UNIVERSIDAD NACIONAL DEL CALLAO

FACULTAD DE CIENCIAS NATURALES Y MÁTEMATICA

UNIDAD DE INVESTIGACIÓN

INFORME FINAL DEL PROYECTO DE INVESTIGACIÓN

"LENGUAJE DE PROGRAMACIÓN C++ APLICADO A LAS ECUACIONES NO LINEALES"

AUTOR: Lic. ELMER ALBERTO LEÓN ZÁRATE

CALLAO, 2018

PERÚ

I. ÍNDICE

LISTA DE GRAFICOS

II. RESUMEN

La presente investigación tuvo como propósito la elaboración de un sistema computacional en el lenguaje de programación C++ que permita implementar los métodos básicos para resolver las Ecuaciones no lineales.

Este trabajo de investigación tiene por título: "LENGUAJE DE PROGRAMACIÓN EN C++ APLICADO A LAS ECUACIONES NO LINEALES" y ha sido preparado para apoyar la formación de los estudiantes de ciencias e ingeniería, quienes podrán aplicar este sistema cuando tengan que resolver problemas matemáticos relacionados con estos temas.

En el capítulo del marco teórico se hace una revisión de los temas del Ecuaciones no lineales y el Lenguaje de programación C++.

En el capítulo de resultados obtenemos el sistema computacional que implementa los métodos básicos utilizados para la solución de ecuaciones no lineales y que son las siguientes:

- Método del punto fijo.
- Método de la bisección.
- Método de Newton.
- Método de Newton modificado.
- Método de la secante.

El sistema computacional implementado servirá para que los alumnos lo utilicen en los cursos del área de matemática cuando tengan que encontrar las raíces de una ecuación no lineal.

Los estudiantes del curso de programación de computadoras también se verán incentivados y tendrán más interés por la programación en un lenguaje de programación científico como el C++.

Palabras clave: Lenguaje de programación C++, Ecuaciones no lineales.

ABSTRACT

The present investigation had as purpose the elaboration of a computer system in the C ++ programming language that allows to implement the basic methods to solve the non-linear equations.

This research work has the title: "PROGRAMMING LANGUAGE IN C ++ APPLIED TO NON-LINEAR EQUATIONS" and has been prepared to support the training of science and engineering students, who will be able to apply this system when they have to solve mathematical problems related to these issues.

In the chapter of the theoretical framework, a review of the topics of nonlinear equations and the C ++ programming language is made.

In the results chapter we obtain the computational system that implements the basic methods used for the solution of nonlinear equations and that are the following:

- Fixed point method.
- Bisection method.
- Newton's method.
- Modified Newton's method.
- Method of the secant.

The implemented computer system will help students use it in the mathematics area when they have to find the roots of a non-linear equation.

Students in the computer programming course will also be encouraged and will have more interest in programming in a scientific programming language such as C++.

Keywords: C ++ programming language, Non-linear equations.

III. INTRODUCCIÓN

El Lenguaje de Programación C++ es Científico y Orientado a Objetos. También se debe tener en cuenta que no se presentan aplicaciones en Lenguaje de Programación C++ para las Ecuaciones no lineales, por lo tanto la bibliografía acerca de este tema es muy escasa.

El Borland C++ que es uno de los Compiladores del Lenguaje de Programación C++ no tiene una librería para los métodos que resuelven las Ecuaciones no lineales y con este trabajo se desea construir esta librería.

¿ES POSIBLE APLICAR LENGUAJE DE PROGRAMACIÓN C++ A LAS ECUACIONES NO LINEALES?

Crear un sistema computacional en Lenguaje de Programación C++ para resolver problemas de las Ecuaciones no lineales, conteniendo las siguientes aplicaciones:

- Implementar el método del punto fijo con una aplicación desarrollada en Lenguaje de Programación C++.
- Implementar el método de la bisección con una aplicación desarrollada en Lenguaje de Programación C++.
- Implementar el método de newton con una aplicación desarrollada en Lenguaje de Programación C++.
- Implementar el método de newton modificado con una aplicación desarrollada en Lenguaje de Programación C++.
- Implementar el método de la secante con una aplicación desarrollada en Lenguaje de Programación C++.

Es importante crear un sistema computacional en Lenguaje de Programación C++ para resolver problemas de las Ecuaciones no lineales y de esta forma apoyar la enseñanza y aprendizaje, con una componente didáctica integrada, que presente un problema en cuya resolución se muestre la necesidad de los conceptos propios de las Ecuaciones no lineales.

Sin embargo, la posibilidad que un medio computacional provee es que bajo un diseño didáctico contribuyen a un ambiente experimental y dinámico, al pasa de los medios tradicionales de pizarrón, tiza, plumón, lápiz y papel a presentar ejemplos y problemas simulados con los que es posible experimentar y explorar sus propiedades; por lo que es una posibilidad de acercamiento a la enseñanza y aprendizaje de las Ecuaciones no lineales si se programan actividades didácticas. Un software tutorial que promueva la enseñanza y el aprendizaje de conceptos importantes de las Ecuaciones no lineales presenta algunas ventajas para los dos agentes del proceso, profesor y alumno.

Un Sistema Computacional como el propuesto en este proyecto de investigación ayuda a que el alumno resuelva problemas por sí mismo, señale sus fallas y corrija sus errores sometiendo sus respuestas de forma repetida; le proporciona ayuda para resolver los problemas en el momento que lo necesite independientemente de la presencia física del profesor pero como si éste lo estuviera acompañando. Además, se adecua a cada estudiante de acuerdo a su nivel de profundidad en el tratamiento de los conceptos y a la necesidad de ayudas; no se imponen soluciones ni métodos. Para el profesor resulta de gran ayuda en la etapa de ejercitación, descargándole de la labor rutinaria y agotadora de evaluar las tareas asignadas y señalar los errores de manera individual, permitiéndole entonces profundizar en el desarrollo conceptual del tema más que en la parte operativa que le requiere tanto tiempo. Así, se contribuye a modificar el papel de los agentes involucrados en el proceso de enseñanza aprendizaje, alumno y profesor, en roles de autoaprendizaje y tutor respectivamente.

El Lenguaje de Programación C++ es Científico y Orientado a Objetos. También se debe tener en cuenta que no se presentan aplicaciones en Lenguaje de Programación C++ para las Ecuaciones no lineales, por lo tanto la bibliografía acerca de este tema es muy escasa.

3.2. Formulación del problema

El Borland C++ que es uno de los Compiladores del Lenguaje de Programación C++ no tiene una Librería para funciones aplicadas a las Ecuaciones no lineales y con este trabajo se desea construir esta librería.

¿ES POSIBLE APLICAR LENGUAJE DE PROGRAMACIÓN C++ A LAS ECUACIONES NO LINEALES?

3.3. Objetivo general y específico

3.3.1. Objetivo general

Crear un sistema computacional en Lenguaje de Programación C++ para resolver Ecuaciones no lineales.

3.3.2. Objetivos específicos

Implementar el método del punto fijo con una aplicación desarrollada en Lenguaje de Programación C++

Implementar el método de la bisección con una aplicación desarrollada en Lenguaje de Programación C++.

Implementar el método de newton con una aplicación desarrollada en Lenguaje de Programación C++.

Implementar el método de newton modificado con una aplicación desarrollada en Lenguaje de Programación C++.

Implementar el método de la secante con una aplicación desarrollada en Lenguaje de Programación C++.

3.4. Importancia

Es importante crear un sistema computacional en Lenguaje de Programación C++ para resolver problemas de Ecuaciones no lineales y de esta forma apoyar la enseñanza y aprendizaje, con una componente didáctica integrada, que presente un problema en cuya resolución se muestre la necesidad de los conceptos propios de las Ecuaciones no lineales.

3.5. Justificación

En este trabajo de investigación estudiaremos algunos métodos numéricos para hallar raíces reales de una ecuación no-lineal en una variable. En la siguiente definición formalizamos el concepto de raíz de una ecuación.

Definición. Dada una función f: R $\rightarrow$ R, resolver la ecuación f(x) = 0, es hallar los valores $\acute{x}$ que anulan a dicha función. A estos valores $\acute{x}$ se les denomina raíces o soluciones de la ecuación, o también, ceros de la función f(x).

Uno de los problemas que se presenta con frecuencia en ingeniería es encontrar las raíces de ecuaciones de la forma f(x) =0, donde f(x) es una función real de una variable x (Nieves Hurtado & Dominguez Sanchez, 1996).

Sin embargo, la posibilidad que un medio computacional provee es que bajo un diseño didáctico contribuyen a un ambiente experimental y dinámico, al pasar de los medios tradicionales de pizarrón, tiza, plumón, lápiz y papel a presentar ejemplos y problemas simulados con los que es posible experimentar y explorar sus propiedades; por lo que es una posibilidad de acercamiento a la enseñanza y aprendizaje de la solución de las Ecuaciones lineales si se programan actividades didácticas. Un software tutorial que promueva la enseñanza y el aprendizaje con algunas ventajas para los dos agentes del proceso, profesor y alumno.

Un Sistema Computacional como el propuesto en este proyecto de investigación ayuda a que el alumno resuelva problemas por sí mismo, señale sus

fallas y corrija sus errores sometiendo sus respuestas de forma repetida; le proporciona ayuda para resolver los problemas en el momento que lo necesite independientemente de la presencia física del profesor pero como si éste lo estuviera acompañando. Además, se adecua a cada estudiante de acuerdo a su nivel de profundidad en el tratamiento de los conceptos y a la necesidad de ayudas; no se imponen soluciones ni métodos. Para el profesor resulta de gran ayuda en la etapa de ejercitación, descargándole de la labor rutinaria y agotadora de evaluar las tareas asignadas y señalar los errores de manera individual, permitiéndole entonces profundizar en el desarrollo conceptual del tema más que en la parte operativa que le requiere tanto tiempo. Así, se contribuye a modificar el papel de los agentes involucrados en el proceso de enseñanza aprendizaje, alumno y profesor, en roles de autoaprendizaje y tutor respectivamente.

IV. MARCO TEÓRICO

4.1. Antecedentes

Existen Software aplicados a la solución de la Ecuaciones no lineales tales como:

a. Métodos de cálculo numérico 3.1.3 creado por Marcelo Calniquer es una aplicación especialmente diseñada para asistir a los estudiantes y profesores en la solución de ecuaciones lineales y no lineales.
b. Octave es un software para la solución de problemas de ingeniería (al estilo de MATLAB).Comparte la sintaxis de MATLAB pero es más poderoso en el sentido de su orientación a objetos.
c. Octave es software libre y actualmente dispone de una interfaz de usuario hecha en QT y muy amigable. Para el trazado de gráficos emplea la herramienta GNUplot, también libre y de calidad en la generación de gráficas científicas.Es un programa multiplataforma ya que corre bajo Windows, Linux y MacOS entre otros. Se maneja por línea de comando, aunque existen numerosas GUI's1, como qtOctave (Ubuntu)
d. El Compilador Borland C++ del Lenguaje de Programación C++ no tiene una librería de funciones para aplicar a las Ecuaciones no lineales, es por ese motivo que planteamos implementarlo en este trabajo de investigación.

