

REGISTRO CTIVITAE

INICIO GUÍA CALIFICACIÓN RENACYT HENRY HIGINIO QUISPE RAMOS Manual de uso Cerrar Sesión

NOVEDADES

- Integridad ante todo. La información falsa o imprecisa en tu CTI Vitae puede afectar tu reputación académica.
- Te invitamos a las Capacitaciones Personalizadas de la Biblioteca Virtual del Concytec, fechas disponibles -> <https://biblioteca.concytec.gob.pe/solicite-una-capacitacion>
- Calendario de sesiones de capacitación y asistencia técnica para correcto llenado del CTI Vitae - 2025 <https://bit.ly/AgendaTuCapacitacion2025>
- 2025 APEC 5G Smart Manufacturing Seminar, October 15-16, 2025. Oportunidad de participación en evento (se financia el pasaje) -> <https://2025apec-5gsmartrfg.org>
- Encuesta de satisfacción de usuarios sobre los servicios de información del Sinacti -> <https://goo.su/Wpuas9>

PERFIL

HENRY HIGINIO QUISPE RAMOS

Cargando foto



Calificación, Clasificación y Registro de Investigadores

Solicitar Incorporación

Elegir archivo No se ha seleccionado ningún archivo

Agregar foto

INICIO GUÍA CALIFICACIÓN RENACYT HENRY HIGINIO QUISPE RAMOS

Elegir archivo No se ha seleccionado ningún archivo

Agregar foto

Resumen

2000 quedan todavía

DATOS PERSONALES (FUENTE: RENIEC)

Nombres: HENRY HIGINIO

Apellido paterno: QUISPE

Apellido materno: RAMOS

DNI: 75327974 [Validar DNI](#)

Domicilio: JR. RAMON CASTILLA 1061 PISO 2

Nacionalidad: PERU

Departamento: PUNO Provincia: SAN ROMAN Distrito: JULIACA

DATOS PERSONALES (FUENTE: AUTOREFERENCIADA)

11-10-2004

INDICE H DE DOCENTES

El índice h es una métrica ampliamente utilizada para evaluar la productividad y el impacto académico de los investigadores en función de sus publicaciones y las citas que estas reciben. En el contexto académico actual, el índice h se ha convertido en un referente importante para medir la calidad y relevancia de los trabajos científicos en diversas disciplinas.

A continuación, se presenta la recopilación de los índices h de los profesores de la Facultad de Ingeniería Estadística e Informática, obtenidos a través de la base de datos Scopus. Esta información permite valorar el nivel de investigación de cada docente y su contribución al desarrollo del conocimiento en su área, proporcionando así una visión general del potencial académico de la facultad.

NOMBRE	indice h	documentos p	citas	id scopus
CANQUI FLORES BERNABE	3	8	20	57214094525
IBAÑES QUISPE VLADIMIRO	5	21	52	57201897173
TUMI FIGUEROA ERNESTO NAYER	3	6	23	57003617100
CHOQUEJAHUA-ACERO REMO	1	2	2	58491331500
APAZA TARQUI ALEJANDRO	1	5	6	58491331500
MENDOZA MOLLOCONDO CHARLES	3	8	17	57562709900
CARPIO VARGAS EDGAR ELOY	3	9	27	57221112735
GONZALES LEONID ALEMAN	0	4	0	58898481200
COYLA IDME LEONEL	1	5	1	58930847700
HUATA PANCA PERCY	2	3	14	57571711100
PARI CONDORI ELQUI YEYE	1	3	1	58284529900
TITO LIPA JOSE PANFILO	0	3	0	58088282100
VILLASANTE SARAVIA FREDY HERIC	2	2	7	58671288200
JUAREZ VARGAS JUAN CARLO	1	3	2	58898176700
LOPEZ CUEVA MILTON ANTONIO	1	6	4	58671914800
TORRES CRUZ FRED	4	40	74	57214066496

ARTICULOS RELACIONADOS AL CURSO

AUTOR	TITULO	INDICE H
Xueyan Ru	Improved Newton-Raphson-based optimizer with gap evolution strategy for multiple engineering problems	2
Mohsen Madadi	A new method for rooting nonlinear equations based on the Bisection method	5

METODO REGULA FALSI

```
2 class RegulaFalsi:
3     def __init__(self, expr, x0, x1, tol=1e-6, max_iter=100):
4         self.expr = expr.replace("^", "**") # permitir potencias como x^2
5         self.x0 = x0
6         self.x1 = x1
7         self.tol = tol
8         self.max_iter = max_iter
9     def f(self, x):
10        return eval(self.expr)
11    def calcular(self):
12        f0 = self.f(self.x0)
13        f1 = self.f(self.x1)
14        if f0 * f1 > 0:
15            print("Error: f(x0) y f(x1) deben tener signos opuestos.")
16            return None
17        print("\nIter\t x0\t x1\t x2\t f(x2)")
18        print("-----")
19        for i in range(1, self.max_iter + 1):
20            # Fórmula de Regula Falsi
21            x2 = self.x1 - (f1 * (self.x0 - self.x1)) / (f0 - f1)
22            f2 = self.f(x2)
23            print(f"\t{i}\t {self.x0:.6f}\t {self.x1:.6f}\t {x2:.6f}\t {f2:.6f}")
24            if abs(f2) < self.tol:
25                print(f"\nRaíz aproximada encontrada: {x2:.6f}")
26                return x2
27            if f0 * f2 < 0:
28                self.x1 = x2
29                f1 = f2
30            else:
31                self.x0 = x2
32                f0 = f2
33        print("\nNo se encontró la raíz dentro del límite de iteraciones.")
34        return None
35 if __name__ == "__main__":
36     print("== MÉTODO DE REGULA FALSI ==")
37     expr = input("Ingresa la función f(x): ") # Ej: x**3 - x - 2
38     x0 = float(input("Valor inicial x0: "))
39     x1 = float(input("Valor inicial x1: "))
40     tol = float(input("Tolerancia (ej. 0.0001): "))
41     max_iter = int(input("Número máximo de iteraciones: "))
42     metodo = RegulaFalsi(expr, x0, x1, tol, max_iter)
43     metodo.calcular()
```

