Tarea 2: Método de Raphson multivariable

IDI 2

Alumno: Ricardo De León

Descripción:

Realice código en Python que, recibiendo un sistema de n ecuaciones no lineales $f_i\,(x_1,\ldots,x_n)=0$, un valor inicial X_0 y una exactitud (error) dado E, encuentre (si existe) mediante el método de Newton-Raphson una aproximación de exactitud menor a E para una solución del sistema. Asegúrese que cuenta el número de iteraciones realizadas.

Use su código para resolver los siguientes ejercicios (en todos los casos indique el(los) valor(es) inicial(es) que utilizó y el número de iteraciones que fueron necesarias para alcanzar la respuesta).

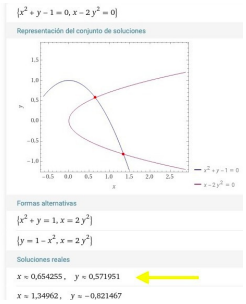
Escriba sus respuestas con **5 cifras significativas**.

Encuentre ***todas*** las soluciones exactas dentro de 10^{-4} para:

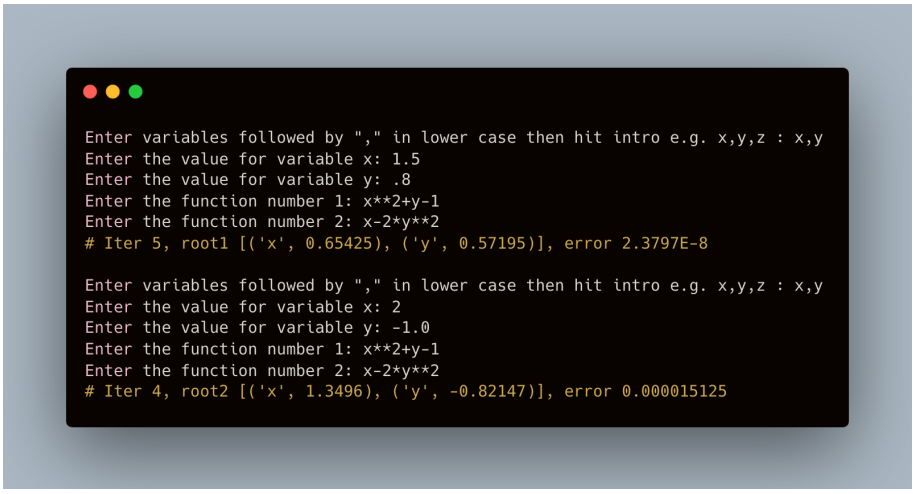
1. $x^2+y=1, x-2y^2=0$
2. $x^2-10x+y^2=-5, xy^2+x-10y=-8$
3. $x\sin y=1, x^2+y^2=4$
4. $y^2\ln x=3, y=x^2$
5. $x+y-z=-2, x^2+y=0, z-y^2=1$

1.- $x^2+y-1=0,\ x-2y^2=0$

Con **WolframAlpha**:



Con Python **Newton Raphson:**



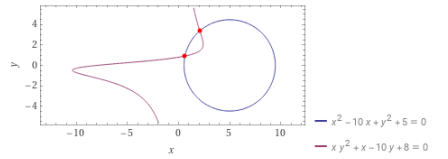
Tarea 2: Método de Raphson multivariable

2.- $x^2 - 10x + y^2 - 5 = 0,\ xy^2 + x - 10y + 8 = 0$

Con **WolframAlpha**:

