TRABAJO ENCARGADO N6-UNIDAD_2

JOB EDWARD APAZA CURTIHUNCA

1. TRABAJO ENCARGADO
   En este trabajo encontraremos el puntaje que tienen los docentes de la facultad de estadística e informática y con su respectivo "índice h"

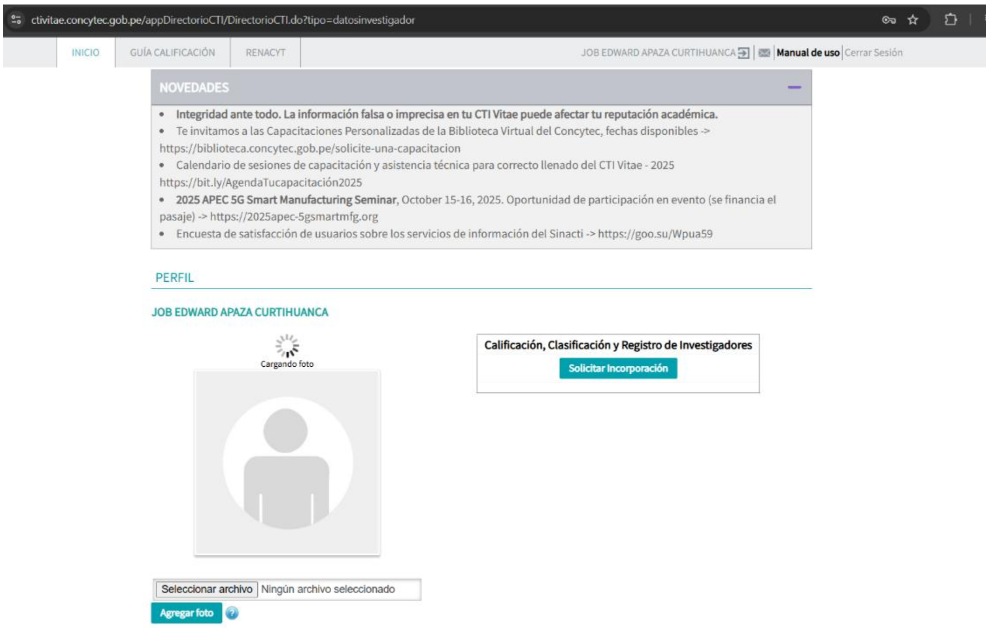
A	B	C	D	E	F
NUMERO	NOMBRES	APELLIDOS	indice h	documentos	cita de documentos
n.1	Juan carlos	JUAREZ VARGAS	1	3	2
n.2	Milton antonio	LOPEZ CUEVA	1	6	4
n.3	Edgar eloy	CARPIO VARGAS	3	9	27
n.4	Bernabé	CANQUI FLORES	3	8	20
n.5	Vladimiro	IBÁÑEZ QUISPE	5	21	52
n.6	Ernesto nayer	TUMI FIGUEROA	3	6	23
n.7	Juan reynaldo	PAREDES QUISPE	0	0	0
n.8	Remo	CHOQUEJAHUA ACERO	1	2	2
n.9	Alejandro	APAZA TARQUI	1	5	6
n.10	Charles ignacio	MENDOZA MOLLOCONDO	3	8	17
n.11	Leonid	ALEMÁN GONZALES	0	4	0
n.12	Leonel	COYLA DME	1	5	1
n.13	Percy	HUATA PANCA	2	3	14
n.14	Elqui yeye	PARI CONDORI	1	3	1
n.15	José pánfilo	TITO LIPA	0	3	0
n.16	Fredy heric	VILLASANTE SARAVIA	2	2	7
n.17	Ramiro	Laura Murillo	1	2	1
n.18	Ángel	Javier Quispe Carita	1	1	0
n.19	Romel P.	Melgarejo-Bolívar	3	6	0
n.20	Fred	Torres-Cruz	4	40	0

