

1. On crée la fonction `lentilleconvergente(d, r)`

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def lentilleconvergente(d, r):
5     e = r - r * np.sqrt(1 - (d**2) / (4 * r**2))
6     y = np.linspace(-d / 2, d / 2, 500)
7     x_convexe = np.sqrt(r**2 - y**2) - r + e
8
9     plt.plot(np.zeros_like(y), y, 'k-', lw=1.5)
10    plt.plot(x_convexe, y, 'k-', lw=1.5)
11    plt.plot([0, x_convexe[0]], [-d / 2, -d / 2], 'k-', lw=1.5)
12    plt.plot([0, x_convexe[-1]], [d / 2, d / 2], 'k-', lw=1.5)
13
14

```

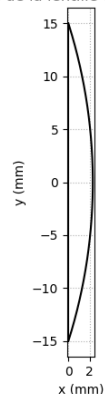
Test de la fonction :

```

17 plt.figure(figsize=(7, 5))
18 d_test, r_test = 30.0, 50.0
19
20 lentilleconvergente(d_test, r_test)
21
22 plt.gca().set_aspect('equal')
23 plt.xlabel("x (mm)")
24 plt.ylabel("y (mm)")
25 plt.title("Profil de la lentille convergente")
26 plt.grid(True, ls=':')
27 plt.show()

```

Profil de la lentille convergente



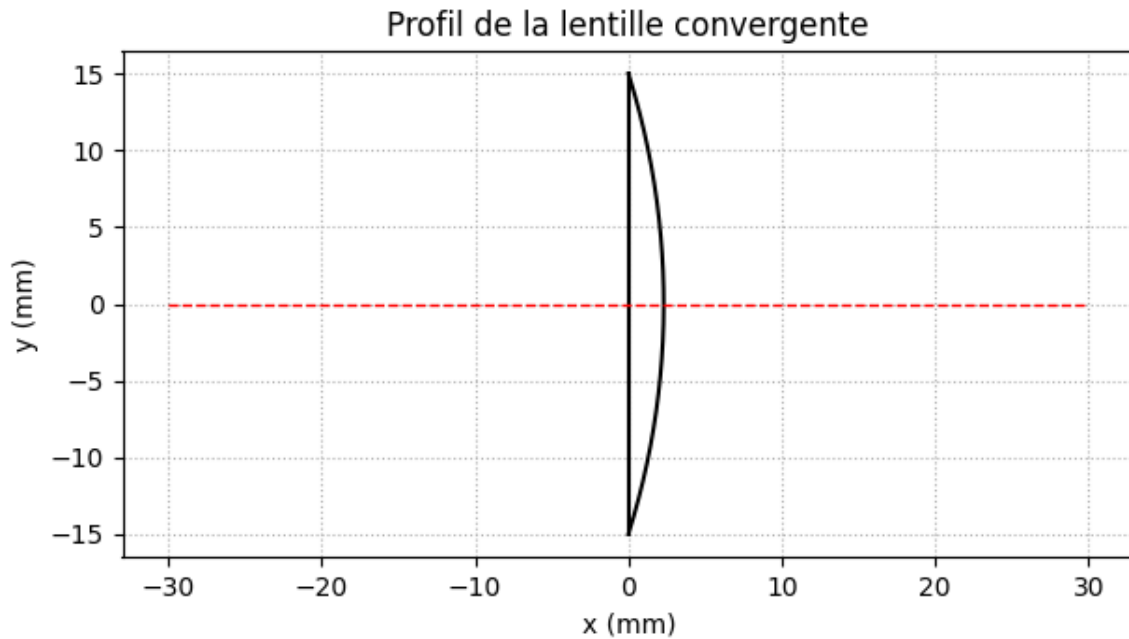
2. On ajoute l'axe optique

```

4 def lentilleconvergente(d, r):
5     e = r - r * np.sqrt(1 - (d**2) / (4 * r**2))
6     y = np.linspace(-d / 2, d / 2, 500)
7     x_convexe = np.sqrt(r**2 - y**2) - r + e
8
9     plt.plot(np.zeros_like(y), y, 'k-', lw=1.5)
10    plt.plot(x_convexe, y, 'k-', lw=1.5)
11    plt.plot([0, x_convexe[0]], [-d / 2, -d / 2], 'k-', lw=1.5)
12    plt.plot([0, x_convexe[-1]], [d / 2, d / 2], 'k-', lw=1.5)
13
14    plt.plot([-d, d], [0, 0], 'r--', lw=1, label='Axe optique')
15

```

Test de la fonction :



3. On crée la fonction `tracerrayon(z, d, r, n)`

```

16 def tracer_rayon(z, d, r, n, couleur='b'):
17     if abs(z) > d / 2:
18         return
19
20     plt.plot([-1.5 * r, 0], [z, z], color=couleur, lw=1)
21
22     # Angles et conditions d'émergence
23     sin_i1 = z / r
24     if abs(sin_i1) > 1:
25         return
26     i1 = np.arcsin(sin_i1)
27
28     # Existence d'un rayon émergent
29     if n * np.sin(i1) >= 1:
30         return
31
32     i2 = np.arcsin(n * z / r)
33
34     # Coordonnées du point de sortie M
35     e = r - r * np.sqrt(1 - (d**2) / (4 * r**2))
36     x_M = e - r + r * np.cos(i1)
37     y_M = z

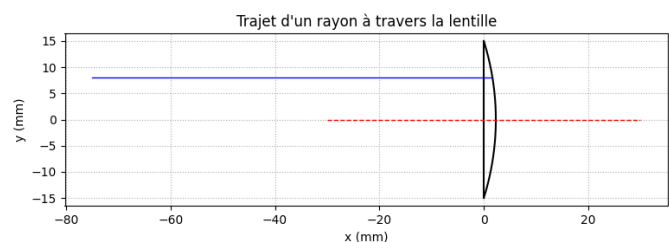
```

Test de la fonction :

```

48 plt.figure(figsize=(9, 5))
49 d_test, r_test, n_test = 30.0, 50.0, 1.5
50
51 lentilleconvergente(d_test, r_test)
52 tracer_rayon(z=8.0, d=d_test, r=r_test, n=n_test)
53
54 plt.gca().set_aspect('equal')
55 plt.xlabel("x (mm)")
56 plt.ylabel("y (mm)")
57 plt.title("Trajet d'un rayon à travers la lentille")
58 plt.grid(True, ls=':')
59 plt.show()

