# Molecular Matching with Ion Mobility Spectrometry using Python

## Introduction

The idea of this research is to analyze and identify how useful the Python programming language can be in comparing and identifying substances using Ion Mobility Spectrometry. Ion Mobility Spectrometry is an analytical research method that helps in separating and identifying ionized molecules.

Ion Mobility Spectrometry (IMS) is widely used in identification of molecules in fields like security, military, research etc. and is extensively used in detection of explosives, drugs and chemical weapons.

This research explores the idea of a Python-based workflow for molecular matching with IMS Data with the following steps

1. Data retrieval from an SQLite database:

The SQLite database should have two tables, *measurement* and *library* where the measurement table holds the IMS Data with which we will be working on identifying the relevant data that we need to compare and identify the substance.

1. Calculation of reduced ion mobility:

Reduced ion mobility is an important parameter with which each molecule can be differentiated.

1. Library matching:

Compare the calculated reduced ion mobility values against a reference database (which should be the library table from the SQLite database) and assign the most likely molecular identity.

## Background and Theoretical Framework

### 2.1 What is Ion Mobility Spectrometry

According to James N Dodds and Erin S Baker, Ion Mobility Spectrometry (IMS) is the study of how ions move in gases under the influence of an electric field. The idea is the setup consists of multiple chambers.[[1]](#footnote-1) When we consider a rudimentary setup of an IMS, the first chamber is where the sample gas enters. Then the sample gas is ionized using an external source by exciting the molecules. Then the ionized gas enters another chamber which has an electric field. Each molecule has a different *drift velocity* with which they travel this chamber from start to end. We should know the voltage of this electric field, the temperature, pressure and multiple factors that are required to calculate the reduced ion mobility ion mobility of each molecule in that sample.

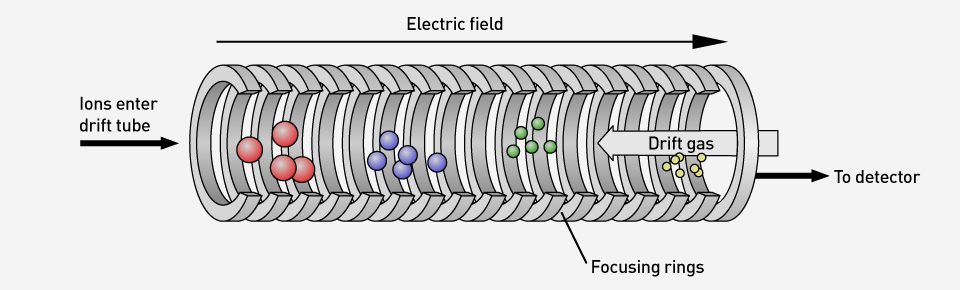


Figure Ion Mobility Spectrometry cross-section [[2]](#footnote-2)

### Principles of Ion Mobility Spectrometry

In IMS, ions that are generated from a sample are introduced in the drift region. In a uniform electric field, each ion’s drift velocity is proportional to the electric field strength . is the proportionality factor between the drift velocity and the electric field strength.[[3]](#footnote-3)

The reduced ion mobility is calculated when other factors such as the temperature and pressure during drift, standard temperature *T*0 = 273 K and standard pressure *p*0 = 1013 hPa are known.

To find the value, we can use the following formula.

Where is the drift length (the length of the chamber or tube where the ionized particles move), is the drift time and is the potential difference in the chamber.

### 2.3 Molecular Matching using

Molecular matching values is straightforward. We get the value of each spectrum using the above-mentioned calculations. We match the values against a reference library. is the value calculated. For each measured value , the algorithm searches for the library entry that minimizes . Each substance in the library is represented by up to 3 positive-mode and 3 negative-mode entries, which shows slight experimental variations. A simple molecular matching absolute-deviation criterion will be used to assign the *closest* substance to each measurement.

## 3. Python for Molecular Matching

Python can be an extremely effective tool in the process of molecular matching substances in Ion Mobility Spectrometry. There are multiple reasons to use Python as the main tool for the procedure.

### 3.1 File Parsing & Data Handling

* It is possible that the values necessary for the calculation of values may not necessarily be from an SQL database. Some IMS devices have serial communication capabilities with their own APIs. Data can be stored in files which can be simple text *.txt* or *.csv* or sometimes *JSON* files. Python has a vast set of libraries for parsing different types of data and it is extremely reliable as well.
* *Pandas* *DataFrames* are very efficient in converting data into tabular form providing indexing, filtering, grouping and even being part of the I/O (CSV <-> DataFrame <-> SQL) with very few lines of code. Pandas is widely used for data handling in Python.

### 3.2 Scientific & Statistical Computation

* Mathematical calculations and operations can be done fairly easily using the Python numeric stack. Some libraries like the *math* library and *numpy* library lets us implement a lot of formulas easily and can be used to make hundreds or thousands of measurements very quickly.

### 3.3 Database Integration

* For our project, database integration helps us a lot by defining tables in the *SQLite* Database which can be easily connected to our project using very good libraries. One such library is the *SQLAlchemy* library which helps us in defining the tables that are necessary to be read and or updated in python classes.
* Portability is also a very important advantage of using Python’s database integration. Switching between SQLite to PostgreSQL or MySQL or almost any other fairly known and commonly used database frameworks can be done with minor tweaks and corrections.

