Experiment No 04

BECS 32461

Paper C

**IMPLEMENTATION OF N - POINT FFT AND IFFT ALGORITHM**

Student Name: W. K. G. K. Jayawardana

Student No: EC/2021/006

Date Performed: 2025/10/17

Date Submitted: 2025/10/17

**PROCEDURE**

F01.

dt = 0.001;

t = 0:dt:1;

x\_clean = sin(2\*pi\*50\*t) + sin(2\*pi\*120\*t);

y\_noisy = x\_clean + 2.5\*randn(size(t));

figure;

plot(t, x\_clean, 'b-', 'LineWidth', 1.5);

hold on;

plot(t, y\_noisy, 'r-', 'LineWidth', 1);

legend('Clean', 'Noisy');

xlim([0, 0.25]);

ylim([-5, 5]);

A graph showing a red and blue line

AI-generated content may be incorrect.

F02.

dt = 0.001;

t = 0:dt:1;

x\_clean = sin(2\*pi\*50\*t) + sin(2\*pi\*120\*t);

y\_noisy = x\_clean + 2.5\*randn(size(t));

N = length(t);

Y = fft(y\_noisy, N);

PSD = Y .\* conj(Y) / N;

freq = (0:N-1) / (dt\*N);

figure;

plot(freq, PSD, 'b-', 'LineWidth', 1.5);

xlabel('Frequency (Hz)');

title('Power Spectrum');

xlim([0, 500]);

ylim([0, 300]);

A graph of a power spectrum

AI-generated content may be incorrect.

F03.

dt = 0.001;

t = 0:dt:1;

x\_clean = sin(2\*pi\*50\*t) + sin(2\*pi\*120\*t);

y\_noisy = x\_clean + 2.5\*randn(size(t));

N = length(t);

Y = fft(y\_noisy, N);

PSD = Y .\* conj(Y) / N;

freq = (0:N-1) / (dt\*N);

threshold = 60;

indices = PSD > threshold;

PSD\_filtered = PSD .\* indices;

figure(3);

plot(freq, PSD, 'b-', 'LineWidth', 1);

hold on;

plot(freq, PSD\_filtered, 'r-', 'LineWidth', 1.5);

xlabel('Frequency (Hz)');

title('Power Spectrums');

legend('Noisy', 'Filtered');

grid on;

xlim([0, 500]);

ylim([0, 350]);

A graph of a power spectrum

AI-generated content may be incorrect.

F04.

dt = 0.001;

t = 0:dt:1;

x\_clean = sin(2\*pi\*50\*t) + sin(2\*pi\*120\*t);

rng(42);

y\_noisy = x\_clean + 2.5\*randn(size(t));

N = length(t);

Y = fft(y\_noisy, N);

PSD = Y .\* conj(Y) / N;

threshold = 60;

indices = PSD > threshold;

Y\_filtered = Y .\* indices;

y\_filtered = real(ifft(Y\_filtered));

figure;

subplot(2,1,1);

plot(t, x\_clean, 'b-', 'LineWidth', 1.5);

hold on;

plot(t, y\_noisy, 'r-', 'LineWidth', 0.5);

legend('Clean', 'Noisy');

xlim([0, 0.25]);

ylim([-5, 5]);

subplot(2,1,2);

plot(t, x\_clean, 'b-', 'LineWidth', 1.5);

hold on;

plot(t, y\_filtered, 'r-', 'LineWidth', 1.5);

legend('Clean', 'Filtered');

xlim([0, 0.25]);

ylim([-5, 5]);

A close-up of a graph

AI-generated content may be incorrect.

**EXERCISE**

E01.

f1 = 20; f2 = 200; f3 = 400;

A = 1;

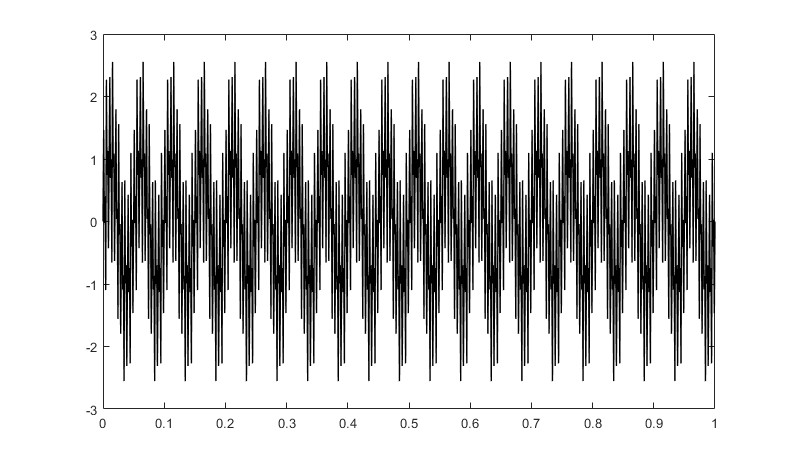
fs1 = 900;

