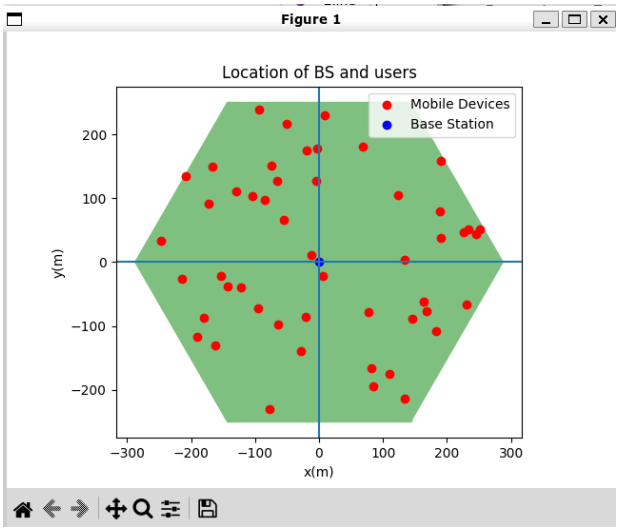
上次作業是將所有圖畫在同一個頁面上，但發現不易觀察。因此我這次一張圖就開一個視窗。以下圖的名字和run程式碼跑出來的圖名相同

(一)、

1.

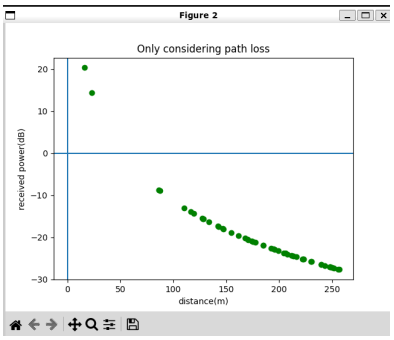


第一題畫圖，我還另外加了兩個三角形和一個長方形組成的六角形作為背景。

2.第二題用到two-ray ground的model，使用以下path loss的公式。和作業上次不一樣的是，這次有很多不同距離的Users。

$$g(d)=\frac{(h_t h_r)^2}{d^4} \qquad P_R = g(d)P_T G_T G_R$$

結果如下



我使用python的散佈圖的函式來畫圖

```
fig2 = plt.figure()
distance = []
powerReceived = []
for i in range(devices):
    distance.append((x[i]**2 + y[i]**2)**0.5)
    # The formula of two-ray ground model
    gainChannelDB = 10*m.log10((H_base*H_mobile)**2 / distance[i]**4)
    # Using sum in dB
    powerReceived.append(powerTx + gainRx + gainTx + gainChannelDB)

plt.scatter(distance, powerReceived, c='g')
plt.xlabel('distance(m)')
plt.ylabel('received power(dB)')
plt.title("Only considering path loss")
plt.axhline(y=0)
plt.axvline(x=0)
```

3.第三題要考慮不同cell的BS所造成的干擾，上課有說到，通常只考慮最近一圈的BS造成的干擾。所以我把最近六座BS的座標列出，要來計算距離。

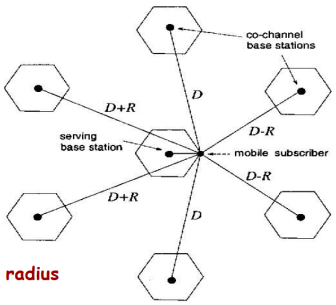
```
# The position of surrounding BS
BS_x = []
BS_x.append(-1.5 * cellRadius)
BS_x.append(0)
BS_x.append(1.5 * cellRadius)
BS_x.append(1.5 * cellRadius)
BS_x.append(0)
BS_x.append(-1.5 * cellRadius)
BS_y = []
BS_y.append(3**0.5 * cellRadius/2)
BS_y.append(3**0.5 * cellRadius)
BS_y.append(3**0.5 * cellRadius/2)
BS_y.append(-3**0.5 * cellRadius/2)
BS_y.append(-3**0.5 * cellRadius)
BS_y.append(-3**0.5 * cellRadius/2)

interference = []
```

inteference計算可以參考這個公式，但因為還要考慮thermal noise，所以還要修正一下。

Interference [CCI] is

$$\Lambda = \frac{1}{2}\frac{R^{-\beta}}{(D-R)^{-\beta}+D^{-\beta}+(D+R)^{-\beta}}$$

