

DUPIED-VIEIRA4

October 9, 2026

0.1 Partie1

```
[1]: import numpy as np
import matplotlib.pyplot as plt

def lentille_convergente(d,r):
    e = r-r*np.sqrt(1-(d**2)/(4*r**2))
    y = np.linspace(-d/2, d/2, 100)
    yP=np.linspace(-1.5*d,1.5*d, 100)
    x_courbe = np.sqrt(r**2 - y**2) - r + e

    plt.figure(1)
    plt.plot(x_courbe, y, "b")
    plt.plot (np.zeros(100), y, 'b')
    plt.plot(yP, np.zeros(100), '--k')
```

0.2 Partie2

```
[2]: def tracer_rayon(z,d,r,n):

    # Epaisseur de la lentille
    e = r-r*np.sqrt(1-d**2/(4*r**2))

    # Angle du rayon incident
    i1 = np.arcsin(z/r)

    # On vérifie qu'il existe un rayon émergent
    if n*np.sin(i1) >= 1:
        print("Pas de rayon émergent pour z =", z)
        return

    # Angle du rayon émergent
    i2 = np.arcsin(n*np.sin(i1))

    # Coordonnée du point M
    xM = e-r+r*np.cos(i1)
```

```

# Coordonnée de F avec  $x_F = 3r$ 
yF = z-(3*r+xM)*np.tan(i2-i1)

# Rayon incident
x_incident = np.linspace(-1.5*r,xM,100)
y_incident = np.ones(100)*z

# Rayon émergent
x_emergent = np.linspace(xM,3*r,100)
y_emergent = z-(x_emergent-xM)*np.tan(i2-i1)

# Tracé
plt.plot(x_incident,y_incident,'r')
plt.plot(x_emergent,y_emergent,'r')

# Point M
plt.plot(xM,z,'ko')

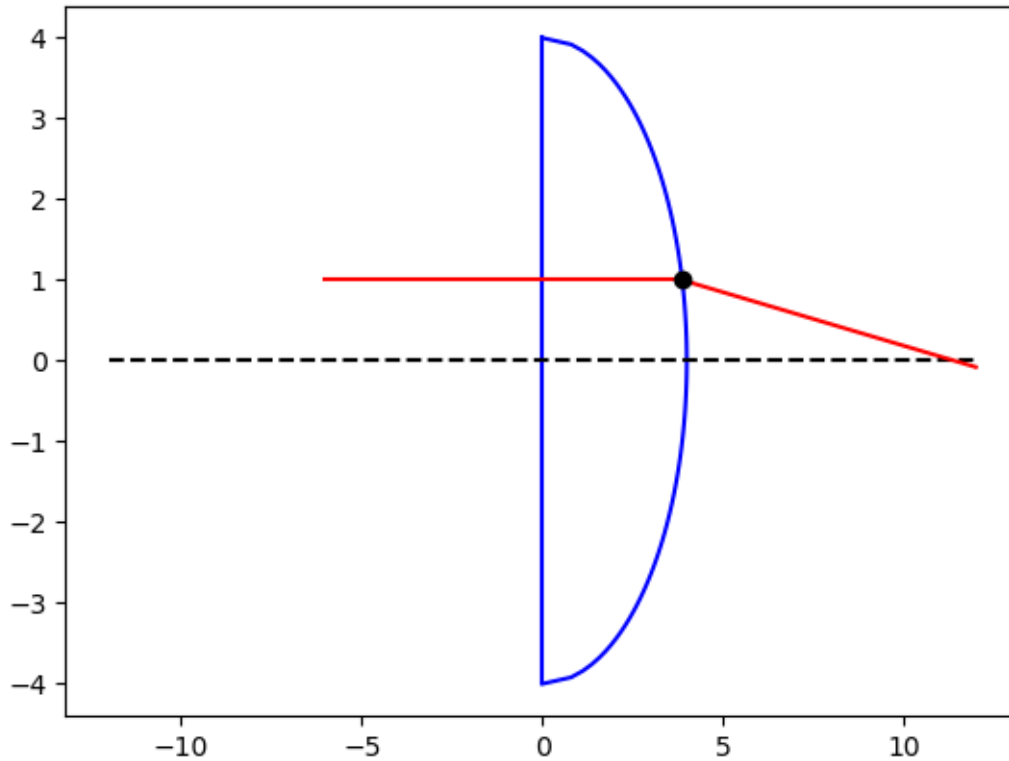
return xM,yF

d = 8
r = 4
z = 1
n = 1.5

lentille_convergente(d, r)
tracer_rayon(z, d, r, n)

plt.show()

