Variabilité

I Capacité numérique :

Simuler, à l'aide d'un langage de programmation ou d'un tableur,
un processus aléatoire permettant de caractériser la variabilité
de la valeur d'une grandeur composée

II Chute libre

II.1 Dispositif expérimental

On cherche à mesurer l'accélération de la pesanteur g en étudiant la chute libre d'un corps dans le vide. Le dispositif consiste en :

- un objet de masse m en chute libre dans le vide ;
- est lâché sans vitesse initiale d'une altitude h_1 ;
- sa vitesse v est mesurée par un capteur spécifique quand il passe en un point d'altitude $h_2 < h_1$.

II.2 Modèle

Montrer qu'on a :
#+ipynb-newcell $v^2 = 2g(h_1 - h_2)$

II.3 Incertitudes

On cherche à observer l'effet sur l'incertitude sur la mesure de g des sources d'incertitude suivantes :

- incertitude sur les lectures des altitudes h_1 et h_2 sur une règle ;
- incertitude sur la mesure de la vitesse par un capteur.

On **simule** ici numériquement les répartitions des valeurs mesurées pour ces grandeurs si on reproduisait un grand nombre de fois la manipulation, en supposant connue la loi de répartition des erreurs.

1. Lecture des altitudes sur une règle

   On suppose une répartition uniforme d'erreurs entre deux graduations de la règle distantes de Δh.

   On rappelle que l'incertitude-type vaut alors :

   #+ipynb-newcell

   $\Delta h/(2\sqrt{3})$.

   On utilisera la fonction `random.random_sample` pour tirer un nombre flottant aléatoire correspondant à une altitude lue entre deux graduations de la règle.

2. Mesure de la vitesse

   On suppose une répartition d'erreurs autour d'une vitesse v donnée par une loi normale d'incertitude-type Δv.

   On utilisera la fonction `random.normal` pour tirer un nombre flottant aléatoire correspondant cette loi normale.

III Code

III.1 Bibliothèques nécessaires

```
%matplotlib inline
import numpy as np
from matplotlib import pyplot
```

Pour utiliser un affichage scientifique

```
sci_format = ":.3e"
np.set_printoptions(formatter={'float':sci_format})
```

III.2 Fonctions permettant de réaliser les tirages aléatoires

```python
# tirage aléatoire pour une loi normale de moyenne X[0] et d'incertitude-type X[1]
def tirage_normal(X):
  return X[0] + X[1]*np.random.normal()

# tirage aléatoire pour une répartition uniforme entre X[0] et X[1]
def tirage_uniforme(X):
  return X[0] + (X[1]-X[0])*np.random.random_sample()
```

III.3 Paramètres

On suppose pour cette simulation connue les valeurs vraies de g, h_1 et h_2. On en déduit celle de v.

```python
# Accélération de la pesanteur
g = 9.80665 #en m/s, valeur normale de la CGPM

# Hauteur de chute
h1 = 3 # en m
h2 = 2 # en m
h0 = h1-h2
Deltah = 5e-3 # en m, demi-largeur de la lecture de h

# Vitesse atteinte
v0 = np.sqrt(2* g * h0) # en m/s
Deltav = 1e-2 # en m/s, incertitude-type sur la mesure de v
```

III.4 Simulation des mesures

On crée des listes `numpy array` de mesures de h_1, h_2 et v pour faciliter leur manipulation.

```python
# Mesures
N = 10000 # nombre de points de mesure
g_calculee = np.array([])

h1min = h1-Deltah
h1max = h1+Deltah
h2min = h2-Deltah
h2max = h2+Deltah

MesuresV = np.array([tirage_normal([v0,Deltav]) for i in range(N)])
MesuresH1 = np.array([tirage_uniforme([h1min,h1max]) for i in range(N)])
MesuresH2 = np.array([tirage_uniforme([h2min,h2max]) for i in range(N)])
```

III.5 Étude de g

on calcule les valeurs de g, leur valeur moyenne et l'écart-type de leur distribution.

```python
# Valeurs de $h = h1-h2
MesuresH = MesuresH1 - MesuresH2
# On pourrait faire la même chose sans numpy array avec:
# MesuresH = [MesuresH1[i] - MesuresH2[i] for  i in range(N)]
# ou
# for i in range(N)
#    MesuresH[i] = MesuresH1[i] - MesuresH2[i]

# Valeurs calculées de g = v^2/(2h)
g_calculees = MesuresV**2/(2*MesuresH)
g_moyenne = np.average(g_calculees)
Stdevg = np.std(g_calculees)
print('moyenne des valeurs calculées de g:',"{:.3e}".format(g_moyenne))
print('incertitude-type sur les valeurs calculées de g:',"{:.3e}".format(Stdevg))
```

On affiche un histogramme de ces valeurs.

```python
pyplot.hist(g_calculees, bins = 50, color = 'blue', edgecolor = 'black')
pyplot.xlabel('g (m/s^2)')
pyplot.ylabel('effectif')
pyplot.title('Pour '+str(N)+ ' iterations')
# pyplot.show()
```

IV Questions

IV.1 Incertitudes composées

1. Afficher l'écart-type relative des mesures de g.

```python
StdevRelatg = Stdevg/g_moyenne
print('écart-type relatif sur g:', "{:.3e}".format(StdevRelatg))
```

2. Afficher les incertitudes-types et les incertitudes-types relatives sur v, $h1$, $h2$.

```python
print('incertitude-type sur v:',Deltav)
DeltaRelatv = Deltav/v0
print('incertitude-type relative sur v:',DeltaRelatv)
Stdevh = Deltah/np.sqrt(3) # attention au facteur $\sqrt{3}$
StdevRelath1 = Stdevh/h1
StdevRelath2 = Stdevh/h2
print('incertitudes-types sur h1,h2:',Stdevh)
print('incertitudes-types relatives sur h1,h2:',StdevRelath1,StdevRelath2)
```

3. Afficher l'écart-type des valeurs mesurées de h et comparer à l'incertitude-type composée à partir des incertitudes-types de h_1 et h_2. Calculer l'incertitude-type relative sur h.

```python
Stdevh0 = np.std(MesuresH)
print('écart-type des mesures de h1-h2' ,Stdevh0)
Deltah0 = Deltah*np.sqrt(2)/np.sqrt(3) # attention au facteur $\sqrt{3}$
print('incertitude-type composée sur h1-h2' , Deltah0)
```

Les deux grandeurs précédentes doivent correspondre (à la dispersion statistique près).

4. Déterminer et afficher l'incertitude-type relative composée sur g en fonction des incertitudes-types relatives précédentes et comparer à l'écart-type relatif sur g.

   On a $g = v^2/(2h)$, soit $\Delta g/g = \sqrt{4(\Delta v/v)^2 + (\Delta h/h)^2}$, qu'on compare à l'écart-type des valeurs calculées de v.

```python
StdevRelath0 = Stdevh0/h0
incertitude_relative_composee = np.sqrt(4*DeltaRelatv**2 + StdevRelath0**2)
print('incertitude-type relative composée sur g:',incertitude_relative_composee)
print('écart-type relatif sur g:',StdevRelatg)
```

Les deux grandeurs doivent correspondre (à la dispersion statistique près).

5. Augmenter d'un facteur 10 la précision sur la vitesse ? Quel est l'effet sur la précision de la détermination de g. Commenter.

   Les incertitudes-types relatives sur la vitesse et la hauteur étaient du même ordre de grandeur. Quand on diminue celle sur v d'un facteur 10, elle devient négligeable devant celle sur h et l'incertitude-type relative globale n'est que faiblement diminuée.

IV.2 Effet du nombre de mesures

1. Changer le nombre d'itérations N d'un facteur 100 dans un sens ou dans l'autre. Cela change-t-il les résultats précédents ?

   On passe de $N = 1e4$ à $Np = 100$:

```python
Np = 100 # nombre de points de mesure
gp_calculee = np.array([])

MesuresVp = np.array([tirage_normal([v0,Deltav]) for i in range(Np)])
MesuresH1p = np.array([tirage_uniforme([h1min,h1max]) for i in range(Np)])
MesuresH2p = np.array([tirage_uniforme([h2min,h2max]) for i in range(Np)])
MesuresHp = MesuresH1p - MesuresH2p

g_calculeesp = MesuresVp**2/(2*MesuresHp)
g_moyennep = np.average(g_calculeesp)
Stdevgp = np.std(g_calculeesp)
print('moyenne des valeurs calculées de g (N=100)',g_moyennep)
print('incertitude-type sur les valeurs calculées de g (N=100):',Stdevgp)
```

```python
pyplot.hist(g_calculeesp, bins = 50, color = 'blue', edgecolor = 'black')
pyplot.xlabel('g (m/s^2)')
pyplot.ylabel('effectif')
pyplot.title('Pour '+str(Np)+ ' iterations')
# pyplot.show()
```

La moyenne et l'écart-type ne changent pas.

2. Pour un jeu de $N = 1e4$ mesures, effectuer des moyennes de $m = 100$ et étudier l'incertitude-type des $N/m = 100$ mesures obtenues. Vérifier qu'il est réduit d'un rapport $\sqrt{m}$.

   On crée un tableau des valeurs des moyennes sur $m = 100$ valeurs.

```python
m = 100
Nm = int(np.floor(N/m))

MesuresVm = np.zeros(Nm)
MesuresH1m = np.zeros(Nm)
MesuresH2m = np.zeros(Nm)
MesuresHm = np.zeros(Nm)

for i in range(Nm):
    for j in range(m):
        MesuresVm[i] += MesuresV[m*i+j]
        MesuresH1m[i] += MesuresH1[m*i+j]
        MesuresH2m[i] += MesuresH2[m*i+j]

MesuresHm = (MesuresH1m - MesuresH2m)/m
MesuresVm = MesuresVm/m
```

On calcule la moyenne et l'écart-type des valeurs de g calculées. On vérifie que la moyenne ne change pas, mais que l'écart-type est réduit d'un facteur $\sqrt{m} = 10$.

```python
g_calculeesm = MesuresVm**2/(2*MesuresHm)
print('moyenne des moyennes de $m$ mesures', np.average(g_calculeesm))
print('écart-type des moyennes de $m$ mesures', "{:.3e}".format(np.std(g_calculeesm)))
print('écart-type des mesures brutes', "{:.3e}".format(np.std(g_calculees)))
```