Astrochemistry Tutorial 2: Maxwell A. Fine 14880725

In [2]:
```
# imports
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.optimize import curve_fit

data_file = "Methanol_fluxes.rtf"
```

```
/home/afinemax/anaconda3/lib/python3.9/site-packages/scipy/__init__.py:146: Us
erWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version o
f SciPy (detected version 1.26.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:
```
# read in the data
data = []
with open(data_file) as infile:
    counter = 0
    for line in infile:

        # Skip lines that start with '{', '}', '\'
        if line[0] == '{' or line[0]=='}' or line[0]== '\\':
            continue

        counter += 1
        if counter == 1 :
            unit = line
            continue

        if counter == 2 :
            unit = line
            continue



        # Split the line into columns using spaces as the delimiter
        columns = line.strip().split()

        # Append the columns to the data list
        data.append(columns)

        # Remove '\\' from each string and convert to float
cleaned_data = [[float(value.rstrip('\\')) for value in row] for row in data]

# Convert the list of lists to a NumPy array
print(unit)
data = np.array(cleaned_data, dtype=float)
```

```
  (GHz)     (K)    (s-1)        (K km s-1)  \
```

1. frequencies (ν in GHz)
2. energies of the upper level (Eu in K)
3. Einstein-A coefficients (Aul in s −1)
4. degeneracies (gu)

5. integrated intensities (W = ∫ Tmb dv in K km s −1)
6. 1σ error associated to W

Q1:

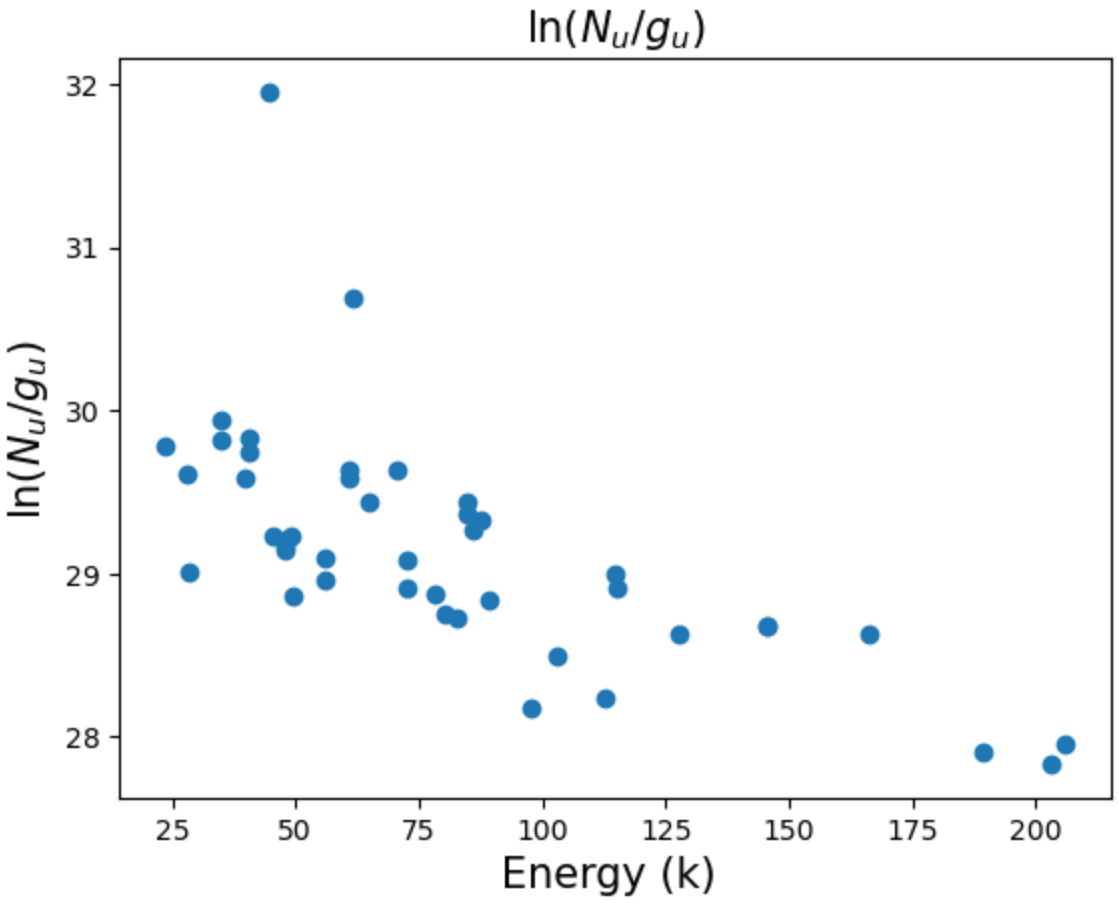
Write a small program (per example in Python), to plot a diagram of $ln(Nu/gu)$ as a function of Eu

$$\frac{N_\mu}{g_\mu} = C\frac{(\nu^2 W)}{A_{ul} g_u}$$

c = 1942.75

In [4]:
```
freq = data[:, 0]
eu = data[:, 1]
aul = data[:, 2]
gu = data[:, 3]
w = data[:, 4]
w_sigma = data[:,5]

