【ノート】 【令和5年度 先端技術等調査研究事業】

GC-MSのデータ解析用プログラムの作成
ー ガスクロマトグラムの解析における高度化・省力化の検討 ー

浅野 壮宏*
食品バイオ技術部(*現 企画・事業推進部)

ガスクロマトグラムのデータ解析において、GC-MSおよびGC-FIDの結果を複合的に解析することを目指し、まず、GC-MSのm/z毎の保持時間と強度の解析プログラムの作成に取り組んだ。その結果、データ変換・整形、ノイズ除去、ピーク検出、ピークのリスト化まで可能となったが処理時間は目標を上回った。

キーワード: GC-MS、クロマトグラム解析、Python

1 緒言

ガスクロマトグラフ(以下、GC)は香気分析等に広く利用されている。検出器の質量分析器(以下、MS)は定性・定量に広く用いられているが定量には標準物質が必要である。一方、汎用的に用いられる水素炎イオン化検出器(以下、FID)は定性に標準物質が必要だが、化合物の構造がわかっている場合には有効炭素数による半定量が可能である。そのため、MSとFIDのデータを組み合わせれば、標準物質がなくとも定性・半定量が行える。しかし、この解析では化合物毎に手動による照らし合わせが必要で、その作業は煩雑である。

そこで、GC-FIDとGC-MSの分析結果の照合を手作業に寄らず行うために、まずGC-MSのm/z毎の保持時間と強度の解析を行うプログラムの作成を試みた。

2 実験方法

2.1 解析プログラムの作成

GC-MSの解析を行うプログラムの作成にはPythonを使用し、ChatGPTを援用した。メーカー純正の解析ソフトでは保持時間と全イオン信号強度(トータルイオンカレント、TIC) のデータは得られるが、GC-MS の特徴である質量電荷比(m/z)分布の情報が出力されない。そこで、m/z 毎の保持時間と強度の解析プログラムの作成を以下の順番で行うこととした。

①データ変換・整形(汎用形式への変換)
②ノイズ除去(スムージング:Savitzky-Golay 法)
③ピーク検出(高次微分法による検出)
④ピークのリスト化(ピークと時間を整理)
⑤ピークの帰属(時間ごとにピークを整理)
⑥ピーク面積計算(カーブフィッティング)
⑦化合物推定(データベース MSbank を利用)
⑧ピークアライメント(分析データ間での位置あわせ)
⑨基準ピーク選択(装置間での位置合わせ)

作成プログラムは EXEフォーマット(.exe型式)とすることで環境に依存せず実行可能とし、また、グラフィカルユーザインタフェース(GUI)で操作できるようにした。

解析に用いたモデルデータは昨年度実施した研究課題[1)]で得た枝豆のデータを利用した。

3 結果と考察

メーカーの解析ソフトでは、内部情報として保存されている時間毎のm/zの値を出力するには煩雑な手作業が必要である。また、解析ソフトでは、総和であるTICに基づくクロマトグラム解析となるため、微小なピークや重なったピークを見落としたり一つのピークに誤認したりする可能性がある。そこでGC-MSで得られるm/z毎のクロマトグラムを手作業に寄らず得ることを目標に、まずGC-MSの専用形式から汎用書式であるCSV形式への変換を行った。GC-MSのメーカーに依存しないファイル形式のうち、mzXML形式からの変換を行った。出力されるCSVの書式は、m/z を行、保持時間を列、強度をセルに持つピボットテーブルとした。次に、得られた配列に対し、スペクトルに対するノイズ除去として広く知られているSavitzky-Golay 法によるスムージングを行った。

さらに、ピーク検出のため、スペクトルの解析手法として一般的な高次微分法によるピーク分離では2回微分を行った。高次微分後の波形に対しピークの検出およびリスト化を試みた結果、GC-MSで75分間の分析デー

タに対し、処理時間は3時間半以上かかり、目標時間を上回ったため、⑤以降の作業を断念した。時間短縮には異なるアルゴリズムによる処理が必要と考えられた。表1にGUI からの操作および任意ファイル名での読み込み、出力の要素を追加したプログラムを記載した。

