TP2Bessel

September 21, 2024

1 Methode de bessel pour les lentilles convergentes

On exploite la méthode de Bessel pour détemriner la distance focale de la lentille. On réalise pour une seule distance D entre l'objet et l'écran, les deux situations de conjugaison entre l'objet et l'image nette sur l'écran. On note d la distance entre les deux positions de la lentille.

La distance focal de la lentille est alors exprimée par la relation

$$f' = \frac{D^2 - d^2}{4 * D}$$

[1]:
```
#on commence classiquement par importer la library numpy (sous l'alias np) pour␣
↪le calcul numérique
#son sous module numpy.random pour effectuer les tirages aléatoires selon des␣
↪lois bien controlées
#et la library matplotlib.pyplot (sous l'alias pl) pour la réalisation de␣
↪graphique.
import numpy as np
import numpy.random as rd
import matplotlib.pyplot as pl
```

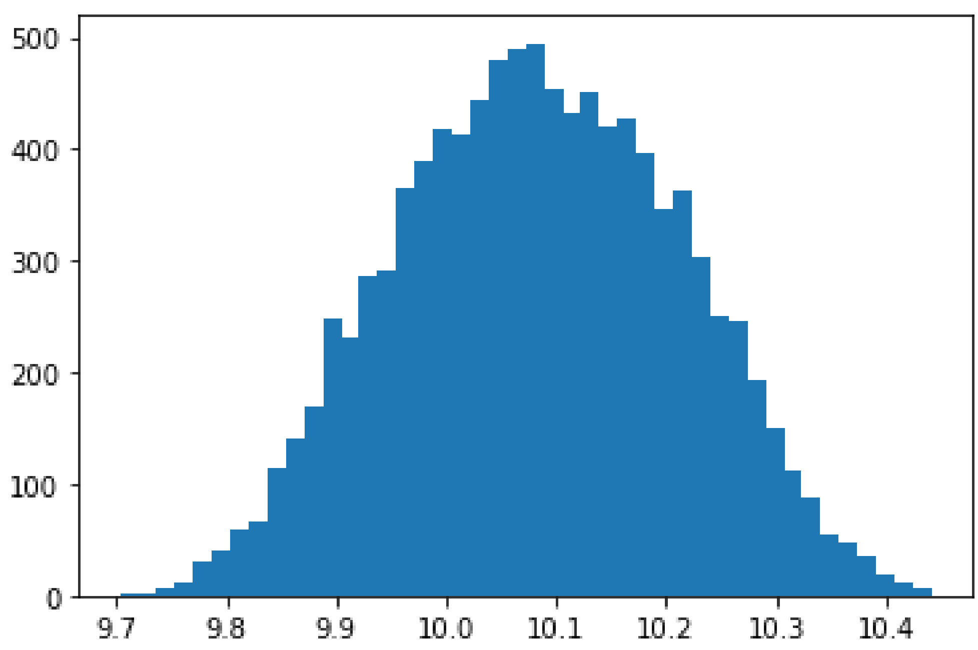
1.1 Méthode 1 , sans le viseur.

Dans la première méthode, on mesure le long du banc les positions de : - l'objet en Xo avec une précision lXo mauvaise puisqu'on fait la projection hasardeuse de l'objet sur le banc - l'écran en Xe avec une précision lXe moyenne puisqu'on ne sait pas si l'écran est exactement au dessus du repère. - la lentille en position X1 et la lentille en position X2, et comme on utilise le même pied, seule la précision de lecture et la profondeur de champ limite la précision.

La formule faisant le lien entre Xo, Xe, X1, X2 et f' étant complexe, on doit passer par une simulation Monte Carlo.

[2]:
```
N=10000     # nombre de tirages aléatoires utilisés

Xo,lXo=32.0,0.5 # position, demi largeur de l'intervalle pour l'objet en cm
Xe,lXe=98.4,0.3 # position, demi largeur de l'intervalle pour l'image en cm
X1,lX1=44.1,0.2 # position, demi largeur de l'intervalle pour la première␣
↪position de la lentille en cm
```

```python
X2,lX2=85.7,0.2 # position, demi largeur de l'intervalle pour la seconde␣
 ↪position de la lentille en cm

#on construit les 4 listes de tirages
Xo_MC=Xo+rd.uniform(-lXo,lXo,N)
Xe_MC=Xe+rd.uniform(-lXe,lXe,N)
X1_MC=X1+rd.uniform(-lX1,lX1,N)
X2_MC=X2+rd.uniform(-lX2,lX2,N)

#on construit la liste des valeurs obtenues pour la distance focale
f_MC=((Xe_MC-Xo_MC)**2-(X2_MC-X1_MC)**2)/(4*(Xe_MC-Xo_MC))

pl.hist(f_MC,bins='rice')
#et en demander l'affichage.
pl.show()

#on estime la distance focale par la moyenne sur les tirages de Monte Carlo
moyenne_f=np.average(f_MC)
print("le meilleur estimateur pour f' est la moyenne ",moyenne_f,' cm')

#on estime l'incertitude par l'écart type sur les tirages de Monte Carlo
u_f=np.std(f_MC,ddof=1)
print("l'incertitude sur f' est ",u_f,' cm')
```

