Assignment 6: Simulation - The Tube Light Model

J.Phani Jayanth - EE19B026

April 16, 2021

Abstract

This week's Python Assignment involves the following:

- Simulating a 1-Dimensional model of a tubelight
- Plotting the Light Intensity (I) in the tubelight against the position (x)
- Plotting the Electron density (E) in the tubelight against position (x) and Electron phase space i.e., velocity (v) vs position (x)

Introduction

In the 1-Dimensional model of the tubelight, a uniform electric field is present, that accelerates the electrons. Electrons are emitted by the cathode and get accelerated in this field. When they get beyond a threshold energy, the can drive atoms into exited states. The relaxation of these atoms results in light emmision.
Electrons reaching the anode are absorbed and lost. At each time step, an average of N electrons are introduced at the cathode. The actual number of electrons is determined by finding the integer part of a random number that is normally distributed with a certain standard deviation and mean.

Code Description

We create the simulation universe as follows:

- The tubelight is divided into n sections
- In each time instant, M electrons are injected
- We run the simulation for nk turns
- The electrons require a minimum velocity (u0), called threshold velocity to be capable of exciting the atoms
- Beyond this velocity, there is a probability "p", that a collision will occur and an atom is excited. After the collision, electon's velocity reduces to zero

The default values of these parameters are:

```
n = 100  # spatial grid size
M = 5    # number of electrons injected per turn
nk = 500 # number of turns to simulate
u0 = 5   # threshold velocity
p = 0.25 # probability that ionization will occur
```

The entire simulation and its steps are enclosed within the function *Tubelight()*, whose inputs are the parameters mentioned above. An additional parameter *Msig* is defined as the variance in M i.e., the number of injected electrons at each time step.
Additionally, all electrons are constrained between $0 < x < L$. Any electrons out of this range are set to $x = 0$ and $u = 0$.
We assume that the electric field creates an acceleration of 1 unit. We find the displacement (dx), and the subsequent position (x) and velocity (u) of the electron as follows:

The code is as follows:

```
import sys
from pylab import *
import numpy as np
if (len(sys.argv))==1:
    n = 100
    M = 5
    nk = 500
    u0 = 5
    p = 0.25
elif (len(sys.argv))==6:
    try:
        n = int(sys.argv[1])
        M = int(sys.argv[2])
        nk = int(sys.argv[3])
        u0 = float(sys.argv[4])
        p = float(sys.argv[5])
    except Exception:
        print(f'ERROR! Wrong Usage!!\nCorrect usage: python3 {sys.argv[0]} n M nk
     u0 p')
        print('n = Spatial grid size\nM = Number of electrons injected per turn\
    nnk = Number of turns to simulate\nu0 = Threshold velocity\np = Probability
    that Ionization will occur')
        exit()
else:
    print(f'ERROR! Wrong Usage!!\nCorrect usage: python3 {sys.argv[0]} n M nk u0
    p')
    print('n = Spatial grid size\nM = Number of electrons injected per turn\nnk =
     Number of turns to simulate\nu0 = Threshold velocity\np = Probability that
    Ionization will occur')
    exit()
```

```
Msig = 1 # Variance

def Tubelight(n,M,nk,u0,p,Msig):
    xx = np.zeros(n*M)
    u = np.zeros(n*M)
    dx = np.zeros(n*M)
    I = []
    X = []
    V = []
    for turn in range(nk):
        # Injecting Electrons
        m = int(randn()*Msig+M)
        # Add them to free slots
        ee = where(xx==0)[0]
        # Initialize their position to 1
        xx[ee[0:m]] = 1
        # Finding electrons present in the chamber
        ii = where(xx>0)[0]
        # Adding their positions and velocities to X and V
        X.extend(xx[ii].tolist())
        V.extend(u[ii].tolist())
        # Calculate the displacement during this turn
        dx[ii] = u[ii] + 0.5
        # Update Position and Velocity
        xx[ii] += dx[ii]
        u[ii]+=1
        # Determine particles which have hit the anode
        jj = where(xx>=n)[0]
        xx[jj]=0
        u[jj]=0
        dx[jj]=0
        # Electrons with velocity greater than threshold velocity
        kk = where(u>=u0)[0]
        ll = where(rand(len(kk))<=p)[0]
        kl = kk[ll]
        # Reset the velocity of these electrons to zero (inelastic collision)
        u[kl]=0
        # Determining the point of collision
        rho = rand(1)
        xx[kl] -= dx[kl]*rho
        # Emissions
        I.extend(xx[kl].tolist())

    return X,V,I
```

The function *Tubelight* accomplishes all the steps required in the simulation and returns the Electron position (X), Electron velocity (V) and the Intensity of emitted light (I). This data is used to obtain the plots as shown in the next section.

