Computational Photonics

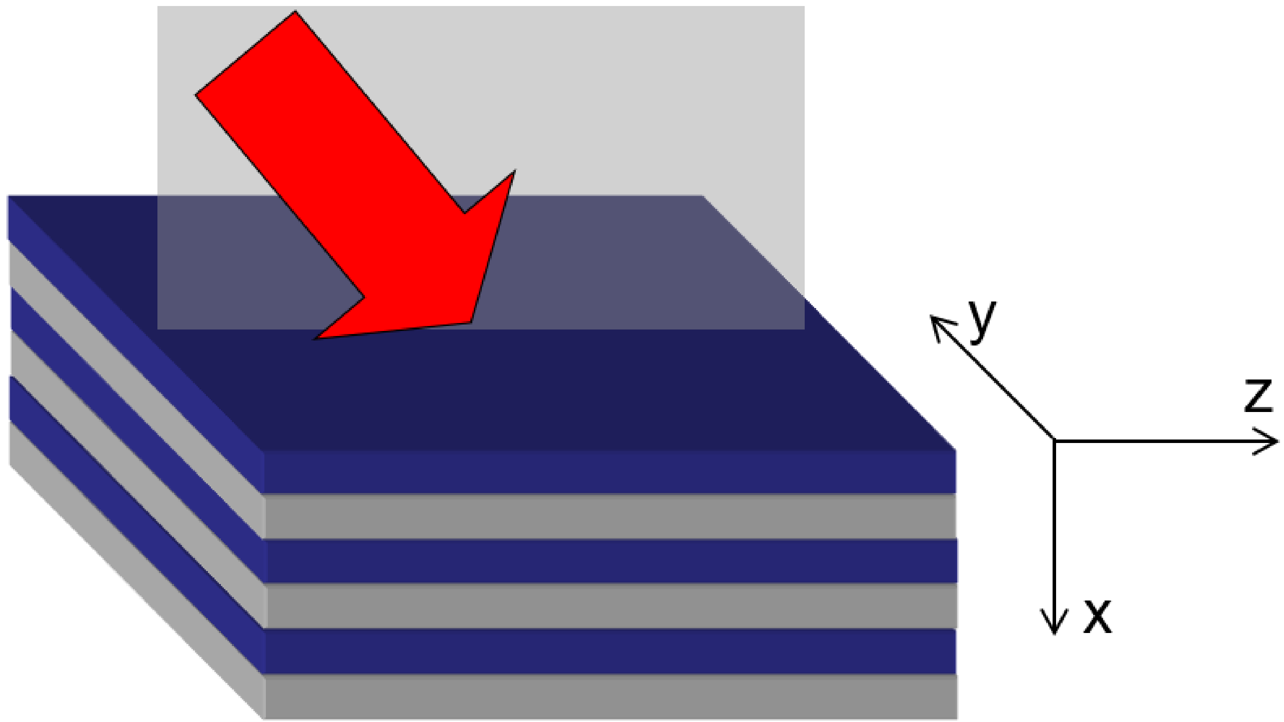
Seminar 02

Implementation of the Matrix Method

- calculation of the transfer matrix
- calculation of reflection and transmission characteristics of stratified media
- calculation of fields inside layers

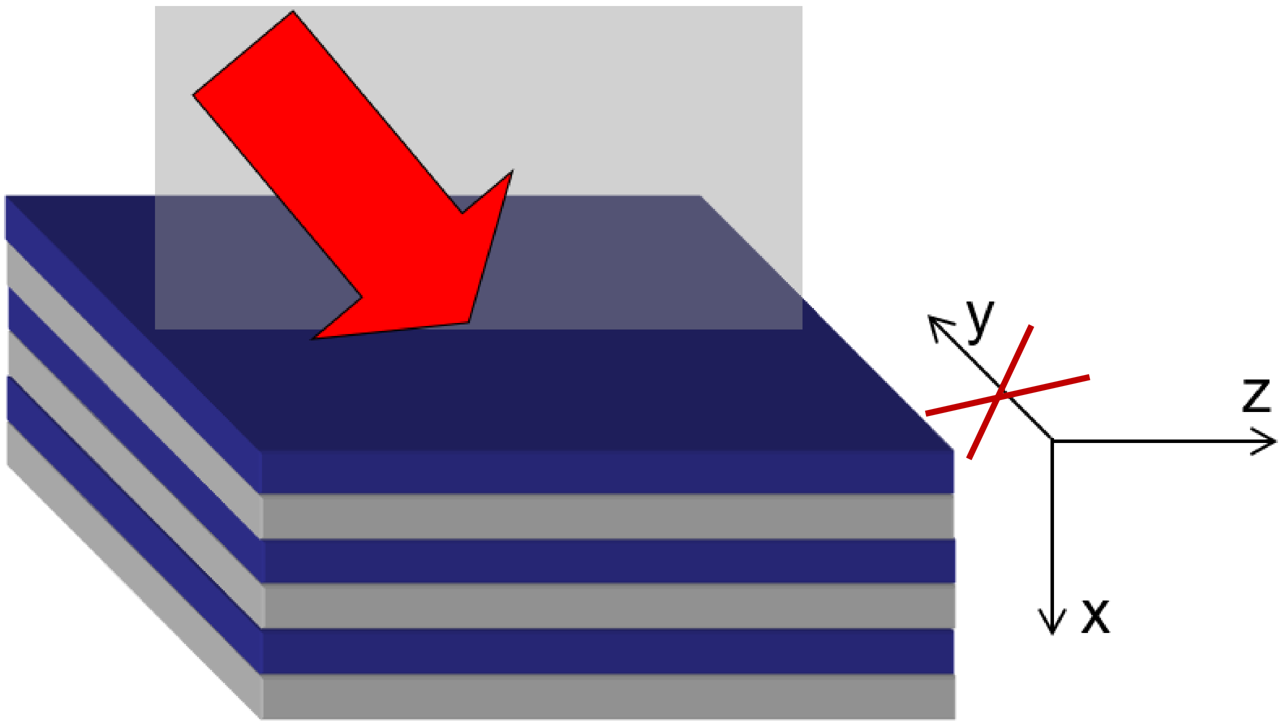
Abbe School of Photonics | JENA
Friedrich-Schiller-Universität

Optics in stratified media



- Bragg mirror
- mirror with chirp for compensating dispersion
- interferometer

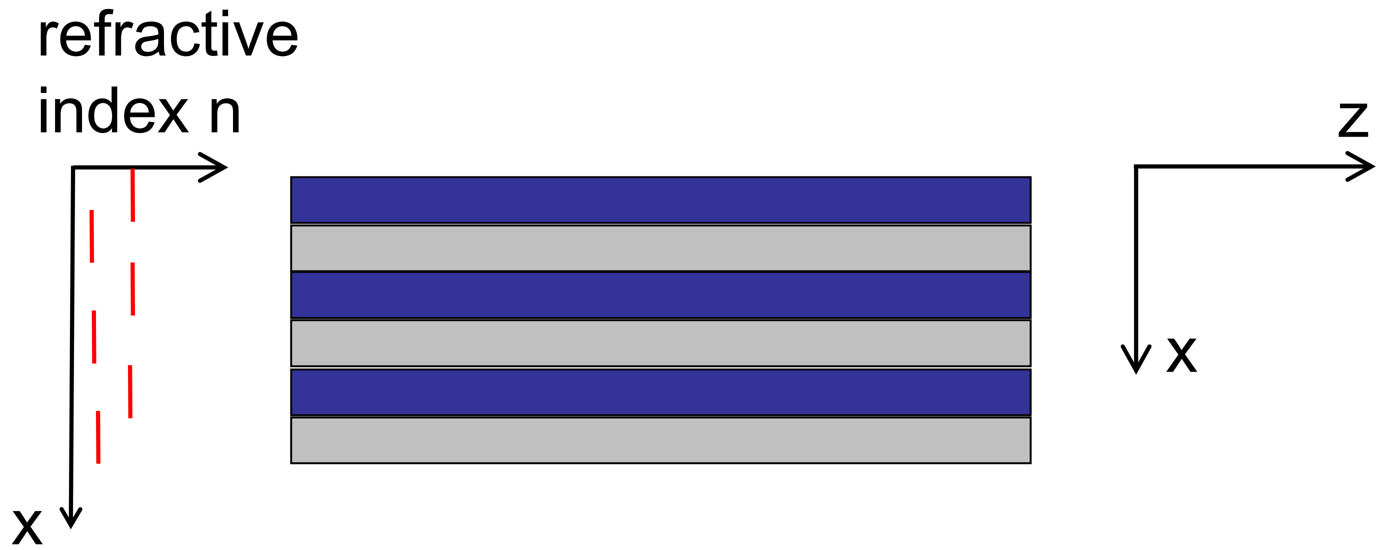
Optics in stratified media



Plane of incidence = x-z-plane

$\Rightarrow$ no y-dependency

A stratified (layered) medium

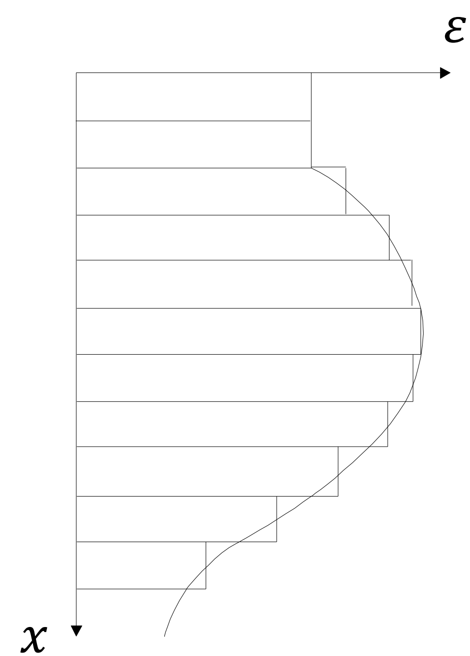


A stratified (layered) medium

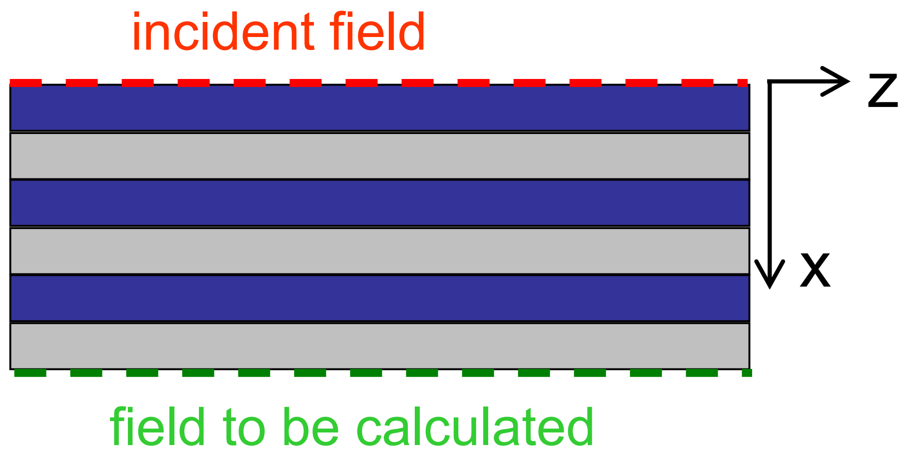
each layer with index i is characterized by its thickness d_i and its dielectric constant $\varepsilon_i(\omega)$

an arbitrary continuous variation of the refractive index can be discretized with a sufficient large number of layers

→ important for so called 'GRIN' – graded index waveguides



EM fields in the stratified media



Requirements:

- stationarity
- infinite extension of the layers in the z-y-plane
- illuminating field incident in the x-z-plane

Ansatz: $\mathbf{E}_{\mathbf{real}}(x,z,t) = \mathrm{Re}\left[\mathbf{E}(x)\exp\left(ik_z z - i\omega t\right)\right]$

$$\mathbf{H}_{\mathbf{real}}(x,z,t) = \mathrm{Re}\left[\mathbf{H}(x)\exp\left(ik_z z - i\omega t\right)\right]$$

Separation into TE und TM polarization

TE: $\mathbf{E}_{\mathrm{TE}} = \begin{pmatrix} 0 \\ E_{\mathbf{y}} \\ 0 \end{pmatrix}, \quad \mathbf{H}_{\mathrm{TE}} = \begin{pmatrix} H_{\mathbf{x}} \\ 0 \\ H_{\mathbf{z}} \end{pmatrix}$

