalación que aparecen en la pantalla.

Puede ejecutar el programa de instalación tantas veces como sea necesario. Por ejemplo, puede ejecutar el programa de instalación para volver a instalar SAN ALFA en otro directorio, pero es redundante.

6.2.2. Iniciar SAN ALFA

En cuanto termine el procedimiento de instalación, puede iniciar SAN ALFA mediante el botón Inicio de la barra de tareas de Windows.

Antes de ejemplificar como acceder a las diferentes opciones de los métodos numéricos se describirá una herramienta indispensable en la implementación de los mismos: el Analizador de Expresiones Matemáticas que quedará como software reutilizable para nuevos proyectos similares.

6.2.3. El Analizador de Expresiones Matemáticas

Acepta como entrada cualquier cadena que represente una expresión aritmética o algebraica; retorna un valor numérico en doble precisión. Puede utilizarse para calcular fórmulas dadas en tiempo de corrida, por ejemplo para trazar y tabular funciones, para hacer cálculos numéricos y más.

Expresiones matemáticas típicas que acepta el analizador son:

Tabla 14: Ejemplos de expresiones matemáticas que admite el analizador de expresiones.

`1+(2-5)*3+8/(5+3)^2`	`Sqrt(2)`
`(a+b)*(a-b)`	`X^2+3*x+1`
`(-1)^(2n+1)*x^n/n`	`(3000000)/144`
`256.33*Exp(-t/12)`	`(1+(2-5)*3+8/(5+3)^2)/sqrt(5^2+3^2)`
`2+3x+2x^2`	`0.25x + 3.5y + 1`
`sin(2*pi()*x)+cos(2*pi()*x)`	`Sqrt(4^2+3^2)`

El analizador de expresiones matemáticas tiene como antecesor el analizador de expresiones matemáticas versión 2.0, distribuida en Internet sin fines de lucro por el profesional mejicano A. David Garza Marín. Esta versión de distribución fue modificada sustancialmente para agregarle características de robustez, transparencia, confiabilidad, eficiencia y soporte para varias características útiles, tales como, funciones seccionadas, constantes en notación científica y otras. De tal forma que esta versión desarrollada es sustancialmente distinta a la versión mencionada.

6.2.3.1. Evaluación de expresiones

El analizador permite ejecutar expresiones que incluyan exponenciación (^), multiplicación (*), división (/), división de enteros (\), módulo aritmético (%), suma (+) y resta (-). Existe una jerarquía de operadores, esta jerarquía se aplica de izquierda a derecha. Primero se evalúan las operaciones de potencia (^), luego, de izquierda a derecha, las multiplicaciones y divisiones (la que se encuentre primero), luego la división de enteros, seguida por el módulo aritmético y finalmente las sumas y restas (igual al anterior). A su vez, se cuenta con la verificación de la ley de los signos.

Es decir, si se proporciona una expresión como la siguiente:

```
5 + 3 * 8
```

El resultado sería 29 y no 64, dado que primero se ejecuta la multiplicación (3 * 8 = 24) y al resultado se le suma a 5, (5 + 24 = 29). Si deseas que ciertas partes de la expresión sean evaluadas antes, debes encerrarlas entre paréntesis. En el ejemplo anterior, para obtener 64 la expresión sería:

```
(5 + 3) * 8
```

El siguiente ejemplo también es válido:

```
(5 + (4 - 1)) * 8
```

De este modo, se anidan paréntesis en la expresión. Así se ejecutaría inicialmente la expresión entre los paréntesis internos, para luego continuar con lo demás. Se aceptan también, como signos de agrupación, los corchetes [] y las llaves {}, los cuales se pueden anidar.

Importante: por la manera en que esta diseñado el analizador, la exponenciación debe manipularse con sumo cuidado cuando intervienen operadores de resta. Específicamente si la base esta precedida por un signo menos, pero que no forma parte de esta, habría que poner entre paréntesis la operación de exponenciación, por ejemplo: 2 – (3^2); de igual manera, pero menos peligroso, debido a que el analizador advierte de este error, es la exponenciación que tiene un exponente con signo negativo, en este caso habría que poner el exponente entre paréntesis, por ejemplo: 2 + (3^(-2)).

6.2.3.2. Uso de variables

El analizador permite utilizar variables en las expresiones. Para el caso, deberán proporcionarse los valores de cada variable para poder llevar a cabo el proceso. Por ejemplo, si la expresión a evaluar es

$$a^2 + 2ab + b^2$$

se escribiría en el analizador de expresiones

```
(a^2) + (2*(a*b)) + (b^2)
```

Los paréntesis en realidad no son necesarios.

Nótese que los nombres de las variables deben ser cualquier letra desde la a hasta la z. Así, sólo podemos introducir nombres de varibles de un carácter. No utilices caracteres acentuados, especiales o ñ para los nombres de variables.

Los siguientes son nombres válidos de variables:

```
x, y, z, a, b
```

Los nombres de variables no son sensibles a mayúsculas y minúsculas. Por lo tanto es indistinto hacer referencia a una variable como *x* ó *X*, *y* ó *Y*, etc.

Los siguientes no son nombres válidos de variables:

```
10x, Validación.de.datos, Cálculo, Año_&_Mes, Xyz, Valor1.
```

6.2.3.3. Uso de funciones

El Analizador de Expresiones Matemáticas incluye el soporte de funciones matemáticas y trigonométricas, en su mayoría estándar en los lenguajes de Visual Basic. Para utilizar una función, debe establecerse su nombre en la expresión a evaluar. Así, si la expresión a evaluar es :

```
PI() * R ^ 2
```

Primero, se accede a la función intrínseca PI, que devuelve el valor aproximado de π (3.141592654). Nótese que las funciones incluyen paréntesis. Si se omiten los paréntesis, el analizador de expresiones matemáticas fallará en su intento. Si una función necesita argumento y no se especifica se asume implícitamente como cero.

Existen funciones, la mayoría de ellas, que requieren de argumentos entre sus paréntesis. Por ejemplo, la función SQRT obtiene la raíz cuadrada del número indicado. La siguiente expresión:

```
SQRT(25)
```

dará por resultado 5. Siempre que sea necesario, las funciones podrán utilizarse en las expresiones. De hecho, también pueden agregarse expresiones o variables entre los paréntesis. La siguiente expresión:

```
SQRT(x * 12)
```

obtendrá la raíz cuadrada del número x multiplicado por 12. Obviamente, se deberá dar un valor a la variable x para poder obtener el resultado adecuado.

Existen varias funciones predefinidas en el analizador. La siguiente tabla describe el nombre de la función y su propósito:

Tabla 15: Funciones incorporadas en el analizador de expresiones.

Nombre	**Propósito**
ABS(x)	Obtener el valor absoluto de x.
ACOS(x)	Obtener el arco coseno de x (en radianes).
ACOSH(x)	Obtener el arco coseno hiperbólico de x.
ASIN(x)	Obtener el arco seno de x (en radianes).
ASINH(x)	Obtener el arco seno hiperbólico de x.

ACOT(x)	Obtener el arco cotangente de x (en radianes).
ACOTH(x)	Obtener el arco cotangente hiperbólico de x.
ATAN(x)	Obtener el arco tangente de x (en radianes).
ATANH(x)	Obtener el arco tangente hiperbólico de x.
COS(x)	Obtener el coseno de un ángulo dado en x (en radianes).
COSH(x)	Obtener el coseno hiperbólico de x.
COT(x)	Devolver la cotangente de un ángulo dado en x (en radianes).
COTH(x)	Devolver la cotangente hiperbólico de x.
E_()	Devolver el valor aproximado de e (2.71828182845905).
EXP(x)	Devolver el número e (que es la base de los logaritmos naturales) elevado a la potencia x.
INT(x)	Devolver la parte entera de un número
LN(x)	Devolver el logaritmo natural (base e) del número x.
LOG(x)	Devolver el logaritmo común (base 10) del número x.
PI()	Devolver el valor aproximado de π (3.141592654).
SGN(x)	Devolver el signo del número x (-1 si es negativo, 1 si es positivo, 0 si es cero)
SIN(x)	Devolver el seno de un ángulo dado en x (en radianes).
SINH(x)	Devolver el seno hiperbólico de x.
SQRT(x)	Devolver la raíz cuadrada de x.

TAN(x)	Devolver la tangente de un ángulo dado en x (en radianes).
TANH(x)	Devolver la tangente hiperbólica de x.

