TESIS PUCP

PONTIFICIA **UNIVERSIDAD CATÓLICA** DEL PERÚ

Esta obra ha sido publicada bajo la licencia Creative Commons
Reconocimiento-No comercial-Compartir bajo la misma licencia 2.5 Perú.
Para ver una copia de dicha licencia, visite
http://creativecommons.org/licenses/by-nc-sa/2.5/pe/

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

TESIS PUCP

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
ESCUELA DE GRADUADOS

INTEGRALAB: UN SOFTWARE PARA INTEGRACIÓN DE FUNCIONES Y SOLUCIÓN DE ECUACIONES DIFERENCIALES POR MÉTODOS NUMÉRICOS

TESIS PARA OPTAR EL GRADO ACADÉMICO DE MAGÍSTER EN INFORMÁTICA

PRESENTADO POR

EDGAR CRUZ RUIZ LIZAMA

LIMA - PERÚ

2006

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

Dedicatoria

A mis pequeños Edgar Jr. y Marianella;
con todo mi amor.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Agradecimiento

Al Dr. Maynard Kong por su asesoría, dedicación y noble apoyo.

A todos mis profesores de la Maestría en Informática de la PUCP(Hoy Maestría en Ciencias de la Computación); por sus enseñanzas, su experiencia y su calidad profesional.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

INDICE DE CONTENIDO

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

APÉNDICE I

CÓDIGO JAVA PARA LA CLASE PARSER

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
/**
  * Autor :       E. Ruiz Lizama
  * Fecha :       24/6/2005
 */

// parser
import java.util.Stack;
import java.util.StringTokenizer;
import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStreamReader;

// Evaluator class interface: evaluate infix expressions
//
// CONSTRUCTION: with a String
//
// ******************PUBLIC OPERATIONS***********************
// long getValue( )      --> Return value of infix expression
// ******************ERRORS**********************************
// Some error checking is performed

public class Parser
{
    private static final int EOL     = 0;
    private static final int VALUE   = 1;
    private static final int OPAREN  = 2;
    private static final int CPAREN  = 3;
    private static final int EXP     = 4;
    private static final int MULT    = 5;
    private static final int DIV     = 6;
    private static final int PLUS    = 7;
    private static final int MINUS   = 8;
    private static final int FUNCT   = 9;

    private static String[] function=
    {
            "sqrt","sin",
            "cos","tan",
            "asin","acos",
            "atan","log",
            "floor","eXp"
    };

       private String string;
       private Stack opStack;       // Operator stack for conversion
       private Stack postfixStack;  // Stack for postfix machine
       private StringTokenizer str; // StringTokenizer stream

    private static class Precedence
    {
        public int inputSymbol;
        public int topOfStack;

        public Precedence( int inSymbol, int topSymbol )
        {
            inputSymbol = inSymbol;
            topOfStack  = topSymbol;
        }
    }

    // PrecTable matches order of Token enumeration
    private static Precedence [ ] precTable = new Precedence[ ]
    {
        new Precedence(   0, -1 ),   // EOL
        new Precedence(   0,  0 ),   // VALUE
        new Precedence( 100,  0 ),   // OPAREN
        new Precedence(   0, 99 ),   // CPAREN
        new Precedence(   6,  5 ),   // EXP
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
        new Precedence(   3,   4 ),   // MULT
        new Precedence(   3,   4 ),   // DIV
        new Precedence(   1,   2 ),   // PLUS
        new Precedence(   1,   2 ),   // MINUS
        new Precedence(   7,   6 )    // FUNCT
    };

    private static class Token
    {
        public Token( )
        {
            this( EOL );
        }

        public Token( int t )
        {
            this( t, 0 );
        }

        public Token( int t, double v )
        {
            type = t;
            value = v;
        }

        public int getType( )
        {
            return type;
        }

        public double getValue( )
        {
            return value;
        }

        private int type = EOL;
        private double value = 0;
    }

    private static class EvalTokenizer
    {
        public EvalTokenizer( StringTokenizer is,double x,
                              double y,double z)
        {
            str = is;
            equis=x;
            ye=y;
            zeta=z;
        }

        /**
         * Find the next token, skipping blanks, and return it.
         * For VALUE token, place the processed value in currentValue.
         * Print error message if input is unrecognized.
         */
        public Token getToken( )
        {
            double theValue;

            if(!str.hasMoreTokens( ))
                return new Token( );

            String s = str.nextToken();

            if( s.equals( " " ) ) return getToken( );
            if( s.equals( "^" ) ) return new Token(EXP);
            if( s.equals( "/" ) ) return new Token(DIV);
            if( s.equals( "*" ) ) return new Token(MULT);
```

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
            if( s.equals( "(" ) ) return new Token(OPAREN);
            if( s.equals( ")" ) ) return new Token(CPAREN);
            if( s.equals( "+" ) ) return new Token(PLUS);
            if( s.equals( "-" ) ) return new Token(MINUS);
            if( s.equals( "x" ) ) return new Token(VALUE,equis);
            if( s.equals( "y" ) ) return new Token(VALUE,ye);
            if( s.equals( "z" ) ) return new Token(VALUE,zeta);

                    if(Character.isLetter(s.charAt(0)))
            {
             int i=searchFunction(s);
             if(i>=0)
                    return new Token(FUNCT+i);
             else
             {
                    System.err.println( "Parse error" );
                    return new Token();
             }
            }

            try
            {
                    theValue = Double.valueOf(s).doubleValue();
            }
            catch( NumberFormatException e )
            {
                System.err.println( "Parse error" );
                return new Token( );
            }

            return new Token( VALUE, theValue );
        }
       public int searchFunction(String s)
       {
              for(int i=0;i<function.length;i++)
                 if(s.equals(function[i]))
                      return i;
              return -1;
        }

        private StringTokenizer str;
        private double equis;
        private double ye;
        private double zeta;
}

/**
 * Construct an evaluator object.
 * @param s the string containing the expression.
 */
public Parser( String s )
{
    opStack = new Stack( );
    postfixStack = new Stack( );
         string=unary2Binary(s);
    str = new StringTokenizer(string,"+*-/^()xyz ",true);

    opStack.push( new Integer( EOL ) );
}

// The only publicly visible routine
/**
 * Public routine that performs the evaluation.
 * Examine the  postfix machine to see if a single result is
 * left and if so, return it; otherwise print error.
 * @return the result.
 */
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
public double getValue(double x)
{
    return getValue(x,0,0);
}
public double getValue(double x,double y)
{
    return getValue(x,y,0);
}
public double getValue(double x,double y,double z)
{
    // for each call
    opStack = new Stack( );
    postfixStack = new Stack( );
    str = new StringTokenizer(string,"+*-/^()xyz ",true);
    opStack.push( new Integer( EOL ) );

    EvalTokenizer tok = new EvalTokenizer(str,x,y,z);
    Token lastToken;

    do
    {
        lastToken = tok.getToken( );
        processToken( lastToken );
    } while( lastToken.getType( ) != EOL );

    if( postfixStack.isEmpty( ) )
    {
        System.err.println( "Missing operand!" );
        return 0;
    }

    double theResult = postFixTopAndPop( );
    if( !postfixStack.isEmpty( ) )
        System.err.println( "Warning: missing operators!" );

    return theResult;
}
  /**
 * Internal method that unary to binary.
 */
  private String unary2Binary(String s)
  {
   int i;
   s = s.trim();
   if(s.charAt(0) == '-')
        s = "0.0"+s;
   while((i=s.indexOf("(-"))>=0)
        s = s.substring(0,i+1)+"0.0"+
        s.substring(i+1);
   return s;
}
/**
 * Internal method that hides type-casting.
 */
private double postFixTopAndPop( )
{
    return ((Double)(postfixStack.pop())).doubleValue( );
}

/**
 * Another internal method that hides type-casting.
 */
private int opStackTop( )
{
    return ( (Integer) ( opStack.peek( ) ) ).intValue( );
}
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
    /**
     * After a token is read, use operator precedence parsing
     * algorithm to process it; missing opening parentheses
     * are detected here.
     */
    private void processToken( Token lastToken )
    {
        int topOp;
        int lastType = lastToken.getType();

        switch(lastType)
        {
          case VALUE:
            postfixStack.push( new Double( lastToken.getValue( ) ) );
            return;

          case CPAREN:
            while( ( topOp = opStackTop( ) ) != OPAREN && topOp != EOL )
                binaryOp( topOp );
            if( topOp == OPAREN )
                opStack.pop( );  // Get rid of opening parentheseis
            else
                System.err.println( "Missing open parenthesis" );
            break;

          default:    // General operator case
            int last=(lastType>=FUNCT?FUNCT:lastType);
            while(precTable[last].inputSymbol<=

precTable[opStackTop()>=FUNCT?FUNCT:opStackTop()].topOfStack)
                binaryOp( opStackTop() );
            if( lastType != EOL )
                opStack.push( new Integer( lastType ) );
            break;
        }
    }

    /*
     * topAndPop the postfix machine stack; return the result.
     * If the stack is empty, print an error message.
     */
    private double getTop( )
    {
        if ( postfixStack.isEmpty( ) )
        {
            System.err.println( "Missing operand" );
            return 0;
        }
        return postFixTopAndPop( );
    }


    /**
     * Internal routine to compute x^n.
     */
    private static double pow( double x, double n )
    {
        if( x == 0 )
        {
            if( n == 0 )
                System.err.println( "0^0 is undefined" );
            return 0;
        }
        if( n < 0 )
        {
            System.err.println( "Negative exponent" );
            return 0;
        }
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
        if( n == 0 )
            return 1;
        if( n % 2 == 0 )
            return pow( x * x, n / 2  );
        else
            return Math.pow(x, n);
    }

    /**
     * Process an operator by taking two items off the postfix
     * stack, applying the operator, and pushing the result.
     * Print error if missing closing parenthesis or division by 0.
     */
    private void binaryOp( int topOp )
    {
        if( topOp == OPAREN )
        {
            System.err.println( "Unbalanced parentheses" );
            opStack.pop( );
            return;
        }
        if(topOp >= FUNCT )
        {
            double d=getTop();
            postfixStack.push(new Double(functionEval(topOp,d)));
            opStack.pop( );
            return;
        }
        double rhs = getTop( );
        double lhs = getTop( );

        if( topOp == EXP )
            postfixStack.push( new Double(pow(lhs,rhs) ) );
        else if( topOp == PLUS )
            postfixStack.push( new Double(lhs + rhs) );
        else if( topOp == MINUS )
            postfixStack.push( new Double(lhs - rhs) );
        else if( topOp == MULT )
            postfixStack.push( new Double(lhs * rhs) );
        else if( topOp == DIV )
            if( rhs != 0 )
                postfixStack.push( new Double(lhs / rhs) );
            else
            {
                System.err.println( "Division by zero" );
                postfixStack.push( new Double( lhs ) );
            }

        opStack.pop();
    }

    private double functionEval(int topOp,double d)
    {
       double y=0;
              switch (topOp) {
              case 9:
                     y=Math.sqrt(d);break;
              case 10:
                     y=Math.sin(d);break;
              case 11:
                     y=Math.cos(d);break;
              case 12:
                     y=Math.tan(d);break;
              case 13:
                     y=Math.asin(d);break;
              case 14:
                     y=Math.acos(d);break;
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
                case 15:
                       y=Math.atan(d);break;
                case 16:
                       y=Math.log(d);break;
                case 17:
                       y=Math.floor(d);break;
                case 18:
                       y=Math.exp(d);

            }
            return y;
        }

    }  // fin de Parser.java
```

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

APÉNDICE II

CÓDIGO JAVA PARA LA CLASE INTEGRALAB

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
/**
  *Autor : E. Ruiz Lizama
  *Fecha : 24/6/2005
  */
import javax.swing.*;
import javax.swing.border.*;
import java.awt.event.*;
import java.awt.*;
import java.io.*;
import java.text.DecimalFormat;

class integraLAB extends JFrame {
       private static final int WIDTH  =700;
       private static final int HEIGHT =500;
       private int tamaño=12, style=0, negrita=0, cursiva=0;
       private JTextArea areaTexto;
       private JLabel LFuente;
       private JButton cor,cop,peg,nue,gua,ab,acerca,ayuda,neg,cur;
       private JScrollPane scroll;
       private JComboBox tFuente,cFuente,tipoFuente;
       private JTextField nFuente;
       private Font Fuente;
       private Color colorValue[]={Color.black,
                           Color.blue,
                           Color.red,
                           Color.yellow,
                           Color.green,
                           Color.cyan};
       private String[] FontNames={"Arial","TimesRoman","Courier","Helvetica"};
    private KeyHandler listener;
       public integraLAB() {
              super("IntegraLAB");
       try {

       UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAnd
Feel");
              }
       catch (Exception e) {
       JOptionPane.showMessageDialog(null,"Error al intentar cargar L&F");
              }
              Fuente= new Font("Arial",style,tamaño);
              areaTexto= new JTextArea();
              areaTexto.setFont(Fuente);
              menus ();
              barra();
              scroll= new JScrollPane(areaTexto);
              getContentPane().add(scroll,BorderLayout.CENTER);
              JPanel panel= new JPanel();
              JPanel panel1= new JPanel();
              JPanel panel2= new JPanel();
              panel.setBackground(Color.lightGray);
              panel1.setBackground(Color.lightGray);
              panel2.setBackground(Color.lightGray);
              getContentPane().add(panel,BorderLayout.SOUTH);
              getContentPane().add(panel1,BorderLayout.WEST);
              getContentPane().add(panel2,BorderLayout.EAST);

              // handling Key event
              listener = new KeyHandler();
              addKeyListener(listener);
              setFocusable(true);

              setSize(WIDTH,HEIGHT);
              setVisible(true);
              show();
       }
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
	public void menus () {
		JMenuBar menus = new JMenuBar();
		JMenu archivo= new JMenu("Archivo");
		JMenuItem nue= new JMenuItem("Nuevo", new
ImageIcon("images/hoja.gif"));
		nue.addActionListener (
			new ActionListener () {
			public void actionPerformed (ActionEvent e) {
					nuevo();
				}
			}
		);
		JMenuItem sal= new JMenuItem("Salir",new
ImageIcon("images/salir.gif"));
		sal.addActionListener(
			new ActionListener () {
				public void actionPerformed (ActionEvent e) {
				try {
					String d=JOptionPane.showInputDialog("Desea
salir y guardar el archivo S/N");
					if (d.equals("s") || d.equals("S")) {
						Guardar();
						System.exit(0);
					}
					else if(d.equals("n") || d.equals("N")) {
						System.exit(0);
					}
					else {

	JOptionPane.showMessageDialog(null,"Caracter invalido");
						}
					}
				catch (Exception ex) {
				  System.out.println("Cancelo la opcion Salir");
					}
				}
			}
		);

		JMenuItem abr= new JMenuItem("Abrir",new
ImageIcon("images/libro.gif"));
		abr.addActionListener(
			new ActionListener () {
				public void actionPerformed (ActionEvent e) {
					abrir ();
				}
			}
		);
		JMenuItem guar= new JMenuItem("Guardar",new
ImageIcon("images/save.gif"));
		guar.addActionListener (
			new ActionListener () {
				public void actionPerformed (ActionEvent e) {
					Guardar ();
				}
			}
		);
		JMenu editar= new JMenu("Edición");
		JMenuItem cor= new JMenuItem("Cortar", new
ImageIcon("images/cut.gif"));
		cor.addActionListener (
			new ActionListener () {
				public void actionPerformed (ActionEvent e) {
					cortar();
				}
			}
		);
```

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
                JMenuItem          cop=          new          JMenuItem("Copiar",new
ImageIcon("images/copiar.gif"));
                cop.addActionListener (
                        new ActionListener () {
                                public void actionPerformed (ActionEvent e) {
                                        copiar ();
                                }
                        }
                );
                JMenuItem        peg=        new        JMenuItem        ("Pegar",new
ImageIcon("images/paste.gif"));
                peg.addActionListener (
                        new ActionListener () {
                                public void actionPerformed (ActionEvent e) {
                                        pegar ();
                                }
                        }
                );
            JMenuItem        col=        new        JMenuItem        ("Colorear",new
ImageIcon("images/paste.gif"));
                col.addActionListener (
                        new ActionListener () {
                                public void actionPerformed (ActionEvent e) {
                                        colorear ();
                                }
                        }
                );

                JMenu integra= new JMenu("Integración");

                JMenuItem NewtonCotes= new JMenuItem("Newton-Cotes");
                NewtonCotes.addActionListener(
                        new ActionListener () {
                                public void actionPerformed (ActionEvent e) {
                                        newton();
                                }
                        }
                );
                JMenuItem GaussLegendre= new JMenuItem("Gauss-Legendre");
                GaussLegendre.addActionListener(
                        new ActionListener () {
                                public void actionPerformed (ActionEvent e) {
                                        legendre();
                                }
                        }
                );
                JMenuItem GaussLaguerre= new JMenuItem("Gauss-Laguerre");
                GaussLaguerre.addActionListener(
                        new ActionListener () {
                                public void actionPerformed (ActionEvent e) {
                                        laguerre();
                                }
                        }
                );

                JMenu ode= new JMenu("EDO's");

                JMenuItem Basicas= new JMenuItem("Basicas");
                Basicas.addActionListener(
                        new ActionListener () {
                                public void actionPerformed (ActionEvent e) {
                                        ODEbasicas();
                                }
                        }
                );
                JMenuItem SistemaODE= new JMenuItem("Sistema EDO");
                SistemaODE.addActionListener(
                        new ActionListener () {
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
                        public void actionPerformed (ActionEvent e) {
                                ODEsistemas();
                                }
                        }
                );

                JMenu about= new JMenu("Ayuda");
                JMenuItem          ayu=          new          JMenuItem("Ayuda",new
ImageIcon("images/ayuda.gif"));
                ayu.addActionListener (
                        new ActionListener () {
                                public void actionPerformed (ActionEvent e) {
                                        ayuda ();
                                }
                        }
                );
                JMenuItem      acer=      new      JMenuItem("Acerca      de...",new
ImageIcon("images/acerca.gif"));
                acer.addActionListener(
                        new ActionListener () {
                                public void actionPerformed (ActionEvent e) {
                                        acerca();
                                }
                        }
                );
                archivo.add(nue);
                archivo.add(abr);
                archivo.add(guar);
                archivo.addSeparator();
                archivo.add(sal);
                editar.add(cor);
                editar.add(cop);
                editar.add(peg);
                editar.addSeparator();
             editar.add(col);
             integra.add(NewtonCotes);
             integra.add(GaussLegendre);
             integra.add(GaussLaguerre);
             ode.add(Basicas);
             ode.add(SistemaODE);
                about.add(ayu);
                about.add(acer);
                menus.add(archivo);
                menus.add(editar);
                menus.add(integra);
                menus.add(ode);
                menus.add(about);
                setJMenuBar(menus);
        }

        public void barra () {
                JToolBar barras= new JToolBar();

                nue= new JButton ();
                nue.setIcon(new ImageIcon("images/hoja.gif"));
                nue.setMargin(new Insets(3,0,0,0));
                nue.setToolTipText("Nuevo");
                nue.addActionListener(
                        new ActionListener () {
                                public void actionPerformed (ActionEvent e) {
                                        nuevo ();
                                }
                        }
                );
                barras.add(nue);

                ab= new JButton();
                ab.setIcon(new ImageIcon("images/libro.gif"));
```

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
ab.setMargin(new Insets(2,0,0,0));
ab.setToolTipText("Abrir");
ab.addActionListener(
       new ActionListener () {
              public void actionPerformed (ActionEvent e) {
                     abrir ();
              }
       }
);
barras.add(ab);

gua= new JButton();
gua.setIcon(new ImageIcon("images/save.gif"));
gua.setMargin(new Insets(2,0,0,0));
gua.setToolTipText("Guardar");
gua.addActionListener(
       new ActionListener () {
              public void actionPerformed (ActionEvent e) {
                     Guardar ();
              }
       }
);
barras.add(gua);

barras.addSeparator();

cor= new JButton();
cor.setIcon(new ImageIcon("images/cut.gif"));
cor.setMargin(new Insets(2,0,0,0));
cor.setToolTipText("Cortar");
cor.addActionListener (
       new ActionListener () {
              public void actionPerformed (ActionEvent e) {
                     cortar ();
              }
       }
);
barras.add(cor);

cop= new JButton();
cop.setIcon(new ImageIcon("images/copiar.gif"));
cop.setMargin(new Insets(-3,0,0,0));
cop.setToolTipText("Copiar");
cop.addActionListener (
       new ActionListener () {
              public void actionPerformed (ActionEvent e) {
                     copiar ();
              }
       }
);
barras.add(cop);

peg= new JButton();
peg.setIcon(new ImageIcon("images/paste.gif"));
peg.setMargin(new Insets(2,0,0,0));
peg.setToolTipText("Pegar");
peg.addActionListener (
       new ActionListener () {
              public void actionPerformed (ActionEvent e) {
                     pegar ();
              }
       }
);
barras.add(peg);

JButton del= new JButton();
del.setIcon(new ImageIcon("images/borrador.gif"));
del.setMargin(new Insets(2,0,0,0));
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
                del.setToolTipText("BORRAR todo el texto");
                del.addActionListener (
                        new ActionListener () {
                                public void actionPerformed (ActionEvent e) {
                                        areaTexto.setText("");
                                }
                        }
                );
                barras.add(del);
                barras.addSeparator();

                ayuda= new JButton();
                ayuda.setIcon(new ImageIcon("images/ayuda.gif"));
                ayuda.setMargin(new Insets(2,0,0,0));
                ayuda.setToolTipText("Ayuda");
                ayuda.addActionListener (
                        new ActionListener () {
                                public void actionPerformed (ActionEvent e) {
                                        ayuda ();
                                }
                        }
                );
                barras.add(ayuda);

                acerca= new JButton();
                acerca.setIcon(new ImageIcon("images/acerca.gif"));
                acerca.setMargin(new Insets(5,2,0,0));
                acerca.setToolTipText("Acerca de...");
                acerca.addActionListener (
                        new ActionListener () {
                                public void actionPerformed (ActionEvent e) {
                                        acerca ();
                                }
                        }
                );
                barras.add(acerca);
                barras.addSeparator();

                neg= new JButton();
                neg.setIcon(new ImageIcon("images/negrita.gif"));
                neg.setMargin(new Insets(4,2,0,0));
                neg.setToolTipText("Negrita");
                neg.addActionListener (
                        new ActionListener () {
                                public void actionPerformed (ActionEvent e) {
                                        if(negrita == 0) {
                                                style+=Font.BOLD;
                                                negrita = 1;
                                        }
                                        else {
                                                style-=Font.BOLD;
                                                negrita = 0;
                                        }

                                        areaTexto.setFont(new
Font(areaTexto.getFont().getName(),style,tamaño));
                                        System.out.println("Realize el cambio Negrita");
                                                repaint();

                                        }
                        }
                );
                barras.add(neg);
                cur= new JButton();
                cur.setIcon(new ImageIcon("images/cursiva.gif"));
                cur.setMargin(new Insets(4,2,0,0));
                cur.setToolTipText("Cursiva");
                cur.addActionListener (
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
                        new ActionListener () {
                              public void actionPerformed (ActionEvent e) {

                                    if(cursiva == 0) {
                                          style+=Font.ITALIC;
                                          cursiva = 1;
                                    }
                                    else {
                                          style-=Font.ITALIC;
                                          cursiva = 0;
                                    }

                        areaTexto.setFont(new
Font(areaTexto.getFont().getName(),style,tamaño));
                              System.out.println("Realize el cambio CURSIVA");
                                    repaint();

                              }
                        }
                  );
                  barras.add(cur);
                  barras.addSeparator();

                  LFuente = new JLabel();
                  LFuente.setText(" Fuente:  ");
                  barras.add(LFuente);

                  tFuente = new JComboBox();
                  tFuente.addItem("12");
                  tFuente.addItem("14");
                  tFuente.addItem("16");
                  tFuente.addItem("18");
                  tFuente.addItem("20");
                  tFuente.addItem("22");
                  tFuente.addItem("24");
                  tFuente.addItem("26");
                  tFuente.addItem("28");
                  tFuente.addItem("30");
                  tFuente.setToolTipText("Tamaño de fuente");
                  tFuente.addItemListener(
                        new ItemListener () {
                              public void itemStateChanged(ItemEvent e) {
                              int elegido = tFuente.getSelectedIndex();

                              tamaño =10 + (elegido+1)*2;
                              Fuente=    new    Font(areaTexto.getFont().getName(),
style,tamaño);
                                    areaTexto.setFont(Fuente);

                              }
                        }
                  );
                  barras.add(tFuente);
                  barras.addSeparator();
                  cFuente = new JComboBox();
                  cFuente.addItem("Negro");
                  cFuente.addItem("Azul");
                  cFuente.addItem("Rojo");
                  cFuente.addItem("Amarillo");
                  cFuente.addItem("Verde");
                  cFuente.addItem("Cyan");
                  cFuente.setToolTipText("Color de fuente");
                  cFuente.addItemListener(
                        new ItemListener () {
                              public void itemStateChanged(ItemEvent e) {

      areaTexto.setForeground(colorValue[cFuente.getSelectedIndex()]);
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
                        }
                    }
                );
                barras.add(cFuente);
                barras.addSeparator();
                tipoFuente = new JComboBox();
                tipoFuente.addItem("Arial");
                tipoFuente.addItem("Times Roman");
                tipoFuente.addItem("Courier");
                tipoFuente.addItem("Helvetica");
                tipoFuente.setToolTipText("Tipo de fuente");
                tipoFuente.setSelectedIndex(0);
                tipoFuente.addItemListener(
                        new ItemListener () {
                                public void itemStateChanged(ItemEvent e) {

                                        areaTexto.setFont(new
Font(FontNames[tipoFuente.getSelectedIndex()],style,tamaño));
                                }
                        }
                );
                barras.add(tipoFuente);
                barras.addSeparator();
                getContentPane().add(barras,BorderLayout.NORTH);
        }

        public void nuevo () {
                try {
                        String d=JOptionPane.showInputDialog("Desea salir y guardar
el archivo S/N");
                        if (d.equals("s") || d.equals("S")) {
                                Guardar();
                                areaTexto.setText("");
                        }
                        else
                                if(d.equals("n") || d.equals("N"))
                                        areaTexto.setText("");
                                else
                                        JOptionPane.showMessageDialog(null,"Caracter
invalido");
                }
                catch (Exception e) {
                        System.out.println("Cancelo la opcion Nuevo");
                }
        }

        public void abrir () {
                JFileChooser fileChooser = new JFileChooser();
                fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
                int result= fileChooser.showOpenDialog(this);
                if (result== JFileChooser.CANCEL_OPTION) return;
                File name= fileChooser.getSelectedFile();
                if(name.exists()) {
                        if (name.isFile()) {
                                try {
                                        BufferedReader input= new BufferedReader(new
FileReader (name));
                                        StringBuffer buffer= new StringBuffer();
                                        String text;
                                        areaTexto.setText("");
                                        while ((text=input.readLine()) !=null)
                                                buffer.append(text+ "\n");
                                        areaTexto.append(buffer.toString());
                                }
                                catch (IOException ioException) {
                                        JOptionPane.showMessageDialog(null,"Error   en
el archivo", "Error",JOptionPane.ERROR_MESSAGE);
                                }
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
                        }
                        else if (name.isDirectory ()) {
                                String directory[] = name.list();
                                areaTexto.append("\n\nContenido del directorio:\n");
                                for (int i=0;i<directory.length; i++)
                                        areaTexto.append(directory [i]+"\n");
                        }
                        else {
                                JOptionPane.showMessageDialog(null," No existe ","
Error ",JOptionPane.ERROR_MESSAGE);
                        }
                }
        }

        public void Guardar () {
                JFileChooser fileChooser = new JFileChooser();
                fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
                int result= fileChooser.showSaveDialog(this);
                if (result== JFileChooser.CANCEL_OPTION) return;
                        File name= fileChooser.getSelectedFile();
                        try {
                                PrintWriter output= new PrintWriter(new FileWriter(
name));
                                output.write(areaTexto.getText());
                                output.close();
                        }
                        catch (IOException ioException) {
                                JOptionPane.showMessageDialog(null,"Error en el
archivo","Error",JOptionPane.ERROR_MESSAGE);
                        }
        }

        public void cortar () {
                areaTexto.cut();
        }

        public void copiar () {
                areaTexto.copy();
        }

        public void pegar () {
                areaTexto.paste();
        }

        public void colorear () {
                Color color=JColorChooser.showDialog(
                                integraLAB.this,
                              "Colorear",Color.red);
                if(color!=null)
                        areaTexto.setForeground(color);
        }

        public void newton() {
                newtonDialog dlg= new newtonDialog(integraLAB.this);
                dlg.setSize(500,300);
                dlg.show();
        }
        public void legendre() {
                legendreDialog dlg= new legendreDialog(integraLAB.this);
                dlg.setSize(500,300);
                dlg.show();
        }
        public void laguerre() {
                laguerreDialog dlg= new laguerreDialog(integraLAB.this);
                dlg.setSize(500,300);
                dlg.show();
        }
        public void ODEbasicas() {
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
                basicasDialog dlg= new basicasDialog(integraLAB.this);
                dlg.setSize(500,300);
                dlg.show();
        }
        public void ODEsistemas() {
                sistemasDialog dlg= new sistemasDialog(integraLAB.this);
                dlg.setSize(500,300);
                dlg.show();
        }
        public void ayuda () {
                JOptionPane.showMessageDialog(null,"\n  Nuevo:  Abre  una  nueva
ventana\n Abrir: Abre un documento existente\n Guardar: Guarda el documento\n
Salir: Salir del programa\n Cortar: ctrl+x\n Copiar: ctrl+c\n Pegar: ctrl+v\n
Salir sin Guardar: alt+F4 \n");
        }
        public void acerca () {
                acercaDialog dlg= new acercaDialog(integraLAB.this);
                dlg.show();
        }
    class newtonDialog extends JDialog
    {
        String lexema="sqrt(x)";
        int metodo=1;
        JRadioButton r1;
        JRadioButton r2;
        JRadioButton r3;
        JRadioButton r4;
        JTextField salida;
        double limiteA=0.0,limiteB=4.0;
        int n=10;

         public newtonDialog(Frame owner)
        {

                super(owner, "Newton - Cotes", true);

                    JTextField funcionText;
                    final JButton btOK;

                JLabel lbl = new JLabel(new ImageIcon(
                                        "images/imagInteg.jpg"));
                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 p1=new JPanel();
                   p1.add(new JLabel("funcion"));
                   funcionText=new JTextField(lexema,30);
                   funcionText.addActionListener(
                          new ActionListener() {
                                 public void actionPerformed(ActionEvent e)
                                 {
                                        lexema=e.getActionCommand();
                                 }
                          }
                   );
                   p1.add(funcionText);
                   p.add(p1);
                getContentPane().add(p, BorderLayout.NORTH);

