LAB MANUAL & PROCEDURE
$C_4H_3F_7O$ (1,1,2,2-Tetrafluoroethyl 2,2,2-trifluoroethyl ether, TFTFE)
CAS Number: 406-78-0

Aziell Paul S. Limbo, Advisor: Dr. Karine Le Bris
California State University, Los Angeles, Department of Physics and Astronomy, Los Angeles, CA
(Dated: January 2026)

Abstract: Despite their relatively low concentration, fluorinated compounds significantly impact the climate due to their high radiative efficiency and long atmospheric lifetimes. Hydrofluoroethers (HFEs) have been introduced into the market as an alternative to chlorofluorocarbons (CFCs), hydrochlorofluocarbons (HCFCs) and hydrofluorocarbons (HFCs). These new compounds have shorter lifetime and, therefore, do not accumulate as much in the atmosphere. However, due to their high radiative efficiency, they are still greenhouse gases whose impact on the energy balance of the atmosphere needs to be evaluated. Our objective is to obtain the infrared absorption spectrum of HFE-347pcf2 using quantum chemistry computational analysis and experimental measurements via Fourier Transform spectroscopy. This will allow us to improve the reliability of the molecule's global warming potential value (GWP). This information can then be used by legislators to regulate the use and emissions of this compound into the atmosphere.

Contents

I. OBTAINING THEORETICAL CROSS SECTION

GitHub Repository link

`https://github.com/Limbotime2/IR_LAB_2027/`

A. Installation

Install the following (this is assuming `Ubuntu` is already installed, if needed, install `Ubuntu`. There are many different ways to install it if needed at all):

- **Avogadro** This program is used for "drawing" your molecule. Can be done using this link `https://avogadro.cc/`
- **ORCA** This software is the real meat of it all. Most calculations will be done here. I installed it onto `Ubuntu` using this tutorial: `https://spoken-tutorial.org/watch/ORCA+-+Computational+Chemistry/Installation+of+ORCA+on+Linux/English/`

  NOTE: I admit it's a little dated, but the instructions should be similar to the time of writing this manual. I have `ORCA 6.1.0` installed on my machine and used the very same tutorial. Let's hope not much has changed as time goes on.

  GOOD PRACTICE: It's also not a bad idea to read the `ORCA` documentation and run your first calculation at this point. Documentation can be found here: `https://www.faccts.de/docs/orca/6.0/manual/`. For my first run, I read the entire beginning up until I made my first `Hello Water` calculation as described. Documentation also tells you the "official" way of installing `ORCA`, but it's not easy to follow if you're not too sure what you're doing.

I am on Windows, and access `Ubuntu` through the `cmd` prompt. As for the text editor, I use `vim`, but any editor can be used.

B. Building Molecule & Input File

TFTFE has three conformers (I only know this from this source: `https://pubchem.ncbi.nlm.nih.gov/compound/164596`). This means three separate files will need to be created via Avogadro. In turn, we will run `ORCA` three times separately for each conformer.

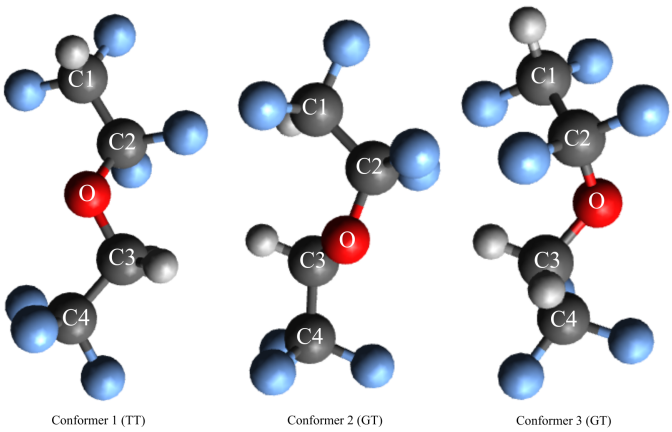

FIG. 1: Conformers of HFE-347pcf2. Visualization is done via Avogadro

NOTE: From `https://www.sciencedirect.com/science/article/pii/S0022285224000274`: by convention, conformers can be labeled depending on the dihedral angles about the C bonds. If the dihedral angle is close to 180 °, it is labeled Trans or Anti (T). If the angle is close to +60°or -60°, it is labeled Gauche(+) (G+) or Gauche (-) (G-) respectively.

Learn how to draw molecules in Avogadro using this online tutorial: `https://www.youtube.com/watch?v=zLBz8WiXbAs`. Personally, I uncheck the "Adjust Hydrogens" setting so I have complete control of what I am inputting - just don't forget your hydrogen atoms where needed!

