Industrial Photonics

Design of Beam Expanders

Ariel Priarone

s274149

(a.y. 2022/2023)

Contents

List of Figures

List of Tables

1 Introduction

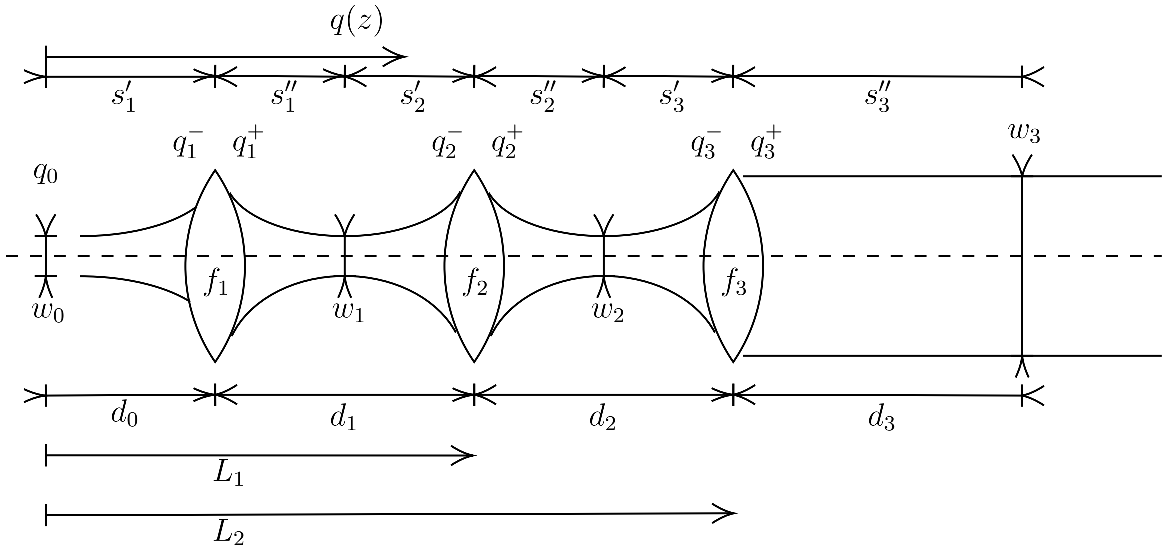

Figure 1: *Schematic of the considered beam expander*

This assignment asks to study the design of a beam expander provided in [1], in particular the behaviour of the magnification w.r.t. the position of the lenses. Then it's asked to study a practical implementation of the design given in the paper, and try to re-design another beam expander based on a given arrangement. To do that it's convenient to write a function that handle a general arrangement of three thin lenses, as depicted in **Figure 1**.

2 Write the function

The function is based on the propagation of the complex parameter $q(z) = z + j \cdot z_r$ on the z axis, thru air and lenses interfaces. The propagation thru the lenses is studied applying the matrix transfer function. As suggested in the paper the design should be optimized for a wavelength, in this case $\lambda_0 = 632.8$nm.

From the theory we can write the Reileigh range of the starting beam:

$$z_r = \frac{\pi w_0^2}{\lambda_0} \tag{1}$$

also the magnification and waist position formulas are used:

$$M_i = \frac{w_{i+1}}{w_i} = \frac{\theta_i}{\theta_{i+1}} = \frac{f_i}{\sqrt{(d_i - f_i)^2 + z_{r,i}^2}} \tag{2}$$

$$s_i' = f_i + M_i^2(s_i'' - f_i) \tag{3}$$

as well as the propagation of the Reileigh range thru lenses:

$$z_r^{i+1} = M_i^2 \cdot z_r^i \tag{4}$$

the initial condition of the beam parameter (at origin, $z = 0$):

$$q_0 = j \cdot z_{r,0} \tag{5}$$

and the propagation of q thru air and lenses:

$$L = \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1/f & 1 \end{bmatrix} \qquad \text{Lens matrix} \tag{6}$$

$$q_i^+ = \frac{A \cdot q_i^- + B}{C \cdot q_i^- + D} \qquad \text{propagation thru lens} \tag{7}$$

$$q_i(z+k) = q_i(z) + K \qquad \text{propagation thru air} \tag{8}$$

$$q_{i+1}^- = q_i^+ + d_i \qquad \text{propagation thru air length} \tag{9}$$

Always referring to **Figure 1**, it is possible to compute the radius thru all the length of the system:

$$w(z) = \begin{cases} \sqrt{-\dfrac{\lambda_0}{\pi \cdot \Im(\frac{1}{q_0+z})}} & z \in [0, d_0) \\ \sqrt{-\dfrac{\lambda_0}{\pi \cdot \Im(\frac{1}{q_1^+ + z - d_0})}} & z \in [d_0, L_1) \\ \sqrt{-\dfrac{\lambda_0}{\pi \cdot \Im(\frac{1}{q_2^+ + z - L1})}} & z \in [L_1, L_2) \\ \sqrt{-\dfrac{\lambda_0}{\pi \cdot \Im(\frac{1}{q_3^+ + z - L2})}} & z \in [L_2, \infty) \end{cases} \tag{10}$$

and for what concern the radius of a real beam:

$$w_r(z) = \begin{cases} \sqrt{M^2} \cdot w_{0,gauss} \sqrt{1 + \left(\dfrac{\lambda \cdot z}{\pi w_{0,gauss}^2}\right)^2} & z \in [0, d_0) \\ \sqrt{M^2} \cdot w_{0,gauss} \sqrt{1 + \left(\dfrac{\lambda \cdot (z - d_0 - s_1'')}{\pi w_{0,gauss}^2}\right)^2} & z \in [d_0, L_1) \\ \sqrt{M^2} \cdot w_{0,gauss} \sqrt{1 + \left(\dfrac{\lambda \cdot (z - L_1 - s_2'')}{\pi w_{0,gauss}^2}\right)^2} & z \in [L_1, L_2) \\ \sqrt{M^2} \cdot w_{0,gauss} \sqrt{1 + \left(\dfrac{\lambda \cdot (z - L_2 - s_3'')}{\pi w_{0,gauss}^2}\right)^2} & z \in [L_2, \infty) \end{cases} \tag{11}$$

The function to plot the beam radius thru the beam expander is implemented in python using the following code:

```python
def BeamExpander(lam0,w0,d0,d1,d2,f1,f2,f3,npoint=1000,fig=None,axs=None,plot=True,MS=1,
    zmin=None,zmax=None):
    # this function aim to produce a plot of a gausiann beam that passes thru three thin lenses,
    the approach used is tho compute the complex beam parameter q and propagate that thru
    air and lenses, then compute the radius and show a plot
    # parameters:
    #   lam0        wavelength considered                            [mm]
    #   w0          initial beam waist                               [mm]
    #   d0          from initial waist to first thin lens            [mm]
    #   d1          between first and second thin lenses             [mm]
    #   d2          between second and third thin lenses             [mm]
```

```
    #   f1          focal length first lens                          [mm]
    #   f2          focal length first lens                          [mm]
    #   f3          focal length first lens                          [mm]
    #   npoint      number of points of the plot (resolution)        [--]
    #   fig=None    figure handle
    #   axs=None    axis handle
    #   plot=True   T=generate plot; F=generate only the data
    #   Ms          quality factor of the beam (ref slide 05/177)    [--]
    #   zmin        z axis limit to consider                         [mm]
    #   zmax        z axis limit to consider                         [mm]
    # returns:
    #   fig         figure handle of the plot
    #   axs         axis handle of the plot
    #   M           overall magnification of the system              [--]
    #   d3 (s3II)   location of the output waist w.r.t. last lens    [mm]
    #   th3*10**5   angle of output beam *10^5                       [mrad*100]
    #   w3          output waist (real beam)                         [mm]
    #   w_end       beam radius at the end of the system (real beam)[mm]

    L1      =   d0+d1                           # second length position
    L2      =   d0+d1+d2                        # third length position
    zr0     =   np.pi*w0**2/lam0                # Rayleigh range
    q0      =   1j*zr0                          # complex beam parameter
    th0     =   lam0/np.pi/w0                   # divergence at the left of the first lens
    M       =   MS**0.5                         # sqrt of quality factor

    M1      =   f1/((d0-f1)**2+zr0**2)**0.5     # magnification first lens
    M1      =   abs(M1)
    w1      =   M1*w0                           # weist of second beam
    zr1     =   zr0*M1**2                       # Rayleigh range right first lens
    th1     =   th0/M1                          # Divergence right first lens
    q1minus =   q0+d0                           # propagate left side first lens
    A,B,C,D =   (1,0,-1/f1,1)                   # matrix entries of first lens
    q1plus  =   (A*q1minus+B)/(C*q1minus+D)     # propagate right side first lens

    s1I     =   d0                              # distance from first lens and waist (on the
    ↪  left)
    s1II    =   f1+M1**2*(s1I-f1)               # distance from first lens and waist (on the
    ↪  right)
    S2I     =   (d1-s1II)                       # distance from second lens and waist (on the
    ↪  left)
    M2      =   f2/((S2I-f2)**2+zr1**2)**0.5    # magnification second lens
    M2      =   abs(M2)
    w2      =   M2*w1                           # weist of third beam
    zr2     =   zr1*M2**2                       # Rayleigh range right second lens
    th2     =   th1/M2                          # Divergence right second lens
    q2minus =   q1plus+d1                       # propagate left side second lens
    A,B,C,D =   (1,0,-1/f2,1)                   # matrix entries of second lens
    q2plus  =   (A*q2minus+B)/(C*q2minus+D)     # propagate right side second lens
```

```python

    s2II    =   f2+M2**2*(S2I-f2)              # distance from second lens and waist (on the
    ↪  right)
    S3I     =   (d2-s2II)                      # distance from third lens and waist (on the
    ↪  left)
    M3      =   f3/((S3I-f3)**2+zr2**2)**0.5   # magnification third lens
    M3      =   abs(M3)
    w3      =   M3*w2                          # weist of third beam
    zr3     =   zr2*M3**2                      # Rayleigh range right second lens
    th3     =   th2/M3                         # Divergence right second lens
    q3minus =   q2plus+d2                      # propagate left side third lens
    A,B,C,D =   (1,0,-1/f3,1)                  # matrix entries of third lens
    q3plus  =   (A*q3minus+B)/(C*q3minus+D)    # propagate right side third lens
    if zmin is None:
        zmin=   0                              # min of z axis
    if zmax is None:
        zmax=   L2+2*f3                        # max of z axis
    z_vect  =   np.linspace(zmin,zmax,npoint)  # points of z axis
    w       =   []                             # initialize beam radius along z
    w_r     =   []                             # this will be the real beam (not gaussian)

    S3II    =   f3+M3**2*(S3I-f3)              # location of output waist w.r.t. last lens (if
    ↪  negative, the beam is already diverging)
    for z in z_vect:
        if      0<=z<d0:
            q   = q0+(z-0)                     # propagate q to z position
            aux = 1/q                          # auxilliary for radius calculation
            w.append((-lam0/(np.pi*aux.imag))**0.5)  # beam radius along z axis
            w_r.append(M*w0*(1+((lam0*z)/(np.pi*w0**2))**2)**0.5)          # real beam radius
            ↪  along z axis
        elif    d0<=z<L1:
            q   = q1plus+(z-d0)                # propagate q to z position
            aux = 1/q                          # auxilliary for radius calculation
            w.append((-lam0/(np.pi*aux.imag))**0.5)  # beam radius along z axis
            w_r.append(M*w1*(1+((lam0*(z-d0-s1II))/(np.pi*w1**2))**2)**0.5)  # real beam radius
            ↪  along z axis
        elif    L1<=z<L2:
            q   = q2plus+(z-L1)                # propagate q to z position
            aux = 1/q                          # auxilliary for radius calculation
            w.append((-lam0/(np.pi*aux.imag))**0.5)  # beam radius along z axis
            w_r.append(M*w2*(1+((lam0*(z-L1-s2II))/(np.pi*w2**2))**2)**0.5)  # real beam radius
            ↪  along z axis
        elif    L2<=z:
            q   = q3plus+(z-L2)                # propagate q to z position
            aux = 1/q                          # auxilliary for radius calculation
            w.append((-lam0/(np.pi*aux.imag))**0.5)  # beam radius along z axis
            w_r.append(M*w3*(1+((lam0*(z-L2-S3II))/(np.pi*w3**2))**2)**0.5)  # real beam radius
            ↪  along z axis
        ymax=max(w)*1.1;    ymin=-0
```

```
    xmin=0; xmax=L2+2*f3
    if plot:                                    # plot if needed, skip if not
        if fig == None or axs == None:
            fig, axs=plt.subplots()
        fig.tight_layout()
        axs.plot(z_vect,w,label=f'$d_1={d1}$; $d_2={d2}$')
        if (MS > 1):
            axs.fill_between(z_vect, w_r, w, alpha=0.2) # if it's a gaussian beam, no need to
             ↪ plot the shade
        axs.set_xlabel('$z$ [mm]')
        axs.set_ylabel('beam radius [mm]')
        axs.set_ylim([ymin,ymax]); axs.set_xlim([xmin,xmax])
        axs.vlines([d0, L1, L2],ymin,ymax,linestyles="dashdot",color="magenta")
        axs.grid(True, 'major')
        axs.legend()
    return fig, axs, M1*M2*M3, S3II, th3*10**5, w3*M, w_r[-1]
```

3 Reproduce the paper result

The second point of the assignment asks to reproduce the results of the paper [1]. To do that i used the already developed function to plot the beam radius along the z axis, with the following code, that produce the plot in **Figure 2**, that is also magnified, for a single configuration, in **Figure 3** to show the hyperbole shape of the beam radius $w(z)$. The code spans the d_1 and d_2 used in the paper.

The results comparison are summarized in **Table 1**.

```
# %% check result for all the row of the table
table=[(10,120.006),
       (20,115.002),
       (30,113.334),
       (40,112.500),
       (50,112.000)]
fig, ax = plt.subplots()
for (d1,d2) in table:
    fig, ax, Mg, dout, thout, wout, w_end =
     ↪ BeamExpander(lam0=0.0006328,w0=0.5,d0=100,d1=d1,d2=d2,f1=-10,f2=10,f3=100
     ↪ ,npoint=1000,fig=fig,axs=ax)
    print(f'Mg={Mg}; dout={dout}; thout={thout}; wout={wout}')

tikzplotlib_fix_ncols(fig)
tikzplotlib.save('Assignment2/PLOT.tex',axis_width='0.9\\textwidth',axis_height ='7cm')
```

3.1 Claim about the magnification

In the paper, the authors claim that "the expected magnification ratio always occurs at a distance equal to the focal length of the rightmost lens". Following the theory, the beam waist distances from the lenses, at which the expected magnification ratio happens, propagate thru the lenses as in

d_1	d_2	M		s''_3		$\theta_3 \cdot 10^5$		w_3	
		paper	my script	paper	my script	paper	my script	paper	my script
10	120.006	10.032	10.0322	100.000	-601.821	4.017	4.015	5.016	5.016
20	115.002	20.071	20.069	100.000	7800.014	2.008	2.007	10.036	10.035
30	113.334	30.111	30.108	100.000	-12190.104	1.338	1.338	15.055	15.054
40	112.500	40.150	40.000	100.000	-171900.000	1.004	1.007	20.075	20.000
50	112.000	50.189	50.000	100.000	-269899.99	0.803	0.805	25.095	25.000
$f_1 = -10$; $f_2 = 10$; $f_3 = 100$; $w_0 = 0.5$; $d_0 = 100$; $\lambda = 0.0006328$									

Table 1: *Comparison between my results and the paper [1] results (all distances in mm)*

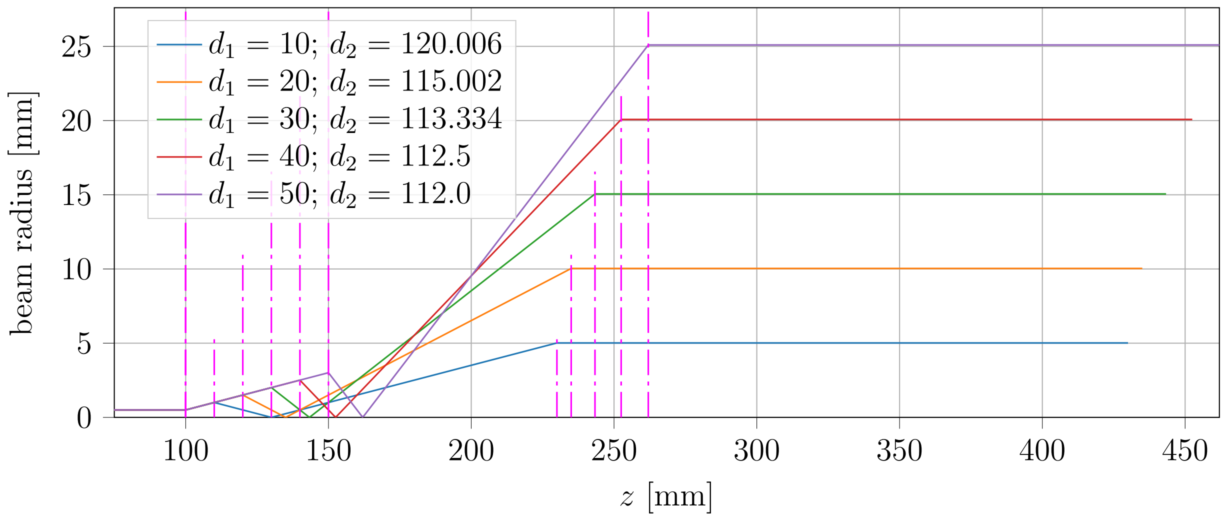

Figure 2: *Plot of the arrangements proposed in [1]*

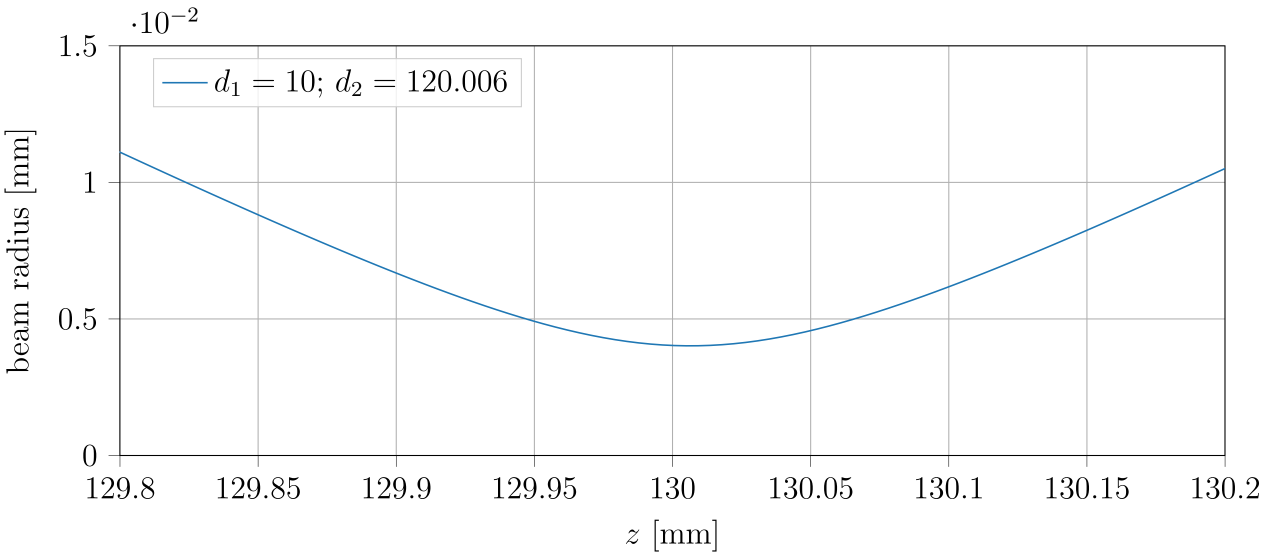

Figure 3: *Plot of the arrangements proposed in [1], detail of the hyperbole shape*

Equation 3. So the distance s''_3 is not always 100 mm, as claimed in the paper, but vary with the configuration of the system. The results are reported in **Table 1**, in the column s''_3. my results are sometimes negative, that means that the beam is already diverging as soon as it exit the rightmost lens, and the distance indicate where the waist would have been (on the left on the lens) if such lens didn't exist in the path of the beam.

The fact that my results, in this case, differ from the one in the paper may be caused by the fact that i did not consider the Gaussian focal shift, that cause the position of the focal point to shift from the theoretical location. This is due to the fact that the Gaussian beam is not a plane wave, but the lens transformation equation that i used are derived using geometrical optics.

3.2 Linear dependency on d_1

Another request of the assignment is to verify that "the magnification of such a system is approximately a linear function of the mutual distance d_1 between the first and the second element of the optical system". To do that I will first consider the theoretical definition of the magnification $M = w_3/w_0$, and then the "practical" magnification at double the focal length of the rightmost lens $M = w(z = L_2 + 2 \cdot f3)/w_0$.

3.2.1 Using theoretical magnification

Using the theory definition, it is easy to show that the behaviour is not linear at all, as shown in **Figure 4**. Note that only half of this plot is referring to an actual beam waist at the right of the rightmost lens, because after the magnification reaches a maximum, the beam becomes diverging and the 'virtual" waist would be on the left of the last lens, where there is another beam.

I performed the calculation with the following script:

```
# %% check linearity
fig, axs=plt.subplots()
fig.tight_layout()
Mg_vect=[]
d2      =   112.5                   # note that this is optimized for 40x !!!
d1_vect =   np.linspace(38,42,500)      # try some d1
for d1 in d1_vect:
    Mg =
    ↪ BeamExpander(lam0=0.0006328,w0=0.5,d0=100,d1=d1,d2=d2,f1=-10,f2=10,f3=100,plot=False)[2]
    Mg_vect.append(Mg)
axs.plot(d1_vect,Mg_vect,label=f'$d_2={round(d2,3)}$')
axs.set_xlabel('$d_1$ [mm]')
axs.set_ylabel('Magnification [-]')
axs.grid(True, 'Both')
axs.legend()

tikzplotlib_fix_ncols(fig)
tikzplotlib.save('Assignment2/dvsm.tex',axis_width='0.9\\textwidth',axis_height ='7cm')
```

The question at this point could be: is it possible that the following statement is true?

$$\forall d_1 \exists d_2 | \max(M) = M(d_1, d_2)$$

i.e. for every d_1, there is a d_2 that places the maximum in the position d_1? and then is it possible that the value of this maximum is linearly dependent on d_1?

In order to have an empirical proof of this statement i performed a gridding of both the parameters (d_1, d_2), obtaining the **Figure 5**, where the linear dependency is quite evident.

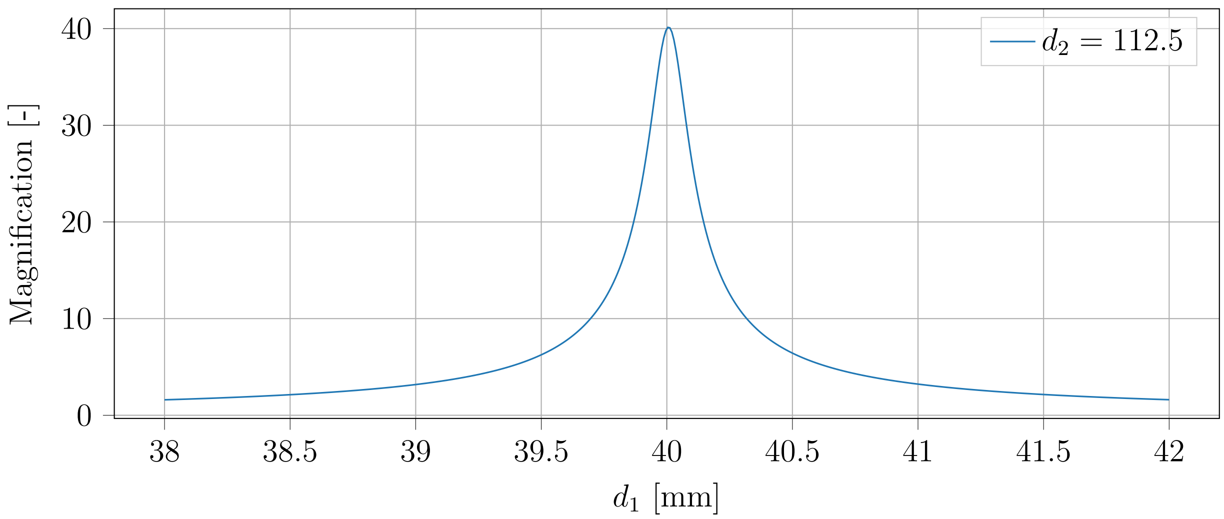

Figure 4: *Plot of the magnification dependency on d_1*

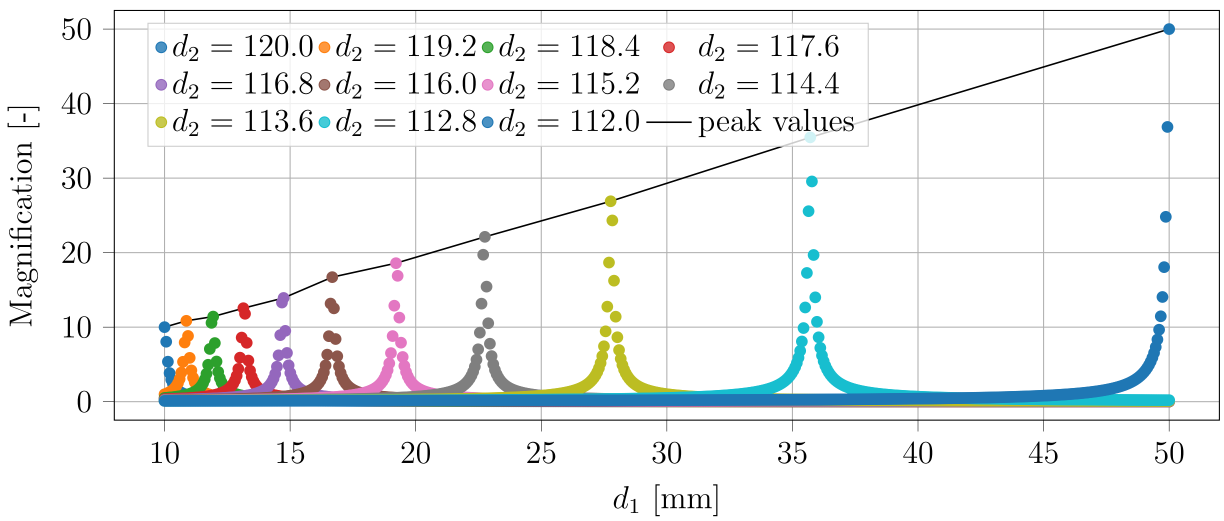

Figure 5: *Plot of the linear approximation*

The following script was used to produce the results:

```
# %% check linearity - envelope
fig, axs=plt.subplots()
fig.tight_layout()
maximum=[]
d_max=[]
for d2 in np.linspace(120,112,11):
    LinRel = [[],[]]
    for d1 in np.linspace(10,50,600):
        Mg = BeamExpander(lam0=0.0006328,w0=0.5,d0=100,d1=d1,d2=d2,
        ↪  f1=-10,f2=10,f3=100,plot=False)[2]
        LinRel[0].append(d1)
```

```
        LinRel[1].append(Mg)
    maximum.append(max(LinRel[1]))
    maxindex=LinRel[1].index(max(LinRel[1]))
    d_max.append(LinRel[0][maxindex])
    axs.scatter(LinRel[0],LinRel[1],marker='.',label=f'$d_2={round(d2,3)}$')
axs.plot(d_max,maximum,'k',label=f'peak values')
axs.set_xlabel('$d_1$ [mm]')
axs.set_ylabel('Magnification [-]')
axs.grid(True, 'Both')
axs.legend(ncol=4)

tikzplotlib_fix_ncols(fig)
tikzplotlib.save('Assignment2/LinApprox.tex',axis_width='0.9\\textwidth',axis_height ='7cm')
```

3.2.2 Using the output beam radius

The previous **subsubsection 3.2.1** applied the definition using the waist radius, but a more practical approach would be to check the magnification in a predetermined point, independently from where the beam is focused. In this case it is the last point of the plot $(L_2 + 2 \cdot f_3)$. Again i used the script below to check the linearity for some configurations, and applying the new definition

$$M = \frac{w(z = L_2 + 2 \cdot f_3)}{w_0}$$

the linear behaviour is immediately evident (**Figure 6**), also because in all the configurations the divergence experienced in the near range of the device is really small.

```
# %% check linearity - definition with useful beam radius
fig, axs=plt.subplots()
fig.tight_layout()
d1_vect =   np.linspace(10,50,10)       # try some d1
d2_vect =   np.linspace(120,112,5)      # try some d2

for d2 in d2_vect:
    Mg_vect =   []
    for d1 in d1_vect:
        W_out = BeamExpander(lam0=0.0006328,w0=0.5,d0=100,d1=d1,d2=d2,f1=-10,f2=10,f3=100, plot=False)[6]
        Mg_vect.append(W_out/0.5)
    axs.plot(d1_vect,Mg_vect,label=f'$d_2={round(d2,3)}$')
axs.set_xlabel('$d_1$ [mm]')
axs.set_ylabel('$\\frac{w(z=L_2+2\\cdot f_3)}{w_0}$ [-]')
axs.grid(True, 'Both')
axs.legend()
tikzplotlib_fix_ncols(fig)
tikzplotlib.save('Assignment2/Woutvsm.tex',axis_width='0.9\\textwidth',axis_height ='7cm')
```

