1 Problem-1, 2

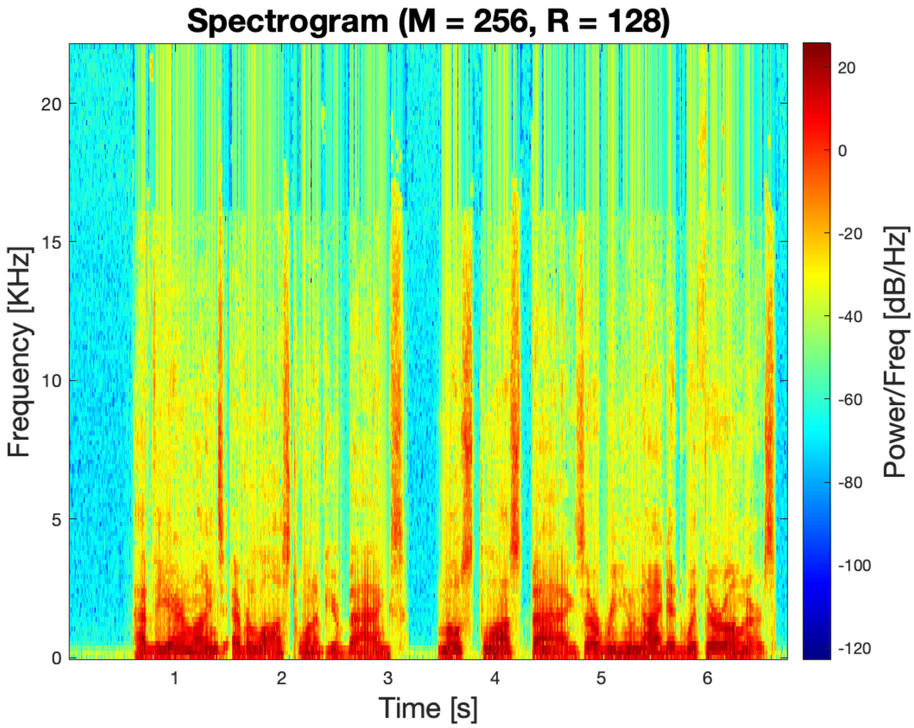

Figure 1: Spectrogram using built-in function (M=256, R=128)

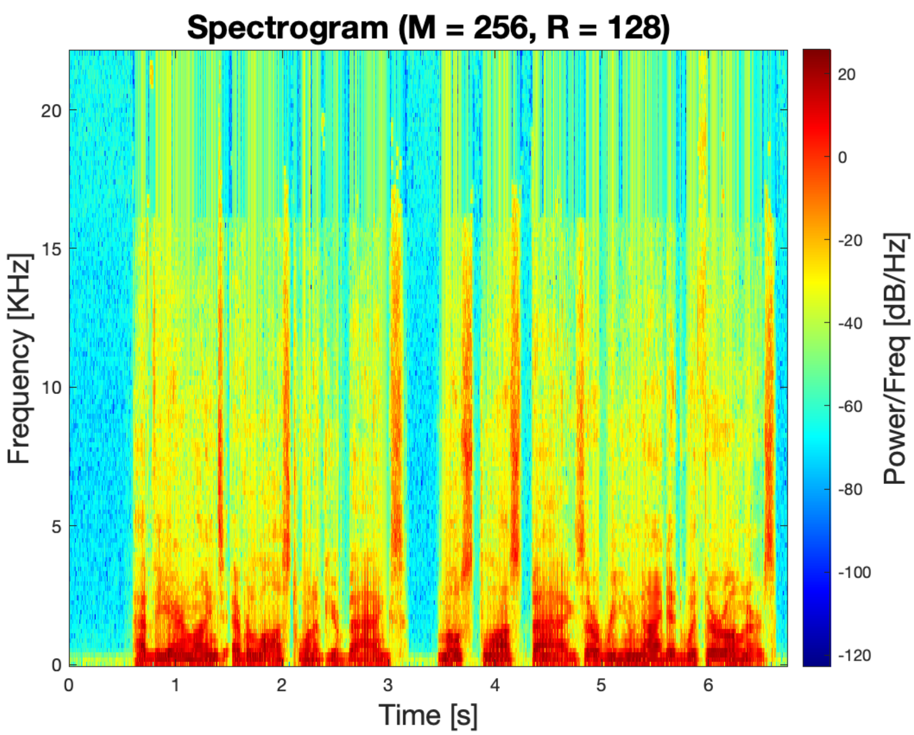

Figure 2: Spectrogram using Own function (M=256, R=128)

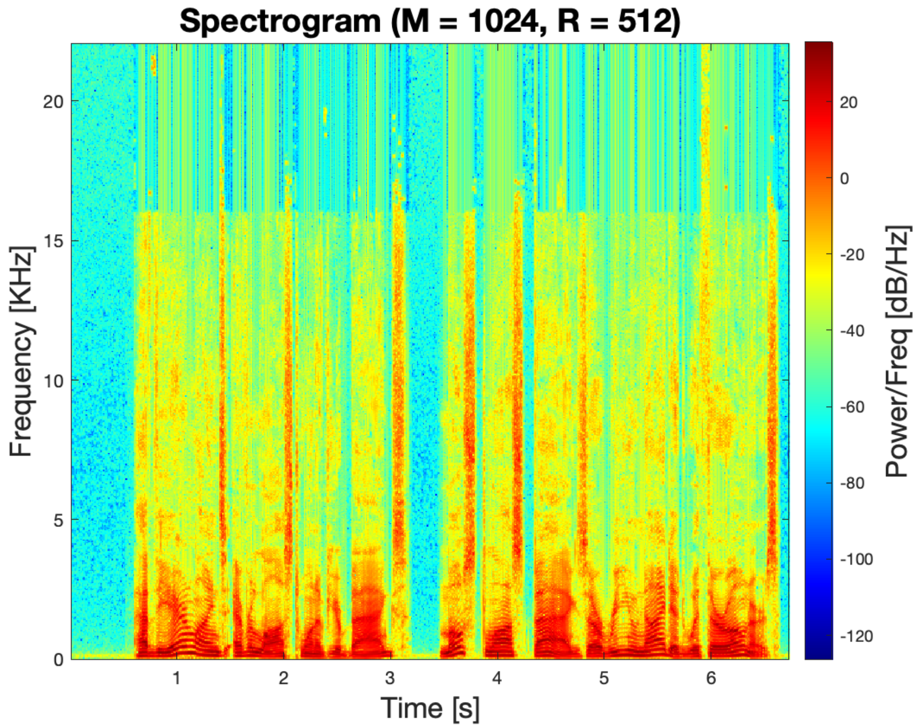

Figure 3: Spectrogram using built-in function (M=1024, R=512)