Simulation and Plots

We plot the following:

- Electron Density - Population plot of X
- Light Intensity
- Electron Phase-Space

```
def Plots(X,V,I):
    # Electron density
    figure()
    hist(X,bins=range(0,n+1),color='r',ec='black')
    title("Electron density")
    xlabel("$x$")
    ylabel("Number of electrons")
    show()
    # Light instensity
    figure()
    ints,bins,trash = hist(I,bins=range(0,n+1),color='yellow',ec='black')
    title("Light Intensity")
    xlabel("$x$")
    ylabel("I")
    show()
    # Electron phase-space
    figure()
    scatter(X,V,marker='x')
    title("Electron Phase Space")
    xlabel("$x$")
    ylabel("$v$")
    show()
    return ints,bins

X,V,I = Tubelight(n,M,nk,u0,p,Msig)
ints, bins = Plots(X,V,I)

# Tabulating Data
xpos = 0.5*(bins[0:-1]+bins[1:])
print("Intensity Data:")
print('xpos',end='    ')
print('count')
for k in range(len(xpos)):
    print(xpos[k],end='     ')
    print(ints[k])
```

The plots obtained for the default values - n = 100, M = 5, nk = 500, u0 = 5, p = 0.25 are:

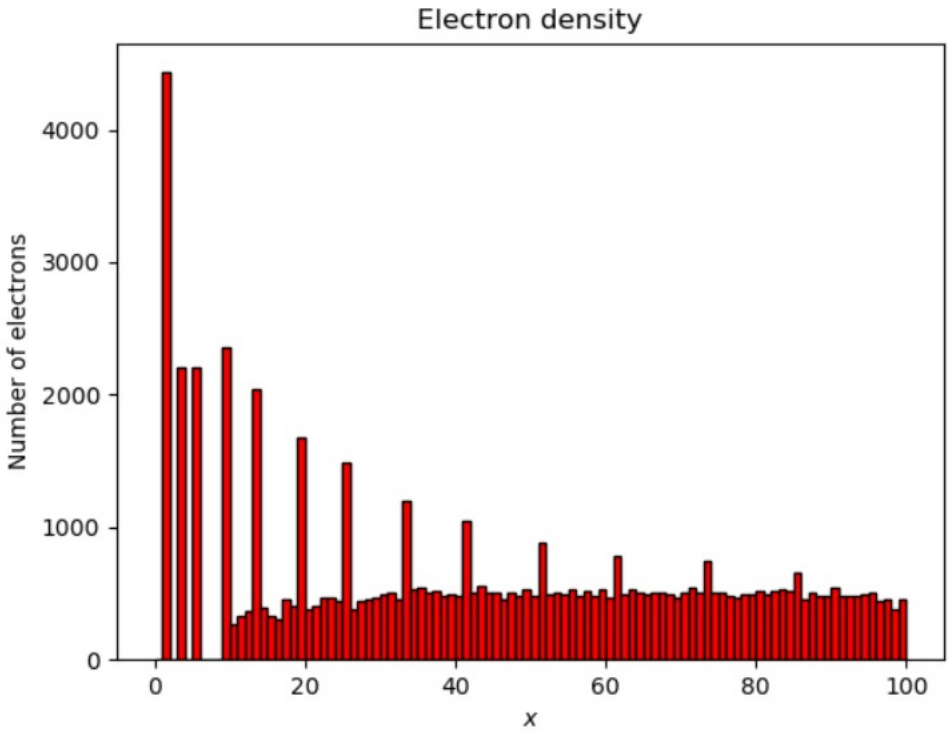

Figure 1: Electron Density - Population Plot of X

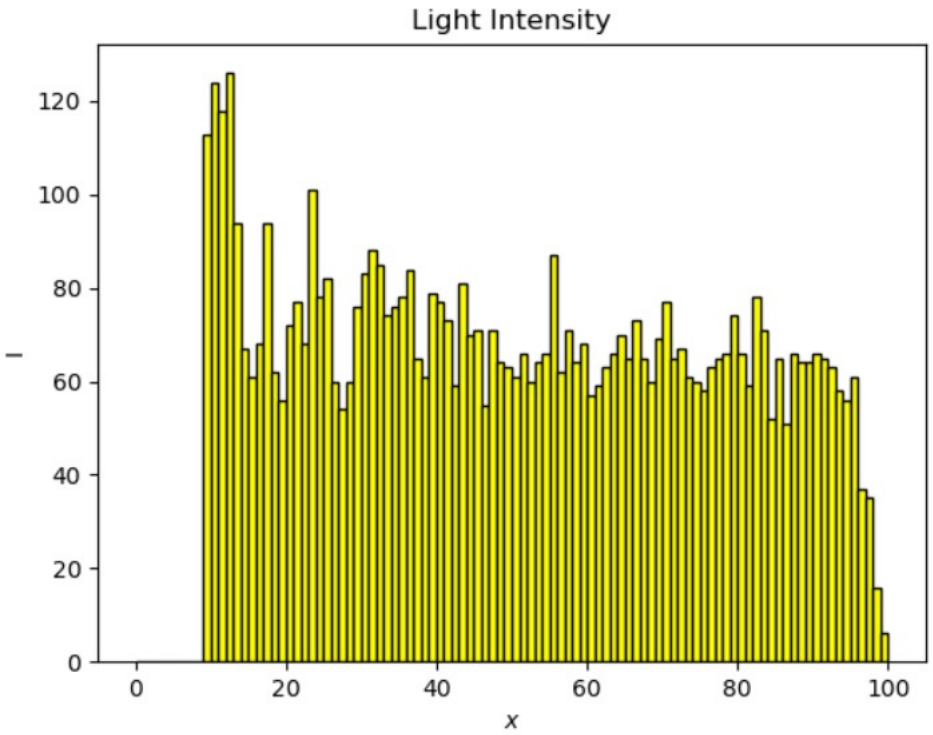

Figure 2: Intensity of Emission

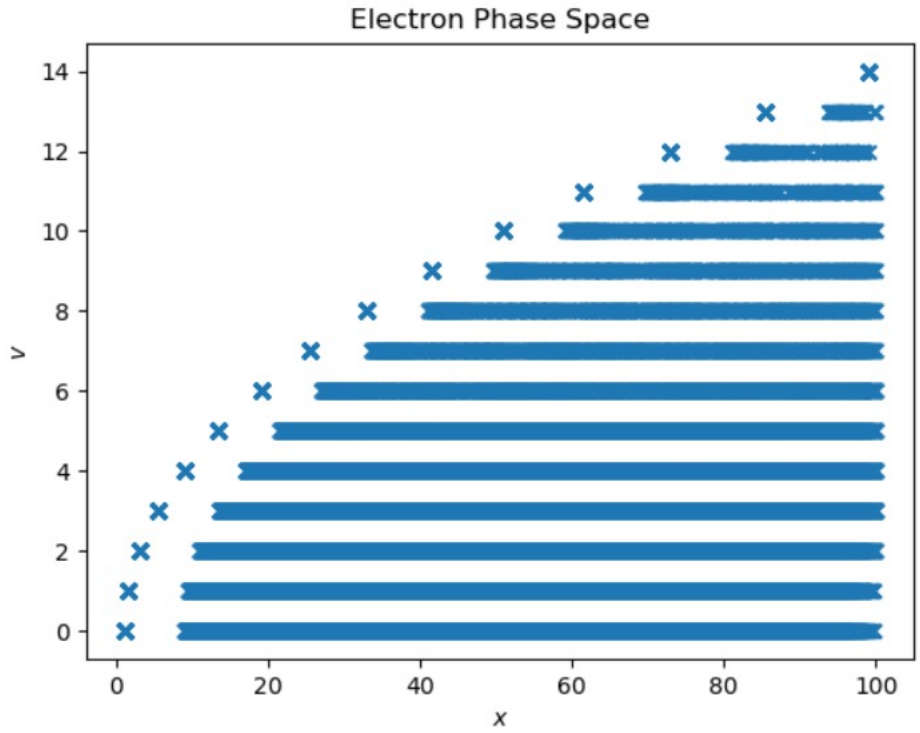

Figure 3: Electron Phase Space

For the values u0 = 7 and p = 0.5, we obtain the following plots:

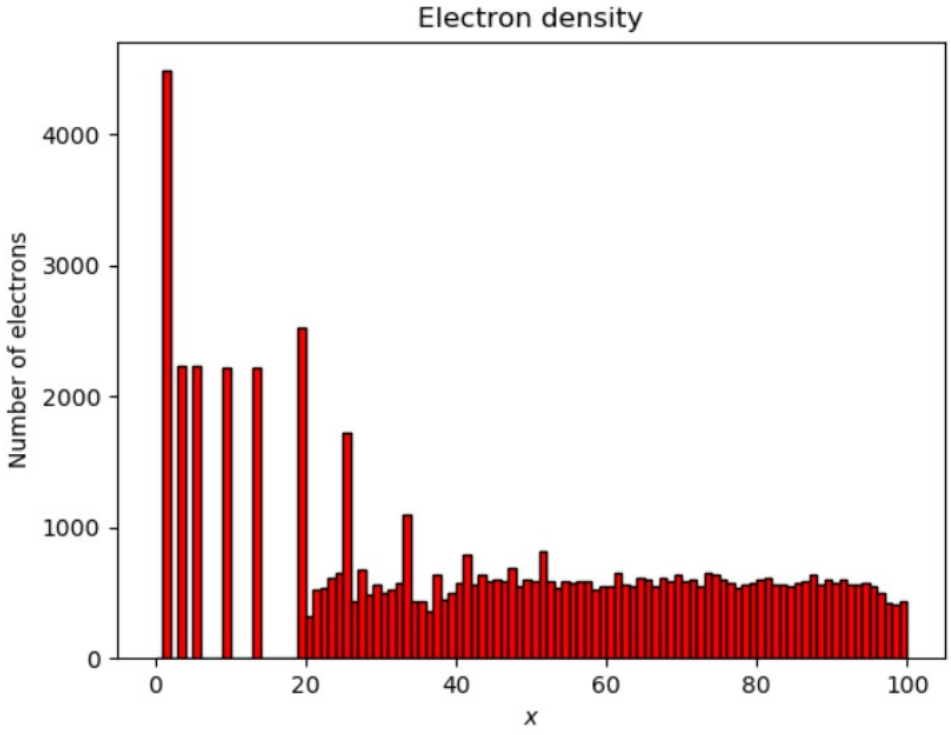

Figure 4: Electron Density - Population Plot of X

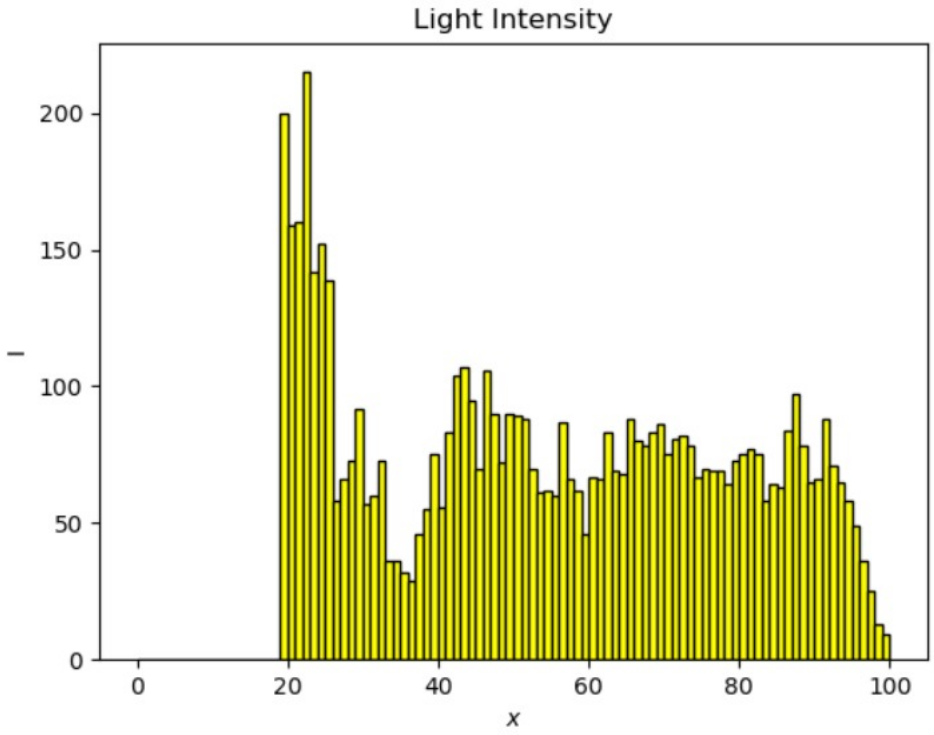

Figure 5: Intensity of Emission

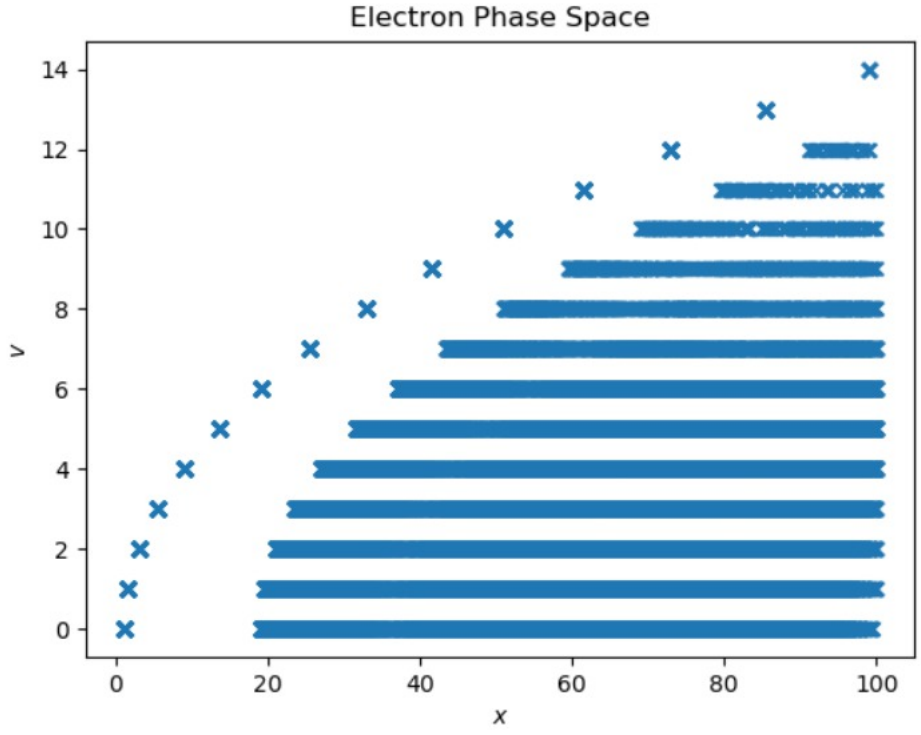

Figure 6: Electron Phase Space

Intensity Data

We print out the Intensity data along with the positions of electrons as follows:

Position	Count
0.5	0.0
1.5	0.0
2.5	0.0
...	...
9.5	100.0
10.5	144.0
11.5	124.0
...	...
97.5	20.0
98.5	24.0
99.5	9.0

We can observe that the intensity is zero till a certain point (around 10). This is because the electrons are building up their energy in this region. After this, they will have sufficient enegrgy to excite atoms and cause emission of light.

Conclusion

This week's assignment shows how Python could be used in simulation of models and the simplification of code through vectorization.

- We simulated a 1-Dimensional model of a tubelight and obtained the necessary data - Electron density, Position and Velocity.
- We also obtained the Emission intensity at various locations in the tubelight and have also observed the presence of dark regions/spaces.
- We observed that as the velocity increases, the number of electrons possessing that velocity decreases, from the phase-space plot.
- We have also seen that varying the various parameters can lead to different distributions of the above mentioned plots.