**MSRESOLVE QuickStart & Example Analysis**

Contents

[0. What is MSRESOLVE Used For? 2](#_Toc101448350)

[1. Installing the Programs MSRESOLVE Needs to Run 2](#_Toc101448351)

[Downloading and Installing Anaconda 2](#_Toc101448352)

[Recommended 4](#_Toc101448353)

[2. Getting Started 4](#_Toc101448354)

[2.1 Files A user Will Normally Change 4](#_Toc101448355)

[2.2 Running MSRESOLVE 5](#_Toc101448356)

[2.2.A. Running MSRESOLVE by using Anaconda Prompt / Terminal 5](#_Toc101448357)

[2.2.B.Running MSRESOLVE by Using Spyder 7](#_Toc101448358)

[2.3 End of QuickStart 10](#_Toc101448359)

[3. Example Analysis & How to Do a Typical Analysis 10](#_Toc101448360)

[3.1 What is the Example Analysis? 10](#_Toc101448361)

[3.2 Steps for a Typical Analysis (when reference file is already perfectly made) 10](#_Toc101448362)

[4. Changing Reference Data and Experimental Data 11](#_Toc101448363)

[4. Steps for making a typical reference file *from* reference patterns. 12](#_Toc101448364)

[4.1. Steps for Making Reference Pattern from Collected Data 12](#_Toc101448365)

[4.1. Concentration Units & Ionization Factors 13](#_Toc101448366)

[End of Example Analysis & How to Do a Typical Analysis 13](#_Toc101448367)

# 0. What is MSRESOLVE Used For?

Mass Spectrometry can be used for several purposes: to monitor chemical reactions, to identify and characterize unknown compounds, and/or to determine the absolute or relative concentrations of molecular species. However, the signals of species can overlap and thus interfere in the interpretation, due to overlapping mass fragmentation patterns. Such overlapping signals increase the difficulty and complexity of discerning individual molecules and their relative concentrations/signals from the full set of detected signals. Efforts to analyze the mass signals generated when there are multiple molecules present has motivated the creation of the MSRESOLVE program. MSRESOLVE is designed to analyze and discern the molecules and their absolute (or relative) concentrations by resolving (“separating the contributions”) within the signals. To do so, MSRESOLVE requires measured patterns (to analyze), reference mass fragmentation patterns (to compare against), and user inputs (for various user choices).

**These things must be in a single directory at the time of running MSRESLOVE.**From a software program point of view, MSRESOLVE takes three required inputs; a reference file (with reference fragmentation patterns), a measured file (the raw signals from mass spectrometry measurements), and a user input file (a text file detailing multiple options/choice for the MSRESOVLE program to execute). Some features may use additional supporting input files. MSRESOLVE reads these inputs and computes, according to the options chosen in the UserInput.py file, the concentrations/relative signals of each molecule. Typically, MSRESOLVE is used for analysis of time-series data, which is data collected as a function of time. The MSRESOLVE program then writes the solutions and data into output files, including for certain intermediate steps, such that the files include Preprocessed Data, Solved Concentrations, and other files..

**This tutorial will guide a user into installing the MSRESOLVE program and its dependencies as well as how to do a simple run. Further detail regarding the use of MSRESOLVE can be located in the MSRESOLVE\_MANUAL document.**

# 1. Installing the Programs MSRESOLVE Needs to Run

## Downloading and Installing Anaconda

MSRESOLVE was made in the python programming language. You will thus first need to install python. For most computers, the easiest way to do that, including the various python packages that MSRESOLVE needs as dependencies, is to install Anaconda. Download Anaconda from the official website. We also recommend that you install Notepad++ and set it as a default for opening .py files. Notepad++ adds color to various variables making it easier to decipher different variables.

