Vicsek Model Simulation

by Mohammad Jafari - Student of Physics , University of Mohaghegh Ardabili

In this project i will Have a look at The Vicsek model. This model is one of the prominent examples of Active Matter. In the begining of the 1990's Hungarian scientist called Thomas Vicsek tried to describe the collective motion of birds . Vicsek was inspired by a theory of the German physicist Werner Heisenberg who developed a theory about the magnetic behavior of materials . He considered an atom as a magnetic bar that each magnetic bar wants to align itself with neighboring Magnets.

in this project we will have a certain number of particles that are randomly defined by their random location and angle. then these particles will move with mean angle of their neighbors in the defined radius.

I start with importing required libraries

In [1]:
```
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

We will use numpy for arraying and randomizing our particles and to visualize our particles we will use pyplot from matplotlib. Now I define required parameters.

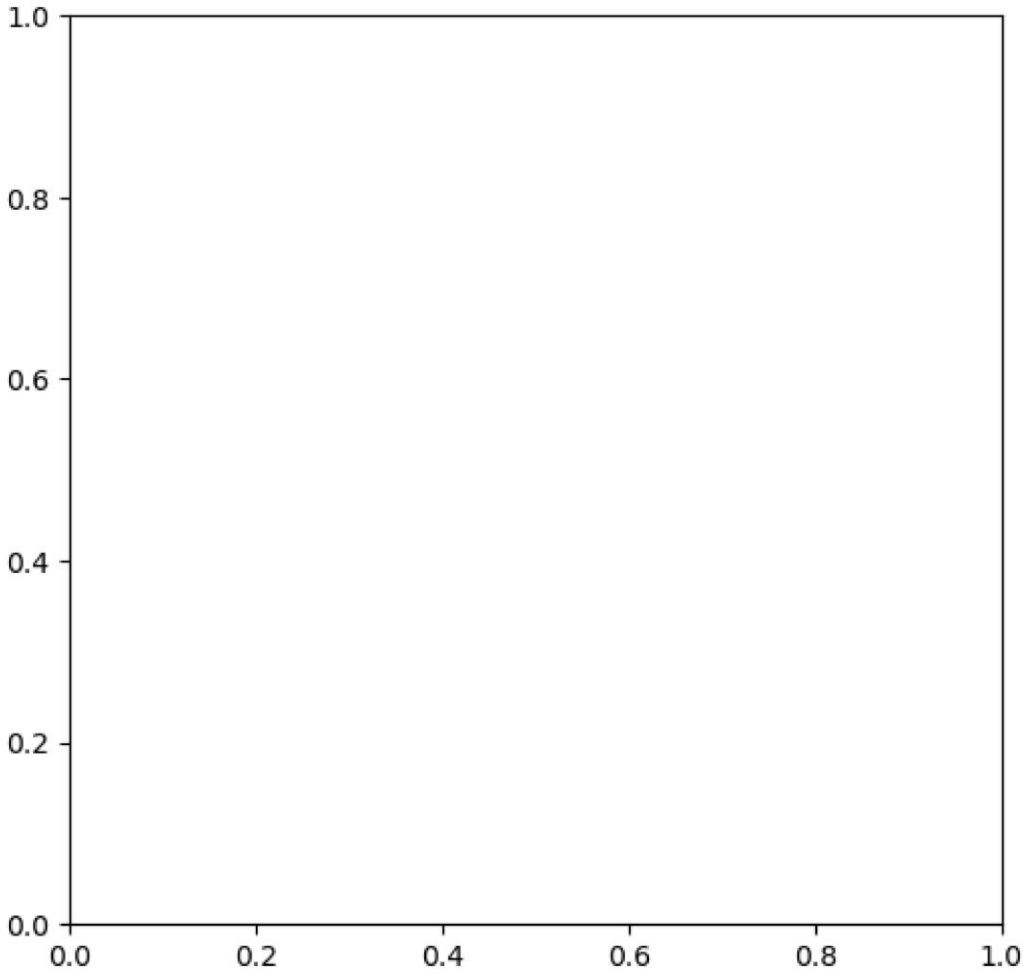
In [52]:
```
T = 10.0 #Time Frame of simulation
n_steps = 100 # steps of simulation
t_axis = np.linspace(0,T,n_steps) #time axis
dt = t_axis[1] - t_axis[0] #dt
n_particles = 500 #number of particles
v0 = 1.0 #velocity of each particle
f_intensity = 0.5 #intensity of fluctuation in angles
radius = 1.0 #radius to find neighbors around
scale = 10.0
```

Now I will randomize directions and positions of particles . I will scale position of particles by 10.

In [17]:
```
np.random.seed(777)
x = np.random.rand(n_particles,1) * scale
y = np.random.rand(n_particles,1) * scale
angles = 2 * np.pi * np.random.rand(n_particles,1)
vx = v0 * np.cos(angles)
vy = v0 * np.sin(angles)
```

After defining random particles , I will create a suitable figure to visualize particles .

In [18]:
```
figr = plt.figure(figsize=(6,6),dpi = 100)
ax = plt.gca()
```

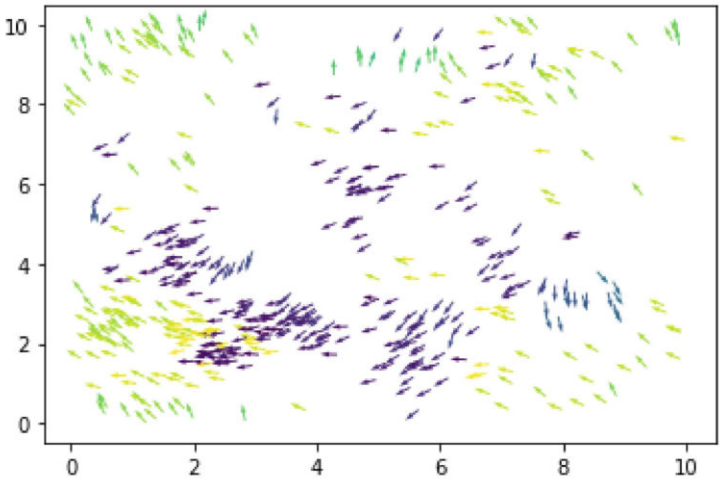


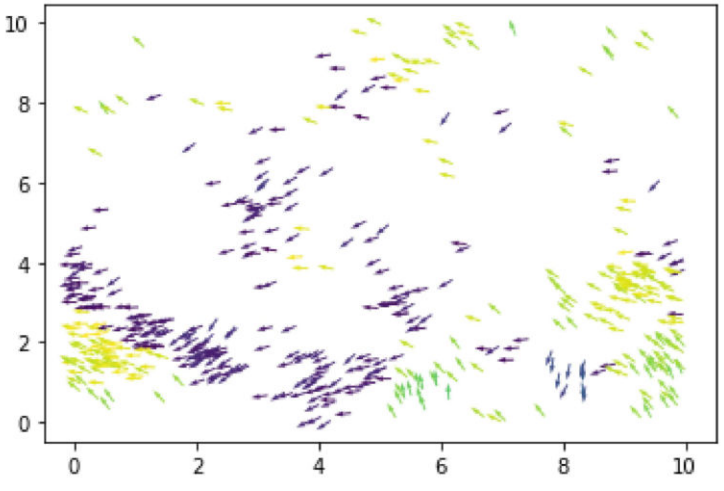
Now it's time to calculate and visualize direction and position of each particle at each timestep.

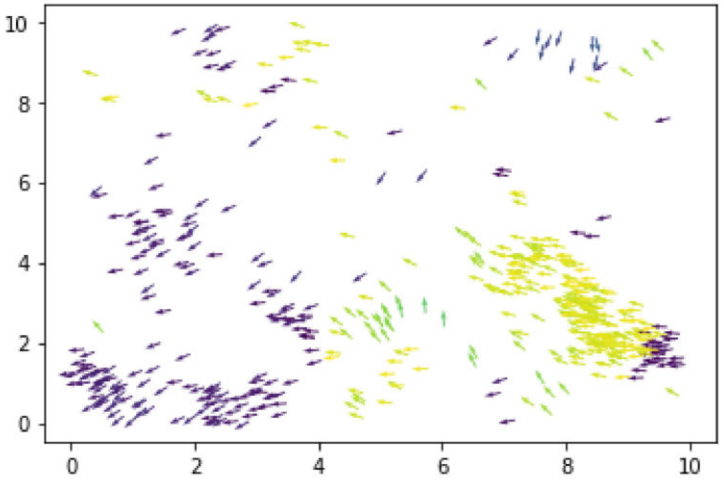
In [32]:

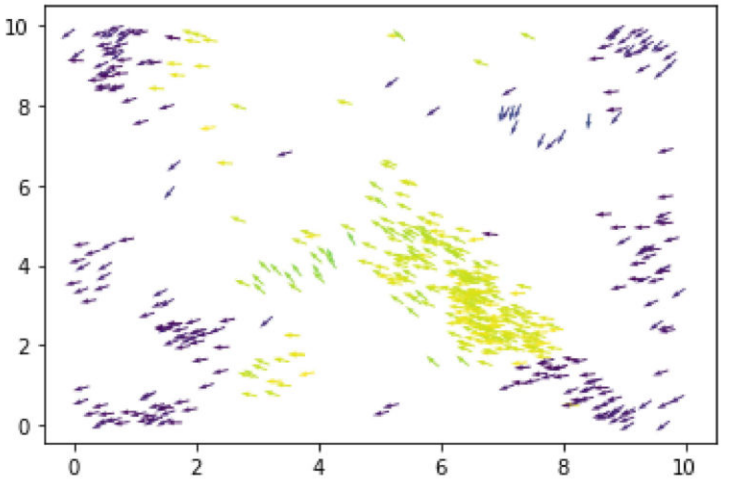
```
for i in range(10):
    x += vx * dt
    y += vy *dt
    #scaling new positions
    x = x % scale
    y = y % scale
    #Find mean angle of particles in defined Radius
    m_angles = angles
    for j in range(n_particles):
        neighbors = (x - x[j]) ** 2 + (y - y[j]) ** 2 < radius ** 2
        sx = np.sum(np.cos(angles[neighbors]))
        sy = np.sum(np.sin(angles[neighbors]))
        m_angles[j] = np.arctan2(sy,sx) #converting vectors to angles
    # Adding Noise to Angles
    angles = m_angles + f_intensity * (np.random.rand(n_particles,1) - f_intensit

    #new Velocities
    vx = v0 * np.cos(angles)
    vy = v0 * np.sin(angles)
    plt.cla()
    plt.quiver(x,y,vx,vy,angles)
    ax.set(xlim=(0,scale) , ylim = (0 , scale))
    ax.set_aspect('equal')
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
```

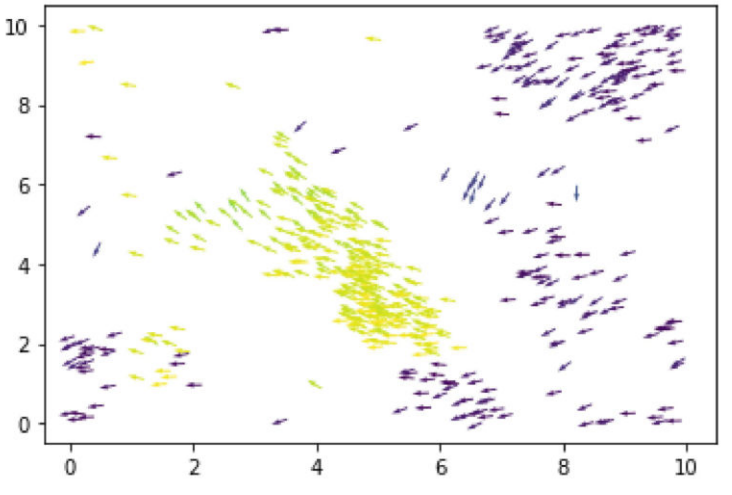
```
    plt.pause(0.001)
plt.show()
```

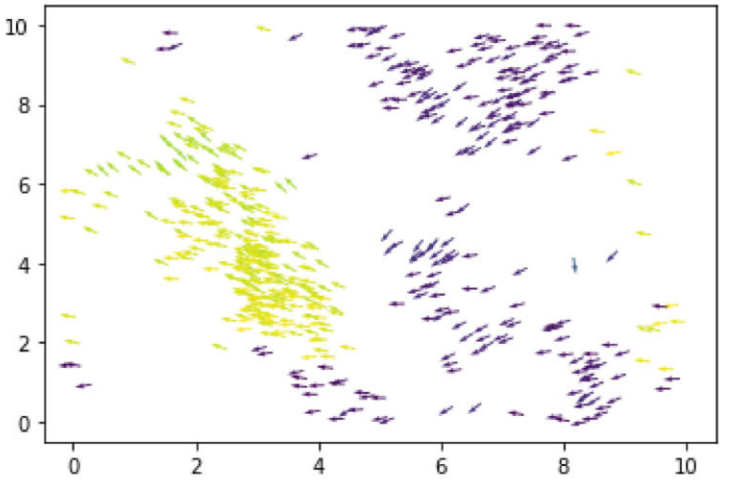


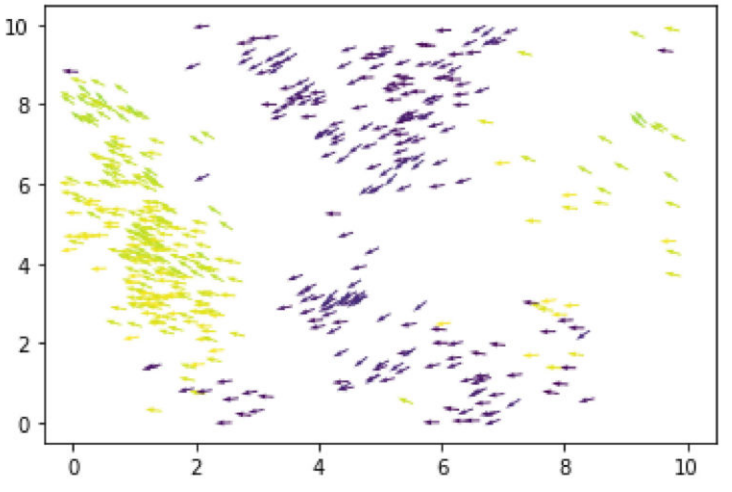


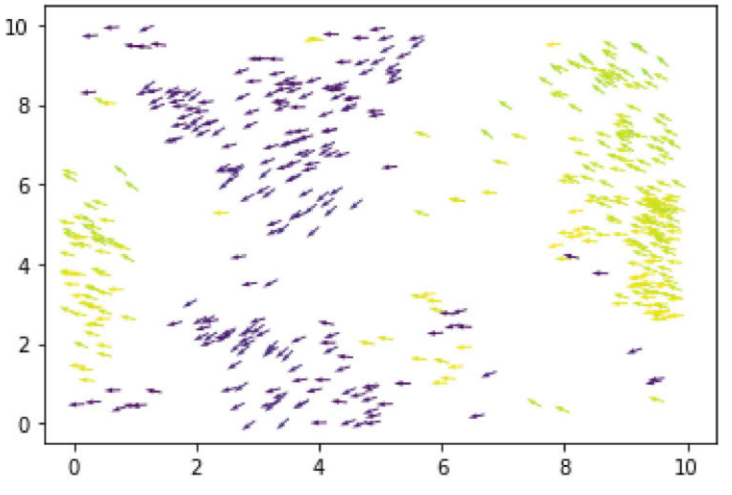


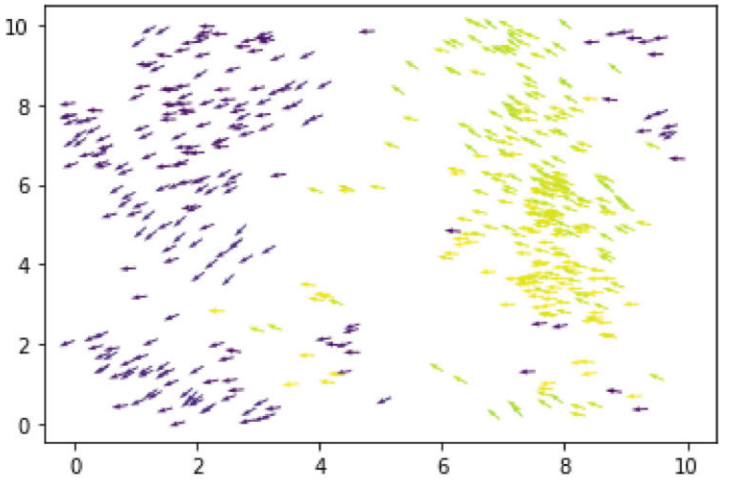

10
8
6
4
2
0
0
2
4
6
8
10

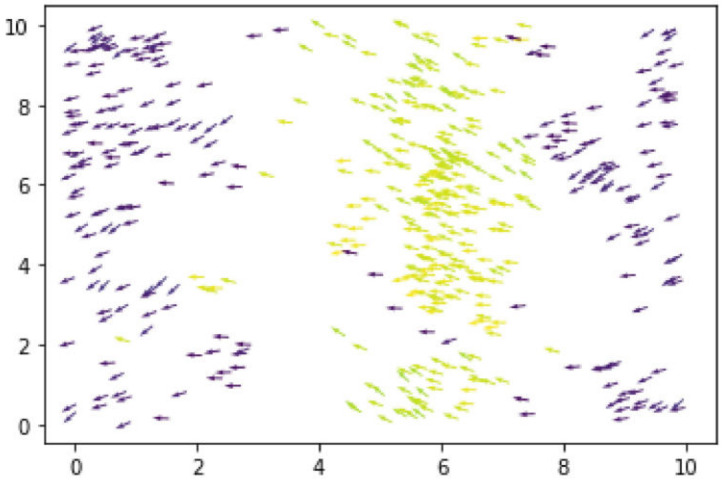

10
8
6
4
2
0
0
2
4
6
8
10


10
8
6
4
2
0
0
2
4
6
8
10


10
8
6
4
2
0
0
2
4
6
8
10


10
8
6
4
2
0
0
2
4
6
8
10


10
8
6
4
2
0
0
2
4
6
8
10



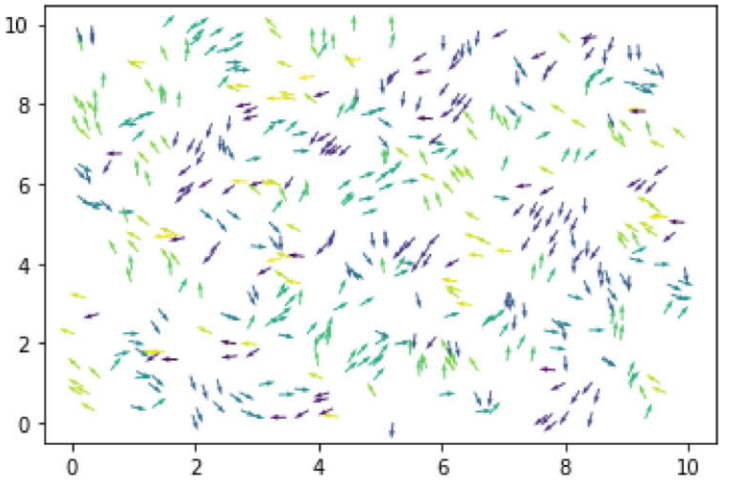
All the graphs in this notebook are presented by 10 steps

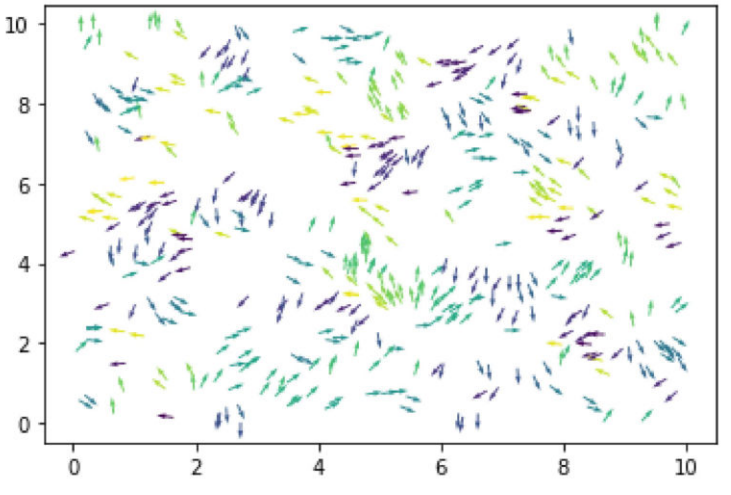
((fully plotted and animated graph of this section will be in "supplementary-material/fig1.mp4"))

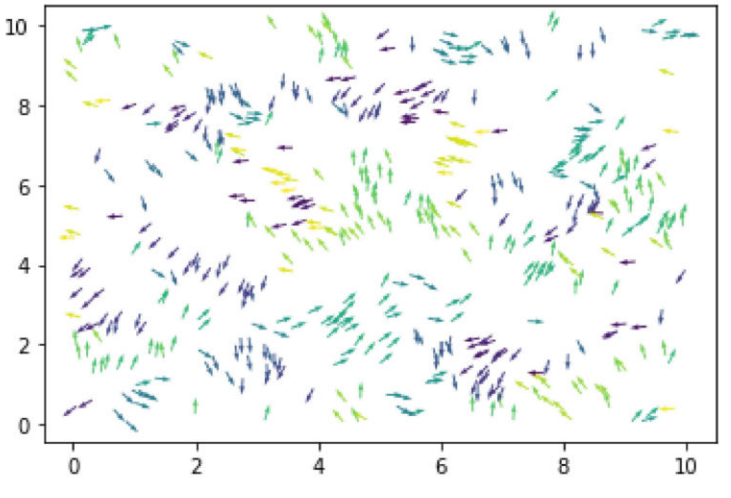
Now I want to change parameters and see how they affect the motion of particles. starting with radius .

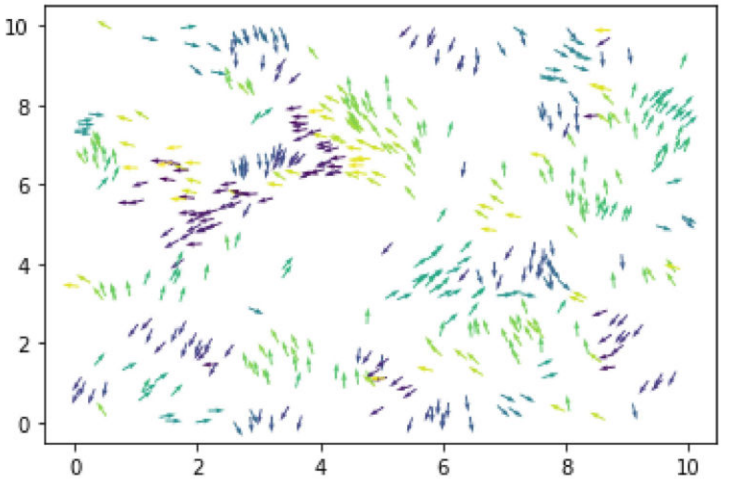
In [35]:

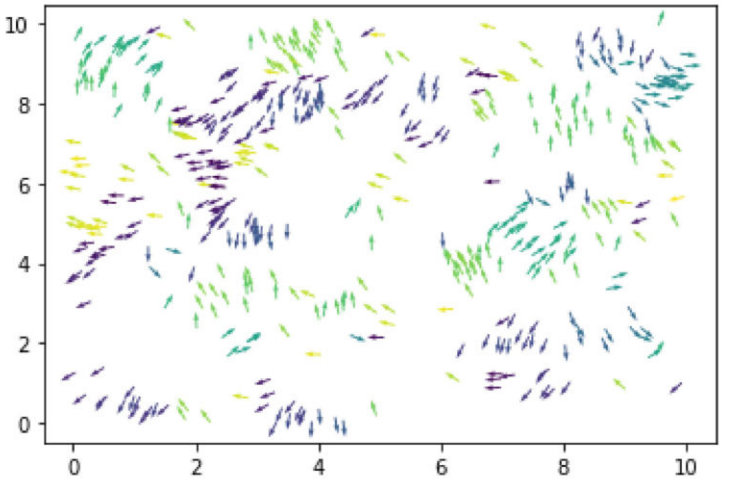
```
radius = 0.5
np.random.seed(777)
x = np.random.rand(n_particles,1) * scale
y = np.random.rand(n_particles,1) * scale
angles = 2 * np.pi * np.random.rand(n_particles,1)
vx = v0 * np.cos(angles)
vy = v0 * np.sin(angles)
for i in range(10):
    x += vx * dt
    y += vy *dt
    #scaling new positions
    x = x % scale
    y = y % scale
    #Find mean angle of particles in defined Radius
    m_angles = angles
    for j in range(n_particles):
        neighbors = (x - x[j]) ** 2 + (y - y[j]) ** 2 < radius ** 2
        sx = np.sum(np.cos(angles[neighbors]))
        sy = np.sum(np.sin(angles[neighbors]))
        m_angles[j] = np.arctan2(sy,sx) #converting vectors to angles
    # Adding Noise to Angles
    angles = m_angles + f_intensity * (np.random.rand(n_particles,1) - f_intensit

    #new Velocities
    vx = v0 * np.cos(angles)
    vy = v0 * np.sin(angles)
    #Visualizing Motion Of Particles
    plt.cla()
    plt.quiver(x,y,vx,vy,angles)
    ax.set(xlim=(0,scale) , ylim = (0 , scale))
