Politecnico di Torino

Industrial Photonics

Design of ARCs

Ariel Priarone

s274149

Contents

List of Figures

List of Tables

1 Introduction

This assignment is about a laser beam delivery system: it is asked to design a dichroic mirror that selectively reflect the laser beam into the working piece, while transmitting the light in the visible spectrum passing in the other direction so that it can reach a camera used for process control. The system is studied for two lasers, both working at $\lambda_0 = 1070$ nm:

Laser	Diameter [mm]	Power [kW]	M^2	BPP [mm·mrad]	Polarization
1	0.014	1	1.20	-	random
2	0.100	5	-	5	random

2 Function

Since a dichroic mirror is usually a device with a big number of interfaces, i will write a function that compute the input reflectivity for any multi layer structure. Since such a function is already provided for matlab in the course material, i will adopt the same procedure, implementing a function in pyhton, following the nomenclature shown in **Figure 1**.

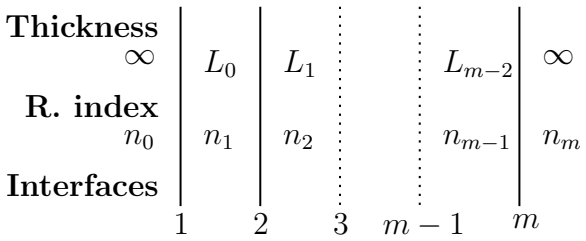

Figure 1: *Nomenclature of the function*

```python
def mLayerReflectivity(n,L,lam0,theta=0,pol='TE'):
    # compute the input reflectivity for a multilayer stack of materials (first and last mediums
    ↪ infinetely thick)
    # imputs:
    #   n     : array-like,       array of reflactive indeces (constant, no chromatic aberration
    ↪ considered)
    #   L     : array-like,       array of thickneses of layers (len(L)=len(n)-2)
    #   lam0  : array-like,       wavelength in vacuum to be tested
    #   theta : float,            angle of incidence (deg), default=0 deg
    #   pol   : str,              "TE" if perpendicular polarization, "TM" for parallel
    ↪ polarization, default = "TE"
    # return:
    #   R     : array-like        input reflectivity

    if not isinstance(n, np.ndarray):        # convert to np.array if needed
        n   =   np.array(n)
    if not isinstance(L, np.ndarray):        # convert to np.array if needed
        L   =   np.array(L)
    if not isinstance(lam0, np.ndarray):     # convert to np.array if needed
        lam0   =   np.array(lam0)
    theta=theta*np.pi/180           # from deg to rad
    if not len(L)==(len(n)-2):      # check consistency of dimensions
```

```
        raise Exception("len(L)!=len(n)-2")
    Z0  =   120*np.pi               # vacuum impedence
    R=[]                            # initialize empty reflectivity
    for lam in lam0:                # iterate for all wavelength of interest
        k0      =   2*np.pi/lam     # k0 inital wave number
        k_vect  =   k0*(n[1:-1]**2-n[0]**2*np.sin(theta)**2)**0.5
        for ii in range(0,len(k_vect)):
            if np.imag(k_vect[ii])>0:        # fix k_z=-j*alpha
                k_vect[ii]=k_vect[ii].conjugate()
        # compute the z_infs
        match pol:
            case "TE":
                zinf = Z0/(n**2-n[0]**2*np.sin(theta)**2)**(0.5)
            case "TM":
                zinf = Z0/(n**2)*(n**2-n[0]**2*np.sin(theta)**2)**(0.5)
            case _ :
                raise Exception('pol is neither "TE" nor "TM"')
        for ii in range(0,len(zinf)):
            if np.imag(zinf[ii])<0:         # fix z_inf
                k_vect[ii]=k_vect[ii].conjugate()
        rho = (zinf[1:]-zinf[0:-1])/(zinf[1:]+zinf[0:-1])
        gamma=[rho[-1]]                    # initialize gamma array - last reflective coefficient
        ↪ is rho
        for r,k,l in zip(reversed(rho[:-1]),reversed(k_vect),reversed(L)):# reverse to propagate
        ↪ backward
            gamma.append((r+gamma[-1]*np.exp(-2j*k*l))/(1+r*gamma[-1]*np.exp(-2j*k*l)))
        R.append(abs(gamma[-1]**2*100))
    return np.array(R)  # reflectivity [%]
```

3 Initialize the script

In order to compute the solutions of this assignment, i used a python script, that i will include in this document. The following is just the preamble of that script. The script is available at `https://github.com/arielpriarone/IndustrialPhotonics/tree/main/Assignment3`.

```
import matplotlib.pyplot as plt
import matplotlib.colors as color
import numpy as np
import matplotlib
import tikzplotlib
import scipy.integrate as integrate
from matplotlib.lines import Line2D
matplotlib.use('Qt5Agg')

def tikzplotlib_fix_ncols(obj):
    """
    workaround for matplotlib 3.6 renamed legend's _ncol to _ncols, which breaks tikzplotlib
```

```
    """
    if hasattr(obj, "_ncols"):
        obj._ncol = obj._ncols
    for child in obj.get_children():
        tikzplotlib_fix_ncols(child)
```

4 Mirror design

The mirror should be designed to obtain ideally a reflectivity $R = 1$ at a wavelength $\lambda = \lambda_0$. In reality obtaining a really high reflectivity is crucial because, in high power applications, the portion of power that is not reflected, is transmitted and will be absorbed by the glass substrate, in part, and then by some other component in the chassis of the device.