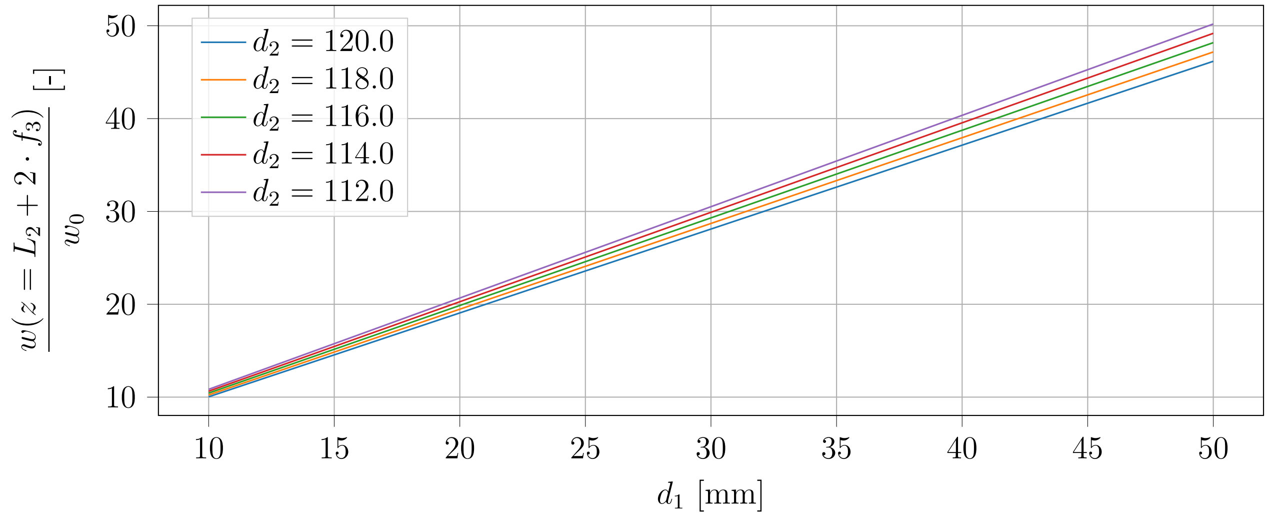

Figure 6: *Plot of the linear behaviour*

4 Practical implementation

The next step of the assignment is to find some commercially available devices that can be used to build the system. the requisites and products that i found are resumed in **Table 2**. These items are just lenses, without supports, so in a real application, also the structure for mounting and tune the positions of the lenses would have to be designed.

Lens	Design			Commercial			
	f [mm]	type	diameter [mm]	manufacturer	f [mm]	type	diameter [mm]
1	-10	negative	>1	Techspec 62-437	-10	negative	6.25
2	10	positive	>6	Techspec stock 63-535	10	positive	10.00
3	100	positive	>50	Thorlabs LB1630-A	100	positive	50.8

Table 2: *Commercial lenses that meet the design requirements*

5 Study of the assignment arrangement

The next request of the assignment is to study a different arrangement of three lens beam expander, in which the middle lens is the negative one. To do that I considered the distances of the arrangement of the paper, and changed the focal length of the lenses to simulate this different configuration. The script is the following and the results are shown in **Figure 7**.

As is, the device, produce a zoom effect, but the collimation of the output beam worsen with the increasing of the magnification.

```python
# %% check result for all the row of the table
table=[(10,120.006),
       (20,115.002),
       (30,113.334),
       (40,112.500),
       (50,112.000)]
fig, ax = plt.subplots()
for (d1,d2) in table:
```

```
    fig, ax, Mg, dout, thout, wout, w_end =
    ↪ BeamExpander(lam0=0.0006328,w0=0.5,d0=100,d1=d1,d2=d2,f1=10,f2=-10,f3=100,
    ↪ npoint=1000,fig=fig,axs=ax)
    print(f'Mg={Mg}; dout={dout}; thout={thout}; wout={wout}')

tikzplotlib_fix_ncols(fig)
tikzplotlib.save('Assignment2/AssArrangment.tex',axis_width='0.9\\textwidth',axis_height ='7cm')
```

