Electromagnetic Waves

In the previous parts we have dealt with mechanics essentially. Even if we have described Brownian motion, this has been done by a particular type of Newtons equation of motion, it is much like mechanics. Now we would like to have a look at some examples from electromagnetic waves. We will not solve the wave equation but look at some solution using the complex notion of the electric field. This shall train our use of complex numbers. The special solutions are the **plane wave** and the **spherical wave** and we will be able to simulate a number of things especially with the spherical waves as they are part of Huuygens principle.

```
#| edit: false
#| echo: false
#| execute: true

import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

# Set default plotting parameters
plt.rcParams.update({
    'font.size': 12,
    'lines.linewidth': 1,
    'lines.markersize': 5,
    'axes.labelsize': 11,
    'xtick.labelsize': 10,
    'ytick.labelsize': 10,
    'xtick.top': True,
    'xtick.direction': 'in',
    'ytick.right': True,
    'ytick.direction': 'in',
})

def get_size(w, h):
    return (w/2.54, h/2.54)
```

Plane waves

We will start with plane waves. Plane waves are solutions of the homogeneous wave equation and are the simplest solutions of the wave equation. They are also the basis for the description of more complicated waves. We will have a look at the electric field of a plane wave and its propagation in space and time.

Equations

A plane wave is a solution of the homogeneous wave equation and is given in its complex form by

$$E = E_0 e^{i\vec{k}\cdot\vec{r}} e^{-i\omega t} \tag{1}$$

where the two exponentials contain a spatial and a temporal phase. E_0 denotes the amplitude of the plane wave. The plane is defined by the shape of the wavefront which is given by $\vec{k}\cdot\vec{r} = const$, which is just the definition of a plane perpendicular to $\vec{k}$.

A wave is a physical quantity which oscillates in space and time. Its energy current density is related to the square magnitude of the amplitude. We will include in the following the spatial and the temporal phase. For plotting just the spatial variation of the electric field, you may just use the spatial part of the equation

$$E = E_0 e^{i\vec{k}\cdot\vec{r}} \tag{2}$$

But since we also want to see the wave propagate, we will directly include also the temporal dependence in our function. In all of the examples below we set the amplitude of the wave $E_0 = 1$.

The propagation of the wave is defined by wavevector $\vec{k}$. In vacuum, the wavevector is just real valued

$$\vec{k}_0 = \begin{pmatrix} k_{0x} \\ k_{0y} \\ k_{0z} \end{pmatrix} \tag{3}$$

The wavevector is providing the direction in which the wavefronts propagate. It is also proportional to the momentum of the wave, which will be important if we consider the refraction process a bit later. The magnitude of the wavevector is related to the wavelength λ.

$$k_0 = \frac{2\pi}{\lambda_0} = \frac{\omega}{c_0} \tag{4}$$

At the same time, its magnitude is also given by the angular frequency divided by the speed of light. The latter is called a dispersion relation.

In a medium, the wavevector is by a factor of n longer, where n is the refractive index. Since the refractive index may be a complex number, e.g. $n = \eta + i\kappa$, the wavevector can be complex as well. It is then given by

$$\vec{k} = n\vec{k}_0 = \begin{pmatrix} k'_x + ik''_x \\ k'_y + ik''_y \\ k'_z + ik''_z \end{pmatrix} \tag{5}$$

The complex refractive index means that there is some damping of the electromagnetic wave due to absorption, for example.