On the other hand, obtaining an ideal reflectivity $R = 0$ among all the visible spectrum is quite impossible in reality, but in this case obtaining a good performance is less crucial because it will only affect the amount of radiation that reaches the camera (it is possible to compensate with high ISO).

A dichroic mirror is a device built stacking a large number of quarter wavelength layers (QWOT). The input reflectivity is function of the number of layers, the thicknesses, the refractive indices and, in the case of an angle of incidence not null, also on the polarization of the light.

I used the following script to analyze the behaviour of the mirror, ad wavelength around the one of design λ_0 (used to tune the QWOT layers). The analysis is carried out for different angle of incidence and different numbers of bi-layers, with random polarization (average reflectivity of the one obtained for both polarizations), as reported in **Figure 2**.

It's evident that increasing the number of bi-layers, the peak of the reflectivity increases, and for 80 bi-layers, the peak is practically 100% in a reasonable band around the design frequency (for $\theta = 0$). The materials were chosen such that the refractive indices[1] were not too different, because using two materials with close refractive indices increase the spacing of the peaks in the frequency periodic pattern of the reflectivity. In this case the materials are **silicon dioxide** and **magnesium fluoride**. As substrate a commonly available BK7 glass is considered.

```
# %% data of the problem
lam0    =   1070*10**(-9)               # design wavelength
nair    =   1                           # left medim refr index
nH      =   1.46                        # refr index layer H (silicon dioxide)
nL      =   1.38                        # refr index layer L (magnesium fluoride)
ng      =   1.5                         # right medim refr index (glass)
dH      =   lam0/4/nH                   # thickness layer H
dL      =   lam0/4/nL                   # thickness layer L

lam     =   np.linspace(lam0*0.80,lam0*1.2,1000)

fig, axs=plt.subplots(3, 1, sharex=True, sharey=True)
fig.tight_layout()
ii = 0
for ncouples in [15,30,80]:
```

[1] from now on i will neglect the chromatic dispersion, so the refractive indices of the materials will be constant at any wavelength

