

---

# About Spectral Data Table (SDT) Files

Henry Bruhns

February 7, 2025

---

Spectral data table (SDT) files are designed as input parameters for a program called Spectral Range Manager (SRM). The SRM is the software solution of a quantitative method to determine a well suited spectral range for sensitive and selective gas analysis, using distributed feedback interband cascade lasers to perform "Tunable Diode Laser Absorption Spectroscopy" (TDLS). The method is described in an article, entitled: "Determining the most suitable spectral range for TDLS - a quantitative approach", published in the Journal of Quantitative Spectroscopy & Radiative Transfer [1]. This note describes the file and data format of SDT files, gives an examples of how to create SDT files from the HITRAN database and shows the usage of SDT files together with the Spectral Range Manager.

---

## 1 Introduction

In connection with absorption spectroscopy, spectra can be understand as functions of the form  $S(\lambda)$  or  $S(\nu)$ , where  $S$  represents the energy absorption intensity and  $\lambda$  or  $\nu$  the energy of electromagnetic radiation as the independent function variable. In spectroscopy, the energy of electromagnetic radiation is defined as the wavelength  $\lambda$  or the wavenumber  $\nu = 1/\lambda$  of an electromagnetic wave. Although spectra are understand as continuous functions (neglecting quantum physics), they are practically treated in the form of discrete functions in data processing, represented as a finite amount of  $(S, \lambda)$ , or rather  $(S, \nu)$  data-pairs. An SDT file is basically a list of such data pairs, supplemented by additional information as described below.

## 2 The SDT file and data format

SDT files are plain "human readable" text files. The file format is the Unicode transformation format – 8-bit (UTF-8). Therefore, a SDT file can be opened and edited with almost any text editor.

### 2.1 Text block

The text is divided into separate text blocks, introduced by a keyword and concluded by an end-statement. Every line of text must end with the control character "line feed" (UTF-8: 0x0A).

```
@keyword:  
content...  
...  
@end
```

## 2.2 Keywords

The keyword describes the type of content between the delimiters `@keyword:` and `@end`.

### 2.2.1 substance

The keyword `@substance:` introduces the name of the molecule or molecule-mix giving the spectral data. For example:

```
@substance:
Test gas: 95.5 ppm ethane in nitrogen
@end
```

### 2.2.2 xunit

The keyword `@xunit:` introduces the unit of the independent variable. If it is a wavelength, the SRM accepts nm as unit. If it is a wavenumber, the SRM accepts 1/cm as unit. For example:

```
@xunit:
1/cm
@end
```

If the unit is 1/cm, indicating a wavenumber, the SRM converts the whole data table to wavelengths with the unit nm and rearranges the data table to ascending values. After processing, the content of the keyword will be changed to nm automatically.

### 2.2.3 yunit

The keyword `@yunit:` introduces the unit of the absorption intensity. If the unit is unknown, the SRM accepts a question mark as unit. For example:

```
@yunit:
?
@end
```

In this case, the SRM normalizes the values of absorption intensity over the entire wavelength or wavenumber range. After processing, the SRM changes the content to:

```
@yunit:
1
@end
```

### 2.2.4 data

The keyword `@data:` introduces the spectral data table. Each line of the content represents a data pair of  $(\lambda, S)$  or  $(\nu, S)$  values. The delimiter between the values must be a single space (UTF-8: 0x20). For example:

```
@data:
3275.2941 0.016210
3276.3453 0.024034
3276.6484 0.014384
3277.4350 0.019541
3277.6073 0.019352
.
.
.
3524.4220 0.004215
3525.5348 0.005484
3525.6830 0.004520
3525.9970 0.005178
3526.6293 0.006075
@end
```

The SRM accepts values written in scientific notation, for example:

```
15.75
1.575E1
1575e-2
-2.5e-3
25E-4
```

The independent variable has always to be sorted in ascending order. There should be no blank lines and no double entries in the table.

## 3 Creating SDT files from the HITRAN database – a practical example

How to obtain spectral data from HITRAN and generate a spectral data table shall be explained with a practical example. We consider that a gas sensor for the detection of carbon dioxide in air shall be designed. The detection-method shall be TDLS. The sensor shall be isotopologue-sensitive and shall cover the three main natural isotopologues  $^{12}\text{C}^{16}\text{O}_2$ ,  $^{13}\text{C}^{16}\text{O}_2$  and  $^{16}\text{O}^{12}\text{C}^{18}\text{O}$ . After creating the necessary SDT files, the SRM should be used to indicate the best suitable center wavelength for the laser diode.

### 3.1 Using the HITRAN Application Programming Interface (HAPI)

With the HITRAN Application Programming Interface (HAPI), the transition lines of molecules can be downloaded from the HITRAN database and processed. The current version of HAPI and the user manual are accessible at [2] and [3]. The usage of HAPI requires a Python environment on your computer. The version should be not less than Python 2.6. Additionally, HAPI requires Numpy. If visualization of the processed spectra is desired, also the Matplotlib needs to be

installed. For details, refer to the HAPI manual and section 5 of this note.

### 3.2 Defining the spectral ranges of interest

It would take very long processing times (several hours) for the SRM to go through the entire infrared band. Therefore, it is recommended to select specific ranges in which carbon dioxide shows strong energy absorption. An overview of a CO<sub>2</sub> spectrum can be found at [4]. The image is given in figure 1.

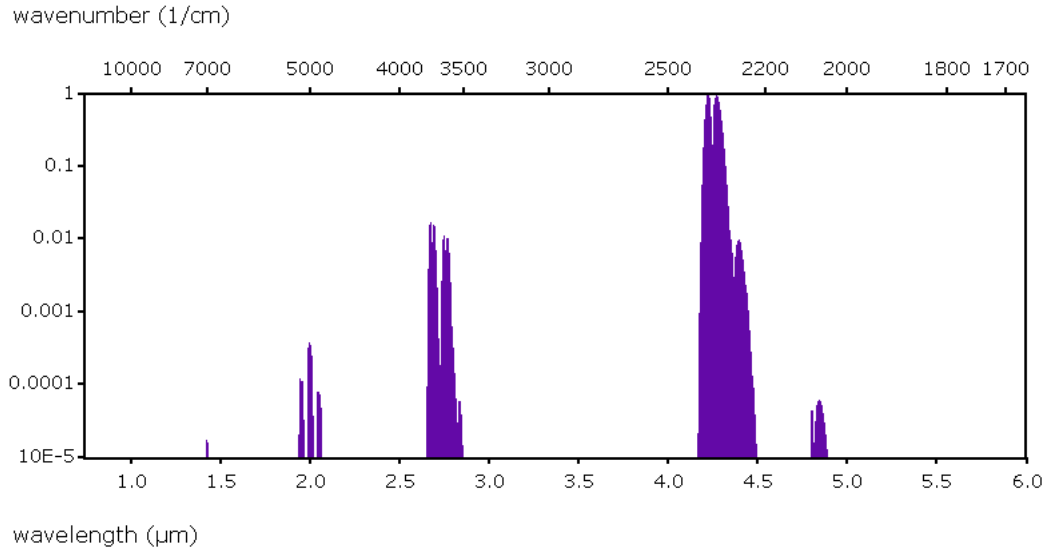


Figure 1: Infrared spectrum of carbon dioxide

A closer look to figure 1 indicates three infrared (IR) regions of interest:

IR reg.	$\lambda$ range	$\nu$ range
g	1850-2200 nm	4545-5406 1/cm
i	2600-2900 nm	3448-3847 1/cm
k	4000-4600 nm	2173-2500 1/cm

The strongest absorption is located in the IR region k. Therefore, detection within IR region k will cause the highest sensitivity for the detector.

### 3.3 Obtaining spectral data from HITRAN

Before data can be downloaded from the HITRAN database, we must know the HITRAN ID of the desired molecule and the identifiers for the isotopologues. They can be found by using the "Data Access, Line-by-Line Search" in HITRAN online [5]. There we will find that the ID for carbon dioxide is 2. If we check this ID in the list, we can proceed to select the isotopologues of CO<sub>2</sub>. In the list, we will find the following isotopologue IDs:

## About Spectral Data Table (SDT) Files

Molecule ID	Isotopologue ID	Formula
2	1	$^{12}\text{C}^{16}\text{O}_2$
2	2	$^{13}\text{C}^{16}\text{O}_2$
2	3	$^{16}\text{O}^{12}\text{C}^{18}\text{O}$

Now we are prepared for downloading spectral data from HITRAN. Start the Python interpreter and enter the following command:

```
>>> import hapi as hp
```

HAPI responds something like:

HAPI version: 1.2.2.2

To get the most up-to-date version please check <http://hitran.org/hapi>

ATTENTION: Python versions of partition sums from TIPS-2021 are now available in HAPI code

MIT license: Copyright 2021 HITRAN team, see more at <http://hitran.org>.

If you use HAPI in your research or software development, please cite it using the following reference:

R.V. Kochanov, I.E. Gordon, L.S. Rothman, P. Wcislo, C. Hill, J.S. Wilzewski, HITRAN Application Programming Interface (HAPI): A comprehensive approach to working with spectroscopic data, J. Quant. Spectrosc. Radiat. Transfer 177, 15–30 (2016)

DOI: 10.1016/j.jqsrt.2016.03.005

ATTENTION: This is the core version of the HITRAN Application Programming Interface. For more efficient implementation of the absorption coefficient routine, as well as for new profiles, parameters and other functional, please consider using HAPI2 extension library.

