Assignment-6 : Tube-Light Simulation

Sai Gautham Ravipati - EE19B053

April 10, 2021

Abstract

A tube-light model is simulated using Python, although the simulation is fairly complex it is done in a few lines of code using Python. Insights are drawn on the results of the simulation which include analysis of intensity of emitted photons with position, dependence of the gas filled and several other properties which shall be discussed in detail. A tube-light is being modelled as a 1D in the rest of the analysis.

1 Introduction

In a tube-light, electrons that are emitted at the cathode are considered to be accelerated by an uniform electric field, starting from zero energy. As they gain a certain threshold energy, the atoms move to the excited state and relaxation of these atoms results in photon emission, where relaxation is assumed to be immediate. The energy of the electron goes to zero and the cycle repeats and when these electrons reach the anode, whole of the energy is absorbed and lost. At each time-step mean number of electrons introduced at the cathode is N and the actual number of electrons is taken as the integer part of a normally distributed random variable with standard deviation of σ.

2 Input Arguments

The input arguments to the code are :

- n (Spatial Grid Size) :
  Passed as : '–n value', default value is 100.
- M (Number of $\bar{e}$ injected per turn) :
  Passed as : '–M value', default value is 5.
- p (Probability that ionization will occur) :
  Passed as : '–R value', default value is 0.25.
- nk (Number of turns to simulate) :
  Passed as : '–nk value', default values is 500.
- u0 (Threshold velocity) :
  Passed as : '–u0 value', default values is 5.

- Msig (Standard deviation of $\bar{e}$ distribution) :
  Passed as : '–Msig value', default value is 2.

The following lines of code does the work of parsing the input arguments :

```
def pf(x):
  try:
    p = float(x)
  except ValueError:
    raise argparse.ArgumentTypeError("%r not a floating-point literal" % x)
  if p < 0.0 or p > 1.0:
    raise argparse.ArgumentTypeError("P %r not in range [0.0, 1.0]" %  x)
  return p

def nat(x):
  try:
    ix = int(x)
  except ValueError:
    raise argparse.ArgumentTypeError("%r not an integer literal" % x)
  if ix <= 0:
    raise argparse.ArgumentTypeError("%s is an invalid positive integer value,
   please enter a postive integer value" % x)
  return ix

def sig(x):
  try:
    s = float(x)
  except ValueError:
    raise argparse.ArgumentTypeError("%r not an integer literal" % x)
  if s <= 0:
    raise argparse.ArgumentTypeError("%s is an invalid positive value, please
   enter a postive integer value" % x)
  return s

parser = argparse.ArgumentParser()
parser.add_argument("--n", metavar = 'n_val', type = nat, default = 100, help =
   'Spatial grid size')
parser.add_argument("--M", metavar = 'M_val', type = nat, default = 5,   help =
   'Number of electrons injected per turn')
parser.add_argument("--p", metavar = 'p_val', type = pf,  default = 0.25, help =
    'Probability that ionization will occur')
parser.add_argument("--nk", metavar = 'nk_val', type = nat, default = 500, help
   ='Number of turns to simulate')
parser.add_argument("--u0", metavar = 'v_thsh', type = sig, default = 5,   help
   = 'Threshold velocity')
parser.add_argument("--Msig", metavar = 'sig_val', type = sig, default = 2, help
    = 'Standard deviation of Electron distribution')
```

The function pf(x) checks that the probability entered is in range [0,1], while nat(x) and sig(x) assert that positive integer and float type inputs are given respectively. The bottom lines parse the input arguments given on command line.

