

# 波形数値化の機械学習用データ作成プログラム (Data\_Digitizer.py) 使用説明書

波形数値化の機械学習には、波形の数により下記の3つのプログラムが存在する。本プログラムはこれらの学習用データを作成するもので、波形本数を入力することで、いずれのプログラムにも対応した学習データを生成できる

- 1波形: Learning\_Digitizer\_1-line.py,
- 2波形: Learning\_Digitizer\_2-lines.py,
- 3波形: Learning\_Digitizer\_3-lines.py

## 目次

1. 機能
2. 動作環境
3. 内容
  - 3.1 構造
  - 3.2 プログラムリストの概要
  - 3.3 入出力
    - 3.3.1 入力及びプログラムの実行
    - 3.3.2 出力
    - 3.3.3 データの出力形式
4. プログラムリスト

## 1. 機能

このプログラム(Data\_Digitizer.py)は、機械学習用の学習データを作成するものであり、作成されたデータは、グラフの波形を数値化する機械学習(Learning\_Digitizer\_1-line.py, Learning\_Digitizer\_2-lines.py, Learning\_Digitizer\_3-lines.py) に用いられる

## 2. 動作環境

1) このプログラムはWindows10, Ubuntu18-04での動作が確認されている

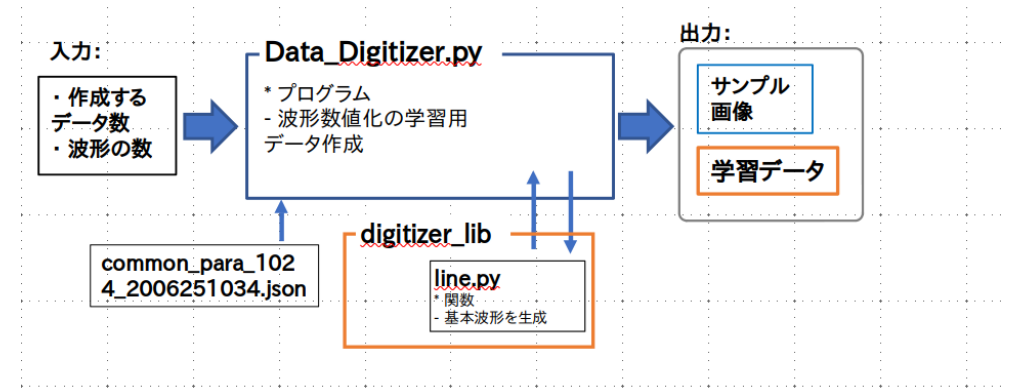
2) このプログラムを実行するには、以下のモジュールがあらかじめ準備された環境が必要である。また、記載されている以外のVersionでの動作確認は行っていないので注意が必要である

- Python 3.6
- Tensorflow-gpu 1.4.0
- Keras 2.2.4
- sklearn 0.19.1, 0.21.2
- matplotlib 2.2.2

## 3. 構造

## 3.1 概要

このプログラムは、入力として、作成するデータ数と波形の数を使用者からのインプットとして受け取り、出力として学習用データとデータのサンプル画像を自動的に出力する  
このプログラムは、画像データを生成するため、画像パラメータファイル(common\_para\_1024\_2006251034.json) から画像パラメータを読み込む  
このプログラムは基本波形を生成するため、関数ライブラリdigitizer\_lib に収納されている波形生成関数 line.py を呼び出して使用する



## 3.2 プログラムリストの概要

このプログラムはデータ作成本体（Data\_Digitizer.py）と関数ライブラリ（digitizer\_lib）に収められている波形生成関数（Line.py）の2つのモジュールに分けられる  
全リストは、4. プログラムリストに示す

### 1. データ作成本体（Data\_Digitizer.py）

1-8 行: 開始コメント  
10-20 行: モジュールのインポート  
28-33 行: 開始準備  
35-98 行: データパラメータの設定  
100-117 行: パラメータの確認  
119-302 行: データの生成  
304-349 行: チェック用の画像出力  
351-372 行: モデルの保存  
374 行: END

### 2. 波形生成関数（Line.py）

1-7 行: 開始コメント  
9-9 行: モジュールのインポート  
15-38 行: 開始準備  
40-69 行: 波形の生成  
71 行: return

## 3.3 入出力

以下、Jupyter Notebook上で実行された例を示して説明する

### 3.3.1 入力およびプログラムの実行

このプログラムの入力、作成するデータ数と波形の数の2項目である

注)波形の数は、このプログラム自体に制限はないが、対応する学習プログラムは1,2,3本の3種類みが作成されている

プログラムを実行すると、下の図の様にコンソール上で各々、データ数と波形の数の入力待ち==>になる。使用者は必要なデータ数と波形の数を入力して実行する  
下の例ではデータ数は500個、波形の数は2本が入力されている

```
< Data Creating of Machine Learning for Waveform Digitizer 200627-1442 >
```

```
Input data number. if <= 99, datas are not saved  
=> 500
```

```
Input Line number   = 2
```

次に、下の様な画像パラメータが表示され、確認の入力待ち==>になる

このパラメータを確認してOKであれば、'y'もしくは'n'以外を入力して実行するとデータ作成が始まる

```
< Data Creating of Machine Learning for Waveform Digitizer 200627-1442 >
```

```
Input data number. if <= 99, datas are not saved  
=> 500  
Input Line number   = 2
```

```
< Data parameter >
```

```
common parameter = common_para_1024_2006251034.json
```

```
Data number = 500  
line number = 2  
line peak   = [2, 3, 4]  
dim         = 1024, dpi = 360 , fig size = 2.84 x 2.84 inch  
line width  = 2.0 p  
font size   = 8.0 p  
marker      = ['+', 'D', '^', 'o', 's', 'v', 'x']
```

```
comment : Digitizer Data 2-lin
```

```
Confirm data parameter, process start OK? y or n -->
```

```
y
```