TM: $\mathbf{H}_{\mathrm{TM}} = \begin{pmatrix} 0 \\ H_{\mathbf{y}} \\ 0 \end{pmatrix}, \quad \mathbf{E}_{\mathrm{TM}} = \begin{pmatrix} E_{\mathbf{x}} \\ 0 \\ E_{\mathbf{z}} \end{pmatrix}$

Boundary conditions

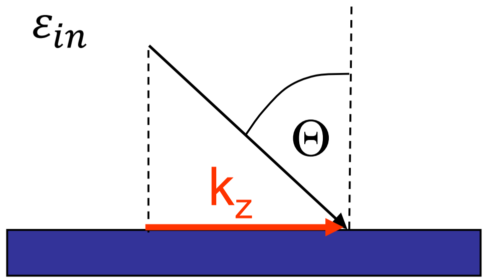
Fields: $\mathbf{E}_t$ and $\mathbf{H}_t$ continuous

TE: $E = E_y$ and H_z
TM: $H = H_y$ and E_z

➔ Performing all computations with the **tangential components**, (if necessary the normal components can be derived)

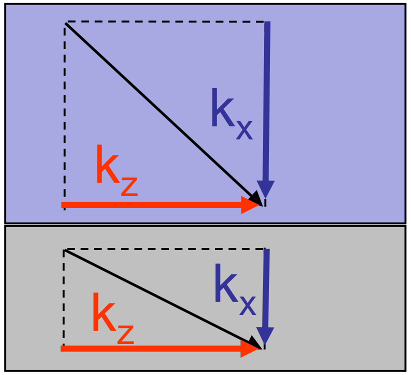
transversal wavevector is constant in the stack and is determined by the angle of incidence:

$$\Rightarrow k_z = \frac{\omega}{c}\sqrt{\varepsilon_{in}}\sin\theta = \frac{2\pi}{\lambda_0}\sqrt{\varepsilon_{in}}\sin\theta$$



normal component varies in the stack:
$\Rightarrow k_x$ depends on the permittivity of each layer
$\Rightarrow$ **dispersion relation**

$$k_x{}^2 = \frac{\omega^2}{c^2}\varepsilon(\omega) - k_z^2$$



Computing the fields by continuous components (TE)

Helmholtz-equation:

$$\left[\frac{\partial^2}{\partial x^2}+\underbrace{\frac{\omega^2}{c^2}\varepsilon(\omega)-k_z^2}_{k_x^2}\right]E_y(x)=0 \qquad i\omega\mu_0 H_z(x)=\frac{\partial}{\partial x}E_y(x)$$