$\{x^2 - 10x + y^2 + 5 = 0,\ x y^2 + x - 10y + 8 = 0\}$

Representación del conjunto de soluciones



Formas alternativas

$\{x^2 + y^2 + 5 = 10x,\ x y^2 + x + 8 = 10y\}$

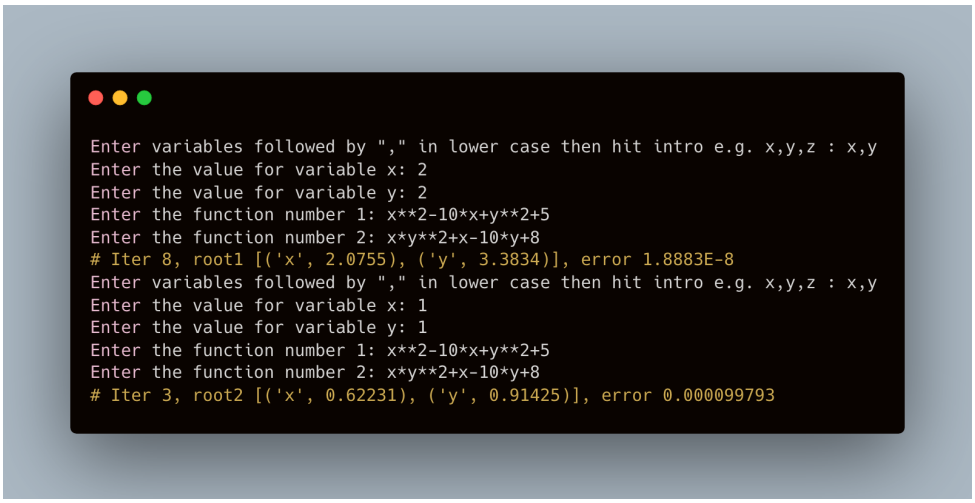
$\{x^2 - 10x + y^2 = -5,\ x(y^2 + 1) = 10y - 8\}$

Soluciones reales

$x \approx 0{,}622312, \quad y \approx 0{,}914247$

$x \approx 2{,}07547, \quad y \approx 3{,}38336$

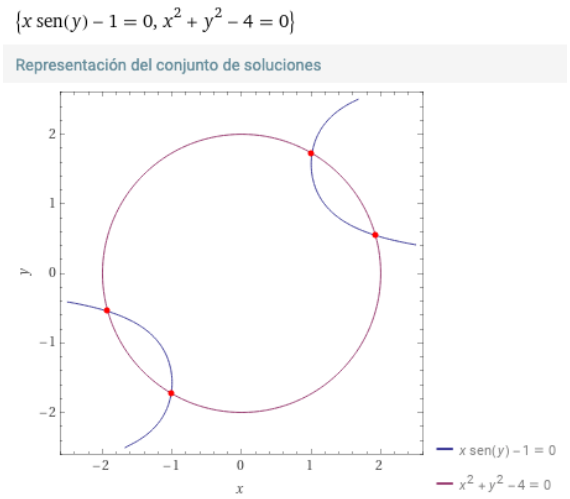
Con Python **Newton Raphson:**



Tarea 2: Método de Raphson multivariable

3.- $\mathbf{x \sin y - 1 = 0}$, $\boldsymbol{x^2 + y^2 - 4 = 0}$

Con WolframAlpha:



Con Python Newton Raphson:

```
Enter variables followed by "," in lower case then hit intro e.g. x,y,z : x,y
Enter the value for variable x: -1
Enter the value for variable y: 0
Enter the function number 1: x*sin(y)-1
Enter the function number 2: x**2+y**2-4
# Iter 6, root [('x', -1.9239), ('y', -0.54660)], error 1.3372E-8
Enter variables followed by "," in lower case then hit intro e.g. x,y,z : x,y
Enter the value for variable x: -1
Enter the value for variable y: -2
Enter the function number 1: x*sin(y)-1
Enter the function number 2: x**2+y**2-4
# Iter 4, root [('x', -1.0120), ('y', -1.7251)], error 1.7850E-7
Enter variables followed by "," in lower case then hit intro e.g. x,y,z : x,y
Enter the value for variable x: 1
Enter the value for variable y: 2
Enter the function number 1: x*sin(y)-1
Enter the function number 2: x**2+y**2-4
# Iter 4, root [('x', 1.0120), ('y', 1.7251)], error 1.7850E-7
Enter variables followed by "," in lower case then hit intro e.g. x,y,z : x,y
Enter the value for variable x: -1
Enter the value for variable y: -1
Enter the function number 1: x*sin(y)-1
Enter the function number 2: x**2+y**2-4
# Iter 9, root [('x', 1.9239), ('y', 0.54660)], error 8.7053E-7
```