\*注) データ数が100個以下が入力された場合は、味見としてデータの保存は実行されず、サンプル画像の表示のみが行われる

### 3.3.2 出力

データ数が100個以上の場合、データは自動的に保存される

データ数が100個以下の場合、味見としてデータの保存は実行されず、サンプル画像の表示のみが行われる

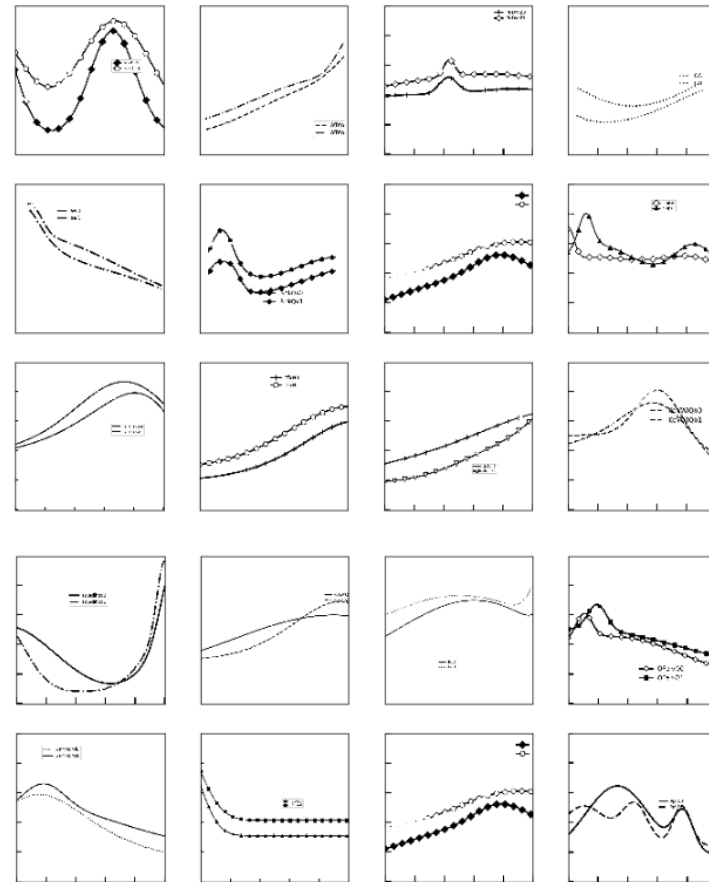
以下に2波形、500個のデータを作成した実際の出力例をしめす

1. 実行時間に続いて画像データの中からランダムに選ばれた20個のサンプル表示される

```
Process start...  
...Finished. Execution time (s) = 30.36
```

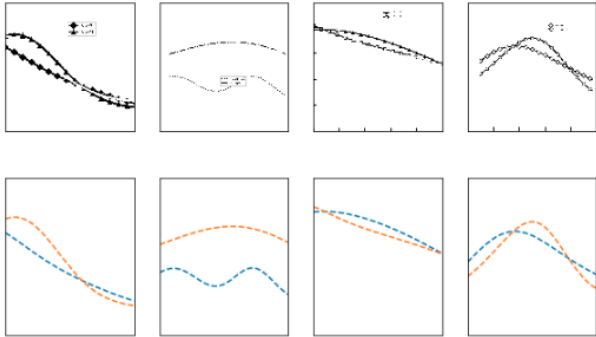
```
< Figure check >
```

```
sample number = [158 75 466 3 159 96 404 293 484 56 1 401 238 13 89 173 124 112 404 121]
```



2. 次に、ランダムに選ばれた4個の画像データとこれに対応する教師データから作られた比較用の画像が表示される

```
< Figure and Data comparison >
. sample number = [172 27 211 294]
. upper = input, lower = output
```



3. 最後に、最下段に自動的に保存されたデータ名が示されプログラムは終了する（この例では ./Digitizer\_Data\_2-lin\_1024\_500\_200627-1442.pik にデータが保存された）

```
Data was saved >>>
./Digitizer_Data_2-lin_1024_500_200627-1442.pik
```

### 3.3.3 データの出力形式

以下に出力されるデータの形式について説明する

参照:4.1 データ作成本体(Data\_Dizizer.py)プログラムリスト,339-357 行:モデルの保存 及び A5,6,7\_Learning\_Digitizer\_Manyual、3.3.1 入力

- pickle形式で保存された辞書型のデータである <データ名.pik >
- 'data','out','para'の3つのキーを持ち、下表のような要素で構成されている

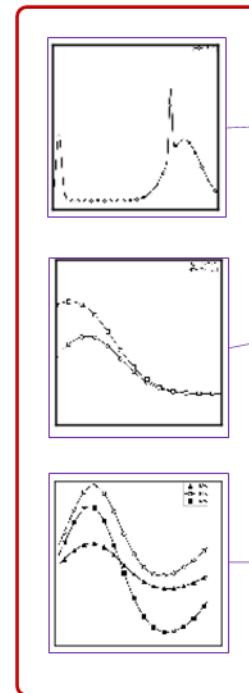
key	item
data	画像データ
out	教師データ
para	画像の大きさ, dpi, コメント, 波形の数, 共通パラメータ, マーカー

### 画像データと教師データの形式

画像データ:  
0-255の整数の数値をとり、そのサイズが1024x1024で1チャンネル(グレイタイプ)の画像配列 (1024,1024,1)

教師データ:  
0-1023の整数値をとり、1024 \* 波形数 (1~3) の整数配列

画像データ:  
(1024 x 1024)



教師データ:  
(1024 x 波形の数)

1波形:

[950, 947, 943, ..., 929, 930, 931]

1024個

2波形:

[608, 607, 606, ..., 837, 837, 837],  
[290, 289, 289, ..., 833, 833, 833]

1024個

3波形:

[558, 557, 556, ..., 547, 547, 546],  
[462, 460, 458, ..., 330, 329, 328],  
[523, 521, 519, ..., 617, 615, 613]

