VaporFit User Guide

Przemysław Pastwa and Piotr Bruździak
Department of Physical Chemistry, Gdańsk University of Technology, Narutowicza 11-12, 80-233
Gdańsk, Poland
piotr.bruzdziak@pg.edu.pl

Contents

1 Introduction

VaporFit is an open-source software tool designed for automated atmospheric correction in FTIR spectroscopy. It employs a **least squares fitting approach** using multiple reference atmospheric/vapor spectra, dynamically adjusting subtraction coefficients to improve accuracy. You are welcome to send us any comments or suggestions.

2 License

This software is distributed under the **GNU GPL v.3.0** license + citation requirement (see LICENSE file). By using this software, you agree to its terms and conditions.

3 Citations

If you use VaporFit in your research, you must cite the following publications:

- Bruzdziak P., Vapor correction of FTIR spectra – A simple automatic least squares approach, *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, 2019, **223**, 1–4, doi:10.1016/j.saa.2019.117373

- Pastwa P., Bruzdziak P., VaporFit: An Open-Source Software for Accurate Atmospheric Correction of FTIR Spectra, *in preparation*

4 Data sets

Three sample data sets are available to download from the project site:

- `test_data_betaine_set.zip` — a set of 29 spectra of betaine aqueous solutions with two atmospheric spectra (2 cm^{-1}, 128 scans, single background measurement),

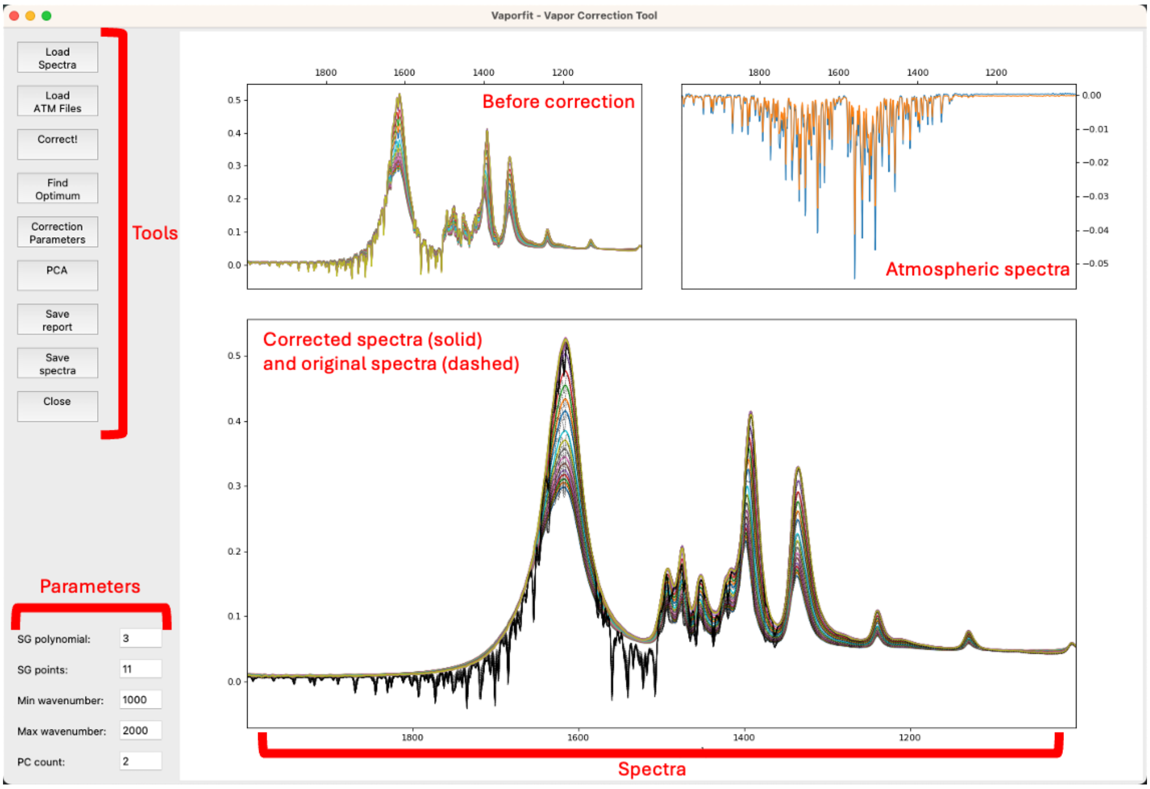
- `test_data_D2O-H2O_set.zip` — a set of 11 spectra of D_2O/H_2O mixtures with three atmospheric spectra (4 cm^{-1}, 128 scans, single background measurement).