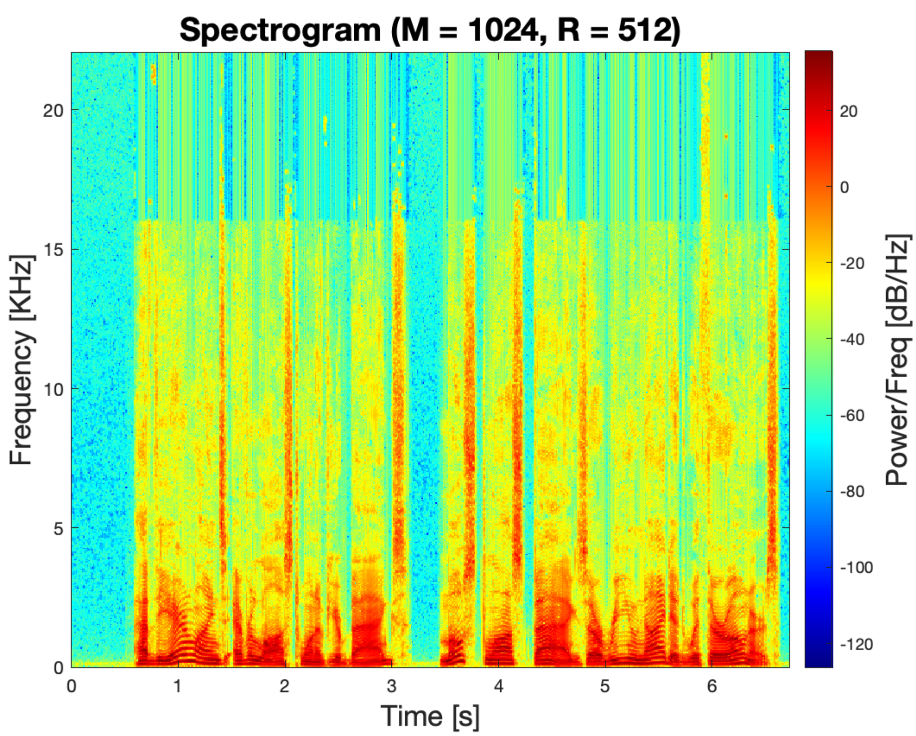

Figure 4: Spectrogram using Own function (M=1024, R=512)

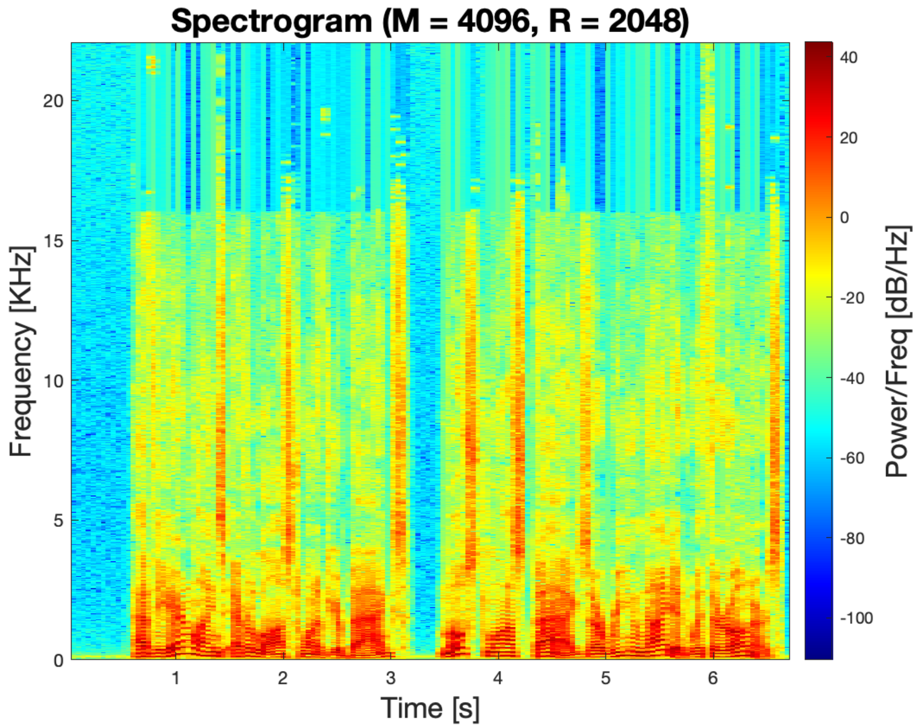

Figure 5: Spectrogram using built-in function (M=4096, R=2048)

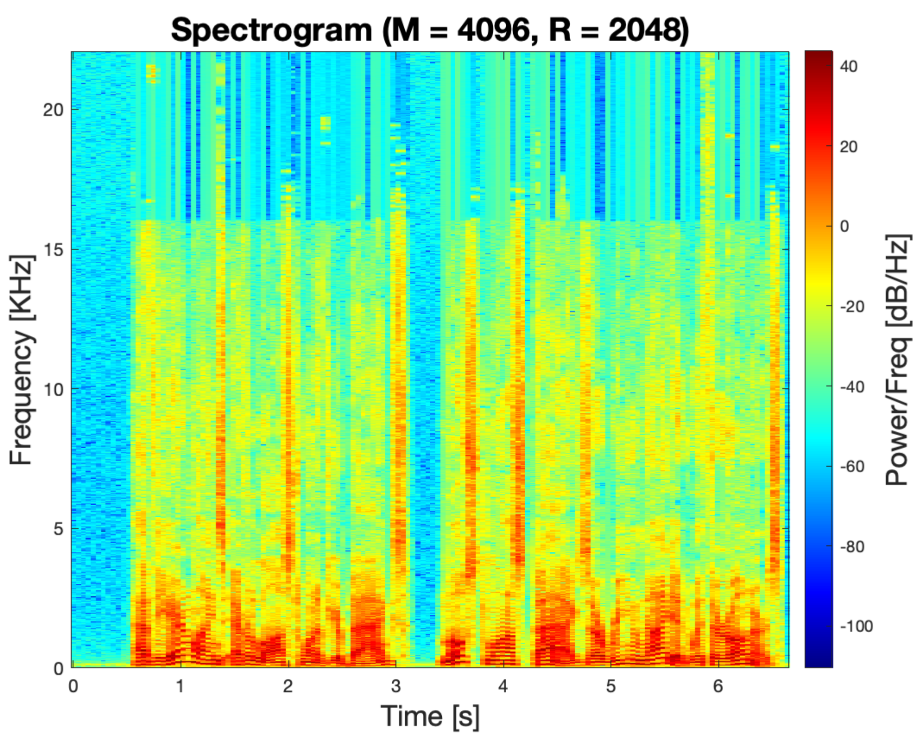

Figure 6: Spectrogram using Own function (M=4096, R=2048)

2 Problem-3

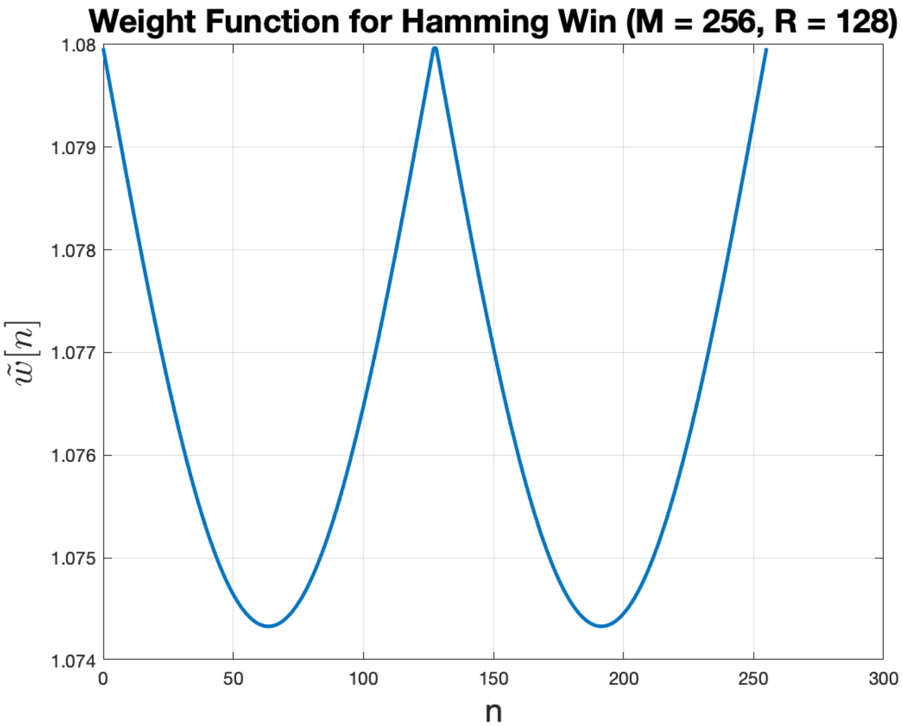

Figure 7: Weight Function for Hamming Window (M = 256, R = 128) shown only for 2 periods. Weight function is periodic with period = R = 128

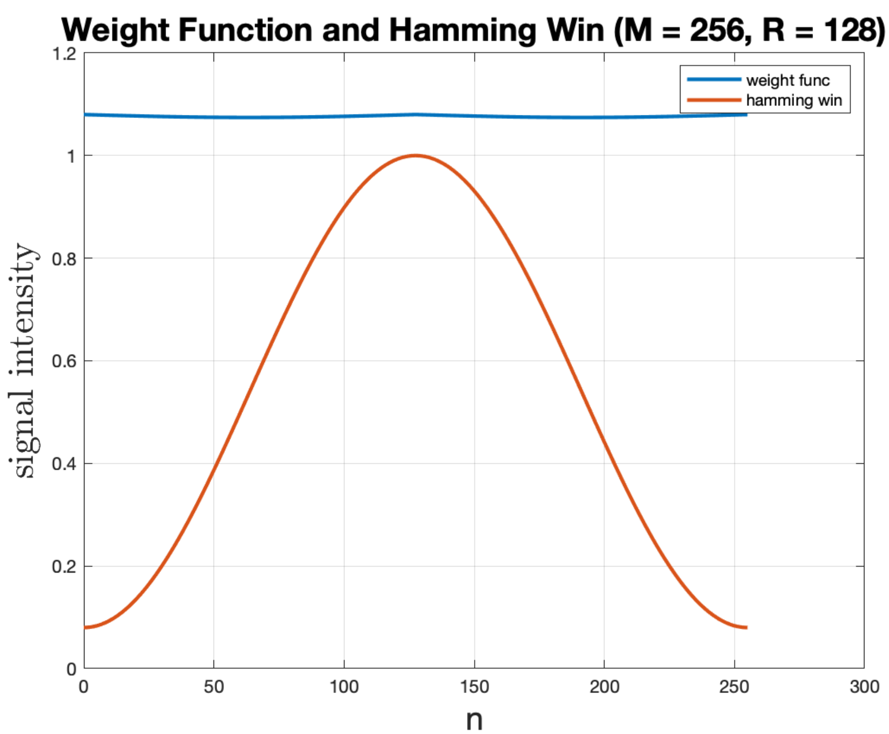

Figure 8: Weight Function vs Hamming Window (M = 256, R = 128). Weight function is nearly constant horizontal line and very nearly qualifies as a simple reconstruction

3 Problem-4

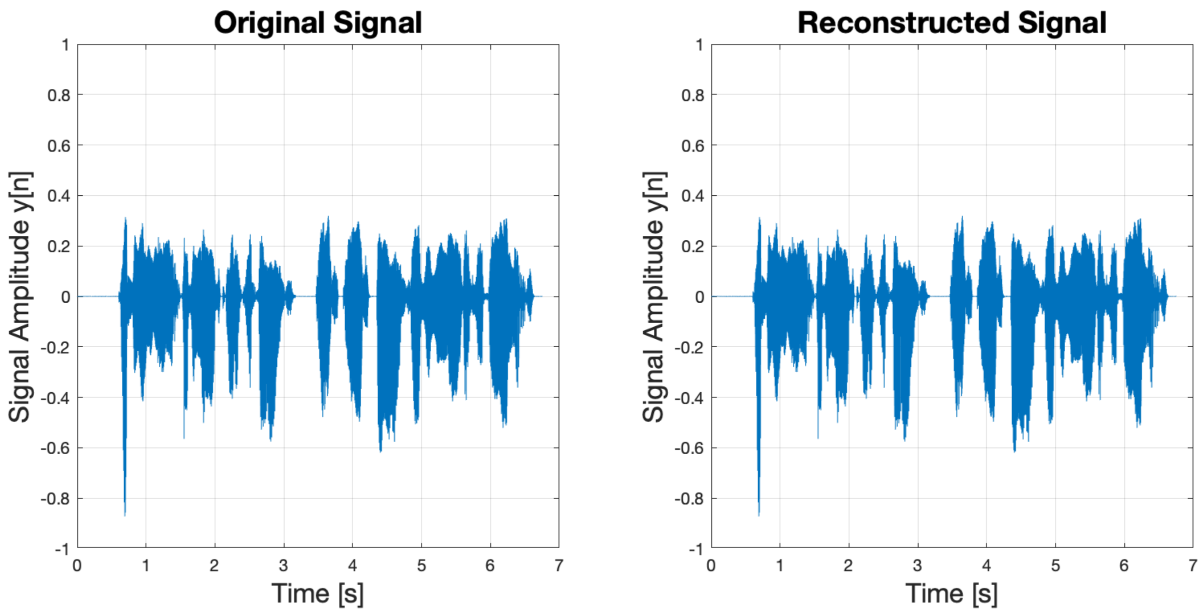

Figure 9: Figure showing original signal (left) and signal reconstructed from spectrogram (right). Exact recovery was achieved with MSE $= 1.5735 \times 10^{-11}$

.