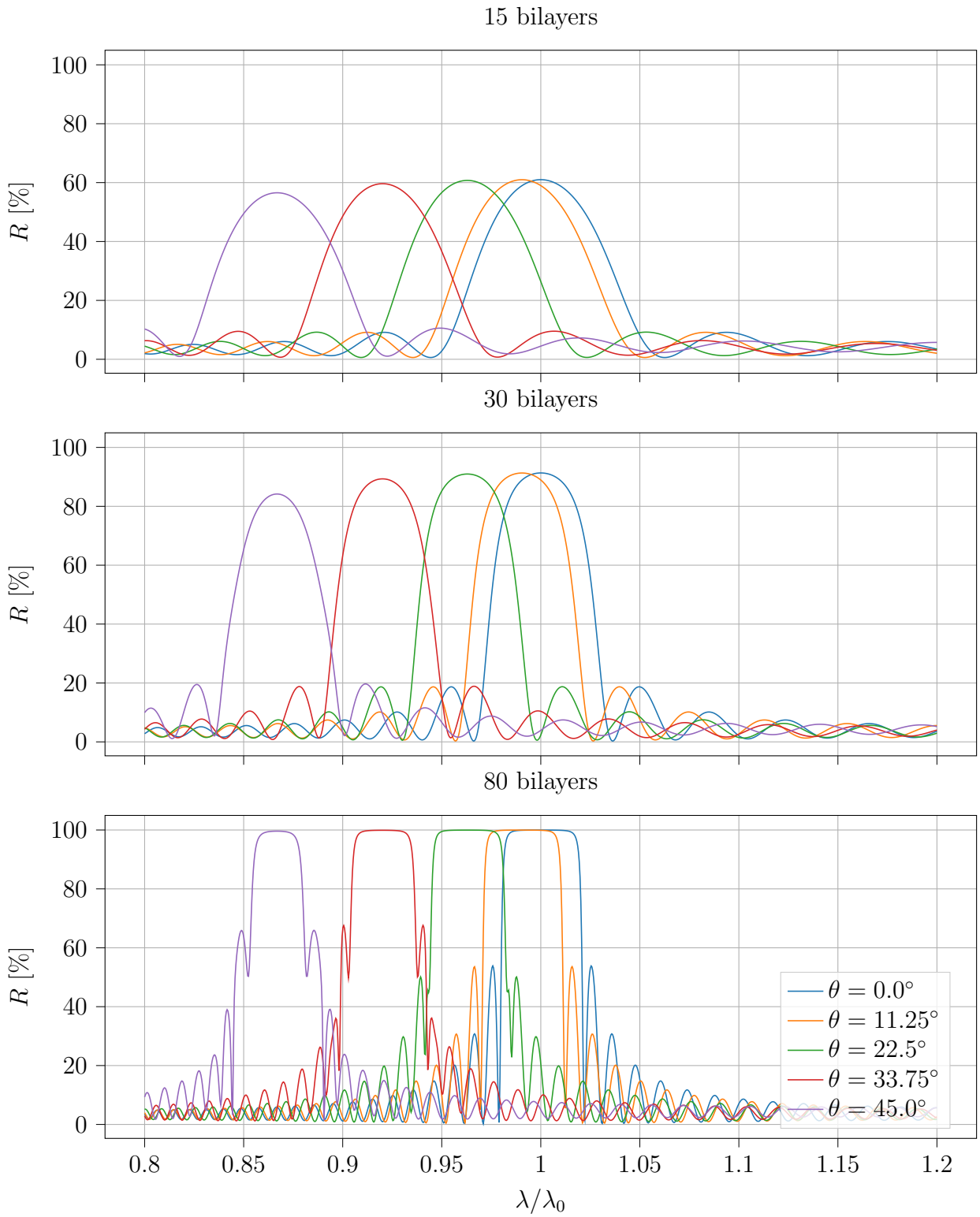

Figure 2: *response of a dichroic mirror for some different number of bilayers and values of the AOI*

```
    for theta in np.linspace(0,45,5):
        n=np.tile([nH, nL],ncouples)    # concatenate bilayers indeces
        d=np.tile([dH, dL],ncouples)    # concatenate bilayers thicknesses
        n=np.concatenate(([nair],n))    # add air medium
        n=np.concatenate((n,[ng]))      # add glass medium

        Rtm=mLayerReflectivity(n,d,lam,theta,'TM')  # tm polarization reflectivity
        Rte=mLayerReflectivity(n,d,lam,theta,'TE')  # te polarization reflectivity

        axs[ii].plot(lam/lam0,(Rte+Rtm)/2,label=f'$\\theta={round(theta,3)}^\\circ$')
        print(f'$\\theta={round(theta,3)}^\\circ$')
    axs[ii].grid(True, 'Both')
    axs[ii].set_title(f'{ncouples} bilayers')
    axs[ii].set_ylabel('$R$ [%]')
    ii+=1
axs[-1].set_xlabel('$\\lambda/\\lambda_0$')
axs[-1].legend(loc='lower right')
tikzplotlib_fix_ncols(fig)
tikzplotlib.save('Assignment3/fig1.tex',axis_width='0.9\\textwidth',axis_height ='7cm')
```

4.1 Angle dependency

At this point the mirror is designed for an AOI $\theta = 0$, but the problem constrain is to use a AOI $\theta = 45°$. Looking at the **Figure 2**, it's possible to see that at this angle of incidence, the peak of the reflectivity has moved towards lower wavelength, so that the design has to be repeated for a corrected wavelength λ_0':

$$\frac{\lambda_{peak}}{\lambda_0} = 0.867 = \frac{\lambda_0}{\lambda_0'} \quad \Longrightarrow \quad \lambda_0' = \frac{\lambda_0}{0.867}$$

The following script re-design the mirror for the corrected wavelength λ_0', and check the reflectivity for both the polarization. The result is displayed in **Figure 3**. At λ_0, the average reflectivity is 99.6%, this means that for the two laser, 4W and 20W will be transmitted thru the mirror, respectively.

```
# %% design of the dichroic mirror
# data of the problem
lam0_c  =   1070*10**(-9)/0.867         # design wavelength - compensated for the AOI
dH      =   lam0_c/4/nH                 # thickness layer H
dL      =   lam0_c/4/nL                 # thickness layer L
ncouples =  80                          # how many bilayers
theta   =   45                          # AOU - degrees
lam     =   np.linspace(lam0*0.9,lam0*1.1,1000) # around laser wavelength
fig, axs=plt.subplots()
fig.tight_layout()
n=np.tile([nH, nL],ncouples)    # concatenate bilayers indeces
d=np.tile([dH, dL],ncouples)    # concatenate bilayers thicknesses
n=np.concatenate(([nair],n))    # add air medium
n=np.concatenate((n,[ng]))      # add glass medium
Rtm=mLayerReflectivity(n,d,lam,theta,'TM')  # tm polarization reflectivity
```

```
Rte=mLayerReflectivity(n,d,lam,theta,'TE')  # te polarization reflectivity
axs.plot(lam,(Rte),label=f'$TE$')
axs.plot(lam,(Rtm),label=f'$TM$')
axs.plot(lam,(Rte+Rtm)/2,label=f'$average$')
axs.vlines([lam0],-5,105,colors='red',linestyles='dashdot')
axs.grid(True, 'Both')
axs.set_title(f'{ncouples} bilayers')
axs.set_ylabel('$R$ [%]')
axs.set_xlabel('$\\lambda$ [m]')
axs.legend(loc='upper right')
tikzplotlib_fix_ncols(fig)
tikzplotlib.save('Assignment3/fig2.tex',axis_width='0.9\\textwidth',axis_height ='7cm')
```

