Punto 2.4

N	a	a-EE	a+EE	EE
2	0,987654321	0,2892772532	0,2892772532	0,7071067812
3	0,987654321	0,4938271605	0,4938271605	0,5
5	0,987654321	0,6384657871	0,6384657871	0,3535533906
10	0,987654321	0,7548619651	0,7548619651	0,2357022604
20	0,987654321	0,8274356335	0,8274356335	0,1622214211
30	0,987654321	0,8579689552	0,8579689552	0,1313064329
50	0,987654321	0,8878861685	0,8878861685	0,1010152545
80	0,987654321	0,9090807621	0,9090807621	0,07955572842
100	0,987654321	0,9174647848	0,9174647848	0,07106690545
1000	0,987654321	0,9655586484	0,9655586484	0,02237186851
3000	0,987654321	0,9749016329	0,9749016329	0,01291209668
5000	0,987654321	0,97777679	0,97777679	0,01000100015
10000	0,987654321	0,9806702011	0,9806702011	0,007071421392
50000	0,987654321	0,9845310526	0,9845310526	0,003162309283
70000	0,987654321	0,9850146849	0,9850146849	0,002672631509

N	a	a-EE	a+EE	EE
2	0,123456789	0,03615965631	0,03615965631	0,7071067812
3	0,123456789	0,0617283945	0,0617283945	0,5
5	0,123456789	0,07980822266	0,07980822266	0,3535533906
10	0,123456789	0,09435774477	0,09435774477	0,2357022604
20	0,123456789	0,1034294532	0,1034294532	0,1622214211
30	0,123456789	0,1072461184	0,1072461184	0,1313064329
50	0,123456789	0,11098577	0,11098577	0,1010152545
80	0,123456789	0,1136350942	0,1136350942	0,07955572842
100	0,123456789	0,114683097	0,114683097	0,07106690545
1000	0,123456789	0,12069483	0,12069483	0,02237186851
3000	0,123456789	0,121862703	0,121862703	0,01291209668
5000	0,123456789	0,1222220976	0,1222220976	0,01000100015
10000	0,123456789	0,122583774	0,122583774	0,00707142139
50000	0,123456789	0,1230663805	0,1230663805	0,00316230928
70000	0,123456789	0,1231268345	0,1231268345	0,00267263150

Para el primer caso se requiere un N del orden de 10000 para tener la segunda cifra significativa. Para el segundo caso se requiere un N del orden de 50000 para obtener las tres cifras significativas.

Punto 3.2

(i) $\int_{-\infty}^{\infty} P_U\,(x,\bar{x},a)dx = \int_{\bar{x}-\frac{a}{2}}^{\bar{x}+\frac{a}{2}} \frac{1}{a}dx = \frac{1}{a}\left(\bar{x}+\frac{a}{2}-\bar{x}+\frac{a}{2}\right) = 1$

(ii) $\mu = \frac{1}{a}\int_{\bar{x}-a/2}^{\bar{x}+a/2} xdx = \frac{1}{2a}\left(\left(\bar{x}+\frac{a}{2}\right)^2 - \left(\bar{x}-\frac{a}{2}\right)^2\right) = \frac{1}{2a}\left(\bar{x}^2+\bar{x}a+\frac{a^2}{4}-\bar{x}^2+\bar{x}a-\frac{a^2}{4}\right) = \frac{1}{2a}(2\bar{x}a) = \bar{x}$

(iii) $\sigma^2 = (\overline{x^2} - \bar{x}^2)$

Pero sabemos que $(\overline{x^2} - \bar{x}^2) = \overline{x^2} - 2\bar{x}^2 + \bar{x}^2 = \overline{x^2} - \overline{2x\bar{x}} + \bar{x}^2 = \overline{(x-\bar{x})^2}$

Luego, usando un cambio de variable $u = x - \bar{x}$ y hallamos el valor esperado de σ^2

$$\sigma^2 = \frac{1}{a}\int_{-a/2}^{a/2} u^2 du = \frac{1}{3a}\left(\left(\frac{a}{2}\right)^3 - -\left(\frac{a}{2}\right)^3\right) = \frac{1}{3a}\frac{2a^3}{8} = \frac{a^2}{12}$$

Por lo tanto $\sigma = \sqrt{\frac{a^2}{12}}$

Punto 3.7

N	Occurrence	NxO	
1	1	1	0,00534358699
2	2	4	0,01925533934
3	3	9	0,04625707957
5	6	30	0,1201281506
6	9	54	0,144291859
7	11	77	0,148556643
8	8	64	0,1338290448
9	8	72	0,1071657868
10	6	60	0,07723327396
11	2	22	0,05060111052
12	1	12	0,03038974741
13	1	13	0,0168473666
TOTAL			
	58	418	
Mean count			
7,206896552			
Probabilidad para el número esperado de occurrencuas de 5 cuentas o menos:			
0,1909841565			
Número de ocurrencias esperadas para 5 cuentas o menos:			
11,07708108			
Probabilidad para el número esperado de occurrencuas de 19 cuentas o menos:			
0,8998989886			
Probabilidad para el número esperado de occurrencuas de 20 cuentas o más:			
0,1001010114			
Número de ocurrencias esperadas para 20 cuentas o más:			
5,805858659			