    ax.set_aspect('equal')
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    plt.pause(0.001)
plt.show()
```

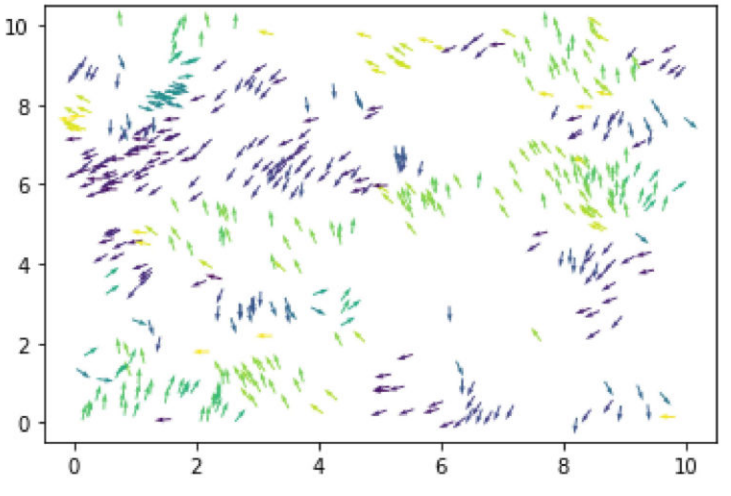

10
8
6
4
2
0
0
2
4
6
8
10

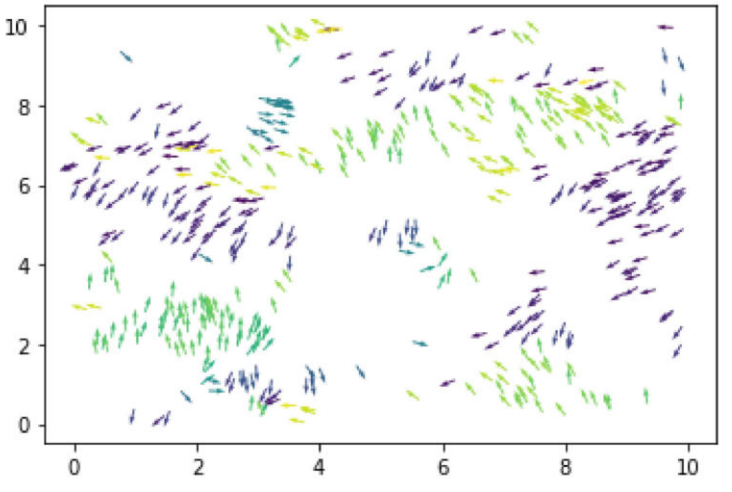

10
8
6
4
2
0
0
2
4
6
8
10

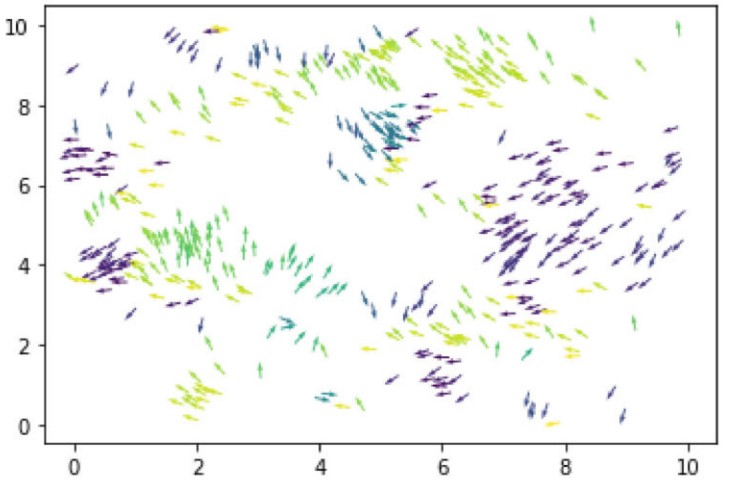

10
8
6
4
2
0
0
2
4
6
8
10

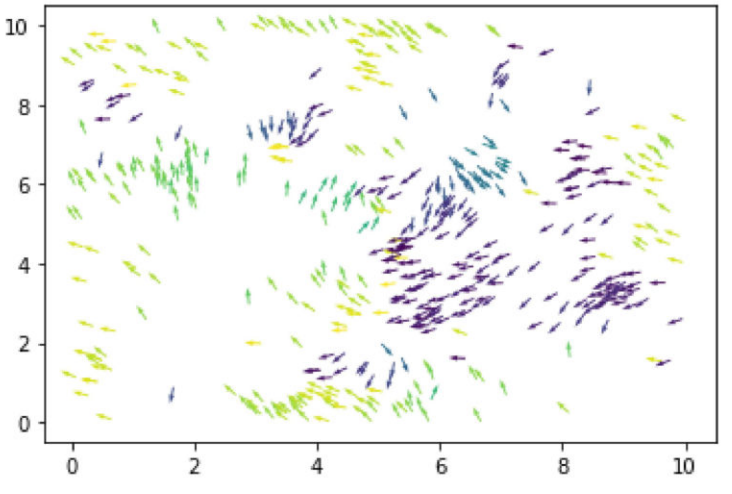

10
8
6
4
2
0
0
2
4
6
8
10

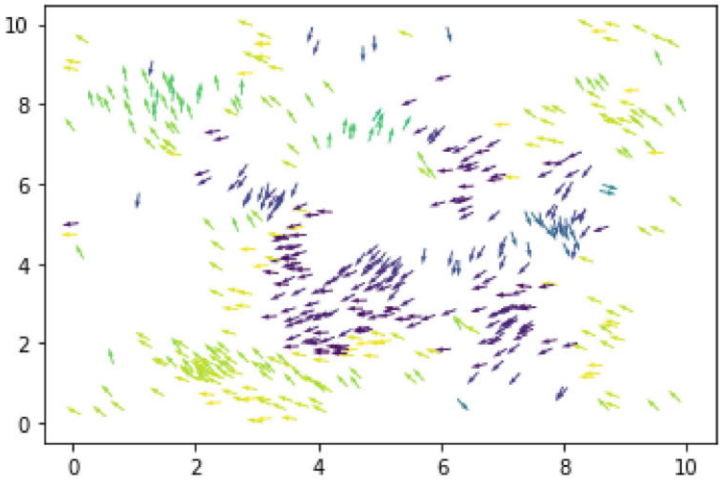

10
8
6
4
2
0
0
2
4
6
8
10


10
8
6
4
2
0
0
2
4
6
8
10


10
8
6
4
2
0
0
2
4
6
8
10


10
8
6
4
2
0
0
2
4
6
8
10

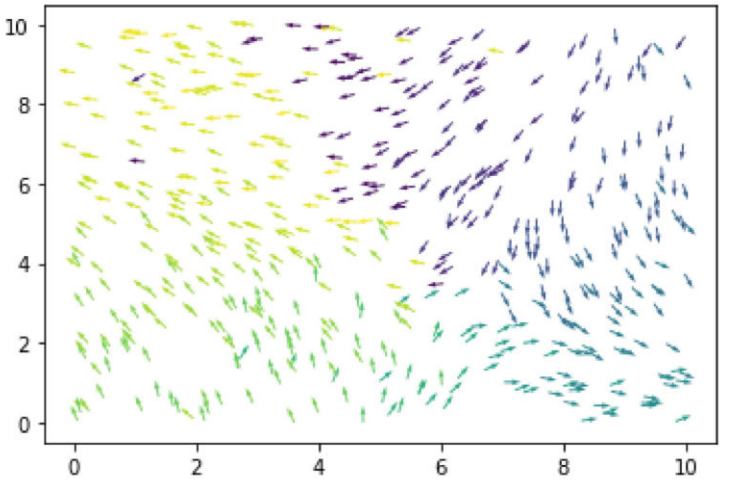

10
8
6
4
2
0
0
2
4
6
8
10

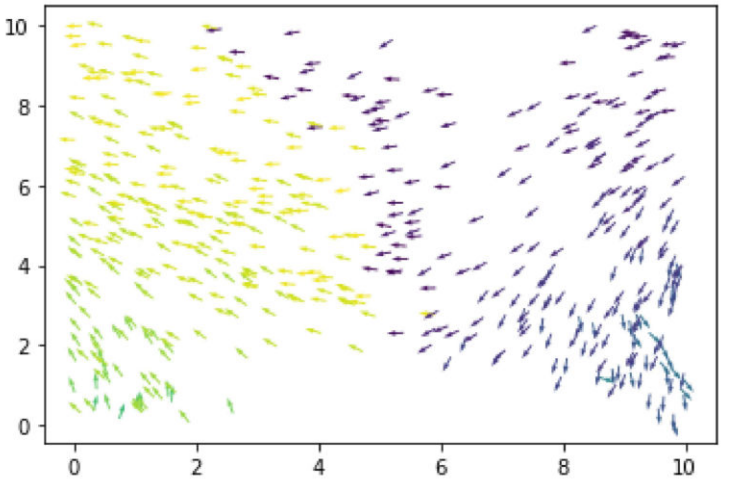


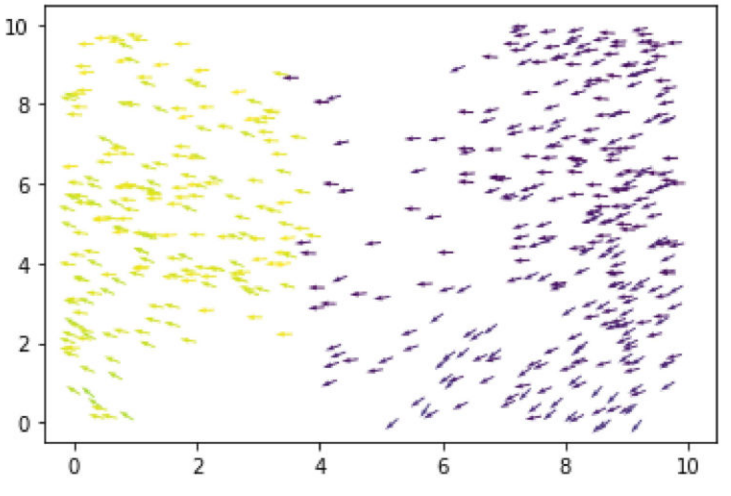
As you can see by decreasing radius , disorder in system increases. also gap between groups of particles with similar direction increases. fully plotted and animated graph of this section will be in "supplementary-material/fig2.mp4"

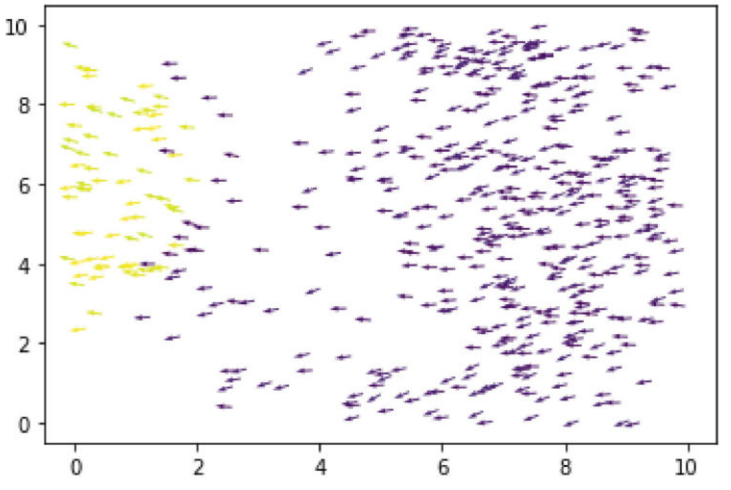
In [45]:

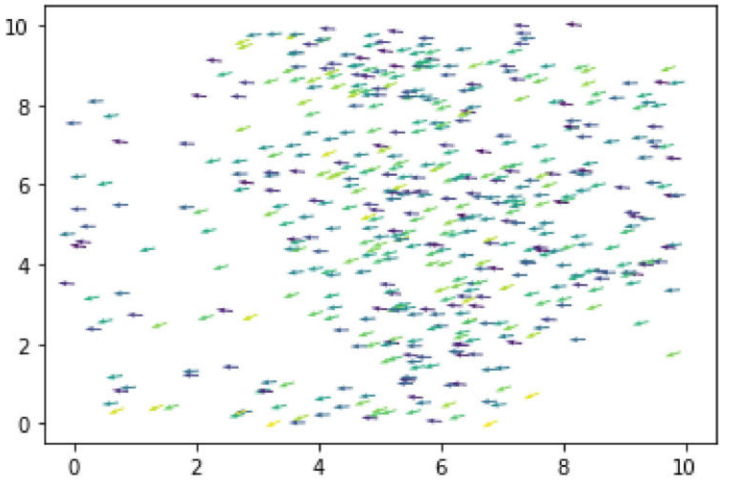
```
radius = 4.0
np.random.seed(767)
x = np.random.rand(n_particles,1) * scale
y = np.random.rand(n_particles,1) * scale
angles = 2 * np.pi * np.random.rand(n_particles,1)
vx = v0 * np.cos(angles)
vy = v0 * np.sin(angles)
for i in range(10):
    x += vx * dt
    y += vy *dt
    #scaling new positions
    x = x % scale
    y = y % scale
    #Find mean angle of particles in defined Radius
    m_angles = angles