The wavelength is then related to

$$\Re(k) = \eta \frac{2\pi}{\lambda_0} \tag{6}$$

and the imaginary part gives the damping

$$\Im(k) = \kappa \frac{2\pi}{\lambda_0} \tag{7}$$

Electric field

```
#| autorun: false
def plane_wave(k,omega,r,t):
    return(np.exp(1j*(np.dot(k,r)-omega*t)))
```

Lets have a look at waves and wave propagation. We want to create a wave, which has a wavelength of 532 nm in vacuum.

```
#| autorun: false
wavelength=532e-9
k0=2*np.pi/wavelength
c=299792458
omega0=k0*c
```

It shall propagate along the z-direction and we wull have a look at the x-z plane.

```
#| autorun: false
vec=np.array([0.0,0.,1.])
vec=vec/np.sqrt(np.dot(vec,vec))

k=k0*vec
```

We can plot the electric field in the x-z plane by defining a grid of points (x,z). This is done by the *meshgrid* function of *numpy*. The meshgrid returns a 2-dimensional array for each coordinate. Have a look at the values in the meshgrid.

```
#| autorun: false
x=np.linspace(-2.5e-6,2.5e-6,300)
z=np.linspace(0,5e-6,300)

X,Z=np.meshgrid(x,z)
r=np.array([X,0,Z],dtype=object)
```

In the last lines, we defined an array of X,0,Z, where X and Z are already 2-dimensional array. This finally gives an array 3D vectors, which we can use to calculate the electric field at any point in space. If we want to plot the electric field, we have to calculate the real part of the complex values, as the electric field is a physical quantity, which is always real. There is not much to see for a plane wave in the intensity plot, as the intensity of a plane wave is constant in space. Yet, if you want to plot it, you have to calculate the magnitude square of the electric field, e.g.

$$I \propto |E|^2 \tag{8}$$

```
#| autorun: false
#| fig-align: "center"
plt.figure(figsize=get_size(8,8))

field=plane_wave(k,omega0,r,4e-16)

extent = np.min(z)*1e6, np.max(z)*1e6,np.min(x)*1e6, np.max(x)*1e6

plt.imshow(np.real(field.transpose()),extent=extent,vmin=-1,vmax=1,cmap='seismic')
plt.xlabel('z-position [µm]')
plt.ylabel('x-position [µm]')

plt.tight_layout()
plt.show()
```

Plane wave propagation

The above graph shows a static snapshot of the plane wave at a time $t = 0$. We know, however, that a plane wave is propagating in space and time. Since we know how to animate things, we may do that using the `ipycanvas` module.

```
x=np.linspace(-2.5e-6,2.5e-6,300)
z=np.linspace(0,5e-6,300)

X,Z=np.meshgrid(x,z)
r=np.array([X,0,Z],dtype=object)
canvas = Canvas(width=300, height=300,sync_image_data=False)
display(canvas)
```

To do the animation I use a little trick to get the same color map as in the matplotlib plotting. The function below uses the matplotlib color map `seismic` and the corresponding mapping of values with a given minimum `vmin` and maximum `vmax` value. The mapping is done in the animation function with `c=m.to_rgba(tmp)`.

```
#normalize the color map to a certain value range
norm = mpl.colors.Normalize(vmin=-1, vmax=1)

cmap = cm.seismic

# do the mapping of values to color values.
m = cm.ScalarMappable(norm=norm, cmap=cmap)
```

This is our animation function, where I provide time and the wavevector as arguments, such that we may change both parameters easily.

```
def animate(k,time):
    for t in time:
        field=plane_wave(k,omega0,r,t)
        tmp=np.real(field.transpose())
        c=m.to_rgba(tmp)
        with hold_canvas(canvas):
            canvas.put_image_data(c[:,:,:3]*255,0,0)
            #canvas.put_image_data(data*255,0,0)
        sleep(0.02)
```

With the call below, you may animate the wave now with different refractive indices.

```
eta=1.
kappa=0.0
n=eta+kappa*1j

k=n*k0*vec
time= np.linspace(0,5e-14,500)
animate(k,time)
```

