Trabajo final física computacional

Errores de medición

Alumno: Gomez Jose Arian

Materia: Fisica computacional

En el siguiente trabajo se les será presentado un programa el cual tiene como finalidad el calcular los errores de medida con la finalidad de agilizar el trabajo a la hora de querer hacer cálculos con exactitud

(errores,pyhton,física,programa)

La importancia de este trabajo y el porque yo lo elegí es porque el tema de errores de medición nos acompaña en toda la carrera teniendo que utilizarlo cada vez que sacamos una medida y teniendo en cuenta como se puede propagar dicho error para así tener una exactitud acorde a los cálculos que precisamos en ese momento

Este tema de física se ve en física l siendo utilizado principalmente para evitar errores a la hora de realizar grandes cálculos con varias medidas, ya que al no tener en cuenta el error que posee el instrumento que nosotros utilizamos podríamos estar propagando el error y esto llevaría a un dato completamente erróneo generando de esta manera que nuestro experimento o medida salga mal

Principalmente las medidas se separan en 2 categorías las directas e indirectas:

Las directas son aquellas las cuales se pueden sacar mediante un elemento de medición, como lo puede ser una regla o un calibre.

Las medidas indirectas se le llaman a las medidas resultantes de someter a las medidas anteriores a cambios matemáticos, como lo pueden ser la resta, suma, multiplicación y división, ya que al implementar estos métodos el error se propaga haciendo que crezca, si esto no se tiene en cuenta puede generar una gran confusión o desperfecto a la hora de contemplar la medida

Representación de las medidas y sus errores:

Una vez que hemos medido una cierta magnitud x y que sabemos que su error es Δx, debemos expresar el resultado como: $x = (x0 \pm \Delta x)$ [unidades]

Los errores relativos son la división del error absoluto entre la medida, esto representa la calidad de la medida

- Error Absoluto $\longrightarrow \Delta x$
- Error relativo $\longrightarrow E_r = \frac{\Delta x}{|x_m|} \longrightarrow$ Calidad de la medición
- Error relativo porcentual $\longrightarrow E_{r\%} = 100\frac{\Delta x}{|x_m|}$

Tanto en la suma como en la resta el error se propaga:

$$A = A_m \pm \Delta A \qquad B = B_m \pm \Delta B$$

$$C = A \pm B$$

$$C_m = A_m \pm B_m \qquad \Delta C = \Delta A + \Delta B$$

Multiplicación por escalar:

$$A = A_m \pm \Delta A \qquad B = B_m \pm \Delta B$$

$$C = A/B$$

$$C_m = A_m/B_m \qquad \frac{\Delta C}{|C_m|} = \frac{\Delta A}{|A_m|} + \frac{\Delta B}{|B_m|}$$

Multiplicacion y dividir entre medidas hace que el error se propage de esta forma:

$$S_m = A_m\ B_m \qquad \frac{\Delta S}{|S_m|} = \frac{\Delta A}{|A_m|} + \frac{\Delta B}{|B_m|}$$

Potencia:

$$A = A_m \pm \Delta A$$

$$B = A^n$$

$$B_m = (A_m)^n \qquad \frac{\Delta B}{|B_m|} = |n|\frac{\Delta A}{|A_m|}$$

Programa de Python:

El programa realiza unas sencillas operaciones que agilizaran el tedioso trabajo de calcular individualmente los errores, reduciendo trabajos que llevarías bastante tiempo a solo algunos minutos.

A continuación adjuntare imágenes y explicare la utilidad de los bloques de código con la mayor brevedad posible:

Inicializando por definir las funciones para una mayor comodidad a la hora de generar las líneas de código

```python
import matplotlib.pyplot as plt
# Función para calcular el error porcentual de una medida
def calcular_error_medida(medida, error):
    error_relativo = abs(error) / medida
    error_porcentual = error_relativo * 100
    return error_porcentual
# Función para realizar la suma de dos medidas
def suma_medidas(medida1, error1, medida2, error2):
    suma = medida1 + medida2
    error = error1 + error2
    return suma, error
# Función para realizar la resta de dos medidas
def resta_medidas(medida1, error1, medida2, error2):
    resta = medida1 - medida2
    error = abs(error1) + abs(error2)
    return resta, error
# Función para realizar la multiplicación de dos medidas
def multiplicacion_medidas(medida1, error1, medida2, error2):
   multiplicacion = medida1 * medida2
   error_relativo = abs(error1 / medida1) + abs(error2 / medida2)
   error = error_relativo * multiplicacion
   return multiplicacion, error
# Función para realizar la división de dos medidas
def division_medidas(medida1, error1, medida2, error2):
    division = medida1 / medida2
    error_relativo = abs(error1 / medida1) + abs(error2 / medida2)
    error = error_relativo * division
    return division, error
# Función para multiplicar por un escalar
def multiplicar_por_escalar(medida, error, escalar):
    multiplicacion = medida * escalar
    error = abs(escalar) * error
    return multiplicacion, error
# Función para realizar la potencia de una medida
def potencia_medida(medida, error, exponente):
    potencia = medida ** exponente
    error_relativo = abs(exponente * error / medida)
    error = error_relativo * potencia
    return potencia, error
# Listas para almacenar los resultados de las operaciones y los errores
resultados = []
```