                JPanel p2=new JPanel();
                p2.setLayout(new BoxLayout(p2,BoxLayout.Y_AXIS));
                p2.setBorder(new TitledBorder(new EtchedBorder(),
                             "Opciones"));

                ButtonGroup group=new ButtonGroup();
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
r1=new JRadioButton("Trapecio",true);
group.add(r1);
p2.add(r1);

r2=new JRadioButton("Simpson 1/3",false);
group.add(r2);
p2.add(r2);

r3=new JRadioButton("Simpson 3/8",false);
group.add(r3);
p2.add(r3);

r4=new JRadioButton("Bool",false);
group.add(r4);
p2.add(r4);

getContentPane().add(p2, BorderLayout.WEST);

RadioButtonHandler handler=new RadioButtonHandler();
r1.addItemListener(handler);
r2.addItemListener(handler);
r3.addItemListener(handler);
r4.addItemListener(handler);

JPanel p3=new JPanel();
p3.setBorder(new TitledBorder(new EtchedBorder(),
          "Datos"));
JPanel p31=new JPanel();
  p31.add(new JLabel("a"));
  JTextField aText=new JTextField("0.0",8);
  aText.addActionListener(
        new ActionListener() {
              public void actionPerformed(ActionEvent e)
              {
                    limiteA=Double.valueOf(
                          e.getActionCommand()).
                          doubleValue();

              }
        }
  );
  p31.add(aText);
  p3.add(p31,BorderLayout.NORTH);
  JPanel p32=new JPanel();
  p32.add(new JLabel("b"));
  JTextField bText=new JTextField("4.0",8);
  bText.addActionListener(
        new ActionListener() {
              public void actionPerformed(ActionEvent e)
              {
                    limiteB=Double.valueOf(
                          e.getActionCommand()).
                          doubleValue();

              }
        }
  );
  p32.add(bText);
  p3.add(p32,BorderLayout.CENTER);
  JPanel p33=new JPanel();
  p33.add(new JLabel("n"));
  JTextField nText=new JTextField("10",8);
  nText.addActionListener(
        new ActionListener() {
              public void actionPerformed(ActionEvent e)
              {
                    n=Integer.parseInt(
                          e.getActionCommand());
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
                        }
                    }
                );
                p33.add(nText);
                p3.add(p33,BorderLayout.SOUTH);

            getContentPane().add(p3, BorderLayout.CENTER);

                JPanel p4=new JPanel();
            p4.setBorder(new TitledBorder(new EtchedBorder(),
                        "Solucion"));

                JButton goOK = new JButton("GO");
                ActionListener rst = new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                  salida.setText((Double.toString(
                              algor(metodo,limiteA,limiteB,n)
                                  )));
                  }
            };
            goOK.addActionListener(rst);
            salida=new JTextField("Solucion",15);
            salida.setEditable(false);
            p4.add(salida,BorderLayout.CENTER);
            getContentPane().add(p4, BorderLayout.EAST);

                btOK = new JButton("OK");
            ActionListener lst = new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                dispose();
              }
            };
            btOK.addActionListener(lst);
            p = new JPanel();
            p.add(goOK);
            p.add(btOK);
            getRootPane().setDefaultButton(btOK);
                getRootPane().registerKeyboardAction(lst,
                        KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0),
                      JComponent.WHEN_IN_FOCUSED_WINDOW);
            getContentPane().add(p, BorderLayout.SOUTH);
                WindowListener wl = new WindowAdapter() {
                        public void windowOpened(WindowEvent e) {
                            btOK.requestFocus();
                        }
                };
                addWindowListener(wl);
                pack();
            setResizable(false);
            setLocationRelativeTo(owner);
        }
        private class RadioButtonHandler implements ItemListener
        {
                public void itemStateChanged(ItemEvent e)
                {
                        if(e.getSource()==r1)
                                metodo=1;
                        else if(e.getSource()==r2)
                                metodo=2;
                        else if(e.getSource()==r3)
                                metodo=3;
                        else if(e.getSource()==r4)
                                metodo=4;
                 }
        }
        // Algoritmos numericos
        double algor(int metodo,double a,double b,int n)
        {
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
        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=bool(a,b,n); break;
        }
        return y;
}
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));
}
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;
}
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;
}
private double bool(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++)
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
                {
                        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;
        }
}

class legendreDialog extends JDialog
{
   String lexema="sqrt(x)";

   JRadioButton r1;
   JRadioButton r2;
   JRadioButton r3;
   JRadioButton r4;
   JRadioButton r5;
   JTextField salida;
   double limiteA=0.0,limiteB=4.0;
   int n=2;

    public legendreDialog(Frame owner)
   {
          super(owner, "Gauss - Legendre", true);

                JTextField funcionText;
                final JButton btOK;

               JLabel lbl = new JLabel(new ImageIcon(
                                    "images/imagInteg.jpg"));
               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 p1=new JPanel();
                  p1.add(new JLabel("funcion"));
                  funcionText=new JTextField(lexema,30);
                  funcionText.addActionListener(
                         new ActionListener() {
                                public void actionPerformed(ActionEvent e)
                                {
                                       lexema=e.getActionCommand();

                                }
                         }
                  );
                  p1.add(funcionText);
                  p.add(p1);
               getContentPane().add(p, BorderLayout.NORTH);

               JPanel p2=new JPanel();
               p2.setLayout(new BoxLayout(p2,BoxLayout.Y_AXIS));
               p2.setBorder(new TitledBorder(new EtchedBorder(),
                            "Numero de puntos"));

               ButtonGroup group=new ButtonGroup();
               r1=new JRadioButton("   Dos    (2)  ",true);
               group.add(r1);
               p2.add(r1);
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
r2=new JRadioButton("   Tres   (3)  ",false);
group.add(r2);
p2.add(r2);

r3=new JRadioButton("   Cuatro (4)  ",false);
group.add(r3);
p2.add(r3);

r4=new JRadioButton("   Cinco  (5)  ",false);
group.add(r4);
p2.add(r4);
r5=new JRadioButton("   Seis   (6)  ",false);
group.add(r5);
p2.add(r5);

getContentPane().add(p2, BorderLayout.WEST);

RadioButtonHandler handler=new RadioButtonHandler();
r1.addItemListener(handler);
r2.addItemListener(handler);
r3.addItemListener(handler);
r4.addItemListener(handler);
r5.addItemListener(handler);

JPanel p3=new JPanel();
p3.setBorder(new TitledBorder(new EtchedBorder(),
          "Datos"));
JPanel p31=new JPanel();
  p31.add(new JLabel("a"));
  JTextField aText=new JTextField("0.0",8);
  aText.addActionListener(
        new ActionListener() {
              public void actionPerformed(ActionEvent e)
              {
                    limiteA=Double.valueOf(
                          e.getActionCommand()).
                          doubleValue();

              }
        }
  );
  p31.add(aText);
  p3.add(p31,BorderLayout.NORTH);
  JPanel p32=new JPanel();
  p32.add(new JLabel("b"));
  JTextField bText=new JTextField("4.0",8);
  bText.addActionListener(
        new ActionListener() {
              public void actionPerformed(ActionEvent e)
              {
                    limiteB=Double.valueOf(
                          e.getActionCommand()).
                          doubleValue();
              }
        }
  );
  p32.add(bText);
  p3.add(p32,BorderLayout.CENTER);

getContentPane().add(p3, BorderLayout.CENTER);

  JPanel p4=new JPanel();
p4.setBorder(new TitledBorder(new EtchedBorder(),
          "Solucion"));

  JButton goOK = new JButton("GO");
  ActionListener rst = new ActionListener() {
 public void actionPerformed(ActionEvent e) {
```

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
            salida.setText((Double.toString(
                        algor(limiteA,limiteB,n)
                            )));
            }
        };
        goOK.addActionListener(rst);
        salida=new JTextField("Solucion",15);
        salida.setEditable(false);
        p4.add(salida,BorderLayout.CENTER);
        getContentPane().add(p4, BorderLayout.EAST);

           btOK = new JButton("OK");
        ActionListener lst = new ActionListener() {
          public void actionPerformed(ActionEvent e) {
            dispose();
          }
        };
        btOK.addActionListener(lst);
        p = new JPanel();
        p.add(goOK);
        p.add(btOK);
        getRootPane().setDefaultButton(btOK);
           getRootPane().registerKeyboardAction(lst,
                   KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0),
                JComponent.WHEN_IN_FOCUSED_WINDOW);
        getContentPane().add(p, BorderLayout.SOUTH);

           WindowListener wl = new WindowAdapter() {
                   public void windowOpened(WindowEvent e) {
                       btOK.requestFocus();
                   }
            };
            addWindowListener(wl);

            pack();
        setResizable(false);
        setLocationRelativeTo(owner);
    }
    private class RadioButtonHandler implements ItemListener
    {
           public void itemStateChanged(ItemEvent e)
           {
                  if(e.getSource()==r1)
                         n=2;
                  else if(e.getSource()==r2)
                         n=3;
                  else if(e.getSource()==r3)
                         n=4;
                  else if(e.getSource()==r4)
                         n=5;
                  else if(e.getSource()==r5)
                         n=6;
            }
    }
    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;
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
                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;
             }

}
class laguerreDialog extends JDialog
{
   String lexema="x^2";

   JRadioButton r1;
   JRadioButton r2;
   JRadioButton r3;
   JRadioButton r4;
   JRadioButton r5;
   JTextField salida;
   double limiteA=1;
   int n=5;

    public laguerreDialog(Frame owner)
   {
         super(owner, "Gauss - Laguerre", true);

               JTextField funcionText;
               final JButton btOK;

            JLabel lbl = new JLabel(new ImageIcon(
                                    "images/imagInteg.jpg"));
            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 p1=new JPanel();
               p1.add(new JLabel("funcion"));
               funcionText=new JTextField(lexema,30);
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
   funcionText.addActionListener(
          new ActionListener() {
                public void actionPerformed(ActionEvent e)
                {
                      lexema=e.getActionCommand();
                }
          }
   );
   p1.add(funcionText);
   p.add(p1);
getContentPane().add(p, BorderLayout.NORTH);

JPanel p2=new JPanel();
p2.setLayout(new BoxLayout(p2,BoxLayout.Y_AXIS));
p2.setBorder(new TitledBorder(new EtchedBorder(),
            "Numero de puntos"));
ButtonGroup group=new ButtonGroup();
r1=new JRadioButton("   Dos    (2)  ",false);
group.add(r1);
p2.add(r1);

r2=new JRadioButton("   Tres   (3)  ",false);
group.add(r2);
p2.add(r2);

r3=new JRadioButton("   Cuatro (4)  ",false);
group.add(r3);
p2.add(r3);

r4=new JRadioButton("   Cinco  (5)  ",true);
group.add(r4);
p2.add(r4);
r5=new JRadioButton("   Seis   (6)  ",false);
group.add(r5);
p2.add(r5);

getContentPane().add(p2, BorderLayout.WEST);

RadioButtonHandler handler=new RadioButtonHandler();
r1.addItemListener(handler);
r2.addItemListener(handler);
r3.addItemListener(handler);
r4.addItemListener(handler);
r5.addItemListener(handler);

JPanel p3=new JPanel();
p3.setBorder(new TitledBorder(new EtchedBorder(),
            "Datos"));
JPanel p31=new JPanel();
   p31.add(new JLabel("a"));
   JTextField aText=new JTextField("1.0",8);
   aText.addActionListener(
          new ActionListener() {
                public void actionPerformed(ActionEvent e)
                {
                      limiteA=Double.valueOf(
                            e.getActionCommand()).
                            doubleValue();
                }
          }
   );
   p31.add(aText);
   p3.add(p31,BorderLayout.NORTH);
   JPanel p32=new JPanel();
   p32.add(new JLabel("b"));
   JTextField bText=new JTextField("Infinito",8);
   bText.setEditable(false);
   p32.add(bText);
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
        p3.add(p32,BorderLayout.CENTER);

    getContentPane().add(p3, BorderLayout.CENTER);

        JPanel p4=new JPanel();
    p4.setBorder(new TitledBorder(new EtchedBorder(),
                "Solucion"));
        JButton goOK = new JButton("GO");
        ActionListener rst = new ActionListener() {
       public void actionPerformed(ActionEvent e) {
         salida.setText((Double.toString(
                    algor(limiteA,n)
                        )));
        }
    };
    goOK.addActionListener(rst);
    salida=new JTextField("Solucion",15);
    salida.setEditable(false);
    p4.add(salida,BorderLayout.CENTER);
    getContentPane().add(p4, BorderLayout.EAST);

        btOK = new JButton("OK");
    ActionListener lst = new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        dispose();
      }
    };
    btOK.addActionListener(lst);
    p = new JPanel();
    p.add(goOK);
    p.add(btOK);
    getRootPane().setDefaultButton(btOK);
        getRootPane().registerKeyboardAction(lst,
                KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0),
               JComponent.WHEN_IN_FOCUSED_WINDOW);
    getContentPane().add(p, BorderLayout.SOUTH);

        WindowListener wl = new WindowAdapter() {
                public void windowOpened(WindowEvent e) {
                        btOK.requestFocus();
                }
        };
        addWindowListener(wl);

        pack();
    setResizable(false);
    setLocationRelativeTo(owner);
}
private class RadioButtonHandler implements ItemListener
{
        public void itemStateChanged(ItemEvent e)
        {
                if(e.getSource()==r1)
                        n=2;
                else if(e.getSource()==r2)
                        n=3;
                else if(e.getSource()==r3)
                        n=4;
                else if(e.getSource()==r4)
                        n=5;
                else if(e.getSource()==r5)
                        n=6;
         }
}
double algor(double a,int n)
{
        double[][] x=new double[7][7];
        double[][] w=new double[7][7];
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
                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]=9.395070922301;
            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);
         }

}

class basicasDialog extends JDialog
{
   String lexema="y*(x*x-1)";
   int metodo=1;
   JRadioButton r1;
   JRadioButton r2;
   JRadioButton r3;
   JRadioButton r4;
   JTextField salida;
   DecimalFormat decimales2;
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
DecimalFormat decimales7;
double limiteA=0.0,limiteB=2.0,yCero=1,h;
int n=2;

String[] arreglo;
 public basicasDialog(Frame owner)
{

      super(owner, "EDO Basicas", true);
            decimales2=new DecimalFormat("0.00");
            decimales7=new DecimalFormat("0.0000000");

            JTextField funcionText;
            final JButton btOK;

          JLabel lbl = new JLabel(new ImageIcon(
                                "images/imagODE.jpg"));
          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 p1=new JPanel();
             p1.add(new JLabel(" f(x,y) "));
             funcionText=new JTextField(lexema,30);
             funcionText.addActionListener(
                    new ActionListener() {
                          public void actionPerformed(ActionEvent e)
                          {
                                lexema=e.getActionCommand();
                          }
                    }
             );
             p1.add(funcionText);
             p.add(p1);
          getContentPane().add(p, BorderLayout.NORTH);

          JPanel p2=new JPanel();
          p2.setLayout(new BoxLayout(p2,BoxLayout.Y_AXIS));
          p2.setBorder(new TitledBorder(new EtchedBorder(),
                     "Opciones"));

          ButtonGroup group=new ButtonGroup();
          r1=new JRadioButton(" Euler ",true);
          group.add(r1);
          p2.add(r1);

          r2=new JRadioButton(" Heun  ",false);
          group.add(r2);
          p2.add(r2);

          r3=new JRadioButton(" Rk 2  ",false);
          group.add(r3);
          p2.add(r3);

          r4=new JRadioButton(" RK 4  ",false);
          group.add(r4);
          p2.add(r4);

          getContentPane().add(p2, BorderLayout.WEST);

          RadioButtonHandler handler=new RadioButtonHandler();
          r1.addItemListener(handler);
          r2.addItemListener(handler);
          r3.addItemListener(handler);
          r4.addItemListener(handler);
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
JPanel p3=new JPanel();
p3.setBorder(new TitledBorder(new EtchedBorder(),
             "Datos"));
JPanel p31=new JPanel();
   p31.add(new JLabel(" a"));
   JTextField aText=new JTextField("0.0",8);
   aText.addActionListener(
          new ActionListener() {
                public void actionPerformed(ActionEvent e)
                {
                      limiteA=Double.valueOf(
                             e.getActionCommand()).
                             doubleValue();
                }
          }
   );
   p31.add(aText);
   p3.add(p31,BorderLayout.NORTH);
   JPanel p32=new JPanel();
   p32.add(new JLabel(" b"));
   JTextField bText=new JTextField("2.0",8);
   bText.addActionListener(
          new ActionListener() {
                public void actionPerformed(ActionEvent e)
                {
                      limiteB=Double.valueOf(
                            e.getActionCommand()).
                            doubleValue();
                }
          }
   );
   p32.add(bText);
   p3.add(p32,BorderLayout.CENTER);
   JPanel p33=new JPanel();
   p33.add(new JLabel(" n"));
   JTextField nText=new JTextField("2",8);
   nText.addActionListener(
          new ActionListener() {
                public void actionPerformed(ActionEvent e)
                {
                      n=Integer.parseInt(
                            e.getActionCommand());
                }
          }
   );
   p33.add(nText);
   p3.add(p33,BorderLayout.SOUTH);
   JPanel p34=new JPanel();
   p34.add(new JLabel("y0"));
   JTextField y0Text=new JTextField("1.0",8);
   y0Text.addActionListener(
          new ActionListener() {
                public void actionPerformed(ActionEvent e)
                {
                      yCero=Double.valueOf(
                             e.getActionCommand()).
                             doubleValue();
                }
          }
   );
   p34.add(y0Text);
   p3.add(p34,BorderLayout.SOUTH);
getContentPane().add(p3, BorderLayout.CENTER);

   JPanel p4=new JPanel();
p4.setBorder(new TitledBorder(new EtchedBorder(),
             "Solucion"));
```

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
        JButton goOK = new JButton("GO");
        ActionListener rst = new ActionListener() {
       public void actionPerformed(ActionEvent e) {
         salida.setText((Double.toString(
                    algor(metodo,limiteA,limiteB,yCero,n)
                        )));
         JList list=new JList(arreglo);
         JScrollPane scrollPane=new JScrollPane(list);
         JOptionPane.showMessageDialog(null,scrollPane,
                                    "EDO",
                JOptionPane.INFORMATION_MESSAGE);
       }
     };
     goOK.addActionListener(rst);
     salida=new JTextField("Solucion",15);
     salida.setEditable(false);
     p4.add(salida,BorderLayout.CENTER);
     getContentPane().add(p4, BorderLayout.EAST);

        btOK = new JButton("OK");
     ActionListener lst = new ActionListener() {
       public void actionPerformed(ActionEvent e) {
         dispose();
       }
     };
     btOK.addActionListener(lst);
     p = new JPanel();
     p.add(goOK);
     p.add(btOK);
     getRootPane().setDefaultButton(btOK);
        getRootPane().registerKeyboardAction(lst,
                 KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0),
               JComponent.WHEN_IN_FOCUSED_WINDOW);
     getContentPane().add(p, BorderLayout.SOUTH);

        WindowListener wl = new WindowAdapter() {
                 public void windowOpened(WindowEvent e) {
                        btOK.requestFocus();
                 }
        };
        addWindowListener(wl);

        pack();
     setResizable(false);
     setLocationRelativeTo(owner);
}
private class RadioButtonHandler implements ItemListener
{
        public void itemStateChanged(ItemEvent e)
        {
                if(e.getSource()==r1)
                        metodo=1;
                else if(e.getSource()==r2)
                        metodo=2;
                else if(e.getSource()==r3)
                        metodo=3;
                else if(e.getSource()==r4)
                        metodo=4;
         }
}
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;
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
        }
        return y;
}
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)+"    "+
                  decimales7.format(y);
        for(int i=1;i<=n;i++)
        {
                y+=h*p.getValue(x,y);
                x+=h;
                arreglo[i]=decimales2.format(x)+"    "+
                    decimales7.format(y);
        }
        return y;
}
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)+"    "+
                  decimales7.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)+"    "+
                    decimales7.format(y);
        }
        return y;
}
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)+"    "+
                  decimales7.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)+"    "+
                    decimales7.format(y);
        }
        return y;
}

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"+
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
                        decimales7.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)+"    "+
                            decimales7.format(y);
                }
                return y;
        }
}

class sistemasDialog extends JDialog
{
   String lexema1="y+2.0*z";
   String lexema2="3.0*y+2.0*z";

   JTextField salida;
   DecimalFormat decimales2;
   DecimalFormat decimales7;
   double limiteA=0.0,limiteB=0.2,h;
   double yCero=6,zCero=4;
   int n=10;

   String[] arreglo;
     public sistemasDialog(Frame owner)
   {

         super(owner, "Sistemas EDO", true);
                decimales2=new DecimalFormat("0.000");
                decimales7=new DecimalFormat("0.0000000");

                JTextField funcionText1;
                JTextField funcionText2;
                final JButton btOK;

            JPanel p = new JPanel();
            p.setLayout(new BorderLayout());

            JPanel p1=new JPanel();
               p1.add(new JLabel(" f(x,y,z) "));
               funcionText1=new JTextField(lexema1,30);
               funcionText1.addActionListener(
                      new ActionListener() {
                             public void actionPerformed(ActionEvent e)
                             {
                                    lexema1=e.getActionCommand();
                             }
                      }
               );
               p1.add(funcionText1);

               JPanel p11=new JPanel();
               p11.add(new JLabel(" g(x,y,z) "));
               funcionText2=new JTextField(lexema2,30);
               funcionText2.addActionListener(
                      new ActionListener() {
                             public void actionPerformed(ActionEvent e)
                             {
                                    lexema2=e.getActionCommand();
                             }
                      }
               );
               p11.add(funcionText2);
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
   p.add(p1,BorderLayout.NORTH);
   p.add(p11,BorderLayout.SOUTH);
getContentPane().add(p, BorderLayout.NORTH);

JPanel p2=new JPanel();
p2.setLayout(new BoxLayout(p2,BoxLayout.Y_AXIS));
p2.setBorder(new TitledBorder(new EtchedBorder(),
             "Datos X"));

JPanel p21=new JPanel();
   p21.add(new JLabel(" a"));
   JTextField aText=new JTextField("0.0",8);
   aText.addActionListener(
          new ActionListener() {
                 public void actionPerformed(ActionEvent e)
                 {
                        limiteA=Double.valueOf(
                               e.getActionCommand()).
                               doubleValue();
                 }
          }
   );
   p21.add(aText);
   p2.add(p21,BorderLayout.NORTH);

   JPanel p22=new JPanel();
   p22.add(new JLabel(" b"));
   JTextField bText=new JTextField("0.2",8);
   bText.addActionListener(
          new ActionListener() {
                 public void actionPerformed(ActionEvent e)
                 {
                        limiteB=Double.valueOf(
                               e.getActionCommand()).
                               doubleValue();
                 }
          }
   );
   p22.add(bText);
   p2.add(p22,BorderLayout.CENTER);

   JPanel p23=new JPanel();
   p23.add(new JLabel(" n"));
   JTextField nText=new JTextField("10",8);
   nText.addActionListener(
          new ActionListener() {
                 public void actionPerformed(ActionEvent e)
                 {
                        n=Integer.parseInt(
                               e.getActionCommand());
                 }
          }
   );
   p23.add(nText);
   p2.add(p23,BorderLayout.SOUTH);
getContentPane().add(p2, BorderLayout.WEST);

JPanel p3=new JPanel();
p3.setBorder(new TitledBorder(new EtchedBorder(),
             " PVI"));

   JPanel p31=new JPanel();
   p31.add(new JLabel("y0"));
   JTextField y0Text=new JTextField("6.0",8);
   y0Text.addActionListener(
          new ActionListener() {
                 public void actionPerformed(ActionEvent e)
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
                {
                        yCero=Double.valueOf(
                                e.getActionCommand()).
                                doubleValue();
                }
        }
    );
    p31.add(y0Text);
    p3.add(p31,BorderLayout.NORTH);

    JPanel p32=new JPanel();
    p32.add(new JLabel("z0"));
    JTextField z0Text=new JTextField("4.0",8);
    z0Text.addActionListener(
          new ActionListener() {
                public void actionPerformed(ActionEvent e)
                {
                        zCero=Double.valueOf(
                                e.getActionCommand()).
                                doubleValue();
                }
          }
    );
    p32.add(z0Text);
    p3.add(p32,BorderLayout.SOUTH);
getContentPane().add(p3, BorderLayout.CENTER);

    JPanel p4=new JPanel();
p4.setBorder(new TitledBorder(new EtchedBorder(),
              "Solucion"));

    JButton goOK = new JButton("GO");
    ActionListener rst = new ActionListener() {
   public void actionPerformed(ActionEvent e) {
     salida.setText((Double.toString(
                      algor(limiteA,limiteB,n
                                ,yCero,zCero))
                    ));
     JList list=new JList(arreglo);
     JScrollPane scrollPane=new JScrollPane(list);
     JOptionPane.showMessageDialog(null,scrollPane,
                                "EDO",
           JOptionPane.INFORMATION_MESSAGE);
   }
};
goOK.addActionListener(rst);
salida=new JTextField("Solucion",15);
salida.setEditable(false);
p4.add(salida,BorderLayout.CENTER);
getContentPane().add(p4, BorderLayout.EAST);

    btOK = new JButton("OK");
ActionListener lst = new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    dispose();
  }
};
btOK.addActionListener(lst);
p = new JPanel();
p.add(goOK);
p.add(btOK);
getRootPane().setDefaultButton(btOK);
    getRootPane().registerKeyboardAction(lst,
             KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0),
          JComponent.WHEN_IN_FOCUSED_WINDOW);
getContentPane().add(p, BorderLayout.SOUTH);

    WindowListener wl = new WindowAdapter() {
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
                        public void windowOpened(WindowEvent e) {
                                btOK.requestFocus();
                        }
                    };
                    addWindowListener(wl);

                    pack();
                setResizable(false);
                setLocationRelativeTo(owner);
            }

            double algor(double a,double b,int n,double y0,
                         double z0)
            {
                    // RK4
                    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)+"   "+
                               decimales7.format(y)+"   "+
                               decimales7.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)+"   "+
                                    decimales7.format(y)+"   "+
                                    decimales7.format(z);
                    }
                    return y;
            }
    }

    class acercaDialog extends JDialog
    {
       public acercaDialog(Frame owner)
       {

            super(owner, "Acerca de", true);

                JLabel lbl = new JLabel(new ImageIcon(
                                        "images/icon.gif"));
                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);
                getContentPane().add(p, BorderLayout.WEST);

                String message = "IntegraLAB and Basic Text Editor
application\n"+
                  "(c) E.Ruiz Lizama 2006";
                JTextArea txt = new JTextArea(message);
                txt.setBorder(new EmptyBorder(5, 10, 5, 10));
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
                txt.setFont(new Font("Helvetica", Font.BOLD, 12));
                txt.setEditable(false);
                txt.setBackground(getBackground());
                   p = new JPanel();
                p.setLayout(new BoxLayout(p, BoxLayout.Y_AXIS));
                p.add(txt);

                message = "JVM version " +
                System.getProperty("java.version") + "\n"+
                            " by " + System.getProperty("java.vendor");
                txt = new JTextArea(message);
                txt.setBorder(new EmptyBorder(5, 10, 5, 10));
                txt.setFont(new Font("Arial", Font.PLAIN, 12));
                txt.setEditable(false);
                txt.setLineWrap(true);
                txt.setWrapStyleWord(true);
                txt.setBackground(getBackground());
                p.add(txt);

                getContentPane().add(p, BorderLayout.CENTER);
                   final JButton btOK = new JButton("OK");
                ActionListener lst = new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                    dispose();
                  }
                };
                btOK.addActionListener(lst);
                p = new JPanel();
                p.add(btOK);
                getRootPane().setDefaultButton(btOK);
                   getRootPane().registerKeyboardAction(lst,
                            KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0),
                          JComponent.WHEN_IN_FOCUSED_WINDOW);
                getContentPane().add(p, BorderLayout.SOUTH);

                  WindowListener wl = new WindowAdapter() {
                          public void windowOpened(WindowEvent e) {
                                 btOK.requestFocus();
                          }
                  };
                  addWindowListener(wl);

                  pack();
                setResizable(false);
                setLocationRelativeTo(owner);
         }
}

private class KeyHandler implements KeyListener
{
        public void keyPressed(KeyEvent event)
        {
           int keyCode = event.getKeyCode();

          // add line segment
           if (keyCode == KeyEvent.VK_G &&
               event.isControlDown())
           {
               GraphDialog dlg=new GraphDialog(integraLAB.this);
               dlg.setSize(500,500);
               dlg.show();
           }
        }

        public void keyReleased(KeyEvent event) {}

        public void keyTyped(KeyEvent event) {}
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
    }


        public static void main (String []args) {
               integraLAB app=new integraLAB();

               app.addWindowListener(
                            new WindowAdapter(){
                                   public void windowClosing(WindowEvent e)
                                   {
                                          System.exit(0);
                                   }
                            }
                     );
        } //fin de main

