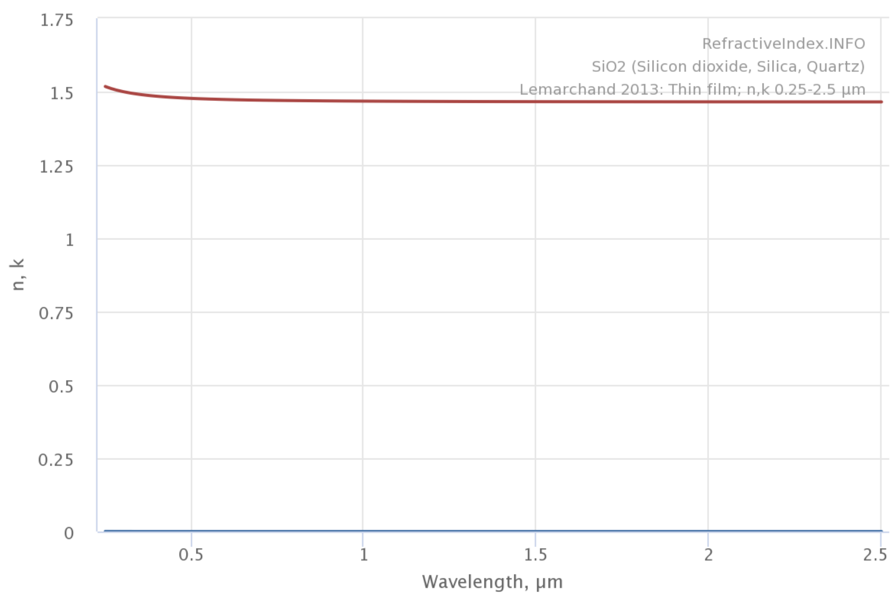
Calcback

Nikolai Weidt

What is this?

This is a script to get the complex refractive index $n = n * ik$ from the ellipsometric parameters Δ and Ψ I got from a simulation. The result for 300nm SiO_2 should look like this:



List of Todos:

TODO Write a loop for all wavelengths after it works for one.

TODO Then take even more wavelengths (rows)

Imports:

```
import numpy as np
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt
```

Defining some variables:

Defining some variables for later use:

```
CSVFILE = "300nmSiO2.csv"  # head300nmSiO2.csv = only 10 rows of data
phi_i = 70 * np.pi / 180  # converting incident angle from deg (first number) to rad
d_L = 300  # thickness of layer in nm
n_air = 1  # refractive index of air
rerange = 5  # upper limit for real part
imrange = 1  # upper limit for imaginary part
i = 0  # only look at one wavelength (row in csv)
```

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)
```

The array looks like this:

```
csv
```

Calculate

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

Change the last number in the "linspaces" to adjust the resolution.

```
lsp_re = np.linspace(1, rerange, 1001)
lsp_im = np.linspace(0.01, imrange, 1001)
re, im = np.meshgrid (lsp_re, lsp_im, copy=False)
n_L = 1j * np.round(im,6) + np.round(re,6)
n_L = n_L.flatten() # create onedimensional array
```

This gives the following matrix:

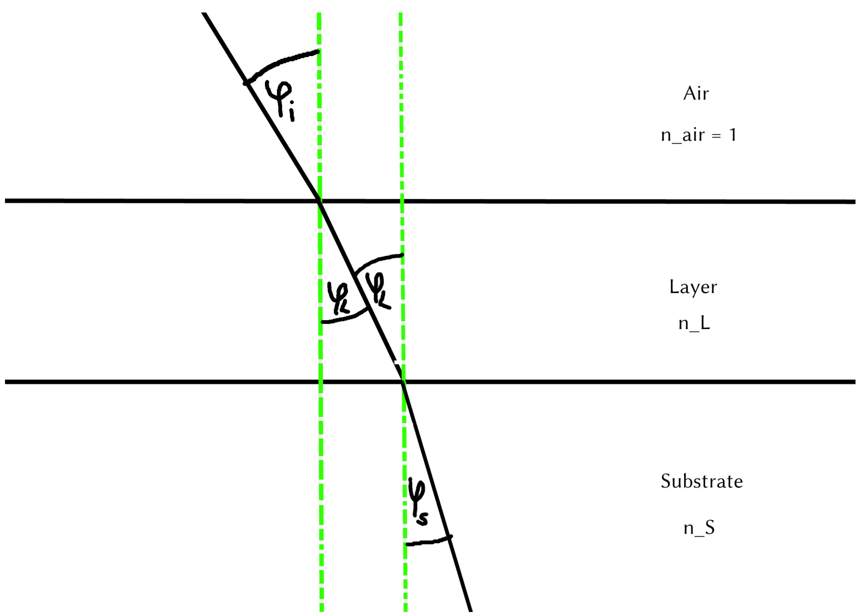
```
n_L
```

Calculate :

First we define some functions:

1. Snell's Law to calculate the refractive angles:

   Phi is the incident angle for the layer, n1 and n2 are refractive indices of first and second medium. Returns the angle of refraction.