3 1D Simulation

For the purpose of performing simulation, the following code is considered where update step is being performed every turn.

```
def TL_sim(n,M,nk,u0,p,Msig) :
  """
  Input arguments as those parsed previously and returns the populated array
   I,V,X along with xx,dx,u.
  """
  xx = np.zeros(n*M)    #Initializing e- postion array with zeros of size nM
  u  = np.zeros(n*M)    #Electron Velocity
  dx = np.zeros(n*M)    #Electron Displacement in current turn

  I = []    #Holds Intensity of Emitted Light
  X = []    #Holds Electron Position
  V = []    #Holds Electron Velocity

  ii = np.where(xx > 0)    #Finding the locations of e- present in the chamber

  for k in tqdm(range(0, nk), desc = "Update step in Progress") :

    """
    The following block of code runs iteratively and updates the variables in
    each turn.
    """
    dx[ii] = u[ii] + 0.5    #Updating the displacement in each turn
    xx[ii] += dx[ii]        #Advancing e- position in each turn
    u[ii]  += 1             #Advancing e- velocity in each turn

    jj = np.where(xx >= n)    #Determing the positions of e- hitting the anode
    xx[jj] = 0                #Setting position to zero
    u[jj]  = 0                #Setting velocity to zero
    dx[jj] = 0                #Setting displacement to zero

    kk = np.where(u >= u0)[0]    #Indices correspoding to the ionised e-
    ll = np.where(np.random.rand(len(kk))<=p)
    kl = kk[ll]         #Indices of energetic e- that suffer a collision

    #Setting the velocity of these e- to zero assuming inelastic collision
    u[kl] = 0

    #Updating xx array, assuming location of collision randomly between previous
     and current locations
    xx[kl] -= dx[kl]*np.random.rand(len(kl))

    I.extend(xx[kl].tolist())    #Adding photon intenities to the I vector

    m  = int(np.random.randn()*Msig+M)    #Injecting new e-
    mm = np.where(xx == 0)                #Finding the unsued indices

    #Finding the lower among number of e- and number of slots availbale
    min_idx = min(len(mm[0]),m)

    xx[mm[0][:min_idx]] = 1       #Setting the position to 1 for these e-
    u[mm[0][:min_idx]]  = 0       #Setting the velocity of these e- to 0

    ii = np.where(xx > 0)         #Finding the indices of e- in the chamber
    X.extend(xx[ii].tolist())     #Populating X,V in each turn
    V.extend(u[ii].tolist())

  return I,X,V,xx,u,dx
```

Throughout the definition, np.where has been used to find out indices satisfying certain properties. The implementation follows parallely to the text given in notes, further to optimise the code, locations of the $\bar{e}$ in the chamber are found only once as shown above in line 51 and the same is used in the next update step. Further the energised $\bar{e}$ are found out using line 25 and constrained by a probability distribution in line 31, give the indices of $\bar{e}$ that suffer collision. Further to add, all the updates are done using vectorised implementation instead of using a for loop.

4 Histogram and Phase Space

The following lines of code has been used to plot the histograms as well as the $\bar{e}$ phase-space curve using the populated I,V,X arrays obtained after performing the update step using the previously defined function call.

```
"""
Code block to plot e- density histogram, plot is saved in directory EE2703_ASN6
"""
plt.figure(0)
plt.hist(X,n,[0,n],edgecolor = 'black')
plt.title("Electron Density",fontsize = 14)
plt.ylabel("Electron Density", fontsize = 12)
plt.xlabel("position", fontsize = 12)
plt.savefig('EE2703_ASN6/fig1.png')

"""
Code block to plot Intensity histogram
"""
plt.figure(1)
count,bins,_ = plt.hist(I,n,[0,n],edgecolor = 'black')
plt.title("Intensity of emitted light",fontsize = 14)
plt.ylabel("I", fontsize = 12)
plt.xlabel("position", fontsize = 12)
plt.savefig('EE2703_ASN6/fig2.png')

"""
Code block to plot e- phase space
"""
plt.figure(2)
plt.plot(X,V,'b.')
plt.title("Electron Phase space",fontsize = 14)
plt.xlabel("position", fontsize = 12)
plt.ylabel("velocity", fontsize = 12)
plt.savefig('EE2703_ASN6/fig3.png')
```

All the histograms are plotted using 'plt.hist' while 'plt.plot' has been used to plot the $\bar{e}$ phase-space. For the default set of parameters the plots obtained are presented in Fig. 1,2,3. It is consistent from the intensity plot that, the intensity of emitted light is zero until certain position, and from the phase space, clearly before this position, the velocity hasn't yet crossed the threshold. After this particular peak in the intensity the emission decays, as fewer energetic electrons reach a point in space before colliding. As we move with position the peaks in the intensity histogram become more diffuse as zero energy location of different electrons is different. Since $\bar{e}$'s haven't gained sufficient energy before they reach the threshold, electron density histogram is peaked initially, further the peaks decay as we move in space, as inferred from the plot. Further the collision of $\bar{e}$'s with atoms to emit photons, is represented

by the change in the electron density plot where electron number tends to be a continuum after the position where the threshold is reached and unlike the part of the histogram before the point of reaching threshold velocities. In the phase-space plot one particular set of discrete points separated from the continuum of points, these correspond to those which don't suffer any collision along the length of the tube.

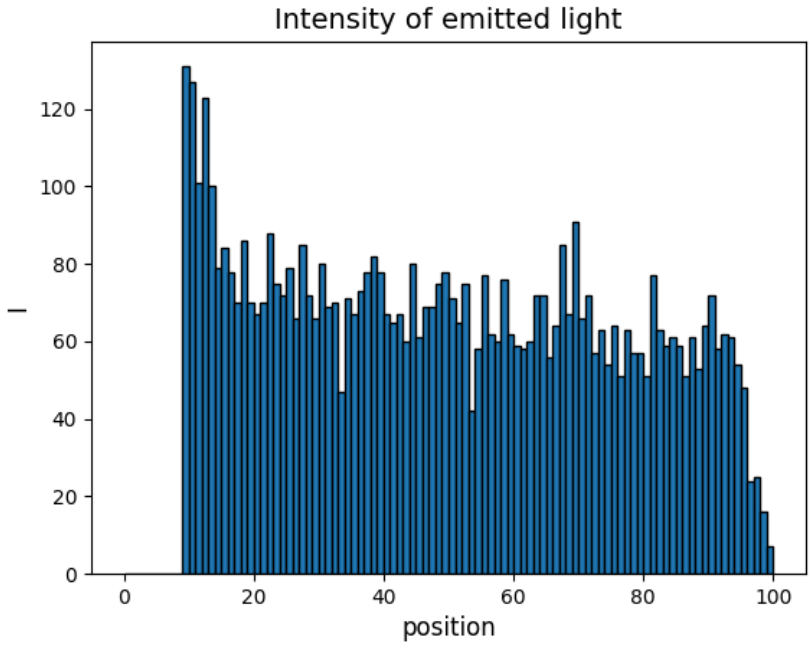

Figure 1: Intensity Histogram with position

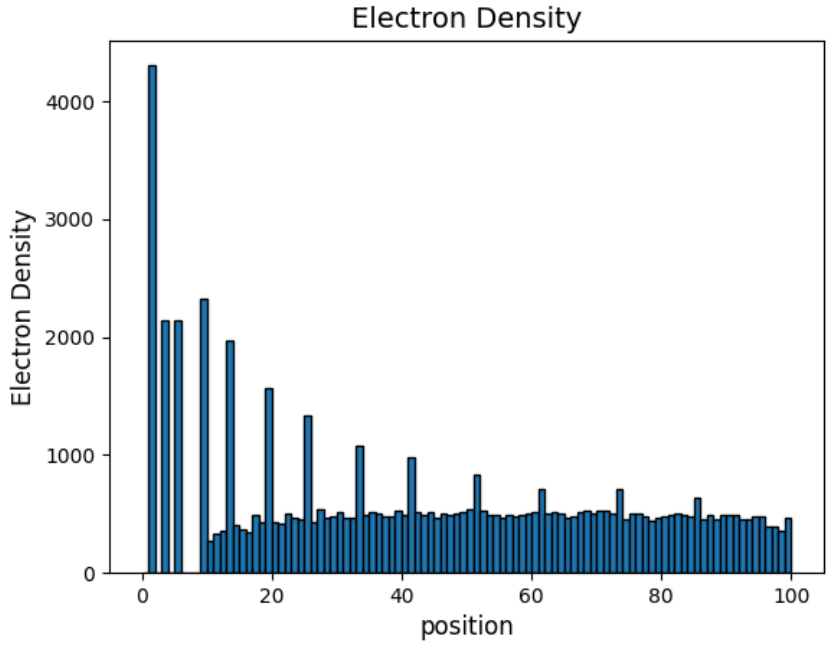

Figure 2: Electron Density with position

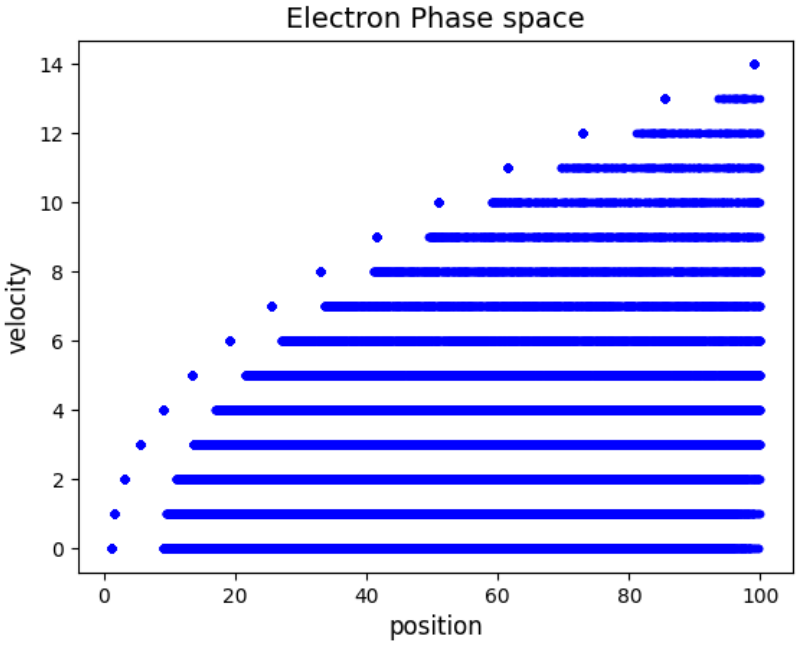

Figure 3: Electron Phase Space

5 Intensity Data

The intensity data with position is obtained using I and bins of histogram. It stored in a file in the directory same as that of plots. Previously unpacked bins from the intensity histogram are operated vectorially to get the mid points of the bins, and tabulate module has been used to present the data in a structured manner. The output for the default set of inputs is presented.

```
Intensity Data :

|   xpos |   count |       |   17.5 |      70 |       |   36.5 |      73 |
|--------+---------|       |   18.5 |      86 |       |   37.5 |      78 |
|    0.5 |       0 |       |   19.5 |      70 |       |   38.5 |      82 |
|    1.5 |       0 |       |   20.5 |      67 |       |   39.5 |      78 |
|    2.5 |       0 |       |   21.5 |      70 |       |   40.5 |      67 |
|    3.5 |       0 |       |   22.5 |      88 |       |   41.5 |      65 |
|    4.5 |       0 |       |   23.5 |      75 |       |   42.5 |      67 |
|    5.5 |       0 |       |   24.5 |      72 |       |   43.5 |      60 |
|    6.5 |       0 |       |   25.5 |      79 |       |   44.5 |      80 |