Son los docentes que encontré en la búsqueda de sabes cual es índice h y numero de documentos que publicaron esta información la saque de SCOPUS:

https://www.scopus.com/home.uri?zone=header&origin=AuthorNamesList

En la cual nos ayuda a saber el índice de los docentes y la cantidad de documentos que estos han realizado atreves de su trayectoria, . y no olvidar que como una prueba tenemos que poner una captura de pantalla de que nos inscribimos a CONCITEC:

https://ctivitae.concytec.gob.pe/appDirectorioCTI/index.jsp



2. TRABAJO ENCARGADO.

El segundo trabajo encargado es de encontrar por lo mínimo 2 docentes que su índice h sea mayor a 10, la cual estaremos buscando tanto a nivel nacional como mundial, y mostrándole 1 documento que publico. y no olvidar que como una prueba tenemos que poner una captura de pantalla de que nos inscribimos a CONCITEC:

https://ctivitae.concytec.gob.pe/appDirectorioCTI/index.jsp

Mauricio Sánchez, David Santos

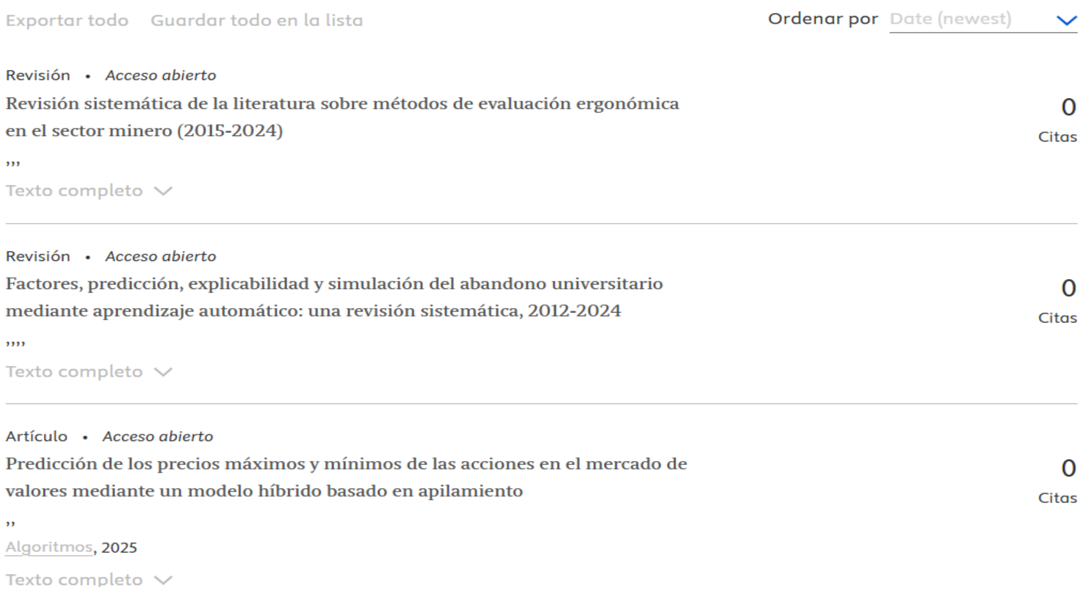
Universidad Nacional Mayor de San Marcos, Lima, Perú • Identificación de Scopus: 55906672000 • 0000-0001-9262-626X

Mostrar toda la información

946 Citas de **896** documentos | 113 Documentos | 15 índice h

Editar perfil ••• Más

Y sus documentos son:



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

Revisión • *Acceso abierto*

Revisión sistemática de la literatura sobre métodos de evaluación ergonómica en el sector minero (2015-2024)

,,,

Texto completo

0 Citas

Revisión • *Acceso abierto*

Factores, predicción, explicabilidad y simulación del abandono universitario mediante aprendizaje automático: una revisión sistemática, 2012-2024

,,,,

Texto completo

0 Citas

Artículo • *Acceso abierto*

Predicción de los precios máximos y mínimos de las acciones en el mercado de valores mediante un modelo híbrido basado en apilamiento

,,

Algoritmos, 2025

Texto completo

0 Citas

El siguiente es un docente extranjero:

