計算機

24cb062h 菅原明

1. 目的

測定をした値には必ず誤差が生じる. 今回は電気回路等で発生する雑音を含んだ電圧データをダウンロード[1] し,それを計算機を用いて解析をし,偶然誤差の性質や誤差伝播法則など誤差を含んだデータの扱い方を習得する.[2] データ解析のツールとして今回は Python を用いる.

2. 原理

2.1. 母集団の平均と標準偏差

母集団の平均値Mに対し,測定したi番目のデータをd_i,データの個数をNとすると,標準偏差σは母集団の平均値Mを用いて,

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (d_i - M)^2}{N}} \tag{1}$$

で与えられる.しかし,事前に測定値の平均値がわかっておらず,測定データの平均mでMを代用するとき,平均値と芳醇偏差は

$$m = \frac{\sum_{i=1}^{N} d_i}{N} \tag{2}$$

$$\sigma_s = \sqrt{\frac{\sum_{i=1}^{N} (d_i - m)^2}{N-1}} \tag{3}$$

で求めることができる.この標準偏差を不変標準偏差といい,分母が$N-1$となっているのは平均値mをデータd_iからきめており,$(d_i - m)$の自由度は 1 減っているからである.
Eq. 1,Eq. 3 の値はデータの数が少なくなると,値が大きくことなくことに注意する.

3. 実験方法

今回,自らでデータの測定は行わず配布されたデータ[1] を使用して計算機を用いて解析する.
Python を使用する場合は,Google Colaboratory(以下 Colab)あるいは,ローカルで仮想環境を作り(参照:https://qiita.com/fiftystorm36/items/b2fd47cf32c7694adc2e), Jupyter あるいは Python3 をもちいて解析をする.以前 python を触ってみた際に Jupyter をしようしたので,今回もこれを使用していく.

3.1. 仮想環境の利用[3]

OS: 24.04.2 LTS
Python 3.9.18

今回仮想環境をつくるにあたり,venv というツールをもちいる.プロジェクトディレクトリに移動いて,noise という今回使用する新しい環境を作る.

```
$ cd [project dir]
$ python3 -m noise
```

また,

```
$ sourse noise/bin/activate
```

とすることで,仮想環境に入った状態になる. バージョンの確認は

```
(noise) $ python -V
```

である.パッケージのインストールは,

```
(noise) $ pip install パッケージ名
```

でインストールすることができる.
仮想環境を終わらせるには

```
(nose) $ deactivate
```

とすれば良いです.

3.2. Python を利用したデータ解析

3.2.1. 課題 A-1,2

```python
# 必要なライブラリのインポート
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# 読み込むCSVファイル名を指定
filename = "noise.csv"

# CSVファイルを読み込む（末尾11行をスキップし、1列目と2列目のみを使用）
# カラム名は 'times', 'voltage' に設定
df = pd.read_csv(filename, skipfooter=11, usecols=[0,1],
names=['times','voltage'], engine='python')

# 電圧データをNumPy配列として取得
data = df['voltage'].values

# 基本的な統計量の計算
data_num = len(data)                    # データの個数
data_min = min(data)                    # 最小値
data_max = max(data)                    # 最大値
data_ave = np.average(data)             # 平均値
data_sig = np.std(data)                 # 標準偏差（母集団）
data_usig = np.std(data, ddof=1)        # 不偏標準偏差（標本用、ddof=1）

# 時系列データのx軸（1から始める連番）
x_data = np.arange(1, data_num + 1, 1)

# 電圧データのプロット（黒線でプロ
```

3.2.1.1. 説明

このコードは csv ファイルから電圧データを読み込み,実験データの基本的な量(平均値,標準偏差,不偏標準偏差)を計算し,時間経過に沿った電圧の変化をグラフに表示させるコードです.

- pandas,matplotlib,numpy というモジュールをインポートし,pandas,matplotlib,numpy をそれぞれ pd,plt,np と略す.