METODO PUNTO FIJO

```
1 import math
2
3 print("METODO DE PUNTO FIJO")
4 print("=====")
5
6 funcion_str = input("Escribe g(x): ")
7
8 x0 = float(input("Valor inicial x0: "))
9
10 tolerancia = float(input("Tolerancia: "))
11
12 max_iter = int(input("Maximo iteraciones: "))
13
14 print("\nCalculando...")
15 print("Iter\tx_nuevo\tDiferencia")
16
17 x_actual = x0
18 encontrado = False
19
20 for i in range(1, max_iter + 1):
21     x_siguiente = eval(funcion_str, {"math": math, "x": x_actual})
22
23     dif = abs(x_siguiente - x_actual)
24
25     print(f"{i}\t{x_siguiente:.6f}\t{dif:.6f}")
26
27     if dif < tolerancia:
28         print(f"\nRaiz encontrada: {x_siguiente:.8f}")
29         print(f"iteraciones: {i}")
30         encontrado = True
31         break
32
33     x_actual = x_siguiente
34
35 if not encontrado:
36     print(f"\nNo converge en {max_iter} iteraciones")
37     print(f"ultimo valor: {x_siguiente:.8f}")
38
39 print("Terminado")
```

METODO SECANTE

```
1 import math # por si usamos sin, cos, exp, etc
2
3 def metodo_secante():
4     print("==> metodo de la secante ==>")
5     expr = input("ingresa la funcion f(x): ") # ej: x**3 - x - 2
6     x0 = float(input("valor inicial x0: "))
7     x1 = float(input("valor inicial x1: "))
8     tol = float(input("tolerancia (ej 0.0001): "))
9     max_iter = int(input("numero maximo de iteraciones: "))
10
11    # definimos la funcion f(x)
12    def f(x):
13        return eval(expr)
14
15    print("\niter\t x0\t x1\t x2\t f(x2)")
16    print("-----")
17
18    for i in range(max_iter):
19        f0 = f(x0)
20        f1 = f(x1)
21
22        if f1 - f0 == 0:
23            print("error: division por cero")
24            break
25        x2 = x1 - f1 * (x1 - x0) / (f1 - f0)
26        f2 = f(x2)
27
28        # mostramos cada paso
29        print(f"{i}\t {x0:.6f}\t {x1:.6f}\t {x2:.6f}\t {f2:.6f}")
30
31        # comprobamos si ya esta dentro de la tolerancia
32        if abs(x2 - x1) < tol:
33            print("\nraiz aproximada encontrada:", round(x2,6))
34            break
35        x0 = x1
36        x1 = x2
37    else:
38        print("\nno se encontro la raiz dentro del limite de iteraciones")
39 metodo_secante()
40
```

METODO BISECCION

```
1 import math
2
3 class Biseccion:
4     def __init__(self, expr, a, b, tol=1e-6, max_iter=100):
5         self.expr = expr.replace("^", "**")
6         self.a = a
7         self.b = b
8         self.tol = tol
9         self.max_iter = max_iter
10    def f(self, x):
11        return eval(self.expr)
12    def calcular(self):
13        fa = self.f(self.a)
14        fb = self.f(self.b)
15        if fa * fb > 0:
16            print("Error: f(a) y f(b) deben tener signos opuestos.")
17            return
18        print("\nIter\t a\t b\t x\t f(x)")
19        print("-----")
20        for i in range(1, self.max_iter + 1):
21            x = (self.a + self.b) / 2
22            fx = self.f(x)
23            print(f"{i}\t {self.a:.6f}\t {self.b:.6f}\t {x:.6f}\t {fx:.6f}")
24
25            if abs(fx) < self.tol or (self.b - self.a) / 2 < self.tol:
26                print(f"\nRaíz aproximada encontrada: {x:.6f}")
27                return
28            if fa * fx < 0:
29                self.b, fb = x, fx
30            else:
31                self.a, fa = x, fx
32        print("\nNo se encontró la raíz dentro del límite de iteraciones.")
33 print("== MÉTODO DE BISECCIÓN ==")
34 expr = input("Ingresa la función f(x): " ) # Ej: x**3 - x - 2
35 a = float(input("Extremo a: "))
36 b = float(input("Extremo b: "))
37 tol = float(input("Tolerancia (ej. 0.0001): "))
38 max_iter = int(input("Número máximo de iteraciones: "))
39
40 metodo = Biseccion(expr, a, b, tol, max_iter)
41 metodo.calcular()
42
```