c = 19942.75

ratio =  c * freq**2 *w  / (aul * gu)

sigma_ratio = np.log(w_sigma/w  * ratio)
sigma_ratio = w_sigma/w
ln_ratio = np.log(ratio)


plt.title('$\ln(N_u/g_u)$', size=15)
plt.scatter(eu, ln_ratio)
plt.xlabel('Energy (k) ', size=15)
plt.ylabel('$\ln(N_u/g_u)$', size=15 )
```

Out[4]:
```
Text(0, 0.5, '$\\ln(N_u/g_u)$')
```



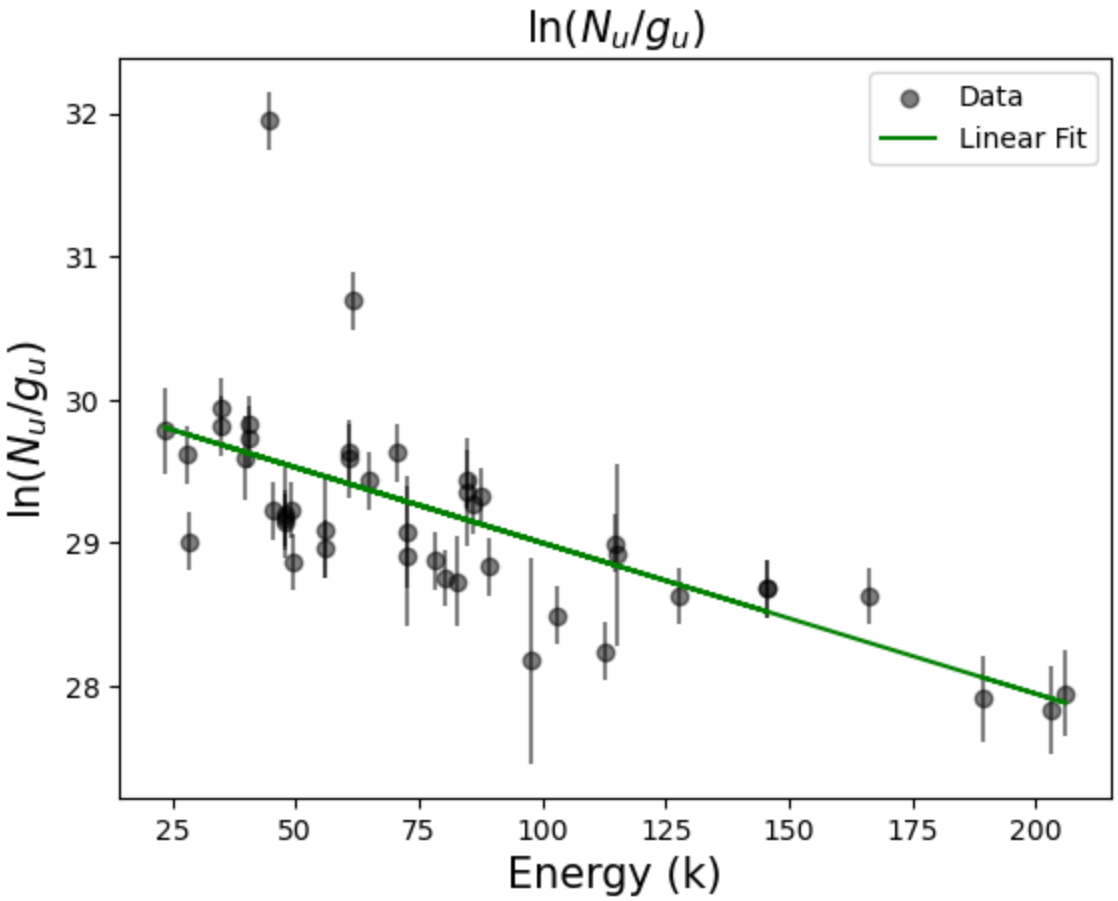
Q2:

Q2: Fit a linear function to the data and obtain the values for a and b such that $ln(Nu/gu) = a + bEu$

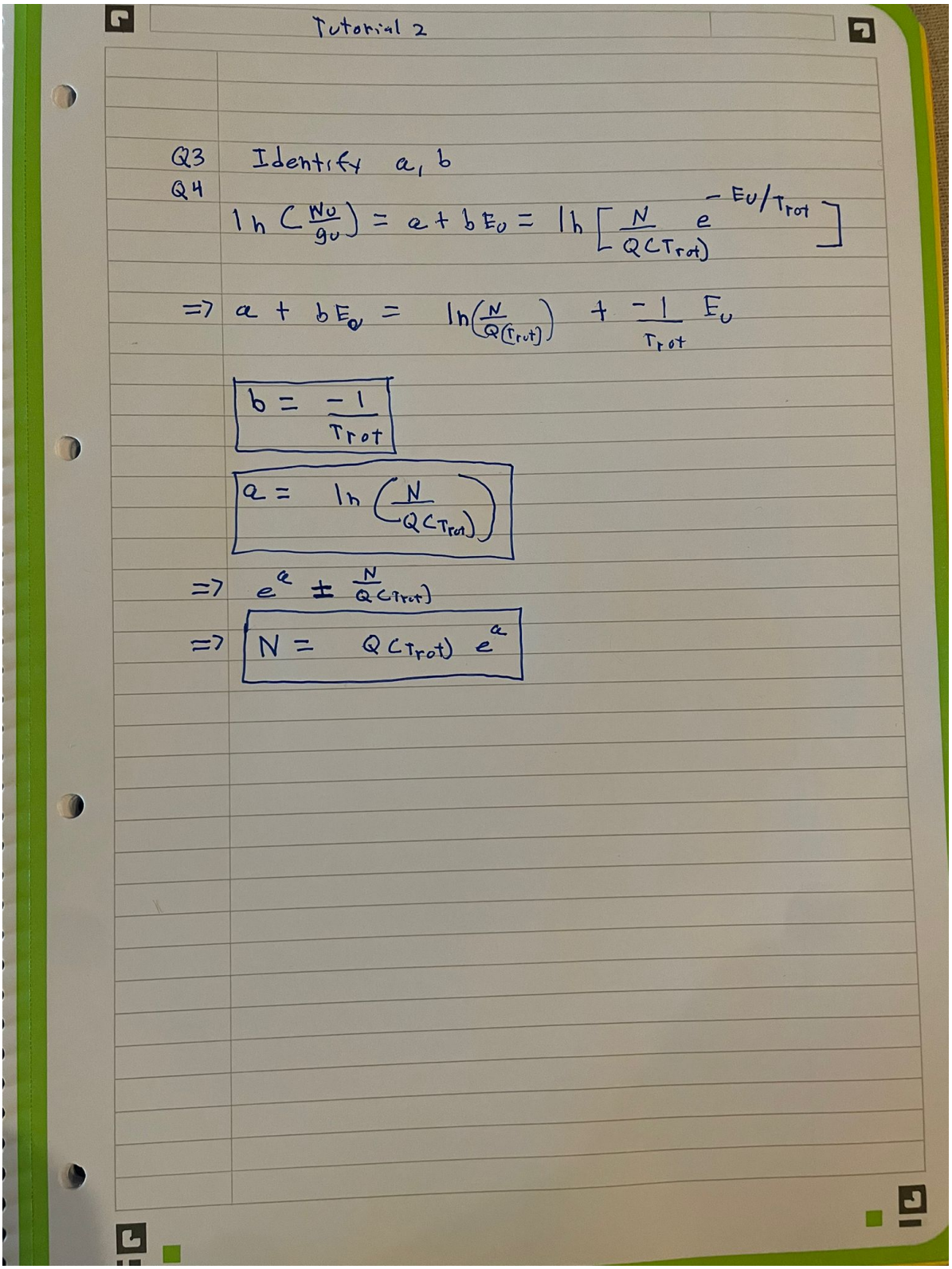
In [16]:
```
# Define the linear function
def linear_function(eu, a, b):
    return a + b * eu

# Use curve_fit with w_sigma as the weights
popt, pcov = curve_fit(linear_function, eu, ln_ratio, sigma=sigma_ratio)

# Extract the values for a and b
a, b = popt

my_t = -1/b
my_a = a # saving a and b for later

# Print the values of a and b
print("a:", a)
print("b:", b)

# Plot the data and the fitted line
plt.title('$\ln(N_u/g_u)$', size=15)
plt.errorbar(eu, ln_ratio, yerr=sigma_ratio, fmt='None', color = 'k',alpha=0.5
plt.scatter(eu, ln_ratio, color='k', label='Data', alpha=0.5)
plt.plot(eu, linear_function(eu, a, b), color='Green', label='Linear Fit') #la
plt.xlabel('Energy (k)', size=15)
```

```
plt.ylabel('$\ln(N_u/g_u)$', size=15)
plt.legend()
plt.show()
```

```
a: 30.054085921293527
b: -0.010534341369827421
```



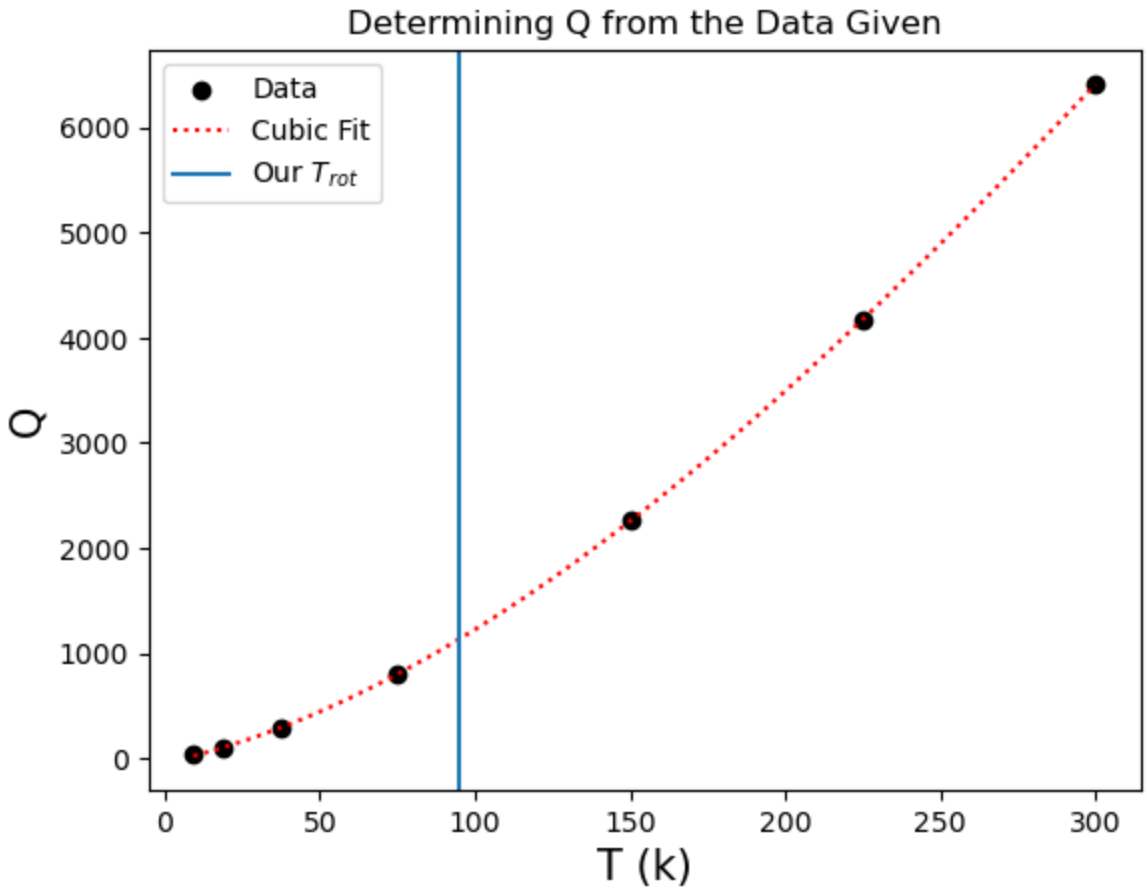
Q3 & Q4:



Q4 part 2:

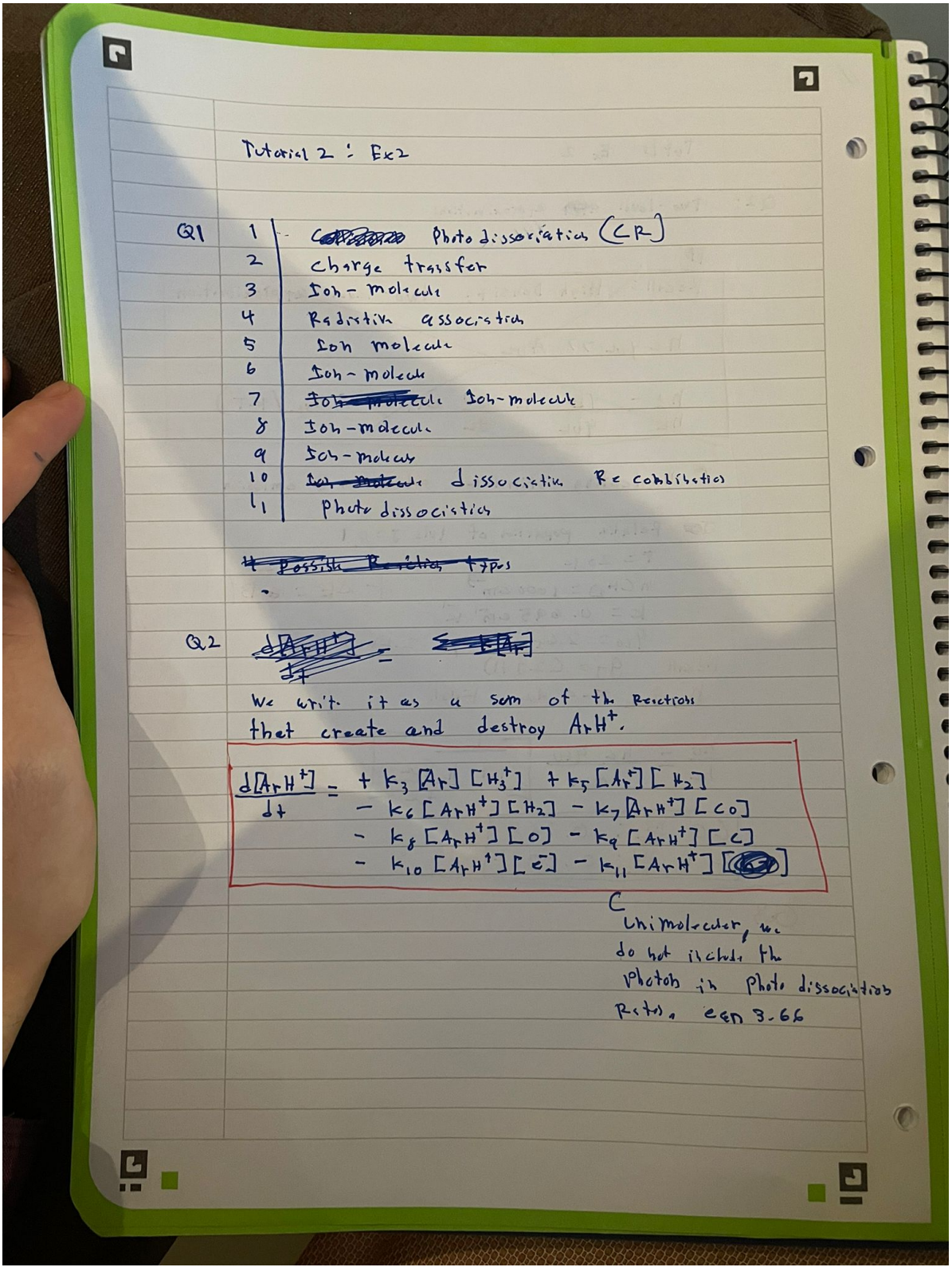
We can fit a cubic function to the data table to determine our Q, based on our T, which we find using their relations to a, and b

In [24]:
```
# Given data
T_table = np.asarray([300.0, 225.0, 150.0, 75.0, 37.5, 18.75, 9.375]) #k
Q_table = np.asarray([6413.573, 4165.815, 2267.776, 801.678, 283.465, 100.207,

# Define the cubic function
def cubic_function(T, a, b, c, d):
    return a * T**3 + b * T**2 + c * T + d

# Fit the cubic function to the data
params, covariance = curve_fit(cubic_function, T_table, Q_table)

# Extract the parameters
a, b, c, d = params

# Generate a smooth curve using the fitted parameters
T_smooth = np.linspace(min(T_table), max(T_table), 100)
Q_smooth = cubic_function(T_smooth, a, b, c, d)

# Plot the original data and the fitted curve
plt.scatter(T_table, Q_table, label='Data', color='k')
plt.plot(T_smooth, Q_smooth, label='Cubic Fit', color='red', linestyle=':')
plt.xlabel('T (k)', size=15)
plt.ylabel('Q', size=15)
plt.title('Determining Q from the Data Given')
plt.axvline(my_t, label='Our $T_{rot}$')
plt.legend()
plt.show()


# Function to determine Q for a given T
def get_Q(T):
    return cubic_function(T, a, b, c, d)

# Example: Determine Q for a given T
resulting_Q = get_Q(my_t)
print('Answer for Q4')
print(f"For T = {my_t} k, Q = {resulting_Q}")

Q = resulting_Q
a = my_a
N = Q * np.exp(a)

print(f"N = {N}")
```




```
Answer for Q4
For T = 94.92762431870788 k, Q = 1134.8932372414174
N = 1.2802025335819972e+16
```

Ex 2



Tutorial 2 : Ex2

Q1

1	[redacted] Photo dissociation (CR)
2	Charge transfer
3	Ion - molecule
4	Radiative association
5	Ion molecule
6	Ion - molecule
7	~~Ion - molecule~~ Ion-molecule
8	Ion-molecule
9	Ion-Molecule
10	~~Ion - Molecule~~ dissociative Recombination
11	Photo dissociation

~~# possible Reactions types~~

Q2

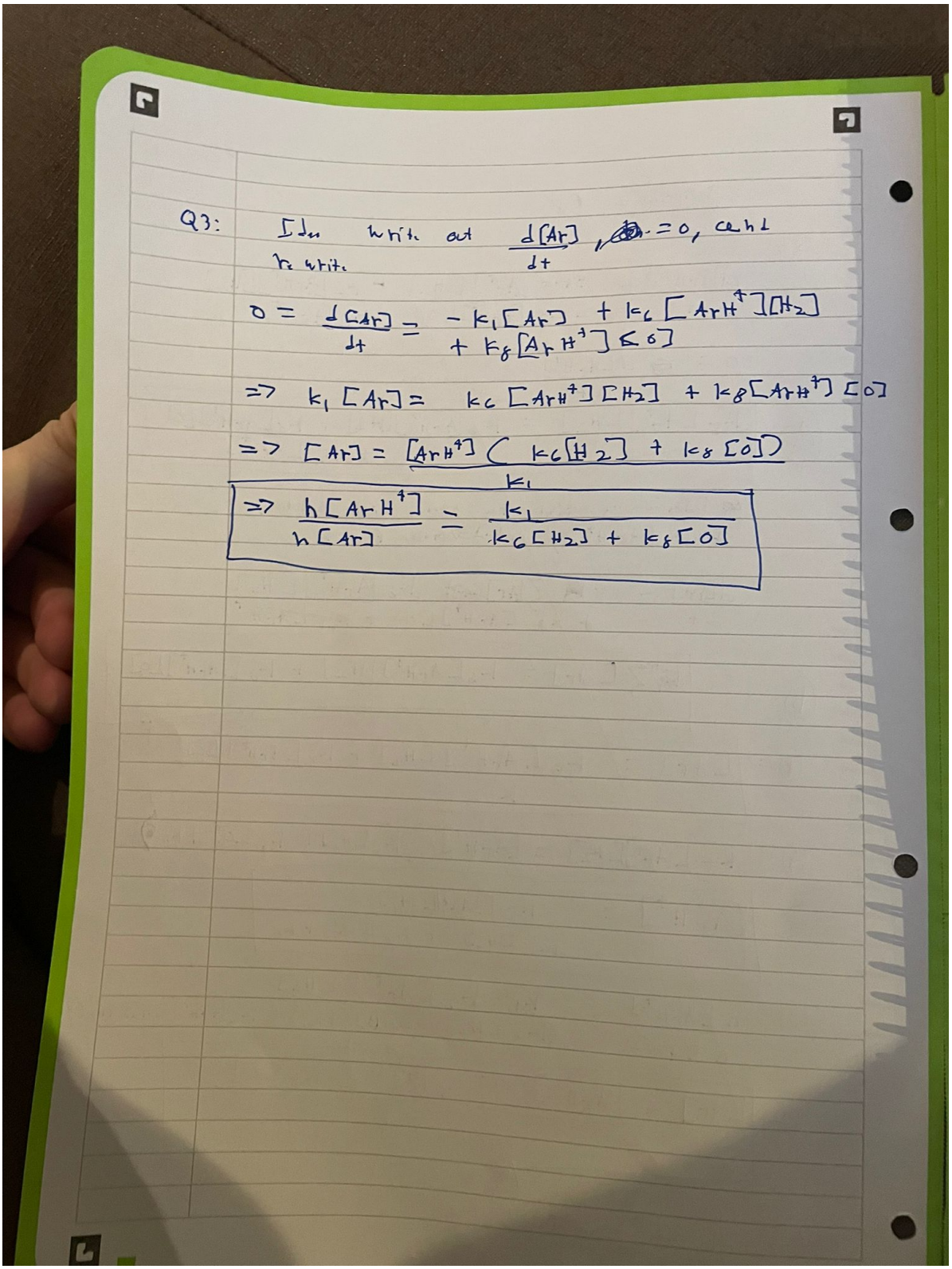
[redacted]

We write it as a sum of the Reactions that create and destroy ArH^+.

$$\frac{d[ArH^+]}{dt} = +k_3[Ar][H_3^+] + k_5[Ar^+][H_2] - k_6[ArH^+][H_2] - k_7[ArH^+][CO] - k_8[ArH^+][O] - k_9[ArH^+][C] - k_{10}[ArH^+][e^-] - k_{11}[ArH^+][\text{[redacted]}]$$

C

Unimolecular, we do not include the photon in photo dissociation Reaction. eqn 3.66



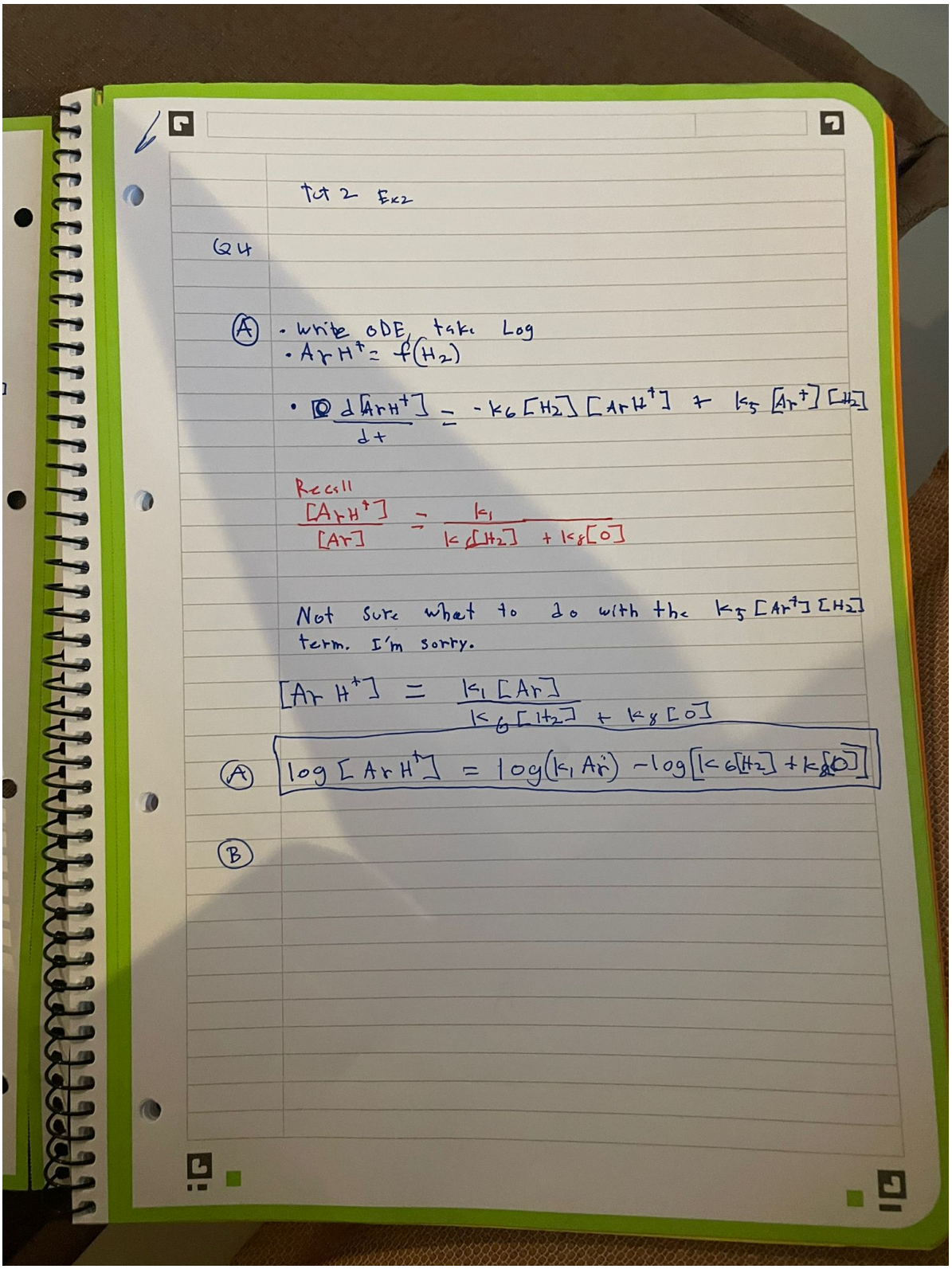
Q3: In the steady state, $\frac{d[Ar]}{dt} = 0$, and hence

$$0 = \frac{d[Ar]}{dt} = -k_1[Ar] + k_6[ArH^+][H_2] + k_8[ArH^+][O] \approx 0$$

$$\Rightarrow k_1[Ar] = k_6[ArH^+][H_2] + k_8[ArH^+][O]$$

$$\Rightarrow [Ar] = \frac{[ArH^+]\left(k_6[H_2] + k_8[O]\right)}{k_1}$$

$$\Rightarrow \frac{h[ArH^+]}{h[Ar]} = \frac{k_1}{k_6[H_2] + k_8[O]}$$



Q4b

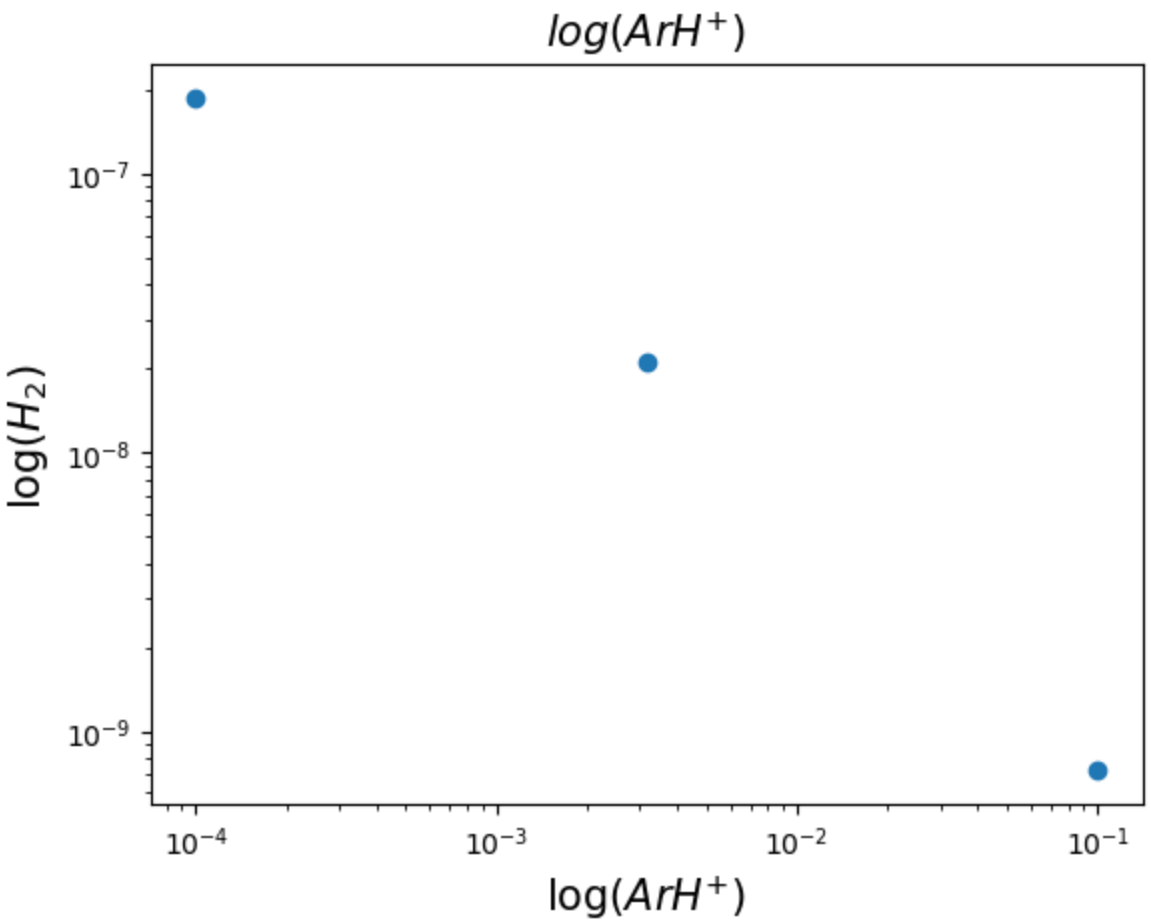
In [25]:
```
#
def get_arh(h2, k1, k6, k8, o, ar):
    'handy function for calculating abduences'
    arh = k1 * ar / (k6*h2 + k8*o)
    return arh
```

In [29]:
```
k1 = 11.4 * 2e-16
k6 = k8 = 1e-10

o = 2.9e-4
ar = 3.2e-6
h2 = np.asarray([-1, -2.5, -4])
h2 = 10**h2

my_arh = get_arh(h2, k1, k6, k8, o, ar)

print('abundance of ArH+')
print(my_arh)
```

```
abundance of ArH+
[7.27490278e-10 2.11338737e-08 1.87076923e-07]
```

In [21]:
```
plt.scatter(h2, my_arh)
plt.xscale('log')
plt.yscale('log')

plt.ylabel('$\log(H_{2})$', size=15)
plt.xlabel('$\log(ArH^{+})$', size=15)

plt.title('$log(ArH^{+})$', size=15)
```

Out[21]:
```
Text(0.5, 1.0, '$log(ArH^{+})$')
```



Q4C

- From theory and from the plot, the $\log(ArH^+)$ is linear with a negative slope. The higher the abundance of ArH+ the lower the abdundance of H2
- we conclude that ArH+ is a good tracer of atomic gas because directly proportional to Ar, and inversely to H2, (we dervied the fracion Arh+/Ar already). Because of this relationship, we can observe ArH+ and infer the amounts of H2, Ar, and O.

In [ ]: