ECE 233 Project Report

Hybrid Digital and Analog Beamforming Design for Large-Scale Antenna Arrays

Ruoye Wang | 605625594

Yida Chen | 005852117

Abstract

The massive multiple-input multiple-output (MIMO) concept has been widely studied and developed. It enables large-scale spatial multiplexing and highly directional beamforming. This project implements a two-stage hybrid beamforming architecture in MATLAB, utilizing the system model and algorithms described in the paper “Hybrid Digital and Analog Beamforming Design for Large-Scale Antenna Arrays” by Foad Sohrabi and Wei Yu.

In this project, the overall beamformer is composed of a digital beamformer, followed by an RF beamformer that uses analog phase shifters for implementation. The finite resolution effect of analog phase shifters is studied. Two scenarios are considered. Firstly, the number of RF chains at the base station (BS) and user end is equal to the number of data streams ($N_t^{RF} = N_r^{RF} = N^{RF} = N_S$). The second scenario is where the number of NF chains is larger than the number of data streams but smaller than twice the total number of data streams ($N_S < N^{RF} < 2N_S$). This report presents the simulation outcomes obtained using MATLAB. Our findings closely correspond to the results documented in the reference paper.

System Model

This project implements a model proposed by the reference paper. This narrowband downlink single-cell multi-user MIMO system model has a two-stage hybrid digital and analog beamforming architecture at the base station (BS) and the user terminals.

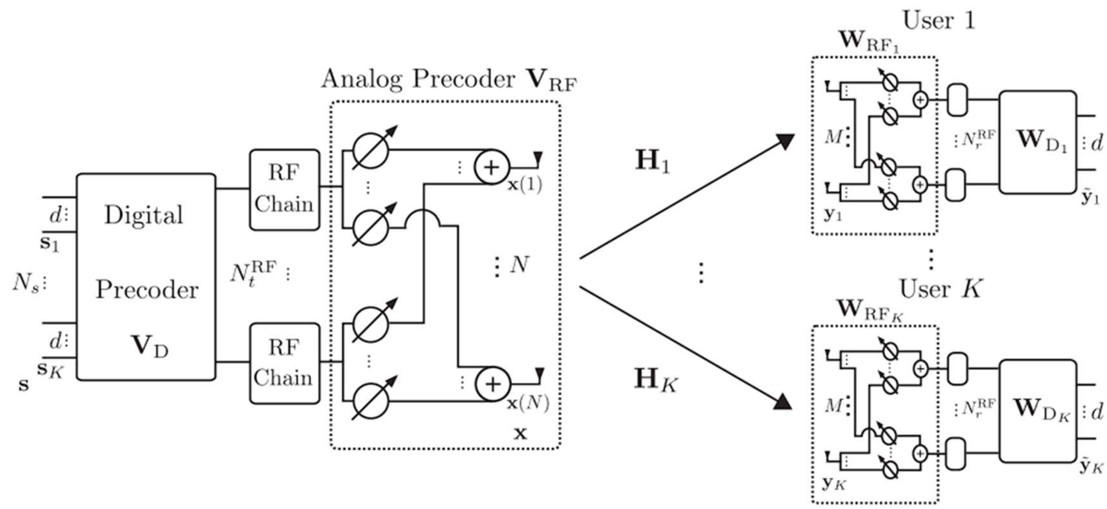

Fig. 1. Block diagram of a multi-user MIMO system with hybrid beamforming architecture at the BS and the user terminals.

Figure 1: Block diagram of the model proposed by the paper

As the figure above shows, the BS has N antennas and N_t^{RF} RF chains and serves K users. Each user is equipped with M antennas and N_r^{RF} RF chains and requires d data streams. The number of data streams required by each user $d \leq N_r^{RF} \leq M$; the total number of data streams $N_S = Kd \leq N_t^{RF} \leq N$.

This project assumes single-user scenario, i.e., $K = 1$. To simplify the notation while preserving the generality, it is assumed that $N_t^{RF} = N_r^{RF} = N^{RF}$. The project first implements the hybrid beamformer design for the case where $N^{RF} = N_S$ to show that, according to Proposition 1, a fully digital beamformer architecture can be realized by a hybrid structure with at least N_S RF chains using a proposed heuristic algorithm. Then the same algorithm is implemented for the case where $N_S < N^{RF} < 2N_S$.

The symbols V_D, V_{RF}, W_D, W_{RF} represent the digital precoder at the BS (size $N_t^{RF} \times N_S$), the RF precoder at the BS (size $N \times N_t^{RF}$), the digital combiner at the user end (size $N_r^{RF} \times N_S$), and the RF combiner at the user end (size $M \times N_r^{RF}$). H

(size $M \times N$) is the matrix of the complex channel gains from the transmit antennas of the BS to the user (note that since $K = 1$, all H_k can be represented by a single H; the same can be applied to other user-specific quantities in the paper).

Then, the transmitted signal can be represented by:

$$\mathbf{x} = \mathbf{V}_{\mathrm{RF}}\mathbf{V}_{\mathrm{D}}\mathbf{s} = \sum_{\ell=1}^{K} \mathbf{V}_{\mathrm{RF}}\mathbf{V}_{\mathrm{D}_\ell}\mathbf{s}_\ell$$

Figure 2: Transmitted signal

Where s is the vector of data symbols. And the final processed signals are obtained as:

$$\widetilde{\mathbf{y}}_k = \mathbf{W}_{t_k}^{H}\left(\mathbf{H}_k\mathbf{V}\mathbf{s} + \mathbf{z}_k\right)$$

Figure 3: Final processed signal

Where z_k represents the additive white Gaussian noise.

Main Part

This paper mainly focuses on maximizing the overall spectral efficiency with total transmit power constrained and H fully known. This requires us to find the optimal solution for precoders at the transmitter end and the combiners at the receiver end, which can be represented by this formula:

$$\begin{aligned}
&\underset{\mathbf{V}_{\mathrm{RF}},\mathbf{V}_{\mathrm{D}},\mathbf{W}_{\mathrm{RF}},\mathbf{W}_{\mathrm{D}}}{\text{maximize}} && \sum_{k=1}^{K} \beta_k R_k && \text{(5a)}\\
&\text{subject to} && \mathrm{Tr}(\mathbf{V}_{\mathrm{RF}}\mathbf{V}_{\mathrm{D}}\mathbf{V}_{\mathrm{D}}^H\mathbf{V}_{\mathrm{RF}}^H) \le P && \text{(5b)}\\
& && |\mathbf{V}_{\mathrm{RF}}(i,j)|^2 = 1, \forall i,j && \text{(5c)}\\
& && |\mathbf{W}_{\mathrm{RF}_k}(i,j)|^2 = 1, \forall i,j,k, && \text{(5d)}
\end{aligned}$$

Figure 4: the formula representing the main problem (K=1, disregarding k)

This formula is calculated under the aforementioned cases $N^{RF} = N_S$ and $N_S < N^{RF} < 2N_S$. It can be simplified for precoder design:

$$\begin{aligned}
\max_{\mathbf{V}_{\mathrm{RF}},\mathbf{V}_{\mathrm{D}}} \quad & \log_2\left|\mathbf{I}_M + \frac{1}{\sigma^2}\mathbf{H}\mathbf{V}_{\mathrm{RF}}\mathbf{V}_{\mathrm{D}}\mathbf{V}_{\mathrm{D}}^H\mathbf{V}_{\mathrm{RF}}^H\mathbf{H}^H\right|\\
\text{s.t.} \quad & \mathrm{Tr}(\mathbf{V}_{\mathrm{RF}}\mathbf{V}_{\mathrm{D}}\mathbf{V}_{\mathrm{D}}^H\mathbf{V}_{\mathrm{RF}}^H) \le P,\\
& |\mathbf{V}_{\mathrm{RF}}(i,j)|^2 = 1, \quad \forall i,j.
\end{aligned}$$

Figure 5: Precoder design formula

$N^{RF} = N_S$ scenario

When $N^{RF} = N_S$, V_{RF} can be calculated by:

$$\begin{aligned}
\max_{\mathbf{V}_{\mathrm{RF}}} \quad & \log_2\left|\mathbf{I} + \frac{\gamma^2}{\sigma^2}\mathbf{V}_{\mathrm{RF}}^H\mathbf{F}_1\mathbf{V}_{\mathrm{RF}}\right|\\
\text{s.t.} \quad & |\mathbf{V}_{\mathrm{RF}}(i,j)|^2 = 1, \forall i,j,
\end{aligned}$$

where $\mathbf{F}_1 = \mathbf{H}^H\mathbf{H}$.

Figure 6: RF precoder design formula

Which is a simplification of Figure. 5 assuming $V_D V_D^H \approx \gamma^2 I$. It is summarized in Algorithm 1:

The formula to calculate $V_{RF}(i, j)$ is under the assumption of infinite phase shifters. Since accurate phase shifters can be expensive to implement in real world, they are commonly replaced by cost effective low resolution phase shifters. To simulate such phase shifters, the design of V_{RF} is quantized:

Algorithm 1. Design of $\mathbf{V}_{\mathrm{RF}}$ by solving (12)

Given: $\mathbf{F}_1, \gamma^2, \sigma^2$

1: Initialize $\mathbf{V}_{\mathrm{RF}} = \mathbf{1}_{N \times N^{\mathrm{RF}}}$.
2: **for** $j = 1 \to N^{\mathrm{RF}}$ **do**
3: Calculate $\mathbf{C}_j = \mathbf{I} + \frac{\gamma^2}{\sigma^2} (\bar{\mathbf{V}}_{\mathrm{RF}}^j)^H \mathbf{F}_1 \bar{\mathbf{V}}_{\mathrm{RF}}^j$.
4: Calculate $\mathbf{G}_j = \frac{\gamma^2}{\sigma^2} \mathbf{F}_1 - \frac{\gamma^4}{\sigma^4} \mathbf{F}_1 \bar{\mathbf{V}}_{\mathrm{RF}}^j \mathbf{C}_j^{-1} (\bar{\mathbf{V}}_{\mathrm{RF}}^j)^H \mathbf{F}_1$.
5: **for** $i = 1 \to N$
6: Find $\eta_{ij} = \sum_{\ell \neq i} \mathbf{G}_j(i, \ell) \mathbf{V}_{\mathrm{RF}}(\ell, j)$.
7: $\mathbf{V}_{\mathrm{RF}}(i, j) = \begin{cases} 1, & \text{if } \eta_{ij} = 0, \\ \frac{\eta_{ij}}{|\eta_{ij}|}, & \text{otherwise.} \end{cases}$
8: **end for**
9: **end for**
10: Check convergence. If yes, stop; if not go to Step 2.

Figure 7: Algorithm 1 to calculate RF precoder, with infinite resolution phase shifter

$$\mathbf{V}_{RF}(i, j) = Q\left(\frac{\eta_{ij}}{|\eta_{ij}|}\right) = \underset{\mathbf{V}_{RF}(i,j) \in \mathcal{F}}{\arg\min} \left| \mathbf{V}_{RF}(i, j) - \frac{\eta_{ij}}{|\eta_{ij}|} \right|^2$$

Figure 8: RF precoder design, with 1-bit resolution phase shifter

Then V_D can be calculated by solving:

$$\begin{aligned} \max_{\mathbf{V}_{\mathrm{D}}} \quad & \log_2 \left| \mathbf{I}_M + \frac{1}{\sigma^2} \mathbf{H}_{\mathrm{eff}} \mathbf{V}_{\mathrm{D}} \mathbf{V}_{\mathrm{D}}^H \mathbf{H}_{\mathrm{eff}}^H \right| \\ \text{s.t.} \quad & \mathrm{Tr}(\mathbf{Q} \mathbf{V}_{\mathrm{D}} \mathbf{V}_{\mathrm{D}}^H) \leq P, \end{aligned}$$

Figure 9: Digital precoder design formula

Where $Q = V_{RF}^H V_{RF}$ and $H_{eff} = H V_{RF}$.

We use a water-filling solution (Reference: Waterfilling) to allocate power P_i to each channel so that the overall channel capacity is maximized. The sum of power P_i satisfies:

$$\sum_{i=1}^{m} P_i = P$$

Figure 10: Constraint on power allocation

Where the power of the first m most efficient channels sum up to use P to the largest extend. Since power cannot be negative:

$$P_i = \max\left(\left(\mu - \frac{\sigma^2}{h_i}\right), 0\right)$$

Figure 11: Formula to calculate allocated power

Combining the above two formulas, μ the water level chosen to satisfy the power sum constraints with equality can be represented by:

$$\mu = \frac{P + \sum_{i=1}^{m} \frac{\sigma^2}{h_i}}{m}$$

Figure 12: Formula to calculate water level

m is obtained by iterating through the number of total channels N_S. In each iteration, μ is calculated using the above formula, and compared with $\frac{\sigma^2}{h_i}$, where $\sigma^2 = \frac{P}{SNR_{linear}}$ represents noise and h_i is the square of right singular value of the SVD result of $H_{eff}Q^{-1/2}$, since in Figure 9, the matrices are applied twice on both left and right. Similarly, the diagonal elements of Γ_e are all square roots of P_i. (Reference: Ruifu Donar Li)

The final value of m is chosen so that μ satisfies the power constraint, as the following figure shows:

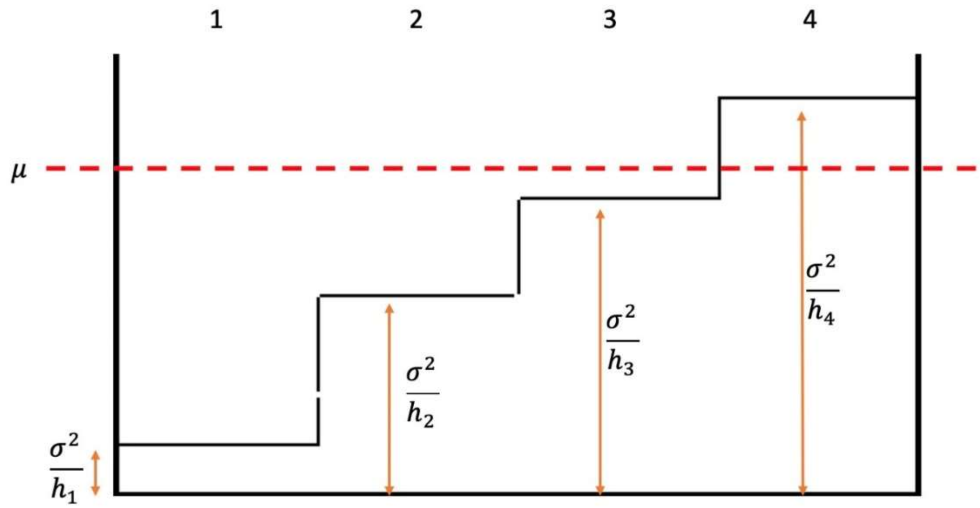

Figure 13: Water-filling solution

Then the matrix of digital precoder can be calculated using the formula proposed by the paper:

$$V_D = Q^{-1/2} U_e \Gamma_e$$

Where U_e is the set of right singular vectors corresponding to the N_S largest singular values of $H_{eff} Q^{-1/2}$ and Γ_e is the diagonal matrix of allocated powers to each stream.

With V_D, V_{RF} designed, the formula in Figure 3 can be simplified to:

$$\begin{aligned} \max_{\mathbf{W}_{\mathrm{RF}}} \quad & \log_2 \left| \mathbf{I} + \frac{1}{\sigma^2} (\mathbf{W}_{\mathrm{RF}}^H \mathbf{W}_{\mathrm{RF}})^{-1} \mathbf{W}_{\mathrm{RF}}^H \mathbf{F}_2 \mathbf{W}_{\mathrm{RF}} \right| \\ \text{s.t.} \quad & |\mathbf{W}_{\mathrm{RF}}(i,j)|^2 = 1, \forall i,j, \end{aligned}$$

Figure 14: RF combiner design formula

Due to its similarity to Figure 6, it can also be solved with Algorithm 1, with F_1 replaced by $F_2 = HV_tV_t^H H^H, V_t = V_{RF}V_D$ and $(W_{RF}^H W_{RF})^{-1} = \frac{1}{M}$ since M is large. The quantized W_{RF} is also calculated with the same method as V_{RF}.

Finally, the optimal digital combiner can be obtained from its MMSE solution:

$$\mathbf{W}_{\mathrm{D}} = \mathbf{J}^{-1}\mathbf{W}_{\mathrm{RF}}^{H}\mathbf{H}\mathbf{V}_{\mathrm{t}},$$

$$\text{where } \mathbf{J} = \mathbf{W}_{\mathrm{RF}}^{H}\mathbf{H}\mathbf{V}_{\mathrm{t}}\mathbf{V}_{\mathrm{t}}^{H}\mathbf{H}^{H}\mathbf{W}_{\mathrm{RF}} + \sigma^{2}\mathbf{W}_{\mathrm{RF}}^{H}\mathbf{W}_{\mathrm{RF}}$$

Figure 15: Digital combiner design formula

After V_D, V_{RF}, W_D, W_{RF} have been obtained, the performance of the model can be evaluated by its spectral efficiency:

$$R = \log_2 \left| \mathbf{I}_M + \frac{1}{\sigma^2}\mathbf{W}_{\mathrm{t}}(\mathbf{W}_{\mathrm{t}}^{H}\mathbf{W}_{\mathrm{t}})^{-1}\mathbf{W}_{\mathrm{t}}^{H}\mathbf{H}\mathbf{V}_{\mathrm{t}}\mathbf{V}_{\mathrm{t}}^{H}\mathbf{H}^{H} \right|$$

$$\text{where } \mathbf{V}_{\mathrm{t}} = \mathbf{V}_{\mathrm{RF}}\mathbf{V}_{\mathrm{D}} \text{ and } \mathbf{W}_{\mathrm{t}} = \mathbf{W}_{\mathrm{RF}}\mathbf{W}_{\mathrm{D}}.$$

Figure 16: Formula of spectral efficiency

The above process is repeated for a range of SNR, and for each SNR value, spectral efficiency is averaged over 100 Monte Carlo trials.

$N_S < N^{RF} < 2N_S$ scenario

As stated in the paper, this scenario can still be implemented using the aforementioned process. It is implemented for $N^{RF} = \{N_S,\ N_S + 1, N_S + 3\}$.

Results and Discussion

1. Plot the spectral efficiency vs. SNR in the range −10 dB to 6 dB, assuming a 64×16 MIMO system and $N^{RF} = N_S = 6$.

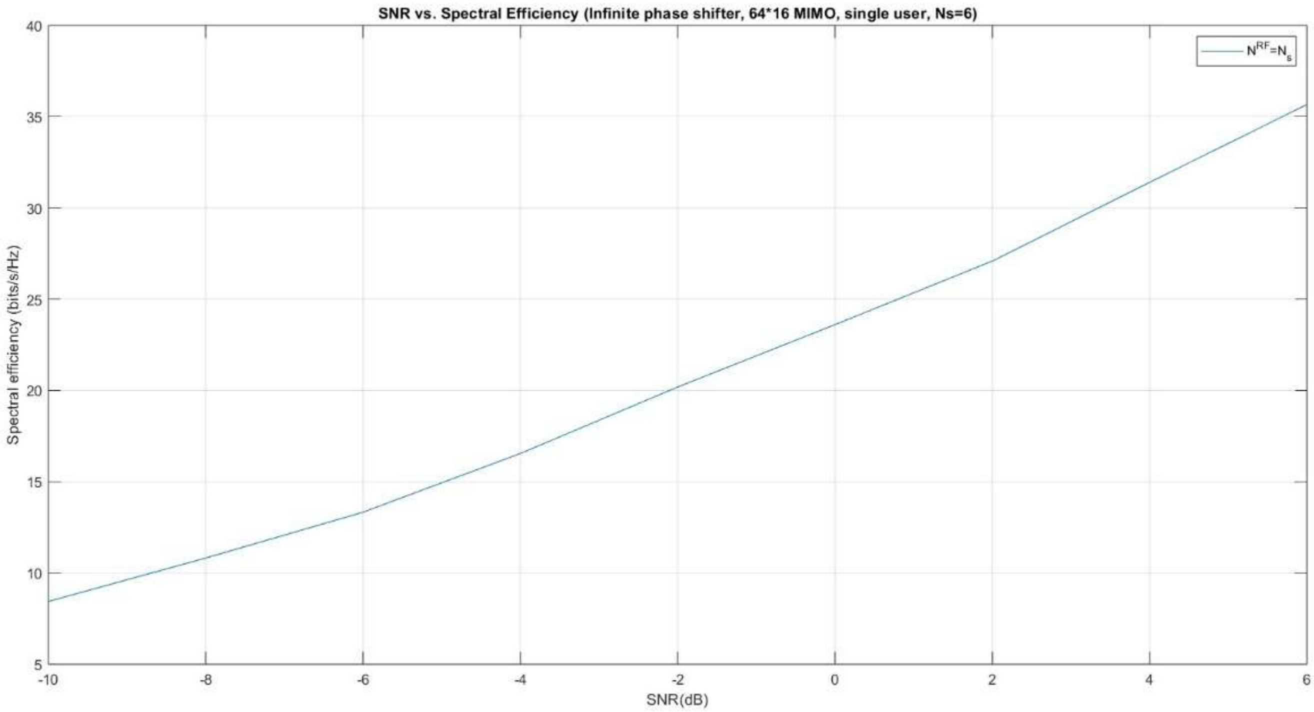

Figure 17: Spectral efficiency vs. SNR for result 1

The resulting figure is similar to Fig. 2 in the paper, with ~1.5dB difference on SNR values.

