03063

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

POSGRADO EN CIENCIA E INGENIERIA DE LA COMPUTACION

ELABORACION DE UN SISTEMA DE COMPUTO INTEGRADO QUE FUNCIONA EN RED, PARA LA ENSEÑANZA DE METODOS NUMERICOS USANDO EL PROCESO DE DESARROLLO DE SOFTWARE: EN ESPIRAL.

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRA EN CIENCIAS (COMPUTACION)

P R E S E N T A :

MARIA DEL CARMEN GOMEZ FUENTES

DIRECTOR DE TESIS:
DR. VLADIMIR TCHIJOV.

MEXICO, D. F. ENERO 2004

Agradezco:

A Dios.

A mi esposo Jorge Cervantes, porque sin su valioso apoyo hubiera sido imposible comenzar siquiera esta maestría.

A mi tutor el Dr. Tchijov porque siempre estuvo en la mejor disposición para supervisar y mejorar este trabajo.

A la M.C.C. Araceli Nivon, por todo el tiempo que me brindó.

Elaboración de un sistema de cómputo integrado que funciona en red, para la enseñanza de métodos numéricos usando el proceso de desarrollo de software: *en espiral.*

CONTENIDO.

1.- OBJETIVOS.

- Desarrollar un sistema de cómputo integrado que incluye la implementación de métodos numéricos, el parser especializado, y la página de Web interactiva que funcione en la red para que los alumnos puedan no solo consultar *en que consiste* un método numérico y *como se usa*, sino también *introducir diferentes datos de entrada* para observar el comportamiento y comparar los diferentes métodos. Para este propósito, emplear el *proceso de desarrollo de software* llamado *"en espiral"* para que se garantice la calidad del sistema.

- Utilizar un *proceso de desarrollo de software* que garantice la calidad del sistema.

- Para la codificación de los métodos numéricos utilizar programación orientada a objetos con JAVABuilder y una herramienta para la elaboración de páginas WEB.

2.- DESCRIPCIÓN.

El sistema consiste en una página de Internet por medio de la cual, los alumnos de las carreras de ingeniería de la Facultad de Estudios Superiores Cuautitlan de la UNAM, pueden aprender métodos numéricos y reforzar su aprendizaje en clase poniéndolos en practica en la computadora para observar su funcionamiento y su comportamiento.

El sistema consta de una parte teórica, que consiste en un conjunto de páginas HTML en donde se exponen breves descripciones de cada uno de los métodos, y de una parte práctica, en donde el alumno introduce los datos necesarios para observar los resultados parciales y finales de un método seleccionado. Es decir, el sistema no sólo es una página Web con ligas a otras páginas, sino que se ejecuta un trabajo en la computadora remota, con esto se obtienen las ventajas de un sistema didáctico que puede correr programas en una red de computadoras.

Teniendo en cuenta que los métodos numéricos son una herramienta para resolver problemas difíciles por medio de *muchos* pasos fáciles, la única manera de observar un ejemplo en donde realmente se ejecuten *muchos* pasos fáciles es en una computadora, ya que en clase, por razones obvias, es necesario limitarse a la resolución "a mano" de problemas sencillos que requieran cuando mucho 5 o 6 iteraciones.

Además, para un buen aprendizaje de la asignatura, es necesario que los alumnos puedan observar cuestiones fundamentales en los métodos numéricos como por ejemplo:

- Que pasa cuando en un mismo problema varía la tolerancia.
- Que variación hay en los resultados utilizando diferentes métodos para un mismo
- Que sucede cuando se dan diferentes valores iniciales.

Todo esto y más, se puede observar claramente con este sistema.

El sistema no solo se usa para la enseñanza de los métodos numéricos, sino también para su aplicación. Por ejemplo, los alumnos de las carreras de Química Industrial, Ingeniería Química, Ingeniería en Alimentos, Químico Farmacéutico Biólogo y Química de la FESC pueden hacer uso de algunos de los métodos que incluye la página para resolver problemas de fisicoquímica.

El sistema tiene las siguientes características:

- Los métodos numéricos están escritos en JAVA. El código con el que se llevan a cabo los métodos es transparente para el usuario, debido a que el objetivo es que el alumno aprenda métodos numéricos, independientemente de sus conocimientos sobre programación.

- El propósito de este trabajo no es la velocidad y optimización de los cálculos numéricos sino su enseñanza, por esto no se usa Fortran o C. Se utiliza JAVA, porque así los programas pueden ser ejecutados desde una página Web.

- El proyecto está basado en el programa de estudios de la asignatura "Métodos Numéricos" de Ingeniero Mecánico Electricista (IME) de la FESC.

Existen herramientas muy poderosas para el uso de los métodos numéricos, como Matlab, Maple, etc., pero estas herramientas además de ser muy costosas (mas de 1000 dolares), no están diseñadas para usarse en red.

El sistema que se presenta en este trabajo fue elaborado siguiendo un *proceso de diseño software*, el método de diseño utilizado fue el resultado de combinar lo leído en la bibliografía (ver referencias) con la experiencia laboral de la autora de esta tesis en el departamento de diseño de software de Alcatel a lo largo de 8.5 años.

El código está abierto para agregar mas métodos sin tener que reescribirlo.

Desde que estuvieron funcionando algunos métodos, el sistema se comenzó a utilizar en clases reales de "Métodos Numéricos" con alumnos de IME en Campo 4 de la FESC, con esto surgieron los detalles de interacción de los alumnos con el sistema que no se pudieron prever al iniciar el trabajo, pero que sin embargo si contribuyeron a mejorar el sistema durante su desarrollo.

3.- ANTECEDENTES.

Se hizo una investigación en Internet buscando en sitios en donde pudieran existir páginas interactivas para la enseñanza de métodos numéricos y se encontró lo siguiente:

Existen varios cursos de métodos numéricos (estos no son interactivos) como:

- "*Exploring Numerical Methods with MathCad*" en
- http://euler.uni_koblenz.de/ictmt3/cd-rom/pdf/bogacki.pdf
- "*Applied Numerical Methods1 (MIME 6150)*" en
- http://memslab.eng.utoledo.edu/~jreed/mime-615/Sylabus.pdf este libro no se puede ver libremente.

La pagina de la Universidad de Plymouth: en:
fttp://www.tech.ply.ac.uk/maths/COURSES/Math1103/math1103.html habla de un software interactivo para PC con salida gráfica usado para el curso de métodos numéricos que se imparte en esta universidad, pero es un software privado, no menciona que pueda ser usado en red.

La Universidad de Pórtland indica en su página:
http://www.me.pdx.edu/~gerry/class/ME352/Sylabus02.pdf: que utilizan MATLAB para la implantación y aplicación de los métodos numéricos.

Lo más parecido a el sistema que se presenta en esta tesis para la enseñanza de métodos numéricos, se encuentra en la página de la Universidad de Newcastle en:
http://lorien.ncl.ac.uk/ming/Dept/swot/optnotes.htm la cual contiene un compendio de métodos:

Integración Numérica (contiene un tutorial elaborado por Joseph L. Zachary, con un simulador de demostración para la integración de varias funciones que vienen en un menú).

Ecuaciones diferenciales ordinarias.

Obtención de raices (Contiene la simulación del método de bisección para la función de demostración).

Además:

1.- En http://uranus.ee.auth.gr/lessons/1/index.html se encuentra el método de Newton-Raphson expuesto ampliamente y con claridad en páginas HTML por los griegos Amarantidis Kostas y Makris Lambros, pero los ejemplos del método vienen en PASCAL y FORTRAN. por lo que es necesario bajar los ejemplos a la máquina del usuario (no es una página interactiva).

2.- Eric A. Carlen publicó en http:/www.math.gatech.edu/~carlen/applets/index.html un applet poco amigable en donde se pretende demostrar el funcionamiento del método de Newton-Raphson. Esta página no trae instrucciones de cómo usar el applet, sin una explicación previa, no se entiende ni el método, ni como utilizar el applet.

3.- Shela V. Belur tiene una página interactiva elaborada en JAVAScript en: http://www.geocities.com/SiliconValley/2902/newton.htm . La presentación del método es mucho mas amigable que la de Carlen, sin embargo, o no funciona correctamente o no trae las instrucciones suficientes como para que el método funcione.

En conclusión:

No se encontró ninguna página que tenga un compendio de todos los métodos que se incluyen en este trabajo, los cuales se requieren para la asignatura de Métodos Numéricos.

De los tres métodos que encontraron con software interactivo, el de bisección solo funciona para una función de demostración, para el método de Newton-Raphson el software que se encontró o no funciona o carece de las instrucciones adecuadas para su ejecución y finalmente para los métodos de integración se encontró el mejor software, sin embargo este también es únicamente para funciones de demostración.

Lo que se pretende con este sistema interactivo para la enseñanza de métodos numéricos, es que los alumnos puedan introducir sus propias funciones y que vean el comportamiento numérico del método observando los resultados parciales, además cada página contiene tanto una breve explicación del método correspondiente, como una explicación de cómo utilizarlo dentro del sistema.

4.- Justificación del uso del "Proceso de diseño software" para la elaboración de este proyecto.

El desarrollo y el mantenimiento de productos de programación requiere de un enfoque mas sistemático que el necesario en el desarrollo de programas para uso personal.

Para un proyecto pequeño se necesita un programador dedicado de uno a seis meses, obteniendo como resultado un producto de entre 1000 y 2000 líneas de código fuente, ejemplos de estos programas son paquetes de aplicaciones científicas escritos por ingenieros para resolver problemas numéricos. Un proyecto pequeño requiere poca interacción con programadores e incluso entre programadores y clientes. Los estándares técnicos de documentación y notaciones, así como las revisiones sistemáticas de los proyectos deben usarse, aunque el grado de formalidad será menor al empleado en proyectos grandes [Fairley,1998].

Existen varios procesos de diseño, desde los mas antiguos (como el diseño en cascada), hasta los mas modernos (como el Proceso Unificado: http://www.enterpriseunifiedprocess.info/).

Para un proyecto elaborado por una sola persona puede servir cualquier proceso de diseño de software preestablecido. Los beneficios de los procesos modernos se notan cuando hay dos o mas personas trabajando para el proyecto ya que es cuando se vuelven necesarias la comunicación y la administración de las versiones. Sin embargo, se quizo investigar y

profundizar un poco sobre los procesos de diseño software modernos para adquirir el conocimiento para trabajos futuros.

5.- Aseguramiento de la calidad del software.

El aseguramiento de la calidad consta de aquellos procedimientos, técnicas y herramientas aplicadas por profesionales para asegurar que el producto cumple con los estándares pre-especificados durante el ciclo de desarrollo del producto con el objeto de alcanzar un nivel de excelencia a nivel comercial [Sommerville1989].

Como procedimiento de aseguramiento de la calidad del sistema desarrollado en esta tesis se utilizó las "Inspecciones software" [Gilb&Graham, 1998].

Se establecieron las reglas de inspección que se encuentran en el apéndice "A" y se generó un documento que se encuentra en el apéndice D: "*registro de las inspecciones*" en donde están registrados:

- Las fechas de las inspecciones.
- Los resultados producidos por cada inspección.
- Los cambios importantes que se le hicieron al proyecto durante su proceso de desarrollo.

6.- Etapas del desarrollo del sistema.

Se llevaron a cabo las tres etapas mencionadas por Antonio de Amescua y coautores [De Amescua y otros, 1995], que son:

6.1.- La *investigación:* en donde se identificaron los objetivos del sistema, su alcance y restricciones:

Objetivos:

- Una página Web en la que los alumnos puedan ejecutar diferentes métodos numéricos y observar resultados parciales y finales con fines didácticos.
- Deberá existir una documentación en donde se explique brevemente cada método y su algoritmo.
- El sistema contendrá una lista de métodos a utilizar agrupados por categorías.
- Los métodos numéricos que requieran una función de entrada deberán tener implantado un "*parser*" para que el alumno pueda introducir las funciónes.

Alcance:

-Esta página deberá poder ser accedida desde cualquier máquina de la UNAM a través de Internet usando el navegador Explorer o Netscape.

Restricciones:

- Se utilizarán los "applets", que son "...pequeños programas que se ejecutan en el contexto de una página Web en cualquier ordenador..."[Ceballos,2002]. Esto tiene las siguientes restricciones:
  1.- No se puede leer ni escribir a ningún archivo de la máquina de los usuarios (dadas las características de seguridad del lenguaje JAVA).
  2.- La velocidad de ejecución del método numérico dependerá de la máquina del usuario.

6.2.- Análisis de requisitos: En donde se analizó qué herramientas software se iban a necesitar para la elaboración del sistema y que lenguaje de programación es el más apropiado.

- Para elaborar las páginas HTML que presentan la información y los applets se seleccionó el Editor de páginas HTML Coffee Cup.

- Para elaborar el código en JAVA que permita la ejecución de métodos numéricos, así como la interface de entrada de datos y presentación de resultados se seleccionó el compilador JAVA Builder versión 7. ya que Jbuilder es un Ambiente de Desarrollo Integrado (IDE: Integrated Development Environment) para programar en JAVA. [Hubbard, 1998][1]

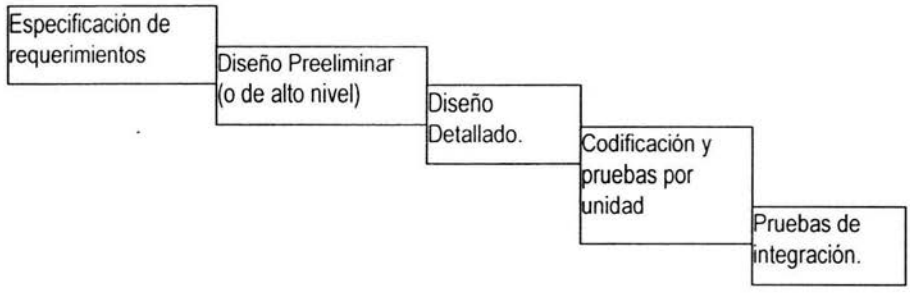
- Para publicar en Internet la página del sistema, se solicitó un espacio en el servidor de la FESC con la siguiente dirección:

http://asesorias.cuautitlan2.unam.mx/mnumericos

[1] Para mayor información puede ver: "Como usar el compilador Jbuilder [Jamsa, 2000, pags 461-466] y la liga de internet: http://www.borland.com/jbuilder/

6.3.- Diseño: E s difícil entender y mapear con anticipación todos los requerimientos de un sistema que se va a producir, hoy en día la mayoría de los investigadores mas importantes de la ingeniería de software piensan que no es posible mapear todos los requerimientos de un sistema antes de empezar a desarrollarlo. Así que los procesos de desarrollo de software más populares actualmente son *los procesos iterativos e incrementales*, donde se pasa varias veces por las fases del "diseño en cascada".

El proceso de "diseño software: en cascada" descritos por Donald Reifer [Reifer, 1993, pag 59] y por John Marciniak [Marciniak, 1994, pag 360] se muestra en la siguiente figura:



Este proceso es muy antiguo e inflexible ya que no permite regresar a etapas anteriores. Una manera de mejorarlo es combinar las fases del proceso de desarrollo en cascada con un *desarrollo incremental*, así, utilizando el ***ciclo de vida evolutivo*** también llamado ***proceso en espiral*** cada vez que se pasa por alguna de las fases se van mejorando los productos, que en este caso son: *el documento* y *el código*.

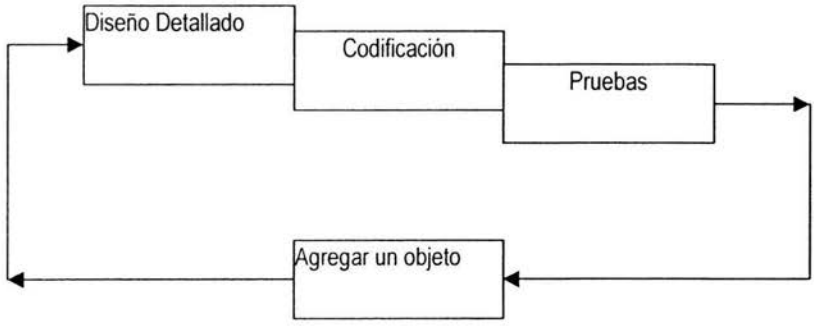
El uso antiguo y convencional del proceso en espiral solo dividía el sistema en "pedazos" y se ejecutaba el proceso en cascada una vez para cada pedazo. Actualmente, las técnicas modernas permiten *modificar los requerimientos* al iniciar una nueva iteración del proceso en espiral, esto permite una mayor flexibilidad en la concepción del proyecto, lo que lo hace mas adaptable a las verdaderas necesidades independientemente de lo que se haya planteado inicialmente.

El ***ciclo de vida evolutivo*** reconoce la necesidad de la revisión del desarrollo conforme éste progresa y sus requerimientos, se presenta por lo general mediante un modelo de "espiral" o caracol, donde se presenta un desarrollo ligado, esto es, un proceso incremental en vez de una secuencia de fases compartidas:

1.-Revisión de los objetivos.
2.-Plan para el desarrollo de la opción seleccionada.
3.-Desarrollo.
4.-Revisión de la entrega de la etapa para verificar que satisfaga los objetivos iniciales

La principal ventaja de adoptar este modelo de ciclo de vida es que permite un considerable nivel de flexibilidad en un proyecto [Norris Rigby 1994].

Al ciclo de vida evolutivo se le adaptó el proceso de "Secuencia típica para un proyecto orientado a objetos" expuesto por Roger Pressman [Pressman, 1998] de tal forma que la secuencia:



Se repite para ir agregando los objetos (que en este caso son los métodos) nuevos al sistema.

En los siguientes dos capítulos: **Diseño de alto nivel** y **Diseño detallado** se describe como quedó el sistema una vez aplicado el proceso de espiral, el cual está reportado en el apéndice C: "*Registro de las diferentes fases del proceso en espiral*".

6.4.- Codificación y pruebas: El código de los programas no se incluye en este trabajo, en su lugar se incluyen los algoritmos de cada uno de los métodos empleados estos pueden consultarse en la sección 7.3.
El proceso de las pruebas puede consultarse en el apéndice F: "*protocolo de pruebas a realizar para cada uno de los métodos (y los resultados obtenidos)*".

7.- DISEÑO PREELIMINAR O DE ALTO NIVEL.

7.1.-Descripción del Sistema.

En el sistema desarrollado existe una página principal HTML en donde hay "ligas" o "enlaces" a las páginas de cada uno de los métodos.

