EE - 2703 Applied Programming Lab
Assignment -5

The Tube Light Model Simulation

Mohammed Khandwawala EE16B117

Contents

1 Introduction

This simulation consist of a 1 -D model of tubelight . In presence of an electric field the electron on the tube accelerates. Electrons are emitted from anode at rest and move towards cathode. In this model assumption is that after electron reaches critical energy it excites the atom (of gas filed in the tube) and comes to rest. Electrons reaching anode are lost. At every step new electrons are introduced in the tube . This number of new electron is given by normally distributed random number. And the process repeats. Tube starts from x = 0 and ends at x = 100, where electron are absorbed in the anode.

2 Parameters for the simulation

The parameters are number of length of the tube(n), mean(M), number of iterations(nk), threshold or critical velocity(u0), probability of collision(p) and standard deviation(Msig). All the inputs are given through command line . Default value of the arguments are already assigned.

```
if(len(sys.argv) == 7):
        n = int(sys.argv[1])
        M = int(sys.argv[2])
        nk = int(sys.argv[3])
        u0 = float(sys.argv[4])
        p = float(sys.argv[5])
    Msig = float(sys.argv[6])
else:
        n=100 #spatial grid size
        M = 5 #number of electron injected per turn
        nk = 500 #number of turns to simulate
        u0 = 5 #threshold velocity
        p = 0.25 #probality that ionization will occur
        Msig = 2 #standard deviation of normal distribution
```

3 Declaring vectors

xx vector stores the position of the electrons in the tube . Similarly u stores velocity and dx displacement in current turn. I,X and V are lists to store intensity at all points , position and velocity of all the electrons passed.

```
xx = np.zeros(n * M)     #to hold position of the electron
u = np.zeros(n * M)      #to hold velocity of electron
dx = np.zeros(n * M)     #displacement in current turn

I = []               #intensity of light
X = []               #Electron position
V = []               #electron veloctiy
```

4 Computation at each step

The below code is in a for loop , to compute each of the quantities for each time step.

4.1 Updating position and velocity

Selecting position of all the electrons that are in the tube i.e. electrons with position greater than 0 and then updating its position as

$$x_i \leftarrow x_i + dx_i$$

for velocity,

$$v_i = u_i + a\Delta t$$

using a = 1 and Δ t = 1,

$$u_i \leftarrow u_i + 1$$

now for dx,

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

taking a = 1.

$$dx_i \leftarrow u_i + \frac{1}{2}$$

```
#initially calculating outside the loop
ii = np.where(xx > 0)[0]

#loop starts
dx[ii] = u[ii] + 0.5
xx[ii] = xx[ii] + dx[ii]
u[ii] = u[ii] + 1
```

4.2 Removing electron reaching anode

selecting the electron with their position greater

```
jj = np.where(xx > n)[0]
dx[jj] = u[jj] = xx[jj] = 0
```

4.3 Exciting atoms

Atoms will get excited if they are hit by electron with velocity greater than the threshold. After the collision the velocity of the electron becomes 0 . The electron with velocity greater than threshold hits the atom with the probability 0.25. generating random probability distribution for electron having velocity greater than the critical velocity. And selecting the electron with probability less than p and making their velocity 0 (considering inelastic collision). This collision could have been anywhere between x$_i$ and x$_i$ - dx$_i$. We will generate random number between 0 and 1 and update the position between x$_i$ and x$_i$ and x$_i$ - dx_i based on the random number. Now these excited atoms will emit photon so, for intensity we can update the intensity (as position) with the position of collision.

```
kk = np.where(u >= u0)
ll = np.where(np.random.rand(len(kk[0])) <= p)
kl = kk[0][ll]
u[kl] = 0

rho = np.random.rand(len(kl))
xx[kl] = xx[kl] - dx[kl] * rho

I.extend(xx[kl].tolist())
```