4 まとめ

GC-MS と GC-FID を組み合わせた解析を容易にすることを目指し、まず GC-MSのm/z毎の保持時間と強度のデータ解析プログラムを作成した。データ変換・整形、ノイズ除去、ピーク検出、ピークのリスト化まで可能となったが、処理時間が目標時間を上回り、計画した全機能を搭載するには至らなかった。

参考文献

1) 浅野壮宏，小山誠司．熱脱着法による香気成分の基礎調査．宮城県産業技術総合センター研究報告．2023. 20. p. 105-108.

表1 作成したソースコード

```
import os
import numpy as np
import pandas as pd
from scipy.signal import savgol_filter, find_peaks
from tkinter import Tk, filedialog, Button, Label, Entry, StringVar

def open_file_dialog(window_size_var):
    root = Tk()
    root.withdraw()  # Hide the root window
    file_path = filedialog.askopenfilename(
        title="Open CSV file",
        filetypes=[("CSV Files", "*.csv")]
    )
    root.destroy()
    if file_path:
        window_size = int(window_size_var.get())
        process_csv(file_path, window_size)

def process_csv(csv_file_path, window_size):
    df = pd.read_csv(csv_file_path, index_col=0)

    # Convert the column labels to numeric type if possible to ensure correct sorting
    df.columns = pd.to_numeric(df.columns, errors="coerce")

    # Ensure window size is odd and greater than polynomial order
    if window_size % 2 == 0:
        window_size += 1

    poly_order = 3  # The polynomial order must be less than window_size - 1
    peaks_results = []

    for mass, intensity in df.iterrows():
        intensity_sg = savgol_filter(intensity.values, window_length=window_size, polyorder=poly_order, deriv=2)
        peaks, _ = find_peaks(-intensity_sg)

        for peak in peaks:
            # Add all peaks to results, we will filter out the zero-intensity ones later
            peaks_results.append({
                'Mass': mass,
                'Time': df.columns[peak],
                'Intensity': intensity_sg[peak]
            })

    # Convert the peaks results to a DataFrame
    peaks_df = pd.DataFrame(peaks_results)

    # Create a pivot table for all peaks
    pivot_peaks = peaks_df.pivot(index='Mass', columns='Time', values='Intensity').fillna(0)
    # Sort the columns
    pivot_peaks = pivot_peaks.reindex(sorted(pivot_peaks.columns), axis=1)

    # Output the pivot table to a CSV file
    peaks_output_file_path = f"{os.path.splitext(csv_file_path)[0]}_peaks.csv"
    pivot_peaks.to_csv(peaks_output_file_path)
    print(f"Peaks were successfully saved to {peaks_output_file_path}")

    # Filter out the zero-intensity peaks
    non_zero_peaks_df = peaks_df[peaks_df['Intensity'] != 0]

    # Group by Time and list Masses for each Time where intensity is non-zero
    grouped_mass = non_zero_peaks_df.groupby('Time')['Mass'].apply(list).reset_index()

    # Output the non-zero intensity Mass list to a CSV file
    mass_output_file_path = f"{os.path.splitext(csv_file_path)[0]}_Mass.csv"
    grouped_mass.to_csv(mass_output_file_path, index=False)
    print(f"Mass list was successfully saved to {mass_output_file_path}")

def create_gui():
    root = Tk()
    root.title("CSV Peak Detection")

    # Window size entry
    Label(root, text="Enter Window Size:").pack()
    window_size_var = StringVar(root, value="51")  # Default value for window size
    window_size_entry = Entry(root, textvariable=window_size_var)
    window_size_entry.pack()

    # Button to open file dialog
    open_file_btn = Button(root, text="Open CSV File", command=lambda: open_file_dialog(window_size_var))
    open_file_btn.pack(expand=True)

    root.mainloop()

if __name__ == "__main__":
    create_gui()
```