IntegraLAB: Un software para integración de funciones y solución de ecuaciones diferenciales por métodos numéricos

Recepción: Agosto de 2006 / Aceptación: Noviembre de 2006

(1) Edgar Ruiz Lizama

RESUMEN

El trabajo presenta el diseño e implementación de un software que tiene por nombre IntegraLAB el cual sirve como una herramienta para resolver problemas de integración de funciones y solución de ecuaciones diferenciales ordinarias aplicando métodos numéricos. El software ha sido probado en cuanto a sus resultados tomando funciones cuyos resultados son conocidos y presentados en la literatura sobre métodos numéricos y luego contrastados con los obtenidos o devueltos por IntegraLAB, encontrado precisión y exactitud en conformidad con los resultados esperados.

Palabras Clave: Software para integración, métodos de integración numérica, graficación de funciones, java.

INTEGRALAB: A SOFTWARE FOR FUNCTIONS' INTEGRATION AND SOLUTION OF DIFFERENTIAL EQUATIONS THROUGH NUMERICAL METHODS.

ABSTRACT

This work presents the design and implementation of a software whose name is IntegraLAB, which can be used as a tool to solve functions' integration problems, and for the solution of ordinary differential equations through applying numerical methods. As for its results, the software has been tested taking functions whose results are known, and have been presented in literature about numerical methods, and then contrasted with those obtained or given back by IntegraLAB. Accuracy and exactness have been found according to the expected results.

Key words: Integration Software, numeric integration methods, functions graphitizing, java.

EL PROBLEMA

El problema es la creación de un software que sirva como herramienta para resolver problemas de integración de funciones y ecuaciones diferenciales ordinarias aplicando métodos numéricos.

SITUACIÓN ACTUAL

La aplicación de los métodos numéricos a problemas de ciencias e ingeniería empleando herramientas computacionales es significativa. Los productos software como MATLAB de MathWorks, MATCAD, EUREKA, SOLVER, y TOOLKIT son muy utilizados.

OBJETIVOS

Objetivo general

El objetivo general es escribir un software que aplique métodos numéricos en la solución de problemas de integración de funciones y ecuaciones diferenciales ordinarias.

Objetivos específicos

1. Escribir un software de calidad acorde con los principios de la Ingeniería de software.
2. Simplificar el empleo de un método numérico aplicado a la evaluación de la integral de una función o la solución de una ecuación diferencial ordinaria. Por ejemplo; utilizando una calculadora de mano, obtener una solución podría tomar algunos minutos. El software elaborado deberá obtener la solución en pocos segundos, reduciendo sustancialmente las tareas de cálculo o proceso.

JUSTIFICACIÓN DEL SOFTWARE

Los programas software que existen en el mercado son hechos por compañías internacionales, no existiendo productos similares desarrollados en el país. Por ello se asume el reto de escribir un software nacional, de calidad que sirva como herramienta en la resolución de problemas de integración de funciones y de ecuaciones diferenciales ordinarias, utilizando los algoritmos que proveen los métodos numéricos, las estructuras de datos y la ingeniería del software.

(1) Magíster en Informática. Profesor del Departamento de Ingeniería de Sistemas e Informática, UNMSM. E-mail: eruizl@unmsm.edu.pe

PERFIL DEL USUARIO

IntegraLAB es un software de propósito específico, dirigido a la solución de problemas ciencias e ingeniería que tengan que ver con el cálculo de integrales y resolución de ecuaciones diferenciales ordinarias. Por ello los usuarios del software deberán tener el siguiente perfil:

1. Conocimiento de los métodos numéricos y sus aplicaciones, lo cual implica también conocimiento básico del cálculo diferencial e integral.
2. Saber formular adecuadamente la ecuación o función que resuelve un problema.
3. Tener capacidad para el análisis e interpretación correcta de la solución que entregue el software.

ESTRATEGIA DE SOLUCIÓN

El entorno de desarrollo para el software es el modo gráfico utilizando: el lenguaje de programación Java, los conceptos de compiladores y sus algoritmos, las estructuras de datos y la utilización de los métodos numéricos para resolver la integración de funciones y la solución de ecuaciones diferenciales ordinarias.

CRITERIOS DE ACEPTACIÓN

1. Teniendo problemas y/o funciones conocidas se usarán sus resultados como testigos, para confrontarlos con los resultados que entregue el programa.
2. El tiempo de proceso de cálculo debe ser mínimo.
3. Facilidad y simplicidad de uso del software.

DISEÑO DEL SOFTWARE IntegraLAB

Para el diseño del software IntegraLAB, se consideran los principios de abstracción, modularidad, ocultamiento de la información, estructura y verificación.

El diseño de pantalla del menú principal (ver la figura 1) contiene las siguientes opciones: Archivo, Edición, Integración, EDO's y Ayuda.

El menú 1: **Archivo**; contiene las opciones: Nuevo, Abrir, Guardar, Salir.
El menú 2: **Edición**; contiene las opciones: Cortar, Copiar, Pegar y Colorear.
El menú 3: **Integración**; contiene las opciones: Newton-Cotes, Gauss-Legendre y Gauss-Laguerre.
El menú 4: **EDO's**; contiene las opciones: Básicas, y Sistemas EDO's.
El menú 5: **Ayuda**; contiene las opciones: Ayuda y Acerca de.

Al elegir la opción: Integración, Newton-Cotes se tiene la ventana que muestra la figura 2, donde se presenta la solución a la raíz cuadrada de x en el intervalo [0,4] por el método del trapecio compuesto con $n = 10$.

Si por el contrario se elige la opción Integración, Gauss-Legendre produce la salida mostrada en la figura 3.