A continuación se presentan los grupos de métodos numéricos que contiene el sistema junto con los métodos de cada grupo:

GRUPO	METODOS DEL GRUPO
Obtención de raíces	○ Newton-Raphson. ○ Biseccción.
Sist. de Ec. lineales	○ Método de Jácobi ○ Método de Gauss-Seidel
Interpolación	○ De Newton ○ De Lagrange
Integración	○ Trapezoidal ○ Simpson 1/3 ○ Simpson 3/8
Ecuaciones Diferenciales	○ Método de Gauss ○ Gauss mejorado ○ Runge-Kutta orden 4

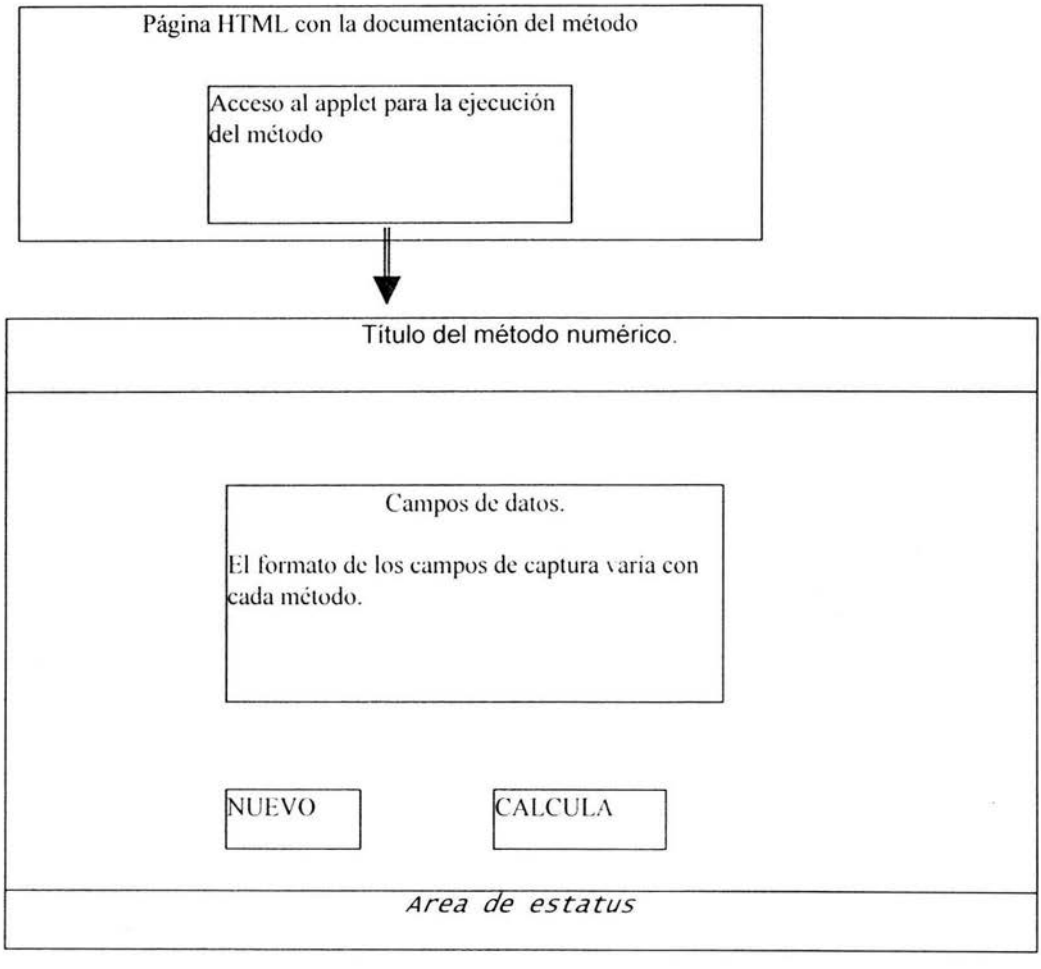
La página de un método contiene 3 partes:

1.- Una breve descripción de en que consiste el método.

2.- Un applet que contiene el software necesario para que el usuario introduzca los datos y obtenga resultados.

3.- Una pequeña guía para el usuario para que sepa como utilizar el applet, es decir como introducir los datos e interpretar los resultados.

Cada applet con un método numérico esta elaborado con el siguiente esquema general:



Los campos de captura varían según el método elegido, a continuación se presentan los campos de captura para cada uno de los métodos implementados:

- Newton-Raphson.

Campos que aparecen en el área de captura:

Función: [ ]

Derivada: [ ]

Valor Inicial: [ ]

Tolerancia: [ ]

Como resultado el sistema despliega una pantalla con los resultados de todos los cálculos parciales que utiliza el método, que son: número de iteración, raíz obtenida y tolerancia (definida como el valor absoluto de $X_{i+1} - X_i$) , donde X_i es el valor aproximado de la raíz obtenida después de *i* iteraciones.

Además despliega una pantalla adicional que indica el resultado final alcanzado.

- Bisección.

Campos que aparecen en el área de captura:

Función: [ ]

Intervalo: [ ] [ ]

Tolerancia: [ ]

Como resultado el sistema despliega una pantalla con todos los cálculos parciales que utiliza el método, que son: número de iteración, los límites del intervalo: X_inferior, X_superior, el punto medio del intervalo: X_media y la tolerancia (definida como el valor absoluto de $X\_media_{i+1} - X\_media_i$).

Además despliega una pantalla adicional que indica el resultado final alcanzado.

- Jacobi

Campos que aparecen en la primera pantalla de captura:

Numero de variables del sistema [ ]

En función del *número de variables* proporcionado se tiene

una segunda pantalla de captura, donde aparecen :los campos necesarios para la captura de:

a).- La matriz A de un sistema Ax=B.
b).- El vector B de este mismo sistema.
c).- El vector X con los valores iniciales.
d).- La tolerancia.

Como resultado el sistema despliega una pantalla con los "vectores solución" parciales indicando el número de iteración.

Además despliega una pantalla adicional con el vector solución alcanzado, en caso de que no haya convergencia se despliega otra pantalla para indicarlo.

- Gauss-Seidel

Las pantallas de captura para este método son exactamente las mismas que para el de Jacobi.

Como resultado el sistema despliega una pantalla con los "vectores solución" parciales indicando el número de iteración.

Además despliega una pantalla adicional con el vector solución alcanzado, en caso de que no haya convergencia se despliega otra pantalla para indicarlo.

- Newton

Campos que aparecen en la primera pantalla de captura:

Numero de datos de la tabla: [ ]

Primera X de la tabla: [ ]

Espaciamiento en X: [ ]

En función del *número de datos de la tabla* proporcionado se tiene una segunda pantalla de captura, donde aparece una tabla con los valores precalculados de X (estos campos no se pueden editar) y campos en blanco para que el usuario introduzca los valores de Y. También se pide:

a).- El valor de la X a interpolar.
b).- La X inicial para el uso del polinomio de Newton.

Nota: los campos de los incisos a) y b) están protegidos, de manera que el usuario no podrá extrapolar ni usar una X inicial que no esté dentro del intervalo de la tabla.

- Lagrange

Campos que aparecen en la primera pantalla de captura:

Numero de datos de la tabla: [ ]

En función del *número de datos de la tabla* proporcionado se tiene una segunda pantalla de captura, donde aparece una tabla con campos en blanco para que el usuario introduzca los valores de X y de Y. También se pide:

a).- El valor de la X a interpolar.
(protegido contra extrapolaciones).

- Trapezoidal

Campos que aparecen en la primera pantalla de captura:

Numero de datos de la tabla: [ ]

Primera X de la tabla: [ ]

Espaciamiento en X: [ ]

En función del *número de datos de la tabla* proporcionado se tiene una segunda pantalla de captura, donde aparece una tabla con los valores precalculados de X (estos campos no se pueden editar) y campos en blanco para que el usuario introduzca los valores de Y. Al oprimir el botón "INTEGRAR" se despliega otra pantalla con el resultado de la integración.

- Simpson 1/3

Campos que aparecen en la primera pantalla de captura:

Numero de datos de la tabla: [ ]

Primera X de la tabla: [ ]

Espaciamiento en X: [ ]

En función del *número de datos de la tabla* proporcionado se

tiene una segunda pantalla de captura, donde aparece una tabla con los valores precalculados de X (estos campos no se pueden editar) y campos en blanco para que el usuario introduzca los valores de Y. Al oprimir el botón "INTEGRAR" se despliega otra pantalla con el resultado de la integración, siempre y cuando el *número de datos de la tabla* sea un número *impar*. En caso contrario se despliega una pantalla de advertencia en lugar de realizar la integración.

- Simpson 3/8

Campos que aparecen en la primera pantalla de captura:

Numero de datos de la tabla: [ ]

Primera X de la tabla: [ ]

Espaciamiento en X: [ ]

En función del *número de datos de la tabla* proporcionado se tiene una segunda pantalla de captura, donde aparece una tabla con los valores precalculados de X (estos campos no se pueden editar) y campos en blanco para que el usuario introduzca los valores de Y. Al oprimir el botón "INTEGRAR" se despliega otra pantalla con el resultado de la integración, siempre y cuando el *número de datos de la tabla –1* sea un *múltiplo de 3*. En caso contrario se despliega una pantalla de advertencia en lugar de realizar la integración.

- Euler

Campos que aparecen en la pantalla de captura

$Y' = f(x,y)$: [ ]

Condición inicial, Y_0: [ ]

Intervalo: [ ] [ ]

Espaciamiento: [ ]

La función f(x,y) es de dos variables. Al oprimir El botón "CALCULAR" aparece una pantalla con una tabla con la función solución usando el *método de Euler* para ecuaciones diferenciales de primer orden con condiciones iniciales, la tabla se obtiene en el intervalo proporcionado con el espaciamiento constante en x indicado en la pantalla de captura de los datos.

- Euler Mejorado

  Los campos que aparecen en la pantalla de captura son exactamente los mismos que los del método de Euler, y el resultado se presenta de la misma manera solo que el resultado se calcula utilizando el método de Euler mejorado.

- Runge Kutta

  Los campos que aparecen en la pantalla de captura son exactamente los mismos que los del método de Euler, y el resultado se presenta de la misma manera solo que el resultado se calcula utilizando el método de Runge Kutta.

7.2.- Justificación de los métodos numéricos elegidos para conformar al sistema:

En el programa de la asignatura "Métodos Numéricos" aprobado por el H. Consejo Técnico para las carreras de Ingeniería Electrónica, Ingeniería Mecánica, Ingeniería Eléctrica e Ingeniería Industrial están especificados, entre otros, los siguientes métodos:

Para la obtención de raíces:

1.- Newton-Raphson.
2.- Bisección.

Resolución de sistemas de ecuaciones lineales:

1.- Método de Jacobi.
2.- Método de Gauss-Seidel.

Métodos de Interpolación:

1.- Interpolación de Newton.
2.- Interpolación de Lagrange.

Métodos de Integración Numérica:

1.- Integración trapezoidal.
2.- Integración de Simpson 1/3.
3.- Integración de Simpsin 3/8.

Métodos para resolver ecuaciones diferenciales de primer orden:

1.- Método de Gauss.
2.- Método de Gauss mejorado.
3.- Método de Runge-Kutta de orden 4.

En la presente tesis, se implementan estos métodos numéricos, aunque el sistema está abierto a la incorporación en un futuro de otros métodos.

7.4.- Descripción del "*PARSER*".

Algunos de los métodos numéricos que componen el sistema desarrollado requieren una función de entrada de una o de dos variables, por esto se desarrolló un "*parser*".

Existen diversos tipos de parsers o intérpretes, desde los mas generales, que sirven para interpretar lenguajes (por ejemplo "pequeño C" [Schildt,2000]), hasta los mas específicos, como el que se requiere en este proyecto, en el que se necesita interpretar funciones de una y de dos variables.

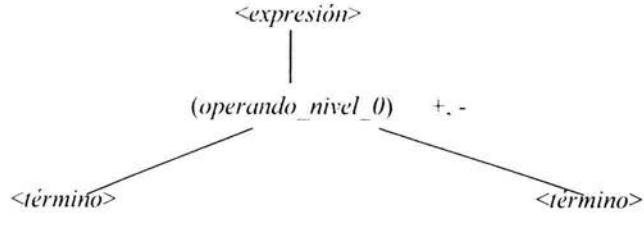
En este proyecto se desarrollo un parser escrito en JAVA que toma en cuenta las necesidades específicas de la tesis, en particular, evaluar funciones de una y de dos variables independientes. El código esta abierto para incluir en el futuro la evaluación de funciones de 3 y más variables para sistemas de ecuaciones diferenciales ordinarias.

Es un parser orientado a objetos "recursivo-descendente"[Schildt,1996] que evalúa expresiones algebraicas simples y complejas.

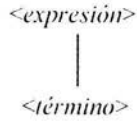
La *clase* **ParserFuncion** hace posible que un usuario instancie funciones reales de una variable real mediante una cadena de caracteres que representa a una función.

La estructura del parser es la siguiente:

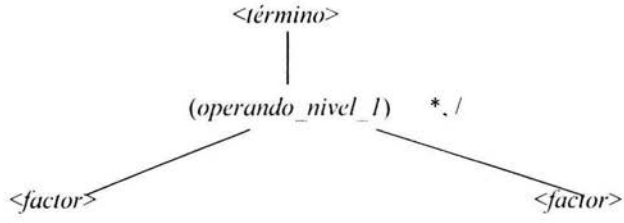
El elemento mas general es una "*expresión*", esta puede constar de un solo "*término*" o de la "*operación de nivel cero*" de dos *términos*:



O bien:

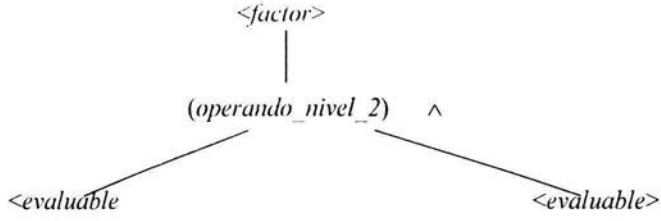


El elemento "*término*", consta de una "*operación de nivel 1*" de dos "*factores*" o bien de un solo "*factor*":

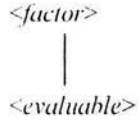


O bien:

El elemento "*factor*", consta de una *"operación de nivel 2"* de dos "*evaluables"* o bien de un solo "*evaluable*":



O bien:



Y finalmente un "*evaluable*" puede ser:

- Un número.
- Una variable.
- La función de una *expresión*.
- (Otra expresión).

Cada *término*, *factor* y *evaluable* pueden tener signo positivo o negativo.
Los números están formados por dígitos del 0 al 9 y pueden tener o no punto decimal.
Las variables pueden ser:

- x (para el parser de una variable)
- x, y (para el parser de dos variables)

La recursividad se lleva a cabo de la siguiente manera:
Se forman *series* de distintos niveles de acuerdo con lo siguiente:

- *Serie de nivel 0*: son sumas o restas de *términos* (series de nivel 1).
- *Serie de nivel 1*: son multiplicaciones o divisiones de *factore* (series de nivel 2)*s*.
- *Serie de nivel 2*: son potencias de *evaluables*.

El parser toma uno o dos parámetros reales que serán los valores de la o las variables con que se solicita la evaluación.

El parser lee la cadena, revisa su sintaxis y en caso de haber error termina la ejecución enviando un mensaje de error.

Si no hay error de sintaxis, la evaluación se ejecuta como sigue:
Se comienza en el estado "leyendo expresión" que de inmediato lee una *serie de nivel 0* en el procedimiento "leyendo una serie de nivel *n*" el cual hace lo siguiente:

Lee una serie de operandos de nivel n con signo, separados entre sí por un operador de nivel *n* y va haciendo las operaciones conforme se van leyendo de izquierda a derecha. Aquí se tienen dos subestados que son:

- Leyendo un **operando** de nivel *n*.
- Leyendo un **operador** de nivel *n*.

Leyendo un operando de nivel *n*:

Si el nivel no es todavía el nivel mas alto, entonces se obtiene el valor de un operando de nivel *n* con signo mediante un llamado recursivo a la misma función "leyendo serie" pero con un nivel superior *n+1*. En otras palabras, primero se efectúan las operaciones de nivel superior y luego las de nivel *n*.

Si ya se esta leyendo el nivel mas alto, entonces el parser pasa al estado "leyendo dato evaluable".

Leyendo dato evaluable:

En este estado se puede leer un número, una variable o recursivamente, una expresión completa si el dato está entre paréntesis. En este último caso se pasa al estado "leyendo expresión" y, cuando se encuentra el paréntesis que cierra, el parser regresa al punto donde encontró el paréntesis que abre teniendo ya el valor del dato evaluable para continuar con el estado "leyendo un operador de nivel *n*". En este estado también puede leerse un llamado a una de las funciones aceptadas en cuyo caso se toma como parámetro el valor de la expresión encerrada entre paréntesis que debe seguir después del nombre de la función.

Leyendo un operador de nivel *n*.

Si se lee un operador de nivel *n* entonces se hace la operación de nivel *n* leída entre el resultado anterior y un nuevo operador que será leído en el estado "leyendo un operando de nivel *n*".

En caso de no ser así se da por finalizada la serie y se regresa el valor calculado de ésta.

Este parser es un intérprete del lenguaje descrito por la gramática siguiente:

```
<programa>    ::= {<Asignación>}
<Asignación>  ::= <variable>=<expresión>
<variable>    ::= <letra>{<caracter>}
<letra>       ::= a..z|A..Z
<caracter>    ::= <letra>|<dígito>
<dígito>      ::= 0..9
<expresión>   ::= <término>{<op_0><término>}
<op_0>        ::= +|-
<término>     ::= {<op_0>}<factor>{<op_1><factor>}
<op_1>        ::= *|/
<factor>      ::= {<op_0>}<evaluable>{<op_2><evaluable>}
<op_2>        ::= ^
<evaluable>   ::=  {<op_0>}<número>
                 | {<op_0>}<variable>
                 | {<op_0>}(<expresión>)
                 | {<op_0>}<función>(<expresión>)
<número>      ::= <dígito>{<dígito>}[.{<dígito>}]
<función>     ::= exp|ln|sin|cos|tan|cot|sec|csc
                 |asin|acos|atan|acot|asec|acsc|sqrt
```

Todas las variables son de tipo real.

La precedencia de los operadores es la siguiente:

( ). f( )	paréntesis y funciones	mayor
+ -	signo de un dato evaluable (unario)	
^	potencia de datos evaluables	
+ -	signo de un factor (unario)	
* /	producto y cociente de factores	
+ -	signo de un término (unario)	
+ -	suma y resta de términos	menor

cuando hay ambigüedad en el orden de la expresión, las series se resuelven de izquierda a derecha.

Por ejemplo:

x = 4/-3*-2 resulta en (4/-3)*-2 = 2.666666666

y x = 4/(-3*-2) resulta en 4/6 = 0.666666666

2^3^3 = (2^3)^4 = 8^4 = 4096

8.- DISEÑO DETALLADO.

8.1.- Porqué se eligió JAVA para elaborar el sistema.