Solution:

$$E_y(x)=C_1\cos\left(k_x x\right)+C_2\sin\left(k_x x\right)$$

$$i\omega\mu_0 H_z(x)=\frac{\partial}{\partial x}E_y(x)=k_x\left[-C_1\sin\left(k_x x\right)+C_2\cos\left(k_x x\right)\right]$$

Determination of C_1, C_2:

$$E_y(0)=C_1 \qquad \left.\frac{\partial}{\partial x}E_y\right|_0=k_x C_2$$

Solution

TE:

$$E_y(x) = \cos\left(k_x x\right) E_y(0) + \frac{1}{k_x}\sin\left(k_x x\right)\frac{\partial}{\partial x}E_y\Big|_0$$

$$\frac{\partial}{\partial x}E_y = -k_x \sin\left(k_x x\right) E_y(0) + \cos\left(k_x x\right)\frac{\partial}{\partial x}E_x\Big|_0$$

TM:

$$H_y(x) = \cos\left(k_x x\right) H_y(0) + \frac{\varepsilon}{k_x}\sin\left(k_x x\right)\frac{1}{\varepsilon}\frac{\partial}{\partial x}H_y\Big|_0$$

$$\frac{1}{\varepsilon}\frac{\partial}{\partial x}H_y = -\frac{k_x}{\varepsilon}\sin\left(k_x x\right) H_y(0) + \cos\left(k_x x\right)\frac{1}{\varepsilon}\frac{\partial}{\partial x}H_y\Big|_0$$

TE/TM:

$$F(x) = \cos\left(k_x x\right) F(0) + \frac{1}{qk_x}\sin\left(k_x x\right) G(0)$$

$$G(x) = -qk_x \sin\left(k_x x\right) F(0) + \cos\left(k_x x\right) G(0)$$

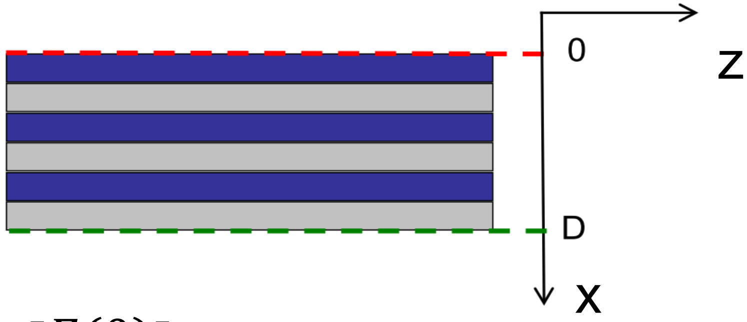
TE: $\quad F = E_y, \quad G = i\omega\mu_0 H_z = \frac{\partial}{\partial x}E_y, \quad q = 1$

TM: $\quad F = H_y, \quad G = -i\omega\varepsilon_0 E_z = q\frac{\partial}{\partial x}H_y, \quad q = 1/\varepsilon$

Summary: Matrix method

Need to know: $F(0)$, $G(0)$,$k_x^{(i)}$, ε_i, d_i

We want to calculate the fields $F(D)$, $G(D)$



$$\begin{bmatrix} F(D) \\ G(D) \end{bmatrix} = \prod_i \hat{m}_i \begin{bmatrix} F(0) \\ G(0) \end{bmatrix} = \hat{M} \begin{bmatrix} F(0) \\ G(0) \end{bmatrix}$$

$$m_i = \begin{bmatrix} \cos\left(k_x^{(i)} d_i\right) & \dfrac{1}{q_i k_x^{(i)}} \sin\left(k_x^{(i)} d_i\right) \\ -q_i k_x^{(i)} \sin\left(k_x^{(i)} d_i\right) & \cos\left(k_x^{(i)} d_i\right) \end{bmatrix}$$

TE: $F = E_y, \quad G = \dfrac{\partial}{\partial x} E_y, \quad q_i = 1$

TM: $F = H_y, \quad G = q_i \dfrac{\partial}{\partial x} H_y, \quad q_i = 1/\varepsilon_i$

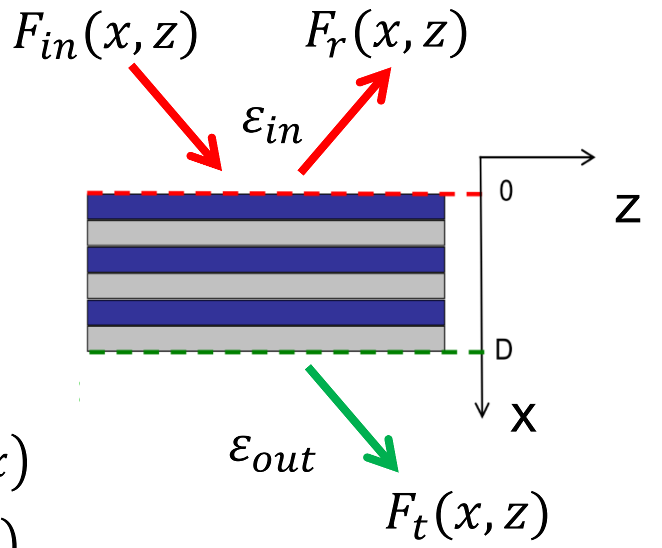
with

$$\left[k_x^{(i)}\right]^2 = \left(\frac{2\pi}{\lambda_0}\right)^2 \varepsilon_i(\omega) - k_z^2$$