} //fin de integralab
```

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

APÉNDICE III

CÓDIGO JAVA PARA LA CLASE GRAPHDIALOG

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
/**
  *Autor : E. Ruiz Lizama
  *Fecha : 19/7/2005
  */
  //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;

         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(/*new ImageIcon(
                                        "images/imagInteg.jpg")*/);
                  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)"));
                     funcionText=new JTextField(lexema,20);
                     funcionText.addActionListener(
                            new ActionListener() {
                                   public void actionPerformed(ActionEvent e)
                                   {
                                          lexema=e.getActionCommand();
                                          panel.setLexical(lexema);
                                   }
                            }
                     );
                     p1.add(funcionText);
                     p3.add(p1);
                     getContentPane().add(p3, BorderLayout.NORTH);

                  JPanel p31=new JPanel();
                     p31.add(new JLabel("a"));
                     JTextField aText=new JTextField("0.0",6);
                     aText.addActionListener(
                            new ActionListener() {
                                   public void actionPerformed(ActionEvent e)
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
                                {
                                        limiteA=Double.valueOf(
                                                e.getActionCommand()).
                                                doubleValue();
                                        panel.setA(limiteA);
                                }
                        }
                );
                p31.add(aText);
                p3.add(p31);

                JPanel p32=new JPanel();
                p32.add(new JLabel("b"));
                JTextField bText=new JTextField("4.0",6);
                bText.addActionListener(
                        new ActionListener() {
                                public void actionPerformed(ActionEvent e)
                                {
                                        limiteB=Double.valueOf(
                                                e.getActionCommand()).
                                                doubleValue();
                                        panel.setB(limiteB);

                                }
                        }
                );
                p32.add(bText);
                p3.add(p32);

                //getContentPane().add(p3, BorderLayout.NORTH);
                goOK = new JButton("GO");
                ActionListener rst = new ActionListener(){
                public void actionPerformed(ActionEvent e){
                panel.graph();
                //getContentPane().add(panel, BorderLayout.CENTER);
                }
                };
                goOK.addActionListener(rst);
                p = new JPanel();
                p.add(goOK);
                btOK = new JButton("OK");
                ActionListener lst = new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                    dispose();
                  }
                };
                btOK.addActionListener(lst);

                p.add(btOK);
                getRootPane().setDefaultButton(btOK);
                getRootPane().registerKeyboardAction(lst,
                        KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0),
                        JComponent.WHEN_IN_FOCUSED_WINDOW);
                getContentPane().add(p, BorderLayout.SOUTH);

                WindowListener wl = new WindowAdapter() {
                        public void windowOpened(WindowEvent e) {
                                btOK.requestFocus();
                        }
                };
                addWindowListener(wl);

                pack();
                setResizable(false);
                setLocationRelativeTo(owner);
        }
} // fin de class GraphDialog
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
// 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();
	}
	// graficando
	public void graph()
	{
		points=new ArrayList();
		lines=new ArrayList();

		computePoints();
		computeScala();
		last=new Point2D.Double(
			X0+scalaX*(((Point2D)points.get(0)).getX()-a),
			Y0+ALTO-scalaY*(((Point2D)points.get(0)).getY()-min));

		for(int i=1;i<points.size();i++)
			add(i);
	}
	public void add(int i)
	{
		Point2D end=new Point2D.Double(
			X0+scalaX*(((Point2D)points.get(i)).getX()-a),
			Y0+ALTO-scalaY*(((Point2D)points.get(i)).getY()-min));
		Line2D line=new Line2D.Double(last,end);
		lines.add(line);
		repaint();
		last=end;
	}

	public void paintComponent(Graphics g)
	{
		super.paintComponent(g);
		Graphics2D g2=(Graphics2D) g; // convertir g a Graphics2D
		axisText(g2);
		g2.setColor(Color.red); //establecer color a ejes

		for(int i=0;i<lines.size();i++)
			g2.draw((Line2D)lines.get(i));
	}
	//calcular puntos a graficar
	public void computePoints()
	{
		Parser p=new Parser(lexema);
		double x=a,y;

		while(x<=b){
			y=p.getValue(x);
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
                        Point2D xy=new Point2D.Double(x,y);
                        points.add(xy);
                        x+=Dx;
                }

        }
        //calcular escala
        public void computeScala()
        {
                min = max=((Point2D)points.get(0)).getY();
                double x=a,y;

                for(int i=1;i<points.size();i++)
                {
                        y=((Point2D)points.get(i)).getY();
                        if(y<min)
                                min=y;
                        if(y>max)
                                max=y;
                }
                scalaY=ALTO/(max-min);
                scalaX=ANCHO/(b-a);
        }
        public void setA(double a)
        {
                this.a=a;
        }
        public void setB(double b)
        {
                this.b=b;
        }
        public void setLexical(String lexema)
        {

                this.lexema=lexema;
        }
        // pintando ejes
        public void axisText(Graphics2D g2)
        {
                g2.setFont(new Font("SansSerif",Font.BOLD,12));
                g2.setColor(Color.blue);
                g2.drawString(Double.toString(a)
                                ,(int)(X0),(int)(ALTO+Y0+20));

                g2.drawString(Double.toString(b)
                                ,(int)(X0+ANCHO-20),(int)(ALTO+Y0+20));

                g2.drawString(decimales4.format(min)
                        //Double.toString(min)
                                ,(int)(X0-40),(int)(Y0+ALTO));
                g2.drawString(decimales4.format(max)
                        //Double.toString(max)
                                ,(int)(X0-40),(int)Y0);
                // frame
                g2.draw(new Line2D.Double(X0,Y0,X0+ANCHO,Y0));
                g2.draw(new Line2D.Double(X0,Y0,X0,Y0+ALTO));
            g2.draw(new Line2D.Double(X0,Y0+ALTO,X0+ANCHO,Y0+ALTO));
            g2.draw(new Line2D.Double(X0+ANCHO,Y0+ALTO,X0+ANCHO,Y0));

        }
        //opcional
        public Dimension getPreferredSize()
        {
                return new Dimension(500,350);
        }

}       // fin de GraphDialog
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

BIBLIOGRAFIA

1. AHO Alfred V., SETHI Ravi, ULMAN Jeffrey D. (1990). “COMPILADORES Principios, técnicas y herramientas” 1ra.edición, Addison-Wesley Iberoamericana, S.A. E.U.A. 820pp.
2. ATKINSON L.V. y HARLEY P.J. (1987). “INTRODUCCIÓN A LOS MÉTODOS NUMÉRICOS CON PASCAL” 1ra. edición, Addison-Wesley Iberoamericana, S.A, E.U.A. 305pp
3. BURDEN Richard L. y FAIRES J. Douglas, (2002). “ANÁLISIS NÚMERICO” 7ma. edición, International Thomsom Editores, S.A. de México, 839pp.
4. CHAPMAN Stephen J., (2004). “JAVA FOR ENGINEERS AND SCIENTISTS” First edition, Pearson Education Inc. U.S.A. 676pp.
5. CHAPRA S. & CANALE R., (1999) “MÉTODOS NUMÉRICOS PARA INGENIEROS” 4ta. edición, Editorial McGrawHill, México, 992pp.
6. DAVIS Philip J. & RABINOWITZ Philip; (1975). “METHODS OF NUMERICAL INTEGRATION” First edition, Academic Press, Inc. U.S.A. 459pp.
7. DEITEL Harvey y DEITEL Paul; (2003). “JAVA HOW TO PROGRAM” Fith Edition, USA. 1536pp.
8. MATHEWS John H. y FINK Kurtis D.; (2000). “MÉTODOS NUMÉRICOS CON MATLAB” 3ra. edición, Prentice-Hall Iberia S.R.L. Madrid 721pp.
9. NAKAMURA Shoichiro, (1992). “MÉTODOS NUMÉRICOS APLICADOS CON SOFTWARE” 1ra. edición, Prentice-Hall Hispanoamericana, S.A. de México 570pp.
10. PALMER Grant, (2003). “TECHNICAL JAVA Developing Scientific and Engineering Applications” First edition, Pearson Education Inc. U.S.A. 496pp.
11. PRESS, William H., TEUKOLSKY Saul A., VETTERLING William T., FLANNERY Brian P.; (2002). “NUMERICAL RECIPES IN C++ The art of Scientific Computing” Second Edition, Cambridge University Press UK. 1002pp.
12. PRESSMAN Roger; (2002). “INGENIERIA DE SOFTWARE, Un enfoque practico” 5ta. edición, McGraw-Hill Hispanoamericana de España S.A. 601pp.
13. SOMMERVILLE Ian; (2002). “INGENIERIA DE SOFTWARE” 6ta edición, Adisson Wesley - Pearson Educación de México S.A. 712pp
14. WEISS Mark Allen; (2000). “ESTRUCTURA DE DATOS EN JAVA™” 1ra. edición, Pearson Educación, S.A. 740pp.

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

CAPITULO 1: INTRODUCCIÓN

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.

En su elaboración se han tenido en cuenta los principios de la ingeniería de software a fin de obtener un software de calidad. Se ha empleado como software de base para escribir el código de los programas: el lenguaje de programación Java; implementando con el, herramientas y técnicas proporcionadas por la teoría de compiladores, los algoritmos, las estructuras de datos y los métodos numéricos. Adicionalmente se ha escrito un ambiente que permite la graficación o ploteo de funciones. Todo el desarrollo se ha realizado en un entrono visual proporcionado por las facilidades de Java Swing.

El software ha sido probado en cuanto a sus resultados teniendo funciones cuyos resultados son conocidos y presentados en la literatura sobre métodos numéricos y contrastados con los obtenidos o devueltos por IntegraLAB, estableciendo las comparaciones del caso, habiéndose encontrado precisión y exactitud con los resultados esperados.

Por ello se estima que el trabajo satisface los objetivos y requerimientos planteados al inicio del proyecto de investigación y desarrollo propuesto a finales del año 2005.

Definición del problema

El problema a resolver 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.

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.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Objetivos específicos

1. Escribir un software de calidad acorde con los principios de la Ingeniería de software.

2. Simplificar la labor de 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 solo unos segundos, reduciendo sustancialmente las tareas de cálculo o proceso.

Justificación del sistema

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 y la ingeniería del software.

Requerimientos del software

1. El programa debe operar en una PC compatible 486 o superior.
2. El sistema operativo de base será Windows 98 o superior.
3. El programa debe poseer facilidad de uso

Funciones de los subsistemas del software

Los subsistemas componentes para el desarrollo de la tesis son: hardware, software y personal.

Hardware

Tiene la función de proporcionar el soporte físico al sistema y consta de los siguientes dispositivos: CPU, monitor, teclado, mouse, impresora.

Software

El software creado tiene por nombre IntegraLAB. La codificación del programa fuente se hace utilizando el lenguaje de programación JAVA™, de Sun Microsystems, teniendo como interfase de desarrollo el JCreator 3.0 LE.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Personal

El desarrollo del proyecto esta a cargo del tesista.

Características 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 las siguientes características:

1. Conocimiento de los métodos numéricos y sus aplicaciones, lo cual implica también conocimiento básico del cálculo.
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 grafico 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 sistema.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

Capitulo 2: MARCO TEÓRICO

2.0 Integración numérica

Las fórmulas de integración numérica se obtienen integrando polinomios que de alguna manera aproximan a una función [ATKINSON].

A menudo es necesario evaluar la integral definida de una función que no tiene una antiderivada explícita, o cuya antiderivada no es fácil de obtener. El método básico con que se aproxima $\int_a^b f(x)dx$ recibe el nombre de **cuadratura numérica** y emplea una suma del tipo $\sum_{i=0}^{n} a_i f(x_i)$ para aproximar $\int_a^b f(x)dx$ [NAKAMURA].

2.1 Métodos de Newton-Cotes

Una forma evidente de obtener la fórmula de integración es integrar un polinomio de integración. Por ejemplo [ATKINSON], supóngase que se puede conseguir una fórmula integrando el polinomio de interpolación de Lagrange *lineal*. Si se define $x_0 \equiv a$ y $x_1 \equiv b$, entonces se tiene

$$\int_a^b f(x)dx \equiv \int_{x_0}^{x_1} f(x)dx$$

$$\approx \int_{x_0}^{x_1} \left[\frac{(x-x_1)}{(x_0-x_1)} f(x_0) + \frac{(x-x_0)}{(x_1-x_0)} f(x_1) \right] dx$$

$$= \frac{(x_1-x_0)}{2} \left[f(x_0) + f(x_1) \right]$$

Si se hace $h = b - a \equiv x_1 - x_0$, entonces esto puede reescribirse como

$$\int_{x_0}^{x_1} f(x)dx \approx \frac{h}{2}(f_0 + f_1) \qquad (2.1)$$

donde, como es usual, $f_0 \equiv f(x_0)$ y $f_1 \equiv f(x_1)$. La fórmula 2.1, se conoce como la ***regla trapezoidal***. El motivo del nombre es evidente, ya que el lado derecho no es más que el área bajo la línea que une a f_0 y f_1; esto es, un trapecio siempre que f_0 y f_1 tengan el mismo signo.

Como es de esperar, puede conseguirse una fórmula más exacta partiendo el intervalo $[a,b]$ en dos subintervalos iguales con un tamaño de

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

$h=(b-a)/2$ e integrando el polinomio cuadrático de interpolación. Si se define $x_0 \equiv a$, $x_1 = a+h$ y $x = a+2h \equiv b$, entonces se tiene que

$$\int_a^b f(x)dx \equiv \int_{x_0}^{x_2} f(x)dx$$

$$\approx \int_{x_0}^{x_2} \left[\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_1-x_2)} f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f(x_1) + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f(x_2) \right] dx$$

$$\approx \frac{h}{3}\left[f_0 + 4f_1 + f_2\right] \qquad (2.2)$$

La fórmula (2.2) es conocida como la regla de ***Simpson 1/3***.

Con un razonamiento similar partiendo el intervalo $[a,b]$ en tres subintervalos iguales con un tamaño de $h=(b-a)/3$ e integrando el polinomio de interpolación. Se tiene

$$\int_a^b f(x)dx \equiv \int_{x_0}^{x_3} f(x)dx$$

$$\approx \frac{3h}{8}\left[f_0 + 3f_1 + 3f_2 + f_3\right] \qquad (2.3)$$

La fórmula 2.3 es conocida como la regla de ***Simpson 3/8***.

Cuando el intervalo $[a,b]$ se parte en cuatro subintervalos iguales con un tamaño de $h=(b-a)/4$ e integrando el polinomio de interpolación se tiene

$$\int_a^b f(x)dx \equiv \int_{x_0}^{x_4} f(x)dx$$

$$\approx \frac{2h}{45}\left[7f_0 + 32f_1 + 12f_2 + 32f_3 + 7f_4\right] \qquad (2.4)$$

donde la fórmula (2.4) es conocida como la Regla de Boole. [MATHEWS]

Las fórmulas (2.1, 2.2, 2.3, y 2.4) son las ***fórmulas cerradas de Newton-Cotes***.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Pueden lograrse fórmulas para el error que se obtiene de los resultados de estas técnicas de integración numérica.

En [MATHEWS] se tiene el siguiente corolario, respecto a la precisión de las fórmulas de Newton-Cotes

Corolario 2.1 (Precisión de las fórmulas de Newton-Cotes). Supongamos que *f*(*x*) es suficientemente derivable; entonces el término del error de truncamiento *E*[*f*] de las fórmulas de Newton-Cotes contiene una derivada de orden superior adecuada evaluada en un cierto punto $c \in (a,b)$. La regla del trapecio tiene un grado de precisión de *n* = 1 y si $f \in C^2[a,b]$, entonces

$$\int_{x_0}^{x_1} f(x)dx = \frac{h}{2}(f_0 + f_1) - \frac{h^3}{12} f^{(2)}(c) \qquad (2.5)$$

La regla de Simpson tiene un grado de precisión de *n* = 3 y si $f \in C^4[a,b]$, entonces

$$\int_{x_0}^{x_2} f(x)dx = \frac{h}{3}[f_0 + 4f_1 + f_2] - \frac{h^5}{90} f^{(4)}(c) \qquad (2.6)$$

La regla de Simpson a 3/8 tiene un grado de precisión de *n* = 3 y si $f \in C^4[a,b]$, entonces

$$\int_{x_0}^{x_3} f(x)dx = \frac{3h}{8}[f_0 + 3f_1 + 3f_2 + f_3] - \frac{3h^5}{80} f^{(4)}(c) \qquad (2.7)$$

La regla de Boole tiene un grado de precisión de *n* = 5 y si $f \in C^6[a,b]$, entonces

$$\int_{x_0}^{x_4} f(x)dx = \frac{2h}{45}[7f_0 + 32f_1 + 12f_2 + 32f_3 + 7f_4] - \frac{8h^7}{945} f^{(6)}(c) \qquad (2.8)$$

2.2 Métodos de Gauss-Legendre [NAKAMURA]

Las cuadraturas de Gauss-Legendre (o simplemente de Gauss) son métodos de integración numérica que utilizan puntos de Legendre (raíces de polinomios de Legendre). Las cuadraturas de Gauss no se pueden utilizar para integrar una función dada en forma de tabla con intervalos de separación uniforme debido a que los puntos de Legendre no están separados de esa

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

manera; sin embargo, son más adecuadas para integrar funciones analíticas. La ventaja de las cuadraturas de Gauss es que su precisión es mayor que la de las fórmulas de Newton-Cotes.

Las cuadraturas de Gauss difieren en forma significativa de las fórmulas de Newton-Cotes ya que los *N* puntos de la retícula (llamados puntos de Gauss) se obtienen mediante raíces del polinomio de Legendre $P_N(x)=0,$ donde $P_N(x)$ es el polinomio de Legendre de orden *N*.

Los polinomios de Legendre en los que se basan las cuadraturas de Gauss Legendre, se escriben como

$$\begin{aligned} P_0(x) &= 1 \\ P_1(x) &= x \\ P_2(x) &= \frac{1}{2}\left[3x^2-1\right] \\ P_3(x) &= \frac{1}{2}\left[5x^3-3x\right] \\ P_4(x) &= \frac{1}{8}\left[35x^4-30x^2+3\right] \\ &\vdots \end{aligned} \qquad (2.9)$$

Cualquier polinomio de Legendre de grado superior puede obtenerse utilizando la fórmula de recursión

$$nP_n(x)-(2n-1)xP_{n-1}(x)+(n-1)P_{n-2}(x)=0 \qquad (2.9)$$

De manera alternativa, un polinomio de Legendre de orden *n* también se puede expresar como

$$P_n(x)=\frac{1}{2^n n!}\frac{d^n(x^2-1)^n}{dx^n} \qquad (2.10)$$

Una propiedad importante de los polinomios de Legendre es la ortogonalidad:

$$\int_{-1}^{1} P_m(x)P_n(x)dx = 0, \text{ para } n\neq m \qquad 2.11$$

$$= \frac{2}{2n+1}, \text{ para } n=m$$

La ecuación anterior indica que la integral en [-1,1] de dos polinomios de Legendre distintos es igual a cero.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Todo polinomio de orden menor o igual que $N - 1$ es ortogonal al polinomio de Legendre de orden N. Esto se puede demostrar fácilmente, ya que un polinomio de orden menor o igual que $N - 1$ se puede expresar como una combinación lineal de polinomios de Legendre a lo más $N - 1$.

La cuadratura de Gauss que se extiende sobre el intervalo [-1, 1] está dada por

$$\int_{-1}^{1} f(x)dx = \sum_{k=1}^{N} w_k f(x_k) \qquad (2.12)$$

donde N es el numero de puntos de Gauss, los w_i son los pesos y las x_i son los puntos de Gauss dados en la Tabla 2.1. Los signos $\pm$ en la tabla significan que los valores de x de los puntos de Gauss aparecen son pares, uno de los cuales es positivo y el otro negativo. Por ejemplo, si $N = 4$, la ecuación (2.12) es

$$\int_{-1}^{1} f(x)dx = 0.34785 f(-0.86113) + 0.65214 f(-0.33998)$$

$$+ 0.65214 f(0.33998) + 0.34785 f(0.86113) \qquad (2.13)$$

Tabla 2.1 Puntos de Gauss y pesos (Ver Abramowitz y Stegun para una tabla más completa)

	$\pm x_i$	w_i
$N = 2$	0.577350269	1.00000000
$N = 3$	0	0.888888889
	0.774596669	0.555555556
$N = 4$	0.339981043	0.652145155
	0.861136312	0.347854845
$N = 5$	0	0.568888889
	0.538469310	0.478628670
	0.906179846	0.236926885
$N = 6$	0.238619186	0.467913935
	0.661209387	0.360761573
	0.932469514	0.171324492
$N = 8$	0.183434642	0.362683783
	0.525532410	0.313706646
	0.796666478	0.222381034
	0.960289857	0.101228536
$N = 10$	0.148874339	0.295524225
	0.433395394	0.269266719
	0.679409568	0.219086363
	0.865063367	0.149451349
	0.973906528	0.066671344

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

La fórmula de Gauss puede aplicarse a cualquier intervalo arbitrario $[a,b]$ con la transformación

$$X = \frac{2z - a - b}{b - a} \qquad (2.14)$$

donde *z* es la coordenada original en $a < z < b$ y *x* es la coordenada normalizada en $-1 \le x \le 1$. La transformación de *x* en *z* es

$$z = \frac{(b-a)x + a + b}{2} \qquad (2.15)$$

Por medio de esta transformación, la integral se puede escribir como

$$\int_a^b f(z)dz = \int_{-1}^{1} f(z)(dz/dx)dx = \frac{b-a}{2}\sum_{k=1}^{N} w_k f(z_k) \qquad (2.16)$$

donde $dz/dx = (b-a)/2$. Los valores de z_k se obtienen al sustituir *x* en la ecuación (2.15) por lo puntos de Gauss; a saber

$$z_k = \frac{(b-a)x_k + a + b}{2} \qquad (2.17)$$

Por ejemplo, supongamos que $N = 2$, $a = 0$ y $b = 2$. Puesto que los puntos de Gauss x_k para $N = 2$ en la coordenada normalizada *x*, $-1 \le x \le 1$, son ± 0.57735 (de la tabla 2.1), los puntos correspondientes en *z* son

$$z_1 = \frac{1}{2}[(2-0)(-0.57735) + 0 + 2] = 0.42265 \qquad (2.18)$$
$$z_2 = \frac{1}{2}[(2-0)(0.57735) + 0 + 2] = 1.57735$$

La derivada de $dz/dx = (b-a)/2 = 1$. Por lo tanto, la cuadratura de Gauss se escribe como

$$\int_0^2 f(z)dz = \int_{-1}^{1} f(z)(dz/dx)dx = (1)[(1)f(0.42264) + (1)f(1.57735)] \qquad 2.19$$

La tabla 2.1 indica que todos los pesos son positivos. Una ventaja de la cuadratura de Gauss es que no existen problemas con el error de redondeo, a menos que el integrando cambie de signo a mitad del dominio, debido a que no se realizan restas de números grandes como en el caso de las fórmulas de Newton-Cotes. Otra ventaja de las cuadraturas de Gauss es que se puede

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

integrar una función con una singularidad en uno o en ambos límites de integración, debido a que no se necesitan los valores en los límites.

En un intervalo de integración grande, el dominio se puede dividir en cierto número de intervalos pequeños y aplicar varias veces la cuadratura de Gauss en cada subintervalo. La idea es la misma que en las reglas extendidas del trapecio y de Simpson.

2.3 Otras cuadraturas de Gauss [NAKAMURA]

Las cuadraturas de Gauss analizadas en la sección anterior se llaman cuadraturas de Gauss-Legendre porque se basan en la ortogonalidad de los polinomios de Legendre. Existen cuadraturas análogas con base en los polinomios de Hermite, de Laguerre [Froeberg] y de Chebyshev, las cuales reciben los nombres de cuadratura de Gauss-Hermite, Gauss-Laguerre y Gauss-Chebyshev, respectivamente.