4.2. Marco Teórico

El Lenguaje de Programación C++ fue creado por Bjarne Stroustrup en 1985 como una extensión del Lenguaje C, el mismo que es más consistente y conocido que otros lenguajes de programación científicos.

El Lenguaje de Programación C++ es de propósito general porque ofrece control de flujo, estructuras sencillas y un buen conjunto de operadores (Monografias.com, 2016).

Este Lenguaje ha sido creado estrechamente ligado al sistema operativo UNIX, puesto que fueron desarrollados conjuntamente. Sin embargo, este lenguaje

no está ligado a ningún sistema operativo ni a ninguna máquina concreta" (Monografias.com, 2016).

En matemáticas, los sistemas no lineales representan sistemas cuyo comportamiento no es expresable como la suma de los comportamientos de sus descriptores. Más formalmente, un sistema físico, matemático o de otro tipo es no lineal cuando las ecuaciones de movimiento, evolución o comportamiento que regulan su comportamiento son no lineales. En particular, el comportamiento de sistemas no lineales no está sujeto al principio de superposición, como lo es un sistema lineal (Wikipedia, 2016).

En diversas ramas de las ciencias la no linealidad es la responsable de la comportamientos complejos y, frecuentemente, impredecibles o caóticos. La no linealidad frecuentemente aparece ligada a la auto interacción, el efecto sobre el propio sistema del estado anterior del sistema. En física, biología o economía la no linealidad de diversos subsistemas es una fuente de problemas complejos, en las últimas décadas la aparición de los ordenadores digitales y la simulación numérica ha disparado el interés científico por los sistemas no lineales, ya que por primera vez muchos sistemas han podido ser investigados de manera más o menos sistemática.

Algunos sistemas no lineales tienen soluciones exactas o integrables, mientras que otros tienen comportamiento caótico, por lo tanto no se pueden reducir a una forma simple ni se pueden resolver. Un ejemplo de comportamiento caótico son las olas gigantes. Aunque algunos sistemas no lineales y ecuaciones de interés general han sido extensamente estudiados, la vasta mayoría son pobremente comprendidos (Wikipedia, 2016).

Las ecuaciones no lineales son de interés en física y matemáticas debido a que la mayoría de los problemas físicos son implícitamente no lineales en su naturaleza. Ejemplos físicos de sistemas lineales son relativamente raros. Las ecuaciones no lineales son difíciles de resolver y dan origen a interesantes fenómenos como la teoría del caos. Una ecuación lineal puede ser descrita usando un operador lineal, L. Una ecuación lineal en algún valor desconocido de u tiene la

forma: Lu=0. Una ecuación no lineal es una ecuación de la forma: F(u)=0. Para algún valor desconocido de u (Wikipedia, 2016).

Para poder resolver cualquier ecuación se necesita decidir en qué espacio matemático se encuentra la solución u. Podría ser que u es un número real, un vector o, tal vez, una función con algunas propiedades (Wikipedia, 2016).

Las soluciones de ecuaciones lineales pueden ser generalmente descritas como una superposición de otras soluciones de la misma ecuación. Esto hace que las ecuaciones lineales sean fáciles de resolver (Wikipedia, 2016).

Las ecuaciones no lineales son mucho más complejas, y mucho más difíciles de entender por la falta de soluciones simples superpuestas. Para las ecuaciones no lineales las soluciones generalmente no forman un espacio vectorial y, en general, no pueden ser superpuestas para producir nuevas soluciones. Esto hace el resolver las ecuaciones mucho más difícil que en sistemas lineales (Wikipedia, 2016).

4.3. Ecuaciones No Lineales

En matemáticas, los sistemas no lineales representan sistemas cuyo comportamiento no es expresable como la suma de los comportamientos de sus descriptores. Más formalmente, un sistema físico, matemático o de otro tipo es no lineal cuando las ecuaciones de movimiento, evolución o comportamiento que regulan su comportamiento son no lineales. En particular, el comportamiento de sistemas no lineales no está sujeto al principio de superposición, como lo es un sistema lineal (Wikipedia, 2016).

En diversas ramas de las ciencias la no linealidad es la responsable de la comportamientos complejos y, frecuentemente, impredecibles o caóticos. La no linealidad frecuentemente aparece ligada a la auto interacción, el efecto sobre el propio sistema del estado anterior del sistema. En física, biología o economía la no linealidad de diversos subsistemas es una fuente de problemas complejos, en las últimas décadas la aparición de los ordenadores digitales y la simulación numérica ha disparado el interés científico por los sistemas no lineales, ya que por primera vez muchos sistemas han podido ser investigados de manera más o menos sistemática.

Algunos sistemas no lineales tienen soluciones exactas o integrables, mientras que otros tienen comportamiento caótico, por lo tanto no se pueden reducir a una forma simple ni se pueden resolver. Un ejemplo de comportamiento caótico son las olas gigantes. Aunque algunos sistemas no lineales y ecuaciones de interés general han sido extensamente estudiados, la vasta mayoría son pobremente comprendidos (Wikipedia, 2016).

Las ecuaciones no lineales son de interés en física y matemáticas debido a que la mayoría de los problemas físicos son implícitamente no lineales en su naturaleza. Ejemplos físicos de sistemas lineales son relativamente raros. Las ecuaciones no lineales son difíciles de resolver y dan origen a interesantes fenómenos como la teoría del caos. Una ecuación lineal puede ser descrita usando un operador lineal, L. Una ecuación lineal en algún valor desconocido de u tiene la forma: Lu=0. Una ecuación no lineal es una ecuación de la forma: F(u)=0. Para algún valor desconocido de u (Wikipedia, 2016).

Para poder resolver cualquier ecuación se necesita decidir en qué espacio matemático se encuentra la solución u. Podría ser que u es un número real, un vector o, tal vez, una función con algunas propiedades (Wikipedia, 2016).

Las soluciones de ecuaciones lineales pueden ser generalmente descritas como una superposición de otras soluciones de la misma ecuación. Esto hace que las ecuaciones lineales sean fáciles de resolver (Wikipedia, 2016).

Las ecuaciones no lineales son mucho más complejas, y mucho más difíciles de entender por la falta de soluciones simples superpuestas. Para las ecuaciones no lineales las soluciones generalmente no forman un espacio vectorial y, en general, no pueden ser superpuestas para producir nuevas soluciones. Esto hace el resolver las ecuaciones mucho más difíciles que en sistemas lineales (Wikipedia, 2016).

4.3.1. Método del punto fijo

Este método se aplica para resolver ecuaciones de la forma

$$x=g(x)$$

Si la ecuación es f(x)=0, entonces puede despejarse x ó bien sumar x en ambos lados de la ecuación para ponerla en la forma adecuada.

Ejemplos:

1) La ecuación cos(x) – x = 0 se puede transformar en cos (x) = x.
2) La ecuación tan x – e^{-x} = 0 se puede transformar en x + tan x - e^{-x} = x.

Dada la aproximación x_i, la siguiente iteración se calcula con la fórmula:

$$x_{i+1} = g(x_i)$$

Supongamos que la raíz verdadera es x_r, es decir, $x_r = g(x_r)$

Restando las últimas ecuaciones obtenemos:

$$x_r - x_{i+1} = g(x_r) - g(x_i)$$

Por el Teorema del Valor Medio para derivadas, sabemos que si g(x) es continua en [a,b] y diferenciable en (a,b) entonces existe $\xi \epsilon (a, b)$ tal que

$$g'(\xi) = \frac{g(b) - g(a)}{b - a}$$

En nuestro caso, existe ξ en el intervalo determinado por x_i y x_r tal que:

$$g'(\xi) = \frac{g(x_r) - g(x_i)}{x_r - x_i}$$

De aquí tenemos que:

$$g(x_r) - g(x_i) = g'(\xi).(x_r - x_i)$$

O bien,

$$x_r - x_{i+1} = g'(\xi).(x_r - x_i)$$

Tomando valor absoluto en ambos lados,

$$|x_r - x_{i+1}| = |g'(\xi)|.|x_r - x_i|$$

Observe que el término $|x_r - x_{i+1}|$ es precisamente el error absoluto en la $(i + 1)$ –ésima iteración, mientras que el término $|x_r - x_i|$ corresponde al error absoluto en la i –ésima iteración.

Por lo tanto, solamente si $|g'(\xi)| < 1$ entonces se disminuirá el error en la siguiente iteración. En caso contrario, el error irá en aumento.

En resumen, el método de iteración del punto fijo converge a la raíz si $|g'(x)| < 1$ para x en un intervalo $[a, b]$ que contiene a la raíz y donde $g(x)$ es contínua y diferenciable, pero diverge si $|g'(x)| > 1$ en dicho intervalo.

Analicemos nuestros ejemplos anteriores:

- En el ejemplo 01, $g(x) = cosx$ y claramente se cumple la condición de que $|g'(x)| < 1$. Por lo tanto el método sí converge a la raíz.
- En el ejemplo 02, $g(x) = x + tan(x) - e^{-x}$ y en este caso, $|g'(x)| = |1 + sec^2 - e^{-x}|$. Por lo tanto, el método no converge a la raíz.

Para aclarar el uso de la fórmula veamos dos ejemplos:

Ejemplo 01

Usar el método de iteración del punto fijo para aproximar la raíz de $f(x) = cos(x) - x$, comenzando con $x_0 = 0$ y hasta que $|\epsilon_a| < 1\%$.

Solución

Como ya aclaramos anteriormente, el método sí converge a la raíz.

Aplicando la fórmula iterativa tenemos,

$$x_1 = g(x_0) = cos\ 0 = 1$$

Con un error aproximado de 100 %.

Aplicando nuevamente la fórmula iterativa tenemos,

$$x_2 = g(x_1) = cos1 = 0.540302305$$

Y un error aproximado de 85.08%.

Intuimos que el error aproximado se irá reduciendo muy lentamente. En efecto, se necesitan hasta 13 iteraciones para lograr reducir el error aproximado menor al 1%. El resultado final que se obtiene es:

$$x_{13} = 0.7414250866$$

Con un error aproximado igual al 0.78%.

Ejemplo 02

Usar el método de iteración del punto fijo para aproximar la raíz de $f(x) = x^2 + 5x - e^x$, comenzando con $x_0 = 0$ y hasta que $|\epsilon_a| < 1\%$.

Solución

Si despejamos la x del término lineal, vemos que la ecuación equivale a

$$\frac{x^2 - e^x}{5} = x$$

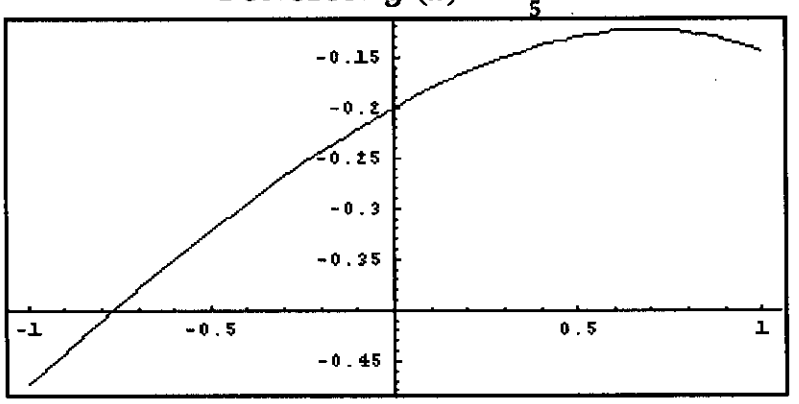
de donde,

$$g(x) = \frac{x^2 - e^x}{5}$$

En este caso, tenemos que $g'(x) = \frac{2x - e^x}{5}$, véase en el siguiente gráfico:

GRAFICO 1

FUNCION $g'(x) = \frac{2x - e^x}{5}$



Fuente propia

nos convence que $|g'(x)| < 1$, para $x \in [-1,1]$, lo que es suficiente para deducir que el método sí converge a la raíz buscada.

Aplicando la fórmula iterativa, tenemos:

$$x_1 = g(x_0) = -0.2$$

Con un error aproximado del 100%.

Aplicando nuevamente la fórmula iterativa, tenemos:

$$x_2 = g(x_1) = -0.1557461506$$

Con un error aproximado igual al 28.41%.

En este ejemplo, el método solo necesita de 5 iteraciones para reducir el error menor al 1%. Resumimos los resultados en la siguiente tabla:

Aprox. a la raíz	Error aprox.
0	
-0.2	100%
-0.1557461506	28.41%
-0.1663039075	6.34%
-0.163826372	1.51%
-0.164410064	0.35%

De donde vemos que la aproximación buscada es:

$$x_5 = -0.164410064$$

4.3.2. Método de la bisección

El método de bisección se basa en el siguiente teorema de Cálculo:

Teorema del Valor Intermedio

Sea $f(x)$ continua en un intervalo $[a, b]$ y supongamos que $f(a) < f(b)$. Entonces para cada z tal que $f(a) < z < f(b)$, existe un $x_0 \in (a, b)$tal que$f(x_0) = z$. La misma conclusión se obtiene para el caso que $f(a) > f(b)$.

Básicamente el Teorema del Valor Intermedio nos dice que toda función contínua en un intervalo cerrado, una vez que alcanzó ciertos valores en los extremos del intervalo, entonces debe alcanzar todos los valores intermedios.

En particular, si $f(a)$ y $f(b)$ tienen signos opuestos, entonces un valor intermedio es precisamente $z = 0$, y por lo tanto, el Teorema del Valor Intermedio nos asegura que debe existir $x_0 \in (a, b)$ tal que $f(x_0 = 0)$, es decir, debe haber por lo menos una raíz de $f(x)$en el intervalo *(a,b)*.

El método de bisección sigue los siguientes pasos:

Sea $f(x)$ continua,

i) Encontrar valores iniciales x_a, $x_b$tales que $f(x_a)$ y $f(x_b)$ tienen signos opuestos, es decir,

$f(x_a).f(x_b) < 0$

ii) La primera aproximación a la raíz se toma igual al punto medio entre x_a y x_b:

$x_r = \frac{x_a + x_b}{2}$

iii) Evaluar $f(x_r)$. Forzosamente debemos caer en uno de los siguientes casos:

$f(x_a).f(x_r) < 0$

En este caso, tenemos que $f(x_a)$ y $f(x_r)$ tienen signos opuestos, y por lo tanto la raíz se encuentra en el intervalo $[x_a, x_r]$.

$f(x_a).f(x_r) > 0$

En este caso, tenemos que $f(x_a)$y $f(x_r)$tienen el mismo signo, y de aquí que $f(x_r)$ y $f(x_b)$tienen signos opuestos. Por lo tanto, la raíz se encuentra en el intervalo$[x_r, x_b]$.

$f(x_a).f(x_r) = 0$

En este caso se tiene que $f(x_r) = 0$ y por lo tanto ya localizamos la raíz.

El proceso se vuelve a repetir con el nuevo intervalo, hasta que:

$$|\epsilon_a| < \epsilon_b$$

es decir,

$$\left|\frac{x_{actual} - x_{previa}}{x_{actual}} \times 100\%\right| < \epsilon_b$$

Ejemplo 01

Aproximar la raíz de $f(x) = e^{-x} - lnx$hasta que $|\epsilon_a| < 1\%$.

Solución

Sabemos por lo visto en el ejemplo 1 de la sección anterior, que la única raíz de $f(x)$ se localiza en el intervalo [1,1.5]. Así que este intervalo es nuestro punto de partida; sin embargo, para poder aplicar el método de bisección debemos checar que y *f(1)* y *f(1.5)* tengan signos opuestos.

En efecto, tenemos que:

$$f(1) = e^{-1} - ln1 = e^{-1} > 0$$

mientras que:

$$f(1.5) = e^{-1.5} - ln1.5 = -0.1823 < 0$$

Cabe mencionar que la función *f(x)* sí es continua en el intervalo [1,1.5] . Así pues, tenemos todos los requisitos satisfechos para poder aplicar el método de bisección. Comenzamos:

i) Calculamos el punto medio (que es de hecho nuestra primera aproximación a la raíz):

$$x_{r_1} = \frac{1 + 1.5}{2} = 1.25$$

ii) Evaluamos $f(1.25) = e^{-1.25} - ln(1.25) = 0.0636 > 0$

iii) Para identificar mejor en que nuevo intervalo se encuentra la raíz, hacemos la siguiente tabla:

f(1)	*f(1.25)*	*f(1.5)*
+	+	-

Por lo tanto, vemos que la raíz se encuentra en el intervalo [1.25, 1.5]. En este punto, vemos que todavía no podemos calcular ningún error aproximado, puesto que solamente tenemos la primera aproximación. Así, repetimos el proceso con el nuevo intervalo [1.25, 1.5].

Calculamos el punto medio (que es nuestra segunda aproximación a la raíz):

$$x_{r_2} = \frac{1.25 + 1.5}{2} = 1.375$$

Aquí podemos calcular el primer error aproximado, puesto que contamos ya con la aproximación actual y la aproximación previa:

$$|\epsilon_a| = \left|\frac{x_{r_2} - x_{r_1}}{x_{r_2}} \times 100\%\right| = 9.09\%$$

Puesto que no se ha logrado el objetivo, continuamos con el proceso.

Evaluamos $f(1.375) = e^{-1.375} - ln(1.375) = -0.06561 < 0$, y hacemos la tabla:

f(1.25)	*f(1.375)*	*f(1.5)*
+	-	-

Así, vemos que la raíz se encuentra en el intervalo [1.25,1.375] .

Calculamos el punto medio,

$$x_{r_1} = \frac{1.25 + 1.375}{2} = 1.3125$$

Y calculamos el nuevo error aproximado:

$$|\epsilon_a| = \left|\frac{x_{r_3} - x_{r_2}}{x_{r_3}} \times 100\%\right| = 4.76\%$$

El proceso debe seguirse hasta cumplir el objetivo.

Resumimos los resultados que se obtienen en la siguiente tabla:

Aprox. a la raíz	Error aprox.
1.25	
1.375	9.09%
1.3125	4.76%
1.28125	2.43%
1.296875	1.20%
1.3046875	0.59%

Así, obtenemos como aproximación a la raíz

$$x_{r_3} = 1.3046875$$

Ejemplo 02

Aproximar la raíz de $f(x) = arctanx + x - 1$hasta que $|\epsilon_a| < 1\%$.

Solución

Como vimos en el ejemplo 2 de la sección anterior, la única raíz de *f(x)* se localiza en el intervalo [0,1]. Para poder aplicar el método de bisección, es importante checar que sí se cumplen las hipótesis requeridas.

Sabemos que *f(x)* es continua en el intervalo [0,1], y checamos que *f(0)* y *f(1)* tengan signos opuestos.

En efecto,

$$f(0) = arctan0 + 0 - 1 = -1 < 0$$

Mientras que,

$$f(1) = arctan1 + 1 - 1 = 0.7853 > 0$$

Por lo tanto, sí podemos aplicar el método de bisección.

Calculamos el punto medio del intervalo [0,1],

$$x_{r_1} = \frac{1+0}{2} = 0.5$$

Que es la primera aproximación a la raíz de *f(x)*.

Evaluamos $f(0.5) = arctan(0.5) + 0.5 - 1 = -0.0363 > 0$

Y hacemos nuestra tabla de signos,

f(0)	*F(0.5)*	*f(1)*
-	-	+

Puesto que *f (0.5)* y *f (1)* tienen signos opuestos, entonces la raíz se localiza en el intervalo [0.5,1].

En este punto, solo contamos con una aproximación, a saber, $x_{r_1} = 0.5$, que es el primer punto medio calculado.

Repetimos el proceso, es decir, calculamos el punto medio ahora del intervalo [0.5,1],

$$x_{r_2} = \frac{1+0.5}{2} = 0.75$$

Que es la nueva aproximación a la raíz de *f(x)*.

Aquí podemos calcular el primer error aproximado:

$$|\epsilon_a| = \left|\frac{0.75 - 0.5}{0.75} \times 100\%\right| = 33.33\%$$

Puesto que no se cumple el objetivo, continuamos con el proceso.

Evaluamos $f(0.75) = arctan(0.75) + 0.75 - 1 = 0.3935 > 0$.

Y hacemos la tabla de signos:

F(0.5)	*F(0.75)*	*f(1)*
-	+	+

Puesto que *f(0.5)* y *f(0.75)* tienen signos opuestos, entonces la raíz se localiza en el intervalo [0.5,0.75].

Calculamos el punto medio,

$$x_{r_3} = \frac{0.5 + 0.75}{2} = 0.625$$

Y el nuevo error aproximado:

$$|\epsilon_a| = \left|\frac{0.625 - 0.75}{0.625} \times 100\%\right| = 20\%$$

El proceso se debe continuar hasta que se logre el objetivo.

Resumimos los resultados que se obtienen en la siguiente tabla:

Aprox. a la raíz	Error aprox.
0.5	
0.75	33.33%
0.625	20%
0.5625	11.11%
0.53125	5.88%
0.515625	3.03%
0.5234375	1.49%
0.51953125	0.75%

De lo cual, vemos que la aproximación buscada es:

$$x_{r_8} = 0.51953125$$

4.3.3. Método de Newton

Según Burden y Faires (2002) afirman que "El método de Newton-Raphson (o simplemente método de Newton) es uno de los métodos numéricos para resolver un problema de búsqueda de raíces f(x)=0 más poderosos y conocidos. Hay muchas formas de introducirlo. La más común consiste en considerarlo gráficamente. Otra posibilidad consiste en derivarlo como una técnica que permite lograr una convergencia más rápida que la que ofrecen otros tipos de iteración funcional. Una tercera forma de introducir el método

de Newton, que estudiaremos a continuación, se basa en los polinomios de Taylor.