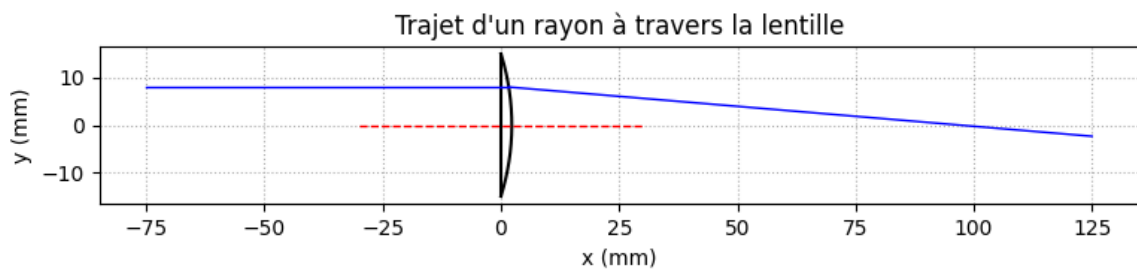
```



4. On complète la fonction

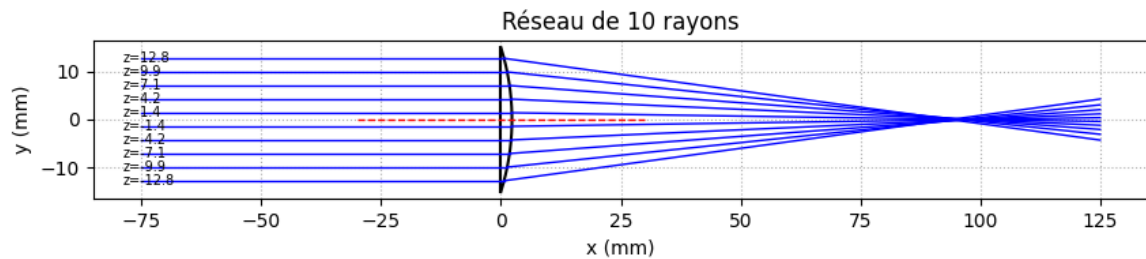
```
16 def tracer_rayon(z, d, r, n, couleur='b'):  
17     if abs(z) > d / 2:  
18         return  
19  
20     plt.plot([-1.5 * r, 0], [z, z], color=couleur, lw=1)  
21  
22     sin_i1 = z / r  
23     if abs(sin_i1) > 1:  
24         return  
25     i1 = np.arcsin(sin_i1)  
26  
27     if n * np.sin(i1) >= 1:  
28         return  
29  
30     i2 = np.arcsin(n * z / r)  
31  
32     e = r - r * np.sqrt(1 - (d**2) / (4 * r**2))  
33     x_M = e - r + r * np.cos(i1)  
34     y_M = z  
35  
36     plt.plot([0, x_M], [z, y_M], color=couleur, lw=1)  
37  
38     x_F = 2.5 * r  
39     y_F = z - (x_F + x_M) * np.tan(i2 - i1)  
40     plt.plot([x_M, x_F], [y_M, y_F], color=couleur, lw=1)  
41
```

Test de la fonction :



5. Traçage des 10 rayons :

