Computational Physics - Exercise 7
Yee algorithm

Christian Gorjaew (354259)
christian.gorjaew@rwth-aachen.de

Julius Meyer-Ohlendorf (355733)
julius.meyer-ohlendorf@rwth-aachen.de

June 21, 2020

1. Introduction

In the following we simulate the transmission and reflection of light by a glass plate in 1D using the `Yee algorithm`. Two different thicknesses of the glass plate are considered. The electric field E is plotted at different times and the reflection coefficient is calculated for the thicker plate. Furthermore, the stability condition of the algorithm is investigated.

2. Simulation model and method

Model:
Maxwell's equations make up the base of classical electromagnetism and describe the propagation and interaction of electromagnetic waves. The Maxwell equations in 1D can be derived from the transverse electric mode in 2D (c.f. lec. 14, slides 30-36):

$$\begin{aligned}\frac{\partial H_y(x,t)}{\partial t} &= \frac{1}{\mu(x)}\left(\frac{\partial E_z(x,t)}{\partial t} - \sigma(x)^\star H_y(x,t)\right)\\ \frac{\partial E_z(x,t)}{\partial t} &= \frac{1}{\epsilon(x)}\left(\frac{\partial H_y(x,t)}{\partial t} - J_{\text{source},z} - \sigma(x)E_z(x,t)\right),\end{aligned} \tag{1}$$

with magnetic permeability $\mu(x)$, electrical permittivity $\epsilon(x)$, magnetic loss $\sigma^\star(x)$, electrical conductivity $\sigma(x)$ and electrical current density $J_{\text{source},z}$. Those equations correspond to a transverse electromagnetic wave in 1D (note that the polarization of the electric field was chosen into z-direction, that of the magnetic field into y-direction). Furthermore, divergence free fields are assumed.
We model the transmission and reflection of light by a glass plate in 1D. For that, a simulation box in x-direction of length $X = 100\lambda$ with $\lambda = 1$ (due to this all following quantities are dimensionless) is considered. At both ends of the box, matched boundary layers with thickness 6λ for reflectionless absorption are assumed. Therefore, $\sigma(x)$ and $\sigma(x)^\star$ is set to:

$$\sigma(x) = \sigma(x)^\star = \begin{cases}1 & \text{if} \quad 0 \le x \le 6\lambda\\ 0 & \text{if} \quad 6\lambda < x < L\Delta - 6\lambda\\ 1 & \text{if} \quad L\Delta - 6\lambda \le x \le L\Delta,\end{cases} \tag{2}$$

where Δ is the spatial resolution or spacing of the Yee grid (c.f. following paragraph). An electric source with current density $J_{\text{source},z}(x_s,t) = \sin(2\pi t f)e^{-((t-30)/10)^2}$ and frequency $f = 1$ creating a wave packet is installed at $x_s = 20\lambda$.
For the glass plate two different cases are considered. In the first case, a thin plate of thickness 2λ is placed in the middle of the simulation box (results in section 3.1):

$$\epsilon(x) = \begin{cases}1 & \text{if} \quad 0 \le x < L\Delta/2\\ n^2 & \text{if} \quad L\Delta/2x < L\Delta/2 + 2\lambda\\ 1 & \text{if} \quad L\Delta/2 + 2\lambda \le x \le L\Delta\end{cases} \tag{3}$$

In the second case, a thick plate spanning over half of the simulation box is regarded (results in section 3.2):

$$\epsilon(x) = \begin{cases}1 & \text{if} \quad 0 \le x < L\Delta/2\\ n^2 & \text{if} \quad L\Delta/2 \le x < L\end{cases} \tag{4}$$

In both cases the refraction index is set to $n = 1.46$.