- pandas を用いて,csv を読み込み,不要な行をなくす.
- Numpy を用いて,実験データの基本的な量を計算.
- matplotlib で時間軸に対する電圧変化をプロットし,画像として保存

3.2.2. 課題 A-3

```python
# 必要なライブラリのインポート
import pandas as pd                 # データ操作用（pandasライブラリ）
import matplotlib.pyplot as plt     # グラフ描画用（matplotlibライブラリ）

# 読み込むCSVファイル名を指定
filename = "noise.csv"

# CSVファイルを読み込み
#  ・末尾11行は不要な情報としてスキップ（skipfooter=11）
#  ・0列目と1列目のみ使用（usecols=[0,1]）
#  ・列名を'times'と'voltage'に設定
df = pd.read_csv(filename, skipfooter=11, usecols=[0, 1], names=['times',
'voltage'], engine='python')

# 電圧データのみをNumPy配列として取り出す
data = df['voltage'].values

# ヒストグラムの作成（10個のビンに分けて頻度をカウント）
n, bins, _ = plt.hist(data, bins=10)

# 軸ラベルの設定
plt.xlabel('Voltage[V]', fontsize=14)   # x軸は「電圧（ボルト）」
plt.ylabel('Counts', fontsize=14)      # y軸は「出現回数（度数）」

# グラフを画像ファイルとして保存（事前に"figure"フォルダを作成しておくこと）
plt.savefig("figure/fig2.png")
```

このコードでは電圧データの分布を確認するためにヒストグラムを作成した.電圧の値の分布を 10 個のピンに分類した.
- plt.hist を使って,ヒストグラムを描写する.

3.2.3. 課題 A-4

```python
# 必要なライブラリの読み込み
import pandas as pd                         # 表形式のデータ処理用
import matplotlib.pyplot as plt             # グラフ描画用
from scipy.stats import norm                # 正規分布（ガウス関数）を扱うためのモジュール
import numpy as np                          # 数値計算用

# 読み込むCSVファイル名
filename = "noise.csv"

# CSVファイルからデータを読み込み
#  ・末尾11行を除外（skipfooter=11）
#  ・最初の2列（0列と1列）を使用
#  ・列名を'times'と'voltage'に設定
df = pd.read_csv(filename, skipfooter=11, usecols=[0,1],
names=['times','voltage'], engine='python')

# 電圧データをNumPy配列として抽出
data = df['voltage'].values

# データの基本統計量を算出
data_num = len(data)                    # データ数
data_min = min(data)                    # 最小値
data_max = max(data)                    # 最大値
```

```
data_ave = np.average(data)            # 平均値
data_sig = np.std(data)                # 標準偏差（母標準偏差）
data_usig = np.std(data, ddof=1)       # 不偏標準偏差（サンプルからの推定）

# ヒストグラムの描画（相対度数で表示、線のみのスタイル）
# density=True により縦軸は確率密度（合計1）に正規化される
n, bins, _ = plt.hist(data, bins=10, histtype='step', density=True,
                      label='measured value', color='k')

# ガウス関数（正規分布）を描画
x_g = np.linspace(data_min, data_max, 100)    # 最小値〜最大値まで100点のx軸データを生成
y_g = norm.pdf(x_g, data_ave, data_usig)      # 平均値と不偏標準偏差に基づく正規分布の確率密度を計算
plt.plot(x_g, y_g, color='r')                 # 赤色の線でプロット

# グリッド線の設定
plt.grid(ls='--')                             # 破線のグリッドを表示

# 軸ラベルの設定
plt.xlabel('Voltage/V', fontsize=14)
plt.ylabel('Relative frequency', fontsize=14)

# 画像ファイルとして保存（事前にfigureフォルダが必要）
plt.savefig('figure/fig4.png')
```

このコードは,測定された電圧データの分布をヒストグラムとして可視化し,得られたデータにたいし,正規分布をフィッティングしている.

