ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

Дисциплина: Моделирование беспроводных сетей

Студент: Дидусь Кирилл Валерьевич

Группа: НПМмд-02-22

In [68]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats
import math
from numpy.linalg import norm
from numpy import arccos, dot, pi, cross
```

In [69]:
```
guard_size=10 #размер области
poisson_lam_block= 0.3 # параметр распределения
radius_block=0.5
distance_Tx_Rx=4
list_point=np.array # массив для точек
```

In [70]:
```
def plot_circle(x,y,r):
    angles=np.linspace(0, 2*np.pi, 50)
    x_cir=x+r*np.cos(angles)
    y_cir=y+r*np.sin(angles)
    plt.plot(x_cir, y_cir, "red")
```

In [71]:
```
def poisson_point_process(lambda0, area_size):
    Number_block=np.random.poisson(lambda0*area_size**2) #количество блокаторов
    x=np.random.uniform(0, area_size, size=Number_block)
    y=np.random.uniform(0, area_size, size=Number_block)
    return x,y
```

In [72]:
```
def coord_rectangle(x_point_1, y_point_1, x_point_2, y_point_2, angle):
    difference_angle=2*np.pi-angle
    reverse_angle=np.pi/2 - difference_angle
    opposite_angle=reverse_angle+np.pi
    x_rectang_A=x_point_1+radius_block*np.cos(opposite_angle)
    y_rectang_A=y_point_1+radius_block*np.sin(opposite_angle)
    x_rectang_B=x_point_1+radius_block*np.cos(reverse_angle)
    y_rectang_B=y_point_1+radius_block*np.sin(reverse_angle)
    x_rectang_C=x_point_2+radius_block*np.cos(reverse_angle)
    y_rectang_C=y_point_2+radius_block*np.sin(reverse_angle)
    x_rectang_D=x_point_2+radius_block*np.cos(opposite_angle)
    y_rectang_D=y_point_2+radius_block*np.sin(opposite_angle)
    return (x_rectang_A, y_rectang_A, x_rectang_B, y_rectang_B,
            x_rectang_C, y_rectang_C, x_rectang_D, y_rectang_D)
```

In [73]:
```
#считаем расстояние через нормаль
def check_distance(A, B, C):
    CA=(C-A)/norm(C-A)
    BA=(B-A)/norm(B-A)
    CB=(C-B)/norm(C-B)
    AB=(A-B)/norm(A-B)
    if arccos(dot(CA, BA))>1:
        return norm(C-A)
    if arccos(dot(CB, AB))>1:
        return norm(C-B)
    return norm(cross(A-B, A-C))/norm(B-A)
```

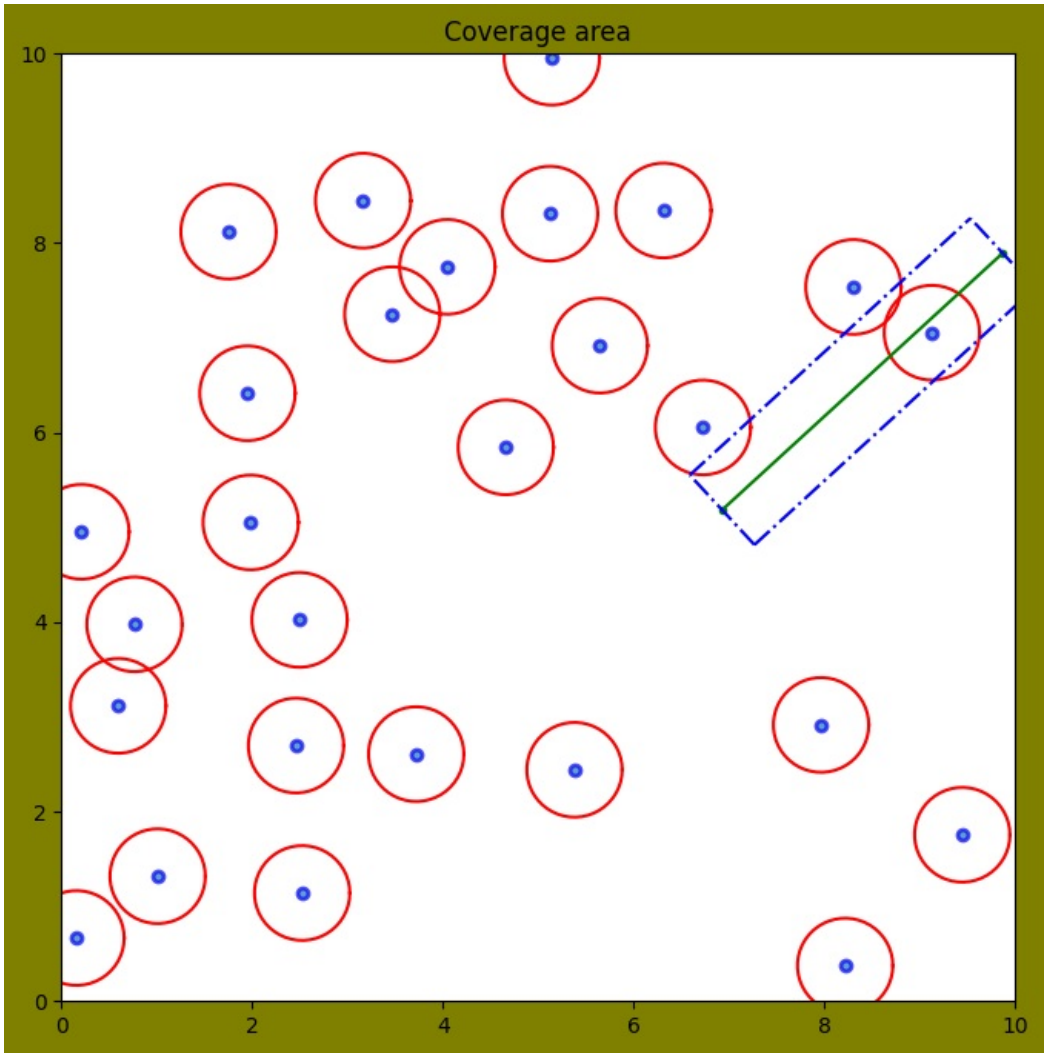
In [74]:
```
#проверка блокировки
def crossing():
    circle_point=[]
    point_1=[]
    point_2=[]
    point_1.extend([x1, y1])
    point_2.extend([x2, y2])
    for i in range (len(x)):
        circle_point.append([x[i], y[i]])
    for i in range (len(x)):
        if (np.round(check_distance(list_point(point_1), list_point(point_2),
                                    list_point(circle_point[i])), 1) <= radius_block):
            return True
```

In [75]:
```
x, y = poisson_point_process(poisson_lam_block, guard_size) #потенциальные блокаторы
x1=np.random.uniform (0, guard_size)
y1 = np.random.uniform (0, guard_size)
```