t1 = 0:1/fs1:1;

x1 = A\*sin(2\*pi\*f1\*t1) + A\*sin(2\*pi\*f2\*t1) + A\*sin(2\*pi\*f3\*t1);

figure;

plot(t1, x1, 'k-', 'LineWidth', 1);



f1 = 20; f2 = 200; f3 = 400;

A = 1;

fs1 = 900;

t1 = 0:1/fs1:1;

x1 = A\*sin(2\*pi\*f1\*t1) + A\*sin(2\*pi\*f2\*t1) + A\*sin(2\*pi\*f3\*t1);

fs2 = 450;

t2 = 0:1/fs2:1;

x2 = A\*sin(2\*pi\*f1\*t2) + A\*sin(2\*pi\*f2\*t2) + A\*sin(2\*pi\*f3\*t2);

N1 = length(x1);

N2 = length(x2);

X1 = fft(x1, N1);

freq1 = (0:N1-1) \* fs1 / N1;

X2 = fft(x2, N2);

freq2 = (0:N2-1) \* fs2 / N2;

figure;

subplot(2,1,1);

plot(freq1(1:N1/2), abs(X1(1:N1/2)), 'k-', 'LineWidth', 1.5);

xlabel('Frequency (Hz)');

ylabel('Magnitude');

title('Frequency Spectrum - Sampling Rate: 900 Hz');

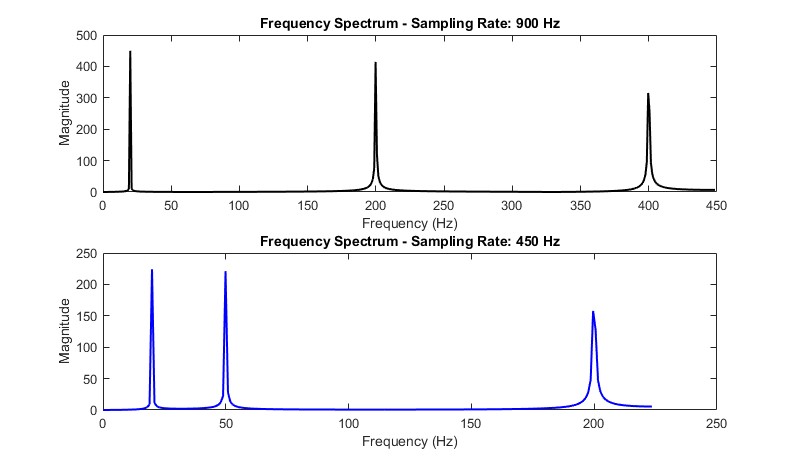
subplot(2,1,2);

plot(freq2(1:N2/2), abs(X2(1:N2/2)), 'b-', 'LineWidth', 1.5);

xlabel('Frequency (Hz)');

ylabel('Magnitude');

title('Frequency Spectrum - Sampling Rate: 450 Hz');



Nyquist rate for highest frequency (400 Hz): 800 Hz

When sampling at 450 Hz (below Nyquist rate), aliasing occurs.

The 400 Hz component appears as a lower frequency due to folding.

General rule: Sampling rate must be at least twice the highest frequency.

E02.

x\_original = [1, 1, 0, 0];

N = length(x\_original);

X = fft(x\_original);

x\_reconstructed = ifft(X);

fprintf('Original sequence: [');

fprintf('%.1f ', x\_original);

fprintf(']\n');

fprintf('Reconstructed sequence: [');

fprintf('%.6f ', real(x\_reconstructed));

fprintf(']\n');

figure;

subplot(3,1,1);

stem(0:N-1, x\_original, 'r', 'filled', 'LineWidth', 2);

title('Original Sequence');

xlabel('Discrete Time Index n');

ylabel('Amplitude');

xlim([-0.5, N-0.5]);

subplot(3,1,2);

stem(0:N-1, abs(X), 'k', 'filled', 'LineWidth', 1.5);

title('Magnitude of Frequency Components');

xlabel('Frequency Bin');

ylabel('Magnitude');

xlim([-0.5, N-0.5]);

subplot(3,1,3);

stem(0:N-1, real(x\_reconstructed), 'm', 'filled', 'LineWidth', 1.5);

title('Reconstructed Sequence');

xlabel('Discrete Time Index n');

ylabel('Amplitude');

xlim([-0.5, N-0.5]);

tolerance = 1e-10;

if max(abs(real(x\_reconstructed) - x\_original)) < tolerance

fprintf('The reconstructed sequence matches the original sequence\n');

else

fprintf('The reconstructed sequence does not match the original sequence\n');

end

A diagram of a sequence of sequence

AI-generated content may be incorrect.

A black and white text

AI-generated content may be incorrect.

E03.

x\_original = [1, 0, 1, 0];

N\_original = length(x\_original);

x\_padded5 = [x\_original, 0];

x\_padded10 = [x\_original, zeros(1, 6)];

X\_original = fft(x\_original);

X\_padded5 = fft(x\_padded5);

X\_padded10 = fft(x\_padded10);

x\_recon\_original = ifft(X\_original);

x\_recon\_padded5 = ifft(X\_padded5);

x\_recon\_padded10 = ifft(X\_padded10);

figure;

subplot(2,3,1);

stem(0:N\_original-1, x\_original, 'k', 'filled', 'LineWidth', 2);

title('Original Sequence (N=4)');

xlabel('n'); ylabel('Amplitude');

xlim([-0.5, 3.5]);

subplot(2,3,2);

stem(0:4, x\_padded5, 'k', 'filled', 'LineWidth', 2);

title('Zero-Padded Sequence (N=5)');

xlabel('n'); ylabel('Amplitude');

xlim([-0.5, 4.5]);

subplot(2,3,3);

stem(0:9, x\_padded10, 'k', 'filled', 'LineWidth', 2);

title('Zero-Padded Sequence (N=10)');

xlabel('n'); ylabel('Amplitude');

xlim([-0.5, 9.5]);

subplot(2,3,4);

freq\_orig = 0:N\_original-1;

freq\_pad5 = 0:4;

freq\_pad10 = 0:9;

stem(freq\_orig, abs(X\_original), 'r', 'filled', 'LineWidth', 2);

hold on;

stem(freq\_pad5, abs(X\_padded5), 'b', 'LineWidth', 2);

stem(freq\_pad10, abs(X\_padded10), 'g', 'LineWidth', 2);

title('Magnitude Spectrum');

xlabel('Frequency Bin'); ylabel('Magnitude');

legend('N=4', 'N=5', 'N=10');

subplot(2,3,5);

stem(freq\_orig, angle(X\_original), 'r', 'filled', 'LineWidth', 2);

hold on;

stem(freq\_pad5, angle(X\_padded5), 'b', 'LineWidth', 2);

stem(freq\_pad10, angle(X\_padded10), 'g', 'LineWidth', 2);

title('Phase Spectrum');

xlabel('Frequency Bin'); ylabel('Phase (rad)');

legend('N=4', 'N=5', 'N=10');

subplot(2,3,6);

stem(0:N\_original-1, real(x\_recon\_original), 'r', 'filled', 'LineWidth', 2);

hold on;

stem(0:4, real(x\_recon\_padded5), 'b', 'LineWidth', 2);

stem(0:9, real(x\_recon\_padded10), 'g', 'LineWidth', 2);

title('Reconstructed Signals');

xlabel('n'); ylabel('Amplitude');

legend('N=4', 'N=5', 'N=10');

error\_original = max(abs(real(x\_recon\_original) - x\_original));

error\_padded5 = max(abs(real(x\_recon\_padded5(1:4)) - x\_original));

error\_padded10 = max(abs(real(x\_recon\_padded10(1:4)) - x\_original));

fprintf('Maximum absolute differences:\n');

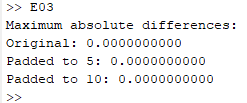
fprintf('Original: %.10f\n', error\_original);

fprintf('Padded to 5: %.10f\n', error\_padded5);

fprintf('Padded to 10: %.10f\n', error\_padded10);

A group of graphs with lines

AI-generated content may be incorrect.