Once a rough 3D drawing is made, the geometry of the molecule needs to be optimized. It's unlikely you've drawn it the exact mathematical way if you used just your mouse to draw your molecule. Go to:

```
Extensions >> Optimize Geometry
```

It should now look like a legit molecule with mathematically correct angles of bonds and atoms. Assuming everything looks correct, feel free to use this visual in final papers and posters by exporting it as a `png` (I totally did this in Fig. 1).

NOTE DEFINITELY save this file! Avogadro saves it's own files using the `.cml` extension.

Now it's time to set everything up for `ORCA`. Go to:

```
Extensions >> Orca >> Generate Orca Input...
```

Hopefully by this point you've learned a thing or two about how to configure your calculations from the `Hello Water` example described in the documentation for `ORCA`.

This time you edit the input file within the `Orca Input Parameters` pop-up window instead of manually writing an input file as you may have done with the `Hello Water` example using some arbitrary text editor. Around line 4 (where the exclamation point is) is where you tell `ORCA` what to calculate and more importantly *how* to calculate. Anything following a # symbol is a comment. For my first example, I used this input:

```
.
.
.
! B3LYP def2-SVP OPT FREQ
.
.
.
```

NOTE: More of a confession really. At the time of writing this, I don't fully understand the various DFT and basis set calculations yet. I have arbitrarily chosen the `B3LYP` method as it was referenced in my prior reading into doing any kind of calculations. `def2-SVP` was arbitrarily chosen at it is one (of the many) method(s) used in Dr. Le Bris's previously published papers. Hopefully this note gets deleted very soon so I may say otherwise.

The `OPT` and `FREQ` directions are taken as we are interested in furthering optimizing the geometry of the molecule (apparently Avogadro does not do this fully, think of it as an added step to ensure all calculations are accurate) as well as calculating the various frequencies of the molecule.

Around line 6 (where the asterisk, *, is) shows the xyz coordinates, charge (usually zero), and multiplicity (usually 1) - in that order. I've left this setting for all conformers (for now):

```
.
.
.
* xyz 0 1
.
.
.
```

Now generate the input file, and ensure the file is saved as an `.inp` file. I recommend saving it as (or something similar):

```
[molecule name].inp
```

Now it's time to run `ORCA`! Now run your input file as you may have done in the `Hello Water` example. For me, I run this command in `Ubuntu`:

```
orca [molecule file name].inp > [your molecule file name].out
```

It took me 30 minutes to an hour for each conformer. So it may take a while!

C. Extracting Raw Spectral Data From ORCA Output

Ensure that the IR spectrum data is available in the `ORCA` output file. It should have a header followed by data like this:

```
.
.
.
-----------
IR SPECTRUM
-----------

 Mode    freq    eps    Int   ...
        cm**-1 ...
------------------------------...
[DATA]
.
.
.
```

Now the goal here is to generate a spectra graph for your molecule. Ideally we want to extract this data into a 2 column format so we can plot it. This can be done by hand, but I've made a `MATLAB` program that accomplishes this by scanning and extracting the data and outputs it into a `.dat` file. The program is entitled `IR_DATA_EXTRACT.m`. The source code is provided in the accompanying GitHub repository.

For plotting the data. I used `xmgrace`. However, any plotting software can be used. The output of this is in Fig. 2. There is an additional output to this code to avoid redundant multiples of I/O. It is defaulted to output the total enthalpy, which will be needed later on for calculating population percentages in the next section.

D. Calculating Population Percentage

Now we wish to find the population percentage of each conformer. It's a simple calculation that needs no program to do. For a given conformer, i, the population percentage is:

$$\text{Population}\%_i = \frac{g_i \cdot exp(-\frac{G_i - G_{min}}{kT})}{z} \tag{1}$$

where g_i is the degeneracy value for the ith conformer, and z is the partition function:

$$z = \sum_i g_i exp(-\frac{G_i - G_{min}}{kT}) \tag{2}$$

k is the Boltzmann constant and G_{min} is the lowest Gibbs free energy from each conformer, and this value is provided by `ORCA`. T is temperature - and we assume room temperature for now at the value 298.15K. So each G_i corresponds to to the Gibbs free energy for the ith conformer. For TFTFE there are three conformers, so there are three Gibbs free energies. To ensure the population percentage calculation is correct, the sum of these percentages should of course be equal to 1. This is all calculated in `pop_perc.m`. All results are displayed in Table I.

NOTE `ORCA` gives each G value in Hartrees instead of Joules! So the conversion $1He = 4.359748199E - 18J$ must be applied.