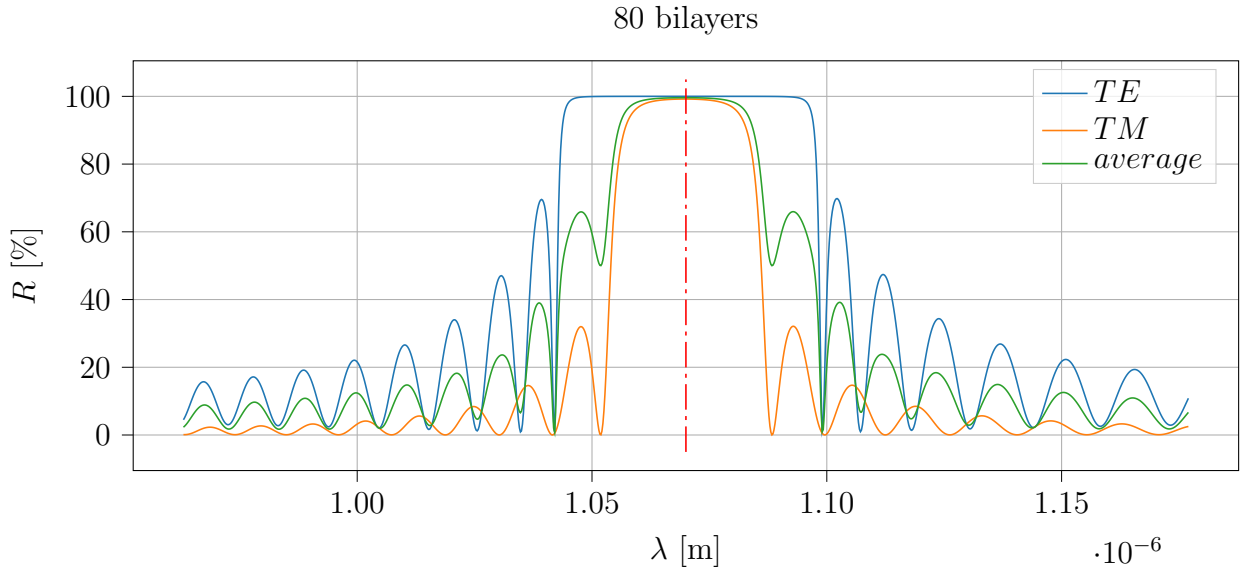

Figure 3: *response of the designed mirror around laser wavelength*

4.2 Check reflectivity in the visible spectrum

At this point it remains to check that the reflectivity is low enough in the visible spectrum range. The following script compute the reflectivity in this range. The result are reported in **Figure 4**. This values are not zero, but considering also that the material being welded will emit a lot of radiation in the visible spectrum, the reflectivity is low enough to allow for a camera to capture the image. The average reflectivity in the visible spectrum is 4%.

The computations are done with the following code:

```
# %% check reflectivity in visible range
# data of the problem
lam     =   np.linspace(380,700,1000)*10**(-9) # visible range
fig, axs=plt.subplots()
fig.tight_layout()
Rtm=mLayerReflectivity(n,d,lam,theta,'TM')  # tm polarization reflectivity
Rte=mLayerReflectivity(n,d,lam,theta,'TE')  # te polarization reflectivity
```

```
8
9  axs.plot(lam,(Rte),label=f'$TE$')
10 axs.plot(lam,(Rtm),label=f'$TM$')
11 axs.plot(lam,(Rte+Rtm)/2,label=f'$average$')
12 axs.grid(True, 'Both')
13 axs.set_ylabel('$R$ [%]')
14 axs.set_xlabel('$\\lambda$ [m]')
15 axs.legend(loc='upper right')
16 tikzplotlib_fix_ncols(fig)
17 tikzplotlib.save('Assignment3/fig3.tex',axis_width='0.9\\textwidth',axis_height ='7cm')
```

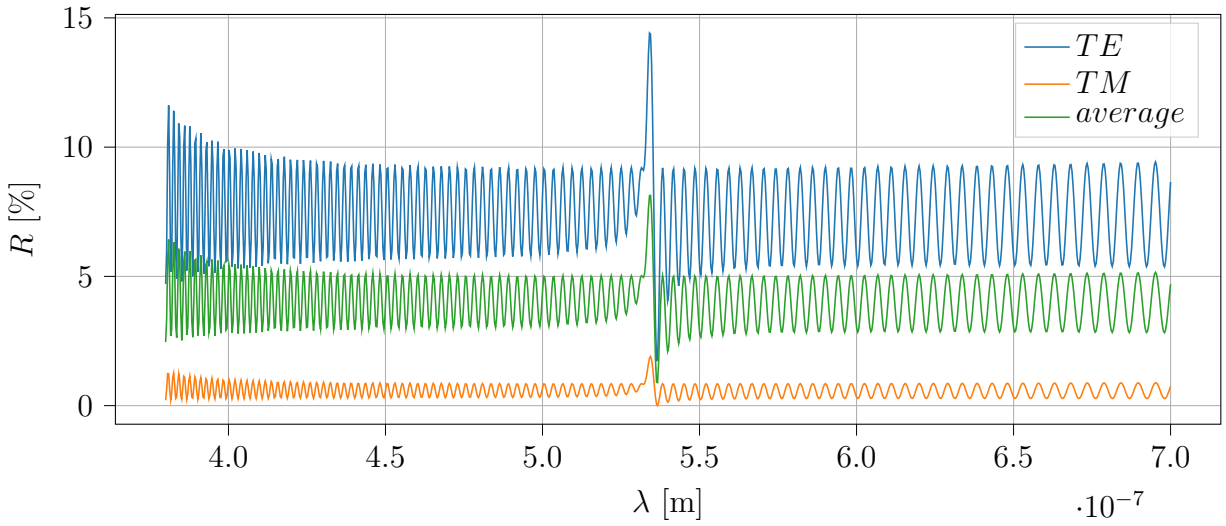

Figure 4: *response of the designed mirror in visible spectrum*

5 Irradiance distribution

At this point the assignment asks to verify if the two lasers are powerful enough to weld, so if they are able to provide a power density of at least 1kW/cm^2 at the focal point. This under the assumption that both lasers are equipped with a collimator that provide a collimated beam of 12mm and a focusing lens with focal length 200 mm.

Regarding the first laser, the quality factor is known:

$$M_{L1}^2 = 1.2$$

and for the second laser it can be computed knowing the beam parameter product:

$$BPP = 5\text{mm} \cdot \text{mrad} \implies M_{L2}^2 = \frac{BPP}{BPP_{gauss}} = \frac{BPP \cdot \pi}{\lambda_0} = 14.68$$

Knowing the quality factor, the spot size can be computed:

$$w_f = M^2 \frac{\lambda_0 \cdot f}{\pi \cdot w_l}$$

where w_l is the collimated radius before the lens with focal length f. The irradiance as function of the z axis and the radial position r follow as:

$$\frac{\mathrm{d}P}{\mathrm{d}\Sigma} = \frac{2 \cdot P}{\pi \cdot w(z)} e^{\frac{-2 \cdot r^2}{w(z)^2}} \tag{1}$$

Computing the irradiance at the focus point, the radial distribution is shown in **Figure 5**.