Teniendo en cuenta los requerimientos, es necesario que los alumnos que accesan la página del sistema en Internet, puedan ejecutar los métodos numéricos de una manera sencilla, sin necesidad de bajar programas a la máquina en donde están trabajando.
Se eligió JAVA por las siguientes razones:

JAVA posee una ***estructura compatible con la Web***, que libera al programador de la carga de tener que considerar soluciones particulares a problemas comunes. JAVA ***genera ejecutables extremadamente cortos*** para facilitar una transferencia mas rápida sobre líneas de comunicación potencialmente lentas. JAVA ***es independiente de la máquina***, el mismo programa puede ser ejecutado en una PC, en una Mac o incluso en una Unix (en tanto haya en la máquina un explorador que admita JAVA) [Davis,1997].

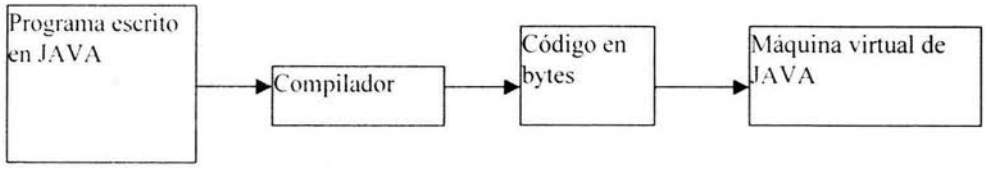
Se requieren páginas Web mas interactivas y dinámicas. JAVA es la forma mas poderosa de lograrlo, existen otras herramientas, como HTML dinámico y ActiveX, pero actualmente JAVA sigue siendo la herramienta mas poderosa y flexible [Cowell,1999].

JAVA es ***robusto,*** puesto que no permite el manejo directo de memoria, en lugar de apuntadores se manejan arreglos y cadenas verdaderas, esto hace que los programas sean seguros en su ejecución. JAVA es ***seguro***, fue pensado para ser un lenguaje de red, por lo que intrínsecamente cuenta con características de seguridad que evitan la infección por virus o trampas de programación, comunes en programas hechos en otros lenguajes [Becerril,1998].

8.2.- El lenguaje JAVA.

JAVA es un lenguaje de programación de alto nivel con el que se pueden escribir tanto programas convencionales, como para Internet.

JAVA incluye dos elementos, un "*compilador*" y un "*intérprete*". El compilador (programa traductor) produce un código en bytes que se almacena en un archivo para ser ejecutado por el intérprete JAVA denominado ***máquina virtual de JAVA.***



Los códigos de bytes de JAVA son conjuntos de instrucciones correspondientes a un lenguaje de máquina que no es específico de ningún procesador, sino de la máquina virtual de JAVA ¿ Donde se consigue esta máquina virtual? Hoy en dia casi todas las compañías

de sistemas operativos y de navegadores han implementado máquinas virtuales según las especificaciones publicadas por Sun Microsystems, propietario de JAVA, para que sean compatibles con el lenguaje JAVA.

Para las aplicaciones por Internet (denominadas applets) la máquina virtual esta incluida en el navegador [Ceballos,2002].

8.3.- Los "Applets".

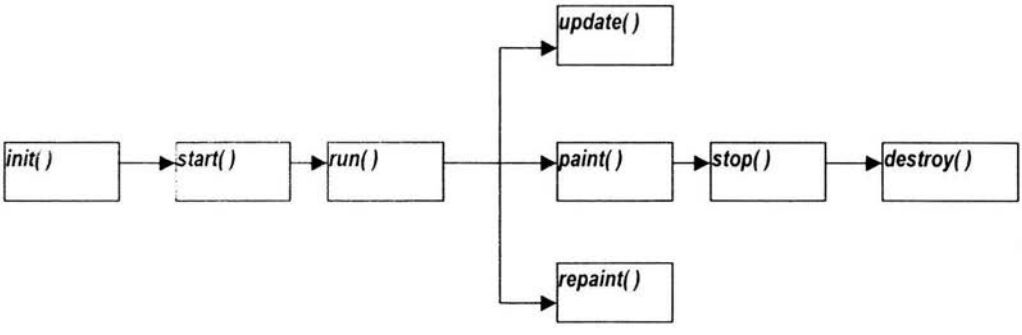
El lenguaje JAVA se usa en esta tesis para desarrollar los applets.

Un applet es un miniprograma escrito en JAVA que un navegador obtiene y transporta a través de la red y lo ejecuta en tiempo real; además puede interactuar con el usuario. Para ejecutarse, un applet requiere un archivo HTML (Lenguaje para marcación de hipertexto o "HiperText Markup Language") [Becerril, 1998].

JAVA puede generar dos tipos de programas. Un programa por sí mismo, independiente de la Web se denomina ***aplicación*** de JAVA, pero JAVA empieza a destacar cuando se usa para generar programas diseñados para ejecutarse como parte de la página de la Web. Estos programas se llaman ***applets***. [Davis, 1997].

8.4.- El ciclo de trabajo de un Applet.

A continuación se presenta un diagrama con el ciclo de trabajo de un applet



[Becerril, 1998]

El método ***init( )*** se activa desde el navegador después de cargar el applet. Este es el lugar perfecto para hacer todos los preparativos: cargar sonidos e imágenes, inicializar los objetos necesarios, insertar controles, etc [Backer, 1996].

Se llama al método ***start( )*** después de ***init( )*** y como punto de partida después de que se haya detenido un applet. Mientras que a ***init( )*** se le llama una vez (la primera vez que se carga el applet), a ***start( )*** se le llama cada vez que se visualiza en pantalla un documento HTML con el applet. Por lo tanto, si un usuario abandona una página de red y vuelve, el applet comienza la ejecución en ***start( )***.

Paint( ) se llama cada vez que se daña un applet. El AWT supervisa el sistema de ventanas y detecta problemas, por ejemplo, que otras ventanas cubran al applet y llama a ***paint( )*** para que repare todos los daños cuando el applet quede al descubierto.

Update( ) significa actualizar.

Stop( ) se llama cuando un visualizador de red abandona el documento HTML que contiene al applet.

Destroy( ) se llama cuando el entorno determinó que es necesario eliminar completamente al applet, se liberan todos los recursos ocupados por el applet.

Repaint( ) se llama para obligar a que se vuelva a pintar el applet, a su vez ***repaint( )*** llama a ***update( )*** [Naughton, 1996].

8.5.- El lenguaje de marcación de hipertexto (HTML).

El menú para tener acceso al sistema elaborado en esta tesis se encuentra en una página "Web" escrita en lenguaje HTML en internet.

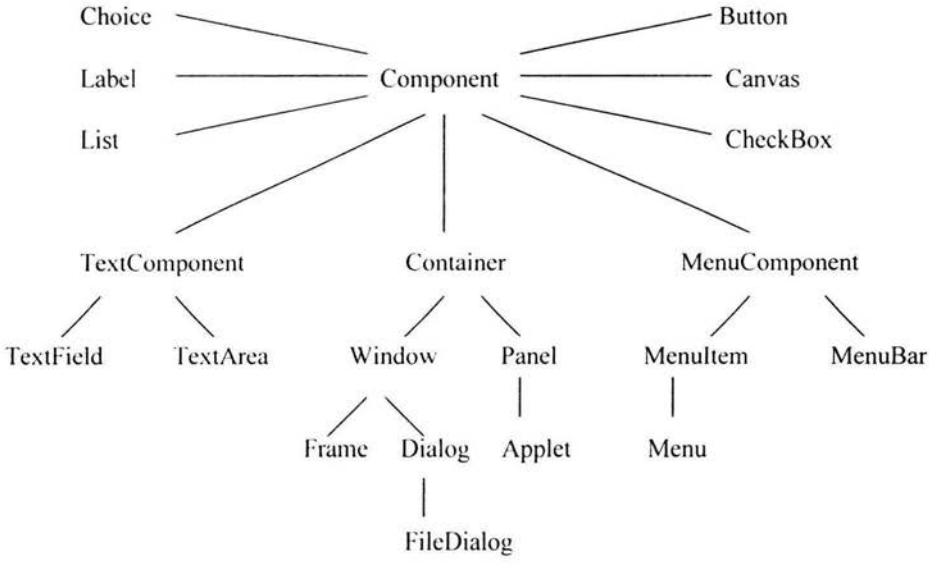
World Wide Web es un concepto que se utiliza para hacer referencia a documentos, sus vínculos y los navegadores usados, para verlos e interactuar con ellos.

Una de las características mas atrayentes de World Wide Web es que para tener una página Web bastan una computadora, un editor de texto y un navegador. Los navegadores Web están diseñados para reconocer un lenguaje basado puramente en texto, el *lenguaje de marcación de hipertexto (HTML)*, los elementos de este lenguaje se denominan etiquetas, las cuales consisten en palabras clave y otra información entre paréntesis angulares < >, todo lo que no es una etiqueta se considera texto sencillo y el navegador lo despliega como tal [Decker&Hirshfield].

8.6.-El equipo de herramientas de ventana abstractas (AWT).

Una de las fortalezas de JAVA es que es independiente de la plataforma. El equipo de herramientas de ventana abstractas (en inglés: Abstract Windowing Toolkit) proporciona un conjunto de clases independientes de la plataforma que manejan las operaciones gráficas...el paquete AWT incluye clases para dibujar líneas y figuras geométricas, así como para crear botones, menús, cuadros de diálogo y demás elementos [Chan, Griffith & Iasi,1998].

El AWT proporciona un conjunto de métodos que se utilizan para realizar operaciones gráficas, a continuación se muestra la jerarquía de clases del AWT, la cual da una buena idea de la gran variedad de características gráficas disponibles [Daconta,1996].



El uso del AWT consta de 3 partes:

1.- *Componentes*.- Cualquier elemento gráfico que pueda ser puesto en la interfaz del usuario, incluyendo botones, listas deslizables, menús pop-up, Checkboxes y campos de texto.

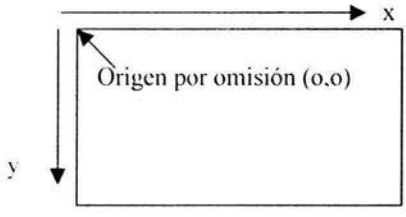
2.- *Contenedores*.- Un componente que puede contener a otros componentes, como paneles, cajas de diálogo y ventanas independientes.

3.- *Administradores de Diseño.*- Son los que controlan la ubicación de los componentes añadidos a un contenedor [Lemay&Cadenhead, 1999].

Los *componentes* se agrupan dentro de los *contenedores* , pero además hay que tomar en cuenta que los contenedores son a su vez *componentes* y pueden volver a ser agrupados en otros *contenedores* [Joyanes&Fernández, 2001].

8.7.- El sistema de coordenadas de JAVA.

Muchos de los métodos de las clases AWT utilizan un sistema de coordenadas "X-Y" para indicar la posición del elemento a dibujar. El origen (0,0) esta predeterminado en la esquina superior izquierda de la pantalla, con la *x* positiva extendiéndose a la derecha y la *y* positiva extendiéndose hacia abajo, como se muestra en la siguiente figura:



8.8.- Las "PÁGINAS DE CAPTURA DE DATOS".

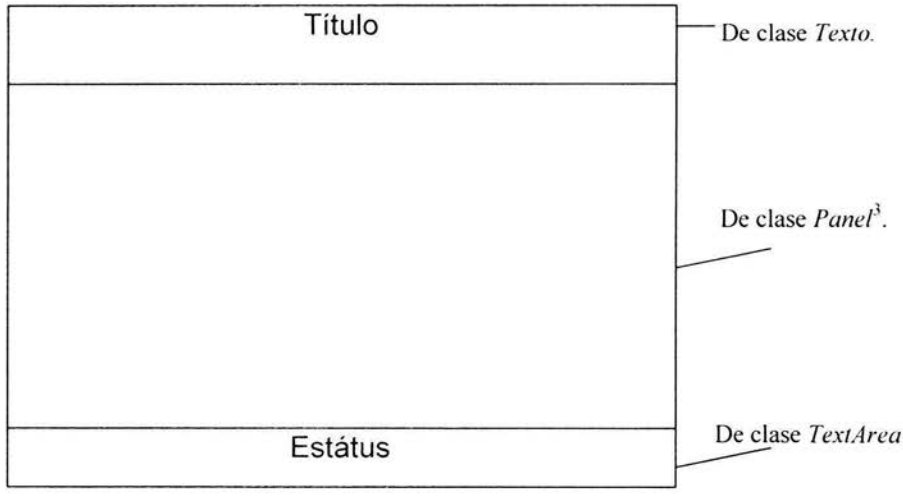
El diseño de estas páginas, se basó en los métodos descritos por Denis Kafura en el capítulo "Building user Interfaces in JAVA" [Kafura, 2000].

Las partes fundamentales de un sistema IDE (en Español, Ambiente de Desarrollo Integrado) son:

1.- Un área de control (menús).
2.- Un área de trabajo.
3.- Un área de estatus.

Apegándose a la estructura de un IDE, las páginas de captura de datos del sistema desarrollado tienen 3 partes:

1.- El ***título del método***.- El cuál es un parámetro que se pasa a través de la página HTML.
2.- La ***pantalla***.- La cuál contiene los campos de captura y botones de control.
3.- Un ***área de estatus*** en la parte inferior.- En donde se envían mensajes al usuario para dar las instrucciones o para indicar la operación que esta realizando el sistema, esta es un "*Area de Texo*" (*TextArea*)[2].



La clase ***Texto*** es la clase derivada de ***Canvas***[4]. En su constructor genera un arreglo de texto cuyo tamaño es la longitud del título y especifica el tamaño y tipo de letra. Con la función *paint* de esta clase, se dibujan las letras de manera aleatoria y a colores.

---

[2] Las *Áreas de Texto* proporcionan un área para manipular múltiples líneas de texto [Deitel&Deitel, 1998].
[3] Un *Panel* es un contenedor al cual se le puede agregar componentes, incluidos otros paneles. El constructor *Panel* no recibe argumentos y no esta sobrecargado. [Deitel&Deitel, 1998.]. A diferencia de *Frame* y *Dialog*, *Panel* es un contenedor que no crea una ventana separada [Flanagan, 1999].

La ***Pantalla*** es una función que devuelve un ***Panel***. El ***Panel*** contiene los diferentes campos de captura (que dependen del método) y dos botones de control, que son:

Los objetos del ***panel*** están montados sobre un ***GridBagLayout***[5].

La propagación de un evento desde un objeto *fuente* hasta un objeto *receptor* involucra la llamada a un método en el objeto receptor y el paso de una instancia de una subclase de eventos (un objeto) que define el tipo de evento generado [Froufe, 2000].
En este caso, el "manejador de eventos" *ActionListener* se encarga de recibir el clic del mouse en el botón y el método en donde se da tratamiento al evento es en *actionPerformed*.

El *área de campos de captura* puede contener la introducción de una función, en este caso se utiliza un ***parser*** para procesar la función introducida.

Para capturar sistemas de ecuaciones lineales de la forma:
$Ax = b$ (donde A es una matriz nxn, x y b son matrices de nx1)

se utilizaron paneles anidados, es decir, en una pantalla se presentan 3 paneles, el primero consta de los campos para introducir los coeficientes de la matriz A, el segundo panel contiene los campos del vector "b" y el tercer panel los del vector inicial "x", cada uno de estos paneles es de tamaño variable, ya que dependen del número de ecuaciones del sistema. Actualmente el sistema acepta hasta 12 ecuaciones lineales con 12 incógnitas, para sistemas mas grandes los campos de A, x y b no caben en la pantalla.

El botón CALCULA sirve para que se ejecute el método seleccionado con los datos introducidos en la pantalla de captura y el botón LIMPIA sirve para borrar todos los campos del area de campos de captura.

---

[4] Un *Canvas* o *Lienzo* es un componente en el que pueden dibujarse gráficas y que puede recibir eventos de raton [Deitel&Deitel, 1998]. Aunque un applet puede dibujar directamente sobre un área que aparece en pantalla, por lo general es mejor definir un área de dibujo independiente mediante un objeto *Canvas* (lienzo) si se dibuja directamente sobre el applet, podria dibujarse sobre otros componentes del AWT o estos podrian encimarse sobre el dibujo [Chan, Griffith&Iasi, 1998].

[5] Un *GridBagLayout* es el *layout manager* mas poderoso y complicado en el paquete JAVA.awt. Divide el contenedor en una rejilla de filas y columnas (que no necesariamente tienen la misma altura y ancho) y coloca los componentes en esta rejilla ademas de ajustar el tamaño de las celdas de la rejilla, según sea adecuado para asegurar que los componentes no se traslapen [Flanagan, 1999]. La descripción completa del manejo de restricciones para *GridBaglayout* puede consultarse en [Wehling, Bharat y otros, 1998].

8.9.- CAPTURA DE MATRICES.

Para capturar un vector o una matriz, fue necesario elaborar un *Vector*[6] de campos de texto (TextFields), para el caso de la matriz bidimensional, se utilizó un vector de vectores.

Si desea información detallada de las operaciones con matrices puede consultar: [Duane,1999] y [Duane,2000].

8.10.- FORMATEO DE LOS RESULTADOS.

Se utilizó **DecimalFormat** para fijar la longitud de la parte entera y de la parte fraccionaria de los resultados parciales cuando se despliegan resultados en cada iteración. "DecimalFormat" tiene muchas opciones para dar formato a los números, la información detallada se puede consultar en [Ceballos,2000] .

8.11.- CAMPOS DE TEXTO NO EDITABLES.

En algunos métodos, como el de interpolación de Newton o los métodos de integración, se presentan unas tablas en donde los valores de X están precalculados y solo se necesita la captura de los valores de Y. En estos casos, fue necesario dar a los campos de texto con los valores de X la característica de "*No editable*" para evitar que el usuario altere su valor.

Se cambió el color de fondo en estos campos para hacer mas evidente que no son de captura.

---

[6] Un *Vector* es un arreglo dinámico que puede crecer y encogerse para atender las necesidades de almacenamiento variables de un programa [Deitel&Deitel,1998, pag. 448].

8.12.- Algoritmos de los métodos del sistema

A continuación se presenta cada uno de los métodos numéricos que componen al sistema. Se incluye una breve descripción del método y su algoritmo en pseudocódigo. Este pseudocódigo se transformó a lenguaje JAVA para implementar los applets.

8.12.1.- Algoritmo del Método de Newton Rapshon.

Este método consiste en obtener una mejor aproximación de la raíz tomando un valor inicial x_0 y utilizando la siguiente fórmula con la función y la derivada de la función:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

El método termina cuando se cumple con alguna condición de paro, en este programa la condición es la tolerancia :

$$|x_{i+1} - x_i| \leq \varepsilon$$

Datos: F(x), F'(x), Valor inicial de x, tolerancia.

```
NewtonRaphson
 begin

  iteración := 1
  x_anterior := valor_inicial
  do
    x_nueva := x_anterior – F(x_anterior) / F'(x_anterior)
    error := | x_nueva – x_anterior |
    x_anterior := x_nueva
    iteración := iteración + 1
  while((error>tolerancia) OR ( iteración = 100))

end NewtonRaphson
```

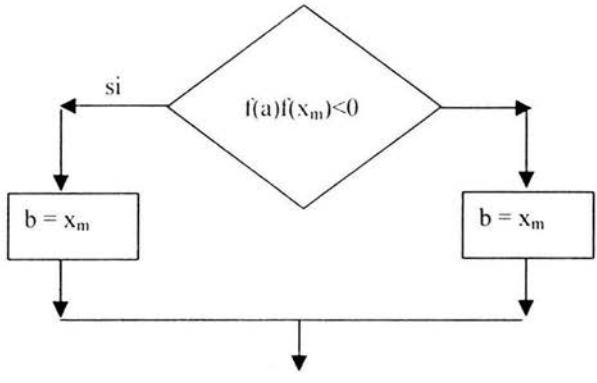
8.12.2.- Algoritmo del Método de Bisección.

Este método consiste en obtener una mejor aproximación de la raíz a partir de un intervalo inicial (a,b) en el cual hay un cambio de signo en la función, es decir: f(a)f(b)<0.

Se obtiene el punto medio:

$$x_m = \frac{a+b}{2}$$

x_m es la nueva aproximación a la raíz, y se vuelve a tomar un intervalo, pero ahora mas pequeño, considerando que siga existiendo un cambio de signo en la función, es decir, el nuevo intervalo queda determinado por:



El método termina cuando se cumple con alguna condición de paro, en este programa la condición es la tolerancia :

$$|x_{i+1} - x_i| \leq \varepsilon$$

Datos: F(x), (x_inferior, x-superior), tolerancia.

```
Bisección
 begin
  if ( F(x_inferior) * F(x_superior) > 0)
    enviar una pantalla de error con el mensaje:
      "Debe haber cambio de signo en el intervalo"
    terminar
  endif

  iteración := 1
  x_media := (x_superior + x_inferior)/2

  do
    if ( F(x_inferior) * F(x_media) < 0)
       x_superior := x_media
    else
        if ( F(x_inferior) * F(x_media) = 0)
          x_nueva := x_media
          terminar
        else
          x_inferior := x_media
        endif
    endif

    x_nueva:= (x_superior+ x_inferior) / 2
    error := | x_nueva – x_media |
    x_media := x_nueva
    iteración := iteración + 1
  while((error>tolerancia) OR ( iteración = 100))

end Bisección
```

8.12.3.- Algoritmo del Método de Jacobi.

Muchos problemas relacionados con el campo de la ingeniería se pueden expresar en términos de sistemas de ecuaciones algebraicas lineales.

Cuando se resuelven numéricamente ecuaciones diferenciales pueden surgir sistemas lineales con 20,000 variables. Los equipos de cómputo disponibles en la actualidad podrían requerir incluso días para resolver estos sistemas por métodos directos (como eliminación o factorización).

El *método de Jácobi* es un método iterativo con el cual se resuelve el sistema lineal

$$A\mathbf{x} = \mathbf{b}$$

Comienza con una aproximación inicial $\mathbf{x}^{(0)}$ a la solución $\mathbf{x}$ y genera una sucesión de vectores $\mathbf{x}^{(k)}$ que convergen a la solución $\mathbf{x}$.

Un sistema de ecuaciones algebraicas lineales es un conjunto de ecuaciones de la forma:

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n &= b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \ldots + a_{3n}x_n &= b_3 \\
\quad :: \qquad\qquad :: \qquad\quad :: & \\
a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{nn}x_n &= b_n
\end{aligned}$$

O bien en su forma matricial:

$$\begin{matrix}
a_{11} & a_{12} & a_{13} & \ldots & a_{1n} & x_1 & b_1 \\
a_{21} & a_{22} & a_{23} & \ldots & a_{2n} & x_2 & b_2 \\
a_{31} & a_{32} & a_{33} & \ldots & a_{3n} & x_3 & = b_3 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
a_{n1} & a_{n2} & a_{n3} & \ldots & a_{nn} & x_4 & b_n
\end{matrix}$$

Que a su vez se puede expresar como:

$$A\mathbf{x} = \mathbf{b}$$

Donde "A" es la matriz de coeficientes, $\mathbf{x}$ es el vector de incógnitas y $\mathbf{b}$ el vector de términos independientes.

La solución del sistema de ecuaciones es un conjunto de n valores $x_1, x_2, x_3, \ldots, x_n$ que satisfacen simultáneamente todas las ecuaciones.

En la solución de estos problemas pueden presentarse 3 casos:

1.- *Solución única* ⟶ Sistema compatible determinado.
2.- Mas de una *solución* (numero infinito de soluciones) ⟶ Sistema compatible e indeterminado.
3.- Sin *solución* ⟶ Sistema incompatible.

Ilustrando el método de Jácobi con un sistema de ecuaciones de 3x3, si el vector:

$$x^{(k)} = \begin{pmatrix} x_1^k \\ x_2^k \\ x_3^k \end{pmatrix}$$

Es el vector aproximación a la solución **x** después de *k* iteraciones, entonces se tiene que para la siguiente aproximación:

$$x_i^{k+1} = \begin{pmatrix} x_1^{k+1} \\ x_2^{k+1} \\ x_3^{k+1} \end{pmatrix} = \begin{pmatrix} \frac{1}{a_{11}}(b_1 - a_{12}x_2^k - a_{13}x_3^k) \\ \frac{1}{a_{22}}(b_2 - a_{21}x_1^k - a_{23}x_3^k) \\ \frac{1}{a_{33}}(b_3 - a_{31}x_1^k - a_{32}x_2^k) \end{pmatrix}$$

Para un sistema de *n* ecuaciones con *n* incógnitas se tiene la siguiente fórmula (usando una notación mas compacta):

$$x_i^{k+1} = -\frac{1}{a_{ii}}(-b_i + \sum_{\substack{j=1 \\ j \neq i}} a_{ij}x_j^k)$$

Para $1 \leq i \leq n$

Tanto en el método de Gauss-Seidel como en el de Jácobi, el valor que se le dé al vector inicial carece de importancia, ya que el método convergirá a la solución rápidamente no obstante que el vector inicial tenga valores muy lejanos a la solución. Por esto se acostumbra usar el vector **0** como vector inicial.

Datos: Número de incógnitas: "n", matriz $A_{n,n}$, Vector X_n , Vector B_n , tolerancia.

```
Sumatoria( i)
  suma:= 0
  for j:=0 to n-1
     if(j<>i)
       suma := suma + (A[i,j] / A[i,i]) * X[j]
     endif
   endfor
   return ( suma )
end Sumatoria

Norma( )
  suma:= 0
  for i:=0 to n-1
     suma := suma + (X_nueva[i] – X[i])^2
   endfor

   return ( sqrt ( suma ) )
end Norma

Jacobi
 begin

  iteración := 1
  do

    for i := 0 to n-1
      X_nueva[i] := B[i] / A[i,i] – Sumatoria (i)
    endfor

    error := Norma( )

    for i := 0 to n-1
      X_[i] := X_nueva[i]
    endfor

    iteración := iteración + 1
  while((error>tolerancia) OR ( iteración <= 50))
 end Jacobi
```

8.12.4.- Algoritmo del Método de Gauss-Seidel.

Ilustrando el método de Gauss-Seidel con un sistema de ecuaciones de 3x3, si el vector:

$$x^{(k)} = \begin{pmatrix} x_1^k \\ x_2^k \\ x_3^k \end{pmatrix}$$

Es el vector aproximación a la solución **x** después de k iteraciones, entonces se tiene que para la siguiente aproximación:

$$x_i^{k+1} = \begin{pmatrix} x_1^{k+1} \\ x_2^{k+1} \\ x_3^{k+1} \end{pmatrix} = \begin{pmatrix} \frac{1}{a_{11}}(b_1 - a_{12}x_2^k - a_{13}x_3^k) \\ \frac{1}{a_{22}}(b_2 - a_{21}x_1^{k+1} - a_{23}x_3^k) \\ \frac{1}{a_{33}}(b_3 - a_{31}x_1^{k+1} - a_{32}x_2^{k+1}) \end{pmatrix}$$

Para un sistema de n ecuaciones con n incógnitas se tiene la siguiente fórmula (usando una notación mas compacta):

$$x_i^{k+1} = -\frac{1}{a_{ii}}(-b_i + \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} + \sum_{j=i+1}^{n} a_{ij}x_j^k) \qquad \text{Para } 1 \leq i \leq n$$

Datos: Número de incógnitas: "n", matriz $A_{n,n}$, Vector X_n, Vector B_n, tolerancia.

```
Sumatoria1( i)
  suma:= 0
  for j:= i+1 to  j < n-1
     if(j<>i)
       suma := suma + (A[i,j] / A[i,i]) * X[j]
     endif
   endfor
   return ( suma )
end Sumatoria1
Sumatoria2( i)
  suma:= 0
  for j:= 0 to  i-1
     if(j<>i)
       suma := suma + (A[i,j] / A[i,i]) * X-nueva[j]
     endif
   endfor
   return ( suma )
end Sumatoria2
Norma( )
  suma:= 0
  for i:=0 to n-1
     suma := suma + (X_nueva[i] – X[i])^2
   endfor
   return ( sqrt ( suma ) )
end Norma

Seidel
 begin
  iteración := 1
  do
    for i := 0 to n-1
      X_nueva[i] := B[i] / A[i,i] – Sumatoria1(i) - Sumatoria2(i)
    endfor
    error := Norma( )
    for i := 0 to n-1
      X_[i] := X_nueva[i]
    endfor
    iteración := iteración + 1
  while((error>tolerancia) OR ( iteración <= 50))
 end Seidel
```

8.12.5.- Algoritmo del Método de Interpolación de Newton.

En algunas ocasiones, no se tiene una función continua, sino valores de la función específicos y(x) para una x dada. A estas funciones se les conoce como *funciones tabulares*, y son de la siguiente forma:

x	y(x)
x_0	y_0
x_1	y_1
x_2	y_2
::	
x_n	y_n

En la práctica tenemos como ejemplo los resultados de experimentos en un laboratorio, o el censo de la población cada 5 años.

La *interpolación* requiere el cálculo de los valores de una función y(x) para argumentos entre $x_0, x_1, x_2, \ldots, x_n$ en los cuales se conocen los valores $y_0, y_1, y_2, \ldots, y_n$. en otras palabras, interpolar es recuperar los valores de una función en puntos intermedios dada una tabla de valores de esta función.

Por ejemplo, a veces es imposible o muy costoso hacer experimentos de laboratorio para valores intermedios de x. También sería muy costoso hacer un censo de la población cada año, sin embargo, si tenemos el tamaño de la población en 1980, 1985 y 1990, podemos interpolar para obtener el tamaño de la población en 1983.

Para poder realizar una ***interpolación de Newton*** es necesario que los valores de las x dadas en la función tabular tengan un ***espaciamiento constante***, es decir, la diferencia entre una x_i y una x_{i+1} debe ser siempre el mismo, al espaciamiento se le denota con la letra ***h***.

El primer paso para la interpolación de Newton es obtener la ***tabla de diferencias***. El formato estándar de una tabla de diferencias finitas es el siguiente:

x_i	y_i	Δy_i	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^4 y_i$
x_0	y_0				
		Δy_0			
x_1	y_1		$\Delta^2 y_0$		
		Δy_1		$\Delta^3 y_0$	
x_2	y_2		$\Delta^2 y_1$		$\Delta^4 y_0$
		Δy_2		$\Delta^3 y_1$	
x_3	y_3		$\Delta^2 y_2$		
		Δy_3			
x_4	y_4				

De esta tabla se puede observar que las *k-ésimas* diferencias de una función tabular, están dadas por:

$$\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i$$

Por otra parte, una vez teniendo una $x_{inicial}$ en la tabla y una $x_{Interpolar}$, podemos obtener la $y_{interpolada}$ utilizando el *polinomio de Newton* que es el siguiente:

$$y_{interpolada} = y_0 + \binom{k}{1}(\Delta y_0) + \binom{k}{2}(\Delta^2 y_0) + \binom{k}{3}(\Delta^3 y_0) + \ldots + \binom{k}{l}(\Delta^l y_0)$$

En este polinomio y_0 se refiere a $y(x_{inicial})$. Mientras que k esta dada por la siguiente fórmula:

$$k = \frac{x_{Interpolar} - x_{inicial}}{h}$$

$\binom{k}{l}$ son las *combinaciones* de k elementos combinados de l en l. Como en este caso k es comúnmente un número fraccionario, entonces se usa la fórmula:

$$\binom{k}{l} = \frac{k(k-1)(k-2)\ldots(k-l+1)}{l!}.$$

Normalmente usando un polinomio de Newton de 3er grado, es decir, hasta las terceras diferencias, se obtiene una interpolación bastante aceptable, mientras mas diferencias se usen, mejor será el resultado. Por esto la $x_{inicial}$ debe escogerse lo mas cercano a la $x_{Interpolar}$ pero sin sacrificar la cantidad de diferencias.

Datos: Numero de datos de la tabla: numDatos, x_inicial, x_a_interpolar, Y, espaciamiento.

```
PotenciaFactorial ( k, m)
  result := 1
  for i:=0 to m-1
    result := result * (k-i)
  endfor
  return (result)
end PotenciaFactorial

Factorial ( m)
  fact := 1
  for i:=1 to m
    fact := fact * i
  endfor
  return (fact)
end Factorial

Combinacion( k, m)
  return ( PotenciaFactorial(k,m) / Factorial(m) )
end PotenciaFactorial

Newton
 Begin
    // Generar el vector X
    X[0] := x_inicial
    for i:= 1 to (numDatos-1)
       X[i] := x_inicial + i*espaciamiento
    endfor

    // Encontrar el indice de la x_inicial
    for i := 0 to (numDatos –1)
      if X[i] = x_inicial
        indice := i
      endif
    endfor
```

```
  // Inicializar la tabla
  for i := 0 to (numDatos –1)
    for j := 0 to (numDatos –1)
      Tabla[i,j]:= 0
    endfor
  endfor
  // Llenar la tabla con las columnas X, Y y las diferencias
  for i := 0 to numDatos
     Tabla[i,0] := X[i]
     Tabla[i,1] := Y[i]
  endfor

  for i := 2 to (numDatos –1)
    for j := 0 to (numDatos –i + 1)
      Tabla[j,i]:= Tabla[j+1,i-1] – Tabla[j,i-1]
    endfor
  endfor

  //Obtener el valor de la Y interpolada usando el polinomio de Newton
  k:= (x_a_interpolar – x_inicial) / espaciamiento

 y_interpolada := 0

 for j:= 0 to  (numDatos-1)
    y_interpolada := y_interpolada + Combinacion(k,j) * Tabla[indice, j+1]
  endfor


end Newton
```

8.12.6.- Algoritmo del Método de Interpolación de Lagrange.

Para poder realizar una *interpolación de Newton* es necesario que los valores de las x dadas en la función tabular tengan un *espaciamiento constante* mientras que una ***interpolación de Lagrange*** se puede llevar a cabo sin importar si el espaciamiento es constante o variable.

La *interpolación de polinomios de Lagrange* es una reformulación del polinomio de Newton que evita el cálculo de la tabla de diferencias, el polinomio de Lagrange se expresa como:

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

Donde:

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

Π es el símbolo de "multiplicatoria" y significa *el producto de*.
Por ejemplo, el polinomio de Lagrange de primer grado es:

$$f_1(x) = \frac{x - x_1}{x_0 - x_1} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1)$$

Mientras que el polinomio de Lagrange de segundo grado es:

$$f_1(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f(x_0) + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f(x_1) + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f(x_2)$$

En este caso $f_n(x)$ es la $y_{interpolada}$ y la x es la $x_{Ainterpolar}$. Mientras mas datos se tengan en la tabla, se podrá usar un polinomio de mayor grado, lo que dará mejores resultados.

Datos: Numero de datos de la tabla: numDatos, x_a_interpolar, X, Y.

```
Producto ( i )
  prod1:= 1
  prod2:= 1
  for k:= 0 to (numDatos-1)
    if(k<>i)
       prod1:= (x_a_interpolar – X[k]) * prod1
       prod2:= (X[i] – X[k]) * prod2
    endif
 endfor

 return ( prod1/prod2)

end Producto



Lagrange
   y_interpolada := 0

   for i:= 0 to  (numDatos-1)
      y_interpolada := y_interpolada + producto(i) * Y[i]
   endfor


 end Lagrange
```

8.12.7.- Algoritmo del Método de Integración Trapezoidal.

Los ingenieros encuentran con frecuencia el problema de integrar funciones que están definidas en forma tabular o en forma gráfica y no como funciones explícitas, se pueden utilizar métodos gráficos, pero los métodos numéricos son mucho mas precisos.

La **integración numérica** consiste en encontrar una buena aproximación al área bajo la curva que representa una función *f(x)*, que ha sido determinada a partir de datos experimentales o a partir de una expresión matemática.

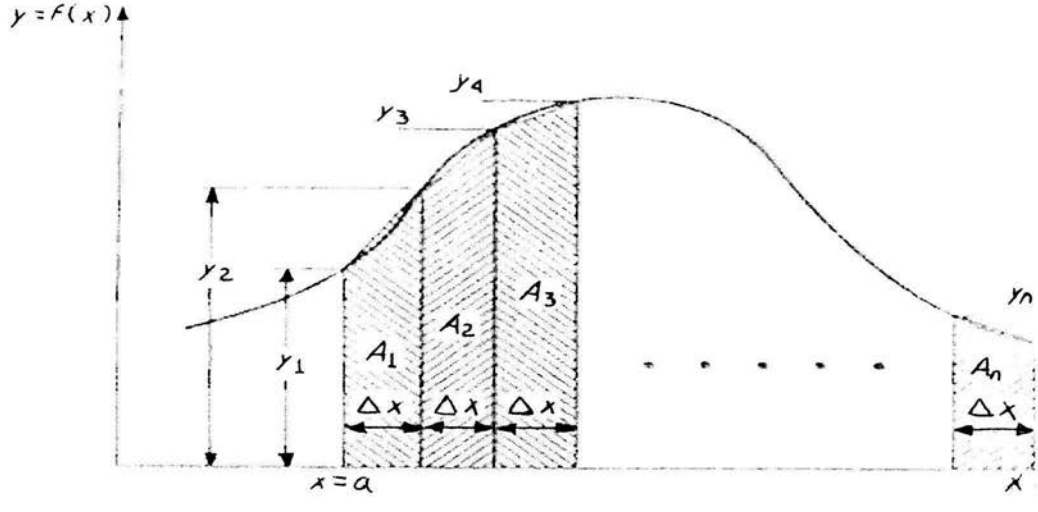
Las *formulas de cuadratura de Newton-Cotes* son los procedimientos mas comunes de integración numérica, se basan en la estrategia de reemplazar una función complicada o datos tabulados con una función aproximada que sea fácil de integrar, estas son:

- La regla de integración Trapezoidal.
- La regla de Simpson.