```



0.3 Partie3

```
[3]: plt.figure()

# lentille
e = r-r*np.sqrt(1-d**2/(4*r**2))
y = np.linspace(-d/2,d/2,100)
x = np.sqrt(r**2-y**2)-r+e

plt.plot(x,y,'b')
plt.plot(np.zeros(100),y,'b')

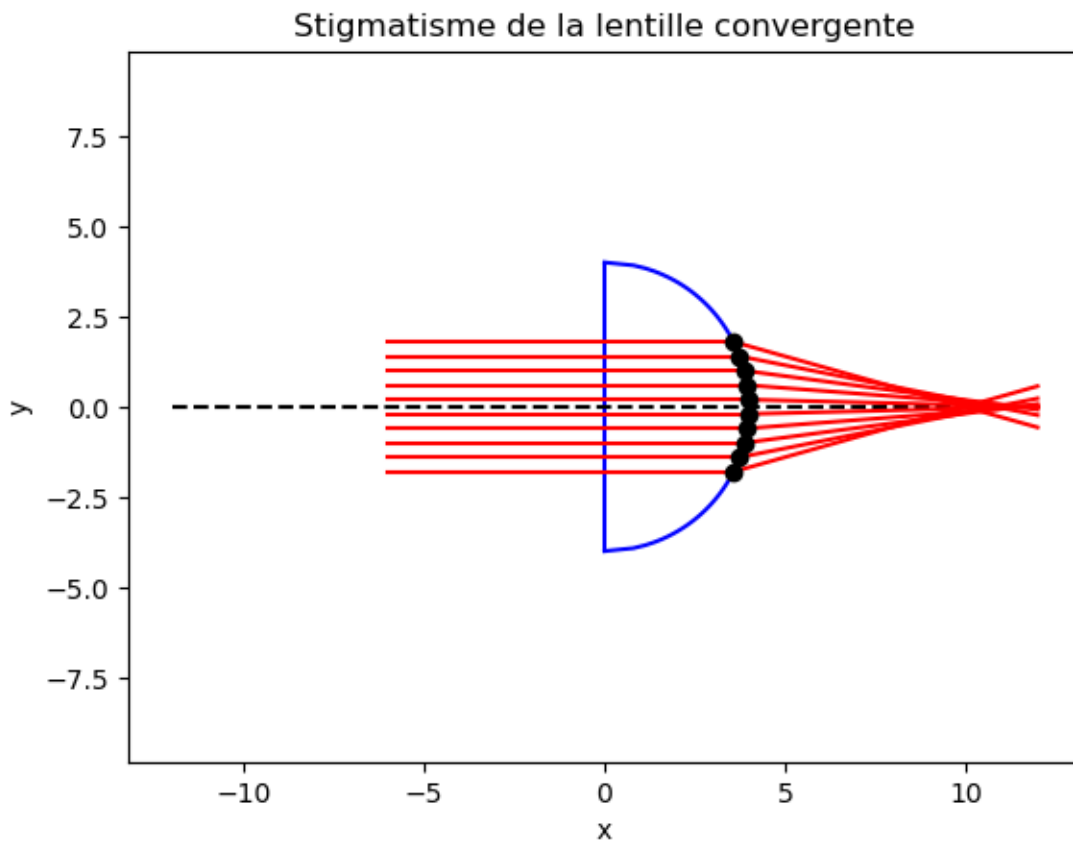
# axe optique
yp = np.linspace(-1.5*d,1.5*d,100)
plt.plot(yp,np.zeros(100),'--k')

# 10 rayons à différentes hauteurs
hauteurs = np.linspace(-1.8,1.8,10)

for z in hauteurs:
    tracer_rayon(z,d,r,n)
    print("Altitude du rayon :", z)
```

```
plt.axis('equal')
plt.xlabel("x")
plt.ylabel("y")
plt.title("Stigmatisme de la lentille convergente")
plt.show()
```

Altitude du rayon : -1.8
 Altitude du rayon : -1.4
 Altitude du rayon : -1.0
 Altitude du rayon : -0.5999999999999999
 Altitude du rayon : -0.19999999999999996
 Altitude du rayon : 0.19999999999999996
 Altitude du rayon : 0.60000000000000003
 Altitude du rayon : 1.0000000000000002
 Altitude du rayon : 1.4000000000000001
 Altitude du rayon : 1.8



0.3.1 Q6: Pour cette lentille, le stigmatisme rigoureux n'existe pas : on n'a qu'un stigmatisme approché valable uniquement si l'on se place dans les conditions de Gauss (rayons proches de l'axe optique).