4.4 Injecting new electrons

New electron are added at the of each step. Number of electrons added are one standard deviation around the mean. In this assignment, standard deviation is around mean 5. These injected electrons are added to our position vectors which has empty spots , if in case it does not have space for newly injected electrons in the array than just inject the electrons as many as the vacant spaces. injected electron have initial position one and velocity and dx both initially zero.

```
m = int(np.random.randn() * Msig + M)

empty = np.where(xx == 0)[0]
xx[empty[:min(m,len(empty))]] = 1
u[empty[:min(m,len(empty))]] = 0
dx[empty[:min(m,len(empty))]] = 0
```

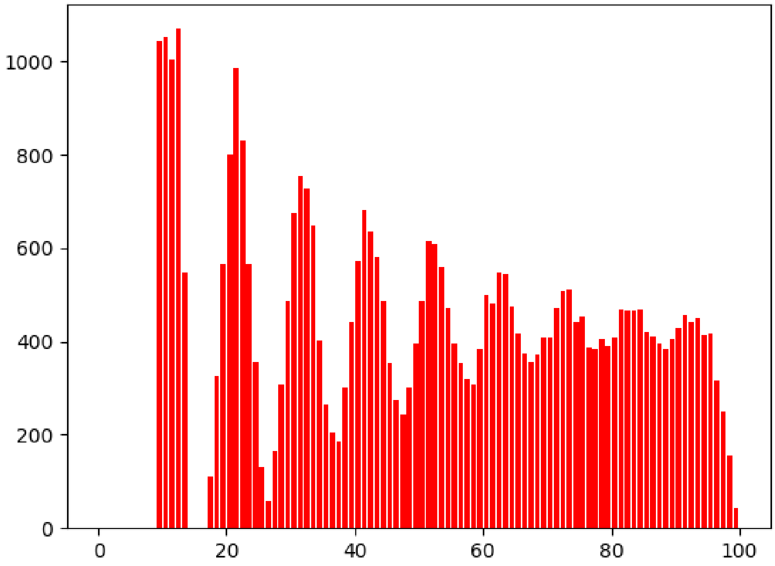
4.5 Updating X and V

Selecting all the positive positions for electron and its corresponding velocity and updating it in X and V list. Index of positive values in position vector computed here is used in the beginning of the next loop.

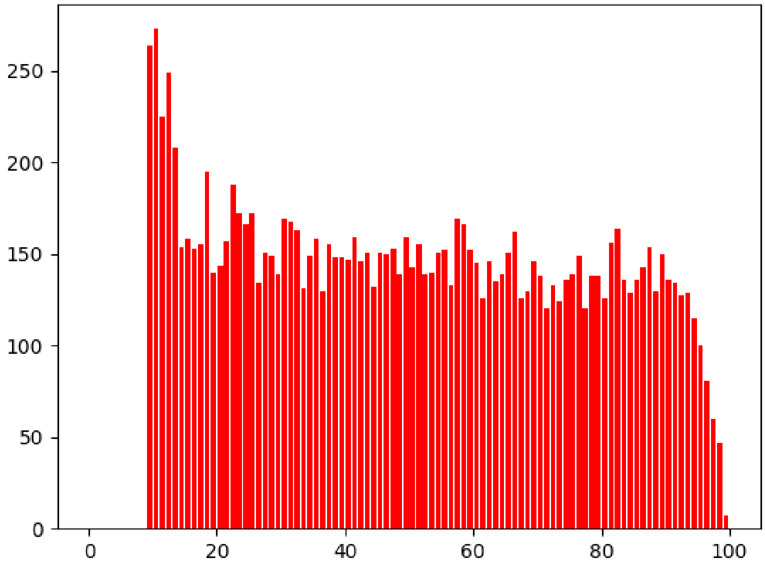
```
ii = np.where(xx > 0)[0] #selecting electron in the tube
X.extend(xx[ii].tolist())
V.extend(u[ii].tolist())
```

4.6 Histogram of Intensity

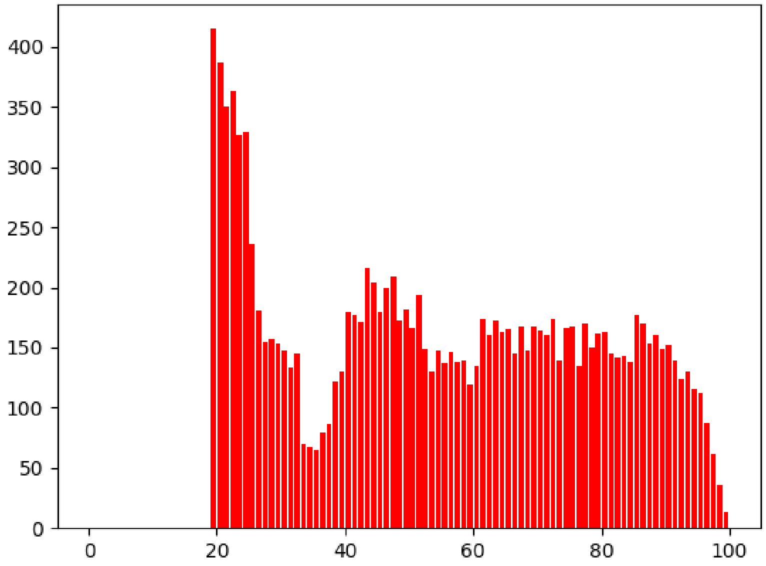
Looking at the plot we can see that the emission starts somewhere near 16 as the before that no electron has reached the critical velocity required to excite the atom.



Above is the histogram for probability of collision 1. We see dark/dim spots. As the electrons starts accelerating from cathode it reaches critical velocity and the collides as p = 1 . And since velocity becomes zero it again starts accelerating reaches critical velocity and the process continues. These slowing down of electron results in dark spaces. Since continuously electrons are injected they will have a similar distribution and the overall intensity pattern will be superposition. Which results in these dark/dim spots.



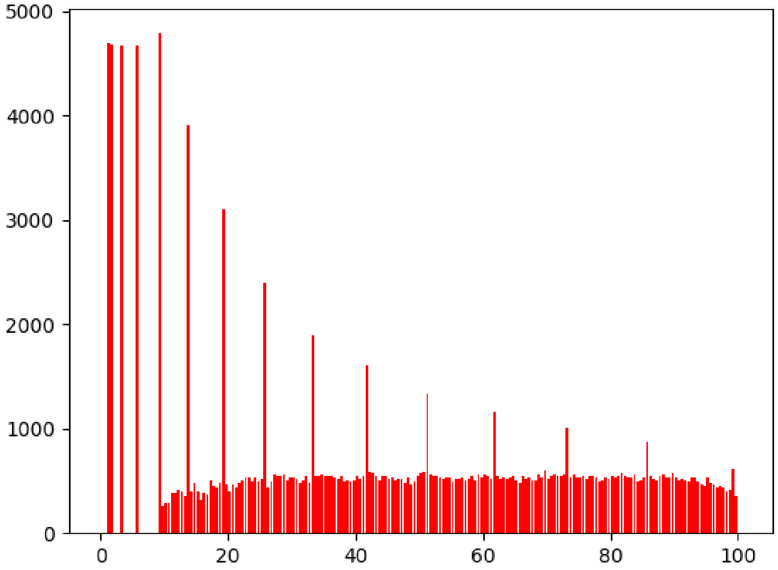
In case of probability 0.25 of collision the pattern followed is same as for probability 1. But minima's in the intensity is not visible because since the probability is 0.25 each curve spreads out and superposition of multiple of them (from continuously injected electrons) give almost constant intensity after first electrons achieves critical velocity. A peak of intensity is observed and then it is almost constant.



