

FP Reflectance matching - n & k determination

Technique author: Carlota Ruíz de Galarreta Fanjúl

Code authors: Emanuele Gemo, Liam Trimby

University of Exeter, 06/06/2020

LOG

- v0.1 – stable version

TO DO

- set the search space as an additional variable
- calculate the errors on user request

This method determines the material property of a FP cavity material, which consists of a reflective layer upon which the cavity material needs to be deposited. The reflection of the FP cavity is then to be experimentally determined.

This code takes such FP cavity reflection experimental *spectrum*, and by with the provision of a set of possible n/k dispersions (here called *references*), determines the closest match (*solution*) to the cavity material, by calculation of the weighted average of the provided references. The method is based on the simulation of the cavity spectrum by transfer-matrix-method, which also requires fairly precise characterization of the thickness of each involved material layer, and the knowledge of the non-investigated materials optical properties.

The requirement of the FP cavity arises from the peculiar features of its reflection spectrum, which contains phase information encoded in the amplitude spectrum, which (around FP resonance) permit to calculate a unique n/k solution.

Index

1. Using the code, standard method
2. List of functions
3. Variables
 - 3.1. `range` variable
 - 3.2. `structure` variable
 - 3.2.1. *Standard use*
 - 3.3. `spectrum` variable
 - 3.3.1. *Standard use*
 - 3.3.2. *Optional parameters*
 - 3.4. `reference` variable
 - 3.4.1. *Standard use*
 - 3.4.2. *Optional parameters*
 - 3.5. *Additional (optional) variables*
 - 3.5.1. `comparison` variable
 - 3.5.2. `search_space` variable
 - 3.5.3. `flag_plot` variable
 - 3.5.4. `force_e2_above_0` variable
 - 3.6. *Compulsory variables of optional functions*
 - 3.6.1. `fit_degree` variable
4. `load_data` function
5. `cavity_patternsearch_v1` function
6. `build_artificial_reference` function

INSTRUCTIONS

1. Using the code, standard method

What follows is an example on how the code is meant to be used:

```
range = (800e-9 : 5e-9 : 1600e-9)'; % wavelength [m];
% structure data
structure.material = {'SiO2',[],'Al','Si'}; % material filenames; investigated material - EMPTY
structure.thickness = [10e-9,200.8e-9,224.5e-9,525e-5]; % thickness [m]
% experimental data
spectrum.file = 'ref_cGeTe_205_10min300deg'; % measurement filename (.mat)
% reference data
reference.material = {'cGeTe_Jafari','cGeTe_Shportko','cGeTe_Park','cGeTe_Peinado_3','cGeTe_Robertson'}; %
refs filenames
% comparsion data
comparison.material = {'cGeTe_liam'}; % reference or ellipsometry data, to compare with calculation output

%% calculate
% load data
[range,structure,spectrum,reference,comparison] = load_data(range,structure,spectrum,reference,comparison);

% build artificial reference
[artificial_reference,reference] = build_artificial_reference(...
    range,structure,spectrum,reference,3);

% calculate solution
solution = cavity_patternsearch_v1(structure,spectrum,reference,comparison,1,true);
```

The details will be better explored in the sections below.

2. List of functions

This version of the code counts two compulsory function and one optional function.

These are:

- `[range, structure, spectrum, reference, comparison] = ... load_data(range, structure, spectrum, reference, comparison);`
- `solution = cavity_patternsearch_v1(structure, spectrum, reference,... comparison, search_space, flag_plot);`
- (optional) `[artificial_reference,reference] = ... build_artificial_reference(range, structure, spectrum, reference,... fit_degree, force_e2_above_0, flag_plot);`

The section below explores the user-provided variables and provides the necessary information on how to set up the script.

3. Variables

3.1. range variable

This variable is a column array containing the values of the wavelengths (in meters) which will be investigated by the code. An example below:

```
range = (800e-9 : 5e-9 : 1600e-9)'
```

3.2. structure variable

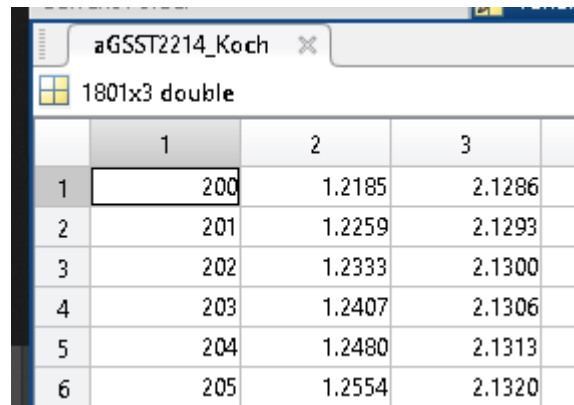
3.2.1. Standard use

`structure` is a struct variable which describes the FP cavity geometry and compositions. It contains the following fields:

i. `structure.material = {'SiO2', [], 'Al', 'Si'};`

`material` is a cell array, that sorts from left to right the FP layers as faced in the optical path from the top to the bottom (i.e. in this case, `SiO2` is the top layer, and `Si` is the bottom layer. The investigated material (or materials) MUST be left empty valued, as the second entry in the cell array of the example above.

Each cell in particular contains the name of a .mat file, which when loaded must contain a variable with the same name of the file. This variable must be a column array with three entries: the wavelength, the n value and the k value (see example below, Fig. 1). It is suggested to store within the same .mat file also a string variable (possibly called “source”), containing the reference from which such dataset is extracted. Wavelength unit can be either meters, or nanometers, but not anything else at the moment. *The code recognizes the nanometer value if it finds the first entry below unity*, so its not totally error-proof.



	1	2	3
1	200	1.2185	2.1286
2	201	1.2259	2.1293
3	202	1.2333	2.1300
4	203	1.2407	2.1306
5	204	1.2480	2.1313
6	205	1.2554	2.1320

Figure 1. Example of material variable called `aGSST2214_Koch`

ii. `structure.thickness = [10e-9, 200.8e-9, 224.5e-9, 525e-5];`

`thickness` is a row array of numerical (positive) variables, which points to the thickness of each layer of the FP cavity, in the same order as in the material variable.

3.2.2. Optional fields

Whilst these fields above are compulsory, it is also possible to add a few other fields:

iii. `structure.l_column = [1];`

iv. `structure.n_column = [2];`

v. `structure.k_column = [3];`

these variables state that, for ALL material files, the wavelength is found in the `l_column` variable; the refractive index is found in the `n_column` variable; and the extinction coefficient is found in the `k_column` variable. These are assumed to be 1,2, and 3 by default, but can be override if expressed as specified.

vi. `structure.adjust_parameter = [1,1,0.95,1.25]`

this variable allows to multiply the n and k variables of the relative loaded material by its value. It can either contain a single value (which will therefore be applied to EACH of the device materials, less the cavity material); or, it can contain as many variables as the material list contains, as in the example above (once more, the entry for the investigated materials will be ignored).

3.3. `spectrum` variable

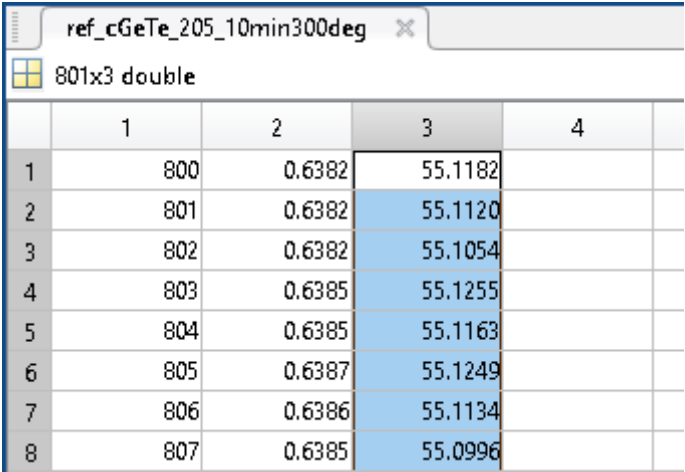
3.3.1. Standard use

`spectrum` is another struct variable which contains the information about the experimental reflectance of the FP cavity. It contains the following fields:

i. `spectrum.file = 'ref_cGeTe_205_10min300deg';`

`file` is a variable which contains a single string, referring to the .mat file which will be loaded (similarly to the use of `structure.material`, less the use of a cell array).

In the same way as the material files, the reflectances must be .mat files containing a column array variable with the same name, which have the wavelength values on its first column, and the reflectance values on its second column (see example below, Fig. 2). Any other column in such file, by default, is ignored. As in material .mat files, wavelength must be reported either in meters or nanometers, and the same method of unit determination is applied. Other variables can be optionally stored in the reflectance .mat files.



	1	2	3	4
1	800	0.6382	55.1182	
2	801	0.6382	55.1120	
3	802	0.6382	55.1054	
4	803	0.6385	55.1255	
5	804	0.6385	55.1163	
6	805	0.6387	55.1249	
7	806	0.6386	55.1134	
8	807	0.6385	55.0996	

Figure 2. Reflectance material example. Blue column is ignored by the code, by default.

