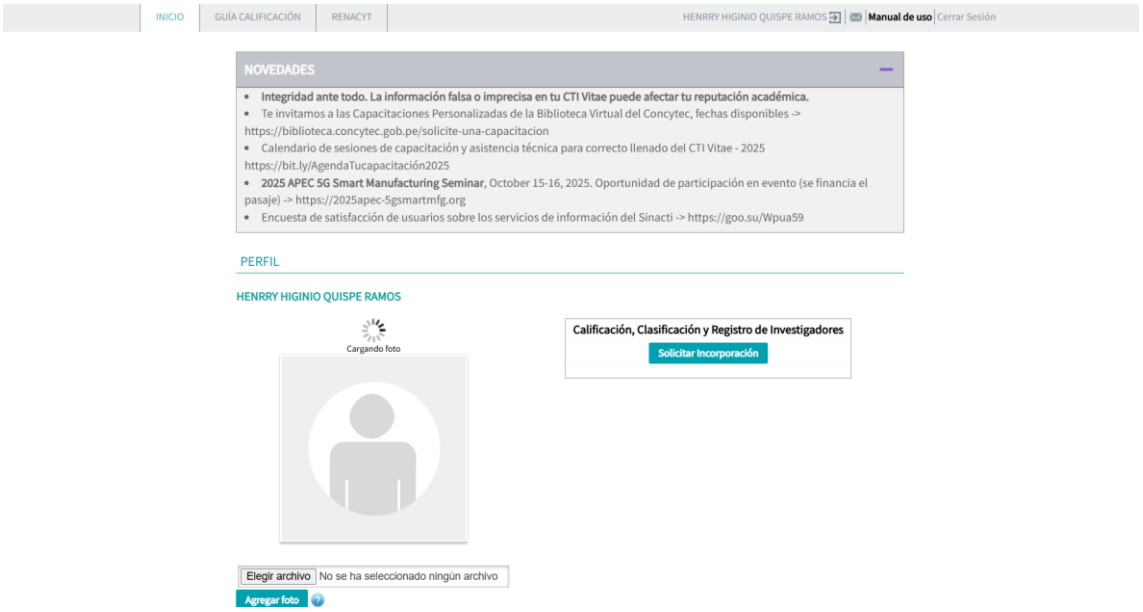
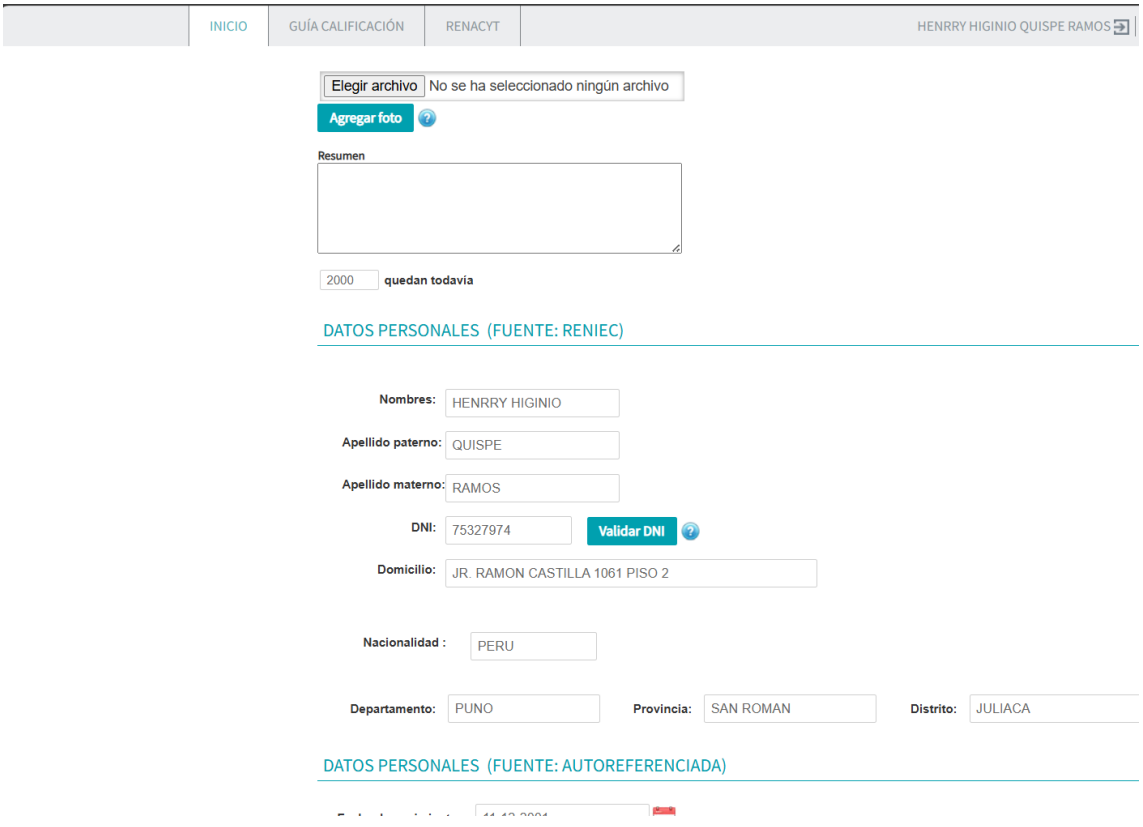
REGISTRO CTIVITAE





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

```python
class RegulaFalsi:
    def __init__(self, expr, x0, x1, tol=1e-6, max_iter=100):
        self.expr = expr.replace("^", "**")  # permitir potencias como x^2
        self.x0 = x0
        self.x1 = x1
        self.tol = tol
        self.max_iter = max_iter
    def f(self, x):
        return eval(self.expr)
    def calcular(self):
        f0 = self.f(self.x0)
        f1 = self.f(self.x1)
        if f0 * f1 > 0:
            print("Error: f(x0) y f(x1) deben tener signos opuestos.")
            return None
        print("\nIter\t x0\t\t x1\t\t x2\t\t f(x2)")
        print("-----------------------------------------------------------")
        for i in range(1, self.max_iter + 1):
            # Fórmula de Regula Falsi
            x2 = self.x1 - (f1 * (self.x0 - self.x1)) / (f0 - f1)
            f2 = self.f(x2)
            print(f"{i}\t {self.x0:.6f}\t {self.x1:.6f}\t {x2:.6f}\t {f2:.6f}")
            if abs(f2) < self.tol:
                print(f"\nRaíz aproximada encontrada: {x2:.6f}")
                return x2
            if f0 * f2 < 0:
                self.x1 = x2
                f1 = f2
            else:
                self.x0 = x2
                f0 = f2
        print("\nNo se encontró la raíz dentro del límite de iteraciones.")
        return None
if __name__ == "__main__":
    print("=== MÉTODO DE REGULA FALSI ===")
    expr = input("Ingresa la función f(x): ")  # Ej: x**3 - x - 2
    x0 = float(input("Valor inicial x0: "))
    x1 = float(input("Valor inicial x1: "))
    tol = float(input("Tolerancia (ej. 0.0001): "))
    max_iter = int(input("Número máximo de iteraciones: "))
    metodo = RegulaFalsi(expr, x0, x1, tol, max_iter)
    metodo.calcular()
```

METODO PUNTO FIJO

```python
import math

print("METODO DE PUNTO FIJO")
print("====================")

funcion_str = input("Escribe g(x): ")

x0 = float(input("Valor inicial x0: "))

tolerancia = float(input("Tolerancia: "))

max_iter = int(input("Maximo iteraciones: "))

print("\\nCalculando...")
print("Iter\\tx_nuevo\\t\\tDiferencia")

x_actual = x0
encontrado = False

for i in range(1, max_iter + 1):
    x_siguiente = eval(funcion_str, {"math": math, "x": x_actual})

    dif = abs(x_siguiente - x_actual)

    print(f"{i}\\t{x_siguiente:.6f}\\t{dif:.6f}")

    if dif < tolerancia:
        print(f"\\nRaiz encontrada: {x_siguiente:.8f}")
        print(f"Iteraciones: {i}")
        encontrado = True
        break

    x_actual = x_siguiente

if not encontrado:
    print(f"\\nNo converge en {max_iter} iteraciones")
    print(f"Ultimo valor: {x_siguiente:.8f}")

print("Terminado")
```

METODO SECANTE

```python
import math   # por si usamos sin, cos, exp, etc

def metodo_secante():
    print("=== metodo de la secante ===")
    expr = input("ingresa la funcion f(x): ")   # ej: x**3 - x - 2
    x0 = float(input("valor inicial x0: "))
    x1 = float(input("valor inicial x1: "))
    tol = float(input("tolerancia (ej 0.0001): "))
    max_iter = int(input("numero maximo de iteraciones: "))

    # definimos la funcion f(x)
    def f(x):
        return eval(expr)

    print("\\niter\\t x0\\t\\t x1\\t\\t x2\\t\\t f(x2)")
    print("-----------------------------------------------------")

    for i in range(max_iter):
        f0 = f(x0)
        f1 = f(x1)

        if f1 - f0 == 0:
            print("error: division por cero")
            break
        x2 = x1 - f1 * (x1 - x0) / (f1 - f0)
        f2 = f(x2)

        # mostramos cada paso
        print(f"{i}\\t {x0:.6f}\\t {x1:.6f}\\t {x2:.6f}\\t {f2:.6f}")

        # comprobamos si ya esta dentro de la tolerancia
        if abs(x2 - x1) < tol:
            print("\\nraiz aproximada encontrada:", round(x2,6))
            break
        x0 = x1
        x1 = x2
    else:
        print("\\nno se encontro la raiz dentro del limite de iteraciones")
metodo_secante()

```

METODO BISECCION

```python
import math

class Biseccion:
    def __init__(self, expr, a, b, tol=1e-6, max_iter=100):
        self.expr = expr.replace("^", "**")
        self.a = a
        self.b = b
        self.tol = tol
        self.max_iter = max_iter
    def f(self, x):
        return eval(self.expr)
    def calcular(self):
        fa = self.f(self.a)
        fb = self.f(self.b)
        if fa * fb > 0:
            print("Error: f(a) y f(b) deben tener signos opuestos.")
            return
        print("\nIter\t a\t\t b\t\t x\t\t f(x)")
        print("-----------------------------------------------------------------")
        for i in range(1, self.max_iter + 1):
            x = (self.a + self.b) / 2
            fx = self.f(x)
            print(f"{i}\t {self.a:.6f}\t {self.b:.6f}\t {x:.6f}\t {fx:.6f}")

            if abs(fx) < self.tol or (self.b - self.a) / 2 < self.tol:
                print(f"\nRaíz aproximada encontrada: {x:.6f}")
                return
            if fa * fx < 0:
                self.b, fb = x, fx
            else:
                self.a, fa = x, fx
        print("\nNo se encontró la raíz dentro del límite de iteraciones.")
print("=== MÉTODO DE BISECCIÓN ===")
expr = input("Ingresa la función f(x): ")  # Ej: x**3 - x - 2
a = float(input("Extremo a: "))
b = float(input("Extremo b: "))
tol = float(input("Tolerancia (ej. 0.0001): "))
max_iter = int(input("Número máximo de iteraciones: "))

metodo = Biseccion(expr, a, b, tol, max_iter)
metodo.calcular()
```