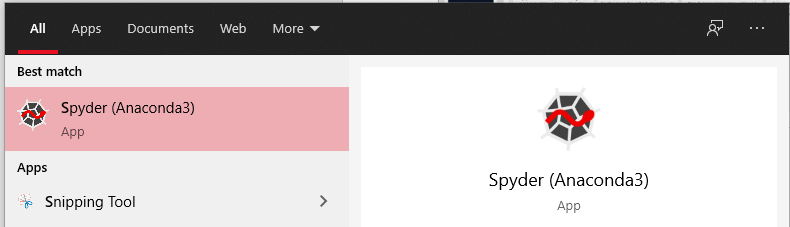
URL for Anaconda: <https://www.anaconda.com/products/individual>

Graphical user interface, text, application

Description automatically generated

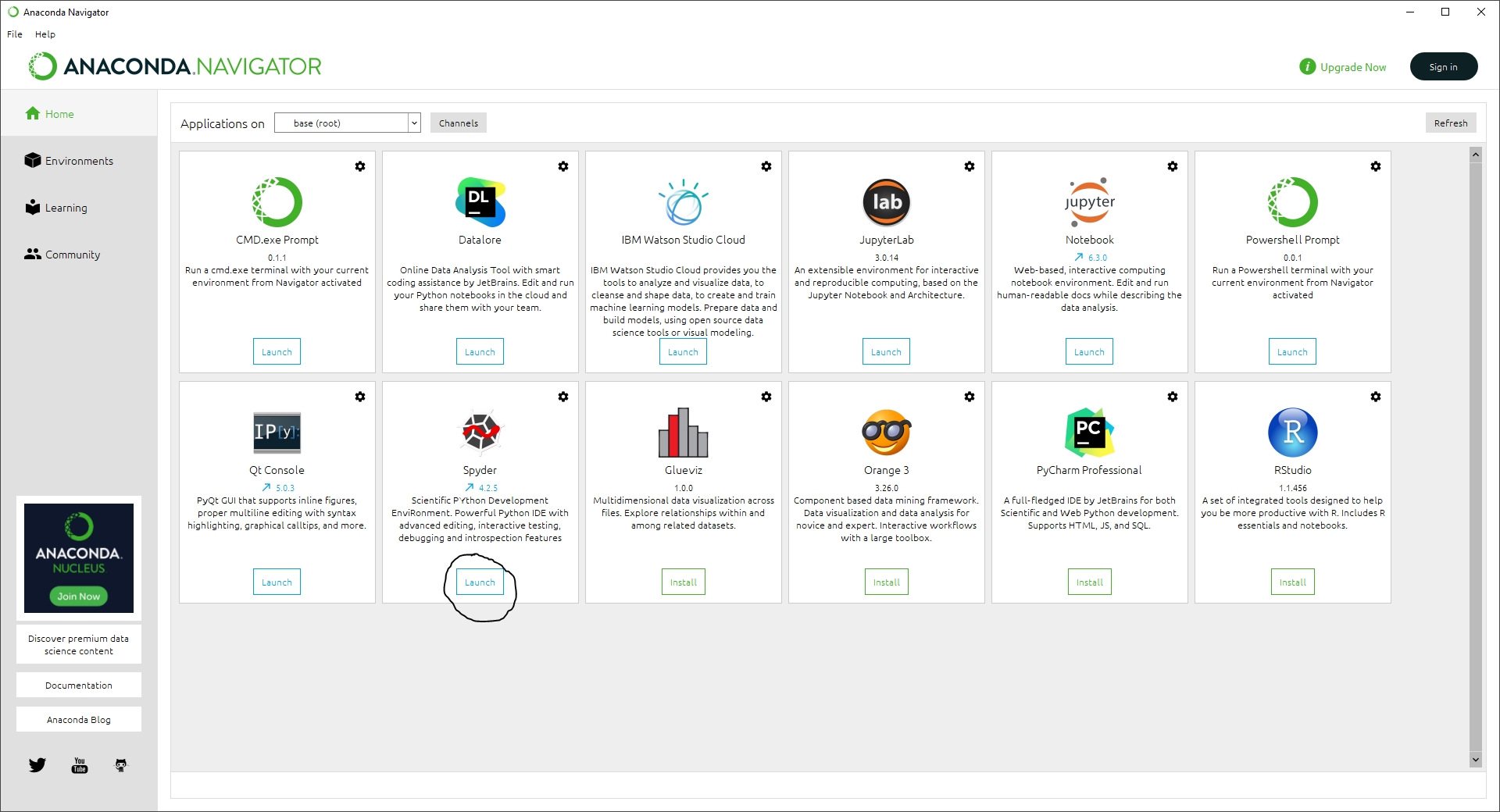
**Figure 1: Anaconda Site**

Click download and install. installing, make sure that **Anaconda Prompt** and **Spyder** are present on your computer. If you are using Windows, this can be done by typing in “Anaconda Prompt” into the Window search bar. The **Anaconda Prompt** should show up as an application (don’t open the program, yet). Now, try typing **Spyder** in the window search bar (don’t open the program, yet).



