INSTITUTO TECNOLÓGICO SUPERIOR DE JEREZ

4to Semestre

Ingeniería en Sistemas Computacionales

MÉTODOS NUMÉRICOS

Actividad: Reporte “Programa Ecuación Diferencial”

Docente: I. S. C. Jesús Aranda Gamboa

Alumna: Pritschella Berenice Flores Estrada

Correo Electrónico: prits99@hotmail.com

No. Control: S17070169

Jerez De García Salinas, Zac.

31/05/ 2019.

INTRODUCCIÓN

En el presente documento, se habla acerca de un programa que resuelve una ecuación diferencial a través del método de Euler.

Éste método necesita funciones, un punto inicial, una condición inicial, un punto final y un número de intervalos.

OBJETIVOS

Objetivo General:

Se planea hacer un programa que resulta una ecuación diferencial a través del método de Euler, poniendo en práctica lo que se aprendió en la asignatura "métodos numéricos".

Objetivo Específico:

Explicar el funcionamiento del programa.

Dar a conocer los resultados de dicho programa.

RESULTADOS

En primera instancia, se pusieron en práctica los métodos en clase, para así poder comprenderlos de una mejor forma.

Posteriormente se realizó un programa que realiza lo que se puso en práctica en las clases. Dicho programa se muestra a continuación.

```
import javax.swing.*;

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.lang.reflect.Method;
import java.util.LinkedList;
import java.util.Queue;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

class Expresion{ //Clase que obtiene la función y la evalúa
    enum TipoToken  {   NUMERO, VARIABLE, FUNCION, ADD, SUB, MUL, DIV,
                        EXP, P_IZQ, P_DER, ERROR };
    class Token {
        TipoToken tipo;
        String    texto;
        Token(TipoToken ti, String te) { tipo=ti; texto=te;}
```

```
        Token(TipoToken ti) { tipo = ti; }
    }
    Queue<Token>    colaTokens;
    String          cadenaFuncion;
    double          variable;

    Expresion(String c) throws Exception {//solo recibe como parametro la
funcion y la ingresa en una variable de instancia
        cadenaFuncion = c;
    }

    public void generarTokens() throws Exception {//separa la funcion en
"tokens" y por tipo caracter que son
        colaTokens = new LinkedList<Token>();//inicializar la lista
        StringBuffer entrada   = new StringBuffer(cadenaFuncion);//variables de
instancia

        Pattern pNumero = Pattern.compile("\\-?\\d+(\\.\\d+)?");
        Pattern pID     = Pattern.compile("\\p{Alpha}\\w+");

        while(entrada.length()>0) {//ciclo que recorre toda la funcion
            Matcher      m  = pNumero.matcher(entrada);
            if(m.lookingAt()) {//compara que sea numero
                colaTokens.add(new Token(TipoToken.NUMERO,m.group()));
                entrada.delete(0, m.end());
                continue;
            }
            if(entrada.charAt(0) == 'x' || entrada.charAt(0) == 'X') {//compara
que la variable sea x o X
                colaTokens.add(new Token(TipoToken.VARIABLE,"x"));
                entrada.deleteCharAt(0);
                continue;
            }
            if(entrada.charAt(0) == '+') {//compara el signo de + para saber si
es suma
                colaTokens.add(new Token(TipoToken.ADD));
                entrada.deleteCharAt(0);
                continue;
            }
            if(entrada.charAt(0) == '-') {//compara el signo de menos para
realizar la resta
                colaTokens.add(new Token(TipoToken.SUB));
                entrada.deleteCharAt(0);
                continue;
            }
            if(entrada.charAt(0) == '*') {//compara el * para saber que es
multiplicacion
                colaTokens.add(new Token(TipoToken.MUL));
                entrada.deleteCharAt(0);
                continue;
            }
            if(entrada.charAt(0) == '/') {//el signo / es para la division
                colaTokens.add(new Token(TipoToken.DIV));
                entrada.deleteCharAt(0);
                continue;
```

```
            }
            if(entrada.charAt(0) == '(') {//identifica los parentesis abiertos
                colaTokens.add(new Token(TipoToken.P_IZQ));
                entrada.deleteCharAt(0);
                continue;
            }
            if(entrada.charAt(0) == ')') {//identifica los parentesis de cierre
                colaTokens.add(new Token(TipoToken.P_DER));
                entrada.deleteCharAt(0);
                continue;
            }
            if(entrada.charAt(0) == '^') {//identifica el exponencial para
realizar su calculo
                colaTokens.add(new Token(TipoToken.EXP));
                entrada.deleteCharAt(0);
                continue;
            }
            m = pID.matcher(entrada);
            if(m.lookingAt()) {//si la variable m contenga el valor por decirlo
de alguna manera y lo elimina de "entrada"
                colaTokens.add(new Token(TipoToken.FUNCION, m.group()));
                entrada.delete(0, m.end());
                continue;
            }
            throw new Exception("Elemento no reconocido en la entrada: "
                                + entrada.charAt(0));//esta ecepcion es por que
el elemento no e pudo identificar
        }//fin ciclo
    }//fin metodo generarTokens

    public double evaluar(double x) throws Exception {//este metodo es el que se
manda a llamar paara evaluar la funcion con

      //el valor deseado
        generarTokens();
        variable = x;
        return expresion();
    }//fin metodo evaluar

    public double expresion() {//realiza las operaciones de suma y resta
        double respuesta=termino();
        while(!colaTokens.isEmpty() ) {
            switch(colaTokens.element().tipo) {
                case ADD:   colaTokens.remove();
                            respuesta+=termino();
                            continue;
                case SUB:   colaTokens.remove();
                            respuesta-=termino();
                            continue;
            }
            break;
        }
        return respuesta;
    }//fin termino expresion
```

```
    public double termino() {//realiza las operaciones de multiplicacion y
division
        double respuesta=factor();
        while(!colaTokens.isEmpty() ) {
            switch(colaTokens.element().tipo) {
                case MUL:   colaTokens.remove();
                            respuesta*=factor();
                            continue;
                case DIV:   colaTokens.remove();
                            respuesta/=factor();
                            continue;
                default:
            }
            break;
        }
        //System.out.println("Termino respuesta: "+respuesta);
        return respuesta;
    }//fin metodo termino

    public double factor() {//realiza las operaciones de exponentes
        double respuesta=valor();
        while(!colaTokens.isEmpty() ) {
            switch(colaTokens.element().tipo) {
                case EXP:   colaTokens.remove();
                            respuesta=Math.pow(respuesta,valor());
                            //System.out.println("Factor respuesta "+respuesta);
                            continue;
            }
            break;
        }

        return respuesta;
    }//fin metodo factor

    public double valor() {//el metodo se encarga de separar los tokens por
tippo y posicion
        Token token;
        try {
            double respuesta = 0;
            token     = colaTokens.poll();
            switch(token.tipo) {
                case P_IZQ:     respuesta = expresion();
                                leaToken(TipoToken.P_DER);
                                return respuesta;
                case NUMERO:    return Double.parseDouble(token.texto);
                case VARIABLE:  return variable;
                case FUNCION:   leaToken(TipoToken.P_IZQ);
                                double argumento=expresion();
                                leaToken(TipoToken.P_DER);
                                Method m = java.lang.Math.class.