3.3.2. Optional parameters

It is possible to specify the following optional fields:

- ii. `spectrum.l_column = [1];`
- iii. `spectrum.r_column = [2];`

these force the code to look for the wavelength variable in the `l_column` column, and the reflectance variables in the `r_column` column. By default, these are 1 and 2 respectively.

3.4. *reference variable*

3.4.1. *Standard use*

The `reference` variable allows the user to specify which references the code can use to calculate the problem solution. It is a struct variable, containing the following fields:

- i. `reference.material = {'aGST_Gholipour', 'Se_Ciesielski'};`

The `material` variable contains a row cell array, which contains a set of strings each one specifying the name of the .mat file which will be loaded and then used as references. Its manipulation is exactly the same as in `structure.material`, and therefore the code expects to find the .mat files containing a column array variable with the same name, which contains: wavelength on the 1st column; n on the 2nd column; k on the 3rd column.

3.4.2. *Optional parameters*

The following fields can be specified, with the very same use and conditions as in `structure.material`:

- ii. `reference.l_column = [1];`
- iii. `reference.n_column = [2];`
- iv. `reference.k_column = [3];`
- v. `reference.adjust_parameter = [1,1.05]`

note – the number of values contained in `adjust_parameter` must be either 1 to be applied to EACH of the reference materials, or must have the same number of values as the cell array in `reference.material` (in the reported example, we have 2 entries for both variables).

3.5. *Additional (optional) variables*

The following optional variables can also be used:

Whilst the first three have already been fully covered, the remaining will be described below.

3.5.1. *comparison variable*

This value is used only for plot purposes. Its structure, sub-fields and handling are identical to `reference.material`, so they will not be reiterated here. If the code is required to plot the

results, the comparison materials will also be plotted, to provide the user a picture of how close or distant from the comparison materials values the found solutions are.

3.5.2. `search_space` variable

This value is by default set to 1. It expresses the sum of the weights applied to the references ensemble, thus with a `search_space` variable set to 1 the code will span through the search space comprised within the references. By setting it to be larger than one, the code forces the weights sum to match the `search_space` variable, thus expanding the search space further from the space comprised within the references.

Note: in the current version, the code is bound to set the weight sum to `search_space`; thus, even if an optimal solution exists with a different weight sum, the code won't be able to find it.

3.5.3. `flag_plot` variable

This value is a logical, which enables the code to plot a graphical representation of the solutions and calculations. If set to be *true* (the default option is *false*), the code opens the necessary figures to graphically represent its outputs, such as the found weights and related n/k/reflectance solutions.

3.5.4. `force_e2_above_0` variable

This variable is used only in the `build_artificial_reference` function. As its name suggests, it forces the artificial reference imaginary component above 0, which is useful when such element is expected. By default, its value is set to `true`. To deactivate such behavior, this variable must be set to `false`.

3.6. Compulsory variables of optional functions

3.6.1. `fit_degree` variable

This variable is used only in the `build_artificial_reference` function. As the yielded reference is affected by a high degree of noise, a fit is compulsory. Consequently, the function fits the `artificial_reference` data with a `fit_degree` order polynomial.

4. `load_data` function

This function is called as follows (optional variables indicated in *italic*):

```
[range, structure, spectrum, reference, comparison] = ...  
load_data(range, structure, spectrum, reference, comparison);
```

As its name suggests, this function perform the load operations as described in the variable section. The load operation also perform an akima-spline interpolation of the provided datapoints, and extracts the values indicated by the range.

Another operation that `load_data` performs is including in the `range` the FP cavity resonance wavelength, which, at this version, is simply (and possibly, erroneously) found as the minimum in the reflection spectrum (after interpolation with the akima-spline method, on a 1e6 point raster). A third operation is the restriction of the `range` based on the provided `spectrum` (i.e. if the `spectrum` does not contain all points as indicated by the `range`, the `range` is then trimmed outside the `spectrum` provided range).

5. `cavity_patternsearch_v1` function

This function performs the calculation of the n/k values, and stores these on the `solution` variable. The function is called as follows (optional parameters indicated in *italic*):

```
solution = cavity_patternsearch_v1(structure, spectrum, reference, ...  
comparison, search_space, flag_plot);
```

6. `build_artificial_reference` function

This optional function integrates the reference pool by constructing the best guess based on both the provided references and the reflection spectrum. It is called as follows (optional parameters indicated in *italic*):

```
[artificial_reference, reference] = build_artificial_reference(range, ...  
structure, spectrum, reference, fit_degree, force_e2_above_0, flag_plot);
```

The function output includes both the modified `reference` variable, and separately the `artificial_reference` variable, which can then be saved by the user if needed be (this function is actually a time-consuming task, so the user might want to avoid to call it twice).