Saad, Youcef

University of Minnesota Twin Cities, Minneapolis, United States • Scopus ID: 7006674191 • 0000-0002-8614-5360

Show all information

14,511 Citations by **10,616** documents | 257 Documents | 62 h-index

Edit profile ••• More

0 of 0 documents Limited access

Export all   Save all to list   Sort by Date (newest)

Article • *Open access*

Acceleration methods for fixed-point iterations

1 Citations

Acta Numerica, 2025

Full text

Article • *Open access*

A rational-Chebyshev projection method for nonlinear eigenvalue problems

,

1 Citations

Numerical Linear Algebra with Applications, 2024

Full text

Article

A PARALLEL ALGORITHM FOR COMPUTING PARTIAL SPECTRAL FACTORIZATIONS OF MATRIX PENCILS VIA CHEBYSHEV APPROXIMATION

, , ,

1 Citations

SIAM Journal on Scientific Computing, 2024

Full text

3. TRABAJO ENCARGADO

En el tercer trabajo encargado tenemos que subir los 4 metodos, que expusieron mis compañeros:

- METODO FALSA POSICION
- METODO DE LA SECATE
- METODO DEL PUNTO FIJO
- METODO DE LA BISECCION

Los métodos estarán hecho en un lenguaje de programación llamado PYTHON EN LA CUAL DEJARE EL CODIGO Y COMO MUESTRA EL RESULTADO EN UNA TABLA DE ITERACIONES:

- METODO DE LA FALSA POSICION.

El **método de la falsa posición** (o **regula falsi**) es un método **numérico para encontrar raíces de una ecuación** $f(x)=0f(x) = 0f(x)=0$.

Funciona así:
Se eligen **dos puntos iniciales** aaa y bbb tales que f(a)f(a)f(a) y f(b)f(b)f(b) tengan **signos opuestos** (es decir, la raíz está entre ellos). Luego se **traza una línea recta** entre los puntos (a,f(a))(a, f(a))(a,f(a)) y (b,f(b))(b, f(b))(b,f(b)), y se calcula el punto donde esa recta **corta el eje x**.

Esa intersección se toma como una **nueva aproximación** a la raíz, y se repite el proceso reemplazando uno de los extremos según el signo de la función.

Código:

```
def falsa_posicion(f, a, b, tol=1e-6, max_iter=100):
    """
    Encuentra la raíz usando el método de falsa posición
    f: función
    a, b: intervalo inicial [a, b]
    tol: tolerancia
    max_iter: número máximo de iteraciones
    """
    print("\n=== MÉTODO DE LA FALSA POSICIÓN ===")



    if f(a) * f(b) > 0:
        print("Error: f(a) y f(b) deben tener signos opuestos")
        return None

    print(f"{'Iter':<6} {'a':<15} {'b':<15} {'c':<15} {'f(c)':<15} {'Error':<15}")
    print("-" * 95)

    c_anterior = a

    for i in range(max_iter):
        fa = f(a)
        fb = f(b)
```

```
        # Fórmula de la falsa posición
        c = (a * fb - b * fa) / (fb - fa)
        fc = f(c)

        if i > 0:
            error = abs(c - c_anterior)
        else:
            error = abs(b - a)

        print(f"{i+1:<6} {a:<15.8f} {b:<15.8f} {c:<15.8f} {fc:<15.8e} {error:<15.8e}")

        if error < tol or abs(fc) < tol:
            print(f"\nConvergió en {i+1} iteraciones")
            print(f"Raíz aproximada: x = {c:.8f}")
            return c
        if fa * fc < 0:
            b = c
        else:
            a = c
        c_anterior = c
    print(f"\nNo convergió en {max_iter} iteraciones")
    return c
# Ejemplo de uso: Encontrar raíz de x^2 - 2 = 0
def f(x):
    return x**2 - 2
# Ejecutar
raiz = falsa_posicion(f, 1, 2)
```