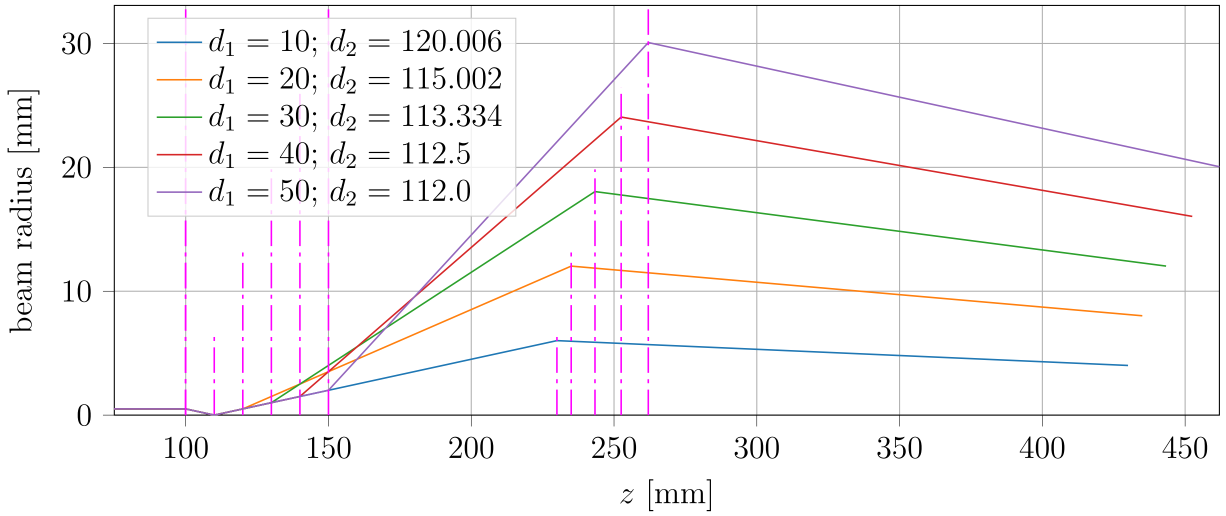

Figure 7: *Plot of the behaviour of the arrangement with negative second lens*

6 Design the beam expander

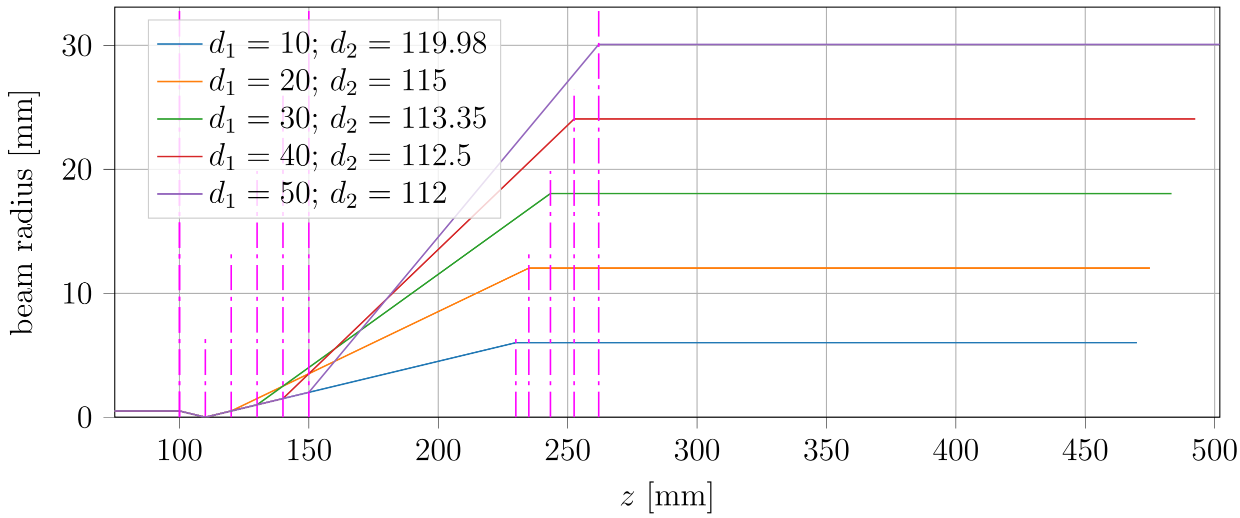

Figure 8: *Plot of the behaviour of the designed system*

At this point the request is to find a strategy to improve the previous configuration to obtain a better beam expander. To optimize the device i used a manual bisection method on the parameter d_2, trying to minimize the output divergence, for all the d_1 values.

d_1 [mm]	d_2 [mm]	M	θ_3 [mrad]	w_3 [mm]
10	119.98	11.47	3.51	5.7
20	115.00	24.00	1.67	12.0
30	113.35	17.51	2.29	8.7
40	112.50	48.00	0.83	24.0
50	112.00	60	0.67	30.0
$f_1 = 10$; $f_2 = -10$; $f_3 = 120$;				
$w_0 = 0.5$; $d_0 = 100$; $\lambda = 0.0006328$ all [mm]				

Table 3: *Performance at some configurations of my design*

With the following values i managed to obtain the results shown in **Figure 8** and resumed in **Table 3**.

```
# %% try to optimize oyher expander
table=[(10,119.98),
       (20,115),
       (30,113.35),
       (40,112.5),
       (50,112)]
fig, ax = plt.subplots()
for (d1,d2) in table:
    fig, ax, Mg, dout, thout, wout, w_end =
    ↪ BeamExpander(lam0=0.0006328,w0=0.5,d0=100,d1=d1,d2=d2,f1=10,f2=-10,f3=120,
    ↪ npoint=1000,fig=fig,axs=ax)
    print(f'Mg={Mg}; dout={dout}; thout={thout}; wout={wout}')

tikzplotlib_fix_ncols(fig)
tikzplotlib.save('Assignment2/Mydesign.tex',axis_width='0.9\\textwidth',axis_height ='7cm')
plt.show()
```

References

[1] Antonin Miks and Pavel Novak. Paraxial properties of three-element zoom systems for laser beam expanders. *Optics Express*, 22, 09 2014.