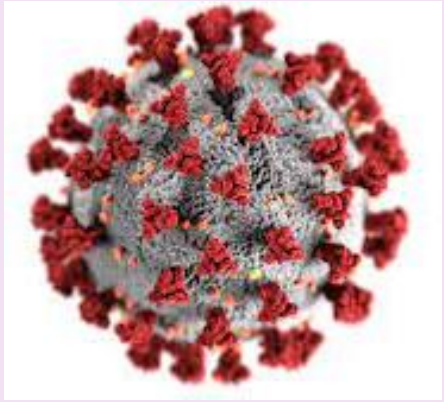
Lisa LUNARDI 11706

Thème : La ville

ETUDE DE PIGMENTS THERMOCHROMIQUES: MESURES DE TEMPÉRATURE ET D'HYGROMÉTRIE

oceasoft.com

Crise du COVID-19



who.int

Contrôle de l'aération des pièces → Mesure d'hygrométrie

Guerre en Ukraine

francebleu.fr

Restrictions énergétiques → Mesure de température

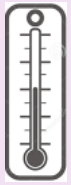
ETUDE DE PIGMENTS THERMOCHROMIQUES: MESURES DE TEMPÉRATURE ET D'HYGROMÉTRIE

Comment estimer la température et l'humidité à l'aide d'un capteur au complexe de cobalt pour optimiser le chauffage d'une pièce ?

- Coefficients d'absorption molaire

fr.vwr.com

- Couleur en fonction de la température



- Grandeurs thermochimiques

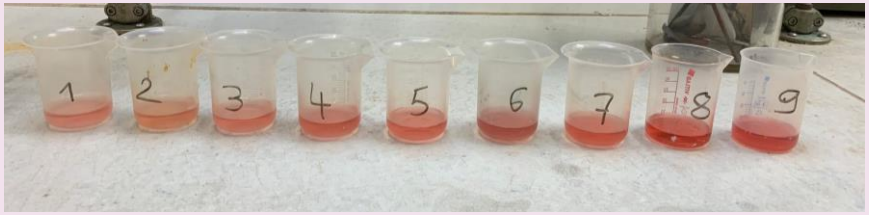
python™

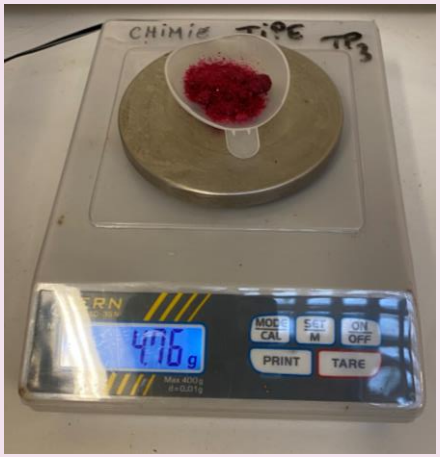
- Couleur en fonction de l'humidité

?

$$Co(H_2O)_6^{2+} + 4Cl^- \rightleftarrows Co(Cl)_4^{2-} + 6H_2O$$

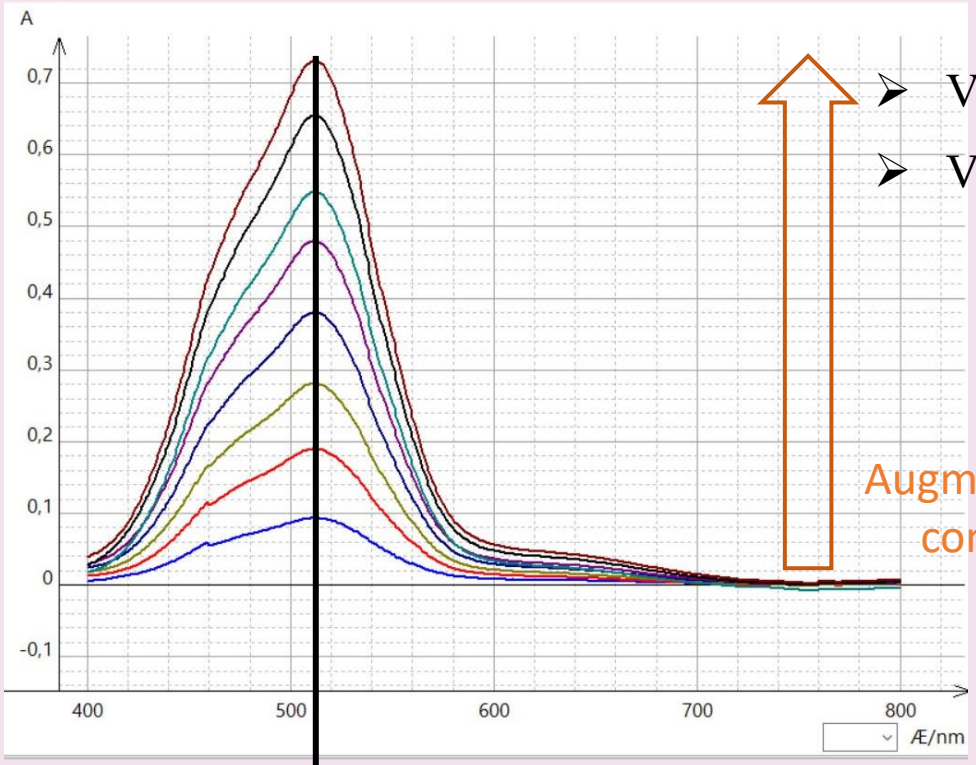
Déterminer $\varepsilon(Co(H_2O)_6^{2+})$





Déterminer $\varepsilon(CoCl_4^{2-})$

Courbes A = f(λ) pour différentes concentrations en $Co(H_2O)_6^{2+}$



- Valeur : $\varepsilon(Co(H_2O)_6^{2+}) = 4,57 \pm 0,13 L/mol/cm$
- Valeur attendue :