1024個

## 4. プログラムリスト

以下に、データ作成本体 (Data\_O.py) と関数ライブラリ (digitizer\_lib) に収められている波形生成関数 (Line.py) の全プログラムリストを示す

### 4.1 データ作成本体 (Data\_Digitizer.py)

#### 概要

1-8 行: 開始コメント  
 10-20 行: モジュールのインポート  
 28-33 行: 開始準備  
 35-98 行: データパラメータの設定  
 100-117 行: パラメータの確認  
 119-302 行: データの生成  
 304-349 行: チェック用の画像出力  
 351-372 行: モデルの保存  
 374 行: END  
 <br>

## 全リスト

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Jan.18,2020 / Fixed on Jun.30,2020
4
5  < 波形を数値化するための機械学習データを作成 >
6
7  @author: T.Kono
8  """
9
10 import cv2
11 import numpy as np
12 import matplotlib.pyplot as plt
13 import sys, string, random
14 import pickle, time
15 import json
16 from datetime import datetime
17
18 from keras.preprocessing.image import img_to_array
19
20 import kn_lib.line as line
21
22 # -----
23 #
24 #       Data Creating of Machine Learning for Waveform Digitizer
25 #
26 # -----
27
28 # --- Opening -----
29 # Date and time
30 today = datetime.today().strftime('%y%m%d-%H%M')
31
32 # start comment
33 print("\n< Data Creating of Machine Learning for Waveform Digitizer ",today,'>')
34
35 # --- Parameter input and define -----
36 # data dimension
37 dim = 1024 # dim = int(input(' Input Dimension of data => '))
38
39 # data number input
40 print("\n Input data number. if <= 99, datas are not saved')
41 loop = int(input(' => '))
42
43 # line number input
44 li_n = int(input(' Input Line number  = '))
45

```

```

46 comment = 'Digitizer_Data_'+str(li_n)+'-lin_'
47
48 # common parameter
49 '''同じdimの学習データは画像のパラメータを共有させる
50 ...
51 com_n = 'common_para_1024_2006251034.json'
52 with open(com_n, mode='r') as f:
53     com_para = json.load(f)
54
55 # parameter import
56 ndpi = com_para['para'][1]
57 lw = com_para['para'][2]
58 f_s = com_para['para'][3]
59 weigt = com_para['para'][4]
60 g_val = com_para['para'][5]
61 '''g_val[0][1] はマーカー間隔の最小値と最大値(pixel)
62 g_val[2]はマーカーの大きさ
63 ...
64 pix_poit = round(ndpi/72) # pixel number of point
65 ww = dim/ndpi ; hh = ww # graph width (inchs)
66
67 # peak number
68 '''1つの波形を作るために重ね合わせるガウス関数の数
69 ...
70 cho_p = [2,3,4]
71
72 # disturbance width
73 '''[0][1]:周波数摂動幅の上限下限
74 [2][3]:ピークの尖度摂動幅の上限下限
75 [4][5]:ライン間隔摂動幅の上限下限
76 ライン本数が3以下は摂動幅大きく,ライン本数が4以上は小さくする
77 ...
78 large = [-30,130,
79          50,200,
80          10,500]
81
82 small = [-5,105,
83          20,100,
84          30,100]
85
86 if li_n <= 3 :
87     lt = large
88 else :
89     lt = small
90
91 # marker setting
92 marker = ['+', 'D', '^', 'o', 's', 'v', 'x']
93 ''' maker sample
94 ['o', 's', 'v']
95 ['+', 'D', '^', 'o', 's', 'v', 'x']
96 ['*', '+', ',', '<', '>', 'D', 'H', '^', '_', '\
97      'd', 'h', 'o', 'p', 's', 'v', 'x', '|']
98 ...
99
100 # --- Parameter print and comfirmation -----
101 print("\n < Data parameter >')
102 print(' common parameter =',com_n)
103 print("\n Data number =',loop )
104 print(' line number =',li_n)
105 print(' line peak  =', cho_p)
106 print(' dim      =',dim, ', dpi =',ndpi,

```

