

Signals and Systems 1

MATLab Project

Sean Greenwood

EECE - 3620

Section: 201

Signals and Systems Project

Submission Date: 11/07/2025

I. Introduction

This MATLAB project demonstrates the process of frequency translation and low-pass filtering for a continuous-time signal using numerical Fourier Transform methods. The signal

Signal 1**:** $x(t) = e^{-200t}\cos(100\pi t) + 1.2e^{-250t},\ 0 \le t \le 0.2$

is analyzed by computing its continuous-time Fourier Transform (CTFT) and by simulating the modulation and filtering operators in a system.

II. Student Written Code

```
% Shift X(ω) RIGHT by +100 Hz → Y(ω) = X(ω − ωc)
% (here ωc = 2π·fc with fc = 100 Hz)
YW = interp1(w, XW, w - wc, 'linear', 0);
% W(ω): passband −200 Hz to +200 Hz
fcut_W = 200;                  % Hz
wc_W   = 2*pi*fcut_W;          % rad/s
H      = double(abs(w) <= wc_W);
% Low-pass filter: W(ω) = Y(ω) · H(ω)
WW = YW .* H;
% Compute F by mixing W(t) down in TIME
wt = ICTFourierTransform(w, dw, WW, t);    % W(t)
ft = wt .* exp(-1i*wc*t);                  % multiply by e^{-j ωc t}
FW = CTFourierTransform(t, dt, ft, w);     % F(ω)
f = w/(2*pi);
figure; hold on;
plot(f, abs(XW), 'r', 'LineWidth', 1.5, 'DisplayName', '|X(j\omega)|');
plot(f, abs(YW), 'g', 'LineWidth', 1.5, 'DisplayName', '|Y(j\omega)|');
plot(f, abs(WW), 'b', 'LineWidth', 1.5, 'DisplayName', '|W(j\omega)|');
plot(f, abs(FW), 'k', 'LineWidth', 1.5, 'DisplayName', '|F(j\omega)|');
xline(+200, ':b'); xline(-200, ':b');   % LPF band edges in Hz
xlabel('f (Hz)'); ylabel('Magnitude');
title('Spectra on One Plot ');
legend('show', 'Location', 'best');
grid on; xlim([-fw fw]); hold off;
```

III. Full Code

```
clear all;
close all;
% Given constants
fc = 100;                % Cutoff frequency (Hz)
f0 = 200;                % Reference frequency (Hz)
wc = 2*pi*fc;            % Cutoff angular frequency
w0 = 2*pi*f0;            % Reference angular frequency
dw = 0.1*pi;             % Frequency resolution
fw = 400;                % Max frequency for evaluation (Hz)
ww = 2*pi*fw;            % Angular frequency limit
w  = -ww:dw:ww;          % Angular frequency vector
dt = 0.0005;             % Time step (s)
t  = 0:dt:0.2;           % Time vector
% Define the signal x(t)
xt = exp(-200*t).*cos(100*pi*t) + 1.2*exp(-250*t);
% Continuous-time Fourier Transform
XW = CTFourierTransform(t, dt, xt, w);
% Plot |X(ω)|
figure;
plot(w/(2*pi), abs(XW), 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('|X(ω)|');
title('Magnitude Spectrum of X(ω)');
grid on;
% Inverse Fourier Transform to verify reconstruction
xtr = ICTFourierTransform(w, dw, XW, t);
figure;
hold on;
plot(t, xt, 'r', 'LineWidth', 1);
plot(t, real(xtr), 'b--', 'LineWidth', 1);
xlabel('Time (s)');
ylabel('Amplitude');
title('Original and Reconstructed Signals');
legend('Original x(t)', 'Reconstructed x_{r}(t)');
grid on;
hold off;
% ===== STUDENT SECTION (paste in your script) =====
% Shift X(ω) RIGHT by +100 Hz → Y(ω) = X(ω − ωc)
% (here ωc = 2π·fc with fc = 100 Hz)
YW = interp1(w, XW, w - wc, 'linear', 0);
% W(ω): passband −200 Hz to +200 Hz
fcut_W = 200;                  % Hz
wc_W   = 2*pi*fcut_W;          % rad/s
H      = double(abs(w) <= wc_W);
% Low-pass filter: W(ω) = Y(ω) · H(ω)
WW = YW .* H;
% Compute F by mixing W(t) down in TIME
wt = ICTFourierTransform(w, dw, WW, t);    % W(t)
ft = wt .* exp(-1i*wc*t);                  % multiply by e^{-j ωc t}
FW = CTFourierTransform(t, dt, ft, w);     % F(ω)
f = w/(2*pi);
figure; hold on;
plot(f, abs(XW), 'r', 'LineWidth', 1.5, 'DisplayName', '|X(j\omega)|');
plot(f, abs(YW), 'g', 'LineWidth', 1.5, 'DisplayName', '|Y(j\omega)|');
plot(f, abs(WW), 'b', 'LineWidth', 1.5, 'DisplayName', '|W(j\omega)|');
plot(f, abs(FW), 'k', 'LineWidth', 1.5, 'DisplayName', '|F(j\omega)|');
```

```
xline(+200, ':b'); xline(-200, ':b');    % LPF band edges in Hz
xlabel('f (Hz)'); ylabel('Magnitude');
title('Spectra on One Plot ');
legend('show', 'Location', 'best');
grid on; xlim([-fw fw]); hold off;
% ===== END STUDENT SECTION =====
% ===== Helper Functions =====
% CTFourierTransform (from CTF.txt)
function FT = CTFourierTransform(t,dt,xt,w)
LengthW = length(w);
FT = zeros(1,LengthW);
for len = 1:LengthW
   wl = w(len);
   FT(len)= sum(xt.*exp(-1j*wl*t))*dt;
end
end
% ICTFourierTransform (from ICT.txt)
function xt = ICTFourierTransform(w,dw,XW,t)
LengthT = length(t);
xt = zeros(1,LengthT);
for len = 1:LengthT
   tt = t(len);
   xt(len)= sum(XW.*exp(1j*w*tt))*dw/(2*pi);
end
end
```

IV. Figures

Figure 1.

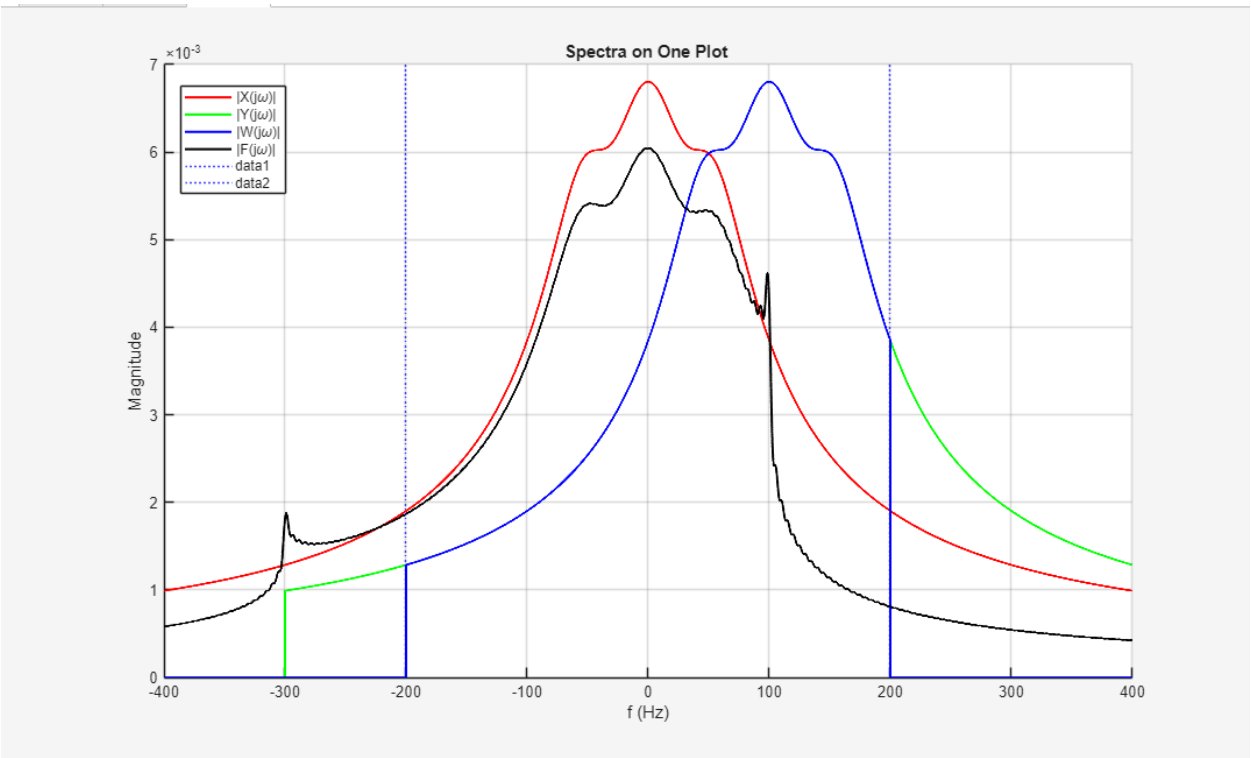


Figure 1 displays all four signals measured in the system, along with how signal 1 is manipulated.

Figure 2. Original and Reconstructed x(t)

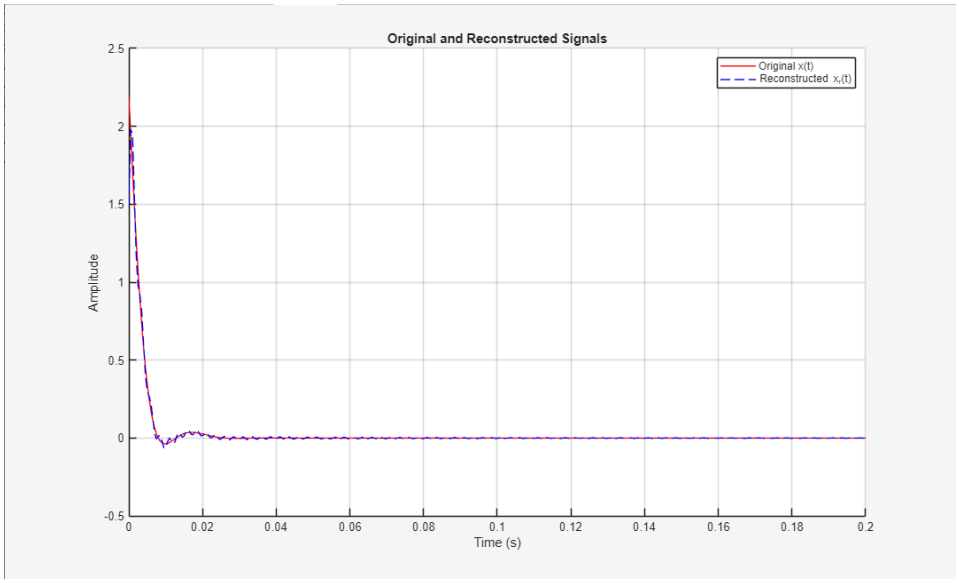


Figure 2 *displays the original and reconstructed x(t)*

Figure 3. Magnitude Spectrum of X(jω)

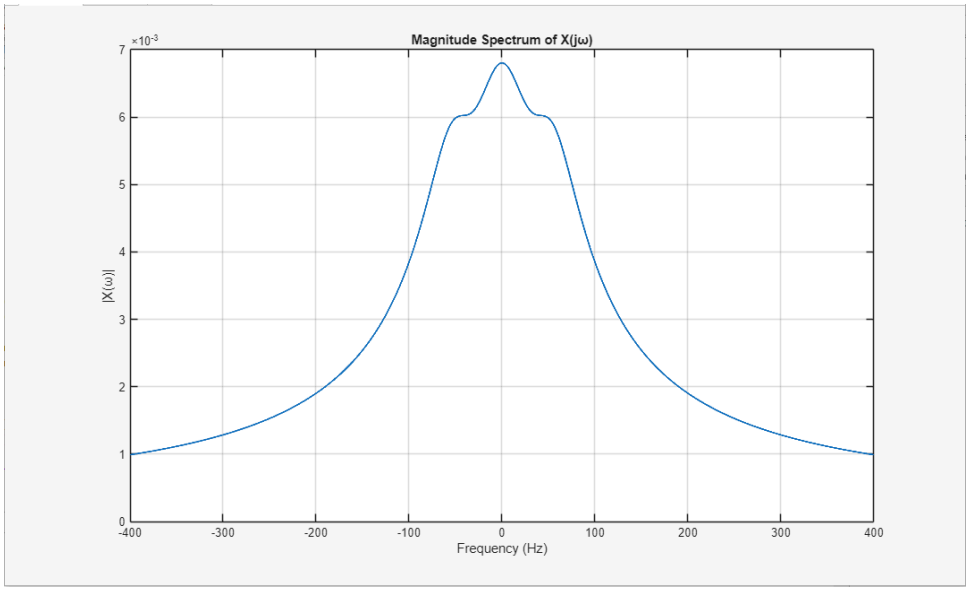
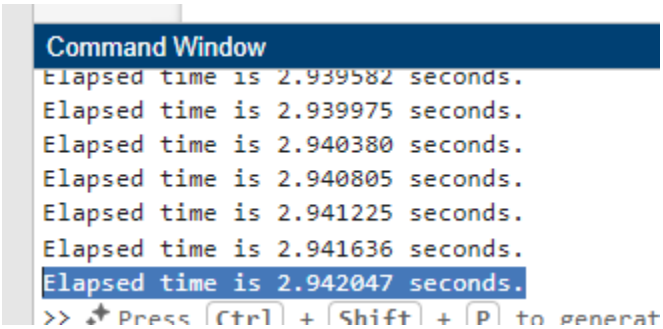


Figure 3 *displays the magnitude of the Fourier transform of the function described by* ***signal 1****.*

V. System Information

Total Elapsed Run Time: 2.942047 Seconds



This was measured using the “tic” and “toc” commands on MATLAB.

System Clock: 2.80Ghz

System Cores: 4 cores

Logical Processors: 8 logical processors