    for j in range(n_particles):
        neighbors = (x - x[j]) ** 2 + (y - y[j]) ** 2 < radius ** 2
        sx = np.sum(np.cos(angles[neighbors]))
        sy = np.sum(np.sin(angles[neighbors]))
        m_angles[j] = np.arctan2(sy,sx) #converting vectors to angles
    # Adding Noise to Angles
    angles = m_angles + f_intensity * (np.random.rand(n_particles,1) - f_intensit

    #new Velocities
    vx = v0 * np.cos(angles)
    vy = v0 * np.sin(angles)
    #Visualizing Motion Of Particles
    plt.cla()
    plt.quiver(x,y,vx,vy,angles)
    ax.set(xlim=(0,scale) , ylim = (0 , scale))
    ax.set_aspect('equal')
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    plt.pause(0.001)
plt.show()
```

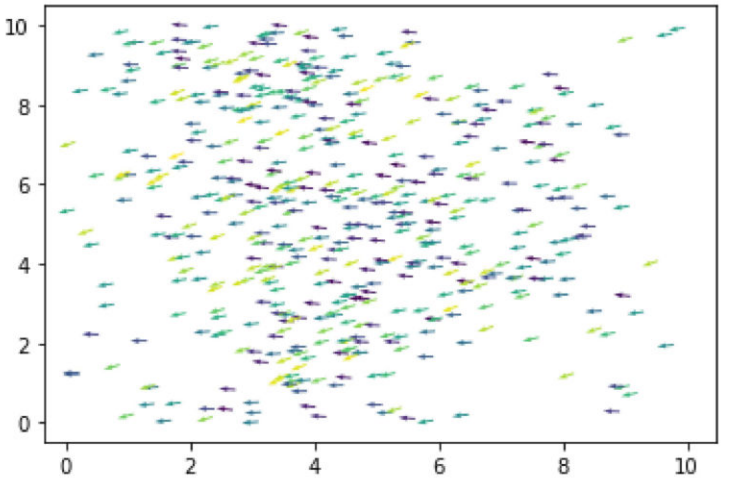

10
8
6
4
2
0
0
2
4
6
8
10

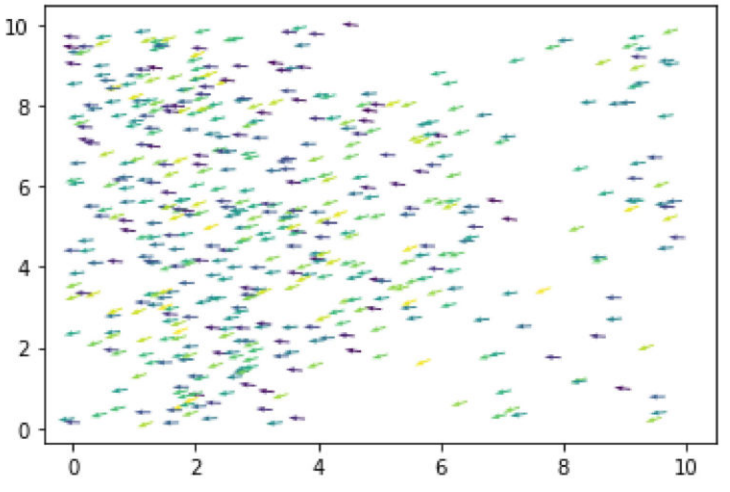

10
8
6
4
2
0
0
2
4
6
8
10

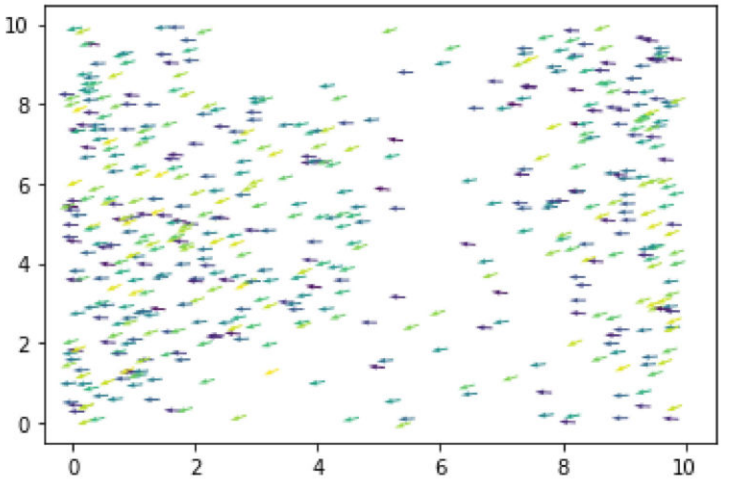

10
8
6
4
2
0
0
2
4
6
8
10

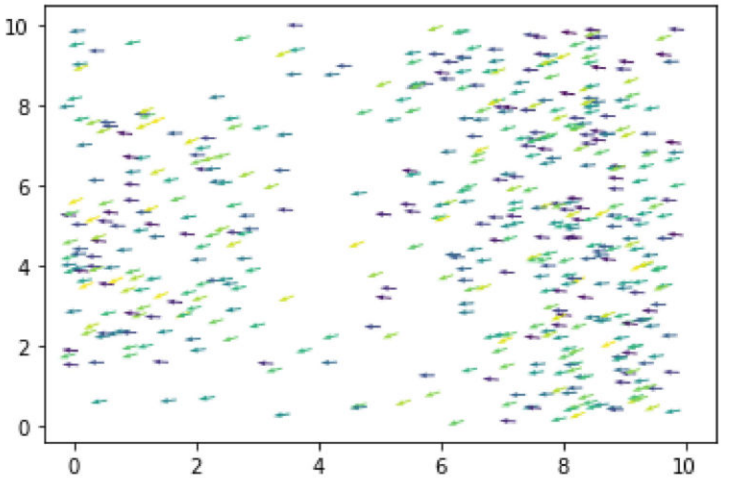

10
8
6
4
2
0
0
2
4
6
8
10

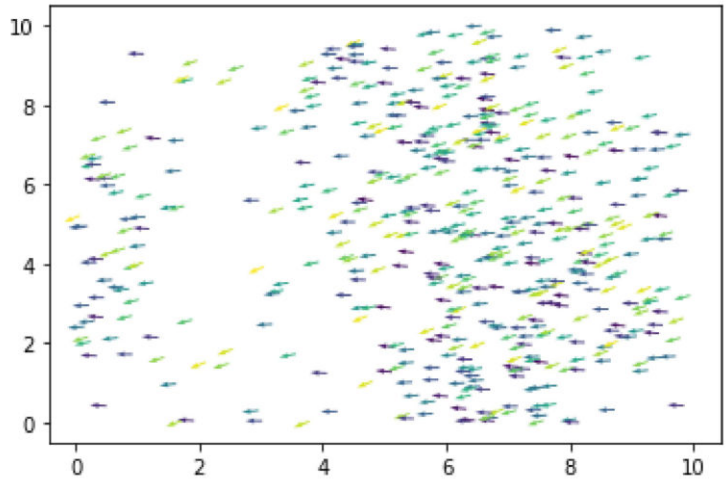

10
8
6
4
2
0
0
2
4
6
8
10


10
8
6
4
2
0
0
2
4
6
8
10


0
2
4
6
8
10


0
2
4
6
8
10


0
2
4
6
8
10

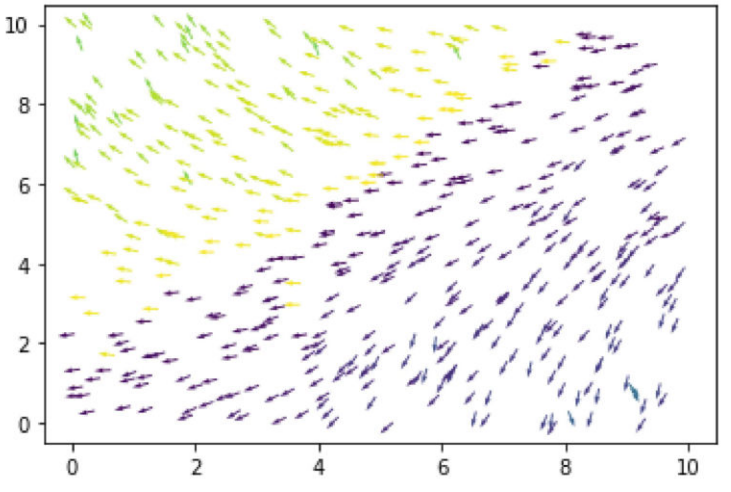


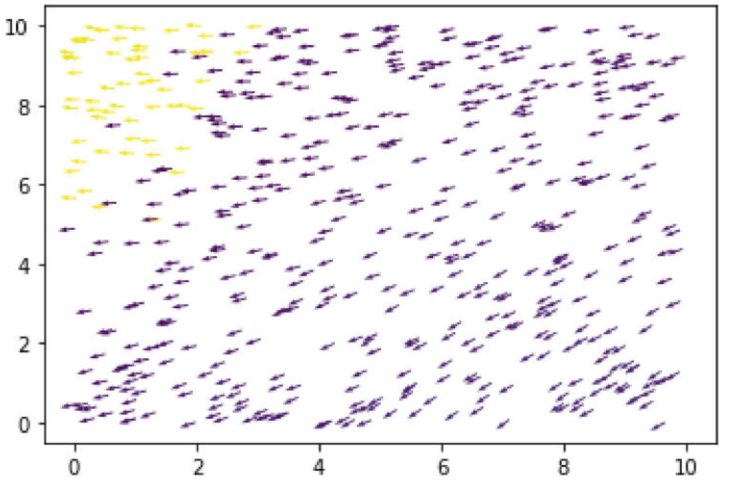
As you can see by increasing radius , disorder in system decreases. fully plotted and animated graph of this section will be in "supplementary-material/fig3.mp4"

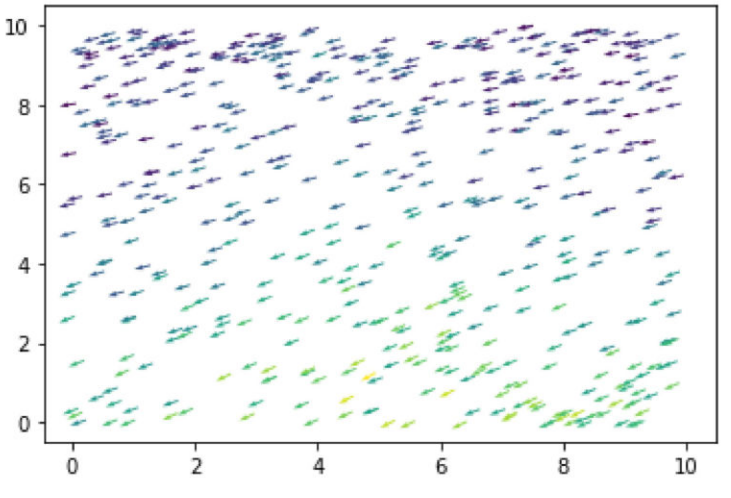
Now i want to change noise intensity and see what happens

