ALBA Synchrotron
MIRAS

Raül Perea Causín
July 3, 2017

RMieS-EMSC

Resonant Mie Scattering Extended Multiplicative Signal Correction (RMieS-EMSC) was developed in Matlab by Paul Bassan during his PhD thesis [1–3]. This method has been developed in Python at ALBA for being used in Orange. This document summarizes the implementation of this algorithm, and it is divided in two parts. First, scattering correction will be introduced. Afterwards, its implementation will be described.

Theoretical basis of the algorithm

Before introducing resonant Mie scattering, the correction algorithm used non-resonant Mie scattering theory [4]. Van de Hulst derived the following approximation of the Mie extinction efficiency [7], which is simpler than the original theory published by Mie [6].

$$Q_{\text{ext}}\left(\tilde{\nu}\right) \approx 2 - \frac{4}{\rho}\sin\rho + \frac{4}{\rho^2}\left(1-\cos\rho\right) \tag{1}$$

where

$$\rho = \alpha\tilde{\nu} \tag{2}$$

$$\alpha = 4\pi r\left(n_r - 1\right) \tag{3}$$

Here n_r and r refer to the refractive index of the particle relative to the medium and the radius of the particle, respectively.

In resonant Mie scattering, the change in the real refractive index is taken into account [1]. According to Bassan, the real refractive index can be obtained by a Kramers-Kronig transform of the imaginary refractive index, which is proportional to the reference spectrum. Thus, the real part of the refractive index, n, can be obtained from the following equation:

$$n = a + bn_{\text{KK}} \tag{4}$$

where n_{KK} is the Kramers-Kronig transform of the imaginary part of the refractive index, a is the average real part of the refractive index, and b is the amplification factor of n_{KK}. Since the real part of refractive index cannot be lower than 1, b must be controlled. Considering that n_{KK} is arbitrarily normalized so that its minimum value is -1, then $b \in (0, a-1)$.

Since a, b and the cell's diameter d are not known, a range of these values is used to calculate the Mie extinction, leading to the extinction matrix $\mathbb{Q}_{\text{ext}}$. In order to consider all the possible contributions, the principal components ($\vec{p}_i$) of $\mathbb{Q}_{\text{ext}}$ are computed and introduced in the EMSC algorithm multiplied by parameters g_i that will determine their weight in the fit.

The EMSC algorithm fits the apparent spectra with a reference spectrum ($h\vec{Z}_{\text{ref}}$), a baseline ($c + m\vec{\nu}$), and the Mie scattering contributions ($g_i\vec{p}_i$):

$$\vec{Z}_{\text{raw}} = c + m\vec{\nu} + h\vec{Z}_{\text{ref}} + \sum_{i=1}^{N_{\text{PC}}} g_i\vec{p}_i + \vec{\varepsilon} \tag{5}$$

where $\vec{\varepsilon}$ accounts for the un-modelled residuals and N_{PC} is the number of principal components of $\mathbb{Q}_{\mathrm{ext}}$'s PCA.

After fitting the apparent spectra with Eq. (5), its correction is performed by subtracting the baseline, the scattering effects and by cancelling out the "reference amplification factor":

$$\vec{Z}_{\mathrm{corr}} = \frac{\vec{Z}_{\mathrm{raw}} - \left(c + m\vec{\nu}\right) - \sum_{i=1}^{N_{\mathrm{PC}}} g_i \vec{p}_i}{h} \tag{6}$$

The reference spectrum, however, must be a very clean sample representative of the dataset. It may have unnoticeable scattering effects, but it still will have a baseline that needs to be corrected. To correct the reference, Kohler used this same method without the contribution of the reference. In order to perform the fitting correctly, only the regions that do not contain chemical information of the samples is considered (*e.g.* 1780-2683 cm^{-1} and 3680-4000 cm^{-1}). Then, the reference is fitted and corrected as follows:

$$\vec{Z}_{\mathrm{ref}} = c + m\vec{\nu} + \sum_{i=1}^{N_{\mathrm{PC}}} g_i \vec{p}_i + \vec{\varepsilon} \tag{7}$$

$$\vec{Z}_{\mathrm{ref}}^{(\mathrm{corr})} = \vec{Z}_{\mathrm{ref}} - \left(c + m\vec{\nu}\right) - \sum_{i=1}^{N_{\mathrm{PC}}} g_i \vec{p}_i \tag{8}$$

In practice, the wavenumber term has not been used for the reference correction because it caused the algorithm to fail.

