Multilayer optical calculations

The Transfer Matrix Method

N. Fressengeas

Laboratoire Matériaux Optiques, Photonique et Systèmes
Unité de Recherche commune à l'Université de Lorraine et à CentraleSupélec

1. Physics behind the software
   - Wave propagation theory
   - Single-interface reflection and transmission amplitudes
   - Multilayer thin films
2. The `tmm` module
   - Setting `tmm` up
   - Example 1 : thin non-absorbing layer, on top of a thick absorbing layer
   - Example 2 : here is an example where we plot absorption and Poynting vector as a function of depth

Many thanks to Steve

A class based on Steve Byrnes paper for the Transfer Matrix Method `tmm` python module

Multilayer optical calculations

Steven J. Byrnes

Current affiliation: Charles Stark Draper Laboratory, Cambridge, Massachusetts, USA

Contact: steven.byrnes@gmail.com

September 8, 2020

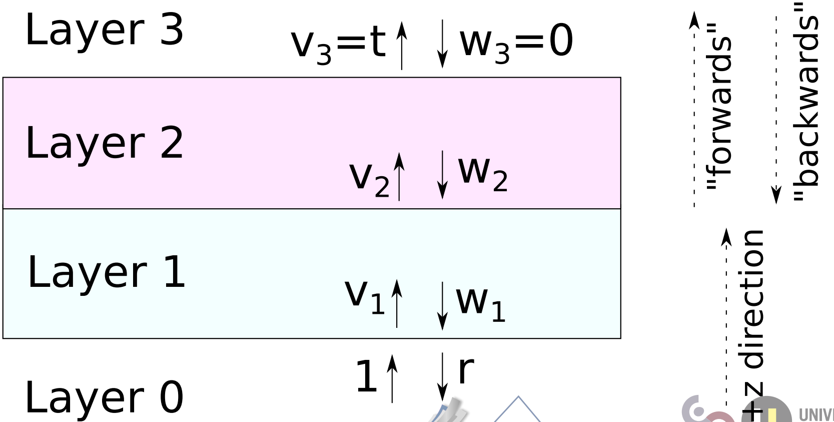
Abstract

When light hits a multilayer planar stack, it is reflected, refracted, and absorbed in a way that can be derived from the Fresnel equations. The analysis is treated in many textbooks, and implemented in many software programs, but certain aspects of it are difficult to find explicitly and consistently worked out in the literature. Here, we derive the formulas underlying the transfer-matrix method of calculating the optical properties of these stacks, including oblique-angle incidence, absorption-vs-position profiles, and ellipsometry parameters. We discuss and explain some strange consequences of the formulas in the situation where the incident and/or final (semi-infinite) medium are absorptive, such as calculating $T > 1$ in the absence of gain. We also discuss some

1. Physics behind the software
   - Wave propagation theory
   - Single-interface reflection and transmission amplitudes
   - Multilayer thin films
2. The `tmm` module
   - Setting `tmm` up
   - Example 1 : thin non-absorbing layer, on top of a thick absorbing layer
   - Example 2 : here is an example where we plot absorption and Poynting vector as a function of depth

Wave propagation in a multilayer stack

Assumption : everything occurs in the *xz* plane



Optical wave electric field

Waves have two components : a *forward* one and a *backward* one

Sum of forward and backward traveling waves

$$\mathbf{E}(\mathbf{r}) = \mathbf{E}_f^0 e^{i\mathbf{k}_f\cdot\mathbf{r}} + \mathbf{E}_b^0 e^{i\mathbf{k}_b\cdot\mathbf{r}}$$

- $\mathbf{E}_f^0$ and $\mathbf{E}_b^0$ are constants
- $\mathbf{k}_f$ and $\mathbf{k}_b$ are wave vectors lying in the xz plane

Optical wave electric field

Waves have two components : a *forward* one and a *backward* one

Sum of forward and backward traveling waves

$$\mathbf{E}(\mathbf{r}) = \mathbf{E}_f^0 e^{i\mathbf{k}_f\cdot\mathbf{r}} + \mathbf{E}_b^0 e^{i\mathbf{k}_b\cdot\mathbf{r}}$$

- $\mathbf{E}_f^0$ and $\mathbf{E}_b^0$ are constants
- $\mathbf{k}_f$ and $\mathbf{k}_b$ are wave vectors lying in the xz plane