- `test_data_urea_evaporation_set.zip` – a set of 20 spectra of a 20 µL urea solution (initial concentration 1 M), freely evaporated on the ATR crystal for 20 minutes, with two atmospheric spectra (resolution: 4 cm^{-1}, 64 scans, single background measurement).

All spectra were measured on a Bruker Invenio-R FTIR spectrometer equipped with a single reflection diamond ATR accessory. The spectrometer was purged with dry nitrogen (Claind Brezza NiGen LCMS 40.1) for 30 minutes before and during the measurement.

5 User Guide

5.1 VaporFit window overview



SG polynomial: 3

SG points: 11

Min wavenumber: 1000

Max wavenumber: 2000

PC count: 2

Figure 1: **Left**: tools and parameters. **Right**: original spectra before correction (upper left panel), atmospheric spectra (upper right panel), correction results (bottom panel; solid lines–corrected spectra, dashed lines–spectra before corrections).

Use the **left mouse button** to select and zoom a fragment of the spectrum (each panel is independent). Use the **right mouse button** to reset the view. Zooming in or out does not affect the wavenumber range for correction indicated in the section of the VaporFit window.

5.2 Load Spectral Files

Upload spectra that require correction.

Supported formats:

- CSV
- DPT

Important: Ensure that the spectral resolution, wavenumber range, number of points, and units are **consistent across all files** to avoid errors.

5.3 Load Atmospheric Spectra

Upload atmospheric spectra.

- The format and spectral parameters must match those of the spectra being corrected.

5.4 Set SG Parameters and Wavenumber Range

Adjust the **desired wavenumber range** and the **Savitzky-Golay (SG) parameters** in the Parameters Section of the main window.

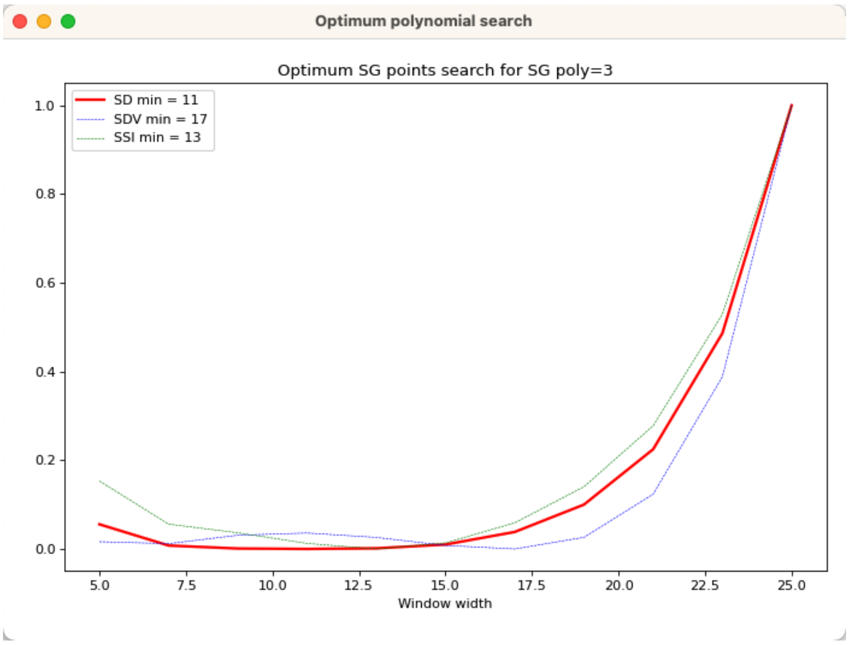
- Default settings are predefined for standard atmospheric corrections for the amide I region of the spectra. Adjust as needed.
- If values exceed the range of loaded spectra, they will be limited to the closest minimum or maximum wavenumber.
- It is **not recommended** to perform correction over the entire spectrum range (typically 4000–400 cm^{-1}). Choosing a narrower region improves result quality.
- Modifying SG parameters may improve results, but incorrect settings can introduce artifacts. A good rule of thumb is to use the lowest possible SG parameter values for optimal correction.

5.5 Run the Correction Algorithm

Press **“Correct!”** to apply the atmospheric correction algorithm.

- The resulting graph will display:
  - **Original spectra** (black dashed lines)
  - **Corrected spectra** (color-coded)
- The correction process **automatically optimizes subtraction coefficients** based on least squares fitting.

5.6 (Optional) Run “Optimum Search”Tool



This tool generates a **graph showing the dependence of four spectral smoothness metrics on the SG window size** (+/-10 points) for a given **SG polynomial order**. Values represent the sum of these metrics for all corrected spectra.

- The minima of these functions (shown in the legend) suggest **potential optimal SG window sizes**.
- However, these values **should not be used blindly**. Verify with **PCA analysis** to ensure meaningful correction.
- The four available smoothness metrics may have different applications depending on the spectral characteristics. For amide I band spectra, the SD parameter works best.
- Depending on computational power, this step may take some time.

5.6.1 Smoothness Functions

1. **Standard Deviation of the Residue After Smoothing (SD)**:

$$\sigma_{\text{residue}} = \text{std}(\text{spectrum} \;\; - \text{smoothed spectrum})$$

where smoothed spectrum is the result of applying the Savitzky-Golay filter to the spectrum Spectrum (with default values of polynomial order 3 and window width 11).

2. **Spectral Smoothness Index (SSI)**:

$$\text{SSI} = \frac{\sum \left(\frac{d}{dx}\text{spectrum}\right)^2}{\sum(\text{spectrum})^2}$$

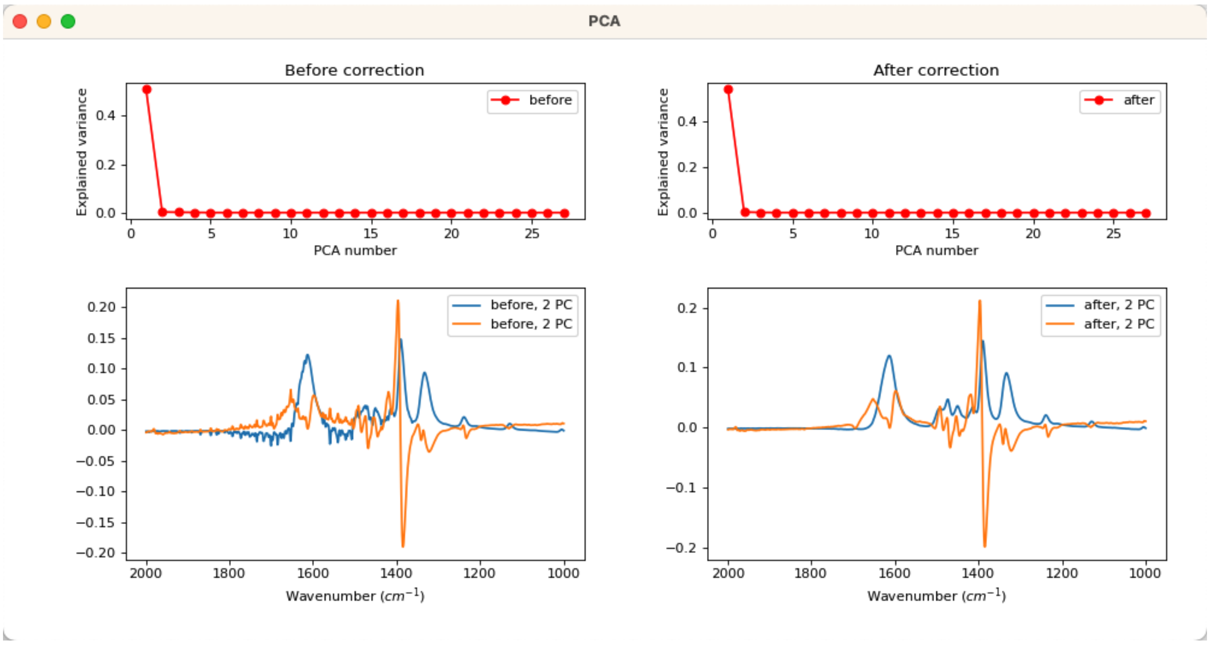
where $\sum$ denotes the sum of all data points of the spectrum or its function, and $\frac{d}{dx}$spectrum is the first derivative of the spectrum, i.e., the differences between consecutive values of the spectrum.

3. **Second Derivative Variance (SDV)**:

$$\text{Variance of second derivative} = \text{Var}\left(\frac{d^2}{dx^2}\text{spectrum}\right)$$

where $\frac{d}{dx}$spectrum is the second derivative of the spectrum, i.e., the differences between consecutive values of the $\frac{d}{dx}$spectrum.

5.7 (Optional) Perform PCA Analysis



After correction, press **“PCA”** to analyze the **principal components (PCs)** of the spectral series **before** (left) and **after** (right) correction. You can set the number of extracted PC spectra in the main window. Use the **left mouse button** to select and zoom a fragment of the spectrum. Use the **right mouse button** to reset the view. Zooming in or out in one spectrum graph will affect the other.

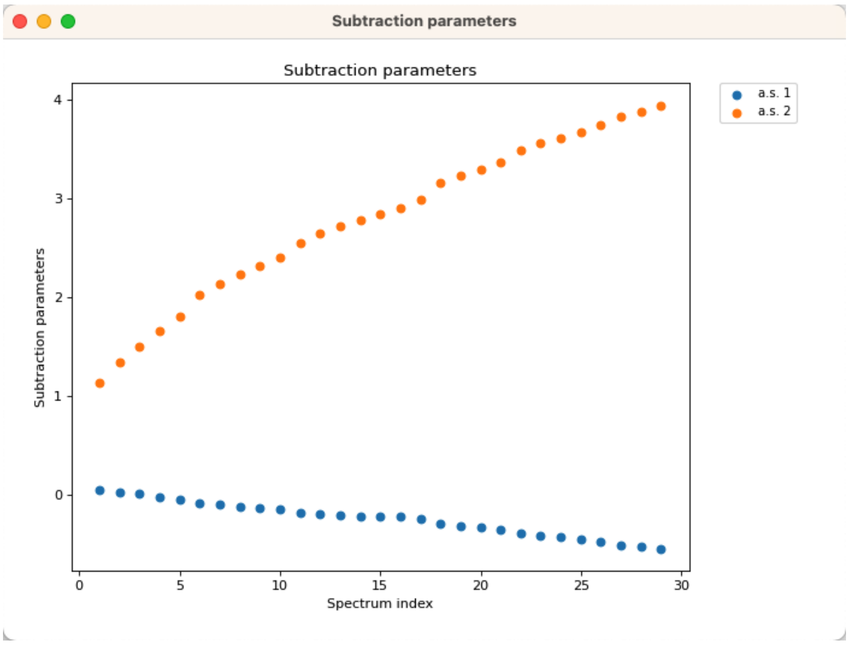
PCA window includes:

- **Explained Variance (EV) plots** (top)—A sharp drop suggests the **minimum number of significant PCs**.
- **Principal component shapes** (bottom)—Before correction, atmospheric components should be recognizable in PCs. After proper correction, they should no longer be present.

