Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática
Docente: Fred Torres Cruz
Alumno: Clyde Neil Paricahua Pari

Trabajo Encargado - Nº 004

Método de Newton-Raphson para encontrar raíces

Genera un programa en Python orientado a objetos para hallar la raíz de una función de una variable utilizando el método de Newton-Raphson. El programa calcula automáticamente la derivada, realiza iteraciones hasta que la raíz se aproxima a un valor aceptable según una tolerancia fija y presenta una tabla con todas las iteraciones.

Código en Python

```python
import sympy as sp

class NewtonRaphson:
    def __init__(self, fx_str, iter_max=100, tol=1e-6):
        self.x = sp.Symbol('x')
        self.f_expr = sp.sympify(fx_str)
        self.df_expr = sp.diff(self.f_expr, self.x)
        self.f = sp.lambdify(self.x, self.f_expr, 'math')
        self.df = sp.lambdify(self.x, self.df_expr, 'math')
        self.iter_max = iter_max
        self.tol = tol
        self.iteraciones = []

    def newton(self, x0):
        delta = 0.1
        while self.df(x0) == 0:
            x0 += delta

        self.iteraciones = []
        for i in range(1, self.iter_max + 1):
            f_val = self.f(x0)
            df_val = self.df(x0)
            if df_val == 0:
                print(f"Derivada cero en x = {x0}. Interrumpiendo.")
                break
            x1 = x0 - f_val / df_val
            error = abs(x1 - x0)
            self.iteraciones.append((i, x0, f_val, df_val, x1, error))
            if error < self.tol:
                break
            x0 = x1
        return x0

    def mostrar_tabla(self):
        print("\nIteraci n |    x_n     |   f(x_n)    |   f'(x_n)   |   x_(n+1)   |
            Error")
        print("-------------------------------------------------------------------------
            ")
        for it in self.iteraciones:
            print(f"{it[0]:9d} | {it[1]:10.6f} | {it[2]:11.6f} | {it[3]:11.6f} | {it
                [4]:11.6f} | {it[5]:10.6e}")
```

```
        if self.iteraciones:
            print(f"\nRa z aproximada: {self.iteraciones[-1][4]:.10f}")
            print(f"N mero de iteraciones: {len(self.iteraciones)}")

# --- Entrada de usuario ---
fx_str = input("Ingrese f(x): ")
x0_input = input("Ingrese el valor inicial x0 (Enter para usar x0=0): ")
x0 = float(x0_input) if x0_input.strip() != "" else 0

nr = NewtonRaphson(fx_str)
raiz = nr.newton(x0)
nr.mostrar_tabla()
```

Ejemplo de ejecución

Para la función:

$$f(x) = x^3 - 2x^2 + 2$$

y valor inicial $x_0 = 0$, se obtiene la siguiente tabla de iteraciones:

Comprobación en Exel

Iteracion	x	F(x)=x^3 - 2x^2 + 2	f'(x)	x_(n+1)	
x_0	0,1	1,981	-0,37	5,4540541	
x_1	5,4540541	104,7467292	67,423901	3,9004992	
x_2	3,9004992	30,91399294	30,039685	2,8713941	
x_3	2,8713941	9,184560582	13,249136	2,1781745	
x_4	2,1781745	2,845338858	5,5206343	1,6627738	

Figura 1: Enter Caption

x_39	1,3925977	0,8220475	0,2475941	-1,9275434
x_40	-1,9275434	-12,592487	18,856444	-1,2597352
x_41	-1,2597352	-3,172981	9,7997396	-0,9359531
x_42	-0,9359531	-0,5719188	6,3718367	-0,8461958
x_43	-0,8461958	-0,0380108	5,532925	-0,8393259
x_44	-0,8393259	-0,0002139	5,4707071	-0,8392868
x_45	-0,8392868	-6,905E-09	5,4703538	-0,8392868

Figura 2: Enter Caption

La raíz aproximada obtenida es:

$$x \approx -0{,}8392868$$

Ejecución de código

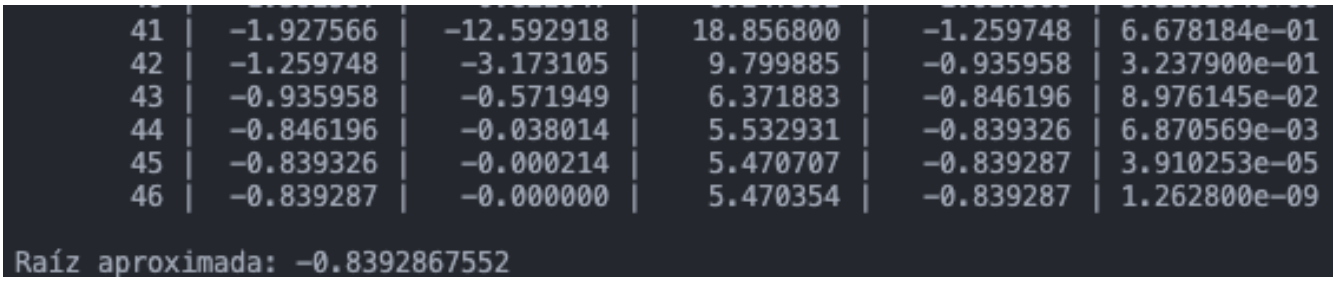

Figura 3: Enter Caption

Se realizaron aproximadamente 12 iteraciones hasta cumplir con la tolerancia 1×10^{-6}.