HAPI2 package is available at <http://github.com/hitranonline/hapi2>

Now HAPI is accessible through the hp object. At first, the download-folder for the data files has to be declared. We will name it "HitranFiles":

```
>>> hp.db_begin('HitranFiles')
```

HAPI responds:

Using HitranFiles

The first data table to download shall be the isotopologue  $^{12}\text{C}^{16}\text{O}_2$  in IR region k. We will name the file '12c16o2\_irk', declare the molecule (2) and isotopologue (1) ID and the lower (2173) and upper (2500) boundaries of the wavenumber range, because HAPI deals with wavenumbers instead of wavelengths:

```
>>> hp.fetch('12c16o2_irk',2,1,2173,2500)
```

HAPI responds:

Data is fetched from <http://hitran.org>

BEGIN DOWNLOAD: 12c16o2\_irk

65536 bytes written to HitranFiles/  
12c16o2\_irk.data

.

.

.

Header written to HitranFiles/

12c16o2\_irk.header

END DOWNLOAD

Lines parsed: 18621

PROCESSED

After this procedure, the two files "12c16o2\_irk.data" and "12c16o2\_irk.header" can be found in the folder "HitranFiles" (refer to subsection 3.5 to check the file path). Now we can deal with them. Let us process the data to get a spectrum of  $^{12}\text{C}^{16}\text{O}_2$ , diluted in air at default temperature and pressure and use a Lorentz profile for line broadening. The resulting data table shall be named "12c16o2\_irk.sdt" and stored in our folder "HitranFiles":

```
>>> nu,coef=hp.absorptionCoefficient_Lorentz(SourceTables='12c16o2_irk',Diluent={'air':1.0},File='./HitranFiles/12c16o2_irk.sdt')
```

### 3.4 Verifying the processed spectra

After successful data processing, a list with wavenumbers (nu) and absorption coefficients (coef) is accessible to the Python interpreter and also stored in the file "12c16o2\_irk.sdt". To verify the list, we can visualize the calculated function, provided that Matplotlib is installed together with Python:

```
>>> import matplotlib.pyplot as plt
>>> plt.plot(nu,coef)
```

Pyplot responds:

```
[<matplotlib.lines.Line2D object at
0x000001358C55F430>]
```

Now let show the plotted function with:

```
>>> plt.show()
```

Figure 2 shows the result, which is the absorption spectrum of  $^{12}\text{C}^{16}\text{O}_2$  in the infrared region 2173/cm to 2500/cm.

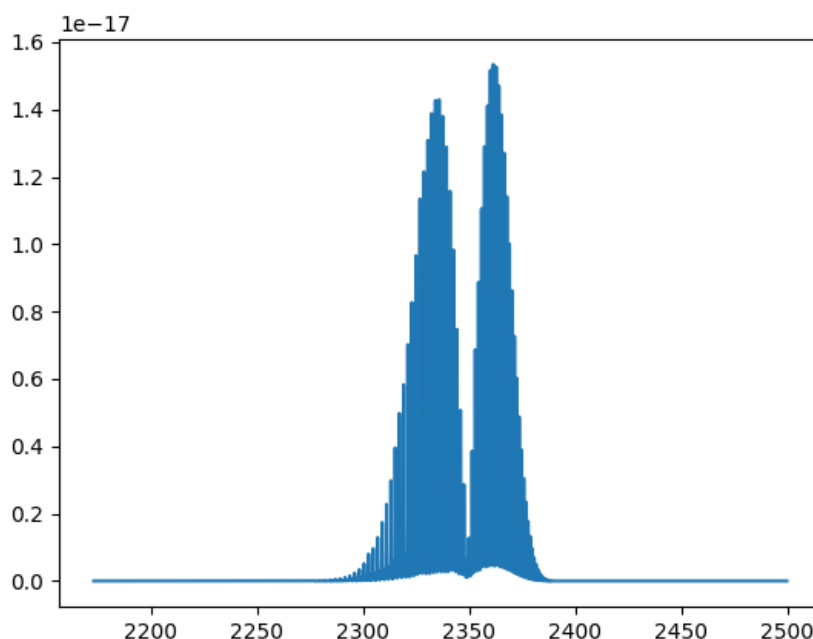

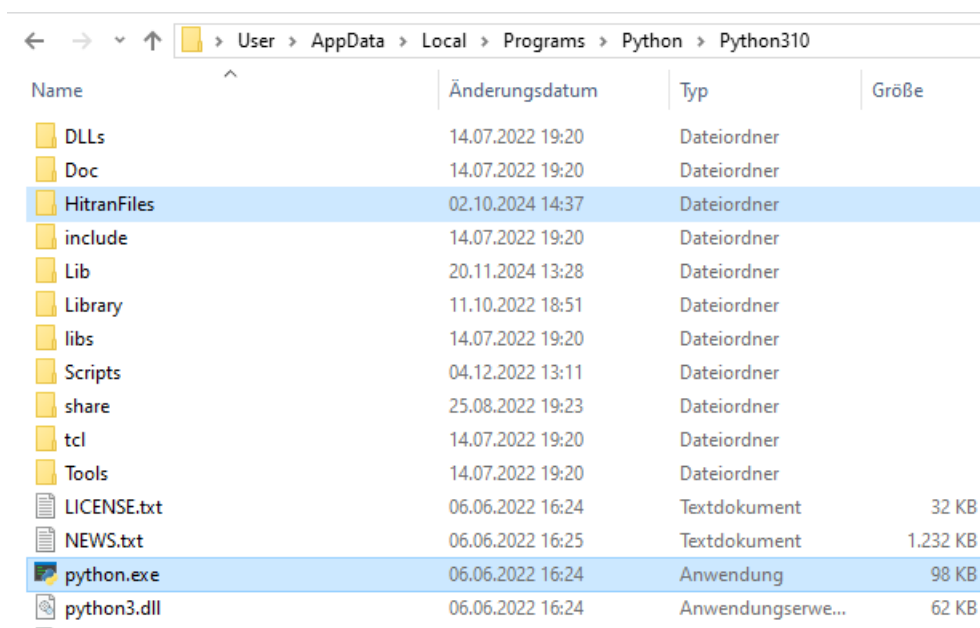


Figure 2: Matplotlib plot of the source table '12c16o2\_irk'

### 3.5 Where to find the SDT file?

Remember that the SDT file resides in a folder called **HitranFiles**. This folder can be found in the directory, where the file **python.exe** is located. You can search for both – the folder **HitranFiles** or the file **python.exe** – with the Windows Explorer or by opening a console and enter the command: **where python**

 . The answer is illustrated in figure 10. If you try to find the file **python.exe** in the first declared folder in this example, you will find out, that it is a fake. This **python.exe** file contains 0 bytes. If it is executed, it leads to an application finder from which Python can be downloaded. The second path contains an executable Python version, like shown in figure 3



Name	Änderungsdatum	Typ	Größe
DLLs	14.07.2022 19:20	Dateiordner	
Doc	14.07.2022 19:20	Dateiordner	
HitranFiles	02.10.2024 14:37	Dateiordner	
include	14.07.2022 19:20	Dateiordner	
Lib	20.11.2024 13:28	Dateiordner	
Library	11.10.2022 18:51	Dateiordner	
libs	14.07.2022 19:20	Dateiordner	
Scripts	04.12.2022 13:11	Dateiordner	
share	25.08.2022 19:23	Dateiordner	
tcl	14.07.2022 19:20	Dateiordner	
Tools	14.07.2022 19:20	Dateiordner	
LICENSE.txt	06.06.2022 16:24	Textdokument	32 KB
NEWS.txt	06.06.2022 16:25	Textdokument	1.232 KB
python.exe	06.06.2022 16:24	Anwendung	98 KB
python3.dll	06.06.2022 16:24	Anwendungserwe...	62 KB

Figure 3: Exploring Python and HitranFiles

## 3.6 Editing the obtained SDT file

The procedure described above creates only the list with data pairs. If we open the file 12c16o2\_irk.sdt with a text editor, we will only see this data pairs. The file has to be completed manually by adding the keywords and their end statements. We have to give the spectrum a meaningful name and have to declare the units:

```
@substance:
Carb. Dioxi. 12C 1602, IR-range: k
@end
@xunit:
1/cm
@end
@yunit:
?
@end
@data:
2173.054432000000 7.071918e-26
2173.064432000000 7.750664e-26
2173.074432000000 8.520922e-26
2173.084432000000 9.395100e-26
2173.094432000000 1.038477e-25
.
.
```

```
.
2499.824432000000 3.995657e-28
2499.834432000000 4.498222e-28
2499.844432000000 4.968758e-28
2499.854432000000 5.336310e-28
2499.864432000000 5.527451e-28
@end
```

After editing and storing the SDT file, it can be used as input for the SRM. Because we want to compare this spectrum with the spectra of  $^{13}\text{C}^{16}\text{O}_2$  and  $^{16}\text{O}^{12}\text{C}^{18}\text{O}$  in the same IR region k, the procedure described above has to be repeated accordingly using HAPI – for example:

```
>>> hp.fetch ('13c16o2_irk',2,2,2173,
2500)

and

>>> hp.fetch ('16o12c18o_irk',2,3,2173,
2500)
```

## 4 Using SDT files with the SRM

Following up the example, the SDT files obtained with HAPI and edited manually, shall be used as

## About Spectral Data Table (SDT) Files

input for the Spectral Range Manager. Therefore, it is necessary to copy or relocate the SDT files to the "spectra" folder of the SRM. After starting the SRM, they should appear in the "Available

spectra" list of the program window. After highlighting the files with a mouse click, they can be selected for processing like illustrated in figure 4.

