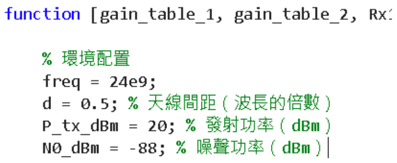
Commit 1: finish task with weight sum=1：把 task1-3 完成，並且有*1/N

Commit 2: finish comparison and not *1/N：把 comparison 完成

Commit 3: final_tx_power/2 把 task 全部打包成 function，如下面所示

(Task 1-3 all in function beamforming_simulation)

先把 TA section 裡面的配置照搬過去



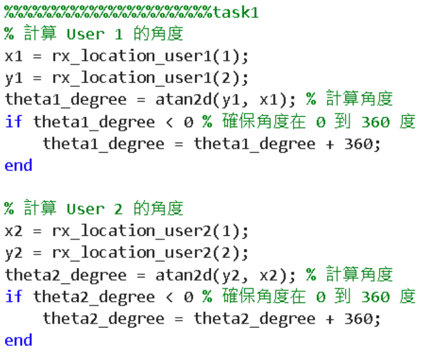
```
function [gain_table_1, gain_table_2, Rx

    % 環境配置
    freq = 24e9;
    d = 0.5; % 天線間距（波長的倍數）
    P_tx_dBm = 20; % 發射功率（dBm）
    N0_dBm = -88; % 噪聲功率（dBm）
```

Task 1

TODO: Find the actual angle θ1and θ2 for user 1 and user 2：

用 atan2d 直接找 user 實際度

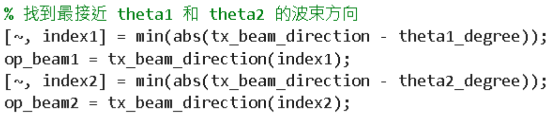


```
%%%%%%%%%%%%%%%%%%%%%%task1
% 計算 User 1 的角度
x1 = rx_location_user1(1);
y1 = rx_location_user1(2);
theta1_degree = atan2d(y1, x1); % 計算角度
if theta1_degree < 0 % 確保角度在 0 到 360 度
    theta1_degree = theta1_degree + 360;
end

% 計算 User 2 的角度
x2 = rx_location_user2(1);
y2 = rx_location_user2(2);
theta2_degree = atan2d(y2, x2); % 計算角度
if theta2_degree < 0 % 確保角度在 0 到 360 度
    theta2_degree = theta2_degree + 360;
end
```

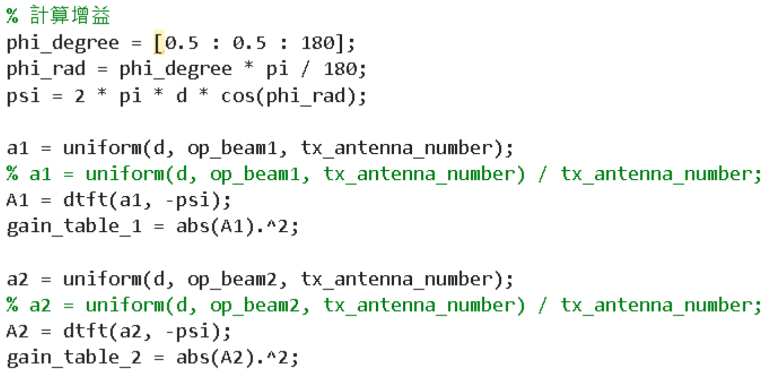
TODO: Find the beam with the closest beam angle：

直接找 code book 裡面角度跟 user1(theta1_degree) user2(theta2_degree) 最接近的 beam



```
% 找到最接近 theta1 和 theta2 的波束方向
[~, index1] = min(abs(tx_beam_direction - theta1_degree));
op_beam1 = tx_beam_direction(index1);
[~, index2] = min(abs(tx_beam_direction - theta2_degree));
op_beam2 = tx_beam_direction(index2);
```

Cont. 計算剛剛 optimal beam 的 gain table 並且畫出來



```
% 計算增益
phi_degree = [0.5 : 0.5 : 180];
phi_rad = phi_degree * pi / 180;
psi = 2 * pi * d * cos(phi_rad);

a1 = uniform(d, op_beam1, tx_antenna_number);
% a1 = uniform(d, op_beam1, tx_antenna_number) / tx_antenna_number;
A1 = dtft(a1, -psi);
gain_table_1 = abs(A1).^2;

a2 = uniform(d, op_beam2, tx_antenna_number);
% a2 = uniform(d, op_beam2, tx_antenna_number) / tx_antenna_number;
A2 = dtft(a2, -psi);
gain_table_2 = abs(A2).^2;
```

phi_degree、phi_rad 將 0.5 度到 180 度切分為 360 等分

psi: 計算每個角度對應的相位移，公式為 $2\pi d\cos(\phi)$

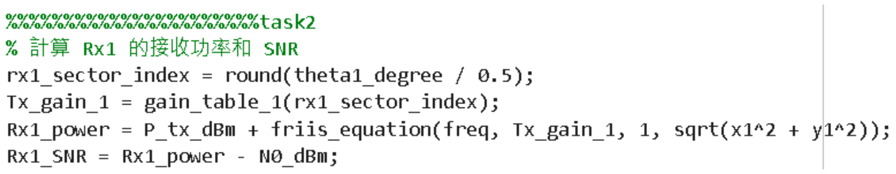
，其中 d 是天線之間的距離。

Uniform 生成均勻間隔的 array weight (a)

dtft(ai, -psi): 計算 array factor (A)

$g(\phi) = |A(\phi)|^2 = \left|a_0 + a_1 e^{jkd\cos\phi}\right|^2$ 最後取決對質平方拿到 gain

Task 2



```
%%%%%%%%%%%%%%%%%%%%%%task2
% 計算 Rx1 的接收功率和 SNR
rx1_sector_index = round(theta1_degree / 0.5);
Tx_gain_1 = gain_table_1(rx1_sector_index);
Rx1_power = P_tx_dBm + friis_equation(freq, Tx_gain_1, 1, sqrt(x1^2 + y1^2));
Rx1_SNR = Rx1_power - N0_dBm;
```

TODO: Identify the Tx gain of users 1 from the optimal beam θ1*

從剛剛的 gain table 找到 user1 對應的 Tx gain

Gain table index：1-360 對應到 resolution360 (0.5：0.5：180)

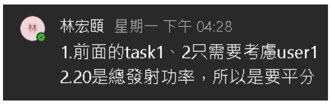
我選擇先乘以二再四捨五入

TODO: Calculate the Rx power and SNRdB for user 1

帶入 friis equation 算出 pathloss，再加上 Tx power 算出 Rx power，並用固定的 noise power 算出 SNR

Task 3 我後來改成有平分 Tx power，report 寫完才看到教授說這次不用平分，

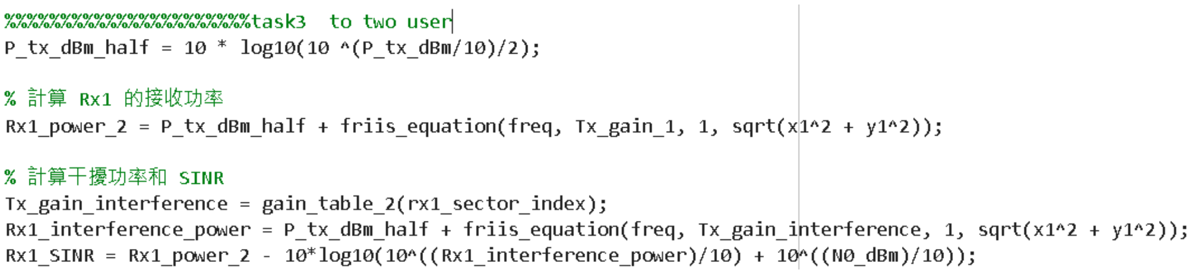
所以我還是有平分 power(只有 task3)



林宏頤 星期一 下午 04:28

1.前面的task1、2只需要考慮user1

2.20是總發射功率，所以是要平分



```
%%%%%%%%%%%%%%%%%%%%%%task3  to two user
P_tx_dBm_half = 10 * log10(10 ^(P_tx_dBm/10)/2);

% 計算 Rx1 的接收功率
Rx1_power_2 = P_tx_dBm_half + friis_equation(freq, Tx_gain_1, 1, sqrt(x1^2 + y1^2));

% 計算干擾功率和 SINR
Tx_gain_interference = gain_table_2(rx1_sector_index);
Rx1_interference_power = P_tx_dBm_half + friis_equation(freq, Tx_gain_interference, 1, sqrt(x1^2 + y1^2));
Rx1_SINR = Rx1_power_2 - 10*log10(10^((Rx1_interference_power)/10) + 10^((N0_dBm)/10));
```

先算出一半的 Tx power 是多少(換成 mW 除以二再換回 dBm)

用新的 Tx power 重算 user 1 的 Rx power

TODO: Identify the Tx gain of users 2 from the optimal beam θ2*

跟 task 2 一樣找到對應的 Tx gain (瞄準 user2 時，user1 方向的 gain)

TODO: Calculate the interference power and SINRdB for user 1

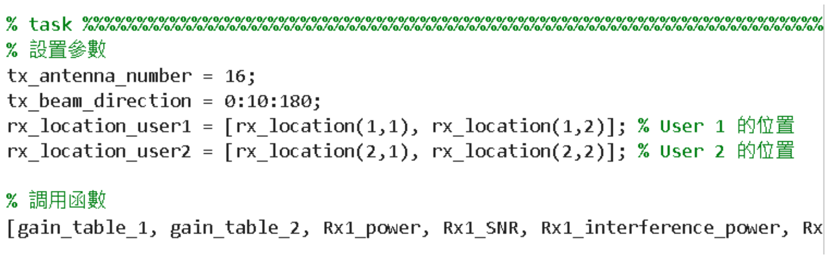
跟 task 2 一樣找到 user1 的 interference power

換成 mW 算出 SINR 再換回 dBm

以上均為 function 裡面的內容，我是把需要用到的答案都傳到外面，再畫圖

配置不同參數並帶入 function：

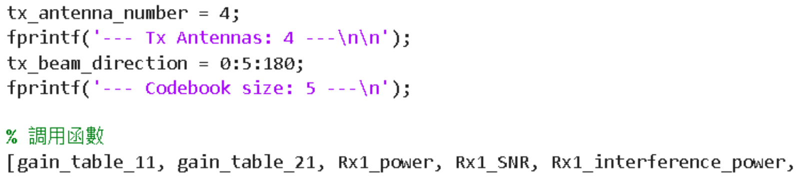
--task



```
% task %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 設置參數
tx_antenna_number = 16;
tx_beam_direction = 0:10:180;
rx_location_user1 = [rx_location(1,1), rx_location(1,2)]; % User 1 的位置
rx_location_user2 = [rx_location(2,1), rx_location(2,2)]; % User 2 的位置

% 調用函數
[gain_table_1, gain_table_2, Rx1_power, Rx1_SNR, Rx1_interference_power, Rx
```

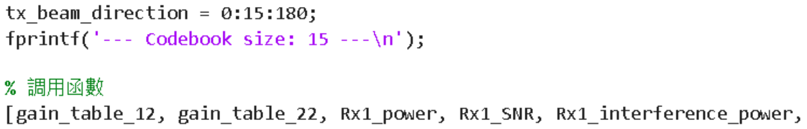
--comparison



```
tx_antenna_number = 4;
fprintf('--- Tx Antennas: 4 ---\n\n');
tx_beam_direction = 0:5:180;
fprintf('--- Codebook size: 5 ---\n');

% 調用函數
[gain_table_11, gain_table_21, Rx1_power, Rx1_SNR, Rx1_interference_power,
```

……

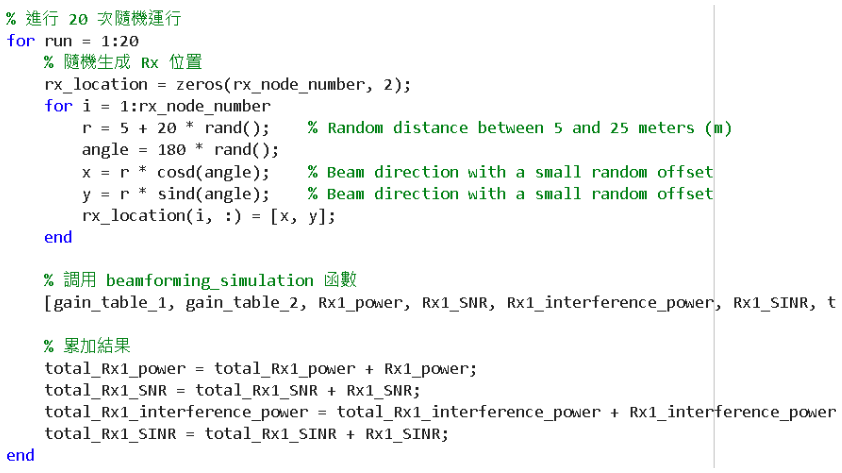


```
tx_beam_direction = 0:15:180;
fprintf('--- Codebook size: 15 ---\n');

% 調用函數
[gain_table_12, gain_table_22, Rx1_power, Rx1_SNR, Rx1_interference_power,
```

各紀錄 user1 user2 的四個狀況的 gain table 再畫成圖表(四條線在同一張)

--random 20 runs



```
% 進行 20 次隨機運行
for run = 1:20
    % 隨機生成 Rx 位置
    rx_location = zeros(rx_node_number, 2);
    for i = 1:rx_node_number
        r = 5 + 20 * rand();    % Random distance between 5 and 25 meters (m)
        angle = 180 * rand();
        x = r * cosd(angle);    % Beam direction with a small random offset
        y = r * sind(angle);    % Beam direction with a small random offset
        rx_location(i, :) = [x, y];
    end

    % 調用 beamforming_simulation 函數
    [gain_table_1, gain_table_2, Rx1_power, Rx1_SNR, Rx1_interference_power, Rx1_SINR, t

    % 累加結果
    total_Rx1_power = total_Rx1_power + Rx1_power;
    total_Rx1_SNR = total_Rx1_SNR + Rx1_SNR;
    total_Rx1_interference_power = total_Rx1_interference_power + Rx1_interference_power
    total_Rx1_SINR = total_Rx1_SINR + Rx1_SINR;
end
```

照搬 TA section 生成位置的 code，帶入 function 並累計結果再平均