```

107     ', fig size =','{:3.3}'.format(ww),
108     'x','{:3.3}'.format(ww),'inch')
109 print(' line width =',lw,'p','\n font size =',f_s,'p')
110 print(' marker =',marker)
111 print('\n comment :',comment)
112
113 yn = input(' Confirm data parameter, process start OK? y or n --> ')
114 if yn == 'n':
115     print('Parameter is wrong') ;sys.exit()
116
117 print('\n Process start ...'); s_time = time.time()
118
119 # --- Data Create Loop -----
120 # initial array set
121 ima_data = []
122 out_data = []
123 arrayimage = np.zeros([dim,dim,3],dtype=np.int8)
124 xxyy = np.zeros([2])
125
126 # mail loop
127 for l in range(0,loop):
128     plt.close()
129     fig = plt.figure(figsize=(ww,hh), dpi=ndpi)
130
131     # marker on/off
132     mk_on_off = random.choice([1,0])
133
134     # Base line create
135     p_n = np.random.choice(cho_p)
136     ff = line.line_n(dim, p_n)
137     fu = ff[0]
138     yn = ff[1]
139     pf = ff[2]
140     pk = ff[3]
141     ofs = ff[4]
142
143     weigt_ = []
144     yy = []
145     y_m = 0
146     # ライン間隔をランダムに変える
147     d_of = (np.random.randint(lt[4],lt[5]))/1000
148     for j in range(0,li_n+1):
149         y = np.zeros([dim])
150         for i in range(0,p_n):
151             """ベースとなる波形パラメータから下記のパラメータをランダムに
152             摂動させて波形を生成
153             """
154             # 周波数の摂動量
155             d_f = (np.random.randint(lt[0],lt[1]))/1000
156             # 尖度の摂動量
157             d_k = (np.random.randint(lt[2],lt[3]))/100
158
159             # 1つのピークを持った波形の生成
160             y_add = np.exp(-((fu-pf[i]+d_f)**2)/
161                             ((2*(pk[i]*d_k)**2))
162                             )
163             # p_n個のピーク波形の重ね合わせ
164             y = y + y_add + d_of
165
166             # オフセットを加えて波形を収納
167             yy.append(y + np.full(dim,d_of*j))

```

```

168
169 yy = np.asarray(yy)
170 ymax = np.max(yy)
171
172 """# 最小値がX軸に張り付かないための調整
173 ymin = np.min(yy)-(np.random.randint(5,20))/100
174 """
175 # レンジをを0-1に入れるための調整
176 yn = yy/(ymax*np.random.randint(101,
177                                 150)/100)
178
179 # 教師データを取得
180 out = (dim - (yn[0:li_n,:]*(dim-1))).astype(np.int32)
181
182 # draw scale randame choice
183 scale = np.random.choice(weigt,
184                           4)/(np.random.randint(75,
185                                                   150)/100)
186
187 # figure position
188 ax = fig.add_axes([0,0,1,1])
189
190 # marker style
191 if mk_on_off == 1 and li_n <= 3:
192     mak_ = np.sort(np.random.choice(marker,
193                                     li_n+1,
194                                     replace=False)
195
196 )
197
198 else:
199     mak_ =['','','','','']
200
201 # maker の間隔を設定範囲内でランダムに変える
202 dif = np.random.randint(g_val[0],g_val[1])
203
204 # legend position
205 lg_x = (np.random.randint(100,900))/1000
206 lg_y = (np.random.randint(100,900))/1000
207
208 # 3-9個のラベル文字をランダムに生成する
209 moj = np.random.randint(3,9)
210 d_n="".join(random.choices(string.ascii_letters, k=moj))
211
212 # line drawing loop
213 for k in range(0, li_n+1):
214     if mk_on_off == 1:
215         """マーカーがある場合は線種は実線で固定
216         マーカーがない場合は線種はランダムに選択する
217         """
218         ls_ = 'solid'
219     else:
220         ls_ = np.random.choice(['-', '-.', ':', '--'])
221
222     """li_n番目 はダミーライン
223     ダミーラインの線とマーカを白に,レジェンドのラベルを無効
224     """
225     l_col = 'k'
226     lab = d_n+str(k)
227     if k == li_n:
228         l_col = 'w'
229         lab = ''
230         fc_ = 'w'

```

```

229     ms = 5
230 else :
231     '''マーカ-の白抜きと黒塗り-をランダム-に選択
232     '''
233     fc_ = np.random.choice(['w','auto'])
234
235     ax.plot(fu, yn[k,:],
236             l_col,
237             ls = ls_,
238             linewidth = lw*scale[0],
239             label = lab,
240             markevery = dif,
241             marker = mak_[k],
242             mfc = fc_,
243             mec = l_col,
244             ms = g_val[2]*scale[1]
245     )
246 # tic drawing
247 sel = np.random.choice( ['in','out'] ) # 目盛線の方向
248 sel2 = np.random.choice( [True, False] ) # Y軸の目盛り数値のON/OFF
249 plt.tick_params(width = lw*scale[0],
250                 length = 8*scale[1],
251                 direction = sel,
252                 labelleft = sel2
253                 )
254
255 # legend drawing
256 fr_ch = np.random.choice([1,0])
257 ax.legend(loc = (lg_x,lg_y),
258           fontsize = f_s*scale[2]*0.8,
259           framealpha = fr_ch,
260           frameon = 'False'
261           )
262
263 # outer frame off
264 ax.spines["right" ].set_color('none')
265 ax.spines["top" ].set_color('none')
266 ax.spines["left" ].set_color('none')
267 ax.spines["bottom"].set_color('none')
268
269 plt.ylim(0,1) ; plt.xlim(0,1)
270
271 # Buffer からfigureデータ-を取得
272 fig.canvas.draw()
273 buf = np.frombuffer (fig.canvas.tostring_rgb(),
274                     dtype=np.uint8
275                     )
276 imar = np.reshape(buf,(dim,dim,-1))
277
278 flg = l%3
279 if flg == 0 :
280     '''3個に1個の割合でデータ-の前後にランダム-な空白-を挿入
281     '''
282     of_x = np.random.randint(0,dim*0.10,2)
283     cv2.rectangle(imar,(0,0),(of_x[0],dim),
284                  (255,255,255),-1) # left
285     cv2.rectangle(imar,(dim-of_x[1],0),(dim,dim),
286                  (255,255,255),-1) # right
287
288 gray_image = cv2.cvtColor(imar,cv2.COLOR_BGR2GRAY)
289 arrayimage = img_to_array(gray_image).astype(np.uint8)

```

```

290
291     ima_data.append(arrayimage)
292     out_data.append(out)
293
294 # 学習データ
295 ima_data = np.asarray(ima_data)
296 # 教師データ
297 out_data = np.asarray(out_data)
298
299 # Execution time
300 e_time = time.time()
301 print(' ...Finished. Execution time (s) = ',
302       '{:6.2f}'.format(e_time - s_time) )
303
304 # ---- Figure check -----
305 plt.close()
306 print('\n < Figure check > ')
307 n = 20
308 fig = plt.figure(figsize=(12,15))
309 n_list = np.random.randint(0,len(ima_data),n)
310 print(' . sample number = ',n_list)
311 for i in range(n):
312     nn = n_list[i]
313     ax1 = plt.subplot(5,4,i+1)
314     ax1.imshow(np.reshape(ima_data[nn,:],(dim,-1)),cmap='gray')
315     # plt.gray()
316     ax1.get_xaxis().set_visible(False)
317     ax1.get_yaxis().set_visible(False)
318
319 plt.show()
320
321 # ---- Data check -----
322 print('\n < Figure and Data comparison > ')
323 n = 4
324 fig = plt.figure(figsize=(10,6))
325 n_list = np.random.randint(0,len(ima_data),n)
326 print(' . sample number = ',n_list)
327 print(' . upper = input, lower = output')
328 for i in range(n):
329     nn = n_list[i]
330     ax1 = plt.subplot(2,n,i+1)
331     ax1.imshow(np.reshape(ima_data[nn,:],(dim,-1)),cmap='gray')
332     ax1.get_xaxis().set_visible(False)
333     ax1.get_yaxis().set_visible(False)
334
335     ax = plt.subplot(2, n, i+1+n)
336     for ii in range(0,li_n):
337         plt.plot(fu,dim-out_data[nn,ii,:], '--')
338     plt.ylim(0,dim-1) ; plt.xlim(0,1)
339     plt.gray()
340     ax.get_xaxis().set_visible(False)
341     ax.get_yaxis().set_visible(False)
342 plt.show()
343
344 # Special exit
345 '''データ-の点数が99個以下は、データ-の保存は行わず強制終了する
346 '''
347 if loop <= 99:
348     print('\n *** Datas were NOT saved, because Data numbers < 100.')
349     sys.exit()
350

```