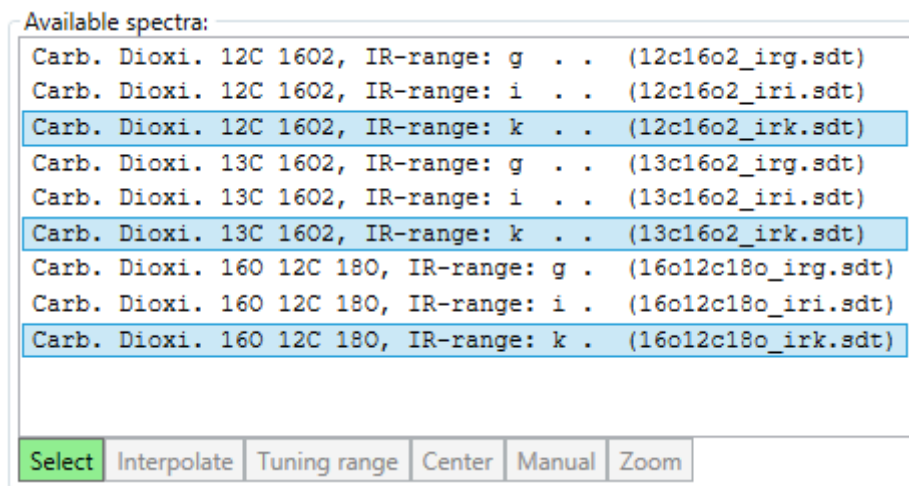


Figure 4: Selection of the input files in the SRM

### 4.1 Automatic conversion of SDT files

After the "Select" button is clicked, the SRM analyzes the SRM files and notes, that the data tables of the three selected spectra have the x unit 1/cm and the y unit is unknown. Therefore, the x values of the data tables will be converted from wavenumbers to wavelengths and rearranged in ascending order. Subsequently, the corresponding y values will be normalized. The SDT files in the "spectra" folder will be modified automatically and will appear in an editor like the following:

```
@substance:
Carb. Dioxi. 12C 1602, IR-range: k
@end
@xunit:
nm
@end
@yunit:
1
@end
@data:
4000.216921 0
4000.232922 0
```

```
4000.248924 0
4000.264926 0
4000.280929 0
.
.
.
4234.624571 0.9561
4234.642503 0.9951
4234.660435 1
4234.678368 0.9698
4234.6963 0.9105
.
.
.
4601.732834 0
4601.75401 0
4601.775187 0
4601.796363 0
4601.81754 0
@end
```

### 4.2 Proceeding data processing with the SRM

After interpolating with a spectral resolution of 0.1 nm, the SRM will show the sum spectrum of

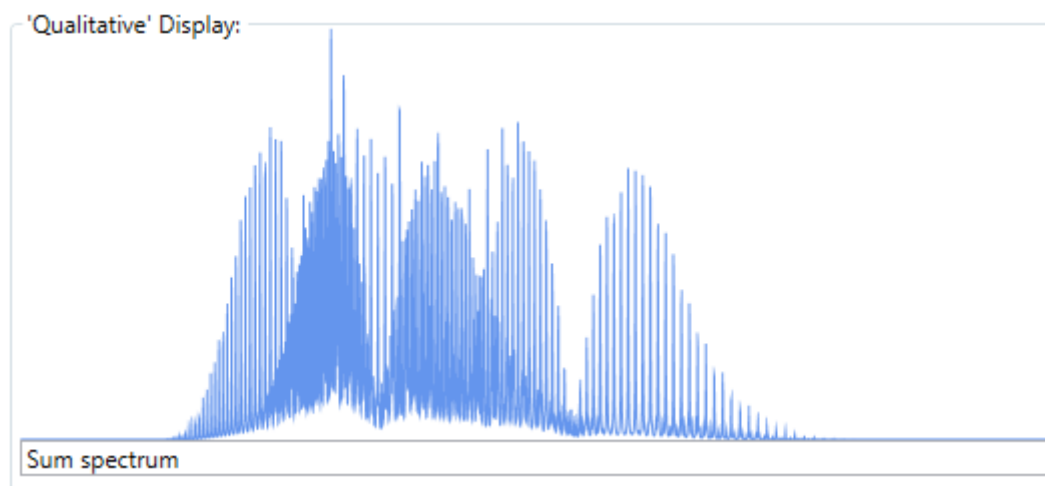


Figure 5: Sum spectrum, plotted by the SRM

all three spectra in the "Qualitative Display", like illustrated in figure 5.

After setting the tuning range to 10.0 nm as a typical value and clicking the "Center" button, the SRM calculates a center wavelength of 4275.9 nm for the laser diode. However, the result is not satisfactory. As we can see in figure 6, which shows the zoomed plot for the isotopologue, there is no absorption for  $^{13}\text{C}^{16}\text{O}_2$  within the laser tuning range. This reveals, that a tuning range of 10.0 nm would be too small to cover all three isotopologues. The SRM provides the following results within the tuning range:

Isotopologue	Wavelength	Rel. absorption
$^{12}\text{C}^{16}\text{O}_2$	4277.8 nm	0.8837
$^{13}\text{C}^{16}\text{O}_2$	4270.9 nm	0
$^{16}\text{O}^{12}\text{C}^{18}\text{O}$	4271.4 nm	0.8962

If the laser tuning range is increased to 200 nm, the SRM calculates a center wavelength of 4347.9 nm and the following relative absorption maxima:

Isotopologue	Wavelength	Rel. absorption
$^{12}\text{C}^{16}\text{O}_2$	4284.2 nm	0.9344
$^{13}\text{C}^{16}\text{O}_2$	4355.7 nm	0.9984
$^{16}\text{O}^{12}\text{C}^{18}\text{O}$	4264.4 nm	0.9973

### 4.3 Further studies

The example above reveals that the typical laser tuning range of 10 nm is insufficient to build a isotopologue sensitive carbon dioxide sensor with a single laser diode. But there are two other IR ranges for  $\text{CO}_2$ , promising a sufficient absorption. If the procedure described above is repeated for the IR regions g and i (refer to figure 4), the SRM will give the following results for tuning range 10 nm and region g:

Isotopologue	Wavelength	Rel. absorption
$^{12}\text{C}^{16}\text{O}_2$	2046.9 nm	0.0189
$^{13}\text{C}^{16}\text{O}_2$	2039.9 nm	0.96
$^{16}\text{O}^{12}\text{C}^{18}\text{O}$	2045.1 nm	0.8852

For Ir region i and a laser the tuning range of 10 nm, the SRM reveals:

Isotopologue	Wavelength	Rel. absorption
$^{12}\text{C}^{16}\text{O}_2$	2708.2 nm	0.592
$^{13}\text{C}^{16}\text{O}_2$	2706.9 nm	0
$^{16}\text{O}^{12}\text{C}^{18}\text{O}$	2711.4 nm	0.9171

## 5 How to get HAPI working with Windows 10

This section provides a tutorial for the installation of Python and the dependencies needed to



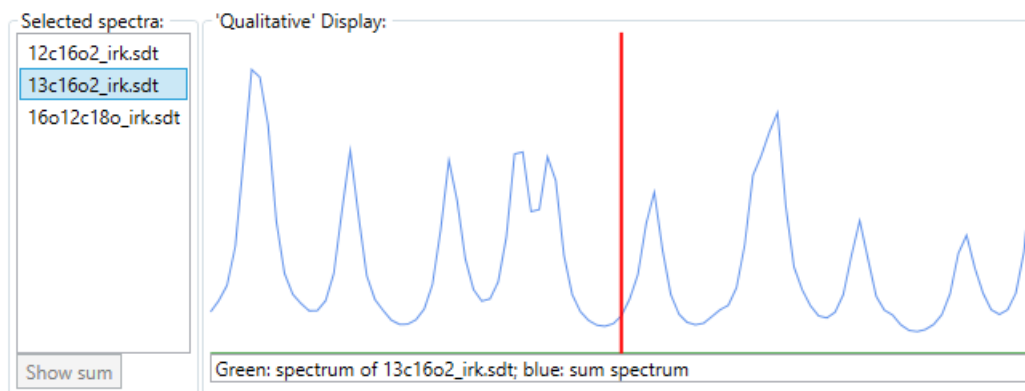



Figure 6: Zoomed spectrum of  $^{13}\text{C}^{16}\text{O}_2$  with 10 nm laser tuning range

get HAPI working on a Windows 10 operating system.

## 5.1 Checking, if Python is already available on the system

There are different ways to detect if Python is already installed. For instance, it may be visible in the application finder, like shown in figure 7. If Python is not available there, you can try to find it anyway by using a console command, to find out if an executable python file exists. Open a console, e.g. through the app finder like shown in figure 8 or by entering `cmd` in the search field, like in figure 9. Open the console and simply ask for Python by typing: `where python` , like illustrated in figure 10. If you don't get any file path information, it is very unlikely that Python is already available on your system. In this case Python should be installed.

## 5.2 Installing Python


To download the latest version of Python for Windows, visit the Python website [6], download the installer, e. g. the executable file `python-3.13.0-amd64.exe` (depending on your

operating system version) and execute it. After following the installation instructions, you should have Python installed on your system. Check the availability by returning to the instructions of subsection 5.1 in this note.

