「計数実験の手引き」：Python で反復計数の実験

日本科学技術振興財団　尾崎　哲

ＧM計数管を使用した、遮へい、距離、プラトー特性、統計的変動の実験に適用できる

■計数実験データをパソコンに記録する

放射線計測の実験では、放射線の統計的変動の性質を踏まえて繰り返して測定する必要があるが、計測器の表示を読み取って記録する場合は、その後のデータ処理をパソコンで行うには改めて手入力することになる。ＧM計数管の出力をパソコンに入力して処理すれば、機械的な処理が可能となって効率のよい実験ができる。

ここでは、プログラミング言語 Python を使用して各種の放射線実験に適用するため、10 秒間の計数を繰り返すプログラムを紹介する。

■Python とは

無料で使用できるオープンソースの言語であり、音声や画像を扱うライブラリーも完備している。コマンドラインに入力するインタープリター言語なので、記述したプログラムがそのまま動作する。言い換えれば、コードの記述がそのままソース・プログラムとなっている。

Python 自体とライブラリーをインストールする必要はあるが、ネット環境から簡単にインストールできる。使用するパソコンやタブレットも、Windows や Android、iOS、Linux など、多くのプラットフォームで使用できる。

ただし、ここでは Windows10 を使用する。Python 自体や必要なライブラリーのインストールの方法、Python の文法などについては添付資料を参考にして調べてほしい。プログラムについても、多くの実例が公開されていて、そのまま利用できるものも少なくない。

■10 秒間の計数を反復するプログラム

これまでに述べてきた遮へいの実験や距離の実験では、通常、条件を変えながら各条件で 6 回の測定を反復する。各測定は 10 秒間として、計数率を CPM で記録することにしており、統計的変動の実験では、条件を変えずに 1000 点程度の計数を続ける。

そこで、プログラムでは、10 秒間の計数を継続して行いながら、計数率を CPM で画面に出力し、ファイルに書き込むことにした。

実行が確認できたプログラムを例として挙げる。#以下はコメント文で、各ラインを説明としている。なお、添付ファイルはテキスト・ファイルで、rename すれば Python ファイルとして実行できる。

まず、ユーザーのホーム・ディレクトリに当該ファイルをコピーしてから、Windows の画面左下にある「ここに入力して検索」をクリックして、「cmd」と入力し、コマンド・プロンプトを起動する。プロンプトにファイル名を入力すれば Python が起動する。以下では、プログラムの流れとポイントを解説する。

冒頭では、必要なライブラリーを呼び込む。なお、cv2 はプログラムの終了時のキー入力に使用する。

```
import pyaudio #マイク入出力処理
import numpy as np
import matplotlib.pyplot as plt #グラフ処理
import cv2 #カメラ画像処理
```

def は関数で、10 秒間サンプリングして、計数率を画面に出力し、その結果をファイルに書き込むまでの処理を行う。

```
    global m, threshold #回数と閾値の変数をグローバルと宣言
    m = 1 #測定回数
    threshold = input('Threshold?') #閾値を入力
    with open('cpm_data.csv', mode='w') as fc: #出力ファイルを新規モードで開く
        fc.write('Time(sec)' + ',' + 'CPM' + '¥n') #ヘッダーを書き込む
    def streamandcount(): #マイク入力から計数までの処理
    #stream
        CHUNK = 1024 #入出力の長さ
        FORMAT = pyaudio.paInt16 #2Byte 整数
        CHANNELS = 2 #ステレオ
        RATE = 44100 #サンプリング周波数(Hz)
        RECORD_SECONDS = 10 #録音時間(秒)
        p = pyaudio.PyAudio()
        stream = p.open(format=FORMAT,
                        channels=CHANNELS,
                        rate=RATE,
                        input=True, #入力のみ
                        frames_per_buffer=CHUNK) #ストリームを開く
        frames = [] #空の list を用意
        for i in range(int(RATE * RECORD_SECONDS // CHUNK)): #CHUNK 単位で切り捨て
            data_raw = stream.read(CHUNK) #CHUNK 単位で音声信号をサンプリング
            frames.append(data_raw) #430 回分を加算
        frames_int16 = np.array(frames) #list を numpy アレイに変換
        stream.stop_stream() #ストリームを閉じる
        stream.close()
        p.terminate()
        data = np.frombuffer(frames_int16, dtype='int16') #バイトデータを整数に変換

    #count
        pulse = 0 #計数を初期化
        data_l = data[::2] #ステレオの L チェンネルを取り出す
        onflag = False #閾値超過フラッグの初期値
        for k in range(len(data_l)): #data_l 全体が範囲
            dat = data_l[int(k)] #data_l の k 番目の値
            if int(dat) > int(threshold) and not onflag: #閾値を超過
                pulse = pulse + 1 #1 カウントを加算
                onflag = True #閾値超過フラッグを立てる
            if int(dat) <= int(threshold): #閾値以下になったら
                onflag = False #閾値超過フラッグを降ろす
        cpm = pulse / RECORD_SECONDS * 60 #CPM に換算
        print('CPM = ' + str(int(cpm))) #CPM を画面に出力
        with open('cpm_data.csv', mode='a') as fc: #出力ファイルを追記モードで開く
            fc.write(str(10 * m) + ',' + str(cpm) + '¥n')
```

バイトデータを整数に変換し、ステレオ録音の片チャンネルを抽出する。計数率は、パルス波高が閾値を超えるとフラッグを立てて、カウントを 1 増やし、閾値を下回るとフラッグを降ろす。10 秒間のカウント数を 6 倍して CPM として画面に表示するとともに、ファイルに書き込む処理を次々と更新する。

「q」をキーボードから入力すると、プログラムは終了する。

```
    while True: #キー入力を待つ
        if cv2.waitKey(1) & 0xFF == ord('q'):
    #q を入力すると、
            break #処理を終了
        streamandcount() #処理を反復
```

■実験方法

７セグメントＬＥＤ表示の計数装置から出力信号を取り出す。イヤホン・ジャックとパソコンのマイク入力端子をケーブルで接続するが、0.022μF 程度のコンデンサを並列に入れて入力電圧が過大(<5V)にならないようにすること。必要があれば、マイクのプロパティーを開き、レベルを調整する。