|    7.5 |       0 |       |   26.5 |      66 |       |   45.5 |      61 |
|    8.5 |       0 |       |   27.5 |      85 |       |   46.5 |      69 |
|    9.5 |     131 |       |   28.5 |      72 |       |   47.5 |      69 |
|   10.5 |     127 |       |   29.5 |      66 |       |   48.5 |      75 |
|   11.5 |     101 |       |   30.5 |      80 |       |   49.5 |      78 |
|   12.5 |     123 |       |   31.5 |      69 |       |   50.5 |      71 |
|   13.5 |     100 |       |   32.5 |      70 |       |   51.5 |      65 |
|   14.5 |      79 |       |   33.5 |      47 |       |   52.5 |      75 |
|   15.5 |      84 |       |   34.5 |      71 |       |   53.5 |      42 |
|   16.5 |      78 |       |   35.5 |      67 |       |   54.5 |      58 |
```

	55.5	77		70.5	66		85.5	59
	56.5	62		71.5	72		86.5	51
	57.5	60		72.5	57		87.5	61
	58.5	76		73.5	63		88.5	53
	59.5	62		74.5	54		89.5	64
	60.5	59		75.5	64		90.5	72
	61.5	58		76.5	51		91.5	58
	62.5	60		77.5	63		92.5	62
	63.5	72		78.5	57		93.5	61
	64.5	72		79.5	57		94.5	54
	65.5	56		80.5	51		95.5	48
	66.5	64		81.5	77		96.5	24
	67.5	85		82.5	63		97.5	25
	68.5	67		83.5	59		98.5	16
	69.5	91		84.5	61		99.5	7

6 Dependence on Parameters

The simulation is run for a different set of parameters, and effect of parameters is inferred through the plots obtained, primarily the effect of increased threshold velocity, probability of collision.

- n: 100, M: 5, p: 0.25, nk: 500, n0: 10, Msig: 2

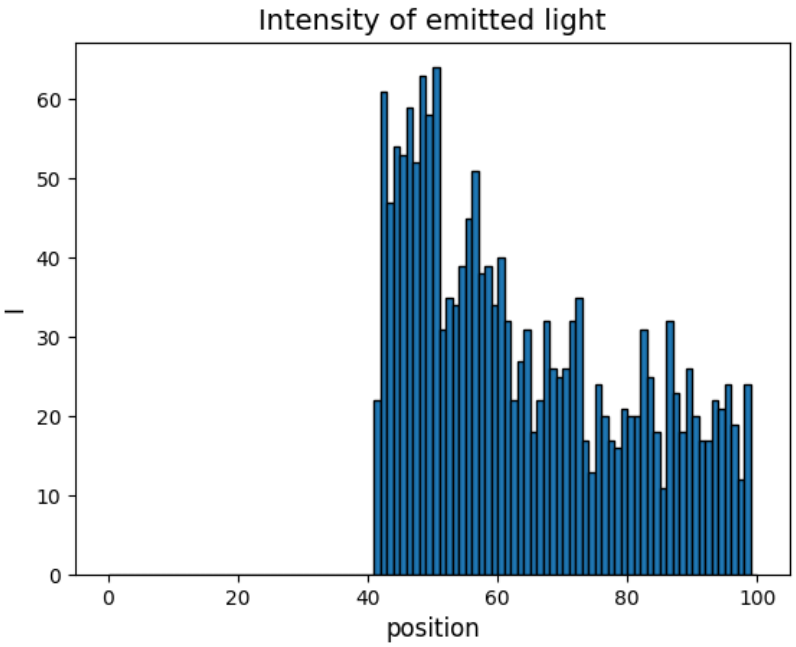

Figure 4: Intensity Histogram with position for n0(10)

Clearly from the plots, since the threshold velocity was double that of the default case, the photon emission occurs at a greater distance in space as a result of which intensity

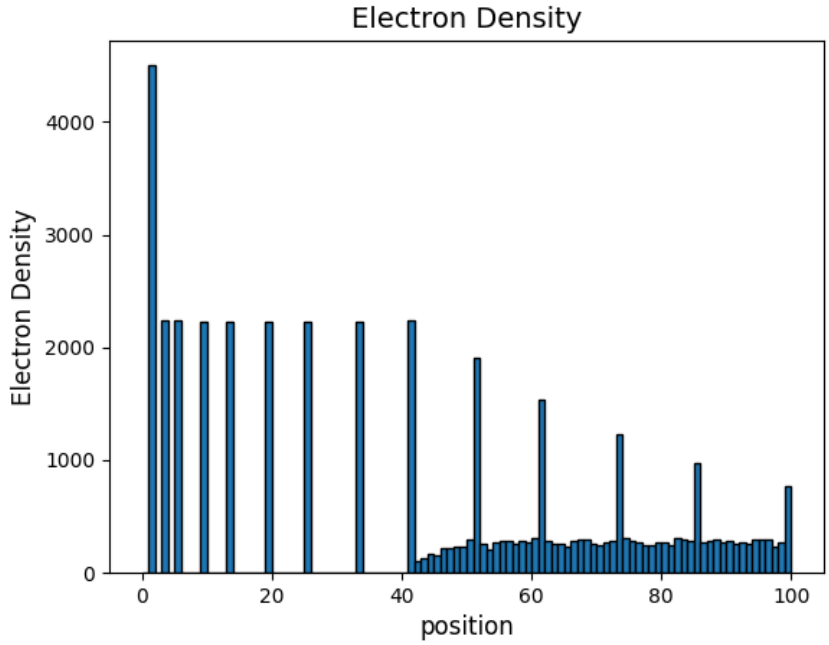

Figure 5: Electron Density with position for n0(10)

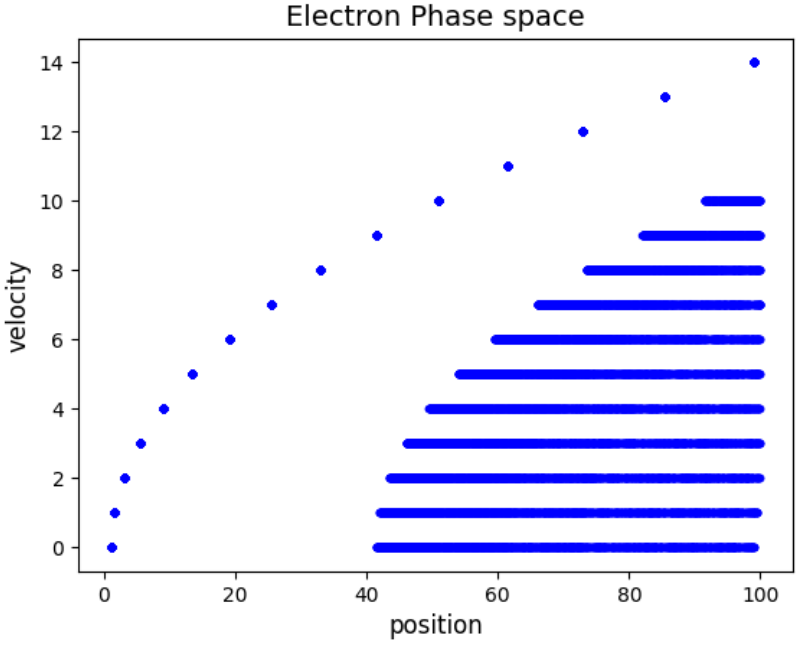

Figure 6: Electron Phase Space for n0(10)

of emitted light is zero till 40. Further to add when compared to the previous case, the height of the bins vary rapidly at a higher threshold velocity when compared to that of a lower one. Similar results apply to electron density as well, where the number is almost constant and electrons are crowded at specific locations before 40, while they tend to diffuse out once the electrons attain threshold velocity, as collision with atoms takes place. The phase trajectory of those electrons which don't suffer any collision remains

the same while the rest change.

- n: 100, M: 5, p: 0.75, nk: 500, n0: 5, Msig: 2

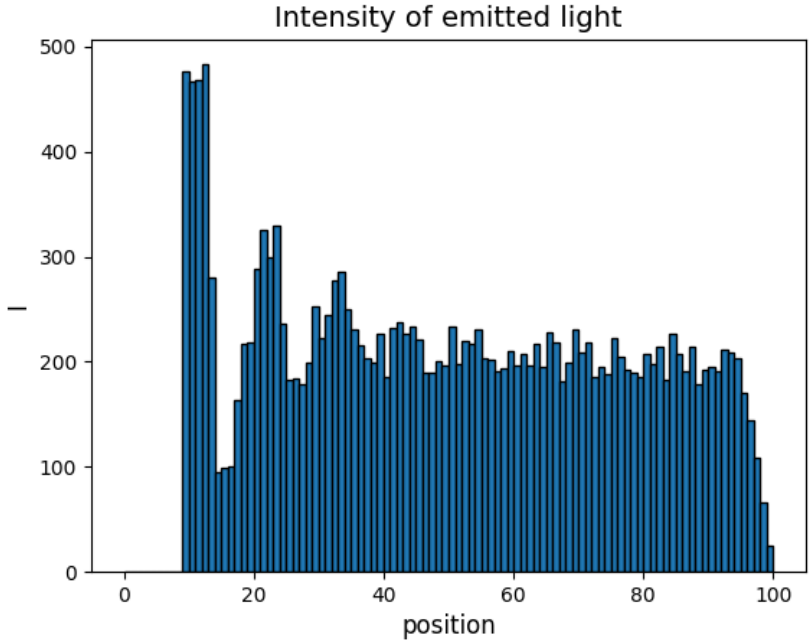

Figure 7: Intensity Histogram with position for p(0.75)

