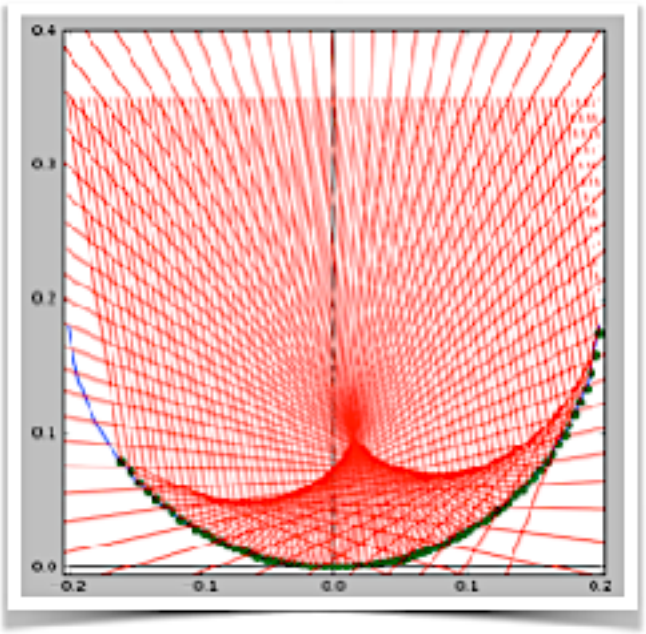
Simulation numérique : Optique Géométrique

Réflexion de rayons lumineux sur une surface quelconque

L'objet de ce problème est de simuler numériquement la réflexion d'un faisceau de rayons parallèles sur une surface réfléchissante de forme quelconque. Le profil de la surface sera donc fixé par une fonction mathématique. Le faisceau lumineux tombera sur la surface avec un angle arbitraire.



On peut ainsi retrouver la « **caustique** » ou zone d'accumulation de lumière se formant dans une tasse de café, là où se croisent les rayons.

Rq : Le seul ingrédient « mathématique » de ce problème est la loi de Descartes pour la réflection, le but étant de faire une **simulation numérique** de la réflection sur une surface complexe et non d'illustrer graphiquement le trajet des rayons pour une surface simple.

1 Première partie : lumière incidente sur le miroir

a - Définition & Tracé des axes

Tracer en noir les axes des abscisses et des ordonnées pour les intervalles [a, b] en abscisse, et [c,d] en ordonnée. On donne a = -0.2 ; b = 0.2; c = -0.05; d = +0.4 où a, b, c, et d sont en mètres. **Rq : on placera toujours en fin de code la commande plt.show(), et plt.close() au tout début.**

b - Vecteurs rayons

On note xStart et yStart les listes des coordonnées des points de départs des rayons.
On souhaite nRay=10 rayons d'abscisses entre a et b inclus, et l'ordonnée à 0.35 pour tous.
Construire les listes par la méthode des compréhensions de liste.

Tous les rayons partent par hypothèse dans la même direction, il faut donc choisir un vecteur.
Introduire dans votre code et commenter le code suivant :

```
vX = 0.25;        vY = -1.;                #C1 :
norme = (vX**2 + vY**2)**(0.5)             #C2 :
vectStart = (vX/norme, vY/norme)           #C3:
```

c - Surface du miroir

Écrire une fonction **surface**(x) qui prend en argument l'abscisse x et qui renvoie l'ordonnée y de la surface en x. Vous pouvez choisir une parabole ou toute fonction de votre choix

d - Tracé du miroir en bleu

On souhaite tracer le miroir avec nMiroir = 50 points. Construire, toujours par compréhension, les listes des abscisses et des ordonnées permettant le tracé le miroir. Complétez votre code pour introduire le tracé en bleu du miroir. [Penser à placer plt.show() à la fin].

