b10505005_hw2_readme.pdf

Description

Use the file to plot figures of

- plot, received power of base station at (0,0) and SINR by considering other 18 base stations interference in other cell, using Two-ray-ground model
- plot, received power of mobile devices and SINR by considering other 49 mobile devices interference in the same cell, using Two-ray-ground model
- plot, received power of mobile devices and SINR by considering different mobile devices and different base station interference in 19 cells, using Two-ray-ground model

Requirement

Python 3.8.10

Main Function

a. basic plot of the coordinates of base station and mobile devices

```
def hexagonal_grid(ISD):
    coords = [(0, 0)]
    for i in range(6):
        angle = -np.pi / 6 + (np.pi / 3 * i)
        x = np.cos(angle) * ISD
        y = np.sin(angle) * ISD
        coords.append((x, y))

    for i in range(6):
        angle = (np.pi / 3 * i)
        x = np.cos(angle) * ISD * np.sqrt(3)
        y = np.sin(angle) * ISD * np.sqrt(3)
        coords.append((x, y))

    for i in range(6):
        angle = (-np.pi / 6+ np.pi / 3 * (i+1))
```

```
        x = np.cos(angle) * 1000
        y = np.sin(angle) * 1000
        coords.append((x, y))

    return np.array(coords)
```

```
def in_hexagon(x, y, radius):

    if np.abs(y) > radius * (np.sqrt(3)/2) and np.abs(x) <
0.5*radius:
        return False
    if np.abs(x) > radius - np.abs(y) * (1/np.sqrt(3)):
        return False
    else:
        return True
```

```
def hexagon_vertices(side_length):
    return [
        (side_length * np.cos(np.pi / 3 * i), side_length *
np.sin(np.pi / 3 * i))
        for i in range(6)
    ]
```

```
def generate_points_in_hexagon(num_points, radius):
    x_points = []
    y_points = []

    while len(x_points) < num_points:

        x = np.random.uniform(-radius * np.sqrt(3) / 2, rad
ius * np.sqrt(3) / 2)
        y = np.random.uniform(-radius, radius)

        if in_hexagon(x, y, radius):
            x_points.append(x)
            y_points.append(y)
```

```
    return np.array(x_points), np.array(y_points)
```

b. power calculation

```
def received_power_two_ray(d):
    return Pt * Gr * Gt * (h_device * h_base) ** 2 / (d **
4)
```

```
//qustion1
def calculate_interference(x_devices, y_devices, bs_coord
s):
    interference_power = np.zeros(len(x_devices))
    for x_bs, y_bs in bs_coords:
        distances_to_bs = np.sqrt((x_devices - x_bs) ** 2 +
(y_devices - y_bs) ** 2)
        Pr_interference = received_power_two_ray(distances_
to_bs)
        interference_power += Pr_interference
    return interference_power

//qustion2
  def calcculate_interference_power(D):
    sum = np.sum(D)
    print(sum)
    for i in range(num_devices):
        SINR = D[i]/(sum-D[i]+N)
        SINR_db = 10* np.log10(SINR)
        SINRs.append(SINR_db)
calcculate_interference_power(P_receive)

//question3
def calculate_power_indiff_bs(x_devices, y_devices, bs_coor
ds):
    Pr_power = []
    distances = []

    for i in range(0, len(bs_coords)):
```

```
        x_chunk = x_devices[i]
        y_chunk = y_devices[i]
        x_bs, y_bs = bs_coords[i]
        distances_to_bs = np.sqrt((x_chunk - x_bs) ** 2 +
(y_chunk - y_bs) ** 2)
        distances.append(distances_to_bs)
        power = received_power_two_ray(distances_to_bs)
        Pr_power.append(power)
    Pr_power1 = np.concatenate(Pr_power)
    distances = np.concatenate(distances)
    Pr_power_db = 10 * np.log10(Pr_power1)
    return Pr_power_db, distances, Pr_power1
```

Compile

```
  python 1.1.py
  python 2.py
  python 3.py
```

Contact：ESOE b10505005 蔣依倢