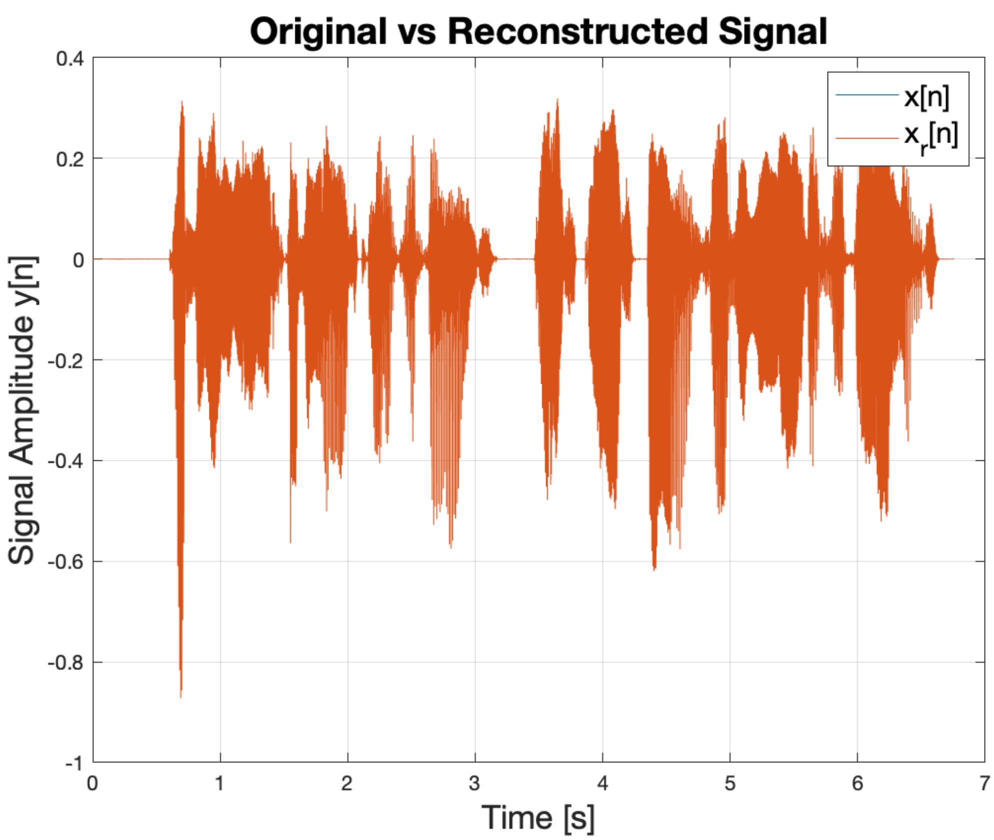

Figure 10: Original signal and Reconstructed signal plotted in the same plot. The original signal (in blue) is not visible due to overlapping

4 Problem-5

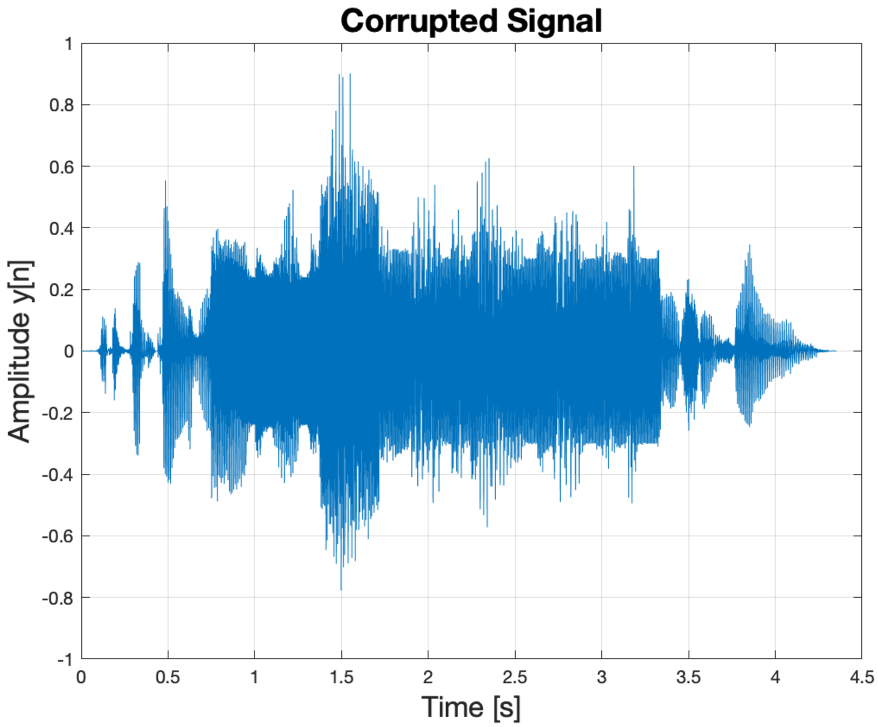

Figure 11: Time-domain Corrupted Signal (part5.wav)

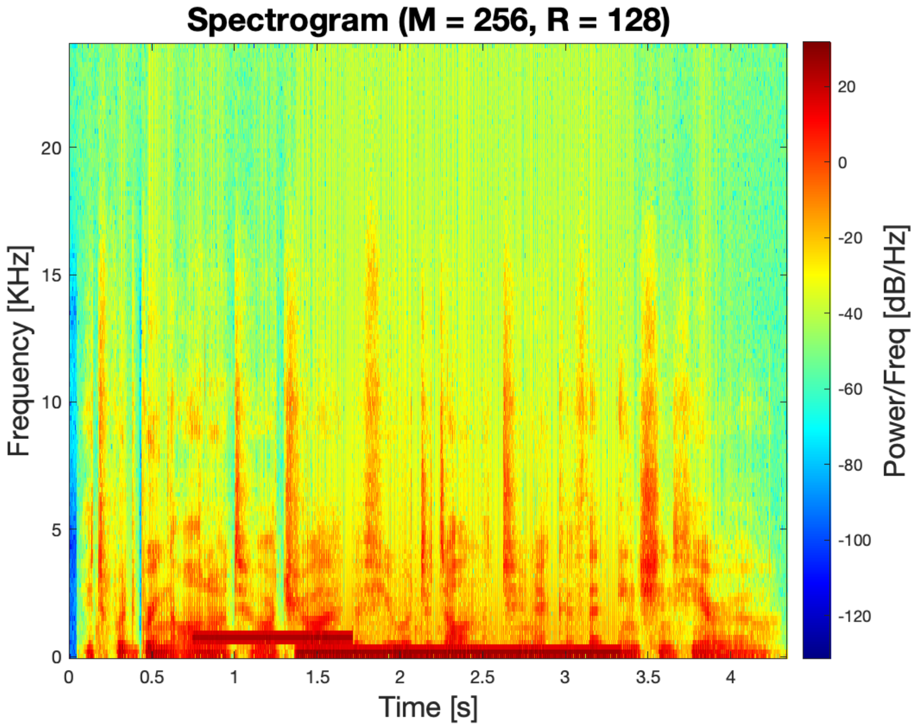

Figure 12: Spectrogram of the corrupted signal. It shows the frequency bands (0 - 1000 Hz) consisting of noise tones between 0.75 s to 1.75 s and another tone between 1.5 s to 3.4 s

