Catálogo Grupal de Algoritmos

Integrantes:

- Josué Araya García - Carnet
- Jonathan Guzmán Araya - Carnet
- Mariano Muñoz Masís - Carnet
- Daniel Prieto - Carnet

1. Tema 1: Ecuaciones no Lineales

1.1. Método 1: Bisección

Código 1: Lenguaje M.

```
%{
    Metodo de la Biseccion
    Parametros de Entrada
    @param f: funcion a la cual se le aplicara el algoritmo
    @param a: limite inferior del intervalo
    @param b: limite superior del intervalo
    @param MAXIT:  iteraciones maximas
    @param TOL:  tolerencia del algoritmo

    Parametros de Salida
    @return xAprox: valor aproximado de x
    @return error: porcentaje de error del resultado obtenido
%}

clc;
clear;

function [xAprox, iter] = biseccion(f, a, b, MAXIT, TOL)

    if(f(a) * f(b) < 0)

        iter = 1;
        iterl = [];
        err = 0;
        errl = [];

        while(iter < MAXIT)
            xAprox = (a + b) / 2;
            fx = f(xAprox);

            if(f(a) * fx < 0)
```

```
            b = xAprox;
        elseif(f(b) * fx < 0)
            a = xAprox;
        elseif(abs(fx) < TOL)
            return;
        endif

        iterl(iter) = iter;
        errl(iter) = err;

        iter = iter + 1;
        err = (b - a)/ (2);

        plot(iterl, errl);
        title("Metodo de la Biseccion");
        xlabel("Iteraciones");
        ylabel("% Error");

  endwhile
else
    error("Condiciones en los parametros de entrada no garantizan el cero de la funcion.")
endif
return;
endfunction

a = 0;
b = 2;
MAXIT = 100;
TOL = 0.000001;
funct = @(x) e^x - x - 2;
[xAprox, iter] =  biseccion(funct, a, b, MAXIT, TOL);
printf('xAprox = %f\nIteraciones = %i \n', xAprox, iter);
```

Código 2: Lenguaje Python.

```
# Metodo de Newton-Raphson
# Entradas:
            # func: es la funcion a analizar
            # x0: valor inicial
            # MAXIT: es la cantidad de iteraciones maximas a realizar
            # TOL: es la tolerancia del algoritmo
# Salidas:
            # xAprox: es la solucion, valor aproximado de x
            # error: pocentaje de error del resultado obtenido

################################################################################
import math
import matplotlib.pyplot as plt
from scipy.misc import derivative
import numpy as np
import sys
################################################################################

def newtonRaphson(func, x0, MAXIT, TOL):
    itera = 0
```

```
    err = 1
    iterl = []
    errl = []
    xAprox = x0

    while (err > TOL):
        xk = xAprox
        fd = derivative(func, xk, dx=1e-6)
        xAprox = xk - (func(xk)) / (fd)

        err = (abs(xAprox - xk)) / (abs(xAprox))

        iterl.append(itera)
        errl.append(err)

        itera = itera + 1

    plt.plot(iterl, errl)
    plt.xlabel("Iteraciones")
    plt.ylabel("% Error")
    plt.title("Metodo de Newton-Raphson")
    plt.show()
    return xAprox, itera


if __name__ == '__main__':
    # Valor inicial
    x0 = 3 / 4
    # Tolerancia
    TOL = 0.0000000001
    # Maximo iteraciones
    MAXIT = 100
    # Funcion
    func = lambda x: (math.cos(2 * x)) ** 2 - x ** 2
    # Llamado de la funcion
    xAprox, itera = newtonRaphson(func, x0, MAXIT, TOL)
    print('xAprox = {}\nIteraciones = {}'.format(xAprox, itera))
```

Código 3: Lenguaje C++.

```
#include<iostream>
#include<cmath>

using namespace std;

double F(double x) {
    return exp(x) - x - 2;
}

double Biseccion(double a, double b, int MAXIT, double TOL) {
    int cont = 1;
    double x;
    double fx;

    while(cont < MAXIT) {
```

```
        x = (a + b)/ 2;
        fx = F(x);
        if(F(a) * fx < 0) {
            b = x;
        }
        if(F(b) * fx < 0) {
            a = x;
        }
        if(abs(fx) < TOL) {
            return x;
        }
        cont = cont + 1;
    }
    return x;
}

int main (int argc, char *argv[]) {
    cout<< Biseccion(0, 2, 100, 0.000001)<<endl;
    system("pause");
    return 0;
}
```