The above histogram is for probability 0.5 and critical velocity 7 dim area after the peak is more significant. And from all the histogram , the nature of intensity variation can be well understood.

```
plt.hist(I,bins=np.arange(0,101,1),rwidth=0.8,color = 'r')
plt.show()
```

4.7 Histogram of Position

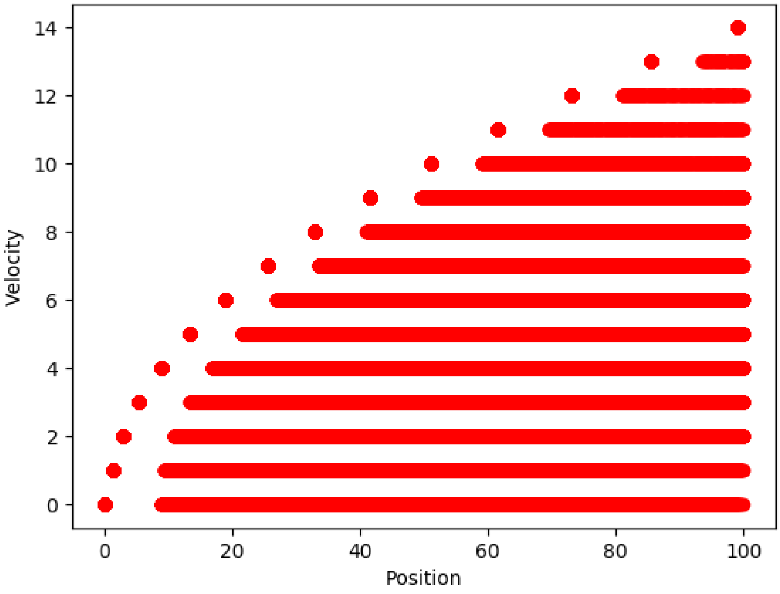


The above histogram is calculated for critical velocity = 5, mean = 10, standard deviation = 2, bin width = 0.5. The above histogram is for number of electron at a position in the tube. Before the position where electrons reach critical velocity the bars are of equal height and discrete gaps which shows uniformly accelerated electrons with no collisions. After collision the continuously decreasing peak (geometrically) which shows the fraction of each peak colliding. So the peak keeps of decreasing by the same fraction which should be equal to the probability of collision.

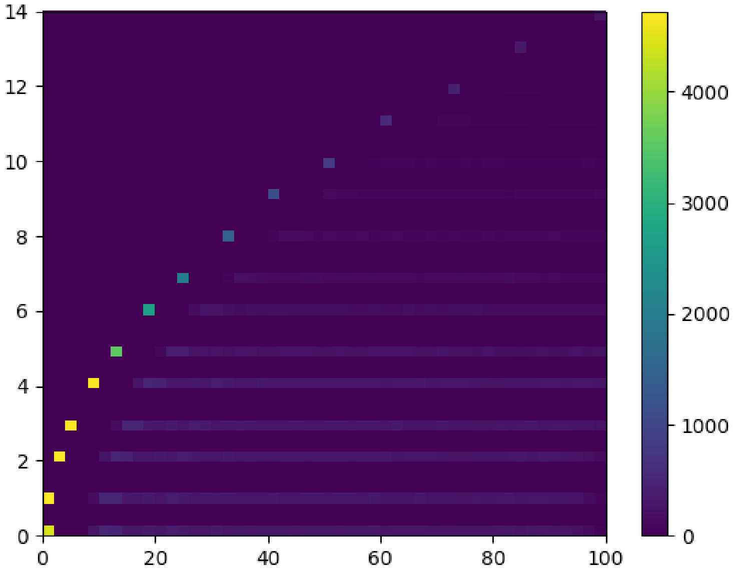
```
plt.hist(X,bins=np.arange(0,101,1),rwidth=0.8,color = 'r')
plt.show()
```