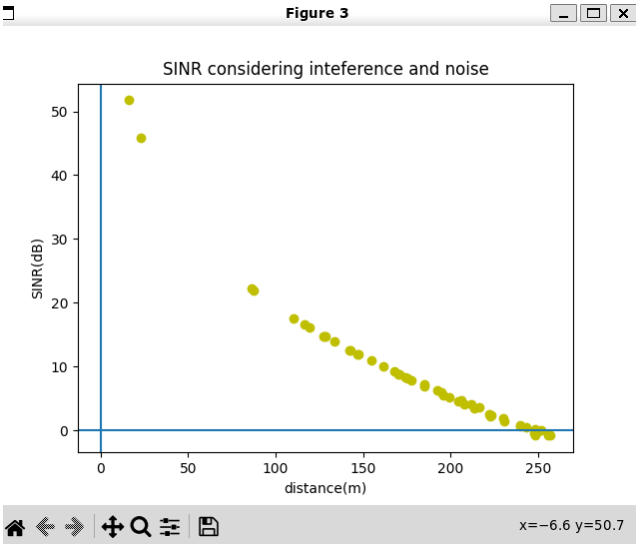
Worst case CCI
on the forward channel



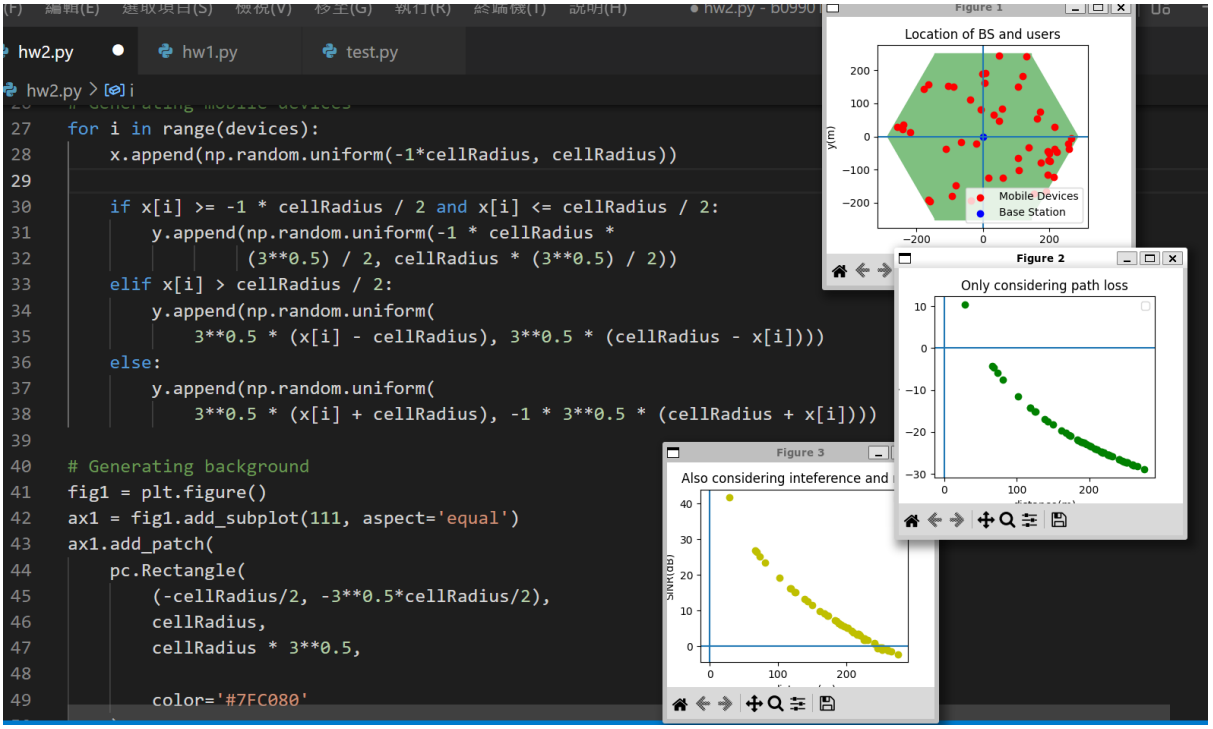
因為要把inteference和thermal noise相加，所以要把分貝換回W，相加後再換回分貝。最後SINR便是原訊號和雜訊相減(分貝)

```
for i in range(devices):
    # Interference from the 6 BS
    I = []
    for k in range(6):
        d = ((x[i] - BS_x[k])**2 + (y[i] - BS_y[k])**2)**0.5
        iInW = 10**(powerTx/10) * ((H_base*H_mobile)**2/d**4) * \
            10**(gainRx/10) * 10**(gainTx/10)
        # Turning the gain into W
        I.append(iInW)
    interference.append(sum(I) + thermalNoise)


# Turning W into dB
interference = [10 * m.log10(i) for i in interference]

SINR = []
for i in range(devices):
    SINR.append(powerReceived[i] - interference[i])
```