6.2.3.4. Definición de funciones personalizadas

Este analizador también permite la definición de funciones propias o personalizadas. Esto amplía el panorama en sí y permite extender su funcionalidad. Para definir una función propia disponemos de una interfaz en cada método invocada por el botón Personalizar función..., en la cual podemos agregar, modificar y eliminar funciones personalizadas. Ejemplo de función personalizada:

```
Nombre       : LOG10
Definición   : LN(x)/LN(10)
```

En el ejemplo, definimos una función que obtiene el logaritmo base 10. La primera cadena establece el nombre de la función (las convenciones para el nombre de función son las mismas que tienen las variables, con excepción, de que debe poseer mas de 1 carácter) y la segunda la definición propia de la función. En caso de que se incurra en un error de sintaxis en el nombre o en la definición de la función se generara un error, por ejemplo: "Nombre incorrecto asignado a la función". Solo se pueden incluir funciones incorporadas en el analizador dentro de la definición de la función, además de los operadores binarios que se mencionaron en una sección previa.

Nótese que las funciones personalizadas sólo pueden tener una variable llamada x en su definición que es el valor con el que la función trabajará. En el ejemplo dado, x es el número del que se calculará el logaritmo base 10. Si se intenta escribir algún otro nombre de variable, se obtendrá un error 10: "La definición de la función es incorrecta". Se puede colocar la variable x donde se desee y cuantas veces se desee. Sin embargo cuando se utilice la función personalizada en una expresión su argumento puede ser cualquier expresión válida. No se podrán usar funciones personalizadas en su definición. Así, la siguiente definición de función será correcta:

```
Nombre       : MiFuncion

Definición   : (x * 10) / (PI() * x ^ 2)
```

Las funciones propias son persistentes. Esto quiere decir que si agregas alguna, no se eliminará cuando finalices tu trabajo con el Analizador de Expresiones Matemáticas. La siguiente vez que lo utilices, allí estará la función tal como la hayas definido. No obstante, cabe aclarar que las funciones definidas por el usuario se almacenan en un archivo de texto plano llamado ANALIZEM.FUN que deberá estar en la misma carpeta (o directorio) donde se encuentre la aplicación.

Al ser un archivo de texto plano, puedes cargarlo con cualquier editor de textos compatible y ver su contenido. Pero no intentes modificarlo a mano (excepto cuando sepas exactamente lo que haces), pues podría haber problemas con la ejecución subsecuente de las funciones. No se agregarán aquellas funciones que hayan sido modificadas erróneamente desde el archivo.

6.2.3.5. Expresiones que representan funciones seccionadas

El analizador de expresiones matemáticas soporta funciones seccionadas, es decir funciones que están definidas por más de una regla de correspondencia. Las distintas secciones se separan por medio de un punto y como “;” y los intervalos por medio de una coma “,”.

Ejemplos:

