Signals Final Project

By David Michael 9299 & Ziad AbdElhaleem 9634

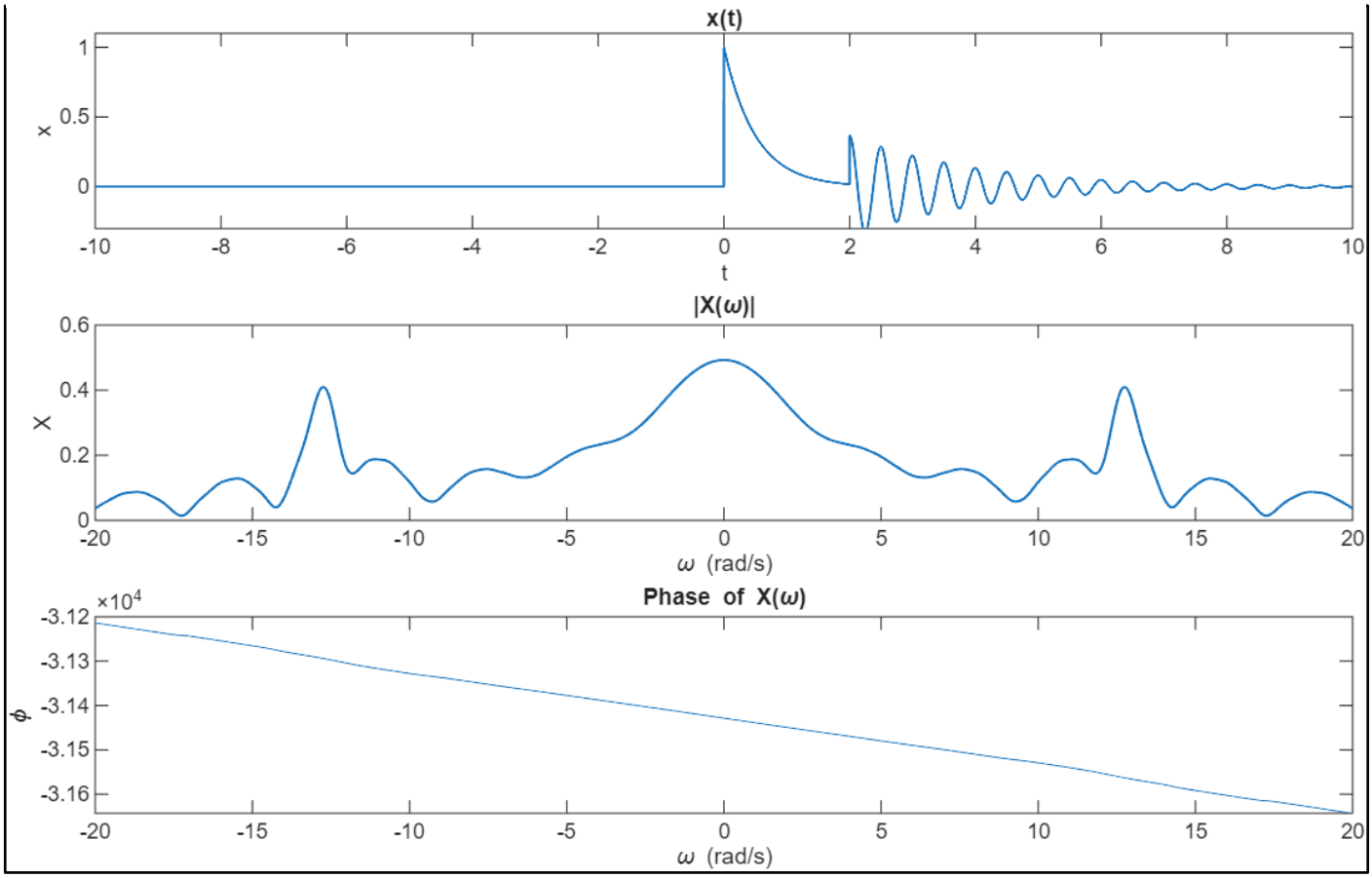
Part I

Question 1:

```
%% Question 1

% Parameters
dt = 1e-3;
t = -10:dt:10;              % time domain
fs = 1/dt;

% (a) time-domain signal
% f(t) = e^{-2t} u(t) t<2; cos(4πt) e^{-0.5t} t>=2;
f = exp(-2*t) .* double(t >= 0 & t < 2) + cos(4*pi*t) .* exp(-0.5*t) .* double(t>=2);

Nf = 2^18;
F = fft(f, Nf) * dt;    % continuous-FT approximation
F = fftshift(F);        % center graph

% Frequency domain
df = fs / Nf;
f_axis = (-Nf/2 : Nf/2 - 1) * df;   % Hz
w = 2*pi * f_axis;                  % rad/s
```

```
figure("Name", "Q1", "NumberTitle","off");
subplot(3,1,1);
plot(t,f, 'LineWidth', 1);
title("x(t)");
xlabel("t"); ylabel("x");
xlim([-10 10]);
ylim([-0.3 1.1]);

subplot(3,1,2);
plot(w,abs(F), 'LineWidth', 1);
title("|X(\omega)|");
xlabel("\omega (rad/s)"); ylabel("X");
xlim([-20 20]);

subplot(3,1,3);
plot(w, unwrap(angle(F)));
title('Phase of X(\omega)');
xlabel('\omega (rad/s)'); ylabel("\phi")
xlim([-20 20]);
```

Question 1



Topic : Date : / /

Q1)

a) $x(t) = e^{-2t}[u(t) - u(t-2)] + \cos(4\pi t)\, e^{-0.5t}\, u(t-2)$

$t < 0$ $u(t) = 0 \longrightarrow x(t) = 0$

$0 < t < 2$ $u(t) = 1,\ u(t-2) = 0 \longrightarrow x(t) = e^{-2t}$

$2 < t$ $u(t) - u(t-2) = 1 - 1 = 0 \longrightarrow x(t) = \cos(4\pi t)\, e^{-0.5t}$

$$x(t) = e^{-2t}(u(t) - u(t-2)) + e^{-0.5t}\cos(4\pi t)\, u(t-2)$$

b) $x(t) = x_1(t) + x_2(t)$

$x_1(t) = e^{-2t}(u(t) - u(t-2))$ $\quad x_2(t) = \cos(4\pi t)\, e^{-0.5t}\, u(t-2)$

x_1 :- $e^{-2t}u(t) \xrightarrow{F} \frac{1}{2+j\omega}$

$e^{-2t}u(t-2) = e^{-4}e^{-2(t-2)}u(t-2)$

$e^{-2t}u(t-2) \xrightarrow{F} e^{-j2\omega}e^{-4}\frac{1}{2+j\omega}$

$$X_1(\omega) = \frac{1 - e^{-4}e^{-j2\omega}}{2+j\omega}$$

x_2 :- let $T = t-2$ $\quad t = T+2$

$x_2(t) = e^{-0.5(T+2)}\cos(4\pi(T+2))\, u(T)$

$\cos(4\pi T + 8\pi) = \cos(4\pi T)$

$x_2(t) = e^{-1}e^{-0.5T}\cos(4\pi T)\, u(T)$ with $e^{-j2\omega}$ shift

$X_2(\omega) = e^{-j2\omega}e^{-1}\, F\{e^{-0.5t}\cos(4\pi t)\, u(t)\}$

$\cos(4\pi t) = \frac{1}{2}(e^{j4\pi t} + e^{-j4\pi t})$

Topic : Date : / /

$$e^{-\alpha t} e^{\pm j\Omega t} u(t) \xrightarrow{F} \frac{1}{\alpha + j(\omega \mp \Omega)}$$

$$e^{-0.5t}\cos(4\pi t)\,u(t) \xrightarrow{F} \frac{1}{2}\left(\frac{1}{0.5+j(\omega+4\pi)} + \frac{1}{0.5+j(\omega+4\pi)}\right)$$

$$X_2(\omega) = e^{-1} e^{-j2\omega} \cdot \frac{1}{2}\left(\frac{1}{0.5+j(\omega+4\pi)} + \frac{1}{0.5+j(\omega-4\pi)}\right)$$

$$X(\omega) = \frac{1-e^{-4}e^{-j2\omega}}{2+j\omega} + e^{-1}e^{-j2\omega}\cdot\frac{1}{2}\left(\frac{1}{0.5+j(\omega+4\pi)} + \frac{1}{0.5+j(\omega-4\pi)}\right)$$

c)

$$E = \int_{-\infty}^{\infty} |x(t)|^2 dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} |X(\omega)|^2 d\omega$$

$$\frac{1}{2\pi}\int_{-\infty}^{\infty} |X(\omega)|^2 d\omega = 0.95E$$

The MATLAB plot clearly suggest that almost all significant energy is within approximately $|\omega| \leq 15$

d) * Time shift (start at $t=2$) introduces a multiplicative factor $e^{-j2\omega}$ in the FT-phase

