UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

MAESTRÍA EN INFORMÁTICA

TESIS

DISEÑO DE APLICACIÓN MULTIMEDIA EN VISUAL BASIC – EXCEL PARA HALLAR SOLUCIONES NUMÉRICAS DE ECUACIONES DIFERENCIALES ORDINARIAS CON VALORES INICIALES

PRESENTADA POR:

ADOLFO CANAHUIRE CONDORI

PARA OPTAR EL GRADO ACADÉMICO DE:

MAGISTER SCIENTIAE EN INFORMÁTICA
MENCIÓN EN INFORMÁTICA EDUCATIVA

PUNO, PERÚ

2018

UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

MAESTRÍA EN INFORMÁTICA

TESIS

DISEÑO DE APLICACIÓN MULTIMEDIA EN VISUAL BASIC – EXCEL PARA HALLAR SOLUCIONES NUMÉRICAS DE ECUACIONES DIFERENCIALES ORDINARIAS CON VALORES INICIALES

PRESENTADA POR:

ADOLFO CANAHUIRE CONDORI

PARA OPTAR EL GRADO ACADÉMICO DE:
MAGISTER SCIENTIAE EN INFORMÁTICA

MENCIÓN EN INFORMÁTICA EDUCATIVA

APROBADA POR EL SIGUIENTE JURADO:

PRESIDENTE

Dr. EDGAR ELOY CARPIO VARGAS

PRIMER MIEMBRO

Dr. LEONEL COYLA IDME

SEGUNDO MIEMBRO

Mg. OLIVER AMADEO VILCA HUAYTA

ASESOR DE TESIS

M. Sc. PEDRO LEONARDO QUISPE TICONA

Puno 15 de junio de 2018

ÁREA: Diseño y Evaluación de Procesos de Software Educativo
TEMA: Diseño de Aplicaciones

DEDICATORIA

A mi madre:

Ana Condori Enriquez

A mi padre:

Gerardo Canahuire Santander

A mis hermanos:

Clodoaldo Claudio (Q.D.D.G)

Isabel

Gertrudes

Estanislao

Domitila

AGRADECIMIENTOS

- Agradecer a la Universidad Nacional del Altiplano, por permitirme lograr un grado académico de mucha importancia para mi desarrollo como profesional.
- Agradecer, de igual forma a la Escuela de Post Grado de la Universidad Nacional del Altiplano, por materializar mi anhelo de lograr un grado académico que me permitirá seguir adelante en el desempeño de mi profesión.
- Manifiesto entera gratitud a todos mis docentes del programa de Maestría en Informática Educativa, por sus enseñanzas y a todos mis condiscípulos de esta maestría por compartir momentos gratos, útiles e inolvidables.
- Expreso mis agradecimientos a los docentes del jurado: Dr. Edgar Eloy Carpio Vargas, M. Sc. Leonel Coyla Idme y el Mg. Oliver Amadeo Vilca Huayta, por sus observaciones y sugerencia constructivas; sobre todo por la comprensión y la magnanimidad manifestada por cada uno de ellos.
- A mi docente asesor M. Sc. Pedro Leonardo Quispe Ticona, por su gran ayuda y colaboración en las gestiones académicas y administrativas.
- Agradecer especialmente a todas aquellas personas dentro de mi familia y fuera de ella que me incentivaron a seguir estudios de posgrado, y siempre estuvieron dándome cotidianamente un apoyo moral y material.

ÍNDICE GENERAL

ÍNDICE DE CUADROS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

En esta tesis se expone la potencialidad del Visual Basic en Excel para la implementar una aplicación sencilla, que permita hallar soluciones aproximadas con algoritmos del análisis numérico, de unos problemas que constan de una ecuación diferencial en derivadas ordinarias provista de una condición inicial, problemas conocidos con las siglas PVI. Al disponer de la hoja de cálculo del Excel se facilita el trabajo de mostrar los resultados aproximados, ya que estos resultados se muestran en tablas que contienen los pasos iterativos de la solución de un determinado problema. Por otro lado como es de amplio conocimiento, el lenguaje de programación Visual Basic se caracteriza por su sencillez y fácil comprensión, lo que garantiza una fácil implementación de algunos algoritmos. Para completar y hacer una aplicación en VB/Excel orientado a las Matemáticas, se dispone de intérpretes matemáticos para Visual Basic, uno de ellos es el “clsMathParser”; al disponer con este interprete llamado también analizador de fórmulas matemáticas, es posible ingresar las fórmulas o expresiones matemáticas desde un control del formulario de usuario, sin tener que estar escribiendo en el código del programa la fórmula matemática para cada problema o caso que tenga que resolverse. Se ha elegido particularmente a los denominados “problema con valor inicial”, porque en la experiencia docente del suscrito, se ha podido percibir la potencialidad de una aplicación que permita realizar los cálculos reiterativos y mostrar la gráfica del resultado aproximado, todo esto sin la necesidad de tener que recurrir a una programación relativamente avanzada. En este trabajo de tesis se expone el diseño de un formulario sencillo que permita ingresar una ecuación diferencial de primer orden en derivadas ordinarias, ingresar la condición inicial y con un comando del formulario hacer posible que se obtengan una tabla y un gráfico de la solución aproximada.

PALABRAS CLAVE: Método, numérico, problema, Runge – Kutta, valor inicial.

ABSTRACT

In this thesis exposes the potential of Visual Basic in Excel to implement a simple application, which allows to find approximate solutions with algorithms of numerical analysis, of problems that consist of a differential equation in ordinary derivatives provided with an initial condition, known problems with the abbreviations PVI. Having the Excel spreadsheet facilitates the work of showing the approximate results, since these results are shown in tables that contain the iterative steps of solving a specific problem. On the other hand, as it is widely known, the Visual Basic programming language is characterized by its simplicity and easy comprehension, which guarantees an easy implementation of some algorithms. To complete and make an application in VB / Excel oriented to Mathematics, there are mathematical interpreters for Visual Basic, one of them is the "clsMathParser"; to have this interpreter also called analyzer of mathematical formulas, it is possible to enter the formulas or mathematical expressions from a control of the user form, without having to be writing in the program code the mathematical formula for each problem or case that has to be resolved. The so-called "problem with initial value" has been particularly chosen, because in the teaching experience of the undersigned, it has been possible to perceive the potentiality of an application that allows repetitive calculations and display the graph of the approximate result, all without the need of having to resort to relatively advanced programming. In this thesis the design of a simple form that allows to enter a differential equation of first order in ordinary derivatives is presented, enter the initial condition and with a command of the form make it possible to obtain a table and a graph of the approximate solution .

Keywords: Initial value, Method, numeric, problem, Runge – Kutta.

CAPÍTULO I

PROBLEMÁTICA DE INVESTIGACIÓN

1.1. PLANTEAMIENTO DE LA INVESTIGACIÓN

En la programación de los algoritmos en los métodos numéricos, la primera expectativa que se tiene es el realizar cientos, miles o hasta cientos de miles de cálculos aritméticos, para resolver problemas complejos que se presentan en Física, Matemáticas, Estadística y otros campos del conocimiento humano. Entre estos problemas se tiene a las ecuaciones diferenciales, En este trabajo de tesis se propone al software Visual Basic para Aplicaciones como una alternativa de programación para problemas matemáticos que contienen una ecuación diferencial ordinaria de primer orden, junto con una condición inicial. El problema es simplificar el trabajo de hallar la solución aproximada a una ecuación diferencial que por lo general no puede ser resuelto analíticamente ya sea por su forma compleja o porque una solución con métodos de integración puede resultar muy engorrosa por la cantidad de cálculos.

1.2. JUSTIFICACIÓN

Existen muchas aplicaciones de las ecuaciones diferenciales ordinarias y de las ecuaciones diferenciales parciales. Para el propósito de exponer y argumentar

esta justificación así como el desarrollo de este trabajo, sólo se considerarán las aplicaciones más sencillas de las ecuaciones diferenciales ordinarias de primer orden, aquellas aplicaciones que se presentan con más frecuencia en los estudios universitarios de pregrado.

Para empezar, una de las primeras aplicaciones que se puede mencionar es la denominada Ley de crecimiento exponencial que está descrita por la siguiente ecuación diferencial ordinaria de primer orden:

$$\frac{dy}{dt} = \alpha y$$

Cuya solución es la función

$$y = ke^{\alpha t}$$

Donde *k* representa a una constante, denominada constante de integración.

Esta ley de crecimiento exponencial, con modificaciones del contexto, tiene un gran número de aplicaciones, tales como el crecimiento de una célula, la propagación y control de una enfermedad, la absorción de algunos productos químicos, el crecimiento y el decrecimiento de poblaciones de bacterias, etc.

Por otro lado, como un segundo caso, se puede citar a la Ley de Newton sobre el enfriamiento de un cuerpo en el transcurso del tiempo, esta ley dice que la razón de cambio de la disminución de temperatura de un cuerpo con respecto al tiempo, es proporcional a la diferencia de temperatura *T* del cuerpo y la temperatura T_m del medio ambiente. Esta ley se expresa mediante la ecuación diferencial:

$$\frac{dT}{dt} = -k(T - T_m)$$

Donde *k* es una constante de proporcionalidad positiva.

Otro caso que se puede mencionar en la Física, es la conocida Ley de la desintegración radioactiva, que refiriéndose al número de núcleos atómicos *dN*, dice que el número de núcleos atómicos que se desintegran en un corto intervalo de tiempo *dt*, es proporcional al número de núcleos presentes *N(t)*. Esta Ley se expresa mediante la siguiente ecuación diferencial.

$$\frac{dN(t)}{dt} = -kN(t)$$

Donde *k* también es una constante de proporcionalidad.

Y para seguir citando ejemplos concretos de ecuaciones diferenciales presentes en leyes de la Física o Química, en la Tabla PT7.1 que se halla en la página 700 de (Chapra C, 2010) se mencionan de forma directa y concisa los siguientes casos:

La segunda ley del movimiento de Newton.

$$\frac{dv}{dt} = \frac{F}{m}$$

Donde, *v* es la velocidad, *F* la fuerza y *m* la masa.

La ley del calor de Fourier.

$$q = -k' \frac{dT}{dx}$$

Donde, *q* es la densidad de la corriente de energía, *k'* es una constante propia del material denominada conductividad térmica, *T* es la temperatura.

Ley de difusión de Fick.

$$J = -D\frac{dC}{dx}$$

Donde, *J* representa a la densidad de la corriente de partículas, *D* es el coeficiente de difusión, y *C* es la concentración molar del material.

Ley de la caída de tensión a través de un inductor.

$$\Delta V_l = L\frac{di}{dt}$$

Donde, ΔV_l es la caída de voltaje, *L* la inductancia e *i* la corriente.

Las leyes citadas anteriormente, se expresan mediante ecuaciones diferenciales ordinarias de primer orden, a las que naturalmente se les tiene que proveer de alguna condición inicial. Pero además de estas leyes, se pueden mencionar otros casos o modelos matemáticos que involucran a las ecuaciones diferenciales, como pueden citarse al modelo matemático que describe las oscilaciones de un péndulo, a las leyes de Kirchhoff en circuitos eléctricos, la deflexión de vigas, a los problemas de cables colgantes, a las ecuaciones de flujo de calor, así como el análisis de las trayectorias ortogonales a una familia de curvas en el plano, problemas de concentración de sal en un tanque, problemas de caída de cuerpos con resistencia del aire, el movimiento vibratorio de sistemas mecánicos. En algunos de estos problemas, las ecuaciones diferenciales son sencillas de resolver mediante métodos analíticos, sin embargo se presentan casos en los cuales no es posible aplicar método analítico alguno, o puede suceder que la solución analítica es complicadísima, o en otras ocasiones se debe trabajar con datos discretos obtenidos con mediciones en laboratorio; por estas razones de limitación en las técnicas de cálculo y otras limitaciones o consideraciones, se justifica aplicar métodos numéricos, estos métodos implican el cálculo repetitivo

de grandes cantidades de sumas, multiplicaciones, etc. Para facilitar estos cálculos repetitivos (iterativos) se propone el desarrollo de programas para PC, y precisamente en este trabajo se plantea hacer el prototipo de una aplicación que permita resolver ecuaciones diferenciales ordinarias de primer orden provistas de una condición inicial aplicando los métodos numéricos de Runge – Kutta.

Así mismo se debe mencionar que con la experiencia en la docencia universitaria en el área de Matemáticas, es innegable reconocer que la cantidad de los problemas que se estudian, analizan teóricamente y resuelven analíticamente son pocos, comparables con la gran cantidad de problemas de la cotidianidad que no siempre están enmarcados o cumplen con las condiciones establecidas en la teoría. Para superar estas limitaciones y poder resolver más problemas que se presentan en problemas concretos, se justifica el estudio de los métodos asistidos por computadora, y que como está registrado en la Historia, estos métodos han empezado a tener considerable desarrollo después de la segunda guerra mundial, justamente con el desarrollo de las primeras computadoras, y ahora que estas nuevas tecnologías se han hecho cada vez más versátiles y de fácil acceso, se apertura muchas posibilidades para la investigación científica.

1.3. OBJETIVOS

1.3.1. Objetivo general

Desarrollar una aplicación multimedia con el editor de código Visual Basic integrado al Microsoft Excel, que puede ser utilizado como un recurso de aprendizaje para resolver numéricamente problemas de valor inicial con ecuaciones diferenciales ordinarias de primer orden.

1.3.2. Objetivos específicos

a) Desarrollar un prototipo de aplicación con una interfaz amigable en VBA/Excel, con el propósito de resolver problemas con valores iniciales, interfaz que pueda influir en lograr las competencias en el aprendizaje de los métodos numéricos aplicados a la solución de ecuaciones diferenciales ordinarias con condiciones iniciales.

b) Aplicar el analizador de ecuaciones CLSMATHPARSER en su versión disponible para desarrollar la aplicación multimedia con formularios de VBA/Excel, para resolver problemas con valores iniciales.

CAPÍTULO II

MARCO TEÓRICO

2.1. INVESTIGACIÓN

Las ecuaciones diferenciales están relacionadas con el cálculo diferencial, por lo tanto estos temas están asociados a Isaac Newton y Gottfried Leibniz.

Newton en su obra “The Method of Fluxioms and Infinite Series with its Application to the Geometry of Curve – Lines”, publicada en 1736, considera varios ejemplos de ecuaciones diferenciales. En 1768, Leonhardi Euleri, en su obra “INSTITUTIONES CALCULI INTEGRALIS” Capítulo VII (De integratione aequation differentialum per approximationem) página 424, describe un método numérico sencillo de un paso, que con el tiempo, habría de llamarse “Método de Euler”. Los métodos numéricos de Runge – Kutta que se estudian actualmente, se originan a finales del siglo XIX e inicios del siglo XX. Primero, el matemático, físico y espectroscopista alemán Carl David Tolmé Runge (1856 – 1927), en el tercer capítulo de su libro “GRAPHICAL METHODS”, publicado en New York por la COLUMBIA UNIVERSITY en el año 1912, expone la solución de una ecuación diferencial de primer orden mediante métodos gráficos, en este libro aparecen las fórmulas recursivas del método que ahora se conoce con el nombre de “método de Runge – Kutta de cuarto orden”. Luego el físico matemático alemán

de origen polaco, Wilhelm Martin Kutta (1867 – 1944), en su tesis doctoral contribuye con mejorar los métodos de Runge, por esta razón, los nombres de ambos científicos están ligados a estos métodos. En la década de 1940 a 1950 aparecen los primeros ordenadores, desarrollándose un interés en la aplicación computarizada de los métodos de los métodos numéricos, haciendo factible la aparición de una nueva disciplina autónoma llamado “Análisis Numérico”. Por otra parte, los métodos numéricos multipaso en ecuaciones diferenciales, están ligados históricamente a los nombres de Adams y Bashforth (1883) y Moulton (1926).

Figura 1. Carl David Tolmé Runge (1856–1927)

(fuente: O'Connor & Robertson, 2016)

Figura 2. Wilhelm Martin Kutta (1867 – 1944)

(fuente: O'Connor & Robertson, 2016).

2.2. ANTECEDENTES

1. Estuardo Javier Gan Rodríguez, en la Universidad Autónoma de México (UNAM). (Gan Rodriguez, 2014), desarrolla una Aplicación en VBA/Excel, denominado “RUNGE – KUTTA SOLUCIONADOR”. En la información general de esta aplicación se lee: “El programa está diseñado para integrar numéricamente ecuaciones diferenciales ordinarias de primer orden con condiciones iniciales.

2. Proyecto de investigación del Departamento de Ingeniería de Comunicaciones en la Facultad de Ingeniería Eléctrica y Electrónica de la Universidad de TUN HUSSEIN de Malasia (Tay, 2015). Título del proyecto: “THE FOURTH ORDER RUNGE – KUTTA SPREADSHEET CALCULATOR

USING VBA PROGRAMING FOR ORDINARY DIFFERENTIAL EQUATIONS"; traducido al español es: "Calculadora de hoja de cálculo para programar ecuaciones diferenciales ordinarias de cuarto orden con el método de Runge – Kutta, usando la programación VBA".