SALIDA DEL PROGRAMA:

```
= RESTART: C:/Users/Admin/Documents/UNAP/PRIGRAMACION NUMERICA/tarea segundo/METODO_FALS
A_POSICION.py

=== MÉTODO DE LA FALSA POSICIÓN ===
Iter   a               b               c               f(c)            Error
---------------------------------------------------------------------------------------
-------
1      1.00000000      2.00000000      1.33333333      -2.22222222e-01 1.00000000e+00
2      1.33333333      2.00000000      1.40000000      -4.00000000e-02 6.66666667e-02
3      1.40000000      2.00000000      1.41176471      -6.92041522e-03 1.17647059e-02
4      1.41176471      2.00000000      1.41379310      -1.18906064e-03 2.02839757e-03
5      1.41379310      2.00000000      1.41414141      -2.04060810e-04 3.48310693e-04
6      1.41414141      2.00000000      1.41420118      -3.50127797e-05 5.97692905e-05
7      1.41420118      2.00000000      1.41421144      -6.00728684e-06 1.02550429e-05
8      1.41421144      2.00000000      1.41421320      -1.03068876e-06 1.75949467e-06
9      1.41421320      2.00000000      1.41421350      -1.76838272e-07 3.01881780e-07

Convergió en 9 iteraciones
Raíz aproximada: x = 1.41421350
```

METODO DE LA SECANTE:

El **método de la secante** es un **método numérico iterativo** para encontrar una raíz de una función f(x)=0f(x) = 0f(x)=0. Utiliza **dos valores iniciales** y una **línea secante** que pasa por los puntos correspondientes de la función para aproximar la raíz. En cada paso, la intersección de la secante con el eje xxx se usa como nueva aproximación, repitiendo el proceso hasta obtener la precisión deseada.

CODIGO:

```
# Método de la Secante
def secante(f, x0, x1, tol=1e-6, max_iter=100):
    """
    Encuentra la raíz usando el método de la secante
    f: función
    x0, x1: dos valores iniciales
    tol: tolerancia
    max_iter: número máximo de iteraciones
    """
    print("\n        MÉTODO DE LA SECANTE        ")
    print(f"{'Iter':<6} {'x_n-1':<15} {'x_n':<15} {'x_n+1':<15} {'f(x_n+1)':<15} {'Error':<15}")
    print("-" * 95)
```

```
    for i in range(max_iter):
        fx0 = f(x0)
        fx1 = f(x1)



        if abs(fx1 - fx0) < 1e-12:
            print("\nError: División por cero")
            return x1



        x2 = x1 - fx1 * (x1 - x0) / (fx1 - fx0)
        fx2 = f(x2)
        error = abs(x2 - x1)

        print(f"{i+1:<6} {x0:<15.8f} {x1:<15.8f} {x2:<15.8f} {fx2:<15.8e} {error:<15.8e}")

        if error < tol or abs(fx2) < tol:
            print(f"\nConvergió en {i+1} iteraciones")
            print(f"Raíz aproximada: x = {x2:.8f}")
            return x2



        x0 = x1
        x1 = x2

    print(f"\nNo convergió en {max_iter} iteraciones")
    return x2

# Encontrar raíz de x^2 - 2 = 0
def f(x):
    return x**2 - 2

# Ejecutar
```

```
raiz = secante(f, 1, # Método de la Secante
def secante(f, x0, x1, tol=1e-6, max_iter=100):
    """
    Encuentra la raíz usando el método de la secante
    f: función
    x0, x1: dos valores iniciales
    tol: tolerancia
    max_iter: número máximo de iteraciones
    """
    print("\n         MÉTODO DE LA SECANTE        ")
    print(f"{'Iter':<6} {'x_n-1':<15} {'x_n':<15} {'x_n+1':<15} {'f(x_n+1)':<15} {'Error':<15}")
    print("-" * 95)

    for i in range(max_iter):
        fx0 = f(x0)
        fx1 = f(x1)



        if abs(fx1 - fx0) < 1e-12:
            print("\nError: División por cero")
            return x1



        x2 = x1 - fx1 * (x1 - x0) / (fx1 - fx0)
        fx2 = f(x2)
        error = abs(x2 - x1)

        print(f"{i+1:<6} {x0:<15.8f} {x1:<15.8f} {x2:<15.8f} {fx2:<15.8e} {error:<15.8e}")

        if error < tol or abs(fx2) < tol:
            print(f"\nConvergió en {i+1} iteraciones")
            print(f"Raíz aproximada: x = {x2:.8f}")
            return x2
```

```
            x0 = x1
            x1 = x2

    print(f"\nNo convergió en {max_iter} iteraciones")
    return x2

# Encontrar raíz de x^2 - 2 = 0
def f(x):
    return x**2 - 2

# Ejecutar
raiz = secante(f, 1, 2)2)
```