* Truncation (the $u(t)-u(t-2)$ factor) multiplies the original $e^{-2t}u(t)$ by finite window, producing the factor $(1-e^{-4}e^{-j2\omega})$ ifrequency causes spectral ripples

* Damping ($e^{-0.5t}$) broadens the spectral lobe and controls lobe width; larger damping → wider lobe

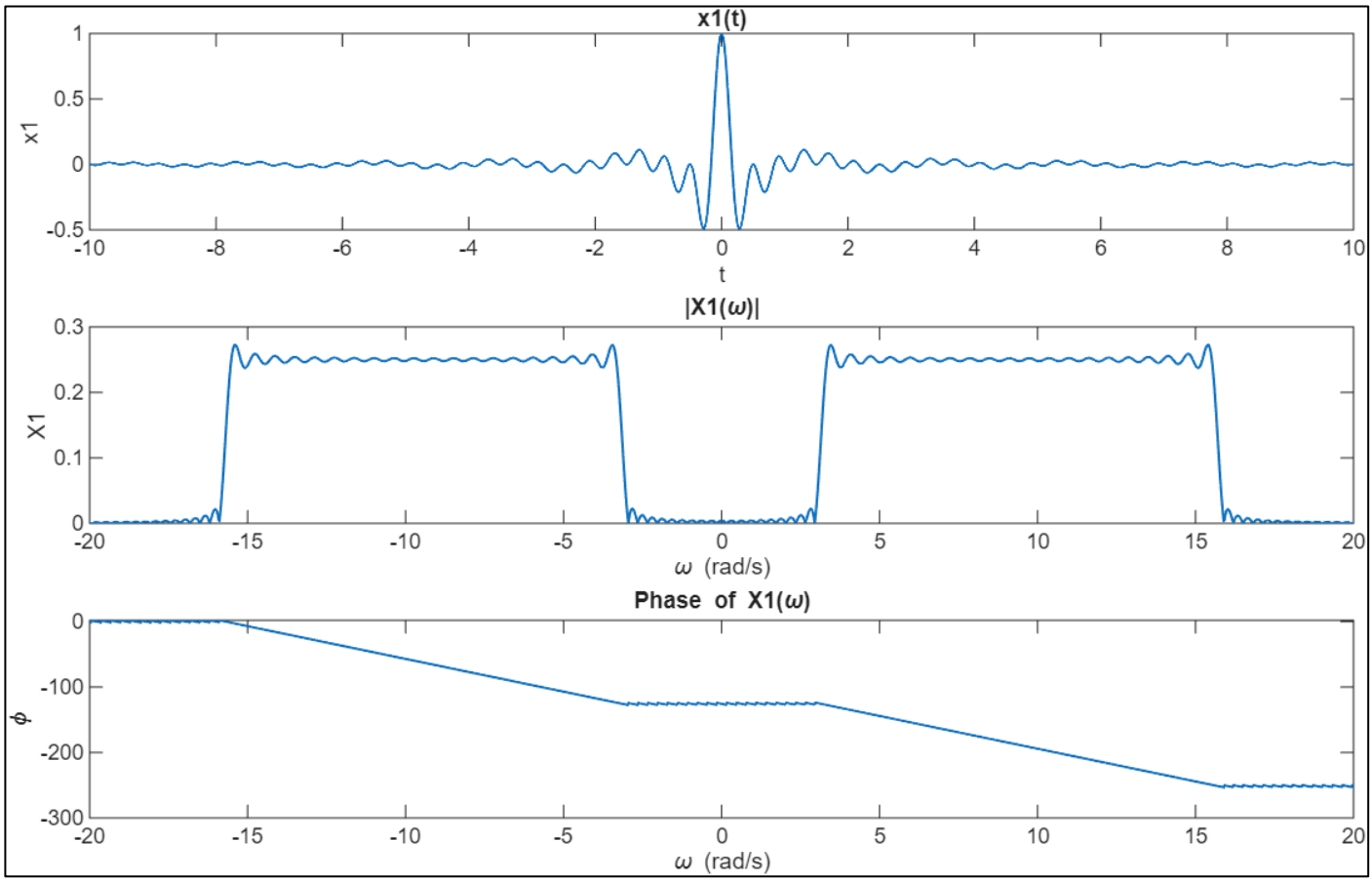
CamScanner

Question 2:

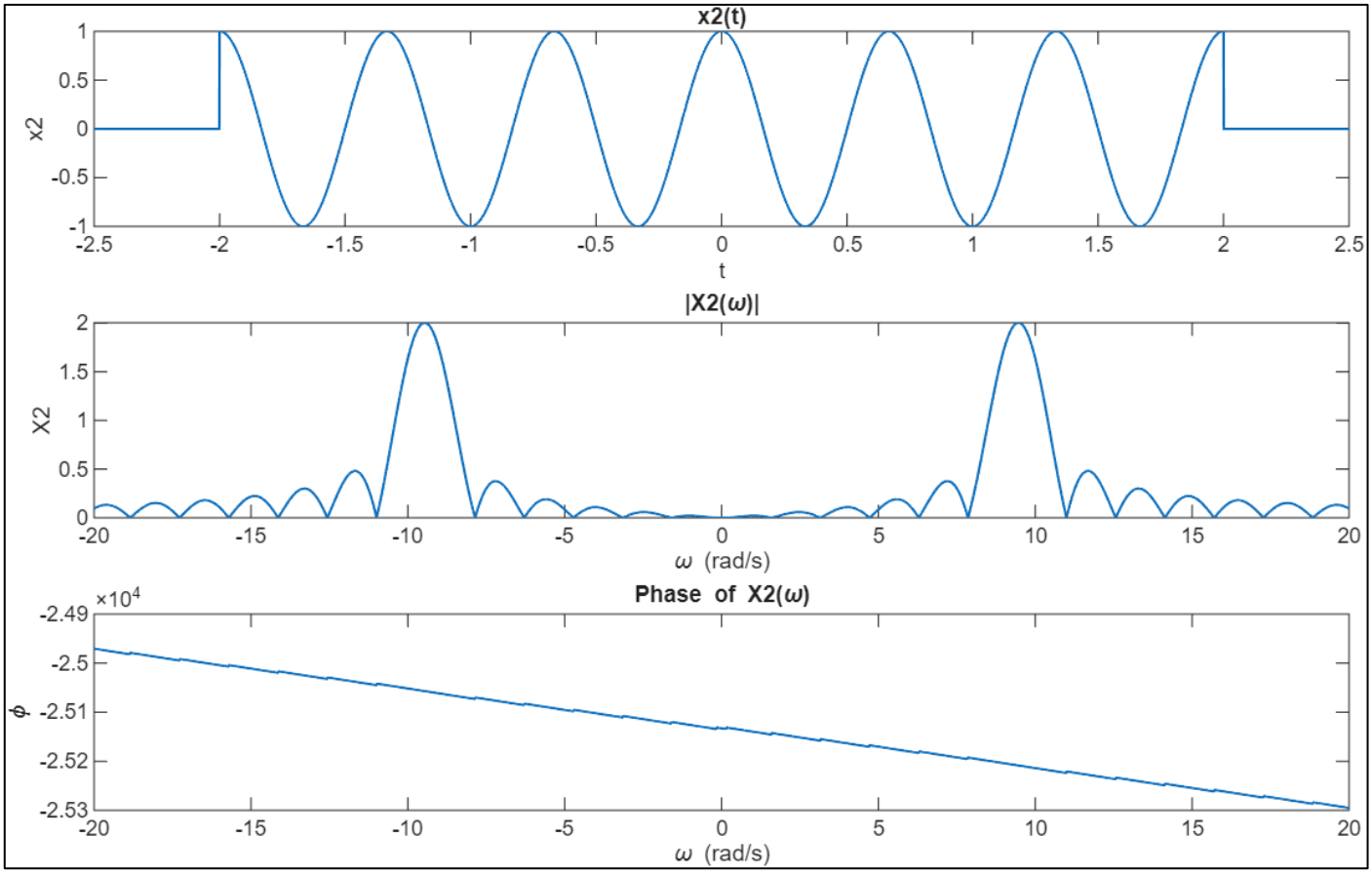
```
%% Question 2

% Parameters
dt = 1e-3;
t = -10:dt:10;              % time domain
fs = 1/dt;

% (a) time-domain signal
% x1(t) = sin2t * cos(3πt)
% x2(t) = rect(t/4) * cos(3πt)

x1 = sinc(2*t) .* cos(3*pi* t);

rect = double(t >= -2) .* double(t <= 2); % -2 -> 2
x2 = rect .* cos(3*pi* t);


Nf = 2^18;
X1 = fft(x1, Nf) * dt; X2 = fft(x2, Nf) * dt;    % continuous-FT approximation
X1 = fftshift(X1); X2 = fftshift(X2);        % center graph

% Frequency domain
df = fs / Nf;
f_axis = (-Nf/2 : Nf/2 - 1) * df;    % Hz