x^2 +2, x < 0; x, x > 3

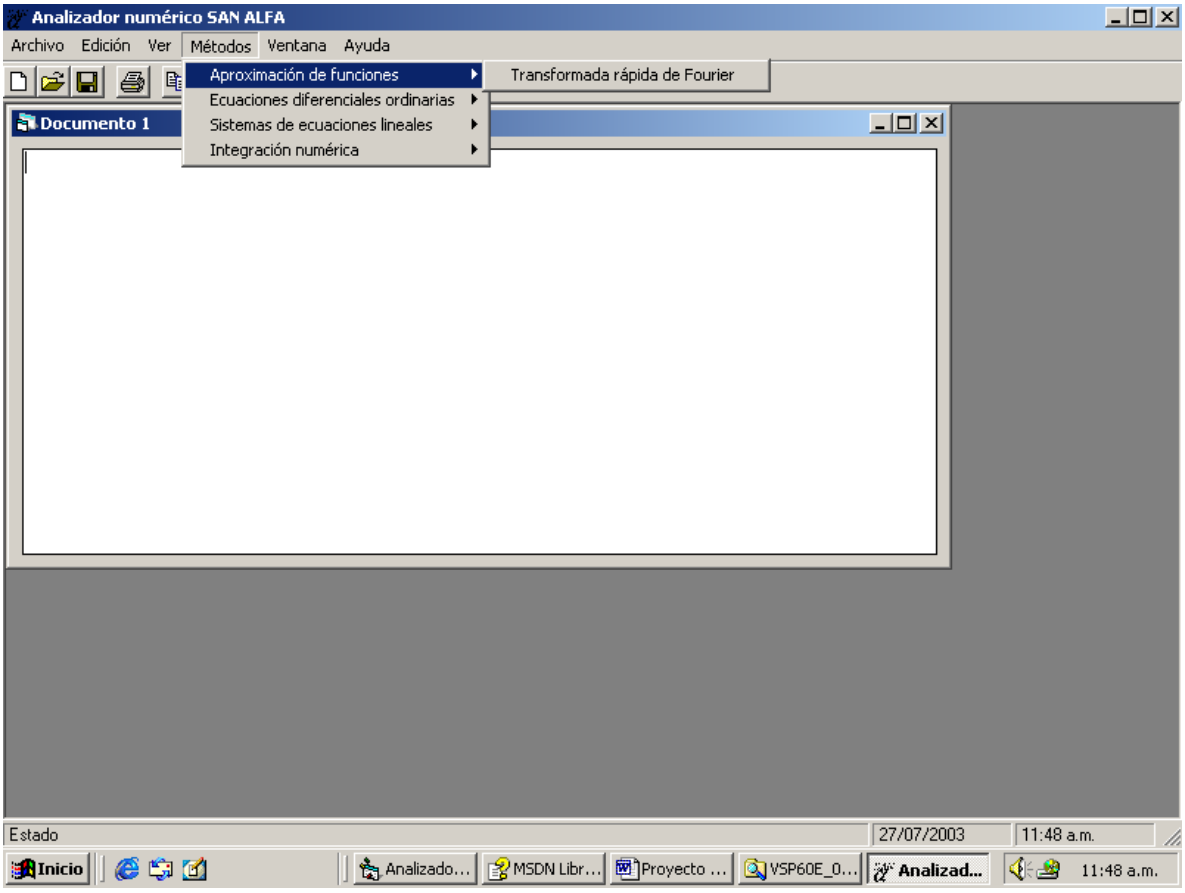
que representa la función seccionada $f(x)=\begin{cases} x^2+2, & x<0; \\ x \quad , & x>3 \end{cases}$

0, x = pi(); x, -pi() < x < pi(); 0, x = pi()

que representa la función seccionada $f(x)=\begin{cases} 0, & x=-\pi; \\ x, & -\pi<x<\pi; \\ 0, & x=\pi \end{cases}$

6.2.4. ¿Cómo ejecutar y trabajar con los distintos métodos?

La pantalla principal de SAN ALFA muestra una barra de menú familiar para un usuario de Microsoft Windows, con excepción del menú métodos que es como nuestro taller. En este menú encontramos los diferentes métodos con los cuales podemos trabajar y que hemos mencionado durante el desarrollo de este documento.



Refiérase al capítulo IV para una descripción de las diferentes interfaces de los métodos. El manual en el menú ayuda de la aplicación muestra esta sección en este preciso lugar.

6.2.5. Ejemplos

Refiérase al capítulo V para ver ejemplos de problemas típicos resueltos con SAN ALPHA. El manual en el menú ayuda de la aplicación muestra esta sección en el lugar correspondiente.

Análisis de resultados

- ✓ Se desarrolló un software de análisis numérico que implementa los siguientes métodos de interés:
  - Iteraciones de Jacobi
  - Iteraciones de Gauss-Seidel
  - De Sobrerrelajación Sucesiva (SOR)
  - De Runge Kutta-Fehlberg
  - Algoritmo de Romberg
  - Cuadratura de Gauss-Legendre
  - Transformada rápida de Fourier
  - Diferencias Finitas
- ✓ El software tiene todas las características para servir como una herramienta visual interactiva para el aprendizaje, enseñanza y aplicación de los métodos anteriormente mencionados.
- ✓ El software se desarrolló con el lenguaje de programación Visual Basic orientado al ambiente del sistema operativo Windows con el objetivo de aprovechar las ventajas que este lenguaje y el sistema operativo poseen: su fácil uso, su popularidad, su estandarización de aplicaciones, etc.
- ✓ Se incluyó un sistema de ayuda (textos de ayuda desplegables, opción de menú de ayuda con índice y contenido) que contiene teoría detallada acerca de los fundamentos matemáticos de los métodos y un manual de usuario; íconos para la ejecución rápida de comandos incluidas en barras de herramientas y menús contextuales.
- ✓ Se pondrá a disposición de la comunidad universitaria y demás personas interesadas, un software gratuito, que podrá ser utilizado en la resolución de problemas de análisis numérico en las áreas seleccionadas, o para fines académicos.