0.4 Partie 4

```
[4]: def lentille_divergente(d,r):

    y = np.linspace(-d/2,d/2,100)

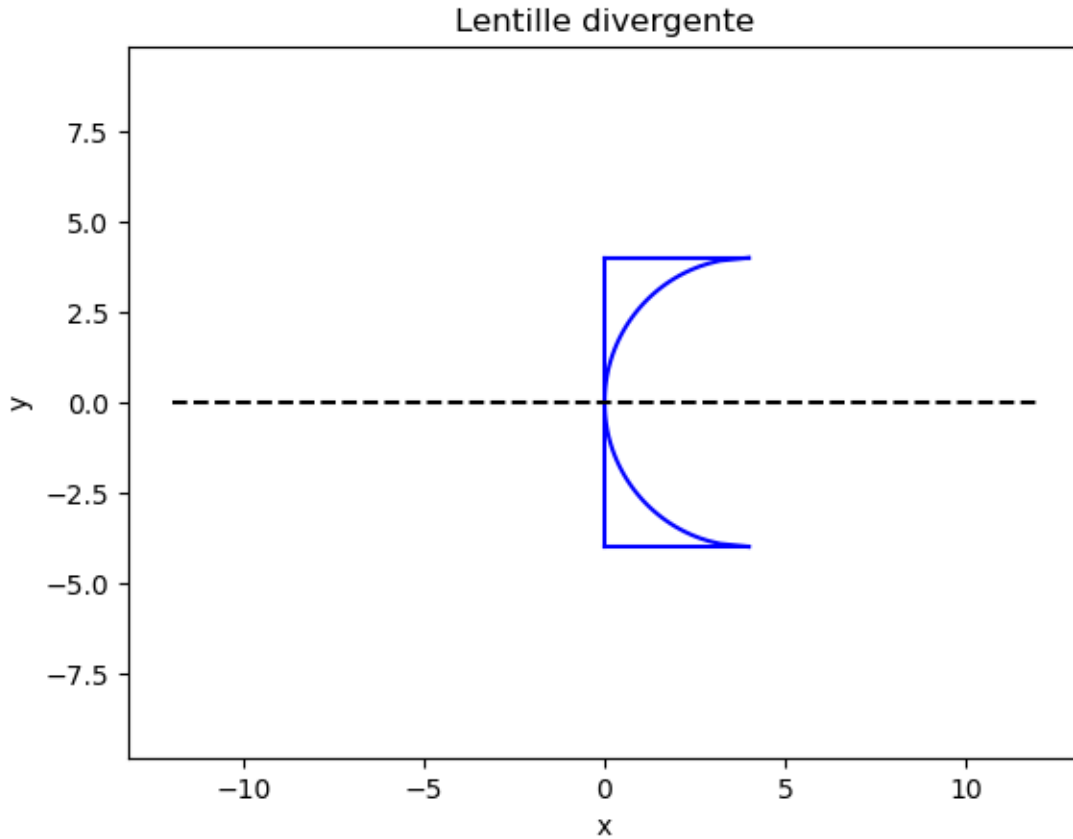
    # Portion de cercle
    x = r-np.sqrt(r**2-y**2)

    plt.plot(x,y,'b')
    plt.plot([0, 0], [-d/2, d/2], 'b')
    plt.plot([x[0], 0], [y[0], y[0]], 'b')
    plt.plot([x[-1], 0], [y[-1], y[-1]], 'b')

    # axe optique
    yp = np.linspace(-1.5*d,1.5*d,100)
    plt.plot(yp,np.zeros(100),'--k')

    plt.axis('equal')
    plt.xlabel("x")
    plt.ylabel("y")
    plt.title("Lentille divergente")

lentille_divergente(8,4)
```



```
[5]: def tracer_rayondivergent(z,d,r,n):

    # Angle incident
    i1 = np.arcsin(z/r)

    # Angle réfracté
    i2 = np.arcsin(n*np.sin(i1))

    # Coordonnée de M
    xM = r-r*np.cos(i1)

    # Coordonnée de F avec  $x^F = r$ 
    yF = (r-xM)*np.tan(i2-i1)+z

    # Rayon incident
    x_incident = np.linspace(-1.5*r,xM,100)
    y_incident = np.ones(100)*z

    # Rayon émergent
    x_emergent = np.linspace(xM,r,100)
```

```

y_emergent = z+(x_emergent-xM)*np.tan(i2-i1)

# Prolongement du rayon émergent
# jusqu'à l'axe optique
xG = xM-z*(r-xM)/(yF-z)

x_prolongement = np.linspace(xG,xM,100)
y_prolongement = z+(x_prolongement-xM)*np.tan(i2-i1)

# Tracé
plt.plot(x_incident,y_incident,'r')
plt.plot(x_emergent,y_emergent,'r')
plt.plot(x_prolongement,y_prolongement,'--r')

plt.plot(xM,z,'ko')

return xM,yF,xG

```

```

[6]: y = np.linspace(-d/2,d/2,100)
x = r-np.sqrt(r**2-y**2)

plt.plot(x,y,'b')
plt.plot(np.zeros(100),y,'b')

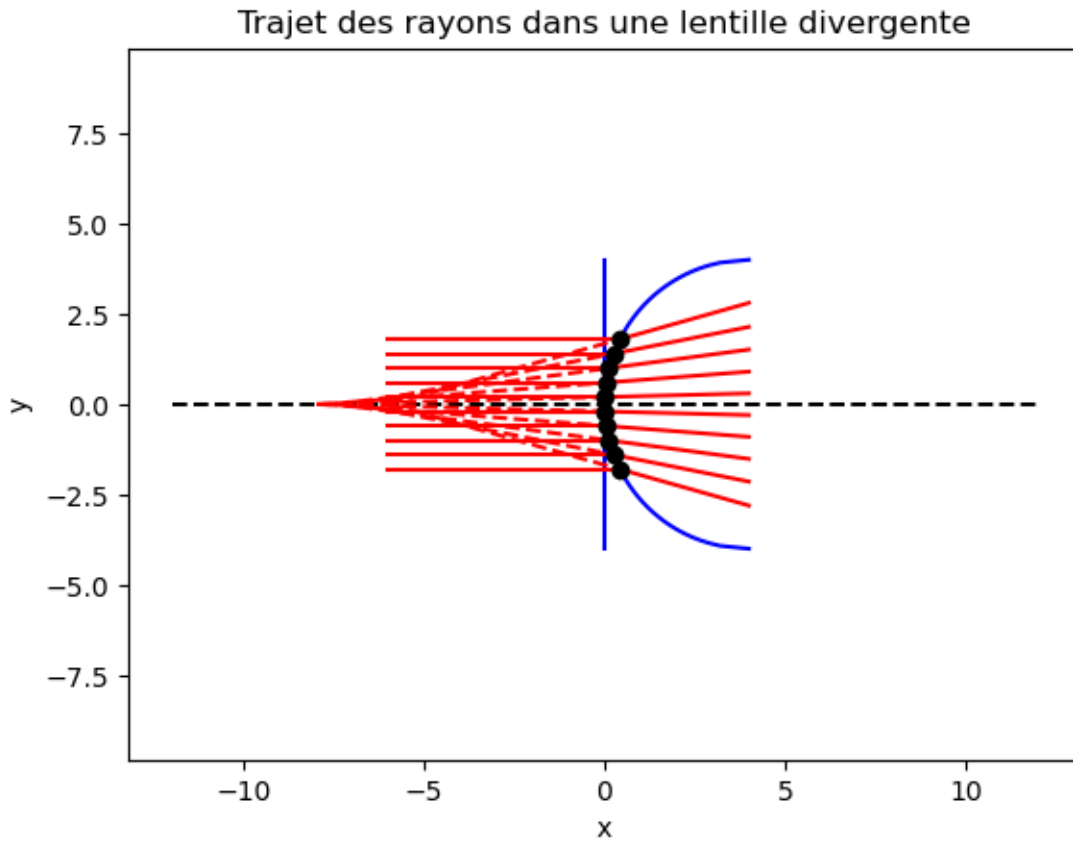
yp = np.linspace(-1.5*d,1.5*d,100)
plt.plot(yp,np.zeros(100),'--k')

hauteurs = np.linspace(-1.8,1.8,10)

for z in hauteurs:
    tracer_rayondivergent(z,d,r,n)

plt.axis('equal')
plt.xlabel("x")
plt.ylabel("y")
plt.title("Trajet des rayons dans une lentille divergente")
plt.show()

