Optique géométrique: aberration de sphéricité

Respecter les notations de l'énoncé.

On considère une lentille plan convexe, hémisphérique, (figure 1) qui reçoit un faisceau de rayons lumineux parallèles à son axe optique (source ponctuelle à l'infini dans la direction de l'axe optique).
Le but de l'exercice est de tracer les rayons émergents de façon exacte c'est à dire non pas dans l'approximation de Gauss, mais en appliquant les lois de la réfraction au passage des dioptres qui composent la lentille.
Ceci permet de faire apparaître l'aberration de sphéricité et de la comparer dans les deux cas suivants:
I) le faisceau entre dans la lentille par sa face plane
II) le faisceau entre dans la lentille par sa face convexe.
Le système ayant la symétrie de révolution autour de l'axe optique, il suffit de tracer les rayons dans un plan quelconque contenant cet axe.
On utilisera les éléments suivants:

1) Soit un rayon lumineux réfracté au passage d'un dioptre séparant un milieu d'indice n_1 d'un milieu d'indice n_2 (figure 2). On appelle:
$\mathbf{u}_i$ le vecteur unitaire porté par le rayon incident
$\mathbf{u}_n$ le vecteur unitaire normal au dioptre au point d'incidence du rayon
$\mathbf{u}_r$ le vecteur unitaire porté par le rayon réfracté.
(Les vecteurs sont notés en caractères gras.)
La loi de la réfraction peut s'écrire:

$$\mathbf{u}_r = \frac{n_1}{n_2}\mathbf{u}_i + a\mathbf{u}_n$$

avec:

$$a = \sqrt{1 - \left(\frac{n_1}{n_2}\right)^2 \sin^2 i} - \frac{n_1}{n_2}\cos i \qquad \text{et} \qquad \cos i = \mathbf{u}_i . \mathbf{u}_n$$

Si le terme sous la racine est négatif, il y a réflexion totale.
(Pour montrer cela il suffit d'exprimer que $n_1\mathbf{u}_i - n_2\mathbf{u}_r$ est parallèle à $\mathbf{u}_n$ et que $\mathbf{u}_r.\mathbf{u}_n \geq 0$ (figure 3)).

2) Pour utiliser des vecteurs avec Maple il faut charger la bibliothèque d'algèbre linéaire par:
```
with(linalg);
```
On a ensuite la correspondance suivante entre les opérations mathématiques et l'écriture en Maple:

$\mathbf{v}$ a pour composantes x et y	`v:=vector(2,[x,y]);`
x	`v[1]`
y	`v[2]`
$\mathbf{v}_1 = \mathbf{v}_2$	`v1:=v2;`
$a\mathbf{v}$	`scalarmul(v,a);`
$a_1\mathbf{v}_1 + a_2\mathbf{v}_2$	`matadd(v1,v2,a1,a2);`
$\mathbf{v}_1.\mathbf{v}_2$	`dotprod(v1,v2);`

3) Pour faire tracer les segments de droite joignant les points $M_0(x_0, y_0)$, $M_1(x_1, y_1)$, $M_2(x_2, y_2)$, etc..., on utilise:
```
plot([[x0,y0],[x1,y1],[x2,y2],etc...]);
```
Mais ceci ne permettra de tracer qu'un seul rayon. Pour tracer tout une séquence de rayons on utilise:
```
plot({seq(f(dt*m),m=min..max)});
```
où `f(t)` est une procédure dont le résultat est une liste de la forme:

```
[[x0,y0],[x1,y1],[x2,y2],etc...]
```
dépendant du paramètre t.
Il faut donc, si M_0, M_1, M_2, etc... sont les points qui définissent le trajet d'un rayon lumineux (figure 4), écrire une procédure à laquelle on fournit en argument x_0 et y_0 et dont le résultat est:
```
[[x0,y0],[x1,y1],[x2,y2],etc...]
```

La lentille est plongée dans l'air d'indice 1, a un indice $n = 1.43$ et un rayon de courbure $R = 1$.
La face plane de la lentille est toujours placée dans le plan $x = 0$.
Tracer sur le même schéma:
- un nombre suffisant de rayons lumineux pour pouvoir faire apparaître la focalisation
- la lentille
- la position du foyer image dans l'approximation de Gauss.

Traiter successivement les deux cas:

I) Les rayons entrent par la face plane de la lentille

(Dans ce cas, le foyer image, dans l'approximation de Gauss, est à la distance $R/(n-1)$ de la face de sortie de la lentille.)

II) Les rayons entrent par la face convexe de la lentille

(Dans ce cas, le foyer image, dans l'approximation de Gauss, est à la distance $R/n(n-1)$ de la face de sortie de la lentille.)

III) Conclusion

Au vu des résultats obtenus indiquer si l'on vérifie la règle:

Plus plat plus près

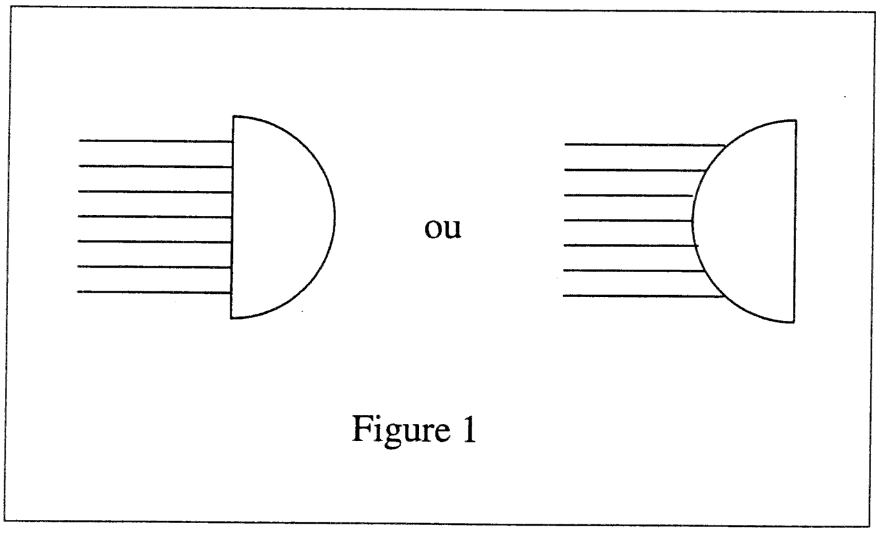


FIG. 1 –

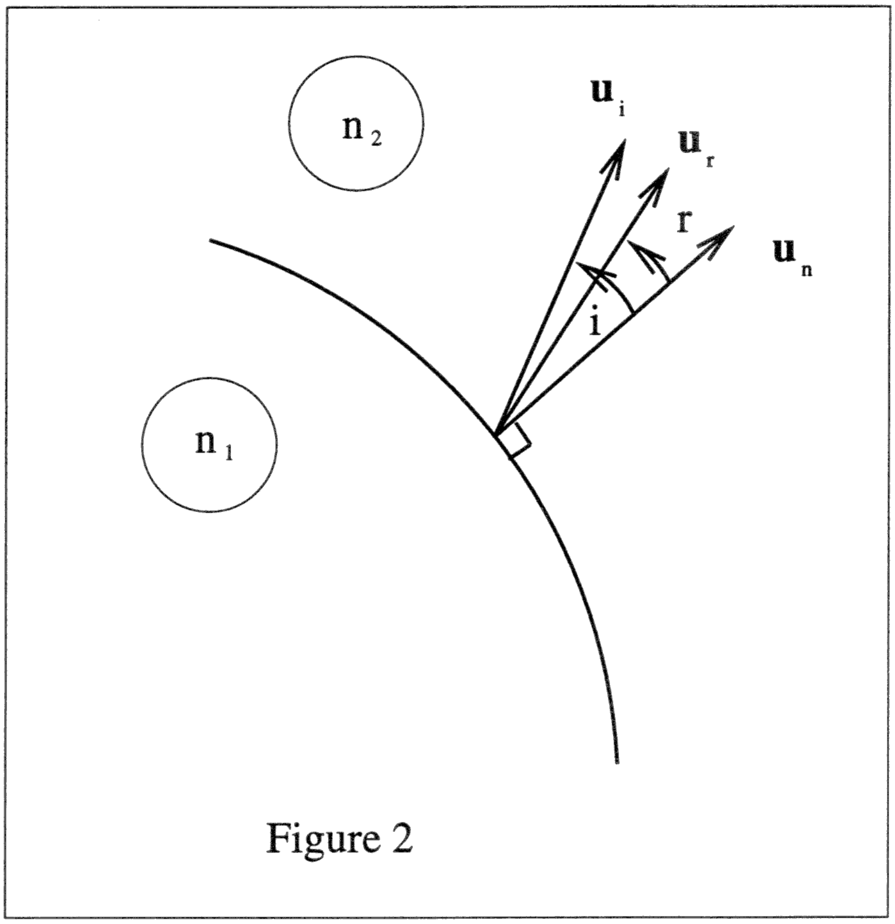


FIG. 2 –