3.2.4. 課題 A-5

```
# 必要なライブラリをインポート
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# 入力ファイル名
filename = "noise.csv"

# データの読み込み
# ・末尾11行は無視
# ・0列と1列のみを読み込む（時刻と電圧）
# ・列名を'times'と'voltage'に指定
df = pd.read_csv(filename, skipfooter=11, usecols=[0,1],
names=['times','voltage'], engine='python')

# 電圧データのみ取り出す
data = df['voltage'].values

# 基本統計量を計算
data_num = len(data)                    # データ数
data_min = min(data)                    # 最小値
data_max = max(data)                    # 最大値
data_ave = np.average(data)             # 平均値
data_sig = np.std(data)                 # 標準偏差（母標準偏差）
data_usig = np.std(data, ddof=1)        # 不偏標準偏差（標本の標準偏差）

# --- データを分割し、それぞれの標本平均の標準偏差を求める関数 --- #
def splitting_data(data, N):
    data = data
    N = int(N)

    data_num = len(data)                    # 全体のデータ数を取得
```

```
    remainder = data_num % N              # Nで割った余りを計算
    cut_num   = int(data_num - remainder) # Nで割り切れる長さに調整
    data_cut  = data[:cut_num]            # 調整後のデータを抽出

    sp_num    = int(cut_num / N)          # 分割後のブロック数
    data_sp   = data_cut.reshape(sp_num, N)  # N個ごとのブロックに分割

    sp_ave_list = np.average(data_sp, axis=-1)                # 各ブロックの平均を求める
    sp_sig_list = np.std(data_sp, ddof=1, axis=-1) / np.sqrt(N) # 各ブロックの標本平均の標準偏差

    sp_sig = np.std(sp_ave_list, ddof=1)  # 全ブロックの平均値に対する標準偏差（評価用）

    sp_x = np.arange(0, cut_num, N) + (N / 2.)  # 各ブロックの中心点をx軸として返す

    return sp_x, sp_ave_list, sp_sig_list, sp_sig

# N（1ブロックあたりのデータ数）ごとに標準偏差を計算
sig_list = []                                  # 結果を格納するリスト
n_list = [5, 10, 20, 50, 100, 200]             # 検討する分割数のリスト

for n in n_list:
    sp_x, sp_ave_list, sp_sig_list, sp_sig = splitting_data(data, n)
    sig_list.append(sp_sig)                    # 分割平均の標準偏差をリストに追加

# 結果のプロット
plt.plot(n_list, sig_list, "ko-")              # 黒丸+線グラフ
plt.xlabel('Number of segments', fontsize=14)
plt.ylabel('Unbiased standard deviation', fontsize=14)
plt.grid(ls='--')
plt.savefig('figure/fig5.png')                 # グラフを画像として保存
```

このコードは電圧データを一定の点数ごと(5,10,20,50,100,200)に分割し,それぞれのブロックごとに平均値を求める.ここで,平均値の標準偏差を求め,分割数を変化させたときの不偏標準偏差の変化を調べる.

3.2.5. 課題 B-1

```
# 必要なライブラリの読み込み
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# データファイル名の指定
filename = "noise.csv"

# CSVファイルからデータを読み込む（末尾11行をスキップ、0列目と1列目のみ使用）
# カラム名を 'time', 'voltage' に指定
df = pd.read_csv(filename, skipfooter=11, usecols=[0,1],
names=['time','voltage'], engine='python')

# 電圧データを NumPy 配列として取り出す
data = df['voltage'].values

# 基本統計量の計算
data_num  = len(data)             # データの個数
data_min  = min(data)             # 最小値
data_max  = max(data)             # 最大値
data_ave  = np.average(data)      # 平均値
data_sig  = np.std(data)          # 標準偏差
```

```python
data_usig = np.std(data, ddof=1) # 不偏標準偏差（ddof=1 を指定）

#------------------------------------------------------------------------------
# 関数: データをN個ずつに分割し、各分割の平均値と誤差を求める
def splitting_data(data, N):
    N = int(N)                          # 分割数を整数化
    data_num = len(data)                # データの総数
    remainder = data_num % N            # 余りを計算
    cut_num   = data_num - remainder    # 端数を切り捨てて使うデータ数を決定
    data_cut  = data[:cut_num]          # 端数を取り除いたデータ
    sp_num    = cut_num // N            # 分割できる回数
    data_sp   = data_cut.reshape(sp_num, N) # N個ずつ行にして2次元配列化

    # 各分割の平均値と標本平均の標準誤差を計算
    sp_ave_list = np.average(data_sp, axis=-1)                      # 各グループの平均値
    sp_sig_list = np.std(data_sp, ddof=1, axis=-1) / np.sqrt(N)    # 各グループの誤差

    # 全体の標準偏差（各平均値のばらつき）を計算
    sp_sig = np.std(sp_ave_list, ddof=1)

    # グラフのx軸用：各平均値の中心位置
    sp_x = np.arange(0, cut_num, N) + (N / 2.)

    return sp_x, sp_ave_list, sp_sig_list, sp_sig

# 分割数を指定し、データの分割・平均・誤差を取得
N = 100
sp_x, sp_ave_list, sp_sig_list, sp_sig = splitting_data(data, N)

# 分割平均とその誤差をエラーバー付きでプロット
plt.errorbar(sp_x, sp_ave_list, yerr=sp_sig_list, fmt='ko') # 'ko' = 黒丸
plt.xlabel('Time series data', fontsize=14)
plt.ylabel('Voltage / V', fontsize=14)
plt.grid(ls='--')
plt.savefig('figure/fig6.png')  # 図の保存

#------------------------------------------------------------------------------
# 関数: 最小二乗法で直線フィッティングを行い、パラメータと誤差を求める
def cal_leastsq(xdata, ydata, sigma):
    """
    最小二乗法による直線フィッティングとパラメータ誤差の計算

    Parameters:
        xdata : x軸データ（NumPy配列）
        ydata : y軸データ（平均値）
        sigma : yの誤差（標準誤差）

    Returns:
        a : 傾き
        b : 切片
        a_err : 傾きの誤差
        b_err : 切片の誤差
        chisq_min : 最小カイ二乗値
    """
    x = xdata
    y = ydata
    s = sigma

    # フィッティング係数を計算するためのΔ（分母）
    delta = (np.sum(x**2 / s**2) * np.sum(1 / s**2)) - (np.sum(x / s**2))**2
```

```
    # 傾きと切片、およびその誤差を計算
    a = (np.sum(x * y / s**2) * np.sum(1 / s**2) - np.sum(x / s**2) * np.sum(y / s**2)) / delta
    a_err = np.sqrt(np.sum(1 / s**2) / delta)
    b = (np.sum(x**2 / s**2) * np.sum(y / s**2) - np.sum(x * y / s**2) * np.sum(x / s**2)) / delta
    b_err = np.sqrt(np.sum(x**2 / s**2) / delta)

    # フィットの理論値と残差による最小カイ2乗の計算
    fit_values = a * x + b
    residuals = (y - fit_values) / s
    chisq_min = np.sum(residuals**2)

    return a, b, a_err, b_err, chisq_min

# 最小二乗法によるフィッティングを実行
xdata = sp_x
ydata = sp_ave_list
sigma = sp_sig_list

a, b, a_err, b_err, chisq_min = cal_leastsq(xdata, ydata, sigma)

# 結果を出力
print(f'a = {a:.5f} ± {a_err:.5f}')
print(f'b = {b:.5f} ± {b_err:.5f}')
print(f'chi^2 = {chisq_min:.3f}')
print(f'reduced chi^2 = {chisq_min/(len(ydata)-2):.3f}')

# フィット結果の描画
y_fit_data = a * xdata + b
plt.errorbar(xdata, ydata, yerr=sigma, fmt='ko', label='data') # 測定値
plt.plot(xdata, y_fit_data, 'r-', label='fit results')         # フィット線
plt.xlabel(r'$\rm xdata$', fontsize=14)
plt.ylabel(r'$\rm ydata$', fontsize=14)
plt.legend(fontsize=14)
plt.grid(ls='--')

# 最終的な図の保存
plt.savefig('figure/fig.png')
```

mの値に時間的変化があるかどうかを調べるために,データを$N = 100$ずつに分割し,それぞれの標本平均と標準偏差を求める. 次に平均値をエラーバー付きでプロットし,最小二乗法を用いてフィッティングをする.フィッティングの式は

$$y = a \cdot x + b \tag{4}$$

でフィッティングをする.

4. 結果

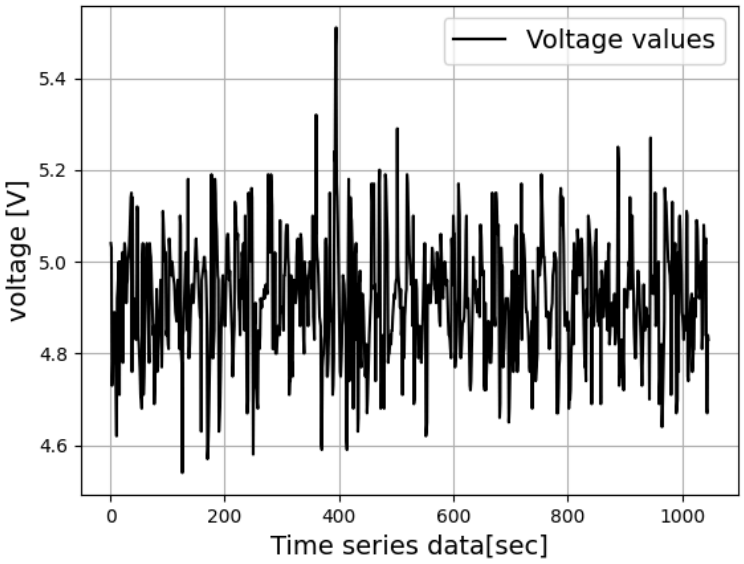

Fig. 1: 課題 A-1:電圧と時間の関係

課題 2 は下のようになった

```
Number of data = 1046
Maximum value = 5.51 [V]
Minimum value = 4.54 [V]
Average value of data = 4.92 [V]
Standard Deviation = 0.13
Unbiased Standard Deviation = 0.13
```

Listing 1: 課題 A-2:基本的な数値

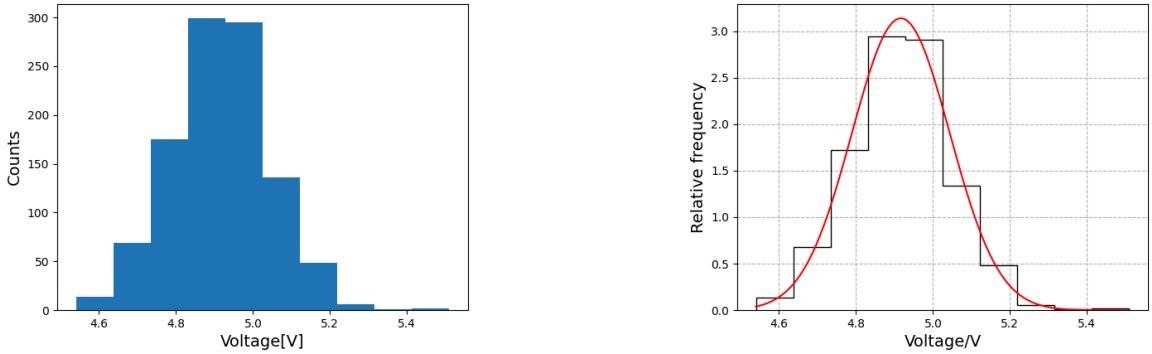

Fig. 2: 課題 A-3,A-4:電圧がどのように分散しているのかのヒストグラム(左)と,ヒストグラムをガウス分布とを比べたグラフ(右)