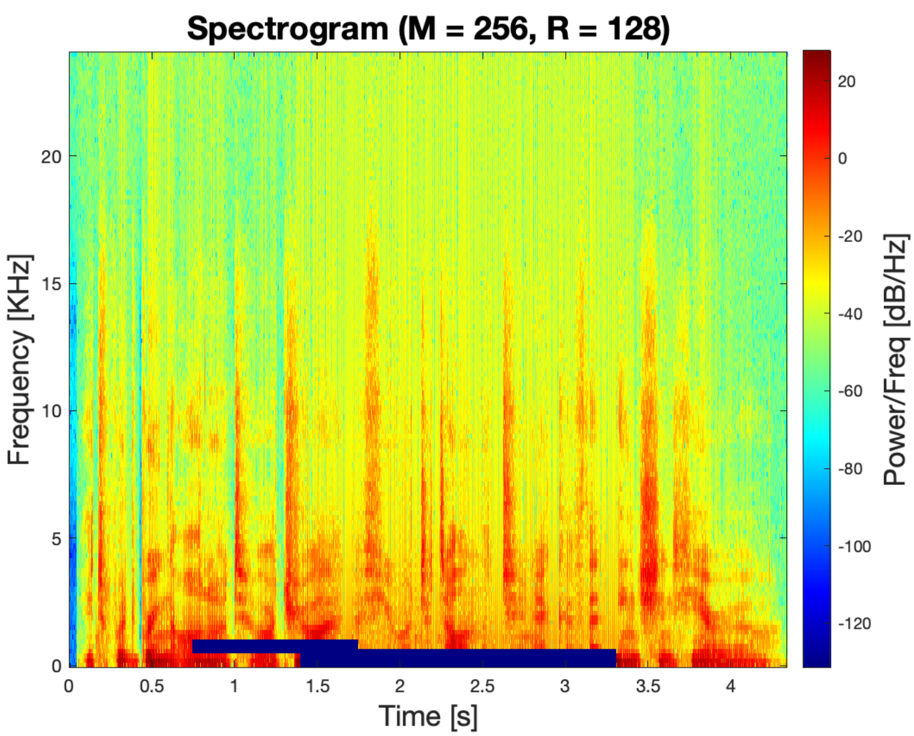

Figure 13: Spectrogram of the clean signal obtained using time-frequency processing. Used the spectrogram to identify which freq bins contains corrupting signal and zeroed those frequency bins and reconstructed the signal

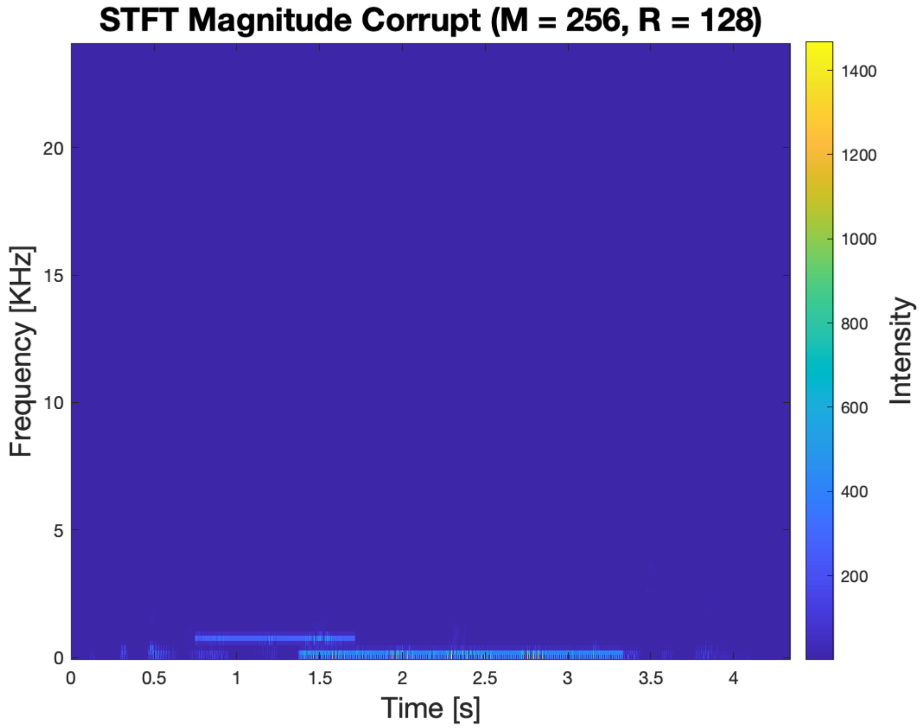

Figure 14: STFT Magnitude Squared of the corrupted signal

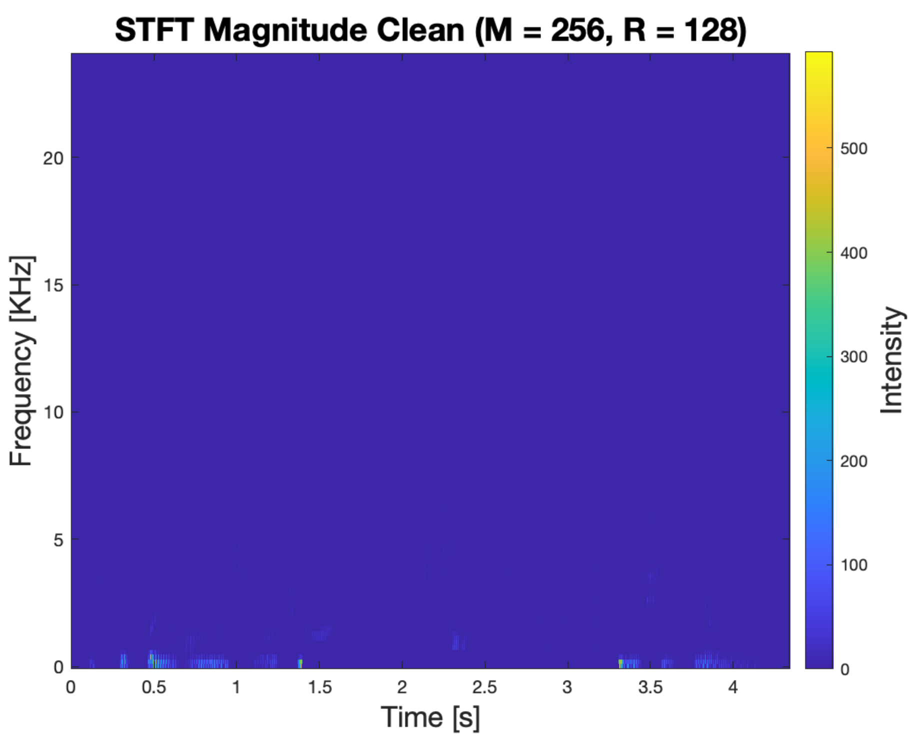

Figure 15: STFT Magnitude Squared of the cleaned signal

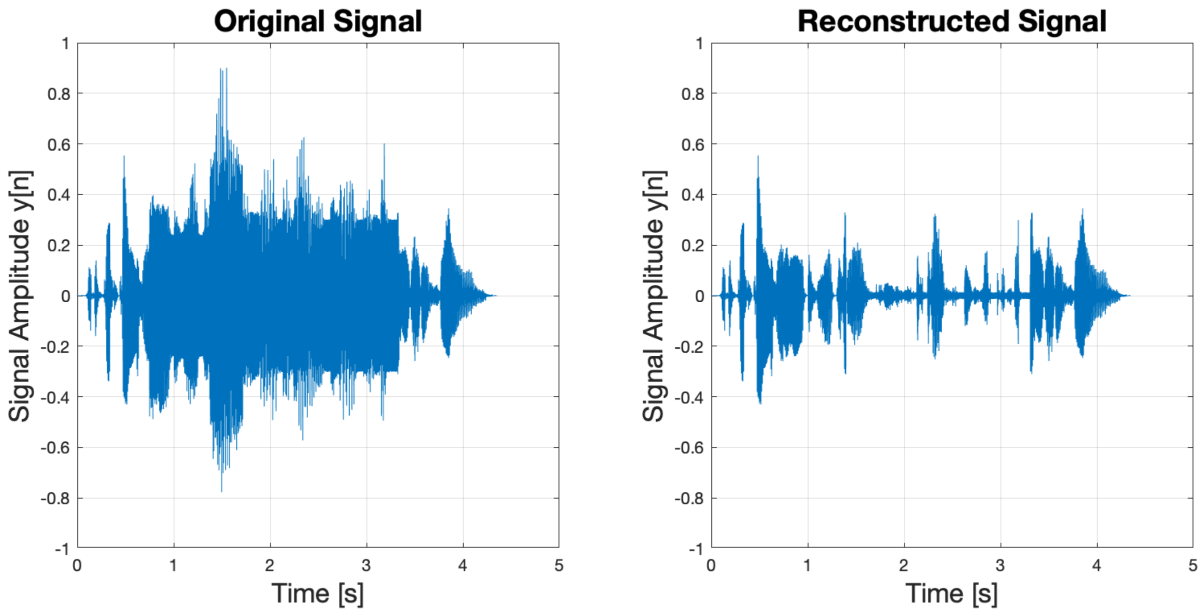

Figure 16: Figure showing original corrupt audio waveform (left) and clean audio waveform (right)

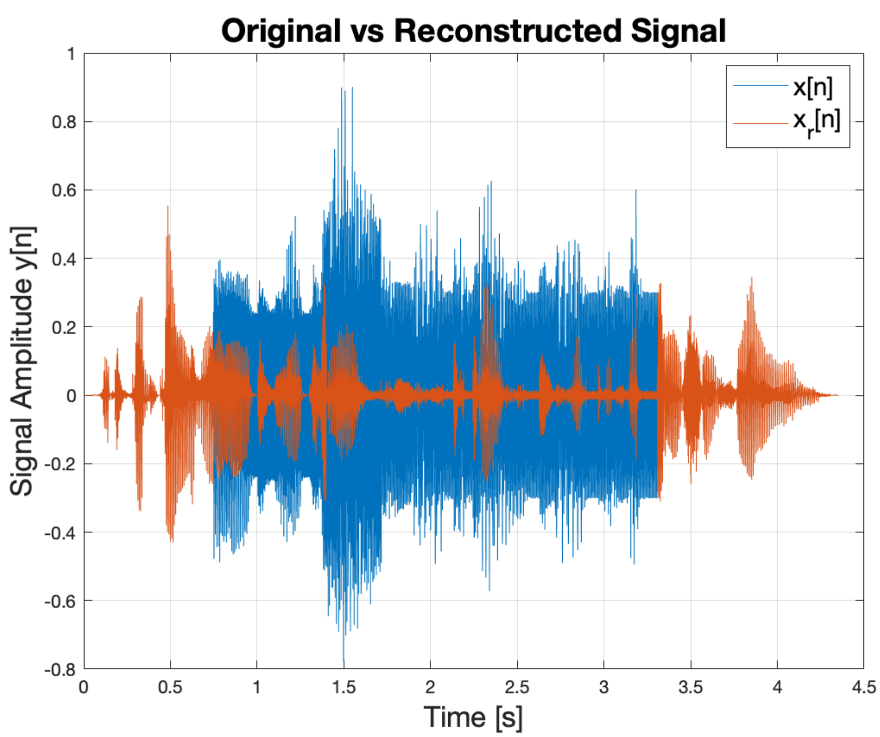

Figure 17: Figure showing original corrupt audio waveform and clean audio waveform overlapped