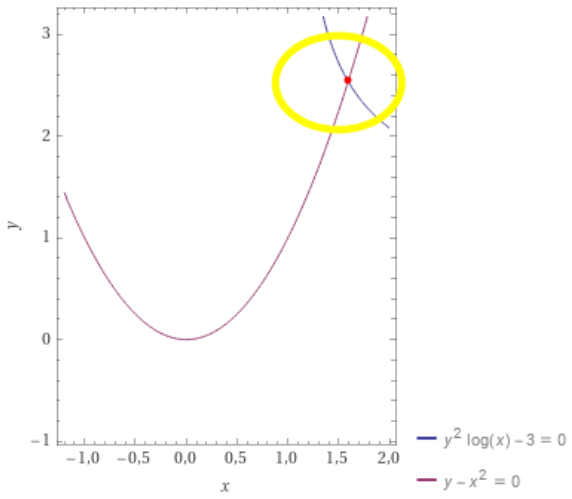
Tarea 2: Método de Raphson multivariable

4.- $y^2\ln(x) - 3 = 0, y - x^2 = 0$

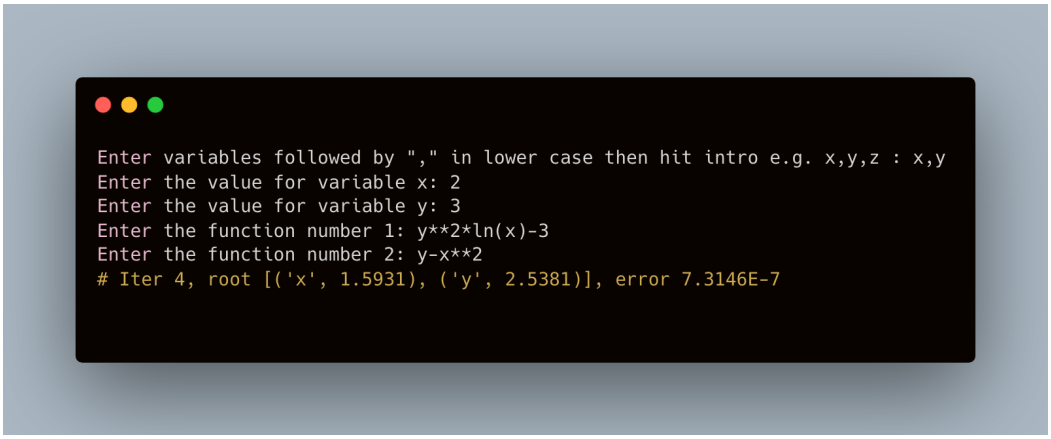
Con WolframAlpha:

$$\{y^2 \log(x) - 3 = 0,\ y - x^2 = 0\}$$

Representación del conjunto de soluciones



Con Python Newton Raphson:



Tarea 2: Método de Raphson multivariable

5.- $\mathbf{x + y - z + 2 = 0, x^2 + y + z0 = 0, z - y^2 + x0 - 1 = 0}$

Con **WolframAlpha**:

Entrada

$\{x + y - z + 2 = 0, x^2 + y + z \times 0 = 0, x \times 0 + z - y^2 - 1 =$

Resultado

$\{x + y - z + 2 = 0, x^2 + y = 0, -y^2 + z - 1 = 0\}$

Formas alternativas

$\{x + y + 2 = z, x^2 + y = 0, y^2 + 1 = z\}$

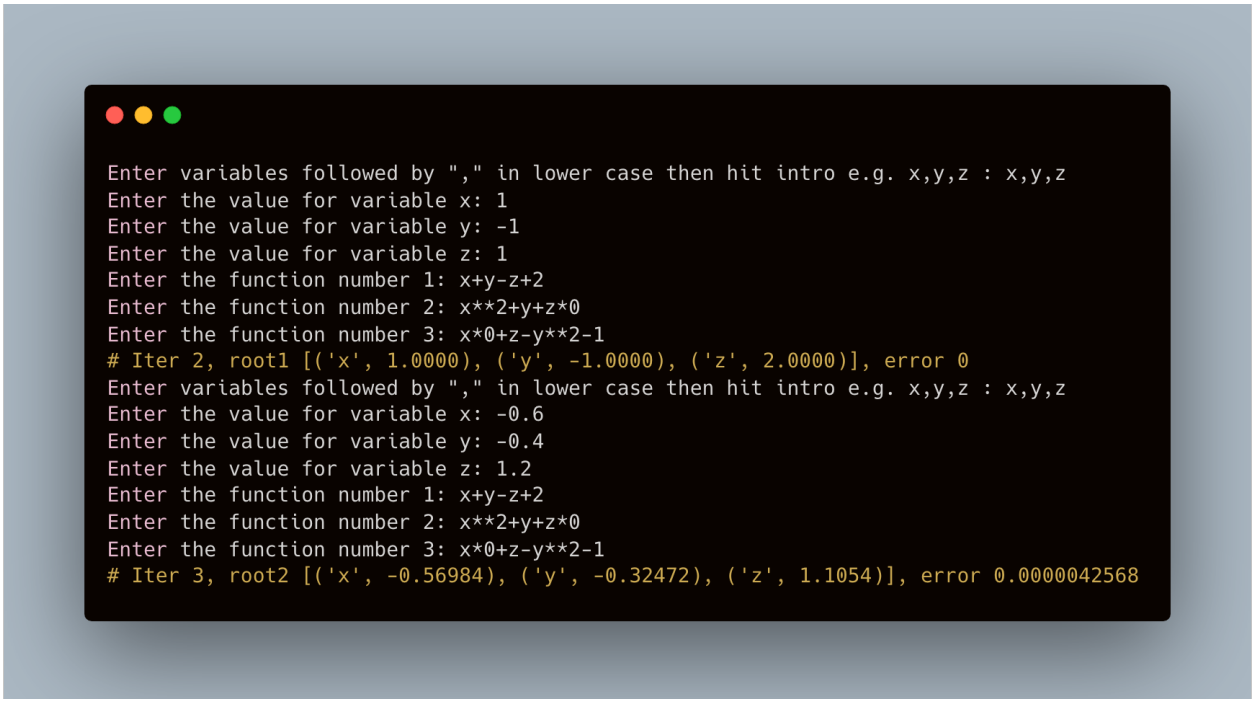
$\{z = x + y + 2, y = -x^2, z = y^2 + 1\}$

Soluciones reales

$x = 1, \quad y = -1, \quad z = 2$

$x \approx -0{,}56984, \quad y \approx -0{,}324718, \quad z \approx 1{,}10544$

Con Python **Newton Raphson:**



Tarea 2: Método de Raphson multivariable

Código:

```python
import sympy as sp
import numpy as np

func_variables = []
variable_values = []
tolerance = 0.0001
max_sig_xifres = 5

def init_func_variables():
    variable = input('Enter variables followed by "," in lower case then hit intro e.g. x,y,z : ')
    global func_variables
    func_variables = variable.split(',')
    for i in func_variables:
        i = sp.symbols(i)
    return func_variables

def create_func_matrix():
    func_list = []
    for i  in range(len(func_variables)):
        func = input(f'Enter the function number {i + 1}: ')
        func_list.append(func)
    return sp.Matrix(func_list)

def create_jacobian(matrix):
    return matrix.jacobian(func_variables)

def calc_inverse_by_matrix_func(jacobian, matrix):
    return jacobian.inv() * matrix

def create_tuple_of_variable_and_values():
    global variable_values
    for i  in func_variables:
        value = float(input(f'Enter the value for variable {i}: '))
        variable_values.append(value)
    return list(zip(func_variables, variable_values))

def newton_raphson_no_linear():

    init_func_variables()
    init_points = create_tuple_of_variable_and_values()
    matrix = create_func_matrix()
    jacob = create_jacobian(matrix=matrix)
    m_inverse = calc_inverse_by_matrix_func(jacobian=jacob, matrix=matrix)
    jacob_eval = jacob.subs(init_points)
    jacob_det=jacob_eval.det()

    if(jacob_det != 0):
        iter_count = 0
        delta = tolerance * 2
        X0 = sp.Matrix(variable_values)
        while not(delta <= tolerance):
            iter_count += 1
            X1 = X0 - m_inverse.subs(init_points)
            delta = np.max(abs(X1 - X0))
            init_points = list(zip(func_variables, X1))
            X0 = X1
            # print(f'X0 {X0}')
        print(f'# Iter {iter_count}, root {list(zip(func_variables, X1.evalf(max_sig_xifres)))}, error
{delta.evalf(max_sig_xifres)}')

newton_raphson_no_linear()
```