Complex wave vector, or complex refractive index

to take absorption and attenuation into account

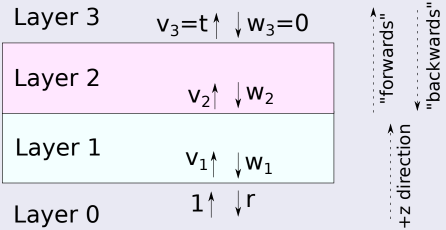
Usual link between refractive index and wave vector

$$\mathbf{k}_f = \frac{2\pi n}{\lambda_{vac}}(\hat{\mathbf{z}}\cos\theta + \hat{\mathbf{x}}\sin\theta) \qquad \mathbf{k}_b = \frac{2\pi n}{\lambda_{vac}}(-\hat{\mathbf{z}}\cos\theta + \hat{\mathbf{x}}\sin\theta)$$

Complex refractive index

- $\Re(n)$ is usual refractive index
- $\frac{2\pi\Im(n)}{\lambda_{vac}}$ is absorption coefficient

Multilayer stack



s and p polarizations

$$\mathbf{H} = \frac{1}{\mu_0\omega}\mathbf{k} \times \mathbf{E}$$

s-polarization: $\mathbf{E} \parallel \hat{\mathbf{y}}$

- $\mathbf{E}_f = E_f\hat{\mathbf{y}}$
- $\mathbf{E}_b = E_b\hat{\mathbf{y}}$
- $\mathbf{H}_f \propto nE_f\left(-\cos\theta\hat{\mathbf{x}} + \sin\theta\hat{\mathbf{z}}\right)$
- $\mathbf{H}_b \propto nE_b\left(\cos\theta\hat{\mathbf{x}} + \sin\theta\hat{\mathbf{z}}\right)$

p-polarization: $\mathbf{H} \parallel \hat{\mathbf{y}}$

- $\mathbf{E}_f = E_f(-\sin\theta\hat{\mathbf{z}} + \cos\theta\hat{\mathbf{x}})$
- $\mathbf{E}_b = E_b(-\sin\theta\hat{\mathbf{z}} - \cos\theta\hat{\mathbf{x}})$
- $\mathbf{H}_f \propto nE_f\hat{\mathbf{y}}$
- $\mathbf{H}_b \propto nE_b\hat{\mathbf{y}}$

1. Physics behind the software
   - Wave propagation theory
   - Single-interface reflection and transmission amplitudes
   - Multilayer thin films
2. The `tmm` module
   - Setting `tmm` up
   - Example 1 : thin non-absorbing layer, on top of a thick absorbing layer
   - Example 2 : here is an example where we plot absorption and Poynting vector as a function of depth

Single-interface reflection and transmission amplitudes

Reflection and refraction when traveling from medium 1 to medium 2

$$r_s = \frac{n_1 \cos\theta_1 - n_2 \cos\theta_2}{n_1 \cos\theta_1 + n_2 \cos\theta_2} \qquad r_p = \frac{n_2 \cos\theta_1 - n_1 \cos\theta_2}{n_2 \cos\theta_1 + n_1 \cos\theta_2}$$

$$t_s = \frac{2 n_1 \cos\theta_1}{n_1 \cos\theta_1 + n_2 \cos\theta_2} \qquad t_p = \frac{2 n_1 \cos\theta_1}{n_2 \cos\theta_1 + n_1 \cos\theta_2}$$

If light is coming from both sides at the same time

Linearity allows to derive the relationships

Single-interface reflection and transmission amplitudes

Reflection and refraction when traveling from medium 1 to medium 2

$$r_s = \frac{n_1 \cos\theta_1 - n_2 \cos\theta_2}{n_1 \cos\theta_1 + n_2 \cos\theta_2} \qquad r_p = \frac{n_2 \cos\theta_1 - n_1 \cos\theta_2}{n_2 \cos\theta_1 + n_1 \cos\theta_2}$$

$$t_s = \frac{2 n_1 \cos\theta_1}{n_1 \cos\theta_1 + n_2 \cos\theta_2} \qquad t_p = \frac{2 n_1 \cos\theta_1}{n_2 \cos\theta_1 + n_1 \cos\theta_2}$$

If light is coming from both sides at the same time

Linearity allows to derive the relationships

Interface mix

$$v_{n+1} = (v_n e^{i\delta_n}) t_{n,n+1} + w_{n+1} r_{n+1,n}$$
$$w_n e^{-i\delta_n} = w_{n+1} t_{n+1,n} + (v_n e^{i\delta_n}) r_{n,n+1}$$

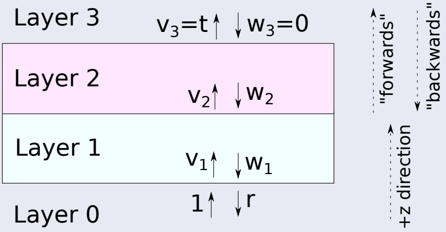
Linear relationship

$$\begin{pmatrix} v_n \\ w_n \end{pmatrix} = M_n \begin{pmatrix} v_{n+1} \\ w_{n+1} \end{pmatrix}$$

$r_{a,b} = -r_{b,a}$ $\quad t_{a,b} t_{b,a} - r_{a,b} r_{b,a} = 1$

$$M_n = \begin{pmatrix} e^{-i\delta_n} & 0 \\ 0 & e^{i\delta_n} \end{pmatrix} \begin{pmatrix} 1 & r_{n,n+1} \\ r_{n,n+1} & 1 \end{pmatrix} \frac{1}{t_{n,n+1}}$$

Multilayer stack



As simple as a matrix product

- Input wave, reflected wave and transmitted waves are linked through a matrix product
- As well as waves inside the stack

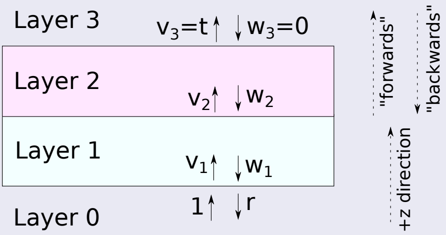
Interface mix

$$
\begin{aligned}
v_{n+1} &= (v_n e^{i\delta_n}) t_{n,n+1} + w_{n+1} r_{n+1,n} \\
w_n e^{-i\delta_n} &= w_{n+1} t_{n+1,n} + (v_n e^{i\delta_n}) r_{n,n+1}
\end{aligned}
$$

Linear relationship

$$
\begin{pmatrix} v_n \\ w_n \end{pmatrix} = M_n \begin{pmatrix} v_{n+1} \\ w_{n+1} \end{pmatrix}
$$

Multilayer stack



$r_{a,b} = -r_{b,a}$ $\qquad$ $t_{a,b} t_{b,a} - r_{a,b} r_{b,a} = 1$

$$
M_n = \begin{pmatrix} e^{-i\delta_n} & 0 \\ 0 & e^{i\delta_n} \end{pmatrix} \begin{pmatrix} 1 & r_{n,n+1} \\ r_{n,n+1} & 1 \end{pmatrix} \frac{1}{t_{n,n+1}}
$$

As simple as a matrix product

- Input wave, reflected wave and transmitted waves are linked through a matrix product
- As well as waves inside the stack

Interface mix

$$v_{n+1} = (v_n e^{i\delta_n}) t_{n,n+1} + w_{n+1} r_{n+1,n}$$
$$w_n e^{-i\delta_n} = w_{n+1} t_{n+1,n} + (v_n e^{i\delta_n}) r_{n,n+1}$$

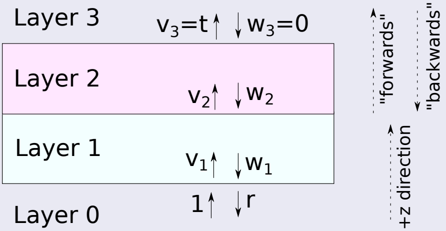
Linear relationship

$$\begin{pmatrix} v_n \\ w_n \end{pmatrix} = M_n \begin{pmatrix} v_{n+1} \\ w_{n+1} \end{pmatrix}$$

$r_{a,b} = -r_{b,a}$ $\quad$ $t_{a,b}t_{b,a} - r_{a,b}r_{b,a} = 1$

$$M_n \equiv \begin{pmatrix} e^{-i\delta_n} & 0 \\ 0 & e^{i\delta_n} \end{pmatrix} \begin{pmatrix} 1 & r_{n,n+1} \\ r_{n,n+1} & 1 \end{pmatrix} \frac{1}{t_{n,n+1}}$$

Multilayer stack



As simple as a matrix product

- Input wave, reflected wave and transmitted waves are linked through a matrix product
- As well as waves inside the stack

Interface mix

$$\begin{aligned} v_{n+1} &= (v_n e^{i\delta_n}) t_{n,n+1} + w_{n+1} r_{n+1,n} \\ w_n e^{-i\delta_n} &= w_{n+1} t_{n+1,n} + (v_n e^{i\delta_n}) r_{n,n+1} \end{aligned}$$

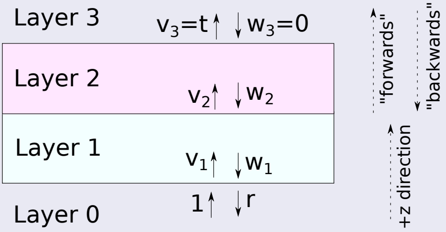
Linear relationship

$$\begin{pmatrix} v_n \\ w_n \end{pmatrix} = M_n \begin{pmatrix} v_{n+1} \\ w_{n+1} \end{pmatrix}$$

$r_{a,b} = -r_{b,a}$ $\quad$ $t_{a,b} t_{b,a} - r_{a,b} r_{b,a} = 1$

$$M_n \equiv \begin{pmatrix} e^{-i\delta_n} & 0 \\ 0 & e^{i\delta_n} \end{pmatrix} \begin{pmatrix} 1 & r_{n,n+1} \\ r_{n,n+1} & 1 \end{pmatrix} \frac{1}{t_{n,n+1}}$$

Multilayer stack



As simple as a matrix product

- Input wave, reflected wave and transmitted waves are linked through a matrix product
- As well as waves inside the stack

Calculating Poynting vector

Understanding the power flow in the layer

$$\mathbf{S}\cdot\hat{\mathbf{z}} = \frac{1}{2}\Re[\hat{\mathbf{z}}\cdot(\mathbf{E}^*\times\mathbf{H})]$$

s-polarization

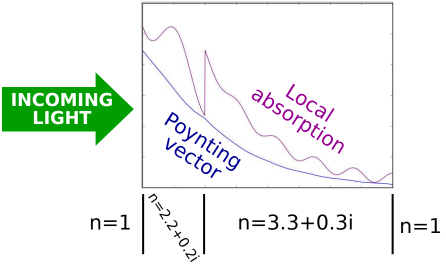
$$\mathbf{S}\cdot\hat{\mathbf{z}} = \frac{\Re\left[(n)(\cos\theta)(E_f^*+E_b^*)(E_f-E_b)\right]}{\Re[n_0\cos\theta_0]}$$

p-polarization

$$\mathbf{S}\cdot\hat{\mathbf{z}} = \frac{\Re\left[(n)(\cos\theta^*)(E_f+E_b)(E_f^*-E_b^*)\right]}{\Re\left[n_0\cos\theta_0^*\right]}$$

Local absorption

Local absorption is the negative z derivative of the above



Calculating Poynting vector

Understanding the power flow in the layer

$$\mathbf{S}\cdot\hat{\mathbf{z}} = \frac{1}{2}\Re[\hat{\mathbf{z}}\cdot(\mathbf{E}^*\times\mathbf{H})]$$

s-polarization

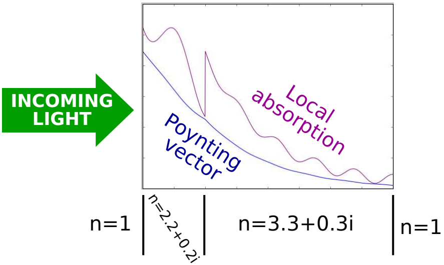
$$\mathbf{S}\cdot\hat{\mathbf{z}} = \frac{\Re\left[(n)(\cos\theta)(E_f^*+E_b^*)(E_f-E_b)\right]}{\Re[n_0\cos\theta_0]}$$

p-polarization

$$\mathbf{S}\cdot\hat{\mathbf{z}} = \frac{\Re\left[(n)(\cos\theta^*)(E_f+E_b)(E_f^*-E_b^*)\right]}{\Re\left[n_0\cos\theta_0^*\right]}$$

Local absorption

Local absorption is the negative z derivative of the above



1. Physics behind the software
   - Wave propagation theory
   - Single-interface reflection and transmission amplitudes
   - Multilayer thin films
2. The `tmm` module
   - Setting `tmm` up
   - Example 1 : thin non-absorbing layer, on top of a thick absorbing layer
   - Example 2 : here is an example where we plot absorption and Poynting vector as a function of depth

When everything else fails, read the instructions.
https://pythonhosted.org/tmm/

1 Introduction

Written by Steve Byrnes, originally in 2012. Please email me any feedback. My website is http://sjbyrnes.com, and the code lives at https://github.com/sbyrnes321/tmm.

This is a group of programs written in Python / NumPy for simulating light propagation in planar multilayer thin films, including the effects of multiple internal reflections and interference, using the "Transfer Matrix Method". It can also simulate combinations of thin and thick films (e.g. a thick piece of glass with a multi-layer antireflection coating on one side and a mirror on the other side), or purely thick films.

In addition to calculating how much light is transmitted and reflected, the program can calculate, at any given point in the structure, how much light is being absorbed there. This is a very important feature for solar-cell modeling, for example.

It can also calculate the parameters measured in ellipsometry.

I wrote out derivations and discussions of the formulas and calculations implemented by this program at: http://arxiv.org/abs/1603.02720. You are encouraged to cite that paper if you publish results that come from this software. :-)

2 Files and API

There are four files: (1) tmm_core.py contains all the main programs, (2) color.py has additional color-related functions, like calculating the RGB color of a thin film under re-

1. Physics behind the software
   - Wave propagation theory
   - Single-interface reflection and transmission amplitudes
   - Multilayer thin films

2. The tmm module
   - Setting tmm up
   - Example 1 : thin non-absorbing layer, on top of a thick absorbing layer
   - Example 2 : here is an example where we plot absorption and Poynting vector as a function of depth

Installation

Open linux command line or anaconda python installer

```
pip install tmm
```

or

```
pip3 install tmm
```

Where to start ?

Find tmm source files

```
import tmm
tmm.__file__
```

Find the documentation

- Directly in the source code
- https://pythonhosted.org/tmm/tmm.html

Where to start ?

Find tmm source files

```
import tmm
tmm.__file__
```

Find the documentation

- Directly in the source code
- https://pythonhosted.org/tmm/tmm.html

1. Physics behind the software
   - Wave propagation theory
   - Single-interface reflection and transmission amplitudes
   - Multilayer thin films

2. The tmm module
   - Setting tmm up
   - Example 1 : thin non-absorbing layer, on top of a thick absorbing layer
   - Example 2 : here is an example where we plot absorption and Poynting vector as a function of depth

Imports

```
from tmm import (coh_tmm, unpolarized_RT)
#coh_tmm :  the transfer-matrix-method calculation in the
    coherent case (i.e. thin films)
#unpolarized_RT : Calculates reflected and transmitted power
     for unpolarized light.

from numpy import pi, linspace, inf, array
from scipy.interpolate import interp1d
import matplotlib.pyplot as plt
%matplotlib inline
```

Setup

Really small deal

```
try:
import colorpy.illuminants
import colorpy.colormodels
from tmm import color
colors_were_imported = True
except ImportError:
# without colorpy, you can't run sample5(), but everything
    else is fine.
colors_were_imported = False


# "5 * degree" is 5 degrees expressed in radians
# "1.2 / degree" is 1.2 radians expressed in degrees
degree = pi/180
```

Thin non-absorbing layer, on top of a thick absorbing layer

with air on both sides

```
# list of layer thicknesses in nm
d_list = [inf,100,300,inf]
# list of refractive indices
n_list = [1,2.2,3.3+0.3j,1]
```

Initialize calculations

```
# list of wavenumbers to plot in nm^-1
ks=linspace(0.0001,.01,num=400)
# initialize lists of y-values to plot
Rnorm=[] # for normal incidence
R45=[] # for 45 degree incidence
```

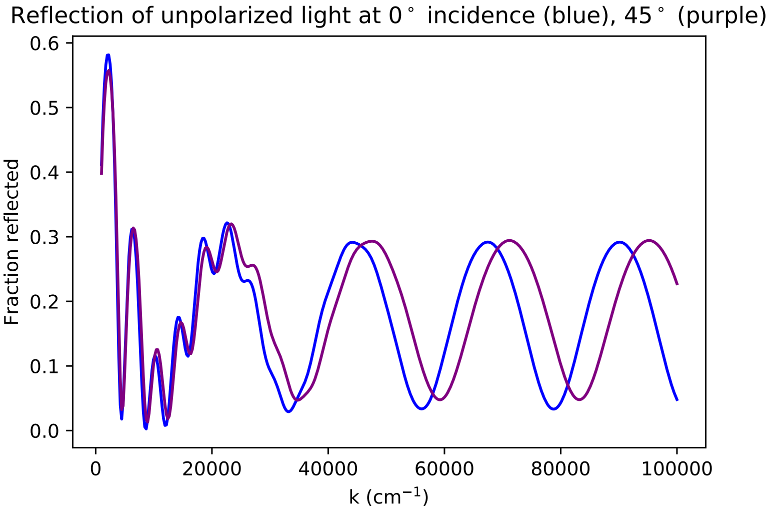
Run the actual calculations

```
for k in ks:
        # For normal incidence, s and p polarizations are
          identical.
        # I arbitrarily decided to use 's'.
        Rnorm.append(coh_tmm('s',n_list, d_list, 0, 1/k)['R'
          ]) #for polarized or unpolarized light
        # 's' is polarization, could be 'p'
        # n_list and d_list are  material indices and
          thicknesses in the order in which light passes
          through them
        # 0 is incidence angle in radian : here, normal
          indidence
        # 1/k is the wavelength to plot
        # REMEMBER : KEEP LENGTH UNITS CONSISTANT
        # ['R'] selects the reflected power in a dictionnary
           (among r,t,R,T...)
        R45.append(unpolarized_RT(n_list, d_list, 45*degree,
           1/k)['R'])
        # same for 45 degree incident unpolarized light
```

Plot the results

```
kcm = ks * 1e7 #ks in cm^-1 rather than nm^-1
plt.figure()
plt.plot(kcm,Rnorm,'blue',kcm,R45,'purple')
plt.xlabel('k␣(cm$^{-1}$)')
plt.ylabel('Fraction␣reflected')
plt.title('Reflection␣of␣unpolarized␣light␣at␣0$^\circ$␣
    incidence␣(blue),␣'
'45$^\circ$␣(purple)');
```

Output


Reflection of unpolarized light at 0 ° incidence (blue), 45 ° (purple)
Fraction reflected
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
20000
40000
60000
80000
100000
k (cm^{-1})

1. Physics behind the software
   - Wave propagation theory
   - Single-interface reflection and transmission amplitudes
   - Multilayer thin films

2. The tmm module
   - Setting tmm up
   - Example 1 : thin non-absorbing layer, on top of a thick absorbing layer
   - Example 2 : here is an example where we plot absorption and Poynting vector as a function of depth

Initialization

Two different absorbing layers ; incident light at a 45° angle

```
d_list = [inf, 100, 300, inf] #in nm
n_list = [1, 2.2+0.2j, 3.3+0.3j, 1]
th_0=pi/4
lam_vac=400
pol='p'
```

Computation

Compute all the needed data ; then inspect within the structure

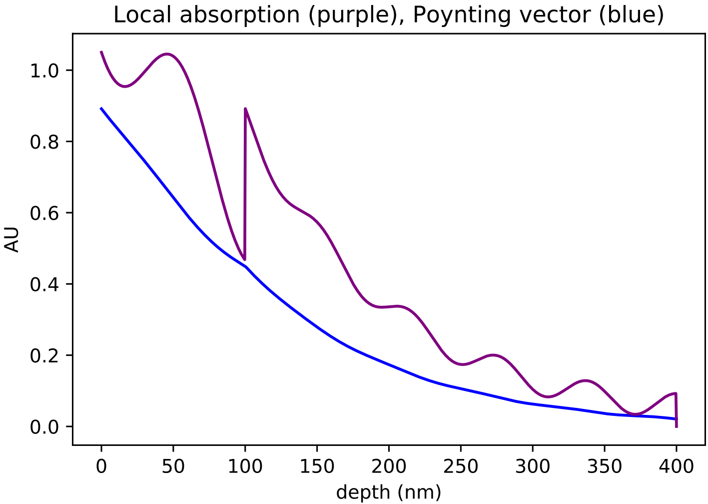
```
coh_tmm_data = coh_tmm(pol,n_list,d_list,th_0,lam_vac)
# This computes all the needed data
ds = linspace(0,400,num=1000) #position in structure
poyn=[]
absor=[]
for d in ds:
        layer, d_in_layer = find_in_structure_with_inf(
            d_list,d)
        data=position_resolved(layer,d_in_layer,coh_tmm_data
            )
        poyn.append(data['poyn'])
        absor.append(data['absor'])
```

Plot

Convert to numpy arrays for plotting both the power density and the local absorption

```
# convert data to numpy arrays for easy scaling in the plot
poyn = array(poyn)
absor = array(absor)
plt.figure()
plt.plot(ds,poyn,'blue',ds,200*absor,'purple')
plt.xlabel('depth␣(nm)')
plt.ylabel('AU')
plt.title('Local␣absorption␣(purple),␣Poynting␣vector␣(blue)
    ');
```

Output


Local absorption (purple), Poynting vector (blue)
AU
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300
350
400
depth (nm)