Tube Light Model Simulation

Mohammed Muqeeth
EE16B026
Electrical Engineering
IIT Madras

March 13, 2018

Abstract

In this report we present simulations done in python.We take tube light model as problem statement to simulate light intensity as function of position and plot electron phase space for each electron.

1 Introduction

- We consider 1D tube light with uniform electric field from anode to cathode. For each time step an average of 5 electrons are introduced at cathode.These electrons accelerate in presence of electric field.
- The accelerated electrons which when reach the threshold energy collide with atoms and excite them.The excited atoms come to ground state by emmitting photons.The collision is taken as perfectly inelastic.The relaxation time for excited atoms is taken as zero ie they emit photons as soon as collision occurs.
- The electrons reaching the anode gets destroyed.The actual number of electrons introduced for each time step is a random number which is normally distributed with standard deviation of 2 and mean 5.
- The position of collided electron is assumed to be uniformly distributed.The total number of time steps for simulation is 500.
- Simulation returns three lists namely Intensity of emitted light I,Electron position X,Electron velocity V. I, X, V record emitted light,electrons postion,electrons velocity for each time step.
- Next we plot Intensity,Electron density using *hist* function in python.

2 Method

2.1 Import Packages

- we import pylab,matplotlib,sys,numpy packages for this simulation.The python code to import them is below

```
#import packages
from pylab import *
from numpy import *
from matplotlib import *
import sys
```

2.2 User input

- We use sys.agrv to take user inputs. If user does not provide input default values defined in the program are taken.sys.agrv returns a list with arguments passed through terminal from second index.
- The variable p below is probability that collision results in ionisation of atoms for light emission.
- The standard deviation is taken as $Msig = 2$

```
#user input using sys.argv
try:
    n = int(sys.argv[1])
    M = int(sys.argv[2])
    nk = int(sys.argv[3])
    u0 = int(sys.argv[4])
    p = float(sys.argv[5])
except:
    n=100 # spatial grid size.
    M=5 # number of electrons injected per turn.
    nk=500 # number of turns to simulate.
    u0=7 # threshold velocity.
    p=0.5 # probability that ionization will occur
Msig=2#standard deviation
```

2.3 Create vectors to hold electron information.Define vectors to accumulate simulation data.

- Three vectors Electron position xx,Electron velocity u,Electron displacement in each time step dx are defined of length n*M to store information in each turn
- The information from simulation is stored in lists defined as Intensity of emitted light I,Electron position X,Electron velocity V for every time step.They are defined as lists because their size gets extended in every time step.
- The variable m1 is defined to hold number of electrons introduced till the current injection of electrons.This helps to place incoming electrons in position greater than m1 for xx list.
- The tube light length is taken to be of size 100.The position of electron from 1 to 100 represents state inside tubelight . The position 0 of electron means not injected.The position greater than or equal to 100 represents electron reaching anode and absorbed.

```
#Create vectors to hold the electron information
xx = zeros((n*M))
u = zeros((n*M))
dx = zeros((n*M))
#information from simulation
I = []#Intensity of emitted light
X = []#Electron position
V = []#Electron velocity
m1 = 0#keep track of number of electrons introduced till loop start again
```

2.4 The Main iteration loop

- Find electrons present inside tube light using *where* command.Assuming the acceleration caused by unifrom electric field as 1, the displacement of i^{th}electron is calculated using kinematic equations.
$$dx_i = u_i\triangle t + \frac{1}{2}a(\triangle t)^2 = u_i + 0.5$$
$$v_i = u_i + a\triangle t = u_i + 1$$

- The postion of electron of electron is added with $dx_i$and velocity is increased by 1.
$$x_i \leftarrow x_i + dx_i$$
$$u_i \leftarrow u_i + 1$$

- The particles whose positions are greater than or equal to 100 have hit anode their position are reset to 0.

- Since electron accelerates it may reach threshold energy.Those electrons with energy greater or equal to threshold if collided with atoms and resulted in ionisation will lose velocity to zero.