SALIDA DEL PROGRAMA:

```
= RESTART: C:/Users/Admin/Documents/UNAP/PRIGRAMACION NUMERICA/tarea segundo/METODO_SECA
NTE.py

         MÉTODO DE LA SECANTE
Iter   x_n-1            x_n              x_n+1            f(x_n+1)         Error
----------------------------------------------------------------------------------------
-------
1      1.00000000       2.00000000       1.33333333       -2.22222222e-01 6.66666667e-01
2      2.00000000       1.33333333       1.40000000       -4.00000000e-02 6.66666667e-02
3      1.33333333       1.40000000       1.41463415       1.18976800e-03  1.46341463e-02
4      1.40000000       1.41463415       1.41421144       -6.00728684e-06 4.22707867e-04
5      1.41463415       1.41421144       1.41421356       -8.93145558e-10 2.12358245e-06

Convergió en 5 iteraciones
Raíz aproximada: x = 1.41421356
```

METODO DEL PUNTO FIJO:

El **método del punto fijo** es un **método iterativo** que busca una raíz de una ecuación (f(x) = 0) transformándola en una forma equivalente (x = g(x)).

Luego se aplica la **iteración** (x_{n+1} = g(x_n)) repetidamente, hasta que el valor de (x) **no cambie significativamente**, lo que indica que se ha alcanzado el **punto fijo**, es decir, la raíz aproximada de la función.

CODIGO:

File Edit Format Run Options Window Help

```python
# Método de Punto Fijo
def punto_fijo(g, x0, tol=1e-6, max_iter=100):
    """
    Encuentra la raíz usando el método de punto fijo
    g: función de iteración g(x)
    x0: valor inicial
    tol: tolerancia
    max_iter: número máximo de iteraciones
    """
    print("\n=== MÉTODO DE PUNTO FIJO ===")
    print(f"{'Iter':<6} {'x':<15} {'g(x)':<15} {'Error':<15}")
    print("-" * 55)

    x = x0
    for i in range(max_iter):
        x_nuevo = g(x)
        error = abs(x_nuevo - x)

        print(f"{i+1:<6} {x:<15.8f} {x_nuevo:<15.8f} {error:<15.8e}")

        if error < tol:
            print(f"\nConvergió en {i+1} iteraciones")
            print(f"Raíz aproximada: x = {x_nuevo:.8f}")
            return x_nuevo

        x = x_nuevo

    print(f"\nNo convergió en {max_iter} iteraciones")
    return x


def g(x):
    return (x + 2/x) / 2

#  valor inicial cercano a sqrt(2)
raiz = punto_fijo(g, 1.5)
```

SALIDA DEL PROGRAMA:

```
= RESTART: C:/Users/Admin/Documents/UNAP/PRIGRAMACION NUMERICA/tare
O_FIJO.py

=== MÉTODO DE PUNTO FIJO ===
Iter   x              g(x)           Error
-------------------------------------------------------
1      1.50000000     1.41666667     8.33333333e-02
2      1.41666667     1.41421569     2.45098039e-03
3      1.41421569     1.41421356     2.12389982e-06
4      1.41421356     1.41421356     1.59494640e-12

Convergió en 4 iteraciones
Raíz aproximada: x = 1.41421356
```

METODO DE LA BISECCION:

El **método de la bisección** es un **procedimiento numérico** para encontrar una raíz de una función continua (f(x) = 0).
Se parte de **dos puntos (a) y (b)** donde (f(a)) y (f(b)) tienen **signos opuestos**, lo que asegura que hay una raíz entre ellos. Luego se **divide el intervalo a la mitad**, se evalúa el punto medio y se **elige el subintervalo** donde la función cambia de signo. Este proceso se repite hasta que la raíz esté **suficientemente aproximada**.