                                           getMethod(token.texto, Double.TYPE);
                                return (Double) m.invoke(null, argumento);
            }

        }
```

```
        catch(Exception ex) {
            System.err.println("Error: "  + ex.getMessage());
            System.exit(0);
        }
        return 0;
    }//fin  metodo valor

    public boolean leaToken(TipoToken t) {
        Token token = colaTokens.poll();
        if(token.tipo.equals(t)) {
            return true;
        }
        else {
            System.err.println("Error: elemento no permitido "  + token.texto );
            return false;
        }
    }//fin metodo lea tokens

}//fin class expresion

class VentanaInicio extends JFrame implements ActionListener{ //Clase donde se
implementará la interfaz

                                        //Componentes que se van a utilizar
      JTextField cajaDerivada, cajaXCero, cajaYCero, cajaN, cajaXi,
cajaResultado;  //Cajas de texto que se utilizarán
      JButton solucion; //Botón solución


      public VentanaInicio() { //Método que da incio a la ventana de la
interfaz

                        //Configuración de la pantalla principal
            getContentPane().setLayout(null);
            setSize(500, 500);  //Medida de la interfaz
            setTitle("ECUACIONES DIFERENCIALES");  //Título de la interfaz
            setLocationRelativeTo(null);
            setVisible(true);
            setDefaultCloseOperation(EXIT_ON_CLOSE);

                        //Configuración de componentes

            //Explicacón las variables que se van a utilizar

            JLabel texto = new JLabel("Ingresa: ");  //Texto Ingresa
            texto.setBounds(80, 10, 80, 20);  //Tamaño y posicion del texto
Ingresa
            add(texto);        //colocación del componente en la interfaz

            JLabel dxdy = new JLabel("dx/dy: Funciones de x y de y"); //Texto
dx/dy
            dxdy.setBounds(20, 40, 160, 20);
            add(dxdy);

            JLabel x0 = new JLabel("X0: Punto inicial"); //Texto x0
```

```
            x0.setBounds(20, 60, 150, 20);
            add(x0);

            JLabel y0 = new JLabel("y0: Condición inicial"); //Texto y0
            y0.setBounds(20, 80, 150, 20);
            add(y0);

            JLabel xi = new JLabel("Xi: Punto final"); //Texto Xi
            xi.setBounds(20, 100, 150, 20);
            add(xi);

            JLabel n = new JLabel("n: Número de intervalos");  //Texto n
            n.setBounds(20, 120, 150, 20);
            add(n);


        //*****************************************************************

            //Componentes que se van a utilizar

            JLabel derivada = new JLabel("dx/dy = "); //Texto dx/dy
            derivada.setBounds(20, 150, 50, 20); //tamaño y posición del texto
derivada
            add(derivada); //Agregación del componente a la interfaz

            cajaDerivada = new JTextField(); //Caja donde se ingresará la
derivada
            cajaDerivada.setBounds(65, 150, 150, 20);
            cajaDerivada.addActionListener(this);
            add(cajaDerivada);

            JLabel xYy = new JLabel("(X, Y)"); //Texto que está debajo de la
caja
            xYy.setBounds(120, 165, 150, 20);
            add(xYy);

            JLabel yX0 = new JLabel("Y ("); //Texo de punto inicial
            yX0.setBounds(20, 185, 20, 20);
            add(yX0);

            cajaXCero = new JTextField(); //Caja donde se ingresa X0
            cajaXCero.setBounds(35, 185, 60, 20 );
            cajaXCero.addActionListener(this);
            add(cajaXCero);

            JLabel parentesis = new JLabel(")  = "); //Terminacion del texto de
punto inicial
            parentesis.setBounds(95, 185, 20, 20);
            add(parentesis);

            JLabel xCero = new JLabel("(X0)"); //Texto que está debajo de la
caja de X0
            xCero.setBounds(53, 200, 35, 20);
            add(xCero);
```

```
        cajaYCero =new JTextField();  //Caja de Y0
        cajaYCero.setBounds(120, 185, 95, 20);