結果如下

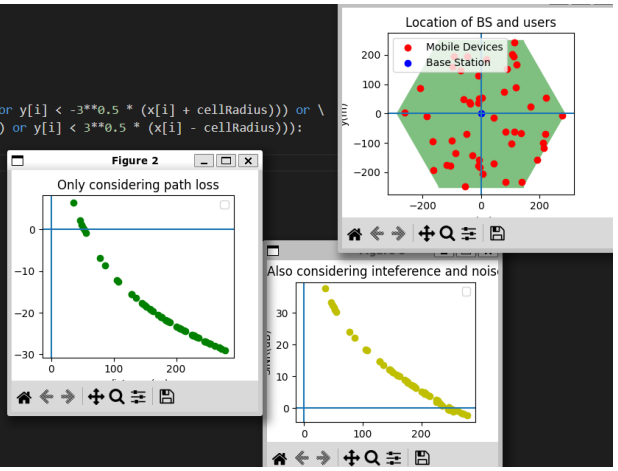


我做到這理發現我在邊邊上的點特別多，回去看了一下自己選users位置的方式是不uniform的，原本是先隨機挑x的座標軸再去根據六邊形的邊界去限制y座標，但這樣x軸偏左和偏右的地方會比較多點點(原本超出的都只能往內縮多比較小的區域)。如下圖



於是我x和y都一起考慮，一起隨機選，超出範圍就重新挑一個點。實驗結果有稍微好一些，邊邊的點少一些，離中間比較近的點多一些。如下圖。

```
# Generating mobile devices
for i in range(devices):
    x.append(np.random.uniform(-1*cellRadius, cellRadius))
    y.append(np.random.uniform(-3**0.5 * cellRadius/2, 3**0.5 * cellRadius/2))
    while ((x[i] <= -1 * cellRadius / 2) and (y[i] > 3**0.5 * (x[i] + cellRadius) or y[i] < -3**0.5 * (x[i] + cellRadius))) or \
            ((x[i] >= 1 * cellRadius / 2) and (y[i] > -3**0.5 * (x[i] - cellRadius) or y[i] < 3**0.5 * (x[i] - cellRadius))):
        x.pop(i)
        y.pop(i)
        x.append(np.random.uniform(-1*cellRadius, cellRadius))
        y.append(np.random.uniform(-3**0.5 * cellRadius/2, 3**0.5 * cellRadius/2))
    '''if x[i] >= -1 * cellRadius / 2 and x[i] <= cellRadius / 2:
        y.append(np.random.uniform(-1 * cellRadius *
                 (3**0.5) / 2, cellRadius * (3**0.5) / 2))
    elif x[i] > cellRadius / 2:
        y.append(np.random.uniform(
            3**0.5 * (x[i] - cellRadius), 3**0.5 * (cellRadius - x[i])))
    else:
        y.append(np.random.uniform(
            3**0.5 * (x[i] + cellRadius), -1 * 3**0.5 * (cellRadius + x[i])))'''

# Generating background
fig1 = plt.figure()
ax1 = fig1.add_subplot(111, aspect='equal')
ax1.add_patch(
    pc.Rectangle(
```

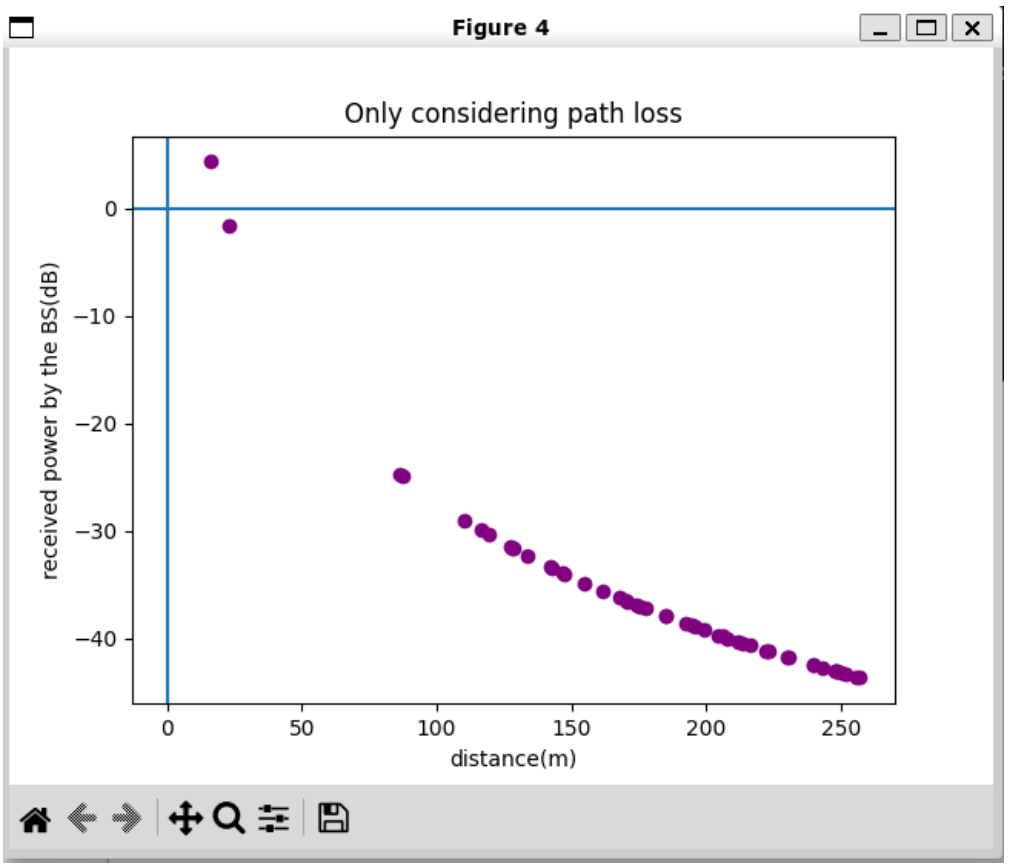


(二)、

1.我用和(一)1.一樣的圖

2.這題是用BS當receiver，所以power要換成mobile device。結果如下

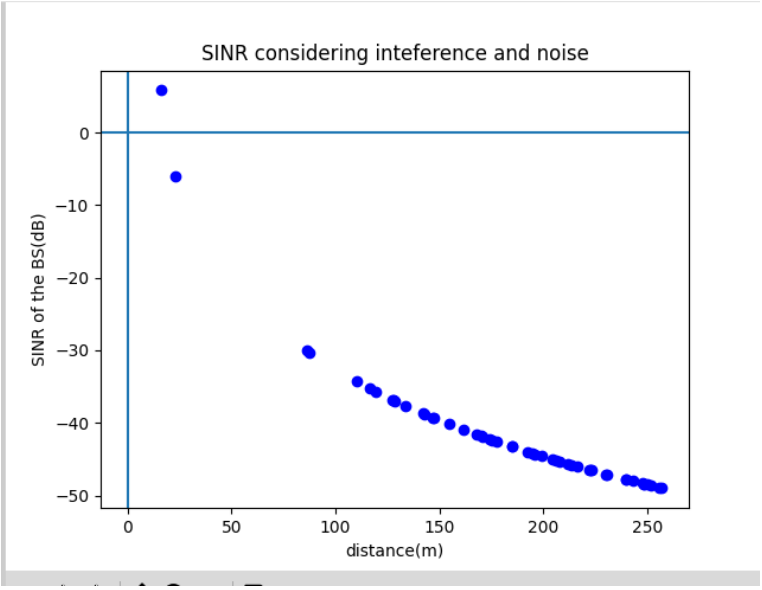
$$g(d) = \frac{(h_t h_r)^2}{d^4} \quad P_R = g(d) P_T G_T G_R$$



3.這次要考慮其他users也用同一個頻段的干擾。所以我先將BS能收到的所有訊號能量相加，再減去作為分子的我們關心的user發出的能量，再加上thermal noise。

```
fig5 = plt.figure()

# Interference from the 50 users
interferenceByMs = []
SINROfBS = []
powerReceivedByBSInW = [10**(i/10) for i in powerReceivedByBS]
sumOfPower = sum(powerReceivedByBSInW)
for i in range(devices):
    # Turning into dB; All the signals are inteference except for the original one
    interferenceByMs.append(
        10 * m.log10(sumOfPower - 10**(powerReceivedByBS[i]/10) + thermalNoise))
    SINROfBS.append(powerReceivedByBS[i] - interferenceByMs[i])

plt.scatter(distance, SINROfBS, c='blue')
plt.xlabel('distance(m)')
plt.ylabel('SINR of the BS(dB)')
plt.title("SINR considering inteference and noise")
plt.axhline(y=0)
plt.axvline(x=0)
```