NOTE Mathematically calculating the degeneracy value, g_i, requires knowledge of group theory, a completely stand alone field. However, for relatively small molecules, it is often easier to just find the degeneracy by rotating each group and see if that gives you the same molecule by mirror image or 180-degree rotation of the whole molecule.

- If it is the same molecule, you now have two states sharing the same energy $\rightarrow g_i = 2$
- If it is a different molecule, you have a new rotamer with its own harmonic modes and bands strength.

E. Gibbs Free Energy Verification & Reliability Test

`Gaussian`, another similar quantum chemistry calculation software, apparently outputs inconsistent and unreliable values for the Gibbs free energy value, G. It is absolutely worth checking if `ORCA` does the same. Otherwise, the results of population percentages will not be accurate. The best way to see if `ORCA` provides reliable values for G or not is to check the G values for different basis set calculations. It should absolutely be similar or the same. If not, then enthalpy values will be used in place of G for equations (1) and (2). To test `ORCA`, two new runs will be made, but this time I'll use the cc-pVDZ to cc-pVTZ basis sets and compare the G values of these runs. B3LYP will also be used for both runs. Results are shown in Table II

F. Generating Theoretical Cross Section for Molecule

At this time, it is assumed that the Gibbs free energy and corresponding population percentage values are good enough. Based off of Table II, I may have to switch to using the enthalpy values and their corresponding population percentages instead.

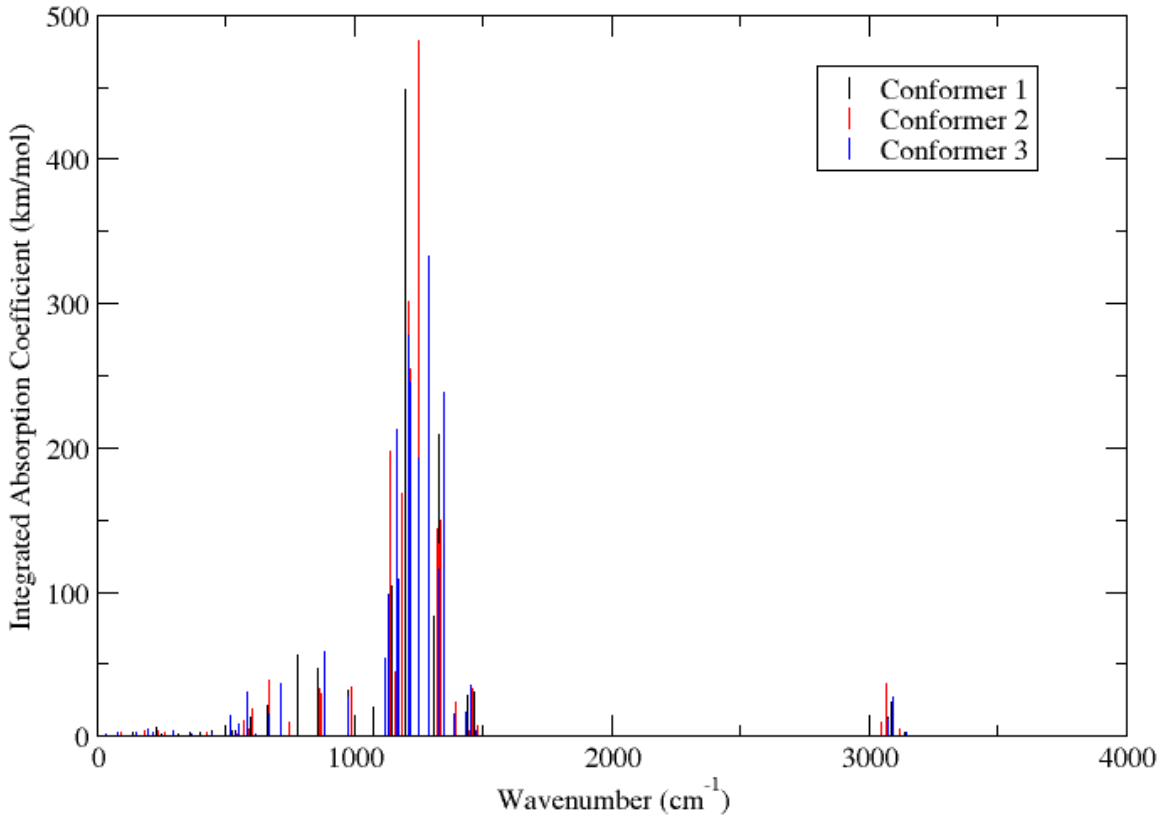

FIG. 2: Raw spectral graph of TFTFE produced by `xmgrace`. Data is directly from `ORCA`. I've plotted all coefficients as infinitesimally small peaks as these are only discrete values, obtaining continuous values are described later.