Las cuadraturas de Gauss-Hermite son adecuadas para

$$\int_{-\infty}^{\infty} \exp(-x^2) f(x) dx \qquad (2.20)$$

y están dadas por

$$\int_{-\infty}^{\infty} \exp(-x^2) f(x) dx = \sum_{k=1}^{N} w_k f(x_k) \qquad (2.21)$$

En la ecuación (2.21), los x_k son raíces del polinomio de Hermite de orden *N* y los w_k son los pesos, algunos de los cuales se presentan en la tabla 2.2

Tabla 2.2: Puntos de Hermite y sus pesos

	$\pm x_i$	w_i
N = 2	0.70710678	0.8862692
N = 3	0.00000000 1.22474487	1.18163590 0.29540897
N = 4	0.52464762 1.65068012	0.80491409 0.08131283
N = 5	0.00000000 0.95857246 2.02018287	0.94530872 0.39361932 0.01995324

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

Las cuadraturas de Gauss-Laguerre son adecuadas para

$$\int_0^\infty \exp(-x) f(x) dx \qquad (2.22)$$

y están dadas por

$$\int_0^\infty \exp(-x) f(x) dx = \sum_{k=1}^{N} w_k f(x_k) \qquad (2.23)$$

donde los x_k son las raíces del polinomio de Laguerre de orden *N* y los w_k son los pesos, algunos de los cuales se muestran en la tabla 2.3

Tabla 2.3: Puntos de Laguerre y sus pesos

	x_i	w_i
N = 2	0.58578643 3.41421356	0.85355339 0.14644660
N = 3	0.41577455 2.29428036 6.28994508	0.71109300 0.27851773 0.10389256*E* - 1
N = 4	0.32254768 1.74576110 4.53662029 9.39507091	0.60315410 0.35741869 0.3887908*E* - 1 0.53929470*E* - 3

La cuadraturas de Gauss-Chebyshev son adecuadas para

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x) dx \qquad (2.24)$$

y están dadas por

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x) dx = \sum_{k=1}^{N} w_k f(x_k) \qquad (2.25)$$

En la ecuación (2.25), los x_k son las raíces del polinomio de Chebyshev de orden *N* y los w_k son los pesos. Las raíces del polinomio de Chebyshev de orden *N* son

$$x_k = \cos\frac{k-1/2}{N}\pi, \quad k = 1,2,...,N \qquad (2.26)$$

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Los pesos son

$$w_k = \frac{\pi}{N} \quad \text{para toda } k \tag{2.27}$$

Así la ecuación (2.25) se reduce a

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x)dx = \frac{\pi}{N} \sum_{k=1}^{N} f(x_k) \tag{2.28}$$

Los límites de integración [-1, 1] se pueden cambiar a un dominio arbitrario mediante la ecuación (2.15).

Las tres cuadraturas de Gauss explicadas en esta subsección son exactas si *f*(*x*) es un polinomio de orden menor igual que 2*N* - 1.

2.4 Ecuaciones Diferenciales Ordinarias[NAKAMURA]

Los problemas de ecuaciones diferenciales ordinarias (EDO) se clasifican en problemas con condiciones iniciales y problemas con condiciones en la frontera. Muchos de los problemas con condiciones iniciales dependen del tiempo; en ellos, las condiciones para la solución están dadas en el tiempo inicial. Los métodos numéricos para los problemas con condiciones iniciales difieren en forma significativa de los que se utilizan para los problemas con condiciones en la frontera. En el trabajo presentado, se examina los problemas con condiciones iniciales.

El problema con condiciones iniciales de una EDO de primer orden se puede escribir en la forma

$$y'(t))f(y,t), \quad y(0) = y_0 \tag{2.29}$$

donde $f(t,y)$ es una función de *y* en tanto que *t* y la segunda ecuación es una condición inicial. En la ecuación (2.29) la primera derivada de *y* esta dada como una función conocida de y y t y queremos calcular la función incógnita y integrando numéricamente $f(t,y)$. Si f fuera independiente de y, el cálculo seria simplemente una de las integraciones directas vistas en las secciones anteriores. Sin embargo, el hecho de que f sea una función de la función desconocida y hace que la integración sea distinta.

2.4.1 Métodos de Euler

Los métodos de Euler son adecuados para una programación rápida debido a su sencillez. Sin embargo, es necesario precisar que el método acumula errores apreciables a lo largo del proceso, pero constituye un método de partida para métodos más precisos y exactos. Los métodos de Euler tienen

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

tres versiones: a) Euler hacia adelante, b) Euler modificado y c) Euler hacia atrás.

a) Método de Euler hacia adelante

El método de Euler hacia adelante para la ecuación $y' = f(y,t)$ se obtiene reescribiendo la aproximación por diferencias hacia delante,

$$\frac{y_{n+1} - y_n}{h} \approx y_n' \qquad (2.30)$$

como

$$y_{n+1} = y_n + hf(y_n, t_n) \qquad (2.31)$$

donde se usa $y_n' = f(y_n, t_n)$. Mediante la ecuación (2.31), se calcula y_n en forma recursiva como

$$y_1 = y_0 + hy_0' = y_0 + hf(y_0, t_0)$$
$$y_2 = y_1 + hf(y_1, t_1)$$
$$y_3 = y_2 + hf(y_2, t_2)$$
$$\vdots$$
$$y_n = y_{n-1} + hf(y_{n-1}, t_{n-1})$$

Aunque el método de Euler hacia delante es muy sencillo, debe utilizarse cuidadosamente para evitar dos tipos de errores. El primer tipo lo forman los errores de truncamiento, el segundo tipo lo constituye la posible inestabilidad, que aparece cuando la constante de tiempo es negativa (la solución tiende a cero si no hay término fuente), a menos de que el intervalo de tiempo *h* sea suficientemente pequeño.

b) Método de Euler modificado

El método de Euler modificado tiene dos motivaciones. La primera es que es más preciso que el anterior. La segunda es que es más estable.

Este método se obtiene al aplicar la regla del trapecio para integrar $y' = f(y,t)$:

$$y_{n+1} = y_n + \frac{h}{2}\left[f(y_{n+1}, t_{n+1}) + f(y_n, t_n)\right] \qquad (2.32)$$

Si f es lineal en y, la ecuación (2.32) se puede resolver fácilmente en términos de y_{n+1}.

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Si f es una función no lineal de y, la ecuación (2.32) es una función no lineal de y_{n+1}, por lo que se requiere un algoritmo para resolver ecuaciones no lineales. Uno de los métodos ampliamente utilizados para resolver ecuaciones no lineales es el de la sustitución sucesiva:

$$y_{n+1}^{(k)} - y_n = \frac{h}{2}\left[f(y_{n+1}^{(k-1)}, t_{n+1}) + f(y_n, t_n)\right] \qquad (2.33)$$

donde $y_{n+1}^{(k)}$ es la *k*-ésima aproximación iterativa de y_{n+1}, y y_{n+1}^{0} es una estimación inicial de y_{n+1}. Esta iteración se obtiene si $\left|y_{n+1}^{(k)} - y_{n+1}^{(k-1)}\right|$ es menor que una cierta tolerancia preestablecida. Entonces en el primer paso de iteración, es idéntico al método de Euler hacia delante. En el caso de que sólo se utilice un paso más de iteración, el esquema se convierte en el método de Runge-Kutta de secundo orden o, en forma equivalente, en el método predictor-corrector de Euler. Pero en el método de Euler modificado, la iteración sigue hasta satisfacer la tolerancia.

c) Método de Euler hacia atrás

Este método se basa en aproximación por diferencias hacia atrás y se escribe como

$$y_{n+1} = y_n + hf(y_{n+1}, t_{n+1}) \qquad (2.34)$$

Si f es una función no lineal de y, debe utilizarse un esquema iterativo en cada paso, justo como en el método de Euler modificado. Sin embargo las ventajas son: a) el método es estable para los problemas rígidos, y b) la solución es positiva si la solución exacta es positiva.

El error local (generado en cada paso) en el método de Euler hacia adelante es proporcional a h^2, mientras que el error global es proporcional a h; en tanto que el error local del método de Euler modificado es proporcional a h^3 y su error global es proporcional a h^2. El orden del error del método de Euler hacia atrás es igual al del método de Euler hacia delante.

2.4.2 Método de Heun [MATHEWS]

El método de Heun, introduce una idea nueva en la construcción de un algoritmo para resolver el problema de valor inicial.

$$y'(t) = f(t, y(t)) \quad \text{en} \quad [a,b] \quad \text{con} \quad y(t_0) = y_0 \qquad (2.35)$$

Para obtener el punto (t_1, y_1), podemos usar el teorema fundamental del cálculo e integrar $y'(t)$ en $[t_0, t_1]$ de manera que

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

$$\int_{t_0}^{t_1} f(t,y(t))dt = \int_{t_0}^{t_1} y'(t)dt = y(t_1) - y(t_0), \qquad (2.36)$$

donde hemos usado como primitiva de $y'(t)$ la función deseada $y(t)$. Despejando $y(t_1)$ en la igualdad (2.36), nos queda

$$y(t_1) = y(t_0) + \int_{t_0}^{t_1} f(t,y(t))dt \qquad (2.37)$$

Ahora podríamos usar un método de integración para aproximar la integral definida en la expresión (2.37). Si usamos la regla del trapecio con incremento $h = t_1 - t_0$, entonces el resultado es

$$y(t_1) \approx y(t_0) + \frac{h}{2}(f(t_0,y(t_0)) + f(t_1,y(t_1))) \qquad (2.38)$$

Nótese que en el lado derecho de la fórmula (2.38) aparece el valor $y(t_1)$ que queremos determinar; lo que hacemos es usar una estimación de $y(t_1)$, para nuestro propósito, la aproximación de Euler es suficiente. Al sustituir esta en (2.38), obtenemos una fórmula de aproximación a $y(t_1)$ que se llama ***método de Heun***.

$$y_1 = y(t_0) + \frac{h}{2}(f(t_0,y_0) + f(t_1, y_0 + hf(t_0,y_0))) \qquad (2.39)$$

El mismo que como se ha visto en 2.4.1(b) es el método de Euler modificado.

Repitiendo el proceso en (2.39) se genera una sucesión de puntos que aproximan la solución $y = y(t)$. En cada paso la aproximación dada por el método de Euler se usa como una predicción del valor que queremos calcular y luego la regla del trapecio se usa para hacer una corrección y obtener el valor definitivo. El paso general del método de Heun es

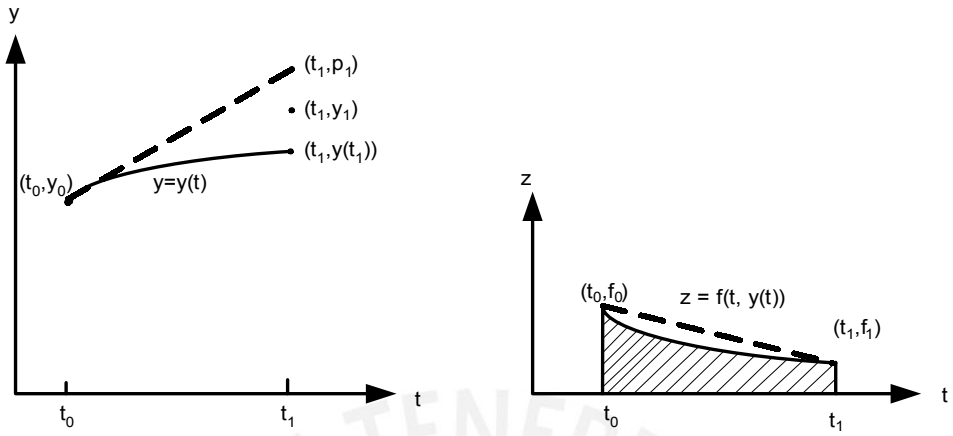
$$\begin{aligned} p_{k+1} &= y_k + hf(t_k,y_k), \qquad t_{k+1} = t_k + h, \\ y_{k+1} &= y_k + \frac{h}{2}(f(t_k,y_k) + f(t_{k+1},p_{k+1})). \end{aligned} \qquad (2.40)$$

Ahora, examinemos el papel que juegan la derivación y la integración en el método de Heun. Dibujamos la recta tangente a la gráfica de la solución $y = y(t)$ en el punto (t_0,y_0) y la usamos para construir el punto (t_1,p_1). Ahora miramos la grafica de $z = f(t,y(t))$ y consideramos los puntos (t_0,f_0) y (t_1,f_1), donde $f_0 = f(t_0,y_0)$ y $f_1 = f(t_1,p_1)$. El área del trapecio con vértices (t_0,f_0) y (t_1,f_1) se usa como aproximación de la integral que aparece en (2.37) que, a su

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

vez, se usa para obtener el valor final dado por la expresión (2.39). Las gráficas correspondientes se muestran en la figura 2.1.



(a) Valor predictor:

$p_1 = y_0 + hf(t_0, y_0)$

(b) Valor corrector:

$y_1 - y_0 = h(f_0 + f_1)/2$

Figura 2.1 Graficas $y = y(t)$ y $z = f(t, y(t))$ que se usan en la construcción del método de Heun.

2.4.3 Métodos de Runge-Kutta

Una desventaja de los métodos de Euler y Heun consiste en que los órdenes de precisión son bajos. Esta desventaja tiene dos facetas. Para mantener una alta precisión se necesita una h pequeña, lo cual aumenta el tiempo de cálculo y provoca errores de redondeo.

En los métodos de Runge-Kutta, el orden de precisión aumenta al utilizar puntos intermedios en cada intervalo. Una mayor precisión implica además que los errores decrecen más rápido al reducir h, en comparación con los métodos con precisión baja [NAKAMURA].

Los métodos de Runge-Kutta se construyen a partir de un método de Taylor, digamos de orden *N*, de tal manera que el error global final sea del mismo orden $O(h^N)$ pero se evite la evaluación de las derivadas parciales; esto puede conseguirse a cambio de evaluar, en cada paso la función en varios puntos. Aunque estos métodos se pueden construir para cualquier orden se examinan los métodos de orden *N* = 4 y el de orden *N* = 2 [MATHEWS].

a) Método de Runge-Kutta RK-4

Es el método más popular y es también, para propósitos generales, una buena elección ya que es bastante preciso, estable y fácil de programar. Aun más, la mayoría de los expertos dicen que no es necesario trabajar con métodos de orden superior porque el aumento del costo computacional no

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

compensa la mayor exactitud; si es necesario obtener mayor precisión en la solución aproximada, entonces es mejor usar un tamaño de paso menor o un método adaptativo.

El método de cuarto orden de Runge-Kutta (que llamaremos RK4) simula la precisión del método de la serie de Taylor de orden *N* = 4 y consiste en calcular la aproximación y_{k+1} de la siguiente manera:

$$y_{k+1} = y_k + w_1 k_1 + w_2 k_2 + w_3 k_3 + w_4 k_4 \tag{2.41}$$

donde k_1, k_2, k_3, k_4 son de la forma

$$\begin{aligned} k_1 &= hf(t_k, y_k), & k_3 &= hf(t_k + a_2 h, y_k + b_2 k_1 + b_3 k_2), \\ k_2 &= hf(t_k + a_1 h, y_k + b_1 k_1), & k_4 &= hf(t_k + a_3 h, y_k + b_4 k_1 + b_5 k_2 + b_6 k_3). \end{aligned} \tag{2.42}$$

Emparejando estos coeficientes con los del método de la serie de Taylor de orden *N* = 4 de manera que el error de truncamiento local sea de orden $O(h^5)$, Runge y Kutta fueron capaces de obtener el siguiente sistema de ecuaciones:

$$\begin{aligned} &b_1 = a_1, \\ &b_2 + b_3 = a_2, \\ &b_4 + b_5 + b_6 = a_3, \\ &w_1 + w_2 + w_3 + w_4 = 1, \\ &w_2 a_1 + w_3 a_2 + w_4 a_3 = \frac{1}{2}, \\ &w_2 a_1^2 + w_3 a_2^2 + w_4 a_3^2 = \frac{1}{3}, \\ &w_2 a_1^3 + w_3 a_2^3 + w_4 a_3^3 = \frac{1}{4}, \\ &w_3 a_1 b_3 + w_4 (a_1 b_5 + a_2 b_6) = \frac{1}{6}, \\ &w_3 a_1 a_2 b_3 + w_4 a_3 (a_1 b_5 + a_2 b_6) = \frac{1}{8}, \\ &w_3 a_1^2 b_3 + w_4 (a_1^2 b_5 + a_2^2 b_6) = \frac{1}{12}, \\ &w_4 a_1 b_3 b_6 = \frac{1}{24}. \end{aligned} \tag{2.43}$$

Este sistema tiene 11 ecuaciones con 13 incógnitas, así que debemos añadir dos condiciones adicionales para resolverlo. La elección más útil resulta ser

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

$$a_1 = \frac{1}{2} \qquad \text{y} \qquad b_2 = 0. \tag{2.44}$$

Entonces los valores de la solución para las demás variables son

$$a_2 = \frac{1}{2},\ a_3 = 1,\ b_1 = \frac{1}{2},\ b_3 = \frac{1}{2},\ b_4 = 0,\ b_5 = 0,\ b_6 = 1,$$
$$w_1 = \frac{1}{6},\ w_2 = \frac{1}{3},\ w_3 = \frac{1}{3},\ w_4 = \frac{1}{6}, \tag{2.45}$$

Sustituyendo en las expresiones (2.41) y (2.42) los valores dados en (2.44) y (2.45), obtenemos la fórmula para el método de Runge-Kutta de orden N = 4 estándar. A partir del punto inicial (t_0, y_0) se genera la sucesión de aproximación usando la fórmula recursiva

$$y_{k+1} = y_k + \frac{h(f_1 + 2f_2 + 2f_3 + f_4)}{6}, \tag{2.46}$$

donde

$$\begin{aligned} f_1 &= f(t_k, y_k), \\ f_2 &= f(t_k + \frac{h}{2}, y_k + \frac{h}{2} f_1), \\ f_3 &= f(t_k + \frac{h}{2}, y_k + \frac{h}{2} f_2), \\ f_4 &= f(t_k + h, y_k + h f_3). \end{aligned} \tag{2.47}$$

Resumiendo las siguientes dos versiones del método de Runge-Kutta de cuarto orden son las de uso más popular. La primera se basa en la regla de 1/3 de Simpson y se escribe como

$$\begin{aligned} k_1 &= hf(y_n, t_n) \\ k_2 &= hf(y_n + \frac{k_1}{2}, t_n + \frac{h}{2}) \\ k_3 &= hf(y_n + \frac{k_2}{2}, t_n + \frac{h}{2}) \\ k_4 &= hf(y_n + k_3, t_n + h) \\ y_{n+1} &= y_x + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4] \end{aligned}$$

La segunda versión se basa en la regla de 3/8 de Simpson y se expresa como

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

$$k_1 = hf(y_n, t_n)$$
$$k_2 = hf(y_n + \frac{k_1}{3}, t_n + \frac{h}{3})$$
$$k_3 = hf(y_n + \frac{k_1}{2} + \frac{k_2}{3}, t_n + \frac{2h}{3})$$
$$k_4 = hf(y_n + k_1 - k_2 + k_3, t_n + h)$$
$$y_{n+1} = y_n + \frac{1}{8}[k_1 + 3k_2 + 3k_3 + k_4]$$

b) Método de Runge-Kutta RK-2 [MATHEWS]

El método de Runge-Kutta de segundo orden, llamado RK2, simula la precisión del método de serie de Taylor de orden *N* = 2. Aunque no es un método tan bueno como el RK4, los razonamientos que nos conducen a su desarrollo son más fáciles de entender y sirven para ilustrar las ideas involucradas en los métodos de Runge-Kutta. Empezamos escribiendo el desarrollo en serie de Taylor para $y(t+h)$:

$$y(t+h) = y(t) + hy'(t) + \frac{1}{2}h^2 y''(t) + C_T h^3 + \ldots, \qquad (2.48)$$

donde C_T es una constante que incluye la derivada tercera de $y(t)$ y los demás términos corresponden a las potencias h^j para $j > 3$.

Vamos a expresar las derivadas $y'(t)$ e $y''(t)$ de la ecuación (2.47) con respecto a t usando la regla de la cadena para funciones de dos variables, obtenemos

$$y''(t) = f_t(t,y) + f_y(t,y)y'(t).$$

Esta podemos escribirlo, usando la igualdad (2.48), como

$$y''(t) = f_t(t,y) + f_y(t,y)f(t,y). \qquad (2.49)$$

Sustituyendo las derivadas (2.48) y (2.49) en la expresión (2.47), obtenemos una nueva expresión del desarrollo de Taylor para $y(t+h)$:

$$y(t+h) = y(t) + hf(t,y) + \frac{1}{2}h^2 f_t(t,y) + \frac{1}{2}h^2 f_y(t,y)f(t,y) + C_T h^3 + \ldots \qquad (2.50)$$

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

En el método de Rungen-Kutta de orden N = 2 se utiliza una combinación lineal de dos funciones que nos permita expresar $y(t+h)$:

$$y(t+h) = y(t) + Ahf_0 + Bhf_1, \tag{2.52}$$

donde

$$\begin{aligned} f_0 &= f(t,y), \\ f_1 &= f(t+Ph, y+Qhf_0). \end{aligned} \tag{2.53}$$

Una forma canónica para el método de RK2 es

$$\begin{aligned} k_1 &= hf(y_n, t_n) \\ k_2 &= hf(y_n + k_1, t_{n+1}) \\ y_{n+1} &= y_n + \frac{1}{2}[k_1 + k_2] \end{aligned} \tag{2.54}$$

El método de Runge-Kutta de segundo orden es idéntico al método predictor-corrector de Heun, también es equivalente al método modificado de Euler, con únicamente dos pasos de iteración.

2.5 Sistemas de Ecuaciones Diferenciales

La aplicación del método de Runge-Kutta de cuarto orden a un conjunto de ecuaciones diferenciales ordinarias es análoga a la aplicación del método de segundo orden. Con el fin de simplificar la explicación, consideremos un conjunto de dos ecuaciones:

$$\begin{aligned} y' &= f(y,z,t) \\ z' &= g(y,z,t) \end{aligned} \tag{2.55}$$

El método de Runge-Kutta de cuarto orden para este conjunto es:

$$\begin{aligned} k_1 &= hf(y_n, z_n, t_n) \\ l_1 &= hg(y_n, z_n, t_n) \\ k_2 &= hf(y_n + \frac{k_1}{2}, z_n + \frac{l_1}{2}, t_n + \frac{h}{2}) \\ l_2 &= hg(y_n + \frac{k_1}{2}, z_n + \frac{l_1}{2}, t_n + \frac{h}{2}) \\ k_3 &= hf(y_n + \frac{k_2}{2}, z_n + \frac{l_2}{2}, t_n + \frac{h}{2}) \\ l_3 &= hg(y_n + \frac{k_2}{2}, z_n + \frac{l_2}{2}, t_n + \frac{h}{2}) \\ k_4 &= hf(y_n + k_3, z_n + l_3, t_n + h) \\ l_4 &= hg(y_n + k_3, z_n + l_3, t_n + h) \end{aligned} \tag{2.56}$$

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

$$y_{n+1} = y_n + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4] \quad (2.57)$$

$$z_{n+1} = z_n + \frac{1}{6}[l_1 + 2l_2 + 2l_3 + l_4] \quad (2.58)$$

Incluso cuando el número de ecuaciones es un conjunto mayor que dos, el método de Runge-Kutta de cuarto orden es esencialmente el mismo.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

CAPITULO 3: 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 contiene las siguientes opciones:

Archivo
Edición
Integración
EDO's
Ayuda

Al respecto ver la figura 3.1

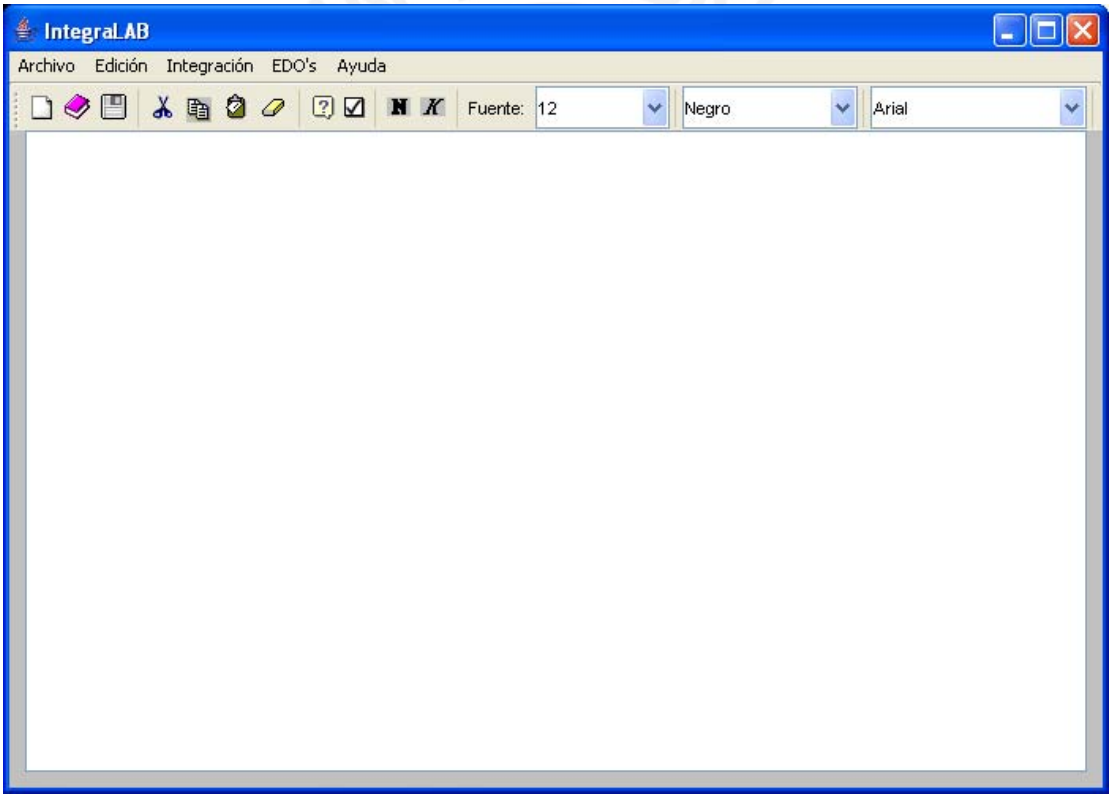


Figura 3.1: Ventana principal de IntegraLAB

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.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

El menú 4: EDO's; contiene las opciones: Básicas, y Sistemas EDO's.

El menú 5: Ayuda; contiene las opciones: Ayuda y Acerca de.

Las figuras 3.2, 3.3, 3.4 y 3.5 muestran las ventanas de los menús 3 y 4

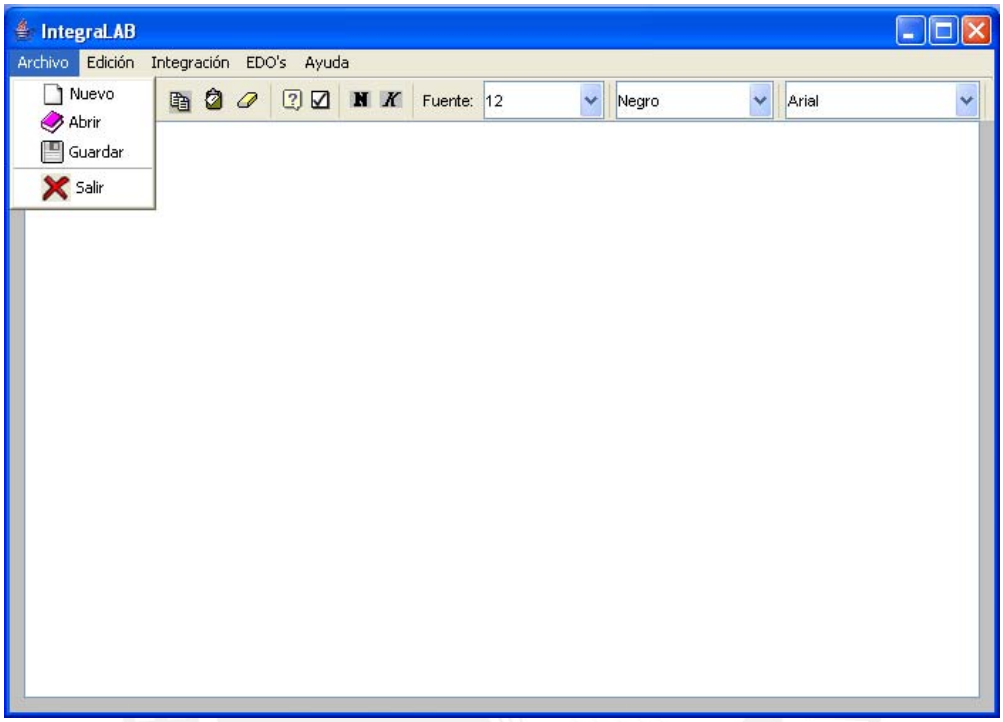


Figura 3.2: Menú Archivo

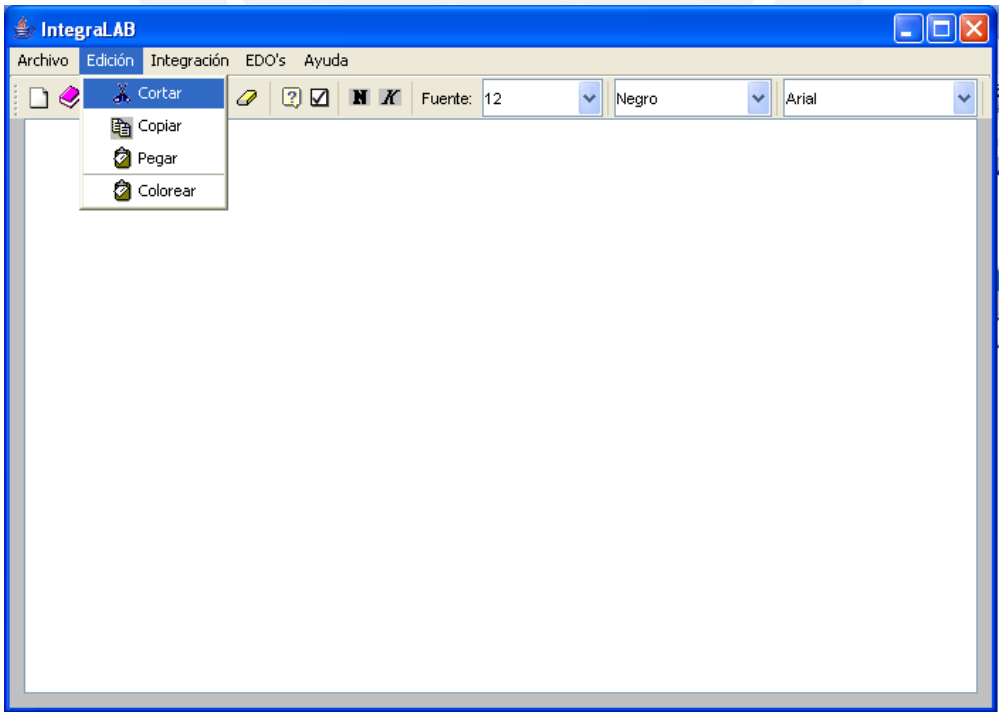


Figura 3.3: Menú Edición

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

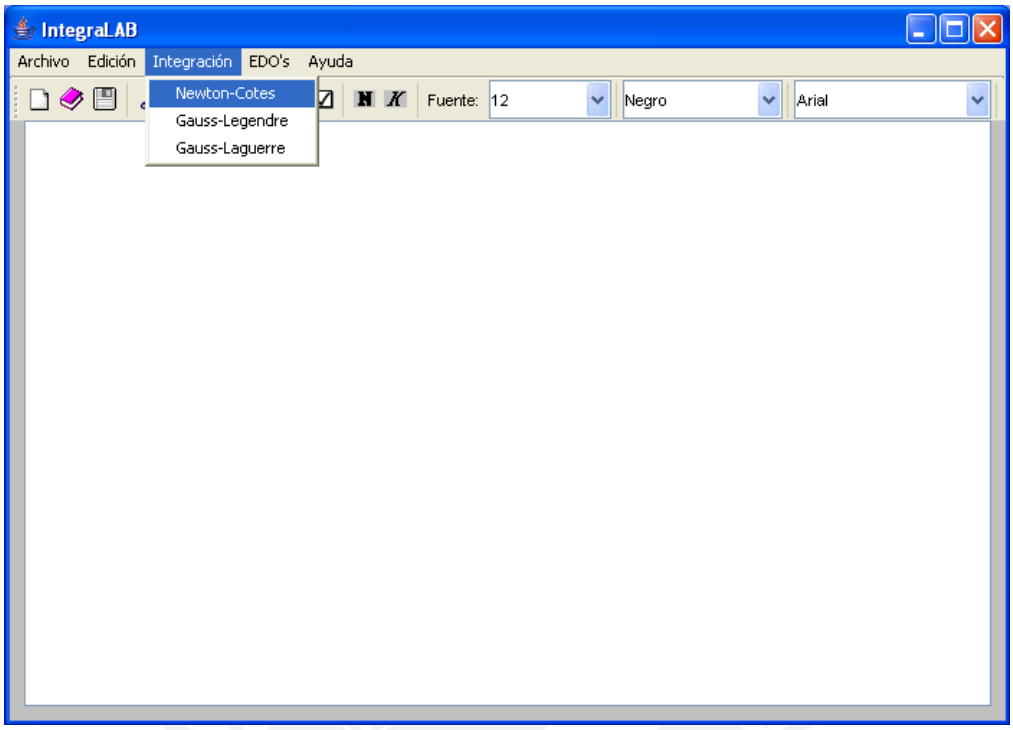


Figura 3.4: Menú integración

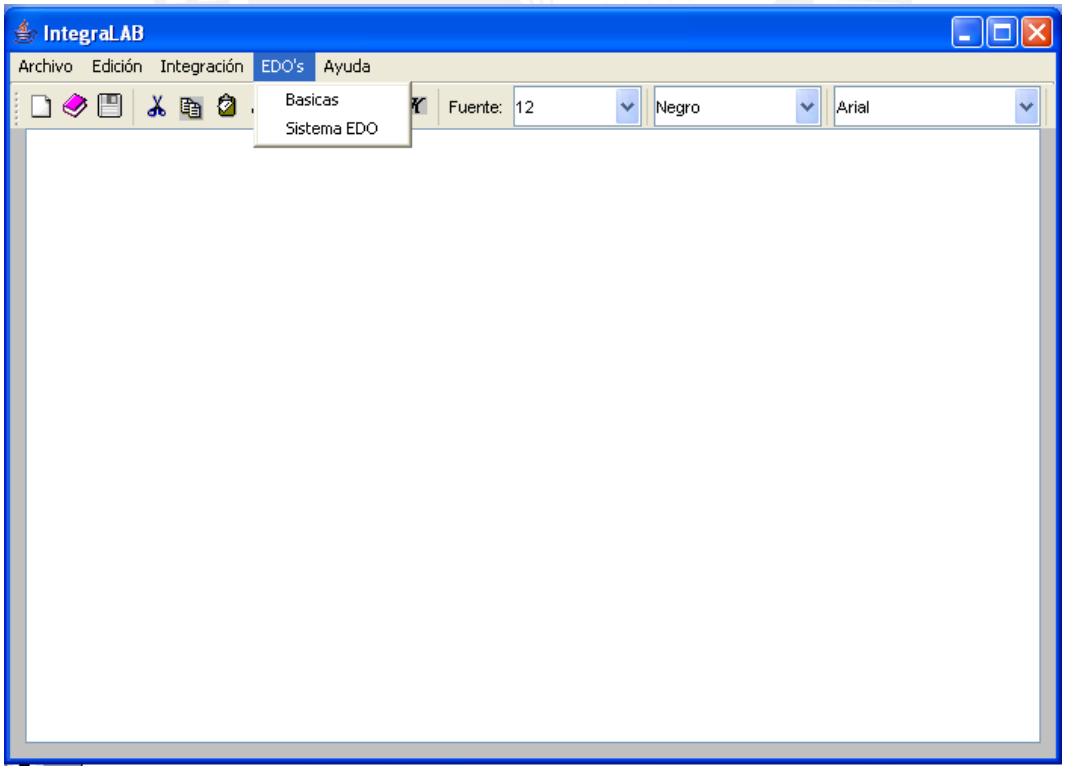


Figura 3.5: Menú EDO's

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

Al elegir la opción: Integración, Newton-Cotes se tiene la siguiente ventana, donde se presenta la solución a la raíz cuadrada de x en el intervalo 0 y 4, por el método del trapecio compuesta con n = 10.

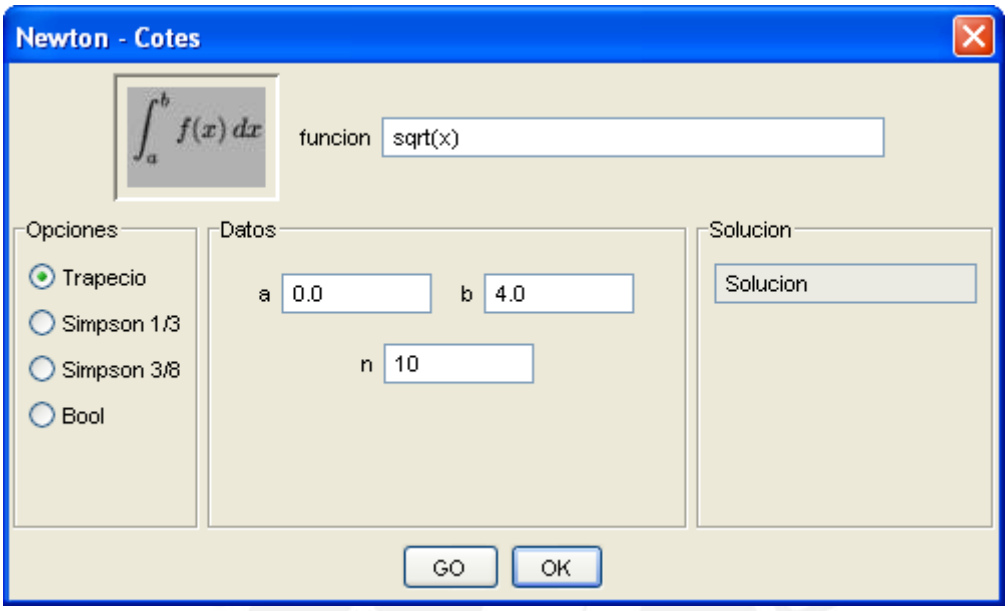


Figura 3.6: Menú Integración Newton-Cotes

La opción Integración, Gauss-Legendre produce una salida como la de la figura 3.7.

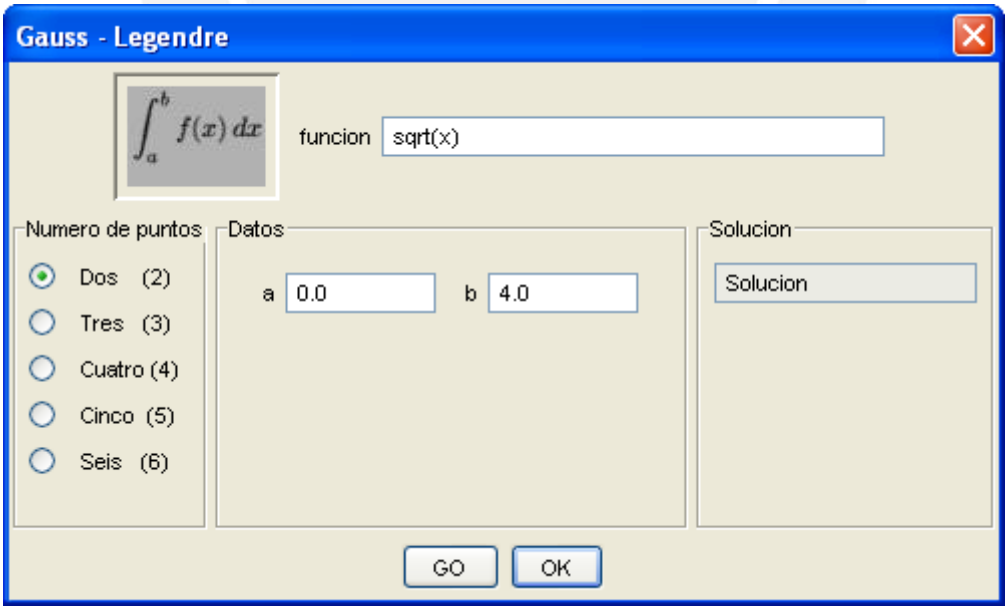


Figura 3.7: Menú Integración Gauss-Legendre

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

La opción Integración impropia, Gauss-Laguerre para evaluar la integral de X^2

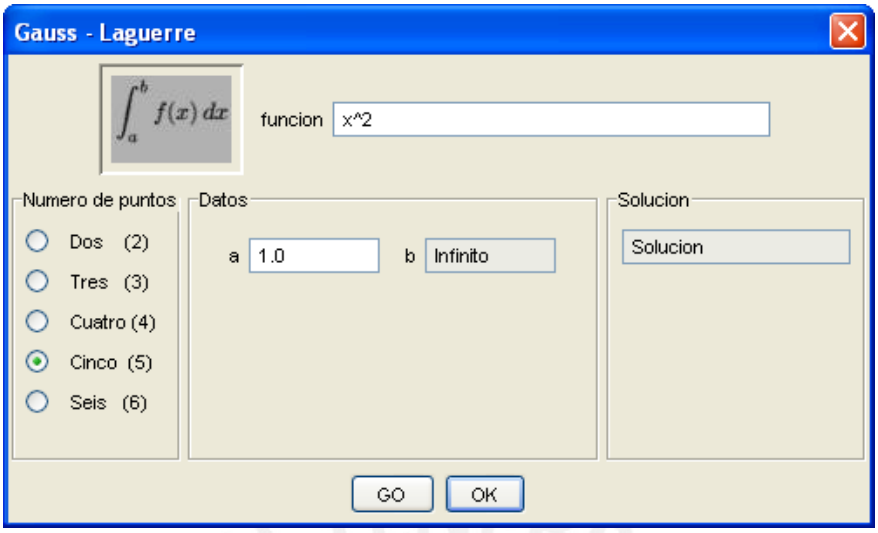


Figura 3.8: Menú Integración Gauss-Laguerre

La figura 3.9 presenta el menú para resolver las ecuaciones diferenciales ordinarias.

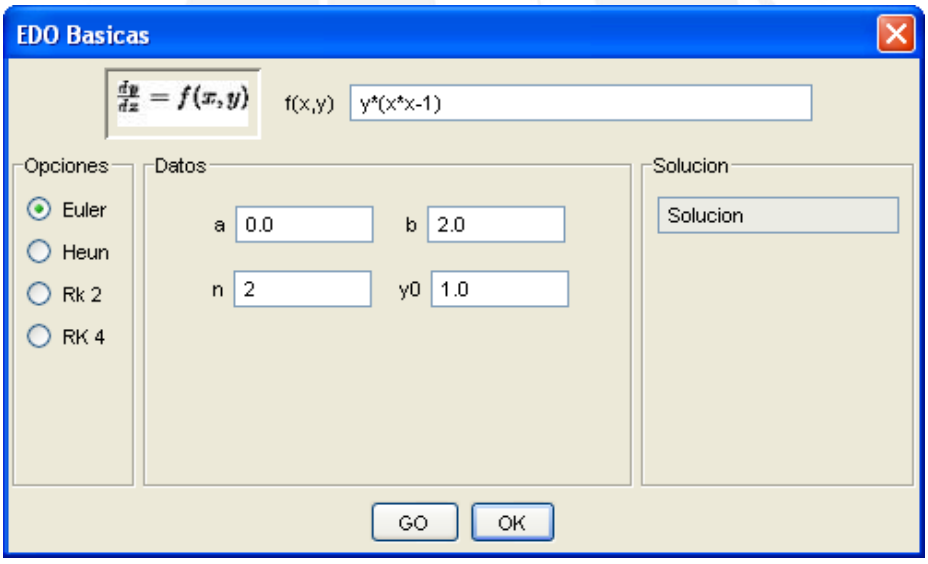


Figura 3.9: Menú para resolver las Edo's Básicas

En la figura 3.10 se presenta la ventana para resolver sistemas de ecuaciones diferenciales ordinarias.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

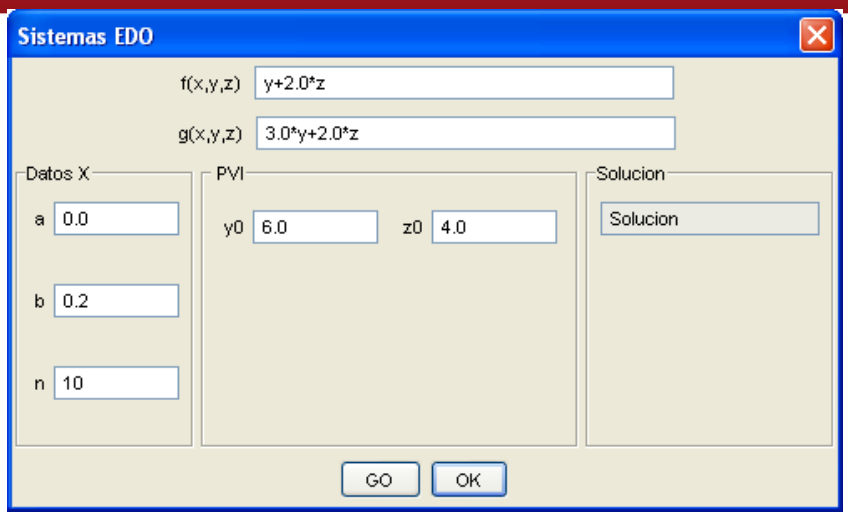


Figura 3.10: Menú Sistemas EDO.

Graficando funciones

El software IntegraLAB, posee un ambiente que permite visualizar la grafica de una función. El archivo GraphDialog.java se encarga de graficar funciones.

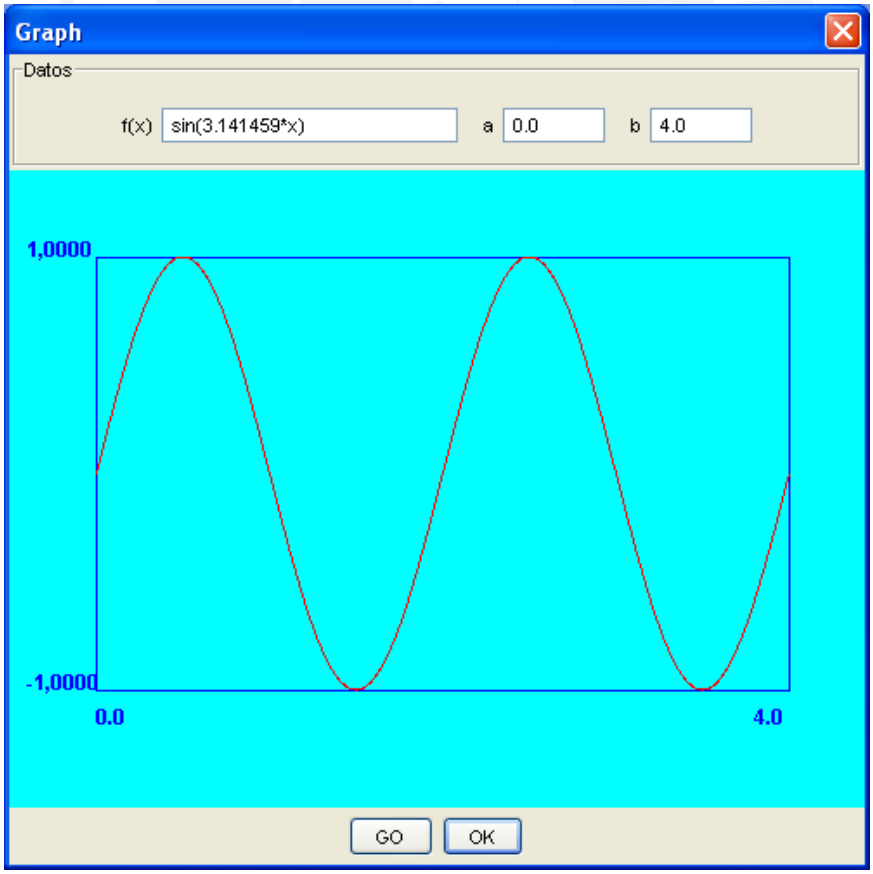


Figura 3.11: Graficando una función

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

CAPITULO 4: IMPLEMENTACIÓN DEL SOFTWARE

4.0 Explicación de IntegraLAB

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

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

Las mismas que son descritas en este capitulo.

4.1 La clase Parser

Para evaluar expresiones, se hace uso de algunas técnicas utilizadas en el diseño de compiladores. Las expresiones según su notación pueden ser:

Notación infija.- operando-operador-operando. Ejemplo: A + B
Notación prefija.- operador-operando-operando. Ejemplo: + A B
Notación postfija.- operando-operando-operador.: Ejemplo: A B +

Al hacer uso de las expresiones infijas, la expresión 1 + 5; consiste de un operador binario junto con sus operandos (argumentos). Una expresión más elaborada es: 1 + 5 * 2; la cual matemáticamente es evaluada a 11, debido a que, el operador * (de multiplicación) tiene mayor precedencia que el de suma. Si no se tiene en cuenta la precedencia, la respuesta será 12. Esto nos lleva a concluir que una simple evaluación de izquierda a derecha no basta.

En expresiones donde están presentes los operadores de resta y potencia ¿Cuál de ellos debe evaluarse primero? Para responder a la pregunta considere las expresiones

(1) 6 – 3 – 1,
(2) 2 ^ 3 ^ 2

Para la expresión (1) el resultado es 2; pues las restas se evalúan de izquierda a derecha. En el caso (2), las potencias se evalúan de derecha a izquierda; y la expresión reflejará 2^{3^2} en lugar de $(2^3)^2$.

Las expresiones en postfija, proporcionan un mecanismo directo de evaluación; existiendo algoritmos de pila para la resolución.

En la implementación de IntegraLAB, la clase `Parser`, escrita en Java; evalúa expresiones infijas que contengan operadores de adición, resta, multiplicación, división y de exponenciación, adicionalmente acepta paréntesis y llamada a funciones. Se utiliza un algoritmo de evaluación de expresiones con precedencia entre operadores.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

A continuación en la figura 4.1; se presenta una porción de la clase `Parser`.

```
public class Parser
{
    private static final int EOL     = 0;
    private static final int VALUE   = 1;
    private static final int OPAREN  = 2;
    private static final int CPAREN  = 3;
    private static final int EXP     = 4;
    private static final int MULT    = 5;
    private static final int DIV     = 6;
    private static final int PLUS    = 7;
    private static final int MINUS   = 8;
    private static final int FUNCT   = 9;
    private static String[] function =
    {
            "sqrt","sin",
            "cos","tan",
            "asin","acos",
            "atan","log",
            "floor","eXp"
    };
    private String string;
    private Stack opStack;       // Operator stack for conversion
    private Stack postfixStack;  // Stack for postfix machine
    private StringTokenizer str; // StringTokenizer stream
    // . . . continua codigo de los metodos de la clase . . .
    public double getValue(double x)
    {
        return getValue(x,0,0);
    }
    public double getValue(double x,double y)
    {
        return getValue(x,y,0);
    }
    public double getValue(double x,double y,double z)
    {
        // for each call
        opStack = new Stack( );
        postfixStack = new Stack( );
        str = new StringTokenizer(string,"+*-/^()xyz ",true);
        opStack.push( new Integer( EOL ) );

        EvalTokenizer tok = new EvalTokenizer(str,x,y,z);
        Token lastToken;
        do
        {
            lastToken = tok.getToken( );
            processToken( lastToken );
        } while( lastToken.getType( ) != EOL );
        if( postfixStack.isEmpty( ) )
        {
            System.err.println( "Missing operand!" );
            return 0;
        }
        double theResult = postFixTopAndPop( );
        if( !postfixStack.isEmpty( ) )
            System.err.println( "Warning: missing operators!" );
        return theResult;
    }
}
// . . . continua codigo de los metodos de la clase . . .
```

Figura 4.1: Declaración de la clase `Parser`.

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

En la figura 4.2 se muestra el objeto p, que es una instancia a la clase Parser.

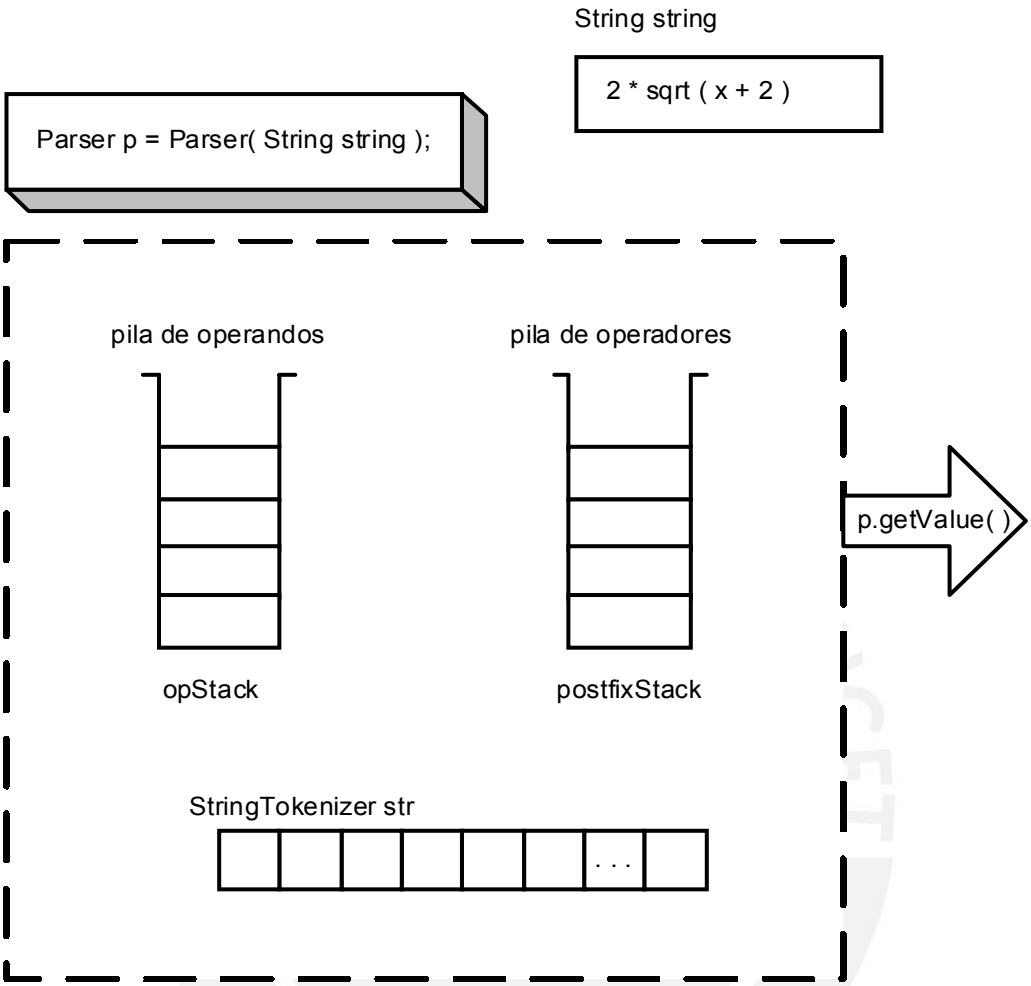


Figura 4.2: Objeto a la clase `Parser`.

4.1.1 Máquinas postfijas

Una expresión postfija esta formada por una serie de operandos y operadores. Se evalúa usando una máquina postfija, en la forma siguiente: cuando se encuentra un operando, se apila en la pila; cuando se encuentra un operador, el número de operandos (según el operador) son sacados de la pila; se realiza la operación, y el resultado se apila de nuevo en la pila. Cuando la expresión postfija completa ha sido procesada, el resultado deberá de ser el único valor en la pila.

Considere la expresión 1 + 5 * 2; que en notación postfija es,

1 5 2 * +

La evaluación es como sigue: el 1, el 5 y el 2 son apilados en ese orden, en la pila de operandos. Al leerse el operador de multiplicación; el 2 y el 5 son

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

desapilados, efectuándose el producto de 2*5, siendo el resultado 10; ahora, 10 es metido a la pila. Continuando con el algoritmo; se lee en la entrada el operador de suma, por lo que 10 y 1 son sacados de la pila, procediéndose a efectuar la suma entre estos; siendo el resultado 11; el cual es apilado nuevamente en la pila. En consecuencia el resultado de la evaluación es 11. La figura 4.3, ilustra este hecho.

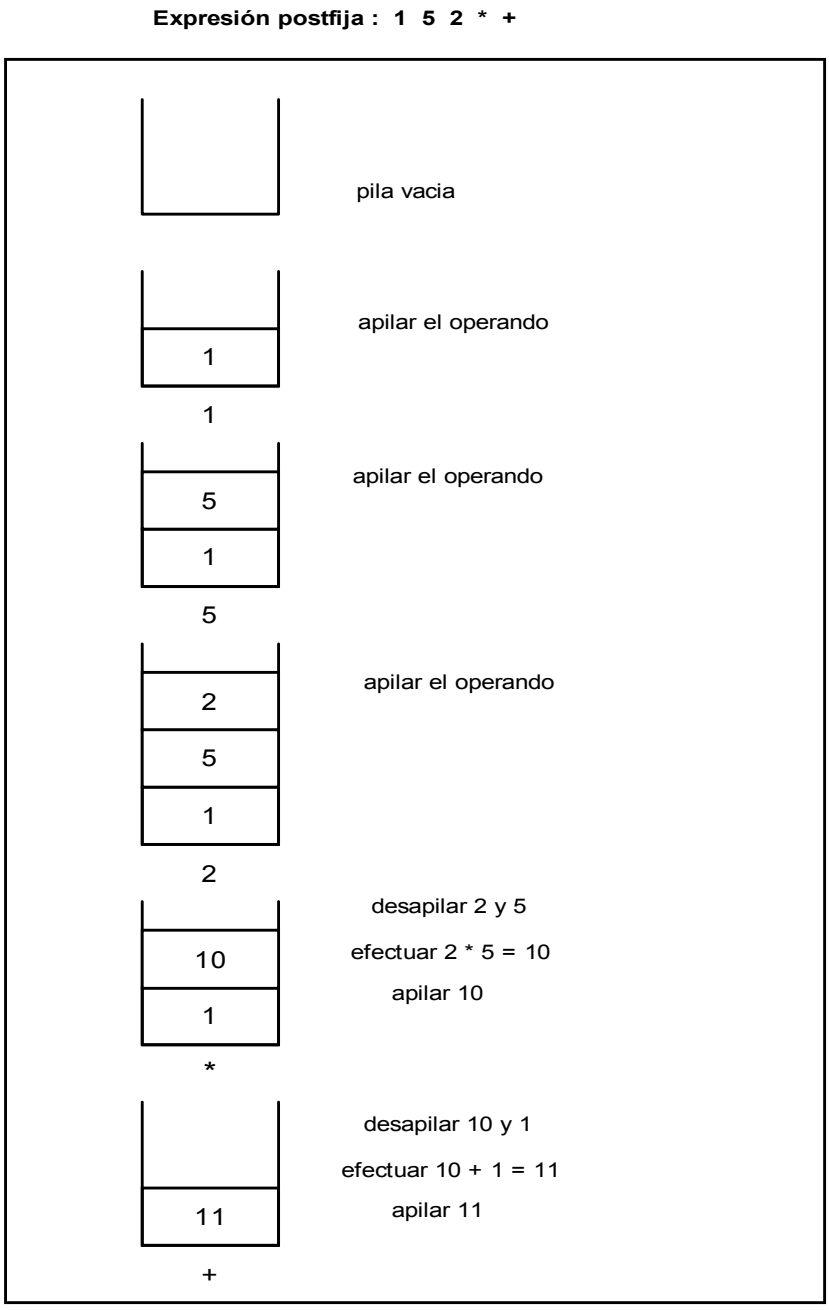


Figura 4.3: Maquina postfija para evaluar 1 5 2 * +

En la figura 4.4 se presenta el código para el algoritmo de evaluación en postfija.

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
/**
  * Process an operator by taking two items off the postfix
  * stack, applying the operator, and pushing the result.
  * Print error if missing closing parenthesis or division by 0.     */
 private void binaryOp( int topOp )
 {
     if( topOp == OPAREN )
     {
         System.err.println( "Unbalanced parentheses" );
         opStack.pop( );
         return;
     }
     if(topOp >= FUNCT )
     {
         double d=getTop();
         postfixStack.push(new Double(functionEval(topOp, d)));
         opStack.pop( );
         return;
     }
     double rhs = getTop( );
     double lhs = getTop( );
     if( topOp == EXP )
         postfixStack.push( new Double(pow(lhs,rhs) ) );
     else if( topOp == PLUS )
         postfixStack.push( new Double(lhs + rhs) );
     else if( topOp == MINUS )
         postfixStack.push( new Double(lhs - rhs) );
     else if( topOp == MULT )
         postfixStack.push( new Double(lhs * rhs) );
     else if( topOp == DIV )
         if( rhs != 0 )
            postfixStack.push( new Double(lhs / rhs) );
         else
         {
            System.err.println( "Division by zero" );
            postfixStack.push( new Double( lhs ) );
         }
     opStack.pop();
 }
 private double functionEval(int topOp, double d)
 {  double y = 0;
          switch (topOp) {
          case 9:
                y = Math.sqrt(d); break;
          case 10:
                y = Math.sin(d); break;
          case 11:
                y = Math.cos(d); break;
          case 12:
                y = Math.tan(d); break;
          case 13:
                y = Math.asin(d); break;
          case 14:
                y = Math.acos(d); break;
          case 15:
                y = Math.atan(d); break;
          case 16:
                y = Math.log(d); break;
          case 17:
                y = Math.floor(d); break;
          case 18:
                y = Math.exp(d);
    }     return y;
 }
```

Figura 4.4 Código para el algoritmo de evaluación en postfija.

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

4.1.2 Conversión de notación infija a postfija [AHO y WEISS]

Cuando se encuentra un operando, podemos pasarlo inmediatamente a la salida. Cuando se encuentra un operador, no es posible pasarlo a la salida; pues se debe esperar a encontrar su segundo operando, por lo que se hace necesario una estructura de datos temporal. Una pila es la estructura de datos adecuada para almacenar operadores pendientes.

La expresión infija 2 ^ 4 – 1, tiene su expresión en postfija: 2 4 ^ 1 -. De acuerdo al procesamiento de un operador de entrada; se debe sacar de la pila aquellos operadores que deben ser procesados atendiendo a las reglas de precedencia y asociatividad. Este, es el principio básico en el algoritmo sintáctico de expresiones por precedencia de operadores. Al respecto ver la figura 4.5

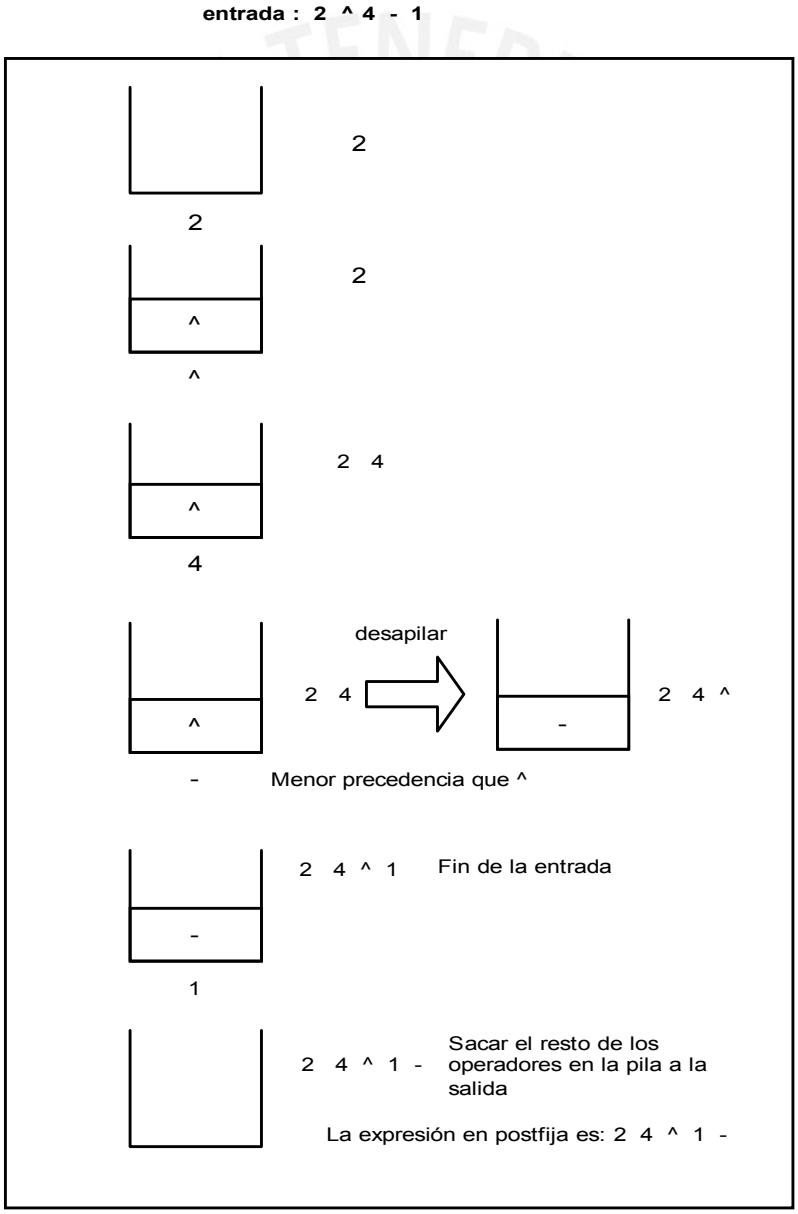


Figura 4.5: Conversión de infija a postfija.

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

4.1.3 Análisis sintáctico por precedencia de operadores [AHO]

Sea la siguiente gramática para expresiones

E → E A E
| (E)
| - E
| F(E)
| id
A → + | - | * | / |^

donde, F es el operador de llamada a función.

Un analizador sintáctico para gramáticas, tiene la propiedad que ningún lado derecho de la producción es ∈, ni tiene dos no terminales adyacentes. A esto se denomina gramática de operadores.

Se observa que la producción E → E A E tiene dos no terminales consecutivos.

La conversión de la gramática anterior a la gramática de operadores es la siguiente:

E → E + E
E - E
E * E
E / E
E ^ E
(E)
- E
F (E)
id

El algoritmo de análisis sintáctico por precedencia de operadores, tiene como entrada una cadena de caracteres *s* y una tabla de relaciones de precedencia, y como salida, una estructura de árbol de análisis sintáctico. Inicialmente la pila contiene $ y el buffer de entrada, la cadena *s*.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
        /**
         * Construct an evaluator object.
         * @param s the string containing the expression.      */
        public Parser( String s )
        {
           opStack = new Stack( );
           postfixStack = new Stack( );
           string = unary2Binary(s);
           str = new StringTokenizer(string,"+*-/^()xyz ",true);

           opStack.push( new Integer( EOL ) );
        }
```

Figura 4.6: Constructor para la clase `Parser`.

Para realizar el análisis sintáctico, se ejecuta algoritmo en pseudo código presentado en la figura 4.7.

```
Algoritmo análisis sintáctico
    Apuntar al primer símbolo de s
    while true
          if $ se encuentra en la cima de la pila y la entrada apunta a
               fin de cadena then
                return
          else
                a = símbolo terminal en la cima de la pila
                b = símbolo que apunta a la entrada
                          •
          if a <· b or a = b then
                apilar b en la pila
                avanzar al siguiente símbolo en la entrada
          else
                if a ·> b
                      repeat
                         extraer el elemento de la cima de la pila
                      until el terminal de la cima de la pila <·
                else
                      error ( )
end Algoritmo análisis sintáctico
```

Figura 4.7: Algoritmo del parser por precedencia de operadores.

En la figura 4.8 se presenta el código en java que implementa el algoritmo para evaluar expresiones.