Supongamos que $f \in C^2[a,b]$. Sea $\bar{x} \in [a,b]$ una aproximación de p tal que $f'(\bar{x}) \neq 0$ y $|\bar{x} - p|$ es "pequeño". Consideremos el primer polinomio de Taylor para f(x) expandida alrededor de x,

$$f(x) = f(\bar{x}) + (x - \bar{x})f'(\bar{x}) + \frac{(x-\bar{x})^2}{2} f''(\xi(x)),$$

Donde $\xi(x)$ esta entre x y $\bar{x}$. Dado que $f(p) = 0$ esta ecuación, con x=p, da

$$0 = f(\bar{x}) + (x - \bar{x})f'(\bar{x}) + \frac{(x-\bar{x})^2}{2} f''(\xi(x)),$$

Derivamos el método de Newton suponiendo que, como $|p - \bar{x}|$ es tan pequeño, el termino que contiene $(p - \bar{x})^2$ es mucho menor y que

$$0 \approx f(\bar{x}) + (p - \bar{x})f'(\bar{x}),$$

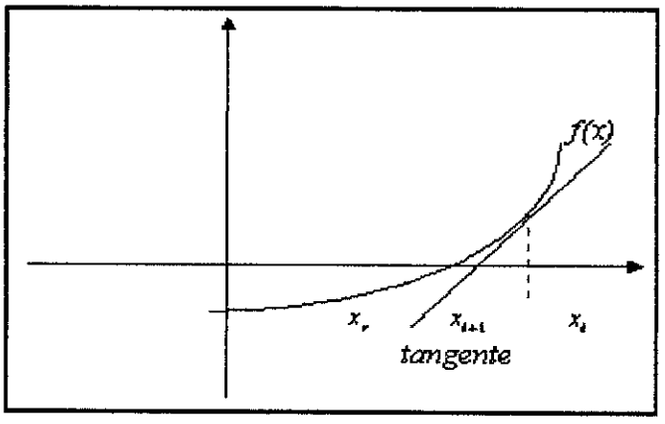
Despejando p de esta ecuación obtenemos

$$p \approx \bar{x} - \frac{f(\bar{x})}{f'(x)},$$

Esto nos prepara para introducir el método de Newton-Raphson, el cual comienza con una aproximación inicial p_0 y genera la sucesión $\{p_n\}$ definida por

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}, \text{ para } n \geq 1 \text{ ," (p.65-66).}$$

GRAFICO 2
RECTA TANGENTE A LA FUNCION f(x)



Fuente propia

También observe que en el caso de que $f'(x_i) = 0$, el método no se puede aplicar. De hecho, vemos geométricamente que esto significa que la recta tangente es horizontal y por lo tanto no intersecta al eje x en ningún punto, a menos que coincida con éste, en cuyo caso x_i es una raíz de $f(x)$.

Ejemplo 01

Usar el método de Newton-Raphson, para aproximar la raíz de $f(x) = e^{-x} - lnx$, comenzando con $x_0 = 1$ y hasta que $|\epsilon_a| < 1\%$.

Solución

En este caso, tenemos que

$$f'(x) = -e^{-x} - \frac{1}{x}$$

De aquí tenemos que:

$$x_{i+1} = x_i - \frac{e^{-x_i} - ln(x_i)}{-e^{-x_i} - \frac{1}{x_i}} = x_i + \frac{e^{x_i} - ln(x_i)}{e^{-x_i} + \frac{1}{x_i}}$$

Comenzamos con $x_0 = 1$ y obtenemos:

$$x_1 = x_0 + \frac{e^{-x_0} - ln(x_0)}{e^{-x_0} + \frac{1}{x_0}} = 1.268941421$$

En este caso, el error aproximado es,

$$|\epsilon_a| = \left|\frac{1.268941421 - 1}{1.268941421} \times 100\%\right| = 21.19\%$$

Continuamos el proceso hasta reducir el error aproximado hasta donde se pidió. Resumimos los resultados en la siguiente tabla:

Aprox. a la raíz	Error aprox.
1	
1.268941421	21.19%
1.309108403	3.06%
1.309799389	0.052%

De lo cual concluimos que la aproximación obtenida es:

$$x_3 = 1.309799389$$

Ejemplo 02

Usar el método de Newton-Raphson para aproximar la raíz de $f(x) = arctanx + x - 1$, comenzando con $x_0 = 0$ y hasta que $|\epsilon_a| < 1\%$.

Solución

En este caso, tenemos que

$$f'(x) = \frac{1}{1+x^2} + 1$$

La cual sustituimos en la fórmula de Newton-Raphson para obtener:

$$x_{i+1} = x_i - \frac{arctan(x_i) + x_i - 1}{\frac{1}{1+x^2} + 1}$$

Comenzamos sustituyendo $x_0 = 0$para obtener:

$$x_1 = x_0 - \frac{arctan(x_0) + x_0 - 1}{\frac{1}{1+x_0^2} + 1} = 0.5$$

En este caso tenemos un error aproximado de $|\epsilon_a| = \left|\frac{0.5-0}{0.5} \times 100\%\right| = 100\%$

Continuamos con el proceso hasta lograr el objetivo. Resumimos los resultados en la siguiente tabla:

Aprox. a la raíz	Error aprox.
0	
0.5	100%
0.5201957728	3.88%
0.5202689918	0.01%

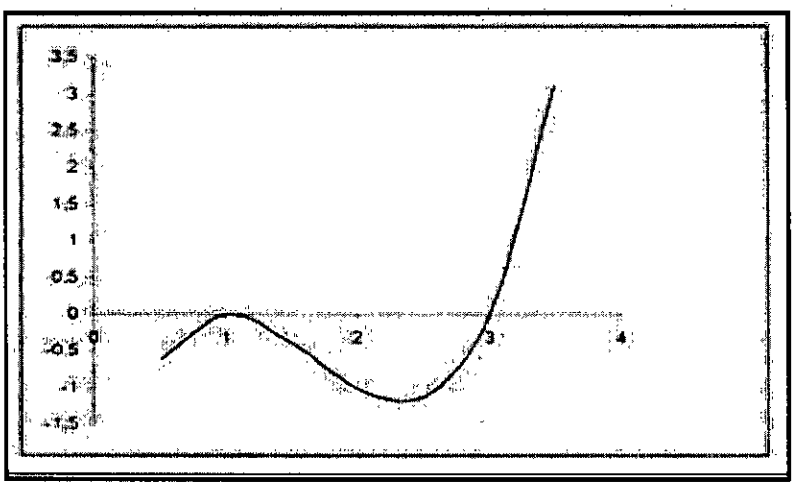
De lo cual concluimos que la aproximación obtenida es:

$$x_3 = 0.5202689918$$

4.3.4. Método de Newton modificado

Para una ecuación polinómica de grado n, se tienen n raíces (entre complejas y reales).

GRAFICO 3
ECUACIÓN CON DOS RAICES



Fuente propia

Se dice que hay una raíz doble, cuando 2 términos de la ecuación son iguales a cero a un valor de *x*.

Se dice que hay una raíz triple, cuando 3 términos de la ecuación son iguales a cero a un valor de *x*.

Cuando la cantidad de raíces es impar, la función cruza al eje; cuando la cantidad es par, no lo cruza.

"Una raíz múltiple corresponde a un punto donde una función es tangencial al eje *x*, y varios valores de *x* hacen que *f(x)* sea cero."

Definimos una función nueva *U(x)*, dada por:

$$U(x) = \frac{f(x)}{f'(x)}$$

Se observa que la función *U(x)* tiene las mismas raíces que *f(x)*, entonces *U(x)* se vuelve cero en cualquier punto que *f(x)* es cero.

Suponiendo ahora que *f(x)* tiene una raíz múltiple en $x = c$ de multicidad *r*. Esto podría ocurrir, por ejemplo, si *f(x)* contiene un factor (x-c) . Entonces, podría fácilmente demostrarse que *U(x)* tiene una raíz en $x = c$ de multicidad *r*, o una raíz simple. Puesto que el método de Newton Raphson es efectivo para raíces simples, podemos aplicar el método de Newton para resolver *U(x)* en lugar de *f(x)*.

$$x_{i+1} = x_i - \frac{f(x_i)f'(x_i)}{[f'(x_i)]^2 - f(x_i)f''(x_i)}$$

derivando la función auxiliar U(x), x_n

$$U'(x) = 1 - \frac{f(x_n).f''(x)}{[f'(x)]^2}$$

Ya que este método está significantemente relacionado con el método de Newton-Raphson, cuando la derivada tiende a cero, tiene problema con la convergencia.

Cuando se tiene existencia de raíces múltiples, tanto el método de Newton-Raphson como el de la secante convergen linealmente.

El metodo de Newton-Raphson modificado el cual se describe acontinuacion consiste en aplicar el metodo de Newton-Raphson univariable dos veces (para el caso de un sistema de n ecuaciones no lineales con n incógnitas, se aplicara n veces), una para cada variable.

Cada vez que se hace esto, se considera las otras variables fijas. Considerese de nuevo el sistema

F1(x,y)=0

F2(x,y)=0

Tomando los valores iniciales x0, y0, se calcula a partir del método de Newton-Raphson univariable un nuevo valor x1 de la forma siguiente:

$$x^1 = x_0 - \frac{f_1(x_0, y_0)}{\frac{\partial f_1}{\partial x}}$$

$$y^1 = y_0 - \frac{f_2(x_1, y_0)}{\frac{\partial f_2}{\partial y}}$$

Este método converge a menudo si x_0,y_0 está muy cerca de x negada y y negada, y requiere la evaluación de solo 2n funciones por paso (cuatro para el caso de dos ecuaciones que se está manejando). Hay que observar que se han empleado

desplazamientos sucesivos, pero los desplazamientos simultáneos también son aplicables.

En la aplicación de este método se pudo tomar f_2 para evaluar x_1 y f_1, a fin de evaluar y_1, así:

Esto puede producir convergencia en alguno de los arreglos y divergencia en el otro. Es posible saber de antemano si la primera o la segunda forma convergirán para el caso de sistemas de dos ecuaciones, pero cuando 3 <= n las posibilidades son varias (n!) y es imposible conocer cuál de estos arreglos tiene viabilidad de convergencia, por lo cual la elección se convierte en un proceso aleatorio. Esta aleatoriedad es la mayor desventaja de este método.