3. Raafat K. Oubida. De University of Babylon, Babil, Iraq, 2016. Plantea una herramienta informática con una interfaz interactiva, para la enseñanza de los métodos numéricos, específicamente para la integración numérica, la solución aproximada de ecuaciones diferenciales ordinarias con el método de Euler, y el método de Runge – Kutta. Keywords: *CSNMS, Teaching tools, Numerical Analysis, Simulation, Integration, Runge – Kutta, Euler*. Información sobre este trabajo se halla en Paper titulado: "COMPUTERIZED SYSTEM FOR NUMERICAL METHODS SIMULATION USING VISUAL BASIC PROGRAMING LANGUAGE".

2.3. MARCO CONCEPTUAL

2.3.1. Problema con valor inicial PVI

En la primera parte del capítulo 7 de (Stoer, 1993), se halla una explicación sugerente sobre un problema con valor inicial. Empieza diciendo que muchos problemas de la matemática aplicada conducen al estudio de las ecuaciones diferenciales ordinarias, y que un caso sencillo y frecuente trata de la determinación de una función diferenciable y = y(t) que tenga que satisfacer a una ecuación diferencial de la forma.

$$\frac{dy}{dt} = f\big(t, y(t)\big)$$

En general existen infinitas funciones *y = y(t)* que satisfacen la ecuación diferencial anterior, sin embargo considerando algún requisito adicional o condición complementaria, se puede particularizar una solución (función) del conjunto de todas las soluciones. Esta particularización se hace mediante una condición inicial: $t = t_0$ y $y = y_0$ la misma que se escribe como $y(t_0) = y_0$.

Así, por ejemplo, el conjunto de funciones $y = c\,e^t - t - 1$, satisface la ecuación diferencial

$$\frac{dy}{dt} = t + y$$

Cualquiera que sea el valor de la constante *c*, llamada constante de integración. La familia de curvas que se muestra en la Figura 3, son los gráficos de las funciones $y = c\,e^t - t - 1$ para algunos valores de la constante *c*. Los gráficos de estas funciones se llaman curvas integrales de la ecuación diferencial, y como se puede deducir, existen infinitas curvas integrales; cuando se tiene particular interés en una de estas curvas, se debe dar un requisito u condición mediante una denominada condición inicial $y(t_0) = y_0$.

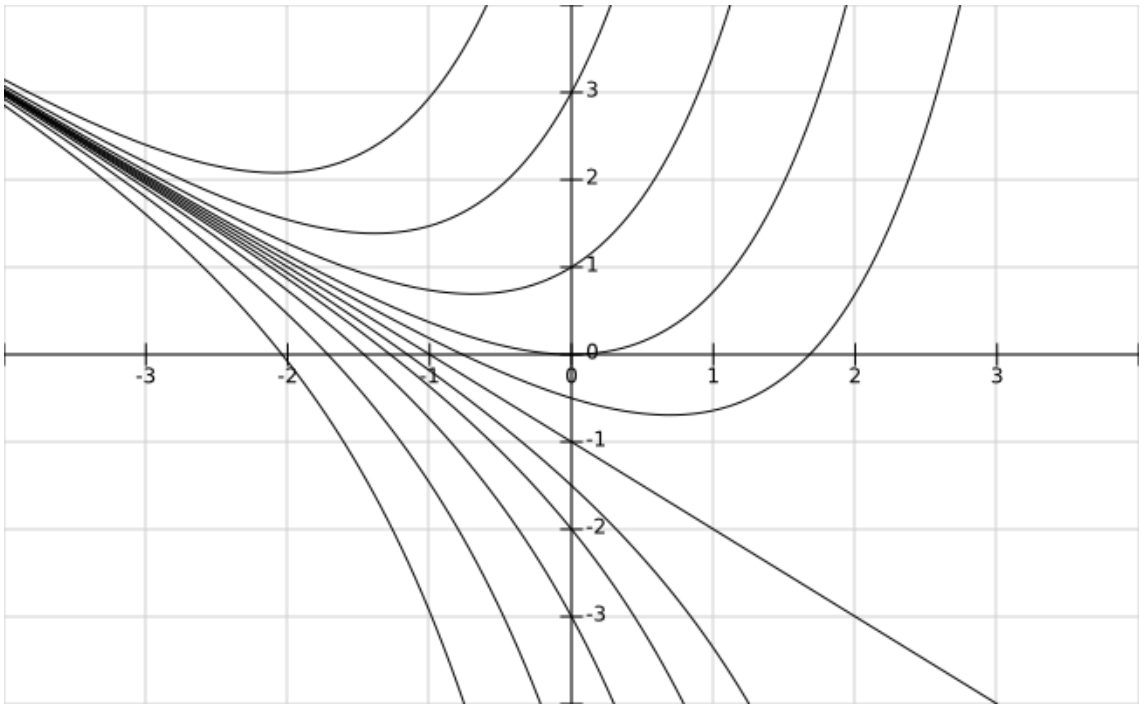


Figura 3. Curvas integrales de la ecuación diferencial $y' = t + y$.

Si la condición inicial fuera $y(0) = 1$, la curva sería aquella que pase por el punto *(0,1)*, cuya ecuación es $y = 2e^t - t - 1$ y en la Figura 3 esta curva es la tercera contando desde la parte superior.

Para el estudio de la teoría elemental de problemas con valor inicial en el capítulo 5 de (Burden & Faires, 2010), se define dos conceptos pertinentes como son función lipschitziana y conjunto convexo en el plano. Se enuncia una versión del teorema fundamental de la existencia y unicidad para ecuaciones diferenciales ordinarias de primer orden. El enunciado del teorema es:

Si $f(t,y)$ es una función continua en $D = \{(t,y) : a \leq t \leq b; -\infty < y < \infty\}$. Si *f* cumple con la condición de Lipschitz en *D* en la segunda variable, entonces el problema con valor inicial.

$$y' = f(t, y); \quad a \leq t \leq b; \quad y(a) = \alpha$$

Tiene una única solución $y(t)$ para $a \leq t \leq b$.

La solución $y(t)$ debe satisfacer las siguientes tres condiciones: Primero $y(a) = \alpha$; segundo, para cualquier t que pertenezca al intervalo cerrado $[a, b]$, el punto $(t, y(t))$ debe pertenecer al conjunto *D*, y tercero, debe satisfacer la ecuación diferencial dada, esto es $y'(t) = f(t, y(t))$.

En la página 6 de (Naón, 2014), se define a un "problema bien planteado".

Un problema que involucra ecuaciones diferenciales, se llama bien planteado, si:

1. La ecuación diferencial tiene solución.
2. La solución es única.
3. La solución depende de modo continuo de las condiciones complementarias (condiciones iniciales) y de todos los parámetros del problema.

En el Teorema 5.6 del capítulo 5 en (Burden, 2010), se dice que cuando la función $f(t, y)$ del problema con valor inicial es continua y lipschitziana en el conjunto *D*, entonces el problema es un problema bien planteado. Para ilustrar este concepto con un ejemplo, se plantea el PVI:

$$\frac{dy}{dt} = t + y; \quad 0 \leq t \leq 2; \quad y(0) = 1$$

Este PVI es un problema bien planteado. La función $f(t, y) = t + y$ es continua y lipschitziana en $D = \{(t, y) : 0 \leq t \leq 2, -\infty < y < \infty\}$. La continuidad de $f(t, y)$ es evidente, y $f(t, y)$ es lipschitziana con respecto a

su segunda variable, tiene una constante de Lipschitz L igual al número 1; pues:

$$\left|\frac{\partial}{\partial y}(t+y)\right| = |1| = 1 = L$$

La ecuación diferencial de este problema bien planteado puede ser resuelta con un método analítico, pues es una ecuación lineal de la forma $y' + p(t)y = q(t)$. Sin embargo,

$$\frac{dy}{dt} = t + \sin y\,; \quad 0 \le t \le 1; \quad y(0) = 1$$

Es un problema bien planteado en $D = \{(t,y) : 0 \le t \le 1, -\infty < y < \infty\}$; pero su ecuación diferencial no puede ser resuelta con método analítico alguno. Para este caso y otros similares se debe recurrir a algún método numérico. Por otro lado,

$$\frac{dy}{dt} = \sqrt{y}; \quad 0 \le t \le 1; \quad y(0) = 1$$

No es un problema bien planteado, pues la función $f(t,y)$ no cumple con la condición de Lipschitz en $D = \{(t,y) : 0 \le t \le 1, -\infty < y < \infty\}$. Se puede ver que la ecuación diferencial de este problema, admite dos soluciones continuas que dependen de la condición inicial $y(0) = 1$. Estas soluciones son:

$$y = \frac{1}{4}(t+2)^2$$

$$y = \frac{1}{4}(t-2)^2$$

Existen ecuaciones diferenciales ordinarias de primer orden que pueden ser de variables separables, homogéneas, lineales de primer orden, de Bernoulli, pero que esto no implica que sean integrables analíticamente; por ejemplo, en el problema

$$\frac{dy}{dt} = y^2 - ty - t; \quad 0 \leq t \leq 1; \quad y(0) = 1$$

Su ecuación diferencial es una ecuación de Riccati, pues está presentada en la forma:

$$\frac{dy}{dt} = P(t) + Q(t)y + R(t)t^2$$

La solución general de esta ecuación puede ser obtenida mediante la sustitución $y = y_1 + z$, donde y_1 es una solución particular, con esta sustitución la ecuación de Riccati se transforma en una ecuación diferencial de Bernoulli. En el problema planteado líneas arriba una solución particular es $y = t + 1$, y la ecuación de Bernoulli correspondiente es:

$$\frac{dz}{dt} + (t+2)z = z^2$$

Esta ecuación no puede ser integrada analíticamente, sin embargo la función es continua y lipschitziana en $D = \{(t,y) : 0 \leq t \leq 1, -2 \leq y \leq 2\}$, por lo tanto este problema es un problema bien planteado, esto implica que tiene una solución, y esta solución tendrá que ser encontrada con un método numérico.

Otro ejemplo que justifica la aplicación de métodos numéricos, es el caso como el que se presenta en el siguiente problema:

Calcular $y(1)$, sabiendo que:

$$\frac{dy}{dt} = \frac{t - y + 2}{2t + y - 5}; \quad -1 \leq t \leq 1; \quad y(0) = 0$$

Si $D = \{(t, y) : -1 \leq t \leq 1, -2 \leq y \leq 2\}$, el problema anterior, es un problema bien planteado en el rectángulo *D*. Por lo tanto tiene una solución $y(t)$. Sin embargo, aunque la ecuación diferencial de este problema puede ser integrado con métodos analíticos, pues esta se reduce a una ecuación diferencial homogénea con una traslación de los ejes coordenados, pero la solución obtenida se presenta como una expresión algebraica complicada; siendo difícil despejar la función $y(t)$.

El problema:

$$\frac{dy}{dt} = \frac{1}{t + y}; \quad t \geq 1; \quad y(1) = 2$$

Es un problema bien planteado en $D = \{(t, y) : t \geq 1, y \geq 1\}$, es más la solución de la ecuación diferencial de este problema puede ser obtenida analíticamente con la sustitución "$u = t + y$". Sin embargo la dificultad es calcular, por ejemplo $y(2)$, pues esto implicaría despejar la variable *y* de la ecuación: $y + \ln(2 + y) = 2 + \ln 3$ como se puede ver, de esta ecuación no es posible despejar la variable "*y*". Este problema es también una justificación o razón de recurrir a algún método numérico para calcular el valor de $y(2)$.

En esta parte del marco conceptual, se ha expuesto en forma resumida la teoría de un problema con valor inicial, las funciones lipschitzianas, el teorema de la existencia y la unicidad de un problema bien planteado. Se ha

justificado con ejemplos, la pertinencia de la aplicación de métodos numéricos. En los siguientes párrafos se citarán, en forma escueta, las fórmulas de los métodos numéricos de Runge - Kutta que se aplicarán para resolver problemas con valor inicial, estos métodos no son únicos, existen otros. Por otro lado los problemas con valor inicial podrían contener ecuaciones diferenciales ordinarias de segundo orden, o contener sistemas de ecuaciones diferenciales, y se pueden presentar otros casos que no son considerados en este trabajo.

2.3.2. Métodos numéricos para resolver un PVI

Los métodos numéricos aplicables a un PVI que se exponen en (Chapra C, 2010) y en (Burden, 2010), y en la mayoría de textos que tratan este tema, son los siguientes:

1. Método de Euler o método de Runge – Kutta de primer orden (RK1).
2. Método de Euler modificado.
3. Método de Heun o método de Runge – Kutta de segundo orden (RK2).
4. Método de Nystrom.
5. Método de Runge – Kutta de tercer orden (RK3).
6. Método de Runge – Kutta de cuarto orden (RK4).

Si bien es cierto, no todos los nombres de los métodos mencionados se refieren a Runge – Kutta, sin embargo, estos métodos y otros están fundamentados y tienen como base a la fórmula de Euler y a los métodos de Runge – Kutta. En las páginas que vienen a continuación, se expone una

relación resumida de las fórmulas iterativas de estos 6 métodos. Las fórmulas iterativas de estos métodos generan sucesiones de puntos $\{t_n\}$ y $\{Y_n\}$, las mismas que dependen de la longitud de paso *h* y de otras consideraciones de contexto.

2.3.3 Método de Runge – Kutta de primer orden

Este método se le denomina también con el nombre de método de Euler, porque fue descrito por Leonard Euler (Simmons, G. & Krantz, 2007). Este método es fácil de programar, pero uno de los inconvenientes es que para obtener aproximaciones razonablemente aceptables, es necesario tomar pasos "*h*" muy pequeños, pero no tan pequeños, porque se corre el riesgo con los errores de redondeo, inevitables en todo cálculo numérico.

Dado un problema con valor inicial. El método de Euler puede ser deducido desarrollando la solución de la ecuación diferencial del PVI en serie de Taylor, alrededor del punto $t_i \in [a, b]$ donde $i = 0, 1, \dots, n - 1$, truncando la serie y denotado por $Y(t)$ el valor aproximado de $y(t)$, y la fórmula recursiva de este método es:

$$Y_{i+1} = Y_i + hf(t_i, Y_i)$$
$$Y_0 = y(t_0)$$

Si *n* es el número de sub intervalos en los que se divide el intervalo $I = [a, b]$, la longitud de paso *h* es calculada con la fórmula:

$$h = \frac{b - a}{n}$$

Además $t_{i+1} = t_i + h$ donde $i = 0, 1, \dots, n$.

2.3.4. Método de Euler modificado

$$Y_{i+1} = Y_i + hf\left(t_i + \frac{h}{2}, Y_i + \frac{h}{2}f(t_i, Y_i)\right)$$
$$Y_0 = y(t_0)$$

2.3.5. Método de Runge – Kutta de segundo orden

Conocido también con el nombre de método de Heun. Asumiendo las consideraciones previas explicadas en el método de Euler, y tomando la fórmula de la cuadratura numérica correspondiente a la regla del trapecio, se llega a obtener la siguiente fórmula iterativa:

$$Y_{i+1} = Y_i + \frac{h}{2}\left[f(t_i, Y_i) + f\left(t_i + h, Y_i + hf(t_i, Y_i)\right)\right]$$
$$Y_0 = y(t_0)$$

Para efectos de facilitar la programación, puede ser conveniente reescribir la primera fórmula de este método como:

$$Y_{i+1} = Y_i + \frac{1}{2}(k_1 + k_2)$$

Donde:

$$k_1 = hf(t_i, Y_i)$$
$$k_2 = hf(t_i + h, Y_i + k_1)$$

Con este método se obtienen aproximaciones más cercanas que las que se obtienen con el método de Euler.

2.3.6. Método de Nystrom

Para obtener este método, se toma en cuenta una sustitución de la derivada $y'(t)$ por:

$$\frac{y(t+h)-y(t-h)}{2h}$$

De esta forma,

$$\frac{Y_{i+1}-Y_{i-1}}{2h}=f(t_i,Y_i)$$

Y la fórmula recursiva que se obtiene es:

$$Y_{i+1}=Y_i+2hf(t_i,Y_i)$$

Este método no se inicia por sí mismo, esta dificultad se resuelve, en particular el valor de Y_i puede obtenerse con el método de Heun. Este método es un método multipaso.

2.3.7. Método de Runge – Kutta de tercer orden

$$Y_{i+1}=Y_i+\frac{1}{6}(k_1+4k_2+k_3)$$

Donde:

$$k_1=hf(t_i,Y_i)$$
$$k_2=hf\left(t_i+\frac{h}{2},Y_i+\frac{k_1}{2}\right)$$
$$k_3=hf(t_i+h,Y_i-k_1+2k_2)$$

2.3.8. Método de Runge – Kutta de cuarto orden

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

Donde:

$$k_1 = hf(t_i, Y_i)$$
$$k_2 = hf\left(t_i + \frac{h}{2}, Y_i + \frac{k_1}{2}\right)$$
$$k_3 = hf\left(t_i + \frac{h}{2}, Y_i + \frac{k_2}{2}\right)$$
$$k_4 = hf(t_i + h, Y_i + k_3)$$

Los métodos que se han expuesto, excepto el método de Nystrom, están comprendidas entre los métodos de un paso, porque cuando se debe calcular un valor de Y_{i+1}, sólo se necesita el valor de la última aproximación Y_i. En los métodos multipaso se necesitan los valores de los puntos previos.

Las fórmulas recursivas de estos métodos numéricos, deberán implementarse mediante funciones en código del lenguaje Visual Basic para aplicaciones.

