EE2703 Applied Programming Lab
Assignment 6

Name: Neham Hitesh Jain
Roll Number: EE19B084

April 11, 2021

Abstract

In this assignment, we model a tubelight in which electrons are continually injected at the cathode and accelerated towards the anode by a constant electric field. Electrons with velocity greater than a threshold can ionize atoms, which will released a photon. The tubelight is simulated for a certain number of timesteps. We look at effects of varying parameters. The tubelight is simulated for a certain number of timesteps from an initial state of having no electrons. The results obtained are plotted and studied.

1 Simulation code

We inject a variable number of electrons into the tubelight at each timestep. Number of electrons follow a normal distribution of mean M and variance $Msig$, these electrons are accelerated through the tubelight and if the velocity of a particular electron is greater than a threshold then, it can collide and cause ionization. We have two main cases:

First Case: No Collision

The displacement and velocity of electron after one time step would be:

$$dx = u_i + 0.5$$

$$x_{i+1} = x_i + dx$$

$$u_{i+1} = u_i + 1$$

Second Case: Collision

For finding electrons which collide, we first find the electrons with velocity greater than threshold and then select electrons with a given probability which take part in collision.

1.1 Using a more physically accurate model for ionized electrons

Considering the fact that an electron will collide after a dt amount of time, and then accelerate after its collision for the remaining time period, we need to perform a more accurate update step. This is done by taking time as the uniformly distributed random variable. Say dt is a uniformly distributed random variable between 0 and 1. Then, the electron would have traveled an actual distance of dx' given by

$$dx'_i = u_i * dt + \frac{1}{2}dt^2$$

as opposed to $dx_i = u_i + 0.5$

We update the positions of collisions using this displacement instead. We also consider that the electrons accelerate after the collision for the remaining $1 - dt$ period of time. We get the following equations for position and velocity updates:

$$dx_i'' = \frac{1}{2}(1 - dt)^2$$

$$u_{i+1} = 1 - dt$$

With the following update rule:

$$xx_{i+1} = xx_i + dx_i' + dx_i''$$

```
def update_displacement_for_collision(self,kl):
    '''Revised model for calculating displacement of collided
        particles'''

    time=np.random.rand(len(kl))
    prev_pos=self.xx[kl]-self.displacement[kl]
    displacement=self.velocity[kl]*time+0.5*time*time
    displacement=displacement+0.5*(1-time)**2
    self.xx[kl]=prev_pos+displacement
    self.velocity[kl]=1-time

def update_timestep(self):
    ''' Things to do at every timestep'''

    #Calculate positions where particle is present
    ii=np.where(self.xx>0)[0]
    self.update_attributes(ii)

    #Checking particles which have reached the end of tubelight and
        applying the end conditions
    end_particles=np.where(self.xx>self.size)[0]
    self.end_conditions(end_particles)

    #Checking for particles which have crossed threshold velocity
    kk=np.where(self.velocity>=self.threshold_velocity)[0]

    #Checking particles that have ionized
    ll=np.where(np.random.rand(len(kk))<=self.probability)[0]
    kl=kk[ll]

    #Applying conditions for electrons which have ionized
    self.update_displacement_for_collision(kl)
```

```
    #Storing locations where photon will be emitted
    self.I.extend(self.xx[kl].tolist())

    #Injecting electrons in the tubelight
    free_index=self.injection()
    new_ii=np.concatenate((ii,free_index))

    #Storing positions of the electrons
    self.X.extend(self.xx[new_ii])
    self.V.extend(self.velocity[new_ii])
```

2 Plotting Graphs and tabulating results

2.1 Modular Code

A modular class to plot the required graphs is written below:

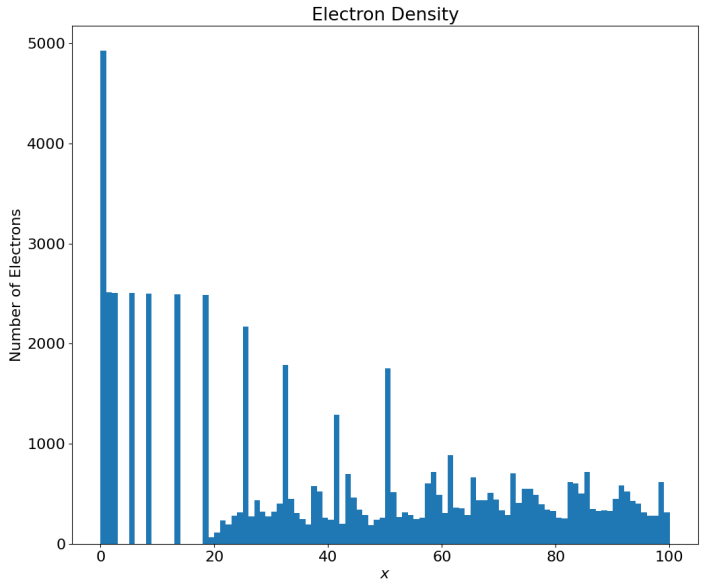
```
class General_Plotter():
''' Class used for plotting different plots. Shortens the code by quite
    a bit'''

    def __init__(self,xlabel,ylabel,title,fig_num=0):
        ''' xlabel,ylabel,title are used in every graph'''

        self.xlabel=xlabel
        self.ylabel=ylabel
        self.title=title
        self.fig=plt.figure(fig_num,figsize=(12,10),dpi=100)
        # Increase font size
        matplotlib.rcParams['font.size'] = 16


    def general_funcs(self,ax):
        ''' General functions for every graph'''

        ax.set_ylabel(self.ylabel)
        ax.set_xlabel(self.xlabel)
        ax.set_title(self.title)
        #self.fig.show()
        self.fig.savefig("plots/"+self.title+".png")

    def population_plot(self,values):
        ''' Plotting histograms'''

        axes=self.fig.add_subplot(111)
        axes.hist(values,bins=100)
        self.general_funcs(axes)
```

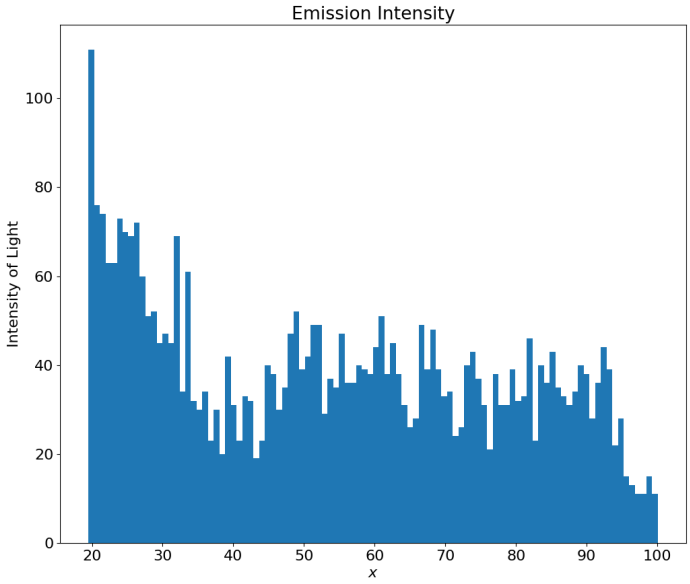
```
def scatter_plot(self,X,Y):
    ''' Plotting the phase plots'''

    axes=self.fig.add_subplot(111)
    axes.scatter(X,Y,c='r',marker='X')
    self.general_funcs(axes)
```

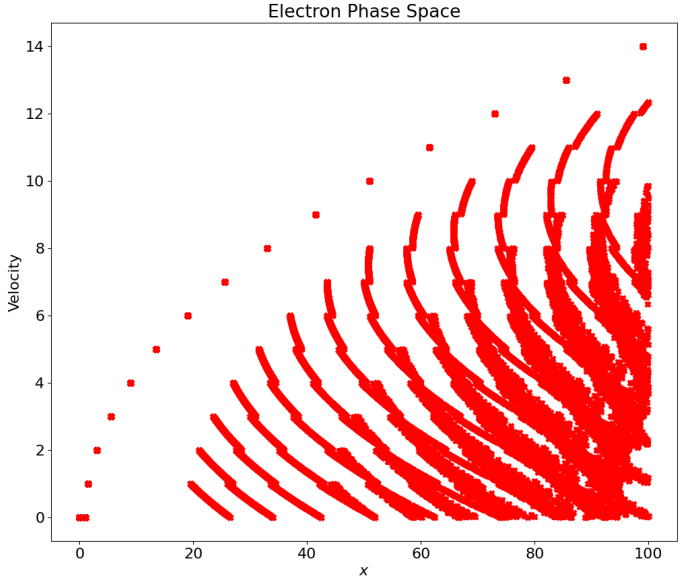
2.2 Graphs

The list of parameters are,
M – Number of electrons injected per turn.
n – Spatial grid size.
n_k – Number of turns to simulate.
u_0 – Threshold velocity.
p – Probability that ionization will occur
Msig – Taking sigma of normal distribution

For, the parameters value M=5 , n=100, n_k=500, u_0=7, p=0.25, Msig=1 the graphs are




Emission Intensity
Intensity of Light
x
0
20
40
60
80
100
20
30
40
50
60
70
80
90
100


Electron Phase Space
Velocity
x
0
2
4
6
8
10
12
14
0
20
40
60
80
100

2.3 Tabulating Results

We create a dataframe using Pandas library and then write the results into a txt file.

```
# Tabulating data for intensity vs position
def tabulate(self,filename="tabulated_data.txt"):
'''Tabulating data for intensity vs position'''
    bins = plt.hist(self.I,bins=np.arange(1,self.size,self.size/100))[1]
        # Bin positions are obtained
    count =
        plt.hist(self.I,bins=np.arange(1,self.size,self.size/100))[0] #
        Population counts obtained

    xpos = 0.5*(bins[0:-1] + bins[1:]) # As no. of end-points of bins
        would be 1 more than actual no. of bins, the mean of bin
        end-points are used to get population of count a particular bin
    df = pd.DataFrame() # A pandas dataframe is initialized to do the
        tabular plotting of values.
    df['Xpos'] = xpos
    df['count'] = count

    base_filename = filename
    with open(base_filename,'w') as outfile:
        df.to_string(outfile) #Writing the Pandas Dataframe to txt file
```

The counts at various positions of x are,

	Xpos	count
0	1.5	0.0
1	2.5	0.0
2	3.5	0.0
3	4.5	0.0
4	5.5	0.0
5	6.5	0.0
6	7.5	0.0
7	8.5	0.0
8	9.5	0.0
9	10.5	0.0
10	11.5	0.0
11	12.5	0.0
12	13.5	0.0
13	14.5	0.0
14	15.5	0.0
15	16.5	0.0
16	17.5	0.0
17	18.5	0.0

18	19.5	106.0
19	20.5	207.0
20	21.5	195.0
21	22.5	166.0
22	23.5	182.0
23	24.5	198.0
24	25.5	153.0
25	26.5	180.0
26	27.5	74.0
27	28.5	68.0
28	29.5	71.0
29	30.5	73.0
30	31.5	90.0
31	32.5	56.0
32	33.5	97.0
33	34.5	32.0
34	35.5	40.0
35	36.5	30.0
36	37.5	34.0
37	38.5	27.0
38	39.5	42.0
39	40.5	27.0
40	41.5	28.0
41	42.5	38.0
42	43.5	16.0
43	44.5	70.0
44	45.5	103.0
45	46.5	105.0
46	47.5	107.0
47	48.5	96.0
48	49.5	110.0
49	50.5	93.0
50	51.5	130.0
51	52.5	78.0
52	53.5	107.0
53	54.5	70.0
54	55.5	84.0
55	56.5	88.0
56	57.5	83.0
57	58.5	104.0
58	59.5	74.0
59	60.5	62.0
60	61.5	60.0
61	62.5	59.0
62	63.5	64.0
63	64.5	47.0

```
64  65.5   60.0
65  66.5   56.0
66  67.5   47.0
67  68.5   43.0
68  69.5   73.0
69  70.5   83.0
70  71.5   86.0
71  72.5   69.0
72  73.5   66.0
73  74.5   69.0
74  75.5   63.0
75  76.5   92.0
76  77.5   99.0
77  78.5   86.0
78  79.5   82.0
79  80.5   80.0
80  81.5   71.0
81  82.5   69.0
82  83.5   85.0
83  84.5   93.0
84  85.5   65.0
85  86.5   68.0
86  87.5   76.0
87  88.5   79.0
88  89.5   70.0
89  90.5   75.0
90  91.5   66.0
91  92.5   56.0
92  93.5   36.0
93  94.5   32.0
94  95.5   37.0
95  96.5   30.0
96  97.5   20.0
97  98.5   25.0
```

3 Observations

We can make the following observations from the above plots:

- The electron density is peaked at the initial parts of the tubelight as the electrons are gaining speed here and are not above the threshold. This means that the peaks are the positions of the electrons at the first few timesteps they experience.

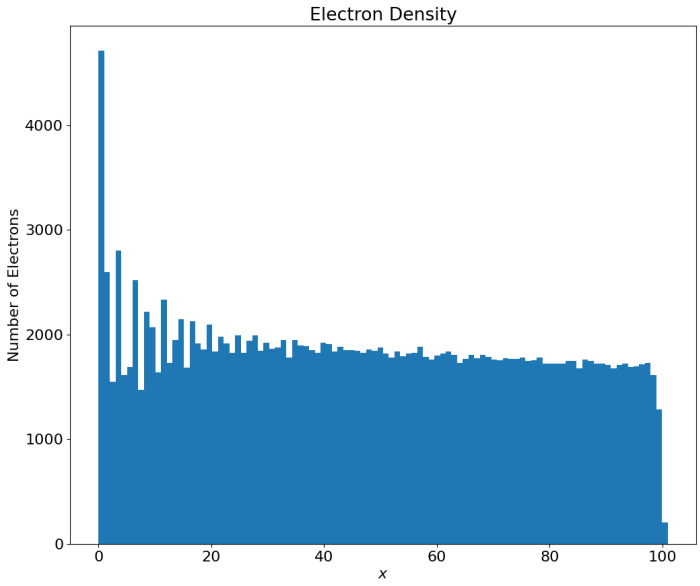
- The peaks slowly smoothen out as x increases beyond 19. This is because the electrons achieve a threshold speed of 7 only after traversing a distance of 19 units. This means that they start ionizing the gas atoms and lose their speed due to an inelastic collision.

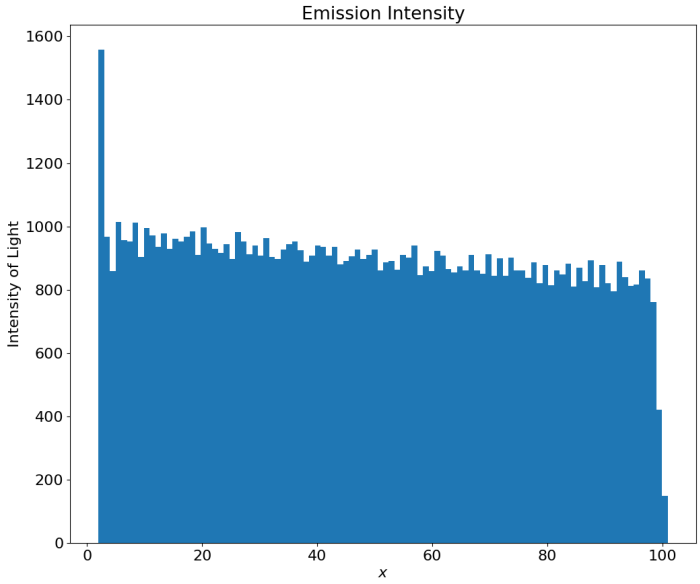
- The emission intensity also shows peaks which get diffused as x increases. This is due the same reason as above. Most of the electrons reach the threshold at roughly the same positions, leading to peaks in the number of photons emitted there.

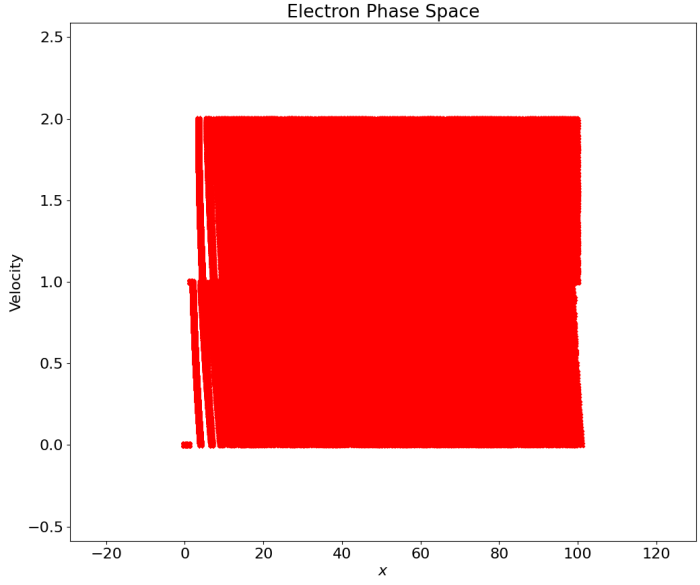
4 Varying parameters

The simulation is repeated using a different set of parameters. Namely, the threshold velocity is greatly reduced to $u_0 = 2$, and the ionization probability is increased to 1. These parameters are quite unrealistic.

Now, the graphs change as follows,




Emission Intensity
Intensity of Light
x
0
200
400
600
800
1000
1200
1400
1600
0
20
40
60
80
100


Electron Phase Space
Velocity
x
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−20
0
20
40
60
80
100
120

5 Conclusion

We can make the following conclusions from the above sets of plots:

- Since the threshold speed is much lower in the second set of parameters, photon emission starts occuring from a much lower value of x. This means that the electron density is more evenly spread out. It also means that the emission intensity is very smooth, and the emission peaks are very diffused.
- Since the probability of ionization is very high, total emission intensity is also relatively higher compared to the first case.
- We can conclude from the above observations that a gas which has a lower threshold velocity and a higher ionization probability is better suited for use in a tubelight, as it provides more uniform and a higher amount of photon emission intensity.
- Coming to the case where the ionization probability is 1, we observe that the emission instensity consists of distinct peaks. The reason that these peaks are diffused is that we perform the actual collision at some time instant within the interval between two time steps. This also explains the slightly diffused phase plot as well.
- The families of phase space plots corresponding to the peaks is also more evident in this unrealistic case.