En general, para un sistema de n ecuaciones con n incógnitas: $x_1, x_2 \ldots\ldots, x_n$, el algoritmo toma la forma:

$$x_i^{k+i} = x_i^k - \frac{f_i\left(x_1^{k+1}, x_2^{k+2}, \ldots., x_{i-1}^{k+1}, x_i^k, \ldots., x_n^k\right)}{\frac{\partial f_1}{\partial x}\left(x_1^{k+1}, x_2^{k+2}, \ldots., x_{i-1}^{k+1}, x_i^k, \ldots., x_n^k\right)} 1 \leq i \leq n$$

Ejemplos:

Tomaremos esta función:

$$f(x) = 3.5x^2 + 6.8x - 8$$

Método matemático:

Primero se calculan las derivadas de la función original

$$f(x_i) = 3.5x^2 + 6.8x - 8$$
$$f'(x_i) = 7x + 6.8$$
$$f''(x_i) = 7$$
$$[f'(x_i)]^2 = 49x^2 + 95.2x + 46.24$$

Para la raíz uno: $x_1 = 0.825621165$

$x_i = 1$

$$x_{i+1} = 1 - \frac{(2.3)(13.8)}{(190.44) - (2.3)(7)} = 0.8179419525$$

$$E = \frac{0.825621165 - 0.8179419525}{0.825621165}(100) = 0.93\%$$

$$x_i = 0.8179419525$$

$$x_{i+1} = 0.8179419525 - \frac{(-0.09639309119)(12.52559367)}{(156.8904967) - (-0.09639309119)(7)}$$

$$= 0.8256708834$$

$$E = \frac{0.825621165 - 0.8256708834}{0.825621165}(100) = 0.0062\%$$

Para la raíz dos:

Método matemático:

$$x_i = -3$$

$$x_{i+1} = -3 - \frac{(3.1)(-14.2)}{(201.64) - (3.1)(7)} = -2.755362899$$

$$E = \frac{-2.768478308 - (-2.755362899)}{-2.768478308}(100) = 0.4737407267\%$$

$$x_i = -2.755362899$$

$$x_{i+1} = -2.755362899 - \frac{(-0.1643812451)(-12.48754029)}{(155.9386626) - (-0.1643812451)(7)}$$

$$= -2.768430097$$

$$E = \frac{-2.768478308 - (-2.768430097)}{-2.768478308}(100) = 0.001741420185\%$$

RESULTADOS Y ERROR:

	Método Matemático	% Error
X_1	0.82563762	0.00616262
X_2	-2.76852652	0.00252

4.3.5. Método de la Secante

Este método se basa en la fórmula de Newton-Raphson, pero evita el cálculo de la derivada usando la siguiente aproximación:

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

Sustituyendo en la fórmula de Newton-Raphson, obtenemos:

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

$$\boldsymbol{x_{i+1} \approx x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}}$$

Que es la fórmula del método de la secante. Nótese que para poder calcular el valor de x_{i+1}, necesitamos conocer los dos valores anteriores x_i y x_{i-1}.

Obsérvese también, el gran parecido con la fórmula del método de la regla falsa. La diferencia entre una y otra es que mientras el método de la regla falsa trabaja sobre intervalos cerrados, el método de la secante es un proceso iterativo y por lo mismo, encuentra la aproximación casi con la misma rapidez que el método de Newton-Raphson. Claro, corre el mismo riesgo de éste último de no converger a la raíz, mientras que el método de la regla falsa va a la segura.

Ejemplo 01

Usar el método de la secante para aproximar la raíz de $f(x) = e^{-x^2} - x$, comenzando con $x_0 = 0, x_1 = 1$ y hasta que $|\epsilon_a| < 1\%$.

Solución

Tenemos que $f(x_0) = 1$ y $f(x_1) = -0.632120558$, que sustituimos en la fórmula de la secante para calcular la aproximación x_2:

$$x_2 = x_1 - \left[\frac{f(x_1)(x_0 - x_1)}{f(x_0) - f(x_1)}\right] = 0.612699837$$

Con un error aproximado de:

$$|\epsilon_a| = \left|\frac{x_2 - x_1}{x_2} \times 100\%\right| = 63.2\%$$

Como todavía no se logra el objetivo, continuamos con el proceso. Resumimos los resultados en la siguiente tabla:

Aprox. a la raíz	Error aprox.
0	
1	100%
0.612699837	63.2%
0.653442133	6.23%
0.652917265	0.08%

De lo cual concluimos que la aproximación a la raíz es:

$$x_4 = 0652917265$$

Ejemplo 02

Usar el método de la secante para aproximar la raíz de *f(x)=arctan x-2x-1*, comenzando con $x_0 = 0$ y $x_1 = 1$, y hasta que $|\epsilon_a| < 1\%$.

Solución

Tenemos los valores $f(x_0) = 1$ y $f(x_1) = -0.214601836$, que sustituimos en la fórmula de la secante para obtener la aproximación x_2:

$$x_2 = x_1 - \left[\frac{f(x_1)(x_0 - x_1)}{f(x_0) - f(\ \ )x_1}\right] = 0.823315073$$

Con un error aproximado de:

$$|\epsilon_a| = \left|\frac{x_2 - x_1}{x_2} \times 100\%\right| = 21.46\%$$

Como todavía no se logra el objetivo, continuamos con el proceso. Resumimos los resultados en la siguiente tabla:

Aprox. a la raíz	Error aprox.
0	
1	100%
0.823315073	21.4%
0.852330280	3.40%
0.853169121	0.09%

De lo cual concluimos que la aproximación a la raíz es:

$$x_4 = 0.853169121$$

4.4. Lenguaje de programación C++

El Lenguaje de Programación C++ fue creado por Bjarne Stroustrup en 1985 como una extensión del Lenguaje C, el mismo que es más consistente y conocido que otros lenguajes de programación científicos.

El Lenguaje de Programación C++ es de propósito general porque ofrece control de flujo, estructuras sencillas y un buen conjunto de operadores. Se puede utilizar en varios tipos de aplicación (Monografias.com, 2018).

Este Lenguaje ha sido creado estrechamente ligado al sistema operativo UNIX, puesto que fueron desarrollados conjuntamente. Sin embargo, este lenguaje no está ligado a ningún sistema operativo ni a ninguna máquina concreta (Monografias.com, 2018).

Sus características principales son:

- Es un Lenguaje que Incluye a versiones anteriores de Lenguaje C.
- Abstracción de Datos.
- Permite definir nuevos tipos de Datos.
- Permite la programación orientada a objetos.
- Varias funciones pueden compartir el mismo nombre.
- Permite definir una función como miembro de una estructura.
- Incluye Operadores para reservar y liberar memoria.

4.4.1. Operadores

Son símbolos que se utiliza para manipular datos. Existen los siguientes tipos de operadores:

a) Operadores de asignación:

Definición. Es el símbolo "=" que se utiliza para dar un valor a una variable.

Ejemplo

A = 3 ;	coloca directamente un valor
A = b ;	se le da un valor de otra variable
A=b=c=3;	da un valor a varias variables
A=b=c=d;	todos toman el valor de la variable d

b) Operadores aritméticos:

Definición. Son símbolos que indican los cálculos que se deben realizar sobre una o más variables dentro de una expresión.

+ suma	suma = a + b
++ Incrementar un uno	x = x + 1; // x = x++; // x +=1;
- resta	resta = a – b
-- Decremento en uno	x = x - 1; // x = x - -; // x -=1;
* Multiplicación	i = i * j ; // i * = j ;
/ División	x= x / 2; // x /=2;
% modulo	Resto de la división de enteros.

c) Operadores de comparación:

Definición. Son símbolos para indicar la relación que hay entre dos o más valores.

= ==	igual que, cumple sin son iguales
!=	distinto que , lo contrario de igual
> , <, >=, <=	mayor, menor, mayor o igual y menor o igual

d) Operadores lógicos:

Definición. Son símbolos que sirven para unir varias comparaciones.

&& &	and	(alt + 38)
|| |	or	(alt + 124)
!	not	

4.4.2. Tipos de datos

Definición 4.2.6 Son los atributos de una parte de los datos que indica al ordenador (y/o el programador) algo sobre la clase de datos sobre los que se va a procesar.

a) Enteros

Para números enteros con el siguiente rango:

int	-32768 .. 32767
long	-2147483648 .. 2147483647

b) Reales (Coma Flotante)

Para números reales con el siguiente rango:

float	-3.40E+38... -1.17E-37 para negativos
float	1.17E-37 ... 3.40E+38 para positivos
double	-1.79E+308 ... –2.22E-307 para negativos
double	2.22E-307 ... 1.79E+308 para positivos

c) Carácter

Para caracteres solos y cadenas de caracteres:

char [cantidad]

4.4.3. Constantes

Definición. Son aquellos datos que no pueden cambiar a lo largo de la ejecución de un programa.

Sintaxis:

Dentro de la Función Principal

Declarar la Variable

Variable = valor_según_declaración

Dentro de las Librerías de Cabecera

#define nombre _ constante Valor _ constante

Ejemplo

1) float pi ; pi = 3.14159
2) #define pi 3.14159

4.4.4. Variables

Definición. Son aquellos datos que puede cambiar su valor dentro de la ejecución del programa.

a) PRIMERA FORMA (Declaración Global): Estas se declaran después de las Librerías y sirven para todo el programa, incluyendo las funciones definidas por el usuario.

```
# include < conio.h >
# include < stdio.h >
tipo_dato  variables globales;
void main()
{
}
```

b) SEGUNDA FORMA (Declaración Local): Estas se declaran dentro de la función principal ó las funciones definidas por el usuario en el programa.

```
# include < conio.h >
# include < stdio.h >
void main()
{
tipo_dato  variables locales;
}
```

4.4.5. Entrada y salida de datos

Las Librerías que se activa para la entrada y salida de datos son:

```
#include < stdio.h >
#include < bcd.h >
```

a) ENTRADA:

- FUNCIÓN CIN : (# include < stdio.h > # include < bcd.h >)

Sintaxis:

```
cin >>apuntador de Variable;
cin >>Variable 1>>Variable 2;
```

Ejemplo

```
cin >> n1 >>n2 >>n3 ;
```

NOTA: No importa qué tipo de datos sea, no se tiene que especificar.

b) SALIDA:

- FUNCIÓN COUT: (# include < stdio.h > - # include< bcd.h >)

Sintaxis:

cout << " mensaje de Salida " ;

cout << " mensaje de Salida " <<Variable <<Variable;

cout << " mensaje de Salida \n " <<Variable << " \n" <<Variable;

Ejemplo

Cout << "los números son " << a << b <<c;

4.4.6. Instrucciones de control

Definición. Las Instrucciones de Control permiten que los programas sean más estructurados y puedan de esta forma solucionar todo tipo de problemas (Joyanes Aguilar, 1994).

Estas instrucciones pueden ser:

Selectivas	(if, if..else, switch)
Repetitivas	(for, while y do..while)
De Salto	(break, continue, goto)

a) SELECTIVAS:

Definición. Se realizan mediante las instrucciones **if, if..else, switch** y permiten evaluar una condición o expresión y en función del resultado de la misma se realizara una acción u otra.

a.1. IF

Toma una decisión referente al bloque de sentencias a ejecutar en un programa, basándose en el resultado (Verdadero o falso) de una Condición.

1ra Sintaxis:

if (Condición _ verdadera)

Sentencias Ejecutables;

2da Sintaxis:

if (Condición)

{ .....

Sentencias Ejecutables;

.....}

else

{.....;

Sentencias Ejecutables

.....;}

Observación

Si la condición a evaluar en el IF necesita más opciones colocar:

&& (and) || (or)

Ejemplo

- Condición para encontrar números pares

```
if (N % 2 ! = 1)
cout <<" NUMERO PAR"
```

- Condición para los números negativos o números cero

```
if ((N < 0) || (N==0))
cout <<" NUMERO NEGATIVO O CERO"
```

- Condición para las personas mayores de edad y que ganen 500 soles, aumentarle 100 soles de lo contrario disminuirle 200.

```
if ((SUELDO==500) && (EDAD<17))
{       SUELDO=SUELDO + 100 ;
cout<<" SU NUEVO SUELDO ES : " << SUELDO ;
}
else
{       SUELDO=SUELDO - 200 ;
cout<<" SU SUELDO SIGUE SIENDO :" << SUELDO ; }
```

a.2. SWITCH

Esta sentencia permite ejecutar una de varias acciones, en función del valor de una expresión.