2.3.9. Visual Basic para aplicaciones

Del texto (Harvey, 2010) y de (Zanini, 2013), se puede resumir la siguiente información: Visual Basic es un lenguaje de programación de alto nivel que proviene del BASIC. La primera versión del Visual Basic fue presentada en el año 1991. Su interfaz gráfica tenía el propósito de simplificar la programación utilizando un entorno de trabajo sencillo, que permitiera crear interfaces gráficas facilitando así la programación de aplicaciones. La creación de interfaces gráficas para diferentes utilidades es una de las principales funciones del Visual Basic y es por esta razón que es muy utilizado en espacios profesionales donde se requieren soportes gráficos para mayor organización de los contenidos y materiales. La programación gráfica se puede llevar a cabo directamente ya que el Visual Basic no requerirá de los usuarios la escritura de los códigos de programación. El

Visual Basic trabaja a partir de lenguajes RAD, en inglés “Rapid Application Development”, o desarrollo rápido de aplicaciones específicas para cada necesidad y función. Al mismo tiempo, el Visual Basic, gracias a su simple lenguaje, es perfectamente adaptable a las plataformas de los sistemas Windows y es fácilmente transformable a otros lenguajes más complejos.

Para acceder a Visual Basic en Excel, se puede pulsar ALT + F11, o hacer clic en la pestaña “Desarrollador” de la cinta de menú y nuevamente clic en el botón “Visual Basic” de la barra de herramientas, de cualquier forma, se abre una ventana de Editor de Visual Basic (Integrated Development Environment IDE), En la Figura 4, se puede observar esta ventana, la misma que contiene una barra de menú, una barra con botones para las herramientas, un explorador de proyectos (Proyecto – VBAProject), una ventana de propiedades (Propiedades - ThisWorkbook), una ventana para escribir el código y en la parte inferior dos ventanas, una de ejecución inmediata llamando “Inmediato”, y la otra denominada “Locales”. Con VBA/Excel se pueden implementar formularios (UserForm) personalizados de usuario, con controles ActiveX de formulario, los controles que se pueden insertar son: etiqueta, cuadro de texto, cuadro de grupo, botón, casilla de verificación, botón de opción, cuadro de lista, cuadro combinado, barra de desplazamiento, control de número, botón para insertar imagen y otros. Ver la Figura 5.

Para cerrar el editor de VBA se puede pulsar ALT + Q, y para guardar un proyecto en Excel que contenga macros debe guardarse el archivo con la extensión “xlsm”.

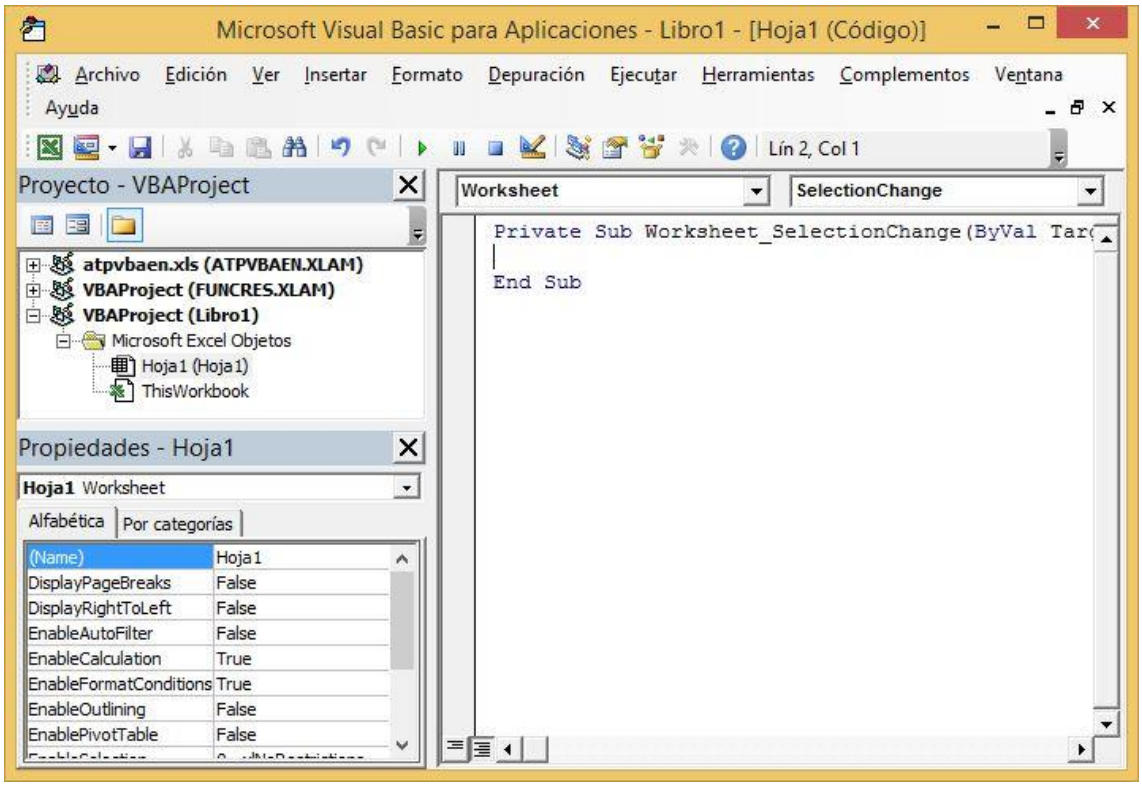


Figura 4. Ventana de editor del Visual Basic para aplicaciones.

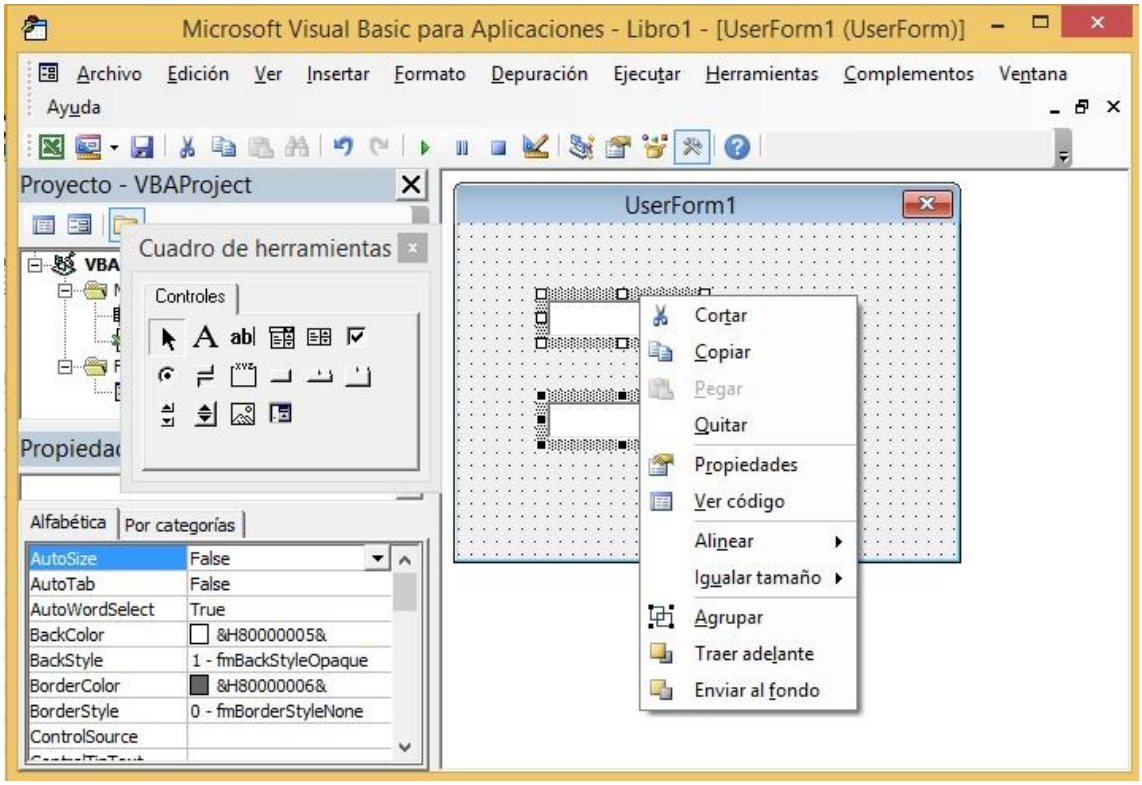


Figura 5. Formularios en VBA/Excel.

2.3.10. Objetos en Visual Basic para Excel

En BVA/Excel un objeto es una unidad de datos que representa a un determinado elemento de una planilla del Excel. Los principales objetos considerados en VBA/Excel, entre otros, son: los rangos, las hojas, los libros, y la aplicación Excel misma. Un rango puede estar constituido por sólo una celda, un conjunto de celdas, una fila o una columna de celdas. Una hoja de trabajo contiene celdas y rangos. Un libro contiene hojas. Estos objetos guardan una estructura jerárquica, como la que se muestra en la Figura 6.

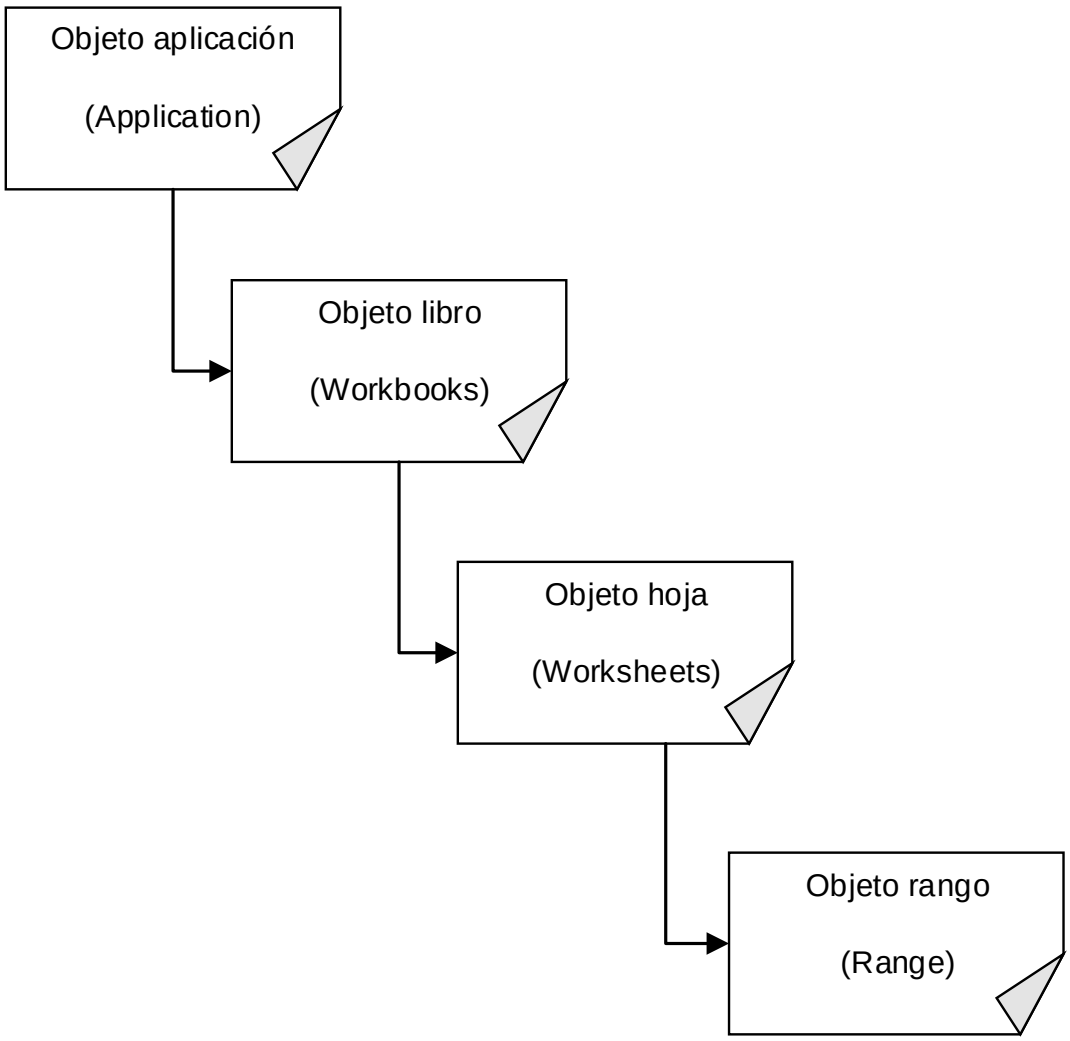


Figura 6. Jerarquía de los objetos en VBA/Excel.

Los objetos en VBA/Excel pueden ser controlados por VBA recurriendo a las propiedades y métodos de cada objeto. Estas propiedades y métodos son tantos que sería imposible memorizarlos, para salvar este inconveniente en el editor de VBA/Excel, se dispone de una ayuda que muestra todos los métodos y las propiedades de un objeto cuando se pone un punto después de escribir el nombre del objeto. En la Figura 7 y en la Figura 8, se puede ver la forma desplegada de estas ayudas contextuales. En estas ayudas, las propiedades están representadas con íconos de color blanco, y los métodos tienen un ícono de color verde (Birnbaun, 2005).

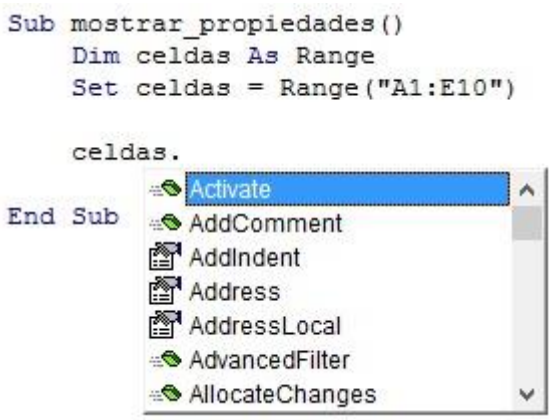


Figura 7. Propiedades y métodos de un Objeto tipo Range llamado celdas.

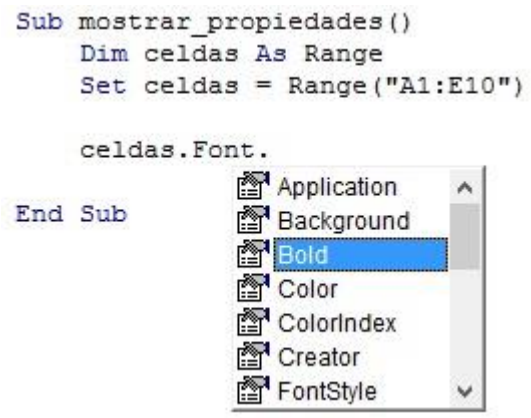


Figura 8. Propiedades del objeto Font

que corresponde al objeto celdas.

Para mayor información, el IDE del VBA/Excel dispone de un examinador de objetos, para acceder a este examinador, se puede seguir cualquiera de los siguientes tres pasos: Uno, seleccionando del menú “Ver” la opción “Examinador de objetos”; segundo, presionando la tecla F2; tercero, haciendo Clic en el botón “Examinador de objetos” de la barra de herramientas estándar. Cuando se abre la ventana del examinador de objetos, se pueden los objetos disponibles en VBA, junto con sus propiedades, métodos y eventos correspondientes.

Para la programación en VBA/Excel se usan fundamentalmente dos tipos de procedimientos. Uno llamado procedimiento “Sub” y otro llamado “Function” cualquiera de estos procedimientos se pueden incluir en un módulo del proyecto.

Un procedimiento “Sub” está formado por una serie de instrucciones colocadas y ejecutadas unas después de otras, pasando a la línea siguiente

cada vez, estas instrucciones tienen el propósito de realizar alguna tarea. La sintaxis es:

```
Public Sub < nombre de la macro > ( )
    < instrucciones de la macro >
End Sub
```

End Sub cierra el procedimiento después de la última instrucción. Cabe mencionar que con “Exit Sub” se puede salir del procedimiento en el transcurso de la ejecución.

Un procedimiento “Function” devuelve un valor, y para definirla es necesario especificar los argumentos de la Function. Su sintaxis es:

```
Public Function < nombre de la función >(argumentos)
    < nombre de la función > = < valor / expresión >
End Function
```

Los procedimientos Sub y las Funciones pueden ser públicas o privadas, si estas fueran privadas, se empieza el código escribiendo la palabra “Private”.

Un evento en VBA/Excel es la acción que puede realizar un objeto, como puede ser al abrir un libro, al pasar de una hoja a otra, al eliminar (delete) una hoja, etc. Los eventos entonces se pueden clasificar como: Eventos de libro, de hoja, de aplicación, de gráfico, de formulario, etc. Así mismo existen eventos no asociados a objetos como son: OnTime y OnKey. Por ejemplo los eventos de un libro, son entre otros, Open, Actívate, BeforeClose, BeforePrint, BeforeSave, etc. El evento por defecto de un libro es Open. Algunos eventos de una hoja son: Activate, Change, SelectionChange, Calculate, BeforeDelete, etc. A los eventos que se asocian a un objeto

determinado, se les puede hallar en un Combo Box en el IDE de VBA/Excel como el mostrado en la Figura 9.

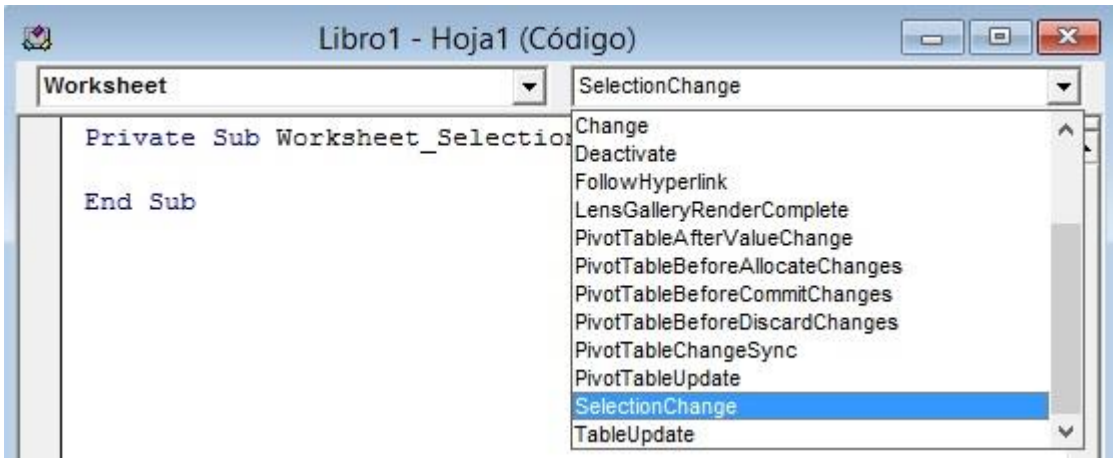


Figura 9. Eventos disponibles para el objeto WorkSheet.

Por ejemplo si un libro tiene varias hojas; con el evento para la hoja 1 cuyo código es:

```
Private Sub Worksheet_Deactivate()
    MsgBox "Usted Salió de la Hoja1"
End Sub
```

La acción (evento) que se desencadena, es que cada vez que el usuario pase de la Hoja 1 a las otras hojas del libro, se le notificará mediante un cuadro de dialogo, que se ha salido de la hoja 1.

Los cuadros de diálogo sirven para interactuar con los usuarios durante la ejecución de un programa. A este respecto, en VBA/Excel existen las funciones “InputBox” y “MsgBox”. El primero es primordialmente para ingresar información, mientras que el segundo es para mostrar mensajes breves, advertencias y/o alertas.

2.3.11. Variables

Las variables sirven para almacenar o manipular información, las variables se declaran en los procedimientos contenidos en los módulos o formularios de un proyecto. El lenguaje VBA puede crear automáticamente variables sin declaración previa, dándole el tipo relacionado con su utilización, se trata de una declaración implícita. Sin embargo, se aconseja declarar las variables antes de su utilización.

La duración de una variable declarada con la instrucción Dim se limita a la ejecución del procedimiento en el que se halla. Esto significa que esta variable se reinicializa en cada ejecución. Si se quiere mantener el valor asignado se debe usar la instrucción Static en la declaración, entonces su duración será durante la ejecución de la aplicación.

El ámbito de una variable, depende de dónde y cómo se le declara. Una variable declarada al interior de un procedimiento sólo puede ser utilizada en ese procedimiento. En cambio, si la variable es declarada en la zona (General) / (Declaraciones) del módulo bajo la línea de código Option Explicit, esta variable es accesible por todos los procedimientos del módulo. Para que una variable sea accesible a todos los módulos de un proyecto, se debe usar la instrucción Public en lugar de la instrucción Dim. Si por el contrario se desea que la variable no sea accesible por otros módulos, se le debe declara como Private.

Las constantes se parecen a las variables, pero no pueden ser modificadas, pues la asignación de un valor a una constante se hace en el momento de su declaración y no durante la ejecución de la aplicación.

2.3.12. Tipos de datos

Los tipos de datos que se usan con mayor frecuencia en la programación VBA/Excel son:

1. Byte, entero de un byte en un intervalo de 0 a 255.
2. Integer, para enteros de 32 bits (4 bytes) con signo que va de – 32768 a 32767, proporciona un rendimiento óptimo en un procesador de 32 bits.
3. Long, entero largo de 4 bytes, de – 2147483648 a 2147483647.
4. Currency, entero de 8 bytes.
5. Single, para números de punto flotante de precisión sencilla de 32 bits (4 bytes) con signo que van de -3,4028235E+38 a -1,401298E-45.
6. Double, contiene números de punto flotante de doble precisión de 64 bits (8 bytes) que van de un valor de –1,79769313486231570E+308 a –4,94065645841246544E-324 para números negativos y de 4,94065645841246544E-324 a 1,79769313486231570E+308 para números positivos. Los números de doble precisión (double) almacenan aproximaciones de números reales..
7. Decimal, números reales de 14 bytes.
8. Bolean, lógico de 2 bytes, puede tomar dos valores, verdadero o falso. (True or False).
9. Date, para las fechas de 8 bytes.
10. String, para datos de texto, de 10 bytes.

2.3.13. Estructuras de control

El Visual Basic, como otros lenguajes de programación, cuenta con estructuras de control, que permiten tomar decisiones o realizar operaciones repetitivas (Zanini, 2013). Así estas estructuras de control pueden ser estructuras de decisión y estructuras de bucle, sin embargo en VBA/Excel se cuenta con otras estructuras como aquella que pueden asignar propiedades y métodos a un objeto.

Excel contiene estructuras de decisión y de bucle, de forma sucinta estas estructuras son:

Estructura condicional If … Else.

```
If < condición >  Then
          < instrucciones 1 >
ElseIf < condición alterna > Then
          < instrucciones 2 >
Else
          < instrucciones 3 >
End If
```

Estructura de decisión Select Case.

```
Select Case < variable >
Case opción 1
        < instrucciones 1 >
Case opción 2
        < instrucciones 2 >
Case Else
```

```
            < instrucciones 3 >
    End Select
```

Estructura Do … Loop.

```
    Do
            < instrucciones >
    Loop < condición >
```

Estructura While … Wend.

```
    While < condición >
            < instrucciones >
    Wend
```

Estructura For … Next.

```
    For < variable = valor inicial > To < valor final >
            < instrucciones >
    Next < contador >
```

Estructura For … Each.

```
    For Each < valor de la variable>
        < instrucciones >
    Next
```

2.3.14. Estructura de bloque

La estructura “Width” sirve para asignar propiedades correspondientes a un objeto, con esta instrucción se asignan varias propiedades. A continuación se muestra el código de un procedimiento en el que se usa una estructura Width anidada dentro de otra.

```
With < objeto >
     .propiedad_1 = valor_1
     .propiedad_2 = valor_2
     ⋮
     .propiedad_n = valor_n
End With
```

CAPÍTULO III

METODOLOGÍA

3.1. TIPO DE INVESTIGACIÓN

Esta investigación es del tipo descriptivo porque describe las características fundamentales de la solución numérica de un problema con valores iniciales, esta técnica de solución se circunscribe en el estudio de las ecuaciones diferenciales ordinarias. La descripción se efectúa utilizando criterios sistemáticos que permiten poner de manifiesto la estructura y el comportamiento de estos problemas.

3.2. METODOLOGÍA

La metodología aplicada en este trabajo, es el método de prototipos; sigue las siguientes etapas: Planteamiento del problema, análisis del contexto, revisión de la literatura, formulación del problema y de los objetivos de la investigación, estructuración del marco teórico de acuerdo al planteamiento del problema, elección y descripción del software, programación de las actividades para el desarrollo de la aplicación; finalmente el análisis y aplicación experimental del prototipo de la aplicación hecha en VBA/Excel. En las siguientes líneas, se explica las fases del desarrollo.

3.3.1. Modelo de desarrollo de la aplicación

Se aplica el modelo de desarrollo de prototipos, el mismo que contempla los siguientes pasos:

Análisis del problema, recolección de requisitos, y la definición de los objetivos.

Diseño y construcción de la aplicación (prototipo).

Revisión del prototipo para refinar los requisitos del software.

El prototipo de la aplicación que se diseña en este trabajo de tesis debe ser funcional y evolutivo, desarrolla un comportamiento que cumple con los requisitos del problema, y las necesidades de utilidad que se han entendido claramente. La construcción del prototipo se va modificando y adecuando en el proceso de desarrollo, según las condiciones y necesidades (Islam, 2015).

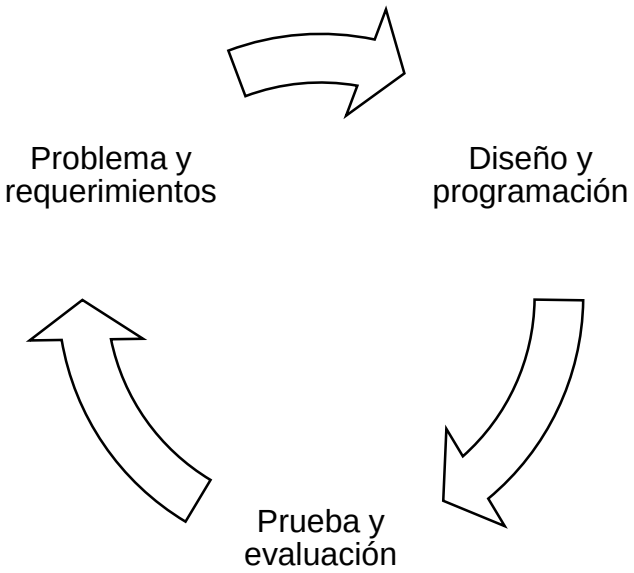


Figura 10. Ciclo de desarrollo de la aplicación.

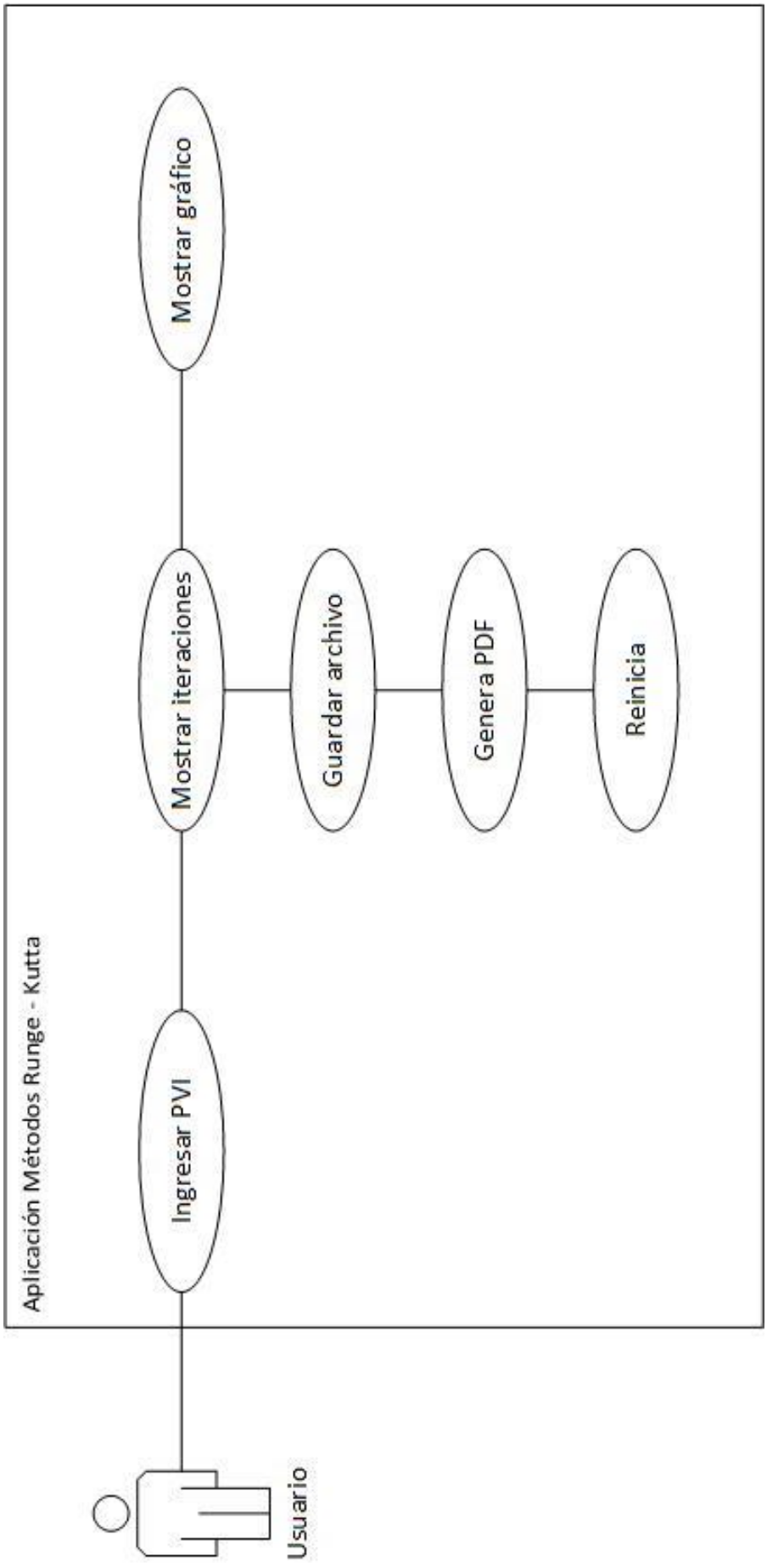


Figura 11. Diagrama de caso de uso.

3.3. HERRAMIENTAS

Para el desarrollo de esta aplicación se consideran las siguientes herramientas de programación:

1. Computadora personal
2. Generador de interfaz VBA.
3. Generador de código VBA.
4. Analizador matemático Cls Math Parser.
5. Herramientas CASE.
6. Bocetos en papel.
7. Cuadernos
8. Archivos
9. Grabaciones en video

3.3.1. Diseño de la interfaz

En el análisis, se indican los siguientes aspectos:

Características de los usuarios.- Estudiantes universitarios de ambos sexos, que tengan conocimientos previos en la solución analítica y numérica, de ecuaciones diferenciales ordinarias de primer orden sujetas a una condición inicial; así como conocimientos básicos en el uso de una hoja electrónica.

Características del entorno del aprendizaje.- En sesiones de aprendizaje en las que se pueda disponer de computadoras, así mismo, en forma libre, en donde existan las condiciones mínimas.

Análisis del contenido.- Se requiere un conocimiento suficiente sobre ecuaciones diferenciales ordinarias sujetas a condiciones iniciales, conocimiento mínimo en el uso del Microsoft Office.

Requerimientos técnicos.- Hardware: Computadora con, Software: Microsoft Office.

En el diseño de la aplicación se debe tener en cuenta el aprendizaje dinámico, mediante una observación de los contenidos conceptuales, en este caso los conceptos de ecuaciones diferenciales ordinarias, problemas con valor inicial, condiciones iniciales, errores absolutos, etc. Al implementar estos conceptos en la programación de sus algoritmos, se puede desarrollar en los estudiantes universitarios, habilidades que demandan las realidades que se viven en estos últimos años.

Se ha procurado que la interfaz sea estándar e intuitiva, trate de ser comprensible y de mostrar la solución de un problema con una sencillez que pueda ser comprendido y entendido rápidamente. Así mismo se procura la sobrecarga de formularios u hojas de trabajo con demasiados controles, pues esto conduciría a confundir al usuario. Por último, se procura mostrar un gráfico, que como se sabe, ayuda a una mejor comprensión del problema y de su solución numérica.

3.3.2. Diseño del formulario

Nueve etiquetas, a los cuales se les ha asignado la propiedad Autosize = True. Estas etiquetas son:

Label1: Con caption “Función Lipschitziana”.

Label2: Con caption “f (x, y) =”.

Label3: Con caption “Solución de la ecuación diferencial ordinaria”.

Label4: Con caption “y (t) =”.

Label5: Con caption “Condiciones iniciales”.

Label6: Con caption “Método numérico de Runge-Kutta”.

Label7: Con caption “t_0 =”.

Label8: Con caption “y_0 =”.

Label9: Con caption “b =”.

Cinco cajas de texto:

txtfuncion: Caja de texto para ingresar la función f(t,y), dimensión 174 x 18 twips.

txtsolucion: Caja de texto para ingresar la solución y=y(t), dimensión 174 x 18 twips.

txttinicial: Para ingresar el valor incial de la variable independiente t, dimensión 72 x 18 twips.

txtyinicial: Para ingresar el valor incial de la variable dependiente y, dimensión 72 x 18 twips.

txtb: Para ingresar el límite derecho del intervalo cerrado [a,b] al cual pertenece la variable independiente t, dimensión 72 x 18 twips.

Un cuadro combinado: cbometodos, este cuadro contiene los siguientes ítems (opciones de métodos a aplicar en los cálculos):

Runge - Kutta 1.

Runge - Kutta 2.

Runge - Kutta 3.

Runge - Kutta 4.

Cinco botones de comando, todos con las mismas dimensione 80 x 24 twips:

cmditera: botón de comando para mostrar iteraciones en una hoja de Excel, habilita los botones gráfico, guardar y llevar a PDF.

cmdgrafica: Para mostrar la gráfica de la solución.

cmdguardar: Para guardar el libro de Excel cue contiene las iteraciones.

cmdapdf: General un PDF de los resultados.

cmdreinicia: Limpia la hoja de Excel que contiene las iteraciones, borra el contenido de las cajas de texto, deshabilita los botones gráfico, guardar y llevar a PDF.

Estos controles (Figura 12), se han dispuesto de una forma que guarde una secuencia en su uso, y las dimensiones de cada control deben estar de acuerdo al propósito o función que debe cumplir.

Figura 12. Formulario de la aplicación.

3.3.3. Sub procedimientos de la aplicación

Esta aplicación contiene 16 sub procedimientos los cuales aparecen en el código del programa en el siguiente orden:

Sub cmdapdf_Click()

Sub cmdgrafica_Click()

Sub cmdguardar_Click()

Sub cmditera_Click()

Sub cmdreinicia_Click()

Sub txtb_Change()

Sub txttinicial_Change()

Sub txtyinicial_Change()

Sub UserForm_Initialize()

Sub PrimerCuadro()

Sub SegundoCuadro()

Sub TercerCuadro()

Sub RK1()

Sub RK2()

Sub RK3()

Sub RK4()

Los primeros cinco sub procedimientos corresponden a las instrucciones que deben realizarse al presionar los cinco botones de comando, los tres siguientes sub procedimientos corresponden a las cajas de texto donde deben ingresarse solo números. El noveno sub procedimiento es para ver cómo estará el formulario al iniciar una sesión. Los sub procedimientos 10, 11 y 12 son para trazar cuadros en los cuales deberán estar los valores obtenidos en las iteraciones.

Los cuatro últimos sub procedimientos son para realizar las iteraciones aplicando los métodos de Runge – Kutta: RK1, RK2, RK3 y RK4. Enseguida se muestran los diagramas de flujo de cada uno de ellos, en cada diagrama, la variable n representa el número de iteraciones a realizarse, se han elegido para esta variable tres valores, estos son: n = 10, n = 20 y n = 40. Esta elección se ha realizado inspirado en las tablas mostradas en las página 98 del texto de ecuaciones diferenciales (Trench, 2013).

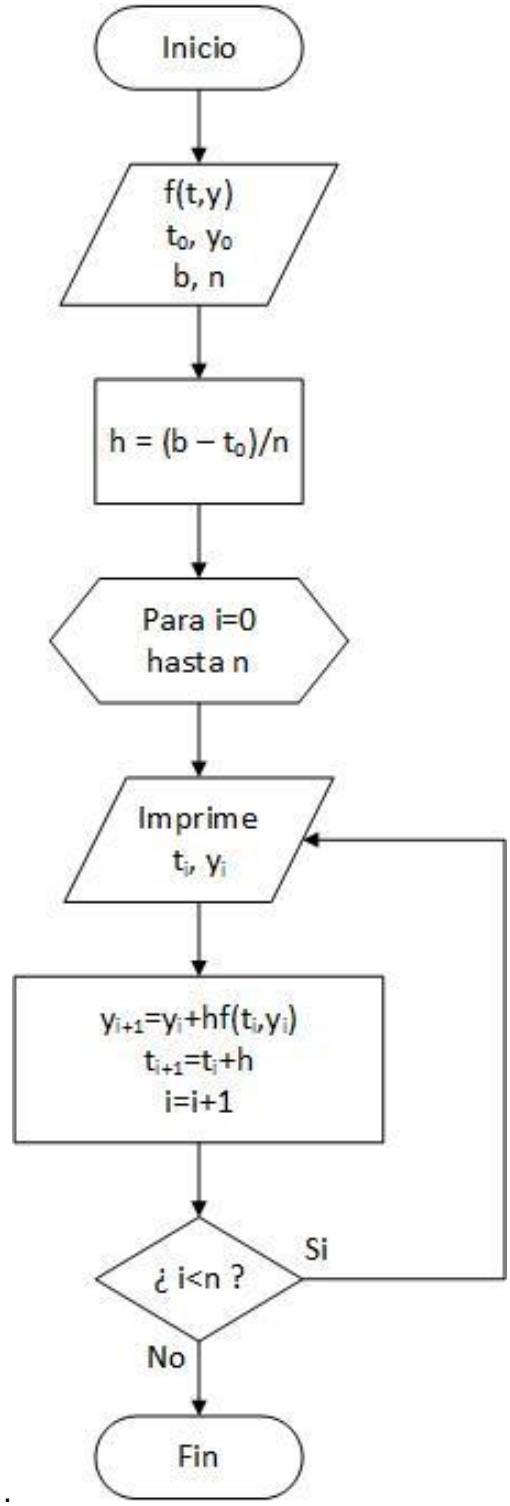


Figura 13. Diagrama de flujo del método Runge – Kutta 1.

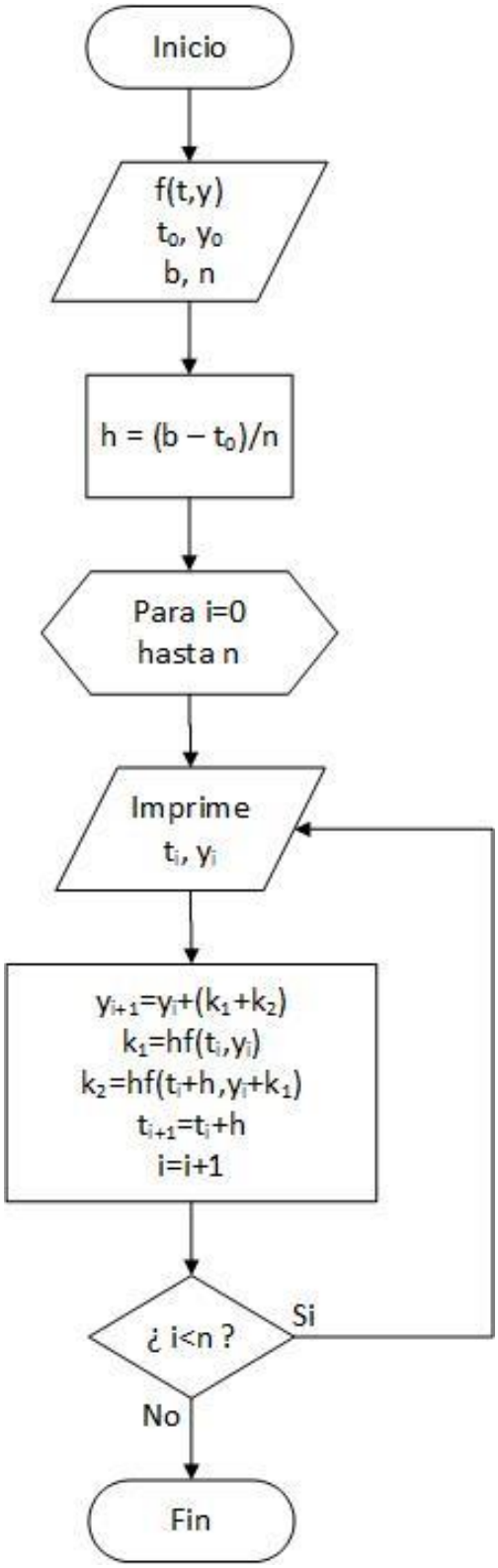


Figura 14. Diagrama de flujo del método Runge – Kutta 2.

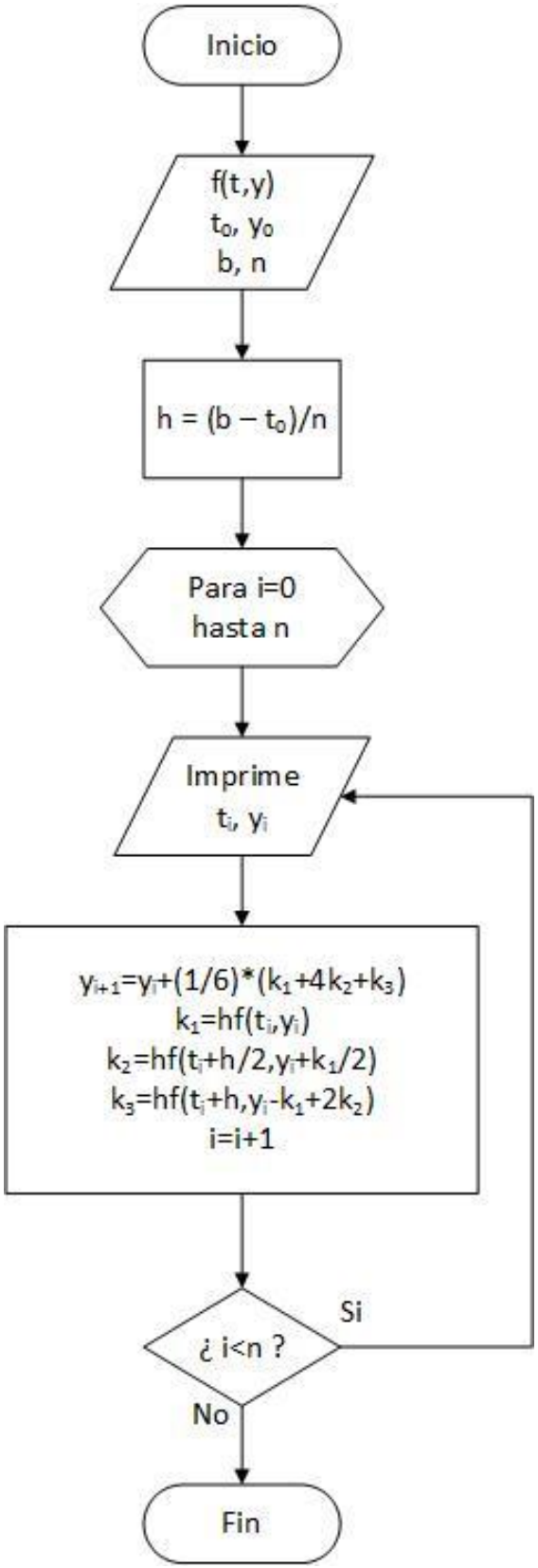


Figura 15. Diagrama de flujo del método Runge – Kutta 3.

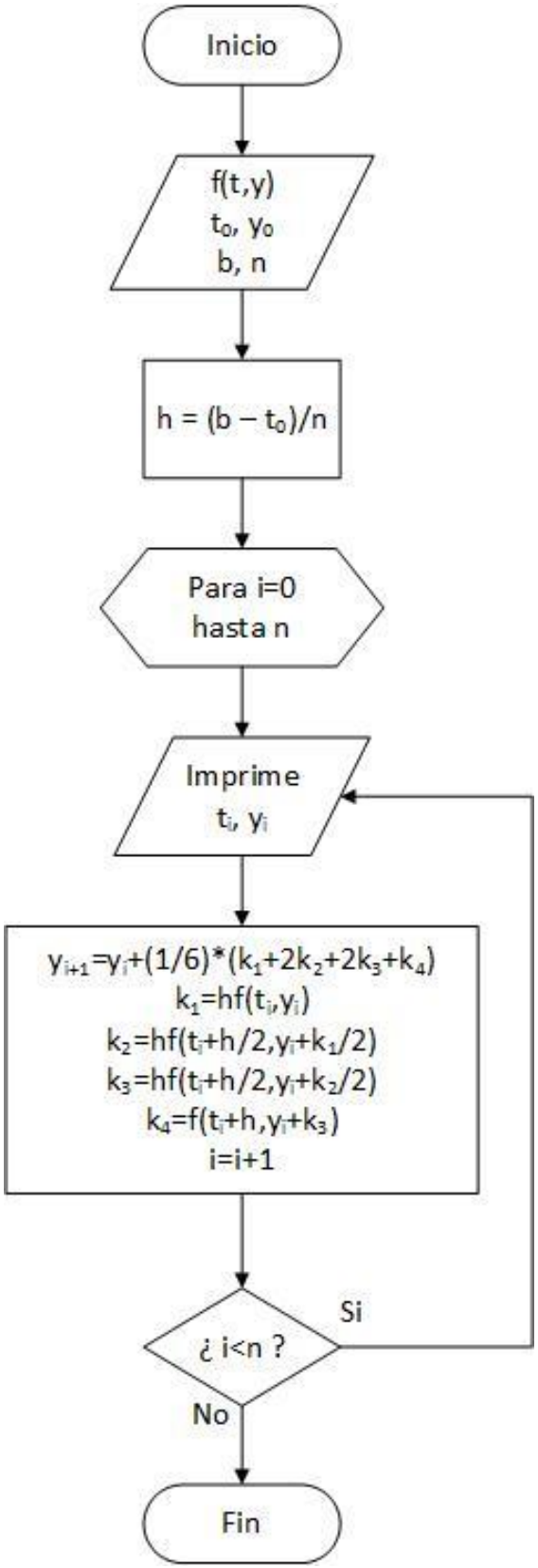


Figura 16. Diagrama de flujo del método Runge – Kutta 4.

3.3.4. Características de los usuarios

Esta aplicación será utilizada por estudiantes de las asignaturas de Análisis Numérico y Métodos Numéricos. Los usuarios deben tener conocimientos en: Solución de ecuaciones diferenciales ordinarias y en el uso de la hoja electrónica Excel. En el diseño y desarrollo de esta aplicación (prototipo) se ha procurado considerar los datos imprescindibles que tienen que ser ingresados para obtener el resultado correcto. Cabe indicar que esta aplicación es fácilmente modificable por el usuario, pues se trata de un prototipo exploratorio y experimental.

3.3.5. Reporte

Con el software Code Line Counter 5.0, se obtiene el reporte de la aplicación (Figura **17**). Y el siguiente cuadro es un resumen traducido de este reporte.

Cuadro 1. Resumen del reporte del código de la aplicación.

Working Report with Code Line Counter	
File name (Nombre archivo)	frmRungeKutta.frm
Languaje (Lenguaje)	VB/VB.NET
Nominal lines (Líneas nominales)	733
Source Code Lines (Líneas de código fuente)	558
Source Code Lines % (Líneas de código fuente %)	76.13%
Comment Lines (Líneas de comentarios)	40
Comment Lines % (Líneas de comentarios %)	5.46%
Todo Lines	0
Todo Lines %	0.00%
Blank Lines (Líneas en blanco)	139
Blank Lines % (Líneas en blanco %)	18.96%
Mixed Lines (Líneas mixtas)	4
Mixed Lines % (Líneas mixtas %)	0.55%
Regular Lines (Líneas regulares)	0
Regular Lines % (Líneas regulares %)	0.00%

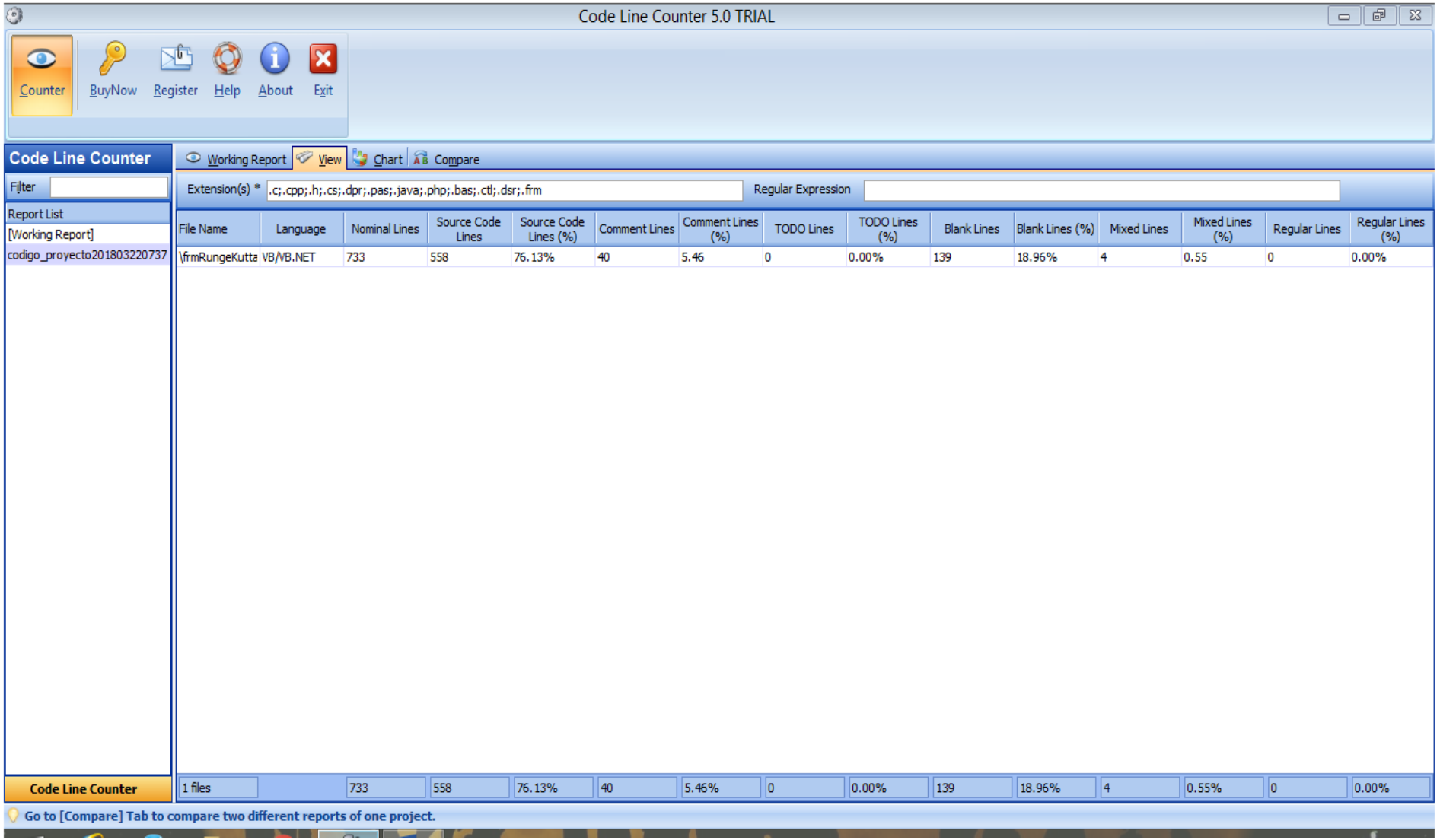


Figura 17. Reportaje de las líneas de código.

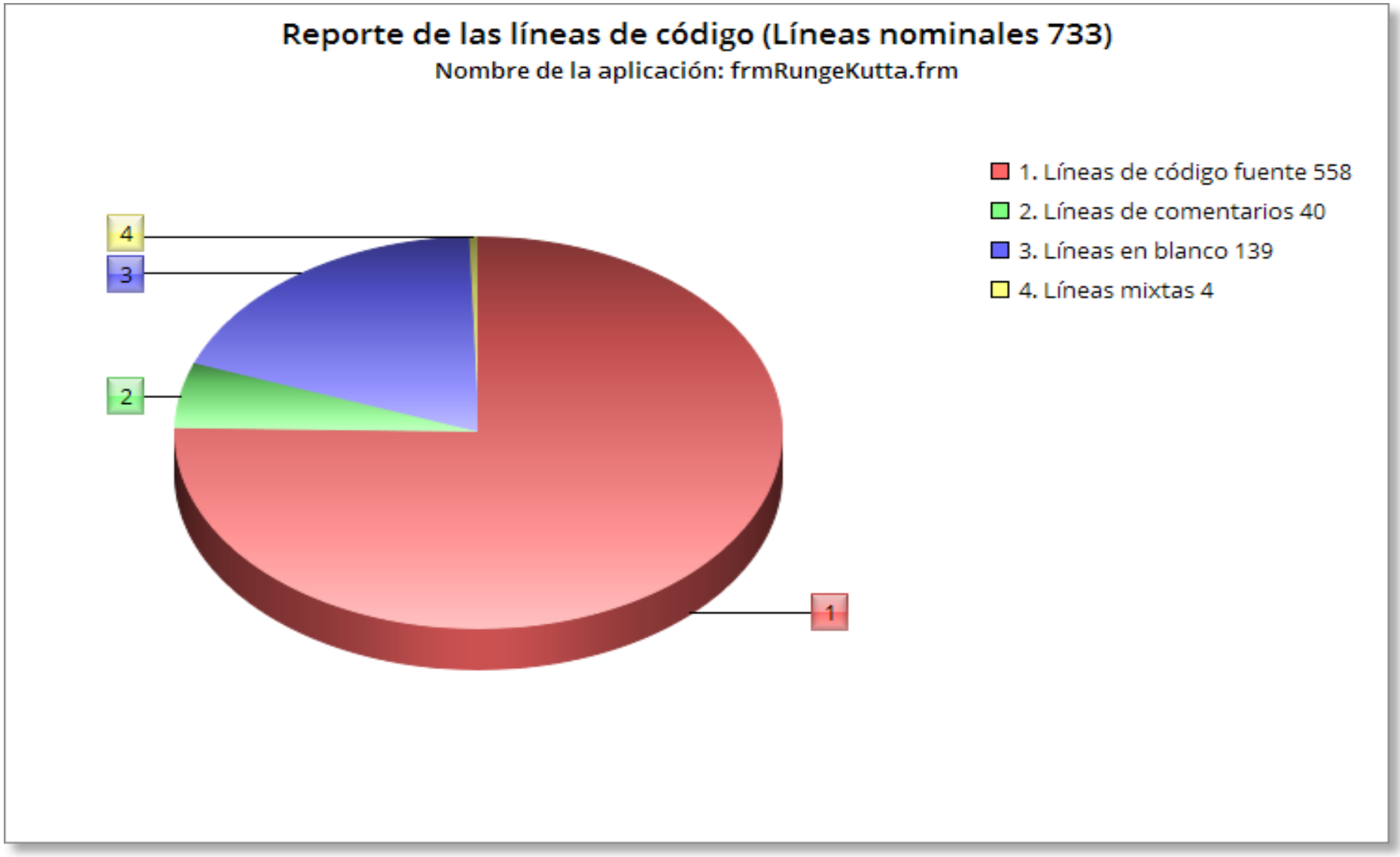


Figura 18. Gráfico circular del reportaje de la aplicación.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

Para ver los resultados del uso de la aplicación, se plantea el problema con valor inicial del ejemplo 3.1.2 que se halla en el texto “ELEMENTARY DIFFERENTIAL EQUATIONS” (Trench, 2013).

Resolver el PVI:

$$\frac{dy}{dt} + 2y = t^3 e^{-2t}; \quad 0 \leq t \leq 1; \quad y(0) = 1$$

Inicialmente se muestra la solución analítica de la ecuación diferencial de este PVI, y enseguida se expone la forma como utilizar la aplicación en VB/Excel que permite realizar los cálculos iterativos y mostrar los resultados aproximados eligiendo uno de los cuatro métodos de Runge – Kutta. La Tabla 2 muestra estos resultados considerándose los valores con seis decimales.

Al resolver analíticamente, se observa que esta ecuación diferencial de este PVI, es una ecuación lineal de la forma.

$$\frac{dy}{dt} + P(t)y = Q(t)$$

Donde:

$$P(t) = 2$$
$$Q(t) = t^3 e^{-2t}$$

Y cuya solución puede ser hallada mediante la aplicación de la fórmula:

$$y(t) = e^{-\int P(t)dt}\left(\int Q(t)e^{\int P(t)dt}dt + C\right)$$

Donde la constante *C* se calcula aplicando la condición inicial $y(0) = 1$, entonces la solución exacta del PVI planteado es:

$$y(t) = \frac{e^{-2t}}{4}(t^4 + 4)$$

En el siguiente cuadro se muestran los valores exactos obtenidos con la solución exacta y los valores aproximados obtenidos con el método de Runge – Kutta 4.

Cuadro 2. Valores obtenidos con el método de Runge - Kutta 4.

i	t_1	Valores exactos	Valores aproximados
0	0.0	1.000000	1.000000
1	0.1	0.818751	0.818754
2	0.2	0.670588	0.670592
3	0.3	0.549923	0.549928
4	0.4	0.452205	0.452210
5	0.5	0.373628	0.373633
6	0.6	0.310953	0.310959
7	0.7	0.261399	0.261405
8	0.8	0.222571	0.222576
9	0.9	0.192412	0.192417
10	1.0	0.169169	0.169173

Por otro lado, si para este mismo PVI se utiliza la aplicación “MÉTODOS DE RUNGE – KUTTA”, se debe acceder a la ventana de macros y ejecutar el macro “METODOS_DE_RUNGE_KUTTA”.

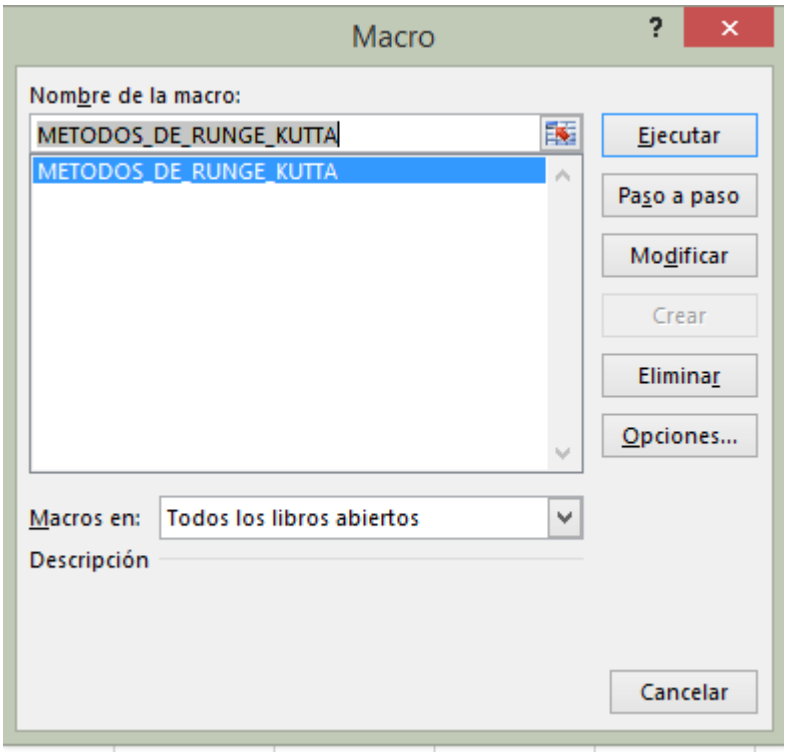
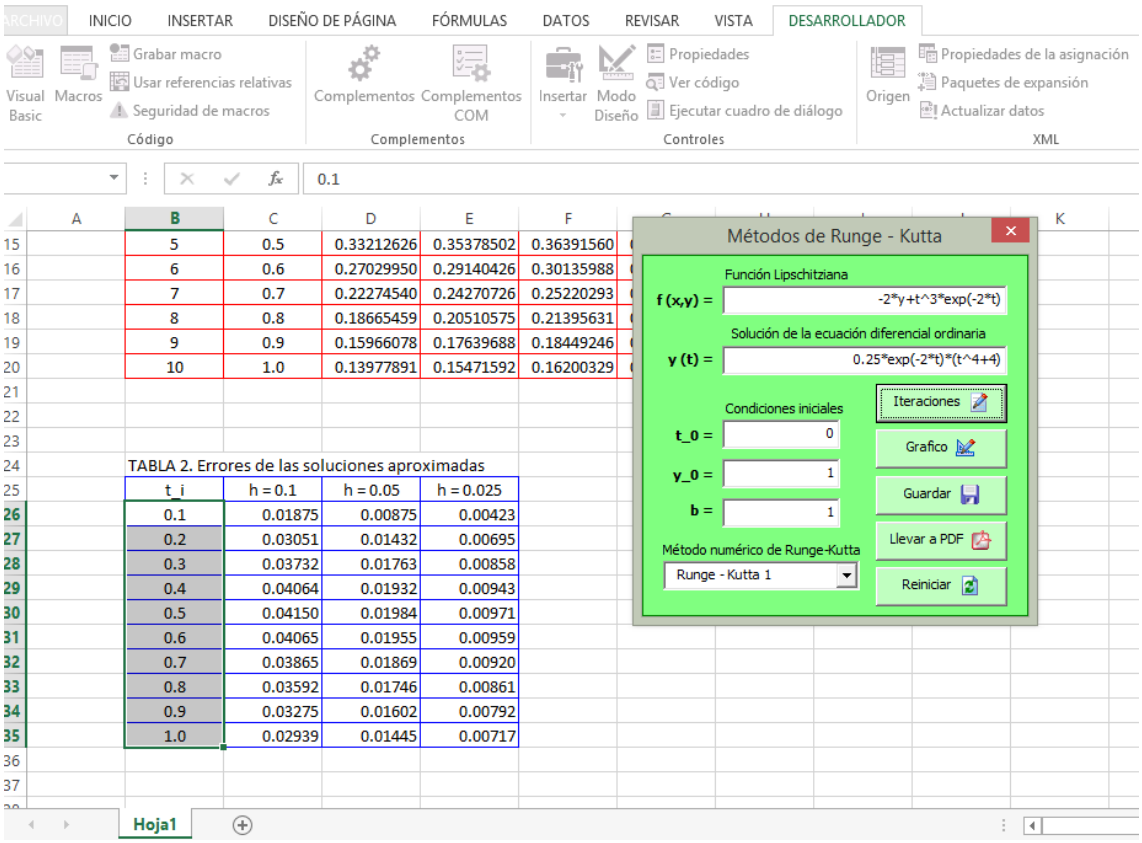


Figura 19. Acceso al formulario de la aplicación.

Enseguida se abre el formulario de la aplicación que consta de: cinco cajas de texto para ingresar en ellos la función $f(t, y)$ del PVI y las condiciones iniciales, cinco botones de comando, de los cuales están habilitadas y los otros tres botones se habilitarán cuando se haya ejecutado el PVI, y un cuadro combinado para elegir el método numérico de Runge – Kutta deseado.

Figura 20. Formulario inicial.

Figura 21. Datos del PVI en el formulario.



t_i	h = 0.1	h = 0.05	h = 0.025
0.1	0.01875	0.00875	0.00423
0.2	0.03051	0.01432	0.00695
0.3	0.03732	0.01763	0.00858
0.4	0.04064	0.01932	0.00943
0.5	0.04150	0.01984	0.00971
0.6	0.04065	0.01955	0.00959
0.7	0.03865	0.01869	0.00920
0.8	0.03592	0.01746	0.00861
0.9	0.03275	0.01602	0.00792
1.0	0.02939	0.01445	0.00717

Figura 22. Iteraciones obtenidas en el Excel.

Se ha obtenido la Tabla 3.1.1 de (Trench, 2013).

1	**CONDICIONES INICIALES DEL PVI**	
2	Función f(t,y):	-2*y+t^3*exp(-2*t)
3	Valor inicial de t:	0
4	Valor inicial de y:	1
5	Extremo derecho de [a,b]	1
6	Solución exacta:	0.25*exp(-2*t)*(t^4+4)

TABLA 1. Soluciones numéricas con el método RK1

i	t_i	h = 0.1	h = 0.05	h = 0.025	Sol. exacta
0	0.0	1.00000000	1.00000000	1.00000000	1.00000000
1	0.1	0.80000000	0.81000566	0.81451835	0.81875122
2	0.2	0.64008187	0.65626644	0.66363595	0.67058817
3	0.3	0.51260175	0.53229098	0.54133949	0.54992298
4	0.4	0.41156320	0.43288706	0.44277477	0.45220467
5	0.5	0.33212626	0.35378502	0.36391560	0.37362756
6	0.6	0.27029950	0.29140426	0.30135988	0.31095290
7	0.7	0.22274540	0.24270726	0.25220293	0.26139895
8	0.8	0.18665459	0.20510575	0.21395631	0.22257072
9	0.9	0.15966078	0.17639688	0.18449246	0.19241204
10	1.0	0.13977891	0.15471592	0.16200329	0.16916910

Figura 23. Iteraciones para tres valores de la longitud de paso h.

Así mismo se obtiene la Tabla 3.1.2 de (Trench, 2013).

TABLA 2. Errores de las soluciones aproximadas

t_i	h = 0.1	h = 0.05	h = 0.025
0.1	0.01875	0.00875	0.00423
0.2	0.03051	0.01432	0.00695
0.3	0.03732	0.01763	0.00858
0.4	0.04064	0.01932	0.00943
0.5	0.04150	0.01984	0.00971
0.6	0.04065	0.01955	0.00959
0.7	0.03865	0.01869	0.00920
0.8	0.03592	0.01746	0.00861
0.9	0.03275	0.01602	0.00792
1.0	0.02939	0.01445	0.00717

Figura 24. Tabla de los errores absolutos.

Así mismo el gráfico para h = 0.1.

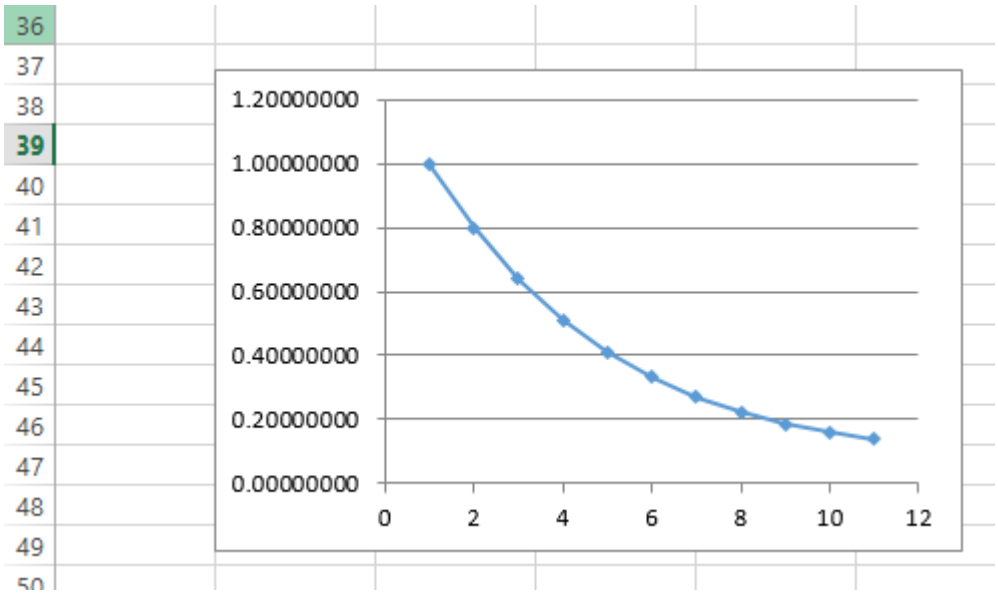
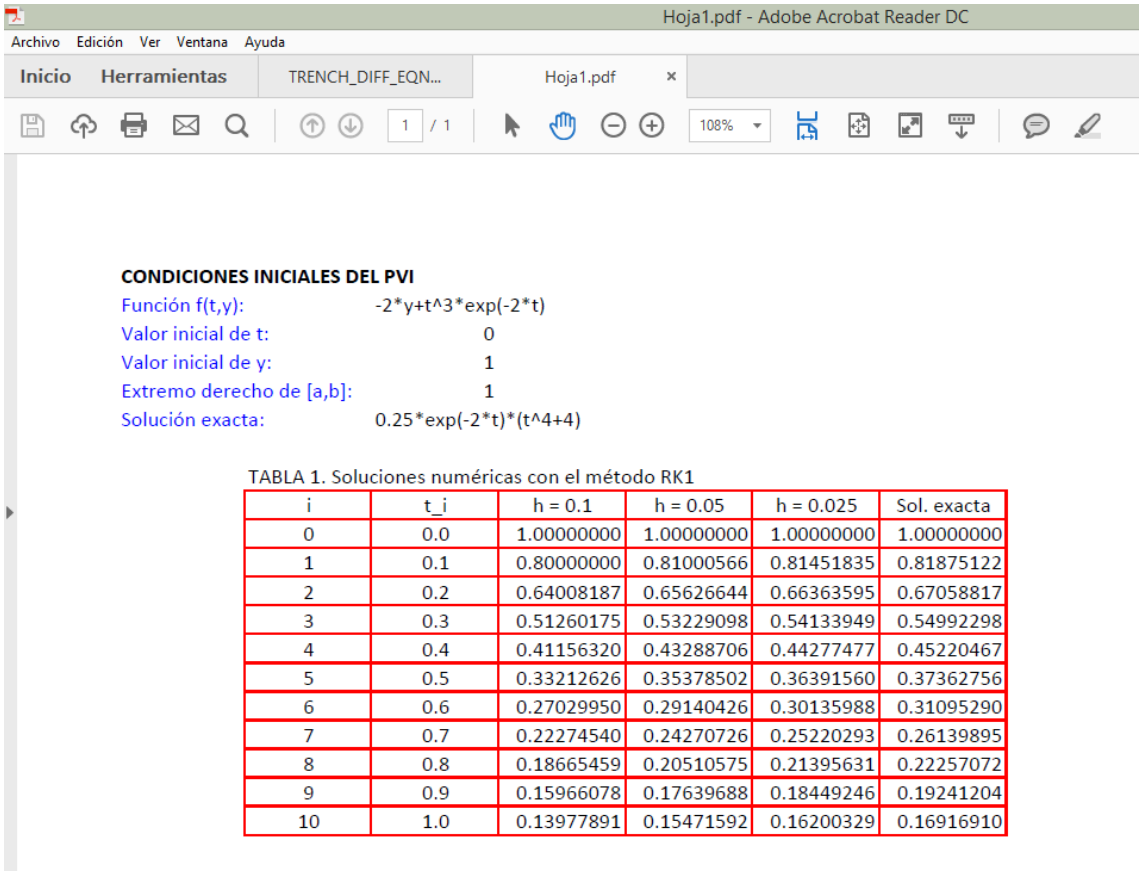


Figura 25. Gráfico de la solución numérica.

Finalmente, el PDF generado.



CONDICIONES INICIALES DEL PVI

Función f(t,y):	-2*y+t^3*exp(-2*t)
Valor inicial de t:	0
Valor inicial de y:	1
Extremo derecho de [a,b]:	1
Solución exacta:	0.25*exp(-2*t)*(t^4+4)

TABLA 1. Soluciones numéricas con el método RK1

i	t_i	h = 0.1	h = 0.05	h = 0.025	Sol. exacta
0	0.0	1.00000000	1.00000000	1.00000000	1.00000000
1	0.1	0.80000000	0.81000566	0.81451835	0.81875122
2	0.2	0.64008187	0.65626644	0.66363595	0.67058817
3	0.3	0.51260175	0.53229098	0.54133949	0.54992298
4	0.4	0.41156320	0.43288706	0.44277477	0.45220467
5	0.5	0.33212626	0.35378502	0.36391560	0.37362756
6	0.6	0.27029950	0.29140426	0.30135988	0.31095290
7	0.7	0.22274540	0.24270726	0.25220293	0.26139895
8	0.8	0.18665459	0.20510575	0.21395631	0.22257072
9	0.9	0.15966078	0.17639688	0.18449246	0.19241204
10	1.0	0.13977891	0.15471592	0.16200329	0.16916910

Figura 26. Parte del PDF generado.

Obviamente esta aplicación en VBA/Excel puede ser aplicada a cualquier problema con valor inicial que conste de:

1. Una ecuación diferencial ordinaria de primer orden de la forma:$y\text{^}' = f(t, y)$, donde la función *f* debe ser Lipschitziana.
2. La mencionada ecuación diferencial debe tener una solución calculada analíticamente (solución exacta).
3. Condiciones iniciales coherentes.

Si para la ecuación diferencial dada no fuera posible hallar su solución exacta, se puede modificar el formulario y el código de la aplicación. Así mismo se puede modificar el código de la aplicación si se desea otros valores de la longitud de paso h. Cabe indicar que en la aplicación considerada, se han tomado los siguientes valores de h: 0.1, 0.05 y 0.025, que corresponden a una partición del intervalo $[a, b]$ del PVI en n=10, n=20 y n=40 subintervalos respectivamente.

CONCLUSIONES

- La programación en VBA/Excel constituye una herramienta versátil en el proceso iterativo de los métodos de Runge – Kutta en la solución y el análisis de un problema con valor inicial. Esto se debe a la combinación del interfaz del Excel y de su editor de Visual Basic. haciéndose más sencillo la administración de los datos y la presentación de los resultados. A estas cualidades se le añade la facilidad con la que se pueden realizar, entre otras cosas, las representaciones gráficas correspondientes a las tablas que se obtienen al realizar los cálculos iterativos.

- La utilización del módulo de clase clsMathParser permite ingresar una fórmula matemática en la ejecución y poder realizar cálculos matemáticos reiterativos con estas fórmulas, sin tener que recurrir a la modificación permanentemente del código para ingresar una función de un PVI y su correspondiente solución exacta.

- Con el desarrollo de la aplicación objeto de este trabajo de tesis, se puede deducir que en la didáctica universitaria orientada a la enseñanza de los métodos numéricos, obviamente el Visual Basic es de fácil comprensión y cuando se trabaja conjuntamente con el Excel, el trabajo aún es más

sencillo por la sencilla razón de contar ya con celdas que sirven para una mejor disposición de los resultados.

RECOMENDACIONES

- La amplia difusión del Excel y su uso cotidiano como hoja de cálculo, así como disponer de la sencillez del lenguaje de programación Visual Basic, esta conjunción hace más flexible el trabajo de implementar macros que usualmente se aplican a problemas que tienen que ver con bases de datos o trabajos de contabilidad financiera, sin embargo es factible aplicar en la estructuración de algoritmos en métodos numéricos. Por lo tanto, se recomienda su aplicación como herramienta de aprendizaje no sólo en la implementación de los algoritmos de Runge – Kutta, sino de otros problemas que se estudian en las asignaturas de análisis numérico, métodos numéricos, programación lineal, optimización y otras. Así mismo, se recomienda que en la programación de algoritmos que tengan que ver con problemas matemáticos en la Física, Química, etc. Se tenga que utilizar analizadores de ecuaciones como el clsMathParser u otros. Además la programación de los algoritmos en VBA/Excel provistos de un analizador matemático exige en el alumno la creatividad, la originalidad, el interés en el estudio y otras habilidades y capacidades, esto constituye una razón para no eludir la responsabilidad de desarrollar el trabajo de

enseñar en las aulas universitarias los fundamentos teóricos, las técnicas y la estructuración de códigos en distintos lenguajes de programación.

- Para concluir, también se recomienda a las personas interesados en este tema, tomar como ejemplo este humilde trabajo y proponerse la creación de prototipos de otras aplicaciones en otros lenguajes de programación que puedan tratar el tema de los métodos numéricos Runge – Kutta, así como otros temas del Análisis Numérico u otros temas en la programación lineal, optimización, etc.

BIBLIOGRAFÍA

Birnbaun, D. (2005). *Microsoft Excel VBA Programming for the Absolute Beginner* (Second; M. Garvey, ed.). Boston: Stacy L. Hiquet.

Burden, R. (2010). *Numerical Analysis* (Ninth; M. Julet, ed.). USA: Richard Stratton.

Chapra C, S. & C. P. R. (2010). *Numerical Methods for Engineers* (Sixth; Mc Graw Hill, ed.). New York: Mc Graw Hill.

Harvey, G. (2010). *Microsoft Excel 2010 All - in -One for Dummies* (I; I. Wiley Publishing, ed.). Indianapolis - Indiana - USA: Wiley Publishing, Inc.

Islam, A. (2015). A Comparative Study on Numerical Solutions of Initial Value Problems (IVP) for Ordinary Differential Equations (ODE) with Euler and Runge Kutta Methods. *American Journal of Computational Mathematics*, *5*, 12. Retrieved from https://www.scirp.org/journal/paperinformation.aspx?paperid=59961

Naón, C. . R. R. . & S. E. (2014). *Ecuaciones Diferenciales en Física* (I; Editorial de la Universidad de la Plata, ed.). Buenos Aires - Argentina: Editorial de la Universidad de la Plata.

Simmons, G. & Krantz, S. (2007). *Ecuaciones Diferenciales* (Primera Ed; McGraw Hill, ed.). México.

Stoer, J. . & B. R. (1993). *Introduction to Numerical Analysis* (Springer, ed.). New York: Springer.

Tay, K. (2015). The Fourth Order Runge-Kutta Spreadsheet Calculator Using VBA Programing for Ordinary Differential Equations. *ScienceDirect*, *204*, 9. Retrieved from https://www.sciencedirect.com/science/article/pii/S187704281504793X

Zanini, V. (2013). *Macros en Excel* (I; RedUSERS - Comunidad de Tecnología, ed.). Buenos Aires - Argentina: RedUSERS - Comunidad de Tecnología.

.

