Franck–Hertz Experiment

In [2]:
```
%reset -f
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import linregress
import pandas as pd
import statistics
```

In [3]:
```
df = pd.read_excel(r'/home/ashwin/Git/general-physics-lab/frank_hertz_2.xlsx')
print(df)
```

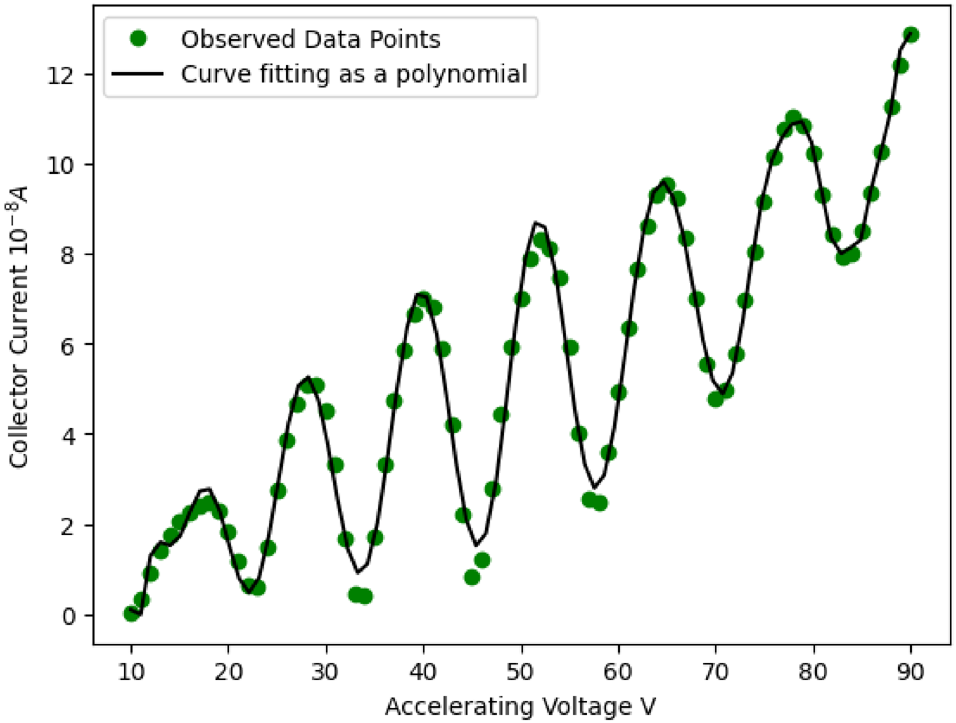
```
    VG2K  CURRENT
0      0     0.00
1      1     0.00
2      2     0.00
3      3     0.00
4      4     0.00
..   ...      ...
86    86     9.35
87    87    10.25
88    88    11.25
89    89    12.18
90    90    12.87

[91 rows x 2 columns]
```

In [4]:
```
VG2K=df['VG2K'].to_numpy()
current=df['CURRENT'].to_numpy()
```

In [16]:
```
x=VG2K[10:]
y=current[10:]
deg=80

result=np.polyfit(x,y,deg)


p = np.poly1d(result)

xp = np.linspace(10, 90, 80)

plt.figure()
plt.plot(x, y, 'og',label="Observed Data Points")
plt.plot(xp, p(xp),'k',label="Curve fitting as a polynomial")
plt.xlabel("Accelerating Voltage V")
plt.ylabel("Collector Current $10^{-8}A$")

plt.legend()
```

