Calcback

Nikolai Weidt

What is this?

I'm coding a program to get the complex refractive index $n = n * ik$ from the ellipsometric parameters Δ and Ψ I got from a simulation.

Imports:

```
import numpy as np
import matplotlib
from matplotlib import pyplot
```

Defining some variables:

Defining some variables for later use:

```
CSVFILE = "head300nmSiO2.csv"
phi_i = 70 * np.pi / 180
d_L = 300
n_air = 1
rerange = 5
imrange = 5
i = 0
```

Read .csv-file:

Read the values into a two dimensional numpy array as [[lambda,Psi,Delta,n_S,k_S],...] (Skip columns 3 and 4)

```
csv = np.loadtxt(CSVFILE, usecols=(0,1,2,5,6),  delimiter=",", skiprows=1)
```

:DEBUG: The array looks like this:

```
print(csv)
```

Calculate ρ

Create a matrix containing every possible refractive index (n+ik):

```
lsp_re = np.linspace(0.1, rerange, 101)
lsp_im = np.linspace(0.1, imrange, 101)
re, im = np.meshgrid (lsp_re, lsp_im, copy=False)
matrix = 1j * im + re
```

This gives the following matrix:

```
print(matrix)
```

Calculate:

```
n_S = (csv[i,3] + 1j* csv[i,4])
```

```
lambda_vac = csv[i,0]
for n_L in matrix.flat:
      # Snell's Law:
      phi_L = np.arcsin((np.sin(phi_i)*n_air)/ n_L)
      phi_S = np.arcsin((np.sin(phi_L)*n_L)/ n_S)
      # Fresnel equations:
      #
      # air/layer:
      rs_al = (n_air * np.cos(phi_i) - n_L * np.cos(phi_L)) / (n_air * np.cos(phi_i) + n_L * np.cos(phi_L))
      rp_al = (n_L * np.cos(phi_i) - n_air * np.cos(phi_L)) / (n_L * np.cos(phi_i) + n_air * np.cos(phi_L))
      # layer/substrate:
      rs_ls = (n_L * np.cos(phi_L) - n_S * np.cos(phi_S)) / (n_L * np.cos(phi_L) + n_S * np.cos(phi_S))
      rp_ls = (n_S * np.cos(phi_L) - n_L * np.cos(phi_S)) / (n_S * np.cos(phi_L) + n_L * np.cos(phi_S))
      # Fujiwara:
      beta = 2 * np.pi * d_L * n_L * np.cos(phi_L) / lambda_vac
      rp_L = (rp_al + rp_ls * np.exp(-2*1j*beta)) / (1 + rp_al * rp_ls * np.exp(-2 * 1j * beta))
      rs_L = (rs_al + rs_ls * np.exp(-2*1j*beta)) / ( 1 + rs_al * rs_ls * np.exp(-2 * 1j * beta))
      rho = rp_L / rs_L
      output = []
      output.append([n_L, rho])

for n_L =  (5+5j)
at lambda =  300.0
phi_L (0.09369049752311029-0.0942436309601521j)
phi_S (0.1516718935900151-0.1754940397472108j)
rs_al (-0.9322788656900732-0.06447800755339925j)
rp_al (0.47076999129408226+0.32915273622391117j)
rs_ls (0.2706645644366405-0.03780574370459625j)
rp_ls (-0.27413124901624036+0.021323198111731292j)
beta (31.139752412112067+31.69455000949363j)
rp_L (1.426723122645158-0.9975355870956931j)
rs_L (-1.067534044700266+0.07383248803644696j)
rho (-1.3944229215529675+0.8379890813445299j)
output [[(5+5j), (-1.3944229215529675+0.8379890813445299j)]]
```

Compare calculated rho with given Δ and ψ: