Document de travail

ENSEA

Beyond Engineering

Digital Communication

Lab 1

Report of the Lab

3ème année - RTS

PETIT Alexandre
LAUMY Arthur

SESSION ONE

I – QAM Modulation and Gray Labelling

a) We need $k = \log_2 M$ to encode M symbols in a M-QAM with k, the number of bits by symbol.

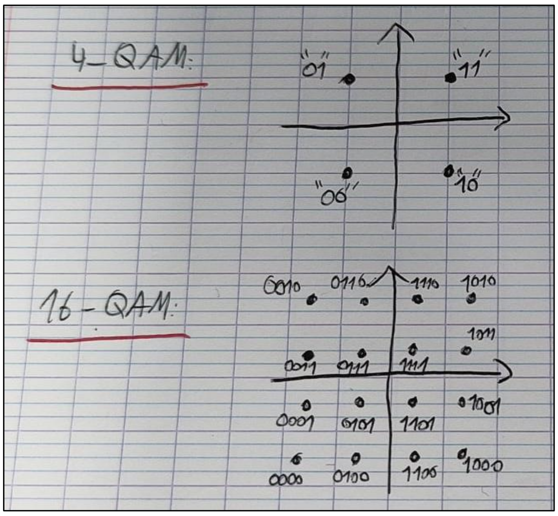

Figure 1 - Drawing of the M-QAM

b)

We created two functions: one to generate a constellation and one to visualize it.

To generate a constellation, first we generate the different levels of the QAM, decided with the M. Next, we use these levels to create our complex numbers for our different symbols and the Gray code labels. In the end, we convert the Gray code labels to binary ones which improve the readability of the constellation.

```
%%  Generation of the M-QAM constellation and associated Gray Code
M = 64;
g0 = 1;
[mPoints, mLabels] = generate_MQAM_constellation(M);
visualize_MQAM_constellation(M, mPoints, mLabels)

```

Figure 2 – Main file extract

```
function [mPoints, mLabels] = generate_MQAM_constellation(M)
    % Verify if M is a power of 4 (because M-QAM is a square constellation)
    if mod(log2(M), 2) ~= 0
        error('M must be a power of 4(for instance: 4, 16, 64, 256, ...)');
    end

    % Size of the matrix √M x √M
    sqrtM = sqrt(M);

    % Generation of the possible levels for the imaginary and real parts
    levels = -(sqrtM-1):2:(sqrtM-1);

    % Matrixes used to store complex points and Gray labels
    mPoints = zeros(sqrtM, sqrtM);  % For complex symbols
    mLabels = strings(sqrtM, sqrtM);  % For Gray labels (in string)

    % Fill matrixes with the points and the labels
    for i = 1:sqrtM
        for j = 1:sqrtM
            % Complex symbols: Combination of real and imaginary parts
            mPoints(i,j) = levels(i) + 1i*levels(j);

            % Labelling complex points
            gray_real = bitxor(i-1, floor((i-1)/2));
            %disp(gray_real);
            gray_imag = bitxor(j-1, floor((j-1)/2));
           % disp(gray_imag);

            % Form one binary word with the real and imaginary Gray bits
            gray_code_real = dec2bin(gray_real, log2(sqrtM));
            gray_code_imag = dec2bin(gray_imag, log2(sqrtM));

            % Gray label fill in a string
            % mLabels(i,j) = bin2dec(strcat(gray_code_real, gray_code_imag));
            mLabels(i,j) = strcat(gray_code_real, gray_code_imag);
        end
    end
end
```

Figure 3 - Generate functions

```
function visualize_MQAM_constellation(M, mPoints, mLabels)
    % M : Size of the constellation (Numbers of points in the M-QAM)
    % mPoints : Matrix containing complex symbols with a size of √M x √M
    % mLabels : Matrix containg Gray labels (in binary code) of size √M x √M

    % Extract real and imaginary parts of the points of the constellation
    x = real(mPoints(:)); % Real part
    y = imag(mPoints(:)); % Imaginary part
    z = mLabels(:);       % Gray labels (in binary, already in a string)

    % Create a new figure
    figure;

    % Print points of the constellation
    scatter(x, y, 50, 'b*'); % 'b*' : Blue with asterisks
    axis([-sqrt(M) sqrt(M) -sqrt(M) sqrt(M)]); % Adjust axis by M

    % Add the Gray labels side to side with the points
    for k = 1:M
        % z(k) already contains of the Gray labels in binary in a string form
        text(x(k) - 0.6, y(k) + 0.3, z(k), 'Color', [1 0 0]); % Red for the labels
    end

    % Add a title and labels for the axis
    title(['Gray Coding for ' num2str(M) '-QAM']);
    xlabel('I (Real Part)');
    ylabel('Q (Imaginary part)');

    % Activer la grille
    grid on;
end
```

Figure 4 - Visualize function

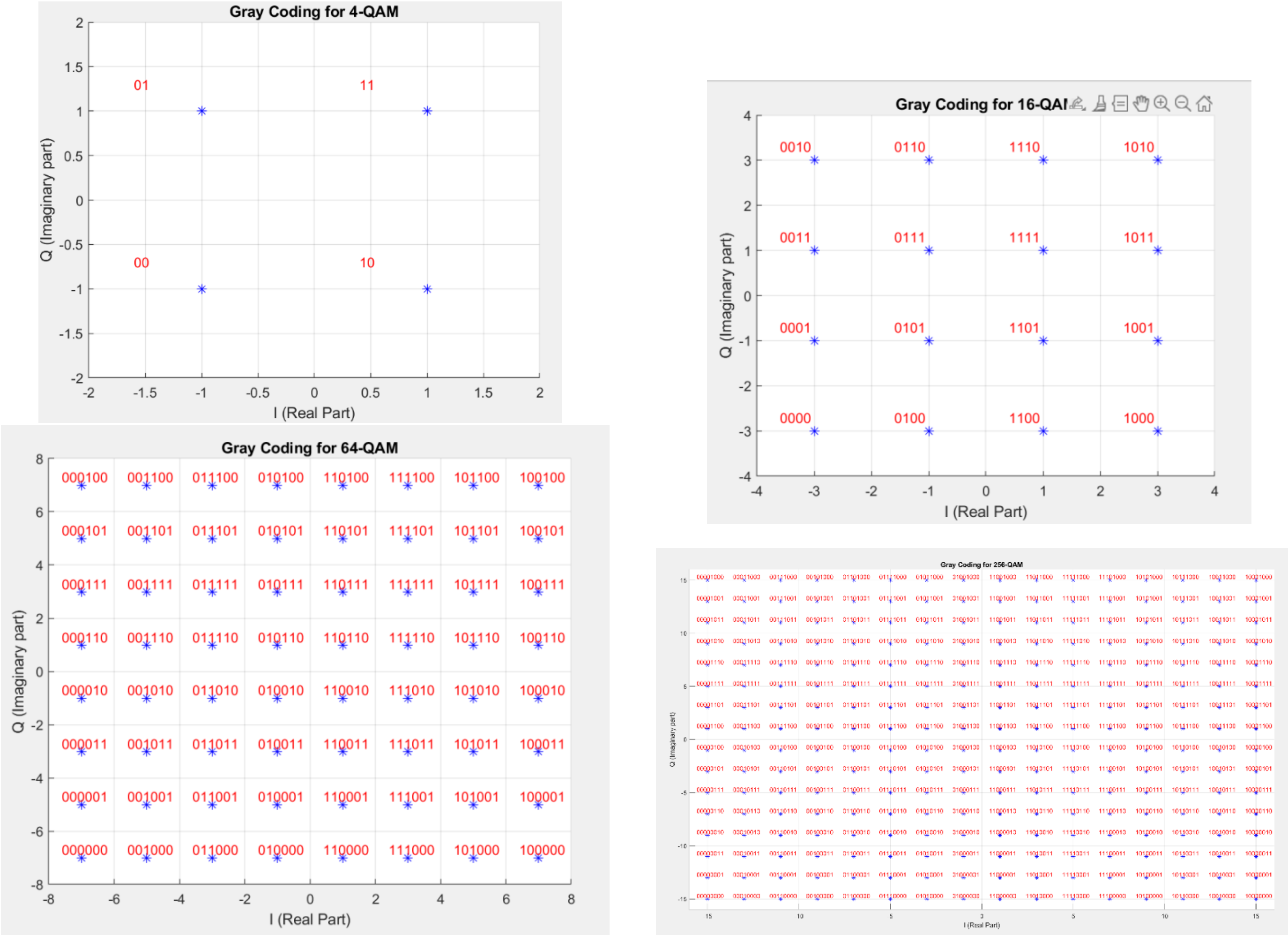

Figure 4 – Different M-QAM

II – Monte Carlo Performance Simulation

a)

First, we write a program to generate the chain mentioned above. This is divided into several parts:

- Modulation,
- Adding AWGN channel noise,
- Demodulation with the block decision

For the **modulation** part, we perform M-QAM modulation with a gain $g(0)$, converting bits into complex symbols based on a given constellation. To achieve this, we define the number of bits per symbol and loop through each group of bits, converting them into a binary string. The corresponding binary string is compared with the Gray code labels (mLabels) to find the coordinates of the symbol in the constellation (mPoints). The symbol in his decimal form is then extracted from the constellation and multiplied by the gain $g(0)$, which is stored in the vector of modulated symbols.

Next, in the main function, we add **Additive White Gaussian Noise** (AWGN) to the modulated symbols based on the SNR expressed in dB.

Regarding **demodulation**: for each received symbol, we calculate the distance between the symbol and all points in the constellation. We find the index of the closest point using the minimum distance. The closest point is stored in closest_points. We then retrieve the Gray label associated with this point, convert it into a bit array, and store it for later comparison between sent bits and received bits to have an error percentage.

```
%% Monte Carlo

% Simulation with random bits
num_bits = 20; %To choose but must be a power of 4
bits_per_symbol = log2(M);
num_symbols = num_bits / bits_per_symbol;
bits = randi([0 1], num_bits, 1);
disp(bits')

% Modulation
symbols = modulate_MQAM(bits, mPoints, mLabels, M, g0);
disp(symbols')

% AWGN Addition
SNR_dB = 0.001;
received_symbols = awgn(symbols, SNR_dB, 'measured');

% Demodulation and block of decision
[demodulated_bits, closest_points] = demodulate_MQAM_with_closest(received_symbols, mPoints, mLabels);
disp(demodulated_bits')

% Print thz M-QAM with the lines of connection and Gray labels
visualize_constellation_and_received(mPoints, mLabels, received_symbols, closest_points);

% Compare in/out bits
comparison = zeros(num_bits,1);
error=0;
for i = 1:num_bits
    difference = mod(bits(i,1) - demodulated_bits(i,1),2);
    comparison = comparison + difference;
    if difference == 1
        error = error + 1;
    end
end
% disp(comparison);
formatSpec = 'Pourcentage of error: %f \n';
fprintf(formatSpec,(error*100)/num_bits)
```

```
function symbols = modulate_MQAM(bits, mPoints, mLabels, M, g0)
    % Modulation M-QAM with gain g(0)
    bits_per_symbol = log2(M);
    num_symbols = length(bits) / bits_per_symbol;

    symbols = zeros(num_symbols, 1);

    for i = 1:num_symbols
        % Extract bits of each symbol from the bit stream
        bit_group = bits((i-1)*bits_per_symbol + 1:i*bits_per_symbol)';
        bit_string = num2str(bit_group(:)', '%d');

        % Find the decimal coordinates for each symbol
        [row, col] = find(mLabels == bit_string);

        % Modulate with the gain g(0)
        symbols(i) = g0 * mPoints(row, col);
    end
end
```

```matlab
function [demodulated_bits, closest_points] = demodulate_MQAM_with_closest(received_symbols, mPoints, mLabels)
    % M-QAM demodulation and find closest points
    num_symbols = length(received_symbols);
    bits_per_symbol = log2(numel(mPoints));

    % Initialize an array to store bits numerically
    demodulated_bits = zeros(num_symbols * bits_per_symbol, 1);  % Storing bits numerically
    closest_points = zeros(num_symbols, 1);  % Storing closest points

    % Index to fill demodulated_bits
    bit_idx = 1;

    for i = 1:num_symbols
        % Find the closest constellation point
        distances = abs(received_symbols(i) - mPoints(:));
        [~, min_index] = min(distances); % Ignore the first variable (~)

        % Get the Gray label of the closest point (as a string)
        gray_label = mLabels{min_index};  % mLabels must be a cell array containing strings

        % Store the closest constellation point
        closest_points(i) = mPoints(min_index);

        % Convert the Gray label (string) to numeric array (bits)
        bits = double(gray_label) - '0';  % Convert binary string to numeric array

        % Store the bits in demodulated_bits
        demodulated_bits(bit_idx:bit_idx + bits_per_symbol - 1) = bits;

        % Update the index for bits
        bit_idx = bit_idx + bits_per_symbol;
    end
end
```

```matlab
function visualize_constellation_and_received(mPoints, mLabels, received_symbols, closest_points)
    % Visualize the M-QAM constellation with Gray labels, received symbols, closest points,
    % and draw connection lines between the received symbols and the closest points.

    % Extract the real and imaginary parts of the constellation points
    x = real(mPoints(:)); % Real part
    y = imag(mPoints(:)); % Imaginary part
    z = mLabels(:);       % Gray labels (in binary, as strings)

    % Calculate the size of the constellation (M)
    M = numel(mPoints);

    % Create a new figure
    figure;

    % Plot the constellation points
    scatter(x, y, 50, 'b*'); % Constellation points in blue
    hold on;
    axis([-sqrt(M) sqrt(M) -sqrt(M) sqrt(M)]); % Adjust axes based on M

    % Add Gray labels next to the points
    for k = 1:M
        text(x(k) - 0.6, y(k) + 0.3, z(k), 'Color', [1 0 0]); % Labels in red
    end

    %  Display the received symbols
    scatter(real(received_symbols), imag(received_symbols), 50, 'rx');  % Received symbols in red

    % Print the received symbols
    scatter(real(closest_points), imag(closest_points), 50, 'go');  % Closest points in green

    % Draw lines connecting each received symbol to its closest point
    for i = 1:length(received_symbols)
        % Coordinates of the received symbol
        x_received = real(received_symbols(i));
        y_received = imag(received_symbols(i));

        % Coordinates of the closest point
        x_closest = real(closest_points(i));
        y_closest = imag(closest_points(i));

        % Draw a line between the received symbol and the closest point
        plot([x_received, x_closest], [y_received, y_closest], 'k--');  % Dashed black line
    end

    % Graph parameters
    title('M-QAM Constellation with Received Symbols and Connection Lines');
    xlabel('Real Part');
    ylabel('Imaginary Part');
    legend('Constellation', 'Received Symbols', 'Closest Points', 'Connection Lines');
    grid on;
    axis equal;
    hold off;
end
```

Figure 5 – Extract from the main files and the functions associated

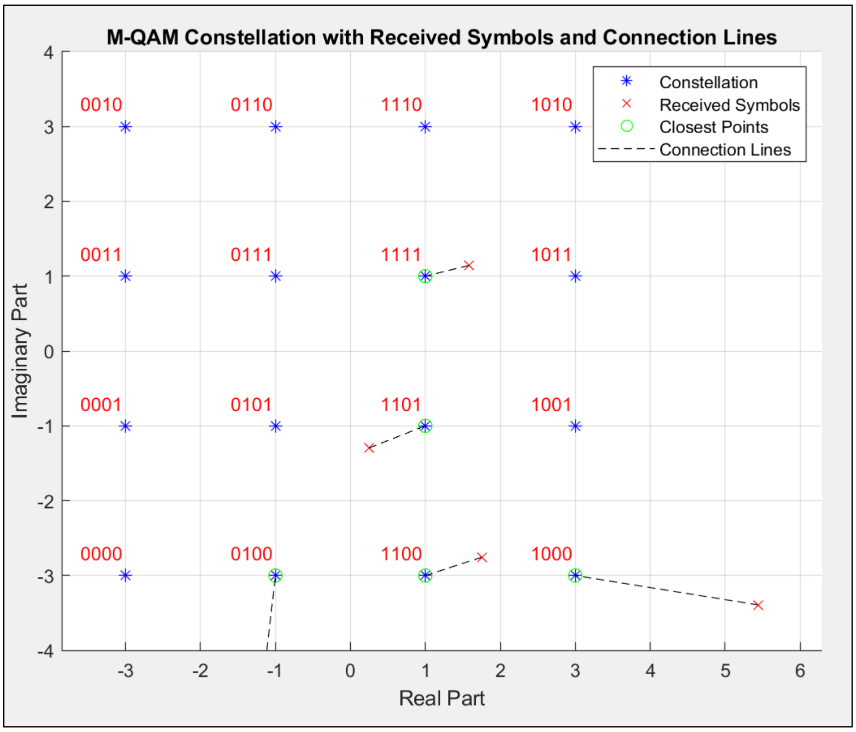

Figure 6 - Result for SNR = 0.01

b) Supposing with the error probability given, $P_e = \frac{1}{N} \rightarrow N = \frac{1}{P_e} \rightarrow 1 \leq N < 10^5$

The expression for the average energy of an M-QAM constellation as a function of the size M is: $E_{moy} = \frac{2(M-1)}{3}$.

We have: $\sigma^2 = \frac{N_0}{2} = \frac{1}{2} * \frac{E_s}{\frac{E_b}{N_0}}$ or $E_b = \frac{E_s}{\log_2(M)} = \frac{2(M-1)}{\log_2(M)}$.

$$\left(\frac{E_b}{N_0}\right)_{db} = 10 \log\left(\frac{E_b}{N_0}\right) \rightarrow \frac{E_B}{N_0} = 10^{\frac{\left(\frac{E_b}{N_0}\right)_{db}}{10}}$$

WIP