בס"ד

$N_A = 6.02214076 \times 10^{23}$ (1)

[בשנת 2017 קבעו את ההגדרה של 1 mol כך שהמספר הזה
בלי אי-ודאות בכלל. מאז אומרים שזה קבוע שככה הוא מדויק
ולא ניתן למדוד". בכלל מאז:]

נניח שהאי-ודאות היא רק בספרה האחרונה, כלומר
פחות מ $0.0000001 \times 10^{23}$.

המדידה הראשונה היא בהפרש של פחות מ"סיגמה"
מהערך המקובל ואין הרבה מקום לפרשנות. אם היא
מבוצעת כמתואר, אין טעם לחלוק.

השני טוען שיש לו שגיאה יחסית קטנה. קודם כל
אנחנו יודעים ששגיאה יחסית מאוד לא מוצגת/משתנה
רלוונטי. שכן האי-ודאות שלו היא כמה סדרי גודל יותר
מאשר של הערך המקובל. לטענו לא סבירה/נכונה.

```
#Some basics for python
import numpy as np
import matplotlib.pyplot as plt
import scipy.special as spsp #I use this for the error function below

import warnings
warnings.simplefilter("ignore")
from math import pow
```

**Problem 1**

A) The basic idea is to perform a calibration, i.e. measure something known. The uncertainty on the calibration is set by the uncertainty of how well known, the reference voltage is and generally speaking there could be statistical uncertainty on the calibration measurements (which could be reduced by multiple measurements).

B) It would be reasonable to call 0.2 V the uncertainty. Calling that 2 sigma, i.e. uncertainty of 0.1 V would also be reasonable. Generally in ambiguous cases we tend to be conservative. The answer for V1 should then be:

```
C=0.84
dC=0.07
v1_meas=17.0
dv1_meas=0.2
v1=v1_meas*C
dv1=v1*np.sqrt(pow(dC/C,2)+pow(dv1_meas/v1_meas,2))
print("v1 = ",v1," +- ",dv1)
print("v1 = ",round(v1,1)," +- ",round(dv1,1))
```

```
    v1 =  14.28  +-  1.2018003161923365
    v1 =  14.3  +-  1.2
```

C) There is no reason to think the uncertainty on v2_meas should be *less* than the uncertainy on v1_meas. One might even think it should be larger; both are reasonable approaches. Since we were conservative with the uncertainty above, let's leave it the same here, i.e. 0.2 V.

```
v2_meas=15.0
dv2_meas=dv1_meas
v2=v2_meas*C
dv2=v2*np.sqrt(pow(dC/C,2)+pow(dv2_meas/v2_meas,2))
print("v2 = ",v2," +- ",dv2)
print("v2 = ",round(v2,1)," +- ",round(dv2,1))
```

```
v2 =  12.6  +-  1.0633550676984618
v2 =  12.6  +-  1.1
```

D) The key here is that the multiplicative correction factor, C, should fully cancel: V1/V2=(C x v1_meas) /(C x v2_meas) = v1_meas/v2_meas . We also assume that dv1_meas and dv2_meas are uncorrelated.

```
ratio=v1_meas/v2_meas
dratio = ratio*np.sqrt(pow(dv1_meas/v1_meas,2)+pow(dv2_meas/v2_meas,2))
print("Ratio = ",ratio," +- ",dratio)
print("Ratio = ",round(ratio,2)," +- ",round(dratio,2))
```

```
Ratio =  1.1333333333333333  +-  0.020152504975563795
Ratio =  1.13  +-  0.02
```

Note that if you had incorrectly included C the central value would of course be unaffected but the uncertainty would appear inflated:

```
wratio=v1/v2
dwratio=wratio*np.sqrt(pow(dv1/v1,2)+pow(dv2/v2,2))
print("Wrong Ratio = ",wratio," +- ",dwratio)
print("Wrong Ratio = ",round(wratio,2)," +- ",round(dwratio,2))
```

```
Wrong Ratio =  1.1333333333333333  +-  0.1350763844261077
Wrong Ratio =  1.13  +-  0.14
```

This is an extreme case of uncertainty correlation where the uncertainty should have cancelled.

**Problem 2**