5 MATLAB Code

5.1 main.m

```
clc; clear;

run_q12 = false;
run_q3 = false;
run_q4 = false;
run_q5 = true;

if run_q12 == true || run_q3 == true || run_q4 == true
    [x_p1, Fs] = audioread("part1.wav");

    x_p1_len = size(x_p1, 1);
    x_p1_dur = x_p1_len / Fs;

    % play the signal
    % sound(x_p1, Fs);
end

% ---------------------------------------------------------------------------
if run_q12 == true
    tosave = true;
    for win_len = [256, 1024, 4096]
        win_shift = win_len / 2;

        % compute the spectrogram using in-built spectrogram function
        num_fft_samp = max(256, 2^nextpow2(win_len));
        [stft_inbuilt, freq_axis, time_axis] = spectrogram(x_p1, hamming(win_len), win_shift,
        spec_single_sided_inbuilt = 20 * log10(abs(stft_inbuilt));
        plot_spectrogram(spec_single_sided_inbuilt, freq_axis/1000, time_axis, win_len, win_sh

        % compute the spectrogram using implemented function
        [stft_own, freq_axis, time_axis] = compute_spectrogram(x_p1, Fs, win_len, win_shift);
        [fft_len, num_seg] = size(stft_own);

        spec_double_sided_own = 20 * log10(abs(stft_own));
        spec_single_sided_own = spec_double_sided_own(1:fft_len/2 + 1, :);

        plot_spectrogram(spec_single_sided_own, freq_axis, time_axis, win_len, win_shift, tosa

        mse = mean((spec_single_sided_own(:) - spec_single_sided_inbuilt(:)).^2);
        disp("MSE between in-built and own spectrogram: "+mse);
    end
end
% ---------------------------------------------------------------------------

% ---------------------------------------------------------------------------
```

```
if run_q3 == true
    % plot the reconstruction weights for hamming window
    tosave = true;
    win_len = 256;
    win_shift = 128;
    weight_func = get_weight_func(win_len, win_shift, "hamming");
    plot_weight_func(weight_func, win_len, win_shift, tosave, "3", "weight_func_hamming_M"+win
end
% --------------------------------------------------------------------------

% --------------------------------------------------------------------------
if run_q4 == true
    % reconstruct signal from spectrogram and check whether exact recovery is achieved
    tosave = true;
    win_len = 256;
    win_shift = 128;

    num_fft_samp = max(256, 2^nextpow2(win_len));
%     [stft_mat, freq_axis, time_axis] = spectrogram(x_p1, hamming(win_len), win_shift, num_ff
    [stft_mat, freq_axis, time_axis] = compute_spectrogram(x_p1, Fs, win_len, win_shift);
%     spec_single_sided = 20 * log10(abs(stft_mat));

    x_p1_reconst = compute_reconst_sig(stft_mat, x_p1_len, win_len, win_shift);

    mse_reconst = mean((x_p1(:) - x_p1_reconst(:)).^2);
    disp("Mean Square Error: "+mse_reconst);

    plot_reconst_sig(x_p1, x_p1_reconst, Fs, win_len, win_shift, tosave, "4");
    sound(x_p1_reconst, Fs);
end
% --------------------------------------------------------------------------

% --------------------------------------------------------------------------
if run_q5 == true
    [x_p5, Fs] = audioread("part5.wav");

    x_p5_len = size(x_p5, 1);
    x_p5_dur = x_p5_len / Fs;

    % sound(x_p5, Fs);

    % plot corrupted signal in time-domain
    fig = figure;
    time_axis = (1:size(x_p5, 1)) / Fs;
    plot(time_axis, x_p5);
    ylim([-1.0 1.0]);
    grid on;
    title("Corrupted Signal", "FontSize", 18);
    xlabel("Time [s]", "FontSize", 16);
    ylabel("Amplitude y[n]", "FontSize", 16);
```

```
    saveas(fig, "../plots/prob5/corrupt_sig.png");
    close;

    % compute spectrogram of corrupted signal
    win_len = 256;
    win_shift = 128;

    [stft_corrupt, freq_axis, time_axis] = compute_spectrogram(x_p5, Fs, win_len, win_shift);
    [fft_len, num_seg] = size(stft_corrupt);   % (256 x 1630)

    stft_mag_sq_corrupt = abs(stft_corrupt).^2;
    spec_double_sided_corrupt = 20 * log10(abs(stft_corrupt));
    spec_single_sided_corrupt = spec_double_sided_corrupt(1:fft_len/2 + 1, :);

    % plot spectrogram of corrupted signal
    plot_spectrogram(spec_single_sided_corrupt, freq_axis, time_axis, win_len, win_shift, true

    % plot STFT magnitude squared of corrupt signal
    fig = figure;
    imagesc(time_axis, freq_axis, stft_mag_sq_corrupt(1:fft_len/2 + 1, :));
    set(gca, 'ydir', 'normal');   % flip the Y Axis so lower frequencies are at the bottom
    title("STFT Magnitude Corrupt (M = "+win_len+", R = "+win_shift+")", 'FontSize', 18);
    xlabel("Time [s]", 'FontSize', 16);
    ylabel("Frequency [KHz]", 'FontSize', 16);
    % colormap(jet(256));
    c = colorbar;
    c.Label.String = 'Intensity';
    c.Label.FontSize = 16;
    saveas(fig, "../plots/prob"+5+"/"+"stft_magsq_corrupt_M"+win_len+"_R"+win_shift+".png");
    close;

    % create a copy of corrupt STFT for processing
    stft_clean = stft_corrupt;

    % remove noise from corrupted signal
    clean_val = 0;
    stft_clean(4:6, 280:656) = clean_val;
    stft_clean(252:254, 280:656) = clean_val;
    stft_clean(1:4, 525:1240) = clean_val;
    stft_clean(254:256, 525:1240) = clean_val;

%     stft_clean(1:4, 175:240) = clean_val;
%     stft_clean(253:256, 175:240) = clean_val;
%     stft_clean(1:2, 275:350) = clean_val;
%     stft_clean(255:256, 275:350) = clean_val;
%     stft_clean(1:3, 110:130) = clean_val;
%     stft_clean(254:256, 110:130) = clean_val;
%     stft_clean(1:2, 1240:1290) = clean_val;
%     stft_clean(255:256, 1240:1290) = clean_val;
%     stft_clean(1:3, 1400:1470) = clean_val;
```

```
%     stft_clean(254:256, 1400:1470) = clean_val;

%     noise_loc = find(stft_clean > 10);
%     stft_clean(abs(stft_clean).^2 > 50) = 0;

    stft_mag_sq_clean = abs(stft_clean).^2;
    spec_double_sided_clean = 20 * log10(abs(stft_clean));
    spec_single_sided_clean = spec_double_sided_clean(1:fft_len/2 + 1, :);

    % plot spectrogram of clean signal
    plot_spectrogram(spec_single_sided_clean, freq_axis, time_axis, win_len, win_shift, true,

    % plot STFT magnitude squared of clean signal
    fig = figure;
    imagesc(time_axis, freq_axis, stft_mag_sq_clean(1:fft_len/2 + 1, :));
    set(gca, 'ydir', 'normal');   % flip the Y Axis so lower frequencies are at the bottom
    title("STFT Magnitude Clean (M = "+win_len+", R = "+win_shift+")", 'FontSize', 18);
    xlabel("Time [s]", 'FontSize', 16);
    ylabel("Frequency [KHz]", 'FontSize', 16);
    % colormap(jet(256));
    c = colorbar;
    c.Label.String = 'Intensity';
    c.Label.FontSize = 16;
    saveas(fig, "../plots/prob"+5+"/"+"stft_magsq_clean_M"+win_len+"_R"+win_shift+".png");
    close;

    fig = figure;
    freq_axis_double = (Fs * (0:fft_len)) / (fft_len * 1000);
    imagesc(time_axis, freq_axis_double, stft_mag_sq_clean);
    set(gca, 'ydir', 'normal');   % flip the Y Axis so lower frequencies are at the bottom
    title("STFT Magnitude Clean (M = "+win_len+", R = "+win_shift+")", 'FontSize', 18);
    xlabel("Time [s]", 'FontSize', 16);
    ylabel("Frequency [KHz]", 'FontSize', 16);
    % colormap(jet(256));
    c = colorbar;
    c.Label.String = 'Intensity';
    c.Label.FontSize = 16;
    saveas(fig, "../plots/prob"+5+"/"+"stft_double_sided_magsq_clean_M"+win_len+"_R"+win_shift
    close;

    % reconstruct clean version of corrupted signal
    x_p5_reconst = compute_reconst_sig(stft_clean, x_p5_len, win_len, win_shift);
%     x_p5_reconst = real(x_p5_reconst);
    sound(x_p5_reconst, Fs);
    plot_reconst_sig(x_p5, x_p5_reconst, Fs, win_len, win_shift, true, 5);
end
% ----------------------------------------------------------------------------
```

5.2 compute_spectrogram.m

```
function [stft_mat, freq_axis, time_axis] = compute_spectrogram(x, Fs, win_len, win_shift)
    x_len = size(x,1);
    num_seg = ceil((x_len - win_len)/win_shift);
    fft_len = win_len;

    stft_mat = zeros(fft_len, num_seg);
%     spec_double_sided = zeros(fft_len, num_seg);
%     spec_single_sided = zeros(fft_len/2 + 1, num_seg);

    hamm_win_samp = hamming(win_len);

    % iterate over all segments and compute dft for each segment
    for r = 0:num_seg-1
        win_start = r * win_shift + 1;
        win_end = r * win_shift + win_len;

        if win_end > x_len
            x_seg = [x(win_start:x_len); zeros(win_end-x_len, 1)] .* hamm_win_samp;
        else
            x_seg = x(win_start : win_end) .* hamm_win_samp;
        end

        assert(size(x_seg, 1) == win_len);
        x_seg_dft = fft(x_seg);
%         x_seg_dft_len = size(x_seg_dft, 1);

        % compute the single-sided spectrum
        stft_mat(:, r+1) = x_seg_dft;
%         spec_double_sided(:, r+1) = 20 * log10(abs(x_seg_dft));
%         spec_single_sided(:, r+1) = spec_double_sided(1:x_seg_dft_len/2 + 1, r+1);
    end

    time_axis = linspace(0, num_seg-1) * win_shift / Fs;
    freq_axis = (Fs * (0:fft_len/2)) / (fft_len * 1000);
end
```

