Υπολογισμός σταθεράς του Planck

και έργου εξαγωγής στο φωτοηλεκτρικό φαινόμενο

Πραγματοποιώντας το πείραμα της Γ Λυκείου μετρήσαμε την τάση αποκοπής V_0 σε συνάρτηση με το μήκος κύματος λ της προσπίπτουσας ακτινοβολίας για διάφορα "χρώματα".

Αυτά ήταν:

```
Μπλέ               : 470nm
Κίτρινο            : 510nm
Πράσινο            : 520nm
Κίτρινο-Πορτοκαλί  : 530nm
Πορτοκαλί          : 580nm
Κόκκινο            : 630nm
```

Στο παρόν **jupyter-lab** έγγραφο θα χρησιμοποιήσουμε την **python** για να μετατρέψουμε τις μετρήσεις μας στην κατάλληλη μορφή ώστε να υπολογίσουμε την σταθερά του Planck h και το έργο εξαγωγής ϕ του μετάλλου του φωτοκυττάρου.

Πρώτα θα εισάγουμε τα πακέτα που θα χρησιμοποιήσουμε: **numpy** για πίνακες και αριθμητική, **sklearn** για τα στατιστικά μοντέλλα που θα χρειαστούμε για να σχεδιάσουμε την ευθεία που ταιριάζει με βέλτιστο τρόπο στις πειραματικές μετρήσεις, **matplotlib** το εξαιρετικό πακέτο για διαγράμματα και τέλος το sympy.interactive.printing για να χρησιμοποιούμε και το $\LaTeX$ γιατί χωρίς αυτό η ζωή είναι ανούσια...

```
# Import the required modules
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
from sympy.interactive import printing
printing.init_printing(use_latex=True)

# This makes the plots appear inside the notebook
%matplotlib inline
```

Καθορίζουμε τους πίνακες με τα δεδομένα του πειράματος:

```
lamda = [470.0, 510.0, 520.0, 530.0, 580.0, 630.0] # nm
V_0 = [1.1, 0.56, 0.70, 0.47, 0.41, 0.31] # Volt
f = np.zeros(6) # frequency in Hz
```

Ορίζουμε τις τιμές της ταχύτητας του φωτός στο κενό c και του φορτίου του ηλεκτρονίου e που θα χρειαστούμε στους υπολογισμούς παρακάτω.

```
c = 3.0 * 10**8
e = 1.6 * 10**-19
```

Υπολογίζουμε τις συχνότητες f από τα μήκη κύματος λ με τον θεμελιώδη νόμο της κυματικής: $f = \frac{c}{\lambda}$

```
# Calculate the frequencies from lambdas
for i in range(len(lamda)):
    f[i] = c/(lamda[i]*10**-9)
print(f)
```

```
[6.38297872e+14 5.88235294e+14 5.76923077e+14 5.66037736e+14
 5.17241379e+14 4.76190476e+14]
```

Γράφουμε τα δεδομένα σε ένα text αρχείο απλά για να υπάρχουν...

```
myfile = open("PhotoElectricExp.txt", "w")
#f.write("Now the file has more content!")
```

```
for k in range(len(f)):
    myfile.write(str(V_0[k])+" "+str(f[k])+"\n")
print('Οι πίνακες γράφτηκαν στο αρχείο')
```

```
Οι πίνακες γράφτηκαν στο αρχείο
```

```
myfile.close()
```

Για να αναλύσουμε στατιστικά τα δεδομένα, μετατρέπουμε τους πίνακες που είχαμε δημιουργήσει σε πίνακες του numpy:

```
f = np.array(f).reshape((-1, 1))
V_0 = np.array(V_0)
```

Ο πίνακας του x (εδώ των συχνοτήτων f) πρέπει να είναι πίνακας στήλη, γι' αυτό και τον αλλάζουμε με το reshape((-1,1)). Πράγματι αν τυπώσουμε τους πίνακες θα δούμε:

```
f, V_0
```

```
(array([[6.38297872e+14],
        [5.88235294e+14],
        [5.76923077e+14],
        [5.66037736e+14],
        [5.17241379e+14],
        [4.76190476e+14]]),
 array([1.1 , 0.56, 0.7 , 0.47, 0.41, 0.31]))
```

Επιλέγουμε στατιστικό μοντέλο το LinearRegression (γραμμική συσχέτιση)

```
model = LinearRegression()
```

Και λέμε στο μοντέλο να προσαρμοστεί στα δεδομένα!

```
model.fit(f, V_0)
```

Οι δύο προηγούμενες εντολές θα μπορούσαν να γραφούν και ως εξής σε μία εντολή:

model = LinearRegression().fit(x, y)

Τώρα απλώς εξάγουμε τα δεδομένα από το μοντέλο μας!

```
r_sq = model.score(f, V_0)
print(f"Συντελεστής αξιοπιστίας: {r_sq:.2f}")
```

```
Συντελεστής αξιοπιστίας: 0.79
```

```
V_1 = model.intercept_
print(f"Τέμνει τον y άξονα: {V_1:.2f}")
```

```
Τέμνει τον y άξονα: -1.89
```

```
slope = model.coef_[0]
print(f"Κλίση: {slope:.2e}")
```

```
Κλίση: 4.43e-15
```

Το μοντέλο προβλέπει για την τάση αποκοπής V_0 ως συνάρτηση της συχνότητας f της μορφής (S.I.):

$$V_0 = 4.43 \cdot 10^{-15} f - 1.89$$

Γνωρίζουμε από την θεωρία ότι η τάση αποκοπής υπολογίζεται από τον τύπο:

$$V_0 = \frac{h}{e} f + \frac{\phi}{e}$$

επομένως η κλίση της γραφικής παράστασης είναι το πηλίκο $\frac{h}{e}$ και η ευθεία τέμνει τον άξονα y (της τάσης αποκοπής V_0) στο σημείο $\frac{\phi}{e}$

Επομένως η σταθερά του Planck υπολογίζεται ως:

$$h = (\text{κλίση}) \cdot e$$

```
# Calculate the Planck constant h
h = slope*e
```

```
print(f"Σταθερά του Planck h = {h:.2e} Js")
```

```
Σταθερά του Planck h = 7.09e-34 Js
```

Και το έργο εξαγωγής είναι απλά η απόλυτη τιμή του σταθερού όρου της συνάρτησης που προβλέπει το μοντέλο σε ηλεκτρονιοβόλτ (eV)

```
# Calculate the ϕ
print(f"Έργο εξαγωγής ϕ = {-V_1:.2f} eV")
```

```
Έργο εξαγωγής ϕ = 1.89 eV
```

Τώρα απομένει να κάνουμε και τις γραφικές μας παραστάσεις. Αρχικά πρέπει να ορίσουμε μερικές τιμές στον x άξονα με την linspace (linear space), 20 τιμές από 0 έως $7 \cdot 10^{14}$ (Hz), και να χρησιμοποιήσουμε την συνάρτηση του μοντέλου για τις τιμές του y (εδώ το V_0)