Method:
In order to solve the Maxwell equations numerically, the `Yee algorithm` is employed. The algorithm, adapted to the present 1D-propagation case, calculates the component of the electric E_z and magnetic field H_y on an interleaved lattice in space (Yee grid) updating the electric field at half integer time steps and the magnetic field at integer time steps. The algorithm's updating scheme is derived by substituting the leapfrog method (centered 3-point formula with half integer steps, which is of 2nd order) for the time and space derivatives in the Maxwell equations. Rearranging the equations and approximating the electrical field at integer time steps and the magnetic field at half integer time steps, leads to the Yee updating scheme:

$$\begin{aligned} E_z\Big|_l^{n+1/2} &= C_l E_z\Big|_l^{n-1/2} + D_l \left[\frac{H_y\Big|_{l+1/2}^n - H_y\Big|_{l-1/2}^n}{\Delta} - J_{\text{source},z}\Big|_l^n \right] \\ H_y\Big|_{l+1/2}^{n+1} &= A_{l+1/2} H_y\Big|_{l+1/2}^n + B_{l+1/2} \left[\frac{E_z\Big|_{l+1}^{n+1/2} - E_z\Big|_l^{n+1/2}}{\Delta} \right], \end{aligned} \tag{5}$$

where

$$C_l = \left[\frac{1 - \frac{\sigma_l \tau}{2\epsilon_l}}{1 + \frac{\sigma_l \tau}{2\epsilon_l}} \right] \qquad D_l = \left[\frac{\frac{\tau}{\epsilon_l}}{1 + \frac{\sigma_l \tau}{2\epsilon_l}} \right] \tag{6}$$

Here, we define $\sigma_l \equiv \sigma(x_l)$ and $\epsilon_l = \epsilon(x_l)$. $A_{l+1/2}$ and $B_{l+1/2}$ are defined analogously but with σ_l and ϵ_l interchanged by $\sigma^*_{l+1/2}$ and $\mu_{l+1/2}$. Here, τ corresponds to the time step and Δ to the lattice spacing of the Yee grid. As one can see, at first the electrical field E_z is updated and afterwards the magnetic field H_y. At a specific half integer time step and a specific integer lattice point the electric field is updated by incorporating neighboring magnetic field values at half integer lattice points form the previous integer time step. Vice versa the same holds for the magnetic field.
The stability of the Yee algorithm is investigated by writing the Maxwell equations in matrix form:

$$\frac{\partial}{\partial t} \vec{\Psi}(t) = \boldsymbol{L} \vec{\Psi}(t), \tag{7}$$

with $\vec{\Psi}(t) = \begin{pmatrix} E_z(x,t) \\ H_y(x,t) \end{pmatrix}$ and $\boldsymbol{L} = \begin{pmatrix} 0 & \frac{\partial}{\partial x} \\ \frac{\partial}{\partial x} & 0 \end{pmatrix}$. The electrical source term $J_{source,z}$ was neglected here.
The solution is given by:

$$\vec{\Psi}(t) = e^{t\boldsymbol{L}} \vec{\Psi}(0), \tag{8}$$

with the matrix exponential $e^{t\boldsymbol{L}}$. For stability of the algorithm, it is demanded that $\|e^{t\boldsymbol{L}}\| \leq 1$ for all times, so that the solution does not diverge. By further mathematical analysis (c.f. lec. 15, slides 38 ff.), it is then derived that the algorithm is stable if it satisfies the Courant condition:

$$\frac{\tau}{\Delta} \leq c, \tag{9}$$

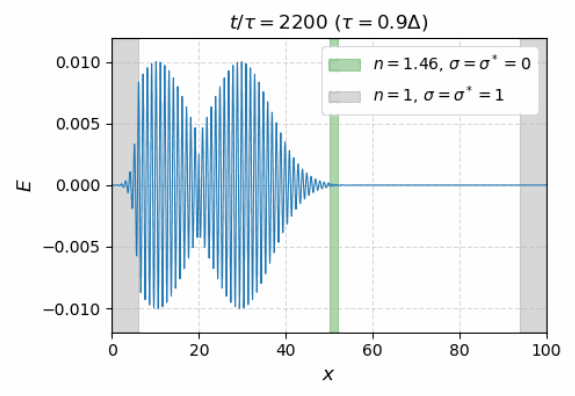
here $c \equiv 1$. This stability condition is investigated in the simulation of the thin glass plate.