Zero-padding effects:

- Increases frequency resolution (more frequency bins)

- Does not add new frequency information

- Provides smoother frequency spectrum interpolation

- Actual frequency components remain unchanged

>>

E04.

f0 = 8;

fs = 400;

duration = 1;

t = 0:1/fs:duration-1/fs;

N = length(t);

square\_wave = square(2\*pi\*f0\*t);

X = fft(square\_wave, N);

freq = (0:N-1) \* fs / N;

figure;

subplot(2,2,1);

stem(freq(1:N/2), abs(X(1:N/2)), 'k', 'LineWidth', 1);

title('Magnitude Spectrum');

xlabel('Frequency (Hz)'); ylabel('Magnitude');

subplot(2,2,2);

stem(freq(1:N/2), angle(X(1:N/2)), 'g', 'LineWidth', 1);

title('Phase Spectrum');

xlabel('Frequency (Hz)'); ylabel('Phase (rad)');

subplot(2,2,3);

stem(freq(1:N/2), real(X(1:N/2)), 'y', 'LineWidth', 1);

title('Real Part of FFT');

xlabel('Frequency (Hz)'); ylabel('Real');

subplot(2,2,4);

stem(freq(1:N/2), imag(X(1:N/2)), 'b', 'LineWidth', 1);

title('Imaginary Part of FFT');

xlabel('Frequency (Hz)'); ylabel('Imaginary');

A diagram of different types of data

AI-generated content may be incorrect.

f0 = 8;

fs = 400;

duration = 1;

t = 0:1/fs:duration-1/fs;

N = length(t);

square\_wave = square(2\*pi\*f0\*t);

X = fft(square\_wave, N);

freq = (0:N-1) \* fs / N;

X\_filtered = zeros(1, N);

harmonic\_indices = [1, 3, 5, 7]; % Harmonics to keep

for k = harmonic\_indices

freq\_component = k \* f0;

idx = round(freq\_component \* N / fs) + 1;

if idx <= N/2

X\_filtered(idx) = X(idx);

X\_filtered(N - idx + 2) = X(N - idx + 2); % Symmetric component

end

end

square\_reconstructed = real(ifft(X\_filtered));

figure;

plot(t, square\_wave, 'b-', 'LineWidth', 1.5, 'DisplayName', 'Original Square Wave');

hold on;

plot(t, square\_reconstructed, 'r--', 'LineWidth', 1.5, 'DisplayName', 'Reconstructed');

xlabel('Time (s)');

ylabel('Amplitude');

title('Square Wave Reconstruction using Selected Harmonics');

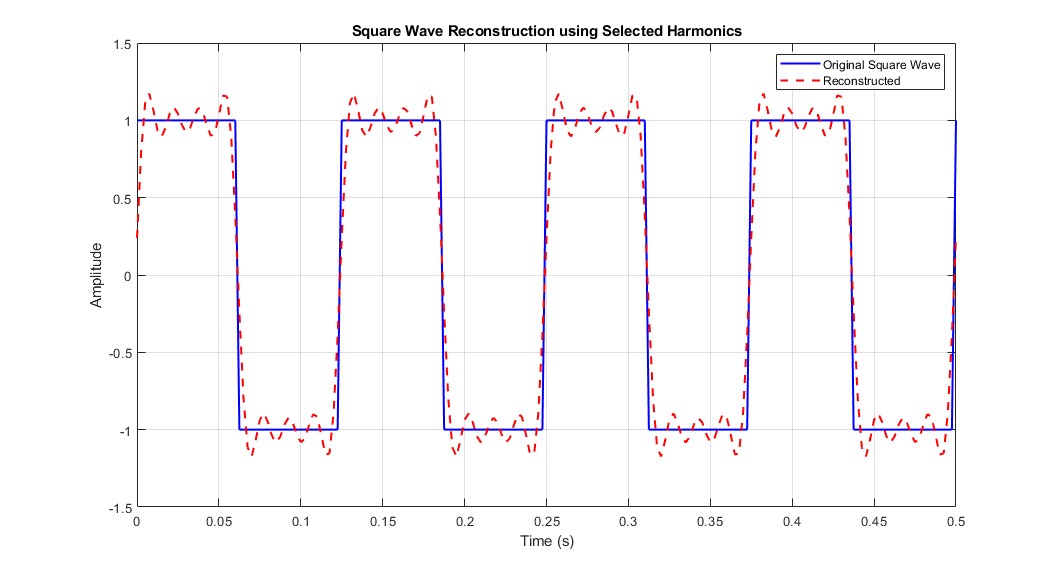
legend('show');

grid on;

xlim([0, 0.5]);

mse = mean((square\_wave - square\_reconstructed).^2);

fprintf('Mean Squared Error: %.6f\n', mse);



Effect of harmonics on reconstruction:

- More harmonics: Better approximation, sharper edges

- Fewer harmonics: Smoother reconstruction, rounded edges

- Gibbs phenomenon: Overshoot at discontinuities persists

- Square wave requires infinite harmonics for perfect reconstruction