### 3.4 Visualization

* Python Visualization frameworks are well-known for easy use and understanding. Plotting drift-time distributions, trends or the spectra in itself can be fairly easily done. Frameworks like *MatPlotLib, Bokeh, Seaborn* can be used to make time-series of identifications or an interactive Bokeh Plot for matching errors.

### 3.5 Reproducability & Testing

* Unit Testing is an important aspect of programming any project. *PyTest* or *unittest* frameworks are really good at writing unit tests to test the project.
  + Parsing of header sections can be tested.
  + Peak-finding logic can be tested for both the positive and the negative spectra.
  + By providing known values, accurate values can be calculated and tested.
  + Library-matching logic can be tested using edge conditions (values that are close to matching but no matches).

## 4. Computational Tools: Python Ecosystem

Since Python offers such a rich ecosystem for some instrument control and data science, we will be able to do the following methodologies in finding the values of each spectrum.

* File Parsing: Existing data if it is in *csv* format or any other format, we can parse the code using modules/libraries like *re*, *numpy* and *csv.* Retrieving data such as raw channel[[4]](#footnote-4) values, IMS parameters and arrays.
* Numerical Computation: For numerical computation, we can either write our own functions which calculate the necessary value using the formulas mentioned earlier. Or for some other operations like drift-time calculation, *numpy* is a very good tool in handling such operations. *Pandas*  can be used to provide us the tabular data management.
* Databases: *SQLAlchemy* is used along with *SQLite* which enable structured data storage of raw measurements and the reference libraries.
* Visualization: *Matplotlib* helps in visualising the spectrum, maybe after Savitzky–Golay[[5]](#footnote-5) filtration and after finding the necessary peaks for which the values are to be calculated.

## 5. Programming

Molecular matching using values is done in the following steps.

1. Raw spectrum data that is received from the sensor, meta data such as temperature, pressure, drift tube length etc. are necessary to calculate values. These values are usually sent by the IMS device used. For example, [AIRSENSE Analytics GmbH](https://airsense.com/en) from Schwerin Germany, work extensively with IMS devices and they send the data using serial communication over RS-232 connection. If the necessary API is available, the necessary data can be parsed.
2. The Raw spectrum is then parsed and saved in an SQLite database. The database has two tables, *measurements* and *library.* The measurements table holds the necessary data that is required to calculate the values and library table holds existing substances with known values for reference.
3. A module is created that will handle all SQL related functions that can be called to get the relevant data for calculation.
4. Another module called *ims* is created to handle all IMS calculations and algorithms.
5. *Scipy* module is used to filter noises from the raw spectrum and can be used to find peaks from the spectrum. It is not necessary to calculate value of every single point on the spectrum.

### 5.1 Raw Data Handling

Firstly the *sqlite\_helper.py* module is created which will be handling any SQLite database related functionalities. The measurements table is created using the following SQL Query.

1. CREATE TABLE measurements (

2. id INTEGER PRIMARY KEY AUTOINCREMENT,

3. measurement\_time TEXT,

4. channel\_1 REAL,

5. channel\_2 REAL,

6. channel\_3 REAL,

7. channel\_4 REAL,

8. channel\_5 REAL,

9. channel\_6 REAL,

10. channel\_7 REAL,

11. channel\_8 REAL,

12. dilution REAL,

13. temperature\_drift\_tube REAL,

14. pressure REAL,

15. pos\_voltage REAL,

16. neg\_voltage REAL,

17. tube\_length REAL,

18. pressure\_offset REAL,

19. pressure\_gradient REAL,

20. pos\_spectrum BLOB,

21. neg\_spectrum BLOB

22. );

23.

This query creates the measurement table with the important parameters like measurement time, temperature of the drift tube during measurement, pressure during measurement, potential difference during measurement (voltage), tube length and the positive and negative spectra.

Similarly, the library table is created with the following query.

CREATE TABLE library (

    id INTEGER PRIMARY KEY AUTOINCREMENT,

    substance\_name TEXT,

    k0\_pos\_1 REAL,

    k0\_pos\_2 REAL,

    k0\_pos\_3 REAL,

    k0\_neg\_1 REAL,

    k0\_neg\_2 REAL,

    k0\_neg\_3 REAL

);

A class is created in python to handle

1. Ion Mobility Spectrometry: Fundamental Concepts, Instrumentation, Applications, and the Road Ahead (<https://pmc.ncbi.nlm.nih.gov/articles/PMC6832852/>) [↑](#footnote-ref-1)
2. https://www.analyticon.eu/en/ion-mobility-spectrometry.html [↑](#footnote-ref-2)
3. https://en.wikipedia.org/wiki/Ion\_mobility\_spectrometry#Ion\_mobility [↑](#footnote-ref-3)
4. Channels are the sensor values that are measured by the multiple sensors present during IMS measurement. In this project, we will be using an 8 channel IMS measurement. [↑](#footnote-ref-4)
5. https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.savgol\_filter.html [↑](#footnote-ref-5)