- ✓ Se espera motivar a los estudiantes de la carrera en Licenciatura Estadística y Computación al desarrollo de software orientado a facilitar y mejorar el proceso educativo de nuesto país.

Conclusiones y recomendaciones

- SAN ALPHA es una aplicación de análisis numérico para sistemas de ecuaciones lineales, integración simple, ecuaciones diferenciales ordinarias y aproximación de funciones que posee una precisión doble, es decir, aproximadamente 15 dígitos significativos.
- Aunque posee una robustez muy bien planeada, tiene limitaciones en cuanto a su aplicación, por ejemplo, solo puede resolver sistemas de ecuaciones lineales con una dimensión de hasta 1700x1700. Si se sobrepasan las limitaciones recibirá un mensaje de error sugiriéndole el posible problema. También es necesario, considerar las especificaciones para las cuales el método funciona de manera satisfactoria, que se mencionan en la determinación de requerimientos.
- SAN ALFA tiene características para poder emplearse como herramienta en la enseñanza de los métodos que en él se implementan y aquellos relacionados. Además, perfectamente, se puede emplear con propósitos prácticos en la solución de problemas propios de esta aplicación.
- Para motivar a los estudiantes a emprender proyectos de su área aplicados a diferentes campos del conocimiento se deben plantear estrategias bien delimitadas para alcanzar ese objetivo. Estrategias que nacen del espíritu o visión de la Licenciatura en Estadística y computación.
- Es recomendable que aunque la aplicación se utilice para propósitos prácticos se conozcan las condiciones bajo las cuales los métodos funcionan eficientemente con cierta precisión. Para este propósito la aplicación SAN ALFA dispone de un sistema de ayuda que presenta la teoría matemática que explica estas condiciones y además, presenta conceptos prácticos del manejo del software.
- Se recomienda que se evalúe el impacto de este proyecto y su producto para que en un futuro pueda ampliarse su ámbito de aplicación a áreas que no se tomaron en cuenta. Todas las herramientas básicas para ello se han proporcionado en este trabajo.

- Este documento se estructuro de forma que sirva de guía metodologica adecuadada para obtener con éxito aplicaciones o productos similares.
- Es recomendable desarrollar software para su aplicación en diversas áreas (educación, comercio, ciencia, etc.) para fomentar el desarrollo de la ciencia de la computación en la facultad.
- Proyectos de esta índole no deben pasar desapercibidos, a pesar de su corto alcance, porque fomenta la generación de ciencia y tecnología. De esta manera, en un futuro prometedor pasaremos a ser productores y no consumidores de productos de buena calidad del extranjero. Esto es válido en cualquier área del conocimiento.

Referencias bibliográficas

- ✓ James A. Senn..
  1992, Análisis y Diseño de Sistemas de Información.
  Segunda Edición. Editorial McGraw Hill.
- ✓ H. M. Deitel / P. J. Deitel.
  Como Programar en C/C++.
  Segunda Edición. Editorial Prentice Hall.
- ✓ Erich R. Bühler.
  Visual Basic .NET. Guía de migración y actualización.
  Primera Edición. Editorial McGraw Hill.
- ✓ Grupo Editoral de Microsoft.
  Manual del Programador de Visual Basic 6.0.
  Editorial Microsoft Press.
- ✓ Timothy M. O' Brien / Steven J. Pogge / Geoffrey E. White.
  Microsoft Access. Desarrollo de soluciones.
  Traducción de la primera edición en Ingles. Editorial McGraw Hill.
- ✓ Richard L. Burden / J. Douglas Faires.
  Análisis Numérico.
  Tercera Edición. Grupo Editorial Iberoamericana.
- ✓ Melvin J. Maron / Robert J. Lopez.
  Análisis numérico. Un enfoque práctico.
  Tercera Edición. Editorial CECSA.
- ✓ Steven C.Chapra / Raymond P. Canale.
  Métodos numéricos para ingenieros. Con aplicaciones en computadoras personales.
  Traducción de la primera edición en Ingles. Editorial McGraw Hill.

Anexos

Junto a este documento se adjunta el producto o software que se logró obtener satisfactoriamente. El CD contiene la siguiente información:

- La aplicación SAN ALFA que viene con un instalador o setup.
- El proyecto de SAN ALFA en Visual Basic.
- Un archivo de este documento.

Si desea obtener el CD dirijase a la Escuela de Matemática de la Facultad de Ciencias Naturales y Matemática y haga su solicitud.