$\varepsilon(Co(H_2O)_6^{2+}) = 4,93 \pm 0,15 L/mol/cm$

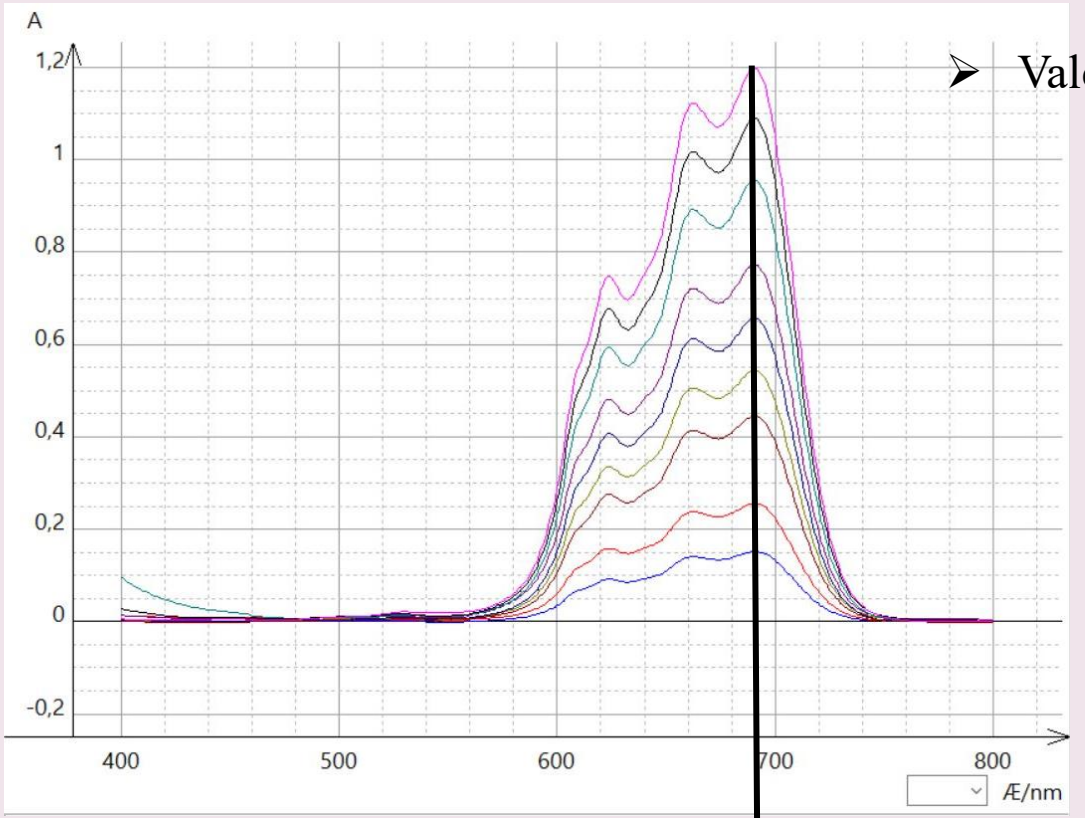
Augmentation de la concentration

511nm

Couleur

Complémentaire

Courbes A = f(λ) pour différentes concentrations en $CoCl_4^{2-}$

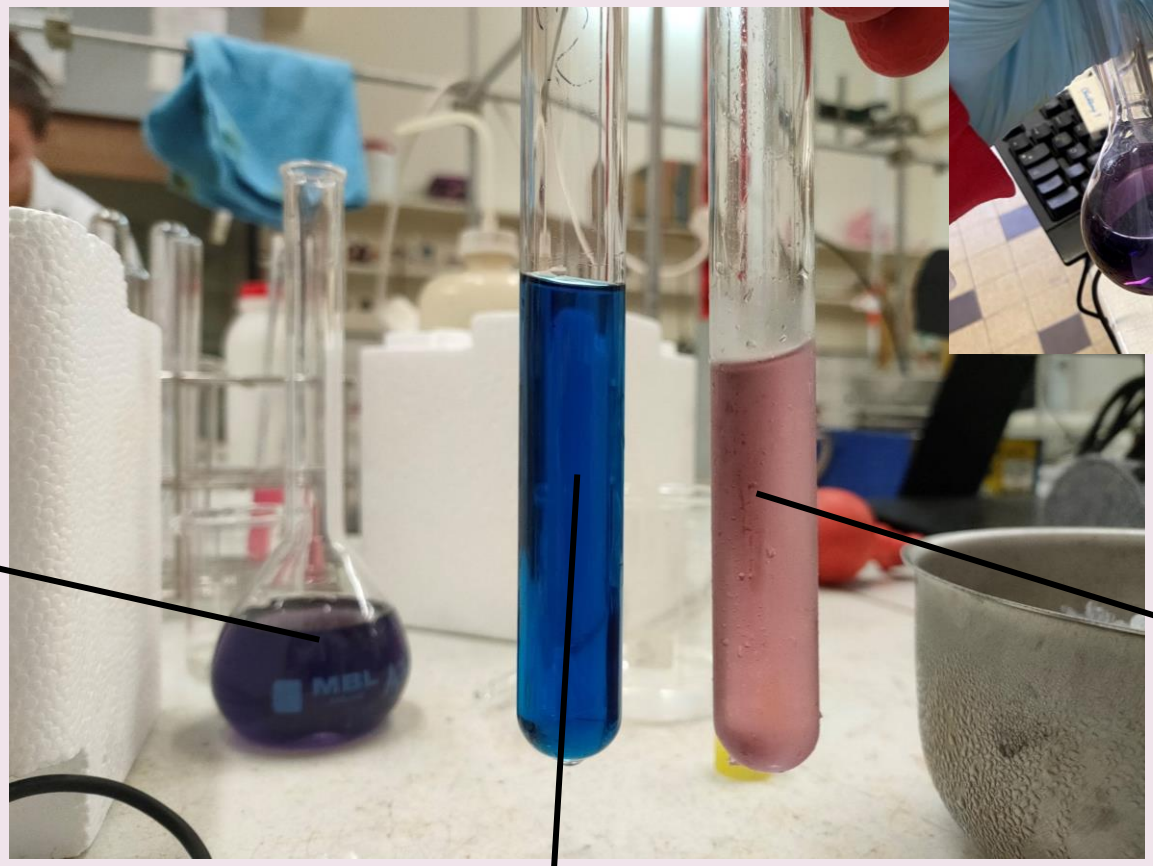


690 nm

➢ Valeur : $\varepsilon(Co(Cl)_4^{2-}) = 664 \pm 0,68$ L/mol/cm

Couleur

Complémentaire

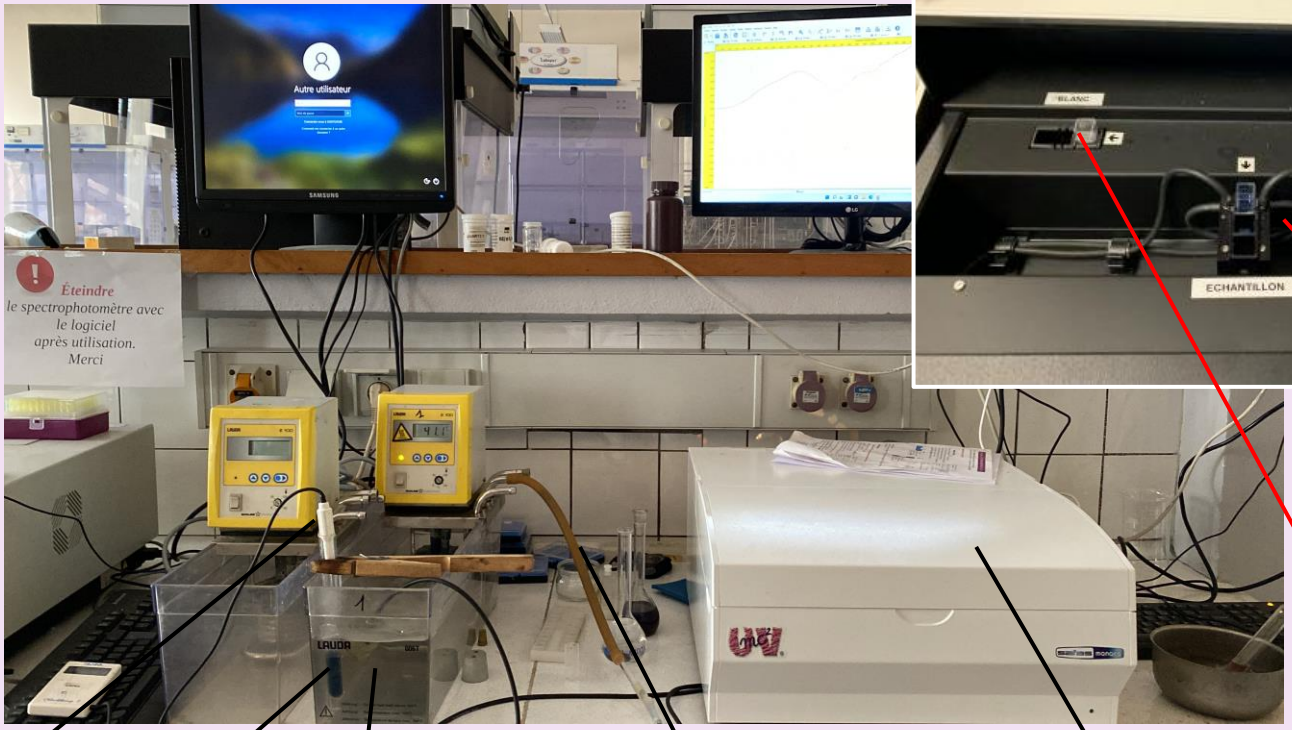


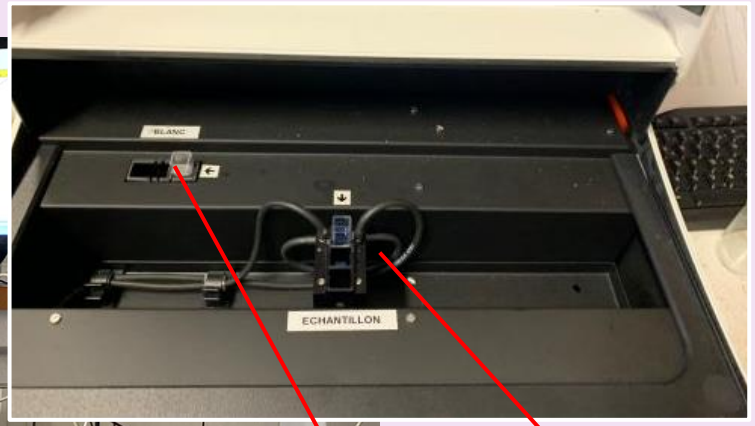
Solution à température ambiante

Solution à basse température

Solution à haute température

Mesurer l'absorbance d'une solution de complexe de cobalt en fonction de sa température





Porte cuve thermostaté

Cuve pour le blanc

thermomètre

solution de complexe de cobalt

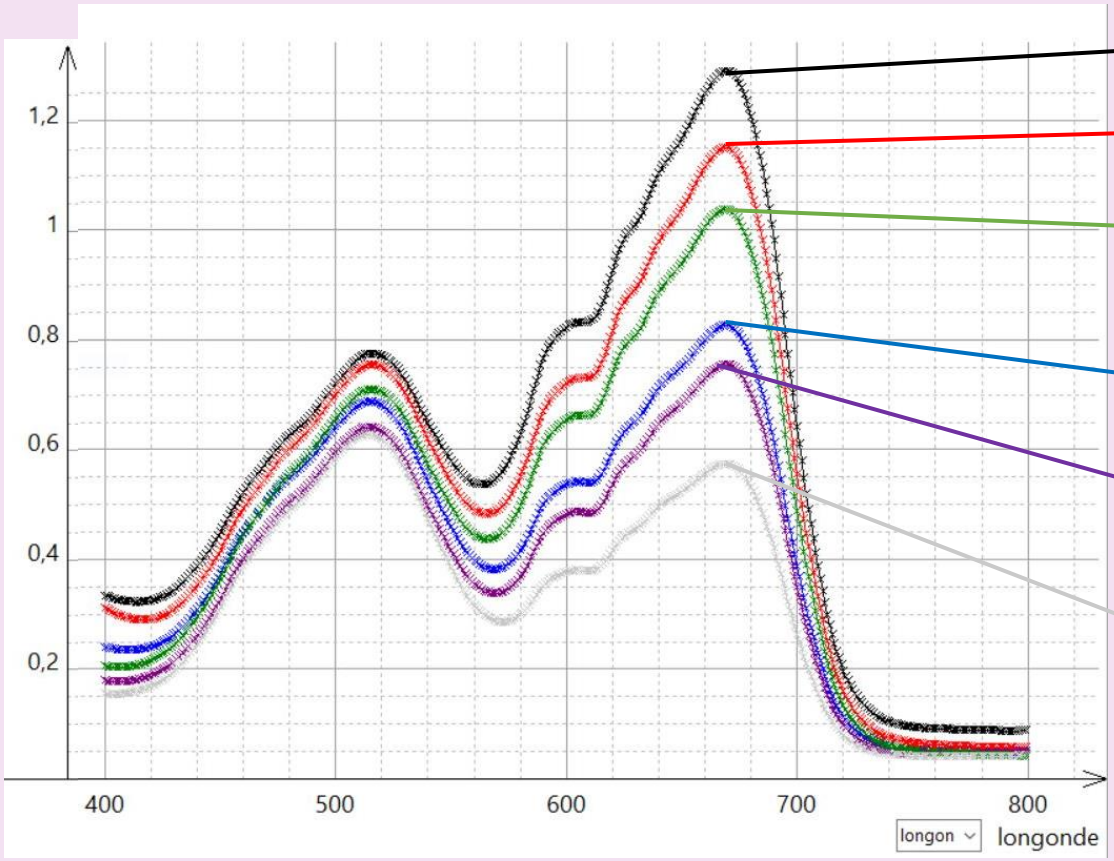
cuve thermostatée

circulation d'eau

spectrophotomètre thermostaté

Courbe A = f(λ) pour différentes températures pour une solution eau:acétone 22:78