2 Seconde partie : Tracking numérique des points d'impact sur la surface

a - cadrage de la zone de calcul

Dans la suite, nous allons tracer les rayons pas à pas. Afin de de savoir si le rayon ne sort pas d'une zone limite comprise dans les intervalles [a, b] en abscisse et [c, d] en ordonnée, écrire une fonction **inTheBox**(P) qui renvoie **True** si le point P est dans la zone limite et **False** sinon. Le point P est un objet tuple (x, y) contenant ses coordonnées.

b - Critère d'impaction sur le miroir

On considère qu'il y a impaction sur le miroir au cours du tracking pas à pas lorsque l'on passe a travers la surface du miroir d'un point P au point suivant nextP.
Écrire une fonction **impactFind**(P, nextP, surface) qui renvoie True lorsque les deux points sont de part et d'autre de la surface et False sinon.

c - Recherche des points d'impact des rayons

La recherche des points d'impact se fait par **tracking numérique** c'est-à-dire en prolongeant le rayon pas à pas depuis son point de départ et dans la direction du vecteur incident. A l'aide d'une boucle while on re-calcule à chaque itération la position du point P et du point nextP en extrapolant **vectoriellement** le vecteur direction : nextP = P + Vstart * dt où dt = 0.001.
Ceci est à faire pour chaque rayon.
Toutefois tous les rayons ne vont pas nécessairement impacter, nous allons donc stocker dans deux listes les coordonnées des points d'impact (**dernier point du tracking**) mais aussi dans une troisième liste un booléen qui vaut True si il y a eu un impact et False sinon. **Pour chaque rayon le tracking prends fin lorsque l'on détecte un impact ou si le rayon est sorti.**

Soient les listes initialement vides : **xImpact=[ ]; yImpact=[ ]; impact=[ ]**
Écrire le code permettant de remplir ces listes pour chaque rayon.
Nous aurons ainsi obtenu pour chaque rayon les points de départ et d'impact (ou de fin de trajet).
RQ : Je vous recommande de commencer par le premier rayon seul puis d'ajouter une boucle sur les rayons.

3 Troisième partie : Réflexion sur le miroir

La gestion physique de la réflexion va être menée vectoriellement. Le point d'impact étant connu, on peut déterminer le vecteur normal à la surface, et à partir du vecteur incident, calculer le vecteur réfléchi. Ces trois vecteurs seront unitaires.

a - Obtenir le vecteur normal

Ecrire une fonction **vectNormal**(x, surface) qui renvoie le vecteur normal unitaire au point x de la surface. Le vecteur sera un tuple (nx, ny). Rq : Le vecteur normal peut être construit à l'aide de la pente (tangente) de la courbe, que l'on calculera pour déplacement élémentaire dx=0.001.

b Construire le vecteur Réfléchi

A l'aide d'un **schéma** exprimer « un vecteur réfléchi » à l'aide du vecteur unitaire normal.
On pourra introduire le vecteur U projeté du vecteur incident sur la normale au point d'impact.
Ecrire une fonction **vectReflect**(P, vIncident, surface) qui prend en argument le point d'impact, le vecteur incident (qui dans notre étude sera toujours le même) ainsi que la surface, et qui renvoie le vecteur unitaire réfléchi (rx, ry).

c - Obtenir le point de sortie de zone (numérique)

Le rayon réfléchi va ensuite sortir de la zone de calcul, en un point d'arrêt permettant de tracer le rayon réfléchi [on ne prend pas en compte les doubles réflexions]. Pour obtenir ce point, on procède à nouveau par **tracking** :
Ecrire une fonction **border**(P, vRef) qui prend en argument le point d'impact et le vecteur réfléchi et qui renvoie le point de sortie de zone (tuple). On mettra à profit les fonctions précédentes.

d - Liste des fins de course des rayons

Soient xStop=[ ] et yStop=[ ] les coordonnées des points d'arrêts. Ecrire le code permettant de remplir ces listes.

5 Tracé complet des rayons

Tracer le cheminement complet des rayons : on distinguera deux cas à l'aide du tableau **impact**.
- Soit le rayon n'impacte pas le miroir => rayons incidents seuls et en tirets avec l'option 'r - - '.
- Soit il impacte => rayons incidents et réfléchis.

Tous les rayons seront tracés en rouge. On placera également un cercle vert sur les points d'impacts.

FIN