Punto 4.2

$$A = 12.3 \pm 0.4$$

$$B = 5.6 \pm 0.8$$

$$C = 89.0 \pm 0.2$$

En general se tiene que

$$\sigma = \sqrt{\left(\frac{\partial Z}{\partial A}\sigma_A\right)^2 + \left(\frac{\partial Z}{\partial B}\sigma_B\right)^2 + \left(\frac{\partial Z}{\partial C}\sigma_C\right)^2}$$

Luego, para la suma y la resta

$$\sigma = \sqrt{(1 * 0.4)^2 + (1 * 0.8)^2} = 0.89$$

Para la división entre la resta y la suma

$$\sigma = \sqrt{\left(\frac{[(17.9) - (6.7)]}{(17.9)^2} * 0.4\right)^2 + \left(\frac{[-(17.9) - (6.7)]}{(17.9)^2} * 0.8\right)^2} = \sqrt{\left(\frac{4.48}{320.41}\right)^2 + \left(\frac{19.68}{320.41}\right)^2}$$
$$= 0.0629$$

Para la división entre la multiplicación y C

$$\sigma = \sqrt{\left(\frac{B}{C} * 0.4\right)^2 + \left(\frac{A}{C} * 0.8\right)^2 + \left(\frac{AB}{C^2} * 0.2\right)^2} = 0.1134$$

Para el arcsin

$$\sigma = \sqrt{\left(\frac{1}{C\sqrt{1 - (B/C)^2}} * 0.8\right)^2 + \left(\frac{1}{C^2\sqrt{1 - (B/C)^2}} * 0.2\right)^2} = 0.00901$$

Para la multiplicación con potencias

$$\sigma = \sqrt{(B^2C^3 * 0.4)^2 + (2ABC^3 * 0.8)^2 + (3AB^2C^2 * 0.2)^2} = 0.7822 * 10^8$$

Para el logaritmo natural

$$\sigma = \sqrt{\left(\frac{BC}{ABC} * 0.4\right)^2 + \left(\frac{AC}{ABC} * 0.8\right)^2 + \left(\frac{AB}{ABC} * 0.2\right)^2} = 0.147$$

Para el exponencial

$$\sigma = \sqrt{(BC\exp(ABC) * 0.4)^2 + (AC\exp(ABC) * 0.8)^2 + (AB\exp(ABC) * 0.2)^2}$$

Dado que los productos y las exponenciales en este caso son muy grandes, se usa el ln(Z) para poder obtener una expresión, de este modo

$$\sigma = \sqrt{(BC * 0.4)^2 + (AC * 0.8)^2 + (AB * 0.2)^2} = 898.27$$

Para el numeral ix

$$\sigma = \sqrt{(1 * 0.4)^2 + \left(\frac{sec^2(B/C)}{C} * 0.8\right)^2 + \left(\frac{-Bsec^2(B/C)}{C^2} * 0.2\right)^2} = 0.4001$$

Para el último numeral ocurre algo similar al caso del exponencial, de modo que usamos un logaritmo en base 10 usando que

$$log_{10}Z = \frac{lnZ}{ln10}$$

De tal modo que

$$log_{10}\sigma = \frac{ln898.27}{ln10} = 390.11$$

(i) $Z = A + B = 17.9 \pm 0.9$

(ii) $Z = A - B = 6.7 \pm 0.9$

(iii) $Z = \frac{A-B}{A+B} = 0.37 \ \pm 0.06$

(iv) $Z = \frac{AB}{C} = 0.77 \ \pm 0.11$

(v) $Z = arcsin\left(\frac{B}{C}\right) = 0.063 \pm 0.009$

(vi) $Z = AB^2C^3 = (2.7 \pm 0.8) \times 10^8$

(vii) $Z = \ln(ABC) = 8.7 \pm 0.1$

(viii) $Z = \exp(ABC) => \ln(Z) = ABC = 6130.32 \pm 898.27$

(ix) $Z = A + \tan\left(\frac{B}{C}\right) = 12.4 \pm 0.4$

(x) $Z = 10^{ABC} => log_{10}Z = ABC = 6130.32 \pm 390.11$

```
import numpy as np
import matplotlib.pyplot as plt
```