4.8 X vs V plot

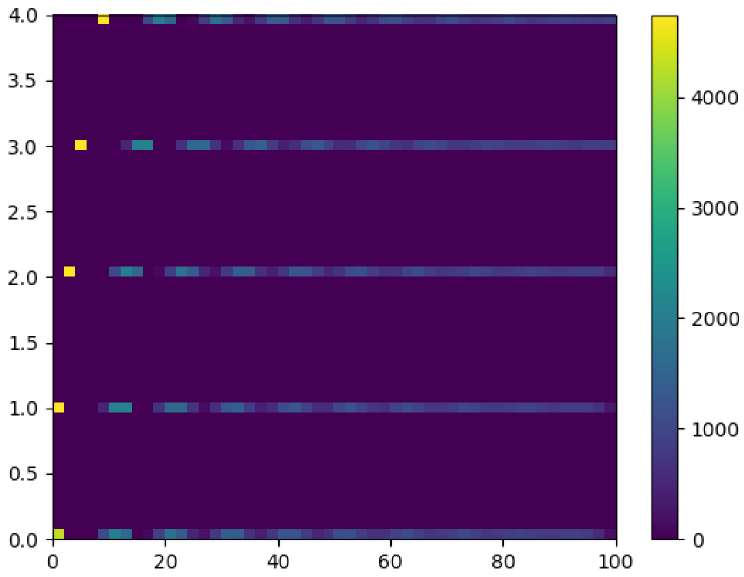
phase - space (position vs momentum) is represented by X vs V. In this plot of X and V the outline (can be seen from equation of motion $v^2 = u^2 + 2*a*x$) of the plot is tracing a parabola and the points inside the parabola are by the electrons colliding and loosing velocity.



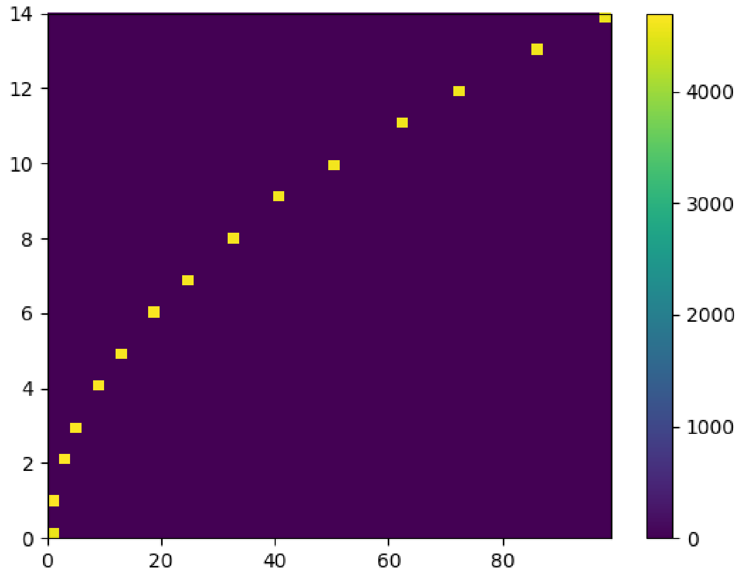
From 2-d histogram it is visible that the outline of the plot (ones which did not collide) have higher proportion which follows the parabola. (with p = 0.25)



If we plot the same for p = 1 , the plot gets cut off at v =45 , as probability of collision is 1 and critical velocity is 5. So every electron reaching v = 5 will collide.



And for p = 0 , parabolic nature is followed perfectly.



```
#phase space of electron
plt.plot(X,V, 'ro')
plt.xlabel("Position")
plt.ylabel("Velocity")
plt.show()

#plotting in 2-d histogram X,V
plt.hist2d(X,V,(50,50))
plt.colorbar()
plt.show()
```

4.9 Intensity Data

The table below shows the count of photons emitted from a particular position. (for p = 0.25, mean = 10, critical velocity = 5 and standard deviation = 2)

```
data = plt.hist(I,bins=np.arange(0,101,1),rwidth=0.8,color = 'r')
bins = data[1]
xpos = 0.5 * (bins[0:-1] + bins[1:])
df = pd.DataFrame({"xpos":xpos, "count":data[0]})
print tabulate(df, headers = 'keys', tablefmt = 'p' )
```