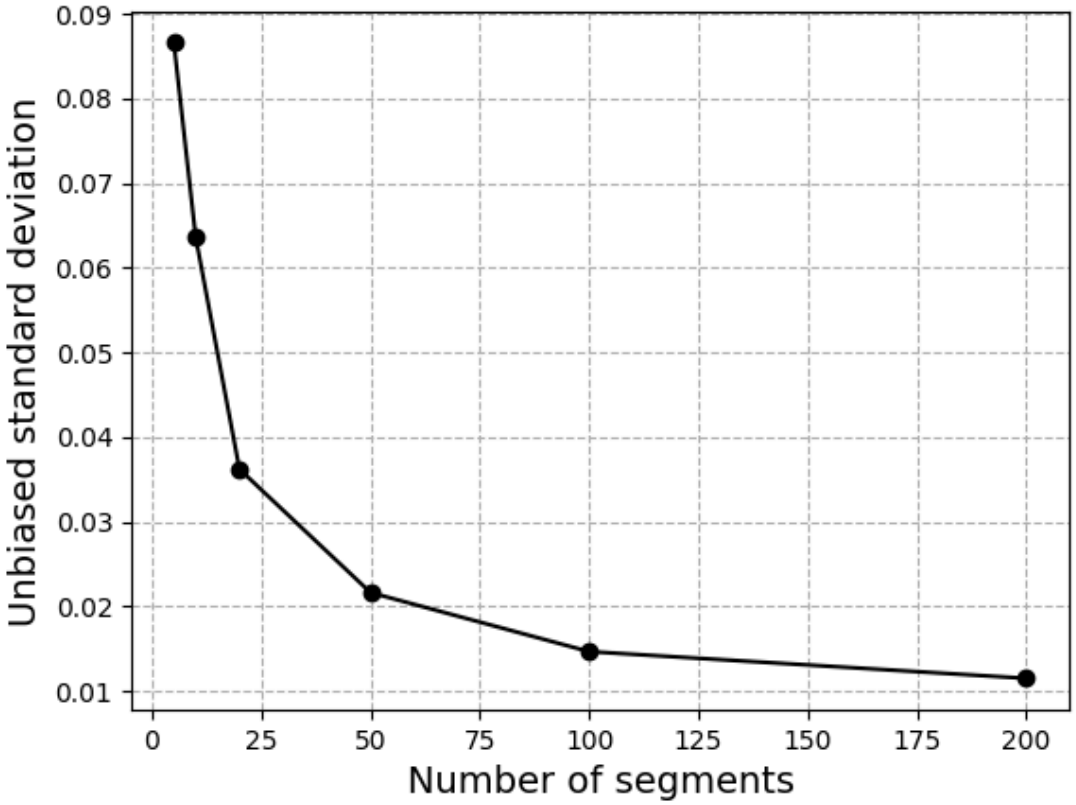

Fig. 3: 課題 A-5:平均値mが使用するデータの個数Nによってどのようにばらつくか

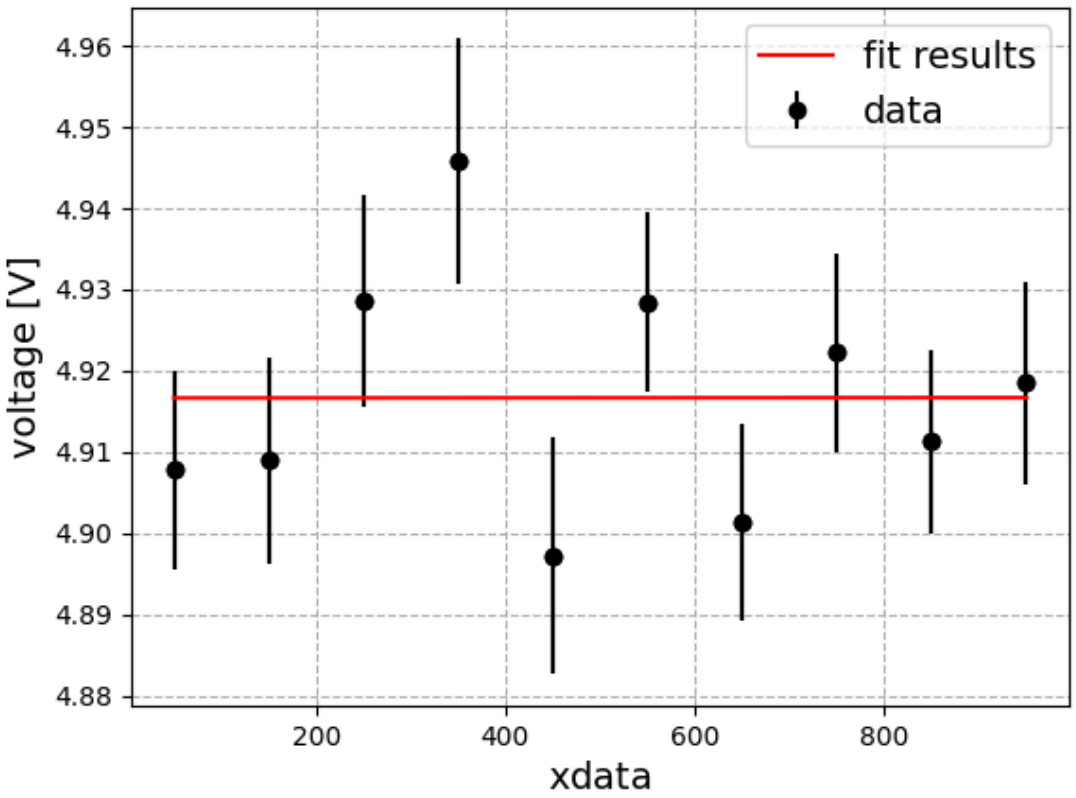

Fig. 4: 課題 B-1:mの値に時間的な変化はあるのかχ^2法を用いて,最小二乗法をし,フィッティング

5. 考察

5.1. 課題 A-1,2

Fig. 1 から電圧の値は時間に対して,大きな誤差はなく比較的平均値に収まっている.Fig. 1 ,Listing 1 より,電圧の最大値は 5.51V,最小値は 4.54V であり,およそ 1V の範囲にデータが分布していることがわかる.標準偏差は 0.127V,不偏標準偏差は 0.1271V とどちらの値もほぼ同じ値を示したことから,データ数が十分にとれており,信ぴょう性がある.

5.2. 課題 A-3,4

Fig. 2 より,不偏標準偏差を用いて計算されたガウス分布は,測定値の分布をほぼ一致しており,データが正規分布に従っている可能性があることがわかる.

5.3. 課題 A-5

Fig. 3 より,分割数を増やすにつれて,不偏標準偏差が減少することから,サイズが大きくなるにつれて,平均値からのずれがすくなくなり,その平均値の信憑性が増していることがわかる.

5.4. 課題 B-1

Fig. 4 より,フィッティングにより,得た直線はx軸にほとんど平行なものである.実際,aの値は$a = 4.21 \cdot 10^{-8} \pm 1.36 \cdot 10^{-5}$がえられ,値が十分小さいため,$a \sim 0$とすることができる.このことから,$m$の値は時間的変化がないと言えるだろう.また,$b$の値は$b = 4.92 \pm 0.0081$であるため,測定された電圧の値と一致した.

```
a=4.212198044156624e-08 +- 1.35939801987801e-05
b=4.916647319323991 +- 0.008089674589483056
chi2_min = 10.393397667763399
reduced chi-squared = 1.2991747084704248
```

また,χ^2の値はおよそ1.3であるため,フィッティングした直線はデータにたいして十分良好なものであると言える.

参考文献

[1] “noise.csv のデータ.” [Online]. Available: http://s.rikkyo.ac.jp/noisedata

[2] 基礎物理実験 立教大学理学部物理学科 2025 年版. 2025.

[3] @. fiftystorm36(Kenya Igarashi), “venv: Python 仮想環境管理.” [Online]. Available: https://qiita.com/fiftystorm36/items/b2fd47cf32c7694adc2e