499 Part 2 Report

Deniz Soysal 2305332

October 28 2024

a)

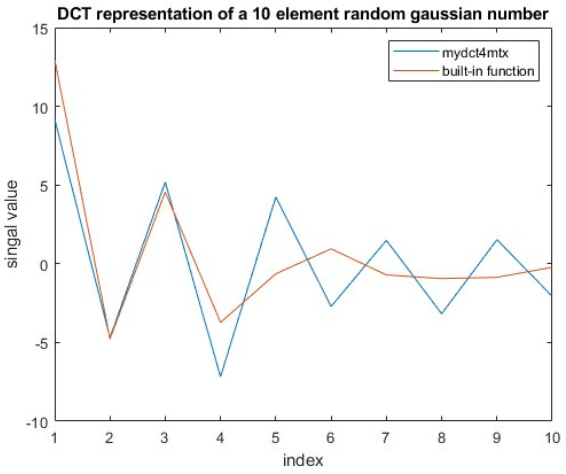
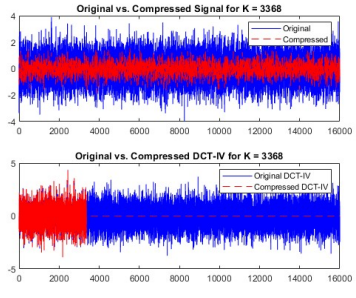


Figure 1: Comparison of results between D matrix generated by hand-written code and built-in function

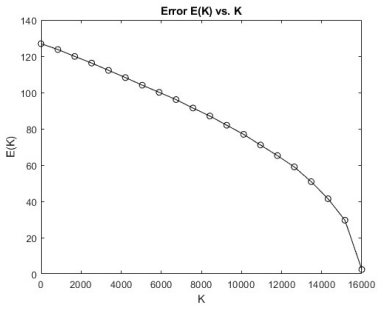
In this part, the handwritten mydct4mtx(N) function and the Matlab duct (x) function are compared. The results show obvious similarities.

b)

- For all signals, the compressed version sounded deeper and lacked bright tones.
- The Best signal for this kind of compression would be speech signals as the critical information of the signal was still present as the words were comprehensible after compression.

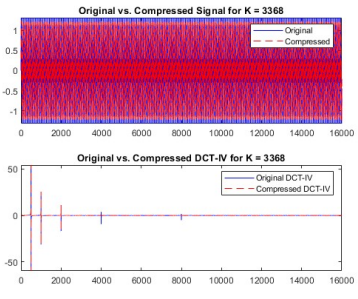


(a) Comparison of the original and compressed signals

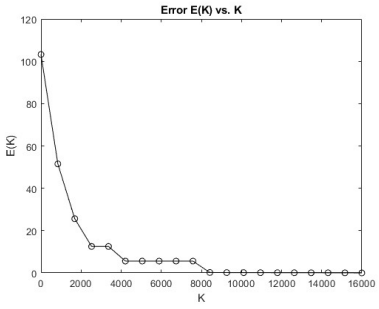


(b) Error plot for increasing values of K

Figure 2: Figures for dct applied on Gaussian signal for compression value of 4/19



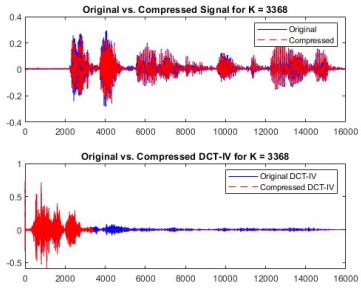
(a) Comparison of the original and compressed signals



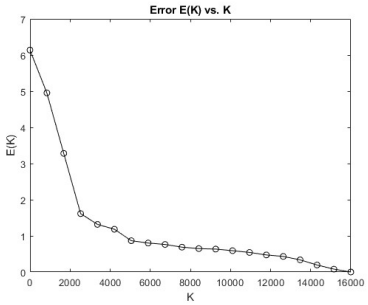
(b) Error plot for increasing values of K

Figure 3: Figures for dct applied on sine signal for compression value of 4/19.