```
43 plt.figure(figsize=(10, 6))  
44 d_test, r_test, n_test = 30.0, 50.0, 1.5  
45  
46 lentilleconvergente(d_test, r_test)  
47  
48 hauteurs = np.linspace(-d_test/2 * 0.85, d_test/2 * 0.85, 10)  
49  
50 for z in hauteurs:  
51     tracer_rayon(z, d_test, r_test, n_test)  
52     plt.text(-1.58 * r_test, z, f"z={z:.1f}", fontsize=7, verticalalignment='center')  
53  
54 plt.gca().set_aspect('equal')  
55 plt.xlabel("x (mm)")  
56 plt.ylabel("y (mm)")  
57 plt.title("Réseau de 10 rayons")  
58 plt.grid(True, ls=':')  
59 plt.show()
```



6. Un système optique est rigoureusement stigmatique si l'image d'un point est un point unique. On a un stigmatisme rigoureux uniquement sous les conditions de Gauss (rayons très proches de l'axe optique et peu inclinés).

7. On refait les parties précédentes pour une lentille divergente.

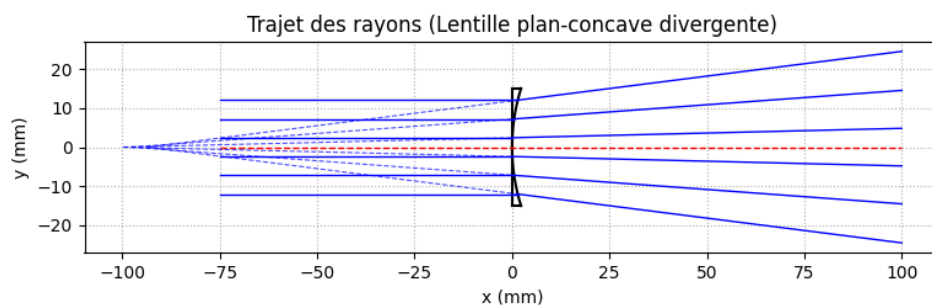
```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def lentille_divergente(d, r):
5     y = np.linspace(-d / 2, d / 2, 500)
6     x_concave = r - np.sqrt(r**2 - y**2)
7     e_max = x_concave[0]
8
9     plt.plot(np.zeros_like(y), y, 'k-', lw=1.5) # Face plane
10    plt.plot(x_concave, y, 'k-', lw=1.5) # Face concave
11    plt.plot([0, x_concave[0]], [-d / 2, -d / 2], 'k-', lw=1.5) # Bord inférieur
12    plt.plot([0, x_concave[-1]], [d / 2, d / 2], 'k-', lw=1.5) # Bord supérieur
13
14    # Axe optique
15    plt.plot([-1.5 * r, 2 * r], [0, 0], 'r--', lw=1, label='Axe optique')
16
17 def tracer_rayon_divergent(z, d, r, n):
18     if abs(z) > d / 2 or abs(z) >= r:
19         return
20
21     sin_i1 = z / r
22     if n * abs(sin_i1) >= 1:
23         return
24
25     i1 = np.arcsin(sin_i1)
26     i2 = np.arcsin(n * sin_i1)
27
28     # Coordonnées du point M
29     x_M = r - r * np.cos(i1)
30     y_M = z
31
32     y_F = (r - x_M) * np.tan(i2 - i1) + z
33     x_G = x_M - z * (r - x_M) / (y_F - z)
34
35     # Rayon incident
36     plt.plot([-1.5 * r, 0], [z, z], 'b-', lw=1)
37
38     # Rayon dans le verre
39     plt.plot([0, x_M], [z, z], 'b-', lw=1)
40
41     # Rayon émergent réel
42     x_fin = 2 * r
43     y_fin = z + (x_fin - x_M) * np.tan(i2 - i1)
44     plt.plot([x_M, x_fin], [z, y_fin], 'b-', lw=1)
45
46     # Prolongement virtuel
47     plt.plot([x_G, x_M], [0, z], 'b--', lw=0.8, alpha=0.7)
48
```

Test du programme :

```

49 plt.figure(figsize=(9, 5))
50 d_test, r_test, n_test = 30.0, 50.0, 1.5
51
52 lentille_divergente(d_test, r_test)
53
54 # Faisceau de rayons parallèles
55 hauteurs = np.linspace(-d_test / 2 * 0.8, d_test / 2 * 0.8, 6)
56 for z in hauteurs:
57     if abs(z) > 0.1:
58         tracer_rayon_divergent(z, d_test, r_test, n_test)
59
60 plt.gca().set_aspect('equal')
61 plt.xlabel("x (mm)")
62 plt.ylabel("y (mm)")
63 plt.title("Trajet des rayons (Lentille plan-concave divergente)")
64 plt.grid(True, ls=':')
65 plt.show()

```



8. Tracage des trois rayons (rayons incidents confondus)

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def lentilleconvergente(d, r):
5     e = r - r * np.sqrt(1 - (d**2) / (4 * r**2))
6     y = np.linspace(-d / 2, d / 2, 500)
7     x_convexe = np.sqrt(r**2 - y**2) - r + e
8
9     plt.plot(np.zeros_like(y), y, 'k-', lw=1.5)
10    plt.plot(x_convexe, y, 'k-', lw=1.5)
11    plt.plot([0, x_convexe[0]], [-d / 2, -d / 2], 'k-', lw=1.5)
12    plt.plot([0, x_convexe[-1]], [d / 2, d / 2], 'k-', lw=1.5)
13    plt.plot([-1.5 * r, 3 * r], [0, 0], 'k--', lw=0.8, label='Axe optique')
14

```

```

15 def tracer_rayon_chromatique(z, d, r, n, couleur, label, epaisseur):
16     if abs(z) > d / 2 or abs(z) >= r:
17         return
18
19     sin_i1 = z / r
20     if n * abs(sin_i1) >= 1:
21         return
22
23     i1 = np.arcsin(sin_i1)
24     i2 = np.arcsin(n * sin_i1)
25
26     e = r - r * np.sqrt(1 - (d**2) / (4 * r**2))
27     x_M = e - r + r * np.cos(i1)
28
29     # Rayon incident et interne avec l'épaisseur adaptée pour éviter le recouvrement
30     plt.plot([-1.5 * r, 0], [z, z], color=couleur, lw=epaisseur, alpha=0.8)
31     plt.plot([0, x_M], [z, z], color=couleur, lw=epaisseur, alpha=0.8)
32
33     # Rayon émergent (se focalise à des positions différentes sur l'axe)
34     x_F = 3 * r
35     y_F = z - (x_F + x_M) * np.tan(i2 - i1)
36     plt.plot([x_M, x_F], [z, y_F], color=couleur, lw=1.5, label=label)
37

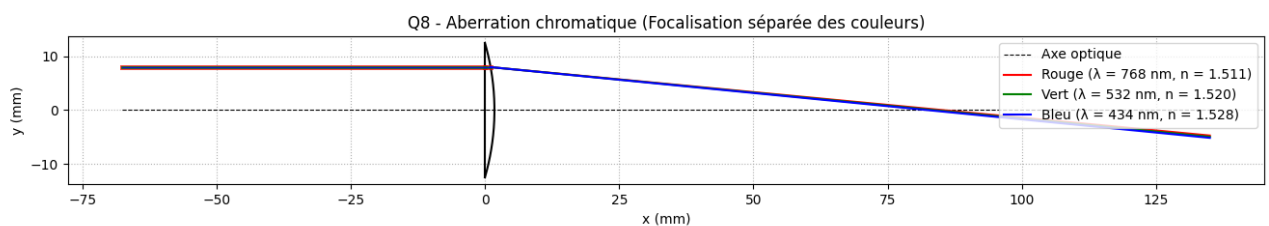
```

Test du programme :

```

38 plt.figure(figsize=(10, 5))
39 d_test, r_test = 25.0, 45.0
40 z_test = 8.0
41
42 lentilleconvergente(d_test, r_test)
43
44 # Épaisseurs différentes pour que le rouge (sous-jacent) soit visible
45 rayons_chromatiques = [
46     (1.511, 'red', 'Rouge (λ = 768 nm, n = 1.511)', 3.5),
47     (1.520, 'green', 'Vert (λ = 532 nm, n = 1.520)', 2.2),
48     (1.528, 'blue', 'Bleu (λ = 434 nm, n = 1.528)', 1.0)
49 ]
50
51 for n_c, c, lab, ep in rayons_chromatiques:
52     tracer_rayon_chromatique(z_test, d_test, r_test, n_c, c, lab, ep)
53
54 plt.gca().set_aspect('equal')
55 plt.xlabel("x (mm)")
56 plt.ylabel("y (mm)")
57 plt.title("Aberration chromatique")
58 plt.legend(loc='upper right')
59 plt.grid(True, ls=':')
60 plt.show()

```



Plus l'indice n est élevé, plus l'angle d'émergence $i_2 = \arcsin(n * \sin i_1)$ est grand. Le rayon bleu est donc plus dévié que le rayon vert, lui-même plus dévié que le rayon rouge. Il en résulte que le point focal pour la lumière bleue se trouve plus proche de la lentille que pour la lumière rouge.

9. L'aberration chromatique est proportionnelle au facteur de dispersion $\Delta n = n_{\text{bleu}} - n_{\text{rouge}}$ dans le domaine visible ($0,4\mu\text{m} \leq \lambda \leq 0,8\mu\text{m}$). Il faut donc choisir un verre avec la pente la plus faible et la plus plate possible sur la courbe $n(\lambda)$. On doit donc utiliser un verre Fluorite crown (FK51A), car ce type de verre présente une faible dispersion optique.