It is evident that the second laser, despite having higher power, produce a lower power density at the focusing point. This is due by the fact that the beam quality is really worse than the one of the first laser. In any case, also the second laser reaches the minimum requirement of $1\mathrm{kW/cm^2}$, but with a less sharp distribution, that will imply also a bigger spot size.

All computations carried out with the following code:

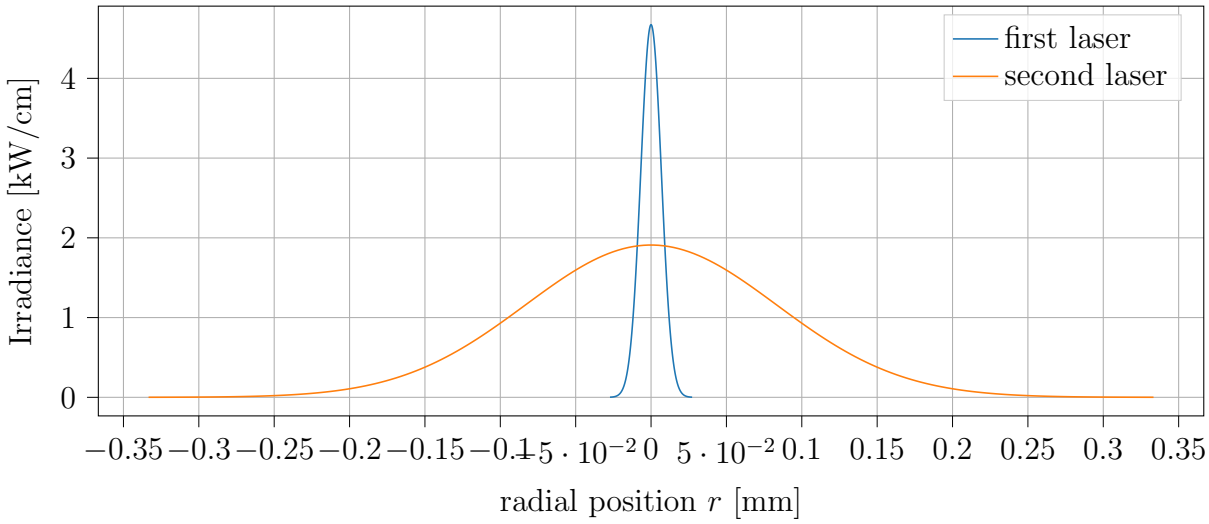

Figure 5: *irradiance distribution for the two lasers*

```
#%% compute the irradiance for the focus spot
# first laser
M2  = 1.2                          # quality factor
f   = 0.2                          # focal length of lens  [m]
wl  = 0.012/2                      # collimated beam radius[m]
wf  = M2*lam0*f/np.pi/wl           # focused spot radius   [m]
P   = 1000                         # laser power           [w]
r   = np.linspace(-2*wf,2*wf,200)  # radial distance from the focus point [m]
I   = 2*P/np.pi/wf*np.exp(-2*r**2/(wf**2)) # irradiance distribution [w/m^2]

fig, axs=plt.subplots()
fig.tight_layout()
axs.plot(r*10**3,I/10**7,label='first laser') # converted in kw/cm^2

# second laser
BPP = 5/10**6                      # beam parameter product [mm*rad]
M2  = BPP/lam0*np.pi               # quality factor
wf  = M2*lam0*f/np.pi/wl           # focused spot radius   [m]
P   = 5000                         # laser power           [w]

r   = np.linspace(-2*wf,2*wf,200)  # radial distance from the focus point [m]
```

```
I   = 2*P/np.pi/wf*np.exp(-2*r**2/(wf**2)) # irradiance distribution [w/m^2]

axs.plot(r*10**3,I/10**7,label='second laser')           # converted in kw/cm^2
axs.grid(True, 'Both')
axs.set_ylabel('Irradiance [kW/cm]')
axs.set_xlabel('radial position $r$ [mm]')
axs.legend()

axs.grid(True, 'Both')
axs.set_ylabel('Irradiance [kW/cm]')
axs.set_xlabel('radial position $r$ [mm]')
axs.legend()


tikzplotlib_fix_ncols(fig)
tikzplotlib.save('Assignment3/fig4.tex',axis_width='0.9\\textwidth',axis_height ='7cm')
```

6 Spot size

At this point the task is to study how real beams are propagated thru a lens system. In particular three cases are considered, a firs lens with 100mm focal length, and a second lens with 125mm, 150mm and 200mm focal length respectively.

In the previous assignment, a function to propagate a beam thru a system of three lenses, has been developed. The problem of having only two lenses can be solved setting the focal length of one of the three lenses to a number that approach infinity, in this case the second lens with $f_2 = s10^{50}$. The distance between the two remaining lens is almost irrelevant because the first lens, if placed at a distance from the beam waist equal to his focal length, climate the beam.

The result is shown in the **Figure 6**. As expected, the first laser, that has a smaller beam waist, is focused into smaller spot, but has also a narrower depth of focus. The solid lines in the figures represent the ideal Gaussian beam radius, while the shading represent the real beam radius, considering its quality factor.

On the other hand, the second laser start from a bigger waist, so in this setup is focused into bigger spots. To obtain the same spot size of the previous it would require much bigger lenses diameters and a firs lens with a greater focal length. In the **Figure 7** it is provided a detail view of the focusing points for both lasers.