結果如下



BONUS

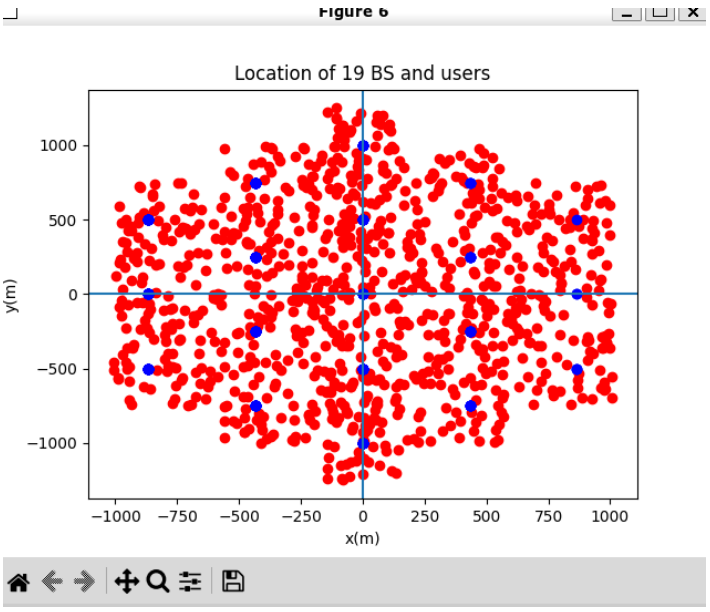
我的基本概念和(一)1.相同，但根據每一個cell的中心點不同，去做平移

```
fig6 = plt.figure()
c = -1
for a in range(5):
    for k in range(b[a]):
        c += 1
        for i in range(devices):

            X[c].append(np.random.uniform(-1*cellRadius + (a+1 - 3) * 1.5 *
                                cellRadius, cellRadius + (a+1 - 3) * 1.5 * cellRadius))

            Y[c].append(np.random.uniform(-3**0.5 * cellRadius/2 + (k-int(b[a]/2))*3**0.5*cellRadius + (a % 2)*3**0.5 *
                                cellRadius/2, 3**0.5 * cellRadius/2 + (k-int(b[a]/2))*3**0.5*cellRadius + (a % 2)*3**0.5*cellRadi

            while ((X[c][i] <= -1 * cellRadius / 2 + (a+1 - 3) * 1.5 * cellRadius) and (Y[c][i] > 3**0.5 * (X[c][i] + (a+1 - 3) * 1.5 * cel
                   ((X[c][i] >= 1 * cellRadius / 2 + (a+1 - 3) * 1.5 * cellRadius) and (Y[c][i] > -3**0.5 * (X[c][i] + (a+1 - 3) * 1.5 * c

                X[c].pop(i)
                Y[c].pop(i)

                X[c].append(np.random.uniform(-1*cellRadius +
                                (a+1 - 3) * 1.5 * cellRadius, cellRadius + (a+1 - 3) * 1.5 * cellRadius))

                Y[c].append(np.random.uniform(-3**0.5 * cellRadius/2 + (k-int(b[a]/2))*3**0.5*cellRadius + (a % 2)*3**0.5 *
                                cellRadius/2, 3**0.5 * cellRadius/2 + (k-int(b[a]/2))*3**0.5*cellRadius + (a % 2)*3**0.5*cell

    #plt.scatter(X, Y, c='r', label="Mobile Devices")
```

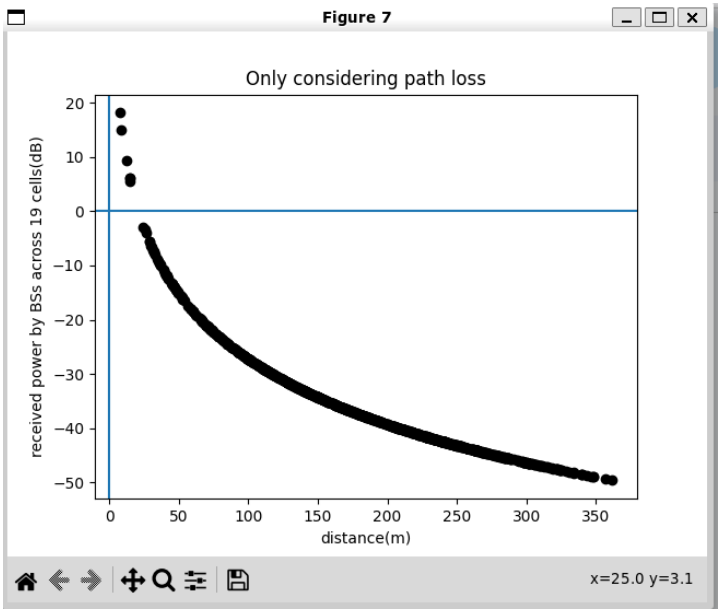
我一開始如上圖程式碼，是將平移的算式全部列出，但還想想其實可以些先寫好，可以比簡潔好除錯。(如下圖center_x,y)

```
for i in range(numberOfCell):
    plt.scatter(X[i], Y[i], c='r', label="Mobile Devices")
center_x = []
center_y = []
for a in range(5):
    for k in range(b[a]):
        center_x.append((a+1 - 3) * 1.5 * cellRadius)
        center_y.append((k-int(b[a]/2))*3**0.5 *
                        cellRadius + (a % 2)*3**0.5 * cellRadius/2)
        plt.scatter(center_x,  center_y, c='b', label="Base Station")
plt.xlabel('x(m)')
plt.ylabel('y(m)')
plt.title("Location of 19 BS and users")
plt.axhline(y=0)
plt.axvline(x=0)
```

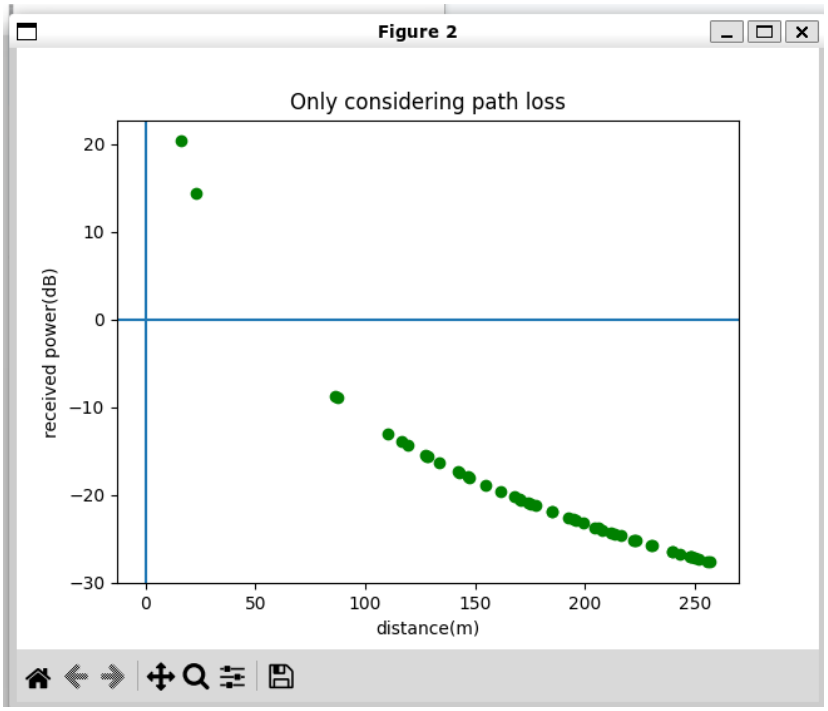
結果如下



2.和(二)2.很像，只是有更多的數據點(因為這題不同cell並沒有干擾，結果如下。



和第二題大致吻合，而且數據更詳細



3.這題則是要考慮所有cell的users，我在每一個cell都開了紀錄一個會被其它cell的50個device干擾。我一樣把一個BS收到的所有干擾相加再減去我們要求的訊號的power。這次也要把分貝換回W，再和noise相加。

```
# Interference from the 50 users
interferenceBonus = []
IBonus = []
SINRBonus = []
for i in range(numberOfCell):
    interferenceBonus.append([])
    SINRBonus.append([])
    for k in range(numberOfCell):
        for j in range(devices):
            # 50 devices for each cell
            interferenceBonus[i].append((H_base*H_mobile)**2/((X[k][j] - center_x[i])**2 + (
                # Turning dB into W
                Y[k][j]-center_y[i])**2)**2 * 10**((powerRx + gainRx + gainTx)/10) + thermalNoise)
    # Sum of inteference 50 devices for 19 cells
    IBonus.append(sum(interferenceBonus[i]))
    print(i)
    print(IBonus[i])

    for l in range(devices):
        SINRBonus[i].append(powerBonus[i][l] - 10 *
                            m.log10(IBonus[i] - 10**(powerBonus[i][l]/10)))

for i in range(numberOfCell):
    col = (np.random.random(), np.random.random(), np.random.random())
    plt.scatter(distanceBonus[i], SINRBonus[i],
                c=col, label=f"{i}line")
```

而我又特別把不同條線用不同顏色區分，可以看到每一區因為local users和BS的距離、和其他cell地理位置的影響所以呈現不一樣的趨勢。結果如下。