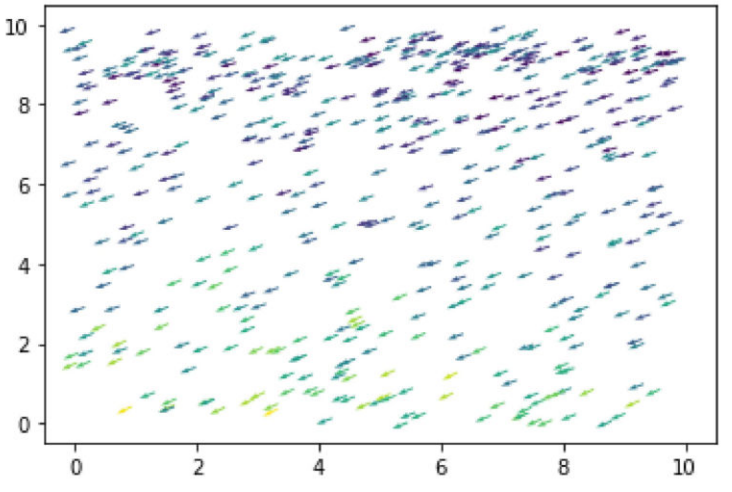
In [46]:

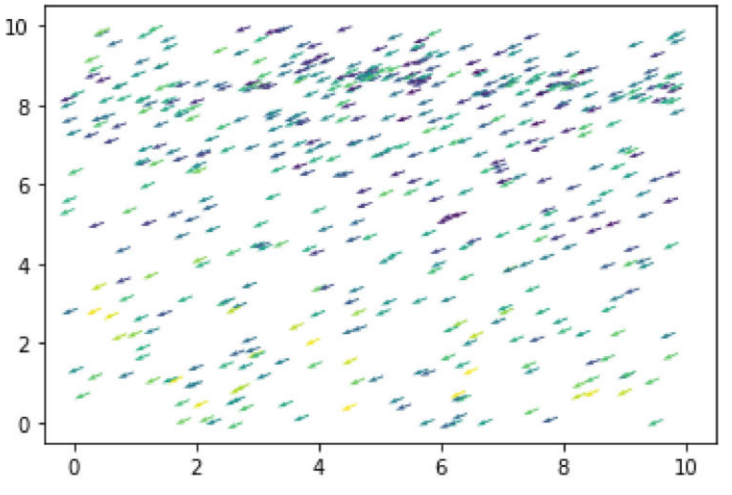
```
f_intensity = 0.1
np.random.seed(777)
x = np.random.rand(n_particles,1) * scale
y = np.random.rand(n_particles,1) * scale
angles = 2 * np.pi * np.random.rand(n_particles,1)
vx = v0 * np.cos(angles)
vy = v0 * np.sin(angles)
for i in range(10):
    x += vx * dt
    y += vy *dt
    #scaling new positions
    x = x % scale
    y = y % scale
    #Find mean angle of particles in defined Radius
    m_angles = angles
    for j in range(n_particles):
        neighbors = (x - x[j]) ** 2 + (y - y[j]) ** 2 < radius ** 2
        sx = np.sum(np.cos(angles[neighbors]))
        sy = np.sum(np.sin(angles[neighbors]))
        m_angles[j] = np.arctan2(sy,sx) #converting vectors to angles
    # Adding Noise to Angles
    angles = m_angles + f_intensity * (np.random.rand(n_particles,1) - f_intensit

    #new Velocities
    vx = v0 * np.cos(angles)
    vy = v0 * np.sin(angles)
    #Visualizing Motion Of Particles
    plt.cla()
    plt.quiver(x,y,vx,vy,angles)
    ax.set(xlim=(0,scale) , ylim = (0 , scale))
    ax.set_aspect('equal')
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    plt.pause(0.001)
plt.show()
```

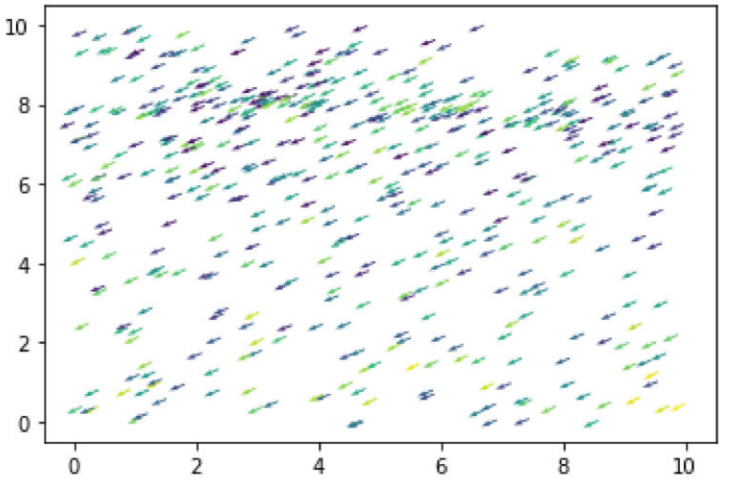

10
8
6
4
2
0
0
2
4
6
8
10

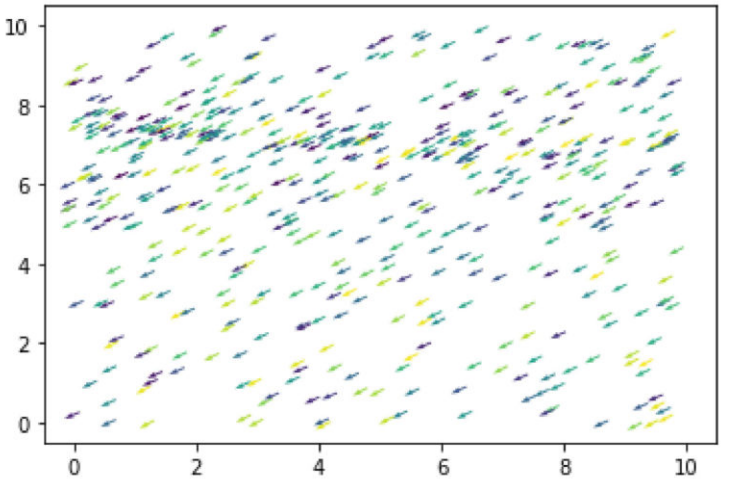

10
8
6
4
2
0
0
2
4
6
8
10

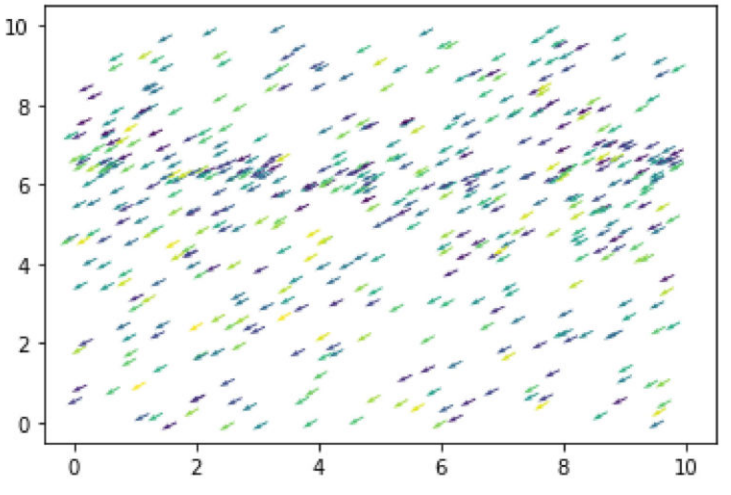

10
8
6
4
2
0
0
2
4
6
8
10