- Programme Python pour importer les données dans Régressi



43 °C

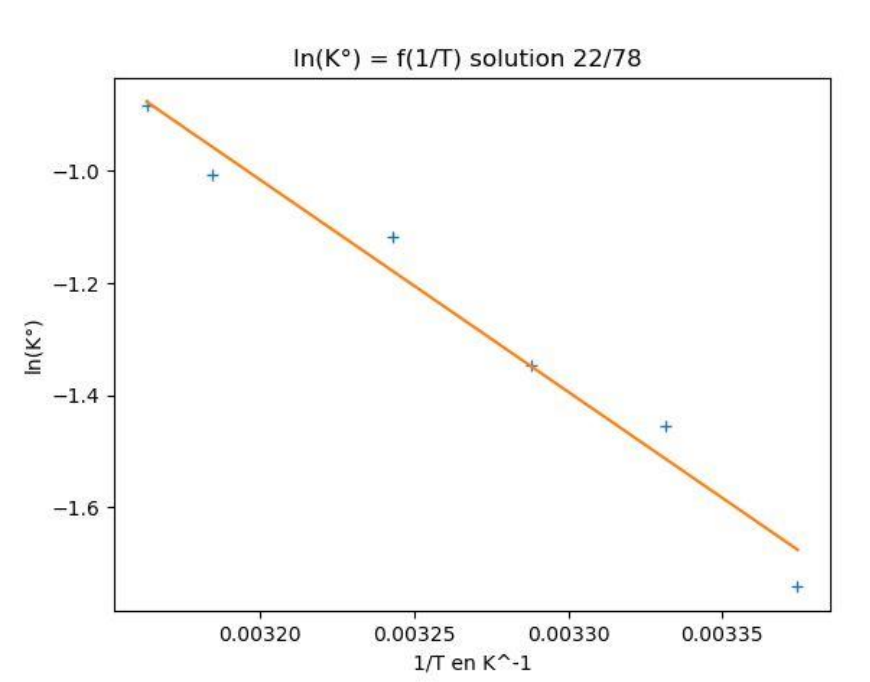
40,9 °C

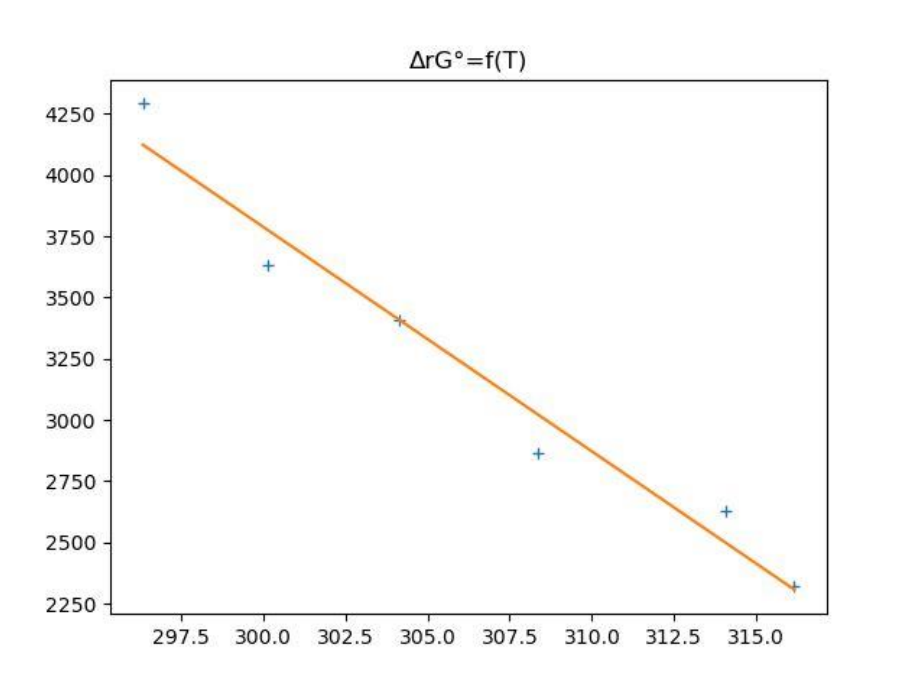
35,2 °C

31 °C

27 °C

23,2 °C





Courbe $\ln(K) = f(1/T)$

Courbe $\Delta rG^0 = f(T)$

$\Delta rG^0 = \Delta rH^0 - T\,\Delta rS^0$

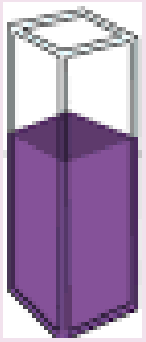
$$Co(H_2O)_6^{2+} + 4Cl^- \rightleftarrows Co(Cl)_4^{2-} + 6H_2O$$

Modélisation :

$\Delta rH^0 = 31.3 \pm 4$ kJ.mol^{-1}

$\Delta rS^0 = 91.7 \pm 15$ J.mol^{-1}.K^{-1}

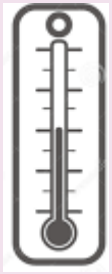
Solution de cobalt



Absorbance

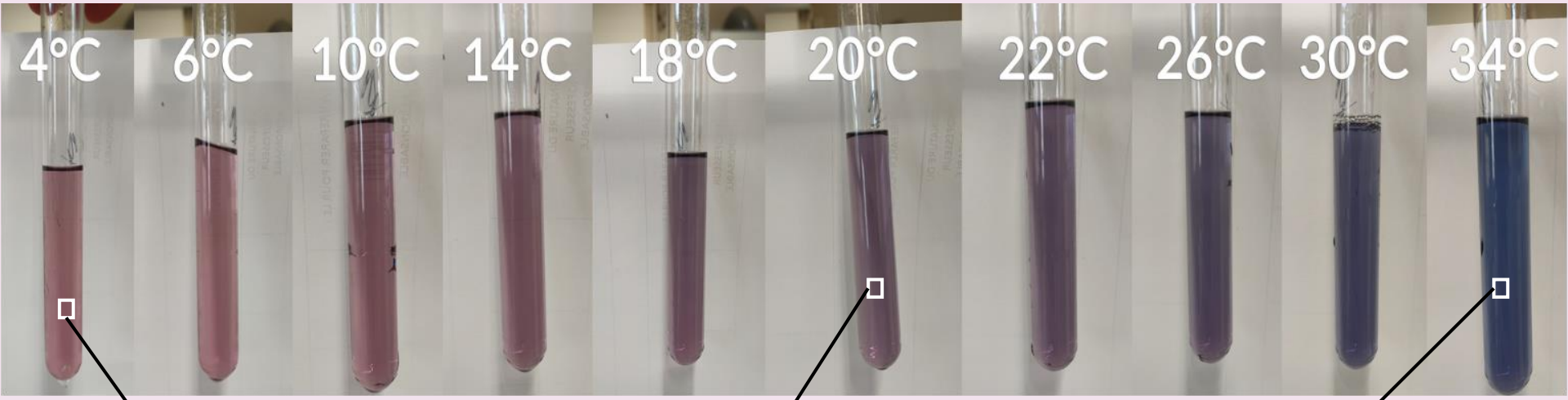
Composantes RGB

Température approximative



4. Mise en œuvre

Création d'un thermomètre au complexe de cobalt

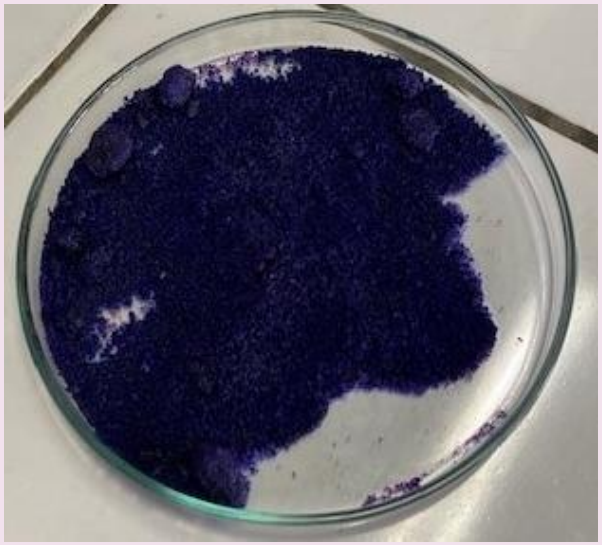


R:148 G:106 B:110

R:95 G:75 B:86

R:43 G:54 B:82

Déterminer le taux d'humidité avec le complexe de cobalt

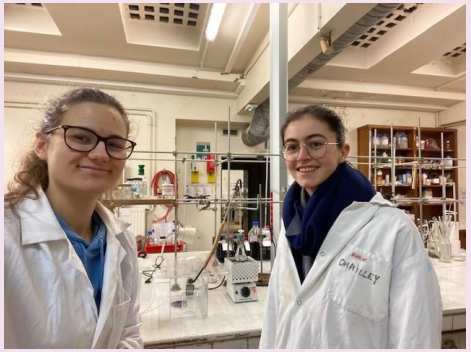


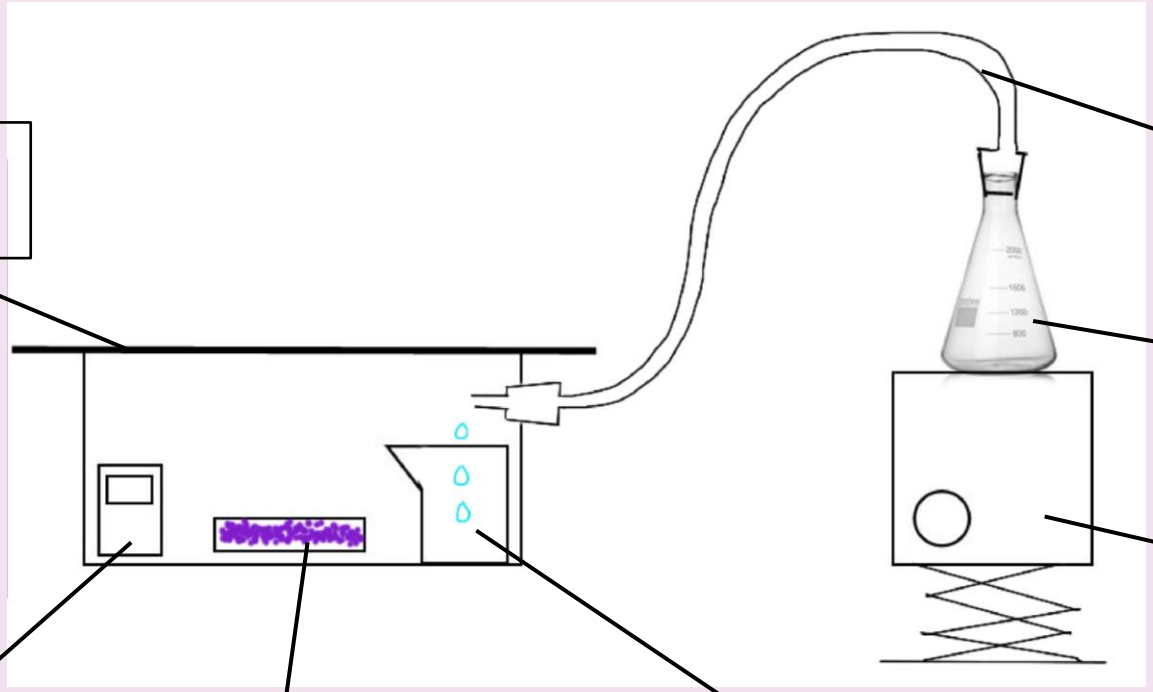
Complexe de cobalt tout juste sorti de l'étuve (T = 14,3°C)

Complexe de cobalt laissé en TP pendant 1 semaine à humidité ambiante

Déterminer le taux d'humidité avec le complexe de cobalt

Tentative 1 : Augmenter l'humidité du milieu





Cuve en plastique

Tuyau

Erlenmeyer contenant de l'eau

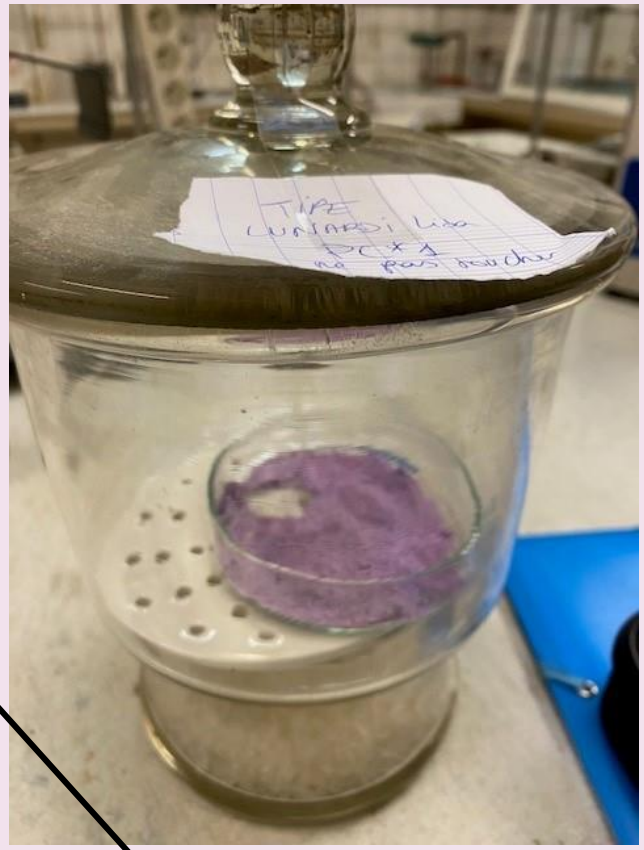
Chauffe-ballon

Hygromètre

Complexe de cobalt

Bécher de récupération

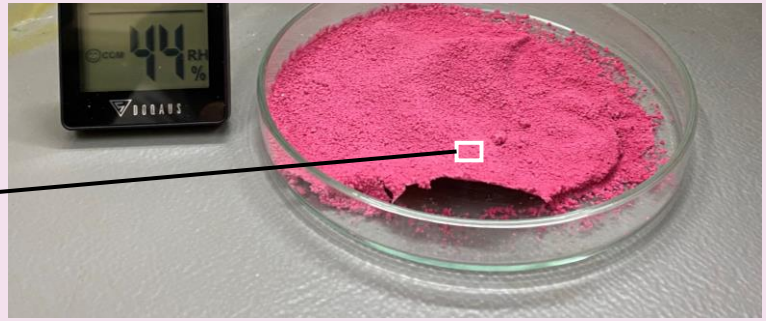
Tentative 2 : Diminuer l'humidité du milieu



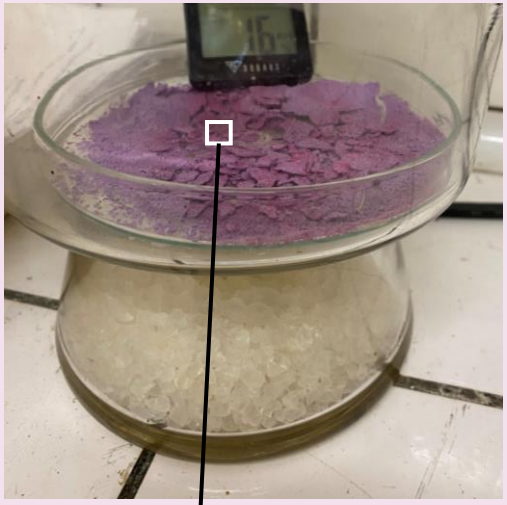
Dessiccateur

Silicagel (agent desséchant)

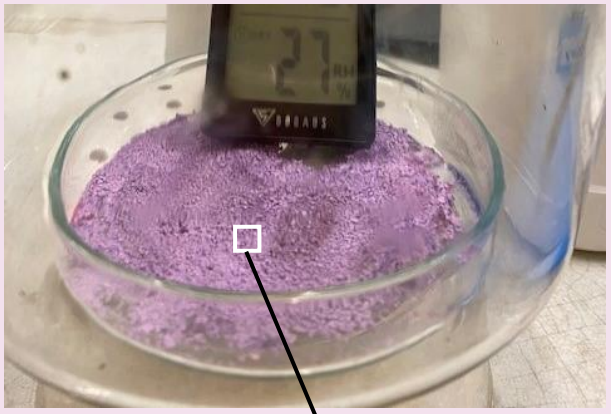
Complexe de cobalt



R:255 G:134 B:175



R:119 G:84 B:104



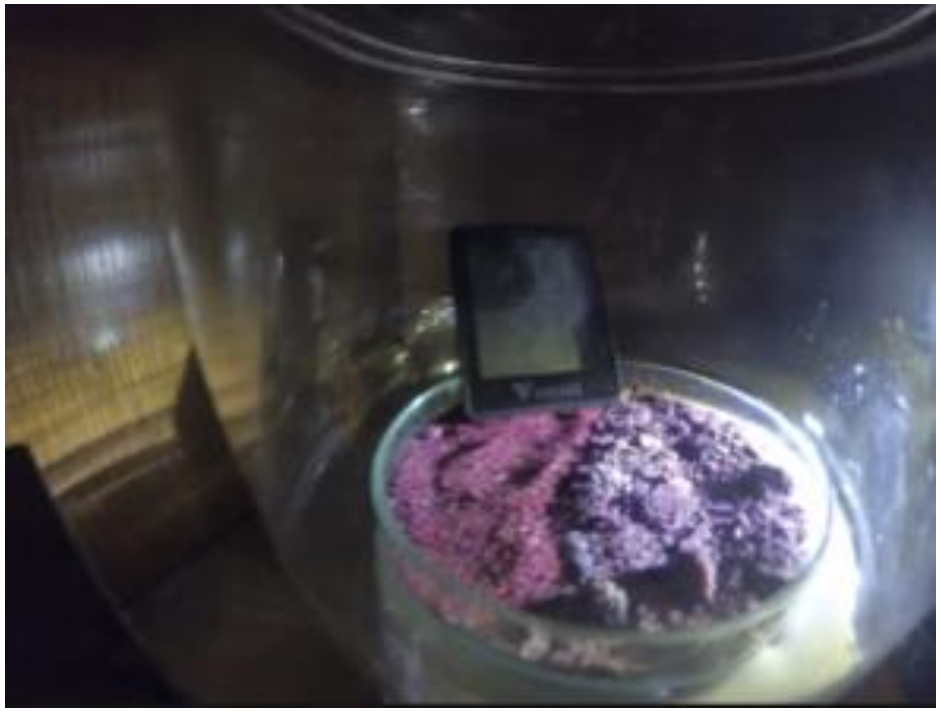
R:200 G:154 B:191

R:227 G:157 B:185

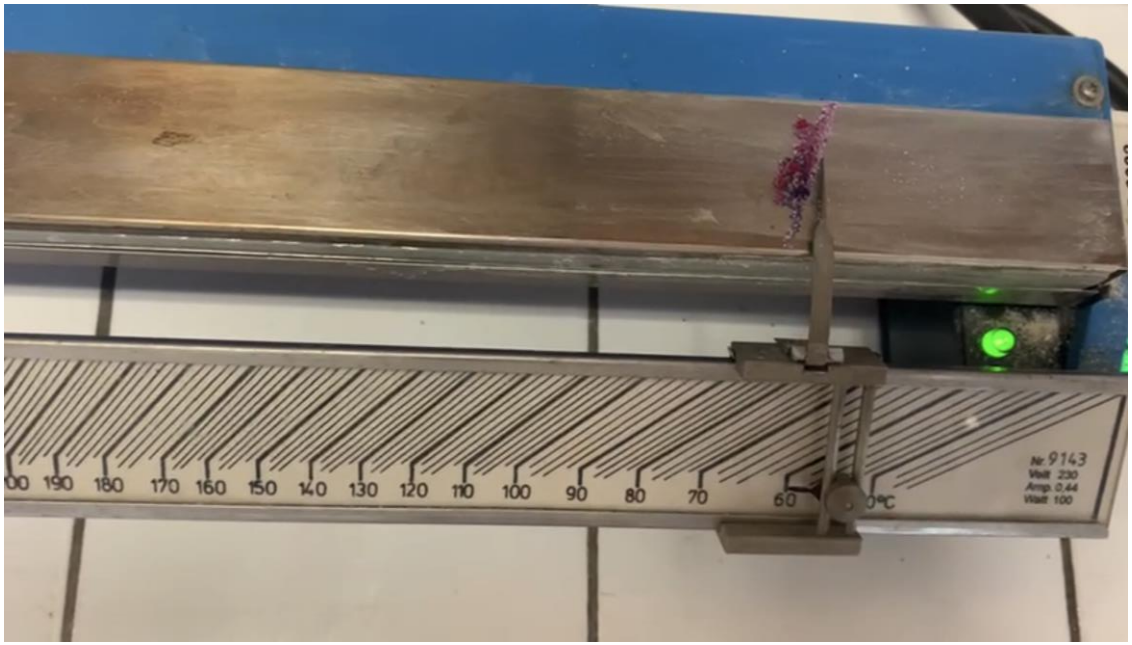
Merci de votre écoute !

Caméra GoPro


TAMPON




MARS 2022
260 250 240 230 220 210 200 190 180 170 160 150 140 130 120 110 100 90 80 70
50°C


190 180 170 160 150 140 130 120 110 100 90 80 70 60
°C
Nr 9143
Volt 230
Amp. 0,44
Watt 100

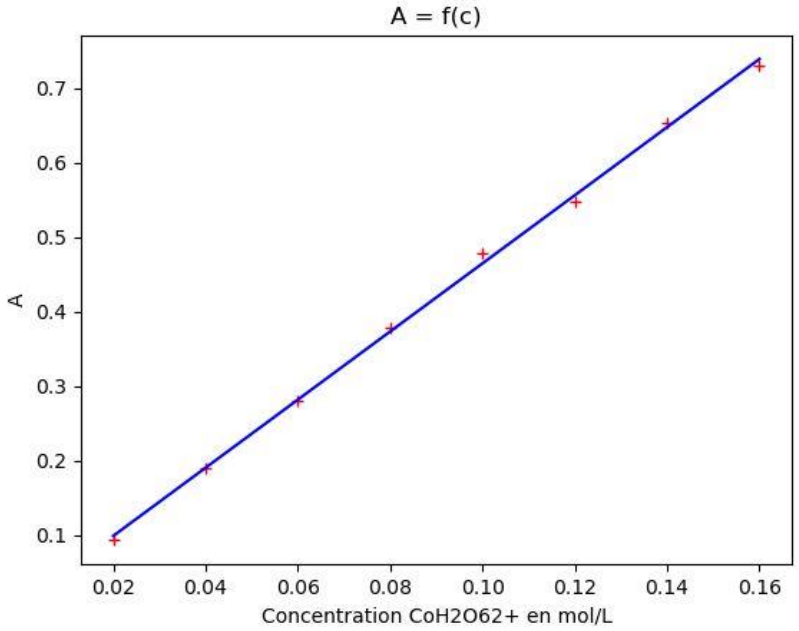
Code extraction des absorbance du SAFAS-UV

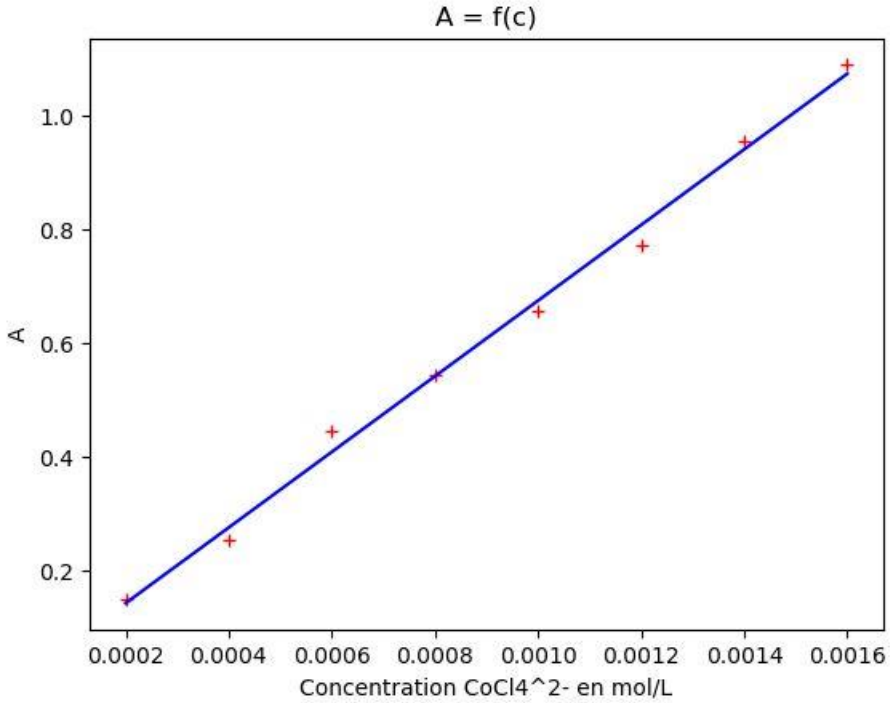
```
import os
os.chdir("C:/Users/lisal/Desktop/CPGE/PCstar/TIPE/séance du 10 10")
os.getcwd()
file1 = "3.8 deg.txt"
file2 = "12.2 deg.txt"
file3 = "13.9 deg.txt"
file4 = "14.8 deg.txt"
file5 = "20.4 deg.txt"
file6 = "27.4 deg.txt"
file7 = "30.2 deg.txt"
file8 = "35.2 deg.txt"
file9 = "38.3 deg.txt"
file10 = "40.9.txt"
```

```
15  def dataSorting (fileName) :
16      file = open(fileName)
17      data = file.readlines()
18      nbData = len(data)
19      start = 17
20      nm = []
21      ABS = []
22      data1 = open('data1.txt', 'w')
23      data2 = open('data2.txt', 'w')
24      for i in range (start, nbData):
25          line = data[i]
26          nm.append(line[0:3])
27          ABS.append(line[4:len(line)-1])
28          data1.write(line[0:3])
29          data1.write('\n')
30          data2.write(line[4:len(line)-1])
31          data2.write('\n')
32
33      file.close()
34      data1.close()
35      data2.close()
36      return(nm, ABS)
```

Code Python thermochimie

```python
import matplotlib.pyplot as plt
import numpy as np
import numpy.random as rd

Aa= np.array([0.09309,0.1895,0.2805,0.3795,0.4792,0.5475,0.6542,0.7306])
Ab= np.array([0.151391,0.2554852,0.4447801,0.5435046,0.6567771,0.7730068,0.9555482,1.089248])


l = 1 # longueur de la cuve en cm
Ca = np.array([2E-2,4E-2,6E-2,8E-2,1E-1,1.2E-1,1.4E-1,1.6E-1]) # Concentrations en CoH2O62+ en mol/L
Cb = np.array([2E-4,4E-4,6E-4,8E-4,1E-3,1.2E-3,1.4E-3,1.6E-3])# Concentrations en CoCl4^2- en mol/L

pa = np.polyfit(Ca,Aa,1)
pb = np.polyfit(Cb,Ab,1)

plt.figure()
plt.plot(Ca,Aa,'r+')
plt.xlabel('Concentration CoH2O62+ en mol/L')
plt.ylabel('A')
plt.title('A = f(c)')
plt.plot(Ca, np.polyval(pa,Ca), 'b')
plt.show()
print("ε = ", pa[0]," L.mol^-1.cm^-1")

plt.figure()
plt.plot(Cb,Ab,'r+')
plt.xlabel('Concentration CoCl4^2- en mol/L')
plt.ylabel('A')
plt.title('A = f(c)')
plt.plot(Cb, np.polyval(pb,Cb), 'b')
plt.show()
print("ε = ", pb[0]," L.mol^-1.cm^-1")
```


A = f(c)
A
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Concentration CoH2O62+ en mol/L


A = f(c)
A
1.0
0.8
0.6
0.4
0.2
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012
0.0014
0.0016
Concentration CoCl4^2- en mol/L

```
A =[0.5732,0.7542,0.8356,1.037,1.15,1.289] #absorbance en fonction des températures

V=100e-3 #volume en L
mi=4.76 #masse de solide Co(H20)6^2+, Cl2^- introduite en g
Ve = 22 # mL
Me = 18 # masse molaire en g.mol^-1
M = 238 # masse molaire en g.mol^-1
Ci = mi/(M*V) #concentration de complexe de cobalt
Ce = Ve/(Me*V) #concentration en eau

eb=pb[0] #coefficient d'absorption molaire en unités SI, à 668 nm de B

xeq=A/eb # avancement final

K=xeq/((Ci - xeq)*(2*Ci - 4*xeq)**4)  # tableau des K° (LAM) à chaque température

logK=np.log10(K)

print("K=", K)
```

$$K^\circ(T) = \frac{\left[\mathrm{Co(Cl)_4^{2-}}\right]_{éq}}{\left[\mathrm{Co(H_2O)_6^{2+}}\right]_{éq}\left[\mathrm{Cl^-}\right]_{éq}^4}$$

$[i]$ en $\mathrm{mol \cdot L^{-1}}$	$\mathrm{Co(H_2O)_6^{2+}}$	+	$\mathrm{4\,Cl^-}$	⇆	$\mathrm{Co(Cl)_4^{2-}}$	+	$\mathrm{6H_2O}$
État initial	0,1		0,2		0		–
État d'équilibre	$0,1 - x_{éq}$		$0,2 - 4\,x_{éq}$		$x_{éq}$		–

```
#Thermochimie

lnK=np.log(K)
T = np.array([296.35,300.15,304.15,308.35,314.05, 316.15])
X = np.array([1/(296.35),1/(300.15),1/(304.15),1/(308.35),1/(314.05), 1/316.15]) #valeurs des 1/T


ΔrG0 = -8.314*T*lnK
coeff = np.polyfit(X, lnK, 1)
a,b = coeff[0], coeff[1]

plt.figure()
plt.plot(X, lnK, '+')
plt.plot(X, a*X+b)
plt.xlabel('1/T en K^-1')
plt.ylabel('ln(K°)')
plt.title('ln(K°) = f(1/T) solution 22/78 ')
plt.show()

coeff2 = np.polyfit(T, ΔrG0, 1)
c,d = coeff2[0], coeff2[1]

plt.figure()
plt.plot(T, ΔrG0, '+')
plt.plot(T, c*(T)+d)
plt.title("ΔrG°=f(T)")
plt.show()

ΔrH0 = d/1000
ΔrS0 = -c

print("ΔrH° = ", ΔrH0 ,"kJ.mol^-1")
print("ΔrS° =", ΔrS0, "J.mol^-1.K^-1")
```

$\Delta G = \Delta G^o + RT \ln Q$ A l'équilibre: $\Delta_r G^\circ = -R.T \ln(K\ )$

$\Delta_r G^\circ = \Delta_r H^\circ - T\Delta_r S^\circ$

```python
#incertitudes Monte-Carlo


uV=0.1e-3/np.sqrt(3) #incertitude sur le volume
umi=0.01/np.sqrt(3) #incertitude sur la masse

N=5000
X=np.array([1/(10+273),1/(18+273),1/(22+273),1/(27+273),1/(30+273)])
T=np.array([273+10,273+18,273+22,273+27,273+30])
u_T=5
asim=[]
bsim=[]
for i in range(N):
    Ksim=[]
    Tsim=[]
    Xsim=[]
    for t in range(5):
        xeqt=xeq[t]
        mi_sim=rd.uniform(mi-umi,mi+umi)
        V_sim=rd.uniform(V-uV,V+uV)
        Ci_sim=mi_sim/(M*V_sim)
        Tsim.append(rd.uniform(T[t]-u_T,T[t]+u_T))
        Xsim.append(1/Tsim[t])
        Ksim.append(xeqt/((Ci_sim - xeqt)*(2*Ci_sim - 4*xeqt)**4))
    lnKsim=np.log(Ksim)
    asim.append(np.polyfit(Xsim,lnKsim,1)[0])
    bsim.append(np.polyfit(Xsim,lnKsim,1)[1])

drHsim=-8.314*np.array(asim)/1000
drSsim=8.314*np.array(bsim)

drHmoy=np.mean(drHsim)
udrH=np.std(drHsim,ddof=1)
drSmoy=np.mean(drSsim)
udrS=np.std(drSsim,ddof=1)

print("ΔrH°sim = ", drHmoy,"±", udrH, "kJ.mol^-1")
```

```
#Calcul du Z-score

zH=abs(drH-drHmoy)/udrH
zS=abs(drS-drSmoy)/udrS

print("zH=", zH, "zS=", zS)
## Incertitudes et validation du modèle

ma = 4.76 #g
mb = 0.48 #g
u_m = 0.01 #g incertitude de la balance
V = 100 #mL
u_V = 0.100 #mL incertitude de la fiole jaugée
M = 238 #g/mol
V_f = 10 #mL
u_V_f = 0.025 #mL incertitude de la fiole jaugée
V_p = np.array([1,2,3,4,5,6,7,8]) #mL
u_V_p = np.array([0.015,0.010,0.025,0.020,0.015,0.030,0.025,0.040])#mL incertitude des pipettes jaugées

# incertitudes des solutions mères

C_0a = ma/(V*M)*10**3 #mol/L
C_0b = mb/(V*M)*10**3 #mol/L
u_C_0a = C_0a*np.sqrt((u_m/ma)**2+(u_V/V)**2) #mol/L
u_C_0b = C_0b*np.sqrt((u_m/mb)**2+(u_V/V)**2) #mol/L

# incertitudes des solutions filles
C_a= C_0a*V_p/V_f
C_b= C_0b*V_p/V_f
u_Ca = (C_0a*V_p/V_f)*np.sqrt((u_C_0a/C_0a)**2+(u_V_p/V_p)**2+(u_V_f/V_f)**2) #mol/L
u_Cb = (C_0b*V_p/V_f)*np.sqrt((u_C_0b/C_0b)**2+(u_V_p/V_p)**2+(u_V_f/V_f)**2) #mol/L

Aa= np.array([0.09309,0.1895,0.2805,0.3795,0.4792,0.5475,0.6542,0.7306])
Ab= np.array([0.151391,0.2554852,0.4447801,0.5435046,0.6567771,0.7730068,0.9555482,1.089248])

u_A = 0.000001
```

```
aasim=[]
absim=[]
N=5000
for i in range(N):
    Asim=[]
    for t in range(5):
        Casim=rd.uniform(C_a-u_Ca,C_a+u_Ca)
        Cbsim=rd.uniform(C_b-u_Cb,C_b+u_Cb)
        Aasim=rd.uniform(Aa-u_A,Aa+u_A)
        Absim=rd.uniform(Ab-u_A,Ab+u_A)
    aasim.append(np.polyfit(Casim,Aasim,1)[0])
    absim.append(np.polyfit(Cbsim,Absim,1)[0])

ea = aasim
eb = absim

eamoy=np.mean(ea)
u_ea=np.std(ea,ddof=1)
ebmoy=np.mean(eb)
u_eb=np.std(eb,ddof=1)

print("ea = ", eamoy,"±", u_ea, "L.mol^-1.cm^-1")
print("eb =", ebmoy*10, "±", u_eb, "L.mol^-1.cm^-1")
```

```
>>> (executing cell "" (line 1 of "python.py"))
ε =  4.575458333333334  L.mol^-1.cm^-1
ε =  664.4801547619048  L.mol^-1.cm^-1
K= [0.17517892 0.23336534 0.25999782 0.32715488 0.36563559 0.41377464]
ΔrH° =  31.317233151030855 kJ.mol^-1
ΔrS° = 91.75894221121713 J.mol^-1.K^-1
ΔrH°sim =  24.459673683946868 ± 4.606505146251941 kJ.mol^-1
zH= 1.5167370791917791 zS= 1.2873063485743634

>>> (executing cell "Incertitudes etv..." (line 141 of "python.py"))
ea =  4.57520809193677 ±  0.13689127360519675 L.mol^-1.cm^-1
eb = 658.693177663025 ± 0.6750882972325659 L.mol^-1.cm^-1
```