```
angle = np.random.uniform(0, 2*np.pi)
x2=x1+distance_Tx_Rx * np.cos(angle)
y2=y1 + distance_Tx_Rx * np.sin(angle)
(x_rectang_A, y_rectang_A, x_rectang_B, y_rectang_B, x_rectang_C, y_rectang_C, x_rectang_D,
 y_rectang_D)=coord_rectangle(x1, y1, x2, y2, angle)

plt.figure(dpi=100, figsize=(8,8), facecolor='olive')
plt.title('Coverage area')
plt.plot(x, y, '.', alpha=0.7, label='first', lw=5, mec='b', mew=2, ms=10)
for i in range (len(x)):
    plot_circle(x[i], y[i], radius_block)
plt.plot([x1,x2],[y1,y2],'.-g') #точки и линия прямой видимости

plt.plot([x_rectang_A, x_rectang_B], [y_rectang_A, y_rectang_B], '-.b')
plt.plot([x_rectang_A, x_rectang_D], [y_rectang_A, y_rectang_D], '-.b')
plt.plot([x_rectang_B, x_rectang_C], [y_rectang_B, y_rectang_C], '-.b')
plt.plot([x_rectang_D, x_rectang_C], [y_rectang_D, y_rectang_C], '-.b')
plt.xlim(0, guard_size)
plt.ylim(0, guard_size)
plt.show()
```



In [81]:
```
#события блокировки
if (crossing()):
    print('LoS is blocked')
else:
    print('LoS is not block')
```

```
LoS is blocked
```

In [82]:
```
#иммитационное моделирование
N = 1000 #@param {type:"integer"}
summa=0
for i in range(N):
    x, y = poisson_point_process(poisson_lam_block, guard_size)
    x1=np.random.uniform (0, guard_size)
    y1 = np.random.uniform (0, guard_size)
    angle = np.random.uniform(0, 2*np.pi)
    x2=x1+distance_Tx_Rx * np.cos(angle)
    y2=y1 + distance_Tx_Rx * np.sin(angle)
    if (crossing()):
        summa+=1
print("Вероятность блокировки: ", summa/N)
```

```
Вероятность блокировки:  0.672
```

In [83]:
```
#вероятность блокировки по аналитической формуле
S=2*radius_block*distance_Tx_Rx
lamb=poisson_lam_block*S
probability=1-np.exp(-lamb)
print("Вероятность блокировки теоретическая ", probability)
```

```
Вероятность блокировки теоретическая  0.6988057880877978
```

Вывод: полученная с помощью иммитационного моделирования вероятность блокировки близка к вычисленной с помощью математической модели вероятности.