Fourier Transform Properties – Gate Function

clear all;

close all;

clc;

% Define parameters

step\_omega = 0.01 \* pi;

step\_size\_t = 0.1;

t = -40:step\_size\_t:40;

omegax = -(1/step\_size\_t)\*pi:step\_omega:(1/step\_size\_t)\*pi;

% Pre-allocate Fourier transform matrix and other matrices

exp\_omega = zeros(length(omegax), length(t));

x\_omega = zeros(1, length(omegax));

% Precompute exponential terms

for ii = 1:length(omegax)

exp\_omega(ii,:) = exp(-1j \* omegax(ii) .\* t);

end

% Define Gate Function (Rectangular Pulse)

width = 1; % Width of the gate

x\_t = (t >= -width/2) & (t <= width/2);

subplot(4, 1, 1), plot(t, x\_t)

title('Original Signal x(t) - Gate Function')

% Compute Fourier Transform

for ii = 1:length(omegax)

temp = x\_t .\* exp\_omega(ii,:);

x\_omega(1,ii) = sum(temp) \* step\_size\_t; % Integral approximation

end

% Plot Magnitude and Phase Spectrum

subplot(4,1,2), plot(omegax, abs(x\_omega));

title('Magnitude Spectrum |X(\omega)|')

subplot(4,1,3), plot(omegax, angle(x\_omega));

title('Phase Spectrum ∠X(\omega)')

% Reconstruct Signal from Fourier Transform

x\_t\_reconstructed = zeros(1, length(t));

for ii = 1:length(t)

expo\_omega\_2(ii,:) = exp(1j \* omegax .\* t(ii));

temp2 = x\_omega .\* expo\_omega\_2(ii,:);

x\_t\_reconstructed(1,ii) = (1 / (2 \* pi)) \* sum(temp2) \* step\_omega;

end

subplot(4,1,4), plot(t, real(x\_t\_reconstructed));

title('Reconstructed Signal from X(\omega)')

% Property 1: Shifting Property

shift = 10; % Example shift amount

x\_t\_shifted = (t - shift >= -width/2) & (t - shift <= width/2);

figure;

subplot(3,1,1), plot(t, x\_t\_shifted);

title(['Shifted Signal x(t - ', num2str(shift), ')']);

x\_omega\_shifted = zeros(1, length(omegax));

for ii = 1:length(omegax)

temp\_shift = x\_t\_shifted .\* exp\_omega(ii,:);

x\_omega\_shifted(1,ii) = sum(temp\_shift) \* step\_size\_t;

end

subplot(3,1,2), plot(omegax, abs(x\_omega\_shifted));

title('Magnitude Spectrum |X(\omega)| after Shifting')

% Phase spectrum for shifted signal

subplot(3,1,3), plot(omegax, angle(x\_omega\_shifted));

title('Phase Spectrum ∠X(\omega) after Shifting')

% Property 2: Scaling Property

scale\_factor = 2; % Example scale factor

x\_t\_scaled = ((t / scale\_factor) >= -width/2) & ((t / scale\_factor) <= width/2);

figure;

subplot(3,1,1), plot(t, x\_t\_scaled);

title(['Scaled Signal x(at) with a = ', num2str(scale\_factor)])

x\_omega\_scaled = zeros(1, length(omegax));

for ii = 1:length(omegax)

temp\_scale = x\_t\_scaled .\* exp\_omega(ii,:);

x\_omega\_scaled(1,ii) = sum(temp\_scale) \* step\_size\_t;

end

subplot(3,1,2), plot(omegax, abs(x\_omega\_scaled));

title('Magnitude Spectrum |X(\omega)| after Scaling')

% Phase spectrum for scaled signal

subplot(3,1,3), plot(omegax, angle(x\_omega\_scaled));

title('Phase Spectrum ∠X(\omega) after Scaling')

% Property 3: Parseval's Theorem

energy\_time\_domain = sum(abs(x\_t).^2) \* step\_size\_t;

energy\_freq\_domain = sum(abs(x\_omega).^2) \* step\_omega / (2 \* pi);

fprintf('Energy in time domain: %.4f\n', energy\_time\_domain);

fprintf('Energy in frequency domain: %.4f\n', energy\_freq\_domain);

% Check if Parseval's theorem holds

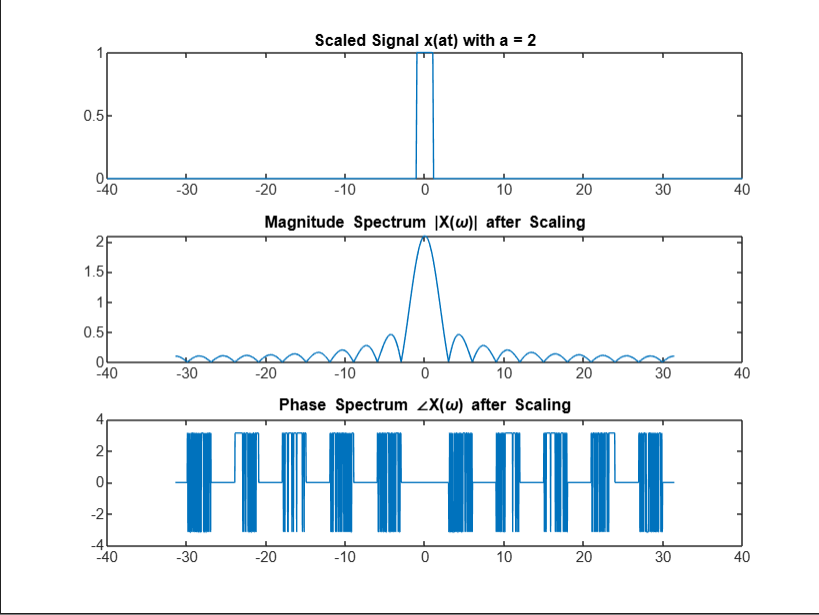
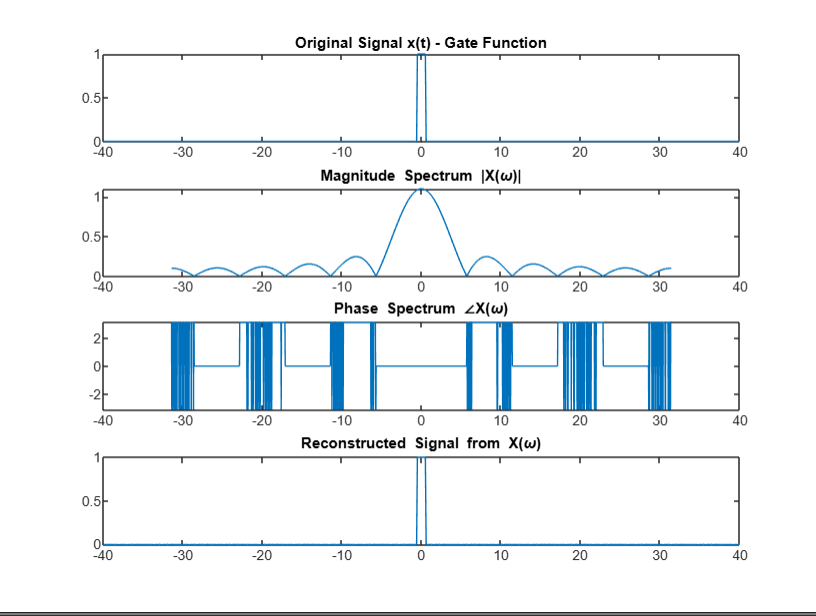
if abs(energy\_time\_domain - energy\_freq\_domain) < 1e-3