**Figure 2: Spyder program’s Icon that appears when Spyder is installed.**

If Spyder does not appear, open the Anaconda Navigator application. If the Spyder icon has the word “Install” shown, rather than “Launch”, then click install. After installing Spyder, if Spyder still does not appear among your windows programs, you may need to always launch it from Anaconda Navigator, or by typing “Spyder” in an anaconda prompt (see next section).



**Figure: Anaconda Navigator view when first opened. The icon demonstrating whether Spyder still needs to be installed or is ready to launch is circled.**

## Recommended

We also recommend that you install the normal (not portable) version of Notepad++ and set it as a default for opening .py (Python) files. Notepad++ adds color to various variables making it easier to decipher different variables.

URL for Notepad++: <https://notepad-plus-plus.org/downloads/>

# 2. Getting Started

## 2.1 Files A user Will Normally Change

In this QuickStart, you will not need to change any files. However, it is good to already learn which three files a user will normally change when starting a new analysis.

Open a windows file explorer and navigate to the directory where your MSRESOLVE files have been downloaded to. Inside this directory, you should see the sub-directory named **“ExampleAnalysis**”. In this directory, you will see a file named MSRESOLVE.py along with many other files. Below are the three file names for the Example Analysis inputs: the reference Data, Collected Data, and User Input.

**Reference Data-** Reference patterns that MSRESOLVEG will use to solve the concentrations of the collected data. File name: (**ExtractedReferencePattern.csv**)

**Collected Data-** Data that the user wants to process that will be loaded into the MSRESOLVEG program. File name: (**Collected**\_**Reference.csv**)

**User Input-** User Choices for guiding how the data will be processed and analyzed when MSRESOLVEG is run. File name: (**UserInput.py**)

## 2.2 Running MSRESOLVE

There are two ways of running MSRESOLVESG and both will be described below. One way is from using an anaconda prompt / terminal, and the other is within Spyder. Experienced Python users may use MSRESOLVE in other ways after following the QuickStart example.

**Reminder:** Whichever method is used, all input files needed for MSRELOVESG to run must be inside the directory where the analysis will be made. The standard practice when using MSRESOLVESG to do an analysis is to copy the entire directory of an old analysis (such as the example) and to then change the files in the new directory prior to run MSRESOLVE for the new analysis.

## 2.2.A. Running MSRESOLVE by using Anaconda Prompt / Terminal

Type “Anaconda Prompt” into the search bar of the Windows Start Menu and open the program. Alternatively open anaconda navigator, go to environments, and selecting the terminal for the python environment you wish to use.

Shape, rectangle

Description automatically generated

**Figure: example of how an Anaconda Terminal looks.**

To run the MSRESOLVE Example Analysis, first change the directory to the directory where the MSRESOLVE example is located. Leaving this terminal open, separately open a windows file explorer and navigate to the directory where your MSRESOLVE files have been downloaded to. Inside this directory, you should see another directory named **“ExampleAnalysis”**.Open the “ExampleAnalysis” directory thenclick on the address bar and copy the directory address.



**Figure: Example view after clicking on address bar within the sub-directory “ExampleAnalysis” located inside the MSRESOLVESG directory. Inside this directory, you should see the file MSRESOLVE.py and many other files.**

Type “cd “in the anaconda prompt (don’t forget the space) and paste the directory address.

Your prompt should look as shown below.

Text

Description automatically generated

**Figure: Example of how the prompt will look once the “ExampleAnalysis” directory is inserted.**

Next, press ENTER.

Text

Description automatically generated

**Figure: Example of how the prompt will look after Enter is pressed.**

Next, type “python MSRESOLVE.py” into the prompt and press ENTER on your keyboard.

Tip: typing “python ms” then pressing TAB will automatically finish the word MSRESOLVE.py.