Imaginary wave vector

If we now create a material, which has an imaginary part of the refractive index, we see that the amplitude decays and the wave fades.

```
#| autorun: false
x=np.linspace(-2.5e-6,2.5e-6,300)
z=np.linspace(0,5e-6,300)
X,Z=np.meshgrid(x,z)
r=np.array([X,0,Z],dtype=object)
plt.figure(figsize=get_size(16,8))

eta=1.5
kappa=0.1
n=eta+kappa*1j
vec=np.array([0.,0.,1.])
vec=vec/np.sqrt(np.dot(vec,vec))
k=n*k0*vec

field=plane_wave(k,omega0,r,0)
extent = np.min(z)*1e6, np.max(z)*1e6,np.min(x)*1e6, np.max(x)*1e6

plt.subplot(1,2,1)
plt.title('electric field')
plt.imshow(np.real(field.transpose()),extent=extent,vmin=-1,vmax=1,cmap='seismic')
plt.xlabel('z-position [µm]')
plt.ylabel('x-position [µm]')

plt.subplot(1,2,2)
plt.title('intensity')
plt.imshow(np.abs(field.transpose())**2,extent=extent,cmap='gray_r')
plt.xlabel('z-position [µm]')
plt.ylabel('x-position [µm]')
plt.tight_layout()
plt.show()
```

The above plots show the electric field amplitude in the x-z plane. We may also have a look the field amplitude and intensity as a function of the z-position by chosing a single x-value. In the plot below, you may notice two things. The first is, that the wave decays exponentially with distance z. Intensity and field decay with different decay length. The field decays with $\exp(-\kappa * k_0 z)$ while the intensity of cause decays twice as fast $\exp(-2\kappa * k_0 z)$ due to the fact the the intensity is the square of the electric field.

```
#| autorun: false
plt.figure(figsize=get_size(12,5))
plt.plot(z*1e6,np.real(field[:,50]))
plt.plot(z*1e6,np.abs(field[:,50]),'k--', label='field amplitude')
plt.plot(z*1e6,np.abs(field[:,50])**2,'--',label='intensity')
plt.xlabel('z [µm]')
plt.ylabel('E,I')
plt.legend()
plt.tight_layout()
plt.show()
```

Animation

Of course, we should not miss the animation.

```
display(canvas)
```

```
k=n*k0*vec
time= np.linspace(0,5e-14,500)
animate(k,time)
```

Interference of two plane waves

It is not very difficult to calculate from the definitions we did above now the interference of two plane waves, which have different directions of the wavevector. The total field in space is then just the sum of the two fields

$$\vec{E} = \vec{E}_1 + \vec{E}_2 \tag{9}$$

The interesting thing is now to look at the intensity which

$$I \propto |\vec{E}|^2 = |\vec{E}_1|^2 + |\vec{E}_2|^2 + \vec{E}_1^* \vec{E}_2 + \vec{E}_2^* \vec{E}_1 \tag{10}$$

```
#| autorun: false
x=np.linspace(-5e-6,5e-6,200)
z=np.linspace(0,10e-6,200)

X,Z=np.meshgrid(x,z)
r=np.array([X,0,Z],dtype=object)

vec=np.array([0.,0.,1.])
vec=vec/np.sqrt(np.dot(vec,vec))
k1=k0*vec

eta=1.
kappa=0.0
n=eta+kappa*1j

k=n*k0*vec
time= np.linspace(0,5e-14,500)

k=n*k0*vec

vec=np.array([1.,0.,1.])
vec=vec/np.sqrt(np.dot(vec,vec))
k2=k0*vec


field1=plane_wave(k1,omega0,r,0)
field2=plane_wave(k2,omega0,r,0)
field=field1+field2

plt.figure(figsize=get_size(16,16))
plt.subplot(2,2,1)
extent = np.min(z)*1e6, np.max(z)*1e6,np.max(x)*1e6, np.min(x)*1e6
plt.imshow(np.real(field1.transpose()),extent=extent,cmap='seismic')
plt.title('Wave 1')
plt.xlabel('z [µm]')
plt.ylabel('x [µm]')
plt.subplot(2,2,2)
extent = np.min(z)*1e6, np.max(z)*1e6,np.max(x)*1e6, np.min(x)*1e6
plt.imshow(np.real(field2.transpose()),extent=extent,cmap='seismic')
plt.title('Wave 2')
plt.xlabel('z [µm]')
plt.ylabel('x [µm]')

plt.subplot(2,2,3)
extent = np.min(z)*1e6, np.max(z)*1e6,np.max(x)*1e6, np.min(x)*1e6
plt.imshow(np.real(field.transpose()),extent=extent,cmap='seismic')
plt.title('Sum')
plt.xlabel('z [µm]')
plt.ylabel('x [µm]')

plt.subplot(2,2,4)
extent = np.min(z)*1e6, np.max(z)*1e6,np.max(x)*1e6, np.min(x)*1e6
plt.imshow(np.abs(field.transpose())**2,extent=extent,cmap='gray')
plt.title('Intensity')
plt.xlabel('z [µm]')
plt.ylabel('x [µm]')

plt.tight_layout()
plt.show()
```