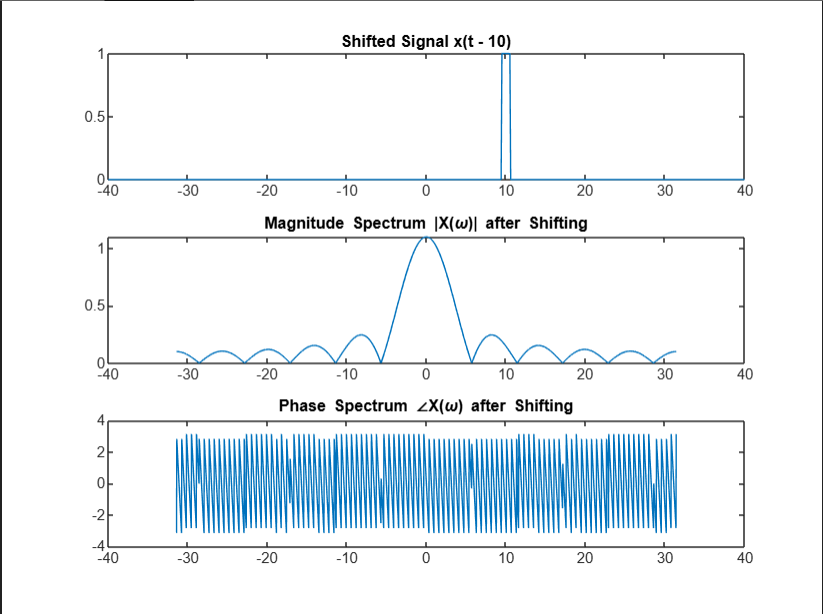
disp('Parseval''s theorem is verified: Energy in time and frequency domains are equal.');

else

disp('Parseval''s theorem is not verified: Energy in time and frequency domains differ.');

end





Fourier Transform Properties – Sinc Function

clear all;

close all;

clc;

% Define parameters

step\_omega = 0.01 \* pi;

step\_size\_t = 0.1;

t = -40:step\_size\_t:40;

omegax = -(1/step\_size\_t) \* pi : step\_omega : (1/step\_size\_t) \* pi;

% Pre-allocate Fourier transform matrix and other matrices

exp\_omega = zeros(length(omegax), length(t));

x\_omega = zeros(1, length(omegax));

% Precompute exponential terms for Fourier transform

for ii = 1:length(omegax)

exp\_omega(ii,:) = exp(-1j \* omegax(ii) .\* t);

end

% Define Sinc Function

for ii = 1:length(t)

if t(ii) == 0

x\_t(ii) = 1; % sinc(0) = 1

else

x\_t(ii) = sin(pi \* t(ii)) / (pi \* t(ii)); % Sinc function

end

end

subplot(4, 1, 1), plot(t, x\_t);

title('Original Signal x(t) - Sinc Function');

% Compute Fourier Transform of Sinc Function

for ii = 1:length(omegax)

temp = x\_t .\* exp\_omega(ii,:);

x\_omega(1, ii) = sum(temp) \* step\_size\_t; % Approximate integral

end

% Plot Magnitude and Phase Spectrum for Sinc Function

subplot(4,1,2), plot(omegax, abs(x\_omega));

title('Magnitude Spectrum |X(\omega)| of Sinc Function');

subplot(4,1,3), plot(omegax, angle(x\_omega));

title('Phase Spectrum ∠X(\omega) of Sinc Function');

% Reconstruct Signal from Fourier Transform

x\_t\_reconstructed = zeros(1, length(t));

for ii = 1:length(t)

temp2 = x\_omega .\* exp(1j \* omegax \* t(ii));

x\_t\_reconstructed(ii) = (1 / (2 \* pi)) \* sum(temp2) \* step\_omega;

end

subplot(4,1,4), plot(t, real(x\_t\_reconstructed));

title('Reconstructed Signal from X(\omega)');

% Property 1: Shifting Property

shift = 10; % Example shift amount

x\_t\_shifted = zeros(1, length(t));

for ii = 1:length(t)

if t(ii) - shift == 0

x\_t\_shifted(ii) = 1; % sinc(0) = 1

else

x\_t\_shifted(ii) = sin(pi \* (t(ii) - shift)) / (pi \* (t(ii) - shift));

end

end

figure;

subplot(3,1,1), plot(t, x\_t\_shifted);

title(['Shifted Signal x(t - ', num2str(shift), ') - Sinc Function']);

x\_omega\_shifted = zeros(1, length(omegax));

for ii = 1:length(omegax)

temp\_shift = x\_t\_shifted .\* exp\_omega(ii,:);

x\_omega\_shifted(ii) = sum(temp\_shift) \* step\_size\_t;

end

subplot(3,1,2), plot(omegax, abs(x\_omega\_shifted));

title('Magnitude Spectrum |X(\omega)| after Shifting');

% Phase spectrum for shifted signal

subplot(3,1,3), plot(omegax, angle(x\_omega\_shifted));

title('Phase Spectrum ∠X(\omega) after Shifting');

% Property 2: Scaling Property

scale\_factor = 2; % Example scale factor

x\_t\_scaled = zeros(1, length(t));

for ii = 1:length(t)

if (t(ii) / scale\_factor) == 0

x\_t\_scaled(ii) = 1; % sinc(0) = 1

else

x\_t\_scaled(ii) = sin(pi \* (t(ii) / scale\_factor)) / (pi \* (t(ii) / scale\_factor));

end

end

figure;

subplot(3,1,1), plot(t, x\_t\_scaled);

title(['Scaled Signal x(at) with a = ', num2str(scale\_factor), ' - Sinc Function']);

x\_omega\_scaled = zeros(1, length(omegax));

for ii = 1:length(omegax)

temp\_scale = x\_t\_scaled .\* exp\_omega(ii,:);

x\_omega\_scaled(ii) = sum(temp\_scale) \* step\_size\_t;

end

subplot(3,1,2), plot(omegax, abs(x\_omega\_scaled));

title('Magnitude Spectrum |X(\omega)| after Scaling');

% Phase spectrum for scaled signal

subplot(3,1,3), plot(omegax, angle(x\_omega\_scaled));

title('Phase Spectrum ∠X(\omega) after Scaling');

% Property 3: Parseval's Theorem for Sinc Function

energy\_time\_domain = sum(abs(x\_t).^2) \* step\_size\_t;

energy\_freq\_domain = sum(abs(x\_omega).^2) \* step\_omega / (2 \* pi);

fprintf('Energy in time domain: %.4f\n', energy\_time\_domain);

fprintf('Energy in frequency domain: %.4f\n', energy\_freq\_domain);

% Check if Parseval's theorem holds

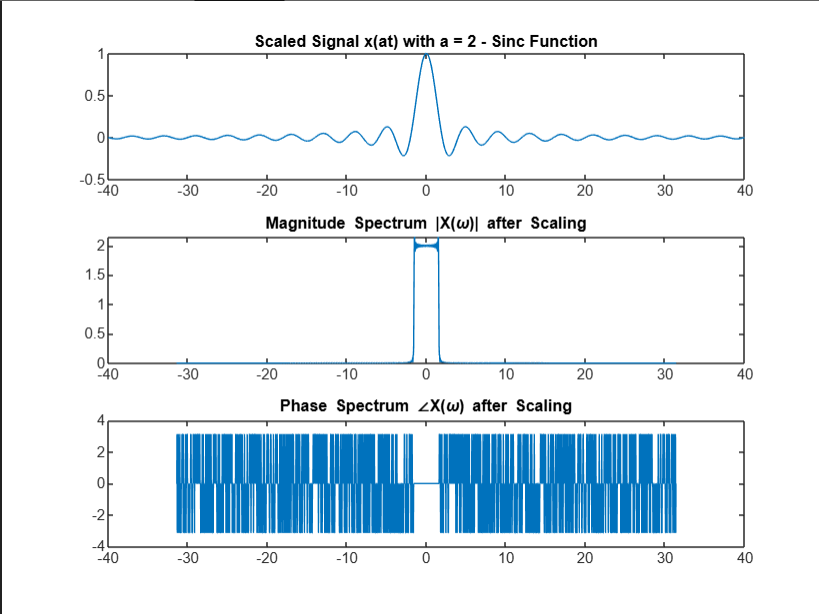
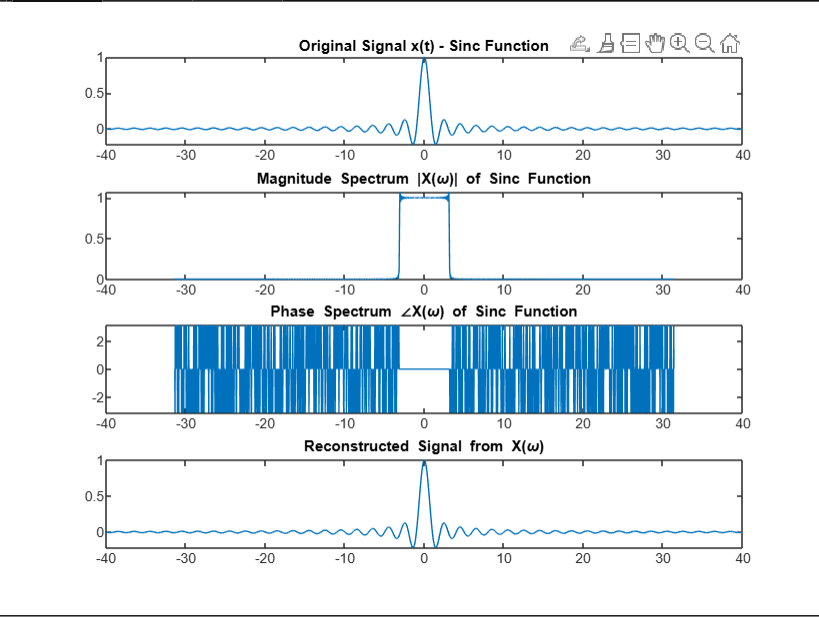
if abs(energy\_time\_domain - energy\_freq\_domain) < 1e-3

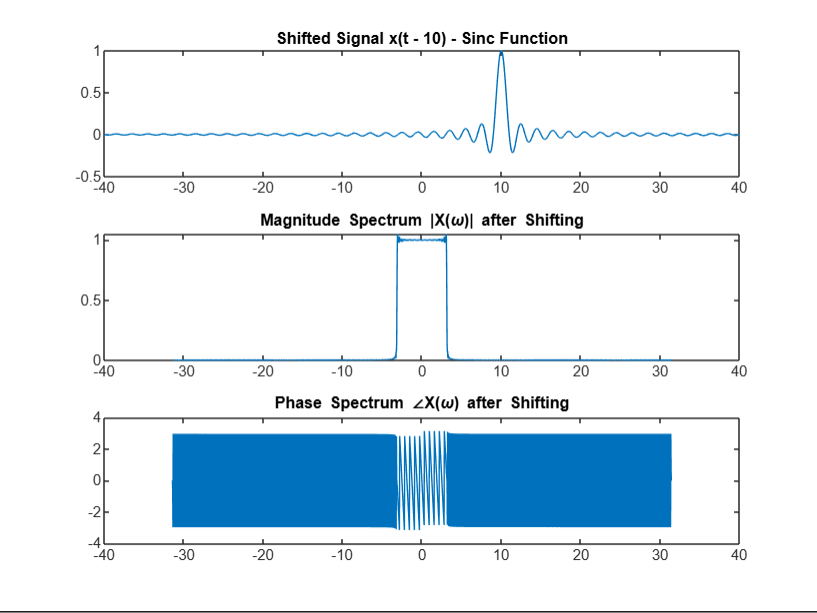
disp('Parseval''s theorem is verified: Energy in time and frequency domains are equal.');

else

disp('Parseval''s theorem is not verified: Energy in time and frequency domains differ.');

end





Fourier Transform Properties – Unit Step Function

clear all;

close all;

clc;

% Define parameters

step\_omega = 0.01 \* pi;

step\_size\_t = 0.1;

t = -40:step\_size\_t:40;

omegax = -(1/step\_size\_t)\*pi : step\_omega : (1/step\_size\_t)\*pi;