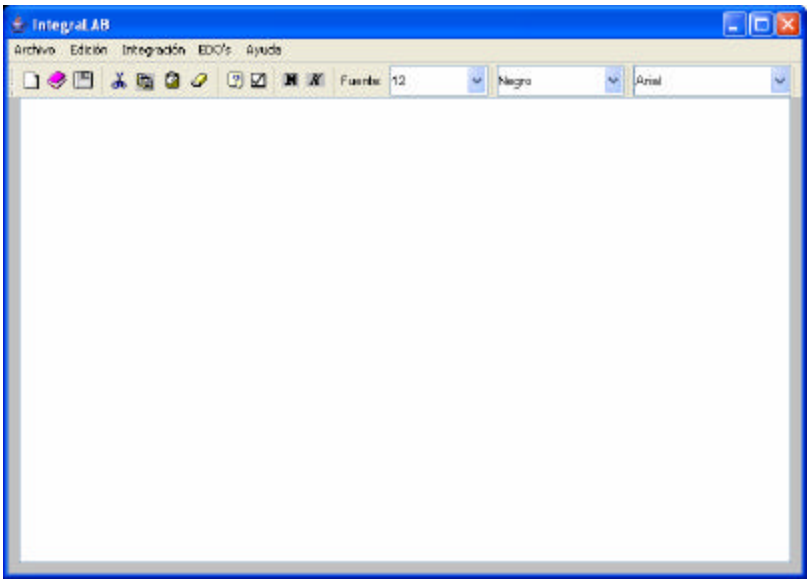

Figura 7. Ventana principal de IntegraLAB.

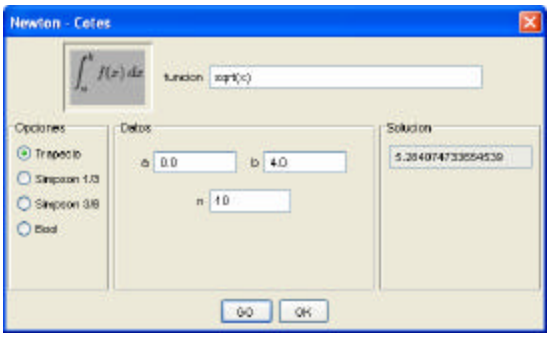

Figura 2. Menú Integración Newton-Cotes.

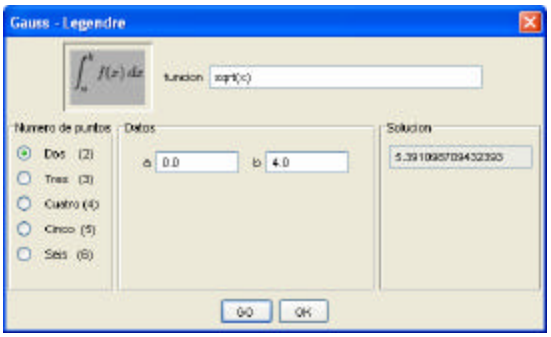

Figura 3. Menú Integración Gauss-Legendre.

El software IntegraLAB, posee un ambiente que permite visualizar la gráfica de una función. El archivo GraphDialog.java se encarga de graficar funciones. La figura 4 presenta una ejecución para la función sin(3.141459*x) en el intervalo [0,4].

IMPLEMENTACIÓN DEL SOFTWARE

IntegraLAB es un software que ha sido escrito en tres grandes clases a saber:

a) La clase Parser
b) La clase IntegraLAB y
c) La clase GraphDialog.

La clase Parser

Para evaluar expresiones, se hace uso de las técnicas utilizadas en el diseño de compiladores. La descripción completa de esta clase se encuentra en [10].

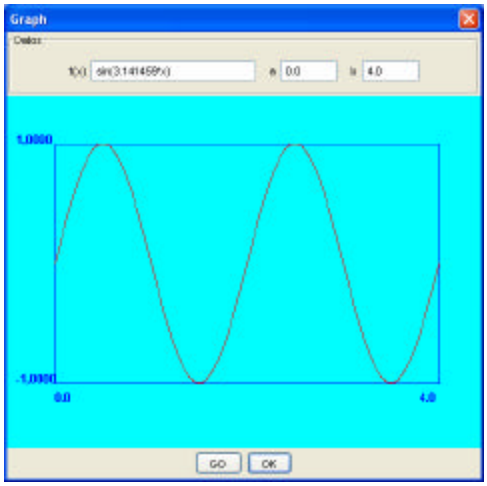

Figura 4. Graficando una función.

La clase IntegraLAB

La clase IntegraLAB permite elaborar la interfase de usuario GUI. Esta hace uso de los paquetes swing.*, awt.*, io.*, que Java posee. Con ellos se construye la Ventana de ejecución del software, junto con sus menús y las diversas opciones que han sido mostradas en el diseño del software IntegraLAB. Allí también se encuentra el código en Java de los métodos de integración y solución de ecuaciones diferenciales ordinarias que se utilizan.

La clase NewtonDialog

Esta clase permite presentar el cuadro de diálogo que permite insertar o introducir en cuadros de texto: la función a integrar, los límites de integración, número de intervalos. Adicionalmente permite escoger las opciones (Trapecio, Simpson 1/3, Simpson 3/8 y Boole), finalmente presenta la solución o respuesta encontrada por el algoritmo seleccionado en opciones.