- The displacement of those collided electron of number len(kl) are calculated as:
$$dt = rand(len(kl))$$
$$dx = u_i * dt + 0.5 * dt * dt$$
$$v_i = 0 + 1 * (1 - dt)$$
$$x_i \leftarrow x_i + dx_i + 0.5 * (1 - dt) * (1 - dt)$$

- The positions where energised electrons collisions occur are added to list I.These positions would represent positions where light is emitted.

- The injection of electrons is done by a normally distributed number with mean 5 and standard deviation 2 as follows
$$m = int(randn() * Msig + M)$$

- The injected electrons are placed in xx where it has zero value.

- This is done under a for loop for 500 iterations.Again find all the existing electrons.Add their positions and velocities to the X and V vectors.

```
#loop begins
for k in range(1,nk+1):
    ii = where(xx>0)[0]
    dx[ii] = u[ii] + 0.5
    xx[ii] = xx[ii] + dx[ii]
    u[ii] = u[ii] + 1
    #set position velocity of elec which reach anode
    jj = where(xx>=n)[0]
    u[jj] = 0
    xx[jj] = 0
    dx[jj] =0
    #those elec whose energy>threshold
    kk = where(u>=u0)[0]
    ll = where(rand(len(kk))<=p)[0]
    kl = kk[ll]
```

```
        xx[kl]=xx[kl]-dx[kl]
        u[kl]=u[kl]-1
        dt = rand(len(kl))
        dx[kl] = u[kl]*dt + 0.5*dt*dt
        xx[kl] =xx[kl]+dx[kl]+0.5*(1-dt)*(1-dt)
        u[kl]=1-dt
        I.extend(xx[kl].tolist())
        m=int(randn()*Msig+M)
        pp = where(xx>0)[0]
        X.extend(xx[pp].tolist())
        V.extend(u[pp].tolist())
        m1 = where(xx==0)[0]
        y=m1[0:m]
        xx[y]=1
```

2.5 Plots and data analysis.

- we use hist function to plot Intensity of emitted light I, electron density X.
- The hist function returns a tuple of three elements. First one is an array of population count and second one is bin position.The second one gives dividing positons between bins. So to make its dimension equal to that of population count we take mid values of each bin.
- The data obtained is tabulated .
- For electron phase space we plot vector X vs vector V.This essentially plots position $x = x_i$and velocity $v = v_i$ for each electron for all time steps.

```
title('Emission Intensity')
xlabel('x')
ylabel('I')
hist(I,100,normed=1,color='white')
show()
title('Electron density')
xlabel('x')
ylabel('Number of electrons that cross x')
hist(X,100,color='white')
show()
title('Electron phase space')
xlabel('x')
ylabel('v')
plot(X,V,'ro',markersize=4)
show()
population_count=hist(I,100)[0]
bins = hist(I,100)[1]
xpos= 0.5*(bins[0:-1]+bins[1:])
for x in range(len(xpos)):
    print "%f  %d\n" %(xpos[x],population_count[x])
```

Figure 1: Emission Intensity for u0=7

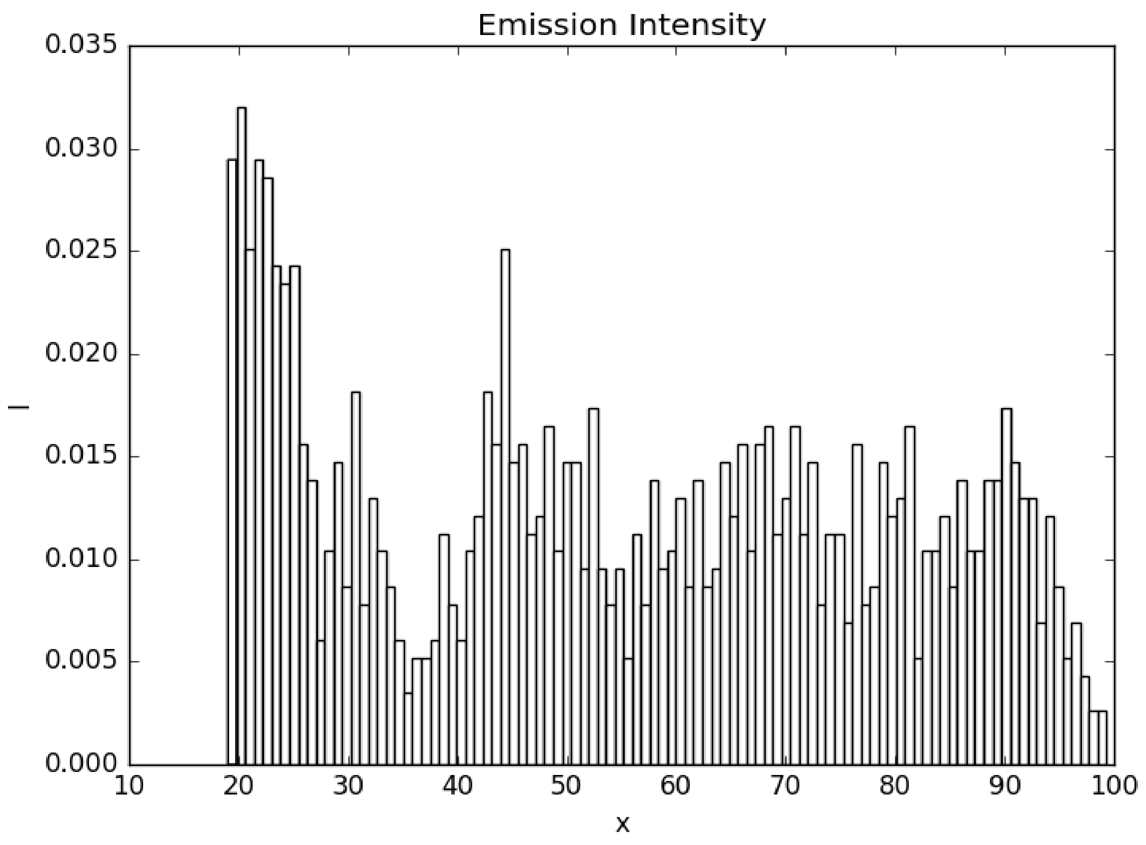


Figure 2: Electron Phase Space

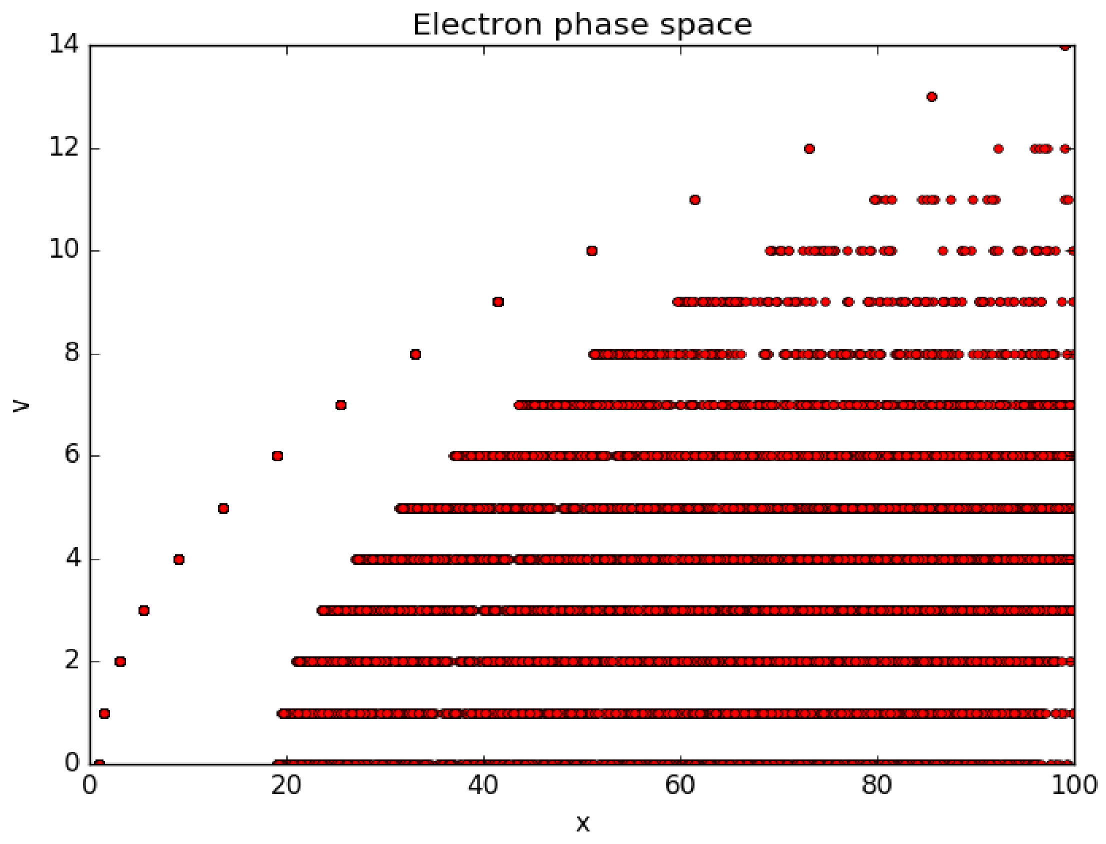


Figure 3: Emission Intensity for u0=5

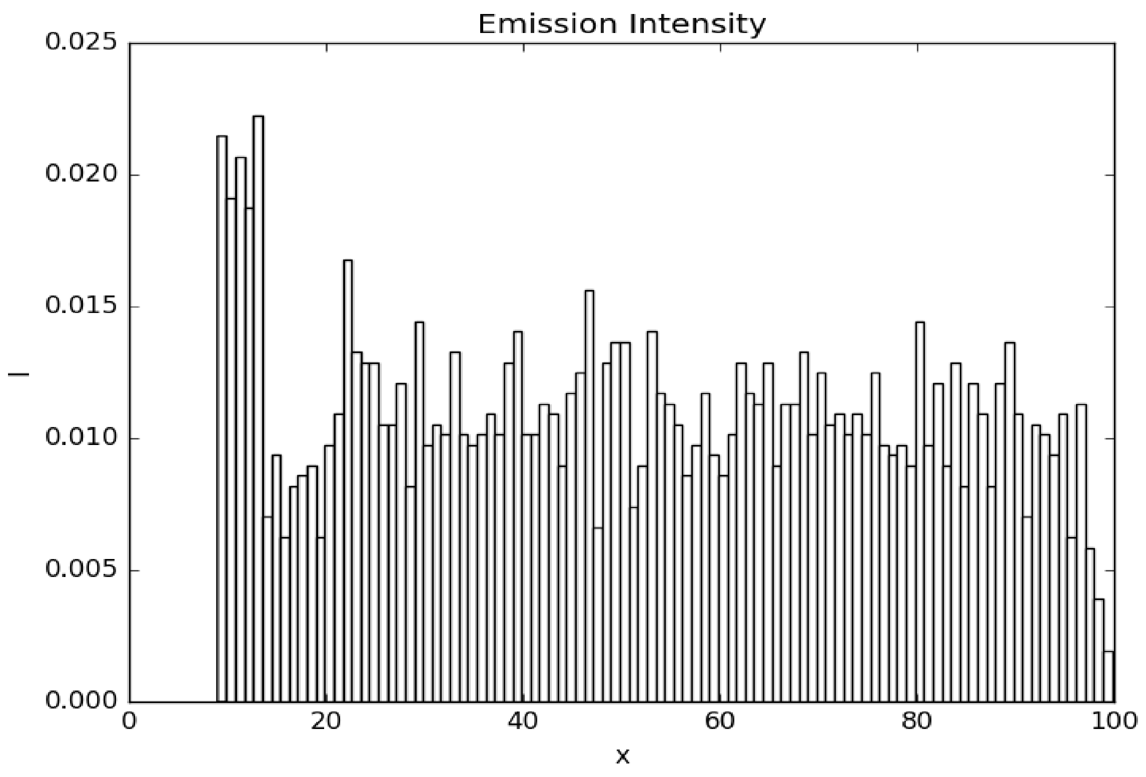
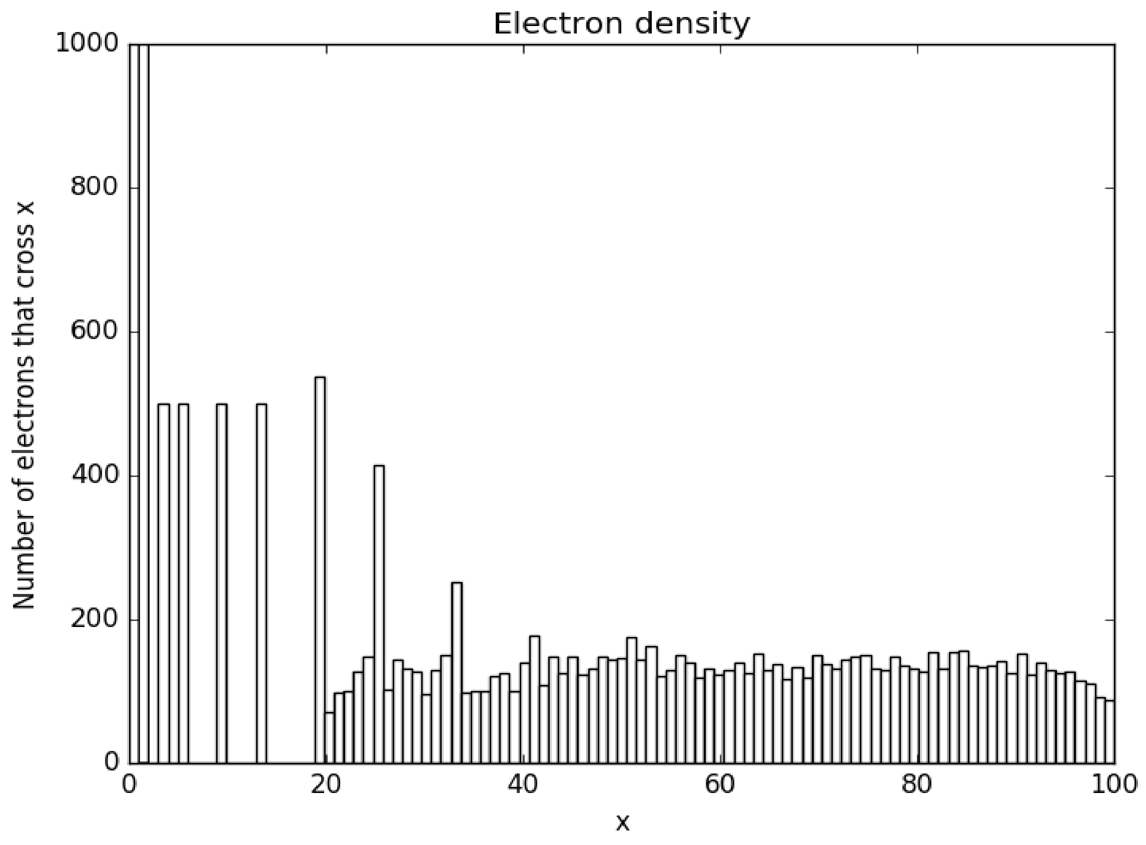


Figure 4: Electron density



3 Results and Discussion

- From figure 1, which is for u0 = 7 and p = 0.5 the conclusion are:
  - There is no emission of light before x = 20 because accelerate along this distance to reach threshold velocity.
  - The peak value of intensity is at x=20,the second peak is at x between 40 to 50.
- From figure 2,we get to know that for an x value only discrete v values are possible.
- From figure3, which is for u0=5 and p = 0.5,the conclusions are:
  - As u0 value changes which can be assumed as changing gas in tube light we can have simulations for different gases in tube light.
  - There is no emssion of light before x=8.i.e electrons travels a distance of 8 to reach threshold velocity.
  - The peak intensity occurs near x=8.Near edge of tube light intensity does vary much.
- From figure 4 we conclude that more electrons cross region near to cathode.Near anode the number of electrons passing remain uniform.
- The above discussion is for assuming electrons are uniformly distributed in space

Figure 5: Emission intensity

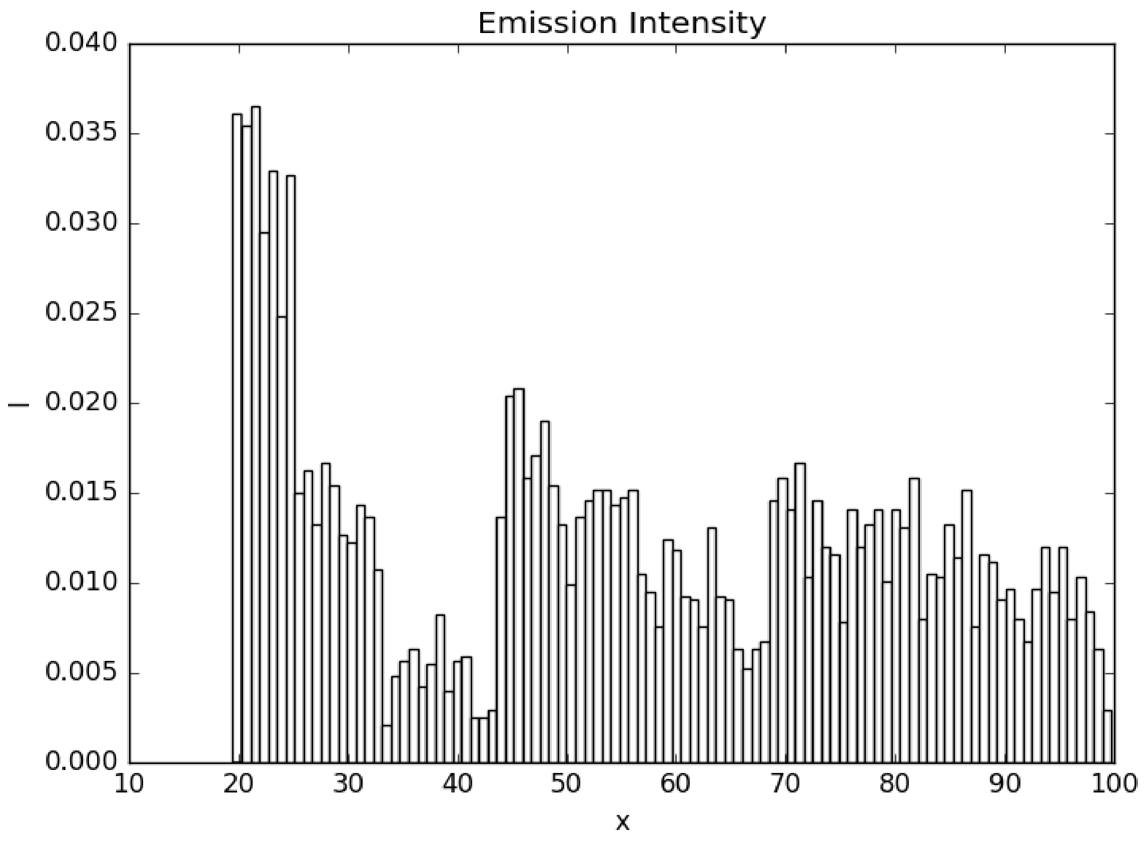


Figure 6: Electron density

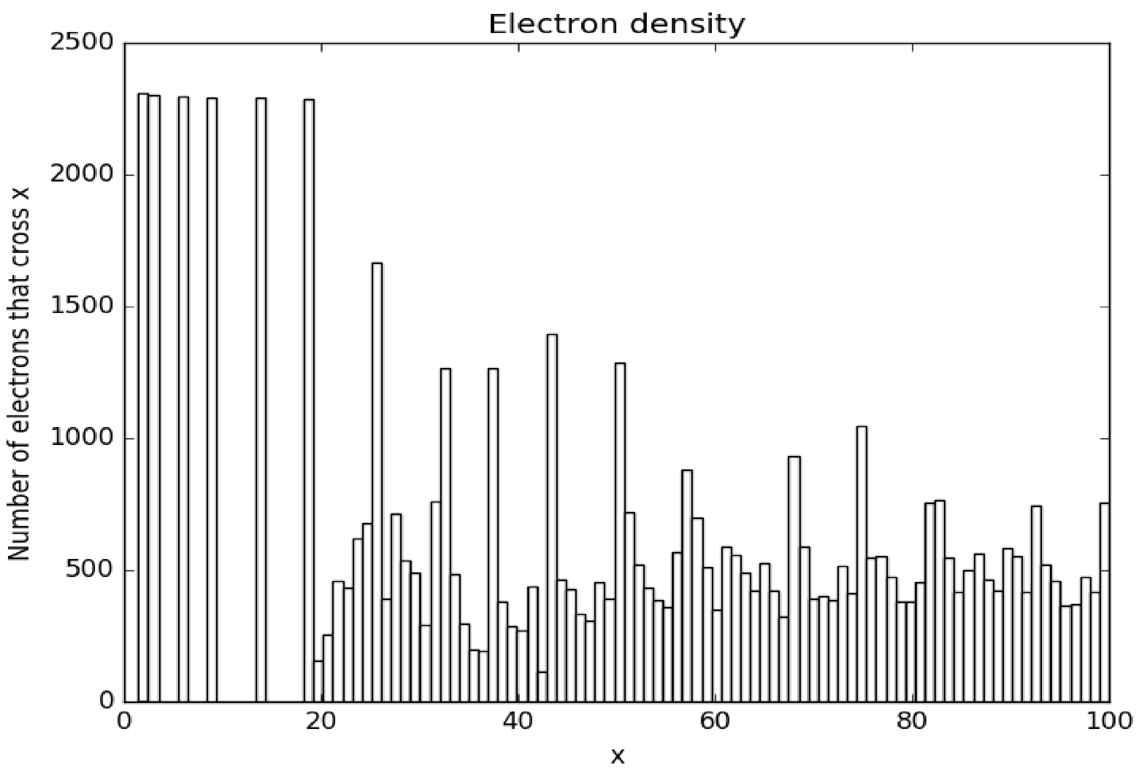
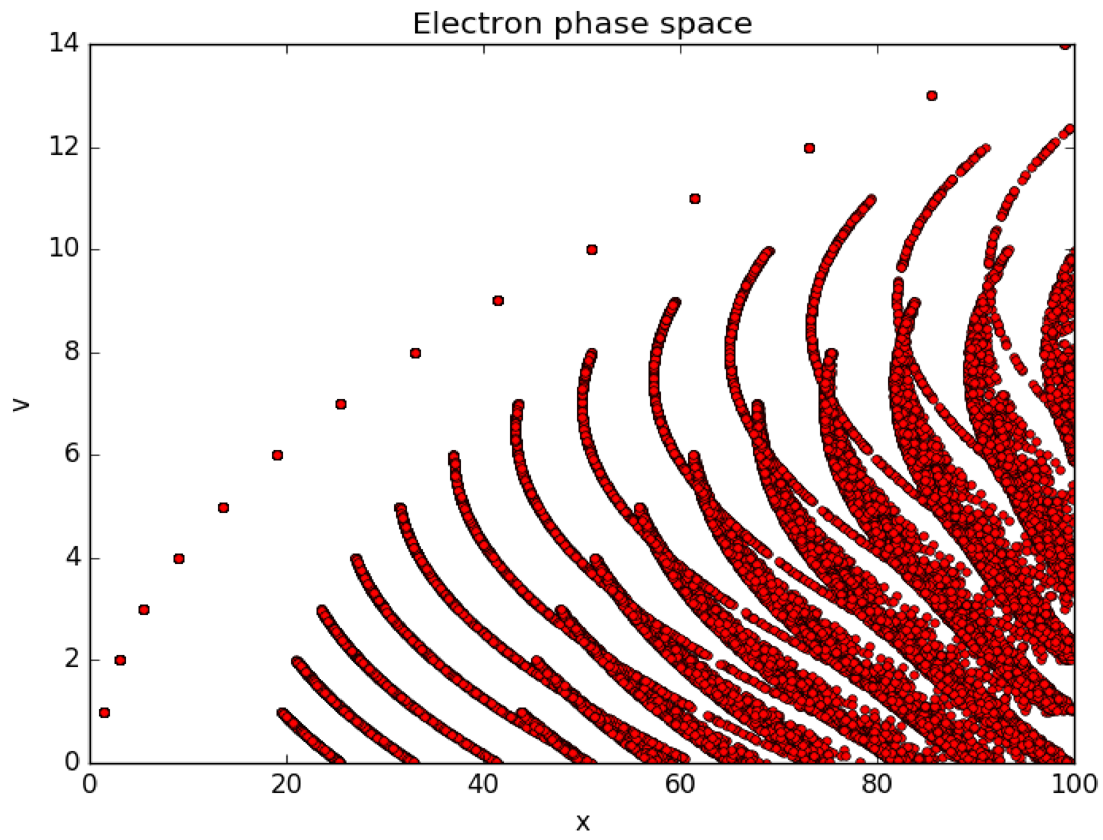


Figure 7: Electron phase space



- The electron phase clearly varies if we assume time is unifromly distributed but not space . It is shown in figure 7 which differs from figure 2.

4 Code:

```
#import packages
from pylab import *
from numpy import *
from matplotlib import *
import sys
#user input using sys.argv
try:
    n = int(sys.argv[1])
    M = int(sys.argv[2])
    nk = int(sys.argv[3])
    u0 = int(sys.argv[4])
    p = float(sys.argv[5])
except:
    n=100 # spatial grid size.
    M=5 # number of electrons injected per turn.
    nk=500 # number of turns to simulate.
    u0=7 # threshold velocity.
    p=0.5 # probability that ionization will occur
Msig=2#standard deviation
#Create vectors to hold the electron information
xx = zeros((n*M))
u = zeros((n*M))
dx = zeros((n*M))
#information from simulation
I = []#Intensity of emitted light
X = []#Electron position
V = []#Electron velocity
#loop begins
for k in range(1,nk+1):
    ii = where(xx>0)[0]
    dx[ii] = u[ii] + 0.5
    xx[ii] = xx[ii] + dx[ii]
    u[ii] = u[ii] + 1
    #set position velocity of elec which reach anode
    jj = where(xx>=n)[0]
    u[jj] = 0
    xx[jj] = 0
    dx[jj] =0
    #those elec whose energy>threshold
    kk = where(u>=u0)[0]
    ll = where(rand(len(kk))<=p)[0]
    kl = kk[ll]
    xx[kl]=xx[kl]-dx[kl]
    u[kl]=u[kl]-1
    dt = rand(len(kl))
    dx[kl] = u[kl]*dt + 0.5*dt*dt
    xx[kl] =xx[kl]+dx[kl]
    u[kl]=1-dt
    I.extend(xx[kl].tolist())
    xx[kl]=xx[kl]+0.5*(1-dt)*(1-dt)
    m=int(randn()*Msig+M)
```

```
        pp = where(xx>0)[0]
        X.extend(xx[pp].tolist())
        V.extend(u[pp].tolist())
        m1 = where(xx==0)[0]
        y=m1[0:min(len(m1),m)]
        xx[y]=1
    title('Emission Intensity')
    xlabel('x')
    ylabel('I')
    hist(I,100,normed=1,color='white')
    show()
    title('Electron density')
    xlabel('x')
    ylabel('Number of electrons that cross x')
    hist(X,100,color='white')
    show()
    title('Electron phase space')
    xlabel('x')
    ylabel('v')
    plot(X,V,'ro',markersize=4)
    show()
    population_count=hist(I,100)[0]
    bins = hist(I,100)[1]
    xpos= 0.5*(bins[0:-1]+bins[1:])
    for x in range(len(xpos)):
        print "%f  %d\n" %(xpos[x],population_count[x])
```

5 Note:

- simulation.py is done assuming space is uniformly distributed.simulation2.py is done assuming only time is uniformly distributed.