% Pre-allocate Fourier transform matrix and other matrices

exp\_omega = zeros(length(omegax), length(t));

x\_omega = zeros(1, length(omegax));

% Precompute exponential terms

for ii = 1:length(omegax)

exp\_omega(ii,:) = exp(-1j \* omegax(ii) .\* t);

end

% Define Unit Step Function

x\_t = (t >= 0); % Unit step function u(t)

subplot(4, 1, 1), plot(t, x\_t)

title('Original Signal x(t) - Unit Step Function')

% Compute Fourier Transform of Unit Step Function

for ii = 1:length(omegax)

temp = x\_t .\* exp\_omega(ii,:);

x\_omega(1,ii) = sum(temp) \* step\_size\_t; % Integral approximation

end

% Plot Magnitude and Phase Spectrum

subplot(4,1,2), plot(omegax, abs(x\_omega));

title('Magnitude Spectrum |X(\omega)|')

subplot(4,1,3), plot(omegax, angle(x\_omega));

title('Phase Spectrum ∠X(\omega)')

% Reconstruct Signal from Fourier Transform

x\_t\_reconstructed = zeros(1, length(t));

for ii = 1:length(t)

expo\_omega\_2(ii,:) = exp(1j \* omegax .\* t(ii));

temp2 = x\_omega .\* expo\_omega\_2(ii,:);

x\_t\_reconstructed(1,ii) = (1 / (2 \* pi)) \* sum(temp2) \* step\_omega;

end

subplot(4,1,4), plot(t, real(x\_t\_reconstructed));

title('Reconstructed Signal from X(\omega)')

% Property 1: Shifting Property

shift = 10; % Example shift amount

x\_t\_shifted = (t - shift >= 0); % Shifted unit step function u(t - shift)

figure;

subplot(3,1,1), plot(t, x\_t\_shifted);

title(['Shifted Signal x(t - ', num2str(shift), ') - Unit Step Function']);

x\_omega\_shifted = zeros(1, length(omegax));

for ii = 1:length(omegax)

temp\_shift = x\_t\_shifted .\* exp\_omega(ii,:);

x\_omega\_shifted(1,ii) = sum(temp\_shift) \* step\_size\_t;

end

subplot(3,1,2), plot(omegax, abs(x\_omega\_shifted));

title('Magnitude Spectrum |X(\omega)| after Shifting')

% Phase spectrum for shifted signal

subplot(3,1,3), plot(omegax, angle(x\_omega\_shifted));

title('Phase Spectrum ∠X(\omega) after Shifting')

% Property 2: Scaling Property

scale\_factor = 2; % Example scale factor

x\_t\_scaled = ((t / scale\_factor) >= 0); % Scaled unit step function u(at)

figure;

subplot(3,1,1), plot(t, x\_t\_scaled);

title(['Scaled Signal x(at) with a = ', num2str(scale\_factor)])

x\_omega\_scaled = zeros(1, length(omegax));

for ii = 1:length(omegax)

temp\_scale = x\_t\_scaled .\* exp\_omega(ii,:);

x\_omega\_scaled(1,ii) = sum(temp\_scale) \* step\_size\_t;

end

subplot(3,1,2), plot(omegax, abs(x\_omega\_scaled));

title('Magnitude Spectrum |X(\omega)| after Scaling')

% Phase spectrum for scaled signal

subplot(3,1,3), plot(omegax, angle(x\_omega\_scaled));

title('Phase Spectrum ∠X(\omega) after Scaling')

% Property 3: Parseval's Theorem

energy\_time\_domain = sum(abs(x\_t).^2) \* step\_size\_t;

energy\_freq\_domain = sum(abs(x\_omega).^2) \* step\_omega / (2 \* pi);

fprintf('Energy in time domain: %.4f\n', energy\_time\_domain);

fprintf('Energy in frequency domain: %.4f\n', energy\_freq\_domain);

% Check if Parseval's theorem holds

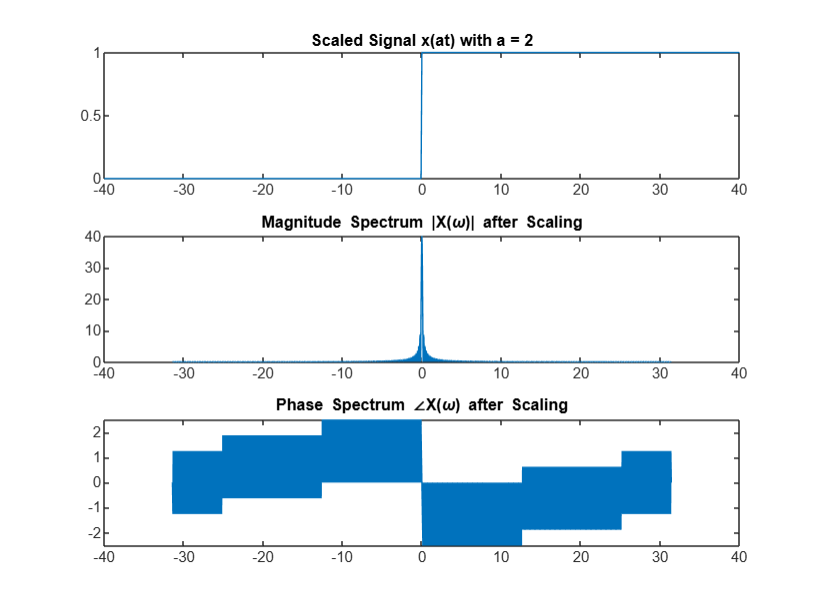
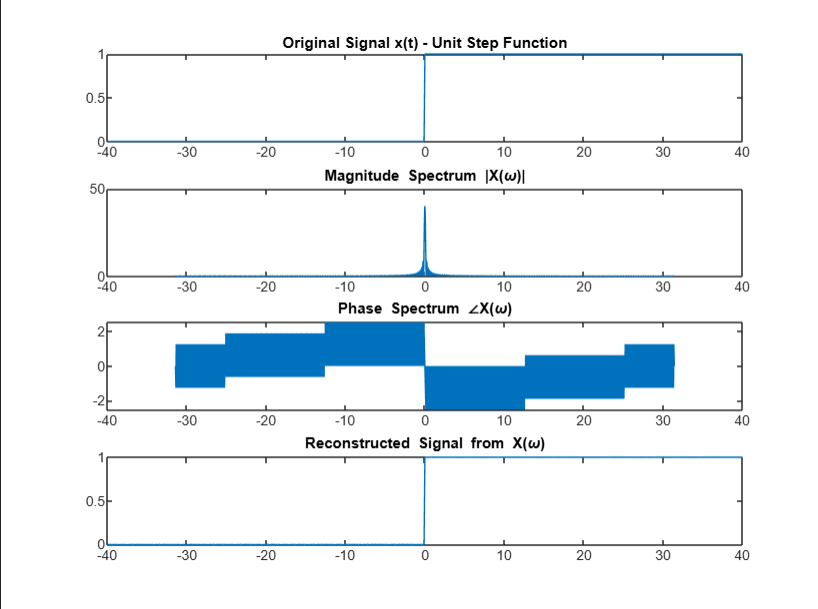
if abs(energy\_time\_domain - energy\_freq\_domain) < 1e-3

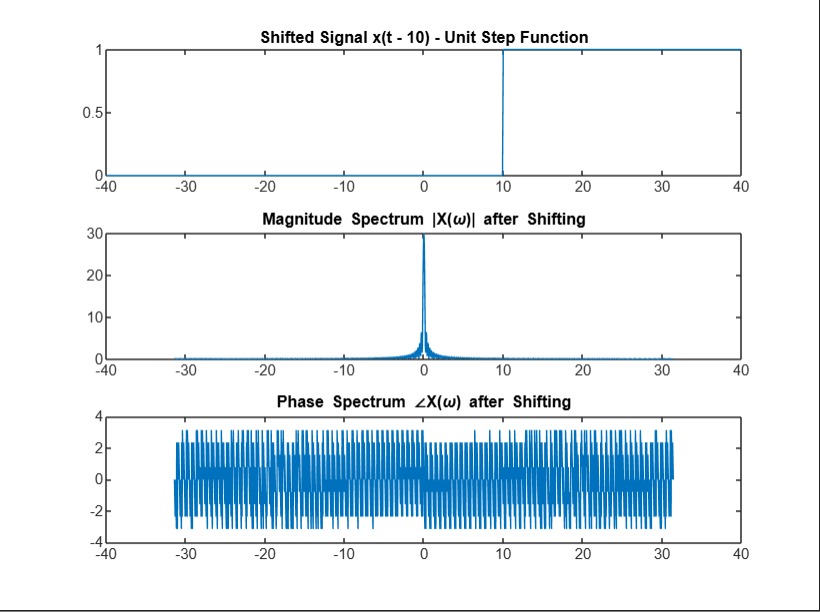
disp('Parseval''s theorem is verified: Energy in time and frequency domains are equal.');

else

disp('Parseval''s theorem is not verified: Energy in time and frequency domains differ.');

end





Fourier Transform Properties – Signum Function

clear all;

close all;

clc;

% Define parameters

step\_omega = 0.01 \* pi;

step\_size\_t = 0.1;

t = -40:step\_size\_t:40;

omegax = -(1/step\_size\_t) \* pi : step\_omega : (1/step\_size\_t) \* pi;

% Pre-allocate Fourier transform matrix and other matrices

exp\_omega = zeros(length(omegax), length(t));

x\_omega = zeros(1, length(omegax));

% Precompute exponential terms for Fourier transform

for ii = 1:length(omegax)

exp\_omega(ii,:) = exp(-1j \* omegax(ii) .\* t);

end

% Define Signum Function

v\_t = zeros(1, length(t));

for ii = 1:length(t)

if t(ii) < 0

v\_t(ii) = -1; % Negative values

elseif t(ii) == 0

v\_t(ii) = 0; % Zero

else

v\_t(ii) = 1; % Positive values

end

end

subplot(4, 1, 1), plot(t, v\_t);

title('Original Signal v(t) - Signum Function');

% Compute Fourier Transform of Signum Function

for ii = 1:length(omegax)

temp = v\_t .\* exp\_omega(ii,:);

x\_omega(1, ii) = sum(temp) \* step\_size\_t; % Approximate integral

end

% Plot Magnitude and Phase Spectrum

subplot(4,1,2), plot(omegax, abs(x\_omega));

title('Magnitude Spectrum |X(\omega)|');

subplot(4,1,3), plot(omegax, angle(x\_omega));

title('Phase Spectrum ∠X(\omega)');

% Reconstruct Signal from Fourier Transform

v\_t\_reconstructed = zeros(1, length(t));

for ii = 1:length(t)

temp2 = x\_omega .\* exp(1j \* omegax \* t(ii));

v\_t\_reconstructed(ii) = (1 / (2 \* pi)) \* sum(temp2) \* step\_omega;

end

subplot(4,1,4), plot(t, real(v\_t\_reconstructed));

title('Reconstructed Signal from X(\omega)');

% Property 1: Shifting Property

shift = 10; % Example shift amount

v\_t\_shifted = zeros(1, length(t));

for ii = 1:length(t)

if t(ii) - shift < 0

v\_t\_shifted(ii) = -1;

elseif t(ii) - shift == 0

v\_t\_shifted(ii) = 0;

else

v\_t\_shifted(ii) = 1;

end

end

figure;

subplot(3,1,1), plot(t, v\_t\_shifted);

title(['Shifted Signal v(t - ', num2str(shift), ')']);

x\_omega\_shifted = zeros(1, length(omegax));

for ii = 1:length(omegax)

temp\_shift = v\_t\_shifted .\* exp\_omega(ii,:);

x\_omega\_shifted(ii) = sum(temp\_shift) \* step\_size\_t;

end

subplot(3,1,2), plot(omegax, abs(x\_omega\_shifted));

title('Magnitude Spectrum |X(\omega)| after Shifting');

% Phase spectrum for shifted signal

subplot(3,1,3), plot(omegax, angle(x\_omega\_shifted));

title('Phase Spectrum ∠X(\omega) after Shifting');

% Property 2: Scaling Property

scale\_factor = 2; % Example scale factor

v\_t\_scaled = zeros(1, length(t));

for ii = 1:length(t)

if (t(ii) / scale\_factor) < 0

v\_t\_scaled(ii) = -1;

elseif (t(ii) / scale\_factor) == 0

v\_t\_scaled(ii) = 0;

else

v\_t\_scaled(ii) = 1;

end

end

figure;

subplot(3,1,1), plot(t, v\_t\_scaled);

title(['Scaled Signal v(at) with a = ', num2str(scale\_factor)]);

x\_omega\_scaled = zeros(1, length(omegax));

for ii = 1:length(omegax)

temp\_scale = v\_t\_scaled .\* exp\_omega(ii,:);

x\_omega\_scaled(ii) = sum(temp\_scale) \* step\_size\_t;

end

subplot(3,1,2), plot(omegax, abs(x\_omega\_scaled));

title('Magnitude Spectrum |X(\omega)| after Scaling');

% Phase spectrum for scaled signal

subplot(3,1,3), plot(omegax, angle(x\_omega\_scaled));

title('Phase Spectrum ∠X(\omega) after Scaling');

% Property 3: Parseval's Theorem

energy\_time\_domain = sum(abs(v\_t).^2) \* step\_size\_t;

energy\_freq\_domain = sum(abs(x\_omega).^2) \* step\_omega / (2 \* pi);

fprintf('Energy in time domain: %.4f\n', energy\_time\_domain);

fprintf('Energy in frequency domain: %.4f\n', energy\_freq\_domain);

% Check if Parseval's theorem holds

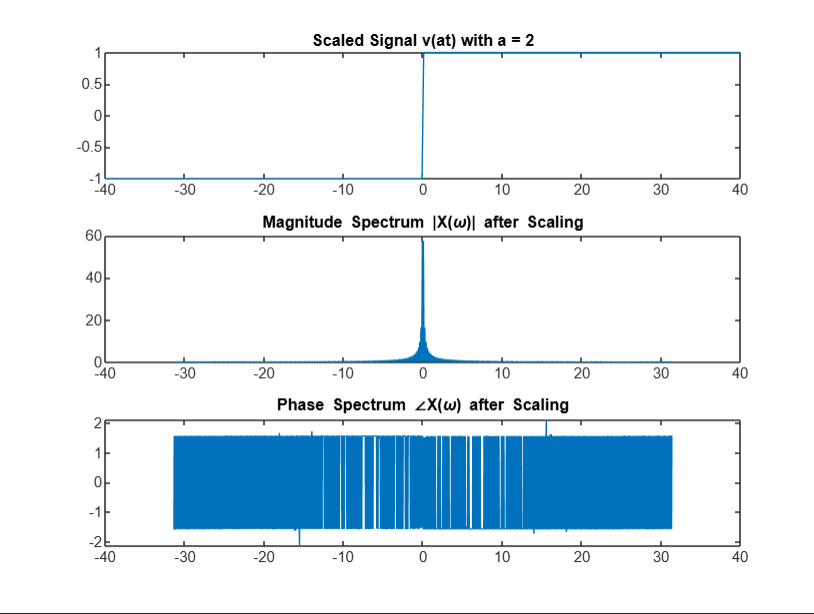
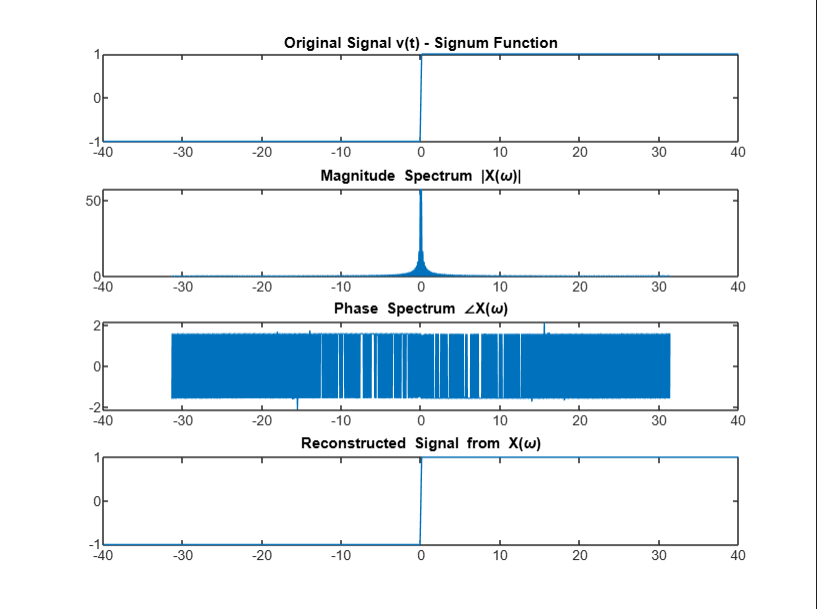
if abs(energy\_time\_domain - energy\_freq\_domain) < 1e-3

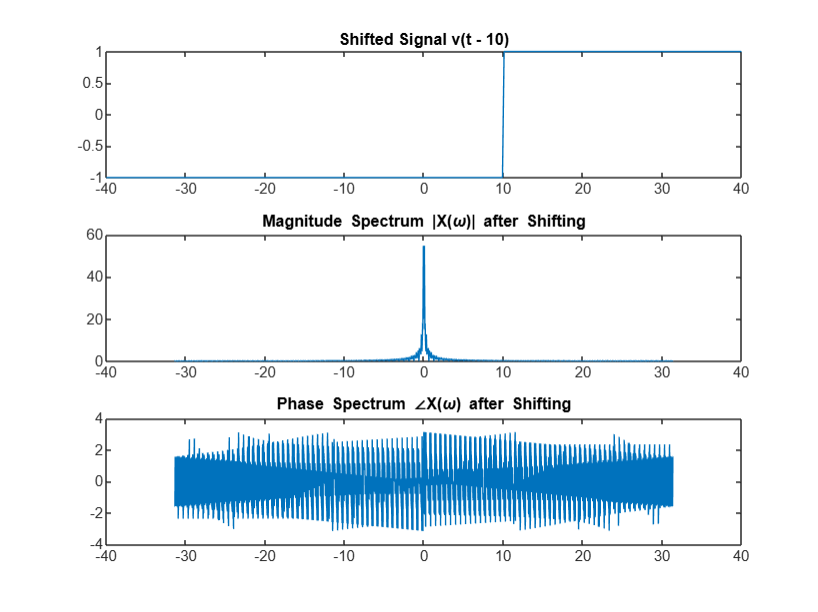
disp('Parseval''s theorem is verified: Energy in time and frequency domains are equal.');

else

disp('Parseval''s theorem is not verified: Energy in time and frequency domains differ.');

end





Laplace Transform :

clc;

clear all;

close all;

% Define parameters

step\_omega = 0.01 \* pi;

step\_size\_t = 0.1;

t = -40:step\_size\_t:40;

length\_t = length(t);

omegax = -(1 / step\_size\_t) \* pi : step\_omega : (1 / step\_size\_t) \* pi;

length\_omega = length(omegax);

sigmax = -5:0.01:5;

% Create mesh grid for plotting

[X, Y] = meshgrid(sigmax, omegax);

% Precompute exponential terms for Fourier transform

expo\_omega = zeros(length\_omega, length\_t);

for ii = 1:length\_omega

expo\_omega(ii, :) = exp(-1j \* omegax(ii) .\* t);

end

expo\_sigma = zeros(length(sigmax), length\_t);

for ii = 1:length(sigmax)

expo\_sigma(ii, :) = exp(-1 \* sigmax(ii) .\* t);

end

% Select Signal: Uncomment only one section at a time

% % Impulse Signal

% x\_t = zeros(1, length\_t);

% x\_t(t == 0) = 1 / step\_size\_t;

% % Unit Step

% x\_t = zeros(1, length\_t);

% x\_t(t >= 0) = 1;

% % Signum Function

% x\_t = zeros(1, length\_t);

% x\_t(t < 0) = -1;

% x\_t(t > 0) = 1;

% % Decaying Exponential

% x\_t = zeros(1, length\_t);

% x\_t(t >= 0) = 3 \* exp(-2 \* t(t >= 0));

% % Double-Sided Exponential

% x\_t = zeros(1, length\_t);

% x\_t(t < 0) = 3 \* exp(2 \* t(t < 0));

% x\_t(t >= 0) = 3 \* exp(-2 \* t(t >= 0));

% % Sinusoid

%x\_t = sin(2 \* pi \* (1 / (2 \* step\_size\_t)) \* 0.5 .\* t);

%% Cosine

%x\_t = cos(2 \* pi \* (1 / (2 \* step\_size\_t)) \* 0.5 .\* t);

%% Sinc Function

%x\_t = sinc(t);

%% Gate Function

%x\_t = (t >= -0.5) & (t <= 0.5);

% Plot the selected signal

% subplot(2, 1, 1);

% plot(t, x\_t);

% title('Selected Signal x(t)');

% xlabel('Time t');

% ylabel('Amplitude');

% Compute Laplace Transform with sigma and j\*omega

X\_s = zeros(length\_omega, length(sigmax));

for ii = 1:length(sigmax)

temp = x\_t.\* expo\_sigma(ii, :);

for jj = 1:length\_omega

temp1 = temp .\* expo\_omega(jj, :);

X\_s(jj, ii) = sum(temp1) \* step\_size\_t; % Approximate integral

end

end

% Plot Laplace Transform magnitude in dB

subplot(2, 1, 2);

surf(X, Y, 10 \* log10(abs(X\_s)), 'EdgeColor', 'none');

xlabel('Sigma');

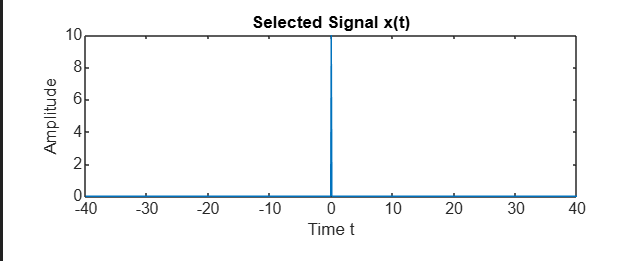
ylabel('j\omega');

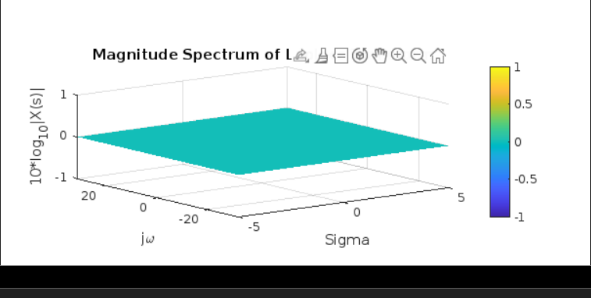
zlabel('10\*log\_{10}|X(s)|');

title('Magnitude Spectrum of Laplace Transform');

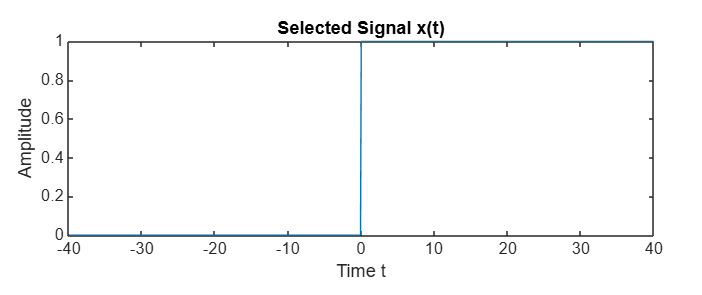
colorbar;

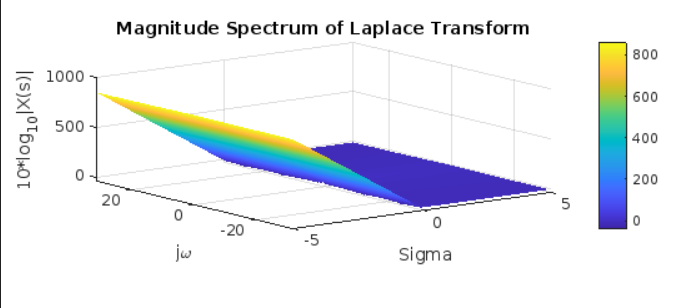
Impulse Function



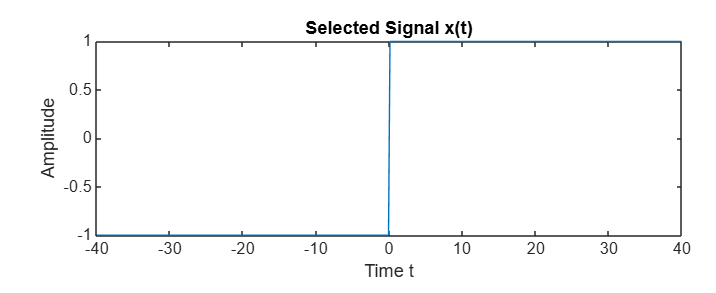


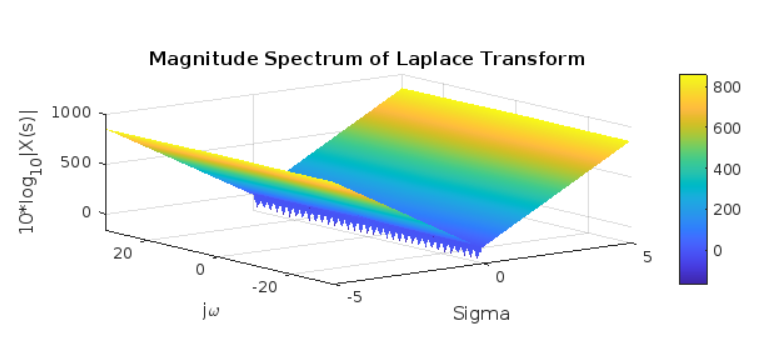
Unit Step Function



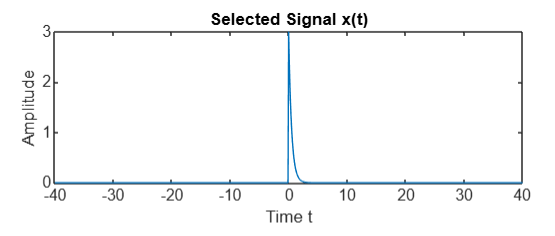


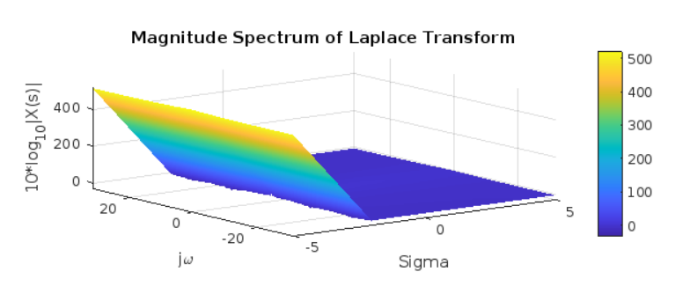
Signum Function



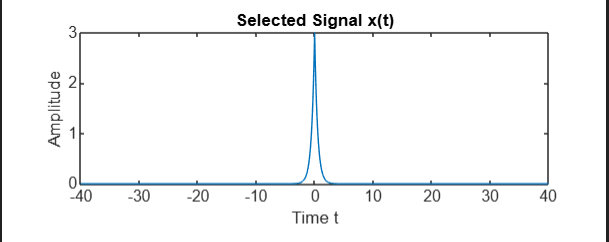


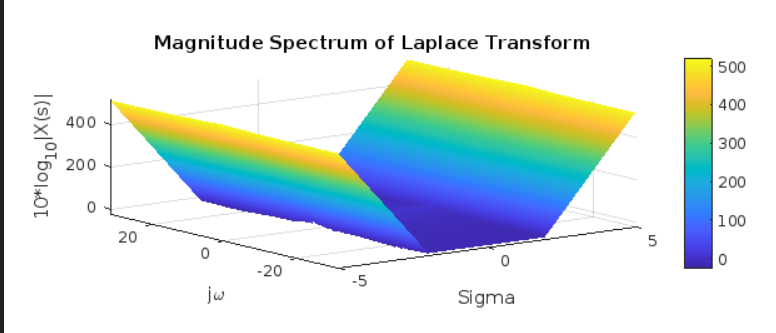
Exponential Decay Function



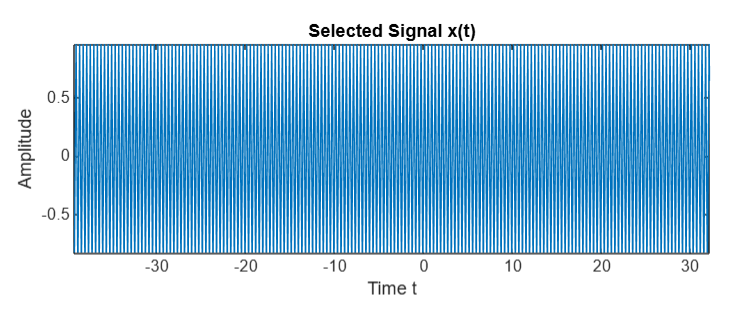


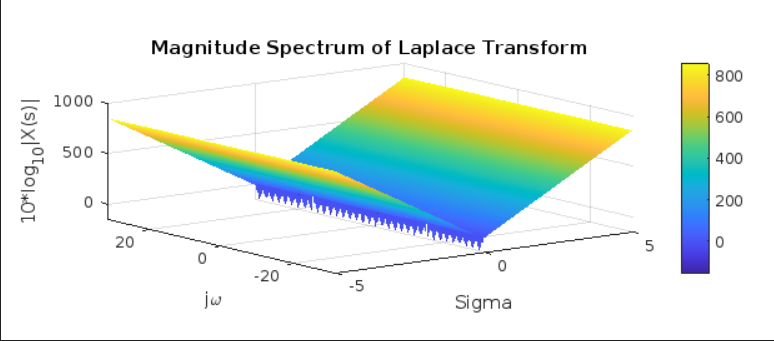
Double Exponential Decay Function



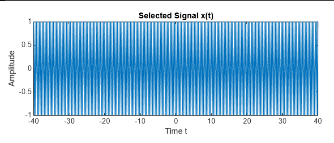


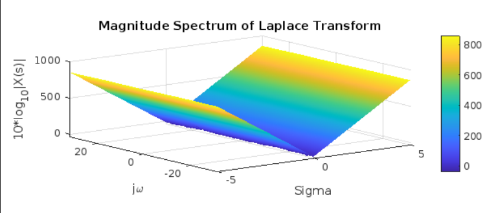
Sine Function



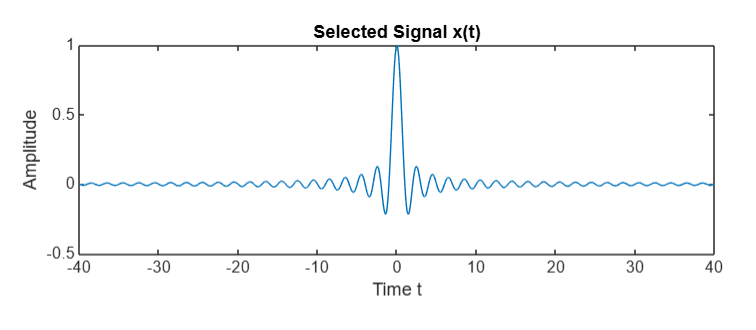


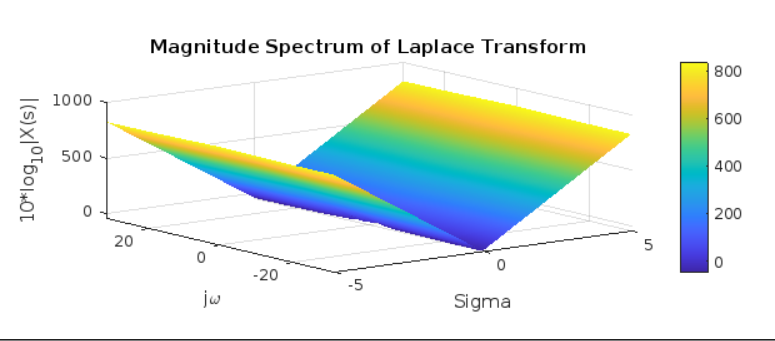
Cosine Function





Sinc Function





Gate Function