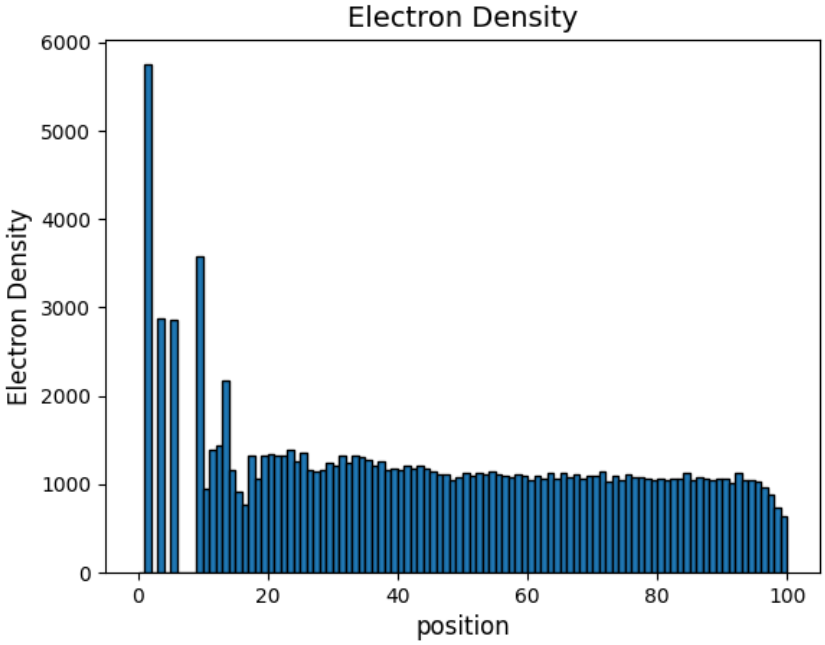

Figure 8: Electron Density with position for p(0.75)

Clearly the magnitude of electron intensity is high in case of greater probability of ionization as inferred from the intensity plot. When compared to the default parameters,

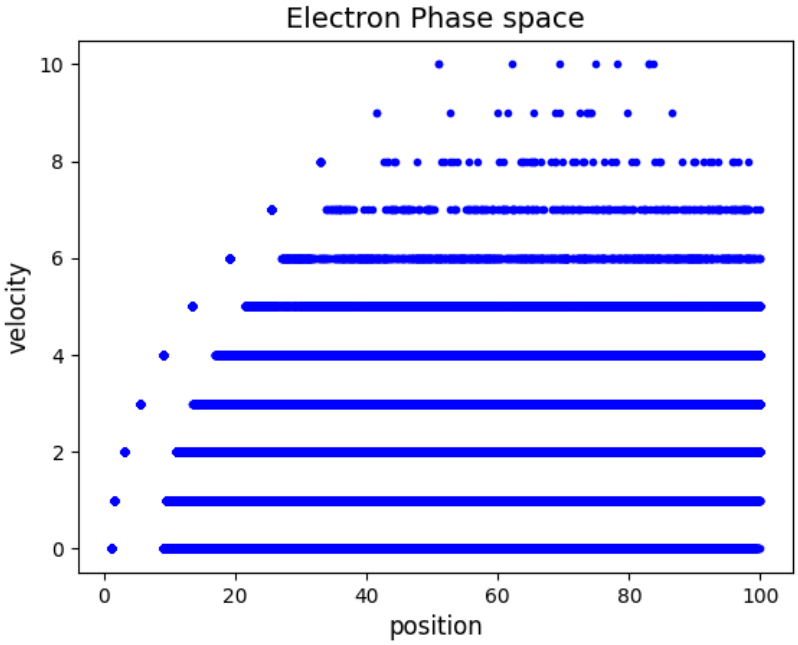

Figure 9: Electron Phase Space for p(0.75)

around at location 17 the intensity is lower when compared to the neighbourhood values, this can be treated as a dark space, and if the probability is increased further number of such regions increase further. When electron density plots are compared in the latter case, the peaks in this curve decay faster than in the case of former.

7 Conclusion

A 1D-simulation of tube-light has been demonstrated, where the majority work was done by the function to update the variables. The histograms and the phase space curves provide great insights about the nature of the histograms, peaks in the histogram, and their relation with threshold velocity. The intensity is zero till the electrons attain the threshold velocity as inferred from the plots. Variation of plots with parameters has been analysed, where higher probability of collision is desirable as it leads to higher intensities, while lower threshold velocities are desirable as the intensity curve is less rapid when compared to the case of higher threshold. Dark-spaces are low intensity regions in the neighbourhood of high intensity regions, and they dominate in high probability of ionisation case. Although the simulation is fairly complex, is done using few lines and provided several insights.