```
def snell(phi, n1, n2):
  """Calculates the refractive angle, parameters are incident angle phi, refractive ind
  phi_ref = np.arcsin((n1/n2)*np.sin(phi))
  return phi_ref
```

2. Calculate r_p and r_s with Fresnel equations:

```
def fresnel(n1, phi1, n2, phi2):
    """Takes refractive indices and angles of two layers to calculate the amplitude ref
    rs = (n1 * np.cos(phi1) - n2 * np.cos(phi2)) / (n1 * np.cos(phi1) + n2 * np.cos(phi
    rp = (n2 * np.cos(phi1) - n1 * np.cos(phi2)) / (n2 * np.cos(phi1) + n1 * np.cos(phi
    return rs, rp
```

3. Calculate for the layer with eq. 5.2 in Spectroscopic Ellipsometry citenum FUJIWARA2009SPECTROSCOPIC

```
def calc_rho(rs_al, rp_al, rs_ls, rp_ls, d, n, phi, lambda_vac, returnbeta=False):
    beta = 2 * np.pi * d * n * np.cos(phi) / lambda_vac
    rp_L = (rp_al + rp_ls * np.exp(-2*1j*beta)) / (1 + rp_al * rp_ls * np.exp(-2 * 1j *
    rs_L = (rs_al + rs_ls * np.exp(-2*1j*beta)) / (1 + rs_al * rs_ls * np.exp(-2 * 1j *
    rho_L = rp_L / rs_L
    return rho_L
```

Then we call these functions one after another to calculate :

Get refractive index of the substrate (n_S) and lambda from the csv:

```
lambda_vac = csv[i][0]
n_S = (csv[i][3] + 1j * csv[i][4])
```

Then call the above defined functions

```
phi_L = snell(phi_i, n_air, n_L)
phi_S = snell(phi_L, n_L, n_S)
# Fresnel equations:
# air/layer:
rs_al, rp_al = fresnel(n_air, phi_i, n_L, phi_L)
# layer/substrate:
rs_ls, rp_ls = fresnel(n_L, phi_L, n_S, phi_S)

rho_L = calc_rho(rs_al, rp_al, rs_ls, rp_ls, d_L, n_L, phi_L, lambda_vac)
```

:DEBUG: nil:END:

Identify the best fitting rho with = tan( ) * $e^{i\Delta}$:

```
# psi is in our csv-file at index 1, delta at index 2 at row "i" for lambda
psi = csv[i][1] * (np.pi/180)
```

```
delta = csv[i][2] * (np.pi/180)
rho_giv = np.tan(psi) * np.exp(1j * delta)
diff = abs(rho_giv - rho_L)  # magnitude of complex number
idx = np.argmin(diff)  # index of the minimum
minimum = diff[idx]
n = n_L[idx]
print("At lambda = ", lambda_vac)
print("the layer has the refractive index n_L = " , n)
```

Plot some things for checking results:

If we use a high resolution, those plots are not showing much, thats why they are only showing the first 10000 values.

Plot Δ & Ψ:

Ψ in blue, Δ in red. Maybe (-1)?

```
fig = plt.figure()
plt.plot(csv[:,0],csv[:,1], 'b')
plt.plot(csv[:,0],csv[:,2], 'r')
plt.savefig("psidelta.png")
```

"psidelta.png"

Plot refractive index of substrate n_S:

Real part n in blue, imaginary part k in red

```
fig = plt.figure()
plt.plot(csv[:,0], csv[:,3], 'b')
plt.plot(csv[:,0], csv[:,4], 'r')
plt.savefig("ns.png")
```

"ns.png"

Plot real and imaginary part of the created n_L matrix:

Real part is blue, imaginary is red.

```
fig = plt.figure()
plt.plot(np.real(n_L[:10000]), c='b')
plt.plot(np.imag(n_L[:10000]), c="r")
plt.savefig('n_L.png')
```

'./n_L.png'

Plot real and imaginary part of $_{\mathrm{L}}$

```
fig = plt.figure()
plt.plot(np.real(rho_L), c='b')
plt.plot(np.imag(rho_L), c='r')
plt.savefig('rho_L.png')
"./rho_L.png"
```

Plot of the difference between $_{\mathrm{L}}$ and the given and determined minimum:

The difference is shown in blue, the red lines show the minimum.

```
fig = plt.figure()
plt.axvline(idx, c='r')
plt.axhline(minimum, c='r')
plt.plot(diff[:idx+10000])
plt.savefig('diff.png')
"./diff.png"
```

Plot refractive angle $\mathrm{phi}_{\mathrm{L}}$ and n_{L}:

n_{L} is shown in green, real part of $\mathrm{phi}_{\mathrm{L}}$ in blue, imaginary in red. A relation between these should be visible.

```
fig = plt.figure()
plt.plot(np.real(phi_L[:5000]), 'b')
plt.plot(np.imag(phi_L[:5000]), 'r')
plt.plot(np.real(n_L[:5000]), c='g')
plt.savefig('phi_L.png')
"phi_L.png"
```

Testing:

Testing with constant n_{L}, $\mathrm{phi}_{\mathrm{i}}$ at i=0

```
[("n_L[0]",n_L[0]),("phi_i",phi_i)]
```

snell():

```
phi_Ltest = snell(phi_i, n_air, n_L[0])
phi_Ltest
```

should be: (1.220429-0.02737074 i)

```
("n_S",n_S)
```

```
phi_Stest = snell(1.220429-0.0273775j,n_L[0],n_S)
phi_Stest
```

should be: (0.151671-0.175494i)

fresnel():

rs_{al}, rp_{al} = fresnel(n_{air}, phi_i, n_L, phi_L)

rs_{ls}, rp_{ls} = fresnel(n_L, phi_L, n_S, phi_S)

```
rs_altest, rp_altest = fresnel(n_air, phi_i, n_L[0], phi_Ltest)
rs_altest
```

should be: (-0.003398-0.04239i)

```
rp_altest
```

should be:

```
rs_lstest, rp_lstest = fresnel(n_L[0], phi_Ltest, n_S, phi_Stest)
rs_lstest
```

```
rp_lstest
```

$calc_{rho}$():

rho_L = $calc_{rho}$(rs_{al}, rp_{al}, rs_{ls}, rp_{ls}, d_L, n_L, $lambda_{vac}$) Just copied this from above with beta returned

```
def calc_rhotest(rs_al, rp_al, rs_ls, rp_ls, d, n, phi, lambda_vac):
    beta = 2 * np.pi * d * n * np.cos(phi) / 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_L = rp_L / rs_L
    return rho_L, beta
```

```
rhotest, betatest = calc_rhotest(rs_altest, rp_altest, rs_lstest, rp_lstest, 300, n_L[0], ph
betatest
```

should be: 2.1558487+0.18312240i

```
rhotest
```

bibliography:forschungspraktikum.bib