Sintaxis:

```
switch (EXPRESION)
{
case 1 : Sentencias Ejecutables;
        getch(); break;
case 2 : Sentencias Ejecutables;
        getch(); break;
case 3 : Sentencias Ejecutables;
        getch(); break;
case 4 : Sentencias Ejecutables;
        getch(); break;
        }
```

Reglas para el uso del Switch :

1. La EXPRESION puede ser una variable o constante.
2. No funciona con datos Flotantes (Reales).
3. El valor en cada CASE debe ser Entero o 'Carácter '.
4. C++ no soporta varios casos en el mismo lugar paras esto se utiliza:

   Case 1:

   Case 2:
5. Necesita utilizar la sentencia BREAK al finalizar cada CASE. Break hace que la ejecución del programa continúe después del SWITCH.
6. Si no se coloca SWITCH la ejecución del programa se reanuda en las demás CASE.
7. El conjunto de sentencias de cada CASE no necesita encerrarse entre llaves.

b) <u>REPETITIVAS</u>:

Definición. Estas instrucciones hacen que una sección del programa se repita un cierto número de veces. La repetición continua mientras una condición sea verdadera, cuando la decisión es falsa el bucle termina y el control pasa a las

sentencias a continuación del bucle, existen tres clases de Bucles **for, while y do..while.**

b.1. FOR

El bucle FOR, ejecuta una sección de código un número fijo de veces.

Sintaxis:

```
for ( exp1 ; exp2 ; exp3 )
        Sentencia Ejecutables;
```

```
for ( exp1 ; exp2 ; exp3 ) for ( exp1, exp A; exp2, exp B; exp3, Exp B )
{       .....;                  {       .....;
Sentencia Ejecutables;          Sentencia Ejecutables;
  .....;                                .....;
}                               }
```

Donde:

- exp1 : Es el Valor inicial (Signo =)
- exp2 : Condición(Signo <=, >=, <, >)
- exp3 : Es el Operador de incremento o decremento (i++, i--)

Ejemplo

1. Imprimir los 10 primeros números.

```
int i ;
for ( i = 1 ; i < = 10 ; i + + )
        cout << "\n" << i ;
```

2. Imprimir los 20 primeros numero en forma descendente

```
int i ;
for ( i = 20  ; i >= 1 ; i - - )
        cout << "\n" << i ;
```

b.2. WHILE

El bucle while ejecuta un bloque de sentencias ejecutables mientras su condición sea verdad.

Sintaxis:

```
while ( condicion_verdadera )
{       .....;
        Sentencias Ejecutables;
        ..... ; }
```

Ejemplo

1. Imprimir los 20 primeros numero enteros ascendente

```
int c=1;
while ( c < = 20 )
        { cout << "\n"  << c ;
                c + +  ;
        }
```

2. Imprimir los 20 primeros enteros en forma descendente

```
int  c = 20 ;
while ( c > = 1 )
{ cout << "\n"  << c ;
        c - - ;   }
```

Observación

¿Cómo ingresar una cadena, si variable se declara como Char?

Utilizar **GETLINE** (VARIABLE, CANTIDAD_DECLARADA);

```
char   nombre[40] ;
        cout << " ingrese su nombre completo :" ;
        cin.getline(nombre,40);
```

¿Cómo controlar la cantidad de decimales de una respuesta ?

Utilizar **#include** < iomanip . h >

```
SETPRECISION (CANTIDAD_DE_DECIMALES)
float  raiz_cuadrada , n ;
cout << " ingrese un numero entero:" ; cin >> n;
raiz_cuadrada =  pow ( n , 0.5 );
cout << "La raiz es :" << setprecision (2) << raiz_cuadrada ;
```

¿Cómo imprimir varias repuestas separados por un espacio determinado?

Utilizar **#include** < iomanip . h >

SETW (CANTIDAD_DE_ESPACIOS)

Cout << a << **setw(5)**<< b << **setw(5)** << c ; // Ancho

Cout << a << " \t " << b << " \t " << c ; //tabulador

b.3. DO... WHILE

Ejecuta un bloque de sentencias, una o más veces, dependiendo de la condición que tiene que ser verdadera.

Sintaxis:

```
do
    {
        ..... ;
        SENTENCIAS EJECUTABLES;
        ..... ;
    } while ( CONDICIÓN _ VERDADERA);
```

Ejemplo 01

Ingresar números y terminar cuando se ingrese un número negativo devolviendo la suma.

```
SUMA = 0 ;
cin >> N ;
do
{       SUMA + = N ;
     cin >> N ;
} while ( N  > 0 );
```

Ejemplo 02

Para colocar ¿desea continuar s/n? Para retornar a un programa.

```
char OP;
do
{       clrscr();
        cout <<" INGRESE UN NUMERO : " ;
```

```
cin >> N ;
cout <<" DESEA SEGUIR S/N : " ;
cin >> OP ;
} WHILE ( OP == ' S ' ) ;
```

c) DE SALTO:

Definición. Estas instrucciones hacen que en un determinado momento de la ejecución del programa se traslade a otra parte o abandone una instrucción repetitiva y pueda continuar con el resto del programa, existen tres clases: **break, continue y goto.**

c.1 BREAK

Hace finalizar la ejecución del Ciclo ó bucle : DO, FOR, SWITCH, WHILE mas interno que lo contenga.

Sintaxis:

```
break;
```

BREAK solamente finaliza la ejecución de la sentencia de donde esta incluida.

Ejemplo

Imprimir los 4 primeros números enteros

```
n=1;
        while( n < 5 )
        {
                cout<< n ;
                if ( n == 4)
                        break;
                else
n++;
        }
```

c.2. CONTINUE

Traslada la ejecución del programa a la ultima línea del Ciclo ó bucle : DO, FOR, SWITCH, WHILE mas interno que lo contenga.

Sintaxis:

```
continue;
```

Ejemplo

Imprimir los números entre 1 y 50 que no sean múltiplos de 5

```
n=1;
    while( n < 50 )
    {
        if ( n % 5 == 0)
            continue;
        else
            cout<< n ;
    n++;
    }
```

c.3. GOTO

Traslada la ejecución del programa a una línea específica identificada por una etiqueta.

Sintaxis:

```
goto etiqueta;
    .....
    .....
    Etiqueta: bloque de SENTENCIAS;
```

Ejemplo

Calcular el área de un triángulo

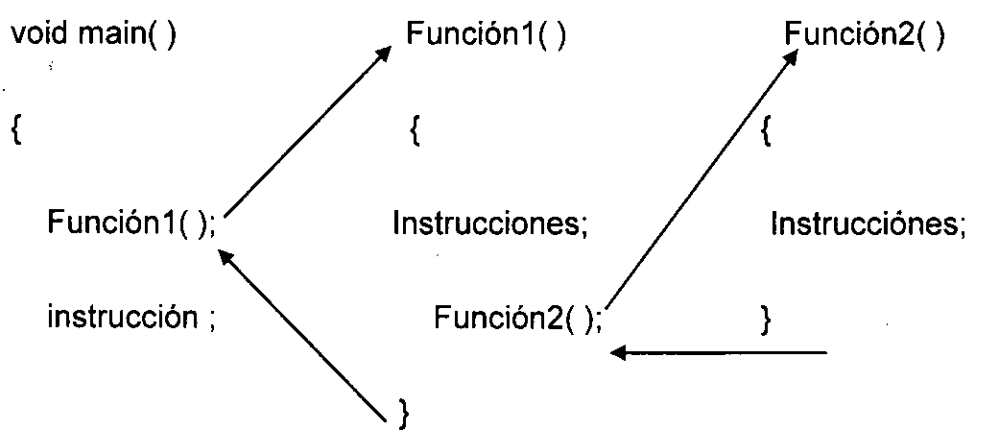
```
Inicio: { cout<<" Ingrese la base del triángulo:";
        cin >>b;
        cout<<"Ingrese la altura del triángulo:";
        cin >>a;
        cout<<"El área del triángulo es : "<<(b*a)/2;
                }
        if (a>0)  goto  Inicio;
        cout<<"FIN DEL PROGRAMA";
```

4.4.7. Funciones

Definición. Es una colección independiente de declaraciones y sentencias, generalmente realiza una tarea específica, En C++ existe una función por naturaleza y es el programa principal:

```
void main ( )
{
}
```

Cuando se llama a una función el control pasa a él mismo para su ejecución, una vez finalizado el trabajo de la función devuelve el control a la función que lo llamó.



Observación

Una función no pertenece al programa

- En el programa sólo se le llama.
- El cuerpo de una función es la parte funcional de un programa.
- Una función siempre debe estar fuera de la función principal main( ).

Pasos para definir una función:

1. Tener en cuenta las variables globales o locales.
2. En C++ se debe declarar una función antes de utilizarla, colocándolo el encabezado de la función.

int suma (int a, int b) ;

3. Comenzar con la función.

Sintaxis:

```
Tipo nombre (tipo variable de entrada)
{
        Declaraciones de variables locales;
        Sentencias ejecutables;
        return [variable;]      // Variables de Salida
}
```

Donde:

- **TIPO:** Indica el tipo de Datos que devolverá la función luego de su uso (no es necesario, solo en casos que se quiera hacer referencia a la salida).
- **NOMBRE**: Es el nombre de la función, trata de no tenerlo como variable.
- **TIPO VARIABLE DE ENTRADA:** Aquí se colocan las Variables con su respectivo tipo de Datos que entraran a la función para su posterior uso, reemplazando a la variable verdadera.

Si dentro de la función se hace uso de variables que no pertenecen a la función MAIN solo se declararan dentro de la función, como una variable local.

Sintaxis:

```
        suma ( );                   → función sin argumentos
int     suma ( ) ;
int     suma ( int  a ) ;           → funcion con 1 entrada
int     suma ( int a,int b) ;       → funcion con 2 entradas
int suma ( int a, float c, int b)   → funcion con 3 entradas
```

A. VALOR RETORNADO POR UNA FUNCIÓN – SENTENCIA RETURN

Indicara que variable retornara con el valor, solo puede haber un solo retorno y en algunos casos no se coloca valor de retorno cuando se trata de varios valores.

Sintaxis:

return variable;

Observación

- Esta variable debe estar declarado dentro de la función o como variable global.
- En el valor de retorno también se le puede alterar su devolución

  return (N)

  return (N + 2)

  return (N * 2)

B. LLAMADA A LA FUNCIÓN

Una llamada se realiza para que la función tenga efecto.

Sintaxis:

[Variable =] Nombre_de_la_función (parámetros o argumentos)

Ejemplo

```
Suma ( )
Cout << suma ( )
Su = suma ( )
```

4.4.8. Librería de funciones creadas por el usuario

Definición. Son un programa fichero que tiene las funciones que serán compiladas en forma separada y pueda ser utilizado en cualquier programa de C++.

Pasos:

1. Se crean solo funciones dentro de un programa, indicando en la parte superior los prototipos de las funciones y los ficheros.

2. Luego, una vez terminado la función grabarlo con un nombre y extensión (**.hpp**)

   Ejemplo

   Sumatoria. hpp

   Factorial. hpp

Observaciones:

a) Se pueden colocar varias funciones dentro de este programa(**.hpp**)
b) No ejecutar un programa **hpp**, mas bien cerrarlo.
c) Dentro de un programa **cpp**, llamar a la función que se encuentra dentro del programa **hpp**.

Sintaxis:

En la Cabecera del programa CPP :

```
# include " NombredelFichero.hpp "
```

4.4.9. Arreglos

Definición. Los arreglos son estructuras de datos, que permiten el almacenamiento masivo de información.

Un arreglo está compuesto por varios componentes, todas del mismo tipo y almacenados consecutivamente en la memoria de la PC.

Ventajas de usar arreglos:

- Almacenamiento de datos en una sola variable.
- A la vez cada dato es independiente de los demás.
- Se puede acceder a cada elemento indicando el número de índice.
- Se pueden manipular con operaciones de cualquier tipo.

Los Arreglos se presentan en dos formas:

A. Vectores o arreglos unidimensionales.

Definición. Son listas de datos que pueden almacenar un número determinado de datos.

	0	1	....	n
NOMBRE DEL ARREGLO	Dato 1	Dato 2	.....	Dato N

Un vector está compuesto:

Nombre del Arreglo

Índice : 0, 1, 2, 3,... n (Utilizar Estructuras Repetitivas)

Datos : Son los datos ingresados al arreglo de un mismo tipo.

Ejemplo

	0	1	2	3	4
NOTAS	08	09	10	11	10
NOMBRES	ANA	LUISA	LIZA	JANET	ELISA
PROMEDIOS	10.5	11.50	3.45	19.55	0.025

Para acceder a cada elemento solo se escribirá el nombre de la variable arreglo y el índice.

- Para acceder a Janet → NOMBRES [3]
- Para acceder a 3.45 → PROMEDIOS [2]

Sintaxis:

TIPO_DE_DATOS NOMBRE_DEL_ARREGLO [TAMAÑO_APROX] ;

Observación

Si el tipo de datos del arreglo es de caracteres su declaración será:

CHAR nombre_del_arreglo **[tamaño_aprox] [cantidad_de_espacios];**

Ejemplo

Declarar el ingreso de 5 notas enteras:

```
int notas [ 5 ] ;
```

Declarar el ingreso de 5 nombres de persona:

```
char nombres [ 5 ] [ 20 ];
```

B. MATRICES O ARREGLOS BIDIMENSIONALES

Definición. Son Arreglos que tienen 2 dimensiones.

	1	2	3	4
1	10	8	13	11
2	10	8	13	11
3	10	8	13	11
4	10	8	13	11

Una Matriz está compuesta:

Nombre del Arreglo

Índice filas :0, 1, 2, 3, ... n(utilizar estructuras repetitivas)

Índice Columnas :0, 1, 2, 3, ... n(utilizar estructuras repetitivas)

Datos :Son los datos ingresados al arreglo de un mismo tipo.

Ejemplo

Nombre del Arreglo: NÚMERO

	1	2	3	4
1	A	E	I	M
2	B	F	J	N
3	C	G	K	O
4	D	H	L	P

Para acceder a cada elemento solo se escribirá el nombre de la variable arreglo y los índices.

- Para acceder a la Letra A (NÚMERO [1][1])
- Para acceder a la Letra L (NÚMERO [4][3])

Sintaxis:

TIPO_DATOS NOM_DEL_ARREGLO [TAMAÑO_FILAS] [TAMAÑO_COLUMNAS];

Observación

Al hacer referencia de cada elemento de un arreglo Bidimensional se coloca el nombre del arreglo luego la fila y la columna.

Ejemplo

Ingresar un arreglo Bidimensional de 3 x 3 y determinar la suma de todos sus Elementos.

```
for( I = 1 ; I < = 3 ; I + + )
for( j = 1 ;j < = 3 ; j + + )
    Cin>> números [I] [j];
for( I = 1 ; I < = 3 ; I + + )
for( j = 1 ;j < = 3 ; j + + )
    suma=suma+números[ I ] [ j ] ;
Cout<<"la suma es " << suma ;
```

V. MATERIALES Y MÉTODOS

5.1. Materiales

- Material de Oficina: papel bond carta, lapiceros, liquid-paper, calculadora.
- Material bibliográfico: Libros de la especialidad de Estadística y el Lenguaje de programación C++.
- Material de cómputo: BORLAND C++, computadora e impresora.

5.2. Método

La elaboración del presente trabajo de investigación demando del suscrito ordenar información recopilada durante su vida profesional como docente del curso de investigación de operaciones en la facultad de ciencias naturales y matemática.

Para cada método para resolver las Ecuaciones no lineales materia de esta investigación se ha resuelto ejercicios donde se puede visualizar como trabaja este sistema computacional.

El método empleado en este trabajo de investigación es inductivo y deductivo que ha hecho posible interpretar el formalismo de la teoría, así como también para analizar las soluciones y poder implementar un sistema computacional.

VI. RESULTADOS

El resultado de la presente investigación es un sistema computacional con los métodos para resolver las Ecuaciones no lineales, el mismo que se adjunta con este informe.

6.1. Librería para todas las aplicaciones

Librería (NOLINEAL.HPP)

```
#include<conio.h>
#include <math.h>
#include <bcd.h>
void puntofijo();
void biseccion();
void newton();
void newtonm();
void secante();
double f(double x);
void evaluapoli(int n, double *f, double x, double &p);
void derivapoli(int n, double *f, double *fd);
void deriva2poli(int n, double *f, double *fd2);

void puntofijo()
{
int it, i, j, z, k, coe, l, f, n, u, op, sw, maxit;
int g[20],N[20],P[20];
double xi,gxi,e,p,b,a,x,fa,fb,fp,tol,x0;
clrscr();
cout<<"Ingrese el polinomio a desarrollar: \n";
cout<<"Ingrese el Grado del polinomio:";
cin>>n;

for(i=0;i<=n;i++)
{
```

```
cout<<"Ingrese el coeficiente "<<i<<":";
cin>>g[i];
}
clrscr();
cout<<"El polinomio es:\n";
for(j=0;j<=n;j++)
{
cout<<"("<<g[j]<<")"<<"x^"<<j<<"+";
}
cout<<"\n Ingrese el valor de x0:";
cin>>x0;
cout<<"\n Ingrese el Máximo de iteraciones:";
cin>>maxit;
cout<<"\n Ingrese tolerancia:";
cin>>tol;
sw=0;
it=0;
while((sw==0)&&(it<=maxit))
{
gxi=0;
for(j=0;j<=n;j++)
{
gxi=gxi+g[j]*pow(x0,j);
}
xi=gxi;
e=fabs(xi-x0);
if(e<=tol)
{
sw=1;
}
else
```

```
{
it=it+1;
x0=xi;
}
cout<<it<<"\t"<<x0<<"\t"<<gxi<<"\t"<<e<<"\n";
}
if (sw==1)
{
cout<<"La raíz es: "<<xi;
}
else
{
cout<<"No Converge";
}
getch();
return;
}

void biseccion()
{
int i,j,n,maxit,nolter;
double a,b,x,tol,c,g[10];
clrscr();
cout<<"Ingrese el polinomio a desarrollar \n";
cout<<"Ingrese el Grado del polinomio:";
cin>>n;
for (i=0;i<=n;i++)
{
cout<<"Ingrese el coeficiente "<<i<<":";
cin>>g[i];
}
```

```
clrscr();
cout<<"El polinomio es:\n";
for(j=0;j<=n;j++)
cout<<g[j]<<"x^"<<j<<"+";
cout<<"\n Ingrese valores a,b del intervalo [a,b]:";
cin>>a>>b;
cout<<"\n Ingrese el Máximo de iteraciones:";
cin>>maxit;
cout<<"\n Ingrese tolerancia:";
cin>>tol;
      int nolter=0;
      do
      {
         c=(a+b)/2;
         if (f(a)*f(c)<0)
         {
           b=c;
         }
         else
         {
           a=c;
         }
    nolter++;
       }while((abs(f(c))>tol)&&(nolter<maxit));
cout<<"La raiz es:"<<c;
getch();
return;
}


double f (double x)
```

```
{
double suma;
suma=0;
for (i=0; i<=n; i++)
  suma=suma+g[i]*pow(x,i);
return(suma);
}

void newton()
{
 int n,i;
 double f[10],fd[10],tol,p=0,y1,y2,verificador,z = 0;
 clrscr();
 cout << "El grado del polinomio y sus coeficientes\n\n";
 cout<< "ingrese el grado = ";
 cin>>n;
 cout << "Coeficientes del polinomio\n\n";
 for (i = n; i > -1; i--)
 {
 cout << "F" << i << " = ";
 cin>>f[i];
  }
  cout << "\n";
  derivapoli (n, f, fd);
  cout << "Coeficientes del polinomio primera derivada\n\n";
  for (i = n - 1; i > -1; i--)
  {
  cout << "Fd" << i << " = "<<fd[i]<<"\n";
  }
  cout << "\n";
  cout << "Introduzca el valor de inicio para la raiz: \n" << endl << "X = ";
```