Punto 6.2

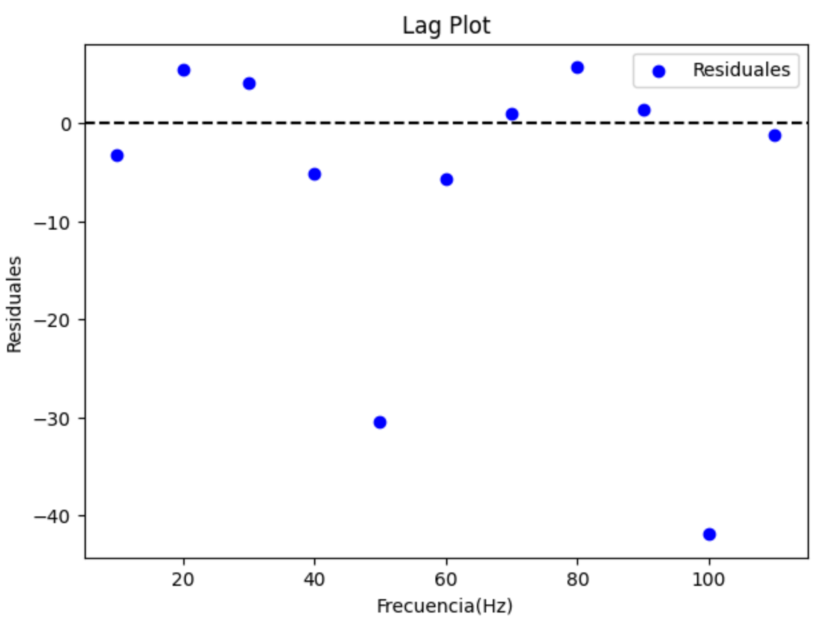
```
Hz = np.array([10,20,30,40,50,60,70,80,90,100,110])
mV = np.array([16,45,64,75,70,115,142,167,183,160,221])
E = np.array([5,5,5,5,30,5,5,5,5,30,5])
N = len(Hz)
ws = 1/(E**2)

S_wx2 = np.sum(ws*Hz**2)
S_wx = np.sum(ws*Hz)
S_wy = np.sum(ws*mV)
S_wxy = np.sum(ws*Hz*mV)
S_w = np.sum(ws)

delta = S_w*S_wx2 - (S_wx**2)
m = (S_w*S_wxy - S_wx*S_wy)/delta
c = (S_wx2*S_wy - S_wx*S_wxy)/delta
a_c = np.sqrt(S_wx2/delta)
a_m = np.sqrt(S_w/delta)
print(m,c,a_m,a_c)
```

```
2.0284648216102745 -0.9474964662767381 0.05197830827721802 3.386858673521736
```

El ajuste resultante sería $$ F = (2.03x\pm 0.05) -(1\pm 3)$$

Comparando el resultado con los valores de la tabla 6.1 b, podemos observar que coinciden.

```
F_fit = m * Hz + c
res = mV - F_fit

plt.figure(figsize=(7,5))
plt.axhline(0, color='black', linestyle='--')
plt.scatter(Hz, res, color='blue', label="Residuales")
plt.xlabel("Frecuencia(Hz)")
plt.ylabel("Residuales")
plt.title("Lag Plot")
plt.legend()
plt.grid(False)
plt.show()
```



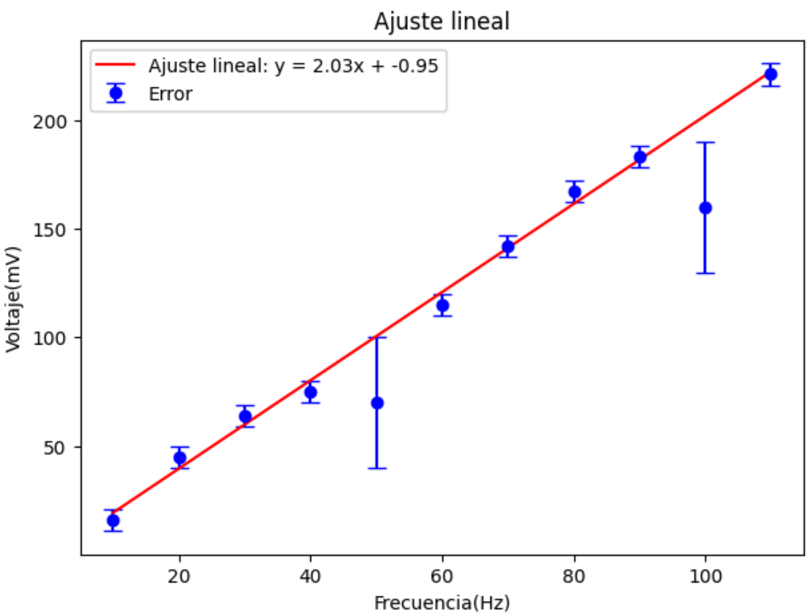

```
D_num = np.sum((res[1:] - res[:-1])**2)
D_den = np.sum(res**2)
```

```
D = (D_num/D_den)
print(D)
```

```
1.772945759877919
```