```

351 # --- Data save -----
352 axis_test_graf = {'data':ima_data,
353                  'out':out_data,
354                  'para':(dim,
355                          ndpi,
356                          comment,
357                          li_n,
358                          com_n,
359                          marker
360                          )
361                  }
362
363 data_n = './'+comment+str(dim)+'_'+str(loop)+'_'+str(today)+''.pik'
364 with open(data_n,'wb') as f1:
365     pickle.dump(axis_test_graf,
366                 f1,
367                 protocol = 4
368                 )
369 print("\nData was saved >>> \n',data_n)
370
371 # del large memory
372 del ima_data; del out_data
373
374 # _____ END OF CODE _____

```

## 4.2 波形生成関数（Line.py）

### 概要

1-7 行: 開始コメント  
 9-9 行: モジュールのインポート  
 15-38 行: 開始準備  
 40-69 行: 波形の生成  
 71 行: return

### 全リスト

```

1  # -*- coding: utf-8 -*-
2  """"
3  Created on Jun.22,2020 / Fixed on Jun.30,2020
4
5  @author: T.Kono
6
7  """"
8
9  import numpy as np
10
11 # ..... Line Create Function .....
12
13 def line_n(nn,p_n):
14
15     '''p_n個のガウス関数を重ね合わせて波形を生成する関数
16
17     入力
18         nn : data number
19         p_n : peak number
20

```

```

21     出力
22         x : xの値
23         y : yの値
24         p_f: ピークの中心値
25         p_k: ピークの尖度
26         g_ : 全体のゲイン
27         ofs: オフセット
28
29     '''
30     # Xの値の生成
31     x = np.linspace(0,1,nn)
32     # yの領域生成
33     y = np.zeros([nn])
34
35     p_f = []
36     p_k = []
37     g_ = []
38     ofs = []
39
40     for i in range(0,p_n):
41
42         ''' ガウス関数で波形を発生させる
43             下のパラメータをランダムに変えて、p_n本の波形を生成して重ね合わせる
44
45         fo: ピークの中心周波数
46         pk: ピークの尖度
47         h : 全体のゲイン
48         ofse: オフセット量
49
50         '''
51
52         fc = (np.random.randint(-30,130))/100
53         pk = (np.random.randint(10,500 ))/1000
54         ga = (np.random.randint(100,150 ))/100
55         ofse = (np.random.randint(10,100))/1000
56
57
58         # ガウス関数の生成
59         y_add = ofse + ga*np.exp(
60             -((x-fc)**2)/
61             (2*(pk)**2)
62             )
63         # ガウス関数の重ね合わせ
64         y = y + y_add
65
66         p_f.append(fc)
67         p_k.append(pk)
68         g_.append(ga)
69         ofs.append(ofse)
70
71     return x, y, p_f, p_k, g_, ofs
72

```

End