## 5.3 Checking if numpy is installed with Python

Numpy is required to use HAPI, as mentioned in subsection 3.1. Numpy provides Python scripts for the calculation of multi-dimensional arrays and matrices and a collection of high-level mathematical functions. Maybe, that numpy was already installed together with Python on your system. There are at least two different ways to check, if numpy is already installed.

## 5.4 Checking for numpy with PIP


The easiest way to look for numpy is the usage of the Python package installer PIP. Open a console and enter the command: `pip show numpy` , like shown in figure 11. If PIP is not installed, you may proceed with subsection 5.7 and get back here afterwards, or you skip to subsection 5.5

## 5.5 Checking for numpy with Python

To check if numpy is available and accessible, you can launch your Python interpreter. Provided Python is successfully installed on your

Windows 10 system, there are several ways to launch it. You can use the application finder, like already described in subsection 5.1, hover your mouse to **Python 3.10 (64-bit)** and left-click it. A fundamental method would be, to use the file explorer, move to the direc-


tory Python310 and start the executable file `python.exe` like shown in figure 12. Python could also be started via the `cmd` method by

opening a console and entering the command `py`  or by starting it through the Windows Power Shell, like shown in figure 13.

If the Python interpreter runs and waits for input (indicated by the `>>>` prompt), try to import numpy and ask for the numpy version as illustrated in figure 14.

If numpy is installed and accessible to Python, the version number will be displayed. In this case, you can proceed with subsection 5.11. Otherwise numpy is probably not installed. In this case, numpy should be installed. There are different ways to manage it. We will use the package installer for Python (PIP) to do so.


## 5.6 Checking if PIP is installed on your sytem

The package installer for Python (PIP) should automatically be installed with Python. However, depending on how Python was installed, PIP may not be available on your system. Therefore it is recommended to check if it is executable. Open a console, like described in subsection 5.1 and type `pip help` . If PIP is present, an overview of all PIP commands should appear in your console window, like illustrated in figure 15.

## 5.7 Installing PIP - first method


If PIP seems to be unavailable, it should be installed on your system. This can be done with a Python script called `get-pip.py`. Open a console and run the curl-command, like illustrated in figure 16.

After execution, a file named `get-pip.py` should be stored somewhere in your system, probably in your user directory. Open a console or the Windows Power Shell. Change to the directory, in which `get-pip.py` is stored and enter the command:


`py get-pip.py` 

## 5.8 Installing PIP - second method


Open a console or the Windows Power Shell. Enter the command:

`py -m ensurepip --upgrade` 


## 5.9 Using PIP to install numpy

If PIP is available on your system, it can be used to install the numpy package. Open a console or the Windows Power Shell and enter the command: `pip install numpy` 


## 5.10 Checking for matplotlib with PIP

Open a console and enter the command: `pip show matplotlib`  like shown in figure 17.

## 5.11 Using PIP to install matplotlib

If PIP is available on your system, it can be used to install the numpy package. Open a console or the Windows Power Shell and enter the command: `pip install matplotlib` 

## 5.12 Installing HAPI with PIP

Provided, PIP is available on your system, open a console and enter the command: `pip install hitran-api` 

## 5.13 Installing HAPI manually

Visit <https://hitran.org> and hover your mouse to "Data Acces" and "HAPI", like illustrated in figure 18. Clicking will lead you to the HAPI website. Download the current version of HAPI and the HAPI manual. If the download is complete, the received file `hapi.py` must be moved to a directory, which

## About Spectral Data Table (SDT) Files

is accessible with Python. All accessible paths can be checked with the following procedure: Open a console or the Windows Power Shell and insert the command: `py` . After the Python interpreter is ready to accept commands, enter: `import sys` . Then enter the command:  
`for p in sys.path: print(p)`

The Python Interpreter will show three dots in a new line. After pressing the return-key, a list of all accessible paths appears, like illustrated in figure 19.

In the example, `hapi.py` is moved to the path:

```
User\AppData\Local\Programs\Python\
Python310\lib
```

## 6 Creating SDT files with Google Colab

If you don't want to install Python on your computer system, there is an option to use Python in a web-based application. This only requires a Google-account and a web browser, which means, that you could even use a smartphone to create your SDT files.

### 6.1 Using Google Colab

Visit the website:

<https://colab.research.google.com>

This will lead to the homepage shown in figure 20. Log in to your Google-account and get a bit familiar with the Colab-environment. Then create a new jupyter-notebook. In this example, the jupyter-notebook is named:

`Generate_sdt_files.ipynb`

### 6.2 Installing HAPI

First you have to install HAPI, using the pip-command described in subsection 5.12 and illustrated in figure 21. Because the notebook assumes Python code in the code-line, the PIP-Command must be declared as non-Python by a prefixed exclamation mark (which means "non" python code). Just enter

```
!pip install hitran-api
```

in the code-line of your jupyter-notebook and execute the code.

### 6.3 Importing and initializing HAPI

In the next two steps, HAPI must be imported and the database must be initialized. Enter a new code-line in your jupyter-notebook and type like described in subsection 3.3:

```
import hapi as hp
```

The result is shown in figure 22

Initialize HAPI by creating a database for your spectral data files, like described above in section 3.3. In this example, the database is called again:

```
HitranFiles
```

The jupyter-notebook operation is shown in figure 23.

### 6.4 Fetching spectral data from HITRAN

Like described in subsection 3.3, a data table can be downloaded to the declared `HitranFiles`-folder, like shown in figure 24. This example fetches the isotopologue  $^{12}\text{C}^{16}\text{O}_2$  in IR region k (2173/cm to 2500/cm).

### 6.5 Calculating a spectrum

According to our example, the next Python-code-line to enter in the jupyter-notebook is

```
nu,coef=hp.absorptionCoefficient_Lorentz(SourceTables='12c16o2_irk',Diluent={'air':1.0},File='./HitranFiles/12c16o2_irk.sdt')
```

like already described above in subsection 3.3 and illustrated here in figure 25

### 6.6 Downloading the SDT file from Google Colab

The SDT file named `12c16o2_irk.sdt` is now available in your workspace. It can be found by opening your `HitranFiles`-folder on the left side of your notebook-window, like shown in figure 26. Simply right-click the file and choose "Download" from the pop-up table. Now you can edit the obtained file, like described above in subsection 3.6 and use it afterwards with the spectral range manager.

## 6.7 Verifying the spectrum with COLAB

As already mentioned in subsection 3.4, you might also use Google Colab to visualize your obtained SDT-file. Just execute the next two command

lines:

```
import matplotlib.pyplot as plt  
plt.plot(nu,coef)
```

The result is visualized in figure 27.