To improve the accuracy, an iterative method is introduced. After each correction, the reference spectrum is updated by the corrected apparent spectrum. However, I still have some doubts about this issue. Is the reference updated only for computing the refractive index, and for the fitting is used the "first" reference? Is the reference correction performed for each updated reference?

Although it is not explained in Bassan's papers, I have found that the fitting has also to be applied only to the specific regions not affected by chemical bonds. This is clear in the results shown by Bassan: only absorbance at these specific regions is zero for all corrected spectra, while at other regions there is discrepancy between the absorbance values. Of course, this difference is desired in the regions that contain chemical information.

Implementation of the algorithm

Here I will comment the code that I have written.

The main function is `Bassan`. This function performs the correction with the input data and returns the corrected spectum. First of all, a copy of the input data is made. In Python, a list variable is a reference to some data. Thus, if we do `a = [1,2,3]` and then assign `b = a`, when we change `a`, `b` will change as well. That's why input data needs to be copied. This is done using numpy's `copy()`.

Since the wavenumbers are usually sorted from larger to lower values, input arrays need to be sorted. First, the wavenumber array is resorted from lower to larger values, taking the indexes. Then, the reference and the apparent spectrum are sorted following these indexes. This is done using the following code:

```
# Copy the input data
wn = np.copy(wavenumbers)
```

```
A_app = np.copy(App)
m_0 = np.copy(m0)
ii = np.argsort(wn)  # Sort the wavenumbers
# Apply the sorting to the input variables
wn = wn[ii]
A_app = A_app[ii]
m_0 = m_0[ii]
```

Afterwards, the reference spectrum is corrected and the weighted spectra are calculated. To get the spectra at the specified regions, the program goes through each pair of values in the variable `w_regions`, which correspond to the minimum and the maximum limits of each region, and it takes the index at which these values are located in the wavenumber's array. With this, a list containing the indexes that have to be considered is created, and then the weighted arrays are built.

```
m_0 = correct_reference(np.copy(m_0), wn, a, d, w_regions)  #
    Correct the reference spectrum as in Kohler method
w_indexes = []
# Get the indexes of the regions to be taken into account
for pair in w_regions:
    min_pair = min(pair)
    max_pair = max(pair)
    ii1 = find_nearest_number_index(wn, min_pair)
    ii2 = find_nearest_number_index(wn, max_pair)
    w_indexes.extend(np.arange(ii1, ii2))
# Take the weighted regions of wavenumbers, apparent and reference
    spectrum
wn_w = np.copy(wn[w_indexes])
A_app_w = np.copy(A_app[w_indexes])
m_0_w = np.copy(m_0[w_indexes])
```

The system range of parameters have also to be initialized: the number of values computed for each parameter, the array of values for a ($a \in [1.1, 1.5]$) and for d ($d \in [2, 8]$ μm).

```
n_loadings = 10  # Number of values to be computed for each
    parameter (a, b, d)
a = np.linspace(1.1, 1.5, n_loadings)  # Average refractive index
d = np.linspace(2.0, 8.0, n_loadings) * 1.0e-4  # Cell diameter
```

Before starting the iterative loop, the extinction matrix's shape is determined and the reference spectrum is initialized. The matrix will have as many rows as the combinations of a, b and d.

```
Q = np.zeros((n_loadings ** 3, len(wn)))  # Initialize the
    extinction matrix
m_n = np.copy(m_0)  # Initialize the reference spectrum, that will
    be updated after each iteration
for iteration in range(iterations):
    ...
```