3. Simulation results

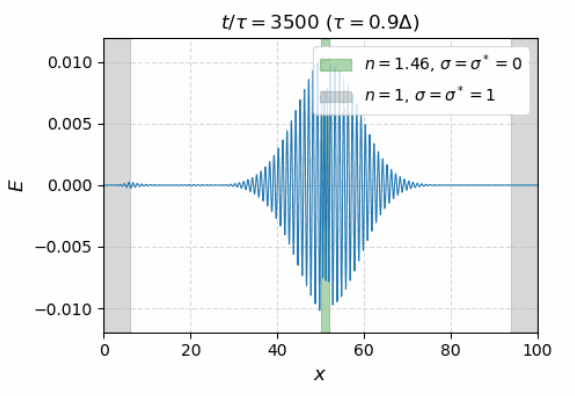
The `python` code used for the simulation can be found in the Appendix A.

3.1. Thin glass plate

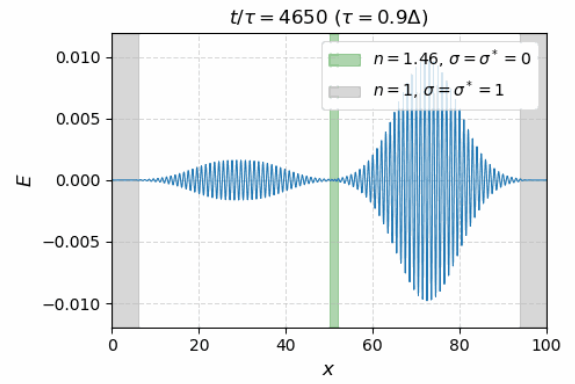
The results for the configuration with a thin glass plate in the center of simulation box, as described in the previous chapter, can be seen in Figure 1. Additionally, an animated `gif` version of the simulation can be seen under the link in the footnote.[1] One sees in Figure 1(a)



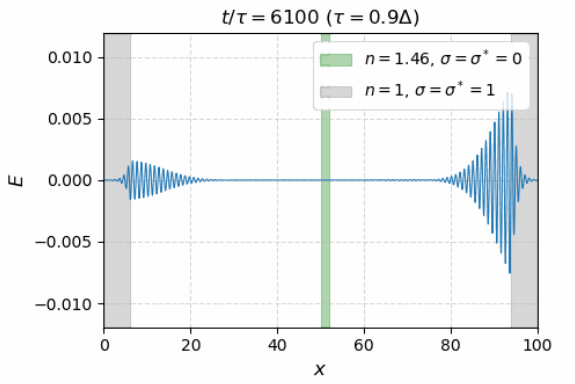
(a) Wave shortly after its creation.



(b) Incident wave during transmission through the glass plate.



(c) Transmitted and reflected parts of the wave.



(d) Absorption of the wave parts at the metallic boundary.

Figure 1: Electric field inside the simulation box at different times for a glass plate of thickness 2λ (green). The chosen time steps display different events occurring in the present setup. The position x is given in units of $\lambda \equiv 1$.

how the wave originates from the source position $x_s = 20\lambda$ and propagates into positive and negative x-directions. The left part is absorbed inside the metallic boundary (grey region) while the right part is propagating towards the glass plate (green region), eventually penetrating it as seen in Figure 1(b). One recognizes that the electric field is attenuated inside the glass since the dielectric constant differs from unity within this region. In Figure 1(c), the reflected and transmitted parts if the incident wave can be seen. As expected, the reflected wave has a

[1] Animated simulation results: `https://rwth-aachen.sciebo.de/s/Km3rUUmFOxiFGXq`. To see the animation, you might have to download the files.

