第 0 版 2026/01/22
山村 一繁

グラフを描いてみよう

この資料では、ライブラリの matplotlib を使ってグラフを描く方法を勉強します。併せてライブラリ pandas を使ったデータ処理の方法についても説明します。

【重要】この pdf ファイルのプログラムをコピペしても、プログラムの実行はできません。

★ データを準備する

まずプロットするデータを準備しましょう。データの個数が少なければリストを使うことができます。

ソースコード 1: データをリストに保存する

```
1  x = [ 1, 2, 3, 4, 5 ]
2  y = [ 2, 4, 9, 16, 25 ]
```

しかし、今回の振動実験の自由振動データは 60,000 個あるので、プログラムの中に書き込むことは現実的に不可能です。データはファイルに保存されていて、内容は次のように、第 1 列に時間（単位：秒）、第 2 列に変位（単位：cm）が書いてあります。

```
0.000000, 0.003662
0.001000, 0.005798
0.002000, 0.005798
0.003000, 0.004272
0.004000, 0.003967
0.005000, 0.003357
   ⋮         ⋮
```

ここで、データファイルの名前を、vibration.csv とします。

python のファイル読込機能を使うと、例えば次の様なプログラムでデータファイルを読み込むことができます。

ソースコード 2: データファイルを読み込む

```
1 coordinate_string = []
2 with open('vibration.csv') as f:
3     for line in f:
4         coordinate_string.append(line.rstrip().split(','))
5
6 coordinate = [[float(item) for item in sublist] for sublist in coordinate_string]
```

まず空のリスト coordinate_string を作ります。ファイルを開き、一行づつ読み込みます。改行文字を削除し、区切り文字を「,」としてリストを作ります。それをリスト coordinate_string に追加します。python でファイルを読み込むときは基本的に文字列として扱われるので、最後に実数に変換します。変数 coordinate は、時間と変位からなるリストのリストです。

このように「素の python」では少々面倒な手続が必要なので、ライブラリ pandas を使ってみましょう。1 行目でライブラリ pandas をインポートし、略称を pd としています。

ソースコード 3: pandas を使ってデータファイルを読み込む

```
1 import pandas as pd
2
3 coordinate = pd.read_csv('vibration.csv', names=('時間 [秒]', '変位 [cm]'))
```

後で使いやすように、各列に名前をつけてあります。pandas を使うと各行を実数として読み込んでくれます。ずいぶん簡単になりましたね。

ここで pandas のデータについて簡単に説明しておきます。python では、ひとつの値を持つスカラ変数、複数の値をもつリストなどを使うことができます。他にタプルや辞書なども利用可能です。pandas をインポートすると、それらに加えてシリーズとデータフレームが使えるようになります。

データフレームは、数学で勉強した「行列」のようなものです。平面的にデータが並んでいます。大切なことは、各行と各列に「名前」がついていることです。ソースコード 3 の 3 行目の変数 coordinate がデータフレームです。列の名前はプログラムで設定した「時間 [秒]」と「変位 [cm]」、行の名前は通し番号（0 から始まることに注意）になっています。

	時間 [秒]	変位 [cm]
0	0.000	0.003662
1	0.001	0.005798
2	0.002	0.005798
3	0.003	0.004272
4	0.004	0.003967
5	0.005	0.003357
	⋮	⋮

★ グラフを描く

グラフを描くにはライブラリ matplotlib を使います。ソースコード 3 の変数 coordinate をプロットしてみましょう。データの読込も再掲します。

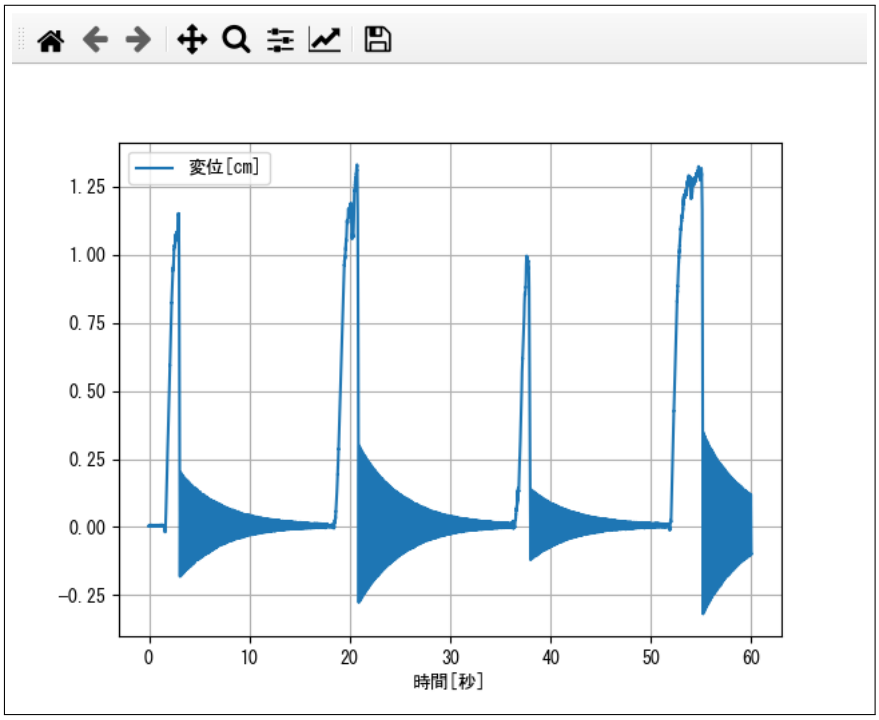
ソースコード 4: グラフを描く

```
1  # ライブラリのインポート
2  import pandas as pd
3  import matplotlib.pyplot as plt
4  import matplotlib
5
6  # フォントの設定
7  matplotlib.rcParams['font.family'] = 'MS Gothic'
8  matplotlib.rcParams['font.size'] = 10
9
10 # データの読み込み。列に名前を附ける。
11 coordinate = pd.read_csv('vibration.csv', names=('時間 [秒]', '変位 [cm]'))
12
13 # x 座標、y 座標の設定
14 coordinate.plot(x='時間 [秒]', y='変位 [cm]')
15
16 # グラフの表示
17 plt.show()
```

注）Spyder でグラフを独立したウィンドウで表示するためには以下の設定をすること。念のため、設定後に Spyder を再起動するとよい。

「ツール (I)」→「設定」→「IPython コンソール」→「プロット中」タブ→「グラフィックスのバックエンド」→「Qt」を選択→「OK」

プログラムを実行すると次のように表示されます。



◇ アイコンの説明


初期状態に戻す。


描画部分を動かす。

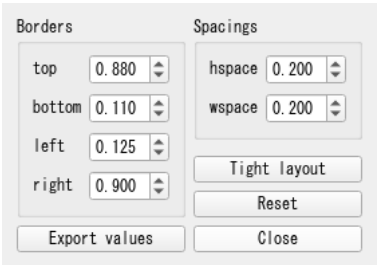

左ボタンでドラッグ＆ドロップした範囲を拡大する。


余白を設定する。


軸、線種を設定する。


グラフを保存する。さまざまなフォーマットが選択できるが、通常は png でよい。

◇ 余白の設定

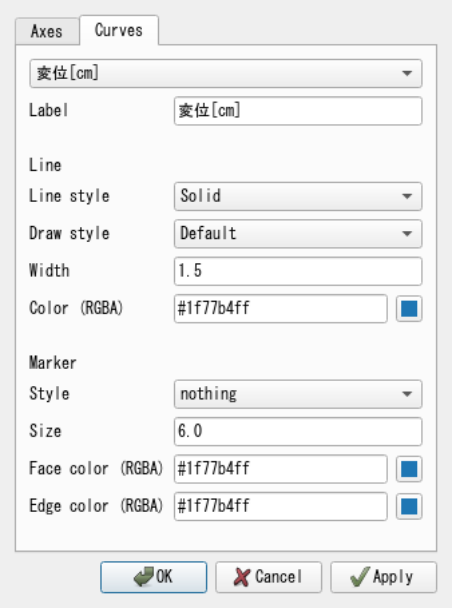


- Borders の top, bottom, left, right で余白を設定する。
- Spacings は複数のグラフを貼りつけたときの設定なので今回は使わない。
- Tight layout でウィンドウ一杯に表示される。
- Reset で初期状態に戻る。

◇ 軸の設定

- Title で図にタイトルをつける。
- X-Axis で、x 軸の範囲やラベルを設定する。
- Y-Axis で、y 軸の範囲やラベルを設定する。
- (Re-)Generate automatic legend で、凡例を調整する。

◇ 線種の設定



- Label で凡例の文字を設定する。
- Line で線の種類、太さ、色などを設定する。Draw style で Steps を選択すると階段状に描画する。
- Marker でマーカーの種類、大きさ、色を設定する。

★ おまけ

以下、説明なしにプログラムを掲載しておく。

ソースコード 5: 移動平均の計算

```
#!/usr/bin/env python3

'''
2024/01/16

自由振動波形の移動平均を計算しプロットする。
計算結果は Excel ファイルに保存する。

'''

# ============================================================================
# 変数

# データのファイル名
fdsc = '.data/vibration.csv'

# ============================================================================
# ライブラリ
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib

matplotlib.rcParams['font.family'] = 'MS Gothic'
matplotlib.rcParams['font.size'] = 10

# ============================================================================
# メインプログラム

# ----------------------------------------------------------------------------
# データの読み込みと変換

# <class 'pandas.core.frame.DataFrame'> で読み込む。
# 時間は '時間 [秒]',変位は '変位 [cm]' と名前をつける。
data = pd.read_csv(fdsc, names=('時間 [秒]', '変位 [cm]'))

# 移動平均を計算し、data に追加する。
window_size = 15
data['変位（移動平均）[cm]'] = data['変位 [cm]'].rolling(window=window_size).mean()

# Excel ファイルに保存する
data.to_excel('MA.xlsx')

# ----------------------------------------------------------------------------
# matplotlib によるプロット

# プロットの準備
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

# 座標と線種などの設定
# 移動平均した波形
data.plot(ax=ax, x='時間 [秒]', y='変位（移動平均）
    [cm]', color='red', linewidth=1, legend=False)
```

```
# オリジナルの波形
data.plot(ax=ax, x='時間 [秒]', y='変位 [cm]', color='blue', linewidth=0.5, legend=False)

# グリッドを描く
ax.grid()

# グラフの表示
plt.show()
```

ソースコード 6: RC の加力試験のデータのプロット

```
#!/usr/bin/env python3

'''
RC の実験の加力試験のデータをプロットする
2024/01/16

subprocess の import を削除してしまったので再追加
2024/02/16
'''

# ----------------------------------------------------------------------
# ライブラリのインポート
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
import subprocess

# インタラクティブモードを off にしておく
plt.ioff()

# ----------------------------------------------------------------------
# フォントの設定
matplotlib.rcParams['font.family'] = 'MS Gothic'
matplotlib.rcParams['font.size'] = 10

# ----------------------------------------------------------------------
# 読み込むファイル

fdsc = '.data/RC 試験体加力実験_2023.xlsx'

# プロットするデータ
X = 2
Y = 1

'''
0 : '計測ｽﾃｯﾌﾟ'
1 : '荷重'
2 : '変位計 1(CDP50)'
3 : '変位計 2(CDP50)'
4 : 'あばら筋（1)'
5 : 'あばら筋（2)'
6 : '主筋（真ん中）'
7 : '主筋（真ん中 2)'
8 : '主筋〈端)'
'''

```

```
47 ##########################################################################
48 def main():
49
50     # データの読込
51     data = pd.read_excel(fdsc, skiprows=[0,2], header=0)
52     # 単位の読込
53     unit = pd.read_excel(fdsc, header=1, nrows=1)
54
55     # 列の名称と単位のリスト
56     Columns = list(data.columns)
57     Unit = list(unit.iloc[0])
58
59     # プロットの準備
60     fig = plt.figure()
61     ax = fig.add_subplot(1, 1, 1)
62
63     # x, y 座標の設定
64     data.plot(ax=ax, x=Columns[X], y=Columns[Y],
65               linewidth=1.0, legend=False)
66
67     # グリッド、タイトル、ラベルの設定
68     ax.grid()
69     ax.set_title(Columns[X] + ' - ' + Columns[Y])
70     ax.set_xlabel('[ ' + Unit[X] + ' ]')
71     ax.set_ylabel('[ ' + Unit[Y] + ' ]')
72
73     # 両軸の範囲の設定
74     ax.set_xlim([0, 40])
75     ax.set_ylim([0, 250])
76
77     # pdf, png, svg ファイルに出力する
78     fig.savefig('RC_graph.pdf')
79     fig.savefig('RC_graph.png')
80     fig.savefig('RC_graph.svg')
81     # svg ファイルを emf ファイルに変換する
82     # windows の場合
83     # subprocess.call('"c:\\Program Files\\Inkscape\\bin\\inkscape.exe" RC_graph.svg -o
           RC_graph.emf', shell=True)
84     # mac/Linux の場合
85     # subprocess.call('/usr/bin/inkscape RC_graph.svg -o RC_graph.emf', shell=True)
86
87     # 画面に描く
88     plt.show()
89
90
91 ##########################################################################
92
93 if __name__ == '__main__':
94
95     main()
```