4.10 Table of count of emitted photons vs position

The table shows count of collision that is equal to the number of emitted photons vs position at which it emitted.

```
+----+---------+--------+
|    |   count |   xpos |
|----+---------+--------|
|  0 |       0 |    0.5 |
|  1 |       0 |    1.5 |
|  2 |       0 |    2.5 |
|  3 |       0 |    3.5 |
|  4 |       0 |    4.5 |
|  5 |       0 |    5.5 |
|  6 |       0 |    6.5 |
|  7 |       0 |    7.5 |
|  8 |       0 |    8.5 |
|  9 |     238 |    9.5 |
| 10 |     248 |   10.5 |
| 11 |     236 |   11.5 |
| 12 |     269 |   12.5 |
| 13 |     203 |   13.5 |
| 14 |     161 |   14.5 |
| 15 |     146 |   15.5 |
| 16 |     180 |   16.5 |
| 17 |     180 |   17.5 |
| 18 |     191 |   18.5 |
| 19 |     131 |   19.5 |
| 20 |     169 |   20.5 |
| 21 |     169 |   21.5 |
| 22 |     181 |   22.5 |
| 23 |     170 |   23.5 |
| 24 |     186 |   24.5 |
| 25 |     161 |   25.5 |
| 26 |     148 |   26.5 |
| 27 |     157 |   27.5 |
| 28 |     136 |   28.5 |
| 29 |     176 |   29.5 |
| 30 |     155 |   30.5 |
| 31 |     157 |   31.5 |
| 32 |     191 |   32.5 |
| 33 |     172 |   33.5 |
| 34 |     143 |   34.5 |
| 35 |     150 |   35.5 |
| 36 |     149 |   36.5 |
| 37 |     142 |   37.5 |
| 38 |     150 |   38.5 |
| 39 |     140 |   39.5 |
| 40 |     159 |   40.5 |
| 41 |      136 |   41.5 |
| 42 |      174 |   42.5 |
| 43 |      160 |   43.5 |
| 44 |      140 |   44.5 |
| 45 |      136 |   45.5 |
| 46 |      146 |   46.5 |
| 47 |      131 |   47.5 |
| 48 |      150 |   48.5 |
| 49 |      156 |   49.5 |
| 50 |      170 |   50.5 |
| 51 |      146 |   51.5 |
| 52 |      146 |   52.5 |
| 53 |      145 |   53.5 |
| 54 |      139 |   54.5 |
| 55 |      147 |   55.5 |
| 56 |      141 |   56.5 |
| 57 |      133 |   57.5 |
| 58 |      151 |   58.5 |
| 59 |      153 |   59.5 |
| 60 |      169 |   60.5 |
| 61 |      136 |   61.5 |
| 62 |      146 |   62.5 |
| 63 |      126 |   63.5 |
| 64 |      134 |   64.5 |
| 65 |      134 |   65.5 |
| 66 |      131 |   66.5 |
| 67 |      166 |   67.5 |
| 68 |      139 |   68.5 |
| 69 |      134 |   69.5 |
| 70 |      143 |   70.5 |
| 71 |      126 |   71.5 |
| 72 |      146 |   72.5 |
| 73 |      121 |   73.5 |
| 74 |      124 |   74.5 |
| 75 |      150 |   75.5 |
| 76 |      129 |   76.5 |
| 77 |      110 |   77.5 |
| 78 |      139 |   78.5 |
| 79 |      155 |   79.5 |
| 80 |      135 |   80.5 |
| 81 |      134 |   81.5 |
| 82 |      164 |   82.5 |
| 83 |      143 |   83.5 |
| 84 |      135 |   84.5 |
| 85 |      141 |   85.5 |
| 86 |      133 |   86.5 |
| 87 |      137 |   87.5 |
| 88 |      151 |   88.5 |
| 89 |      109 |   89.5 |
| 90 |      126 |   90.5 |
| 91 |      143 |   91.5 |
| 92 |      134 |   92.5 |
| 93 |      115 |   93.5 |
| 94 |      115 |   94.5 |
| 95 |      102 |   95.5 |
| 96 |       71 |   96.5 |
| 97 |       65 |   97.5 |
| 98 |       35 |   98.5 |
| 99 |       10 |    99.5 |
+----+---------+--------+
```

5 Modification in existing model

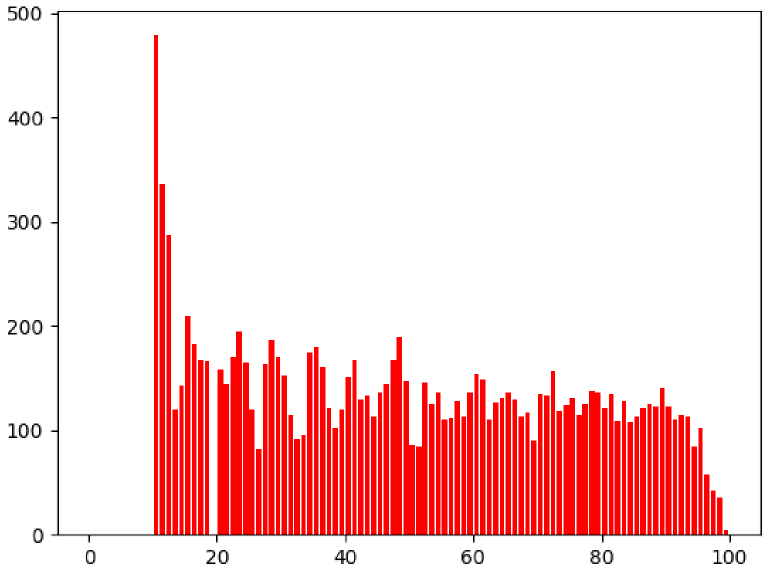
In the model used in this assignment assumes the than the collision between electron and atom happens between and random position between x_i and $x_i + dx_i$. The position between the between these position is selected by a uniformly distributed random number in position. From equation

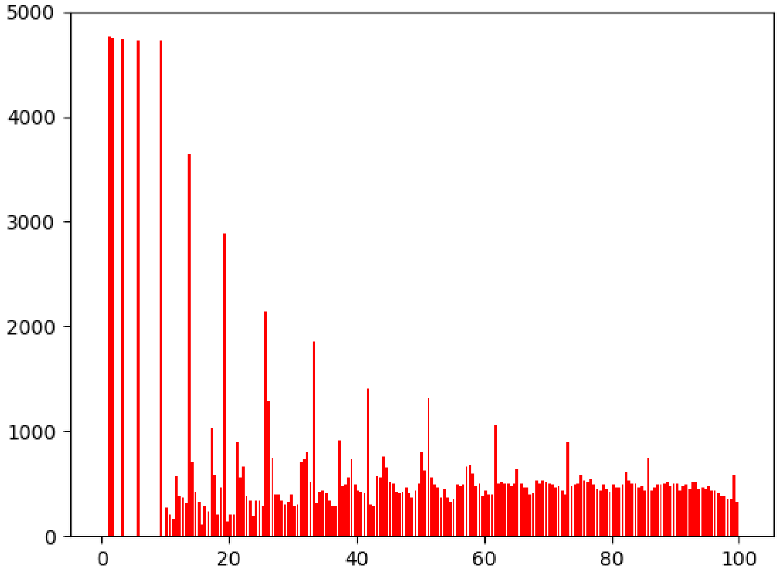
$$v = u + \frac{1}{2}at^2$$

we can select uniformly distributed random number for time and substitute in equation to get position of collision. after colliding its velocity will become 0 and it will again start accelerating till next time step (we are recording velocity an position at integer time steps). so we will update velocity as (1-dt) assuming unit acceleration and dt is random time instant when collision happened between one time step. Similarly position would be updated as, *position of collision* + 0.5(1-dt)^2.

