General information

This c++ library contains functions for calculation of the optical properties of multilayer thin film coatings. The current version is able to perform reflectivity, absorption and transmission calculations, as well as the calculation of ellipsometric parameters tg(Ψ) and cos(Δ). The calculation approach is based on transfer matrix formalism. The results of the calculation were cross-checked with the WinSpall software.
This library is included in the Opal program. The program is available on the GitHub under **https://github.com/mbiednov/opal/releases**
The reserved constants are $Pi = 3.14158$ and $condVac = 0.0027$. Additionally the class name *mycomplex* and the names for data structures *Ellipsometry* and *Period* are used.
The *Ellipsometry* data structure simply contains two members of type *double* to store tg(Ψ) and cos(Δ) values.
The *Period* data structure stores the data about the repeating sequence of thin films. The graphical representation of the term *Period* is given in fig. The inclusion of the *Period* was done in order to simplify working with periodical multilayer structures. A single layer can also be represented by a period, that includes one layer, repeated one time. This makes calculation functions that work with periods more general, compared to the functions that work with vectors.

Examples of usage

Here the examples of usage of all calculation function are provided. The angle of incidence has to be provided in degrees and the imaginary part of the refraction index (k) has to always be negative. Additionally, the thickness of first and last layer of the system has to be zero, because these layers are semi-infinite. The first layer can not be absorbing, meaning that k=0. The thickness of each layer and the wavelength of light must be provided in nanometers.

Basic reflectivity, transmission and absorption calculations

There are three basic functions, called *calculateReflectivity*, *calculateAbsorption* and *calculateTransmission* that accept the same set of arguments and return a value of the type double for the result. For example if we want to know the reflection of the p-polarized 632 nm radiation from the Air-Glass (n=1.5) interface, the input arguments for the calculation function have to be:

n	k	thickness
1.0	0	0
1.5	0	0

The *numberOfLayers* parameter in this case is 2. This parameter has to be provided to the function, so that it knows the length of n, k and *thickness* vectors. The corresponding c++ code for the function call would look like:

```
1 int main()
2 {
3         int N=2; //number of layers
4         double n[2] = {1, 1.5};
```

```
5           double k[2] = {0,0};
6           double h[2] = {0,0};   //thickness
7           double wl = 632; //wavelength
8           double angle = 10; //AOI in degrees
9
10          double result = 0;
11          result =
12          calculateReflectivity(N, n, k, h, angle, wl, true);
13
14          return 0;
15  }
```

The last input parameter *true* tells that the light is considered to be *p*-polarized. For the *s*-polarization the parameter has to be set *false*.

Calculation of ellipsometric parameters

Lets assume that we would like to know the tg(Ψ) and cos(Δ) values for the 100 nm thick SiO_2 layer on top of Si, in Air atmosphere. The code to calculate it would look like this:

```
1  int main()
2  {
3           int N=3; //number of layers
4           double n[3] = {1, 1.46, 3.88};
5           // note that k values are always negative
6           double k[3] = {0, 0, -0.0198};
7           double h[3] = {0, 100, 0};
8           double wl = 632; //wavelength
9           double angle = 70; //AOI in degrees
10
11          Ellipsometry result;
12          result =
13          calculateEllispometricValues(N, n, k, h, angle, wl);
14
15          return 0;
16  }
```

Note that in line 6, the value of the k for the Si substrate is negative.

Periodical structures

If we have to simulate the reflectivity of the periodic multilayer structure $S(L_1L_2L_3)^5(L_1L_3)M$, where S is the substrate layer, L_1, L_2 and L_3 are some layers that are being repeatedly included and M is the medium, we can use the overloaded *calculateReflectivity* function. As can be seen, there are two parts(periods) in our system. The first one is the five times repeated $L_1L_2L_3$ stack. The second part is simply the stack of two layers L_1L_3. The code to perform the calculation would look like this:

```
int main()
{
        //some realistic values for the real and imaginary
        //part of RI for each layer
        double nMedium{1.0};
        double nSubstr{1.5}, kSubstr{0};
        double n1{1.5}, n2{2.3}, n3{3.0};
        double h1{500}, h2{250}, h3{200};
        //initialization of both periods
        Period period1, period2;
        //first period
        int nL_p1=3; //number of layers in the period1
        double n_p1[3] = {n1, n2, n3};
        double k_p1[3] = {0, 0, 0};
        double h_p1[3] = {h1, h2, h3};
        //second period
        int nL_p2=2; //number of layers in the period2
        double n_p2[2]={n1, n3};
        double k_p2[2]={0, 0};
        double h_p2[2]={h1, h3};
        double wl = 632; //wavelength
        double angle = 0; //AOI in degrees
        //number of repetition within each period
        int nRep_p1=5;
        int nRep_p2=1;

        double result=0;
        period1=
        createPeriod(nL_p1,nRep_p1,nMedium,n_p1,k_p1,h_p1,angle,wl,tru
        period2=
        createPeriod(nL_p2,nRep_p2,nMedium,n_p2,k_p2,h_p2,angle,wl,tru
```

```
32
33          //now the vector with all periods
34          Period myPeriodicalSystem[2];
35          myPeriodicalSystem[0]=period1;
36          myPeriodicalSystem[1]=period2;
37
38          //now we can calculate reflectivity
39          result =
40          calculateReflectivity(myPeriodicalSystem, 2,
41           nMedium, nSubstr, kSubstr, angle, wl, true);
42
43
44          return 0;
45 }
```

We can now investigate with a couple of mouse clicks how the reflectivity of the system changes, when we change the number of inclusions of $L_1L_2L_3$ stack (nRep_p1 parameter in the code above).