Estas reglas están diseñadas para casos en los que los datos a integrarse están *espaciados de manera uniforme*.

A continuación se describe la regla trapezoidal para la "*integración cerrada*" es decir, para cuando los valores de la función en los extremos de los límites de integración son conocidos

Con el *método de Integración Trapezoidal* se obtiene una aproximación del área bajo la curva de una función dividiéndola en *n* fajas de ancho Δx y aproximando el área de cada faja mediante un trapecio, como se indica en la siguiente figura:



Dada una función tabular con espaciamientos constantes, de la forma:

x	y(x)
x_0	y_0
x_1	y_1
x_2	y_2
::	::
x_n	y_n

La fórmula de integración Trapezoidal es la siguiente:

$$\int_{x_0}^{x_n} f(x_k)dx_k = \frac{h}{2}(y_0 + y_n + 2\sum_{i=1}^{n-1} y_i) + e_r$$

Datos: Número de datos de la tabla: numDatos, espaciamiento, x_inicial, Y.

```
Sumatoria( )
  suma:= 0
  for i:=1 to  i < numDatos-2
    suma := suma + Y[i]
  endfor
   return ( suma )
end Sumatoria


Trapezoidal
 begin

    // Generar el vector X
    X[0] := x_inicial
    for i:= 1 to (numDatos-1)
       X[i] := x_inicial + i*espaciamiento
    endfor

    // Obtener el valor de la integral usando la fórmula trapezoidal
    integral := (espaciamiento/2)*(Y[0] + Y[numDatos-1] + 2*Sumatoria() )

end Trapeziodal
```

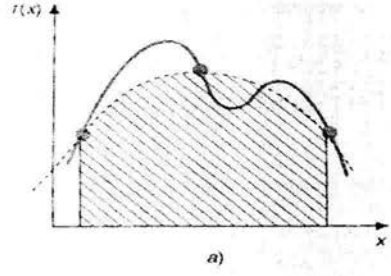
8.12.8.- Algoritmo del Método de Integración de Simpson 1/3.

A continuación se describe la regla de integración de Simpson 1/3 para la "*integración cerrada*" es decir, para cuando los valores de la función en los extremos de los límites de integración son conocidos.

Además de aplicar la regla trapezoidal con segmentación mas fina, otra forma de obtener una estimación mas exacta de una integral es con el uso de polinomios de orden superior para conectar los puntos (en lugar de utilizar líneas para conectarlos).

Las *reglas de Simpson* son las fórmulas que resultan al tomar las integrales bajo los polinomios que conectan a los puntos.

El *método de Integración Simpson 1/3* consiste en tomar el área bajo una parábola que conecta tres puntos, como se muestra en la siguiente gráfica:



Dada una función tabular con espaciamientos constantes, de la forma:

x	y(x)
x_0	y_0
x_1	y_1
x_2	y_2
::	::
x_n	y_n

La fórmula de integración de Simpson 1/3 es la siguiente:

$$\int_{x_0}^{x_n} f(x_k)dx_k = \frac{h}{3}(y_0 + y_n + 4\sum_{indice-impar}^{ordenadas} + 2\sum_{indice-par}^{ordenadas}) + e_r$$

La fórmula de integración de Simpson 1/3 sólo es aplicable cuando el "número de datos de la tabla " sea ***par***.

Datos: Número de datos de la tabla: numDatos, espaciamiento, x_inicial, Y.

```
Sumaimpar( )
  suma:= 0
  for i:=1 to  i < numDatos-2
    suma := suma + Y[i]
    i:= i + 2
  endfor
   return ( suma )
end Sumaimpar

Sumapar( )
  suma:= 0
  for i:=2 to  i < numDatos-3
    suma := suma + Y[i]
    i:= i + 2
  endfor
   return ( suma )
end Sumapar

Simpson13
 begin
    // Verificar que el número de datos sea impar
    if ( numDatos MOD 2 = 0 )
      lanzar una pantalla con el mensaje:
       "El número de datos de la tabla debe ser impar"
      terminar
    endif

    // Generar el vector X
    X[0] := x_inicial
    for i:= 1 to (numDatos-1)
       X[i] := x_inicial + i*espaciamiento
    endfor

    // Obtener el valor de la integral usando la fórmula de Simpson 1/3
    integral:=(espaciamiento/3)*(Y[0] + Y[numDatos-1] + 4*Sumaimpar() + 2*Sumapar() )

end Simpson13
```

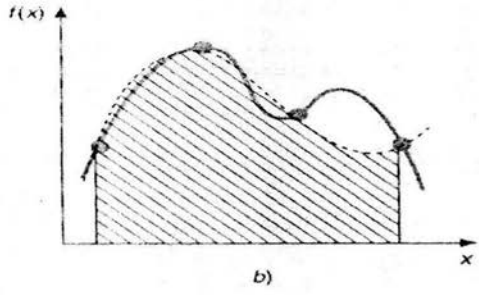
8.12.9.- Algoritmo del Método de Integración de Simpson 3/8.

A continuación se describe la regla de integración de Simpson 3/8 para la "*integración cerrada*" es decir, para cuando los valores de la función en los extremos de los límites de integración son conocidos.

Además de aplicar la regla trapezoidal con segmentación mas fina, otra forma de obtener una estimación mas exacta de una integral es con el uso de polinomios de orden superior para conectar los puntos (en lugar de utilizar líneas para conectarlos).

Las *reglas de Simpson* son las fórmulas que resultan al tomar las integrales bajo los polinomios que conectan a los puntos.

El *método de Integración Simpson 3/8* consiste en tomar el área bajo una ecuación cúbica que conecta cuatro puntos , como se muestra en la siguiente gráfica:



Dada una función tabular con espaciamientos constantes, de la forma:

x	y(x)
x_0	y_0
x_1	y_1
x_2	y_2
::	::
x_n	y_n

La fórmula de integración de Simpson 3/8 es la siguiente:

$$\int_{x_0}^{x_n} f(x_k)dx_k = \frac{3h}{8}(y_0 + y_n + 2\sum_{multiplosde3}^{ordenadas} + 3\sum_{ordenadas}^{resto}) + e_r$$

La fórmula de integración de Simpson 3/8 sólo es aplicable cuando el "número de datos de la tabla menos 1" sea ***múltiplo de 3***.

Datos: Número de datos de la tabla: numDatos, espaciamiento, x_inicial, Y.

```
Sumamultiplos3( )
  suma:= 0
  i:=3
  do
    suma := suma + Y[i]
    i:= i + 3
  while((i<numDatos-1)AND(i <> numDatos-1))
    return ( suma )
end Sumamultiplos3

Sumarestoordenadas( )
  suma:= 0
  for i:=1 to  i < (numDatos-2)
   if ( i MOD 3 <> 0)
     suma := suma + Y[i]
   endif
  endfor
   return ( suma )
end Sumarestoordenadas

Simpson38
 begin
   // Verificar que el número de datos – 1  sea múltiplo de 3
   if ( (numDatos –1)MOD 3 <> 0 )
     lanzar una pantalla con el mensaje:
      "El número de datos de la tabla – 1  debe ser múltiplo de 3"
     terminar
   endif

   // Generar el vector X
   X[0] := x_inicial
   for i:= 1 to (numDatos-1)
      X[i] := x_inicial + i*espaciamiento
   endfor

   // Obtener el valor de la integral usando la fórmula de Simpson 3/8
   integral:=(3*espaciamiento/8)*(Y[0] + Y[numDatos-1] + 2*Sumamultiplos3() +
                                  3*Sumarestoordenadas() )

end Simpson38
```

8.12.10.- Algoritmo del Método de EULER para resolver ecuaciones diferenciales de primer orden.

Los fenómenos que estudia la ingeniería se pueden representar mediante modelos matemáticos, éstos en algunos casos se reducen a una ecuación diferencial, cuya solución es una función que representa el comportamiento del fenómeno.

En la práctica la gran mayoría de las ecuaciones diferenciales no pueden resolverse utilizando "técnicas analíticas" y se recurre a los "métodos numéricos", que permiten la utilización de una computadora para resolver el problema.

Las ecuaciones diferenciales se dividen en dos grupos:

1.- Ecuaciones Diferenciales Ordinarias (EDO) .- En donde aparece sólo una variable independiente (que se denota con x).

2.- Ecuaciones Diferenciales Parciales (EDP) .- En las que aparece mas de una variable independiente.

Las EDO se clasifican y se estudian según su *orden* el cual se define como el entero igual al número máximo de veces que se deriva la variable dependiente de la ecuación.

La solución de una ecuación diferencial es una función que no contiene derivadas o integrales de funciones y que al derivarla coincide con la ecuación diferencial. Como al momento de integrar, aparece una constante, entonces la solución a una ecuación diferencial resulta ser una "familia de curvas" (una curva para cada valor de la constante).

Para determinar una solución particular, es necesaria una **condición inicial** del problema, con lo cual será posible determinar el valor de la constante que corresponde a ese caso particular y con ello seleccionar una sola curva que sea solución de la ecuación diferencial dada.

La Ecuación Diferencial Ordinaria **(EDO)** general de ***primer orden*** es:

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

El ***método de Euler*** es uno de los métodos mas sencillos para resolver EDO´s con *condiciones iniciales*. La solución que ofrece este método, es una tabla de la función solución, con valores de "**y**" correspondientes a valores específicos de "**x**".

x	y(x)
x_0	y_0
x_1	y_1
x_2	y_2
::	::
x_n	y_n

Por esto uno de los requisitos para este método es *especificar el intervalo de x*.

También se requiere de:

- Una *ecuación diferencial de primer orden.*
  $y' = f(x,y)$

- La *condición inicial* ,es decir, el valor de **y** en un punto conocido $\mathbf{x_0}$.
  $y(x_0) = y_0$

La ecuación del ***método de Euler*** es la siguiente:

$$y_{i+1} = y_i + hf(x_i, y_i)$$

Datos: y' = f(x,y), espaciamiento, x_inferior, x_superior, condicion_inicial.

```
Euler
 begin

    // Determinar el número de datos
    numDatos = (x_superior - x_inferior) / espaciamiento  + 1

   // Llenar los valores del vector X
   X[0] := x_inferior
   for i:= 1 to (numDatos-1)
      X[i] := x_inicial + i*espaciamiento
   endfor

   // Aplicar el método de Euler
   Y[0] := condicion_inicial
   for k:= 1 to (numDatos –1)
      Y[k] :=  Y[k-1] + espaciamiento * f ( X[k-1], Y[k-1] )
   endfor

end Euler
```

8.12.11.- Algoritmo del Método de EULER MEJORADO para resolver ecuaciones diferenciales de primer orden.

El ***método de Euler mejorado*** es una modificación del *método de Euler* para resolver EDO's con *condiciones iniciales*. La solución que ofrece este método, es una tabla de la función solución, con valores de "**y**" correspondientes a valores específicos de "**x**".

x	y(x)
x_0	y_0
x_1	y_1
x_2	y_2
::	::
x_n	y_n

Por esto uno de los requisitos para este método es *especificar el intervalo de x*.

También se requiere de:

- Una *ecuación diferencial de primer orden.*
  $y' = f(x,y)$

- La *condición inicial* ,es decir, el valor de **y** en un punto conocido $\mathbf{x_0}$.
  $y(x_0) = y_0$

El método consiste en usar la ecuación de Euler como una ecuación ***predictora*** y usar este resultado en la ecuación ***correctora*** de Euler-Gauss.

Las ecuaciones del ***método de Euler Mejorado*** son las siguientes:

$$y_{i+1_p} = y_i + hf(x_i, y_i)$$

$$y_{i+1_c} = y_i + \frac{h}{2}(f(x_i, y_i) + f(x_{i+1}, y_{i+1_p}))$$

Datos: y' = f(x,y), espaciamiento, x_inferior, x_superior, condicion_inicial.

```
EulerMejorado
 begin

    // Determinar el número de datos
    numDatos = (x_superior – x_inferior) / espaciamiento  + 1

   // Llenar los valores del vector X
   X[0] := x_inferior
   for i:= 1 to (numDatos-1)
      X[i] := x_inicial + i*espaciamiento
   endfor

   // Aplicar el método de Euler mejorado
   Yp[0] := condicion_inicial
   Yc[0] := condicion_inicial
   for k:= 1 to (numDatos –1)
      Yp[k] :=  Yp[k-1] + espaciamiento * f ( X[k-1], Yc[k-1] )
      Yc[k] :=  Yc[k-1] + espaciamiento/2 *( f ( X[k-1], Yc[k-1])+ f ( X[k], Yp[k] )
      Yp[k] = Yc[k]
   endfor

end Eulermejorado
```

8.12.12.- Algoritmo del Método de RUNGE-KUTTA de orden 4 para resolver ecuaciones diferenciales de primer orden.

El ***método Runge-Kutta de orden 4*** es la forma de los métodos de Runge-Kutta de uso más común y así mismo más exactos para obtener soluciones aproximadas de ecuaciones diferenciales. La solución que ofrece este método, es una tabla de la función solución, con valores de "**y**" correspondientes a valores específicos de "**x**".

x	y(x)
x_0	y_0
x_1	y_1
x_2	y_2
::	::
x_n	y_n

Por esto uno de los requisitos para este método es *especificar el intervalo de x.*

También se requiere de:

- Una *ecuación diferencial de primer orden.*
  $y' = f(x,y)$

- La *condición inicial* ,es decir, el valor de **y** en un punto conocido $\mathbf{x_0}$.
  $y(x_0) = y_0$

El ***método de Runge-Kutta de 4º orden*** consiste en determinar constantes apropiadas de modo que una fórmula como:

$$y_{i+1} = y_i + ak_1 + bk_2 + ck_3 + dk_4$$

coincida con un desarrollo de Taylor hasta el término h^4, es decir, hasta el quinto término.

Las ecuaciónes del ***método de Runge-Kutta de 4º orden*** son las siguientes:

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

donde:

$$k_1 = hf(x_i, y_i)$$

$$k_2 = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1)$$

$$k_3 = hf(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2)$$

$$k_4 = hf(x_i + h, y_i + k_3)$$

Datos: y' = f(x,y), espaciamiento, x_inferior, x_superior, condicion_inicial.

```
RungeKutta
 begin

    // Determinar el número de datos
    numDatos = (x_superior – x_inferior) / espaciamiento  + 1

   // Llenar los valores del vector X
   X[0] := x_inferior
   for i:= 1 to (numDatos-1)
      X[i] := x_inicial + i*espaciamiento
   endfor

   // Aplicar el método de Runge-Kutta de orden 4
   Y[0] := condicion_inicial
   for k:= 1 to (numDatos –1)
      k1 := f ( X[k-1], Y[k-1])
      k2 := f ( X[k-1] + espaciamiento/2, Y[k-1] + espaciamiento*k1/2)
      k3 := f ( X[k-1] + espaciamiento/2, Y[k-1] + espaciamiento*k2/2)
      k4 := f ( X[k-1] + espaciamiento/2, Y[k-1] + espaciamiento*k3/2)

      Y[k] :=  Y[k-1] + (espaciamiento/6)* ( k1 + 2*k2 + 2*k3 + k4)
   endfor

end RungeKutta
```

9.- Documentación y pruebas.

Las pruebas se hicieron con el navegador "*Internet Explorer*" y posteriormente con "*Netscape Navigator*", primero en una PC, posteriormente se colocó el applet en el servidor para probar el sistema desde la red, las pruebas se llevaron a cabo conforme a lo establecido en las ***reglas de inspección.***

La documentación se elaboró conforme a las ***reglas de inspección*** establecidas en el **apéndice A.**

10.- CONCLUSIONES.

En esta tesis se desarrolló un sistema cómputo que funciona en red para la enseñanza de métodos numéricos. El desarrollo del sistema se basó en el proceso de desarrollo de software en espiral. En el capítulo del diseño de alto nivel , se describieron cuales son los métodos que incluye el sistema, de que manera los va a seleccionar el usuario y cuales son los datos de entrada y de salida de cada método. Además en el capítulo del diseño detallado se mencionaron los pormenores de las interfaces con el usuario y los algoritmos de los métodos en pseudocódigo junto con una breve descripción de cada uno de ellos. Por último, se incluyó en esta tesis la descripción y código del parser elaborado especialmente para este sistema.

La documentación, la elaboración del parser, la elaboración de los algoritmos de los métodos que componen el sistema, así como la de las interfaces gráficas con el usuario, se llevaron a cabo utilizando el proceso de desarrollo de software : *en espiral*, el cual consiste en ir añadiendo características y funcionalidades por etapas, al mismo tiempo que se va probando y revisando lo que ya esta implantado. La documentación que ilustra de que manera se llevó a cabo el proceso de diseño del sistema se incluye en esta tesis a manera de apéndices.

APÉNDICE A

REGLAS DE INSPECCION PARA EL ASEGURAMIENTO DE LA CALIDAD DEL SOFTWARE.

Documentación.

1.- Se generará la documentación correspondiente cada vez que se complete un ciclo con las siguientes etapas: *especificación de requerimientos->Diseño Detallado-→ Codificación → Pruebas*, generando el documento "***tesis***" el cual contiene la **descripción funcional del proyecto**, de tal forma que el documento se mantiene "vivo", es decir, durante todo el proceso se hacen adiciones, sustracciones y modificaciones.

2.- Al añadir una iteración mas al proyecto se podrán modificar especificaciones, el diseño de alto nivel o el diseño detallado de cualquiera de las iteraciones anteriores, estas modificaciones deberán quedar asentadas con fecha en el documento "***registro de inspecciones***".

3.- Estas reglas también podrán ser modificadas en cada inspección, cada modificación con su correspondiente registro en el *registro de inspecciones.*

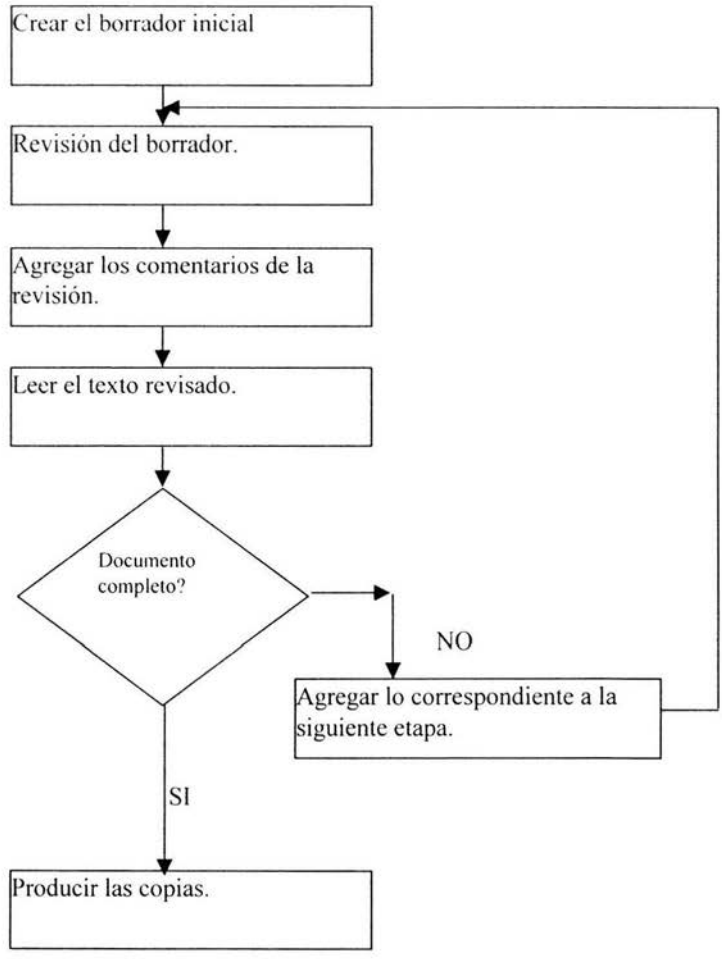
4.- Siempre que se termine un ciclo del proyecto se hará una inspección al documento "***tesis***" para asegurar que cumpla con las reglas enlistadas a continuación, asentando en el *registro de inspecciones* la fecha de la inspección y los resultados producidos por la misma.

Regla 1. claridad.- El documento deberá ser claro y conciso.
Regla 2. corrección.- No deberán existir errores de concepto ni de ortografía.
Regla 3. validez.- Lo documentado debe corresponder a lo implantado en el código.
Regla 4. completes.-Deberá documentarse todo lo implantado sin detalles innecesarios.

5.- La "*documentación funcional*" de los métodos quedará en páginas HTML que podrá consultar el usuario además de documentarse en la "tesis".

6.- La "*guía del usuario*" para el manejo del sistema deberá ser muy breve y fácil de entender y estará en páginas HTML. Se usará la voz activa en lugar de la voz pasiva, tal y como recomienda Ian Somerville [Somerville 1989], es decir, usar "debes introducir un valor inicial" en lugar de "el valor inicial deberá ser introducido".

7.- El documento "***tesis***" deberá pasar por el siguiente proceso de producción:



Código.

1.- El código deberá seguir las "reglas de codificación".del apéndice B

2.- Cada procedimiento que contenga un método numérico debe estar correctamente documentado por medio de un comentario.

3.- Deberán existir comentarios que describan al código con claridad, evitando los comentarios innecesarios.

4.- Deberán estar determinados los criterios de paro necesarios para evitar ciclos infinitos durante la ejecución de los métodos.

5.- El sistema deberá prever lo más posible los datos de entrada inválidos.

Pruebas.

1.- Una vez concluida la elaboración de cada método se efectuarán varias pruebas con datos diferentes, con esto se comprobará:

a).- Que las salidas en pantalla son correctas.

b).- Que las salidas en pantalla estén debidamente formateadas.

2.- Las pruebas se deben realizar conforme al **protocolo de pruebas** descrito en el Apéndice F

APÉNDICE B.

REGLAS DE CODIFICACION.

1.- Los nombres de las clases empiezan con mayúsculas.
2.- Los nombres de las variables empiezan con minúsculas.
3.- Después de las llaves no se requiere punto y coma.
4.- Las llaves se abren en el mismo renglón de la instrucción (clase o función if, for, etc.) y se cierran en un renglón aparte respetando la sangría.
5.- El código debe estar documentado claramente.
6.- Las sangrías deben ser uniformes en todo el código, ésta será de dos espacios.

APÉNDICE C.

REGISTRO DE LAS DIFERENTES FASES DEL PROCESO EN ESPIRAL.

No. de fase	Objetivo	Completada
1	Visualizar un applet desde una Página HTML	8-enero-2003
2	Captura de datos desde un applet (enteros, punto flotante)	29-enero-2003
3	Ejecución del applet del método de ***Newton-Raphson*** desde un menú.	19-marzo-2003

**********INSPECCION**********

No. de fase	Objetivo	Completada
4	Elaboración de todo el sistema de páginas HTML para acceder a todos los métodos	26-marzo-2003
5	Ejecución del applet del método de ***Bisección*** desde un menú HTML	2-abril-2003
6	Elaborar un sistema que pueda capturar matrices.	9-abril-2003
7	Ejecución del applet del método de ***Jacobi*** desde un menú HTML	16-abril-2003
8	Ejecución del applet del método de ***Gauss-Seidel*** desde un menú HTML	23-abril-2003
9	Dar formato a los resultados de los Métodos.	30-abril-2003
	********INSPECCION*******	7-mayo-2003
	Modificaciones y mejoras al sistema como resultado de la inspección	14-mayo-2003
10	Elaborar las pantallas de captura de los datos para los métodos de interpolación	21-mayo-2003
11	Elaborar las pantallas de captura de los datos para los métodos de integración.	21-mayo-2003
12	Elaborar las pantallas de captura de los datos para los métodos de ecuaciones diferenciales.	28-mayo-2003
13	Introducir un *parser* para resolver cualquier Función con el método de Newton-Raphson.	09-junio-2003
14	Introducir el *parser* para le método de Bisección.	09-junio-2003

No. de fase	Objetivo	Completada
15	Ejecución del applet del método de ***interpolación de Newton*** desde un menú HTML	13-junio-2003
16	Ejecución del applet del método de ***interpolación de Lagrange*** desde un menú HTML	16-junio-2003
17	Ejecución del applet del método de ***intergración trapezoidal*** desde un menú HTML	18-junio-2003
18	Ejecución del applet del método de ***intergración de Simpson 1/3*** desde un menú HTML	19-junio-2003
19	Ejecución del applet del método de ***intergración de Simpson 3/8*** desde un menú HTML	19-junio-2003
	********INSPECCION*******	20-junio-2003
	Modificaciones y mejoras al sistema como resultado de la inspección	23-junio-2003
20	Ejecución del applet del método de ***Euler*** para resolver ecuaciones diferenciales desde un menú HTML	24-junio-2003
21	Ejecución del applet del método de ***Euler Mejorado*** para resolver ecuaciones diferenciales desde un menú HTML	25-junio-2003
22	Ejecución del applet del método de ***Runge Kutta*** de orden 4 para resolver ecuaciones diferenciales desde un menú HTML.	30-junio-2003

No. de fase	Objetivo	Completada
	********INSPECCION*******	2-julio-2003
	Modificaciones y mejoras al sistema como resultado de la inspección	4-julio-2003
23	Elaboración de la ***guía de usuario***.	11-julio-2003
24	Elaboración de la *documentación de cada uno de los métodos* en paginas HTML.	18-julio-2003

APÉNDICE D.

REGISTRO DE LAS INSPECCIONES.

Fecha: 20 de marzo de 2003.
Resultados:

1.- Se agregó el capítulo *descripción* al documento "tesis" .

2.- Se encontraron algunas fallas, tanto en la presentación de la información como en el uso de los submenús, en el complicado sistema hecho en JAVA para el manejo de los menús, de tal forma que en la siguiente iteración del diseño se cambió por completo el enfoque y el sistema de elección de los métodos se elaboró de una manera mas fácil y con una mejor presentación para el usuario utilizando el lenguaje HTML. La elaboración del sistema en JAVA no fue en vano, pues sirvió como una experiencia reutilizable en la elaboración de las *páginas de captura de datos* para cada uno de los métodos numéricos.

Fecha: 7 de mayo de 2003.
Resultados:

Reglas de Codificación.

1.- El método de Jacobi y el de Gauss-Seidel no cumplían con la regla 4 del código, ya que faltaba establecer los criterios de paro para evitar ciclos infinitos con sistemas de ecuaciones con los que los métodos no convergen.
2.- Se corrigió un título de pantalla equivocado.
3.- Se modificaron varias sangrías.
4.- Se borró código innecesario.

Pruebas.

1.- No se había observado que en el campo "iteración" de la pantalla de resultados parciales para los métodos de bisección, Jacobi y Gauss-Seidel la numeración no salía correctamente, esto se resolvió aumentando el espacio asignado a este campo.
2.- En el método de bisección, el intervalo inicial no se desplegaba en la pantalla de resultados parciales.

Documentación.

1.- Se corrigieron 3 errores de escritura en el documento "Tesis".
2.- Se actualizó el documento "Tesis" de acuerdo con las "*mejoras al sistema*".
3.- Se agregó la regla 6 (sobre las sangrías) en las reglas de codificación.

Mejoras al sistema.

1.- Se agregó en cada método una pantalla adicional además de la que despliega los resultados parciales, para mostrar mas claramente el resultado final.

2.- Se modificó el diseño de las pantallas de los métodos de Jacobi y Gauss-Seidel de tal forma que la tolerancia se captura en la misma pantalla que los datos de la matriz y el vector, de esta forma se puede modificar la tolerancia sin tener que volver a introducir los datos.

Fecha: 20 de junio de 2003.

Resultados:

Código.

1.- Se agregó la regla 5 para la inspección del código.

2.- No se estaba cumpliendo con la regla 5 en el método de "Bisección", se agregó una "pantalla de advertencia" que se despliega cuando el *intervalo proporcionado es inválido.*

3.- Se protegieron los métodos de interpolación de Newton y de Lagrange contra extrapolaciones.

Pruebas.

1.- Se agregó la regla 2 para la inspección de las pruebas.

2.- Se elaboró el Apéndice F con el protocolo de pruebas que se deben realizar.

Mejoras al sistema.

1.- En el método de interpolación de Lagrange se pusieron dos botones, uno para borrar la tabla y otro para borrar el valor a interpolar.

2.- Se despliega una pantalla adicional para indicar cuando no hay convergencia en los métodos de Gauss-Seidel y de Jácobi.

3.- Se puso en otro color y en campos no editables los valores de X precalculados en las funciones tabulares donde se capturan los valores de Y.

Documentación.

1.- Se corrigió texto en el capítulo "Descripción" del documento "Tesis".

2.- Se actualizó una referencia a internet http://www.borland.com/jbuilder.

3.- Se actualizó el documento "Tesis" de acuerdo con el nuevo código y con las "*mejoras al sistema*".

4.- Se indico en el capítulo "Descripción" que este sistema ya se esta usando en clases reales.

Fecha: 2 de julio de 2003.

Resultados:

Código.

Ya no hubo modificaciones en el código.

Mejoras al sistema.

Ya no hubo modificaciones en el sistema.

Documentación.

1.- Se modificó la regla 5 de la documentación, para que la descripción de los métodos no solo este en la páginas HTML, sino también en el documento "tesis".

2.- Se mejoró la descripción del parser y se eliminó un apartado.

3.- Se agregaron "puentes" entre los diferentes capítulos de la tesis para hacerla mas coherente.

4.- Se modificó la manera en la que estaban hechas las referencias (se borraron números de páginas).

APÉNDICE E.

LISTA CON LOS NOMBRES DE LOS ARCHIVOS QUE COMPONEN AL SISTEMA.

La página HTML principal es: menus.HTML

menus.HTML tiene ligas a las páginas de la primera columna, las cuales pueden solicitar la ejecución de un método por medio de una liga a la página correspondiente indicada en la segunda columna. En la tercera columna se especifica la clase que se instancia cuando se lanza el applet correspondiente, cada clase esta dentro de un paquete, de tal forma que las clases están especificadas de la siguiente manera:

nombrePaquete.nombreClase.class

Página HTML	Página HTML que lanza el applet	clase (en JAVA)
NewtonRaphsonDoc.HTML	NewtonRaphson.HTML	newtonraphson.Raphson.class
BiseccionDoc.HTML	Biseccion.HTML	biseccion.Biseccion.class
JacobiDoc.HTML	Jacobi.HTML	lineal1.Jacobi.class
SeidelDoc.HTML	Seidel.HTML	lineal2.Seidel.class
InterpoNewton.HTML	InterpoNewton.HTML	interpola1.Newton.class
LagrangeDoc.HTML	Lagrange.HTML	interpola2.Lagrange.class
TrapezoidalDoc.HTML	Trapezoidal.HTML	integra1.Trapezoidal.class
Simpson13Doc.HTML	Simpson13.HTML	integra2.Simpson13.class
Simpson38Doc.HTML	Simpson38.HTML	integra3.Simpson38.class
EulerDoc.HTML	Euler.HTML	ecDif1.Euler.class
EulerMDoc.HTML	EulerM.HTML	ecDif2.EulerM.class
RungeKuttaDoc.HTML	RungeKutta.HTML	ecDif3.RungeKutta.class

Cada una de las páginas *Doc.HTML tienen 3 ligas:

1.- A la página en donde se documenta el método.
2.- A la guía de usuario que explica como introducir los datos y como interpretar los resultados.
3.- A la página que lanza el applet que ejecuta el método.

A continuación se lista el nombre de las páginas HTML que contienen la ***descripción*** de cada uno de los métodos:

Nombre de la página	Método que describe
MetodRaphson.html	Newton-Raphson.
MetodBiseccion.html	Bisección
MetodJacobi.html	Jacobi
MetodSeidel.html	Gauss-Seidel
MetodNewton.html	Interpolación de Newton
MetodLagrange.html	Interpolación de Lagrange
MetodTrapecio.html	Integración trapezoidal
MetodSimpson13.html	Integración usando la fórmula de Simpson 1/3
MetodSimpson38.html	Integración usando la fórmula de Simpson 3/8
MetodEuler.html	Euler para ecuaciones diferenciales ordinarias
MetodEulerM.html	Euler mejorado para ecuaciones diferenciales ordinarias
MetodRunge.html	Runge Kutta para ecuaciones diferenciales ordinarias

A continuación se lista el nombre de las páginas HTML que contienen la ***guía de usuario*** de cada uno de los métodos:

Nombre de la página	Guía de usuario para el/los método(s)
UsaRaphson.html	Newton-Raphson
UsaBiseccion.html	Bisección
UsaLineal.html	Jacobi y Gauss-Seidel
UsaNewton.html	Interpolación de Newton
UsaLagrange.html	Interpolación de Lagrange
UsaIntegracion.html	Trapezoidal, Simpson 1/3, Simpson 3/8
UsaDiferenciales.html	Euler, Euler Mejorado, Runge Kutta

Las instrucciones del parser están en la página funciones.html. Para los métodos en los que es necesario introducir alguna función, existe una liga a funciones.html dentro de su guía de usuario correspondiente.

Libros consultados para la descripción de los métodos:

Newton-Raphson. (en MetodRaphson.html).

[Burden,1998] páginas 65-71.
[Chapra 1999] páginas 156-162.
[Curtis,1991] páginas 14-21.
[Iriarte,1990] páginas 35-43.
[James,1973] páginas 154-160.
[Maron, 1995] páginas 73-78.
[Nakamura,1992] páginas 73-76.
[Nieves,1999] páginas 46-49.
[Smith,1988] páginas 115-123.

Bisección (en MetodBiseccion.html).

[Burden,1998] páginas 48-52.
[Chapra,1999] páginas 131-140.
[Iriarte,1990] páginas 23-26.
[Maron, 1995] páginas 86-90.
[Nakamura,1992] páginas 63-67.
[Nieves,1999] páginas 57-59.
[Smith,1988] páginas 76-80.

Jacobi (en MetodJacobi.html).

[Atkinson,1987] páginas 127-129.
[Burden,1998] páginas 444-447.
[Curtis,1991] páginas 127-128.
[Iriarte,1990] páginas 63-67.
[Maron, 1995] páginas 196-200.
[Nieves,1999] páginas 207-222

Gauss Seidel (en MetodSeidel.html).

[Atkinson,1987] páginas 129-131.
[Burden,1998] páginas 447-449.
[Chapra,1999] páginas 311-319.
[Curtis,1991] páginas 128-130.
[Iriarte,1990] páginas 67-69.
[James,1973] páginas 238-243.
[Maron, 1995] páginas 196-200.
[Nieves,1999] páginas 209-222

Interpolación de Newton (en MetodNewton.html).

[Burden,1998] páginas 123-130.
[Curtis,1991] páginas 173-184.
[Iriarte,1990] páginas 99-114.
[Maron, 1995] páginas 289-297.
[Nakamura,1992] páginas 32-40.
[Nieves,1999] páginas 338-354.
[Scheid,1990] páginas 34,58,59,79,80.

Interpolación de Lagrange (en MetodLagrange.html).

[Atkinson,1987] páginas 178-184.
[Burden,1998] páginas 107-119.
[Chapra,1999] páginas 515-519.
[Curtis,1991] páginas 197-199.
[Iriarte,1990] páginas 114-118
[Maron, 1995] páginas 284-287.
[Nakamura,1992] páginas 24-31.
[Nieves,1999] páginas 323-329.
[Smith,1988] páginas 268-278

Integración trapezoidal (en MetodTrapecio.html).

[Curtis,1991] páginas 237-240.
[Chapra,1999] páginas 619-630.
[Iriarte,1990] páginas 127-130.
[James,1973] páginas 303-316.
[Maron, 1995] páginas 389.
[Nakamura,1992] páginas 110-114.
[Nieves,1999] páginas 393-398.
[Scheid,1990] páginas 107.
[Smith,1988] páginas 323-329.

Integración usando la fórmula de Simpson 1/3(en MetodSimpson13.html).

[Curtis,1991] páginas 242-244.
[Chapra,1999] páginas 630-635.
[Iriarte,1990] páginas 130-132.
[James,1973] páginas 316-323.
[Maron, 1995] páginas 390-397.
[Nakamura,1992] páginas 115-118.
[Nieves,1999] páginas 398-402.
[Scheid,1990] páginas 108.
[Smith,1988] páginas 329-332.

Integración usando la fórmula de Simpson 3/8(en MetodSimpson38.html).

[Curtis,1991] páginas 244-246.
[Chapra,1999] páginas 635-638.
[Iriarte,1990] páginas 133-140.
[James,1973] páginas 316-323.
[Maron, 1995] páginas 390-397.
[Nakamura,1992] páginas 119-120.
[Nieves,1999] páginas 398-402.
[Smith,1988] páginas 329-332.

Euler para ecuaciones diferenciales ordinarias (en MetodEuler.html).

[Curtis,1991] páginas 288-292.
[Chapra,1999] páginas 715-725.
[Iriarte,1990] páginas 151-159.
[James,1973] páginas 352-363.
[Maron, 1995] páginas 446-448.
[Nakamura,1992] páginas 292-298.
[Nieves,1999] páginas 470-474.
[Scheid,1990] páginas 197.
[Smith,1988] páginas 416-420.

Euler mejorado para ecuaciones diferenciales ordinarias (en MetodEulerM.html).

[Curtis,1991] páginas 288-292.
[Iriarte,1990] páginas 159-162.
[James,1973] páginas 364-378.
[Maron, 1995] páginas 450-452.
[Nieves,1999] páginas 477-480.
[Smith,1988] páginas 440-444.

Runge Kutta para ecuaciones diferenciales ordinarias (enMetodRunge.html).

[Atkinson,1987] páginas 269-311.
[Chapra,1999] páginas 736-747.
[Curtis,1991] páginas 292-295.
[Iriarte,1990] páginas 165-171.
[James,1973] páginas 379-386.
[Maron, 1995] páginas 463-465.
[Nakamura,1992] páginas 299-311.
[Nieves,1999] páginas 480-484.
[Scheid,1990] páginas 202-204.
[Smith,1988] páginas 455-463.
[Zill, 1982] páginas

APÉNDICE F.

PROTOCOLO DE PRUEBAS A REALIZAR PARA CADA UNO DE LOS MÉTODOS,(Y LOS RESULTADOS OBTENIDOS).

PRUEBAS PREELIMINARES PARA EL PARSER.

Antes de probar el parser con los métodos, es necesario hacer pruebas independientes para verificar que el parser funciona correctamente.
Lista de pruebas:

1.- Con las 14 funciones que soporta.
2.- Con expresiones algebraicas que contengan potencia (^).
3.- Con expresiones algebraicas que contengan multiplicación y división (*, /).
4.- Con expresiones algebraicas que contengan suma y resta (+, -).
5.- Con expresiones algebraicas que contengan una mezcla de los operadores de los puntos 2 al 4 usando paréntesis.
6.- Con expresiones algebraicas que contengan una mezcla de los operadores de los puntos 2 al 4 usando paréntesis combinadas con funciones.
7.- Con expresiones que contengan errores de sintaxis.

MÉTODO DE NEWTON-RAPHSON.

Lista de funciones a probar con una tolerancia de 0.001:

	Función	Valor inicial	Resultado	Num. de iteraciones
1.-	$Cos(x) - x^2 = 0$	8	0.824132	6
2.-	$4Sin(x) - e^x = 0$	5	1.364958	8
3.-	$2x^2 + 1 - e^x = 0$	0.5	0.74085	4
4.-	$xTan(x)-1 = 0$	3.1415	3.4261	35
5.-	$4x-Tan(x) = 0$	1	1.3932	43
6.-	$2x + e^x - e^{2x} = 0$	1	0.4563	6
7.-	$Sin(x) + 1 - x^2 = 0$	0.75	1.4096	5
8.-	$e^{-x+1}Sin(x) + 1 = 0$	-1	-0.2813	4

MÉTODO DE BISECCIÓN.

Lista de funciones a probar con una tolerancia de 0.001:

	Función	Intervalo inicial	Resultado	Num. de iteraciones
1.-	$Cos(x) - x^2 = 0$	(-0.5, 1.5)	0.82324	11
2.-	$4Sin(x) - e^x = 0$	(0.5,2)	1.36499	11
3.-	$2x^2 + 1 - e^x = 0$	(0, 1)	0.74121	10
4.-	$xTan(x)-1 = 0$	(3,4)	3.424804	10
5.-	$4x-Tan(x) = 0$	(1,1.5)	1.39355	9
6.-	$2x + e^x - e^{2x} = 0$	(0,0.5)	0.45605	9
7.-	$Sin(x) + 1 - x^2 = 0$	(0,2)	1.40917	11
8.-	$e^{-x+1}Sin(x) + 1 = 0$	(-1,1)	-0.28222	11

MÉTODO DE JACOBI.

Sistemas de ecuaciones lineales a probar con una tolerancia de 0.001:
Y con el vector inicial **0**.

De dos incógnitas:	Vector solución	Num. de iteraciones.
$2x_1 - x_2 = 1$ $3x_1 + 2x_2 = 5$	$x_1 = 1.00075$ $x_2 = 1.00075$	50
$2x_1 + x_2 = 1$ $3x_1 + 2x_2 = 5$	$x_1 = 0.230601$ $x_2 = 0.538069$	12
$5x_1 + 2x_2 = 7$ $4x_1 - x_2 = 3$	no hubo convergencia	

De tres incógnitas:	Vector solución	Num. de iteraciones.
$6x_1 + 2x_2 + x_3 = 22$ $-x_1 + 8x_2 + 2x_3 = 30$ $x_1 - x_2 + 6x_3 = 23$	$x_1 = 1.99977$ $x_2 = 3.00001$ $x_3 = 4.00014$	10
$7x_1 - x_2 + 4x_3 = 8$ $3x_1 - 8x_2 + 2x_3 = -4$ $-4x_1 + x_2 - 6x_3 = 3$	$x_1 = 2.3773$ $x_2 = 0.908321$ $x_3 = -1.9338$	22
$5x_1 - x_2 + 3x_3 = 4$ $-2x_1 + 5x_2 + 3x_3 = -2$ $3x_1 + 4x_2 + 6x_3 = 2$	$x_1 = 0.59478$ $x_2 = -0.306198$ $x_3 = 0.240485$	13
$10x_1 + x_2 + 2x_3 = 44$ $2x_1 + 10x_2 + x_3 = 51$ $x_1 + 2x_2 + 10x_3 = 61$	$x_1 = 3.00007$ $x_2 = 4.00007$ $x_3 = 5.00007$	9

De 5 incógnitas:	Vector solución	Num. de iteraciones.
$4x_1 + x_2 + x_3 + x_5 = 6$ $-x_1 - 3x_2 + x_3 + x_4 = 6$ $2x_1 + x_2 + 5x_3 - x_4 - x_5 = 6$ $-x_1 - x_2 - x_3 + 4x_4 = 6$ $+2x_2 - x_3 + x_4 + 4x_5 = 6$	$x_1 = 0.787088$ $x_2 = -1.0030357$ $x_3 = 1.8660481$ $x_4 = 1.912449$ $x_5 = 1.989570$	12

MÉTODO DE GAUSS-SEIDEL.

Sistemas de ecuaciones lineales a probar con una tolerancia de 0.001:
Y con el vector inicial **0**.

De dos incógnitas:	Vector solución	Num. de iteraciones.
$2x_1 - x_2 = 1$ $3x_1 + 2x_2 = 5$	$x_1 = 1.00021$ $x_2 = 0.99968$	28
$2x_1 + x_2 = 1$ $3x_1 + 2x_2 = 5$	$x_1 = 0.2309655$ $x_2 = 0.538579$	7
$5x_1 + 2x_2 = 7$ $4x_1 - x_2 = 3$	No hubo convergencia	

De tres incógnitas:	Vector solución	Num. de iteraciones.
$6x_1 + 2x_2 + x_3 = 22$ $-x_1 + 8x_2 + 2x_3 = 30$ $x_1 - x_2 + 6x_3 = 23$	$x_1 = 1.99994$ $x_2 = 2.99997$ $x_3 = 4.000005$	6
$7x_1 - x_2 + 4x_3 = 8$ $3x_1 - 8x_2 + 2x_3 = -4$ $-4x_1 + x_2 - 6x_3 = 3$	$x_1 = 2.37735$ $x_2 = 0.908177$ $x_3 = -1.93354$	10
$5x_1 - x_2 + 3x_3 = 4$ $-2x_1 + 5x_2 + 3x_3 = -2$ $3x_1 + 4x_2 + 6x_3 = 2$	$x_1 = 0.7924$ $x_2 = -0.0754$ $x_3 = -0.01258$	3
$10x_1 + x_2 + 2x_3 = 44$ $2x_1 + 10x_2 + x_3 = 51$ $x_1 + 2x_2 + 10x_3 = 61$	$x_1 = 2.9999$ $x_2 = 4.0000$ $x_3 = 5.0000$	4

De 5 incógnitas:	Vector solución	Num. de iteraciones.
$4x_1 + x_2 + x_3 + x_5 = 6$	$x_1 = 0.78668$	
$-x_1 - 3x_2 + x_3 + x_4 = 6$	$x_2 = -1.002718$	7
$2x_1 + x_2 + 5x_3 - x_4 - x_5 = 6$	$x_3 = 1.86628$	
$-x_1 - x_2 - x_3 + 4x_4 = 6$	$x_4 = 1.91256$	
$+2x_2 - x_3 + x_4 + 4x_5 = 6$	$x_5 = 1.98978$	

MÉTODO DE INTERPOLACIÓN DE NEWTON.

1.- Para la siguiente función tabular con espaciamientos constantes:

x	F(x)
0	-5
1	1
2	9
3	25
4	55
5	105

Encontrar el valor de F(x) para x = 1.5
Para una $X_0 = 1$.

La tabla de diferencias obtenida fue:

x	F(x)	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
0	-5	6	2	6	0
1	1	8	8	6	0
2	9	16	14	6	0
3	25	30	20	0	0
4	55	50	0	0	0
5	105	0	0	0	0

Y el resultado: F(1.5) = 4.375

2.- Para la siguiente función tabular con espaciamientos constantes:

x	F(x)
-5	0
-2	15
1	18
4	15
7	12
10	15

Encontrar el valor de F(x) para x = 7.1
Para una $X_0 = 1$.

La tabla de diferencias obtenida fue:

x	F(x)	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
-50	0	15	-12	6	0
-2	15	3	-6	6	0
1	18	-3	0	6	0
4	15	-3	6	0	0
7	12	3	0	0	0
10	15	0	0	0	0

Y el resultado: F(7.1) =11.97003

3.- Para la siguiente función tabular con espaciamientos constantes:

x	F(x)
-3	-51
-1	-11
1	-11
3	-3
5	61

Encontrar el valor de F(x) para x = 0.5
Para una $X_0 = -1$.

La tabla de diferencias obtenida fue:

x	F(x)	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
-3	-51	40	-40	48	0
-1	-11	0	8	48	0
1	-11	8	56	0	0
3	-3	64	0	0	0
5	61	0	0	0	0

Y el resultado: F(0.5) = -9.875

MÉTODO DE INTERPOLACIÓN DE LAGRANGE.

1.- Para la siguiente función tabular con espaciamientos variables:

X	F(X)
0	5
1	7
2	9
5	15
7	19

Encontrar el valor de F(x) para x = 3
Resultado: "Usando un polinomio de grado 4 F(3)= 11.00"

Encontrar el valor de F(x) para x = 6
Resultado: "Usando un polinomio de grado 4 F(3)= 17.00"

2- Para la siguiente función tabular con espaciamientos variables:

X	F(X)
0.0	1.000
0.05	1.64875
1.0	2.71828
2.0	7.38906

Encontrar el valor de F(x) para x = 0.25
Resultado: "Usando un polinomio de grado 3 F(0.25)= 1.291611"

Encontrar el valor de F(x) para x = 0.75
Resultado: "Usando un polinomio de grado 3 F(0.75)= 2.111083"

MÉTODO DE INTEGRACIÓN TRAPEZOIDAL.

1.- A partir de la siguiente tabla:

X	F(X)
1	-4
2	6
3	18
4	32
5	48
6	66
7	86
8	108
9	132

a.- Encontrar el valor de la integral de la función tabular en el intervalo $1 \leq X \leq 9$
Resultado: 428.00 u^2.

b- Encontrar el valor de la integral de la función tabular en el intervalo $1 \leq X \leq 7$
Resultado: 211.00 u^2.

2.- A partir de la siguiente tabla:

X	F(X)
3	4.2948
4	9.6329
5	17.4742
6	28.0134
7	41.4098

- Encontrar el valor de la integral de la función tabular en el intervalo $3 \leq X \leq 7$
Resultado: 77.9728 u^2.

MÉTODO DE INTEGRACIÓN DE SIMPSON 1/3.

1.- A partir de la siguiente tabla:

X	F(X)
1	-4
2	6
3	18
4	32
5	48
6	66
7	86
8	108
9	132

a.- Encontrar el valor de la integral de la función tabular en el intervalo $1 \leq X \leq 9$
Resultado: 426.00 u^2.

b.- Encontrar el valor de la integral de la función tabular en el intervalo $1 \leq X \leq 7$
Resultado: 210 u^2.

2.- A partir de la siguiente tabla:

X	F(X)
3	4.2948
4	9.6329
5	17.4742
6	28.0134
7	41.4098

- Encontrar el valor de la integral de la función tabular en el intervalo $3 \leq X \leq 7$
Resultado: 77.07939 u^2.

MÉTODO DE INTEGRACIÓN DE SIMPSON 3/8.

A partir de la siguiente tabla:

X	F(X)
1	-4
2	6
3	18
4	32
5	48
6	66
7	86
8	108
9	132

1.- Encontrar el valor de la integral de la función tabular en el intervalo $1 \leq X \leq 7$
Resultado: 210 u^2.

2.- Encontrar el valor de la integral de la función tabular en el intervalo $1 \leq X \leq 4$
Resultado: 37.5 u^2.

MÉTODO DE EULER PARA ECUACIONES DIFERENCIALES DE PRIMER ORDEN.

1.- Encontrar la función tabular solución de la ecuación:

$y' = x + 2xy$

Con la condición inicial: $Y(0.5) = 1$

En el intervalo: $0.5 \leq X \leq 1.0$

Con un espaciamiento de : 0.1

Resultado:

X	F(X)	Sol. exacta
0.5	1	1
0.6	1.15	1.17442
0.7	1.3479	1.40687
0.8	1.60672	1.71547
0.9	1.94379	2.12601
1.0	2.38368	2.67550

2.- Encontrar la función tabular solución de la ecuación:

$y' = -y + x + 1$

Con la condición inicial: $Y(0) = 1$

En el intervalo: $0 \leq X \leq 0.4$

Con un espaciamiento de : 0.1

Resultado:

X	F(X)	Sol. exacta
0	1	1
0.1	1	1.00424
0.2	1.01	1.01873
0.3	1.029	1.04082
0.4	1.0561	1.07032

3.- Encontrar la función tabular solución de la ecuación:

$y' = \sin(x) + e^{-x}$

Con la condición inicial: $Y(0) = 0$

En el intervalo: $0 \leq X \leq 1$

Con un espaciamiento de : 0.5

Resultado:

X	F(X)	Sol. exacta
0	0	0
0.5	0.5	0.5158
1	1.04297	1.09181

4.- Encontrar la función tabular solución de la ecuación:

$y' = -xy + 4x/y$

Con la condición inicial: $Y(0) = 1$

En el intervalo: $0 \leq X \leq 1$

Con un espaciamiento de : 0.25

Resultado:

X	F(X)	Sol. exacta
0	1	1
0.25	1	1.087088
0.5	1.1875	1.289805
0.75	1.4601	1.51348
1	1.7000	1.70187

MÉTODO DE EULER MEJORADO PARA ECUACIONES DIFERENCIALES DE PRIMER ORDEN.

1.- Encontrar la función tabular solución de la ecuación:

$y' = x + 2xy$

Con la condición inicial: $Y(0.5) = 1$

En el intervalo: $0.5 \leq X \leq 1.0$

Con un espaciamiento de : 0.1

Resultado:

X	F(X)	Sol. exacta
0.5	1	1
0.6	1.174	1.17442
0.7	1.40568	1.40687
0.8	1.7128	1.71547
0.9	2.1209	2.12601
1.0	2.6660	2.67550

2.- Encontrar la función tabular solución de la ecuación:

$y' = -y + x + 1$

Con la condición inicial: $Y(0) = 1$

En el intervalo: $0 \leq X \leq 0.4$

Con un espaciamiento de : 0.1

Resultado:

X	F(X)	Sol. exacta
0	1	1
0.1	1.005	1.00424
0.2	1.019024	1.01873
0.3	1.04121	1.04082
0.4	1.07080	1.07032

3.- Encontrar la función tabular solución de la ecuación:

$y' = \sin(x) + e^{-x}$

Con la condición inicial: $Y(0) = 0$

En el intervalo: $0 \leq X \leq 1$

Con un espaciamiento de : 0.5

Resultado:

X	F(X)	Sol. exacta
0	0	0
0.5	0.52148	0.5158
1	1.09525	1.09181

4.- Encontrar la función tabular solución de la ecuación:

$y' = -xy + 4x/y$

Con la condición inicial: $Y(0) = 1$

En el intervalo: $0 \leq X \leq 1$

Con un espaciamiento de : 0.25

Resultado:

X	F(X)	Sol. exacta
0	1	1
0.25	1.09375	1.087088
0.5	1.29485	1.289805
0.75	1.51142	1.51348
1	1.69228	1.70187

MÉTODO DE RUNGE KUTTA DE ORDEN 4 PARA ECUACIONES DIFERENCIALES.

1.- Encontrar la función tabular solución de la ecuación:

$y' = x + 2xy$

Con la condición inicial: $Y(0.5) = 1$

En el intervalo: $0.5 \leq X \leq 1.0$

Con un espaciamiento de : 0.1

Resultado:

X	F(X)	Sol. exacta
0.5	1	1
0.6	1.170026	1.17442
0.7	1.396180	1.40687
0.8	1.695542	1.71547
0.9	2.092272	2.12601
1.0	2.620866	2.67550

2.- Encontrar la función tabular solución de la ecuación:

$y' = -y + x + 1$

Con la condición inicial: $Y(0) = 1$

En el intervalo: $0 \leq X \leq 0.4$

Con un espaciamiento de : 0.1

Resultado:

X	F(X)	Sol. exacta
0	1	1
0.1	1.004043	1.00424
0.2	1.017295	1.01873
0.3	1.03887	1.04082
0.4	1.06797	1.07032

3.- Encontrar la función tabular solución de la ecuación:

$y' = \sin(x) + e^{-x}$

Con la condición inicial: $Y(0) = 0$

En el intervalo: $0 \leq X \leq 1$

Con un espaciamiento de : 0.5

Resultado:

X	F(X)	Sol. exacta
0	0	0
0.5	0.5109	0.5158
1	1.0822	1.09181

4.- Encontrar la función tabular solución de la ecuación:

$y' = -xy + 4x/y$

Con la condición inicial: $Y(0) = 1$

En el intervalo: $0 \leq X \leq 1$

Con un espaciamiento de : 0.25

Resultado:

X	F(X)	Sol. exacta
0	1	1
0.25	1.07467	1.087088
0.5	1.27751	1.289805
0.75	1.50859	1.51348
1	1.70436	1.70187

Nota: Todos los resultados de las pruebas aquí expuestas están validados con cálculos hechos a mano o con resultados de libros que contienen ejercicios resueltos. [Burden,1998] [Chapra, 1999] [Curtis,1991] [Iriarte,1990] [Nieves,1999].

17.- APÉNDICE G: Código del "*PARSER*".

A continuación se presenta el código del parser descrito en la sección 7.4 del capítulo de diseño de alto nuvel. El código esta escrito en JAVA.

```
package metonume;

import JAVA.awt.*;
import JAVA.awt.event.*;
public class ParserFuncion2 {
  String expresion;
  int indice;
  double x,y;

  ////////////////////////////////////////////////////////////////////////
  //
  //  manejo de errores
  //
  // En caso de cualquier error se llama a ésta función
  //
  ////////////////////////////////////////////////////////////////////////
  void ERROR( String mensaje ) throws ParserFuncionException
  {
    // se construye la excepción con el mensaje
    // y se lanza
    throw(new ParserFuncionException(mensaje, indice));
  }
```

```
////////////////////////////////////////////////////////////////////////////
//
// rutinas para clasificar un caracter
//
////////////////////////////////////////////////////////////////////////////

//
// verdadera cuando c es letra
//
boolean es_letra(char c){
  if( (c >= 'A' && c <= 'Z')|| (c >= 'a' && c <= 'z'))
    return(true);
  else
    return(false);
}

//
// verdadera cuando c es digito
//
boolean es_digito(char c){
  if( c >= '0' && c <= '9' )
    return(true);
  else
    return(false);
}

//
// verdadera cuando c es un operador de nivel n
//
boolean es_operador(int n, char c){
  switch( n )
    {
      case 0:{ if(c == '+' || c == '-')
                 return(true);
               else
                 return(false);
             }
      case 1:{ if(c == '*' || c == '/')
                 return(true);
               else
                 return(false);
             }
      case 2:{ if(c == '^')
                 return(true);
               else
                 return(false);
             }

      default: return(false);
    }
}

//
// verdadera cuando c es un separador
//
boolean es_separador(char c){
  if(c == ' ' || c == '\t')
    return(true);
  else
    return(false);
}
```

```
////////////////////////////////////////////////////////////////////////
//
// rutinas para leer de la cadena de caracteres
//
////////////////////////////////////////////////////////////////////////

//
// lee un caracter de la cadena
//
char lee_caracter(){
  if( indice >= expresion.length() )
  {
    indice++;
    return 0;
  }
  return expresion.charAt(indice++);
}

//
// deslee un caracter de la expresion
//
void deslee_caracter(char c){
  --indice;
}

//
// lee un caracter de la expresión sin adelantar el indice
//
char pre_lee_caracter(){
  return expresion.charAt(indice);
}

////////////////////////////////////////////////////////////////////////
//
//  rutinas para leer tokens especificos
//
////////////////////////////////////////////////////////////////////////

//
// lee caracteres mientras sean separadores
//
void lee_separador(){
  char c;
  while( es_separador(c=lee_caracter()))
          ;
  deslee_caracter(c);
}

//
// lee un caracter obligatorio
//
void lee_obligatorio(char esperado) throws ParserFuncionException
{
  lee_separador();
  char c = lee_caracter();
  if( c != esperado ){
    StringBuffer texto = new StringBuffer("Se espera el caracter ");
    texto=texto.append(esperado);
    ERROR(new String(texto));
  }
}
```

```
////////////////////////////////////////////////////////////////////////
//
// rutinas para leer una variable
//
////////////////////////////////////////////////////////////////////////
//
// determina si c puede pertenecer a una cadena alfanumérica
//
boolean alfanumerica(char c){
  if(es_digito( c ) || es_letra( c ))
    return(true);
  else
    return(false);
}

//
// Lee el nombre de una variable
// Regresa una referencia a esa variable
//
double lee_variable() throws ParserFuncionException
{
  if( ! (('X'== pre_lee_caracter() )||('Y'== pre_lee_caracter() )))
    ERROR("El nombre de las variables debe ser x o y");

  if('X'== lee_caracter() ){
    return x;
  }else{
    return y;
  }
}

////////////////////////////////////////////////////////////////////////
//
// rutinas para leer un número
//
////////////////////////////////////////////////////////////////////////
```

```
//
// determina si c puede pertenecer a una cadena numérica
//
boolean numerica(char c){
  if(es_digito( c ))
    return(true);
  else
    return(false);
}

//
// lee una cadena de caracteres numéricos
//
String lee_Cadena_numerica( ) throws ParserFuncionException
{
  String cad = new String();
  char c;

  while( numerica(c = lee_caracter()) )
  {
    cad = cad.concat(String.valueOf(c));
  }
  deslee_caracter(c);

  if( cad.length() > 10 )
    ERROR("Cadena demasiado larga");

  return cad;
}

//
// lee el punto decimal
//
char lee_punto(){
  char c = lee_caracter();
  if( c != '.' )
    {
      deslee_caracter(c); // el punto es opcional
      return(0);          // así que el programa continúa
    }
  return '.';
}

//
// lee un número real
//
double lee_numero() throws ParserFuncionException
{
  String e = lee_Cadena_numerica();
  String p = "";
  if( '.'==lee_punto())
    p = ".";
  String f = lee_Cadena_numerica();
  String numero = new String(e+p+f);
  if( numero.equals(".") )
    ERROR("Punto decimal sin digitos");

  double n = Double.valueOf(numero).doubleValue();
  return(n);
}
////////////////////////////////////////////////////////////////////////////
```

```
  //
  //  rutinas para leer y evaluar un dato evaluable
  //
  ////////////////////////////////////////////////////////////////////////

  //
  // Determina si el nombre dado es el de una función y, de ser así,
  // determina el tipo de función.
  //
  int tipo_funcion( String nombre ){
    if( (nombre.compareTo("EXP"))==0) return 0;
    if( (nombre.compareTo("LN" ))==0) return 1;
    if( (nombre.compareTo("SIN"))==0) return 2;
    if( (nombre.compareTo("COS"))==0) return 3;
    if( (nombre.compareTo("TAN"))==0) return 4;
    if( (nombre.compareTo("COT"))==0) return 5;
    if( (nombre.compareTo("SEC"))==0) return 6;
    if( (nombre.compareTo("CSC"))==0) return 7;
    if( (nombre.compareTo("ASIN"))==0) return 8;
    if( (nombre.compareTo("ACOS"))==0) return 9;
    if( (nombre.compareTo("ATAN"))==0) return 10;
    if( (nombre.compareTo("ACOT"))==0) return 11;
    if( (nombre.compareTo("ASEC"))==0) return 12;
    if( (nombre.compareTo("ACSC"))==0) return 13;
    if( (nombre.compareTo("SQRT"))==0) return 14;
    return -1;
  }

  //
  // Evalúa la función f en el valor dado.
  //
  double evaluar_funcion( int f, double valor ) throws
ParserFuncionException
  {
    switch( f )
    {
      case 0:  return Math.exp( valor );
      case 1:  if( valor <= 0 ) break; return Math.log( valor );
      case 2:  return Math.sin( valor );
      case 3:  return Math.cos( valor );
      case 4:  return Math.tan( valor );
      case 5:  if( Math.tan( valor ) == 0 ) break; return 1/Math.tan(
                                                          valor );
      case 6:  if( Math.cos( valor ) == 0 ) break; return 1/Math.cos(
                                                          valor );
      case 7:  if( Math.sin( valor ) == 0 ) break; return 1/Math.sin(
                                                          valor );
      case 8:  return Math.asin( valor );
      case 9:  return Math.acos( valor );
      case 10: return Math.atan( valor );
      case 11: if( valor == 0 ) break; return Math.atan( 1/valor );
      case 12: if( valor == 0 ) break; return Math.acos( 1/valor );
      case 13: if( valor == 0 ) break; return Math.asin( 1/valor );
      case 14: if( valor <  0 ) break; return Math.asin( 1/valor );
      case -1: ERROR("Funcion desconocida.");
    }
    ERROR("Argumento en llamado a funcion no valido.");

    return 0;
  }
```

```
//
// lee un nombre de funcion y su argumento y regresa su valor.
//
double lee_funcion() throws ParserFuncionException
{
  String nombre = lee_cadena_alfanumerica();
  int f = tipo_funcion( nombre );

  // Leer el argumento de la función
  lee_obligatorio( '(' );          // lee '('
  double valor = lee_expresion();// lee RECURSIVAMENTE una expresión
  lee_obligatorio( ')' );                  // lee ')'

  // Regresar el valor de la función
  return evaluar_funcion( f, valor );
}

//
// lee y evalua un dato evaluable
//
double lee_evaluable() throws ParserFuncionException
{
  char c = pre_lee_caracter();   // sólo pre-lectura

  // numero
  if( es_digito(c) || c == '.' )
    return lee_numero();

  // variable
  if( ('X'==c)||('Y'==c) )
    return lee_variable();

  // función
  if( es_letra(c) )
    return lee_funcion();

  // expresión
  if( c == '(' )
  {
    c = lee_caracter();                                  // lee '('
    double valor = lee_expresion();// lee RECURSIVAMENTE una expresión
    lee_obligatorio( ')' );                  // lee ')'

    return valor;
  }

  ERROR("Falta dato evaluable");
      return 0;
}
```

```
////////////////////////////////////////////////////////////////////////
//
//  rutinas para leer y evaluar una expresión
//
////////////////////////////////////////////////////////////////////////

//
// lee un operador de nivel n
//
char lee_operador(int n){
  lee_separador();
  char c = lee_caracter();
  if( !es_operador(n,c) )
  {
    // c no es un operador de nivel n
    deslee_caracter(c);
    return 0;
  }

  return c;
}

//
// lee signos + o - consecutivos y calcula el signo resultante
//
int lee_signo(){
  int signo = 1;
  char c;
  while( 0 != (c=lee_operador(0)) ){
    if(c=='-')
      signo= signo*(-1);
  }
    return signo;
}

//
// lee un operando para una operación de nivel n
//
double lee_operando(int n) throws ParserFuncionException
{
  int signo = lee_signo();

  // un operando de nivel n puede estar formado por
  // a) una serie de operandos de nivel n+1 separados por
  //    un operador de nivel n+1 ó por
  // b) un dato evaluable.
  if( n<2 )
  // para niveles 0 y 1 se sube el nivel RECURSIVAMENTE
    return signo * lee_serie(n+1);

  // para nivel 2 (el más alto)
  double e = lee_evaluable();
  return signo * e;
}
```

```
  //
  // evalua la operación binaria op con los operandos a y b
  //
  double evaluacion( double a, char op, double b ) throws
ParserFuncionException
  {
    switch( op )
    {
      default:
      case '+': return a+b;
      case '-': return a-b;
      case '*': return a*b;
      case '/': {if( b == 0 ) ERROR("División entre cero");
                 return a/b;
                }
      case '^': return Math.pow(a,b);
    }
  }

  //
  // Lee y evalua una serie de operandos intercalados por un operador
  // Las operaciones se evalúan de izquierda a derecha
  //
  double lee_serie(int n) throws ParserFuncionException
  {
    // por lo menos debe haber un operando
    double valor = lee_operando(n);

    char operador;
    // mientras se lea un operador de nivel n
    while( 0 != (operador = lee_operador(n)) )
    {
      // se acumula en "valor" el resultado de
      // operar "valor" con el siguiente operando
      double op2 = lee_operando(n);
      valor = evaluacion( valor, operador, op2 );
    }
    // ya no hay más operadores de nivel n
    return valor;
  }

  //
  // lee y evalúa una expresión
  //
  double lee_expresion() throws ParserFuncionException
  {
    // una expresión es una serie de operandos de nivel 0 (términos)
    // separados por un operador de nivel 0
    return lee_serie(0);
  }
```

```
  ////////////////////////////////////////////////////////////////////////
  //
  //  Constructor
  //
  ////////////////////////////////////////////////////////////////////////
  public ParserFuncion2(String expr){
    indice=0;
    x=0;
    y=0;
    expresion=new String(expr.toUpperCase());
  }


  ////////////////////////////////////////////////////////////////////////
  //
  // Evaluar con el argumento (x,y)
  //
  ////////////////////////////////////////////////////////////////////////
  public double Evaluar(double X, double Y) throws ParserFuncionException
{
    x = X;
    y = Y;
    indice=0;

    try
    {
      double valor = lee_expresion();
      if( lee_caracter() != 0 ){
        ERROR("No se termino de evaluar la expresion");
      }
      return valor;
    }
    catch(ParserFuncionException e)
    {
      String mensaje= new String("ERROR: posicion:" + e.posicion +" -
                                                                  "+e.C);
      ParserFuncionException Excpt=new ParserFuncionException(mensaje,
                                                                  0);
      throw(Excpt);
    }
  } // fin de Evaluar
} // fin de clase ParserFuncion
```

```
package metonume;

public class ParserFuncionException extends Exception{
  public String C;
  int posicion;

  public
  ParserFuncionException(String mensaje, int i){
    C=mensaje;
    posicion= i;
  }//fin de constructor
}//fin de class ParserFuncionException
```

REFERENCIAS.

1.- [Atkinson,1987].-Atkinson y Harley, *Introducción a los métodos numéricos con PASCAL*, Ed. SITESA, Addison-Wesley Iberoamericana, © México 1987.

2.- [Becerril,1998].-Becerril Francisco, *JAVA a su alcance*, Ed. McGraw Hill, © 1998.

3.- [Becker,1996].-Becker, Data, *JAVA el lenguaje de programación en INTERNET*,, Ed. Marcombo, © 1996.

4.- [Burden,1998].- Burden L. Richard, Faires J. Douglas, *Análisis Numérico*, Ed. Thomson, © 1998 6a edición..

5.- [Ceballos,2002].-Ceballos, Fco. Javier., *El lenguaje de programación JAVA*, Ed. Alfaomega, Ra-Ma, © 2002.

6.- [Chan,Griffith&Iasi,1998].-Chan, Mark, Griffith Steven, Iasi, "*1001 tips para programar con JAVA*", Ed. McGraw-Hill, México, ©1997.

7.- [Chapra, 1999].- Chapra Steven, Canale Raymond, *Métodos numéricos para ingenieros*, Ed. McGraw.Hill , México, ©1999, 3ª edición.

8.- [Cowel,1999].-Cowel, *Essential JAVA 2 fast*, Ed. Springer, © 1999.

9.- [Curtis,1991].- Curtis, F.Gerald, *Análisis Numérico*, Ed. Alfaomega, ©1991 2ª edición.

10.- [Daconta,1996].-Daconta, Michael, *JAVA for C/C++ programmers*, Ed. Wiley Computer Publishing, © 1996.

11.- [Davis,1997].-Davis, Stephen R., *Aprenda JAVA ya*, Ed. Microsoft Press, ©España 1997.

12.- [De Amescua y otros,1997].-De Amescua Seco, Antonio, Garcia Sánchez Luis, Martinez Fernández Paloma, Díaz Pérez Paloma, *Ingeniería del software de Gestion*, "Análisis y Diseño de Aplicaciones", Ed. Paraninfo, ©España 1995.

13.- [Decker&Hirshfield, 2001].- Decker, Hirshfield,.- *Programación con JAVA*", Ed. Thomson, 2ª edición © Mex. 2001.

14.- [Deitel&Deitel,1998].- Deitel y Deitel, *Como programar en JAVA*", Ed. Pearson Education, ©1998.

15.- [Duane,1999].- Duane A. Bailey, *JAVA Structures*, Ed. McGraw-Hill, Singapore, ©1999.

16.- [Duane,2000].- Duane A. Bailey, *JAVA Elements*, Ed. McGraw-Hill, Singapore, ©2000.

17.- [Flanagan,1999].-Flanagan , David, *JAVA en pocas palabras*, McGraw-Hill., ©1999.

18.- [Froufe,2000].- Froufe, Agustín. *JAVA 2, Manual de usuario y tutorial de JAVA*, Ed. Alfaomega- RA-MA, 2ª edición, ©2000.

19.- [Gilb&Graham, 1998].- Gilb Tom, Graham Dorothy . *software Inspection*, Ed. Adison-Wesley, 4ª edición, © Inglaterra 1998.

20.- [Hubbard, 1998].-Hubbard, John, *Programing with JAVA*, McGraw-Hill., Schaum's outlines ©1998.

21.- [Iriarte,1990].- Iriarte V. Balderrama, Rafael, *Métodos Numéricos*, Trillas:UNAM, Facultad de Ingenieria ©1990.

22.- [James, 1973].- James, Merlin L., Smith, M. Gerald, Wolford, James C., *Métodos Numéricos Aplicados a la Computación Digital con FORTRAN*, Ed. Representaciones y Servcios de Ingeniería S.A., ©1973.

23.- [Jamsa, Kris, 2000] . *Aprenda y practique JAVA*, Ed. Oxford, México, © 2000.

24.- [Joyanes&Fernández, 2001].-Joyanes Aguilar Luis, Fernández Azuela Matilde, *JAVA 2 Manual de programación.*, Ed. McGraw Hill, Madrid, © 2001.

25.- [Kafura,2000].- Kafura, Dennis,. *Object Oriented software Design and construction with JAVA, " Web enhanced"*, Ed. Prentice Hall, ©2000.

26.-[Lemay&Cadenhead,1999].-Lemay, Laura and Cadenhead Rogers *Teach yourself JAVA 2 platform in 21 days*, Ed. Sams.net, © 1999.

27.- [Maron, 1995].- Maron, J. Melvin. *Análisis Numérico*, "Un enfoque práctico", Ed. CECSA, ©1995.

28.- [Marciniak,1994].- Marciniak, John (editor in chief), *Encyclopedia of software Engineering*, volume 1 A-N, Ed. Wiley & Sons Inc.., ©1994.

29.- [Nakamura, 1992].- Nakamura, Shoichiro. *Métodos Numéricos Aplicados con software*, Ed. Prentice Hall, ©1992.

30.- [Naughton, 1996]Naughton, Patrick. *Manual de JAVA* Ed. Osborne Mc Graw-Hill, ©España 1996.

31.- [Nieves, 1999]Nieves, Antonio, Dominguez Federico C., *Métodos numéricos aplicados a la ingeniería* Ed. CECSA, 5a edición, ©1999.

32.- [Norris,1994].-Norris Rigby, *Ingeniería del software explicada,* Ed. MEGABYTE., ©1994.

33.- [Pankak Jalote,1997].-Pankaj Jalote *An Integrated Approach to software Egineering* "Undergraduate Text in computer science", 2a Edición, Ed. Springer., ©1997.

34.- [Pressman,1998].-Pressman, Roger, *Ingeniería del software, un enfoque práctico*, 4ª Edición, Ed. McGraw-Hill., ©1998.

35.- [Reifer,1993].-Reifer, Donald, *software Management*., 4ª Edición, Ed. IEEE Computer Society, ©1998.

36.- [Scheid,1990].- Sheid, Francis, *Análisis Numérico*, serie Schaum, Ed. Mc Graw Hill, ©1991 2a edición.

37.- [Schildt,2000].- Schildt Herbert, *C Manual de referencia*, 4ª Edición, Ed. Mc Graw-Hill ., ©2000.

38.- [Schildt,1996].- Schildt Herbert, *Schildt's expert C++*", Ed. Mc Graw-Hill ., ©2000.

39.- [Smith,1988].- Smith, W. Allen. *Análisis Numérico*, Prentice- Hall; ©1998.

40.- [Sommerville,1989].-Sommerville, Ian, *software Engineering*", 3ª Edición, Ed. Addison-Wesley., ©1989.

41.- [Wehling, Bharat y otros, 1998].- Wehling Jason, Vidya Bharat y otros, *Aproveche las noches con JAVA*., Ed. Prentice Hall, ©1998.

42.- [Zill, 1982].- Zill, Dennis G., *Ecuaciones Diferenciales con aplicaciones*, Primer Curso. Ed. WadSoftwarcorth International/Iberoamérica, 1982.