```



1 Partie 5

```
[7]: n_rouge = 1.511
     n_vert = 1.520
     n_bleu = 1.528

def rayon_chromatique(z,d,r,n):

    e = r-r*np.sqrt(1-d**2/(4*r**2))

    i1 = np.arcsin(z/r)

    # Vérification de l'existence du rayon émergent
    if n*np.sin(i1) >= 1:
        return

    i2 = np.arcsin(n*np.sin(i1))
```

```

xM = e-r+r*np.cos(i1)

yF = z-(3*r+xM)*np.tan(i2-i1)

# rayon incident
x1 = np.linspace(-1.5*r,xM,100)
y1 = np.ones(100)*z

# rayon émergent
x2 = np.linspace(xM,3*r,100)
y2 = z-(x2-xM)*np.tan(i2-i1)

plt.plot(x1,y1)
plt.plot(x2,y2)

return xM,yF

plt.figure()

# Lentille
e = r-r*np.sqrt(1-d**2/(4*r**2))
y = np.linspace(-d/2,d/2,100)
x = np.sqrt(r**2-y**2)-r+e

plt.plot(x,y,'b')
plt.plot(np.zeros(100),y,'b')

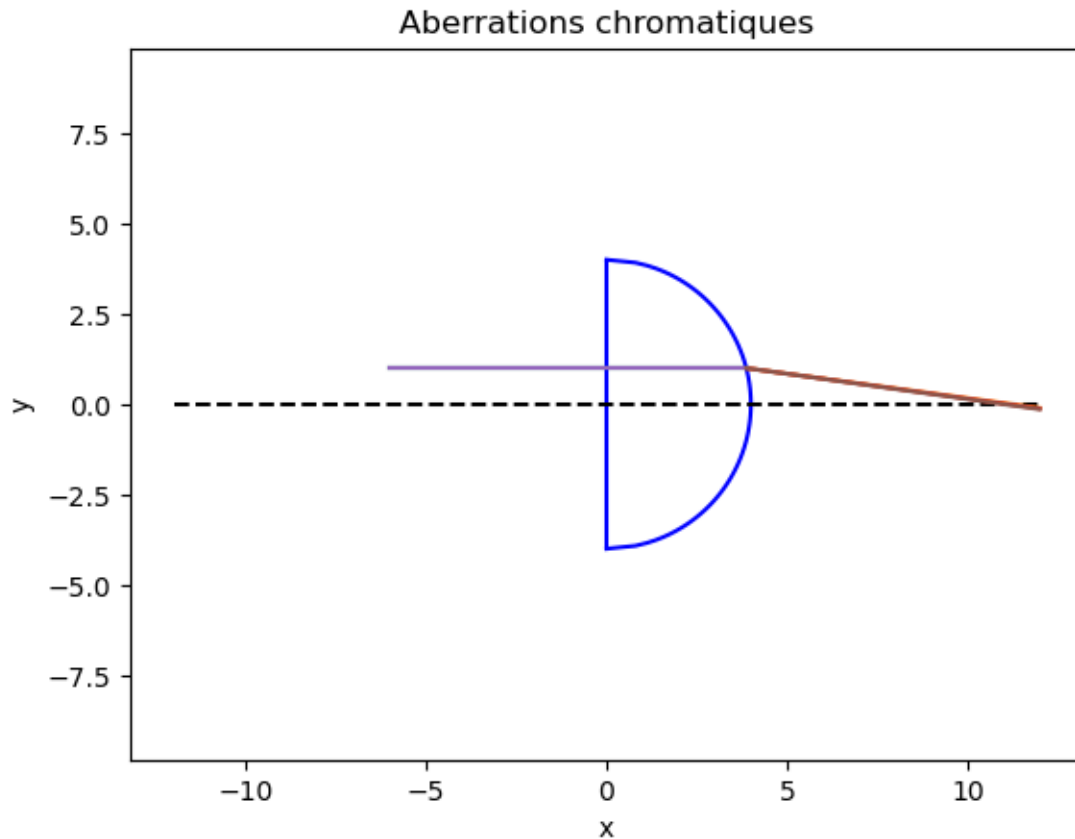
# axe optique
yp = np.linspace(-1.5*d,1.5*d,100)
plt.plot(yp,np.zeros(100),'--k')

# On prend un rayon proche de l'axe
z = 1

rayon_chromatique(z,d,r,n_rouge)
rayon_chromatique(z,d,r,n_vert)
rayon_chromatique(z,d,r,n_bleu)

plt.axis('equal')
plt.xlabel("x")
plt.ylabel("y")
plt.title("Aberrations chromatiques")
plt.show()

```



On voit qu'en sortie de lentille, les rayons bleu et rouges sont presque confondus.

Q9 Pour minimiser la dispersion des couleurs, il faut choisir le matériau dont la courbe $n(\lambda)$ est la plus horizontale (variation d'indice Δn la plus faible sur le spectre visible entre 0.4 μm et 0.8 μm). Sur le graphique, la courbe la plus basse et la plus plate est celle du verre Fluorite crown FK51A. C'est donc ce type de verre qu'il faut utiliser pour réduire au maximum les aberrations chromatiques.