While the field pattern still looks complicated, the intensity pattern is just a set of bright lines.

```
#| autorun: false
plt.figure(figsize=get_size(12,8))
plt.plot(x*1e6,np.abs(field.transpose()[:,-1])**2)
plt.xlabel('distance [µm]')
plt.ylabel('intensity')
plt.tight_layout()
plt.show()
```

Advanced Topics: Plane wave at a boundary

We want to go a bit further now and have a look at the wave at a boundary between vaccum and glass for example. At this boundary, the electromagnetic wave is reflected and refracted such that two new wavevectors arise. These are easily calculated by the law of reflection and the law of refraction. Besides that, also the amplitude of the waves change. To calculate the field we need the so-called Fresnel equations.

Fresnel equations

When electromagnetic waves hit a boundary, they will be reflected and refracted. The amplitude of the reflected and refracted wave is determined by the refractive index of the two materials, the angles and the polarizations. For the latter we differentiate between a polarization in the incident plane (the p-polarization) and perpendicular to the incident plane (s-polarization).

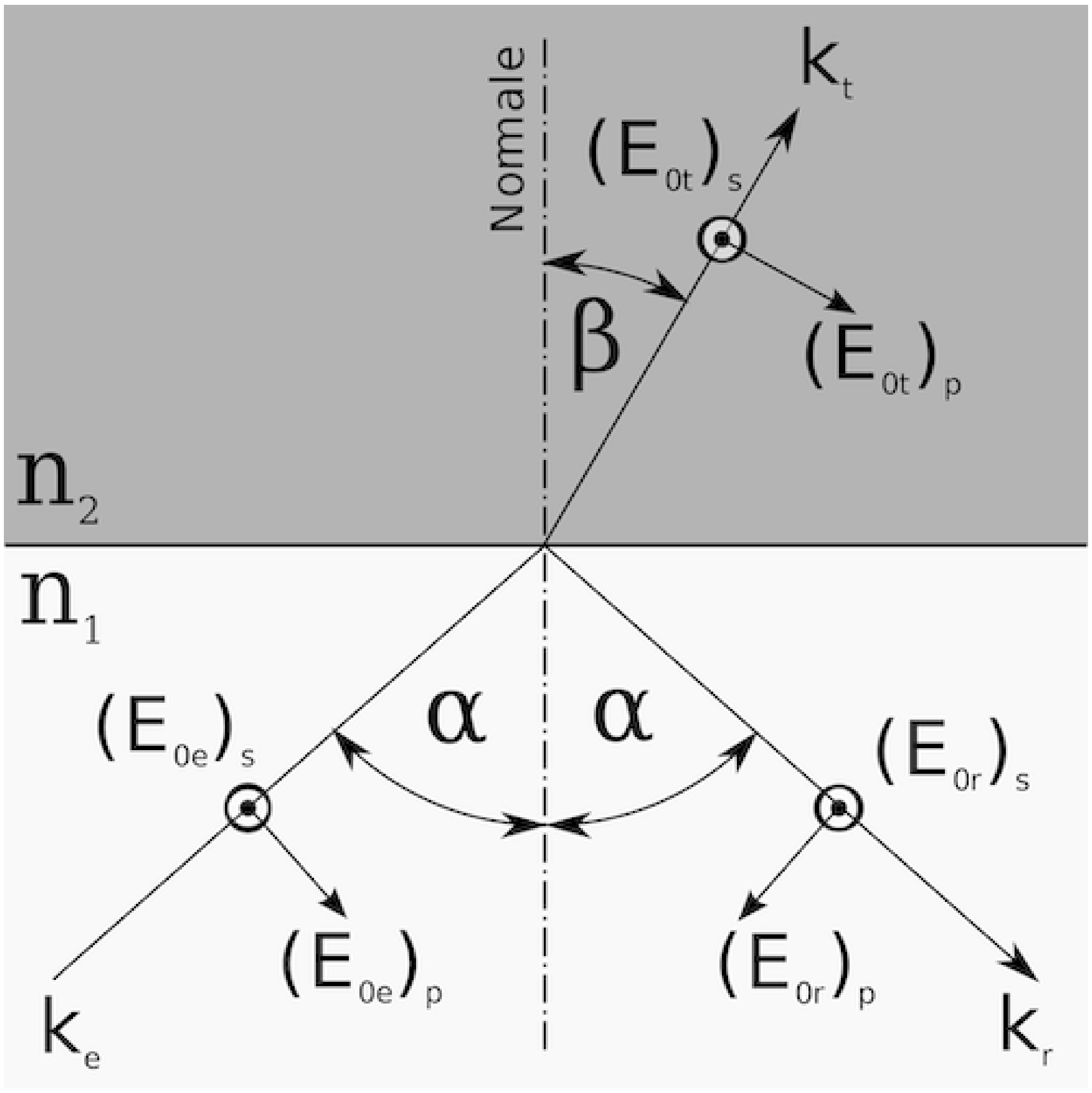

Figure 1: Fresnel

For each of the polarization we in general obtain a coeffcient for the reflection and one for the refraction. To make our calculation a bit simpler, we will assume only s-polarization. Then the two Fresnel coefficients are calculated as

$$\left(\frac{E_{0t}}{E_{0e}}\right)_s = t_s = \frac{2n_1 \cos\alpha}{n_1 \cos\alpha + n_2 \cos\beta} \tag{11}$$

$$\left(\frac{E_{0r}}{E_{0e}}\right)_s = r_s = \frac{n_1 \cos\alpha - n_2 \cos\beta}{n_1 \cos\alpha + n_2 \cos\beta} \tag{12}$$

where α and β are the incident and refraction angles, respectively. Note that the Fresnel coefficients are for the amplitudes and can be negative to account for a phase jump by π. To obtain the coefficients for the intensities, one has to square the Fresnel coefficients. To bring everything correctly together, we therefore have to define a number of things. We will need a function calculating the outgoing angle from Snells law. And we need at least two functions calculating the reflection and transmission coefficient for one polarization. We use the s-polarization, where the electric field is always parallel to the interface.

```
#| autorun: false
## snells law
def snell(n1,n2,alpha):
    tmp=n1*np.sin(alpha)/n2
    return(np.arcsin(tmp,dtype=complex))

## reflection coefficient
def rs(n1,n2,alpha,beta):
    return((n1*np.cos(alpha)-n2*np.cos(beta))/((n1*np.cos(alpha)+n2*np.cos(beta))))

## transmission coefficient
def ts(n1,n2,alpha,beta):
    return(2*n1*np.cos(alpha)/((n1*np.cos(alpha)+n2*np.cos(beta))))
```