The numeric results are resumed in the **Table 1**

Laser	Diameter [mm]	M^2	Spot size [mm]		
			$f_3 = 125$	$f_3 = 150$	$f_3 = 200$
1	0.014	1.20	0.009585	0.011502	0.015336
2	0.100	14.68	0.239468	0.287361	0.383149

Table 1: *Spot sizes for different configurations*

The code to calculate the spot sizes is the following.

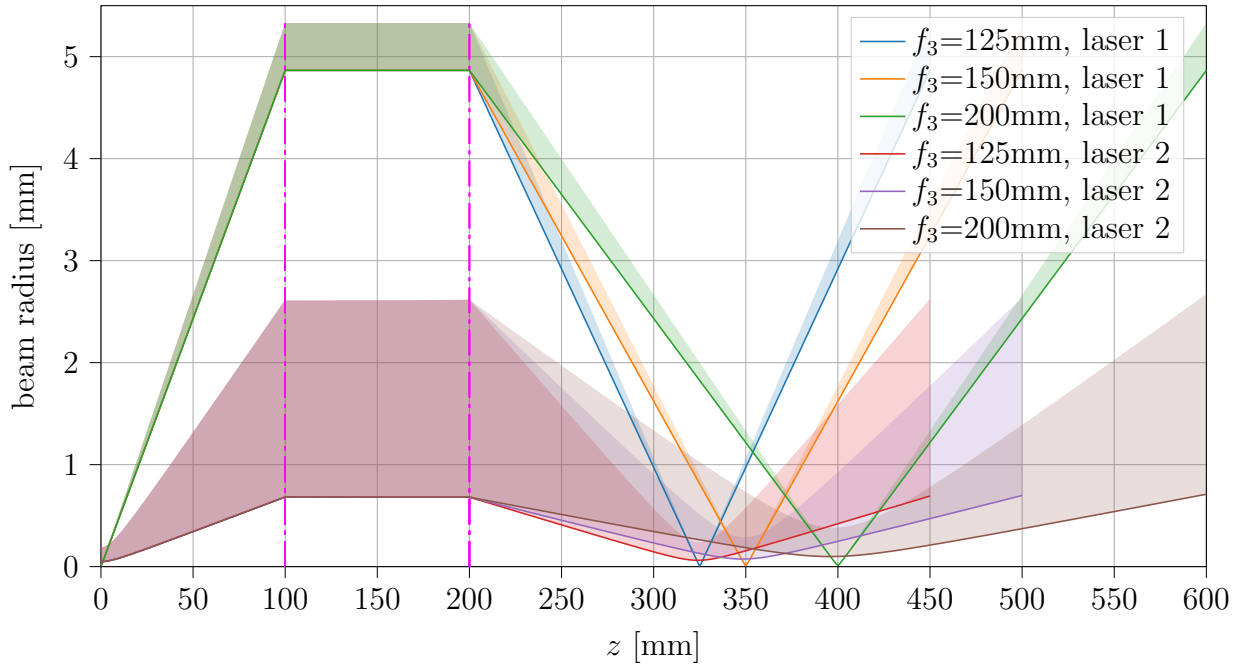

Figure 6: *focusing heads in different configurations for both lasers*

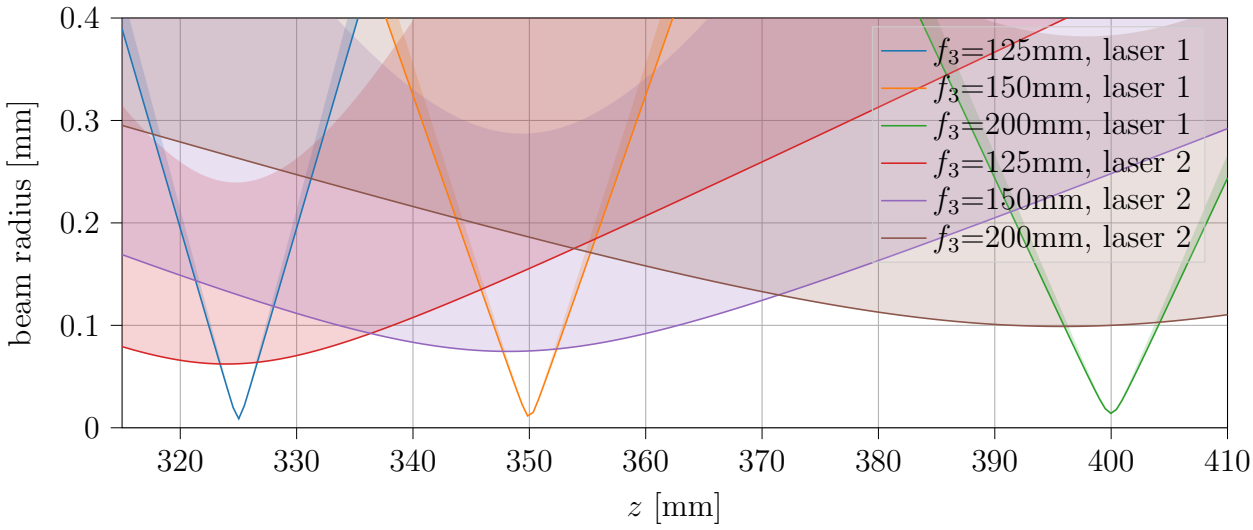

Figure 7: *detail of focusing heads in different configurations for both lasers*

```python
# %% study the lasers with three heads
# first laser
w0  =   14/2/10**3                      # approximate the waist with the core radius [mm]
MS  =   1.2                             # quality factor
(d0,d1,d2)  = (100,0,100)               # spacing configuration
(f1,f2,f3)  = (100,10**50,125)          # lenses configuration 1
fig, axs, Mag, d3, div, w_out, w_end =
↪  BeamExpander(lam0*10**3,w0,d0,d1,d2,f1,f2,f3,npoint=1001,MS=MS)
print(w_out*10**3)
(f1,f2,f3)  = (100,10**50,150)          # lenses configuration 2
fig, axs, Mag, d3, div, w_out, w_end =
↪  BeamExpander(lam0*10**3,w0,d0,d1,d2,f1,f2,f3,npoint=1001,MS=MS,fig=fig,axs=axs)
print(w_out*10**3)
(f1,f2,f3)  = (100,10**50,200)          # lenses configuration 3
fig, axs, Mag, d3, div, w_out, w_end =
↪  BeamExpander(lam0*10**3,w0,d0,d1,d2,f1,f2,f3,npoint=1001,MS=MS,fig=fig,axs=axs)
print(w_out*10**3)

# second laser
w0  =  100/2/10**3                      # approximate the waist with the core radius [mm]
BPP = 5/10**6                           # beam parameter product [m*rad]
MS  = BPP/lam0*np.pi                    # quality factor
(f1,f2,f3)  = (100,10**50,125)          # lenses configuration 1
fig, axs, Mag, d3, div, w_out, w_end =
↪  BeamExpander(lam0*10**3,w0,d0,d1,d2,f1,f2,f3,npoint=1001,MS=MS,fig=fig,axs=axs)
print(w_out*10**3)
(f1,f2,f3)  = (100,10**50,150)          # lenses configuration 2
fig, axs, Mag, d3, div, w_out, w_end =
↪  BeamExpander(lam0*10**3,w0,d0,d1,d2,f1,f2,f3,npoint=1001,MS=MS,fig=fig,axs=axs)
print(w_out*10**3)