Important Notes:

- PCA **will not run** if the correction is incomplete or if fewer than **two spectra** are available.
- Misinterpretation of PCA can lead to incorrect assumptions about correction quality.
- PCA in this software is for visual assessment only. The actual number and shape of principal components should be determined using specialized software or appropriate libraries.

5.8 (Optional) Check Correction Coefficients



Use the **“Parameters Graph”** tool to inspect the correction coefficients applied to each atmospheric reference spectrum.

6 Save Results

6.1 Save Report

Press **“Save report”** to export **multiple CSV files**, which include:

- General information about the correction (date, number of spectra, file names, and paths)
- Spectra **before and after correction** (as a matrix; for the selected wavenumber range)
- **Atmospheric reference spectra** within the same range (as a matrix)
- **Correction parameters** for each spectrum
- **PCA results**, if performed

You will be asked for the report filename. The given name will serve as a prefix to all these files.

6.2 Save Corrected Spectra

Press **“Export Spectra”** to save all corrected spectra, each in a separate CSV file, in the same folder as the original spectra. All filenames will start with the ”corr_” prefix.

7 Exit the Program

Press **“Close”** to exit VaporFit.

8 Introduction to the AtmFitParams Class

The AtmFitParams class constitutes the computational core of the VaporFit software, responsible for the process of fitting and subtracting atmospheric spectra from the analyzed sample spectrum. It has been designed modularly to allow for easy integration with custom analytical scripts in Python, while simultaneously offering advanced atmospheric correction capabilities. Below is a detailed description of its structure, methods, and a practical usage example.

8.1 Class Structure and Initialization

The AtmFitParams class encapsulates all parameters and methods necessary to perform the correction.

8.1.1 Class Source Code

```python
import numpy as np
from scipy.optimize import least_squares as leastsq
from scipy.signal import savgol_filter as savitzky_golay

class AtmFitParams:
    """This class handles the parameters and methods for fitting atmospheric spectra to a given spectrum.
    It includes methods for calculating residuals, fitting the parameters,
    and subtracting the atmospheric spectra from the given spectrum."""
    def __init__(self,wavenb,spectrum,atm_spectra,sg_poly=3,sg_points=11):
        self.wavenb = np.asarray(wavenb,dtype=float) # list of wavenumbers
        self.spectrum = np.asarray(spectrum,dtype=float) # spectrum from which atmosphere (atm) spectra will be subtracted
        self.atm_spectra = np.asarray(atm_spectra,dtype=float) # array of atm. spectra
        self.no_of_atm_spctr = self.atm_spectra.shape[1] #number of vapor spectra
        self.init_fit_params = np.asarray((self.no_of_atm_spctr*[0.1]), dtype=float) # initial parameters of atm fitting
        self.no_of_parameters = np.size(self.init_fit_params) # number of parameters
        self.bounds = (self.no_of_parameters*[-np.inf],self.no_of_parameters*[np.inf]) # initial lower and upper bounds for all parameter set to +/- inf
        self.sg_poly,self.sg_points = sg_poly,sg_points # SG smoothing factors

    def residuals(self,params):
        atm_sum = np.sum((params*self.atm_spectra),axis=1)
        total_sum = self.spectrum - atm_sum
        smoothed_total_sum = savitzky_golay(total_sum,self.sg_points,self.sg_poly)
        return smoothed_total_sum - total_sum  #residual minimized during fitting

    def fit(self):
        fit_result = leastsq(self.residuals,self.init_fit_params,ftol=1.49012e-10, xtol=1.49012e-10,bounds=self.bounds)
        return fit_result.x # Fitted parameters of water vapor subtraction

    def atm_subtract(self):
        atm_sum = np.sum((self.fit()*self.atm_spectra),axis=1)
        return self.spectrum - atm_sum # Vapor-corrected spectrum
```

Listing 1: Source code of the AtmFitParams class

Note: The code above assumes the ***savitzky_golay*** *function is available, typically imported from* ***scipy.signal.savgol_filter*** *or a custom implementation within VaporFit.*

8.1.2 Constructor __init__

Initializes the class object, accepting necessary input data and optional smoothing parameters.