5.3 plot_spectrogram.m

```
function [] = plot_spectrogram(spec_single_sided, freq_axis, time_axis, win_len, win_shift, to

    fig = figure;
    imagesc(time_axis, freq_axis, spec_single_sided);
    set(gca, 'ydir', 'normal');   % flip the Y Axis so lower frequencies are at the bottom

    title("Spectrogram (M = "+win_len+", R = "+win_shift+")", 'FontSize', 18);
```

```
    xlabel("Time [s]", 'FontSize', 16);
    ylabel("Frequency [KHz]", 'FontSize', 16);

    colormap(jet(256));
    c = colorbar;
    c.Label.String = 'Power/Freq [dB/Hz]';
    c.Label.FontSize = 16;

%     spec_max = max(spec_single_sided(:));
%     spec_min = min(spec_single_sided(:));
%     disp(spec_min);
%     disp(spec_max);
%     set(c, 'ylim', [spec_min spec_max]);

    if tosave == true
        saveas(fig, "../plots/prob"+prob+"/"+save_file_name);
        close;
    end
end
```

5.4 get_weight_func.m

```
function [weight_func] = get_weight_func(win_len, win_shift, win_type)
    if win_type == "hamming"
        win_samp = hamming(win_len);
    end

    win_shift_right_R = [zeros(win_shift, 1); win_samp(1:win_len-win_shift, 1)];
    win_shift_left_R = [win_samp(win_shift+1:win_len, 1); zeros(win_shift, 1)];

    if win_shift == win_len / 2
        weight_func = win_samp + win_shift_right_R + win_shift_left_R;
    end
end
```

5.5 plot_weight_func.m

```
function [] = plot_weight_func(weight_func, win_len, win_shift, tosave, prob, save_file_name)
    fig1 = figure;
    plot(0:size(weight_func, 1)-1, weight_func, 'LineWidth', 2.0);
    grid on;
    title("Weight Function for Hamming Win (M = "+win_len+", R = "+win_shift+")", 'FontSize',
    xlabel("n", 'FontSize', 18);
    ylabel('$\tilde{w}[n]$', 'interpreter', 'latex', 'FontSize', 20, 'fontweight', 'bold');

    if tosave == true
```

```
        saveas(fig1, "../plots/prob"+prob+"/"+save_file_name);
        close;
    end

    fig2 = figure;
    plot(0:size(weight_func, 1)-1, weight_func, 'LineWidth', 2.0); hold on;
    plot(0:size(weight_func, 1)-1, hamming(win_len), 'LineWidth', 2.0); hold off;
    grid on;
    title("Weight Function and Hamming Win (M = "+win_len+", R = "+win_shift+")", 'FontSize',
    xlabel("n", 'FontSize', 18);
    ylabel('signal intensity', 'interpreter', 'latex', 'FontSize', 20, 'fontweight', 'bold');
    legend("weight func", "hamming win")

    if tosave == true
        saveas(fig2, "../plots/prob"+prob+"/"+"weight_func_vs_hamming_M256_R128.png");
        close;
    end
end
```

5.6 compute_reconst_sig.m

```
function [x_reconst] = compute_reconst_sig(stft_mat, x_len, win_len, win_shift)

    [fft_len, num_seg] = size(stft_mat);
    x_hat = zeros(x_len, 1);
%     stft_mag = 10.^(spec / 20.0);

    % iterate over each DFT segment to reconstruct the original sequence
    for r = 0:num_seg-1
        x_seg_dft = stft_mat(:, r+1);

%         x_seg_dft_mag_double = zeros(2 * (fft_len-1), 1);
%         x_seg_dft_mag_double(1:fft_len-1) = x_seg_dft_mag(2:end);
%         x_seg_dft_mag_double(fft_len:end) = flip(conj(x_seg_dft_mag(2:end)));
%         x_seg_dft_mag = spec(:, r+1);
%         x_seg = ifft(x_seg_dft_mag_double);

        x_seg = ifft(x_seg_dft);

        win_start = r * win_shift + 1;
        win_end = r * win_shift + win_len;

        if win_end > x_len
            x_hat(win_start:x_len) = x_hat(win_start:x_len) + x_seg(1:x_len - win_start + 1);
        else
            x_hat(win_start:win_end) = x_hat(win_start:win_end) + x_seg;
        end
    end
```

```
    % construct a repeated version of weight function (since it is periodic)
    weight_func = get_weight_func(win_len, win_shift, "hamming");
    weight_func_repeat = zeros(x_len, 1);
    for i = 1:win_len:x_len
        if i + win_len - 1 > x_len
            weight_func_repeat(i:x_len) = weight_func(1:x_len - i + 1);
        else
            weight_func_repeat(i:i + win_len - 1) = weight_func;
        end
    end

    assert(size(x_hat, 1) == size(weight_func_repeat, 1));
    x_reconst = x_hat ./ weight_func_repeat;
end
```

5.7 plot_reconst_sig.m

```
function [] = plot_reconst_sig(x, x_reconst, Fs, win_len, win_shift, tosave, prob)
    % plot the original and reconstructed signal
    fig1 = figure;
    time_axis = (1:size(x,1)) / Fs;
    plot(time_axis, x); hold on;
    plot(time_axis, x_reconst); hold off;
    grid on;
    title("Original vs Reconstructed Signal", "FontSize", 18);
    xlabel("Time [s]", "FontSize", 16);
    ylabel("Signal Amplitude y[n]", "FontSize", 16);
    legend("x[n]", "x_r[n]", "FontSize", 14);

    if tosave == true
        saveas(fig1, "../plots/prob"+prob+"/origvsrecons_M"+win_len+"_R"+win_shift+".png");
    end

    fig2 = figure;
    time_axis = (1:size(x,1)) / Fs;

    subplot(1,2,1);
    plot(time_axis, x);
    ylim([-1.0 1.0]);
    grid on;
    title("Original Signal", "FontSize", 18);
    xlabel("Time [s]", "FontSize", 16);
    ylabel("Signal Amplitude y[n]", "FontSize", 16);

    subplot(1,2,2);
    plot(time_axis, x_reconst);
    ylim([-1.0 1.0]);
```

```
    grid on;
    title("Reconstructed Signal", "FontSize", 18);
    xlabel("Time [s]", "FontSize", 16);
    ylabel("Signal Amplitude y[n]", "FontSize", 16);

    pos = get(gcf, 'Position');
    set(gcf, 'Position', pos+[300 -50 300 -50]);

    if tosave == true
        saveas(fig2, "../plots/prob"+prob+"/orig_recons_M"+win_len+"_R"+win_shift+".png");
        close all;
    end
end
```