The first thing that is done once the iterative loop is started is to calculate the Kramers-Kronig transform of the reference spectrum. According to Konevskikh [5], this can be computed through

a Hilbert transform. Scipy's Hilbert function returns the hilbert transform in the imaginary part. Besides, the transformation from the imaginary real index (proportional to the reference spectrum) to the real one is done through an inverse Kramers-Kronig transform, which is the same but with opposite sign. Afterwards, the variable n_{KK} is normalized so that its minimum value lays at -1.

```
# Compute the scaled real part of the refractive index by Kramers-
    Kronig transform:
nkk = -1.0 * np.imag(hilbert(m_n))
nkk -= np.min(nkk) + 1.0
```

The next step is to construct the extinction matrix. In order to do this, the code iterates over each combination of the values of a, b and d. In each a iteration, b is recomputed, since it depends on the a value. With each combination of these values, a new row of the extinction matrix is computed.

```
# Build the extinction matrix
n_row = 0
for i in range(n_loadings):
   b = np.linspace(0.0001, a[i] - 0.9999, 10)  # Range of
       amplification factors of nkk
   for j in range(n_loadings):
      for k in range(n_loadings):
         n = a[i] + b[j] * nkk  # Compute the real refractive
             index
         alpha = 2.0 * np.pi * d[k] * (n - 1.0)
         rho = alpha * wn
         #  Compute the extinction coefficients for each
             combination of a, b and d:
         Q[n_row] = 2.0 - np.divide(4.0, rho) * np.sin(rho) + \
         np.divide(4.0, rho ** 2.0) * (1.0 - np.cos(rho))
         n_row += 1
```

The extinction matrix, however, needs to be orthogonalized with respect to the reference spectrum in order to avoid competition between the parameters h and g_i when performing the PCA of Q_{ext}. Hence, each spectrum of the extinction matrix (*i.e.* each row) is orthogonalized with respect to the reference spectrum. Considering $\vec{v}$ as any vector and $\vec{Z}_{\mathrm{ref}}$ sa the reference spectrum, then the orthogonalized v can be computed as:

$$\vec{v}^{(\mathrm{ort})} = \vec{v} - \frac{\vec{v} \cdot \vec{Z}_{\mathrm{ref}}}{\vec{Z}_{\mathrm{ref}} \cdot \vec{Z}_{\mathrm{ref}}} \vec{Z}_{\mathrm{ref}} \tag{9}$$

Prove:

$$\vec{v}^{(\mathrm{ort})} \cdot \vec{Z}_{\mathrm{ref}} = \vec{v} \cdot \vec{Z}_{\mathrm{ref}} - \frac{\vec{v} \cdot \vec{Z}_{\mathrm{ref}}}{\vec{Z}_{\mathrm{ref}} \cdot \vec{Z}_{\mathrm{ref}}} \vec{Z}_{\mathrm{ref}} \cdot \vec{Z}_{\mathrm{ref}} = \vec{v} \cdot \vec{Z}_{\mathrm{ref}} - \vec{v} \cdot \vec{Z}_{\mathrm{ref}} = 0$$

```
# Orthogonalization of the extinction matrix with respect to the
    reference spectrum:
for i in range(n_loadings ** 3):
   Q[i] -= np.dot(Q[i], m_0) / np.linalg.norm(m_0) ** 2.0 * m_0
```

Once the extinction matrix is built and orthogonalized, its principal components can be computed. I use the sklearn library, concretely the skl_decomposition package for the PCA computation. The procedure to follow in order to obtain the principal components is shown in the following piece of code:

```
# Perform PCA of the extinction matrix
pca = skl_decomposition.IncrementalPCA(n_components=n_components)
pca.fit(Q)
p_i = pca.components_  # Get the principal components
```

Since for the fitting step we are only interested in some specific regions, new vectors containing the principal components only in these regions are needed. This is done in the same way as when building the weighted wavenumber array:

```
# Take the indexes of the specified regions
w_indexes = []
for pair in w_regions:
    min_pair = min(pair)
    max_pair = max(pair)
    ii1 = find_nearest_number_index(wn, min_pair)
    ii2 = find_nearest_number_index(wn, max_pair)
    w_indexes.extend(np.arange(ii1, ii2))
p_i_w = np.copy(p_i[:, w_indexes])  # Get the principal components
     of the extinction matrix at the
# specified regions
```