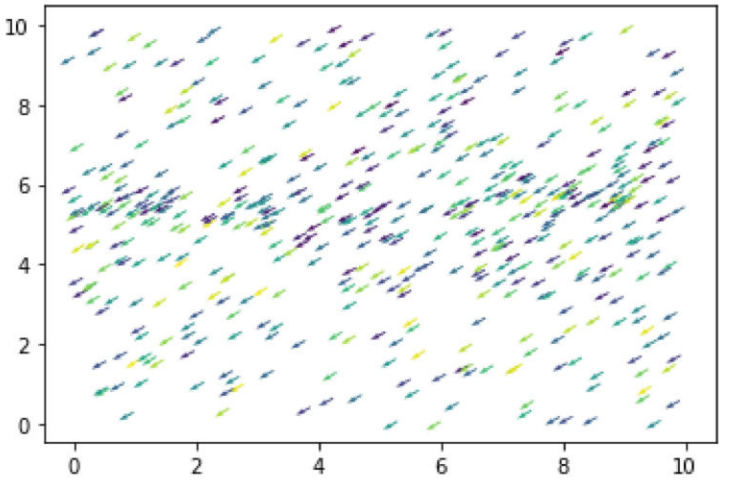

10
8
6
4
2
0
0
2
4
6
8
10

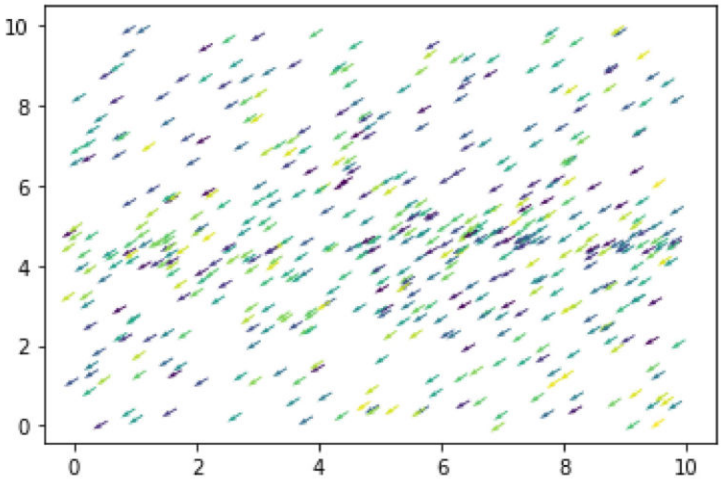

10
8
6
4
2
0
0
2
4
6
8
10


10
8
6
4
2
0
0
2
4
6
8
10


10
8
6
4
2
0
0
2
4
6
8
10


10
8
6
4
2
0
0
2
4
6
8
10

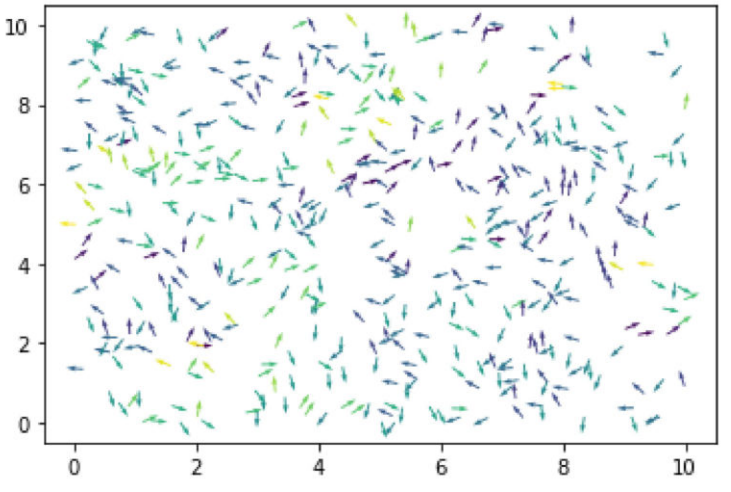

10
8
6
4
2
0
0
2
4
6
8
10

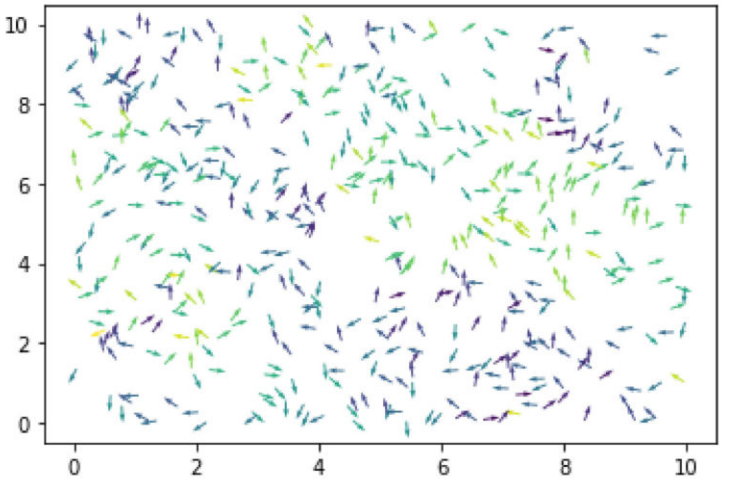


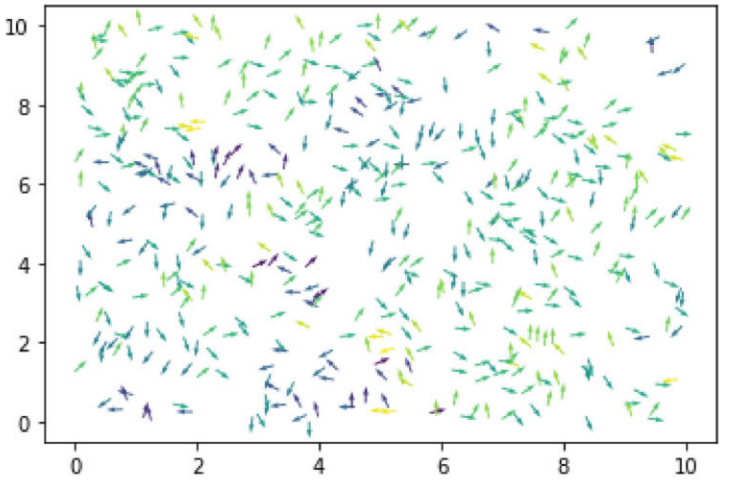
by decreasing of noise intensity particles motion became more ordered . fully plotted and animated graph of this section will be in "supplementary-material/fig4.mp4"

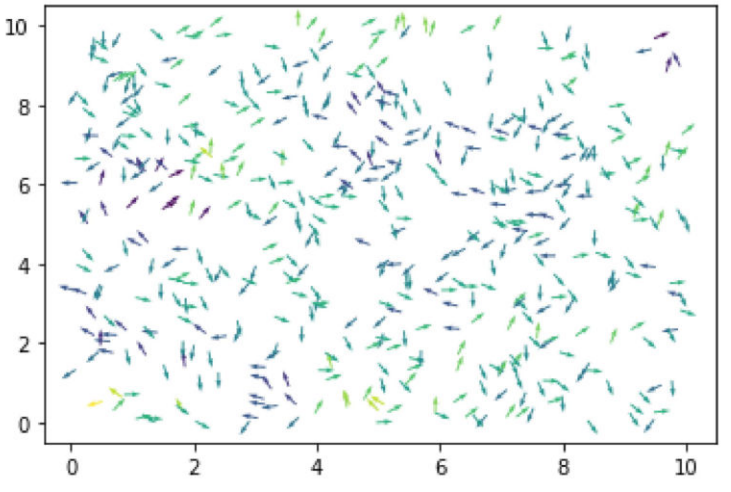
In [53]:

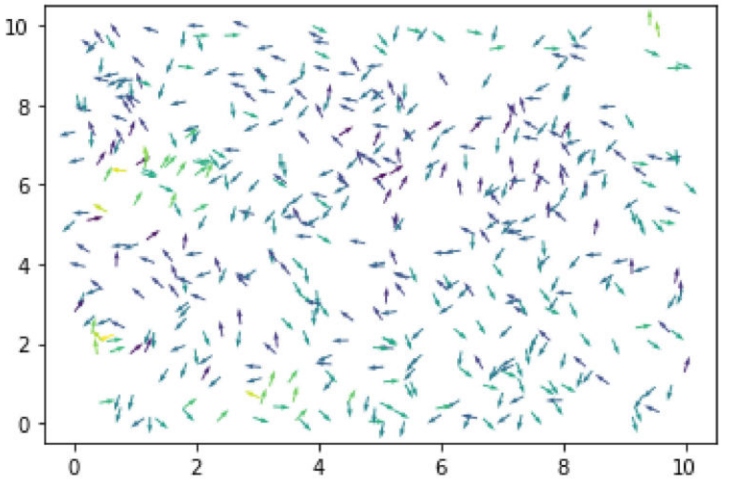
```
f_intensity = 4.0
np.random.seed(777)
x = np.random.rand(n_particles,1) * scale
y = np.random.rand(n_particles,1) * scale
angles = 2 * np.pi * np.random.rand(n_particles,1)
vx = v0 * np.cos(angles)
vy = v0 * np.sin(angles)
for i in range(10):
    x += vx * dt
    y += vy *dt
    #scaling new positions
    x = x % scale
    y = y % scale
    #Find mean angle of particles in defined Radius
    m_angles = angles
    for j in range(n_particles):
        neighbors = (x - x[j]) ** 2 + (y - y[j]) ** 2 < radius ** 2
        sx = np.sum(np.cos(angles[neighbors]))
        sy = np.sum(np.sin(angles[neighbors]))
        m_angles[j] = np.arctan2(sy,sx) #converting vectors to angles
    # Adding Noise to Angles
    angles = m_angles + f_intensity * (np.random.rand(n_particles,1) - f_intensit