Todas estas funciones tienen la utilidad tanto de sumar,restar,multiplicar,dividir las medidas entre si y adicionalmente también están las opciones de multiplicar la medida por un escalar a la vez que también se puede elevarlo a la potencia que uno quiera.

Todos estos datos son almacenados en dos listas, una llamada resultados y otra llamada operaciones

```python
# Listas para almacenar los resultados de las operaciones y los errores
resultados = []
errores = []
operaciones = ['Suma', 'Resta', 'Multiplicación', 'División', 'Multiplicación por Escalar', 'Potencia']
```

Todo esto para que una vez el usuario ingrese el valor de las medidas junto con los errores establecidos queden guardados para poder ser utilizados de manera eficaz a la hora de realizar las posteriores operaciones anteriormente mencionadas.

Esta compuesto por un menú para facilitar la navegación por el código.

```
# Menú principal
while True:
    print("Menu:")
    print("1. Ingresar medidas")
    print("2. Sumar medidas")
    print("3. Restar medidas")
    print("4. Multiplicar medidas")
    print("5. Dividir medidas")
    print("6. Multiplicar por escalar")
    print("7. Calcular potencia")
    print("8. Mostrar gráfico de resultados")
    print("9. Mostrar gráfico de errores relativos")
    print("10. Mostrar resultados de las medidas")
    print("11. Salir")
    opcion = int(input("Ingrese una opción: "))
```

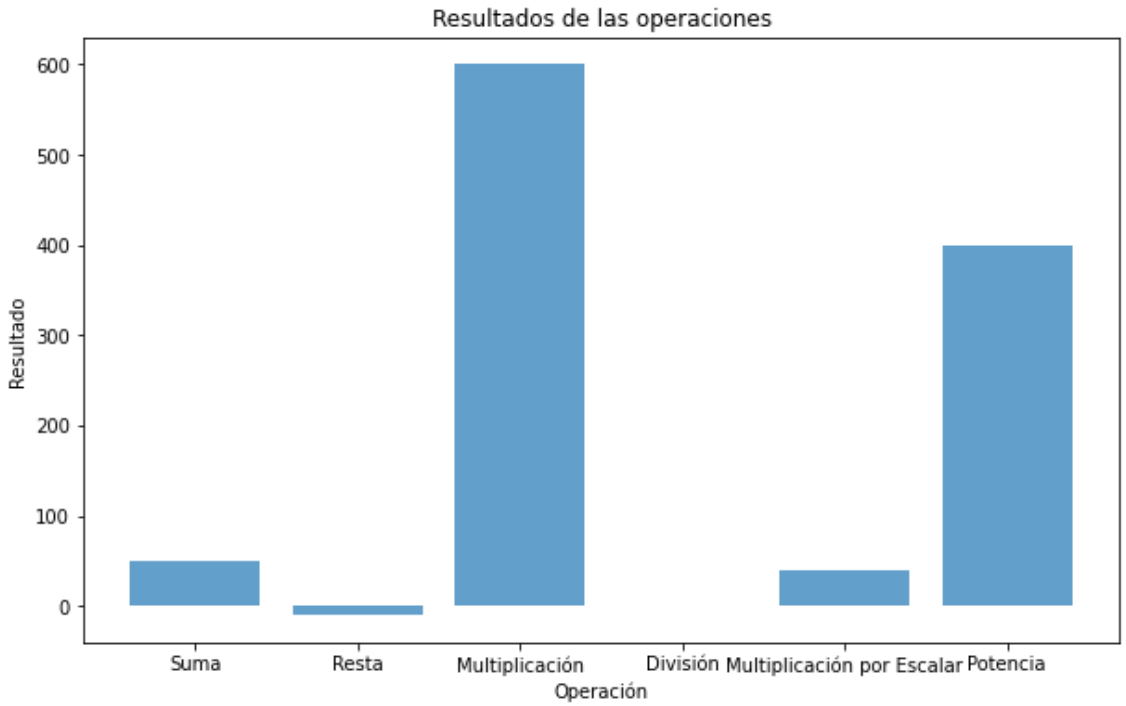
Luego se utiliza bloques de código para ilustrar el valor de las medidas y sus errores mediante graficos, como los mostrados a continuación:

Valores agregador

Medida1 = (20±0.1)

Medida2=(30±0.5)

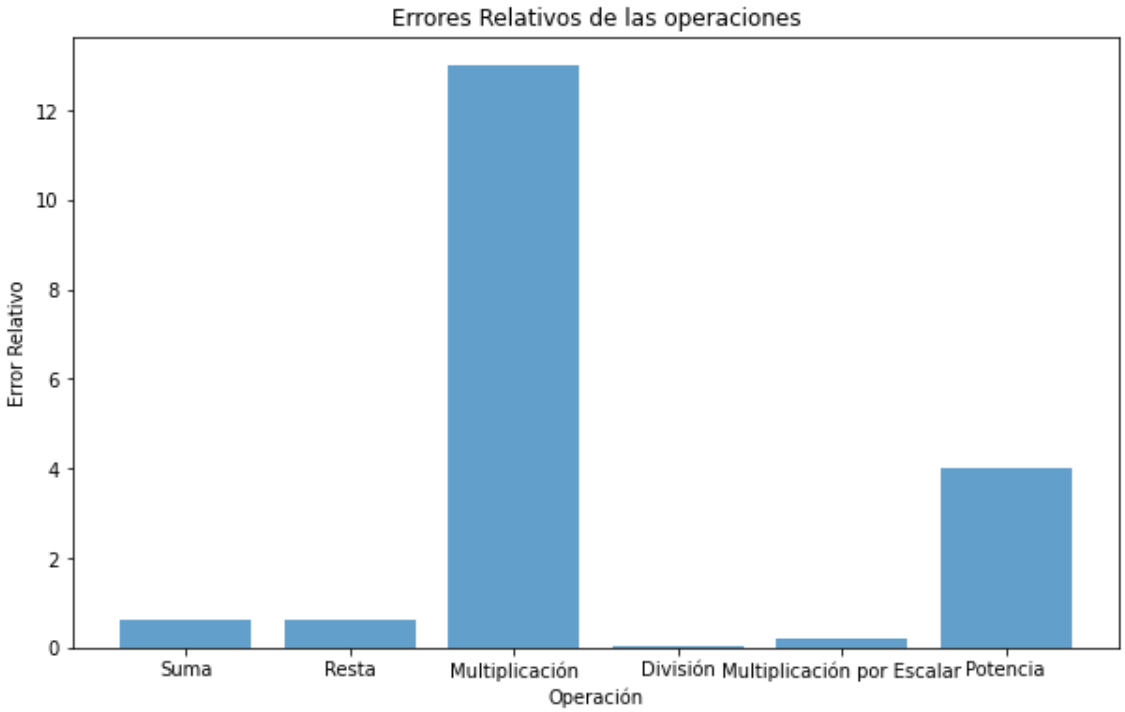
Grafico de resultados (medidas sin los errores)



resultados de las operaciones(medidas)

```
Suma: 50.0
Resta: -10.0
Multiplicación: 600.0
División: 0.6666666666666666
Multiplicación por Escalar: 40.0
Potencia: 400.0
```

Grafico de errores(sin medidas)



Código utlizado

```
import matplotlib.pyplot as plt
# Función para calcular el error porcentual de una medida
def calcular_error_medida(medida, error):
    error_relativo = abs(error) / medida
    error_porcentual = error_relativo * 100
    return error_porcentual
# Función para realizar la suma de dos medidas
def suma_medidas(medida1, error1, medida2, error2):
    suma = medida1 + medida2
```

```
    error = error1 + error2
    return suma, error
# Función para realizar la resta de dos medidas
def resta_medidas(medida1, error1, medida2, error2):
    resta = medida1 - medida2
    error = abs(error1) + abs(error2)
    return resta, error
# Función para realizar la multiplicación de dos medidas
def multiplicacion_medidas(medida1, error1, medida2, error2):
   multiplicacion = medida1 * medida2
   error_relativo = abs(error1 / medida1) + abs(error2 / medida2)
   error = error_relativo * multiplicacion
   return multiplicacion, error
# Función para realizar la división de dos medidas
def division_medidas(medida1, error1, medida2, error2):
    division = medida1 / medida2
    error_relativo = abs(error1 / medida1) + abs(error2 / medida2)
    error = error_relativo * division
    return division, error
# Función para multiplicar por un escalar
def multiplicar_por_escalar(medida, error, escalar):
    multiplicacion = medida * escalar
    error = abs(escalar) * error
    return multiplicacion, error
# Función para realizar la potencia de una medida
def potencia_medida(medida, error, exponente):
    potencia = medida ** exponente
    error_relativo = abs(exponente * error / medida)
    error = error_relativo * potencia
    return potencia, error
# Listas para almacenar los resultados de las operaciones y los errores
```

```
resultados = []
errores = []
operaciones = ['Suma', 'Resta', 'Multiplicación', 'División', 'Multiplicación por Escalar', 'Potencia']
# Menú principal
while True:
    print("Menu:")
    print("1. Ingresar medidas")
    print("2. Sumar medidas")
    print("3. Restar medidas")
    print("4. Multiplicar medidas")
    print("5. Dividir medidas")
    print("6. Multiplicar por escalar")
    print("7. Calcular potencia")
    print("8. Mostrar gráfico de resultados")
    print("9. Mostrar gráfico de errores relativos")
    print("10. Mostrar resultados de las medidas")
    print("11. Salir")
    opcion = int(input("Ingrese una opción: "))
    if opcion == 1:       # Solicita al usuario ingresar las medidas y sus errores
        medida1 = float(input("Ingrese la primera medida: "))
        error1 = float(input("Ingrese el error de la primera medida: "))
        medida2 = float(input("Ingrese la segunda medida: "))
        error2 = float(input("Ingrese el error de la segunda medida: "))
    elif opcion == 2:       # Realiza la suma de las medidas
        suma,error = suma_medidas(medida1, error1, medida2, error2)
        resultados.append(suma)
        errores.append(error)
        print("La suma de las medidas es:", suma)
        print("El error de la suma es:", error)
        print("El error porcentual de la suma es:", calcular_error_medida(suma, error))
```

```
    elif opcion == 3:       # Realiza la resta de las medidas
        resta,error = resta_medidas(medida1, error1, medida2, error2)
        resultados.append(resta)
        errores.append(error)
        print("La resta de las medidas es:", resta)
        print("El error de la resta es:", error)
        print("El error porcentual de la resta es:", calcular_error_medida(resta, error))
    elif opcion == 4:       # Realiza la multiplicación de las medidas
        multiplicacion,error = multiplicacion_medidas(medida1, error1, medida2, error2)
        resultados.append(multiplicacion)
        errores.append(error)
        print("La multiplicación de las medidas es:", multiplicacion)
        print("El error de la multiplicación es:", error)
        print("El error porcentual de la multiplicación es:", calcular_error_medida(multiplicacion,
error))
    elif opcion == 5:       # Realiza la división de las medidas
        division,error = division_medidas(medida1, error1, medida2, error2)
        resultados.append(division)
        errores.append(error)
        print("La división de las medidas es:", division)
        print("El error de la división es:", error)
        print("El error porcentual de la división es:", calcular_error_medida(division, error))
    elif opcion == 6:       # Multiplicar por escalar
        escalar = float(input("Ingrese el valor del escalar: "))
        multiplicacion, error = multiplicar_por_escalar(medida1, error1, escalar)
        resultados.append(multiplicacion)
        errores.append(error)
        print("El resultado de la multiplicación por escalar es:", multiplicacion)
        print("El error de la multiplicación por escalar es:", error)
        print("El error porcentual de la multiplicación por escalar es:",
calcular_error_medida(multiplicacion, error))
    elif opcion == 7:       # Calcular potencia
```

```
        exponente = float(input("Ingrese el valor del exponente: "))
        potencia, error = potencia_medida(medida1, error1, exponente)
        resultados.append(potencia)
        errores.append(error)
        print("El resultado de la potencia es:", potencia)
        print("El error de la potencia es:", error)
        print("El error porcentual de la potencia es:", calcular_error_medida(potencia, error))
    elif opcion == 8:       # Muestra el gráfico de resultados
        if len(resultados) == 0:
            print("No hay resultados para mostrar. Realice alguna operación primero.")
        else:
            plt.figure(figsize=(10, 6))
            plt.bar(operaciones, resultados, align='center', alpha=0.7, capsize=10)
            plt.xlabel('Operación')
            plt.ylabel('Resultado')
            plt.title('Resultados de las operaciones')
            plt.show()
    elif opcion == 9:       # Muestra el gráfico de errores relativos
        if len(errores) == 0:
            print("No hay errores para mostrar. Realice alguna operación primero.")
        else:
            plt.figure(figsize=(10, 6))
            plt.bar(operaciones, errores, align='center', alpha=0.7, capsize=10)
            plt.xlabel('Operación')
            plt.ylabel('Error Relativo')
            plt.title('Errores Relativos de las operaciones')
            plt.show()
    elif opcion == 10:      # Muestra los resultados de las medidas
        print("Resultados de las medidas:")
        for i in range(len(resultados)):
            print(f"{operaciones[i]}: {resultados[i]}")
```

```
elif opcion == 11:
    break
else:       print("Opción inválida. Por favor, seleccione una opción válida.")
```

información: apuntes física l, apuntes y trabajos física computacional