Before performing the fit of the apparent spectrum, the function describing this fit needs to be defined. There is a function that directly fits some data with the desired function, but I have obtained very bad results with this. On the contrary, I have obtained good results when building my own least squares function and minimizing it. Actually, what we want to find is the following:

$$\underset{c,m,h,g_i}{\arg\min} \left\| \vec{Z}_{\text{app}} - \vec{Z}^{(\text{fit})}_{\text{app}}(c, m, h, g_i) \right\|^2$$

The function to be minimized is defined in the code as follows:

```
def min_fun(x):
    """
    Function to be minimized for the fitting
    :param x: fitting parameters (offset, baseline, reference's
        linear factor, PCA scores)
    :return: squared norm of the difference between the apparent
        spectrum and its fitting
    """
    cc, mm, hh, g = x[0], x[1], x[2], x[3:]
    return np.linalg.norm(A_app_w -
    apparent_spectrum_fit_function_Bassan(wn_w, m_0_w, p_i_w, cc,
        mm, hh, g)) ** 2.0
```

To have a more stable method, a initial guess is defined. The function used for minimization is SciPy's minimize, with the Powell method. The result of this contains different data about how did the minimization process go.

```
p0 = np.append([1.0, 0.0005, 0.9], np.ones(n_components))  #
    Initial guess for the fitting
res = scipy.optimize.minimize(min_fun, p0, method='Powell')  #
    Perform the fitting

# print(res)  # Print the result of the minimization
# assert(res.success) # Raise AssertionError if res.success ==
    False

c, m, h, g_i = res.x[0], res.x[1], res.x[2], res.x[3:]  # Take the
     fitted parameters
```

Finally, the fitting parameters are used to correct the apparent spectrum as follows:

```
Z_corr = (A_app - c - m * wn - np.dot(g_i, p_i)) / h  # Apply the
    correction
```

In order to keep iterating, the reference spectrum is updated by the corrected spectrum:

```
m_n = np.copy(Z_corr)  # Take the corrected spectrum as the
    reference for the next iteration
```

References

[1] Bassan, P., Byrne, H. J., Bonnier, F., Lee, J., Dumas, P., and Gardner, P. Resonant Mie scattering in infrared spectroscopy of biological materials–understanding the 'dispersion artefact'. *Analyst 134*, 8 (2009), 1586–1593.

[2] Bassan, P., Kohler, A., Martens, H., Lee, J., Byrne, H. J., Dumas, P., Gazi, E., Brown, M., Clarke, N., and Gardner, P. Resonant Mie scattering (RMieS) correction of infrared spectra from highly scattering biological samples. *Analyst 135*, 2 (2010), 268–277.

[3] Bassan, P., Sachdeva, A., Kohler, A., Hughes, C., Henderson, A., Boyle, J., Shanks, J. H., Brown, M., Clarke, N. W., and Gardner, P. FTIR microscopy of biological cells and tissue: data analysis using resonant Mie scattering (RMieS) EMSC algorithm. *Analyst 137*, 6 (2012), 1370–1377.

[4] Kohler, A., Sule-Suso, J., Sockalingum, G., Tobin, M., Bahrami, F., Yang, Y., Pijanka, J., Dumas, P., Cotte, M., Van Pittius, D., et al. Estimating and correcting Mie scattering in synchrotron-based microscopic Fourier transform infrared spectra by extended multiplicative signal correction. *Applied spectroscopy 62*, 3 (2008), 259–266.

[5] Konevskikh, T., Lukacs, R., Blümel, R., Ponossov, A., and Kohler, A. Mie scatter corrections in single cell infrared microspectroscopy. *Faraday discussions 187* (2016), 235–257.

[6] Mie, G. Beiträge zur optik trüber Medien, speziell kolloidaler Metallösungen. *Annalen der physik 330*, 3 (1908), 377–445.

[7] van de Hulst, H. C. *Light scattering by small particles*. Courier Corporation, 1957.