Arguments:

- wavenb: A 1D array (or a list convertible to a NumPy array) containing wavenumber values corresponding to the spectral points. Data type: float.
- spectrum: A 1D array (or list) with absorbance (or transmittance) values of the analyzed sample spectrum. Data type: float.

- `atm_spectra`: A 2D NumPy array where each column represents a single atmospheric spectrum used as a reference in the correction process. The number of rows must match the length of `wavenb` and `spectrum`. Data type: `float`.
- `sg_poly` (optional): The polynomial order used in the Savitzky-Golay filter for smoothing. Default value: `3`. Data type: `int`.
- `sg_points` (optional): The number of points (window size) used in the Savitzky-Golay filter. Must be an odd number and greater than `sg_poly`. Default value: `11`. Data type: `int`.

Instance Attributes (most important):

- `self.wavenb`: Stores the wavenumber axis.
- `self.spectrum`: Stores the sample spectrum to be corrected.
- `self.atm_spectra`: Stores the array of atmospheric spectra.
- `self.no_of_atm_spctr`: The number of atmospheric spectra used.
- `self.init_fit_params`: Initial values of the parameters (scaling coefficients for each atmospheric spectrum) used in the optimization process. By default, initialized to 0.1 for each spectrum.
- `self.sg_poly`, `self.sg_points`: Savitzky-Golay smoothing parameters.

8.2 Class Methods

8.2.1 Method `residuals(self, params)`

This is a key internal method used by the optimization algorithm (least squares method). It calculates the residuals that are minimized during the fitting process.

Arguments:

- `params`: A 1D array containing the current values of the scaling coefficients for the individual atmospheric spectra, passed by the optimization function.

Operation:

1. Calculates the weighted sum of atmospheric spectra (`atm_sum`) using the current coefficients `params`.
2. Subtracts this sum from the original sample spectrum, obtaining a pre-corrected spectrum (`total_sum`).
3. Smooths the pre-corrected spectrum using the Savitzky-Golay filter (`smoothed_total_sum`) with the parameters `self.sg_points` and `self.sg_poly`.
4. Returns the difference between the smoothed spectrum and the spectrum before smoothing. This difference is minimized by the `least_squares` algorithm, leading to an estimation of the ”ideal” sample spectrum devoid of sharp atmospheric peaks.

Returns:

- A 1D array of residuals.

8.2.2 Method `fit(self)`

Performs the actual fitting (optimization) process of the scaling coefficients for the atmospheric spectra, using the least squares method implemented in the `scipy.optimize.least_squares` function.

Operation:

1. Calls the `least_squares` function, passing it the `self.residuals` method as the function to be minimized and `self.init_fit_params` as the initial parameter values.
2. The parameters `ftol` and `xtol` control the convergence tolerance of the algorithm.
3. `bounds` define the limits for the optimized parameters (unbounded by default).

Returns:

- A 1D NumPy array containing the optimized scaling coefficients $\mathtt{a_n}$ (see Equation 1 in the publication) for each of the reference atmospheric spectra.

8.2.3 Method atm_subtract(self)

Subtracts the weighted sum of atmospheric spectra (using the optimized coefficients from the fit() method) from the original sample spectrum.

Operation:

1. First, it calls the self.fit() method to obtain the optimal scaling coefficients.
2. Calculates the weighted sum of atmospheric spectra (atm_sum) using these optimized coefficients.
3. Subtracts atm_sum from the original self.spectrum.

Returns:

- A 1D NumPy array representing the sample spectrum after atmospheric correction.

8.3 Using the AtmFitParams Class in Custom Code

The modular design of the AtmFitParams class makes it an ideal tool for integration with existing or newly developed spectroscopic data analysis workflows in Python. Users can easily import the class and utilize its methods for batch processing of spectral series, automating analysis, or building more complex applications.

8.3.1 Usage Example

The following example illustrates the basic way to use the AtmFitParams class to perform atmospheric correction on a single spectrum.

```python
import numpy as np
from scipy.optimize import least_squares as leastsq
from scipy.signal import savgol_filter as savitzky_golay
# Assuming AtmFitParams class is defined above or imported
# from vaporfit_core import AtmFitParams

# --- Helper function to create Gaussian peaks ---
def gaussian(x, mu, sig, amp):
    return amp * np.exp(-np.power(x - mu, 2.) / (2 * np.power(sig, 2.)))

# 1. Prepare input data
wvn = np.linspace(4000, 400, 1800) # Wavenumbers

# Sample spectrum with a main band and two atmospheric artifact sets
sample_spec = np.random.rand(1800) * 0.001 # Base noise
sample_spec += gaussian(wvn, 1650, 100, 0.5)  # First sample band
sample_spec += gaussian(wvn, 3000, 250, 0.3)  # Second sample band
# Atmospheric set 1
atm_peaks1 = gaussian(wvn, 3200, 5, 0.3) + gaussian(wvn, 1600, 3, 0.2)
# Atmospheric set 2
atm_peaks2 = gaussian(wvn, 3250, 5, 0.25) + gaussian(wvn, 1610, 3, 0.22)
sample_spec += atm_peaks1 + atm_peaks2

# Reference atmospheric spectra (simulating slight variations)
atm_ref1 = np.random.rand(1800) * 0.001 + atm_peaks1 * 0.9
atm_ref2 = np.random.rand(1800) * 0.001 + atm_peaks2 * 1.1
atm_array = np.vstack((atm_ref1, atm_ref2)).T

# 2. Initialize AtmFitParams object
corrector = AtmFitParams(wvn, sample_spec, atm_array)
# With custom SG:
# corrector = AtmFitParams(wvn, sample_spec, atm_array, sg_poly=3, sg_points=15)

# 3. Perform correction
corrected_spec = corrector.atm_subtract()

#4. Further analysis or visualization (example)
import matplotlib.pyplot as plt
plt.plot(wvn, sample_spec, label='Original')
plt.plot(wvn, corrected_spec, label='Corrected')
plt.legend(); plt.show()
```

```python
fitted_coeffs = corrector.fit()
print(f"Fitted coefficients: {fitted_coeffs}")

print("Correction finished. Corrected spectrum in 'corrected_spec'.")
```

Listing 2: Concise example of using the `AtmFitParams` class in Python

Steps in the example:

1. **Data Preparation**: Example data is created for the wavenumber axis, the sample spectrum (with simulated sample bands and atmospheric artifacts), and two reference atmospheric spectra. In practice, this data would be loaded from measurement files.

2. **Object Initialization**: An object `corrector` of the `AtmFitParams` class is created, passing the prepared data.

3. **Performing Correction**: The `atm_subtract()` method is called, which executes the entire fitting and subtraction process. The result (corrected spectrum) is assigned to the `corrected_spectrum` variable.

4. **Further Actions**: The corrected spectrum is ready for further analysis, visualization (e.g., using `matplotlib`), or saving to a file. The example also shows how to obtain the fitting coefficients themselves using the `fit()` method.

This example demonstrates the simplicity with which the `AtmFitParams` class can be used for advanced atmospheric correction of FTIR spectra, paving the way for more precise and reliable quantitative and qualitative analyses.

9 Final Notes

- VaporFit is designed for atmospheric corrections in **FTIR spectroscopy** but can process any CSV spectral files.
- If you experience issues with correction quality, double-check **input spectra consistency** and consider adjusting **wavenumber ranges** or **reference spectra selection**.
- If correction is unsatisfactory, ensure you have enough atmospheric spectra covering varied conditions; set the SG polynomial order to 3 and the SG window size to 5, or try correction with a single atmospheric spectrum.
- The software is actively developed, and new features may be added in future releases.
- We tested and compiled the software on MacOS v.15.3.1 (Mac Mini M4) and Windows 11 (Intel Core i5-8365).
- If you are experiencing issues with the compiled versions, particularly security warnings on Windows (e.g., being blocked as potential malware), try running the Python source code directly in your Python environment.