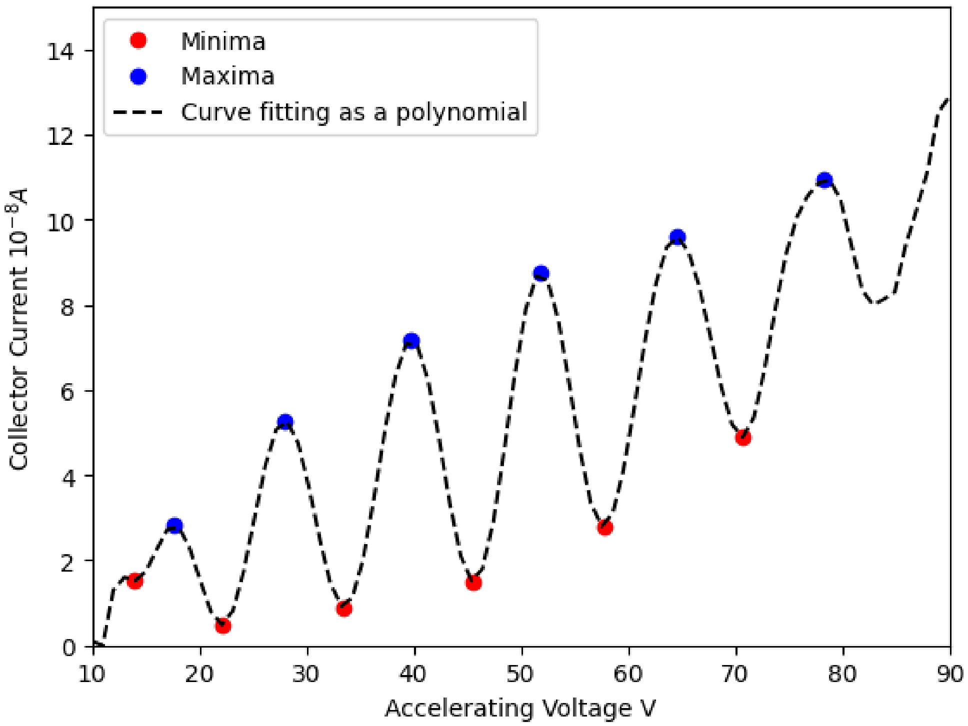
```
/home/ashwin/.local/lib/python3.10/site-packages/IPython/core/interactiveshell.py:3378: RankWarning: Polyfit may be poorly conditioned
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Out[16]: <matplotlib.legend.Legend at 0x7f74d99a8370>



In [15]:

```python
crit = p.deriv().r
r_crit = crit[crit.imag==0].real
test = p.deriv(2)(r_crit)


# compute local minima
# excluding range boundaries
x_min = r_crit
x_min=x_min[::-1]
x_min=sorted(x_min)
#Removing the ourlier values.
x_min=x_min[2:-1]

y_min = p(x_min)
plt.plot( x_min[::2], y_min[::2], 'or' , label="Minima" )
plt.plot( x_min[1::2], y_min[1::2], 'ob' , label="Maxima " )


plt.plot(xp, p(xp),'k--',label="Curve fitting as a polynomial")
plt.xlabel("Accelerating Voltage V")
plt.ylabel("Collector Current $10^{-8}A$")

plt.xlim([10,90])
plt.ylim([0,15])
plt.legend()

plt.show()
```



In [7]:
```
print(df)
```

```
    VG2K  CURRENT
0      0     0.00
1      1     0.00
2      2     0.00
3      3     0.00
4      4     0.00
..   ...      ...
86    86     9.35
87    87    10.25
88    88    11.25
89    89    12.18
90    90    12.87

[91 rows x 2 columns]
```

In [65]:
```
diff=[]
for i in range(1,len(x_min)-2):
    diff.append(abs(x_min[i]-x_min[i+2]))
print(diff)
print(len(diff))
```

```
[10.311111527425563, 11.389201767235946, 11.815394039170151, 12.09308971360467, 12.097065908940685, 12.10628727347192, 12.665408674048408, 12.925463151598443, 13.769943646212766]
9
```

In [66]:
```
statistics.stdev(diff)
```
Out[66]:
```
0.9726228575481425
```
In [67]:
```
statistics.mean(diff)
```
Out[67]:
```
12.130329522412062
```
In [68]:
```
print("Percentage error compared with literature is \n {0}%".format(round(abs((statistics.mean(diff)-11.83)/11.83*100),2)))
```
```
Percentage error compared with literature is
 2.54%
```
In [69]:
```
print("The Energy drop on inelastic collision of electron on Argon atom is :  : \n  {0} +- {1} ".format(round(statistics.mean(diff),2),roun
```
```
The Energy drop on inelastic collision of electron on Argon atom is :  :
  12.13 +- 0.97
```

This Concludes and quantization of energy in atomic model as per Quantum theory.