Reflection and transmission coefficients of the fields

transmission coefficient $t = \dfrac{F_T}{F_{\text{in}}}$

reflection coefficient $r = \dfrac{F_R}{F_{\text{in}}}$



fields at ε_{in}:

$$F_{in}(x,z) = F_{in}(x)\exp(ik_z z) \qquad F_{in}(x) = F_{in}\exp\left(ik_x^{in}x\right)$$

$$F_r(x,z) = F_r(x)\exp(ik_z z) \qquad F_r(x) = F_r\exp\left(-ik_x^{in}x\right)$$

field at ε_{out}:

$$F_t(x,z) = F_t(x)\exp(ik_z z) \qquad F_t(x) = F_t\exp(ik_x^{out}(x-D))$$

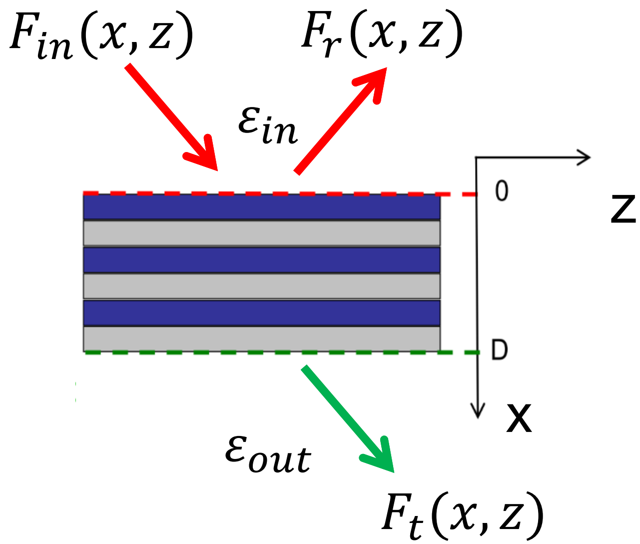
connection of fields at $x = 0$ and $x = D$ by transfer matrix

$$\begin{bmatrix} F(D) \\ G(D) \end{bmatrix} = \hat{M}\begin{bmatrix} F(0) \\ G(0) \end{bmatrix} \longrightarrow \begin{bmatrix} F_t \\ iq_{out}k_x^{out}F_t \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}\begin{bmatrix} F_{in}+F_r \\ iq_{in}k_x^{in}(F_{in}-F_r) \end{bmatrix}$$

Reflection and transmission coefficients of the fields

transmission coefficient $t = \frac{F_T}{F_{\mathrm{in}}}$

reflection coefficient $r = \frac{F_R}{F_{\mathrm{in}}}$

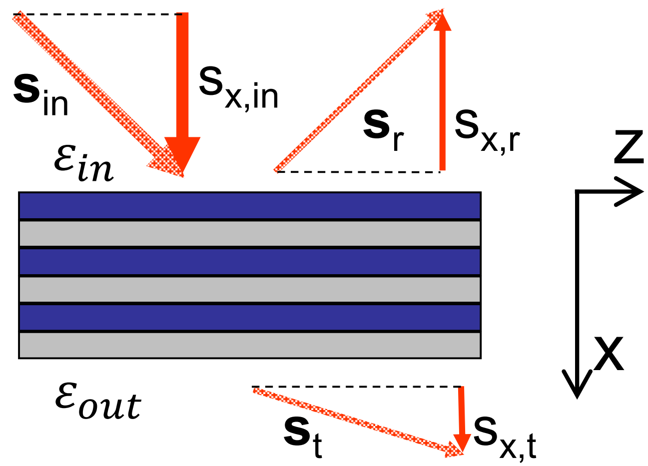


$$r = \frac{q_{in}k_x^{in}M_{22} - q_{out}k_x^{out}M_{11} - i\left(M_{21} + q_{in}k_x^{in}q_{out}k_x^{out}M_{12}\right)}{q_{in}k_x^{in}M_{22} + q_{out}k_x^{out}M_{11} + i\left(M_{21} - q_{in}k_x^{in}q_{out}k_x^{out}M_{12}\right)}$$

$$t = \frac{2q_{in}k_x^{in}}{q_{in}k_x^{in}M_{22} + q_{out}k_x^{out}M_{11} + i\left(M_{21} - q_{in}k_x^{in}q_{out}k_x^{out}M_{12}\right)}$$

Energy flux

defined via the normal component of the Poynting vector **S**



- Reflectivity:

$$R = \frac{S_{x,r}}{S_{x,in}} \qquad\qquad R = |r|^2$$

- Transmissivity:

$$T = \frac{S_{x,t}}{S_{x,in}} \qquad\qquad T = \frac{q_{out}\,\mathrm{Re}\left(k_x^{out}\right)}{q_{in}\,\mathrm{Re}\left(k_x^{in}\right)}|t|^2$$

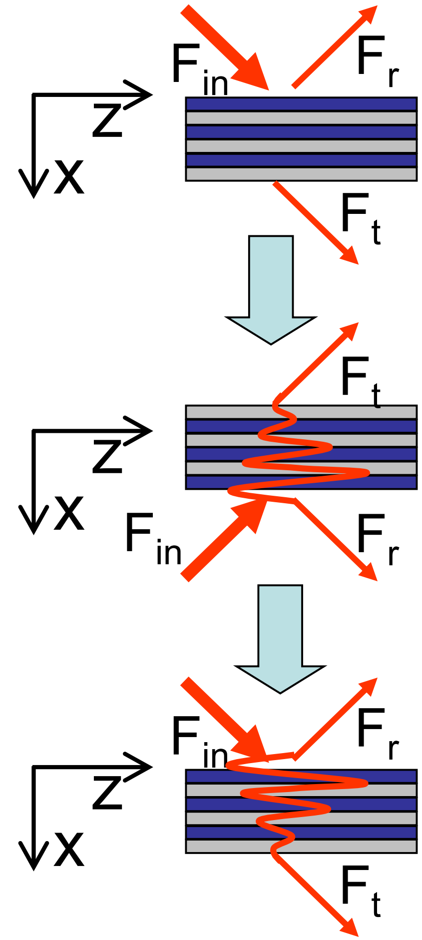
Field distribution

Goal: Computation of $F(x)$ inside the entire structure,
(the absolute values can be scaled)

Initial point: Take the known entries of the transmitted amplitude

$$\begin{bmatrix} F(D) \\ G(D) \end{bmatrix} = \begin{bmatrix} F_t \\ q_{out} \dfrac{\partial F_t}{\partial x} \end{bmatrix} = F_t \begin{bmatrix} 1 \\ i q_{out} k_x^{out} \end{bmatrix}$$

Now: $F_t = 1$



Approach:

1. Reverse the structure (incident vector becomes $(1, -i q_{out} k_x^{out})$)
2. Calculate the field vector up to the next interface
3. From there, calculate the field to the next x-point of interest
4. Save the first value of the vector for this x-point
5. Iterate until all x-values are calculated and reverse the structure and the field

The real field

The observable (real) field

$$\mathbf{E_r}(x,z,t) = \mathrm{Re}\left[\mathbf{E}(x)\exp\left(ik_z z - i\omega t\right)\right]$$
$$\mathbf{H_r}(x,z,t) = \mathrm{Re}\left[\mathbf{H}(x)\exp\left(ik_z z - i\omega t\right)\right]$$

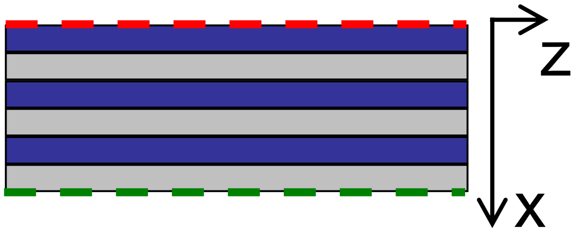
What you have actually calculated is the complex value of a certain component:

TE: $\mathbf{E}\left(x\right) = F\left(x\right)\mathbf{e_y}$

TM: $\mathbf{H}\left(x\right) = F\left(x\right)\mathbf{e_y}$

Task I : Transfer matrix

Goal : calculation of $\hat{M}$



MATLAB

```
function M = transfermatrix(thickness, epsilon, polarisation, lambda, kz)
% M = transfermatrix(thickness, epsilon, polarisation, lambda, kz)
% Computes the transfer matrix for a given stratified medium.
% All dimensions are in µm.
% Arguments:
%     thickness    : Thicknesses of the layers (vector)
%     epsilon      : Dielectric permittivity of the layers (vector)
%     polarisation : Polarisation of the field to be computed (string: 'TE' or 'TM')
%     lambda       : Wavelength of the light (scalar)
%     kz           : Transverse wavevector [1/µm] (scalar)
% Returns:
%     M : Transfer matrix (matrix)
```

(potentially) useful functions:

`eye(N)`: creates the N-dimensional unity matrix

`error('message')`: prints `'message'` on the screen and interrupts the program

`strcmp(variable,'string')`: compares variable against `'string'`

Task I : Transfer matrix

Goal : calculation of $\widehat{M}$

Python

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

def transfermatrix(thickness, epsilon, polarisation, wavelength, kz):
    '''Computes the transfer matrix for a given stratified medium.

    Parameters
    ----------
    thickness : 1d-array
        Thicknesses of the layers in µm.
    epsilon : 1d-array
        Relative dielectric permittivity of the layers.
    polarisation : str
        Polarisation of the computed field, either 'TE' or 'TM'.
    wavelength : 1d-array
        The wavelength of the incident light in µm.
    kz : float
        Transverse wavevector in 1/µm.