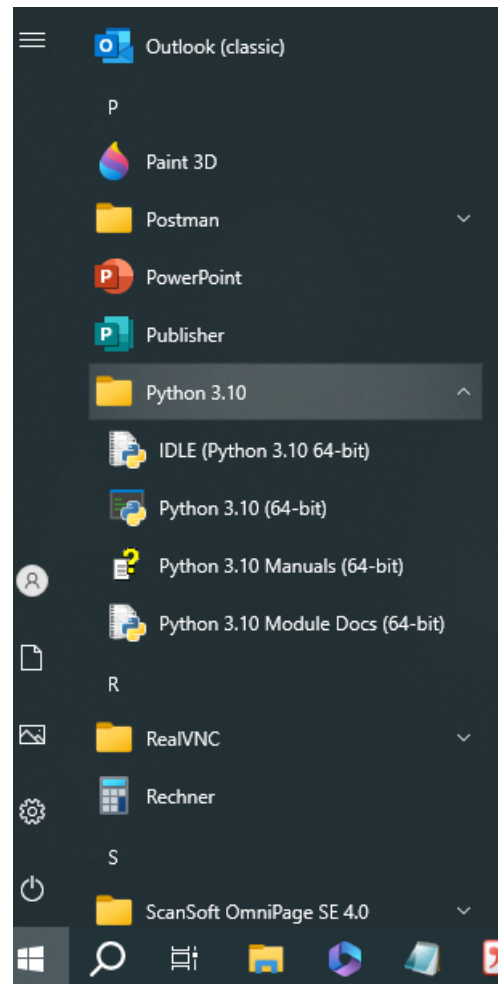


Figure 7: Python in the application finder

## About Spectral Data Table (SDT) Files

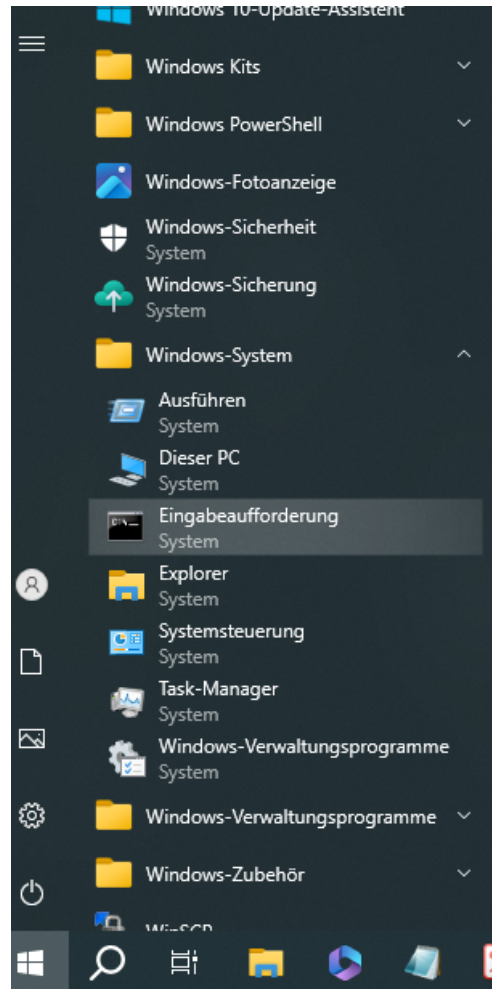


Figure 8: Console command in the application finder

## About Spectral Data Table (SDT) Files

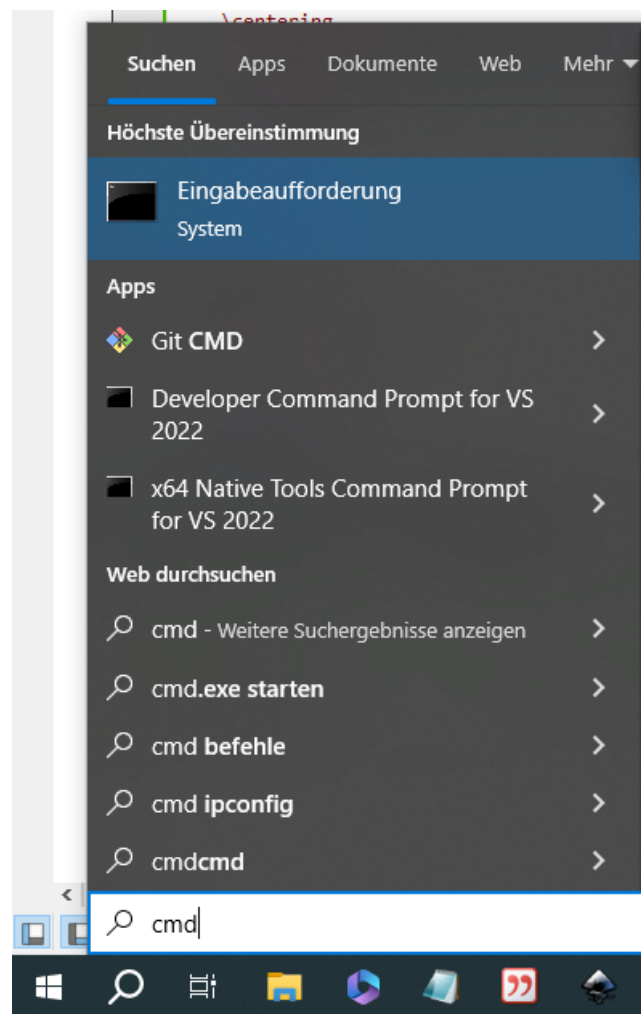


Figure 9: Console command in the search field

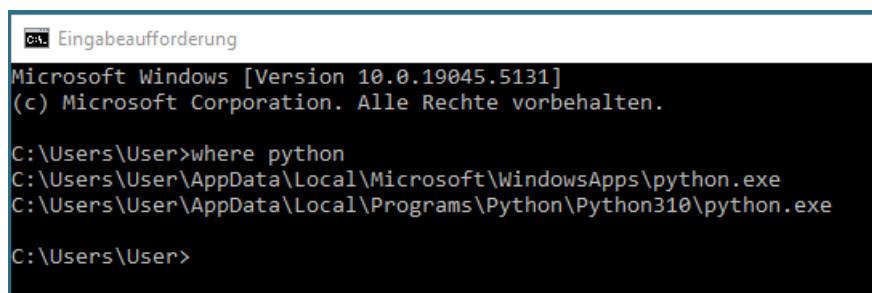
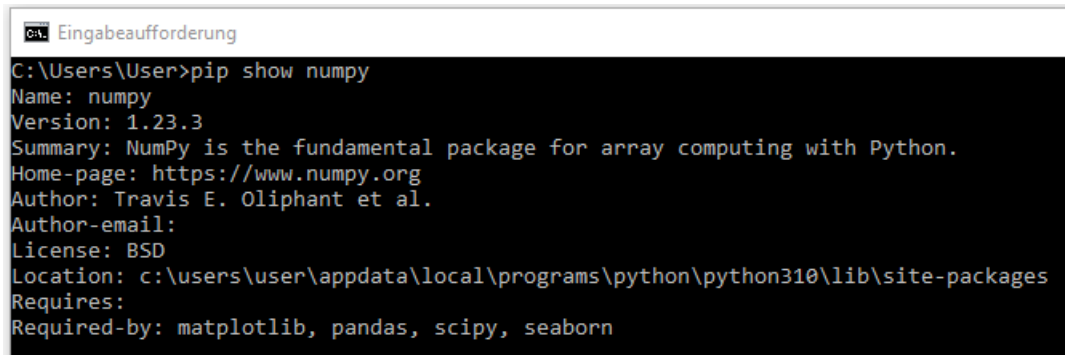


Figure 10: Asking for Python, using the console



## About Spectral Data Table (SDT) Files



```
C:\Users\User>pip show numpy
Name: numpy
Version: 1.23.3
Summary: NumPy is the fundamental package for array computing with Python.
Home-page: https://www.numpy.org
Author: Travis E. Oliphant et al.
Author-email:
License: BSD
Location: c:\users\user\appdata\local\programs\python\python310\lib\site-packages
Requires:
Required-by: matplotlib, pandas, scipy, seaborn
```

Figure 11: Using PIP to check for numpy

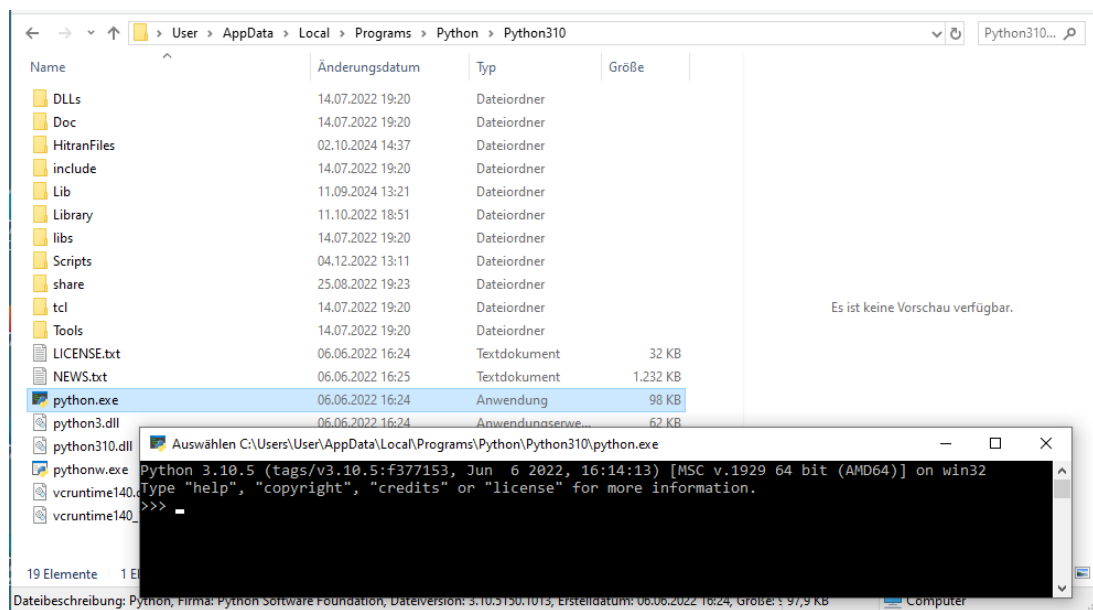
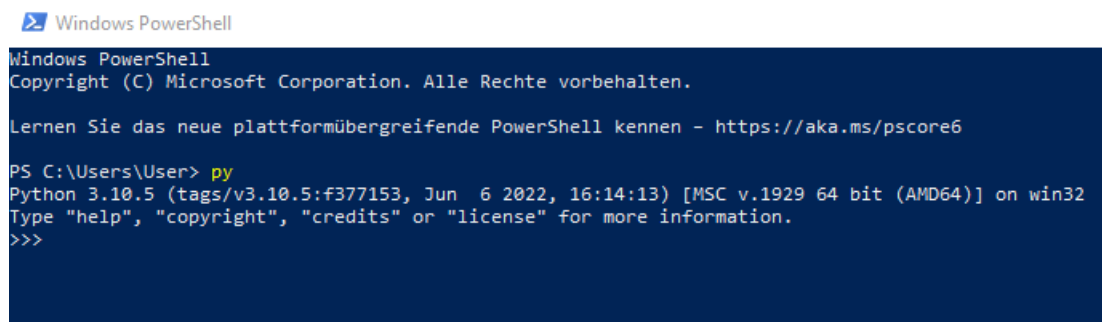


Figure 12: Launching Python by starting the python.exe file



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Alle Rechte vorbehalten.

Lernen Sie das neue plattformübergreifende PowerShell kennen - https://aka.ms/pscore6

PS C:\Users\User> py
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 13: Launching Python through the Windows Power Shell

## About Spectral Data Table (SDT) Files

```
PS C:\Users\User> py
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> print(np.__version__)
1.23.3
>>>
```

Figure 14: Try numpy by asking for it's version number

```
ca Eingabeaufforderung
Microsoft Windows [Version 10.0.19045.5131]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\User>pip help

Usage:
  pip <command> [options]

Commands:
  install           Install packages.
  download          Download packages.
  uninstall         Uninstall packages.
  freeze            Output installed packages in requirements format.
  inspect           Inspect the python environment.
  list              List installed packages.
  show              Show information about installed packages.
  check             Verify installed packages have compatible dependencies.
  config            Manage local and global configuration.
  search            Search PyPI for packages.
  cache             Inspect and manage pip's wheel cache.
  index             Inspect information available from package indexes.
  wheel             Build wheels from your requirements.
  hash              Compute hashes of package archives.
  completion        A helper command used for command completion.
  debug             Show information useful for debugging.
  help             Show help for commands.

General Options:
  -h, --help        Show help.
  --debug           Let unhandled exceptions propagate outside the main subroutine, instead of logging them to stderr.
  --isolated        Run pip in an isolated mode, ignoring environment variables and user configuration.
  --require-virtualenv
                    Allow pip to only run in a virtual environment; exit with an error otherwise.
  --python <python> Run pip with the specified Python interpreter.
  -v, --verbose     Give more output. Option is additive, and can be used up to 3 times.
  -V, --version     Show version and exit.
  -q, --quiet       Give less output. Option is additive, and can be used up to 3 times (corresponding to WARNING, ERROR, and CRITICAL logging levels).
  --log <path>     Path to a verbose appending log.
  --no-input        Disable prompting for input.
  --proxy <proxy>   Specify a proxy in the form scheme://[user:passwd@]proxy.server:port.
  --retries <retries>
                    Maximum number of retries each connection should attempt (default 5 times).
  --timeout <sec>  Set the socket timeout (default 15 seconds).
  --exists-action <action>
                    Default action when a path already exists: (s)witch, (i)gnore, (w)ipe, (b)ackup, (a)bort.
  --trusted-host <hostname>
                    Mark this host or host:port pair as trusted, even though it does not have valid or any HTTPS.
  --cert <path>     Path to PEM-encoded CA certificate bundle. If provided, overrides the default. See 'SSL Certificate Verification' in pip documentation for more information.
  --client-cert <path>
                    Path to SSL client certificate, a single file containing the private key and the certificate in PEM format.
  --cache-dir <dir> Store the cache data in <dir>.
  --no-cache-dir    Disable the cache.
  --disable-pip-version-check
                    Don't periodically check PyPI to determine whether a new version of pip is available for download. Implied with --no-index.
  --no-color        Suppress colored output.
  --no-python-version-warning
                    Silence deprecation warnings for upcoming unsupported Pythons.
  --use-feature <feature>
                    Enable new functionality, that may be backward incompatible.
  --use-deprecated <feature>
                    Enable deprecated functionality, that will be removed in the future.

C:\Users\User>
```

Figure 15: Checking PIP with the help command

## About Spectral Data Table (SDT) Files

```
C:\> Eingabeaufforderung

C:\Users\User>curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total       Spent    Left     Speed
100 2222k  100 2222k    0     0  6079k      0  --:--:-- --:--:-- --:--:-- 6156k
```

Figure 16: Downloading get-pip.py

```
C:\Users\User>pip show matplotlib
Name: matplotlib
Version: 3.5.3
Summary: Python plotting package
Home-page: https://matplotlib.org
Author: John D. Hunter, Michael Droettboom
Author-email: matplotlib-users@python.org
License: PSF
Location: c:\users\user\appdata\local\programs\python\python310\lib\site-packages
Requires: cyclor, fonttools, kiwisolver, numpy, packaging, pillow, pyparsing, python-dateutil
Required-by: seaborn
```

Figure 17: Using PIP to check for matplotlib

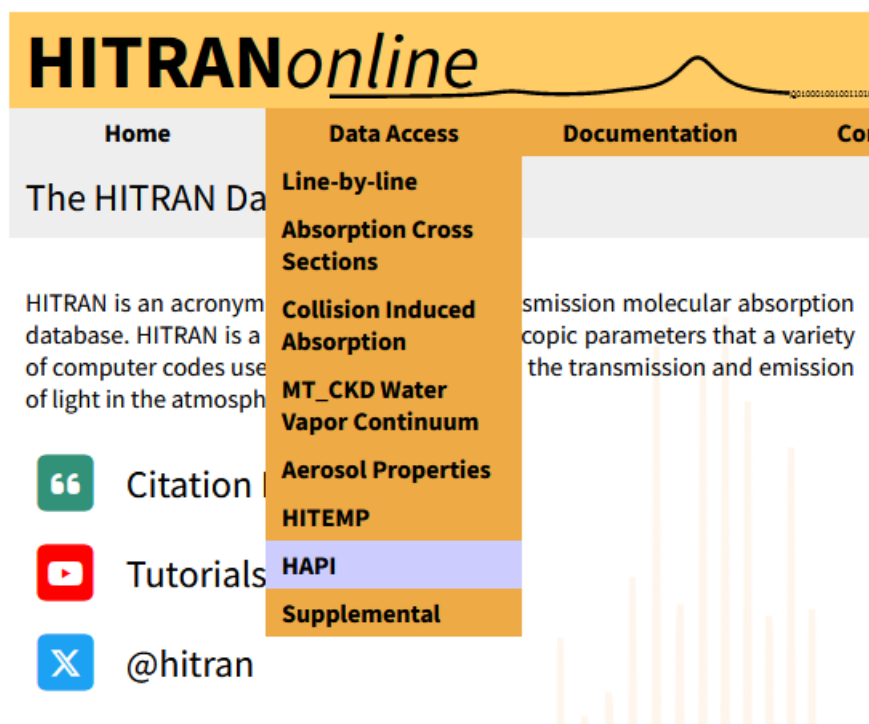
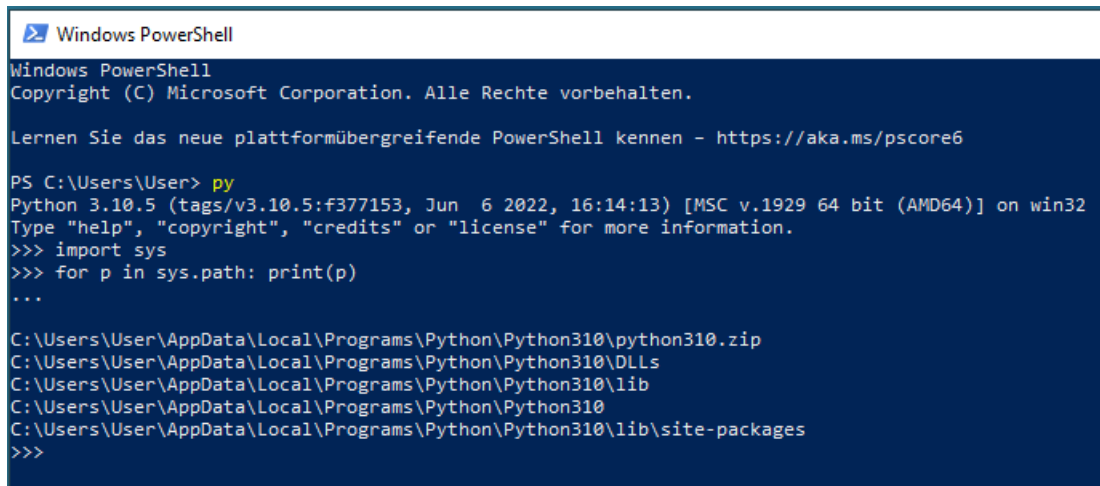


Figure 18: Where to find HAPI on the HITRAN Website

## About Spectral Data Table (SDT) Files



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Alle Rechte vorbehalten.

Lernen Sie das neue plattformübergreifende PowerShell kennen - https://aka.ms/pscore6

