LSPR_library-tutorial

June 1, 2020

1 LSPR_library - tutorial

1.1 Introduction - what is LSPR_library?

LSPR_library is a handful of tools to approximate Transmittance/Absorbance of a metal-dielectric composite. The dielectric is considered isotropic, therefore, usually glasses apply to that rule. The metal can be introduced in the for of nanoparticles with desired concentration, mean diameter and standard deviation of the size distribution.

Several conditions are imposed on the model: 1. Mie dipole approximation is used. 2. The reflectance is dominated by the glass refractive index. 3. The angle of incidence is 90° 4. All the scattered and absorbed light is collected and adds to the absorbance. 5. The matrix and nanoparticle absorbances are additive.

1.2 Dependencies - what you need to run it smoothly?

To run LSPR_library you will need: 1. Pandas - dataframes/Series and their manipulations 2. NumPy - operations on vectors 3. SciPy - Sellmeier model fitting, integration of a function involving product of a normal distribution

1.3 How does it work? 5 steps

1.3.1 In short:

1. Load the data for metal and dielectric.
2. Ensure both of them have the same Wavelength subsets.
3. Create nanoparticle population.
4. Create a measurement.
5. Calculate the Transmittance/Absorbance.

1.3.2 In long:

1. Load the refractive index data (n,k) using OpticalMaterial class, e.g.

   a. `Silver = OpticalMaterial("Silver", "Ag_refractive_index.csv")`

   b. `Glass = OpticalMaterial("Glass", "Soda_lime_glass_refractive_index.csv")`

2. Valid results can be calculated only when Dielectric and Metal datasets (Glass and Silver in this example) hold values for the same wavelength. Make sure that both datasets have the same Wavelength subsets in the spectral region of interest. If not, then:

a. use either `OpticalMaterial.regular_xspacing` or `SpectralMeasurement.align_datasets`. These use linear interpolation so be careful when using it on sparsely populated spectral regions, e.g.:

```
evenly_spaced_refractive_index = Silver.regular_xspacing(Silver.refractive_index, 0.200, 0.801, 0.010)
```

This will fit the Silver.refractive_index data to the Wavelengths=[0.200,0.210, 0.220, …, 0.790, 0.800] and linearly interpolate the missing data. You can use this data to recreate a new dataset

```
Silver_regular = OpticalMaterial("Silver_r", evenly_spaced_refractive_index)
```

or override the existing object

```
Silver = OpticalMaterial("Silver_r", evenly_spaced_refractive_index)
```

b. for glasses use Sellmeier class to fit 3rd order Sellmeier equation and create a theoretical fit of the refractive index for the desired wavelength set, e.g.:

```
Sellmeier_fitting = Sellmeier(Glass.refractive_index) # load the data

Sellmeier_fitting.fit_data()   # Fit the 3rd order Sellmeier eq. to get the B, C constants

Sellmeier_soda_lime = Sellmeier_fitting.model(x_subset_of_choice)   # Calculate the refractive index based on the x_subset_of_choice
```

Create a new material instance with properly spaced data

```
Glass_regular = OpticalMaterial("Glass_r", Sellmeier_soda_lime)
```

or override the existing one

```
Glass = OpticalMaterial("Glass_r", Sellmeier_soda_lime)
```

3. Create a Nanoparticle object, with specified size, distribution and concentration, e.g.

   a. `nAg = Nanoparticle("Silver_nanoparticles", material=Silver, size=11, stdev=2, concentration=3.58e13`

4. Create a SpectralMeasurement instance to set the measurement, e.g.:

   a. `NewMeasurement = SpectralMeasurement("Measurement_1", matrix=Glass, nanoparticle=nAg, thickness=21)`

5. Calculate the transmittance/absorbance:

   a. `transmittance = NewMeasurement.T(R_corr=True)   # Reflectance corrected transmittance`

   b. `absorbance = NewMeasurement.Abs   #Absorbance of the composite plate with thickness 21μm`

1.4 Library contents - classes and methods

1.4.1 1. OpticalMaterial

1.4.2 2. Nanoparticle

1.4.3 3. Sellmeier

1.4.4 4. SpectralMeasurement

[1]:

[ ]: