pyFresnel

Robert Steed

06/04/13

pyFresnel is a collection of python modules for modelling the reflectivities and transmissions of dielectric interfaces. Increasingly, there are two parts for the library, those modules for modelling a simple interface or an etalon, and the modules for modelling a series of dielectric layers (also known as a thin film coating). The reflectivities and transmittances of thin film structures can be found using a transfer matrix calculation which represents the interfaces and layers as a series of 2x2 matrices. Many such codes exist already but this one was developped for modelling uniaxial layers (layers with a bi-refringent or uniaxial refractive index) which have an extraordinary axis along the growth direction of the stack. This is useful for certain types of semiconductor structures and in my case, optical transitions in semiconductor quantum well structures.

Usage

Since there is no GUI interface, this library requires some knowledge of python in order to be used effectively. The library depends upon the python libraries numpy, matplotlib and scipy.

Each module contains example code that demonstrates the modules in action. There are more examples in the examples directory, they use the module init.py to find the pyFresnel modules in case you have chosen not to install the package into your python site-packages directory or include it on your system path. The classes rely on python's flexibility with regard to types and so the inputs are not rigorously checked for correctness (this is not a good example of rigorous programming). Despite that, the code works pretty well and I have striven instead to write code that can be easily understood.

There are many possible choices of spectral units (Hz,THz,meV,nm,cm^{-1}) but here I have chosen to use natural frequency which is 2π times the real frequency and normally uses a symbol ω. The natural frequency is useful theoretically but will normally need to be converted to something else for comparison with measurements. Different communities use different spectral units and I decided to stick to one unit rather than try to make the code aware of different spectral scales.

The codes can normally be called w.r.t. to a range of frequencies or a range of incident angles but not both at the same time.

The complex refractive index is defined by $n + i\kappa$ rather than $n - i\kappa$ or $n\,(1 + i\kappa)$. I have also several times written duplicate code that uses dielectric constants instead of refractive indices and so it is up to the user which he/she uses. Since the relation between the two is trivially $n = \sqrt{\varepsilon}$ maybe it is not really necessary but many theoretical models of the dielectrics yield dielectric constants.

Finally, there should shortly be a separate document which gives the theoretical notes behind the code.

Modules

Modules to model the optics of simple dielectric interfaces:

- fresnel.py - contains the class Interface for modelling dielectric interfaces. We can instantiate this class using Interface(n1=1.0,n2=1.0,theta=0.0) where n1 is the initial refractive index, n2 is the final refractive index and theta is the angle of incidence in radians.
- fresnel_uniaxial.py - contains a class to model an interface with a uniaxial medium which has its extraordinary axis perpendicular to the interface.
- materials.py - classes to model dielectric constants/refractive indices of materials. The main class is Material which contains the methods to convert between the dielectric constant or the refractive index, but this shouldn't be directly instantiated. There is a subtle issue regarding whether to use natural or real frequencies. Natural frequencies are 2*pi* real frequency and they crop up all of the time in physics. In some of the models, one or other may be implicitly assumed but I will try to have left some comments when this happens. Look at the file to see the materials defined.
- optical_plate.py - contatins the Plate class for modelling a slab of dielectric (also known as an etalon). We can create an object of this class using Plate(n1,d,w,theta,n0=1.0,n2=1.0) where n0 is the refractive index before, n1 is the refractive index of the plate/slab/layer, n2 is the refractive index after the plate, d is the thickness of the plate, w is the natural frequency and theta is the angle in radians.
- effective_medium.py - contains two classes for calculating an effective dielectric for a stack of thin layers under the assumption that the layer thicknesses are all much less than the wavelength of the light considered. The main class is EffectiveMedium(layers) where layers is a tuple of tuples like (n,thickness (m)) or tuples like (nzz,nxx,thickness (m)) or Layer objects from transfer_matrix.py. The other class is EffectiveMedium_eps(layers) where layers is a tuple of tuples like (epszz, epsxx, thickness (m)).
- uniaxial_plate.py - contains 2 classes, AnisoInterface and AnisoPlate. These allow us to model an interface or a plate which has an uniaxial medium under the rather limiting condition that the uniaxial medium has its optical axis perpendicular to the interface. We call these classes using AnisoInterface(n1o,n1e,n2o,n2e,theta) AnisoPlate(n1o,n1e,d,w,theta,n0=1.0,n2=1.0) where n1o and n2o are the ordinary refractive indices, n1e and n2e are the extraordinary refractive indices and the other quantities are as before. For AnisoPlate, the refractive indices are the media before and after the layer\slab\plate are assumed to isotropic.
- uniaxial_plate2.py - contains a class AnisoPlate that is very similar to the class in uniaxial_plate but comes from a different derivation. It is called using AnisoPlate(n_xx,n_zz,d,w,theta,n_b=1.0) or AnisoPlate_eps(eps_xx,eps_zz,d,w,theta,eps_b=1.0) eps_zz is the dielectric constant perpendicular to plate's sides, equivalent to the extraordinary refractive index squared, while eps_xx is the dielectric constant for electric fields parallel to the plate's sides which is equivalent to the ordinary refractive index squared. This code doesn't allow us to separately set the properties of the medium either side of the plate, instead we use eps_b to set the dielectric constant of media either side of the plate.

Tranfer matrix modules

- transfer_matrix.py - Contains 6 classes Layer, LayerUniaxial, Layer_eps, LayerUniaxial_eps, Filter_base and Filter. This is an Optical Transfer Matrix code. It takes a description of the layers and calculates the transmission and reflection. Unusually it include a very special anisotropic/uniaxial

case where the dielectric is different along the perpendicular/ layer stack axis than to the in-plane directions; this is so we can describe quantum well intersubband absorptions. It can also plot the electric field and the absorptivity versus depth within the structure. There is also the possibility to model partially coherent layers (see the transfer matrix examples folder for details).

- incoherent_transfer_matrix.py - Contains IncoherentFilter (based on Filter_base) - to module a structure with coherent and incoherent layers use this as the top class, although it has less features than the Filter class in transfer_matrix.

- layer_types.py - contains material types (basically materials.py adjusted for transfer_matrix compatability). There is also a class for loading Sopra refractive index data files (which can be downloaded from http://www.sspectra.com/sopra.html).

Other modules

- constants.py - physical constants library by Herman J.C. Berendsen, <www.hjcb.nl/python> (released under GPL and so compatible with this library)

- finite_well.py - Calculates the conduction band levels of a finite AlGaAs-GaAs quantum well and the associated dielectric constant.

Examples

Tranfer Matrix Examples

There are examples for the transfer matrix code in the folder 'transfer_matrix examples'. These include replications of the results in various articles and the examples from the freesnell program. Therefore we can be relatively sure of the program for isotropic layers.

- Isotropic_dielectric_filters.py - dielectric thin film filter examples from freesnell

- Isotropic_metallic_filters.py - metallic thin film filter examples from freesnell

- Isotropic_incoherent_filters.py - incoherent effects example from freesnell.

- testing_Transfer_Matrix.py - models an absorbing layer using a Lorentz oscillator (compares with optical_plate.py)

- testing_Transfer_Matrix2.py - models a simple interface (compares with fresnel.py)

- testing_Transfer_Matrix3.py - models a uniaxial absorbing layer using a Lorentz oscillator (compares with uniaxial_plate.py)

- Anisotropic_ThinFilm_example.py - as it says.

- TIR.py - model total internal reflection

- Pettersson1999.py - Replicate some graphs from Pettersson's 1999 paper "Modeling photocurrent action spectra of photovoltaic devices based on organic thin films. Nb. resize the figures to get a better fit, I couldn't get the overlayed scales to work properly...

- Ohta1990.py - Replicates some graphs from Ohta's 1990 paper "Matrix Formulism for calculation of electric field intensity of light in stratified multilayered films"

Links

There are many other thin-film modelling programs and libraries to be found; and if you are only interested in layers with isotropic layers, many are more developped than this one

Free/Open Source

(many from S Byrne's manual)

- Openfilters- a userfriendly gui program with many advanced techniques for optimising optical filter designs.
- Freesnell- a command-line scheme program with many good examples of thin film filters.
- slabs
- TMMmode solver- matlab and python code.
- http://www.stanford.edu/group/mcgehee/transfermatrix/- matlab and python code
- EMpy- An electromagnetism python library, also includes anistropic transfer matrices and rigorous coupled wave analysis.
- Multilayer thin film optics calculator by S Byrnes- python and matlab code. Has a very good manual.
- refFIT- transfer matrix and ellipsometry*
- NKDstack
- lightmachinery optical-calculations- a page of optical calculatorsightmachinery
- openTMM- python module
- FilmStar(free version)
- Puma

Online

- Thinfilm- thinfilm calculator
- Luxpop- A very useful site of optical calculators and material refractive indices

Commercial

(many from Optalix - a very useful page of optical software)

- Essential Macleod- Thin film design and analysis software. Macleod also wrote a very important book on thin film filters.
- TFCalc- Software for designing and manufacturing optical thin film coatings.
- Optalix- Raytracing software with integrated thin film modelling.
- Setfos- Models active devices such as detectors and emitters.

- Film Wizard- Software for optimization and synthesis of optical thin film coatings. Other products are Film Tec 2000, Film Monitor, Film Ellipse.
- FilmStar A suite of Windows programs
- Multilayer
- OptiLayer
- Woollam- Ellipsometry software
- SemiconSoft- TFCompanion software, thin-film data analysis for ellipsometry, reflectance and transmittance measurements.
- RP coating- program from RPphotonics who also have an amazing and freely available encyclopedia on optics.

*Ellipsometry is a method of measuring a sample in order to analyse its structure. So it is almost the reverse of what we have been proposing. A sample has measured it's reflectivities in both polarisations as well as the relative phase difference between the polarisations; this might be done for a range of angles. Then starting from a guess of the sample's structure, a fitting procedure is used to find the layer thicknesses/refractive indices that best fit the measured data. As we might imagine, this is much harder to do well than a simple application of the transfer matrix method to a designed structure.