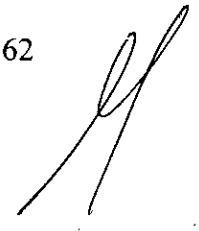
```
double x;
int contador = 0;
cin >> x;
cout << endl;
cout<<"Ingrese la tolerancia:";
cin>>tol;
do{ evaluapoli(n, f, x, p);
 y1 = p;
 evaluapoli(n-1, fd, x, p);
 y2 = p;
 x = x - y1/y2;
 verificador = fabs(x - z);
 z = x;
 contador += 1;
 } while (verificador > tol);
 cout << "La raiz es x="<<x<< " y se obtuvo en " << contador << " iteraciones\n";
 getch();
 return;
}

void evaluapoli(int n, double *f, double x, double &p)
{
 int i;
 double *a;
 a = new double [n];
 for (i = n; i > -1; i--)
  {
   a[i] = f[i];
  }
 for (i = n; i > 0; i--)
  {
```

```
 p = a[i] * x + a[i - 1];
 a[i - 1] = p;
 }
}

void derivapoli(int n, double *f, double *fd)
{
int i;
for (i = n; i > 0; i--)
{
fd[i-1] = i * f[i];
}
}

void newtonm()
{
 int n,i;
 double p=0, y1, y2, y3,f[10], fd[10], fd2[10], tol, verificador, z = 0;
clrscr();
cout << "El grado del polinomio y sus coeficientes\n\n";
 cout<< "ingrese el grado = ";
 cin>>n;
 double ;
 cout << "Coeficientes del polinomio\n\n";
for (i = n; i > -1; i--)
{
cout << "F" << i << " = ";
cin>>f[i];
 }
 cout << "\n";
 derivapoli (n, f, fd);
```

```
cout << "Coeficientes del polinomio primera derivada\n\n";
for (i = n - 1; i > -1; i--)
{
cout << "Fd" << i << " = "<<fd[i]<<"\n";
}
cout << "\n";
deriva2poli (n-1, fd, fd2);
cout << "Coeficientes del polinomio segunda derivada\n\n";

for (i = n - 2; i > -1; i--)
{
cout << "Fdd" << i << " = "<<fd2[i]<<"\n";
}
cout << "\n";
cout << "Introduzca el valor de inicio para la raiz: \n" << endl << "X = ";
double x;
int contador = 0;
cin >> x;
cout << endl;
cout<<"Ingrese la tolerancia:";
cin>>tol;
do {
 evaluapoli(n, f, x, p);
 y1 = p;
 evaluapoli(n-1, fd, x, p);
 y2 = p;
 evaluapoli (n-2, fd2, x, p);
 y3 = p;
 x = x - ((y1*y2)/(pow(y2,2)-(y1*y3)));
 verificador = fabs(x - z);
 z = x;
```

```
 contador += 1;
 } while (verificador > tol);
 cout << "La raiz es x="<< x << "y se obtuvo en " << contador << " iteraciones\n";
 getch();
 return;
}
void deriva2poli(int n, double *fd, double *fd2)
{
int i;
for (i = n; i > 0; i--){
fd2[i-1] = i * fd[i];
}
return;
}

void secante()
{
int i, j, z, k, coe, l, f, n, u, op, sw, maxit;
int V[20],P[20];
double xi,xa,fxi,fx1,e,p,b,a,x,fa,fb,fp,tol,x0,x1;
clrscr();
cout<<"Ingrese el grado del polinomio:";
cin>>n;
cout<<"ingrese los coeficientes:\n";
for (i=0;i<=n;i++)
{
cout<<"x^"<<i<<":";
cin>>V[i];
}
cout<<"Tolerancia:";
cin>>tol;
```

```
cout<<"Ingrese el valor del p inicial x0:";
cin>>x0;
cout<<"Ingrese el valor del segundo punto:";
cin>>x1;
cout<<"Ingrese el número máximo de iteraciones:";
cin>>maxit;
z=0;
sw=0;
while((sw==0)&&(z<=maxit))
{
fxi=0;
for (j=0;j<=n;j++)
  fxi=fxi+V[j]*pow(x0,j);
fx1=0;
for (k=0;k<=n;k++)
 fx1=fx1+V[k]*pow(x1,k);
xi=x1-(fx1*(x1-x0))/(fx1-fxi);
e=fabs(xi-x1);
cout<<z<<" "<<x0<<" "<<x1<<" "<<fxi<<"  "<<fx1<<"  "<<xi<<"  "<<e<<"\n";
if (e<=tol)
sw=1;
else
{
z=z+1;
x0=x1;
x1=xi;
}
}

if (sw==1)
cout<<"x="<<xi;
```

```
else
cout<<"No Converge";
getch();
return;
};
```

6.2. Aplicación completa

Programa (menunolineal.cpp)

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <bcd.h>
#include <stdlib.h>
#include"NOLINEAL.hpp"

void main()
{
do
{
clrscr();
gotoxy(30,7);cout<<"MENU METODOS PARA ECUACIONES NO
LINEALES";
gotoxy(30,8);cout<<"*****************";
gotoxy(30,10);cout<<"1.-PUNTO FIJO";
gotoxy(30,12);cout<<"2.-BISECCION";
gotoxy(30,14);cout<<"3.-NEWTON";
gotoxy(30,16);cout<<"4.-NEWTON MODIFICADO";
gotoxy(30,18);cout<<"5.-SECANTE";
gotoxy(30,20);cout<<"6.-SALIDA";
gotoxy(35,22);cout<<"su opcion es:";
cin>>op;
switch(op)
```

```
{
case 1:{clrscr();
  puntofijo();
getch();
  break;
     }
case 2:{
  clrscr();
  biseccion();
  getch();
  break;
  }
case 3:{
  clrscr();
  newton();
  getch();
  break;
  }
case 4:{
  clrscr();
  newtonm();
  getch();
  break;
  }
case 5:{
  clrscr();
  secante();
  getch();
  break;
  }
case 6:{
```

```
  clrscr();
  cout<<"SALIENDO DEL MENU";
  op=6;
  getch();
  break;
  }
default:{
  clrscr();
  cout<<"opcion incorrecta\a\a\a\a\a\a\a";
  }
};
}while(op!=6);
}
```

VII. DISCUSIÓN

En los compiladores del Lenguaje de Programación C++, tales como: Borland C++, Microsoft C++ o Visual C++ no tienen una librería para calcular los métodos para resolver las ecuaciones no lineales, la bibliografía acerca de este tema es muy escasa.

Existen software para aplicar al área de las ecuaciones no lineales tales como: SOLVESYS y MATLAB que pueden resolver las ecuaciones no lineales.

El sistema propuesto en este trabajo de investigación está desarrollado en el lenguaje de programación C++ y por consiguiente tiene una mejor consistencia y ciclo de vida útil.

El resultado obtenido en este trabajo de investigación es un sistema computacional que tiene una librería con los métodos para resolver las ecuaciones no lineales.

VIII. REFERENCIALES

- Borr, C. y Conte, S. D. (1974) *Análisis Numérico*. México: McGraw-Hill, 2ª edición.
- Burden, R. L. y Faires, J. D. (2002) *Análisis Numérico*. México: Thomson Learning, 7ma edición, 2002.
- Cevallos S., F. (1997) *Curso de programación C++*. Madrid, España: RA-MA, 1ra edición, 1997.
- Domínguez, F. y Nieves, A. (1995) *Métodos Numéricos*. México: CECSA.
- Ford, W. y TOPP, W. (2002) *Data structures with C++*. México: Prentice – Hall, 2ª edición.
- Guardia, L. (2016) *Programación: Lenguajes*. Recuperado de https://www.monografias.com/docs110/programacion-lenguajes/programacion-lenguajes.shtml#introducca
- Joyanes A., L. (1994) *C++ a su alcance*, Madrid, España: McGraw-Hill, 1ra edición.
- Kincaid D.,C. W. (1994) *Análisis Numérico: Las matemáticas del cálculo científico*. México: Addison-Wesley Iberoamérica.
- López, R. J. y maron, M. J. (1998) *Análisis Numérico*. México: Continental, 1ra reimpresión.
- Mathews, J. (1992) *Numerical Methods*. México: Prentice-Hall,1ª edición.
- Musser, D.R. (1997) *Effective C++*. Madrid, España: Addison – Wesley
- Stroustrup, B. (2000) *The C++ Programming Language*. Madrid, España: Addison-Wesley, 3ra edición.
- Stroustrup, B. (2002) *El Lenguaje de programación C++*. Madrid, España: Addison-Wesley, 3ra edición.
- Stroustrup, B. (1994) *The Design and Evolution of C++*. Madrid, España: Addison-Wesley, 1ra edición.
- Wikipedia (2016) *Artículo de Sistema No Lineal*. Recuperado de https://es.wikipedia.org/wiki/Sistema_no_lineal

IX. APÉNDICES

MÉTODOS NUMÉRICOS PARA RESOLVER ECUACIONES NO LINEALES

MÉTODOS	FÓRMULA
PUNTO FIJO	Este método se aplica para resolver ecuaciones de la forma $x_{i+1} = g(x_i)$ Si la ecuación es f(x)=0, entonces puede despejarse x ó bien sumar x en ambos lados de la ecuación para ponerla en la forma adecuada.
BISECCIÓN	Sea $f(x)$ continua en un intervalo $[a,b]$ y supongamos que $f(a) < f(b)$. Entonces para cada z tal que $f(a) < z < f(b)$, existe un $x_0 \in (a,b)$ tal que $f(x_0) = z$. La misma conclusión se obtiene para el caso que $f(a) > f(b)$ La primera aproximación a la raíz se toma igual al punto medio entre x_a y x_b: $x_r = \frac{x_a + x_b}{2}$
NEWTON	La fórmula iterativa de Newton para calcular la siguiente aproximación: $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$ Donde: $f'(x_i) \neq 0$
NEWTON MODIFICADO	La fórmula iterativa de Newton modificado para calcular la siguiente aproximación: $Xi+1 = Xi - \frac{f(Xi)(f'(Xi))}{[f'(Xi)]^2 - f(Xi)f''(Xi)}$
SECANTE	En esta fórmula iterativa de la Secante se necesita dos valores iniciales: $x_{i+1} \approx x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$

X. ANEXOS

ANEXO 01: MATRIZ DE CONSISTENCIA

PROBLEMA	OBJETIVOS	HIPOTESIS	VARIABLES	METODOLOGÍA
¿ES POSIBLE APLICAR LENGUAJE DE PROGRAMACIÓN C++ A LAS ECUACIONES NO LINEALES?	**Objetivo general:** - Crear un sistema computacional en Lenguaje de Programación C++ para resolver Ecuaciones no lineales. **Objetivo específicos:** -Implementar el método del punto fijo con una aplicación desarrollada en Lenguaje de Programación C++. - Implementar el método de bisección con una aplicación desarrollada en Lenguaje de Programación C++. - Implementar el método de newton con una aplicación	**Hipótesis general:** - Existe evidencia que se puede mejorar la enseñanza de la solución de las ecuaciones no lineales utilizando un Sistema Computacional. Se pretende crear un Sistema Computacional y probar la eficiencia de sus aplicaciones, resolviendo problemas y comparando los resultados con los de otros software. **Hipótesis específicas:** - Crear una función para el método del punto fijo en	**Definición de las variables:** En la presente investigación los datos ingresados son la variable independiente y el valor de cada una de las funciones para los métodos numéricos a implementar en el Lenguaje de programación C++ son las variables dependientes. **Operacionalización de variables:** Los datos son escalares de tipo real o	**Tipo y diseño de la investigación:** La investigación es aplicada y el Diseño de investigación no experimental. La Metodología a emplear será el enfoque inductivo (introducir definiciones, observaciones, etc). Así como también el enfoque debe ser deductivo (deducir la demostración de las funciones a través de un algoritmo implementado en Lenguaje de Programación C++).

	desarrollada en Lenguaje de Programación C++. - Implementar el método de newton modificado con una aplicación desarrollada en Lenguaje de Programación C++. - Implementar el método de la secante con una aplicación desarrollada en Lenguaje de Programación C++.	Lenguaje de Programación C++. - Crear una función para el método de bisección en Lenguaje de Programación C++. - Crear una función para el método de newton en Lenguaje de Programación C++. - Crear una función para el método de newton modificado en Lenguaje de Programación C++. - Crear una función para el método de la secante en Lenguaje de Programación C++.	vectores (variable independiente) y el valor que devuelven las funciones de los métodos numéricos que solucionan las ecuaciones no lineales son escalares de tipo de real o vectores (variable dependiente).	