- Error function dropped significantly earlier at audio and sinusoidal signals compared to the Gaussian. Which might imply that those signals arguably were better suited for this kind of compression
- An alternative would be a decision algorithm for choosing the best value of k by setting an error threshold for optimizing the information loss.



(a) Comparison of the original and compressed signals



(b) Error plot for increasing values of K

Figure 4: Figures for dct applied on audio signal for compression value of 4/19.

Appendix: scripts

mydct4mtx.m

```
function D = mydct4mtx(N)
    D = zeros(N, N); %initialize the matrix D according to the input value
    constant = sqrt(2 / N);

    %loop over each element to compute the matrix entries
    for k = 0:N-1
        for n = 0:N-1
            D(k+1, n+1) = constant * cos(pi / (4 * N) * (2 * n + 1) * (2 * k + 1));
        end
    end
end
```

Test script for problem_3_c.m

```
close all
clear all
clc

N = 10; % define dimension N
x = randi([0, 9], N, 1);


D = mydct4mtx(N); % compute dct by using mydct4mtx generted matrix D
y_mydct = D * x;

y_builtin = dct(x); % compute by built-in function for comparison

% plots
figure;
plot(y_mydct)
hold on

plot(y_builtin)
hold on

title("DCT representation of a 10 element random gaussian number")
legend("mydct4mtx","built-in function")
xlabel("index")
ylabel("singal value")
```

DCT_IV_compression_example.m

```
function [compressed_signal] = DCT_IV_compression_example(x, Fs)
% compression parameters
N = length(x);                    % signal length
K_vals = round(linspace(0, N-1, 20)); % 20 equal steps for K values
D = mydct4mtx(N);                 % get DCT-IV matrix from the previous function
y = D * x;                        % compute DCT-IV of x
errors = zeros(1, length(K_vals)); % pre-allocate error array


for i = 1:length(K_vals)
    K = K_vals(i);
    % compression
    y0K = y;
    y0K(K+1:end) = 0;

    % obtaining original signal after compression
    x0K = D' * y0K;

    % error
    errors(i) = sqrt(sum((x - x0K).^2));

    % plot results for one example K
    if i == 5 % time step to cut in compression
        % plot original and compressed signal
        figure;
        subplot(2, 1, 1);
        plot(x, 'b'); hold on;
        plot(x0K, 'r--');
        title(['Original vs. Compressed Signal for K = ', num2str(K)]);
        legend('Original', 'Compressed');

        % plot original and compressed DCT-IV
        subplot(2, 1, 2);
        plot(y, 'b'); hold on;
        plot(y0K, 'r--');
        title(['Original vs. Compressed DCT-IV for K = ', num2str(K)]);
        legend('Original DCT-IV', 'Compressed DCT-IV');

        compressed_signal = x0K;
    end
end

% final error plot
figure;
```

```
    plot(K_vals, errors, 'k-o');
    title('Error E(K) vs. K');
    xlabel('K'); ylabel('E(K)');
end
```

Test script for DCT_IV_compression_example.m

```
close all
clear all
clc

% % Generate Gaussian signal
% N = 16000; % length
% x = randn(N, 1); % signal
% Fs = 8000; % Sampling frequency
%
% x_compressed = DCT_IV_compression_example(x, Fs);

% % sine waves
% N = 16000;
% n = 0:N-1;
%
% % loop for summing sine waves
% x = 0;
% for i = 0:4
%     x = x + (1/2^i) * sin(2 * pi * 2^i / 64 * n);
% end
% x = x(:); % convert to column vector
% Fs = 8000;
%
% x_compressed = DCT_IV_compression_example(x, Fs);

% record audio
audioDuration = 2; % in seconds
[x, Fs] = recorder(audioDuration); % record

x_compressed = DCT_IV_compression_example(x, Fs);

soundsc(x, Fs);  % play the original signal
pause(3.0);
soundsc(x_compressed, Fs);  % play the compressed signal
```