La función miembro algor( ) se encuentra sobrecargada y permite la selección de los algoritmos numéricos, escritos para el software. En las figuras 5, 6, 7, 8 y 9; se presenta el código de cada uno de los métodos numéricos empleados por esta clase.

```
// Algoritmos numericos implementados
double algor(int metodo,double a,double b,int n)
{     double y=0;
      switch(metodo){
            case 1: y = trapecio(a,b,n); break;
            case 2: y = simpson13(a,b,n); break;
            case 3: y = simpson38(a,b,n); break;
            case 4: y = boole(a,b,n); break;
      }
      return y;
}
```

Figura 5. Función miembro algor.

```
private double trapecio(double a,double b,int n)
{     double h = (b-a)/n;
      double x;
      Parser p = new Parser(lexema);
      double suma = 0;
      for(int i=1;i<n;i++)
      {      x=a+i*h;
             suma=suma+p.getValue(x);
      }
      return h*0.5*(p.getValue(a)+2*suma+ p.getValue(b));
}
```

Figura 6. Función miembro para el método del trapecio.

```
    private double simpson13(double a,double b,int n)
     {     double h=(b-a)/n;
           double x;
           Parser p=new Parser(lexema);
           double suma=0;
           for(int i=1;i<n;i++)
           {      x=a+i*h;
                  if(i%2==0)
                         suma+=2*p.getValue(x);
                  else
                         suma+=4*p.getValue(x);
           }
           return h*(p.getValue(a)+suma + p.getValue(b))/3;
     }
```

Figura 7. Función miembro para el método de Simpson 1/3.

La clase LegendreDialog

Esta clase en cuanto al cuadro de diálogo que presenta al ser seleccionada, es idéntico al cuadro de diálogo presentado por la clase newtonDialog; pero se diferencia en que, el frame para las "opciones" o métodos, es titulado ahora "número de puntos" (dos, tres, cuatro, cinco y seis) acerca de los cuales se quiere tener en cuenta para los cálculos.

Con la finalidad de abreviar y no distraer al lector en la figura 10, se presenta la función miembro algor() de esta clase.

La clase laguerreDialog

Esta clase presenta un cuadro de diálogo similar al de la clase anterior. En la figura 11 se presenta el código que hace posible los cálculos.

La clase basicasDialog

Esta clase permite a IntegraLAB la solución de ecuaciones diferenciales ordinarias para el problema del valor inicial. En ella se escribe el código Java que permite mostrar el cuadro de dialogo correspondiente para el ingreso de la función a evaluar, las opciones (Euler, Heun, RK2, y RK4), asimismo los datos (el intervalo, el número de segmentos y el valor inicial), asimismo; al presionar el botón [Go] presenta una caja con la malla de puntos encontrados por el algoritmo elegido.

La función miembro algor( ), mostrada en la figura 12; permite la selección de uno de los algoritmos numéricos para resolver EDO's.

En las figuras 13, 14, 15 y 16; se presenta el código Java de los algoritmos de Euler, Heun, Runge Kutta 2, y Runge Kutta 3.

La clase sistemasDialog

Esta clase implementa la solución de un sistema de ecuaciones diferenciales ordinarias por el método de Runge Kutta RK4. En ella de modo similar que las clases anteriores se implementa el código Java necesario para mostrar el cuadro de diálogo que permite la solución de un sistema de dos ecuaciones diferenciales ordinarias.

El código de la función miembro algor( ), se presenta en la figura 17.

Las figuras 18 y 19, presentan una ejecución para un sistema de dos ecuaciones EDO's.

```
private double simpson38(double a,double b,int n)
 {     double h=(b-a)/n;
       double x;
       Parser p=new Parser(lexema);
       double suma=0;
       for(int i=1;i<n;i++)
       {      x=a+i*h;
              if(i%3==0)
                     suma+=2*p.getValue(x);
              else   suma+=3*p.getValue(x);
       }
       return 3*h*(p.getValue(a) + suma + p.getValue(b))/8;
 }
```

Figura 8. Función miembro para el método de Simpson 3/8.

```
private double boole(double a,double b,int n)
 {     double h=(b-a)/n;
       double x;
       Parser p=new Parser(lexema);
       double suma=0;
       for(int i = 1;i<n;i++)
       {      x = a + i*h;
              if(i%4 == 0)
                     suma+=14*p.getValue(x);
              else if(i%4 == 2)
                            suma+=12*p.getValue(x);
                     else   suma+=32*p.getValue(x);
       }
       return 4*h*(7*p.getValue(a)+suma+7*p.getValue(b))/90;
 }
```

Figura 9. Función miembro para el método de Boole.

```
double algor(double a,double b,int n)
{       double[][] x=new double[7][4];
        double[][] w=new double[7][4];
        x[2][1]=0.577350269189626;   w[2][1]=1.000000000000000;

         x[3][1]=0.000000000000000;   w[3][1]=0.888888888888888;
         x[3][2]=0.774596669241483;   w[3][2]=0.555555555555555;

         x[4][1]=0.339981043584856;   w[4][1]=0.652145154862546;
         x[4][2]=0.861136311594053;   w[4][2]=0.347854845137454;

         x[5][1]=0.000000000000000;   w[5][1]=0.568888888888889;
         x[5][2]=0.538469310105683;   w[5][2]=0.478628670599366;
         x[5][3]=0.906179845938664;   w[5][3]=0.236926885056189;

         x[6][1]=0.238619186083197;   w[6][1]=0.467913934572691;
         x[6][2]=0.661209386466265;   w[6][2]=0.360761573048139;
         x[6][3]=0.932469514203152;   w[6][3]=0.171324492379170;
        Parser p=new Parser(lexema);
        double suma;
        double y=0,c,d,z;
        c=0.5*(a+b);    d=0.5*(b-a);
        z=d*x[n][1];
        if(Math.floor(n/2)!= Math.floor((n+1)/2))
           suma=d*w[n][1]*p.getValue(z+c);
        else
           suma=d*w[n][1]*(p.getValue(-z+c)+ p.getValue(z+c));
        for(int i=2;i<=Math.floor((n+1)/2);i++){
           z=c*x[n][i];
           suma+=d*w[n][i]*(p.getValue( -z+c) + p.getValue(z+c));
        }
        return suma;
}
```