Conformer #	g_i	Gibbs Free Energy (J)	Population Percentage	Enthalpy (J)	Population Percentage
1	1	$-4.04248581 \times 10^{-15}$	83.2358451254%	$-4.0422778686 \times 10^{-15}$	97.17891360%
2	1	$-4.04245942 \times 10^{-15}$	0.136797200712%	$-4.0422433201 \times 10^{-15}$	0.02200837%
3	1	$-4.04247918 \times 10^{-15}$	16.6273576739%	$-4.0422632667 \times 10^{-15}$	2.79907803%

TABLE I: Table showing the Gibbs Free Energy in Joules for each conformer. Method used: B3LYP def2-SVP. $G_{min} = G_1$. Enthalpy values and their corresponding population percentages are also included for this run in case the Gibbs free energy values calculated are not reliable. See section **Gibbs Free Verification & Reliability Test** for more info.

So by this point, we have the theoretical relevant wavenumbers of interest, the integrated absorption coefficients, and conformer population weights. It's time to put it all together. The goal is to calculate the cross section. The roadblock is that we need to convert the absorption coefficients (y-axis) into the cross-section. A MATLAB code has been provided (authored by Dr. Le Bris), so the following are the mathematical notes based off that. That file is entitled `generate_spec_conformers.m` and has been uploaded to the GitHub repo located in the folder `KLB_m_source_codes`. However, actual implementation/calculations about to be described in this section are used in the program entitled `THEORETICAL_CROSS_SECTION.m`. The results are visualized in Fig. 3.

At present, the coefficients calculated by `ORCA` suggests that we only have discrete values, and we need continuous values as when it comes time to gather actual experimental data, the values will be given as a con-

	cc-pVDZ				cc-pVTZ			
Conformer	G.F.E (J)	Pop. Perc.	Enthalpy (J)	Pop. Perc.	G.F.E. (J)	Pop. Perc.	Enthalpy (J)	Pop. Perc.
1	$-4.0457459125 \times 10^{-15}$	91.0146294371%	$-4.0455266677 \times 10^{-15}$	83.86733969%	$-4.047156367 \times 10^{-15}$	97.28800450 24%	$-4.0469377964 \times 10^{-15}$	96.34310512%
2	$-4.0457169541 \times 10^{-15}$	0.080149947 06%	$-4.0454995632 \times 10^{-15}$	0.11587021%	$-4.0471263421 \times 10^{-15}$	0.06668778143 06%	$-4.0469089654 \times 10^{-15}$	0.08750983%
3	$-4.0457363444 \times 10^{-15}$	8.905220615 89%	$-4.0455198526 \times 10^{-15}$	16.01679010%	$-4.0471415104 \times 10^{-15}$	2.645307716 22%	$-4.0469242307 \times 10^{-15}$	3.56938505%

TABLE II: Table comparing the G values with it's corresponding calculated population percentage between the cc-pVDZ and cc-pVTZ basis sets. Enthalpy values and their corresponding population percentages are included. B3LYP is still used for both basis sets. $G_{min} = G_1$ for both basis sets.

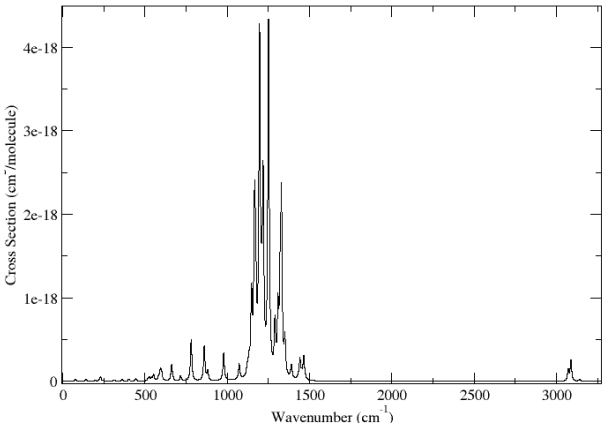

FIG. 3: Raw cross section output from `THEORETICAL_CROSS_SECTION.m`for B3LYP def2-SVP method. This here would serve as the theoretical data and will be compared to the upcoming experimental measured spectra.

tinuous spectrum (cross section). In other words, real spectra have broadened peaks instead of infinitesimally small peaks shown in Fig. 2. Mathematically broadening these infinitesimally small peaks is done via three different methods, Gaussian (not to be confused with `Gaussian`, the other computational quantum chemistry software mentioned earlier), Lorentzian, and Voigt. In the language of math and in general,

$$\sigma_{\text{total}}(\nu) = \sum_{i=1}^{N} w_i \cdot \sigma_i(\nu) \tag{3}$$

where σ_{total} is the cross section as a function of the wavenumber (e.g. what we want!!!), N is the number of conformers, w_i is the population percentage for the ith conformer, and $\sigma_i(\nu)$ is the spectrum of conformer i alone:

$$\sigma_i(\nu) = \sum_{j=1}^{M} S_{i,j} \cdot L(\nu - \nu_{i,j}) \tag{4}$$

where M is the number of discrete present for that particular ith conformer, $S_{i,j}$ is the integrated absorption coefficients, and $L(\nu - \nu_{i,j})$ is the broadening lineshape function (Gaussian, Lorentzian, or Voigt).

So in the special case where there is only one conformer:

$$\sigma_{\text{total}}(\nu) = \sigma_1(\nu) = \sum_{j=1}^{M} S_j \cdot L(\nu - \nu_{i,j}) \tag{5}$$

Broadening Lineshape Functions

We'll start with the Gaussian lineshape. The normalized Gaussian function is from the provided

`generate_spec_conformers.m` MATLAB function.

$$L_G(\nu - \nu_0) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(\nu-\nu_0)^2}{2\sigma^2}\right) \quad (6)$$

where σ is the standard deviation (defined as `sigma=(1/2.355)*FWHM` in code), ν is the wavenumber to be evaluated, and ν_0 are the known wavenumbers obtained through quantum calculation (from `ORCA`). The FWHM is the Full Width at Half Maximum of the band in cm^{-1}. This code recommends to begin at a value of 10 to start, then it can be adjusted. However, I've found that the HITRAN documentation (`https://hitran.org/docs/definitions-and-units/`) has alot more to say about that. I will definitely be exploring this value at a later date, but for now I will use 10 as the code suggests.

The Lorentzian function is:

$$L_L(\nu - \nu_0) = \frac{1}{\pi}\frac{\gamma}{(\nu-\nu_0)^2 + \gamma^2} \quad (7)$$

where γ is the Half Width at Half Maximum (HWHM) which is just FWHM divided by 2. Once again, the same HITRAN documentation has a bit more to say about this value.

As for the Voigt lineshape, HITRAN doesn't say much about it, and doesn't even give it's own equation for it. It does, however, note that when using the HITRAN database, the Voigt line shape is usually assumed. The Gaussian lineshape function is a little different by HITRAN, yet the Lorentzian lineshape function is the same by HITRAN. From the provided code,

$$L_V(\nu-\nu_0) = \text{ratio}\cdot L_L(\nu-\nu_0) + (1-\text{ratio})\cdot L_G(\nu-\nu_0) \quad (8)$$

where the ratio (according to the code) is the fraction Lorentzian versus Gaussian in the Voigt function. A value of 1 would correspond to a pure Lorentzian function. If a ratio is not specified, the default ratio value is 0.5.

Now it brings the question, which function does one use? HITRAN suggests that in the lower atmosphere, pressure broadening of spectral lines dominates and a Lorentz profile can be used. Conversely, in the low-pressure environment of the upper atmosphere, Doppler-broadening dominates the line shape and a Gaussian profile can be used. I'm going to be completely honest, I don't know where my molecule, TFTFE, has been detected other than it has been mentioned that is is used in batteries. So I'll assume a Lorentz profile because I'm assuming there aren't many batteries in the upper atmosphere that is causing concern.

G. Extracting Data from HITRAN

If the molecule has been studied before, there's a chance the cross section data is available on HITRAN. To access it, you must create an account (it's totally free!). For example, TFTFE is available for three different temperatures. However the output `.xsc` is formatted in a very specific way. Details can be found on this page of HITRAN's documentation: `https://hitran.org/docs/cross-sections-definitions/`.

NOTE The headers of the `.xsc` files is VERY important! It specifies the number of points, but each line contains a set of 10 points, but not all data points will be a multiple of 10, so the last record of the file will likely be extended with zeroed out values - source: `https://www.sciencedirect.com/science/article/abs/pii/S0022407319307642#preview-section-snippets`). This paper also provides more info on how to use HITRAN.

The code `read_HITRAN.m` parses data from the `.xsc` file, downloaded from HITRAN, and creates an output two column file that is ready to be plotted. The plot for TFTFE is shown in Fig. 4.

Some calibration is needed in order to fit my data to HITRAN's data. This can be done by modifying values for variables `a` and `b` in the code `THEORETICAL_CROSS_SECTION.m`. Mathematically finding these two values is described in the next section.

H. Calibrations

Now I wish to find values that fits `a` and `b` so that my calculated values matches the published theoretical values provided by HITRAN. I'll adopt a least squares method described by Cheney and Kincaid's *Numerical Mathematics and Computing*. I wish to calibrate my data such that there are two variables, a and b, where:

$$x_{\text{HITRAN}} = a \cdot x_{\text{ORCA}} + b \quad (9)$$

I don't wish to calibrate the cross section intensities, just the x-axis wavenumbers. The coefficients a and b are to be determined according to the principle of least squares:

$$\varphi(a,b) = \sum_{k=0}^{m}(a \cdot x_{\text{ORCA}_m} + b - x_{\text{HITRAN}_m})^2 \quad (10)$$

For a and b:

$$\frac{\partial\varphi}{\partial a} = 2\sum_{k=0}^{m}(a \cdot x_{\text{ORCA}_m} + b - x_{\text{HITRAN}_m}) \cdot x_{\text{ORCA}_m} = 0$$

$$= \sum_{k=0}^{m} x_{\text{ORCA}_m}(a \cdot x_{\text{ORCA}_m} + b - x_{\text{HITRAN}_m}) = 0$$

ORCA (cm^{-1})	3088.2	1464.4	1441.6	1166.6	1276.5	978.1
HITRAN (cm^{-1})	2999.2011	1436.4430	1418.0923	1144.6405	1256.1908	970.7764

TABLE III: Peaks and valleys selected to "line up" to calibrate.

$$a\sum_{k=0}^{m} x^2_{\text{ORCA}_m} + b\sum_{k=0}^{m} x_{\text{ORCA}_m} = \sum_{k=0}^{m} x_{\text{ORCA}_m} \cdot x_{\text{HITRAN}_m} \tag{11}$$

$$\frac{\partial \varphi}{\partial b} = 2\sum_{k=0}^{m}(a \cdot x_{\text{ORCA}_m} + b - x_{\text{HITRAN}_m}) \cdot 1 = 0$$

$$= \sum_{k=0}^{m}(a \cdot x_{\text{ORCA}_m} + b - x_{\text{HITRAN}_m}) = 0$$

$$a\sum_{k=0}^{m} x_{\text{ORCA}_m} + b \cdot \sum_{k=0}^{m} 1 = \sum_{k=0}^{m} x_{\text{HITRAN}_m} \tag{12}$$

Now it can be seen that equations 11 and 12 forms a linear system:

$$\begin{bmatrix} \sum x^2_{\text{ORCA}_m} & \sum x_{\text{ORCA}_m} \\ \sum x_{\text{ORCA}_m} & \sum 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum x_{\text{ORCA}_m} \cdot x_{\text{HITRAN}_m} \\ \sum x_{\text{HITRAN}_m} \end{bmatrix}$$

Now there is another problem. Currently, the number of entries in x_{ORCA} and x_{HITRAN} do not match, and they are on different grids with different step sizes. So instead of using the entire spectrum, I'll select an arbitrary number of peaks (and some valleys) I wish to line up manually. These chosen wavenumbers are in Table III This entire method is coded in `CALIBRATE_source_code.m`. Values for the coefficient of determination, R^2, both before and after calibration are also calculated within this code. Results are shown in Fig. 5.

II. CALCULATING DATA FROM THE CROSS SECTION

A. Calculating The Integrated Band Strength

Integrating the wavenumber vs. cross section curve gives the integrated band strength.

$$S(\tilde{\nu}_1, \tilde{\nu}_2, T) = \int_{\tilde{\nu}_1}^{\tilde{\nu}_2} \sigma(\tilde{\nu}, T) \cdot d\tilde{\nu} \tag{13}$$

However, the current issue is that we only have data as a set of x- and y- values. To integrate, you'd need a *continuous* function. So either regression or interpolation is needed.

Polynomial Regression

This method, at least on my machine, did not work. The degree needed to fully express the spectrum as a polynomial is way higher than the current code can work. I have yet to fully diagnose and debug exactly what is going on, but as of now I believe the problem lies as a question of numerical stability and memory allocation. I adopt Cheney and Kincaid's method of polynomial regression. For consistency, I write this algorithm in MATLAB, and it is saved as `poly_regres.m`.

Essentially, the goal is to represent the trend of the table of wavenumber vs. cross-section as a polynomial:

$$p_N = \sum_{i=0}^{N} a_i x^i \tag{14}$$

where N is the degree used that best describes the data. This constructs a standard basis of $\{1, x, x^2, x^3, ..., x^N\}$ and one could probably do a method of least squares to find the appropriate coefficients a_i. However Cheney and Kincaid constructs orthogonal polynomials $\{q_0(x), q_1(x), ..., q_{m-1}\}$ using a recurrence relation:

$$q_1(x) = 1$$

$$q_2(x) = x - \alpha_0$$

$$q_n(x) = (x - \alpha_{n-1})q_{n-1}(x) - \beta_{n-2}q_{n-2}(x) \tag{15}$$

where α and β are defined as:

$$\alpha_{n-1} = \frac{\sum_{i=1}^{m} x_i[q_{n-1}(x_i)]^2}{\sum_{i=1}^{m}[q_{n-1}(x_i)]^2} \tag{16}$$

$$\beta_{n-2} = \frac{\sum_{i=1}^{m} x_i q_{n-1}(x_i) q_{n-2}(x_i)}{\sum_{i=1}^{m}[q_{n-2}(x_i)]^2} \tag{17}$$

where m is the number of data points. Therefore equation 14 becomes:

$$p_N = c_0 q_0(x) + c_1 q_1(x) + ... + c_N q_N(x) \tag{18}$$

Instead of a_i's now there is now c, and the orthogonality property makes it easier to compute:

$$c_k = \frac{\sum_{i=1}^{m} y_i q_k(x_i)}{\sum_{i=1}^{m} [q_k(x_1)]^2} \tag{19}$$

Now the looming question still remains: what degree of N should we use? The algorithm then tests polynomials of increasing degrees $(1, 2, 3, ..., N)$ and calculates the variance (mean squared error) of each:

$$\sigma_n^2 = \frac{1}{m-n} \sum_{i=1}^{m} [y_i - p_n(x_i)]^2 \tag{20}$$

The optimal degree used is one where adding more terms to p_n doesn't significantly improve the fit. This means that a degree of $m-1$ is totally possible, but it may be overfitting the data. This is where the issue lies. For this particular spectrum, the code wants to end at degree of 48, yet it calculates to degree 52 before data becomes "NaN" but it is still not high enough of a degree to accurately describe the data. Just by looking at the (attempted) fitted curve, I predict a degree of 60-70 is truly what I need. I have not attempted to integrate it since is such a low approximation.

Cubic Splines Interpolation

Since the previous method is (presumably) numerically unstable for our data set, I then try to come up with a function, really functions (plural), using Cheney and Kincaid's method of cubic spline interpolation. For my own convenience, I now switch programming languages to Fortran as I actually already had programmed a module containing procedures of interpolation by splines. This module is entitled `splines.f95` and the source code for this is `interpolate_integrate.f95`. The method is described once again by Cheney and Kincaid.

For each data point, (t_i, y_i), $i = 1, 2, ..., N$, there is the interval in between each point $[t_i, t_{i+1}]$. The natural cubic spline on each of these intervals is:

$$S_i(x) = a_i + b_i(x - t_i) + c_i(x - t_i)^2 + d_i(x - t_i)^3 \tag{21}$$

Each cubic spline has these conditions:

1. **Interpolation Condition**: $S(t_i) = y_i$ denotes that the spline shares boundaries with the data points
2. **Continuity Condition**: $\lim_{x \to t_i^-} S^{(k)}(t_i) = lim_{x \to t_i^+} S^{(k)}(t_i)$ $(k = 0, 1, 2)$. Note that this is *only* for the interior knots; all the x-data points except the lowest and highest values.
3. **Boundary Condition**: $S''(t_0) = S''(t_n) = 0$. The second derivative at the lowest and highest x value are both equal to zero.

Since S'' is continuous, the following set is defined:

$$z_i \equiv S''(t_i) \tag{22}$$

It is desirable to get these z_i values. And that is exactly what the subroutine `spline3_coef` does. It does this be solving the tridiagonal system:

$$\begin{cases} z_0 = 0 \\ h_{i-1} z_{i-1} + u_i z_i + h_i z_{i+1} = v_i \\ z_n = 0 \end{cases} \tag{23}$$

where h_i, u_i, v_i, and b_i are defined as:

$$h_i = t_{i+1} - t_i$$

$$u_i = 2(h_{i-1} + h_i)$$

$$v_i = 6(b_i - b_{i-1})$$

$$b_i = \frac{1}{h_i}(y_{i+1} - y_i)$$

The tridiagonal system, and thus `spline3_coef`, is solved as described in the text by Cheney and Kincaid.

Now that the z_i's have been obtained, another subroutine is called: `MC_integral` where the z_i's are used to evaluate the function and to simultaneously solve equation 13. This is done through Monte Carlo integration. The result of this integral is the band strength which is displayed on the terminal screen. For this example, the integrated band strength is $3.5318805515246983 \times 10^{-16} \pm 1.9566570362548759 \times 10^{-48}$. This value falls within literature (see `https://www.sciencedirect.com/science/article/abs/pii/S0022285221000783?via%3Dihub`)

III. PROGRAMS WORKFLOW

Now that the technique has been described, It's now time to talk about how to actually gather this data. It's very easy; all that needs to be done is to run `CALCULATE_source_code.m`. It should use nearly all the mentioned codes (just the functions). Then `CALIBRATE_source_code.m` is used for calibrations. Specific instructions are in the comments of each source code.