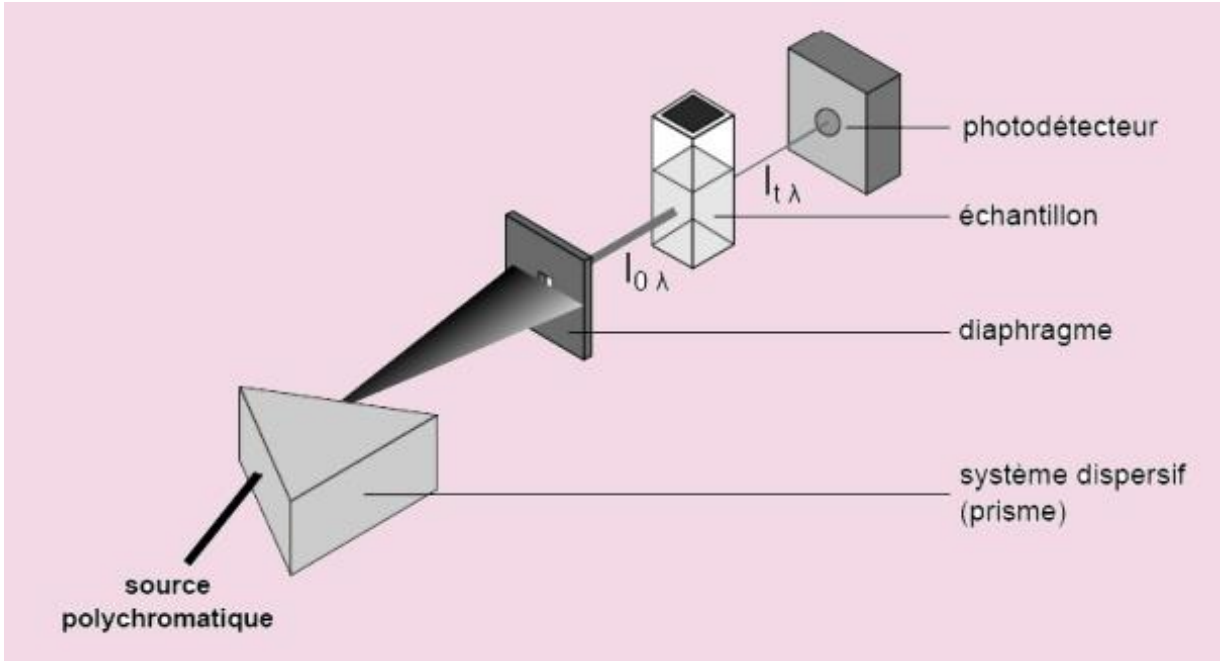
Code Python programme thermomètre

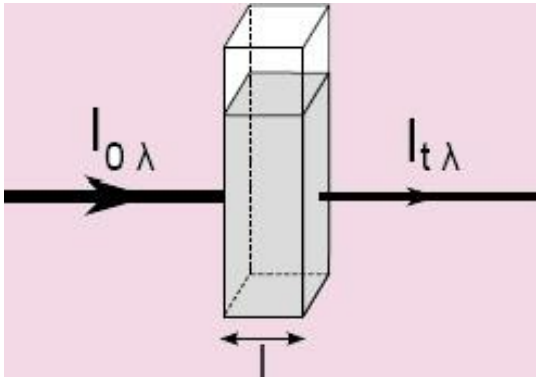
```
def temperature(R,G,B):
    eps = 15
    if R>135-eps and R < 135+eps and G>97-eps and G<97+eps and B> 96-eps and B<96+eps:
        return 10.0

    if R>157-eps and R < 157+eps and G>91-eps and G<91+eps and B> 119-eps and B<119+eps:
        return 14.5

    if R>91-eps and R < 91+eps and G>55-eps and G<55+eps and B> 67-eps and B<67+eps:
        return 15.3

    if R>87-eps and R < 87+eps and G>56-eps and G<56+eps and B> 71-eps and B<71+eps:
        return 16.0

    if R>67-eps and R < 67+eps and G>41-eps and G<41+eps and B> 89-eps and B<89+eps:
        return 18.6

    if R>102-eps and R < 102+eps and G>102-eps and G<102+eps and B> 102-eps and B<102+eps:
        return 19.00

    if R>74-eps and R < 74+eps and G>56-eps and G<56+eps and B> 98-eps and B<98+eps:
        return 20.0

    if R>78-eps and R < 78+eps and G>67-eps and G<67+eps and B> 143-eps and B<143+eps:
        return 21.0

    if R>91-eps and R < 91+eps and G>80-eps and G<80+eps and B> 97-eps and B<97+eps:
        return 25.0

    if R>44-eps and R < 44+eps and G>47-eps and G<47+eps and B> 66-eps and B<66+eps:
        return 30.0

    if R>33-eps and R < 33+eps and G>46-eps and G<46+eps and B> 78-eps and B<78+eps:
        return 35.0

    else:
        return "Not yet defined"
```

Correspondances longueurs d'onde et RGB

```
396 775,0.34375,0,0,
397 776,0.33499999999999996,0,0,
398 777,0.32625,0,0,
399 778,0.3175,0,0,
400 779,0.30874999999999997,0,0,
401 780,0.3,0,0,
402 781,0,0,0
403 ]
404
405
406 def dataSorting():
407     longonde,R,G,B = [],[],[],[]
408     i = 0
409     while i != len(corres):
410         longonde.append(corres[i])
411         i += 1
412         R.append(corres[i])
413         i+= 1
414         G.append(corres[i])
415         i += 1
416         B.append(corres[i])
417         i +=1
418     return
419
420 def Couleur(int(long)):
421     i = 0
422     while long != longonde[i]:
423         i += 1
424     return ("RGB =", (255-R[i],255-G[i],255-B[i]))
```

Fonctionnement d'un spectrophotomètre UV-Visible





https://culturesciences.chimie.ens.fr/thematiques/chimie-analytique/spectroscopies/introduction-a-la-spectroscopie-uv-visible

Diagramme orbitalaire de M-L_6