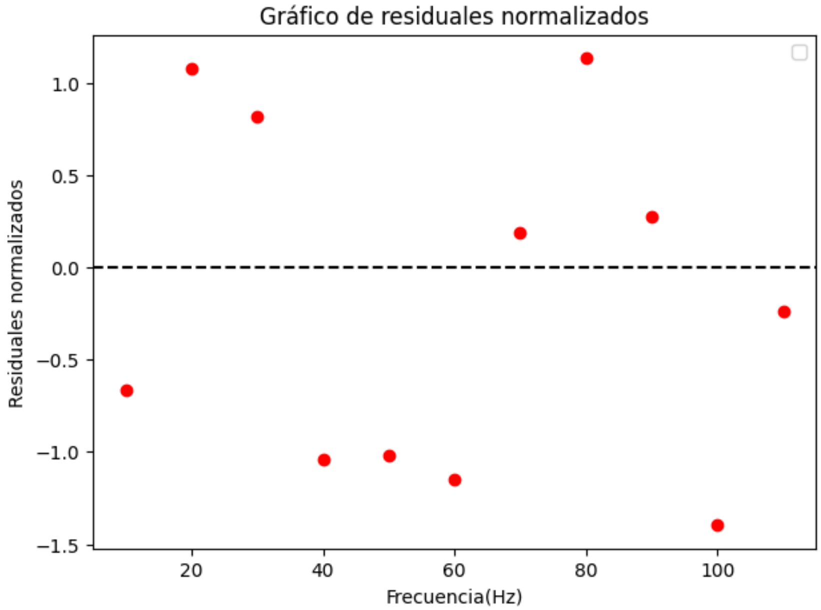
Punto 6.3

```
Hz_fit = np.linspace(min(Hz), max(Hz), 200)
mV_fit = m * Hz_fit + c

plt.figure(figsize=(7,5))
plt.errorbar(Hz, mV, yerr=E, fmt='o', capsize=5, label="Error", color="blue")
plt.plot(Hz_fit, mV_fit, 'r-', label=f"Ajuste lineal: y = {m:.2f}x + {c:.2f}")
plt.xlabel("Frecuencia(Hz)")
plt.ylabel("Voltaje(mV)")
plt.title("Ajuste lineal")
plt.legend()
plt.grid(False)
plt.show()
```




```
res_norm = (mV - F_fit)/E

plt.figure(figsize=(7,5))
plt.axhline(0, color='black', linestyle='--')
plt.scatter(Hz, res_norm, color='red')
plt.xlabel("Frecuencia(Hz)")
plt.ylabel("Residuales normalizados")
plt.title("Gráfico de residuales normalizados")
plt.legend()
plt.grid(False)
plt.show()
```

```
/tmp/ipython-input-2149316552.py:9: UserWarning: No artists with labels found to put in legend.  Note that artists whose label start wit
  plt.legend()
```



Laboratorio Intermedio
Tarea Estadistica
September 3, 2025

Name: *Sergio Montoya Ramirez*
Angelica Lopez Duarte

Nota

Durante el transcurso de este trabajo se hara uso de varios scripts en python (Particularmente para usar sympy que opinamos que es una de las maravillas del mundo moderno). Se pondra aqui el codigo utilizado sin embargo recomendamos fervientemente que lo mire en el repo de github: `https://github.com/demon-s1e7j/Universidad/tree/main/Semestre10/LabIntermedio/Tareas/Tarea2/code`. Este ya esta comodamente organizado para tener un proyecto de uv con el que manejar los requerimientos. Eh de confesar que no estan pensados para que alguien mas los ejecute y entienda lo que retorna pero creo que sigue siendo mucho mejor que simplemente quedarse con la implementacion mostrada en este documento.

