layerstack demo

February 15, 2019

0.1 # Transfer matrix calculation for thinfilm stacks

- calculate reflectance and transmittance
- both symbolic **and** numeric

0.1.1 Assumptions (for now)

- coherent
- normal incidence, AOI = 0ř

```
In [45]: import sympy as sp
         sp.init_printing()
```

```
In [46]: from layerstack.stack import *
```

```
In [47]: air = Material("air", refractive_index_value=1)
         cSi = Material("cSi")
```

```
In [48]: layers = [Layer("air", air),
                   Layer("bulk", cSi),
                   Layer("air", air)]

         lstack = Layerstack(layers)
```

0.1.2 Reflectance

```
In [49]: lstack.reflectance_amplitude()
```

```
Out[49]:
```

$$\frac{\frac{\left(-\frac{n_{cSi}}{2}-\frac{1}{2}\right)(n_{cSi}-1)e^{\frac{2i\pi d_{bulk}n_{cSi}}{\lambda vac}}}{2n_{cSi}}+\frac{\left(\frac{n_{cSi}}{2}-\frac{1}{2}\right)(n_{cSi}+1)e^{-\frac{2i\pi d_{bulk}n_{cSi}}{\lambda vac}}}{2n_{cSi}}}{\frac{\left(-\frac{n_{cSi}}{2}-\frac{1}{2}\right)(n_{cSi}+1)e^{-\frac{2i\pi d_{bulk}n_{cSi}}{\lambda vac}}}{2n_{cSi}}+\frac{\left(\frac{n_{cSi}}{2}-\frac{1}{2}\right)(n_{cSi}-1)e^{\frac{2i\pi d_{bulk}n_{cSi}}{\lambda vac}}}{2n_{cSi}}}$$

```
In [50]: sp.simplify(_)
```

Out[50]:

$$\frac{(n_{cSi}-1)(n_{cSi}+1)\left(-e^{\frac{4i\pi d_{bulk}n_{cSi}}{\lambda_{vac}}}+1\right)}{(n_{cSi}-1)^2 e^{\frac{4i\pi d_{bulk}n_{cSi}}{\lambda_{vac}}}-(n_{cSi}+1)^2}$$

```
In [51]: lstack.reflectance_amplitude().free_symbols
```

Out[51]:

$$\{d_{bulk}, \lambda_{vac}, n_{cSi}\}$$

0.1.3 Add an ARC

```
In [52]: thickness_ARC = sp.Symbol('d_ARC')
         thickness_bulk = sp.Symbol('d_bulk')
```

```
In [53]: SiNx = Material("SiNx")

         layers = [Layer("air", air),
                   Layer("ARC", SiNx, thickness_ARC),
                   Layer("bulk", cSi, thickness_bulk),
                   Layer("air", air)]
```

```
In [54]: lstack_with_ARC = Layerstack(layers)
         reflectance = lstack_with_ARC.reflectance_amplitude()
         sp.simplify(reflectance)
```

Out[54]:

$$\frac{\left((n_{cSi}-1)\left((n_{SiNx}-1)(n_{SiNx}-n_{cSi})-(n_{SiNx}+1)(n_{SiNx}+n_{cSi})e^{\frac{4i\pi d_{ARC}n_{SiNx}}{\lambda_{vac}}}\right)e^{\frac{2i\pi(d_{ARC}n_{SiNx}+2d_{bulk}n_{cSi})}{\lambda_{vac}}}+(n_{cSi}+1)\left(\right.\right.}{(n_{cSi}-1)\left((n_{SiNx}-1)(n_{SiNx}+n_{cSi})e^{\frac{4i\pi d_{ARC}n_{SiNx}}{\lambda_{vac}}}-(n_{SiNx}+1)(n_{SiNx}-n_{cSi})\right)e^{\frac{4i\pi d_{bulk}n_{cSi}}{\lambda_{vac}}}+(n_{cS}}$$

```
In [55]: lstack_with_ARC.reflectance_amplitude().free_symbols
```

Out[55]:

$$\{d_{ARC}, d_{bulk}, \lambda_{vac}, n_{SiNx}, n_{cSi}\}$$

0.1.4 Introduce numbers

```
In [56]: RI_substitutions = [
             (SiNx.n_symbol, 1.804),
             (cSi.n_symbol, 4.217 + 0.038j)
         ]
```

units need to be consistent
here use ţm

```
In [57]: length_substitutions = [
             (LAMBDA_VAC, 515e-3),
             (thickness_ARC, 74e-3),
             (thickness_bulk, 200)
         ]
```

0.1.5 Calculate actual values

substitute symbols with numbers

```
In [58]: reflectance_num = reflectance.subs(RI_substitutions).subs(length_substitutions)
```

Reflectance from complex amplitude

```
In [59]: sp.sqrt(sp.N(sp.conjugate(reflectance_num) * reflectance_num))
```

```
Out[59]:
```

$$0.137353521726773$$

0.1.6 Build functions

```
In [60]: x = sp.Symbol('x')
         wavelength = 515e-3

         RI_substitutions = [
             (SiNx.n_symbol, 1.804),
             (cSi.n_symbol, 4.217 + 0.038j)
         ]
         length_substitutions = [
             (LAMBDA_VAC, wavelength),
             (thickness_ARC, x), # <---- use Symbol instead of value
             (thickness_bulk, 200)
         ]
```

```
In [61]: R = sp.sqrt(sp.conjugate(reflectance) * reflectance)

         reflectance_of_x = R.subs(RI_substitutions).subs(length_substitutions)
```

```
In [62]: reflectance_of_x.free_symbols
```

```
Out[62]:
```

$$\{x\}$$

0.1.7 Lambdify!

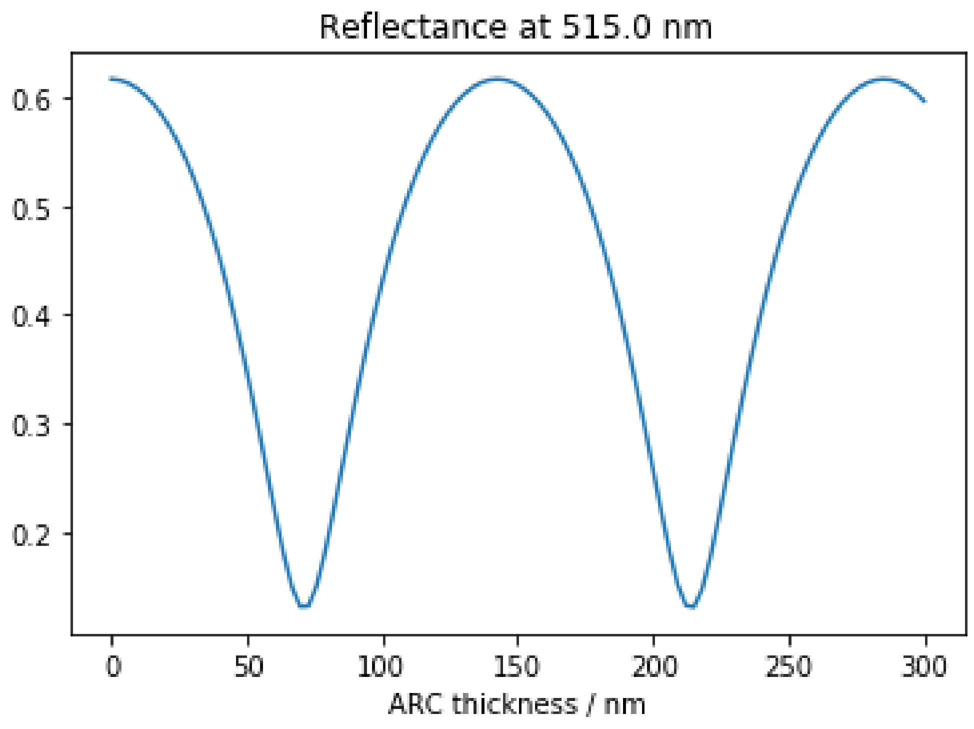
```
In [63]: import numpy

         reflectance_of_x_function = sp.lambdify(x, reflectance_of_x, numpy)

In [67]: import matplotlib.pyplot as plt

         x_values = numpy.linspace(0, 300e-3, num=100)
         y_values = reflectance_of_x_function(x_values)

         plt.plot(x_values*1e3, y_values)
         plt.xlabel("ARC thickness / nm")
         plt.title("Reflectance at %s nm" % (wavelength*1e3))
         plt.show()
```


Reflectance at 515.0 nm
ARC thickness / nm