EE2703 Assignment 6 : Tubelight Simulation

Aman Kumar EE19B066

April 17, 2021

1 The Assignment

1.1 Introduction

In this assignment we have to simulate a tubelight. We use a 1-Dimensional model of the tubelight. A uniform electric field is present, that accelerates electrons. Electrons are emitted by the cathode with **zero energy**(i.e. zero velocity), and accelerated in this field. When they get beyond a **threshold energy E0**(i.e. threshold velocity *u0*), they can drive atoms to excited states. The relaxation of these atoms results in light emission. In our model, we have to assume that the relaxation is immediate. The collision is inelastic, and thus the electron loses all its energy after the collision and the process starts again. The electrons reaching the anode are lost.

One of the important things that we learn in this assignment is learning to work with index vectors. We also learn how to run simulations in time.

1.2 Things to do

Mainly the following things are asked in this assignment:

- Electron density plot
- Population plot of light intensity. i.e. Light intensity as a function of space after the process reaches steady state.
- Electron phase plot.
- Intensity data table.

2 Assignment questions

2.1 The simulation universe

- The tube is divided into **n** sections.
- In each time instant, on average **M** electrons are injected with a standard deviation of **Msig**.
- We run the simulation for **nk** turns.
- The electrons are unable to excite the atoms till they have a threshold velocity of **u0**.
- Beyond this velocity, there is a probability **p** in each turn that a collision will occur and an atom excited.

The electron's velocity reduces to zero if it collides.

Each of six parameters have default values, and if the user wants to update some parameter(s) then all 6 parameters have to passed with updated values. They should be passed in this order-

```
n M Msig nk u0 p
```

If the arguments are not passed correctly, then the default values are taken for **all** of them. The python code snippet for doing this:

```
if len(argv) != 7:                      #Checking if all 6 parameters have been passed
    n,M,Msig,nk,u0,p = 100,5,2,500,5,0.25    #Default values
    print("Usage: python",argv[0],"n M Msig nk u0 p")
    print("The program will run using default parameter values.")
else:
    #Updating the values user has passed
    n = int(argv[1])        #spatial frid size
    M = int(argv[2])        #avg number of electrons injected per turn
    Msig = float(argv[3])   #stddev of number of electrons injected each turn
    nk = int(argv[4])       #number of turns
    u0 = float(argv[5])     #threshold velocity
    p = float(argv[6])      #probability of ionization
```

2.2 Vectors to store information

We need vectors to hold the electron information. We need vectors for:

- Electron position xx
- Electron velocity u
- Displacement in current turn dx

We also need vectors to accumulate information as part of simulation. Three vectors are needed for storing:

- Intensity of emitted light, I
- Electron position, X
- Electron velocity, V

```
#Creating vectors to hold the electron information
xx = np.zeros(n*M)                  #Electron position
u = np.zeros(n*M)                   #Electron velocity
dx = np.zeros(n*M)                  #Displacement in current turn
#Lists to accumulate information as part of the simulation
I = []                              #Intensity of emitted light
X = []                              #Electron position
V = []                              #Electron velocity
```

2.3 The Loop

This is the main loop of the program. It performs the simulation and Saves the Intensity, Position and Velocity data in I, X, V respectively. We are assuming that the Electric field creates an acceleration of 1. Hence, the displacement of the i^{th} electron is given by

$$dx_i = u_i \Delta t + \frac{1}{2} a (\Delta t)^2 = u_i + 0.5$$

```
ii = np.where(xx > 0)               #Finding indices of existing electrons
for k in range(nk):
    #ii = np.where(xx > 0)
    dx[ii] = u[ii] + 0.5            #Finding displacement for the electrons in this turn
```

```
xx[ii] = xx[ii] + dx[ii]        #Updating the electron position
u[ii] = u[ii] + 1               #Updating velocity.(acceleration = 1 unit)

jj = np.where(xx >= n)          #Electrons that have reached Anode are lost.
xx[jj], dx[jj], u[jj] = 0,0,0   #Resetting there position, velocity and displacement

kk = np.where(u >= u0)          #Finding electrons having atleast threshold energy
ll = np.where(np.random.rand(len(kk[0]))<=p)
kl = kk[0][ll]                  #Electrons at these indices will suffer collision

u[kl] = 0                       #After inelastic collision they lose all energy
xx[kl] = xx[kl] - dx[kl]*rd.random()
I.extend(xx[kl].tolist())       #Light is emitted where collision happens.

m = round(pl.randn()*Msig + M)    #Numner of electrons to be injected
zz = np.where(xx == 0)          #Finding available space in the array.
xx[zz[0][:m]] = 1               #Newly injected electrons are at position 1
dx[zz[0][:m]] = 0               #Newly injected electrons have displacement = 0
u[zz[0][:m]] = 0                #Newly injected electrons have velocity = 0

ii = np.where(xx > 0)           #Existing elecrons
X.extend(xx[ii].tolist())       #Adding there position to X
V.extend(u[ii].tolist())        #Adding there velocity to V
```

3 Plots and Results(*Parameters* : 100 5 2 500 5 0.25)

3.1 Electron density

The number of electrons between i and i+1. We do this using the `hist` function. The position data is accumulated in the vector `X`.

```
pl.hist(X,n)
pl.title("Population plot for Electron Density")
```

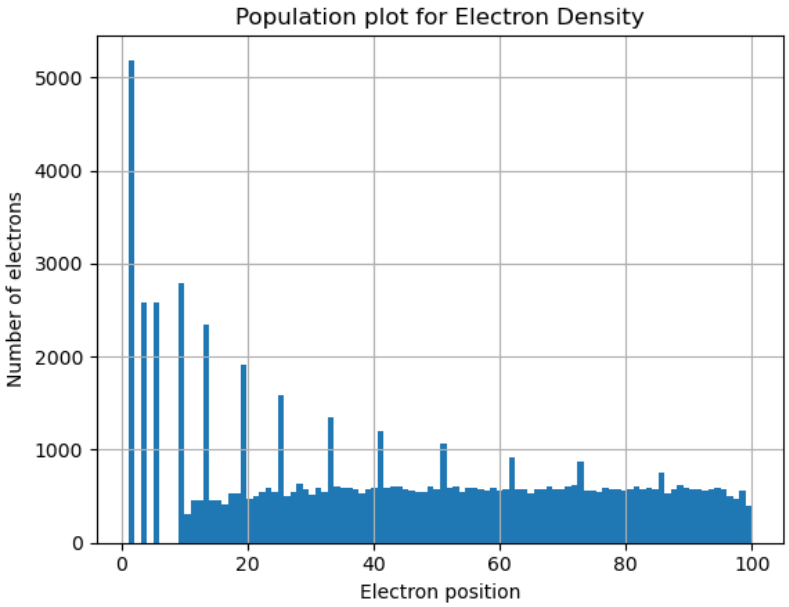

Figure 1: Electron Density

3.2 Light Intensity

The intensity data is accumulated in the vector `I`. That is it stores the positions where excitation happened after collision. These positions are stored the number of times a collision happens there.

```
count,bins,rec=pl.hist(I,n)
pl.title("Population plot for Light Intensity")
```

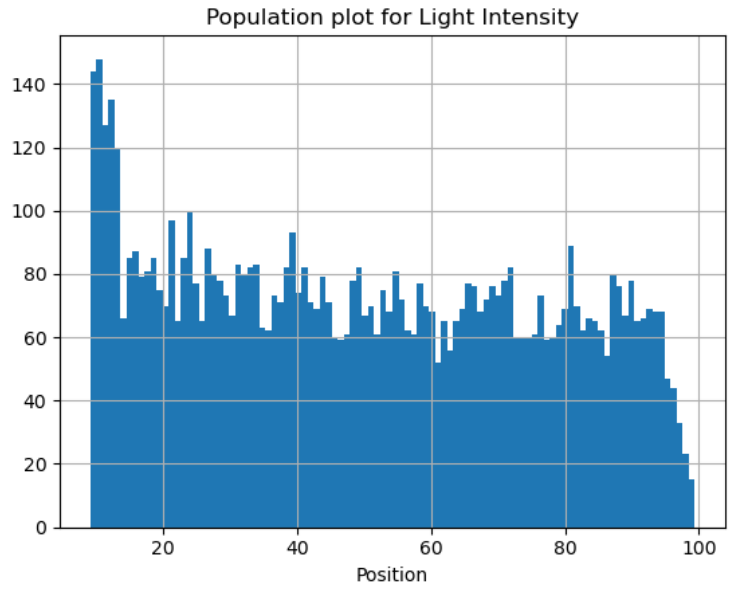

Figure 2: Light Intensity

3.3 Electron phase space

For each electron we plot a "X" corresponding to $x = x_i$, $y = v_i$.

```
pl.scatter(X,V,marker="x")
pl.title("Electron Phase space")
```

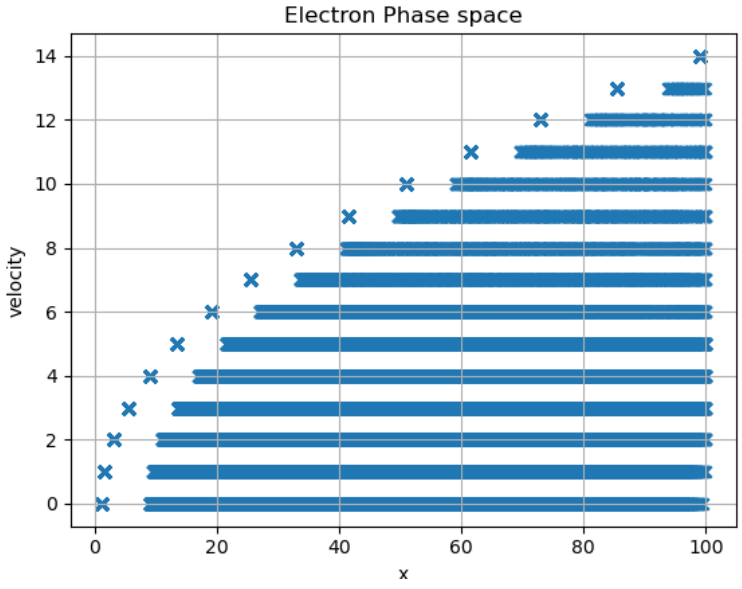

Figure 3: Electron phase space

3.4 Intensity Table

We need to print the population count against average bin positions.Python code snippet for printing the intensity table.

```
xpos = 0.5*(bins[0:-1]+bins[1:])    #Converting bin positions to mid point values
print("\nIntensity data:\n")
print("\txpos\tcount")
ogm=np.c_[xpos, count]              #Concatenating the xpos and count vectors column-wise
s1 = str(ogm)
s2 = s1.replace(’], [’,’\n’)
s3 = s2.replace(’[’, ’’)
s4 = s3.replace(’]’,’’)
s5 = s4.replace(’, ’,’\t’)
print(s5)
```

An example run of the program with intensity table:

```
(base) H:\Aman 1\4th Semester\EE2703 APL\Assignment 6>python EE2703_ASSIGN6_EE19B066.py 100 5 2 500 5 0.25

Intensity data:

        xpos    count
  9.46305809 144.
  10.3671875  148.
  11.27131691 127.
  12.17544632 135.
  13.07957573 120.
  13.98370514  66.
  14.88783455  85.
  15.79196396  87.
  16.69609337  79.
  17.60022278  81.
  18.50435219  85.
  19.4084816   75.
  20.31261101  70.
  21.21674042  97.
  22.12086983  65.
  23.02499924  85.
  23.92912865 100.
  24.83325806  77.
  25.73738747  65.
  26.64151688  88.
  27.54564629  80.
  28.4497757   78.
  29.35390511  73.
  30.25803452  67.
  31.16216393  83.
  32.06629334  80.
  32.97042275  82.
  33.87455216  83.
  34.77868157  63.
  35.68281098  62.
  36.58694039  73.
```

Figure 4: Example Run

3.5 Some more plots

I am plotting the above three plots again, this time with-

```
u0 = 7
p = 0.75
All other parameters are kept as it is
```