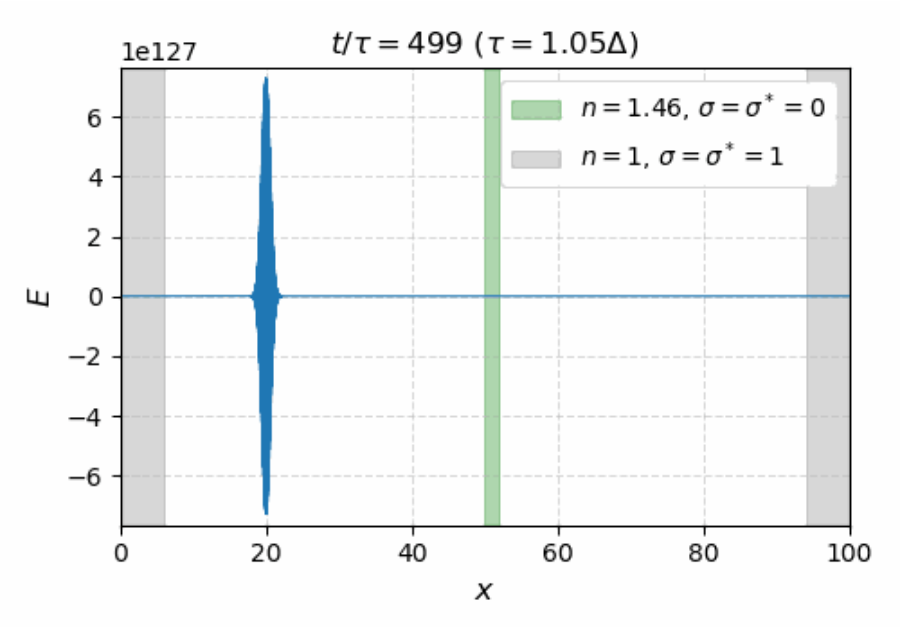


Figure 2: Simulated electric field after 500 steps employing a time step $\tau = 1.05\Delta$.

much lower amplitude compared to the transmitted wave. Figure 1(d) shows the absorption of the reflected and transmitted waves when reaching the metallic boundary. In the further development which is not shown here, one sees further oscillations whose amplitudes are smaller by some orders of magnitude (with respect to the incident, reflected and transmitted waves). These might be caused by delayed transmitted parts of the wave that are reflected multiple times inside the glass plate before being transmitted or a result of incomplete absorption at the boundary.

In Figure 2, an instance of the simulation with a time step that does not meet the Courant condition is shown. As can be seen, the solution of the electric field diverges quickly after approximately 500 steps (note the factor 10^{127} applied to the y axis). This meets the expectation of stability stating that the Yee algorithm is only stable if $\tau \leq c/\Delta$ ($c = 1$ in present case).

3.2. Thick glass plate

Here, we model the same system as before except that the glass plate now extends over the right half of the simulation domain. This can be seen in Figure 3 for two different time steps. These time steps were chosen to get an instance of the incident wave and the reflected wave in order to calculate the reflectivity (see below). The animated version can be found under the link in Footnote 1 as well.

Comparing the transmitted waves in Figure 3(b) to that in Figure 1(c) one recognizes that the former wavepackage extension in x-direction is decreased because the wavelength is reduced inside the glass as a result of the increased permittivity. Investigating the animations, one also recognizes that the transmitted wave reaches the metallic boundary later than the reflected wave due to the decreased propagation speed inside the medium.

Using the snapshots of the waves in Figure 3, the reflectivity R was determined by taking the maximum of the absolute value of the part of the wave in the region $x = 0$ to 50. This leads to

$$R = \frac{|E_{\text{refl}}^{\max}|^2}{|E_{\text{inc}}^{\max}|^2} \approx 0.0350, \tag{10}$$

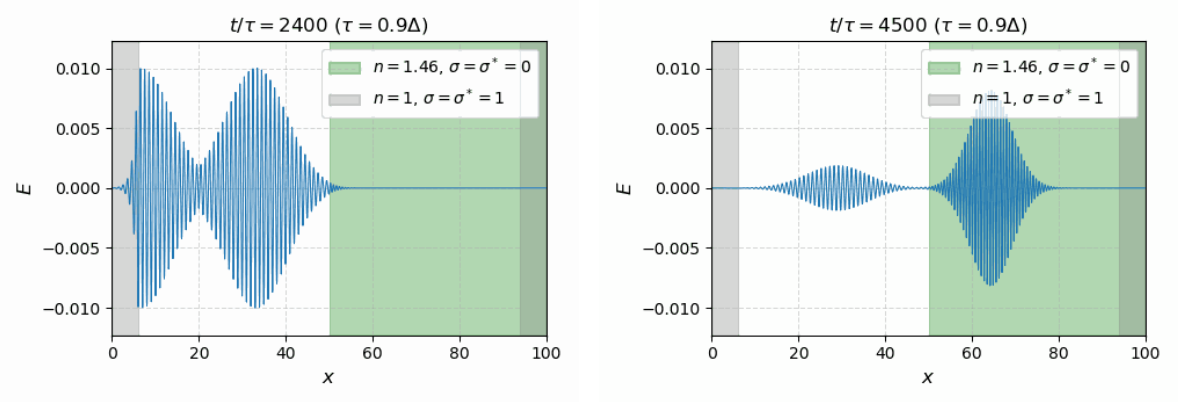


Figure 3: Electric field inside the simulation box for a thick glass plate at two different time steps that were used to determine the reflectivity. Simulation and physical parameter as in Figure 1.

where $E_{\text{refl}}^{\max}$ is the the maximum of the reflected wave at step $t/\tau = 4500$ and $E_{\text{inc}}^{\max}$ the maximum of the incident wave at step $t/\tau = 2400$. This value agrees with the theoretical prediction for the reflectivity of a wave propagating in vacuum and being reflected on the surface of a medium with refractive index n when invading perpendicular:

$$R_{\text{theo}} = \frac{|1-n|}{|1+n|} \overset{n=1.46}{\approx} 0.0349 \tag{11}$$

4. Discussion

The simulation of the discussed situation could be implemented and executed successfully using Yee's algorithm. It could be shown that the algorithm is in fact unstable if the Courant condition is not met, leading to strong divergence of the solution. In turn, the solution stays bounded already for time step sizes $\tau = 0.9\Delta$ which is still close to the stability boundary of $\tau = c\Delta$ ($c \equiv 1$). For the thick glass plate, the reflectivity was determined to $R = 0.0350$ which agrees with the theoretical value for an perpendicular incident wave on a glass surface.

A. Code used for simulation and animation

```
"""
Python 3.7.7

@author: Christian Gorjaew, Julius Meyer-Ohlendorf
"""

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import h5py as h5

from tqdm import tqdm
from numpy import sin, exp, pi

def yee_step(E, H, l_s, J, A, B, C, D, delta):
    """
    Updates the Yee grid by one time step
    """
    E[1:-1] *= C[1:-1]
    E[1:-1] +=  D[1:-1] * (H[1:] - H[0:-1]) / delta
    E[l_s] -= D[l_s] * J
    H *= A
    H +=  B * (E[1:] - E[0:-1]) / delta

"""
###### Defining physical and simulation parameters on grid #####
###### Change parameters here ####
"""

L = 5000
m = 10000

lamb = 1.
delta = lamb / 50
step_fac = 0.9  # Change here for different time step

n_refr = 1.46

tau = step_fac * delta
tarr = np.arange(0, m * tau, tau)  # array with time values

# Setting sigma(x)
left_bound = int(6 * lamb / delta)
sigma = np.zeros(L + 1)
sigma[0: left_bound] = 1.
sigma[L - left_bound:] = 1.

# setting sigma*(x)
sigma_star = np.zeros(L)
sigma_star[0: left_bound-1] = 1.
sigma_star[L - left_bound:] = 1.

# setting permittivity epsilon(x)
left_bound_eps = int(np.ceil(L / 2))  # Change to -1 for thick glas plate
right_bound_eps = int(np.ceil(L / 2 + 2 * lamb / delta))
eps = np.ones(L + 1)
eps[left_bound_eps:right_bound_eps] = n_refr**2

# setting magnetic permeability mu(x)
mu = np.ones(L)
```

1

```
# setting parameters A, B, C, D employed in Yee steps
A = (1 - sigma_star * tau / 2 / mu) / (1 + sigma_star * tau / 2 / mu)
B = tau / mu / (1 + sigma_star * tau / 2 / mu)
C = (1 - sigma * tau / 2 / eps) / (1 + sigma * tau / 2 / eps)
D = tau / eps / (1 + sigma * tau / 2 / eps)

# setting source for whole duration of simulation in time steps of tau
x_s = 20 * lamb
i_s = int(x_s / delta)
J_s = sin(2 * pi / lamb * tarr) * exp(-((tarr - 30.) / 10.)**2)

# initializing E and H fields with zeros
E = np.zeros(L + 1)
H = np.zeros(L)

# setting up storage frame
storage_frame = 50
storage_steps = int(m / storage_frame)

# storage path and file name, change 'extra_string' for different configuration
extra_string = "_0.9_thin"
path = "./yee_data" + extra_string + ".hdf5"

#%%  <-- signature lets you excecute code cells with some python interpreters

# excecuting the actual simulation
with h5.File(path, "a") as file:
    # setting up the storage file of type 'hdf5'
    file.create_dataset("step", shape=(storage_steps,), maxshape=(None,),
                        chunks=True, dtype=int)
    file.create_dataset("t", shape=(storage_steps,), maxshape=(None,),
                        chunks=True, dtype=float)
    file.create_dataset("E", shape=(storage_steps, L+1), maxshape=(None, L+1),
                        chunks=True, dtype=float)
    file.create_dataset("H", shape=(storage_steps, L), maxshape=(None, L),
                        chunks=True, dtype=float)
    k_storage = 0

    # actual simulation loop, 'try-except' to catch exception due to divergence
    # when Courant condition is not met
    try:
        for step in tqdm(range(m)):

            yee_step(E, H, i_s, J_s[step], A, B, C, D, delta)

            if (step % storage_frame) == 0:
                file["step"][k_storage] = step
                file["t"][k_storage] = tarr[step]
                file["E"][k_storage] = E
                file["H"][k_storage] = H

                k_storage += 1
    except:
        pass

#%%
xx = np.arange(0., (L+1)*delta, delta)  # simulation domain

def animate_E(path, extra_string=""):
    """
    Generates a gif displaying the solution for the electric field that was
```

```
    simulated with the Yee algorithm above
    """
    fig, ax = plt.subplots(figsize=(5,3.5))
    fig.set_tight_layout(True)


    lineE, = ax.plot(xx, np.zeros_like(xx), linewidth=0.75)

    def update(i):
        step = int(t[i] / tau)
        title = "$t / \\tau = ${0} ($\\tau = ${1}$\Delta$)".format(step, step_fac)
        lineE.set_ydata(E_d[i])
        ax.set_title(title, fontsize=12)

        return (lineE, ax)

    data = h5.File(path, "r")

    t = data["t"][:]
    E_d = data["E"][:]
    data.close()
    ax.set_xlim(0,xx[-1])
    ax.set_ylim(np.min(E_d)*1.05, np.max(E_d)*1.05)
    ax.set_xlabel("$x$", fontsize=12)
    ax.set_ylabel("$E$", fontsize=12)
    ax.axvspan(xx[left_bound_eps], xx[right_bound_eps], color="g", alpha=0.3,
               label="$n = ${0}, $\\sigma = \\sigma^* = 0$".format(n_refr))
    ax.axvspan(0., xx[left_bound], color="grey", alpha=0.3,
               label="$n = 1$, $\\sigma = \\sigma^* = 1$".format(n_refr))
    ax.axvspan(xx[L - left_bound], xx[-1], color="grey", alpha=0.3)
    ax.grid(linestyle="--", alpha=0.5)
    ax.legend(loc=1, fontsize=10)

    anim = FuncAnimation(fig, update, frames=np.arange(t.shape[0]))
    anim.save("E" + extra_string + ".gif", dpi=80)

animate_E(path, extra_string)

#%%
# calculation of the reflectivity
if extra_string == "_0.9_thick":
    # setting time steps at which incident and reflected wave are clearly
    # visible
    step_inc = 2400
    step_refl = 4500

    # loading data
    data = h5.File(path, "r")
    steps = data["step"][:]
    E_d = data["E"][:]
    ind_inc = np.where(steps == step_inc)[0]
    ind_refl = np.where(steps == step_refl)[0]

    # determining maximum of absolute values of the waves
    E_inc_max = np.max(np.abs(E_d[ind_inc][0,0:left_bound_eps]))
    E_refl_max = np.max(np.abs(E_d[ind_refl][0,0:left_bound_eps]))

    R = E_refl_max**2 / E_inc_max**2

    print("Reflection coefficient R = {0}".format(R))
```