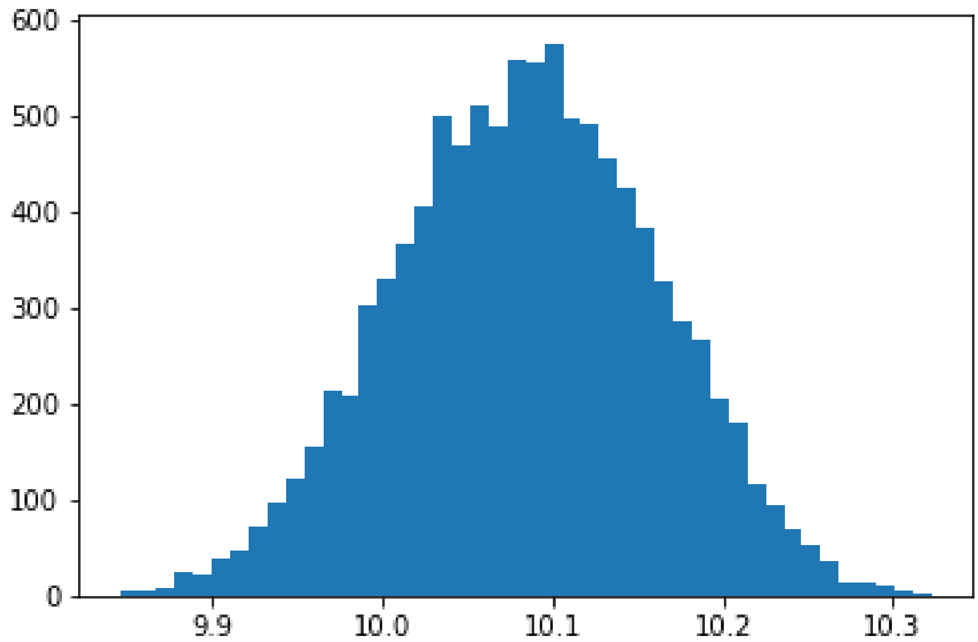



```
le meilleur estimateur pour f' est la moyenne  10.083718966404478  cm
l'incertitude sur f' est  0.12838456923946515  cm
```

1.2 Méthode 2 , avec le viseur.

Dans la première méthode, on mesure le long du banc les positions de : - l'objet en Xo avec une précision lXo bien meilleure puisqu'on exploite le viseur qui pointe sur l'objet. Seules la lecture et la profondeur de champ influence la précision. - l'écran en Xe avec une précision lXe bien meilleure puisqu'on exploite le viseur qui pointe sur l'image. Seules la lecture et la profondeur de champ influence la précision. - la lentille en position X1 et la lentille en position X2, et comme on utilise le même pied, seule la précision de lecture et la profondeur de champ limite la précision.

[4]:
```
Xo,lXo=32.0,0.2 # position, demi largeur de l'intervalle pour l'objet en cm
Xe,lXe=98.4,0.2 # position, demi largeur de l'intervalle pour l'image en cm
X1,lX1=44.1,0.2 # position, demi largeur de l'intervalle pour la première␣
 ↪position de la lentille en cm
X2,lX2=85.7,0.2 # position, demi largeur de l'intervalle pour la seconde␣
 ↪position de la lentille en cm

#on construit les 4 listes de tirages
Xo_MC=Xo+rd.uniform(-lXo,lXo,N)
Xe_MC=Xe+rd.uniform(-lXe,lXe,N)
X1_MC=X1+rd.uniform(-lX1,lX1,N)
X2_MC=X2+rd.uniform(-lX2,lX2,N)

#on construit la liste des valeurs obtenues pour la distance focale
f_MC=((Xe_MC-Xo_MC)**2-(X2_MC-X1_MC)**2)/(4*(Xe_MC-Xo_MC))

pl.hist(f_MC,bins='rice')
#et en demander l'affichage.
pl.show()

#on estime la distance focale par la moyenne sur les tirages de Monte Carlo
moyenne_f=np.average(f_MC)
print("le meilleur estimateur pour f' est la moyenne ",moyenne_f,"cm")

#on estime l'incertitude par l'écart type sur les tirages de Monte Carlo
u_f=np.std(f_MC,ddof=1)
print("l'incertitude sur f' est ",u_f,'cm')
```




```
le meilleur estimateur pour f' est la moyenne  10.084543319642341 cm
l'incertitude sur f' est  0.07609678975309234 cm
```

1.3 Comparer avec votre voisin.

Les lentilles étudiées sortent tous d'un même lot, elles présentent toutes une distance focale annoncée à 10,0cm, mais l'incertitude sur cette valeur n'est pas connue. Pour vérifier que deux évaluations d'une grandeur sont cohérentes, on peut introduire le Z-score qui compare l'écart entre les évaluations obtenues et les incertitudes associées à ces évaluations :

- le groupe 1 trouve une valeur de f_1' avec une incertitude $u_1(f')$
- le groupe 2 trouve une valeur de f_2' avec une incertitude $u_2(f')$

Le Z-score associé à ce couple d'évaluation s'écrit :

$$Z_{1/2} = \frac{|f_1' - f_2'|}{\sqrt{u_1^2(f') + u_2^2(f')}}$$

[9]:
```
#groupe 1
f1,u1=10.08,0.08
#groupe 2
f2,u2=10.35,0.06

#calcul du Z-score

Z=np.abs(f1-f2)/np.sqrt(u1**2+u2**2)
```

```python
print(" le Z-score obtenu est évalué à ",Z)

if Z>2 :
    print ("les deux évaluations sont incohérentes")

else :
    print ("les deux évaluations sont cohérentes")
```

```
 le Z-score obtenu est évalué à  2.6999999999999957
les deux évaluations sont incohérentes
```

[ ]:

```python

```