Trabajo en cargado: Cristian Ronaldo Paucar Yupanqui

INDICE “H”

NOMBRES	APELLIDOS	CITAS	DOCUMENTOS	INDICE H
Edgar Eloy	CARPIO VARGAS	27	9	3
Leonel	COYLA DME	1	5	1
Percy	HUATA PANCA	14	3	2
Fredy heric	VILLASANTE SARAVIA	7	2	2
Fred	TORRES CRUZ	9	40	4
Charles ignacio	MENDOZA MOLLOCONDO	17	8	3
Remo	CHOQUEJAHUA ACERO	2	2	1
Juan carlos	JUAREZ VARGAS	2	3	1
José pánfilo	TITO LIPA	1	3	0
Leonid	ALEMAN GONZALES	0	4	0
Juan Reynaldo	PAREDES QUISPE	0	0	0
Ángel	JAVIER QUISPE CARITA	0	1	1
Romel P.	MELGAREJO BOLIVAR	0	6	3
Milton Antonio	LOPEZ CUEVA	4	6	1
Ernesto Nayer	TUMI FIGUEROA	23	6	3
Vladimir	IBAÑEZ QUISPE	52	21	5
Alejandro	APAZA TARQUI	6	5	1
Bernabe	CANQUI FLORES	20	8	3
Ramiro	LAURA MURILLO	1	2	1
Elqui yeye	PARI CONDORI	1	3	1

|2 DOCENTES CON INDICE MAYOR A 10

Romero-Abad, David

universidad san ignacio de loyola, Lima, Perú • Identificación de Scopus: 56682111300 • 0000-0003-0120-5211

Mostrar toda la información

22.603 Citas de 10.300 documentos | 346 Documentos | 77 índice h

Exportar todo Guardar todo en la lista Ordenar por Date (newest)

Artículo

Elucidación de la reflexión coherente e incoherente de ondas planas en medios de espesor finito: una aplicación pedagógica práctica

1 Citas

,,

Revista Europea de Física, 2024

Texto completo

Artículo

La energía de dos pulsos de onda opuestos e invertidos: un enfoque gráfico

0 Citas

,

Profesor de Física, 2024

Texto completo

CODIGOS DE LOS 4 METODOS:

METODO DE LA BISECCION

```
class Biseccion:
    def __init__(self):
        self.funcion_str = input("Ingrese la funcion f(x): ")
        self.a = float(input("Ingrese el límite inferior (a): "))
        self.b = float(input("Ingrese el límite superior (b): "))
        self.tol = float(input("Ingrese la tolerancia: "))

    def f(self, x):
        return eval(self.funcion_str, {"x": x, "__builtins__": {}})

    def resolver(self):
        a, b = self.a, self.b

        if self.f(a) * self.f(b) > 0:
            print("Error: No hay cambio de signo en el intervalo.")
            return

        print(f"\n{'Iter':<6} {'a':<10} {'b':<10} {'c':<10} {'f(c)':<10} {'Error':<10}")
        print("-" * 60)

        iteracion = 0
```

```
        while True:
            c = (a + b) / 2
            fc = self.f(c)
            error = abs(b - a)
            iteracion += 1

            print(f"{iteracion:<6} {a:<10.5f} {b:<10.5f} {c:<10.5f} {fc:<10.5f} {error:<10.5f}")

            if error < self.tol or abs(fc) < self.tol:
                print(f"\nRaiz encontrada: {c:.8f}")
                print(f"  Iteraciones: {iteracion}")
                print(f"  f({c:.8f}) = {fc:.8f}")
                return c
            if self.f(a) * fc < 0:
                b = c
            else:
                a = c
print("=" * 60)
print("MÉTODO DE BISECCIÓN")
print("=" * 60)

metodo = Biseccion()
metodo.resolver()
```

METODO SECANTE:

```
import math

class Secante:
    def __init__(self, f, x0, x1, tol, nmax):
        self.f = f
```

```
        self.x0 = x0
        self.x1 = x1
        self.tol = tol
        self.nmax = nmax

    def resolver(self):
        for i in range(self.nmax):
            f0, f1 = self.f(self.x0), self.f(self.x1)
            if f1 - f0 == 0:
                print("Error: division por cero.")
                return
            x2 = self.x1 - f1 * (self.x1 - self.x0) / (f1 - f0)
            if abs(x2 - self.x1) < self.tol:
                print(f"Convergio en {i+1} iteraciones: x ≈ {x2:.6f}")
                return
            self.x0, self.x1 = self.x1, x2
        print("No convergio.")

f = lambda x: eval(input("Ingrese f(x): "))
x0 = float(input("x0: "))
x1 = float(input("x1: "))
tol = float(input("tolerancia: "))
nmax = int(input("numero máximo de iteraciones: "))

metodo = Secante(f, x0, x1, tol, nmax)
metodo.resolver()
```

METODO PUNTO FIJO:

```
import math
```

```
class PuntoFijo
    def __init__(self, g, x0, tol, nmax):
        self.g = g
        self.x = x0
        self.tol = tol
        self.nmax = nmax

    def resolver(self):
        for i in range(self.nmax):
            x1 = self.g(self.x)
            if abs(x1 - self.x) < self.tol:
                print(f"Convergió en {i+1} iteraciones: x ≈ {x1:.6f}")
                return
            self.x = x1
        print("No convergió.")

g = lambda x: eval(input("Ingrese g(x): "))
x0 = float(input("x0: "))
tol = float(input("tolerancia: "))
nmax = int(input("número máximo de iteraciones: "))

metodo = PuntoFijo(g, x0, tol, nmax)
metodo.resolver()
```

METODO FALSA CUERDA:

```
import math

class FalsaCuerda:
    def __init__(self, f, a, b, tol, nmax):
```

```
        self.f = f
        self.a = a
        self.b = b
        self.tol = tol
        self.nmax = nmax

    def resolver(self):
        fa, fb = self.f(self.a), self.f(self.b)
        if fa * fb > 0:
            print("No hay cambio de signo en el intervalo.")
            return
        for i in range(self.nmax):
            c = self.b - fb * (self.b - self.a) / (fb - fa)
            fc = self.f(c)
            if abs(fc) < self.tol:
                print(f"Convergió en {i+1} iteraciones: x ≈ {c:.6f}")
                return
            if fa * fc < 0:
                self.b, fb = c, fc
            else:
                self.a, fa = c, fc
        print("No convergió.")

f = lambda x: eval(input("Ingrese f(x): "))
a = float(input("a: "))
b = float(input("b: "))
tol = float(input("tolerancia: "))
nmax = int(input("número máximo de iteraciones: "))

metodo = FalsaCuerda(f, a, b, tol, nmax)
```

metodo.resolver()