EE2703: Assignment 5

Tanay Dixit
EE19B123

April 10, 2021

Aim

The aim is to simulate a tubelight in 1-Dimension.

Introduction

Consider electrons being emitted by the cathode with zero energy, and accelerate in this field. When they get beyond a threshold energy, they get excited. The relaxation of these atoms results in emission of photons. In our model, we will assume that the relaxation is immediate. The electron loses all its energy and the process starts again.

Those electrons reaching the anode are absorbed and lost. Each time step, electrons are introduced at the cathode.The displacement and change in its velocity is computed and electrons that have hit the anode are identified.
We now got through the detailed simulation process.

Simulation Process

We begin by creating a environment to contain all our variables.We divide the tube into n sections. At each time instant M electrons are injected and this process is run for nk turns.All electrons that have a velocity greater than $u0$ will excited and will collide with the anode with a probability p.

```
class simulate(plot):
    '''
    Tubelight Simulation Environment
    '''
    def __init__(self,dx,x,u):
        '''

        '''
        super(simulate, self).__init__()
        self.dx  = dx # displacement at time i
        self.u  = u # e speed
        self.x = x # e position
        #I is the intensity vector , X is the Electron position , V is the
    electron Velocity vector
        self.I , self.V, self.X = [], [], []
```

where dx, x, u are the displacement in current turn , electron position and electron velocity respectively. All are of dimensions nM and are initialised with 0s.

At every iteration the information of all electrons are stored in I, V, X vectors.

Electron Identification

To identify electrons every turn we find all the positions that have the values $x = 0$ which represent an empty place. This is done using the *where* command.Note the position values must be $0 < x < L$ hence all electrons that excess L are reset to $x = 0$

```
def findElectrons(self):
    ids = np.where(self.x >0)[0]
```

Displacement Process

Due to the present of an External Electric field the electrons will accelerate(we assume with $1ms/s^2$.Note as only the electrons will move we only shift those ids which are non-zero. We advance the positions by dx computed and increment velocity by 1.

```
def displace(self):
    self.ids = np.where(self.x > 0)[0]
    self.dx[self.ids] = self.u[self.ids]  +0.5
    self.x[self.ids] += self.dx[self.ids]
    self.u[self.ids] += 1
```

Collision and Ionization Process

We identify the particles that have hit the anode by there position values being greater than n. Agiain we make use of the where command, these collided electron's positions are reset to 0.

```
def check_validity(self,n):
    return np.where(self.x > n)

def ionization(self,p, u0):
    velocity_ids = np.where(self.u >= u0)[0]
    ll =  np.where(np.random.rand(len(velocity_ids))<=p)[0]
    collision_ids = velocity_ids[ll]
    self.u[collision_ids] =0

    self.x[collision_ids] -= self.dx[collision_ids]*np.random.rand() #TODO
    improve algo
    # add photon
```

```
12      self.I.extend(self.x[collision_ids].tolist())
```

For indentifying which electrons are ionized we first find the electrons that have a velocity greater than a threshold $u0$. We create

Injection

We inject new electrons every turn based on a random number drawn from a normal distribution with means M and standard deviation sigma.To decide where to inject the electrons into the array we first identify all the empty slots by using the *where* command , if the number of electrons to be injected is less than number of slots available we randomly select n different slots form this lost and assign an electron to them, while if the number of empty slots are less we fill all the available(highly unlikely as the list is of size nM.

```
1   @staticmethod
2   def injection(M, sigma):
3       return np.random.randn()*sigma + M
4
5   def inject(self,M, sigma):
6       num_of_e = int(self.injection(M, sigma))
7       empty_slots = np.where(self.x==0)[0]
8       numOfEadded = random.sample(empty_slots.tolist(),num_of_e) if num_of_e <= len
        (empty_slots) else empty_slots
9
10      self.x[numOfEadded] = 1
11      self.u[numOfEadded] = 0
```

Iterations

Every iteration we collect all electrons positions and velocities in these arrays. If they had a collision, we also record the emitted light. Places where a non zero position is present represents an electron, rest all empty places are $x = 0$.

```
1   def run(self,args):
2       M, sigma, p ,u0 ,n = args.M , args.Msigma, args.p , args.u0, args.n
3       nk = args.nk
4       for _ in tqdm(range(nk)): #range(nk)
5           #dispalce the electrons
6           self.displace()
7
8           #check which have hit the anode
9           hit_ids = self.check_validity(n)
10          self.u[hit_ids] =0
11          self.x[hit_ids] =0
```

```
12          self.dx[hit_ids] =0
13
14          #ionize the electrons that have hit the anode
15          self.ionization(p , u0)
16
17          #inject more electrons
18          self.inject(M, sigma)
19
20          #identify elect properties
21          self.findElectrons()
```

Results and Analysis

We plot the electron density plot and light Intensity plots using the *hist* function for given default inputs of $u0$ =5 and $p = 0.25$

Tabulated Intensity Data

Position	Count
1.5	0.0
2.5	0.0
3.5	0.0
4.5	0.0
5.5	0.0
6.5	0.0
7.5	0.0
8.5	0.0
9.5	95.0
10.5	140.0
..	..
38.5	68.0
39.5	68.0
40.5	74.0
41.5	62.0
42.5	63.0
43.5	65.0
44.5	81.0
45.5	64.0
..	..
..	..
92.5	59.0
93.5	36.0
94.5	48.0
95.5	39.0
96.5	35.0
97.5	39.0
98.5	23.0

We also analyse electron phase plot by plotting electron distance v/s velocity

The intensity histogram reveals that the electrons do not cause excitation of atoms till they cross a particular threshold velocity. Secondly, this gives rise to a peak in intensity just after the first mean length. This is because a majority of electrons collide with atoms at this distance. Further subsequent peaks do exist, but have larger spread and are less prominent. We observer around 2 dark bands in this intensity profile.
Th peak intensity is proportional to $u0$ also it is clear that as p increases , the peak should increase its magnitude as more collisions occur. The electron phase plots show the constant acceleration all electrons initially undergo, and the subsequent random motion post collision. The phase plots are nearly uniformly distributed in the middle portion of the tubelight.

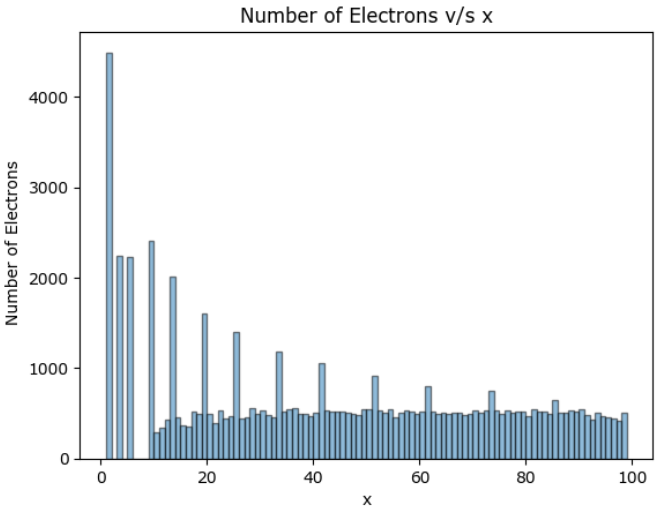

Figure 1: Electron density Plot for $u0 = 5$ and $p = 0.25$

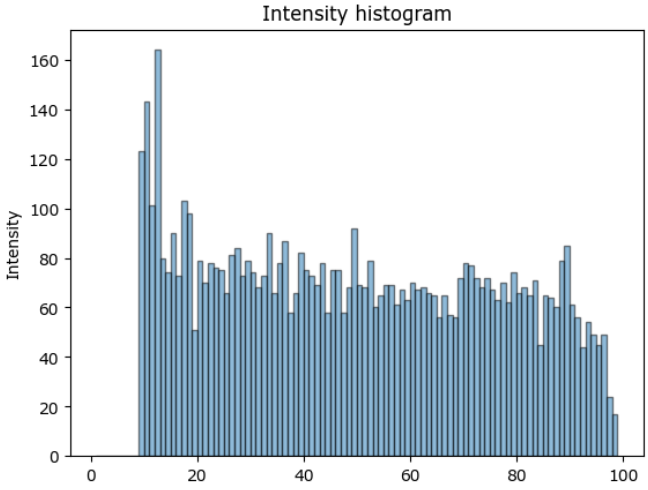

Figure 2: Intensity Plot for $u0 = 5$ and $p = 0.25$

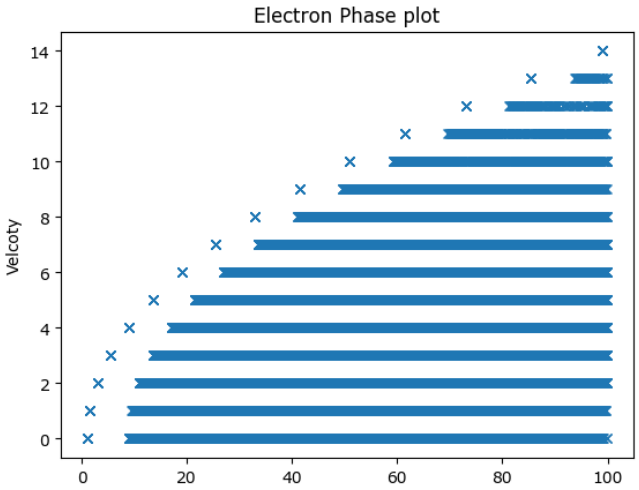

Figure 3: Phase space plot for $u0 = 5$ and $p = 0.25$

Here are the following plots for other examples
Plots for $u0$ =7, $p = 0.25$

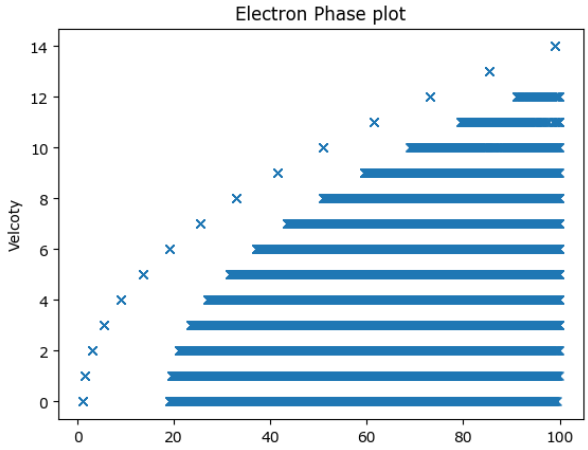

Figure 4: Phase space plot for $u0 = 7$ and $p = 0.25$

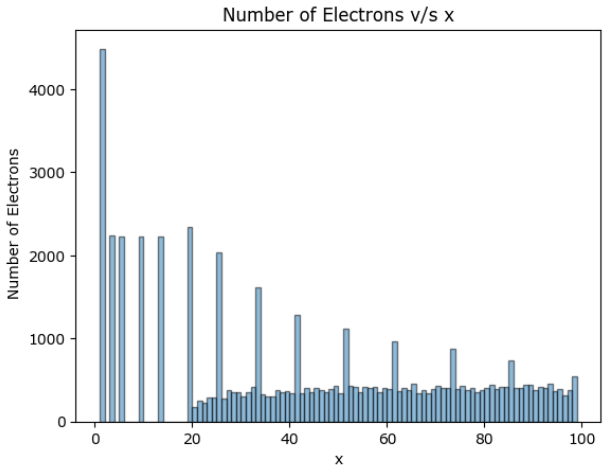

Figure 5: Phase space plot for $u0 = 7$ and $p = 0.25$

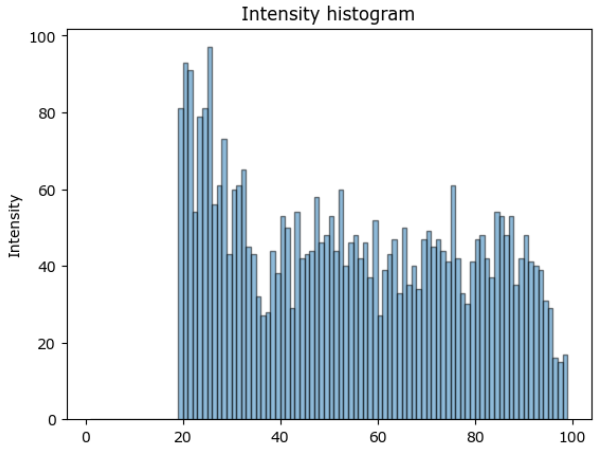

Figure 6: Phase space plot for $u0 = 7$ and $p = 0.25$

Here are the following plots for other examples Plots for $u0$ =10, $p = 1$

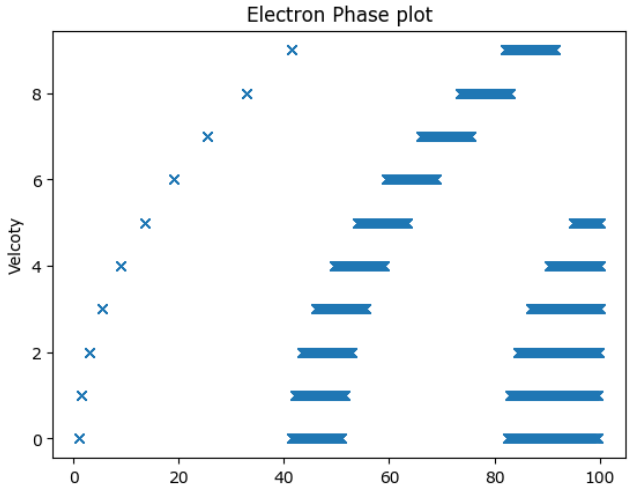

Figure 7: Phase space plot for $u0 = 10$ and $p = 1$

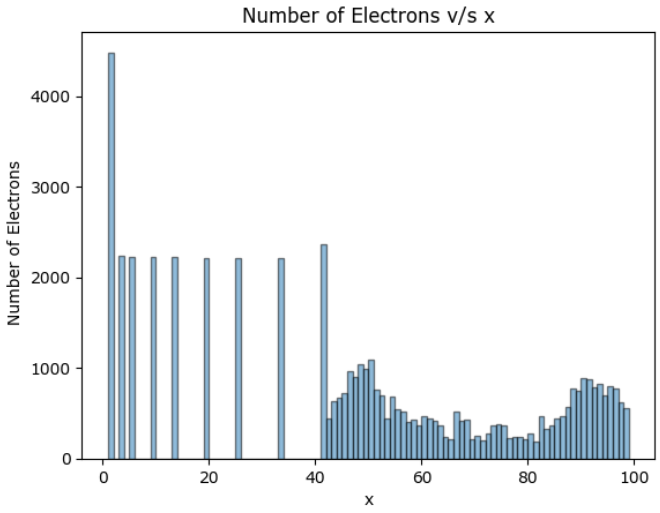

Figure 8: Phase space plot for $u0 = 10$ and $p = 1$

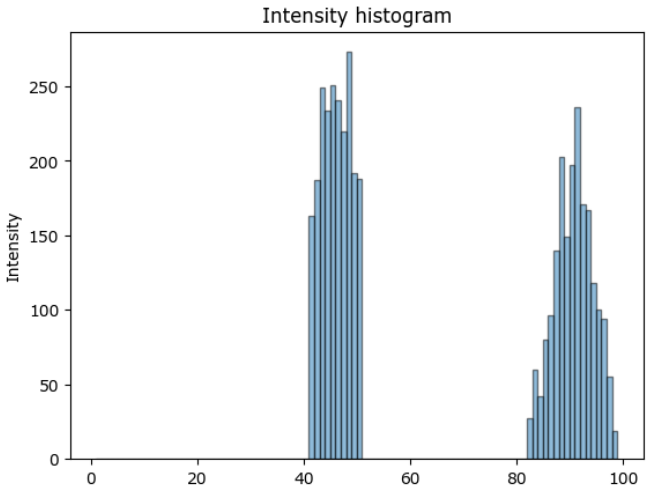

Figure 9: Phase space plot for $u0 = 10$ and $p = 1$

Conclusion

For lower threshold of speed the photon emissions starts occuring from a much lower value of x, on increasing value of p(*probability of Ionization*) the total emission intensity also increases. As a result gases which have a higher ionization probability and lower threshold velocity are more suitable to be used in a tubelight.