Figura 10. Función miembro algor ( ) para el método de Gauss-Legendre

```
double algor(double a,int n)
{     double[][] x=new double[7][7];
      double[][] w=new double[7][7];

      x[2][1]=0.585786437627;   w[2][1]=0.853553390593;
      x[2][2]=3.414213562373;   w[2][2]=0.146446609407;

      x[3][1]=0.415774556783;   w[3][1]=0.711093009929;
      x[3][2]=2.294280360279;   w[3][2]=0.278517733569;
      x[3][3]=6.289945082937;   w[3][3]=0.0103892565016;

      x[4][1]=0.322547689619;   w[4][1]=0.603154104342;
      x[4][2]=1.745761101158;   w[4][2]=0.357418692438;
       x[4][3]=4.536620296921;   w[4][3]=0.0388879085150;
      x[4][4]=9.395070912301;   w[4][4]=0.000539294705561;

     x[5][1]=0.263560319718;    w[5][1]=0.521755610583;
     x[5][2]=1.413403059107;    w[5][2]=0.398666811083;
     x[5][3]=3.596425771041;    w[5][3]=0.0759424496817;
     x[5][4]=7.085810005859;    w[5][4]=0.00361175867992;
     x[5][5]=12.640800844276;   w[5][5]=0.0000233699723858;

     x[6][1]=0.222846604179;    w[6][1]=0.458964673950;
     x[6][2]=1.188932101673;    w[6][2]=0.417000830772;
     x[6][3]=2.992736326059;    w[6][3]=0.113373382074;
     x[6][4]=5.775143569105;    w[6][4]=0.0103991974531;
     x[6][5]=9.837467418383;    w[6][5]=0.000261017202815;
     x[6][6]=15.982073980602;   w[6][6]=0.000000898547906430;
      Parser p = new Parser(lexema);
      double suma=0,z;
      for(int i=1;i<=n;i++){
            z = x[n][i] + a;
            suma += w[n][i]*p.getValue(z);
      }
      return suma*Math.exp(-a);
 }
```

Figura 11. Función miembro algor( ) para el método de Gauss-Laguerre

```
double algor(int metodo,double a,double b,double y0,int n)
 {     double y=0;
       switch(metodo){
             case 1:y = euler(a,b,y0,n); break;
             case 2:y = heun(a,b,y0,n); break;
             case 3:y = rk2(a,b,y0,n) ; break;
             case 4:y = rk4(a,b,y0,n) ; break;
       }
       return y;
 }
```

Figura 12. Función miembro algor ( ) para los métodos de EDO de primer orden.

```
private double euler(double a,double b,double y0,int n)
 {     double h=(b-a)/n;
       double x=a,y=y0;
       Parser p=new Parser(lexema);
       arreglo=new String[n+1];
       arreglo[0]=decimales2.format(x)+"   " + decimales5.format(y);
       for(int i=1;i<=n;i++)
       {      y+=h*p.getValue(x,y);
              x+=h;
              arreglo[i]=decimales2.format(x)+"   "+ decimales5.format(y);
       }
       return y;
 }
```

Figura 13. Función miembro para el método de Euler.

```
private double heun(double a,double b,double y0,int n)
 {     double h=(b-a)/n;
       double x=a,y=y0,pred;
       Parser p=new Parser(lexema);
       arreglo=new String[n+1];
       arreglo[0]=decimales2.format(x)+"   "+ decimales5.format(y);
       for(int i=1;i<=n;i++)
       {      pred=y+h*p.getValue(x,y);
              y+=0.5*h*(p.getValue(x,y) + p.getValue(x+h,pred));
              x+=h;
              arreglo[i]=decimales2.format(x)+"   "+ decimales5.format(y);
       }
       return y;
 }
```

Figura 14. Función miembro para el método de Heun.

```
private double rk2(double a,double b,double y0,int n)
 {
       double h=(b-a)/n;
       double x=a,y=y0;
       double k1,k2;
       Parser p=new Parser(lexema);
       arreglo=new String[n+1];
       arreglo[0]=decimales2.format(x)+"   "+ decimales5.format(y);
       for(int i=1;i<=n;i++)
       {      k1=p.getValue(x,y);
              k2=p.getValue(x+h/2,y+k1*h/2);
              y=y+h*k2;
              x+=h;
              arreglo[i]=decimales2.format(x)+"   " + decimales5.format(y);
       }
       return y;
 }
```

Figura 15. Función miembro para el método RK2.

La clase GraphDialog

Esta clase permite a IntegraLAB graficar curvas mediante la presentación del cuadro de diálogo y el trazo de la función en un objeto a la clase GraphPanel, denominado panel. La figura 20 presenta el diagrama de objetos de la clase GraphDialog.

La clase GraphDialog está compuesta por los siguientes campos: lexema, limiteA, limiteB y panel. Al respecto ver la sección de código mostrada en la figura 21.

En IntegraLAB, por defecto al ejecutar la clase GraphDialog mediante la secuencia [Ctrl] + [G] se presenta la ventana de la figura 4, donde se traza la función sin(3.141459*x), en un intervalo [0, 4].

El constructor de GraphDialog

Las tareas que realiza el constructor de la clase GraphDialog son:

1. Insertar el objeto dentro del marco.
2. Declarar campos de texto, botones, y rótulos.
3. Crear el objeto de gráfico o panel.
4. Declarar los diversos listener.

Los puntos del 1 al 3, son presentados en el código de las figuras 21 y 22.