(f1,f2,f3)  = (100,10**50,200)          # lenses configuration 3
fig, axs, Mag, d3, div, w_out, w_end =
↪  BeamExpander(lam0*10**3,w0,d0,d1,d2,f1,f2,f3,npoint=1001,MS=MS,fig=fig,axs=axs)
print(w_out*10**3)

axs.set_ylim([0,5.5])
axs.legend().remove
legend_elements = [Line2D([0], [0], color='#1f77b4', label='$f_3=125$, laser 1'),
                   Line2D([0], [0], color='#ff7f0e', label='$f_3=150$, laser 1'),
                   Line2D([0], [0], color='#2ca02c', label='$f_3=200$, laser 1'),
                   Line2D([0], [0], color='#d62728', label='$f_3=125$, laser 2'),
                   Line2D([0], [0], color='#9467bd', label='$f_3=150$, laser 2'),
                   Line2D([0], [0], color='#8c564b', label='$f_3=200$, laser 2')]
axs.legend(handles=legend_elements, loc='upper right')
tikzplotlib_fix_ncols(fig)
tikzplotlib.save('Assignment3/fig5.tex',axis_width='0.9\\textwidth',axis_height ='9cm')
```

7 Design a processing head

At this point, i will pretend that for the application, the spot size is more important that the depth of focus, so the second laser is the one that has worst performance.

I will aim tho design a head that provide the same spot size of the first laser with the current processing head. To do that i would need to compensate for both the higher beam quality factor and the higher initial beam waist.

This correction coefficient will be applied to the focal length of the leftmost lens:

$$c = \frac{w_2}{w_1} \cdot \frac{M_2}{M_1}$$

The results are shown in **Figure 8** and **Figure 9**, as usual the solid line is the Gaussian beam, the shade is the propagation of the real beam. This processing head produce the desired beam spot size, but has the problem of needing lenses with a diameter of at least 13cm, and the apparatus has a length of almost 3m. I can imagine that in a real situation, if the laser head needs to move, this design would be impractical.

As far as concern the numeric result, the designed head produce with the second laser, the same performance that the first laser had with the first head. The numeric results are resumed in **Table 2**.