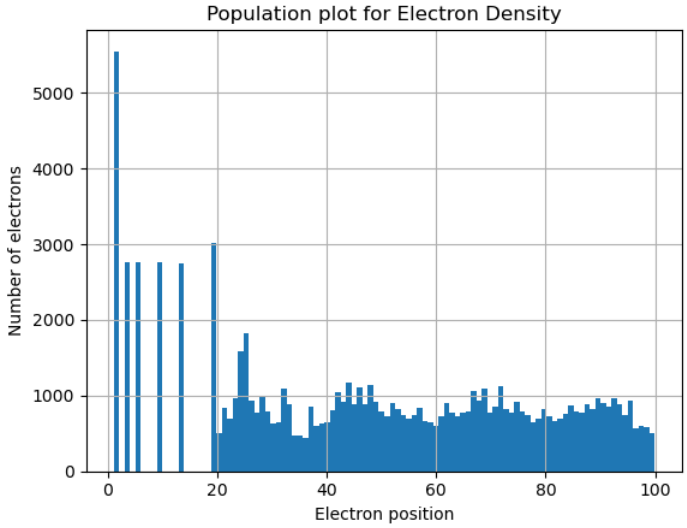

Figure 5: Electron density

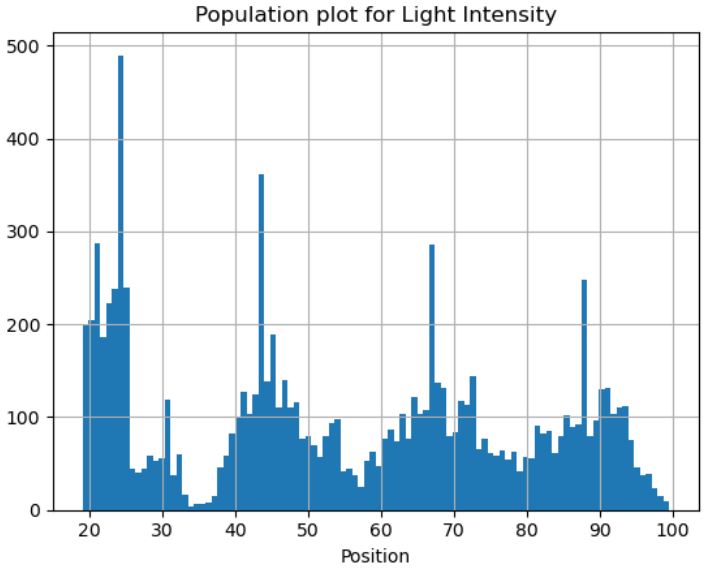

Figure 6: Light Intensity

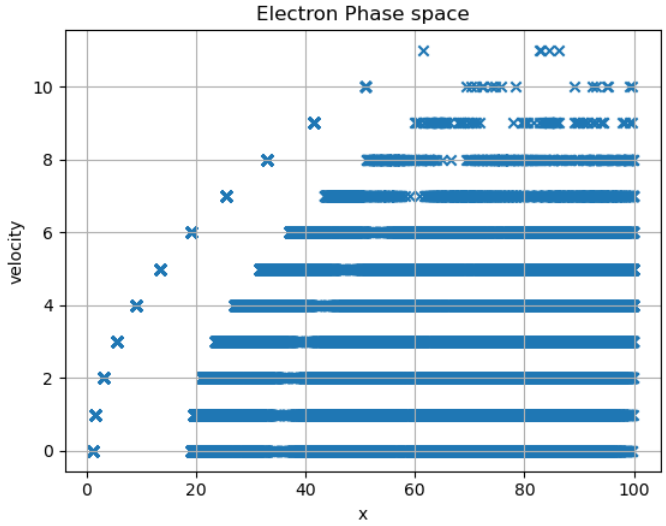

Figure 7: Electron phase space

This time in the Light Intensity plot, we can observe some *dark spaces* in between $x = 30$ and $x = 40$, between $x = 50$ and $x = 60$.

4 Conclusion

In this week we used for python for simulations in time. We learnt to work with index vectors. We simulated the motion of electrons in a tubelight(1 D), and also the illumination. We tried to find the *dark places*. With the parameter values $u0 = 7$ and $p = 0.75$, we got some dark places. In this the tubelight is dark in more of the initial region i.e. till around $x = 20$ there is no illumination.