2. Plot the spectral efficiency vs. SNR in the range 0 dB to 30 dB, assuming a 10 × 10 MIMO system, $N^{RF} = N_S = 2$, and phase shifters with 1-bit and infinite resolutions.

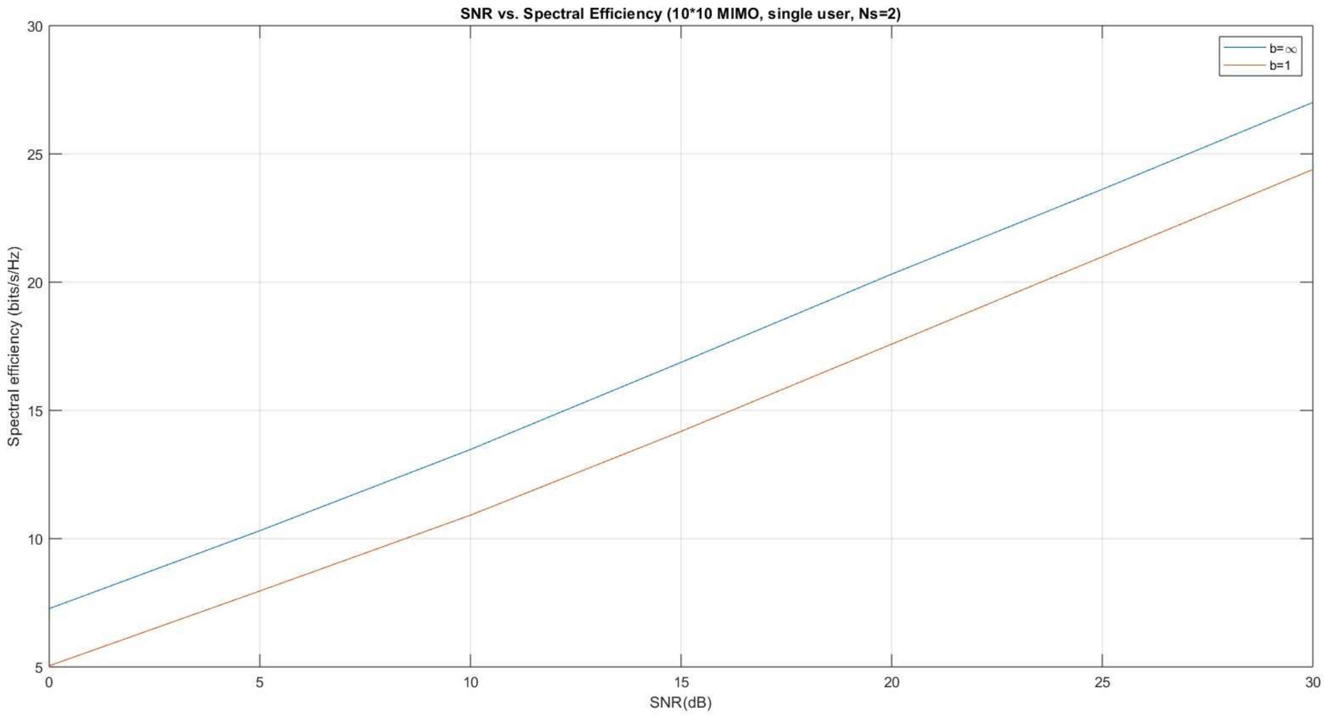

Figure 18: Spectral efficiency (infinite resolution and 1-bit phase shifters) vs. SNR for result 2

In paper Fig. 3, the spectral efficiency of 1-bit phase shifter is higher than that of infinite resolution phase shifter. However, this resulting figure shows the opposite. One possible reason is that, the paper's beamformer is designed under the assumption of infinite resolution phase shifters and each entry of the RF beamformer is quantized to the nearest point of the set of possible phases. However, in this project, the beamformers for infinite resolution phase shifter is not quantized.

3. Plot the spectral efficiency vs. SNR in the range −10 dB to 6 dB, assuming a 64 × 16 MIMO system, $N_S = 4, N^{RF} = \{N_S,\ N_S + 1, N_S + 3\}$, and phase shifters with 1-bit and infinite resolutions.