Now, pressing Enter will run the MSRESOLVE Program. Please note, in this QuickStart example, the directory already contains the three inputs required, including measured data, a Reference Pattern, and certain settings in UserInput.py

During a real analysis, you will need to change the input files or and the settings inside UserInput.py

As MSRESOLVE is running, you will be greeted with popup windows containing graphs generated by MSRESOLVE from the data provided by the **“ExampleAnalysis”.** Initially, the graphs appearing will represent the Example Analysis data during pre-processing steps. Clicking the “X” to close the window of the graph will advance the code, thus showing more graphs. Note,when running MSRESOLVE from an anaconda prompt, the program will (by default) pause when each graph is made. It is thus important to close each graph so that the program can finish. Once all graphs are closed and the program finished, the program will have written various data files. The graphs are also saved into a subdirectory named “Graphs”. Please refer to section **2.3 Files Produced** to learn more about the files created by MSRESOLVE once an analysis is run.

You have run MSRESOLVE on your first example data! There are Solved Concentrations files in the directory. Now let us run this analysis the other way of running MSRESOLVE.

## 2.2.B. Running MSRESOLVE by Using Spyder

Type “Spyder” into the search bar of the Windows Start Menu.

Once Spyder is open, you will be greeted by a screen similar to below. The Spyder interface has several regions. The code window is on the left, a console window on the bottom right, and a small window that is primarily useful for seeing plots on the upper right. These features will be further explained below.A computer screen capture

Description automatically generated with low confidence

**Figure: Spyder Windows**

When using **Spyder**, you may want to change the colors’ theme. To change the theme, click the “Tools” menu dropdown at the top menu bar. Click on “Preference.” An alternative way to enter the “Preference” menu is by clicking the wrench icon located at the top of the application or by pressing **Ctrl+Alt+Shift+P** on your keyboard. Within the preferences, you can enter the “Appearance” menu. Here you can change the theme to your own preference in the sub menu labeled “Main interface”. You can also alter the syntax theme in the “Syntax highlighting theme” menu. The fonts can also be changed in the sub-menu labeled “Fonts”.

Location of wrench icon:

Graphical user interface, application

Description automatically generated

Example of sub directory “Appearance” in **Preference menu**:

A screenshot of a computer

Description automatically generated

After Spyder is open, open the “MSRESOLVESG” directory. Once in the “MSRESOLVESG” directory, navigate to the **“ExampleAnalysis” sub-directory.**

As before, to run the MSRESOLVE Example Analysis, it is necessary to go to the directory where to the example analysis is located. Open a windows file explorer and navigate to the directory where your MSRESOLVE files have been downloaded to. Inside this directory, you should see another directory named **“ExampleAnalysis”.** In this file you will find multiple .py files. Click and drag the MSRESOLVE.py into Spyder. There are also more .py files such as UserInput.py. The UserInput.py is used to make changes and specifications for how you want your data processed. We will not be changing any of these files, yet.

Your screen should look like similar to the image below:

Text

Description automatically generated

**Figure: Spyder window with the MSRESOLVE.py opened.**

Now that MSRESOLVESG.py is opened in **Spyder**, press the GREEN play button located in the upper region of the screen (the 7th icon in the image above). This will start and run the MSRESOLVE program. Once the program is finished running, you will see the statement “LogFile Complete” in the bottom right window of Spyder named “Console 1/A”. Within windows file explorer, look in your “ExampleAnalysis” directory. You will see a file named “LogFile”. This is where you can view some logs about the run. You can also view created graphs in the “Plots” tab on **Spyder** located above the “Console 1/A”. If you would like to refer to the graphs you created later in the future, open the directory where the MSRESOLVESG.py that you ran is located on your device. In this instance, it will be located inside the “ExampleAnalysis” directory. You should see a directory named “Graphs”. This is where all your graphs from your processed data will be located. The “Graphs” directory will show up in the directory that has the MSRESOLVESG.py that you have ran. Please refer to section **2.3 Files Produced** to learn more about the files created by MSRESOLVE once an analysis is run.

## 2.3 Files Produced

The directory in which you ran MSRESOLVESG.py will also have files such as “ScaledConcentrations.csv” and “ScaledConcentrations\_Statistics.csv”. “ScaledConcentrations.csv” includes a table of solved scaled concentrations (relative to Carbon Monoxide) for each molecule relative to time as shown in the figure below.

A screenshot of a computer

Description automatically generated with medium confidence

**Figure:** “ScaledConcentrations.csv” from the “ExampleAnalysis” directory after MSRESOLVE.py has been ran.

The “ScaledConcentrations\_Statistics.csv” file is also a file created after an analysis is ran. This file will include information such as scaled concentration means, standard deviations, and standard errors to name a few. Below is an example of the “ScaledConcentrations\_Statistics.csv” file.

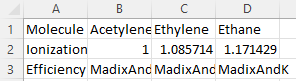
Graphical user interface, text, application, email

Description automatically generated

**Figure:** “ScaledConcentrations\_Statistics.csv” from the “ExampleAnalysis” directory after MSRESOLVE.py has been ran.

There are also several files produced with the tag “Exported” at the beginning of the file name.

The “ExportedIonizationEfficienciesSourcesTypes.csv” file is a file that provides that users with information such as the ionization efficiency factor used in calculating molecule scaled concentration and the method to obtain ionization efficiency as shown below.

****

**Figure:** “ExportedIonizationEfficienciesSourcesTypes.csv” from the “ExampleAnalysis” directory after MSRESOLVE.py has been ran.

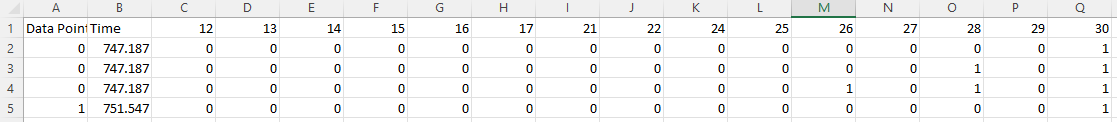
The “ExportedSLSUniqueMassesUsedInSolvingMolecules.csv” is a file that’s exported when the SLS solving method is chosen (Refer to the MSRESOLVE manual to learn more about SLS). This file shows the user the unique masses used in solving molecule concentration based on time. These unique masses play a key factor in solving scaled molecule concentrations.

Table

Description automatically generated

**Figure:** “ExportedSLSUniqueMassesUsedInSolvingMolecules.csv” from the “ExampleAnalysis” directory after MSRESOLVE.py has been ran.

The “ExportedSLSUniqueMassFragmentsUsed.csv” is a file that also shows the user the unique masses used in solving scaled molecule concentration based on time. Unique masses used are represented as “1” while masses not used are represented with “0” in the table.



**Figure:** “ExportedSLSUniqueMassFragmentsUsed.csv” from the “ExampleAnalysis” directory after MSRESOLVE.py has been ran.

The “ExportedSLSUniqueMoleculesAndChosenMassFragments.csv” is a file that shows the molecule solving order used in solving scaled molecule concentration based on time. This solving order is specific to SLS.

Table, Excel

Description automatically generated

**Figure:** “ExportedSLSUniqueMoleculesAndChosenMassFragments.csv” from the “ExampleAnalysis” directory after MSRESOLVE.py has been ran.

**Note: MSRELOVE has several ways to solve data and other advance features as described in the MSRESOLVE\_MANUAL**

## 2.4 End of QuickStart

As noted, MSRESOLVE is a program designed to analyze and discern the molecules and their relative concentrations contained within raw mass signals. MSRESOLVE can be run from Spyder or an Anaconda Prompt. MSRESOLVE has several methods for solving for concentrations, and other advance features. These are described in the main manual. In the next section, we will go into a little more detail for an example analysis.

# 3. Example Analysis & How to Do a Typical Analysis

## 3.1 What is the Example Analysis?

In this example, an Ethane pulse is fed into a reactor.

A reference file has already been made along with uncertainties as can be found in the “ExampleAnalysis” directory as Excel files. We want to see if there is any Ethyne or Ethene in the pulse.

The reference file uncertainties were set at 2 percent relative to the maximum fragment per molecule, within a user setting that will be provided below.

The collected file uncertainties have been set to be taken within a point radius of 2. This small point radius was chosen because the pulse has relatively sharp changes in concentration. Where this was set in the UserInput file will be provided below in section **3.2 Steps for a Typical Analysis**.

As can be seen in the graph named “scaledConcentrationsAfterAnalysis.png”, there is a large spike in Ethane between the time 900 and 950 .

Chart, histogram

Description automatically generated

**Figure:** “scaledConcentrationsAfterAnalysis.png” from the “ExampleAnalysis” directory after MSRESOLVE.py has been ran. A large spike in Ethane can be viewed.