PS C:\Users\User> py
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> for p in sys.path: print(p)
...
C:\Users\User\AppData\Local\Programs\Python\Python310\python310.zip
C:\Users\User\AppData\Local\Programs\Python\Python310\DLLs
C:\Users\User\AppData\Local\Programs\Python\Python310\lib
C:\Users\User\AppData\Local\Programs\Python\Python310
C:\Users\User\AppData\Local\Programs\Python\Python310\lib\site-packages
>>>
```

Figure 19: Printing the paths, accessible with Python

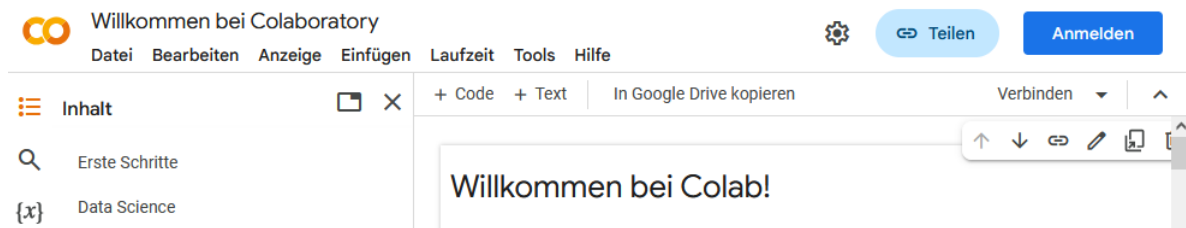


Figure 20: Welcome to Colab!

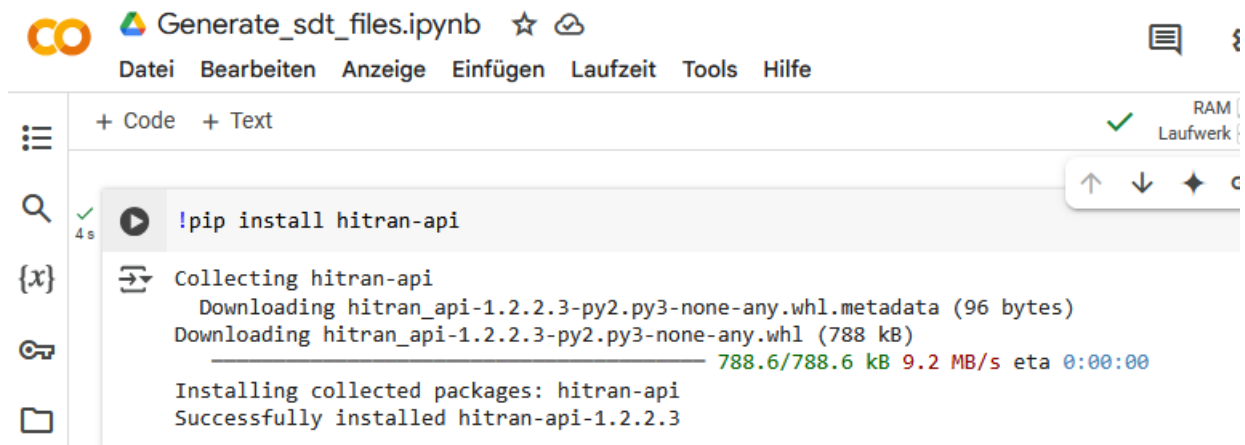
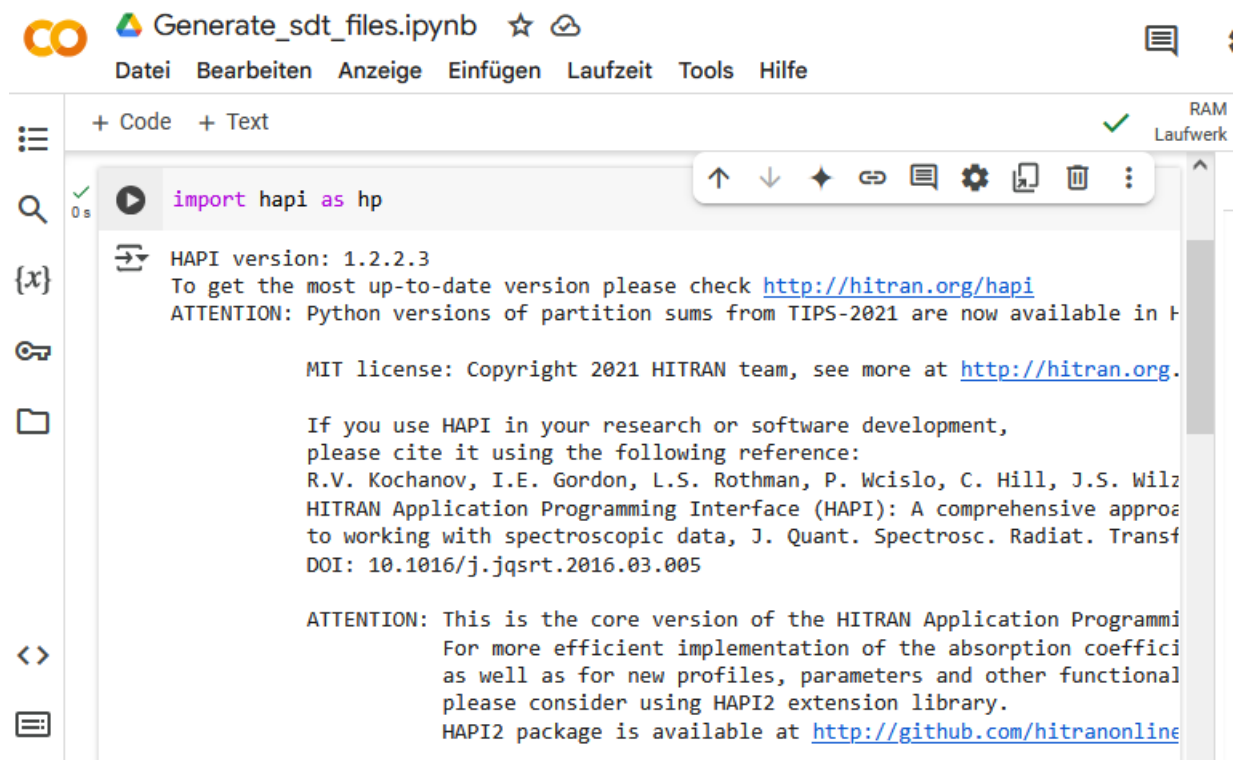


Figure 21: Installing HAPI

## About Spectral Data Table (SDT) Files



```
import hapi as hp
```

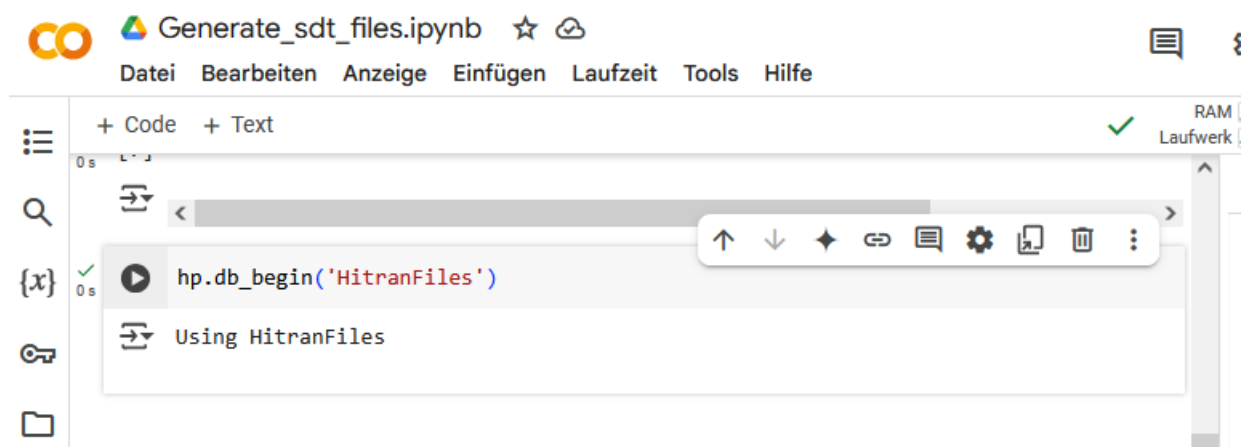
HAPI version: 1.2.2.3  
To get the most up-to-date version please check <http://hitran.org/hapi>  
ATTENTION: Python versions of partition sums from TIPS-2021 are now available in H

MIT license: Copyright 2021 HITRAN team, see more at <http://hitran.org>.

If you use HAPI in your research or software development,  
please cite it using the following reference:  
R.V. Kochanov, I.E. Gordon, L.S. Rothman, P. Wcislo, C. Hill, J.S. Wilz  
HITRAN Application Programming Interface (HAPI): A comprehensive approach  
to working with spectroscopic data, J. Quant. Spectrosc. Radiat. Transf.  
DOI: 10.1016/j.jqsrt.2016.03.005

ATTENTION: This is the core version of the HITRAN Application Programming  
For more efficient implementation of the absorption coefficient  
as well as for new profiles, parameters and other functional  
please consider using HAPI2 extension library.  
HAPI2 package is available at <http://github.com/hitranonline>

Figure 22: Importing HAPI

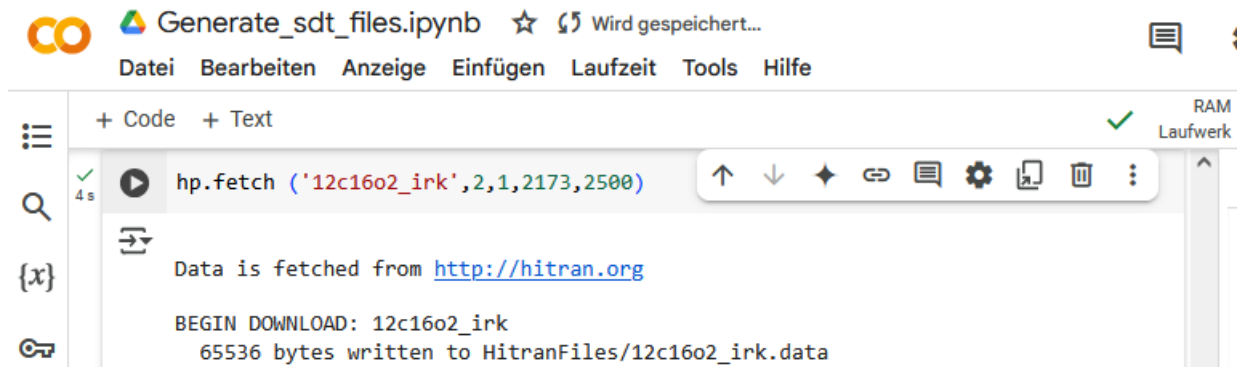


```
hp.db_begin('HitranFiles')
```

Using HitranFiles

Figure 23: Initializing HAPI database

## About Spectral Data Table (SDT) Files



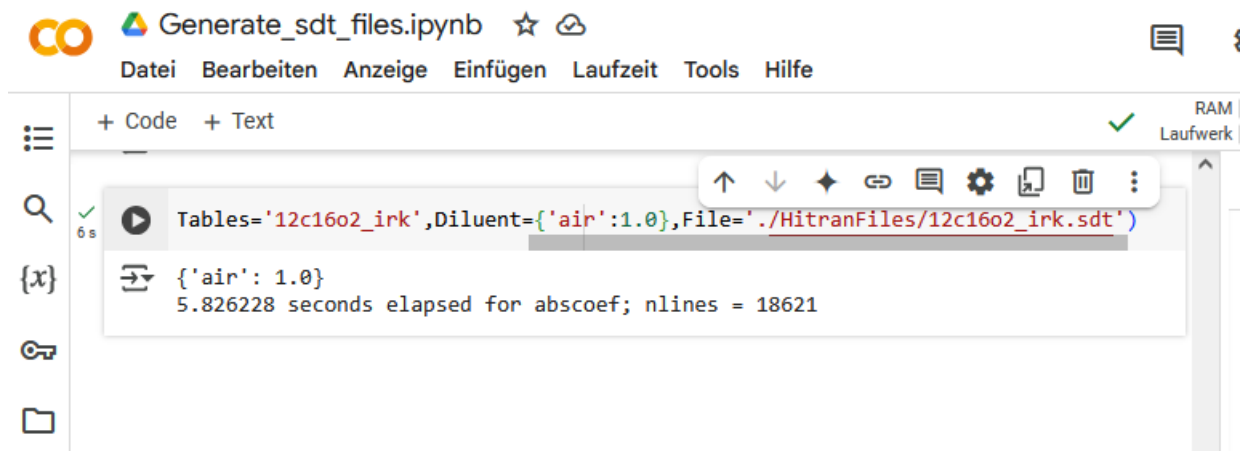
The Jupyter Notebook interface displays the first code cell of 'Generate\_sdt\_files.ipynb'. The code cell contains the following text:

```
hp.fetch ('12c16o2_irk',2,1,2173,2500)
```

The output of the code cell is displayed below the code:

```
Data is fetched from http://hitran.org  
BEGIN DOWNLOAD: 12c16o2_irk  
65536 bytes written to HitranFiles/12c16o2_irk.data
```

Figure 24: Fetching spectral data from HITRAN



The Jupyter Notebook interface displays the second code cell of 'Generate\_sdt\_files.ipynb'. The code cell contains the following text:

```
Tables='12c16o2_irk',Diluent={'air':1.0},File='./HitranFiles/12c16o2_irk.sdt')
```

The output of the code cell is displayed below the code:

```
{'air': 1.0}  
5.826228 seconds elapsed for abscoef; nlines = 18621
```

Figure 25: Calculating a spectrum

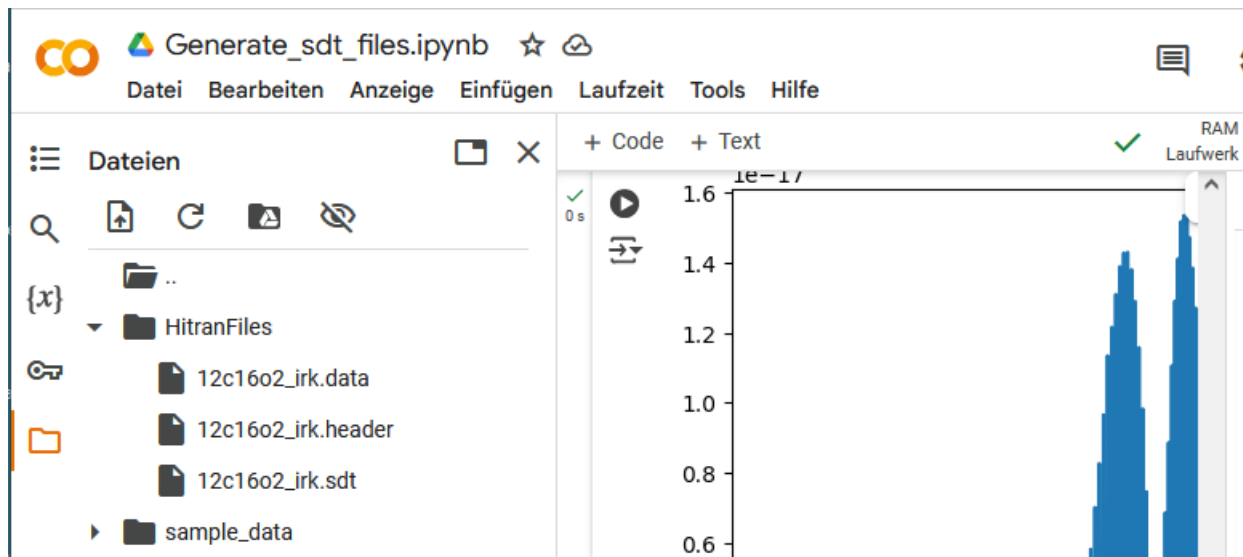


Figure 26: Downloading the SDT file

## About Spectral Data Table (SDT) Files

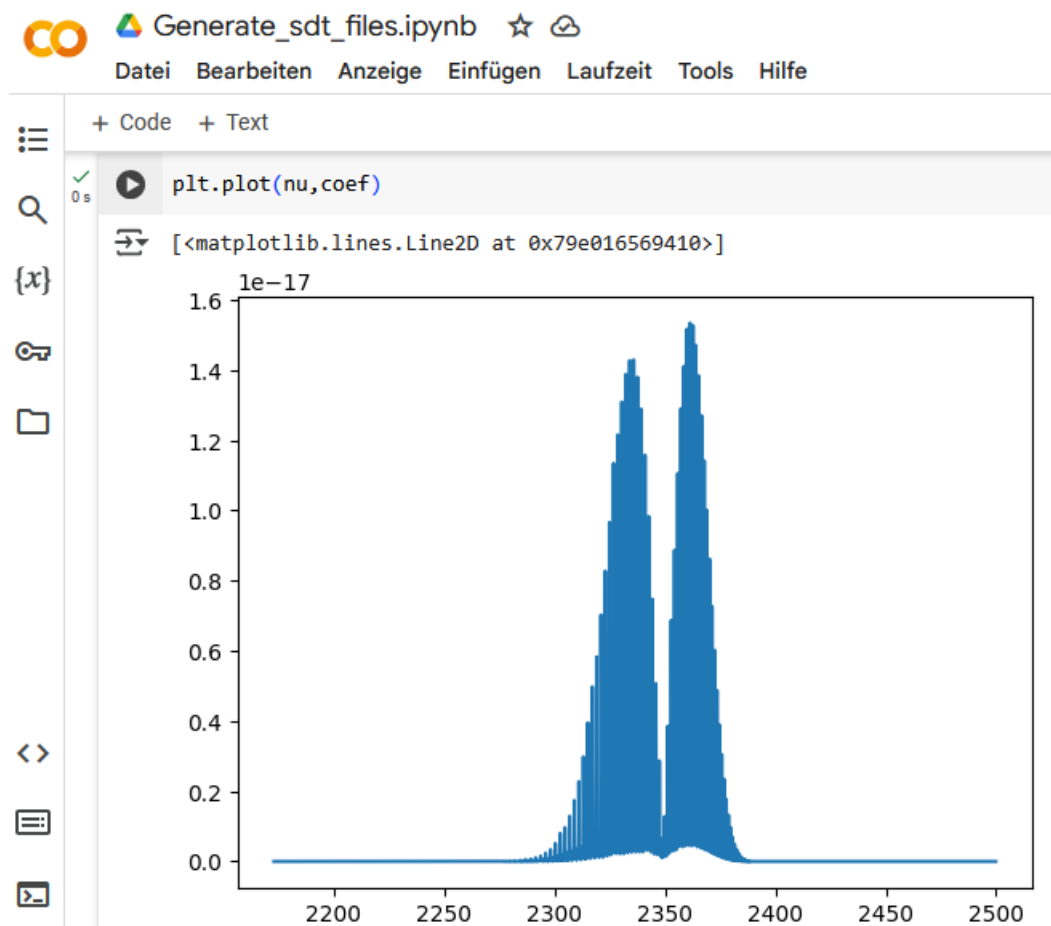


Figure 27: Visualizing the SDT file

## References

- [1] Marc-Simon Bahr, Bernd Baumann, and Marcus Wolff. “Determining the most suitable spectral range for TDLS – a quantitative approach”. In: *Journal of Quantitative Spectroscopy and Radiative Transfer* (Aug. 2022). DOI: <https://doi.org/10.1016/j.jqsrt.2022.108216>. URL: <https://www.sciencedirect.com/science/article/pii/S0022407322001510>.
- [2] *hapi.py V.1.2.2.2*. URL: <https://hitran.org/static/hapi/hapi.py>.
- [3] *HAPI User Guide*. URL: [https://hitran.org/static/hapi/hapi\\_manual.pdf](https://hitran.org/static/hapi/hapi_manual.pdf).
- [4] *Spectrum of carbon dioxide*. URL: <https://nanoplus.com/applications/applications-by-gas/carbon-dioxide>.
- [5] *Data Access, Line-by-Line Search*. URL: <https://hitran.org/lbl/>.
- [6] *Download the latest version for Windows*. URL: <https://www.python.org/downloads/>.