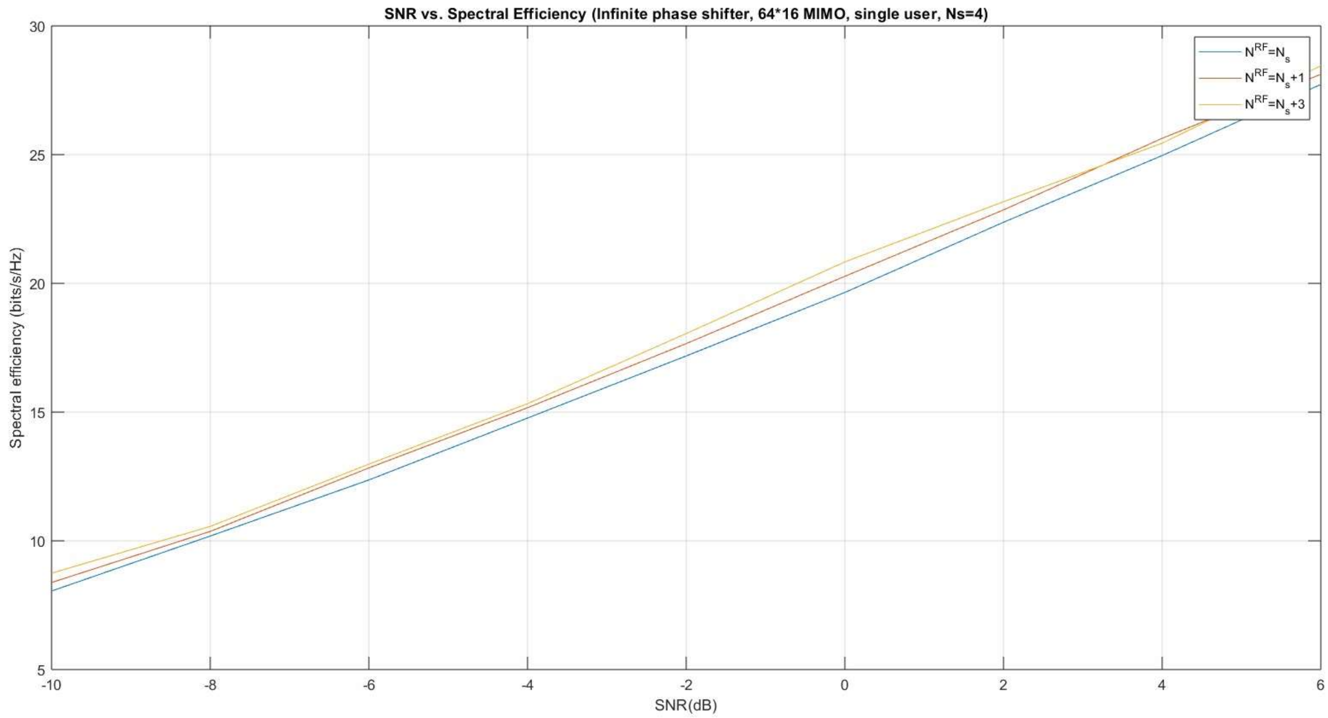

Figure 19: Spectral efficiency (Infinite resolution phase shifter) vs. SNR, 100 Monte Carlo trials

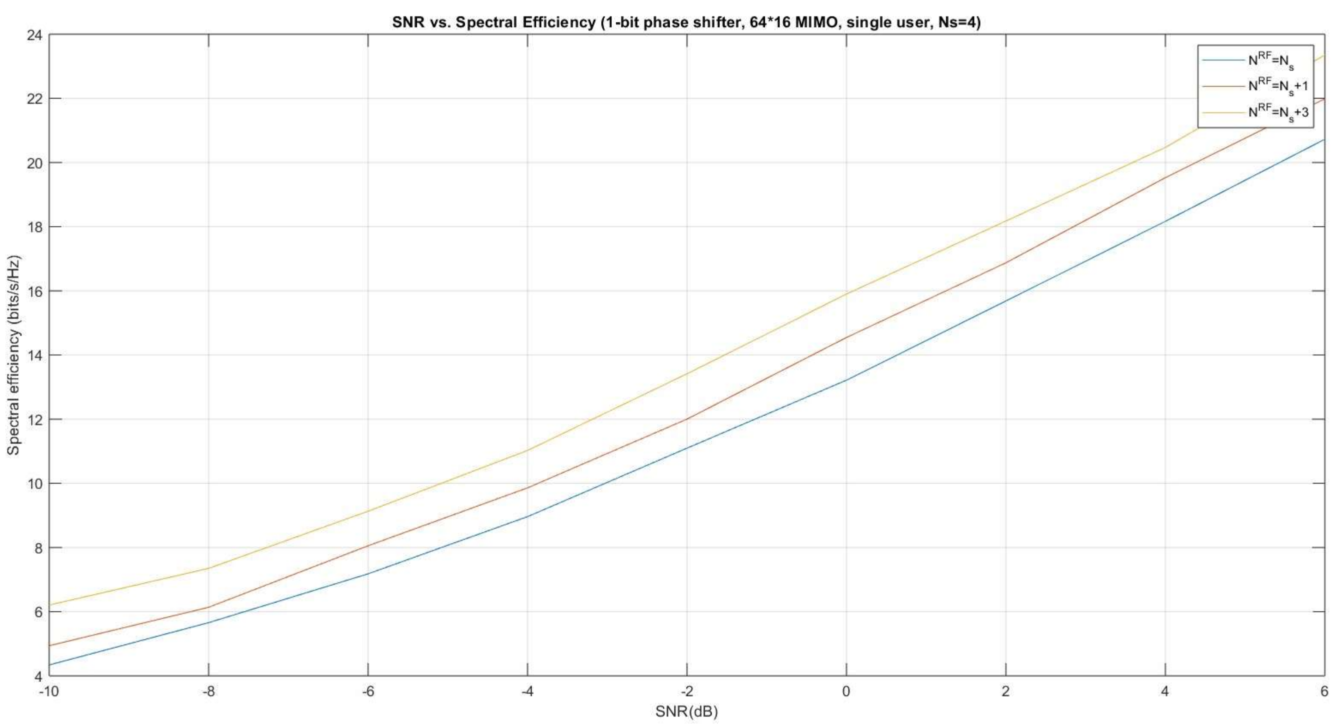

Figure 20: Spectral efficiency (1-bit phase shifter) vs. SNR, 100 Monte Carlo trials

The curves for 1-bit phase shifter are ~1.5dB lower than paper Fig. 4 which is considered acceptable error. Same as Fig. 4, the curves for infinite phase shifters (plotted separately for clarity) are significantly higher than that of 1-bit phase shifters, and the spectral efficiency increases as the number of RF chains increase, so that 1-bit phase shifter with more RF chains has a performance close to infinite phase shifters. This is in accordance with the paper's conclusion that the number of RF chains can be used to trade off the accuracy of phase shifters in hybrid beamforming design.

The above curves are rather unsmooth because the number of Monte Carlo trials are small. After the number of trials is increased to 1000, the figures become:

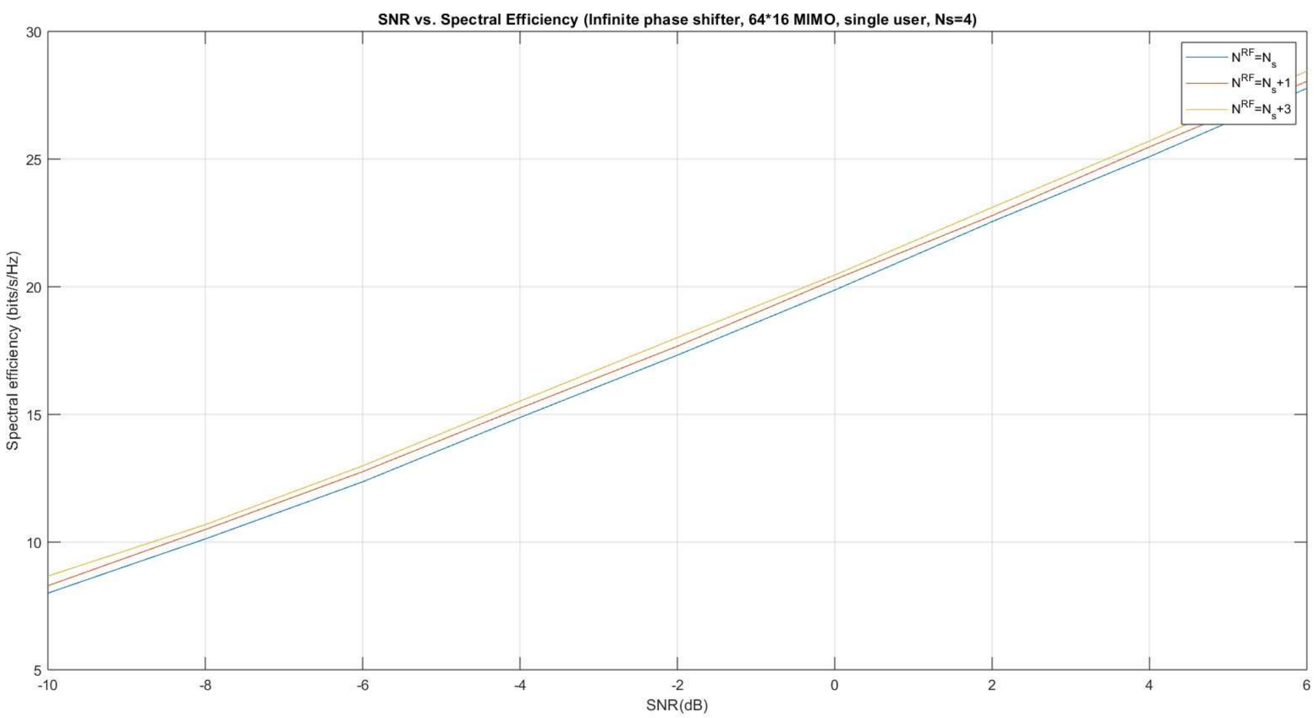

Figure 21: Spectral efficiency (infinite resolution phase shifter) vs. SNR, 1000 Monte Carlo trials

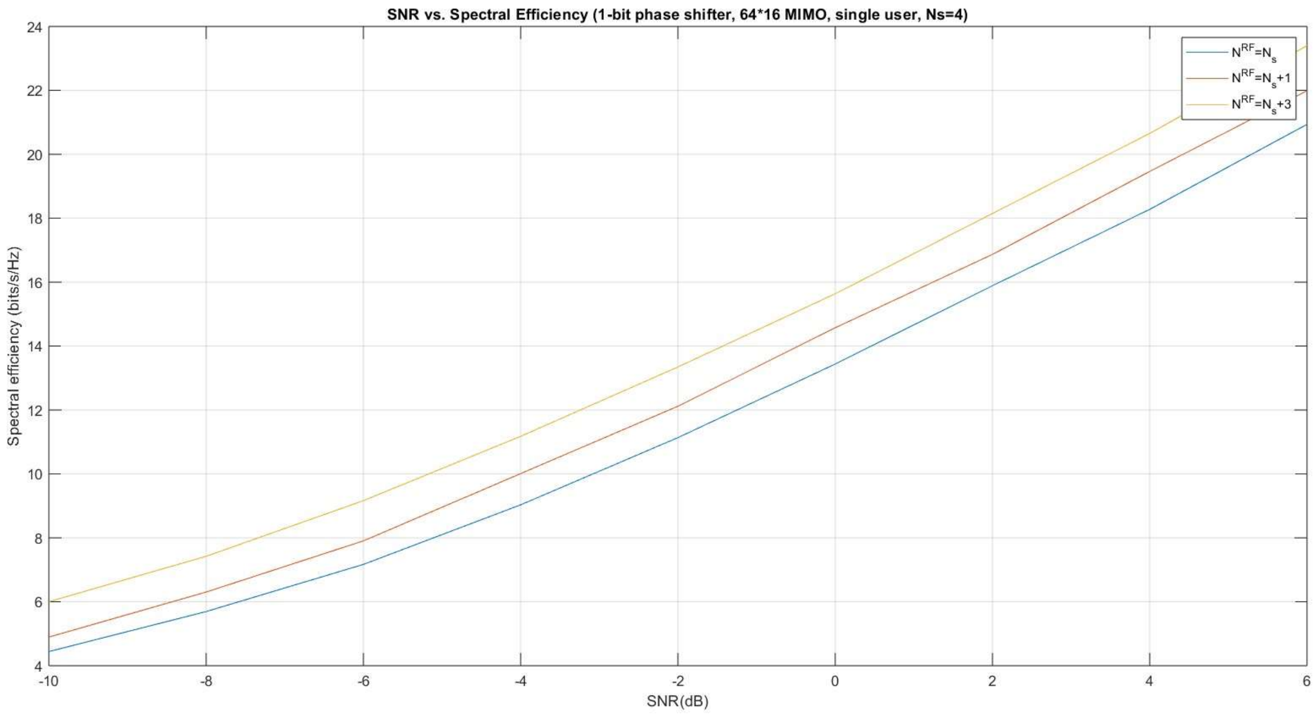

Figure 22: Spectral efficiency (1-bit phase shifter) vs. SNR, 1000 Monte Carlo trials

Which more clearly shows the paper’s discoveries.

Conclusion

This project implements the hybrid beamformer architecture proposed by the reference paper for wireless systems with large-scale antenna arrays. It focuses on single-user scenario and explores the performance of hybrid beamformers. It first implements a hybrid beamformer design in a 64*16 MIMO system assuming the number of RF chains is equal to the number of data streams, then modifies the algorithm to compare the difference between infinite resolution and 1-bit phase shifters. The process is repeated for the scenario that the number of RF chains is less than twice but larger than once of the number of data streams. The simulation results are in accordance with the paper's simulation figures with only minor difference on the range of spectral efficiency. Result 2 shows the performance of infinite resolution phase shifter is better than that of 1-bit phase shifter, which is opposite from paper Fig. 3, because the project does not quantize the infinite resolution phase shifter as in the paper. Result 3 shows that increasing the number of RF chains in hybrid beamformer architectures trades off its insufficiency in accuracy compared to fully digital ones and thus allows it to serve as a sub-optimal solution for massive MIMO systems.

Appendix: code

Note: to produce 3 results in the project specification separately, please refer to the instructions in the comment and change the parameter set accordingly.

```
clear;
%% Project 1
%% NOTE Change parameters according to the results in project spec!!!
N=10; % Number of antennas of the base station | 64 | 10
M=10; % Number of antennas of each user | 16 | 10
K=1; % Number of users
d=2; % Number data streams required by each user | 6 | 2 | 4
Ns=K*d; % Number of total data streams
L=15; % Number of paths (paper suggests 15)
SNR=0:5:30;   % -10:2:6 | 0:5:30
tolerance=1e-6;
num_mc=100; %   Number of Monte Carlo trials | 100 | 1000
phase_list=[1 exp(1j*pi)];
P=1;

% Complex channel
H=zeros(N,M);

% Uncomment for results 1, 2 ↓
N_RF_list=Ns;    % Sec. IV assumes N_RF=Nt_RF=Nr_RF preserves generality
R_list=zeros(1,length(SNR));
R_quant_list=zeros(1,length(SNR));
% Uncomment for results 1, 2 ↑

% Uncomment for result 3 ↓
%N_RF_list=[Ns, Ns+1, Ns+3];
%R_list=zeros(3,length(SNR));
%R_quant_list=zeros(3,length(SNR));
% Uncomment for result 3 ↑


for i_nrf=1:length(N_RF_list)

    N_RF=N_RF_list(i_nrf);
    Nt_RF=N_RF;   % Number of transmit RF chains
    Nr_RF=N_RF;   % Number of receive RF chains
    % NOTE Nt_RF=Nr_RF=N_RF, the total number of RF chains is 2*N_RF which
    % satisfies "if the number of RF chains is twice the total number of data
streams, the
    % hybrid beamforming structure can realize any fully digital beam former exactly"

    % Initialize Tx precoder and combiner
    V_D=zeros(Nt_RF,Ns);    % (14)
    V_RF=zeros(N,Nt_RF);    % (11)
    Vt=V_RF*V_D;


    for i_snr=1:length(SNR)
```

```
        %i_snr
        SNR_lin=10^(SNR(i_snr)/10);
        sigma=sqrt(P/SNR_lin);
        R=0;
        R_quant=0;

        for i_mc=1:num_mc
            %i_mc
            H=zeros(M,N);
            for l=1:L
                alpha=sqrt(1/2)*(randn(1,1)+1j*randn(1,1));
                phi_r=2*pi*randn(1,1);
                phi_t=2*pi*randn(1,1);
                % TODO rand
                a_r=transpose(exp(1j*pi*(0:M-1)*sin(phi_r))/sqrt(M));
                a_t=transpose(exp(1j*pi*(0:N-1)*sin(phi_t))/sqrt(N));
                H=H+alpha*a_r*a_t';
            end
            H=sqrt(N*M/L)*H;
            F1=H'*H;


            %% Part 1: IV.B RF Precoder Design for N_RF=Ns

            % Necessary parameters
            V_RF=ones(N,N_RF);
            V_RF_quant=ones(N,N_RF);

            % Translating algorithm 1 to design V_RF
            not_converge_inf=1;
            not_converge_1bit=1;

            % Convergence loop for infinite phase shifter
            while not_converge_inf
                for j=1:N_RF
                    V_RF_noj=V_RF;
                    V_RF_noj(:,j)=[];
                    C_j=eye(Nr_RF-1,Nr_RF-1)+(SNR_lin/N/N_RF)*V_RF_noj'*F1*V_RF_noj;
                    G_j=(SNR_lin/N/N_RF)*F1-
(SNR_lin/N/N_RF)^2*F1*V_RF_noj*pinv(C_j)*V_RF_noj'*F1;
                    for i=1:N
                        eta_ij=0;
                        for l=1:N
                            if l~=i
                                eta_ij=eta_ij+G_j(i,l)*V_RF(l,j);
                            end
                        end
                        % Infinite phase shifter
                        if eta_ij==0
                            V_RF(i,j)=1;
                        else
                            V_RF(i,j)=eta_ij/abs(eta_ij);
                        end
                    end
                end
```

```
            % Check convergence
            if abs(abs(V_RF).^2-ones(N,Nr_RF)) < tolerance*ones(N,Nr_RF)
                not_converge_inf=0;
            end
        end
        % TODO for each entry of precoder, quantize it to the nearest
        % in F = {1, w, ...}

        % Convergence loop for 1-bit phase shifter
        while not_converge_1bit
            for j=1:N_RF
                V_RF_noj=V_RF;
                V_RF_noj(:,j)=[];
                C_j=eye(Nr_RF-1,Nr_RF-1)+(SNR_lin/N/N_RF)*V_RF_noj'*F1*V_RF_noj;
                G_j=(SNR_lin/N/N_RF)*F1-
(SNR_lin/N/N_RF)^2*F1*V_RF_noj*pinv(C_j)*V_RF_noj'*F1;
                for i=1:N
                    eta_ij_quant=0;
                    for l=1:N
                        if l~=i
                            eta_ij_quant=eta_ij_quant+G_j(i,l)*V_RF(l,j);
                        end
                    end
                    % 1-bit resolution phase shifter
                    min_abs_sq=10000;
                    V_RF_quant_min=V_RF(i,j);
                    for i_phase=1:length(phase_list)
                        candidate=abs(phase_list(i_phase)-
eta_ij_quant/abs(eta_ij_quant))^2;
                        if candidate<min_abs_sq
                            min_abs_sq=candidate;
                            V_RF_quant_min=phase_list(i_phase);
                        end
                    end
                    V_RF_quant(i,j)=V_RF_quant_min;
                end
            end
            % Check convergence
            if abs(abs(V_RF_quant).^2-ones(N,Nr_RF)) < tolerance*ones(N,Nr_RF)
                not_converge_1bit=0;
            end
        end



        %% Part 2: IV.A. Digital Precoder Design for N_RF=Ns
        Q=V_RF'*V_RF;
        Q_quant=V_RF_quant'*V_RF_quant;
        Heff=H*V_RF;
        Heff_quant=H*V_RF_quant;
        [Left,Delta,U_e]=svd(Heff*pinv(sqrtm(Q)));  % TODO U_e' ?
        [Left_quant,Delta_quant,U_e_quant]=svd(Heff_quant*pinv(sqrtm(Q_quant)));
        % REF https://scicoding.com/water-filling-algorithm-in-depth-
explanation/#:~:text=Water%2Dfilling%20is%20a%20generic,in%20a%20technical%20sense%2C
%20orthogonal
```

```
            % REF https://zhuanlan.zhihu.com/p/502453127
            Delta(N_RF+1:M,:)=[];
            Delta_quant(N_RF+1:M,:)=[];
            N0=sigma^2;    % NOTE MIMO.pdf one line below formula 7.12

            lambda_list=diag(Delta);
            lambda_list(Ns+1:end,:)=[];
            lambda_quant_list=diag(Delta_quant);
            lambda_quant_list(Ns+1:end,:)=[];
            lambda_sq_list=lambda_list.^2;
            lambda_quant_sq_list=lambda_quant_list.^2;

            % NOTE water-filling solution: refer to MIMO.pdf formula 7.11 and above
            % NOTE assume Nmin (mu?) interval (0, /2, /2, /2, ...)
            % to obtain Nmin
            Nmin=1;
            mu=0;
            while Nmin<Ns && ~(mu>=sigma^2/lambda_sq_list(Nmin) &&
mu<=sigma^2/lambda_sq_list(Nmin+1))   % TODO >= <= or > < ?
                Nmin=Nmin+1;
                mu=(P+sum(sigma^2./lambda_sq_list(1:Nmin)))/Nmin;   % TODO ./!!!
            end
            Nmin_quant=1;
            mu_quant=0;
            while Nmin_quant<Ns && ~(mu_quant>=sigma^2/lambda_sq_list(Nmin_quant) &&
mu_quant<=sigma^2/lambda_sq_list(Nmin_quant+1))   % TODO >= <= or > < ?
                Nmin_quant=Nmin_quant+1;

mu_quant=(P+sum(sigma^2./lambda_quant_sq_list(1:Nmin_quant)))/Nmin_quant;   %
TODO ./!!!
            end
            P_star_list=max(mu-N0./lambda_sq_list,0);
            P_star_list(Nmin+1:end)=[];
            P_star_quant_list=max(mu_quant-N0./lambda_quant_sq_list,0);
            P_star_quant_list(Nmin_quant+1:end)=[];

            Gamma_e=diag([transpose(sqrt(P_star_list)) zeros(1,N_RF-Nmin)]);
            Gamma_e_quant=diag([transpose(sqrt(P_star_quant_list)) zeros(1,N_RF-
Nmin_quant)]);
            V_D=pinv(sqrtm(Q))*U_e*Gamma_e;
            V_D_quant=pinv(sqrtm(Q_quant))*U_e_quant*Gamma_e_quant;




            %% Part 3: IV.C Hybrid Combining Design for N_RF=NS

            % Necessary parameters
            W_RF=ones(M,N_RF);
            W_RF_quant=ones(M,N_RF);
            Vt=V_RF*V_D;
            Vt_quant=V_RF_quant*V_D_quant;
            F2=H*(Vt)*Vt'*H';
            F2_quant=H*(Vt_quant)*Vt_quant'*H';
```

```
            % Translating algorithm 1 to design W_RF for infinite phase shifter
            not_converge_inf=1;
            while not_converge_inf
                for j=1:N_RF
                    W_RF_noj=W_RF;
                    W_RF_noj(:,j)=[];
                    big=W_RF_noj'*F2*W_RF_noj;
                    C_j=eye(size(big))+(1/M/sigma^2)*big;
                    G_j=(1/M/sigma^2)*F2-
(1/M/sigma^2)^2*F2*W_RF_noj*pinv(C_j)*W_RF_noj'*F2;
                    for i=1:M
                        eta_ij=0;
                        for l=1:M
                            if l~=i
                                eta_ij=eta_ij+G_j(i,l)*W_RF(l,j);
                            end
                        end
                        if eta_ij==0
                            W_RF(i,j)=1;
                        else
                            W_RF(i,j)=eta_ij/abs(eta_ij);
                        end
                    end
                end
                % Check convergence
                if abs(abs(W_RF).^2-ones(M,N_RF)) < tolerance*ones(M,N_RF)
                    not_converge_inf=0;
                end
            end

            %  Translating algorithm 1 to design W_RF for 1-bit resolution phase
shifter
            not_converge_1bit=1;
            while not_converge_1bit
                for j=1:N_RF
                    W_RF_noj_quant=W_RF_quant;
                    W_RF_noj_quant(:,j)=[];
                    big_quant=W_RF_noj_quant'*F2_quant*W_RF_noj_quant;
                    C_j_quant=eye(size(big_quant))+(1/M/sigma^2)*big_quant;
                    G_j_quant=(1/M/sigma^2)*F2_quant-
(1/M/sigma^2)^2*F2_quant*W_RF_noj_quant*pinv(C_j_quant)*W_RF_noj_quant'*F2_quant;
                    for i=1:M
                        eta_ij_quant=0;
                        for l=1:M
                            if l~=i

eta_ij_quant=eta_ij_quant+G_j_quant(i,l)*W_RF_quant(l,j);
                            end
                        end
                        % 1-bit resolution phase shifter
                        min_abs_sq=10000;
                        W_RF_quant_min=W_RF(i,j);
                        for i_phase=1:length(phase_list)
```

```
                            candidate=abs(phase_list(i_phase)-
eta_ij_quant/abs(eta_ij_quant))^2;
                            if candidate<min_abs_sq
                                min_abs_sq=candidate;
                                W_RF_quant_min=phase_list(i_phase);
                            end
                        end
                        W_RF_quant(i,j)=W_RF_quant_min;
                    end
                end
                % Check convergence
                if abs(abs(W_RF_quant).^2-ones(M,N_RF)) < tolerance*ones(M,N_RF)
                    not_converge_1bit=0;
                end
            end

            % Design W_D
            J=W_RF'*H*(Vt)*Vt'*H'*W_RF+sigma^2*(W_RF')*W_RF;

J_quant=W_RF_quant'*H*(Vt_quant)*Vt_quant'*H'*W_RF_quant+sigma^2*(W_RF_quant')*W_RF_q
uant;
            W_D=pinv(J)*W_RF'*H*Vt;
            W_D_quant=pinv(J_quant)*W_RF_quant'*H*Vt_quant;
            Wt=W_RF*W_D;
            Wt_quant=W_RF_quant*W_D_quant;

            % Spectral efficiency
            R=R+log2(det(eye(M,M)+(Wt*pinv(Wt'*Wt)*Wt'*H*(Vt)*Vt'*H')/sigma^2));

R_quant=R_quant+log2(det(eye(M,M)+(Wt_quant*pinv(Wt_quant'*Wt_quant)*Wt_quant'*H*(Vt_
quant)*Vt_quant'*H')/sigma^2));







        end

        % Uncomment for results 1, 2 ↓
        R_list(i_snr)=R/num_mc;
        R_quant_list(i_snr)=R_quant/num_mc;
        % Uncomment for results 1, 2 ↑

        % Uncomment for result 3 ↓
        %R_list(i_nrf,i_snr)=R/num_mc;
        %R_quant_list(i_nrf,i_snr)=R_quant/num_mc;
        % Uncomment for result 3 ↑

    end
end


% Uncomment for result 1 ↓
```

```
%figure;
%plot(SNR,R_list);
%grid on;
%xlabel("SNR(dB)");
%ylabel("Spectral efficiency (bits/s/Hz)");
% NOTE spectral efficiency unit means: bit rate that this frequency can afford
%legend("N^{RF}=N_{s}","N^{RF}=N_{s}+1","N^{RF}=N_{s}+3");
%title("SNR vs. Spectral Efficiency (Infinite phase shifter, "+N+"*"+M+" MIMO, single
user, Ns="+Ns+")");
% Uncomment for result 2 ↑

% Uncomment for result 2 ↓
figure;
plot(SNR,R_list,SNR,R_quant_list);
grid on;
legend("b=\infty","b=1");
xlabel("SNR(dB)");
ylabel("Spectral efficiency (bits/s/Hz)");
% NOTE spectral efficiency unit means: bit rate that this frequency can afford
title("SNR vs. Spectral Efficiency ("+N+"*"+M+" MIMO, single user, Ns="+Ns+")");
% Uncomment for result 2 ↑

% Uncomment for result 3 ↓
%figure;
%plot(SNR,R_list);
%grid on;
%xlabel("SNR(dB)");
%ylabel("Spectral efficiency (bits/s/Hz)");
%legend("N^{RF}=N_{s}","N^{RF}=N_{s}+1","N^{RF}=N_{s}+3");
%title("SNR vs. Spectral Efficiency (Infinite phase shifter, "+N+"*"+M+" MIMO, single
user, Ns="+Ns+")");
%figure;
%plot(SNR,R_quant_list);
%grid on;
%xlabel("SNR(dB)");
%ylabel("Spectral efficiency (bits/s/Hz)");
%legend("N^{RF}=N_{s}","N^{RF}=N_{s}+1","N^{RF}=N_{s}+3");
%title("SNR vs. Spectral Efficiency (1-bit phase shifter, "+N+"*"+M+" MIMO, single
user, Ns="+Ns+")");
% Uncomment for result 3 ↑
```