## 3.2 Steps for a Typical Analysis (when reference file is already perfectly made)

1. Start with minimal working directory (just copy the “ExampleAnalysis” directory). Stick your reference file in there along with your collected data file.
   1. Make sure they are in the right format. Make sure there are no “empty rows” or “empty columns”. Open in notepad to check (excel can hide empty rows/columns).
2. Plot your collected data file in excel. Figure out if you need individual linear baseline corrections for each mass. Record *regions* of time for “early” and “Late” baselines. Like [10.0,15.0] might be an early baseline and [100.0,120.0] might be a late one. You can also just use a single region (just early baseline, no late one – “early” one can even be at the end of data). Zoom in and make as many graphs as needed. Sometimes you need to make one graph for each mass since they might need to have separate background correction factors.
   1. In UserInput.py, turn on **UserChoices['linearBaselineCorrectionSemiAutomatic']**
   2. Fill in the early baseline times and late baseline times.
3. Go to **UserChoices['minimalReferenceValue']['on'] = 'yes'**. A good initial choice is to make:
   1. **UserChoices['minimalReferenceValue']['referenceValueThreshold'] = [1.0]**
   2. **UserChoices['minimalReferenceValue']['referenceSignificantFragmentThresholds'] = [5.0]**
   3. The first of these settings makes values 0 in reference file [but now gets partially or completely added back with the '**implicitSLScorrection**' feature].
   4. The second of these settings specifies how big a peak has to be before it is significant – this \*only\* affects the order of solving the problem. This used to be more important before the uncertainties feature and slsweighting features were added. Now, those two features should usually already solve the problem to reduce the solving errors. Regardless, this feature is still good to use as in some analyses it could make a difference.
4. Turn on uncertainties:
   1. **UserChoices['uncertainties']['calculateUncertaintiesInConcentrations'] = True**
   2. **UserChoices['uncertainties']['referenceFileUncertainties'] = ‘File’** #If you don’t know your reference file’s uncertainties, a good initial choice is 2 or 5.
      1. You can also use the word ‘File’, that means you have to have an uncertainty for \*each\* value, like in the example provided.
   3. **UserChoices['uncertainties']['collectedFileUncertainties'] = 2** 
      1. This has four options:
         1. you can provide an integer (An integer defines a point radius to be used)
         2. You can use ‘Auto’ which simply uses a point radius of 5
         3. You can use ‘File’ and provide a file (will expect same file name as collected file with uncertainties)
         4. You can use ‘None’.
   4. The other two features of uncertainty are not yet implemented.
5. Data Analysis Methods:
   1. **UserChoices['dataAnalysisMethods']['answer'] = 'sls'#'inverse' or 'sls'**; sls is suggested
   2. **UserChoices['dataAnalysisMethods']['uniqueOrCommon'] = 'unique' #'unique' or 'common'**; now ‘unique’ is suggested.
   3. **UserChoices['dataAnalysisMethods']['slsWeighting'] = [1,0,0,0]** #You should probably keep this as [1,0,0,0] or make it [2,1,1,1]
   4. **UserChoices['dataAnalysisMethods']['implicitSLScorrection'] = True**
6. concentrationFinder: #This is only needed if you need to convert the units coming out to something like bar, torr, etc.

# 4. Changing Reference Data and Experimental Data

Changing Reference Data and Experimental Data is a very important part of knowing how to operate MSRESOLVE. To begin, it is very important to note that when preforming a new analysis using MSRESOLVE, it is best to create a completely new Directory that will include the MSRESOLVESG.py file along with the reference files and the user.input.py file if needed. Its best to copy the entire sub-directory named “ExampleAnalysis” and use the files located inside this directory as a reference to incorporate your own data. You would then incorporate your Reference Data and Experimental Data into these directories that you copied from the sub-directory named “ExampleAnalysis” into your newly created directory where you will be preforming your new analysis.

# 4. Steps for making a typical reference file *from* reference patterns.

1. Download JDX Files:
   1. Download JDX Files from NIST webbook for each molecule.
2. Format:
   1. Use JDX Converter to convert them into the right format. <https://github.com/AdityaSavara/JDX_Converter>
3. Reference Patterns From Your Own Instrument:
   1. When possible, make reference patterns from your own instrument. Gather some reference data from your own instrument, then extract the pattern. Under normal circumstances, you will have some reference patterns collected on your own instrument and some that you could not calibrate and had to get from NIST webbook.
4. follow the example inside the full package documentation for 190930TuningCorrectorInstructions.docx (this allows to make a reference file consisting of patterns collected from your instrument and from NIST).

## 4.1. Steps for Making a Reference Pattern from Collected Data

1. Collect a reference pattern from your own instrument.
   1. Advice: Usually you will want to take a scan where there are 10 minutes or longer with just inert gas or vacuum, and then you will introduce your chemical continuously for 10 minutes or longer, followed by another 10 minutes or longer of background again. For some setups 10 minutes is sufficient for each region, for other setups 1 hour is required for each region.
      1. It is okay to collect multiple calibrations in the same file.
2. Create a directory as if you will be making an analysis.
   1. Your collected data will be the “collected data” file.
   2. You will \*still\* need a real or fake reference pattern file. You can make a reference file that has a column for this chemical and then just put all the intensities as 1, for example.
3. Plot the data in excel and get the baseline correction information, as you would for a regular analysis.
4. Fill in the UserInput file with the reference extraction feature on and with export at each step turned on. Run the analysis. You will find that the first “Exported” file is the extracted reference pattern along with the uncertainties for each mass signal (as determined by the standard error of the mean from the region where the extraction occurred).
   1. Be advised that there may be other sources of error like baselines etc., so the true errors can be larger than the uncertainty provided.

## 4.1. Concentration Units & Ionization Factors

1. **Concentration Units:**
   1. In general, one can convert to concentration units using **UserChoices['concentrationFinder']**
      1. the concentrations can be for a single molecule or for a list of molecules. A list of molecules requires using “SeparateMoleculesFactors”
      2. one specifies a mass and a particular molecule and the intensity at that mass, and what concentration it corresponds to. Then all molecules get converted into that unit.
         1. One can also specify multiple molecules using lists for everything (other than the unit’s name). In that case any molecules that are not specified will get normalized according to the 1st molecule specified.
2. **Ionization Factors:**
   1. There are currently two common methods of using molecule Ionization factors with MSRESOLVE. One being Madix & Ko, the other being Ionization values provided by the user. If the user does not provide Ionization values, MSRESOLVE will calculate ionization values using Madix and Ko by default. Madix & Ko ionization is a method of determining ionization using the formula in the figure below. Simply, the equation used by MSRESOLVE is (0.6\*(X/14)) + 0.4 with X being the number of electrons present in the compound.
      1. Note: the ionization factors are not dependent ‘only’ on molecule type. It’s \*mostly\* on number of electrons and \*partially\* on molecule type.

Text

Description automatically generated

**Figure:** The Madix & Ko equation for calculating Ionization efficiency.

Another method that MSRESOLVE uses ionization factor is through the user manually inputting molecule ionization efficiencies. This method can increase the accuracy of the analysis. Though, the user must create an ionization data file. The figure below shows the ionization data file from the root directory of MSRESOLVE. Its best to copy this file and add your own data to it. This will help lower the chance of an error from occurring due to setting up the csv incorrectly. (Note: The name of the compound must match **EXACTLY** as the name of the compound in your reference file. Spaces and all.)

Graphical user interface

Description automatically generated with medium confidence

**Figure:** “181017ProvidedIonizationData.csv” from the root directory of “MSRESOLVESG

Once you have filled out the ionization data file, you will need to input the file name into the UserInput.py ‘ionizationDataFileName’ user option. Below shows an example of how/where you would input the file name for the file “181017ProvidedIonizationData.csv” from the figure above.

Text

Description automatically generated

**Figure**: Representation of where a user would input their ionization date file name into the UserInput.py

## End of Example Analysis & How to Do a Typical Analysis

The Example Analysis examined an Ethane Pulse that was fed into a reactor. A reference file with uncertainties was already made. Those uncertainties were set to be taken from a radius of 2 and it was found that there was no ‘detectable’ Ethene, or Ethyne in the Ethane Pulse as expected. Steps for a typical analysis such as setting up user input and plotting collected data were explained above. Steps for making a typical reference file from reference pattens using the JDX Converter were explained. Steps for making a reference pattern from collected data such as plotting data in Excel and getting a baseline correction formula was explained. Lastly, converting to concentration units was also explained in the text above. There are also many more features advance that MSRESOLVE has. These features can be found in the main manual.