ANEXOS

Anexo 1. Código de la aplicación

```
Option Explicit

Dim i, j, k As Integer
Dim t0, y0, a, b, h As Double
Dim t(1 To 1000), y(1 To 1000) As Double
Dim k1, k2, k3, k4 As Double
Dim func As String
Dim solu As String
Dim f As New clsMathParser
Dim g As New clsMathParser
Dim ok As Boolean
' Declaración de variables para el PDF y el gráfico
Dim NombreArchivo, RutaArchivo As String
Dim GraficoRK As ChartObject

Private Sub cmdapdf_Click()

NombreArchivo = ActiveSheet.Name
RutaArchivo = ActiveWorkbook.Path & "\" & NombreArchivo & ".pdf"
ActiveSheet.ExportAsFixedFormat Type:=xlTypePDF, Filename:=RutaArchivo, _
quality:=xlQualityStandard, includedocproperties:=True, ignoreprintareas:=False, _
openafterpublish:=True
```

```
End Sub

Private Sub cmdgrafica_Click()

Set GraficoRK = ActiveSheet.ChartObjects.Add(Left:=60, Top:=550, Width:=280, Height:=180)

With GraficoRK.Chart
      .SetSourceData Source:=ActiveSheet.Range("D10:D20") 'Datos a graficar
      .ChartType = xlXYScatterLines 'Tipo de grafico
      .Legend.Delete ' La leyenda
End With
End Sub

Private Sub cmdguardar_Click()

'Declaramos las variables.
Dim VentanasProtegidas As Boolean
Dim EstructuraProtegida As Boolean
Dim NombreHoja As String
Dim Confirmacion As String
Dim NombreArchivo As String
Dim GuardarComo As Variant
Dim Extension As String
'
'En caso de error.
```

```
On Error GoTo ErrorHandler
'
'Validamos si la ventana o la estructura del archivo están protegidos.
VentanasProtegidas = ActiveWorkbook.ProtectWindows
EstructuraProtegida = ActiveWorkbook.ProtectStructure
'
'En caso de estar protegidas mostramos mensaje.
If VentanasProtegidas = True Or EstructuraProtegida = True Then
    MsgBox "No se puede ejecutar el comando cuando la estructura del archivo está protegida.", _
           vbExclamation, "EXCELeINFO"
Else
    '
    'Copiamos la hoja y guardamos.
    NombreHoja = ActiveSheet.Name
    Confirmacion = MsgBox("Desea guardar la hoja '" & NombreHoja & "' como archivo nuevo?", _
                          vbQuestion + vbYesNo, "EXCELeINFO")
    Application.ScreenUpdating = False
    If Confirmacion = vbYes Then
        ActiveSheet.Select
        ActiveSheet.Copy
        NombreArchivo = ActiveWorkbook.Name
        GuardarComo = Application.GetSaveAsFilename(InitialFileName:=NombreHoja, _
            fileFilter:="Libro de Excel(*.xlsx), *.xlsx, Libro de Excel habilitado para macros(*.xlsm), *.xlsm, Libro
de Excel 97-2003(*.xls), *.xls,CSV (delimitado por comas)(*.csv),*.csv", _
            Title:="EXCELeINFO - guadar hoja activa como archivo nuevo.")
```

```
        If GuardarComo = False Then
            Workbooks(NombreArchivo).Close SaveChanges:=False
        Else
            With Application.WorksheetFunction
                Extension = .Trim(Right(.Substitute(GuardarComo, ".", .Rept(" ", 500)), 500))
            End With

            Select Case Extension
            Case Is = "xlsx"
                ActiveWorkbook.SaveAs GuardarComo
            Case Is = "xlsm"
                ActiveWorkbook.SaveAs GuardarComo, xlOpenXMLWorkbookMacroEnabled
            Case Is = "xls"
                ActiveWorkbook.SaveAs GuardarComo, xlExcel8
            Case Is = "csv"
                ActiveWorkbook.SaveAs GuardarComo, xlCSV
            Case Else
                ActiveWorkbook.SaveAs GuardarComo
            End Select
        End If
    Else
    End If
    '
End If
'
```

```
Exit Sub
'
'En caso de error mostramos un mensaje.
ErrorHandler:
MsgBox "Ha ocurrido un error: " & Err.Description, vbExclamation, "EXCELeINFO"
Workbooks(NombreArchivo).Close SaveChanges:=False
'
End Sub

Private Sub cmditera_Click()

If txtfuncion = Empty Or (txtsolucion = Empty Or (txttinicial = Empty Or txtyinicial = Empty Or txtb = Empty)) Then
    MsgBox "Faltan datos"
Else
    Select Case cbometodos
    Case "Runge - Kutta 1"
        Call RK1
        ' Poner título a la tabla
        Range("B8").Value = "TABLA 1. Soluciones numéricas con el método RK1"
        Call PrimerCuadro
        Call SegundoCuadro
        Call TercerCuadro

    Case "Runge - Kutta 2"
        Call RK2
```

```
    ' Poner título a la tabla
    Range("B8").Value = "TABLA 1. Soluciones numéricas con el método RK2"
    Call PrimerCuadro
    Call SegundoCuadro
    Call TercerCuadro

Case "Runge - Kutta 3"
    Call RK3
    ' Poner título a la tabla
    Range("B8").Value = "TABLA 1. Soluciones numéricas con el método RK3"
    Call PrimerCuadro
    Call SegundoCuadro
    Call TercerCuadro

Case "Runge - Kutta 4"
    Call RK4
    ' Poner título a la tabla
    Range("B8").Value = "TABLA 1. Soluciones numéricas con el método RK4"
    Call PrimerCuadro
    Call SegundoCuadro
    Call TercerCuadro

Case Else
    MsgBox "Faltan datos"
```

```
    End Select

' Para habilitar los botones de graficar y llevar a PDF
    cmdguardar.Enabled = True
    cmdgrafica.Enabled = True
    cmdapdf.Enabled = True
End If
End Sub

Private Sub cmdreinicia_Click()

' Para borrar celdas de la hoja de trabajo
    ActiveSheet.Cells.Clear

' Para borrar las cajas de texto del formulario
    txtfuncion = Empty
    txtsolucion = Empty
    txttinicial = Empty
    txtyinicial = Empty
    txtb = Empty

' Para desactivar dos botones del formulario
    cmdguardar.Enabled = False
    cmdgrafica.Enabled = False
    cmdapdf.Enabled = False
```

```
' Para reiniciar el combobox
    cbometodos = Empty

' Para dejar el cursor en la primera caja de texto
    txtfuncion.SetFocus

' Para borrar la figura
    ActiveSheet.ChartObjects.Delete

End Sub

Private Sub txtb_Change()

'Para ingresar en la sólo números en la quinta caja de texto
If Not IsNumeric(txtb.Text) And (txtb.Text <> "" And txtb.Text <> "-") Then
    Beep
    MsgBox "Se debe ingresar sólo números"
    txtb = Empty
    txtb.SetFocus
End If

End Sub

Private Sub txttinicial_Change()
```

```
'Para ingresar en la sólo números en la tercera caja de texto
If Not IsNumeric(txttinicial.Text) And (txttinicial.Text <> "" And txttinicial.Text <> "-") Then
    Beep
    MsgBox "Se debe ingresar sólo números"
    txttinicial = Empty
    txttinicial.SetFocus
End If

End Sub

Private Sub txtyinicial_Change()

'Para ingresar en la sólo números en la cuarta caja de texto
If Not IsNumeric(txtyinicial.Text) And (txtyinicial.Text <> "" And txtyinicial.Text <> "-") Then
    Beep
    MsgBox "Se debe ingresar sólo números"
    txtyinicial = Empty
    txtyinicial.SetFocus
End If
End Sub

Private Sub UserForm_Initialize()

cbometodos.AddItem "Runge - Kutta 1"
```

```
cbometodos.AddItem "Runge - Kutta 2"
cbometodos.AddItem "Runge - Kutta 3"
cbometodos.AddItem "Runge - Kutta 4"

cmdgrafica.Enabled = False
cmdguardar.Enabled = False
cmdapdf.Enabled = False

End Sub

Sub PrimerCuadro()

Range("A1").Font.Bold = True
Range("A1").Value = "CONDICIONES INICIALES DEL PVI"
Range("A2:A6").Font.Color = vbBlue
Range("A2").Value = "Función f(t,y): "
Range("A3").Value = "Valor inicial de t: "
Range("A4").Value = "Valor inicial de y: "
Range("A5").Value = "Extremo derecho de [a,b]: "
Range("A6").Value = "Solución exacta: "

End Sub


Sub SegundoCuadro()
```

```
' Para poner bordes de color rojo a las iteraciones
    Range("B9:G20").Borders.Color = vbRed

' Centralizar el texto que se colocará en las celdas
' del anterior rango
    Range("B9:G9").HorizontalAlignment = xlCenter

' Los encabezados en el cuadro de las iteraciones
    Range("B9").Value = "i"
    Range("C9").Value = "t_i"
    Range("D9").Value = "h = 0.1"
    Range("E9").Value = "h = 0.05"
    Range("F9").Value = "h = 0.025"
    Range("G9").Value = "Sol. exacta"

End Sub


Sub TercerCuadro()

    Range("B24").Value = "TABLA 2. Errores de las soluciones aproximadas"

' Para colorear de azul los bordes
    Range("B25:E35").Borders.Color = vbBlue
```

```
' Centralizar texto en los encabezados
    Range("B25:E25").HorizontalAlignment = xlCenter

' Escribir los encabezados
    Range("B25").Value = "t_i"
    Range("C25").Value = "h = 0.1"
    Range("D25").Value = "h = 0.05"
    Range("E25").Value = "h = 0.025"

End Sub

Sub RK1()

    Range("C2").Value = txtfuncion.Text
    Range("C6").Value = txtsolucion.Text
    Range("C3").Value = Val(txttinicial.Text)
    Range("C4").Value = Val(txtyinicial.Text)
    Range("C5").Value = Val(txtb.Text)

    func = Range("C2").Value
    ok = f.StoreExpression(func)

    If Not ok Then
        MsgBox ("Error en la función" + f.ErrorDescription)
```

```
    Exit Sub
End If

t(1) = Range("C3").Value
y(1) = Range("C4").Value

f.Variable("t") = t(1)
f.Variable("y") = y(1)

a = Val(txttinicial.Text)
b = Val(txtb.Text)

solu = Range("C6").Value
ok = g.StoreExpression(solu)

Range("C10:C20").NumberFormat = "###,##0.0"
Range("D10:G20").NumberFormat = "###,##0.00000000"

For i = 1 To 11
    Cells(i + 9, 2).HorizontalAlignment = xlCenter
    Cells(i + 9, 2).Value = i - 1
    Cells(i + 9, 3).HorizontalAlignment = xlCenter
    Cells(i + 9, 3) = t(i)
    Cells(i + 9, 4) = y(i)
    f.Variable("t") = t(i)
```

```
    f.Variable("y") = y(i)
    y(i + 1) = y(i) + (b - a) / 10 * f.Eval()
    t(i + 1) = t(i) + (b - a) / 10
Next i

For i = 1 To 22
    f.Variable("t") = t(i)
    f.Variable("y") = y(i)
    y(i + 1) = y(i) + (b - a) / 20 * f.Eval()
    t(i + 1) = t(i) + (b - a) / 20
    For j = 1 To 11
        Cells(j + 9, 5).Value = y(2 * j - 1)
    Next j
Next i


For i = 1 To 44
    f.Variable("t") = t(i)
    f.Variable("y") = y(i)
    y(i + 1) = y(i) + (b - a) / 40 * f.Eval()
    t(i + 1) = t(i) + (b - a) / 40
    For k = 1 To 11
        Cells(k + 9, 6).Value = y(4 * k - 3)
    Next k
Next i
```

```
    For i = 1 To 11
        y(i + 1) = g.Eval1(t(i))
        t(i + 1) = t(i) + (b - a) / 10
        Cells(i + 9, 7).Value = y(i + 1)
    Next i

    Range("C11:C20").Copy
    Range("B26:B35").PasteSpecial xlPasteAll

    For i = 1 To 10
        Range("C26:E35").NumberFormat = "###,##0.00000"
        Cells(i + 25, 3).Value = Abs(Cells(i + 10, 7).Value - Cells(i + 10, 4).Value)
        Cells(i + 25, 4).Value = Abs(Cells(i + 10, 7).Value - Cells(i + 10, 5).Value)
        Cells(i + 25, 5).Value = Abs(Cells(i + 10, 7).Value - Cells(i + 10, 6).Value)
    Next i
End Sub


Sub RK2()

    Range("C2").Value = txtfuncion.Text
    Range("C6").Value = txtsolucion.Text
    Range("C3").Value = Val(txttinicial.Text)
```

```
Range("C4").Value = Val(txtyinicial.Text)
Range("C5").Value = Val(txtb.Text)

func = Range("C2").Value
ok = f.StoreExpression(func)

If Not ok Then
    MsgBox ("Error en la función" + f.ErrorDescription)
    Exit Sub
End If

t(1) = Range("C3").Value
y(1) = Range("C4").Value

f.Variable("t") = t(1)
f.Variable("y") = y(1)

a = Val(txttinicial.Text)
b = Val(txtb.Text)

solu = Range("C6").Value
ok = g.StoreExpression(solu)

k1 = 0
k2 = 0
```

```
Range("C10:C20").NumberFormat = "###,##0.0"
Range("D10:G20").NumberFormat = "###,##0.00000000"

For i = 1 To 11
    Cells(i + 9, 2).HorizontalAlignment = xlCenter
    Cells(i + 9, 2).Value = i - 1
    Cells(i + 9, 3).HorizontalAlignment = xlCenter
    Cells(i + 9, 3).Value = t(i)
    Cells(i + 9, 4).Value = y(i)
    f.Variable("t") = t(i)
    f.Variable("y") = y(i)
    k1 = (b - a) / 10 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 10
    f.Variable("y") = y(i) + k1
    k2 = (b - a) / 10 * f.Eval()
    y(i + 1) = y(i) + 0.5 * (k1 + k2)
    t(i + 1) = t(i) + (b - a) / 10
Next i

k1 = 0
k2 = 0

For i = 1 To 22
    f.Variable("t") = t(i)
```

```
    f.Variable("y") = y(i)
    k1 = (b - a) / 20 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 20
    f.Variable("y") = y(i) + k1
    k2 = (b - a) / 20 * f.Eval()
    y(i + 1) = y(i) + 0.5 * (k1 + k2)
    t(i + 1) = t(i) + (b - a) / 20
    For j = 1 To 11
        Cells(j + 9, 5).Value = y(2 * j - 1)
    Next j
Next i

k1 = 0
k2 = 0

For i = 1 To 44
    f.Variable("t") = t(i)
    f.Variable("y") = y(i)
    k1 = (b - a) / 40 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 40
    f.Variable("y") = y(i) + k1
    k2 = (b - a) / 40 * f.Eval()
    y(i + 1) = y(i) + 0.5 * (k1 + k2)
    t(i + 1) = t(i) + (b - a) / 40
    For j = 1 To 11
```

```
            Cells(j + 9, 6).Value = y(4 * j - 3)
        Next j
    Next i


    For i = 1 To 11
        y(i + 1) = g.Eval1(t(i))
        t(i + 1) = t(i) + (b - a) / 10
        Cells(i + 9, 7).Value = y(i + 1)
    Next i

    Range("C11:C20").Copy
    Range("B26:B35").PasteSpecial xlPasteAll

    For i = 1 To 10
        Range("C26:E35").NumberFormat = "###,##0.00000"
        Cells(i + 25, 3).Value = Abs(Cells(i + 10, 7).Value - Cells(i + 10, 4).Value)
        Cells(i + 25, 4).Value = Abs(Cells(i + 10, 7).Value - Cells(i + 10, 5).Value)
        Cells(i + 25, 5).Value = Abs(Cells(i + 10, 7).Value - Cells(i + 10, 6).Value)
    Next i

End Sub


Sub RK3()
```

```
Range("C2").Value = txtfuncion.Text
Range("C6").Value = txtsolucion.Text
Range("C3").Value = Val(txttinicial.Text)
Range("C4").Value = Val(txtyinicial.Text)
Range("C5").Value = Val(txtb.Text)

func = Range("C2").Value
ok = f.StoreExpression(func)

If Not ok Then
    MsgBox ("Error en la función" + f.ErrorDescription)
    Exit Sub
End If

t(1) = Range("C3").Value
y(1) = Range("C4").Value

f.Variable("t") = t(1)
f.Variable("y") = y(1)

a = Val(txttinicial.Text)
b = Val(txtb.Text)

solu = Range("C6").Value
```

```
ok = g.StoreExpression(solu)

k1 = 0
k2 = 0
k3 = 0

Range("C10:C20").NumberFormat = "###,##0.0"
Range("D10:G20").NumberFormat = "###,##0.00000000"

For i = 1 To 11
    Cells(i + 9, 2).HorizontalAlignment = xlCenter
    Cells(i + 9, 2).Value = i - 1
    Cells(i + 9, 3).HorizontalAlignment = xlCenter
    Cells(i + 9, 3).Value = t(i)
    Cells(i + 9, 4).Value = y(i)
    f.Variable("t") = t(i)
    f.Variable("y") = y(i)
    k1 = (b - a) / 10 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 20
    f.Variable("y") = y(i) + k1 / 2
    k2 = (b - a) / 10 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 10
    f.Variable("y") = y(i) - k1 + 2 * k2
    k3 = (b - a) / 10 * f.Eval()
    y(i + 1) = y(i) + 1 / 6 * (k1 + 4 * k2 + k3)
```

```
    t(i + 1) = t(i) + (b - a) / 10
Next i

k1 = 0
k2 = 0
k3 = 0

For i = 1 To 22
    f.Variable("t") = t(i)
    f.Variable("y") = y(i)
    k1 = (b - a) / 20 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 40
    f.Variable("y") = y(i) + k1 / 2
    k2 = (b - a) / 20 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 20
    f.Variable("y") = y(i) - k1 + 2 * k2
    k3 = (b - a) / 20 * f.Eval()
    y(i + 1) = y(i) + 1 / 6 * (k1 + 4 * k2 + k3)
    t(i + 1) = t(i) + (b - a) / 20
    For j = 1 To 11
        Cells(j + 9, 5).Value = y(2 * j - 1)
    Next j
Next i

k1 = 0
```

```
k2 = 0
k3 = 0

For i = 1 To 44
    f.Variable("t") = t(i)
    f.Variable("y") = y(i)
    k1 = (b - a) / 40 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 80
    f.Variable("y") = y(i) + k1 / 2
    k2 = (b - a) / 40 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 40
    f.Variable("y") = y(i) - k1 + 2 * k2
    k3 = (b - a) / 40 * f.Eval()
    y(i + 1) = y(i) + 1 / 6 * (k1 + 4 * k2 + k3)
    t(i + 1) = t(i) + (b - a) / 40
    For j = 1 To 11
        Cells(j + 9, 6).Value = y(4 * j - 3)
    Next j
Next i

For i = 1 To 11
    y(i + 1) = g.Eval1(t(i))
    t(i + 1) = t(i) + (b - a) / 10
    Cells(i + 9, 7).Value = y(i + 1)
Next i
```

```
    Range("C11:C20").Copy
    Range("B26:B35").PasteSpecial xlPasteAll

    For i = 1 To 10
        Range("C26:E35").NumberFormat = "###,##0.000000"
        Cells(i + 25, 3).Value = Abs(Cells(i + 10, 7).Value - Cells(i + 10, 4).Value)
        Cells(i + 25, 4).Value = Abs(Cells(i + 10, 7).Value - Cells(i + 10, 5).Value)
        Cells(i + 25, 5).Value = Abs(Cells(i + 10, 7).Value - Cells(i + 10, 6).Value)
    Next i

End Sub

Sub RK4()

    Range("C2").Value = txtfuncion.Text
    Range("C6").Value = txtsolucion.Text
    Range("C3").Value = Val(txttinicial.Text)
    Range("C4").Value = Val(txtyinicial.Text)
    Range("C5").Value = Val(txtb.Text)

    func = Range("C2").Value
    ok = f.StoreExpression(func)

    If Not ok Then
```

```
    MsgBox ("Error en la función" + f.ErrorDescription)
    Exit Sub
End If

t(1) = Range("C3").Value
y(1) = Range("C4").Value

f.Variable("t") = t(1)
f.Variable("y") = y(1)

a = Val(txttinicial.Text)
b = Val(txtb.Text)

solu = Range("C6").Value
ok = g.StoreExpression(solu)

k1 = 0
k2 = 0
k3 = 0
k4 = 0

Range("C10:C20").NumberFormat = "###,##0.0"
Range("D10:G20").NumberFormat = "###,##0.00000000"

For i = 1 To 11
```

```
    Cells(i + 9, 2).HorizontalAlignment = xlCenter
    Cells(i + 9, 2).Value = i - 1
    Cells(i + 9, 3).HorizontalAlignment = xlCenter
    Cells(i + 9, 3).Value = t(i)
    Cells(i + 9, 4).Value = y(i)
    f.Variable("t") = t(i)
    f.Variable("y") = y(i)
    k1 = (b - a) / 10 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 20
    f.Variable("y") = y(i) + k1 / 2
    k2 = (b - a) / 10 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 20
    f.Variable("y") = y(i) + k2 / 2
    k3 = (b - a) / 10 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 10
    f.Variable("y") = y(i) + k3
    k4 = (b - a) / 10 * f.Eval()
    y(i + 1) = y(i) + 1 / 6 * (k1 + 2 * k2 + 2 * k3 + k4)
    t(i + 1) = t(i) + (b - a) / 10
Next i

k1 = 0
k2 = 0
k3 = 0
k4 = 0
```

```
For i = 1 To 22
    f.Variable("t") = t(i)
    f.Variable("y") = y(i)
    k1 = (b - a) / 20 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 40
    f.Variable("y") = y(i) + k1 / 2
    k2 = (b - a) / 20 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 40
    f.Variable("y") = y(i) + k2 / 2
    k3 = (b - a) / 20 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 20
    f.Variable("y") = y(i) + k3
    k4 = (b - a) / 20 * f.Eval()
    y(i + 1) = y(i) + 1 / 6 * (k1 + 2 * k2 + 2 * k3 + k4)
    t(i + 1) = t(i) + (b - a) / 20
    For j = 1 To 11
        Cells(j + 9, 5).Value = y(2 * j - 1)
    Next j
Next i

k1 = 0
k2 = 0
k3 = 0
k4 = 0
```

```
For i = 1 To 44
    f.Variable("t") = t(i)
    f.Variable("y") = y(i)
    k1 = (b - a) / 40 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 80
    f.Variable("y") = y(i) + k1 / 2
    k2 = (b - a) / 40 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 80
    f.Variable("y") = y(i) + k2 / 2
    k3 = (b - a) / 40 * f.Eval()
    f.Variable("t") = t(i) + (b - a) / 40
    f.Variable("y") = y(i) + k3
    k4 = (b - a) / 40 * f.Eval()
    y(i + 1) = y(i) + 1 / 6 * (k1 + 2 * k2 + 2 * k3 + k4)
    t(i + 1) = t(i) + (b - a) / 40
    For j = 1 To 11
        Cells(j + 9, 6).Value = y(4 * j - 3)
    Next j
Next i

For i = 1 To 11
    y(i + 1) = g.Eval1(t(i))
    t(i + 1) = t(i) + (b - a) / 10
    Cells(i + 9, 7).Value = y(i + 1)
```

```
    Next i

    Range("C11:C20").Copy
    Range("B26:B35").PasteSpecial xlPasteAll

    For i = 1 To 10
        Range("C26:E35").NumberFormat = "###,##0.00000000"
        Cells(i + 25, 3).Value = Abs(Cells(i + 10, 7).Value - Cells(i + 10, 4).Value)
        Cells(i + 25, 4).Value = Abs(Cells(i + 10, 7).Value - Cells(i + 10, 5).Value)
        Cells(i + 25, 5).Value = Abs(Cells(i + 10, 7).Value - Cells(i + 10, 6).Value)
    Next i

End Sub
```