CODIGO:

```python
# Método de Bisección
def biseccion(f, a, b, tol=1e-6, max_iter=100):
    """
    Encuentra la raíz usando el método de bisección
    f: función
    a, b: intervalo inicial [a, b]
    tol: tolerancia
    max_iter: número máximo de iteraciones
    """


    if f(a) * f(b) > 0:
        print("Error: f(a) y f(b) deben tener signos opuestos")
        return None

    print(f"{'Iter':<6} {'a':<15} {'b':<15} {'c':<15} {'f(c)':<15} {'Error':<15}")
    print("-" * 90)

    for i in range(max_iter):
        c = (a + b) / 2
        fc = f(c)
        error = abs(b - a) / 2

        print(f"{i+1:<6} {a:<15.8f} {b:<15.8f} {c:<15.8f} {fc:<15.8e} {error:<15.8e}")

        if error < tol or abs(fc) < tol:
            print(f"\nConvergió en {i+1} iteraciones")
            print(f"Raíz aproximada: x = {c:.8f}")
            return c


        if f(a) * fc < 0:
            b = c
        else:
            a = c

    print(f"\nNo convergió en {max_iter} iteraciones")
    return c

# Ejemplo de uso: Encontrar raíz de x^2 - 2 = 0
def f(x):
    return x**2 - 2

# Ejecutar
raiz = biseccion(f, 1, 2)
```

SALIDA DEL PROGRAMA:

```
= RESTART: C:/Users/Admin/Documents/UNAP/PRIGRAMACION NUMERICA/tarea segundo/METODO_BISE
CCION.py
Iter    a               b               c               f(c)            Error
-----------------------------------------------------------------------------------------
--
1       1.00000000      2.00000000      1.50000000      2.50000000e-01  5.00000000e-01
2       1.00000000      1.50000000      1.25000000      -4.37500000e-01 2.50000000e-01
3       1.25000000      1.50000000      1.37500000      -1.09375000e-01 1.25000000e-01
4       1.37500000      1.50000000      1.43750000      6.64062500e-02  6.25000000e-02
5       1.37500000      1.43750000      1.40625000      -2.24609375e-02 3.12500000e-02
6       1.40625000      1.43750000      1.42187500      2.17285156e-02  1.56250000e-02
7       1.40625000      1.42187500      1.41406250      -4.27246094e-04 7.81250000e-03
8       1.41406250      1.42187500      1.41796875      1.06353760e-02  3.90625000e-03
9       1.41406250      1.41796875      1.41601562      5.10025024e-03  1.95312500e-03
10      1.41406250      1.41601562      1.41503906      2.33554840e-03  9.76562500e-04
11      1.41406250      1.41503906      1.41455078      9.53912735e-04  4.88281250e-04
12      1.41406250      1.41455078      1.41430664      2.63273716e-04  2.44140625e-04
13      1.41406250      1.41430664      1.41418457      -8.20010900e-05 1.22070312e-04
14      1.41418457      1.41430664      1.41424561      9.06325877e-05  6.10351562e-05
15      1.41418457      1.41424561      1.41421509      4.31481749e-06  3.05175781e-05
16      1.41418457      1.41421509      1.41419983      -3.88433691e-05 1.52587891e-05
17      1.41419983      1.41421509      1.41420746      -1.72643340e-05 7.62939453e-06
18      1.41420746      1.41421509      1.41421127      -6.47477282e-06 3.81469727e-06
19      1.41421127      1.41421509      1.41421318      -1.07998130e-06 1.90734863e-06
20      1.41421318      1.41421509      1.41421413      1.61741718e-06  9.53674316e-07

Convergió en 20 iteraciones
Raíz aproximada: x = 1.41421413
```