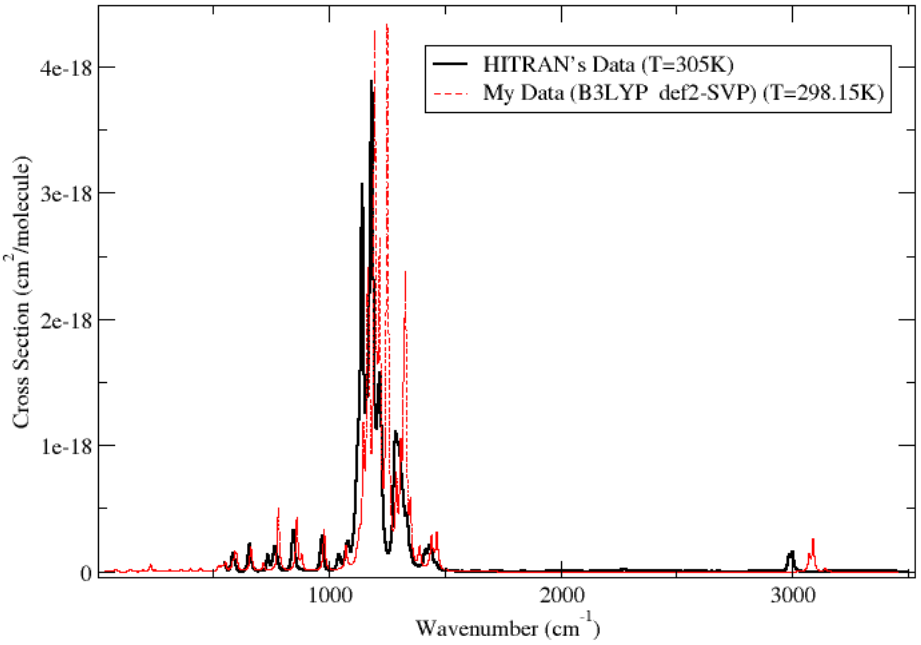

FIG. 4: HITRAN's data vs data plotted in Fig. 3. Obviously some calibration is needed somewhere. $R^2 = 0.996243$

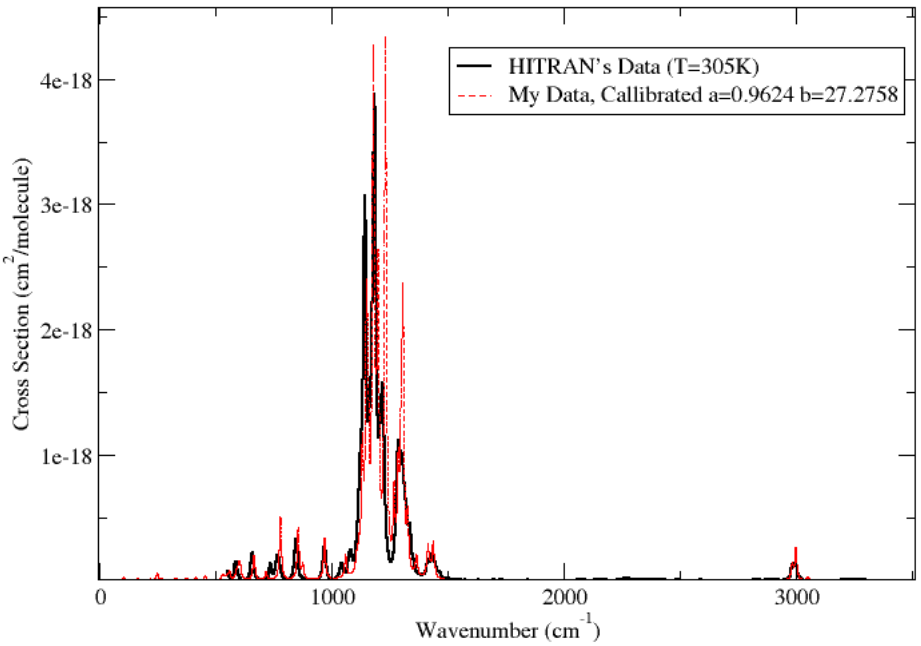

FIG. 5: HITRAN's data vs my callibrated data. $R^2 = 0.999983$