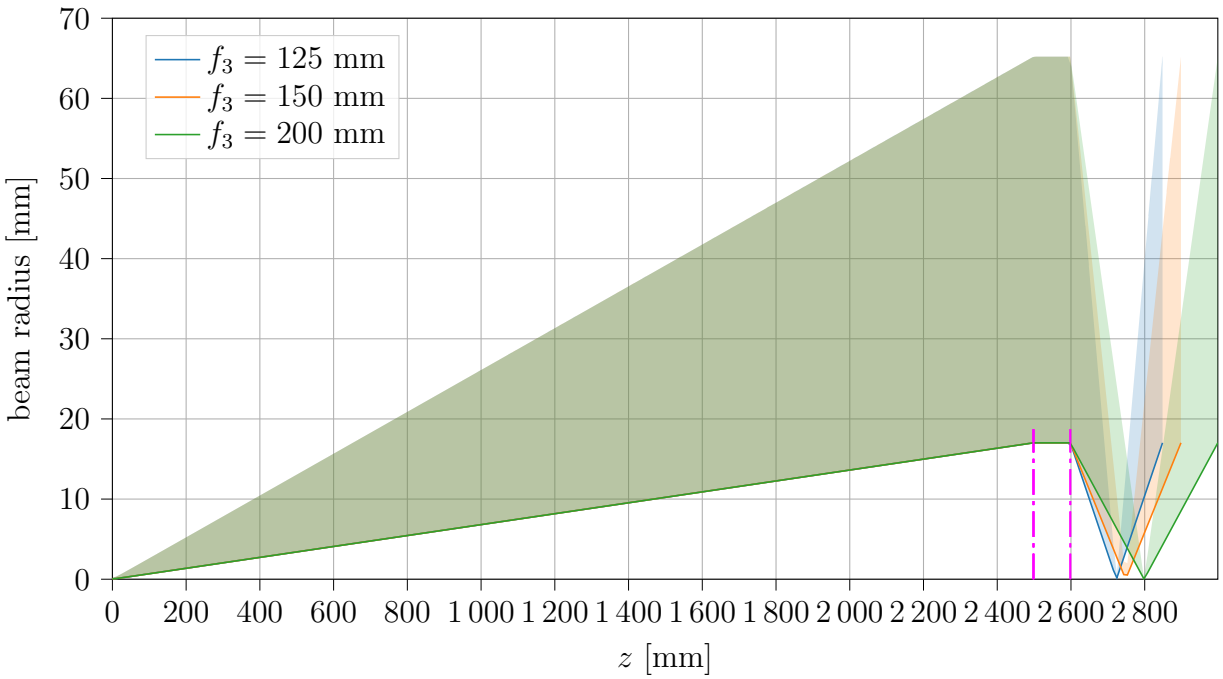

Figure 8: *designed focusing heads in different configurations for laser 2*

Laser	Diameter [mm]	M^2	Spot size [mm]		
			$f_3 = 125$	$f_3 = 150$	$f_3 = 200$
1	0.014	1.20	0.009585	0.011502	0.015336
2	0.100	14.68	0.009585	0.011502	0.015336

Table 2: *Spot sizes for different configurations*

as usual i paste below the code used to design the new processing head:

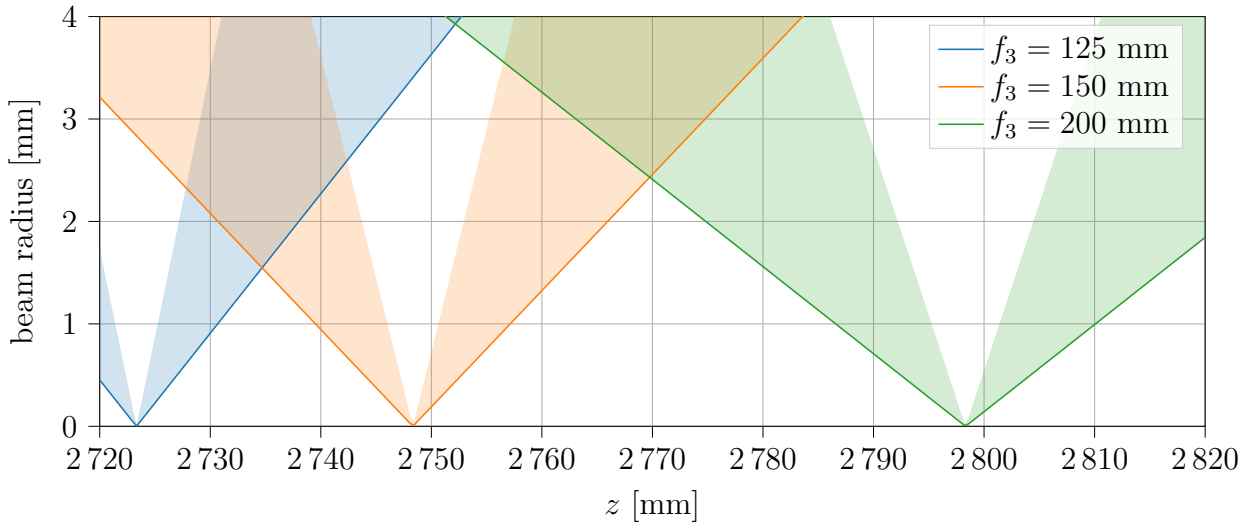

Figure 9: *detail of designed focusing heads in different configurations for laser 2*

```
# %% design head for second laser
# second laser
w0  = 100/2/10**3                       # approximate the waist with the core radius [mm]
BPP = 5/10**6                           # beam parameter product [m*rad]
MS  = BPP/lam0*np.pi                    # quality factor

corr=100/14*(MS**0.5/1.2**0.5)          # correction factor for both the wors quality factor and
↪  the bigger waist
(d0,d1,d2)  = (100*corr,0,100)              # spacing configuration

(f1,f2,f3)  = (100*corr,10**50,125)         # lenses configuration 1
fig, axs, Mag, d3, div, w_out, w_end =
↪  BeamExpander(lam0*10**3,w0,d0,d1,d2,f1,f2,f3,npoint=1001,MS=MS)
print(w_out*10**3)
(f1,f2,f3)  = (100*corr,10**50,150)         # lenses configuration 2
fig, axs, Mag, d3, div, w_out, w_end =
↪  BeamExpander(lam0*10**3,w0,d0,d1,d2,f1,f2,f3,npoint=1001,MS=MS,fig=fig,axs=axs)
print(w_out*10**3)
(f1,f2,f3)  = (100*corr,10**50,200)         # lenses configuration 3
fig, axs, Mag, d3, div, w_out, w_end =
↪  BeamExpander(lam0*10**3,w0,d0,d1,d2,f1,f2,f3,npoint=1001,MS=MS,fig=fig,axs=axs)
print(w_out*10**3)
axs.set_ylim(0,70)
axs.legend(loc='upper left')
tikzplotlib_fix_ncols(fig)
tikzplotlib.save('Assignment3/fig7.tex',axis_width='0.9\\textwidth',axis_height ='9cm')
```