    Returns
    -------
    M : 2d-array
        The transfer matrix of the medium.
    '''
    pass
```

Task II: Reflection and transmission coefficients

Goal: computation of r, t, R, t as a function of the wavelength

MATLAB

```
function [t, r, T, R] = spectrum(thickness,epsilon,polarisation, ...
                                      lambda_vector,angle_inc,n_in,n_out)
% [t, r, T, R] = spectrum(thickness,epsilon,polarisation, ...
%                              lambda_vector,angle_inc,n_in,n_out)
% Computes the reflection and transmission of a stratified medium depending on the
% wavelength. All dimensions are in µm.
% Arguments:
%     thickness     : Thicknesses of the layers (vector)
%     epsilon       : Dielectric permittivity of the layers (vector)
%     polarisation  : Polarisation of the field to be computed (String: 'TE' or 'TM')
%     lambda_vector : Wavelength of the light (vector)
%     angle_inc     : Angle of incidence in degree (scalar)
%     n_in, n_out   : Refractive indices of the input and output layers (scalars)
% Returns:
%     t   : Transmitted amplitude (complex vector)
%     r   : Reflected amplitude (complex vector)
%     T : Transmitted energy (real vector)
%     R : Reflected energy (real vector)
```

Using the function **`transfermatrix`**

Task II: Reflection and transmission coefficients (1/2)

Goal: computation of r, t, R, t as a function of the wavelength

Python

```
def spectrum(thickness, epsilon, polarisation, wavelength, angle_inc, n_in, n_out):

    '''Computes the reflection and transmission of a stratified medium.

    Parameters
    ----------
    thickness : 1d-array
        Thicknesses of the layers in µm.
    epsilon : 1d-array
        Relative dielectric permittivity of the layers.
    polarisation : str
        Polarisation of the computed field, either 'TE' or 'TM'.
    wavelength : 1d-array
        The wavelength of the incident light in µm.
    angle_inc : float
        The angle of incidence in degree (not radian!).
    n_in, n_out : float
        The refractive indices of the input and output layers.
```

Task II: Reflection and transmission coefficients (2/2)

Goal: computation of r, t, R, t as a function of the wavelength

Python

```
Returns
-------
t : 1d-array
    Transmitted amplitude
r : 1d-array
    Reflected amplitude
T : 1d-array
    Transmitted energy
R : 1d-array
    Reflected energy
'''
pass
```

Task III*: Field distribution

Goal: Computation of the complex field f at predefined values of x

MATLAB

```
function [f, index, x] = field(thickness,epsilon,polarisation, ...
                               lambda,kz,n_in,n_out,Nx,l_in,l_out);
% function [f, index, x] = field(thickness,epsilon,polarisation, ...
%                                lambda,kz,n_in,n_out,Nx,l_in,l_out)
% Computes the field in a stratified medium. All dimensions are in µm.
% The stratified medium starts at x = 0 on the entrance side
% (stratified media for x > 0). The transmitted field has a magnitude of unity.
% Arguments:
%     thickness    : Thicknesses of the layers (vector)
%     epsilon      : Dielectric permittivity of the layers (vector)
%     polarisation : Polarisation of the field to be computed (String: 'TE' or 'TM')
%     lambda       : Wavelength (scalar)
%     kz           : Transverse wavevector [1/µm] (scalar)
%     n_in, n_out  : Refractive index of the input and output layers (scalars)
%     Nx           : Number of points where the field shall be computed (integer)
%     l_in, l_in   : Additional thickness of the input and output layers where the
%                    field should be computed (scalars)
% Returns:
%     f      : Field structure (complex vector)
%     index  : Refractive index distribution (complex vector)
%     x      : Spatial coordinate (real vector)
```

Using the functions **`transfermatrix, fliplr`**

Task III*: Field distribution (1/2)

Goal: Computation of the complex field f at predefined values of x

Python

```
def field(thickness, epsilon, polarisation, wavelength, kz, n_in, n_out, Nx, l_in, l_out):
    '''Computes the field inside a stratified medium.

    The medium starts at x = 0 on the entrance side. The transmitted field
    has a magnitude of unity.

    Parameters
    ----------
    thickness : 1d-array
        Thicknesses of the layers in µm.
    epsilon : 1d-array
        Relative dielectric permittivity of the layers.
    polarisation : str
        Polarisation of the computed field, either 'TE' or 'TM'.
    wavelength : 1d-array
        The wavelength of the incident light in µm.
    kz : float
        Transverse wavevector in 1/µm.
```

Task III*: Field distribution (2/2)

Goal: Computation of the complex field f at predefined values of x

Python

```
    n_in, n_out : float
        The refractive indices of the input and output layers.
    Nx : int
        Number of points where the field will be computed.
    l_in, l_out : float
        Additional thickness of the input and output layers where the field will
be computed.

    Returns
    -------
    f : 1d-array
        Field structure
    index : 1d-array
        Refractive index distribution
    x : 1d-array
        Spatial coordinates
    '''
    pass
```

Task IV*: Time animation of the field

Goal: Visualization of the temporal evolution of the field

MATLAB

```
function timeanimation(x,f,index,steps,periods)
% timeanimation(x,f,index,steps,periods)
% Animation of a quasi-stationary field.
% Arguments:
%     x       : Spatial coordinates (real vector)
%     f       : Field (complex vector)
%     index   : Refractive index (complex vector, real part plotted with the field)
%     steps   : Total number of time points (integer)
%     periods : Number of the oscillation periods (integer)
```

Using the functions **max, axis, figure(gcf), pause**

Task IV*: Time animation of the field

Goal: Visualization of the temporal evolution of the field

Python

```python
def timeanimation(x, f, index, steps, periods):
    ''' Animation of a quasi-stationary field.