        cajaYCero.addActionListener(this);
        add(cajaYCero);

        JLabel yCero = new JLabel("(Y0)"); //Texto que está debajo de Y0
        yCero.setBounds(160, 200, 35, 20);
        add(yCero);

        JLabel yXi = new JLabel("Y ("); //Texto
        yXi.setBounds(20, 220, 20, 20);
        add(yXi);

        cajaXi = new JTextField(); //Caja donde se ingresa Xi
        cajaXi.setBounds(35, 220, 60, 20 );
        cajaXi.addActionListener(this);
        add(cajaXi);

        JLabel parentesis2 = new JLabel(")  = "); //Terminacion del texto
de condicion inicial
        parentesis2.setBounds(95, 220, 20, 20);
        add(parentesis2);

        JLabel Xi = new JLabel("(Xi)"); //Texto que está debajo de la caja
de X0
        Xi.setBounds(55, 235, 35, 20);
        add(Xi);


        cajaResultado = new JTextField();
        cajaResultado.setBounds(120, 220, 100, 20);
        cajaResultado.addActionListener(this);
        cajaResultado.setEnabled(false);
        add(cajaResultado);

        JLabel resultado = new JLabel("resultado");
        resultado.setBounds(150, 235, 100, 20);
        add(resultado);

        JLabel n2 = new JLabel("n = ");
        n2.setBounds(300, 185, 30, 20);
        add(n2);

        cajaN = new JTextField();
        cajaN.setBounds(320, 185, 100, 20);
        cajaN.addActionListener(this);
        add(cajaN);

        JLabel intervalo = new JLabel("intervalo");
        intervalo.setBounds(350, 200, 100, 20);
        add(intervalo);


        solucion = new JButton("Resolver");
        solucion.setBounds(180, 300, 100, 30);
```

```
		solucion.addActionListener(this);
		add(solucion);

	}//VentanaInicio

	public void MetodEuler(String funcion, double x0, double x1, double y,
int n, JTextField cajaResultado) {

		try {
			Expresion e = new Expresion(funcion);

			double resultado=0, f, h;
			double X[]= new double[n+1];
			double Y[]= new double[n+1];

			h=(x1 - x0)/n;
			X[0]=x0;
			Y[0]=y;

			for (int i = 0; i < n; i++) {
				f=e.evaluar(X[i])+e.evaluar(Y[i]);

				X[i+1]=X[i]+h;
				Y[i+1]=Y[i]+(h*f);

			}
			resultado=Y[n];
			cajaResultado.setText(String.valueOf(resultado));

		} catch (Exception e2) {
			// TODO Auto-generated catch block
			e2.printStackTrace();
		}

	}

	@Override
	public void actionPerformed(ActionEvent e) {
		// TODO Auto-generated method stub
		if (e.getSource()==solucion) {
			if
(!cajaDerivada.getText().equals("")&&!cajaN.getText().equals("")&&!cajaXCero.get
Text().equals("")&&!cajaXi.getText().equals("")&&!cajaYCero.getText().equals("")
) {

				String funcion=cajaDerivada.getText().toString();
				double x0=Double.parseDouble( cajaXCero.getText()),
xi=Double.parseDouble(cajaXi.getText()),

	y0=Double.parseDouble(cajaYCero.getText());
				int n=Integer.parseInt(cajaN.getText());

				MetodEuler(funcion, x0, xi, y0, n, cajaResultado);

			}else {
```

```
                        JOptionPane.showMessageDialog(rootPane, "Campos
vacios!!!!");
                }


            }
        }


}//ClassVentanaInicio


public class Prueba {

        public static void main(String[] args) {

                SwingUtilities.invokeLater(new Runnable() {

                        @Override
                        public void run() {
                                // TODO Auto-generated method stub
                                new VentanaInicio();
                        }
                });


        }//main

}//clase
```

CONCLUSIÓN

El programa salió como se esperaba. Demostrando que una ecuación diferencial se puede resolver con el programa anterior. Cabe mencionar, que una ecuación se puede resolver por medio de distintos métodos, mismos que no se pusieron debido a la falta de tiempo.

El poder resolver es de suma importancia, ya que si una ecuación es complicada, una ecuación diferencial lo es aún más, es por ello que el programa que se realizó anteriormente puede ser de gran utilidad si se requiere dar solución con el método que éste contiene.