TP n°6. Etude d'un élastique (corrigé)

November 16, 2023

1 Bibliothèques nécessaires

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

2 Mesure de la constante de raideur de l'élastique

[2]:
```
m=np.array([50,100,150,200,250,300,70,170])*10**-3
leq=np.array([15.1,16,17.1,18.5,19.9,21.4,15.6,17.6])*10**-2
mg=m*9.81
```

3 La force exercée par l'élastique est-elle de la forme $-k(\ell - \ell_0)$?

3.1 Mesures

[3]:
```
m=np.array([50,100,150,200,250,300,70,170])*10**-3 # kg
leq=np.array([15.1,16,17.1,18.5,19.9,21.4,15.6,17.6])*10**-2 # m
g=9.81
mg=m*g

l0=13.6*10**-2 #m
```

[4]:
```
k=m*g/(leq-l0)
print(k)
```

```
[32.7        40.875      42.04285714 40.04081633 38.92857143 37.73076923
 34.335      41.6925    ]
```

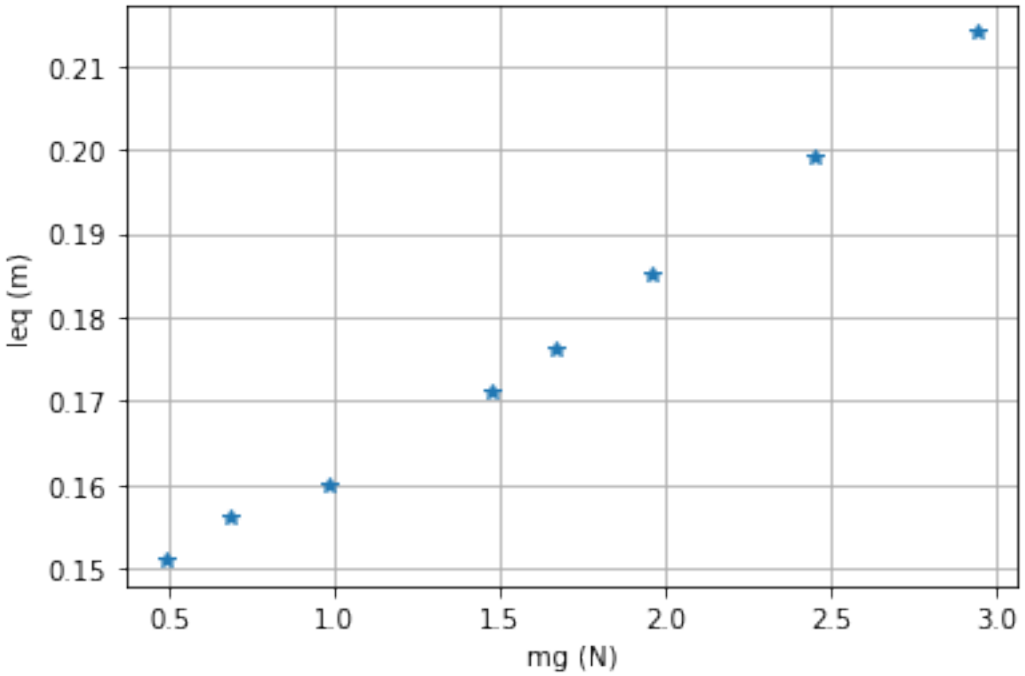
[5]:
```
kmoy=np.mean(k)
print("kmoy=",kmoy)
s_k=np.std(k,ddof=1) # écart type de la série de mesures
u_k=s_k/len(k) # incertitude-type sur la moyenne de k
print("u(k)=",u_k)
```

```
kmoy= 38.54318926609106
u(k)= 0.42914555077729305
```

Résultat de l'expérience : $\bar{k} = 38,54\ \mathrm{N/m}$; $u(\bar{k}) = 0,43\ \mathrm{N/m}$

3.2 Première courbe

[6]:
```
# Réprésentation graphique de leq en fonction du poids mg
plt.plot(mg,leq,'*')
plt.xlabel(r'mg (N)')
plt.ylabel(r'leq (m)')
plt.grid()
plt.show()
```

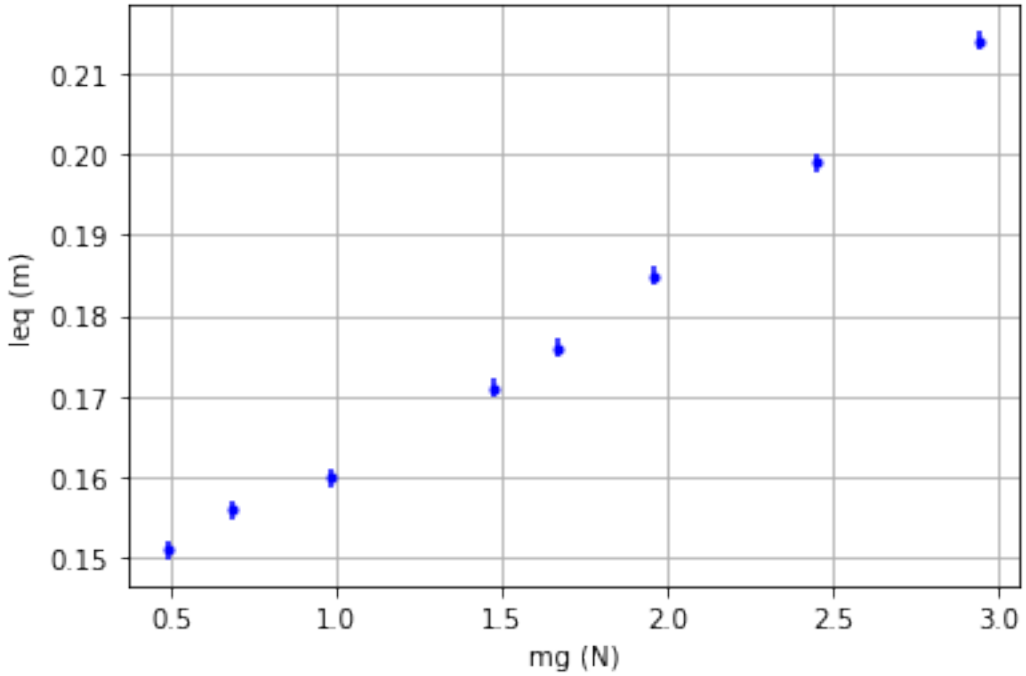


Ajout des barres d'incertitudes

Afin d'exploiter plus quantitativement la courbe, il faut étudier les incertitudes.

[7]:
```
# demie largeur de l'intervalle au sein duquel on est "certain" de trouver le␣
 ↪résultat de la mesure
Delta_leq=0.2*10**-2      # m

# incertitudes type
u_leq= Delta_leq/np.sqrt(3)
```

[8]:
```
# Réprésentation graphique de leq en fonction du poids, avec les barres␣
 ↪d'incertitude
# xerr = valeur ou tableau qui contient les incertitudes-types sur l'abscisse
# yerr = valeur ou tableau qui contient les incertitudes-types sur l'ordonnée
plt.errorbar( mg , leq , yerr = u_leq   , fmt = 'b.')
plt.xlabel(r'mg (N)')
plt.ylabel(r'leq (m)')
plt.grid()
plt.show()
```



Observation : les points peuvent laisser penser qu'ils s'alignent sur une droite (encore que...)

3.3 Régression linéaire

[9]:
```
# Régression linéaire, on modélise par un polynôme de degré 1
reg = np.polyfit( mg ,  leq  ,1)
a = reg[0] # pente
b = reg[1] # ordonnée à l'origine
print('a=',a,'b=',b)
```

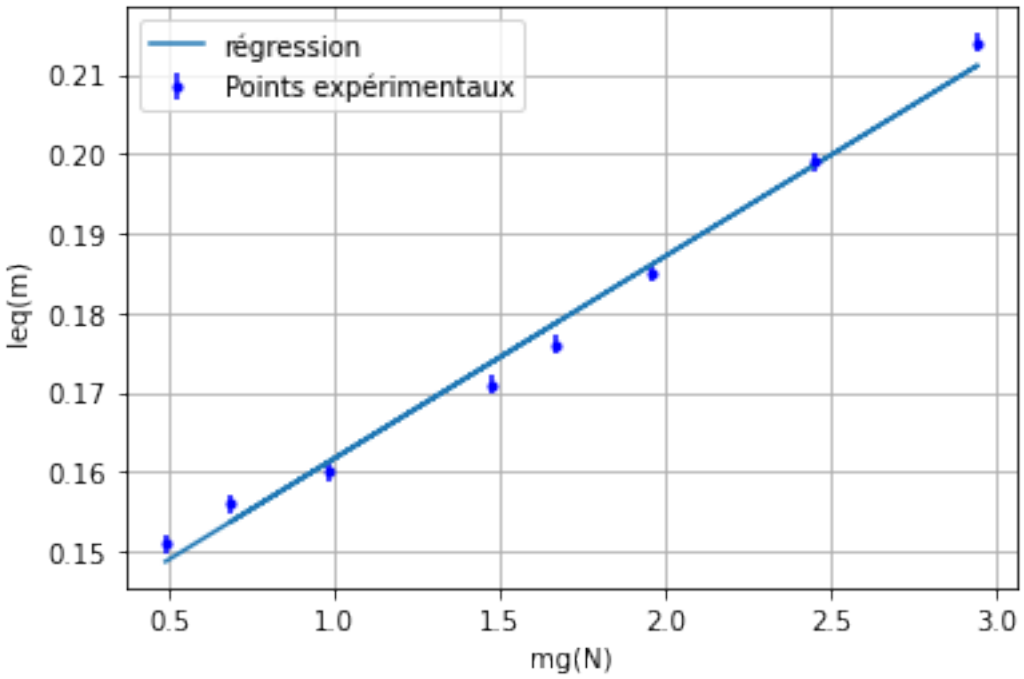
```
a= 0.025451918614118652 b= 0.13623856439127374
```

[10]:
```
# Constante de raideur issue de la régression linéaire
k = 1/a
print('k=',k)
```

```
k= 39.289768883878224
```

[11]:
```
# tableau des valeurs du polynôme reg issu de la régression linéaire
# évalué aux éléments du tableau des poids mg
# ici, reg est un polynôme de degré N=1 :
# tableau des reg[0]*P[i]**1+reg[1]*P[i]**0
tab_leq_reg = np.polyval( reg , mg)
```

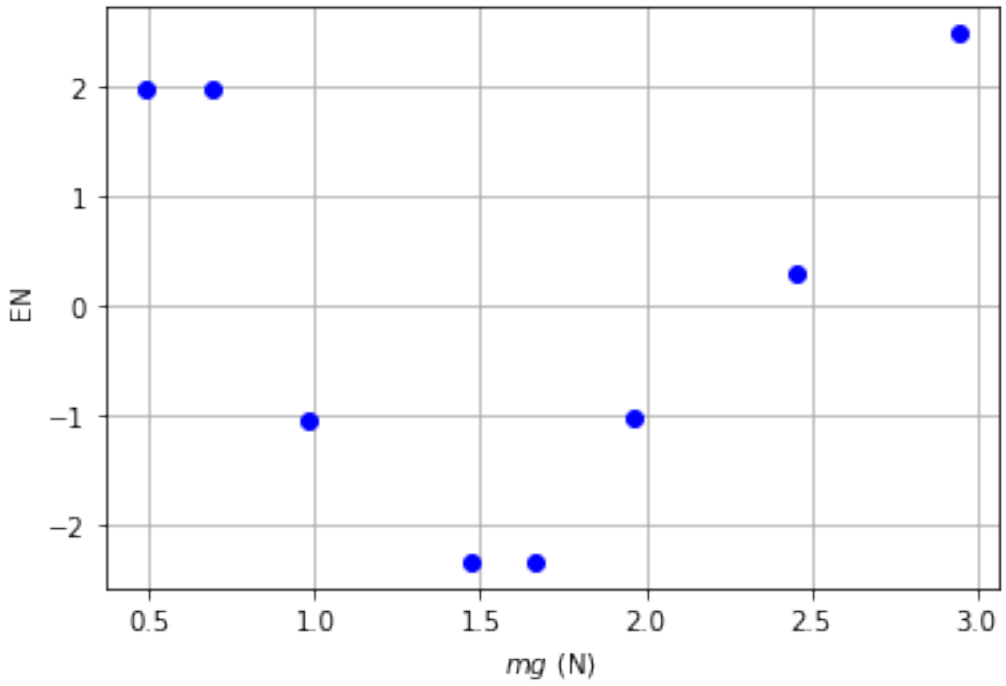
[12]:
```
plt.errorbar( mg , leq , yerr = u_leq   , fmt = 'b.',label='Points␣
 ↪expérimentaux')
plt.plot( mg  , tab_leq_reg  , label='régression')
plt.xlabel(r' mg(N) ')
plt.ylabel(r' leq(m)  ')
plt.legend()
plt.grid()
plt.show()
```



Conclusion : les barres d'incertitudes n'atteignent pas la droite de régression linéaire, cela laisse penser que le modèle linéaire de la force élastique $-k(\ell_q - \ell_0)$ utilisé pour analyser les mesures n'est pas valable.

3.4 Ecarts normalisés

[13]:
```
# Ecart normalisé entre les mesures expérimentales (leq) et les résultats issus␣
 ↪de la régression linéaire
EN=(leq-tab_leq_reg)/u_leq
plt.plot(mg,EN,'bo')
plt.xlabel(r'$mg$ (N)')
plt.ylabel('EN')
plt.grid()
plt.show()
```



On a représenté les écarts normalisés $E_N = \dfrac{\ell_{\text{éq, mesurées}} - \ell_{\text{éq, calculées à partir de la régression}}}{u(\ell_q)}$, en fonction de mg. On constate que pour beaucoup de mesures, cet écart normalisé est en valeur absolue supérieure à 2. Le modèle linéaire n'est pas compatible avec les mesures effectuées.