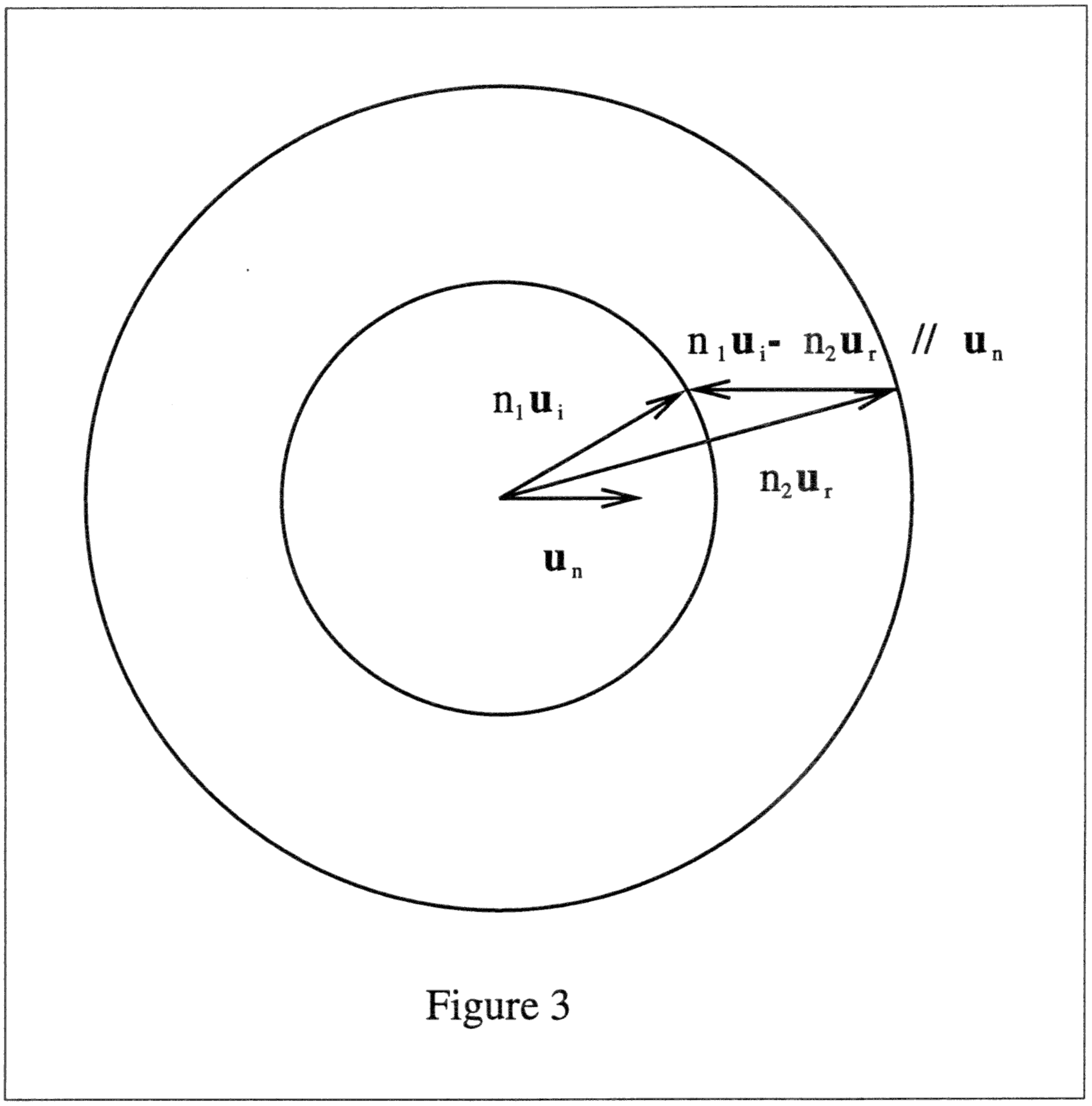


FIG. 3 –

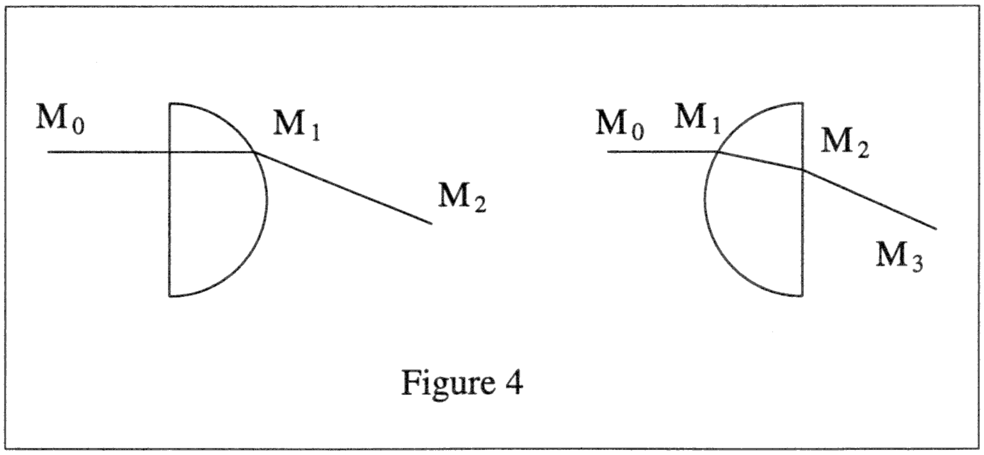
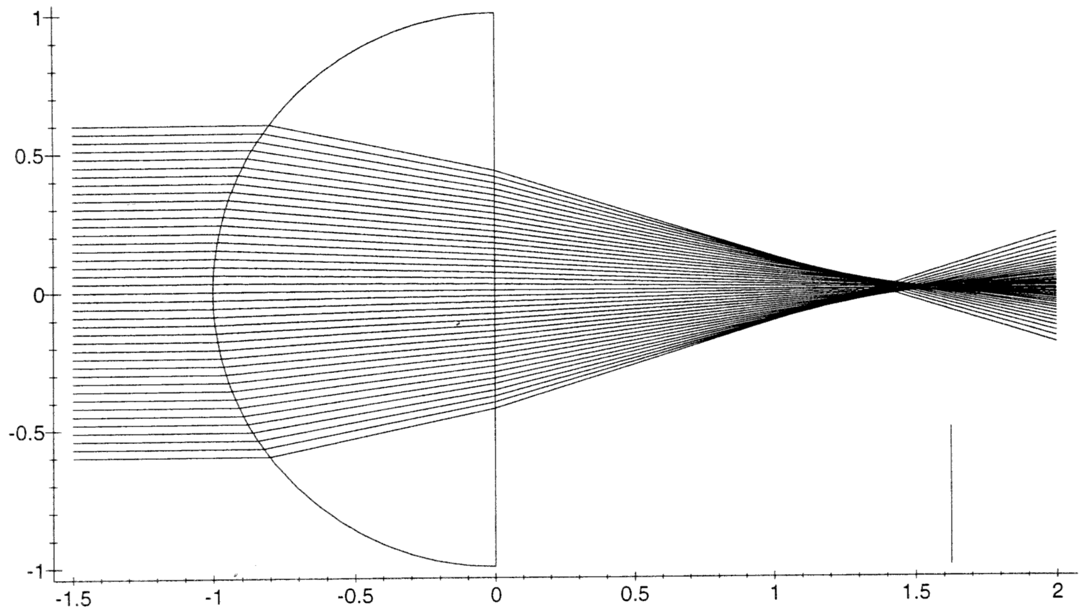
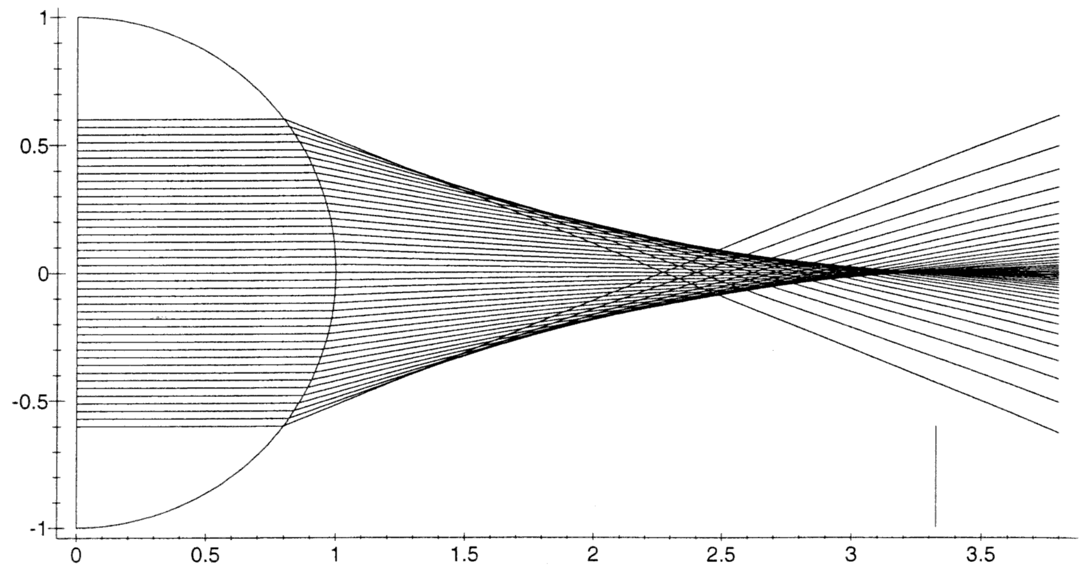


FIG. 4 –

```
> plot({seq(f(t*0.03),t=-20..20),[-sqrt(R^2-y^2),y,y=-1..1],[[0,-1
  ],[0,1]],[[R/n2/(n2-1),-1],[R/n2/(n2-1),-0.5]]},color=black,scal
  ing=constrained,axes=frame);
```




```
>
```


1
0.5
0
-0.5
-1
0
0.5
1
1.5
2
2.5
3
3.5

>