1 Punto 2.1

Podemos usar el codigo

```python
from functools import wraps
from typing import Sequence, Union
import math as mt

Numeric = Union[int, float, complex]


def mean(data: Sequence[Numeric]) -> Numeric:
    return sum(data) / len(data)


def autofill_valmean(func):
    @wraps(func)
    def wrapper(
            data: Sequence[Numeric],
            *args,
            valmean: Numeric | None = None,
            **kwargs):
        if valmean is None:
            valmean = mean(data)
        return func(data, *args, valmean=valmean, **kwargs)
    return wrapper
```

```python
@autofill_valmean
def list_deviation(
        data: Sequence[Numeric],
        valmean: Numeric | None = None) -> Sequence[Numeric]:
    def deviation(x): return abs(valmean - x)
    return [deviation(x) for x in data]


@autofill_valmean
def standard_rough_and_ready(
        data: Sequence[Numeric],
        valmean: Numeric | None = None) -> Numeric:
    return (2 / 3) * (max(list_deviation(data, valmean=valmean)))


@autofill_valmean
def standard_2_3(
        data: Sequence[Numeric],
        valmean: Numeric | None = None) -> Numeric:
    return mt.sqrt(
        (sum(list_deviation(data, valmean=valmean))) / (len(data) - 1))


def standar_error(standard_deviation: Numeric, N: int) -> Numeric:
    return standard_deviation / mt.sqrt(N)


data = [25.8, 26.2, 26.0, 26.5, 25.8, 26.1, 25.8, 26.3]

if __name__ == "__main__":
    promedio = mean(data)
    desviacion_rar = standard_rough_and_ready(data)
    desviacion_2_3 = standard_2_3(data)
    error_rar = standar_error(desviacion_rar, len(data))
    error_2_3 = standar_error(desviacion_2_3, len(data))
    print(f"""\
    {promedio=}
    {desviacion_rar=}
    {desviacion_2_3=}
    {error_rar=}
    {error_2_3=}
    """)
```

lo que nos da:

```
$ uv run punto_2_1.py
```

```
promedio=26.0625
desviacion_rar=0.2916666666666663
desviacion_2_3=0.4928053803045810 4
error_rar=0.10311973892303816
error_2_3=0.17423301310929235
```

2 Punto 2.6

2.1

Numero de datos: 5

- $\alpha = 0.01913 \approx 0.019$
- $\bar{\delta} = 3.27346 \approx 3.273$

Resultado: 3.273 ± 0.019

2.2

Numero de datos: 50

- $\alpha = 0.002506 \approx 0.0025$
- $\bar{\delta} = 3.26513 \approx 3.2651$

Resultado: 3.25513 ± 0.019

2.3

Numero de datos: 500

- $\alpha = 0.000270 \approx 0.000270$
- $\bar{\delta} = 3.26681 \approx 3.26681$

Resultado: 3.273 ± 0.019

3 Punto 3.4

Podemos crear la funcion (sin integrarla) en sympy como

$$\frac{\sqrt{2}e^{-\frac{(-\mu+x)^2}{2\sigma^2}}}{2\sqrt{\pi}\sigma}$$

y con eso podemos implementar un codigo:

```python
import sympy
from tabulate import tabulate
import os

os.system("clear")


x, mean, std_dev = sympy.symbols('x mu sigma')

gaussian = (1 / (std_dev * sympy.sqrt(2 * sympy.pi))) * \
    sympy.exp(-((x - mean)**2) / (2 * std_dev**2))


def replicate_table_sympy(mean_val=0, std_dev_val=1):
    integral_sym = sympy.integrate(gaussian, x)

    ranges = [1, 1.65, 2, 2.58, 3, 4, 5]
    table_data = []

    for r in ranges:
        x_min = mean_val - r * std_dev_val
        x_max = mean_val + r * std_dev_val

        result = (integral_sym.subs(x, x_max) - integral_sym.subs(x, x_min)
        result = result.subs({mean: mean_val, std_dev: std_dev_val})

        fraction_in_range = float(result)

        fraction_out_range = 1 - fraction_in_range

        range_str = f"\\pm {r} \\sigma"
        in_range_percent = f"{fraction_in_range * 100:.2f}%"
        out_range_percent = f"{fraction_out_range * 100:.2f}%"
        table_data.append([range_str, in_range_percent, out_range_percent])

    headers = [
        "Centrado en media",
        "Medidas dentro del rango",
        "Medidas fuera del rango"]

    print(tabulate(table_data, headers=headers, tablefmt="fancy_grid"))
    print(tabulate(table_data, headers=headers, tablefmt="latex"))


replicate_table_sympy()
```

Que nos da como resultado:

Centrado en media	Medidas dentro del rango	Medidas fuera del rango
$\pm 1\sigma$	68.27%	31.73%
$\pm 1.65\sigma$	90.11%	9.89%
$\pm 2\sigma$	95.45%	4.55%
$\pm 2.58\sigma$	99.01%	0.99%
$\pm 3\sigma$	99.73%	0.27%
$\pm 4\sigma$	99.99%	0.01%
$\pm 5\sigma$	100.00%	0.00%

Que coincide con lo que esperamos

4 Punto 4.1

4.1 $Z = 2A$

- $\frac{dZ}{dA} = 2$
- $\delta Z = \left|\frac{dZ}{dA}\right| \_\bar{A}\delta A = |2|(0.005) \approx 0.01$
- $Z \approx 18.54800 \pm 0.01000$

4.2 $Z = A/2$

- $\frac{dZ}{dA} = \frac{1}{2}$
- $\delta Z = \left|\frac{dZ}{dA}\right| \_\bar{A}\delta A = |0.5|(0.005) \approx 0.0025$
- $Z \approx 4.63700 \pm 0.00250$

4.3 $Z = \frac{A-1}{A+1}$

- $\frac{dZ}{dA} = -\frac{A-1}{(A+1)^2} + \frac{1}{A+1}$
- $\delta Z = \left|\frac{dZ}{dA}\right| \_\bar{A}\delta A = |0.018947|(0.005) \approx 9.4737e - 05$
- $Z \approx 0.80533 \pm 0.00009$

4.4 $Z = \frac{A^2}{A-2}$

- $\frac{dZ}{dA} = -\frac{A^2}{(A-2)^2} + \frac{2A}{A-2}$
- $\delta Z = \left|\frac{dZ}{dA}\right| \_\bar{A}\delta A = |0.9244|(0.005) \approx 0.004622$
- $Z \approx 11.82390 \pm 0.00462$

4.5 $Z = \arcsin(\frac{1}{A})$

- $\frac{dZ}{dA} = -\frac{1}{A^2\sqrt{1-\frac{1}{A^2}}}$
- $\delta Z = \left|\frac{dZ}{dA}\right| \_\bar{A}\delta A = |-0.011695|(0.005) \approx 5.8476e-05$
- $Z \approx 0.10804 \pm 0.00006$

4.6 $Z = \sqrt{A}$

- $\frac{dZ}{dA} = \frac{1}{2\sqrt{A}}$
- $\delta Z = \left|\frac{dZ}{dA}\right| \_\bar{A}\delta A = |0.16419|(0.005) \approx 0.00082093$
- $Z \approx 3.04532 \pm 0.00082$

4.7 $Z = \ln(\frac{1}{\sqrt{A}})$

- $\frac{dZ}{dA} = -\frac{1}{2A}$
- $\delta Z = \left|\frac{dZ}{dA}\right| \_\bar{A}\delta A = |-0.053914|(0.005) \approx 0.00026957$
- $Z \approx -1.11361 \pm 0.00027$

4.8 $Z = \exp(A^2)$

- $\frac{dZ}{dA} = 2Ae^{A^2}$
- $\delta Z = \left|\frac{dZ}{dA}\right| \_\bar{A}\delta A = |4.1754e+38|(0.005) \approx 2.0877e+36$
- $Z \approx 2.251e+37 \pm 2.088e+36$

4.9 $Z = A + \sqrt{\frac{1}{A}}$

- $\frac{dZ}{dA} = 1 - \frac{\sqrt{\frac{1}{A}}}{2A}$
- $\delta Z = \left|\frac{dZ}{dA}\right|_{\bar{A}} \delta A = |0.9823|(0.005) \approx 0.0049115$
- $Z \approx 9.60237 \pm 0.00491$

4.10 $Z = 10^A$

- $\frac{dZ}{dA} = 10^A \log(10)$
- $\delta Z = \left|\frac{dZ}{dA}\right|_{\bar{A}} \delta A = |4.3273e + 09|(0.005) \approx 2.1636e + 07$
- $Z \approx 1.879e + 09 \pm 2.164e + 07$

5 Punto 4.4

Podemos reescribir la formula que nos pidieron en sympy y despejar la ecuacion 4.10 y con eso encontrar los resultados. Si lo hacemos para un valor generico (Es decir, θ_i y θ_t) los resultados son

$$R = \frac{\tan^2(\theta_i - \theta_t)}{\tan^2(\theta_i + \theta_t)}$$

$$\delta_R = \sqrt{\delta_{\theta_i}^2 \left(\frac{(2\tan^2(\theta_i - \theta_t) + 2)\tan(\theta_i - \theta_t)}{\tan^2(\theta_i + \theta_t)} + \frac{(-2\tan^2(\theta_i + \theta_t) - 2)\tan^2(\theta_i - \theta_t)}{\tan^3(\theta_i + \theta_t)}\right)^2 + \delta_{\theta_t}^2 \left(\frac{(-2\tan^2(\theta_i - \theta_t) - 2)\tan(\theta_i - \theta_t)}{\tan^2(\theta_i + \theta_t)} + \frac{(-2\tan^2(\theta_i + \theta_t) - 2)\tan^2(\theta_i - \theta_t)}{\tan^3(\theta_i + \theta_t)}\right)^2}$$

Ahora reemplazando a los valores que nos dieron para θ_i y θ_t como puede ver en el siguiente script:

```
import sympy as sp
import os

os.system("clear")

s_theta_i, s_theta_t, delta_theta_i, delta_theta_t = sp.symbols(
```

```
    "theta_i theta_t delta_{theta_i} delta_{theta_t}")


def R(theta_i=s_theta_i, theta_t=s_theta_t):
    num = sp.Pow(sp.tan(theta_i - theta_t), 2)
    den = sp.Pow(sp.tan(theta_i + theta_t), 2)
    return num / den


def dR_dA(R_expr, theta=s_theta_i):
    return sp.diff(R_expr, theta)


def delta_R(R_expr, e_theta_i, e_theta_t):
    dR_dtheta_i = dR_dA(R_expr, theta=s_theta_i)
    dR_dtheta_t = dR_dA(R_expr, theta=s_theta_t)
    return sp.sqrt((dR_dtheta_i * e_theta_i)**2 + (dR_dtheta_t * e_theta_t)


expresion = R()
error = delta_R(expresion, delta_theta_i, delta_theta_t)

print("-" * 60)
sp.print_latex(expresion)
print("\n")
sp.print_latex(error)
print("-" * 60)


def rad_of_grad(x): return (x * sp.pi) / 180


values = {
    s_theta_i: 45.0,
    delta_theta_i: 0.1,
    s_theta_t: 34.5,
    delta_theta_t: 0.2}
values = {key: rad_of_grad(value) for key, value in values.items()}

val_expresion = expresion.evalf(subs=values)
val_error = error.evalf(subs=values)

print(val_expresion)
print(val_error)
sp.print_latex(val_expresion)
```

```
sp.print_latex(val_error)
```

al ejecutar este script nos devuelve:

$$0.00118 \pm 9 \cdot 10^{-5}$$

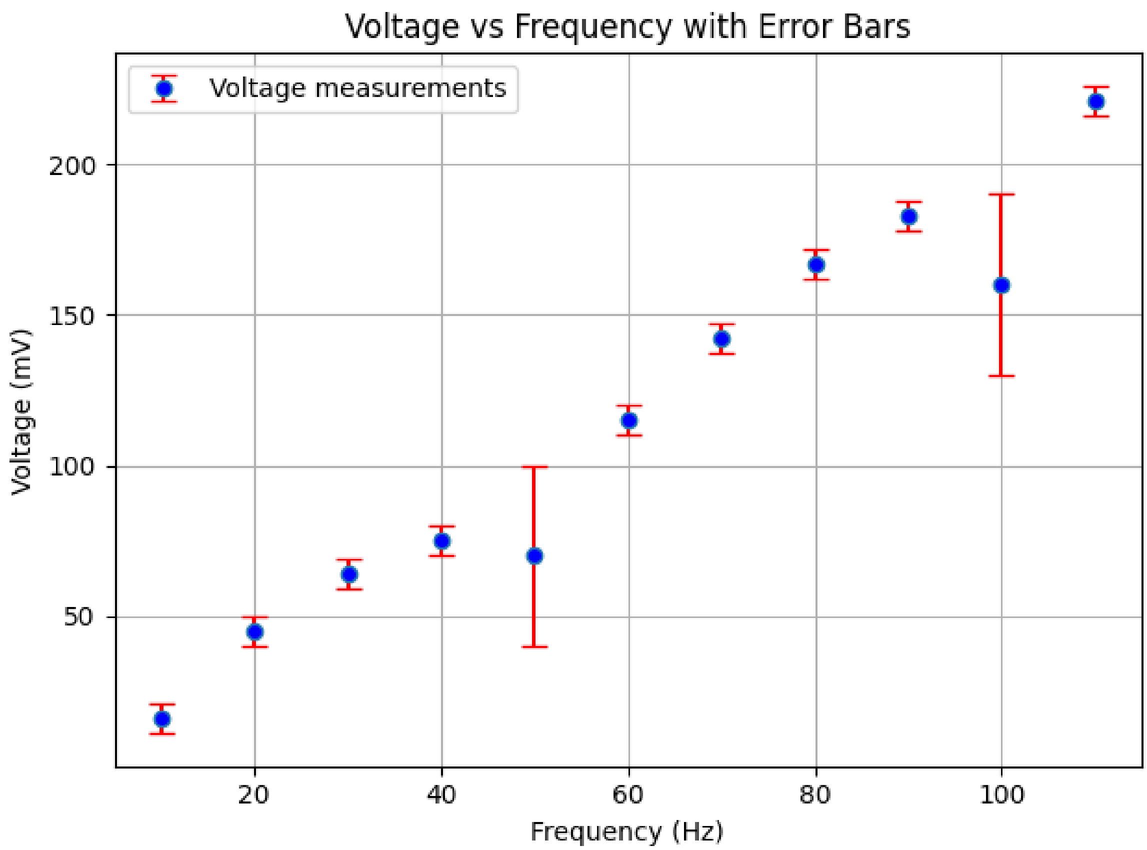
6 Punto 6.1

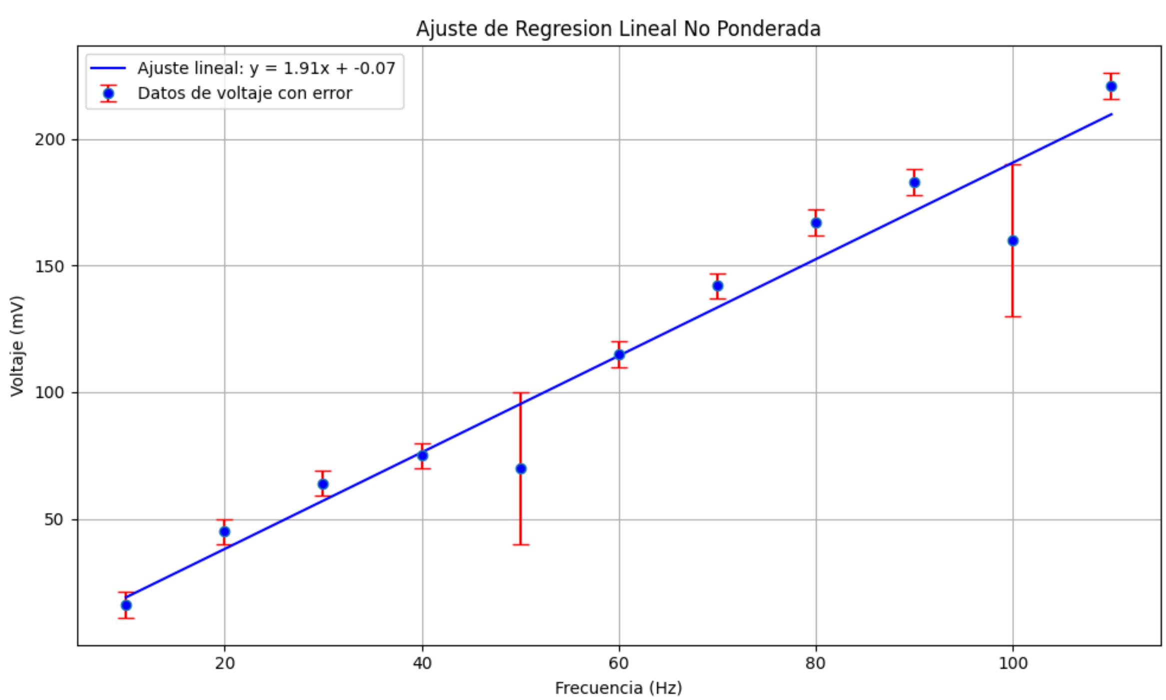
Para esto de nuevo usamos simplemente un script de python utilizando las ecuaciones 5.1 a 5.6 del libro. Las puede encontrar en el script como:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import os

os.system("clear")

data = pd.read_csv("./data_6_1.csv")
print(data)

x, y, yerr = data["Frecuency (Hz)"].to_numpy(
), data["Voltage (mV)"].to_numpy(), data["Error (mV)"].to_numpy()

plt.errorbar(
    x,
    y,
    yerr=yerr,
    fmt='o',
    ecolor='red',
    capsize=5,
    linestyle='none',
    markerfacecolor='blue',
    label="Voltage measurements"
)

plt.xlabel("Frequency (Hz)")
plt.ylabel("Voltage (mV)")
plt.title("Voltage vs Frequency with Error Bars")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.savefig("punto_6_1_a.png")

sum_x, sum_y, sum_x_sq, sum_xy = (np.sum(x) for x in [x, y, x**2, x * y])
```

```
delta = len(x) * np.sum(x**2) - (np.sum(x))**2

num_c = (sum_x_sq * sum_y) - (sum_x * sum_xy)
c = num_c / delta

num_m = (len(x) * sum_xy) - (sum_x * sum_y)
m = num_m / delta

alpha_CU = np.sqrt(1 / (len(x) - 2) * np.sum((y - m * x - c)**2))

alpha_c = alpha_CU * np.sqrt(sum_x_sq / delta)
alpha_m = alpha_CU * np.sqrt(len(x) / delta)

plt.figure(figsize=(10, 6))

plt.errorbar(
    x,
    y,
    yerr=yerr,
    fmt='o',
    ecolor='red',
    capsize=5,
    linestyle='none',
    markerfacecolor='blue',
    label="Datos de voltaje con error"
)

plt.plot(
    x,
    m * x + c,
    color='b',
    label=f'Ajuste lineal: y = {m:.2f}x + {c:.2f}'
)

plt.xlabel("Frecuencia (Hz)")
plt.ylabel("Voltaje (mV)")
plt.title("Ajuste de Regresion Lineal No Ponderada")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.savefig("punto_6_1_b.png")
```

Y esto nos da como respuesta


Voltage vs Frequency with Error Bars
Voltage measurements
Voltage (mV)
Frequency (Hz)
200
150
100
50
20
40
60
80
100


Ajuste de Regresion Lineal No Ponderada
Ajuste lineal: y = 1.91x + -0.07
Datos de voltaje con error
Voltaje (mV)
Frecuencia (Hz)
200
150
100
50
20
40
60
80
100