With the definition of the Fresnel coefficients, we may now plot the reflection and the transmission coefficients. Note that the sum of reflection and transmission coefficients for the intensities have to add up to one if there is no absorption.

```
#| autorun: false
n1=1.5
n2=1
alpha=np.linspace(0,np.pi/2,100)
beta=snell(n1,n2,alpha)

plt.figure(figsize=get_size(16,8))
plt.subplot(1,2,1)
plt.title('reflection')
plt.plot(alpha,np.abs(rs(n1,n2,alpha,beta))**2)
plt.xlabel(r"$\alpha$")
plt.ylabel(r"$R_{s}$")
plt.subplot(1,2,2)
plt.title('transmission')
plt.plot(alpha,1-np.abs(rs(n1,n2,alpha,beta))**2)
plt.xlabel(r"$\alpha$")
plt.ylabel(r"$T_{s}$")
plt.tight_layout()
plt.show()
```

Incident wave

We want to study the electric fields and the intensities at various angles. The most interesting one, is a case where we have total internal reflection. This happens, if light is propagating from the higher refractive index to a lower refractive index. If we start in glass ($n_1 = 1.5$) and transmit to vacuum $n_2 = 1$, then at all angles above $\theta_c = \sin^{-1}(n_2/n_1) = 41.810314895778596$ are total internally reflected.

```
#| autorun: false
x=np.linspace(-5e-6,5e-6,500)
z1,z2=np.linspace(-5e-6,0,250),np.linspace(0,5e-6,250)
X,Z1=np.meshgrid(x,z1)


X,Z2=np.meshgrid(x,z2)
r1=np.array([X,0,Z1],dtype=object)
r2=np.array([X,0,Z2],dtype=object)
```

We may now specify or calculate the corresponding wavevectors for an incident angle of 45°. In general all waves (reflect, refracted) have to match with their phase at the boundary. If the boundary is along the x-direction, we therefore have

$$k_{x,in} = k_{x,r} = k_{x,t} \tag{13}$$

This fixes one component of all wavevectors in the plane. What is then missing, is the z-component of the wavevectors. The incident wavevector is providing $k_{z,in}$.

```
#| autorun: false
alpha=43*np.pi/180
```

```
#| autorun: false
vec=np.array([np.sin(alpha),0.,np.cos(alpha)])
k1=1.5*k0*vec
print(k1,'[1/m]')
```

Reflected wave

For the reflected wave the z-component of the wavevector is just flipped in sign, e.g. $k_{z,r} = -k_{z,in}$.

```
#| autorun: false
k2=1.5*k0*vec
k2[2]=k2[2]*-1
print(k2,'[1/m]')
```

Refracted wave

The magnitude of the z-component of the transmitted wave can be obtained from the conservation of momentum. The momentum of the wave is proportional to the magnitude of the wavevector on both sides.

$$k_1^2 = k_2^2 \tag{14}$$

which is, due to $k = nk_0$ the same as

$$n_1^2(k_{0x,in}^2 + k_{0z,in}^2) = n_2^2(k_{0x,t}^2 + k_{0z,t}^2) \tag{15}$$

from which we get

$$k_{0z,t} = \pm\frac{1}{n_1}\sqrt{n_2^2 k_{0z,in}^2 - (n_1^2 - n_2^2)k_{0x}^2} \tag{16}$$

If we go from a medium with high refrective index to a lower one, the second term in the root may surpass the first one and the whole solution will become imaginary. The wave in the lower refractive index medium n_2 is then evanescent.

```
#| autorun: false
k3=np.zeros([3],dtype=complex)
k3[0]=k1[0]
k3[2]=np.sqrt(n2**2*k1[2]**2+(n2**2-n1**2)*k1[0]**2,dtype=complex)/n1

print(k3,'[1/m]')
```

The total field thus containes three components. In medium 1, the field consists of the incident and the reflected wave. In medium 2, we just have the transmitted wave, with a possible evanescent solution.

```
#| autorun: false
field=np.zeros([500,500],dtype=complex)
field1=plane_wave(k1,omega0,r1,0)
field2=plane_wave(k2,omega0,r1,0)
field3=plane_wave(k3,omega0,r2,0)

beta=snell(n1,n2,alpha)
r=rs(n1,n2,alpha,beta)
t=ts(n1,n2,alpha,beta)

field[:250,:]=field1+r*field2
field[250:,:]=t*field3
```

The plots below show the electric field on the left side and the intensity on the right side. Interestingly, the intensity is that of a standing wave in medium 1, while it is just decaying in medium 2. Note that the electric field is oscillating along the interface in medium 2 but not at all in z-direction. This means that there is no energy transport along the z-direction anymore.

```
#| autorun: false
plt.figure(figsize=get_size(16,8))
extent = np.min(z1)*1e6, np.max(z2)*1e6,np.max(x)*1e6, np.min(x)*1e6
plt.subplot(1,2,1)
plt.imshow(np.real(field.transpose()),extent=extent,cmap='seismic')
plt.title('electric field')
plt.xlabel('z [µm]')
plt.ylabel('x [µm]')

plt.subplot(1,2,2)
plt.imshow(np.abs(field.transpose())**2,extent=extent,cmap='gray')
plt.title('intensity')
plt.xlabel('z [µm]')
plt.ylabel('x [µm]')

plt.tight_layout()
plt.show()
```

We will also have a look ath the propagation of the wave yb defining our animation.

```
canvas = Canvas(width=500, height=500,sync_image_data=True)
display(canvas)
```

```
def animate(k,time):
    for t in time:
        field=np.zeros([500,500],dtype=complex)
        field1=plane_wave(k1,omega0,r1,t)
        field2=plane_wave(k2,omega0,r1,t)
        field3=plane_wave(k3,omega0,r2,t)

        beta=snell(n1,n2,alpha)
        r=rs(n1,n2,alpha,beta)
        t=ts(n1,n2,alpha,beta)

        field[0:250,:]=field1+r*field2
        field[250:,:]=t*field3
        tmp=np.real(field.transpose())

        c=m.to_rgba(tmp)
        with hold_canvas(canvas):
            canvas.put_image_data(c[:,:,:3]*255,0,0)
        sleep(0.02)
```

```
time= np.linspace(0,1e-14,100)
animate(k,time)
```

As it is apparent from our simulation, the wave is longitudinal in medium 2 at this angle. Try to modify the incident angles yourself to see if the wave becomes propagating in medium 2.
In the last plot, we will have a look at the intensity in medium 1 and medium 2. What is nicely visible, is that the intensity decays in medium 2 with increasing distance. As compared to the absorbing case, there is not oscillation of the field in the z-direction, hence no energy transfer. Convince yourself that this is indeed an exponential decay by using the appropriate `semilog` plot.

```
#| autorun: false
n=500
field=np.zeros([n,n],dtype=complex)
field1=plane_wave(k1,omega0,r1,0)
field2=plane_wave(k2,omega0,r1,0)
field3=plane_wave(k3,omega0,r2,0)

beta=snell(n1,n2,alpha)
r=rs(n1,n2,alpha,beta)
t=ts(n1,n2,alpha,beta)

field[0:int(n/2),:]=field1+r*field2
field[int(n/2):,:]=t*field3

plt.figure(figsize=get_size(12,5))
plt.subplot(1,2,1)
plt.plot(z1*1e6,np.abs(field.transpose())[int(n/2),:int(n/2)]**2)
plt.xlabel('distance from interface [µm]')
plt.ylabel('field intensity')

plt.subplot(1,2,2)
plt.plot(z2*1e6,np.abs(field.transpose())[int(n/2),int(n/2):]**2)
plt.xlabel('distance from interface [µm]')
plt.ylabel('field intensity')


plt.tight_layout()
plt.show()
```