    #new Velocities
    vx = v0 * np.cos(angles)
    vy = v0 * np.sin(angles)
    #Visualizing Motion Of Particles
    plt.cla()
    plt.quiver(x,y,vx,vy,angles)
    ax.set(xlim=(0,scale) , ylim = (0 , scale))
    ax.set_aspect('equal')
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    plt.pause(0.001)
plt.show()
```

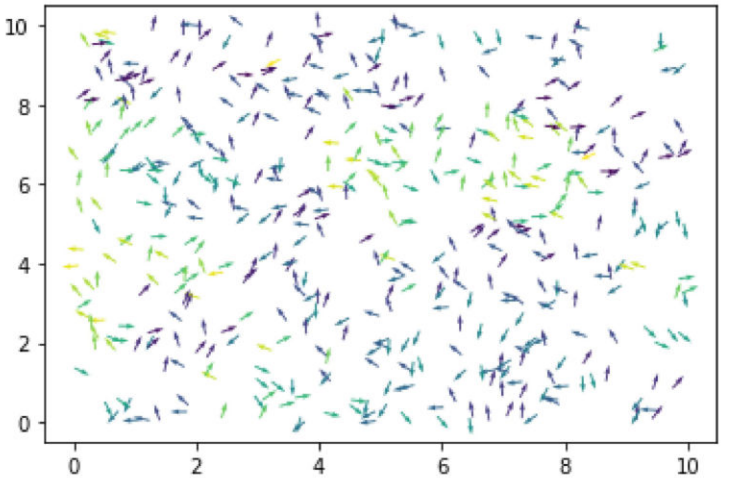

10
8
6
4
2
0
0
2
4
6
8
10

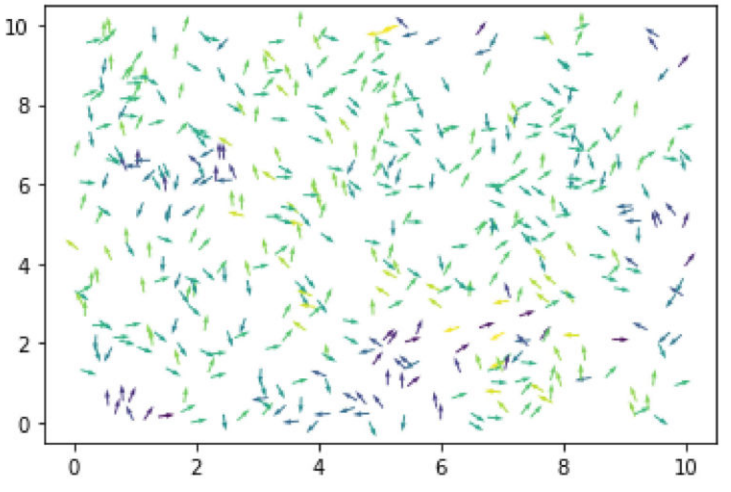

10
8
6
4
2
0
0
2
4
6
8
10

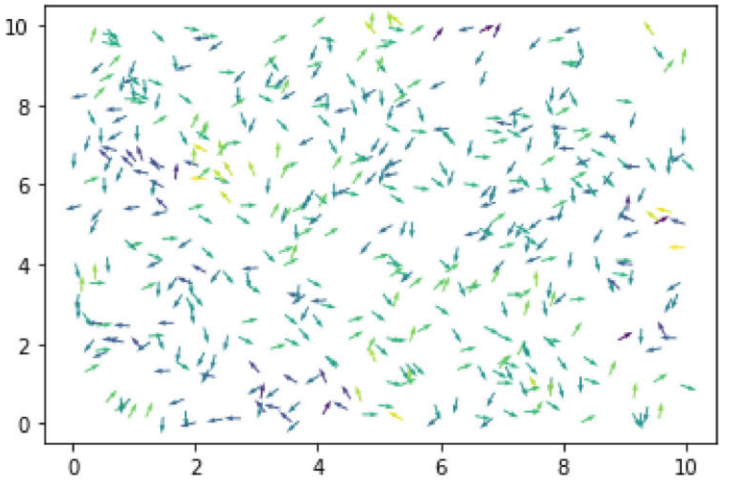

10
8
6
4
2
0
0
2
4
6
8
10

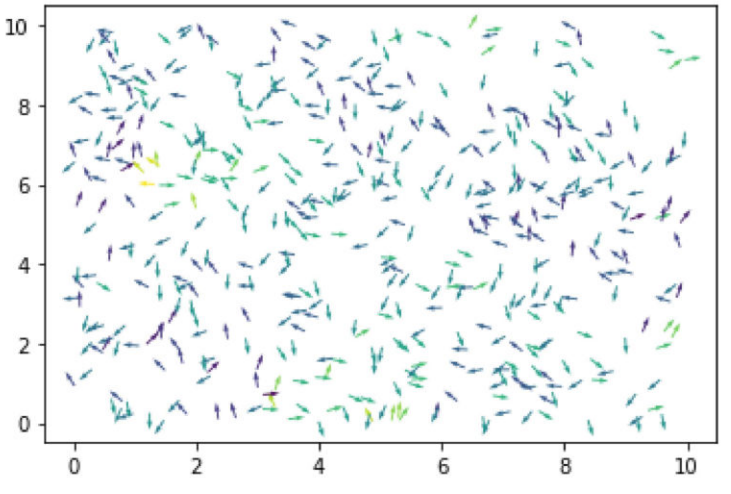

10
8
6
4
2
0
0
2
4
6
8
10

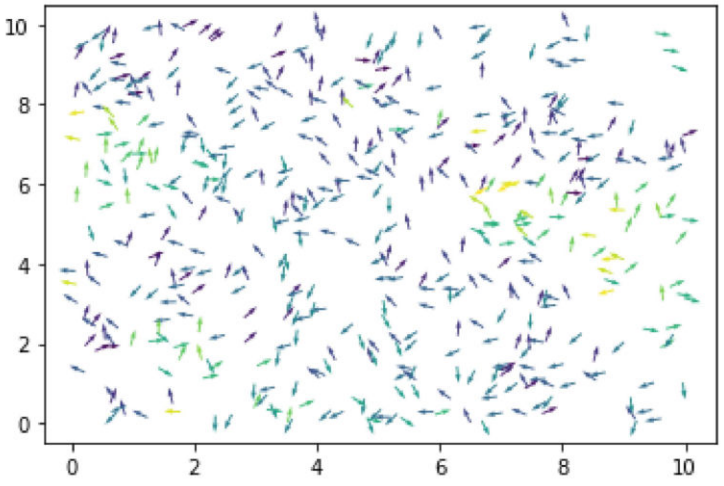

10
8
6
4
2
0
0
2
4
6
8
10


10
8
6
4
2
0
0
2
4
6
8
10


0
2
4
6
8
10


0
2
4
6
8
10


0
2
4
6
8
10



as you can see by increasing fluctuation intensity system became chaotic and there is less order . fully plotted and animated graph of this section will be in "supplementary-material/fig5.mp4"

conclusions

In this model we can see a phase transition. motion of our particles transits from a disordered and random motion to an scaled and ordered motion . also we can see that two parameters, noise intensity and radius, can affect the order of motions . At large noises and small radius , disorder and gap between particles or groups of particles appears. At low noises and large radius , order between particles or groups of particles appears and they globaly align and move in similar directions.