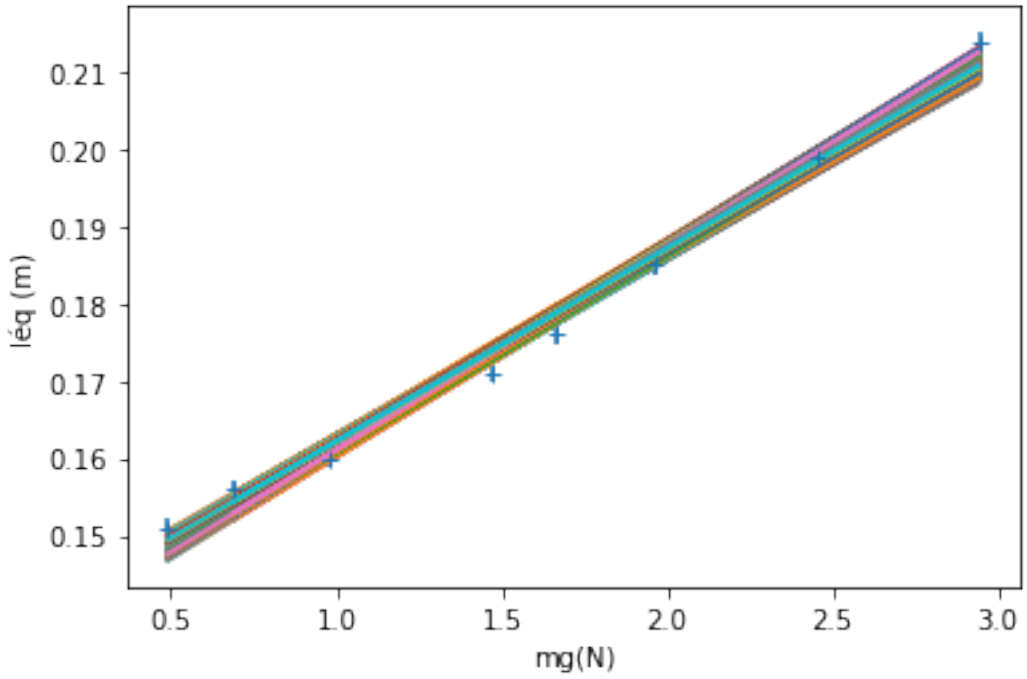
Cela peut venir de : - le modèle de la force de rappel élastique ne s'applique pas à l'élastique étudié ici, dans les conditions dans lequel on l'a étudié, - ou incertitudes sur la mesure de la longueur à l'équilibre sous-évaluées, - ou des mesures mal effectuées (problème de paralaxe important ici !)

Je pencherai plutôt sur la première raison…

3.5 Régression linéaire et Monte-Carlo

Si le modèle linéaire avait validé par les expériences précédentes, pour obtenir les incertitudes-types sur ℓ_0 et sur k (sur les abscisses et les ordonnées à l'origine), il est nécessaire d'utiliser la simulation Monte-Carlo.

[14]:
```
N= 1000    # nombre d'expériences simulées

l0_MC=[] # liste des N ordonnées à l'origine générées par régression linéaire
k_MC=[] # liste des N valeurs de k calculées, à partir de la pente de la␣
 ↪régression linéaire

for i in range(N):
    # chaque valeur de i correspond à une expérience simulée
    # pour chaque valeur de m et léq précédente on génère une valeur
    # dans l'intervalle  [leq-Delta_leq,leq+Delta_leq] pour la longueur à␣
 ↪l'équilibre
    leq_MC=[] # liste des valeurs de leq générées pour une expérience
    for j in range(len(m)):
        leqij=np.random.uniform(leq[j]-Delta_leq,leq[j]+Delta_leq)
        leq_MC.append(leqij)
    p=np.polyfit(mg,leq_MC,1)
    l0_MC.append(p[1])
    k_MC.append(1/p[0])
    plt.plot(mg, np.polyval (p , mg ))

plt.errorbar(mg,leq,yerr=Delta_leq/np.sqrt(3),fmt='+')
plt.xlabel(r'mg(N)')
plt.ylabel(r'léq (m)')
plt.show()

l0_moy=np.mean(l0_MC)
u_l0=np.std(l0_MC,ddof=1)

k_moy=np.mean(k_MC)
u_k=np.std(k_MC,ddof=1)
```



[15]:
```
# Résultats de l'expérience :
print('Constante de raideur :  k=', k_moy   ,' ; u(k)=', u_k  )
print('Longueur à vide  :  l0=', l0_moy  ,' ; u(l0)=', u_l0)
```

```
Constante de raideur :  k= 39.338891741064025  ; u(k)= 0.8123560429234016
Longueur à vide  :  l0= 0.13625897641014859  ; u(l0)= 0.0009479051704359583
```

Résultat :

$k = 39,29\ \mathrm{N/m}$; $u(k) = 0,81\ \mathrm{N/m}$

$\ell_0 = 13,62\ \mathrm{cm}$; $u(\ell_0) = 0,09\ \mathrm{cm}$