```
   // The only publicy visible routine
   /**
    * Public routine that performs the evaluation.
    * Examine the  postfix machine to see if a single result is
    * left and if so, return it; otherwise print error.
    * @return the result.
    */
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
public double getValue(double x)
{
    return getValue(x,0,0);
}

public double getValue(double x,double y)
{
    return getValue(x,y,0);
}

public double getValue(double x,double y,double z)
{
    // for each call
    opStack = new Stack( );
    postfixStack = new Stack( );
    str = new StringTokenizer(string,"+*-/^()xyz ",true);
    opStack.push( new Integer( EOL ) );

    EvalTokenizer tok = new EvalTokenizer(str,x,y,z);
    Token lastToken;
    do
    {
        lastToken = tok.getToken( );
        processToken( lastToken );
    } while( lastToken.getType( ) != EOL );

    if( postfixStack.isEmpty( ) )
    {
        System.err.println( "Missing operand!" );
        return 0;
    }

    double theResult = postFixTopAndPop( );
    if( !postfixStack.isEmpty( ) )
        System.err.println( "Warning: missing operators!" );

    return theResult;
}

/**
* Internal method that unary to binary.
*/
   private String unary2Binary(String s)
   {
    int i;
    s = s.trim();
    if(s.charAt(0) == '-')
         s = "0.0"+s;
    while((i = s.indexOf("(-"))>=0)
        s = s.substring(0,i+1)+"0.0"+
        s.substring(i+1);
    return s;
}

/**
 * Internal method that hides type-casting.
 */
private double postFixTopAndPop( )
{
    return ((Double)(postfixStack.pop())).doubleValue( );
}

/**
 * Another internal method that hides type-casting.
 */
private int opStackTop( )
{
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
        return ( (Integer) ( opStack.peek( ) ) ).intValue( );
    }

    /**
     * After a token is read, use operator precedence parsing
     * algorithm to process it; missing opening parentheses
     * are detected here.
     */
    private void processToken( Token lastToken )
    {
        int topOp;
        int lastType = lastToken.getType();

        switch(lastType)
        {
          case VALUE:
            postfixStack.push( new Double( lastToken.getValue( ) ) );
            return;

          case CPAREN:
            while((topOp = opStackTop( ) ) != OPAREN && topOp != EOL)
                binaryOp( topOp );
            if( topOp == OPAREN )
                opStack.pop( );  // Get rid of opening parentheseis
            else
                System.err.println( "Missing open parenthesis" );
            break;

          default:    // General operator case
            int last=(lastType>=FUNCT?FUNCT:lastType);
            while(precTable[last].inputSymbol<=
precTable[opStackTop()>=FUNCT?FUNCT:opStackTop()].topOfStack)
                binaryOp( opStackTop() );
            if( lastType != EOL )
                opStack.push( new Integer( lastType ));
            break;
        }
    }

    /*
     * topAndPop the postfix machine stack; return the result.
     * If the stack is empty, print an error message.
     */
    private double getTop( )
    {
        if ( postfixStack.isEmpty( ) )
        {
            System.err.println( "Missing operand" );
            return 0;
        }
        return postFixTopAndPop( );
    }

    /**
     * Internal routine to compute x^n.
     */
    private static double pow( double x, double n )
    {
        if( x == 0 )
        {
            if( n == 0 )
                System.err.println( "0^0 is undefined" );
            return 0;
        }
        if( n < 0 )
        {
            System.err.println( "Negative exponent" );
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
            return 0;
        }

        if( n == 0 )
            return 1;
        if( n % 2 == 0 )
            return pow( x * x, n / 2  );
        else
            return Math.pow( x, n );
    }
```

Figura 4.8: Código en Java para evaluar expresiones.

4.1.4 Tabla Driven

Las gramáticas de operadores describen un lenguaje de propósitos especiales, o un subjuego de un lenguaje típico de programación, como por ejemplo snoboll.

Las relaciones de precedencia, son especificadas entre un conjunto de operadores, para esa gramática.

En la precedencia de operadores, el `string` es capturado de izquierda a derecha; existiendo dos estados esenciales:

Estado 1: esperando un operador
Estado 2: esperando un operando

El análisis sintáctico, propone dos pilas: una para operandos y otra para operadores; luego compara la precedencia entre operadores consecutivos. Cuando un operador de más alta precedencia, se encuentra en la cima (el top o tope de la pila), entonces el handle, llamado también mango; como lo llamaremos de aquí en adelante, consiste de dos operandos, combinados con este operador.

Se formaliza las relaciones de precedencia, por la introducción de 3 relaciones disjuntas:

$$<\cdot \qquad (\doteq) \qquad \cdot>$$

donde $a <\cdot b$ significa, que a tiene precedencia menor que b, $a \doteq b$, significa que a tiene la misma precedencia que b, y $a \cdot> b$ significa, que a tiene mayor precedencia que b. Por ejemplo,

$$+ <\cdot * , (\doteq), \text{ y } * \cdot> +$$

En la entrada, el string:

(id + id)

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

las relaciones de precedencia entre los elementos del string:

$$\$ \lessdot (\lessdot \mathbf{id} \gtrdot + \lessdot \mathbf{id} \gtrdot) \gtrdot \$$$

y haciendo uso del algoritmo parsing de precedencia de operadores, se tiene:

1. Explorar el primer $\gtrdot$ y regresar al más cercano $\lessdot$

$$\$ \lessdot (\lessdot \mathbf{id} \gtrdot + \lessdot \mathbf{id} \gtrdot) \gtrdot \$$$

2. Identificar el primer **id** como el mango y utilizando la producción de **E** → **id** la gramática se reduce a **E**:

$$\begin{array}{c} E \\ | \\ \$ (\mathbf{id} + \mathbf{id}) \$ \end{array}$$

3. Ignorar el no terminal para reinsertar, las relaciones de precedencia

$$\$ \lessdot (\lessdot \mathbf{id} + \lessdot \mathbf{id} \gtrdot) \gtrdot \$$$

con lo cual el siguiente identificador **id** es ahora el mango:

$$\begin{array}{c} E \quad E \\ | \quad | \\ \$ (\mathbf{id} + \mathbf{id}) \$ \end{array}$$

luego, la forma de la sentencia es:

$$\$ (E + E) \$$$

Repitiendo el algoritmo

$$\$ \lessdot (\lessdot + \gtrdot) \gtrdot \$$$

el mango es ahora E + E; es decir

$$\$ (E) \$$$

El proceso continua y como E → (E) se tiene,

$$\$ \lessdot (\doteq) \gtrdot \$$$

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

$ E $

finalizando; el árbol del parser es el de la figura 4.9:

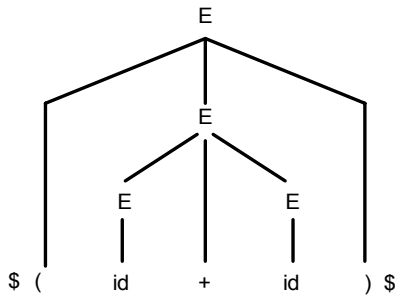


Figura 4.9: Árbol parser para la gramática G.

La tabla de precedencias para la gramática G se presenta en la tabla 4.1.

Gramática G
E → E + E
E → E * E
E → (E)
E → id

	(	id	*	+	)	$
)			·>	·>	·>	·>
Id			·>	·>	·>	·>
*	<·	<·	·>	·>	·>	·>
+	<·	<·	<·	·>	·>	·>
(	<·	<·	<·	<·	=	
$	<·	<·	<·	<·		=

Tabla 4.1: Tabla de precedencias para la gramática G.

Haciendo uso de la tabla 4.1 y del algoritmo de la figura 4.10 se procede a construir el árbol de la figura 4.9

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Algoritmo parsing de precedencia

```
Algoritmo parsing de precedencia
       Insertar relaciones de precedencia
       while la entrada ≠ $ E $
              capturar el ·> más a la izquierda
              capturar el primer <· cercano a la izquierda
              reducir el mango
              reinsertar las relaciones de precedencia ignorando no terminales
       end while
end Algoritmo parsing de precedencia
```

Figura 4.10: Algoritmo parsing de precedencia

4.1.5 Construcción de la tabla de precedencia

Sean las operaciones Leading y trailing:

1. Leading (N), donde N es un no terminal, y corresponde al conjunto de todos los terminales que aparecen primeros en una forma sentencial derivable de N.
2. Trailing (N), donde N es un no terminal, y corresponde al conjunto de todos los terminales que aparecen últimos, en una forma sentencial derivable de N.

Usando la gramática G, de la tabla 4.1, se tiene:

Leading (E) = { + , * , (, id }
Trailing (E) = { + , * ,) , id }

Una producción de la forma

$$N \rightarrow a\ A\ t\ B\ \beta$$

donde t, es un terminal y A y B son no terminales; cumple con la relación

$$\text{Trailing}(A) \cdot\!> t <\!\cdot \text{Leading}(B)$$

Al respecto ver la figura 4.11.

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

$N \rightarrow a\ A\ t\ B\ \beta$

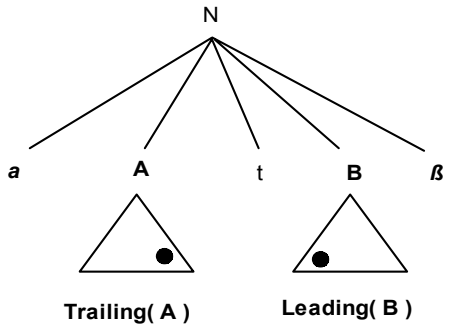


Figura 4.11: Trailing (A) ·> t <· Leading (B)

Se plantean las siguientes reglas:

Regla 1: $t <\cdot$ Leading (B)

Regla 2: Training (A) $\cdot>$ t

Regla 3: Si t_1 y t_2, ambos aparecen en el lado de la mano derecha de la misma producción, entonces

$$t_1 \doteq t_2$$

Se plantea el siguiente algoritmo para construir la tabla de precedencia

```
Algoritmo construcción tabla de precedencia

      for cada producción de la forma N → α A t B β
            computar Leading ( B )
            computar Trailing ( A )
            aplicar las reglas 1, 2 y 3
            $ <· otros terminales
      end for

End Algoritmo construcción tabla de precedencia
```

Figura 4.12: Algoritmo de construcción de la tabla de precedencia.

Aplicando el algoritmo de construcción de la tabla de precedencia a la producción $E \rightarrow E + E$, se tiene la figura 4.13.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

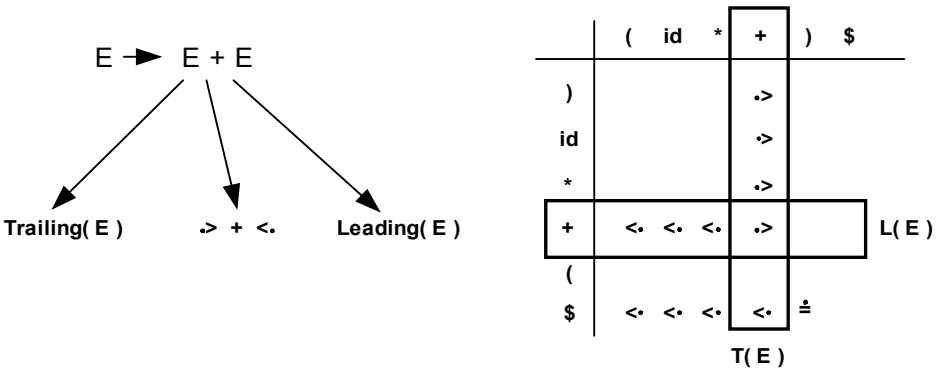


Figura 4.13: Aplicación del algoritmo de construcción para la producción:
$E \rightarrow E + E$

4.1.6 Funciones de precedencia

Los compiladores que utilizan parser por precedencia de operadores, utilizan funciones de precedencia antes que tablas de precedencia. Las funciones de precedencia ***f*** y ***g*** transforman símbolos terminales en enteros. Las funciones ***f*** y ***g***, actúan ante los símbolos a y b como:

1. f (a) < g (b), si a <· b,
2. f (a) = g (b), si a $\doteq$ b, y
3. f (a) > g (b), si a ·>

El algoritmo para encontrar las funciones de precedencia, usa como insumo la tabla de precedencia de operadores; y se denomina algoritmo de construcción de funciones de precedencia.

```
Algoritmo de construcción de funciones de precedencia

1.  Crear los símbolos f_a y g_a, para cada a que sea un terminal o $.
2.  Crear un grafo dirigido, cuyos nodos sean los símbolos f_a y g_a.  Si
    a <· b, colóquese una arista desde g_b. a f_a.   Si a ·> b; colóquese
    una arista de f_a a g_b
3.  Si no hay ciclos, f( a ) es la longitud del camino mas largo que
    comienza en f_a

End Algoritmo de construcción de funciones de precedencia
```

Figura 4.14: Algoritmo de construcción de funciones de precedencia.

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

En la figura 4.15 se presenta el grafo de funciones de precedencia aplicando el algoritmo a la gramática G siguiente

Gramática G:
E → E + E
E → E - E
E → E * E
E → E / E
E → E ^ E
E → (E)
E → id

Como no hay ciclos, existen las funciones de precedencia. Debido a que $f_{\$}$ y $g_{\$}$ no poseen aristas de salida, $f(\$) = g(\$) = 0$.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

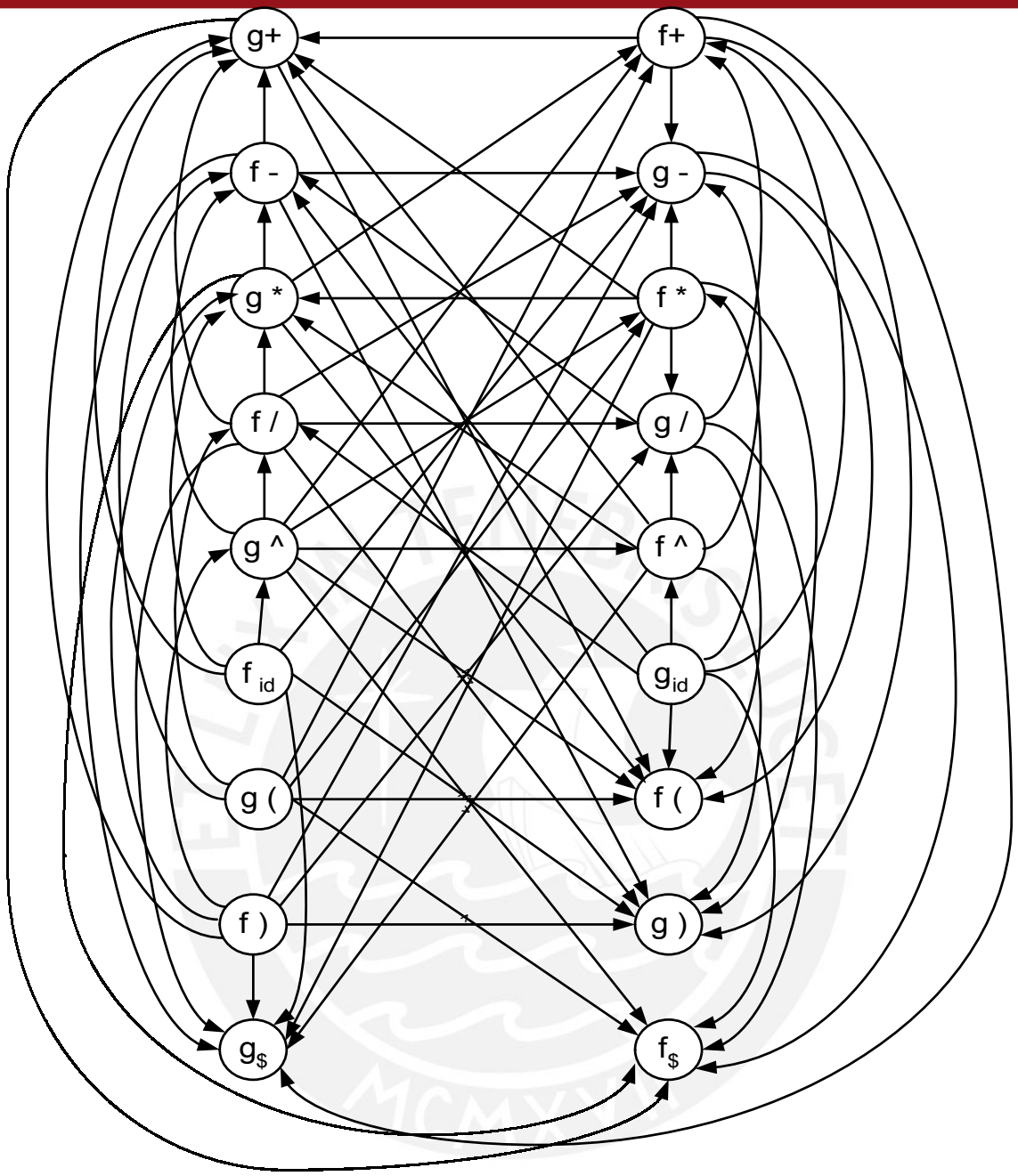


Figura 4.15: Grafo de funciones de precedencia.

La tabla siguiente, es el resultado a aplicar el paso 3 del algoritmo de construcción de funciones de precedencia.

	+	-	*	/	^	(	)	Id	$
f	2	2	4	4	4	0	6	6	0
g	1	1	3	3	5	5	0	5	0

Tabla 4.2: Funciones de precedencia

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

Algunos cálculos de funciones de precedencia, se muestran a continuación.

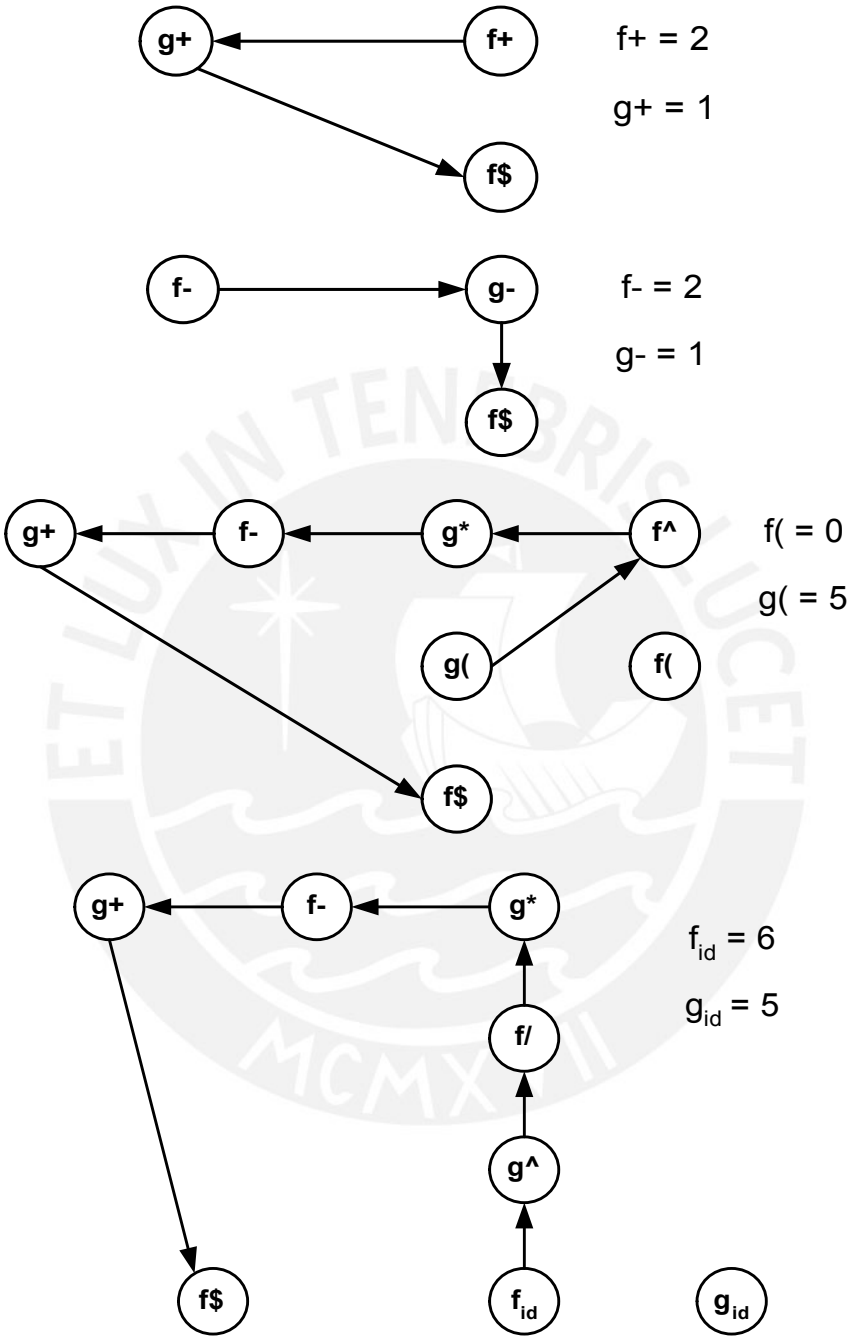


Figura 4.16: Algunos cálculos de funciones de precedencia.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

El código en Java que realiza los cálculos de las funciones de precedencia se presenta en la figura 4.17.

```
private static class Precedence
{
    public int inputSymbol;
    public int topOfStack;

    public Precedence( int inSymbol, int topSymbol )
    {
        inputSymbol = inSymbol;
        topOfStack  = topSymbol;
    }
}

// PrecTable matches order of Token enumeration
private static Precedence [ ] precTable = new Precedence[ ]
{
    new Precedence(   0, -1 ),   // EOL
    new Precedence(   0,  0 ),   // VALUE
    new Precedence( 100,  0 ),   // OPAREN
    new Precedence(   0, 99 ),   // CPAREN
    new Precedence(   6,  5 ),   // EXP
    new Precedence(   3,  4 ),   // MULT
    new Precedence(   3,  4 ),   // DIV
    new Precedence(   1,  2 ),   // PLUS
    new Precedence(   1,  2 ),   // MINUS
    new Precedence(   7,  6 )    // FUNCT
};
```

Figura 4.17: La clase `Precedence`.

4.1.7 Análisis lexicográfico

Existen tres métodos generales de implantación de un analizador léxico. [AHO]

1. Utilizar un generador de analizadores léxicos, como el compilador LEX, utilizado para producir el analizador léxico a partir de una configuración basada en expresiones regulares. En este caso, el generador proporciona rutinas para la entrada y manejarla con buffers.
2. Escribir el analizador léxico en un lenguaje convencional de programación de sistemas, utilizando las posibilidades de entrada y salida de este lenguaje para leer la entrada.
3. Escribir el analizador léxico en lenguaje ensamblador y manejar explícitamente la lectura de la entrada.

Para el desarrollo de IntegraLAB, se considera el segundo método y el lenguaje a utilizar es el JAVA de Sun.

Muchos compiladores son del tipo parser-driven, lo cual significa que el analizador sintáctico llama al analizador lexical.

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Los token, son la unidad básica lexical; así como las palabras y la puntuación son las unidades básicas en una sentencia; sea en el lenguaje español o en el inglés.

La figura 4.18, presenta la clase `Token` para IntegraLAB.

```
private static class Token
{
    public Token( )
    {
        this( EOL );
    }

    public Token( int t )
    {
        this( t, 0 );
    }

    public Token( int t, double v )
    {
        type = t;
        value = v;
    }

    public int getType( )
    {
        return type;
    }

    public double getValue( )
    {
        return value;
    }

    private int type = EOL;
    private double value = 0;
}
```

Figura 4.18: La clase `Token`.

Los token son descritos en dos partes; un tipo y un valor:

Token = (type, value)

Por ejemplo:

Token (MINUS);
Token (VALUE, equiz);
Token (VALUE, zeta);

Esto es presentado en la clase `EvalTokenizer` mostrada en la figura 4.19

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
    private static class EvalTokenizer
    {
        public EvalTokenizer( StringTokenizer is,double x,
                              double y,double z)
        {
            str = is;
            equis=x;
            ye=y;
            zeta=z;
        }

        /**
         * Find the next token, skipping blanks, and return it.
         * For VALUE token, place the processed value in currentValue.
         * Print error message if input is unrecognized.
         */
        public Token getToken( )
        {
            double theValue;

            if(!str.hasMoreTokens( ))
                return new Token( );

            String s = str.nextToken();

            if( s.equals( " " ) ) return getToken( );
            if( s.equals( "^" ) ) return new Token(EXP);
            if( s.equals( "/" ) ) return new Token(DIV);
            if( s.equals( "*" ) ) return new Token(MULT);
            if( s.equals( "(" ) ) return new Token(OPAREN);
            if( s.equals( ")" ) ) return new Token(CPAREN);
            if( s.equals( "+" ) ) return new Token(PLUS);
            if( s.equals( "-" ) ) return new Token(MINUS);
            if( s.equals( "x" ) ) return new Token(VALUE,equis);
            if( s.equals( "y" ) ) return new Token(VALUE,ye);
            if( s.equals( "z" ) ) return new Token(VALUE,zeta);

                    if(Character.isLetter(s.charAt(0)))
            {
             int i=searchFunction(s);
             if(i>=0)
                    return new Token(FUNCT+i);
             else
             {
                    System.err.println( "Parse error" );
                    return new Token();
             }
            }

            try
            {
                    theValue = Double.valueOf(s).doubleValue();
            }
            catch( NumberFormatException e )
            {
                System.err.println( "Parse error" );
                return new Token( );
            }

            return new Token( VALUE, theValue );
        }

      public int searchFunction(String s)
      {
             for(int i=0;i<function.length;i++)
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
                if(s.equals(function[i]))
                    return i;
            return -1;
        }

        private StringTokenizer str;
        private double equis;
        private double ye;
        private double zeta;
    }
```

Figura 4.19: La clase `EvalTokenizer`.

La figura 4.20 presenta el diagrama UML para la clase Parser.

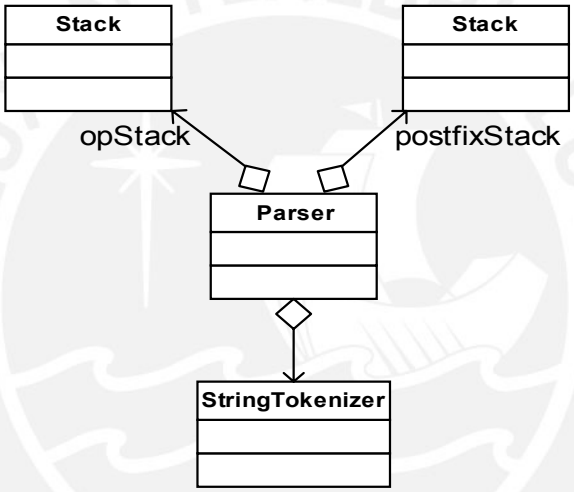


Figura 4.20: Diagrama UML de clases para `Parser`.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

4.2 La clase GraphDialog

Esta clase permite a IntegraLAB graficar curvas mediante la presentación del cuadro de dialogo y el trazo de la función en un objeto a la clase `GraphPanel`, denominado panel. La figura 4.21 presenta el diagrama de objetos de la clase `GraphDialog`.

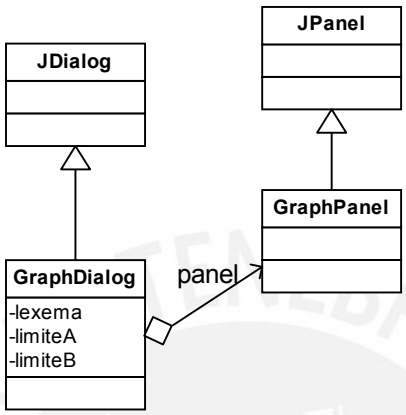


Figura 4.21: Diagrama UML para la clase `Graphdialog`.

La clase `GraphDialog` esta compuesta por los siguientes campos: `lexema`, `limiteA`, `limiteB` y `panel`. Al respecto ver la sección de código mostrada en la figura 4.22.

```
  //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 4.22: Sección de código de la clase `GraphDialog`.

En IntegraLAB, por defecto al ejecutar la clase GraphDialog mediante la secuencia [Ctrl] + [G] se presenta la ventana de la figura 4.23 donde se traza la función sin(3.141459*x), en un intervalo [0.0, 4.0].

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

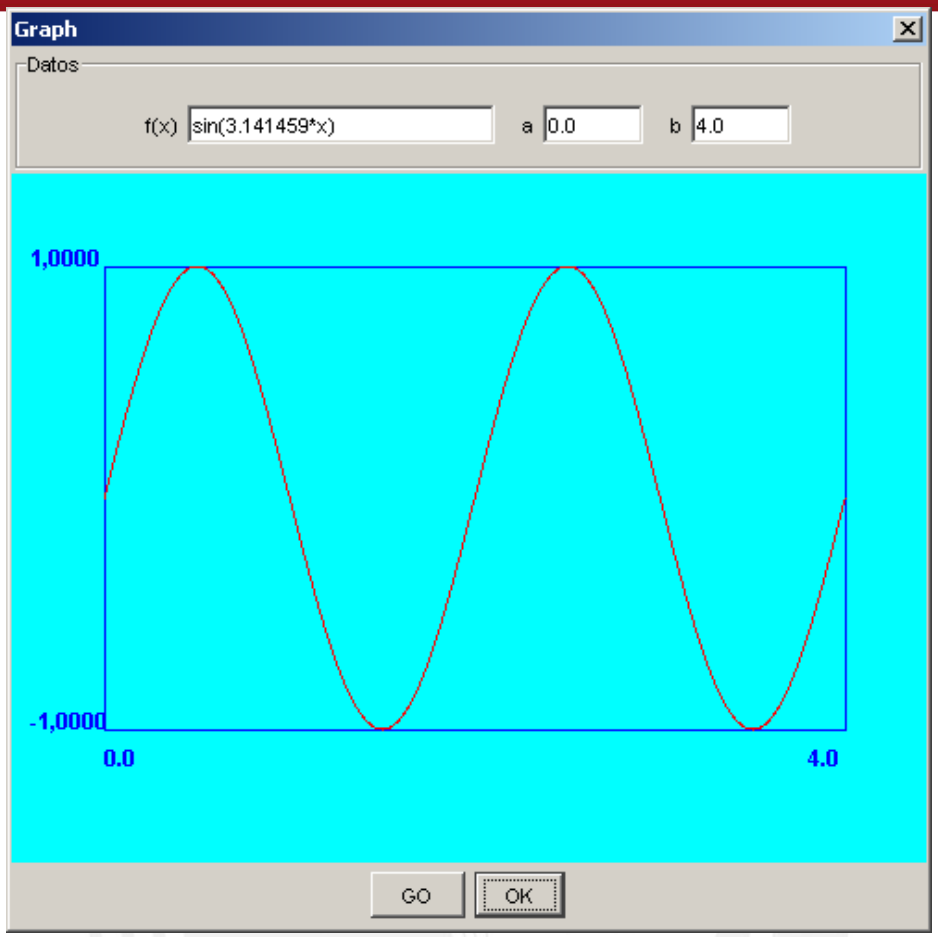


Figura 4.23: Ejecución de `GraphDialog`.

4.2.1 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. Instanciar al objeto de gráfico o panel.
4. Declarar los diversos listener.

Los puntos del 1 al 3, son presentados en el siguiente código.

```
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);
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
getContentPane().add(panel, BorderLayout.CENTER);
panel.setBackground(Color.cyan);

JLabel lbl = new JLabel(/*new ImageIcon(
                          "images/imagInteg.jpg")*/);
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)"));
```

Figura 4.24: Tareas 1, 2 y 3 del constructor `GraphDialog`.

Para instanciar y especificar el método para el evento (listener) a `funcionText`; a parte de leer el nuevo string para lexema, debe de actualizarse el campo al objeto panel. Adicionalmente debe pegar esta componente `funcionText` al objeto p3 (de la clase `JPanel`). Ver figura 4.25.

```
funcionText=new JTextField(lexema,20);
funcionText.addActionListener(
      new ActionListener() {
            public void actionPerformed(ActionEvent e)
            {
                  lexema=e.getActionCommand();
                  panel.setLexical(lexema);
            }
      }
);
p1.add(funcionText);
p3.add(p1);
getContentPane().add(p3, BorderLayout.NORTH);

JPanel p31=new JPanel();
p31.add(new JLabel("a"));
JTextField aText=new JTextField("0.0",6);
aText.addActionListener(
      new ActionListener() {
            public void actionPerformed(ActionEvent e)
            {
                  limiteA=Double.valueOf(
                        e.getActionCommand()).
                        doubleValue();
                  panel.setA(limiteA);
            }
      }
);
p31.add(aText);
p3.add(p31);
JPanel p32=new JPanel();
p32.add(new JLabel("b"));
JTextField bText=new JTextField("4.0",6);
bText.addActionListener(
      new ActionListener() {
            public void actionPerformed(ActionEvent e)
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
                        {
                                limiteB=Double.valueOf(
                                        e.getActionCommand()).
                                        doubleValue();
                                panel.setB(limiteB);

                        }
                }
        );
        p32.add(bText);
        p3.add(p32);
```

Figura 4.25: Instanciar y especificar el método para el evento a `funcionText`.

De manera similar ocurre para los objetos *`aText`*, para recuperar el valor limite inicial del gráfico y *`bText`*, para recuperar el valor limite final del gráfico, tal como se observa en la figura 4.25.

La instancia en el botón *`goOK`*, permite actualizar el gráfico; en cambio la instancia *`btOK`*, permite cerrar la caja de diálogo. Esto se muestra en la figura 4.26.

```
        //getContentPane().add(p3, BorderLayout.NORTH);
        goOK = new JButton("GO");
        ActionListener rst = new ActionListener(){
           public void actionPerformed(ActionEvent e){
              panel.graph();
              //getContentPane().add(panel, BorderLayout.CENTER);
           }
        };
        goOK.addActionListener(rst);
        p = new JPanel();
        p.add(goOK);
        btOK = new JButton("OK");
        ActionListener lst = new ActionListener() {
           public void actionPerformed(ActionEvent e) {
                dispose();
           }
        };
        btOK.addActionListener(lst);
        p.add(btOK);
        getRootPane().setDefaultButton(btOK);
        getRootPane().registerKeyboardAction(lst,
           KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0),
           JComponent.WHEN_IN_FOCUSED_WINDOW);
        getContentPane().add(p, BorderLayout.SOUTH);

        WindowListener wl = new WindowAdapter() {
           public void windowOpened(WindowEvent e) {
                btOK.requestFocus();
           }
        };
        addWindowListener(wl);
        pack();
        setResizable(false);
        setLocationRelativeTo(owner);
    }
} // fin de class GraphDialog
```

Figura 4.26: Instancia para `goOK` y `btOK`.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

4.2.2 La clase GraphPanel

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

El constructor de `GraphPanel` permite especificar los elementos del gráfico, tales como: cadena a evaluar y limites del gráfico; e invocar al método `graph( )` para construir el gráfico. La figura 4.27 presenta este constructor.

```
// 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 4.27: La clase `GraphPanel` y su constructor.

El método `graph( )`, permite construir el gráfico en el objeto *g* a `Graphics`; su algoritmo se presenta en el siguiente pseudo código.

```
	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 4.28 Pseudo código para `graph( )`.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

El código que implementa el algoritmo `graph` se muestra en la figura 4.29.

```
// graficando
public void graph()
{
	points=new ArrayList();
	lines=new ArrayList();

	computePoints();
	computeScala();
	last=new Point2D.Double(
		X0+scalaX*(((Point2D)points.get(0)).getX()-a),
		Y0+ALTO-scalaY*(((Point2D)points.get(0)).getY()-min));

	for(int i=1;i<points.size();i++)
		add(i);
}
```

Figura 4.29: Código para el método `graph`().

El proceso de pintar los puntos de las líneas de la curva, en una forma animada o iterada, lo efectúa el método `add`( ), cuyo código viene a continuación.

```
public void add(int i)
{
	Point2D end = new Point2D.Double(
		X0+scalaX*(((Point2D)points.get(i)).getX()-a),
		Y0+ALTO-scalaY*(((Point2D)points.get(i)).getY()-min));
	Line2D line=new Line2D.Double(last,end);
	lines.add(line);
	repaint();
	last = end;
}
```

Figura 4.30: Código para el método `add`( ).

La invocación a `repaint`( ), implica en Java Swing llamar al método `paintComponent`( ). En esta acción de repintado se grafican los ejes y se grafican las i líneas. La figura 4.31, presenta el código para el pintado de los ejes.

```
public void paintComponent(Graphics g)
{
	super.paintComponent(g);
	Graphics2D g2 = (Graphics2D) g; //convertir g a Graphics2D
	axisText(g2);
	g2.setColor(Color.red); //establecer color a ejes

	for(int i=0;i<lines.size();i++)
		g2.draw((Line2D)lines.get(i));
}
```

Figura 4.31: Código para el método `paintComponent`( ).

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

El pintado de los ejes, está a cargo del método `axisText( )`, usando un objeto a `Graphics2D`.

```
// pintando ejes
public void axisText(Graphics2D g2)
{
	g2.setFont(new Font("SansSerif",Font.BOLD,12));
	g2.setColor(Color.blue);
	g2.drawString(Double.toString(a)
			,(int)(X0),(int)(ALTO+Y0+20));

	g2.drawString(Double.toString(b)
			,(int)(X0+ANCHO-20),(int)(ALTO+Y0+20));

	g2.drawString(decimales4.format(min)
		//Double.toString(min)
				,(int)(X0-40),(int)(Y0+ALTO));
	g2.drawString(decimales4.format(max)
		//Double.toString(max)
				,(int)(X0-40),(int)Y0);
	// frame
	g2.draw(new Line2D.Double(X0,Y0,X0+ANCHO,Y0));
	g2.draw(new Line2D.Double(X0,Y0,X0,Y0+ALTO));
	g2.draw(new Line2D.Double(X0,Y0+ALTO,X0+ANCHO,Y0+ALTO));
	g2.draw(new Line2D.Double(X0+ANCHO,Y0+ALTO,X0+ANCHO,Y0));

}
//opcional
public Dimension getPreferredSize()
{
	return new Dimension(500,350);
}
```

Figura 4.32: Código para el pintado de ejes; método `axisText( )`.

Los métodos `computePoints( )` y `computeScala( )`, permiten el cálculo de los puntos de la curva a graficar y el cálculo de la escala que se ajusta a dicha curva.

```
//calcular puntos a graficar
public void computePoints()
{
	Parser p = new Parser(lexema);
	double x = a,y;

	while(x <= b){
		y = p.getValue(x);
		Point2D xy = new Point2D.Double(x,y);
		points.add(xy);
		x+ = Dx;
	}

}

//calcular escala
public void computeScala()
{
	min = max=((Point2D)points.get(0)).getY();
	double x = a,y;
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
        for(int i=1;i<points.size();i++)
        {
                y = ((Point2D)points.get(i)).getY();
                if(y<min)
                        min = y;
                if(y>max)
                        max = y;
        }
        scalaY = ALTO/(max - min);
        scalaX = ANCHO/(b - a);
}
```

Figura 4.33: Los métodos `computePoints( )` y `computeScala( )`.

Finalmente, los métodos públicos `setA( )`, `setB( )` y `setLexical( )`; son los métodos para modificar los campos *`a`*, *`b`* y *`lexema`* del objeto panel; el cual como se ha mencionado en una instancia a la clase `GraphPanel`. Ver figura 4.34.

```
public void setA(double a)
{
        this.a=a;
}
public void setB(double b)
{
        this.b=b;
}
public void setLexical(String lexema)
{

        this.lexema=lexema;
}
```

Figura 4.34: Métodos públicos *`setA`*`( )`, *`setB`*`( )` y *`setLexical`*`( )`.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

4.3 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 capito 3: Diseño del Software IntegraLAB.

En la figura 4.35, se presenta la sección inicial del código para esta clase

```
/**
  *Autor : E. Ruiz Lizama
  *Fecha : 24/03/2006
  */
import javax.swing.*;
import javax.swing.border.*;
import java.awt.event.*;
import java.awt.*;
import java.io.*;
import java.text.DecimalFormat;

class integraLAB extends JFrame {
       private static final int WIDTH  =700;
       private static final int HEIGHT =500;
       private int tamaño=12, style=0, negrita=0, cursiva=0;
       private JTextArea areaTexto;
       private JLabel LFuente;
       private JButton cor,cop,peg,nue,gua,ab,acerca,ayuda,neg,cur;
       private JScrollPane scroll;
       private JComboBox tFuente,cFuente,tipoFuente;
       private JTextField nFuente;
       private Font Fuente;
       private Color colorValue[]={Color.black,
                                   Color.blue,
                                   Color.red,
                                   Color.yellow,
                                   Color.green,
                                   Color.cyan};
       private String[] FontNames={"Arial","TimesRoman","Courier","Helvetica"};
    private KeyHandler listener;
       public integraLAB() {
              super("IntegraLAB");
              try {

       UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAnd
Feel");
              }
              catch (Exception e) {
                     JOptionPane.showMessageDialog(null,"Error    al    intentar
cargar L&F");
              }
              Fuente= new Font("Arial",style,tamaño);
              areaTexto= new JTextArea();
              areaTexto.setFont(Fuente);
              menus ();
              barra();
              scroll= new JScrollPane(areaTexto);
              getContentPane().add(scroll,BorderLayout.CENTER);
              JPanel panel= new JPanel();
              JPanel panel1= new JPanel();
              JPanel panel2= new JPanel();
              panel.setBackground(Color.lightGray);
              panel1.setBackground(Color.lightGray);
```

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
            panel2.setBackground(Color.lightGray);
            getContentPane().add(panel,BorderLayout.SOUTH);
            getContentPane().add(panel1,BorderLayout.WEST);
            getContentPane().add(panel2,BorderLayout.EAST);

            // handling Key event
            listener = new KeyHandler();
            addKeyListener(listener);
            setFocusable(true);

            setSize(WIDTH,HEIGHT);
            setVisible(true);
            show();
      }

      //. . . continua codigo de las funciones miembro
      //     public void menú()
      //     public void barra() y otras

      //. . .
```

Figura 4.35: código inicial de la clase IntegraLAB.

4.3.1 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, numero 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 figura 4.36, presenta una sección de código de esta clase

```
class newtonDialog extends JDialog
    {
      String lexema="sqrt(x)";
      int metodo=1;
      JRadioButton r1;
      JRadioButton r2;
      JRadioButton r3;
      JRadioButton r4;
      JTextField salida;
      double limiteA=0.0,limiteB=4.0;
      int n=10;

       public newtonDialog(Frame owner)
      {

            super(owner, "Newton - Cotes", true);

                  JTextField funcionText;
                  final JButton btOK;

            JLabel lbl = new JLabel(new ImageIcon(
                                    "images/imagInteg.jpg"));
            JPanel p = new JPanel();
            Border b1 = new BevelBorder(BevelBorder.LOWERED);
            Border b2 = new EmptyBorder(5, 5, 5, 5);
```

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
            lbl.setBorder(new CompoundBorder(b1, b2));
            p.add(lbl);

            JPanel p1=new JPanel();
               p1.add(new JLabel("funcion"));
               funcionText=new JTextField(lexema,30);
               funcionText.addActionListener(
                      new ActionListener() {
                             public void actionPerformed(ActionEvent e)
                             {
                                    lexema=e.getActionCommand();
                             }
                      }
               );
               p1.add(funcionText);
               p.add(p1);
            getContentPane().add(p, BorderLayout.NORTH);

            JPanel p2=new JPanel();
            p2.setLayout(new BoxLayout(p2,BoxLayout.Y_AXIS));
            p2.setBorder(new TitledBorder(new EtchedBorder(),
                        "Opciones"));

            ButtonGroup group=new ButtonGroup();
            r1=new JRadioButton("Trapecio",true);
            group.add(r1);
            p2.add(r1);

            r2=new JRadioButton("Simpson 1/3",false);
            group.add(r2);
            p2.add(r2);

            r3=new JRadioButton("Simpson 3/8",false);
            group.add(r3);
            p2.add(r3);

            r4=new JRadioButton("Boole",false);
            group.add(r4);
            p2.add(r4);

            getContentPane().add(p2, BorderLayout.WEST);

            RadioButtonHandler handler=new RadioButtonHandler();
            r1.addItemListener(handler);
            r2.addItemListener(handler);
            r3.addItemListener(handler);
            r4.addItemListener(handler);

            JPanel p3=new JPanel();
            p3.setBorder(new TitledBorder(new EtchedBorder(),
                        "Datos"));

      // . . . continua codigo de la clase
```

Figura 4.36: Sección de Código de la clase newtonDialog

La función miembro `algor( )`, permite la selección de los algoritmos numéricos, escritos para el software. A continuación se presenta el código de cada uno de los métodos numéricos empleados por esta clase.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
// Algoritmos numericos
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 4.37: 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 4.38: 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 4.39: Función miembro para el método de Simpson 1/3( ).

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP


```
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 4.40: Función miembro para el método de Simpson3/8( ).

```
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 4.41: Función miembro para el método de Boole( ).

4.3.2 La clase LegendreDialog

Esta clase en cuanto al cuadro de dialogo que presenta al ser seleccionada, es idéntico al cuadro de dialogo presentado por la clase newtonDialog; pero se diferencia en que, el frame para las “opciones” o métodos, es titulado ahora “numero 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 4.42 se presenta la función miembro algor( ) de esta clase

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
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 4.42: Función miembro algor( ) para el método de Gauss-Legendre.

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

4.3.3 La clase laguerreDialog

Esta clase presenta un cuadro de dialogo similar al de la clase anterior. A continuación se presenta el código hace posible los cálculos.

```
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 4.42: Función miembro algor( ) para el método de Gauss-Laguerre.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

4.3.4 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 numero de segmentos y el valor inicial), así mismo 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 4.43; permite la selección de uno de los algoritmos numéricos para resolver EDO's.

```
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 4.43: Función miembro algor( ) para los métodos de EDO de primer orden.

En las figuras 4.44, 4.45, 4.46, y 4.47 se presenta el código Java de los algoritmos de Euler, Heun, Runge Kutta 2, y Runge Kutta 3.

```
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 4.44: Función miembro euler( ).

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
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 4.45: Función miembro 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 4.46: Función miembro rk2( ).

```
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);
```

SOME RIGHTS RESERVED

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

```
            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 4.47: Función miembro rk4( ).

4.3.5 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 dialogo que permite la solución de un sistema de dos ecuaciones diferenciales ordinarias.

El código de la función miembro algor( ), se presenta en la figura 4.47.

```
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 4.47: Función miembro algor( ) para el sistema de ecuaciones diferenciales ordinarias.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Las figuras 4.48 y 4.49 presentan una ejecución para un sistema de dos ecuaciones EDO's.

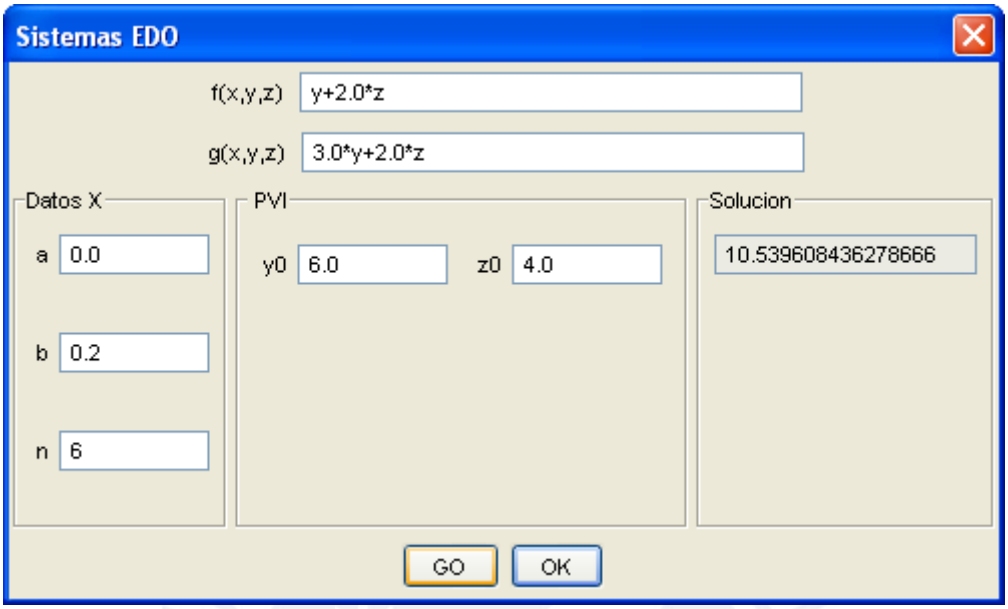


Figura 4.48 Un sistema de dos Ecuaciones EDO's.

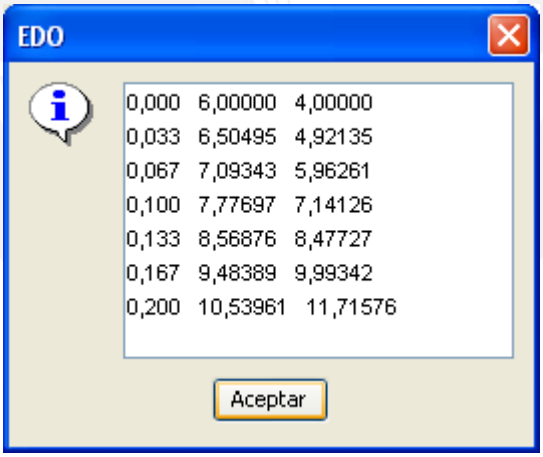


Figura 4.49: Solución reportada por IntegraLAB para el sistema EDO de la figura 4.48.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

CAPITULO 5: 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 cada método numérico implementado. 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

5.1 Prueba para el método del Trapecio

Para probar este método se toma como función testigo, el ejemplo 4.1 [Nakamura S., Pág. 111-112] cuyo enunciado y solución se trascribe a continuación.

Ejemplo 4.1: El cuerpo de revolución que se muestra en la figura 5.1 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. Evalué el error para cada *N*.

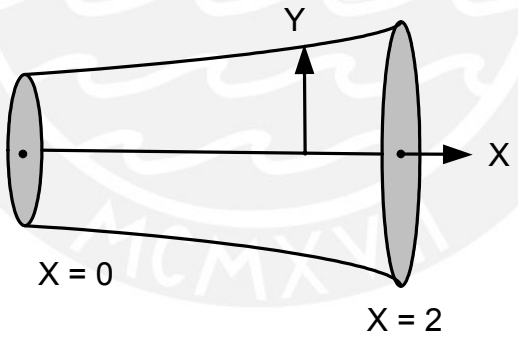


Figura 5.1: Un cuerpo de revolución

(Solución)

El volumen esta dado por

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

donde

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

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

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

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

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

$$= 12.7627$$

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

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

$$= 11.9895$$

Las integraciones con los demás valores de **N** se presentan en la tabla 5.1

Tabla 5.1

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.

Al ejecutar IntegraLAB, y escoger la opción integración, Newton-Cotes, luego escoger el método del trapecio para 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 la siguiente tabla de resultados.

Tabla 5.2: Resultados de IntegraLAB[1] 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

[1] Se han redondeado los resultados a cuatro dígitos significativos

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Como se puede observar los resultados entregados por el software son los esperados y las pequeñas discrepancias se atribuyen a los efectos producidos por los errores de redondeo.

5.2 Prueba para el método de Simpson 1/3

Para ilustrar la Regla de Simpson a 1/3, se toma como función testigo [Nakamura S., Pág. 117-118]; el ejemplo 4.3 se trascribe a continuación

Ejemplo 4.3: Repita el problema del ejemplo 4.1 utilizando la regla extendida de 1/3 de Simpson con **N** = 2, 4, 8, 16, 32, 64

(Solución)

El tamaño del intervalo es $h = 2/N$. Los cálculos para $N = 2$ y 4 son como sigue:

$N = 2$:
$$I = \frac{1}{3}\left[f(0) + 4f(1) + f(2)\right]$$
$$= \frac{1}{3}\pi\left[1 + 4(1.25^2) + 2^2\right] = 11.7809$$

$N = 4$:
$$I = \frac{0.5}{3}\left[f(0) + 4f(0.5) + 2f(1) + 4f(1.5) + f(2)\right]$$
$$= \frac{0.5}{3}\pi\left[1 + 4(1.0625) + 2(1.25)^2 + 4(1.5625)^2 + 2^2\right]$$
$$= 11.7318$$

Los cálculos par N mayores se pueden realizar de manera análoga. Los resultados y evaluaciones del error se muestran a continuación en la tabla 5.3

Tabla 5.3

N	h	I_h	E_h
2	1.	11.7809	-0.0523
4	0.5	11.7318	-0.0032
8	0.25	11.7288	-0.0002
16	0.125	11.7286	0
32	0.0625	11.7286	0
64	0.03125	11.7286	0

Al comparar los resultados anteriores con los del ejemplo 4.1 se puede ver que la regla extendida de Simpson es mucho mas precisa que l regla extendida del trapecio, utilizando el mismo número de intervalos. Por ejemplo, la exactitud de la regla extendida del trapecio con 32 intervalos es equivalente a la de la regla extendida de Simpson con tan solo 4 intervalos. El error de la

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

regla extendida del Simpson es proporcional a h^4, por lo que es de dos órdenes más grande que el de la regla extendida del trapecio. Debido al alto orden del error, la regla extendida de Simpson tiende a la solución exacta en forma más rápida que la regla extendida del trapecio cuando *h* se reduce.

Al ejecutar IntegraLAB, y escoger la opción integración, Newton-Cotes, luego escoger el método de Simpson para 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 la siguiente tabla de resultados.

Tabla 5.4: Resultados de IntegraLAB[2] para el método de Simpson 1/3.

N	***h***	I_h	E_h
2	1.	11.7810	-0.0524
4	0.5	11.7319	-0.0033
8	0.25	11.7288	-0.0002
16	0.125	11.7286	0
32	0.0625	11.7286	0
64	0.03125	11.7286	0
128	0.015625	11.7286	0

Tal como puede comprobarse con los resultados entregados por el software IntegraLAB en esta segunda prueba, son los esperados.

5.3 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 formulas 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

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

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

[2] Se han redondeado los resultados a cuatro dígitos significativos

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

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.

Los resultados al ejecutar la función 1+eXp(0-x)*sin(4*x) en el software IntegraLAB se presentan en la tabla 5.5

Tabla 5.5: 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

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

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

5.4 Prueba para las cuadraturas de Gauss-Legendre

Para probar las cuadraturas de Gauss-Legendre se toma el ejemplo 7.17 [Mathews, John; pag426]

Ejemplo 7.17. Vamos a usar la regla de Gauss-Legendre con dos nodos para aproximar

$$\int_{-1}^{1} \frac{dx}{x+2} = \ln(3) - \ln(1) \approx 1.098612288668$$

y compararemos el resultado que se obtiene con las aproximaciones dadas por la regla del trapecio $T(f,h)$ para $h=2$ y por la regla de Simpson $S(f,h)$ para $h=1$.

Sea $G_2(f)$ la aproximación que proporciona la regla de Gauss-Legendre con dos nodos, entonces

$$\begin{aligned} G_2(f) &= f(-0.57735) + f(0.57735) \\ &= 0.70291 + 0.38800 = 1.9091, \end{aligned}$$

$$\begin{aligned} T(f,2) &= f(-1.00000) + f(1.00000) \\ &= 1.00000 + 0.33333 = 1.33333, \end{aligned}$$

$$S(f,1) = \frac{f(-1) + 4f(0) + f(1)}{3} = \frac{1 + 2 + \frac{1}{3}}{3} = 1.11111.$$

Los errores son 0.00770, -0.23472 y -0.01250, respectivamente, de manera que la regla de Gauss-Legendre es la que proporciona la mejor aproximación. Hagamos notar, además, que en la Regla de Gauss-Legendre sólo se hicieron dos evaluaciones de la función, por tres en la regla de Simpson. En este ejemplo, el tamaño del error de $G_2(f)$ es un 61% del tamaño del error de $S(f,1)$.

La función ingresada a IntegraLAB para esta prueba es: 1/(x+2). Los resultados se presentan en la tabla 5.6.

Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Tabla 5.6: Resultados de IntegraLAB para $\int_{-1}^{1} \frac{dx}{x+2}$

Método numérico	Resultado I_h	E_h
Gauss-Legendre		
$n=2$	1.0909090909090908	0.007703197
$n=3$	0.999999999999999	0.098612288
$n=4$	1.019405654312171	0.079206634
$n=5$	1.0000000000999996	0.098612288
$n=6$	1.006756833477033	0.091855455
Trapecio		
$h=1/2,\ \ n=1$	1.3333333333333333	-0.234721004
Simpson a 1/3		
$h=1/3,\ n=2$	1.1111111111111112	-0.012498822

Como puede verse los resultados calculados por IntegraLAB, son los esperados. Así mismo la regla de Gauss-Legendre con sólo dos nodos de evaluación es la que obtiene una mejor aproximación; no siendo necesario un *n* mayor.

Para probar la regla de Gauss-Legendre con tres puntos o más puntos de evaluación se toma como función testigo el problema 4.18 [Nakamura, S.; pag153].

Problema 4.18. Evaluar la siguiente integral mediante la cuadratura de Gauss-Legendre con N = 3, 4, 5, y 6.

$$\int_0^1 \frac{\ln(1+x)}{x} dx$$

La solución presentada en la página 560, reporta lo siguiente:

N = 4 *I* = 0.82224
N = 6 *I* = 0.82246 I_{exacto} = 0.82246

Ahora la función ingresada a IntegraLAB es log(1+x)/x, con un intervalo de [0,1]. En la tabla 5.7, se presentan los resultados para esta prueba.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

Tabla 5.7: Resultados de IntegraLAB[3] para $\int_0^1 \frac{\ln(1+x)}{x} dx$

Método numérico	Resultado I_h	E_h
Gauss-Legendre		
$n=3$	0.82246	0.0
$n=4$	0.82247	0.00001
$n=5$	0.82247	0.00001
$n=6$	0.82247	0.00001

Como puede comprobarse los resultados de IntegraLAB son nuevamente los esperados y en esta caso Gauss-Legendre con *n* = 3 puntos de evaluación es la mejor.

5.5 Prueba para las cuadraturas de Gauss-Laguerre

Para probar las cuadraturas de Gauss-Laguerre se utiliza los ejemplos 3,6 y 8. [Davis Philip & Rabinowitz Philip; pag. 176-177]

Ejemplo 3: Resolver $e^{-2}\int_2^\infty \frac{dx}{x^{1.01}}$

La respuesta con 4 puntos de evaluación de Gauss-Laguerre es 0.27011936.

Para esta prueba se debe realizar un cambio en la función a evaluar tal que la función se transforma en

$$e^{-2}\int \frac{e^{-x}dx}{e^{-x}x^{1.01}}$$

$$= e^{-2}\int_2^\infty e^{-2} f(x)dx, \qquad f(x) = \frac{1}{e^{-x}x^{1.01}}$$

Con lo cual la función ingresada a IntegraLAB es 1/eXp(-x)*x^1.01 y la respuesta obtenida es 1.9959276520335911. Esta respuesta se multiplica ahora por e^{-2} para obtener 0.270119434.

Una vez más puede verse que el resultado es el esperado.

[3] Los resultados se han redondeado a 5 dígitos significativos

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Ejemplo 6: Resolver $e^{-2}\int_2^{\infty} e^{-x}$ (complementary error function)

La respuesta con 4 puntos de evaluación de Gauss-Laguerre es 0.00051218446.

Para esta prueba se debe realizar un cambio en la función a evaluar tal que la función se transforma en

$$e^{-2}\int_2^{\infty} e^{-x^2}\left(\frac{e^{-x}}{e^{-x}}\right)dx$$

$$= e^{-2}\int_2^{\infty} e^{-x}\left(\frac{e^{-x^2}}{e^{-x}}\right)dx, \quad f(x) = \frac{e^{-x^2}}{e^{-x}}$$

Por lo que la función ingresada a IntegraLAB es: eXp(-x*x)/eXp(-x). Siendo el resultado 0.00378455800852555157.

Ahora procedemos a multiplicar este resultado por e^{-2} y nos da 0.000512784. El cual es el valor esperado.

Ejemplo 8: Resolver $e^{-2}\int_2^{\infty} \frac{xdx}{(e^x - 1)}$ (Debye function)

La respuesta con 4 puntos de evaluación de Gauss-Laguerre es 0.058335177.

Para esta prueba se debe realizar un cambio en la función a evaluar tal que la función se transforma en

$$e^{-2}\int_2^{\infty} e^{-x}\left(\frac{xdx}{1-e^{-x}}\right)$$

y ahora la función ingresada a IntegraLAB es x/(1-eXp(0-x)), siendo la respuesta 0.43104193636242816.

La respuesta obtenida se multiplica por e^{-2} para obtener 0.0583351825444672. Tomando 9 dígitos significativos se tiene 0.058335183, el cual es para efectos prácticos el valor esperado.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

5.6 Prueba para el método de Euler

Para probar el método de Euler se toma el ejemplo 9.1 [Nakamura S., pag. 292-293]

Ejemplo 9.1

a) Resuelva $y' = -20y + 7\exp(-0.5t)$, $y(0) = 5$, por el método de Euler hacia delante con $h = 0.01$ para $0 < t \le 0.02$. Haga el cálculo a mano.

b) Repita lo mismo para $h = 0.01, 0.001,$ y 0.0001 en una computadora para $0 < t \le 0.09$. Evalué los errores de los tres cálculos mediante la comparación con la solución analítica dada por

(Solución)

a) Los primeros cálculos con $h = 0.01$ son los siguientes

$t_0 = 0, \qquad y_0 = y(0) = 5$

$t_1 = 0.01, \qquad y_1 = y_0 + hy_0' = 5 + (0.01)(-20(5) + 7\exp(0)) = 4.07$

$t_2 = 0.02, \qquad y_2 = y_1 + hy_1' = 4.04 + (0.01)(-20(4.07) + 7\exp(-0.005))$
$= 3.32565$

.

.

.

$t_n = nh, \qquad y_n = y_{n-1} + hy_{n-1}'$

b) Los resultados computacionales para los valores seleccionados de t, con tres valores de intervalos de tiempo (espaciamiento de la retícula) se muestran en la tabla 5.8

Tabla 5.8: Método de Euler hacia delante

t	$h = 0.01,$	$h = 0.001,$	$h = 0.0001$
0.01	4.07000 (8.693)	4.14924 (0.769)	4.15617 (0.076)[a]
0.02	3.32565 (14.072)	3.45379 (1.259)	3.46513 (0.124)
0.03	2.72982 (17.085)	2.88524 (1.544)	2.89915 (0.153)
0.04	2.25282 (18.440)	2.42037 (1.684)	2.43554 (0.167)
0.05	1.87087 (18.658)	2.04023 (1.722)	2.05574 (0.171)
0.06	1.56497 (18.125)	1.72932 (1.690)	1.74454 (0.168)
0.07	1.31990 (17.119)	1.47496 (1.613)	1.48949 (0.160)
0.08	1.12352 (15.839)	1.26683 (1.507)	1.28041 (0.150)
0.09	0.96607 (14.427)	1.09646 (1.387)	1.10895 (0.138)
0.10	0.83977 (12.979)	0.95696 (1.261)	0.96831 (0.126)

[a] (error) X 100

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Comentarios: La exactitud del método de Euler hacia delante aumenta al disminuir el intervalo de tiempo h. En efecto, las magnitudes de los errores son aproximadamente proporcionales a h. Sin embargo, una reducción mayor de h, sin el uso de la doble precisión, tiene sus desventajas, puesto que aumenta el error numérico debido al redondeo.

Para esta prueba la función ingresada a IntegraLAB es -20*y+7*eXp(0-0.5*x). En la tabla 5.9 se presentan los cálculos realizados por IntegraLAB.

Tabla 5.9: Resultados de IntegraLAB al utilizar el método de Euler para $y' = -20y + 7\exp(-0.5t)$, $y(0) = 5$ en $[0,0.1]$

t	$h = 0.01,$	$h = 0.001,$	$h = 0.0001$
0.01	4.07000 (8.693)	4.14924 (0.769)	4.15617 (0.076)[a]
0.02	3.32565 (14.072)	3.45379 (1.259)	3.46513 (0.124)
0.03	2.72982 (17.085)	2.88524 (1.544)	2.89915 (0.153)
0.04	2.25282 (18.440)	2.42037 (1.684)	2.43554 (0.167)
0.05	1.87087 (18.658)	2.04023 (1.722)	2.05574 (0.171)
0.06	1.56497 (18.125)	1.72932 (1.690)	1.74454 (0.168)
0.07	1.31990 (17.119)	1.47496 (1.613)	1.48949 (0.160)
0.08	1.12352 (15.839)	1.26683 (1.507)	1.28041 (0.150)
0.09	0.96607 (14.427)	1.09646 (1.387)	1.10895 (0.138)
0.10	0.83977 (12.979)	0.95696 (1.261)	0.96831 (0.126)

[a] (error) X 100

Como se observa en la tabla 5.9 los resultados entregados por IntegraLAB son los esperados.

5.7 Prueba para el método de Heun

Para probar el método de Heun se utiliza el ejemplo 9.6 [Mathews, John; pp484].

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 $h = 1, \frac{1}{2}, \frac{1}{4}, \text{ y } \frac{1}{8}$.

En la figura 9.8 se muestran las graficas de las dos primeras soluciones dadas por el método de Heun y la grafica de la solución exacta $y(t) = 3e^{-t/2} - 2 + t$. En la tabla 9.4 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

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

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

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

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

La iteración continúa hasta que lleguemos al último paso

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

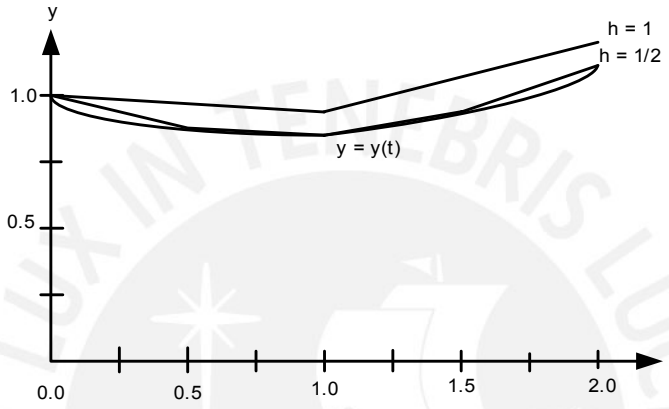


Figura 5.2: Comparación de las soluciones obtenidas con el método de Heun con diferentes tamaños de paso para $y' = (t - y)/2$ en $[0,3]$ con la condición inicial $y(0) = 1$.

Tabla 9.4. 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

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

La función ingresada a IntegraLAB para esta prueba es (x-y)/2. La tabla 5.10 presenta los resultados del software.

Tabla 5.10: 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

Nuevamente observamos que los resultados entregados por IntegraLAB son los esperados.

5.8 Prueba para el método de Runge-Kutta RK2

Para probar Runge-Kutta de orden dos se utiliza el ejemplo 25.6 [CHAPRA Steven. y CANALE Raymond; pag740-742].

Ejemplo 25.6 Enunciado del problema. Use el método de punto medio (25.37) y el método de Ralston (25.38) para integrar numéricamente la ecuación $f(x,y) = -2x^3 + 12x^2 - 20x + 8.5$

$$y_{i+1} = y_i + k_2 h \qquad (25.37)$$

donde

$$k_1 = f(x_i, y_i) \qquad (25.37a)$$

$$k_2 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h) \qquad (25.37b)$$

$$y_{i+1} = y_i + (\frac{1}{3}k_1 + \frac{2}{3}k_2)h \qquad (25.38)$$

donde

$$k_1 = f(x_i, y_i) \qquad (25.38a)$$

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

$$k_2 = f(x_i + \frac{3}{4}h, y_i + \frac{3}{4}k_1 h) \quad (25.38b)$$

desde $x = 0$ hasta $x = 4$ usando un tamaño de paso de 0.5. La condición inicial en $x = 0$ es $y = 1$. Compare los resultados con valores obtenidos con otro algoritmo RK de segundo orden: el método de Heun sin corrector de iteración.

Solución.

El primer paso en el método de punto medio es el uso de la ecuación (25.37a) para calcular

$$k_1 = -2(0)^3 + 12(0)^2 - 20(0) + 8.5 = 8.5$$

Sin embargo, como la EDO es una función solo de x, tal resultado carece de relevancia sobre el segundo paso [el uso de la ecuación (25.37b)] para calcular

$$k_2 = -2(0.25)^3 + 12(0.25)^2 - 20(0.25) + 8.5 = 4.21875$$

Observe que tal estimación de la pendiente es mucho más cercana al valor promedio para el intervalo (4.4375) que la pendiente al inicio del intervalo (8.5) que podría haber sido usada por el procedimiento de Euler. La pendiente en el punto medio puede entonces sustituirse en la ecuación (25.37) para predecir

$$y(0.5) = 1 + 4.21875(0.5) = 3.109375, \qquad \varepsilon_t = 3.4\%$$

El cálculo se repite y los resultados se resumen en la tabla 25.3

Tabla 25.3: Comparación de los valores verdadero y aproximado de la integral de $y' = -2x^3 + 12x^2 - 20x + 8.5$, con la condición inicial de $y = 1$ en $x = 0$. Los valores aproximados se calculan por medio de tres versiones de los métodos RK de segundo orden con un tamaño de paso de 0.5.

		Heun		Punto medio		RK Ralston de segundo orden	
x	$y_{verdadero}$	y	$I_{\varepsilon_t}I(\%)$	y	$I_{\varepsilon_t}I(\%)$	y	$I_{\varepsilon_t}I(\%)$
0.0	1.00000	1.00000	0	1.00000	0	1.00000	0
0.5	3.21875	3.43750	6.8	3.109375	3.4	3.277344	1.8
1.0	3.00000	3.37500	12.5	2.81250	6.3	3.101563	3.4
1.5	2.21875	2.68750	21.1	1.984375	10.6	2.347656	5.8
2.0	2.00000	2.50000	25.0	1.75	12.5	2.140625	7.0
2.5	2.71845	3.18750	17.2	2.484375	8.6	2.855469	5.0
3.0	4.00000	4.37500	9.4	3.81250	4.7	4.117188	2.9
3.5	4.71875	4.93750	4.6	4.609375	2.3	4.800781	1.7
4.0	3.00000	3.00000	0	3.00000	0	3.031250	1.0

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

Por medio del método de Ralston k_1 para el primer intervalo es también igual a 8.5 y [vease ecuación 25.38b]

$$k_2 = -2(0.375)^3 + 12(0.375)^2 - 20(0.375) + 8.5 = 2.58203125$$

La pendiente promedio se calcula por

$$\phi = \frac{1}{3}(8.5) + \frac{2}{3}(2.5803125) = 4.546875$$

la cual se usara para predecir

$$y(0.5) = 1 + 4.5546875(0.5) = 3.27734375 \qquad \varepsilon_t = -1.82\%$$

Los cálculos se repiten y los resultados se resumen en la tabla 25.3. Observe que todos los métodos RK de segundo orden son superiores al método de Euler.

La función ingresada a IntegraLAB es: -2*x^3+12*x^2-20*x+8.5, la misma que es evaluada desde $x = 0$ hasta en $x = 4$, usando un tamaño de paso de 0.5. La condición inicial en $x = 0$ es $y = 1$. La tabla 5.11 presenta los resultados del software.

Tabla 5.11: Resultados de IntegraLAB para la función $f(x, y) = -2x^3 + 12x^2 - 20x + 8.5$ desde $x = 0$ hasta en $x = 4$, usando un tamaño de paso de 0.5. La condición inicial en $x = 0$ es $y = 1$.

		Runge-Kutta de segundo orden	
x	$y_{verdadero}$	y	$I_{\varepsilon_t} I(\%)$
0.0	1.00000	1.00000	0
0.5	3.21875	3.109375	3.4
1.0	3.00000	2.812500	6.3
1.5	2.21875	1.984375	10.6
2.0	2.00000	1.750000	12.5
2.5	2.71845	2.484375	8.6
3.0	4.00000	3.812500	4.7
3.5	4.71875	4.609375	2.3
4.0	3.00000	3.000000	0

Una vez más se puede observar que los resultados de IntegraLAB son los esperados. Esta vez coinciden con los del punto medio presentados en la tabla 25.3, presentada en la página anterior.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

5.9 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].

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 5.12.

Tabla 5.12: 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

La función ingresada a IntegraLAB para esta prueba es: –y +x^2+1; con a = 0.0, b = 2.0, n = 10 y el valor inicial y(0) = 0.5; siendo los resultados los mostrados en la tabla 5.13.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Tabla 5.13: Resultados de evaluar $y' = y - t^2 + 1,\ 0 \le t \le 2,\qquad y(0) = 0.5,$ 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

La tabla 5.13 muestra que los resultados nuevamente son los esperados.

5.10 Prueba para el método de Solución de sistema de ecuaciones diferenciales ordinarias

Para probar este método se realizan dos pruebas.

Prueba 1: Se utiliza el ejemplo 1 [BURDEN Richard L. y FAIRES J. Douglas; pag 316, 317, 318].

Ejemplo 1: La ley de Kirchoff establece que la suma de todos los cambios instantáneos de voltaje alrededor de un circuito cerrado es cero. Esta ley implica que en un circuito cerrado que contenga una resistencia de R ohms, una capacitancia de C faradios, una inductancia de L henrios y una fuente de voltaje de $E(t)$ voltios, la corriente $I(t)$ satisface la ecuación

$$LI'(t) + RI(t) + \frac{1}{C}\int I(t)d(t) = E(t)$$

Las corrientes $I_1(t)$ e $I_2(t)$ en los ciclos izquierdo y derecho, respectivamente, del circuito que se muestra en la figura 5.6 son soluciones del sistema de ecuaciones

$$2I_1(t) + 6\left[I_1(t) + I_2(t) + 2I_1'(t) = 12\right]$$
$$\frac{1}{0.5}\int I_2(t)dt + 4I_2(t) + 6\left[I_2(t) - I_1 t\right] = 0$$

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Supongamos que el interruptor del circuito se encuentra cerrado en el instante $t=0$. Entonces $I_1(0)$ e $I_2(0)$. Se resuelve para $I_1'(t)$, al diferenciar la segunda ecuación y al sustituir la primera en el resultado, se obtiene el sistema

$$I_1' = f_1(t,I_1,I_2) = -4I_1 + 3I_2 + 6, \qquad I_1(0) = 0,$$
$$I_2' = f_2(t,I_1,I_2) = 0.6I_1' - 0.2I_2 = -2.4I_1 + 1.6I_2 + 3.6, \qquad I_2(0) = 0.$$

La solución exacta del sistema es

$$I_1(t) = -3.375e^{-2t} + 1.875e^{-0.4t} + 1.5,$$
$$I_2(t) = -2.25e^{-2t} + 2.25e^{-0.4t}.$$

Aplicaremos el método de Runge-Kutta de cuarto orden con $h=0.1$ a este sistema.

Dado que $w_{1,0} = I_1(0) = 0$ y $w_{2,0} = I_2(0) = 0,$

$$k_{1,1} = hf_1(t_0,w_{1,0},w_{2,0}) = 0.1f_1(0,0,0) = 0.1[-4(0)+3(0)+6] = 0.6,$$
$$k_{1,2} = hf_2(t_0,w_{1,0},w_{2,0}) = 0.1f_2(0,0,0) = 0.1[-2.4(0)+1.6(0)+3.6] = 0.36,$$
$$k_{2,1} = hf_1(t_0+\frac{1}{2}h, w_{1,0}+\frac{1}{2}k_{1,1}, w_{2,0}+\frac{1}{2}k_{1,2}) = 0.1f_1(0.05,0.3,0.18)$$
$$= 0.1[-4(0.3)+3(0.18)+6] = 0.534,$$
$$k_{2,2} = hf_2(t_0+\frac{1}{2}h, w_{1,0}+\frac{1}{2}k_{1,1}, w_{2,0}+\frac{1}{2}k_{1,2}) = 0.1f_2(0.05,0.3,0.18)$$
$$= 0.1[-2.4(0.3)+1.6(0.18)+3.6] = 0.3168,$$

Al generar los valores restantes en una forma semejante, se obtiene

$$k_{3,1} = (0.1)f_1(0.05,0.267,0.1584) = 0.54072,$$
$$k_{3,2} = (0.1)f_2(0.05,0.267,0.1584) = 0.321264,$$
$$k_{4,1} = (0.1)f_1(0.1,0.54072,0.321264) = 0.4800912,$$

y

$$k_{4,2} = (0.1)f_2(0.1,0.54072,0.321264) = 0.28162944,$$

En consecuencia

$$I_1(0.1) \approx w_{1,1} = w_{1,0} + \frac{1}{6}(k_{1,1} + 2k_{2,1} + 2k_{3,1} + k_{4,1})$$
$$= 0 + \frac{1}{6}[0.6 + 2(0.534) + 2(0.54072) + 0.4800912] = 0.5382552$$

y

$$I_2(0.1) \approx w_{2,1} = w_{2,0} + \frac{1}{6}(k_{1,2} + 2k_{2,2} + 2k_{3,2} + k_{4,2}) = 0.3196263$$

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

El resto de los valores presentados en la tabla 5.14 se generan de manera parecida.

Tabla 5.14: Resultado de aplicar el método Runge-Kutta de cuarto orden al sistema de ecuaciones del ejemplo 1.

t_j	$w_{1,j}$	$w_{2,j}$	$\lvert I_1(t_j) - w_{1,j} \rvert$	$\lvert I_2(t_j) - w_{2,j} \rvert$
0.0	0	0	0	0
0.1	0.5382552	0.3196263	0.8907×10^{-5}	0.5744×10^{-5}
0.2	0.9684983	0.5687817	0.1469×10^{-4}	0.9976×10^{-5}
0.3	1.310717	0.7607328	0.1955×10^{-4}	0.1200×10^{-4}
0.4	1.581263	0.9063208	0.2135×10^{-4}	0.1256×10^{-4}
0.5	1.793505	1.014402	0.2205×10^{-4}	0.1345×10^{-4}

Para ingresar el sistema de ecuaciones EDO a IntegraLAB, estas se definen en términos de $f(x,y,z)$ y $g(x,y,z)$. Con lo cual el sistema es

$$f(x,y,z) = y = -4y + 3z + 6, \quad y(0) = 0$$
$$g(x,y,z) = z = 2.4y + 1.6z + 3.6, \quad z(0) = 0$$

Por tanto para $f(x,y,z)$ se ingresa -4.0*y+3.0*z+6.0 y para $g(x,y,z)$ se ingresa -2.4*y+1.6*z+3.6; con $a = 0$, $b = 0.5$, $n = 5$, $y0 = 0$, $z0 = 0$. La tabla 5.15 muestra los resultados devueltos por IntegraLAB.

Tabla 5.15: Resultado de aplicar el método Runge-Kutta de cuarto orden al sistema de ecuaciones del ejemplo 1, con IntegraLAB.

t_j	$w_{1,j}$	$w_{2,j}$	$\lvert I_1(t_j) - w_{1,j} \rvert$	$\lvert I_2(t_j) - w_{2,j} \rvert$
0.0	0	0	0	0
0.1	0.5382552	0.3196262	0.8707×10^{-5}	0.5844×10^{-5}
0.2	0.9684987	0.5687822	0.1429×10^{-4}	0.9476×10^{-5}
0.3	1.3107190	0.7607331	0.1755×10^{-4}	0.1101×10^{-4}
0.4	1.5812652	0.9063206	0.1915×10^{-4}	0.1276×10^{-4}
0.5	1.7935075	1.0144024	0.1955×10^{-4}	0.1305×10^{-4}

De la tabla 5.15, podemos observar que los resultados entregados por IntegraLAB son los esperados y hasta con una mayor precisión.

Prueba 2: Del conjunto de ejercicios 5.9 [BURDEN Richard L. y FAIRES J. Douglas; pag 323], se escoge el problema numero siete cuyo enunciado se presenta a continuación.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

7. El estudio de los modelos matemáticos para predecir la dinámica demográfica de especies antagónicas nació con las obras independientes que, en la primera parte del siglo XX, publicaron A.J. Lotka y V. Volterra. Considere el problema de predecir la población de dos especies, una de las cuales es depredadora y cuya población en el tiempo t es $x_2(t)$ y la otra es la presa, cuya población es $x_1(t)$. Supondremos que la presa dispone siempre de suficiente comida y que su natalidad en cualquier momento es proporcional a la cantidad de presas vivas en ese tiempo; es decir, la natalidad de la presa es $k_1x_1(t)$. La mortalidad de la presa depende del número de presas y de depredadores vivos en ese tiempo. Para simplificar los cálculos, supondremos que la mortalidad (de la presa) es igual a $k_2x_1(t)x_2(t)$. En cambio, la natalidad del depredador depende del suministro de comida $x_1(t)$ y también del número de depredadores que intervienen en el proceso de reproducción. Por tal razón, suponemos que la natalidad (de los depredadores) es $k_3x_1(t)x_2(t)$. Supondremos que su mortalidad es proporcional a la cantidad de depredadores vivos en el tiempo; es decir, mortalidad (de los depredadores) es igual a $k_4x_2(t)$.

Dado que $x_1'(t)$ y $x_2'(t)$ representan, respectivamente, el cambio de las poblaciones de presas y depredadores en el tiempo, el problema se expresa mediante el sistema de ecuaciones diferenciales no lineales.

$$x_1'(t) = k_1x_1(t) - k_2x_1(t)x_2(t) \quad \text{y}$$

$$x_2'(t) = -k_4x_2(t) + k_3x_1(t)x_2(t).$$

Resuelva este sistema para $0 \le t \le 4$, suponiendo que la población inicial de la presa es de 1000 y la de los depredadores es de 500, y que las constantes son $k_1 = 3,\ k_2 = 0.002,\ k_3 = 0.0006$ y $k_4 = 0.5$.

Para resolver este sistema EDO en IntegraLAB, se debe reescribir el sistema en términos de $f(x,y,z)$ y $g(x,y,z)$. Con lo cual el sistema es

$$f(x,y,z) = y = 3y - 0.002yz,\ \ y(0) = 500$$
$$g(x,y,z) = z = -0.5z + 0.0006yz,\qquad z(0) = 1000$$

Por tanto para $f(x,y,z)$ se ingresa 3*y-0.002*y*z, y para $g(x,y,z)$ se ingresa -0.5*z+0.0006*y*z; con $a = 0$, $b = 4$, $n = 4$. La tabla 5.16 presenta los resultados devueltos por IntegraLAB.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Tabla 5.16: Resultado de aplicar el método Runge-Kutta de cuarto orden al sistema de ecuaciones del modelo Lotka Volterra, con IntegraLAB. ([a])

t_i	Depredadores	Presas
0	500	1000
1.0	1428.9	1044.9
2.0	1735.2	1881.3
3.0	498.3	2150.8
4.0	244.7	1582.0

([a]) Los resultados se han redondeado a 1 decimal.

5.11 Prueba para Graficar funciones

Por defecto IntegraLAB presenta una grafica predefinida al invocar el graficador de funciones, con la secuencia de teclas [Ctrl] + [G]. Esta se nuestra en la figura 5.3

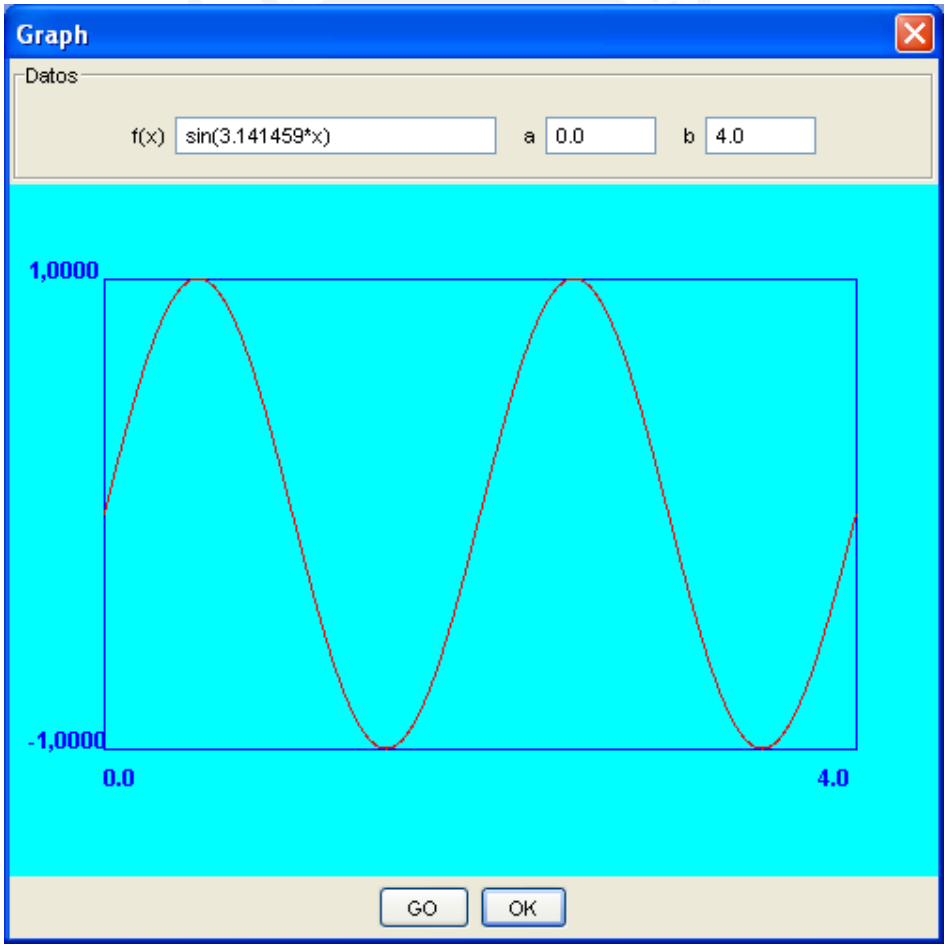


Figura 5.3: Grafica por defecto de IntegraLAB.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

Prueba 1: Graficaremos la función proporcionada en el ejemplo 4.1 [Nakamura S., Pág. 111-112].

Presionamos las teclas [Ctrl] + [G] para invocar al ambiente graficador de IntegraLAB, e ingresamos la función 3.14159*(1+(x/2)^2)^2; con un intervalo de [0, 2]. La figura obtenida se presenta en la figura 5.4.

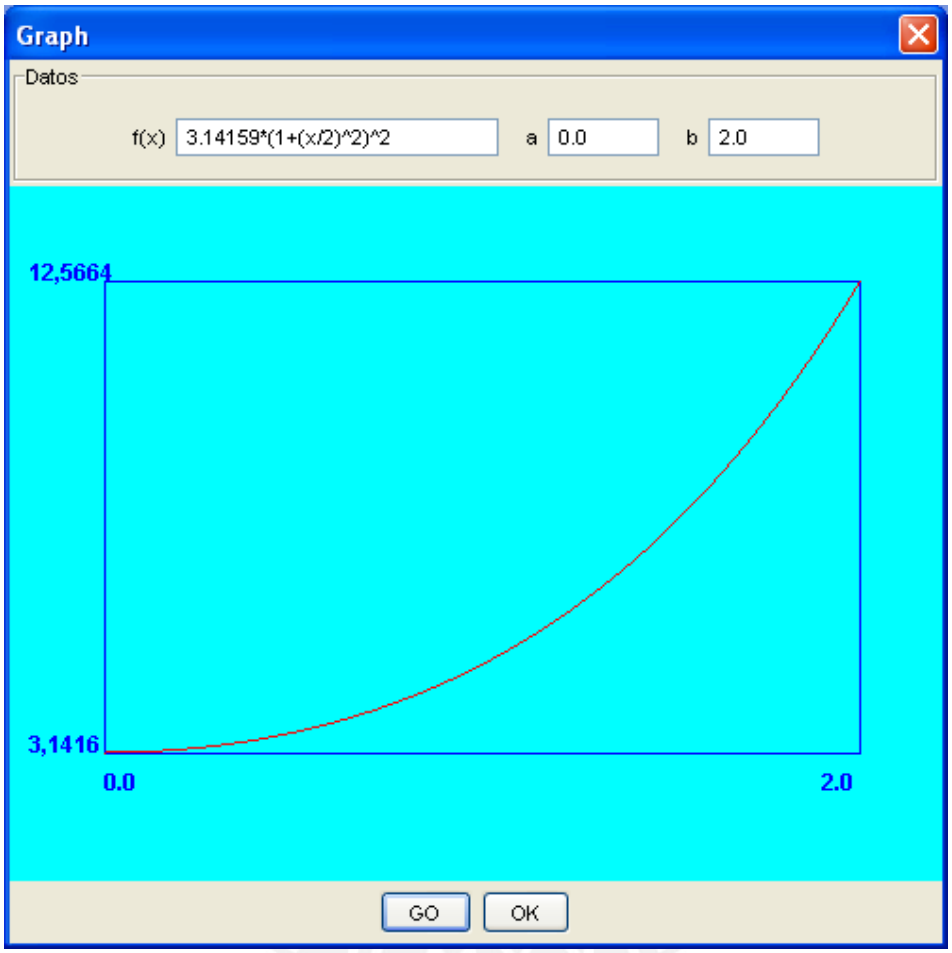


Figura 5.4: Grafica de la función 3.14159*(1+(x/2)^2)^2.

Prueba 2: Graficaremos la función proporcionada en el ejemplo 7.2 [Mathews, John; pag. 377].

Presionamos las teclas [Ctrl] + [G] para invocar al ambiente graficador de IntegraLAB, e ingresamos la función 1+eXp(0-x)*sin(4*x), en el intervalo $a = 0$ y $a = 1$. La figura obtenida se presenta en la figura 5.5.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

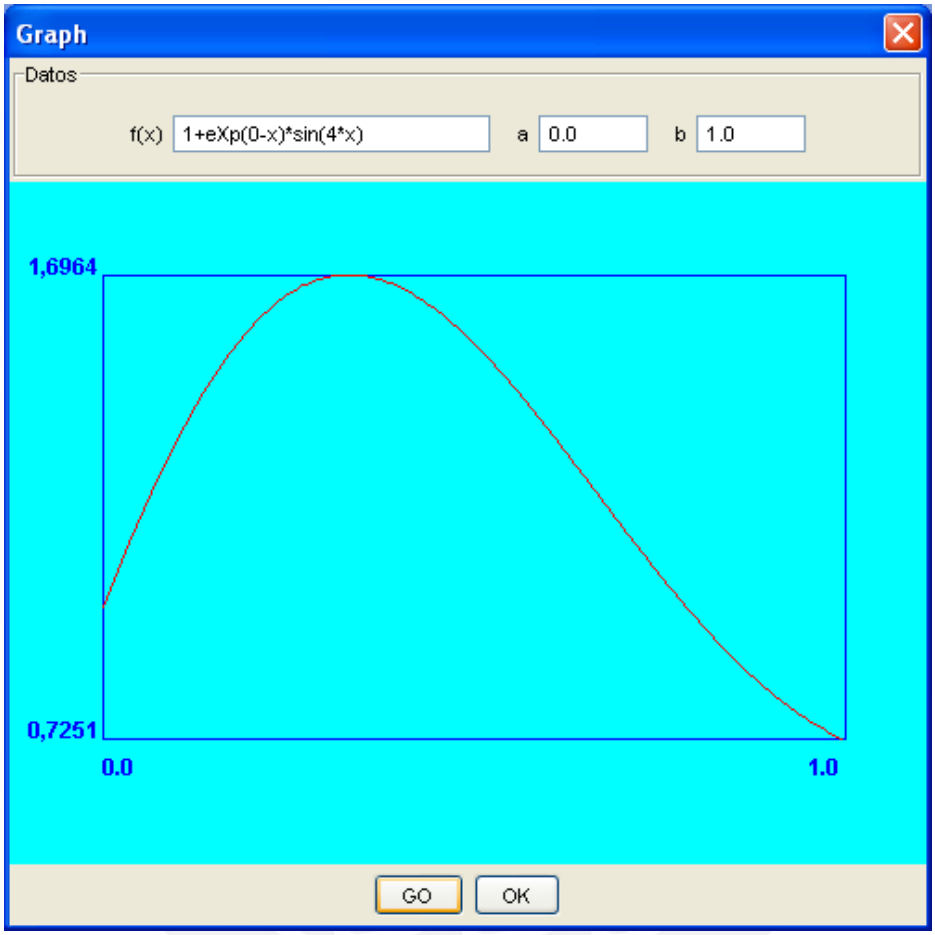


Figura 5.5: Grafica de la función 1+eXp(0-x)*sin(4*x),

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

CAPITULO 6: CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

1. IntegraLAB es un software que implementa algoritmos numéricos para la solución de problemas de integración de funciones y solución de ecuaciones diferenciales

2. Los métodos numéricos implementados para integración de funciones son:

   Cuadraturas de Newton-Cotes: Regla del Trapecio, Regla de Simpson a 1/3, Regla de Simpson a 3/8, Regla de Boole.

   Cuadraturas de Gauss-Legendre

   Cuadraturas de Gauss-Laguerre

3. Los métodos numéricos para la solución de ecuaciones diferenciales ordinarias que implementa IntegraLAB son:

   Método de Euler

   Método de Heun

   Método de Runge-Kutta RK4

   Método de Runge-Kutta RK2

   Solución de sistema de ecuaciones diferenciales ordinarias

4. IntegraLAB utiliza un parser para el manejo de funciones, de modo tal que cada función a evaluar es analizada desde la línea de entrada.

5. Se utiliza la GUI de Java Swing, para lograr un diseño amistoso al usuario.

6. IntegraLAB posee un ambiente de despliegue gráfico el mismo que permite graficar una función desde la línea de entrada.

7. Para el desarrollo de IntegraLAB, se utilizan los principios de la Ingeniería del Software.

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

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis

TESIS PUCP

6.2 RECOMENDACIONES

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

2. 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 caracter.

3. 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 seria muy útil para visualizar el comportamiento de un sistema de ecuaciones diferenciales ordinarias con respecto al tiempo.

SOME RIGHTS RESERVED
Tesis publicada con autorización del autor
Algunos Derechos Reservados. No olvide citar esta tesis