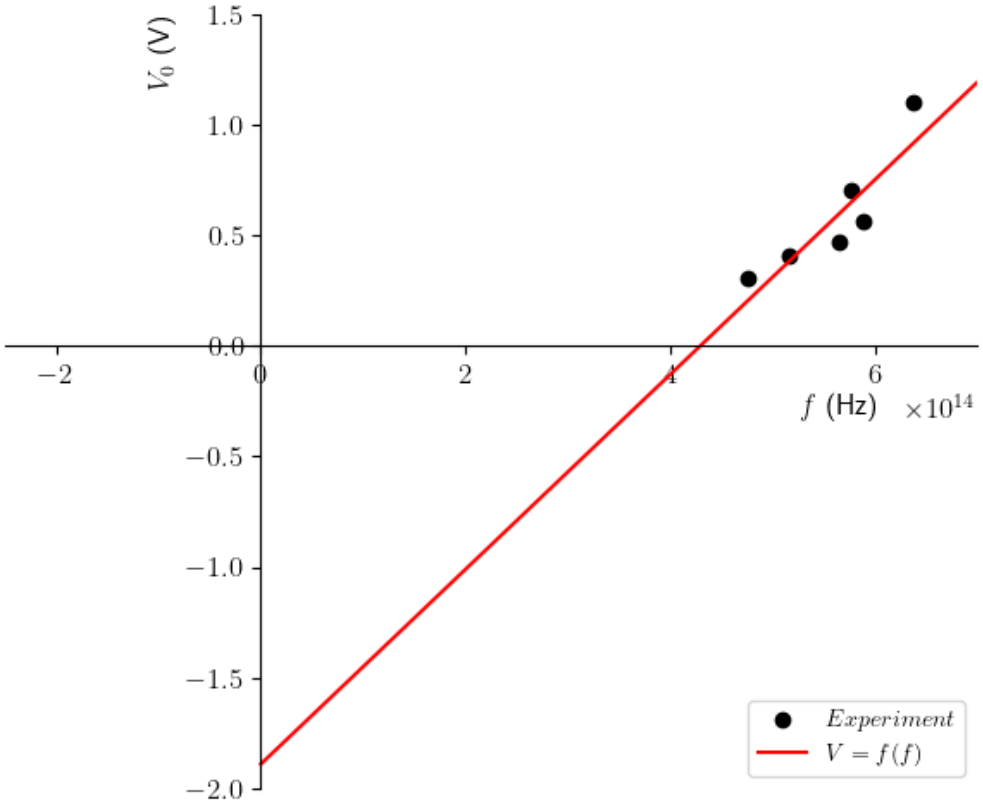
```
x = np.linspace(0, 7*10**14, 20)
y = np.zeros(20)
# give values for the line plot
for k in range(len(x)):
    y[k] = 4.4*10**-15*x[k] - 1.89
print(y)
```

```
[-1.89       -1.72789474 -1.56578947 -1.40368421 -1.24157895 -1.07947368
 -0.91736842 -0.75526316 -0.59315789 -0.43105263 -0.26894737 -0.10684211
  0.05526316  0.21736842  0.37947368  0.54157895  0.70368421  0.86578947
  1.02789474  1.19      ]
```

Και τέλος αφήνουμε το matplotlib να κάνει τα μαγικά του!

(Εντάξει το καθοδηγούμε και λίγο...)

```
# Plot the solution please!
plt.rc('text', usetex=True)
plt.rcParams['figure.figsize'] = (6,5)
plt.rcParams['font.size'] = 12
plt.rcParams['legend.fontsize'] = 10
plt.rcParams['legend.loc'] = 'lower right'
fig, ax1 = plt.subplots()

#ax1.tick_params(axis='y', labelcolor=color)
ax1.set_xlim(-2.5*10**14, 7*10**14)
ax1.set_ylim(-2, 1.5)

color = 'black'
ax1.set_xlabel(r'$f$ (Hz)', loc='right')
ax1.xaxis.set_label_coords(.9, .51) # place x axis label
ax1.set_ylabel(r'$V_0$ (V)', color=color, loc='top')
# Γραφική παράσταση των πειραματικών σημείων
ax1.scatter(f, V_0, color=color, label=r'$Experiment$')

# Καθορίζουμε τους άξονες να τέμνονται στο (0,0)
ax1.spines['left'].set_position('zero')
ax1.spines['bottom'].set_position('zero')
ax1.spines['right'].set_color('none')
ax1.spines['top'].set_color('none')

# Τώρα θα ζωγραφίσουμε και την ευθεία ελαχίστων τετραγώνων του μοντέλου
# σε κόκκινο χρώμα
color = 'red'
ax1.plot(x, y, color, label=r'$V=f(f)$')

# Πού είναι η επεξήγηση?
ax1.legend()

# Για να μην κόβει την εικόνα γύρω-γύρω
fig.tight_layout() # otherwise the right y-label is slightly clipped

# Σώζουμε το δημιούργημά μας σε δύο εικόνες pdf και png ή ότι άλλο θέλουμε
plt.savefig('/home/jorge/PyCode/figures/V_vs_f.pdf')
plt.savefig('/home/jorge/PyCode/figures/V_vs_f.png')
```


V_0 (V)
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2
0
2
4
6
f (Hz)
$\times 10^{14}$
Experiment
$V = f(f)$