    Parameters
    ----------
    x : 1d-array
        Spatial coordinates
    f : 1d-array
        Field
    index : 1d-array
        Refractive index
    steps : int
        Total number of time points
    periods : int
        Number of the oscillation periods.

    '''
    pass
```

Example parameters

Define a Bragg mirror at 780nm:

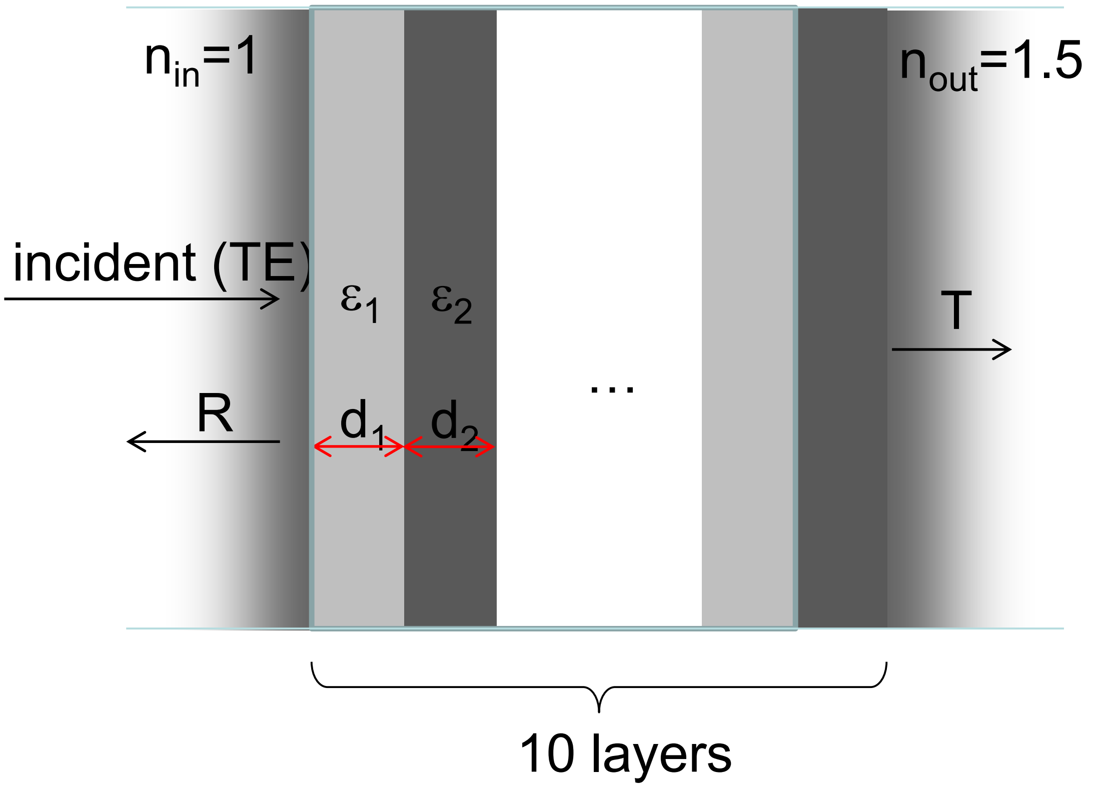
```
>> eps1 = 2.25;
>> eps2 = 15.21;
>> d1 = 0.13;    %[µm]
>> d2 = 0.05;    %[µm]
>> N  = 5;
>> polarisation ='TE';
>> angle_inc = 0.0;
>> n_in = 1.0;
>> n_out = 1.5;
```

Create the arrays

```
>> epsilon = zeros(1, 2*N);
>> epsilon(1:2:2*N)  = eps1;
>> epsilon(2:2:2*N)  = eps2;
>> thickness = zeros(1, 2*N);
>> thickness(1:2:2*N)= d1;
>> thickness(2:2:2*N) = d2;
>> lambda = linspace(0.5, 1.5, 100); %[µm]
```

Now, e.g. calculate the transmission/reflection spectrum:

```
>> [t, r, T, R] = spectrum(thickness, epsilon,
                           lambda_vector, ang]
```



Voluntary Homework (due 9 May 2019)

- These tasks are still **voluntary,** but it is strongly encouraged to solve at least the first two (I and II).
- For each task we require that each student implements a program that solves the problem and documents the code and its result (e.g. with an iPython Notebook).
- The source code and the report must be submitted via email to teaching-nanooptics@uni-jena.de by Thursday (9 May 2019).
- The subject line of the email should have the following format:
  - [family name]; [given names]; [student id]: solution to the assignment of seminar [seminar no.]
- All source code files should be gathered in a single zip archive (no rar, tar, 7z, gz or any other compression format!)