La clase GraphPanel

Esta clase permite a IntegraLAB, la presentación del gráfico en pantalla, haciendo uso de la clase base JPanel.

```
private double rk4(double a,double b,double y0,int n)
 {     double h=(b-a)/n;
       double x=a,y=y0;
       double k1,k2,k3,k4;
       Parser p=new Parser(lexema);
       arreglo=new String[n+1];
       arreglo[0]=decimales2.format(x)+"\t"+ decimales5.format(y);
       for(int i=1;i<=n;i++)
       {      k1=p.getValue(x,y);
              k2=p.getValue(x+h/2,y+k1*h/2);
              k3=p.getValue(x+h/2,y+k2*h/2);
              k4=p.getValue(x+h,y+k3*h);
              y=y+h*(k1+2*k2+2*k3+k4)/6;
              x+=h;
              arreglo[i]=decimales2.format(x)+"   " + decimales5.format(y);
       }
       return y;
}
```

Figura 16. Función miembro para el método RK4.

```
double algor(double a,double b,int n,double y0,double z0)
 {     // RK4 para sistema de ecuaciones EDO
       double h=(b-a)/n;
       double x=a,y=y0,z=z0;
       double k1,k2,k3,k4;
       double q1,q2,q3,q4;
       Parser p1 = new Parser(lexema1);
       Parser p2 = new Parser(lexema2);
       arreglo = new String[n+1];
       arreglo[0]=decimales2.format(x)+"   "+
                  decimales5.format(y)+"   "+ decimales5.format(z);
       for(int i=1;i<=n;i++)
       {      k1=p1.getValue(x,y,z);
              q1=p2.getValue(x,y,z);
              k2=p1.getValue(x+h/2,y+k1*h/2,z+q1*h/2);
              q2=p2.getValue(x+h/2,y+k1*h/2,z+q1*h/2);
              k3=p1.getValue(x+h/2,y+k2*h/2,z+q2*h/2);
              q3=p2.getValue(x+h/2,y+k2*h/2,z+q2*h/2);
              k4=p1.getValue(x+h,y+k3*h,z+q3*h);
              q4=p2.getValue(x+h,y+k3*h,z+q3*h);
              y=y+h*(k1+2*k2+2*k3+k4)/6;
              z=z+h*(q1+2*q2+2*q3+q4)/6;
              x+=h;
              arreglo[i]=decimales2.format(x)+"   "+
                      decimales5.format(y)+"   "+ decimales5.format(z);
       }
       return y;
 }
```

Figura 17. Función miembro algor, para el sistema de ecuaciones diferenciales ordinarias.

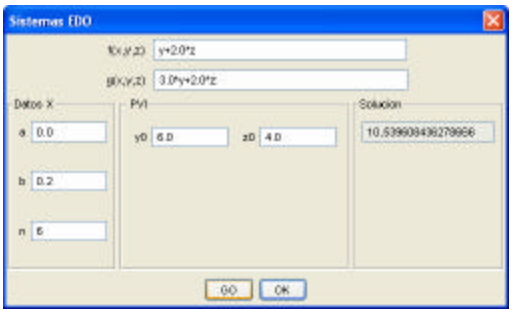

Figura 18. Un sistema de dos Ecuaciones EDO's.

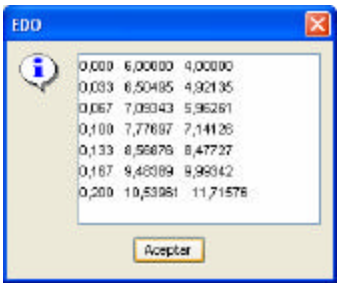

Figura 19. Solución reportada por IntegraLAB para el sistema EDO de la figura 18.

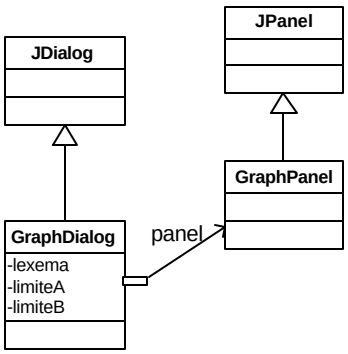

Figura 20. Diagrama UML para la clase Graphdialog

```
//Graficando curvas
import javax.swing.*;
import javax.swing.border.*;
import java.awt.event.*;
import java.awt.*;
import java.awt.geom.*;
import java.util.*;
import java.text.DecimalFormat;
class GraphDialog extends JDialog
{
       String lexema = "sin(3.141459*x)";
       double limiteA = 0;
       double limiteB = 4;
       GraphPanel panel;
       . . .  // continua
```

Figura 21. Sección de código de la clase GraphDialog

El constructor de GraphPanel permite especificar los elementos del gráfico, tales como, cadena a evaluar y los limites del gráfico; e invocar al método graph( ) para construir el gráfico. La figura 23 presenta este constructor.

El método graph( ), permite construir el gráfico en el objeto g a Graphics; su algoritmo se presenta en el pseudo código de la figura 24.

PRUEBAS DEL SOFTWARE

Las pruebas son de suma importancia para todo proyecto software y permiten observar si los resultados o respuestas entregados por el software son o no los esperados o correctos.

Las pruebas hechas a IntegraLAB, se realizan para

```
public GraphDialog(Frame owner)
{            super(owner, "Graph", true);
             JTextField funcionText;
             JTextField aTest;
             JTextField bTest;
             final JButton goOK;
             final JButton btOK;
             panel=new GraphPanel(lexema,limiteA,limiteB);
             getContentPane().add(panel, BorderLayout.CENTER);
             panel.setBackground(Color.cyan);
             JLabel lbl = new JLabel();
             JPanel p = new JPanel();
             Border b1 = new BevelBorder(BevelBorder.LOWERED);
             Border b2 = new EmptyBorder(5, 5, 5, 5);
             lbl.setBorder(new CompoundBorder(b1, b2));
             p.add(lbl);
             JPanel p3=new JPanel();
             p3.setBorder(new TitledBorder(new EtchedBorder(),"Datos"));
             JPanel p1=new JPanel();
             p1.add(new JLabel("f(x)"));
             // . . . continua
```

Figura 22. Tareas 1, 2 y 3 del constructor GraphDialog

```
// panel para el grafico o curva
class GraphPanel extends JPanel
{     private Point2D last;
      private ArrayList lines;
      private ArrayList points;
      private double scalaX, scalaY, max, min;
      private String lexema;
      private double a, b;
      private DecimalFormat decimales4 =
               new DecimalFormat("0.0000");
      private static final double Dx = 0.005;
      private static final double X0 = 50,Y0 = 50;
      private static final double ANCHO = 400, ALTO = 250;
      public GraphPanel(String lexema, double a,double b)
      {     this.lexema = lexema;
            this.a = a;
            this.b = b;
            graph();
      }
```

Figura 23. La clase GraphPanel y su constructor

```
Algoritmo graph ( )
            Computar puntos del grafico
            Computar escala para X y para Y
            Determinar el primer punto del grafico
            Graficar todos los puntos en modalidad animada en el panel.
End graph( )
```

Figura 24. Pseudo código para graph( )

cada método numérico implementado; pero por cuestiones de espacio solo se presentan dos de integración y dos de ecuaciones diferenciales ordinarias. Para ello se emplean funciones tomadas de las referencias como funciones testigo y luego se comparan estos resultados con los entregados por IntegraLAB.

MÉTODOS DE INTEGRACIÓN

Prueba para el método del Trapecio

Para probar este método se toma como función testigo, el ejemplo 4.1 [Nakamura S., Pág. 111-112]. Al ejecutar IntegraLAB, y escoger la opción integración, Newton-Cotes, y luego el método del trapecio para

Ejemplo 4.1: El cuerpo de revolución que se muestra en la figura 25, se obtiene al girar la curva dada por $y = 1 + (x/2)^2, 0 \le x \le 2$, en torno al eje *x*. Calcule el volumen utilizando la regla extendida del trapecio con *N* = 2, 4, 8, 16, 32, 64, 128. El valor exacto es *I* = 11.7286. Evaluar el error para cada *N*.

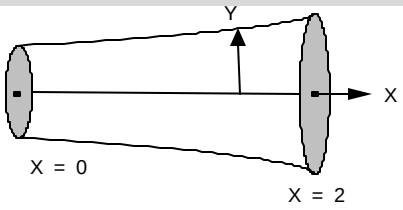

Figura 25: Un cuerpo de revolución

(Solución)

El volumen está dado por

$$I = \int_0^2 f(x)dx$$

Donde,

$$f(x) = \pi\left(1 + \left(\frac{x}{2}\right)^2\right)^2$$

A continuación aparecen los cálculos para N = 2 y 4:

N = 2: $h = 2/2 = 1$

$$I = \frac{1}{2}[f(0) + 2f(1) + f(2)] = 0.5\pi[1 + 2(1.5625) + 4]$$

$$= 12.7627$$

N = 4: $h = 2/4 = 0.5$

$$I = \frac{0.5}{2}[f(0) + 2f(0.5) + 2f(1) + 2f(1.5) + f(2)]$$

$$= 11.9895$$

Las integraciones con los demás valores de **N** se presentan en la tabla 1

Tabla 1: Resultados de aplicar el método del trapecio

N	***h***	I_h	E_h
2	1.	12.7627	-1.0341
4	0.5	11.9895	-0.2609
8	0.25	11.7940	-0.0654
16	0.125	11.7449	-0.0163
32	0.0625	11.7326	-0.0040
64	0.03125	11.7296	-0.0010
128	0.015625	11.7288	-0.0002

Valor exacto 11.7286

Error absoluto E_h

Se puede observar que el error decrece en forma proporcional a h^2.

Cuadro 1. Resultados de IntegraLAB* para el método del trapecio

N	*H*	I_h	E_h
2	1.	12.7627	-1.0341
4	0.5	11.9896	-0.2610
8	0.25	11.7940	-0.0654
16	0.125	11.7450	-0.0164
32	0.0625	11.7327	-0.0041
64	0.03125	11.7296	-0.0010
128	0.015625	11.7289	-0.0003

(*) Se han redondeado los resultados a cuatro dígitos significativos

la función 3.14159*(1+(x/2)^2)^2; en un intervalo de [0,2], con tamaños de N diferentes en cada corrida; se tiene el cuadro 1. Como se observa los resultados entregados por el software son los esperados y las pequeñas discrepancias se atribuyen a los efectos producidos por los errores de redondeo.

Prueba para Simpson 3/8 y la Regla de Boole

Para probar Simpson 3/8 se toma el Ejemplo 7.2 [Mathews, John; pag.377]

Ejemplo 7.2. Queremos integrar la función $f(x)=1+e^{-x}sen(4x)$ en el intervalo [a,b]=[0,1]. Para ello vamos aplicar las fórmulas de las regla del trapecio, regla de Simpson 1/3, regla de Simpson 3/8 y la regla de Boole.

(Solución)

Para la regla del trapecio tenemos $h=1$ y el resultado es

$$\int_0^1 f(x)dx \approx \frac{1}{2}(f(0)+f(1))$$

$$=\frac{1}{2}(1.00000+0.72159)=0.86079$$

Para la regla de Simpson tenemos $h=1/2$ y el resultado es

$$\int_0^1 f(x)dx \approx \frac{1/2}{3}(f(0)+4f(\frac{1}{2})+f(1))$$

$$=\frac{1}{6}(1.00000+4(1.55152)+0.72159)=1.32128$$

Para la regla de Simpson 3/8 tenemos $h=1/3$ y el resultado es

$$\int_0^1 f(x)dx \approx \frac{3(1/3)}{8}(f(0)+3f(\frac{1}{3})+3f(\frac{2}{3})+f(1))$$

$$=\frac{1}{8}(1.00000+3(1.69642)+3(1.23447)+0.72159)=1.31440$$

Para la regla de Boole tenemos $h=1/4$ y el resultado es

$$\int_0^1 f(x)dx \approx \frac{2(1/4)}{45}(7f(0)+32f(\frac{1}{4})+12f(\frac{1}{2})+32f(\frac{3}{4})+7f(1))$$

$$=\frac{1}{90}(7(1.00000)+32(1.65534)+12(1.55152)$$

$$+32(1.06666)+7(0.72159)=1.30859$$

El valor exacto de esta integral definida es

$$\int f(x)dx=\frac{21e-4\cos(4)-sen(4)}{17e}=1.3082506046426...$$

Así que la aproximación 1.30859 dada por la regla de Boole es la mejor.

Cuadro 2. Resultados de IntegraLAB para $f(x)=1+e^{-x}sen(4x)$

Método numérico	Resultado I_h	E_h
Trapecio $h=1,\ n=1$	0.8607939604744832	0.4474566441681168
Simpson a 1/3 $h=1/2,\ n=2$	1.3212758322698817	-0.0130252276272817
Simpson a 3/8 $h=1/3,\ n=3$	1.3143968149336276	-0.0031462102910276
Boole $h=1/4,\ n=4$	1.3085919215646966	-0.0003368611004366

Los resultados al ejecutar la función 1+eXp(0-x)*sin(4*x) en el software IntegraLAB se presentan en el cuadro 2.

Se observa que los resultados son los esperados y similares a los de la función testigo.

MÉTODOS DE SOLUCIÓN DE ECUACIONES DIFERENCIALES ORDINARIAS

Prueba para el método de Heun

Para probar el método de Heun se utiliza el ejemplo 9.6 [Mathews, John; pag.484]. La función ingresada

Ejemplo 9.6: Vamos a usar el método de Heun para resolver el problema

$$y' = \frac{t-y}{2} \text{ en } [0,3] \text{con } y(0)=1$$

y a comparar las soluciones obtenidas con $h=1,\frac{1}{2},\frac{1}{4},$ y $\frac{1}{8}$.

En la figura 26, se muestran las gráficas de las dos primeras soluciones dadas por el método de Heun y la gráfica de la solución exacta $y(t)=3e^{-t/2}-2+t$. En la tabla 3 se recogen los valores de las cuatro soluciones en algunos nodos. Un cálculo típico, que hacemos con el tamaño de paso $h=0.25$, sería

$$f(t_0,y_0)=\frac{0-1}{2}=-0.5, \qquad p_1=1.0+0.25(-0.5)=0.875$$

$$f(t_1,p_1)=\frac{0.25-0.875}{2}=-0.3125,$$

$$y_1=1.0+0.125(-0.5-0.3125)=0.8984375$$

La iteración continúa hasta que lleguemos al último paso

$$y(3)\approx y_{12}=1.511508+0.125(0.619246+0.666840)=1.672269$$

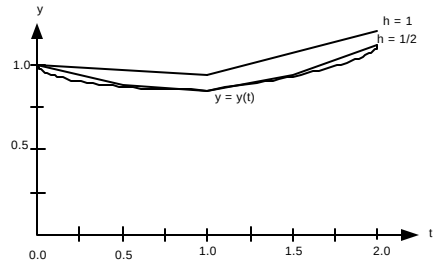

Figura 26: Comparación de las soluciones obtenidas con el método de Heun con diferentes tamaños de paso para $y'=(t-y)/2$ en $[0,3]$ con la condición inicial $y(0)=1$.

Tabla 3. Comparación de las soluciones obtenidas con el método de Heun con diferentes tamaños de paso para $y' = (t-y)/2$ en $[0,2]$ con la condición inicial $y(0)=1$.

t_k	y_k				$y(t_k)$
	$h=1$	$h=1/2$	$h=1/4$	$h=1/8$	Exacto
0.0	1.0	1.0	1.0	1.0	1.0
0.125				0.943359	0.943239
0.25			0.898438	0.897717	0.897491
0.375				0.862406	0.862087
0.50		0.84375	0.838074	0.836801	0.836402
0.75			0.814081	0.812395	0.811868
1.00	0.87500	0.831055	0.822196	0.820213	0.819592
1.50		0.930511	0.920143	0.917825	0.917100
2.00	1.171875	1.117587	1.106800	1.104392	1.103638
2.50		1.373115	1.362593	1.360248	1.359514
3.00	1.732422	1.682121	1.672269	1.670076	1.669390

a IntegraLAB para esta prueba es (x-y)/2.

El cuadro 3 presenta los resultados del software. Nuevamente observamos que los resultados entregados por IntegraLAB son los esperados.

La función ingresada a IntegraLAB para esta prueba es: –y +x^2+1; con a = 0.0, b = 2.0, y el valor inicial ción desde la línea de entrada.

Prueba para el método de Runge-Kutta RK4

Para probar Runge-Kutta de orden cuatro su utiliza el ejemplo 3 [BURDEN Richard L. y FAIRES J. Douglas; pag 278-279].

La función ingresada a IntegraLAB para esta prueba es: –y +x^2+1; con a = 0.0, b = 2.0, y el valor inicial ción desde la línea de entrada. Nuevamente los resultados presentados en el cuadro 4 son los esperados.

CONCLUSIONES Y RECOMENDACIONES

IntegraLAB es un software desarrollado en JAVA lo que permite independencia de la plataforma; por lo que puede ejecutarse en sistemas operativos distintos. Utiliza un parser para el manejo de funciones, de modo tal que cada función a evaluar es analizada desde la línea de entrada.

Cuadro 3. Resultados de IntegraLAB para evaluar por Heun con diferentes tamaños de paso para $y' = (t-y)/2$ en [0,3] con la condición inicial $y(0)=1$.

t_k	y_k				$y(t_k)$
	$h=1$	$h=1/2$	$h=1/4$	$h=1/8$	Exacto
0.0	1.0	1.0	1.0	1.0	1.0
0.125				0.94336	0.943239
0.25			0.89844	0.89772	0.897491
0.375				0.86241	0.862087
0.50		0.84375	0.83807	0.83680	0.836402
0.75			0.81408	0.81240	0.811868
1.00	0.87500	0.83105	0.82220	0.82021	0.819592
1.50		0.93051	0.92014	0.91783	0.917100
2.00	1.17188	1.11759	1.10680	1.10439	1.103638
2.50		1.37311	1.36259	1.36025	1.359514
3.00	1.73242	1.68212	1.67227	1.67008	1.669390

Ejemplo 3: Al aplicar el método de Runge-Kutta de orden cuatro para obtener aproximaciones a la solución del problema de valor inicial

$$y' = y - t^2 + 1, \quad 0 \le t \le 2, \qquad y(0) = 0.5,$$

con $h = 0.2$, $N = 10$ y $t_i = 0.2_i$ obtenemos los resultados y los errores que se proporcionan en la tabla 5.

Tabla 5: Resultados de evaluar $y' = y - t^2 + 1, \quad 0 \le t \le 2, \qquad y(0) = 0.5,$ con $h = 0.2$, $N = 10$ y $t_i = 0.2_i$

t_i	Valores exactos $y_i = y(t_i)$	Método de Runge-Kutta de orden cuatro w_i	Error $\lvert y_i - w_i \rvert$
0.0	0.5000000	0.50000000	0
0.2	0.8292986	0.8292933	0.0000053
0.4	1.2140877	1.2140762	0.0000114
0.6	1.6489406	1.6489220	0.0000186
0.8	2.1272295	2.1272027	0.0000269
1.0	2.6408591	2.6408227	0.0000364
1.2	3.1799415	3.1798942	0.0000474
1.4	3.7324000	3.7323401	0.0000599
1.6	4.2834838	4.2834095	0.0000743
1.8	4.8151763	4.8150857	0.0000906
2.0	5.3054720	5.3053630	0.0001089

Cuadro 4. Resultados de evaluar $y' = y - t^2 + 1, \quad 0 \le t \le 2, \qquad y(0) = 0.5,$ utilizando IntegraLAB.

t_i	Valores exactos $y_i = y(t_i)$	Método de Runge-Kutta de orden cuatro w_i	Error $\lvert y_i - w_i \rvert$
0.0	0.5000000	0.50000000	0
0.2	0.8292986	0.8292933	0.0000053
0.4	1.2140877	1.2140762	0.0000114
0.6	1.6489406	1.6489220	0.0000186
0.8	2.1272295	2.1272027	0.0000269
1.0	2.6408591	2.6408227	0.0000364
1.2	3.1799415	3.1798942	0.0000474
1.4	3.7324000	3.7323401	0.0000599
1.6	4.2834838	4.2834095	0.0000743
1.8	4.8151763	4.8150857	0.0000906
2.0	5.3054720	5.3053630	0.0001089

Se utiliza la GUI de Java Swing, para lograr un diseño amistoso al usuario. IntegraLAB posee un ambiente de despliegue gráfico el mismo que permite graficar una función desde la línea de entrada.

Las pruebas realizadas para el software satisfacen los requerimientos y objetivos previstos en su concepción.

El desarrollo de IntegraLAB es una primera versión por tanto es susceptible de mejoras en un próxima versión.

El alfabeto considerado en el parser para la evaluación de funciones contempla las variables x, y, z; pero podría ampliarse a otros caracteres alfabéticos para que reconozca como variables, por ejemplo r, s, t; o cualquier otro carácter alfabético o incluso una cadena de caracteres.

El gráfico de funciones se realiza en el plano y para una única función a la vez. Podría rescribirse la clase graphDialog, para graficar en un mismo lienzo dos funciones. Esto sería muy útil para visualizar el comportamiento de un sistema de ecuaciones diferenciales ordinarias con respecto al tiempo.

REFERENCIAS BIBLIOGRÁFICAS

1. Burden R. L. y Faires J. D. (2002). Análisis Numérico. 7ma. edición, International Thomsom Editores, S.A. de México, 839pp.

2. Chapman S. J. (2004). JavaTM for Engineers and Scientists. First edition, Pearson Education Inc. U.S.A. 676pp.

3. Chapra S. & Canale R. (1999) Métodos Numéricos para Ingenieros. 4ta. edición, Editorial McGrawHill, México, 992pp.

4. Deitel H. y Deitel P. (2003). JavaTM How To Program. Fifth Edition, Pearson Education Inc. U.S.A. 1536pp.

5. Mathews J. H. y Fink K. D. (2000). Métodos Numéricos con Matlab. 3ra. edición, Prentice-Hall Iberia S.R.L. Madrid 721pp.

6. Nakamura S. (1992). Métodos Numéricos Aplicados con Software. 1ra. edición, Prentice-Hall Hispanoamericana, S.A. de México 570pp.

7. Palmer G. (2003). Technical JavaTM: Developing Scientific and Engineering Applications. First edition, Pearson Education Inc. U.S.A. 496pp.

8. Press, W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P. (2002). Numerical Recipes in C++ The art of Scientific Computing. Second Edition, Cambridge University Press UK. 1002pp.

9. Pressman R. (2002). Ingeniería de Software Un enfoque práctico. 5ta. edición, McGraw-Hill Hispanoamericana de España S.A. 601pp.

10. Ruiz E. y Raffo E. (2006). Una clase parser en Java para evaluar expresiones algebraicas. Industrial Data Vol. 9, Nro.1: 85-96.

11. Sommerville I. (2002). Ingeniería de Software. 6ta edición, Adisson Wesley - Pearson Educación de México S.A. 712pp.

12. Weiss M. A. (2000). Estructura de Datos en JavaTM. 1ra. edición, Pearson Educación, S.A. 740pp.