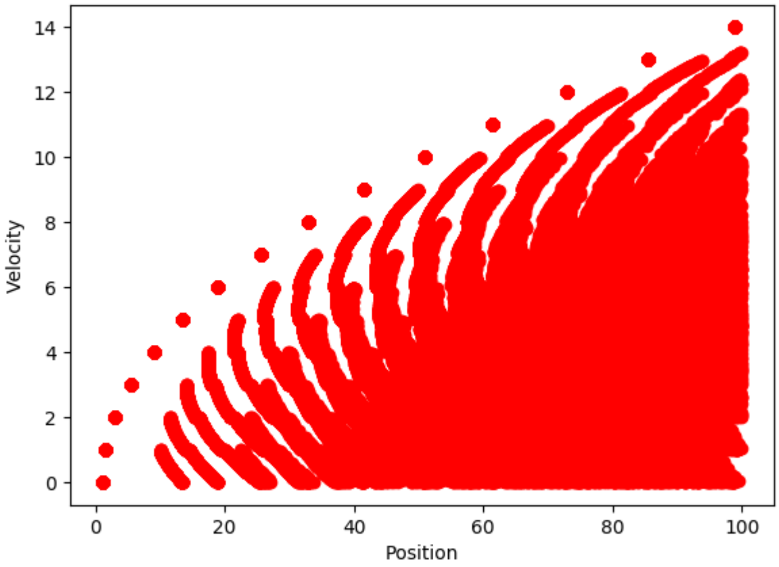
```
rho = np.random.rand(len(kl))# random time
xx[kl] = ( xx[kl] - dx[kl] )  + ( u[kl] - 1 )*rho  + 0.5 * rho ** 2
#finding exact position of collision between xi-1 and xi
xx[kl] =  x[kl] + (1 - rho) ** 2 #after colision velocity will be zero and it will aa
u[kl] = (1 - rho)          #making their velocity at after uniformly
```

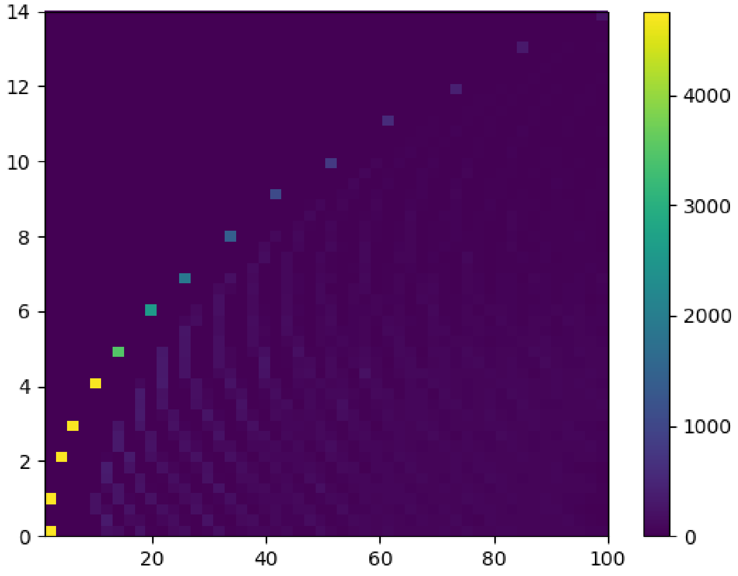
plots obtained from this model are shown below (for p = 0.25, mean = 10, standard deviation = 2, critical velocity = 5)





In the phase space plot velocity is no longer discretized . Continuous phase space in this model gives more accurate picture.




14
12
10
8
6
4
2
0
20
40
60
80
100
4000
3000
2000
1000
0