w = 2*pi * f_axis;                   % rad/s
```

```
figure("Name", "Q2 x1", "NumberTitle","off");
subplot(3,1,1);
plot(t,x1, 'LineWidth', 1);
title("x1(t)");
xlabel("t"); ylabel("x1");

subplot(3,1,2);
plot(w,abs(X1), 'LineWidth', 1);
title("|X1(\omega)|");
xlabel("\omega (rad/s)"); ylabel("X1");
xlim([-20 20]);

subplot(3,1,3);
plot(w, unwrap(angle(X1)), 'LineWidth', 1);
title('Phase of X1(\omega)');
xlabel('\omega (rad/s)'); ylabel("\phi")
xlim([-20 20]);
```

```
figure("Name", "Q2 x2", "NumberTitle","off");

subplot(3,1,1)
plot(t,x2, "LineWidth",1);
title("x2(t)");
xlabel("t"); ylabel("x2");
xlim([-2.5 2.5]);

subplot(3,1,2);
plot(w,abs(X2), 'LineWidth', 1);
title("|X2(\omega)|");
xlabel("\omega (rad/s)"); ylabel("X2");
xlim([-20 20]);

subplot(3,1,3);
plot(w, unwrap(angle(X2)), 'LineWidth', 1);
title('Phase of X2(\omega)');
xlabel('\omega (rad/s)'); ylabel("\phi")
xlim([-20 20]);
```


x1(t)
x1
t
|X1(ω)|
X1
ω (rad/s)
Phase of X1(ω)
ϕ
ω (rad/s)

Question 2


x2(t)
x2
t
|X2(ω)|
X2
ω (rad/s)
Phase of X2(ω)
×10^4
ϕ
ω (rad/s)

Topic : Date : / /

Q2) b)

$$x_1(t) = \text{sinc}(2t)\cos(3\pi t)$$

1. $\text{sinc}(at) \xrightarrow{F} \frac{1}{|a|}\text{rect}\left(\frac{w}{2\pi a}\right)$

$$\text{sinc}(2t) \xrightarrow{F} \frac{1}{2}\text{rect}\left(\frac{w}{4\pi}\right)$$

2. $s(t)\cos(\Omega_0 t) \xrightarrow{F} \frac{1}{2}\left[S(w-\Omega_0)+S(w+\Omega_0)\right]$

$\Omega_0 = 3\pi \quad s(w) = \frac{1}{2}\text{rect}\left(\frac{w}{4\pi}\right)$

$$X_1(w) = \frac{1}{4}\left[\text{rect}\left(\frac{w-3\pi}{4\pi}\right) + \text{rect}\left(\frac{w+3\pi}{4\pi}\right)\right]$$

$$x_2(t) = \text{rect}\left(\frac{t}{4}\right) * \cos(3\pi t)$$

1. $\text{rect}\left(\frac{t}{4}\right) \xrightarrow{F} \int_{-2}^{2} e^{-jwt}\,dt = \frac{2\sin(2w)}{w}$

$$\cos(3\pi t) \xrightarrow{F} \pi\left[\delta(w-3\pi)+\delta(w+3\pi)\right]$$

$$f(t)g(t) \xrightarrow{F} \frac{1}{2\pi} F(w) * G(w)$$

2. $X_2(w) = \frac{1}{2\pi} R(w) * \left[\pi\left(\delta(w-3\pi)+\delta(w+3\pi)\right)\right]$

$$R(w) * \delta(w-w_0) = R(w-w_0)$$

$$X_2 = \frac{1}{2}\left[R(w-3\pi)+R(w+3\pi)\right]$$

$$X_2(w) = \frac{\sin(2(w-3\pi))}{w-3\pi} + \frac{\sin(2(w+3\pi))}{w+3\pi}$$

KMS

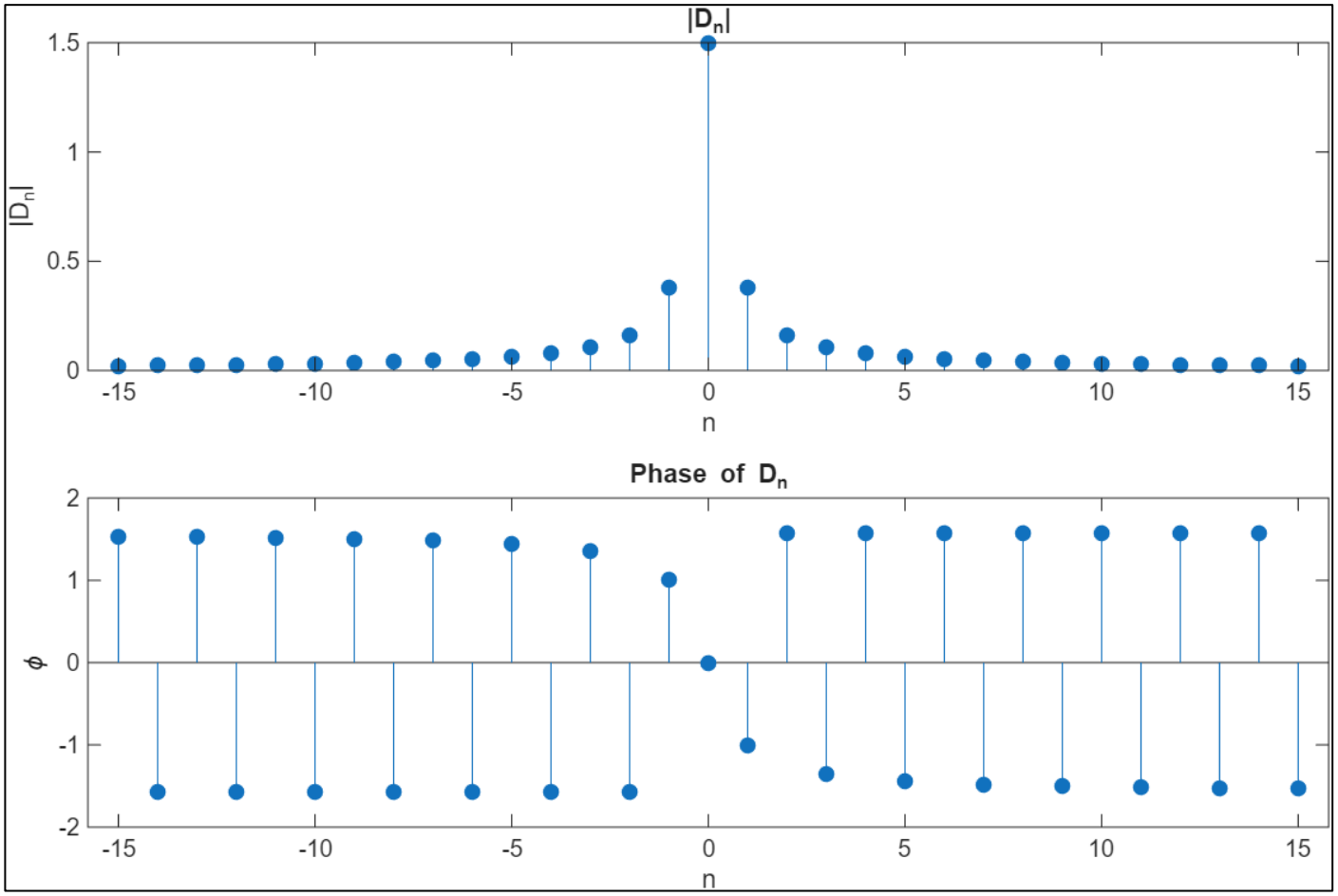
Topic : Date : / /

c) - $X_1(t)$ is more band width-efficient because its spectrum is strictly band-limited and confined to finite intervals

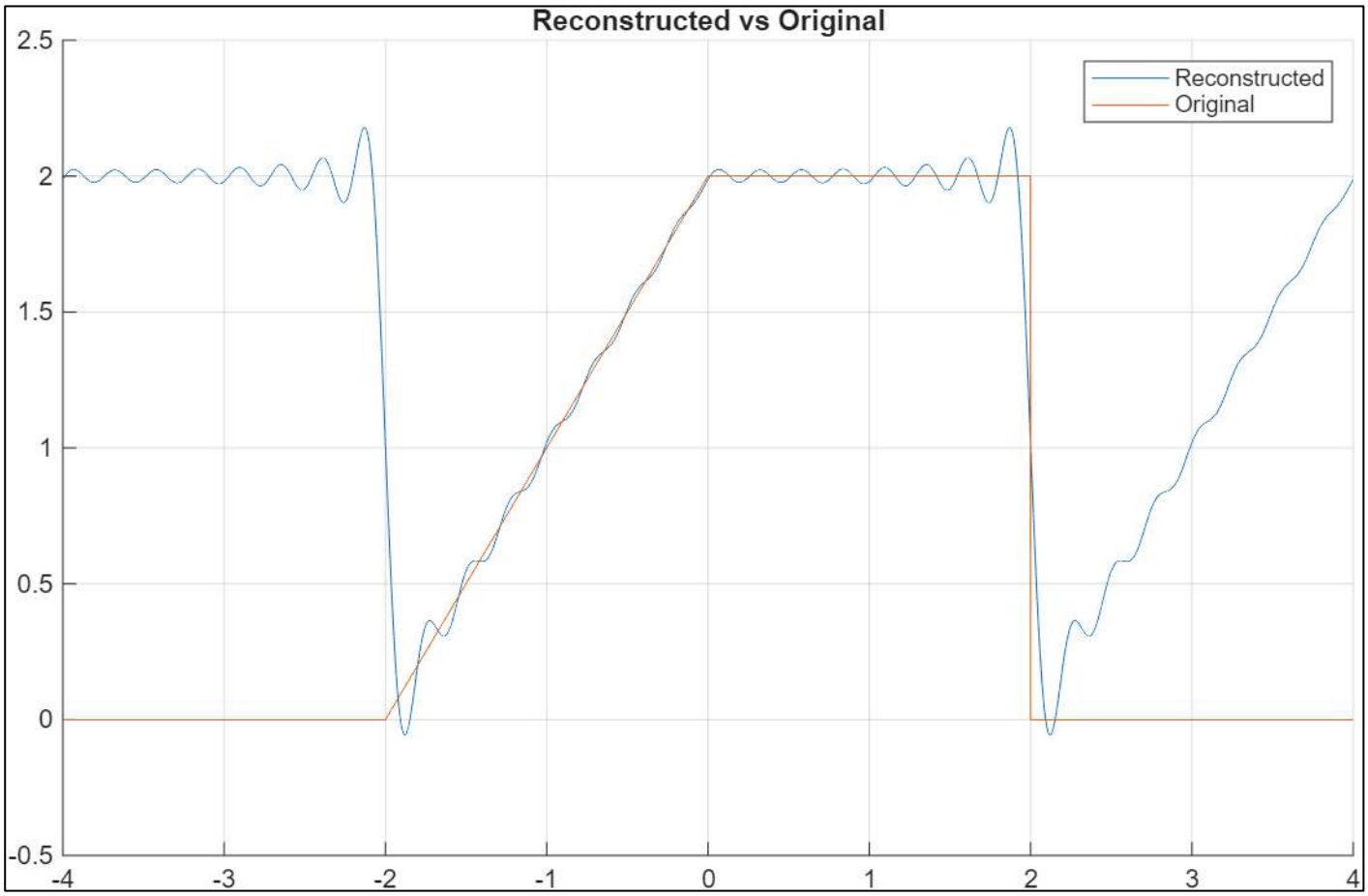
- $X_2(t)$ spreads energy over a wider frequency range due to the sinc side-lobes caused by the rectangular window

Question 3:

```
%% Question 3

n = -15:15;
Dn = ((-1).^(n+1))./(1j*n*pi) - ((-1).^n - 1)./(n * pi).^2;
Dn(n==0) = 3/2;

figure("Name", "Q3 Magnitude and Phase", "NumberTitle","off");
subplot(2,1,1);
stem(n, abs(Dn), 'filled');
title('|D_n|');
xlabel('n'); ylabel('|D_n|');

subplot(2,1,2);
stem(n, unwrap(angle(Dn)), 'filled');
title('Phase of D_n');
xlabel('n'); ylabel('\phi');

dt = 1e-3;
t = -4:dt:4;
% x(t) = (t+2) -2<t<0; 2 0<t<2;
x = (t+2).*double(-2<=t & t<0) + 2*double(0<=t & t<2);
x_recon = zeros(size(t));

for k=-15:15
    if k==0
        Dk=3/2;
    else
        Dk = ((-1)^(k+1))/(1j*k*pi) - ((-1)^k - 1)/(k^2*pi^2);
    end
    x_recon = x_recon + Dk*exp(1j*k*pi/2*t);
end
figure("Name", "Q3 Reconstructed vs Original", "NumberTitle","off");
hold on; grid on;
plot(t, real(x_recon)); plot(t, x);
title('Reconstructed vs Original'); legend('Reconstructed','Original');
```


|Dn|
1.5
1
0.5
0
|Dn|
-15
-10
-5
0
5
10
15
n
Phase of Dn
2
1
0
-1
-2
ϕ
-15
-10
-5
0
5
10
15
n

Question 3


Reconstructed vs Original
Reconstructed
Original
2.5
2
1.5
1
0.5
0
-0.5
-4
-3
-2
-1
0
1
2
3
4

Topic : Date : / /

Q3) a) $$x(t) = \begin{cases} t+2 & -2 \le t < 0 \\ 2 & 0 \le t < 2 \end{cases}$$

$\omega_0 = \frac{2\pi}{T} = \frac{\pi}{2}$ $\quad D_n = \frac{1}{4}\int_{-2}^{2} x(t)\, e^{-jn\omega_0 t}\, dt$

Let $\alpha = n\omega_0 = \frac{n\pi}{2}$ we evaluate for $n=0$, $n\neq 0$

$$D_0 = \frac{1}{4}\int_{-2}^{2} x(t)\, dt = \frac{1}{4}\left(\int_{-2}^{0}(t+2)\, dt + \int_{0}^{2} 2\, dt\right)$$

$$\int_{-2}^{0}(t+2)\, dt = \left[\frac{t^2}{2}+2t\right]_{-2}^{0} = 0-(-2) = 2$$

$$\int_{0}^{2} 2\, dt = 4 \qquad 2+4=6 \qquad D_0 = \frac{6}{4} = \frac{3}{2}$$

$$D_n = \frac{1}{4}\left(\underbrace{\int_{-2}^{0}(t+2)e^{-j\alpha t}\, dt}_{I_1} + \underbrace{\int_{0}^{2} 2e^{-j\alpha t}\, dt}_{I_2}\right)$$

$$I_2 = 2\int_{0}^{2} e^{-j\alpha t}\, dt = 2\left[\frac{e^{j\alpha t}}{-j\alpha}\right]_{0}^{2} = \frac{2(1-e^{-j2\alpha})}{j\alpha}$$

Let $s = t+2$ $\qquad t = s-2$

$$I_1 = \int_{0}^{2} s\, e^{-j\alpha(s-2)}\, ds = e^{j2\alpha}\int_{0}^{2} s\, e^{-j\alpha s}\, ds$$

$B = \int_0^2 s e^{-j\alpha s} ds$ Let $u = s$, $dv = e^{-j\alpha s} ds$

$du = ds$ $\qquad v = \frac{e^{-j\alpha s}}{-j\alpha}$

$$B = \frac{s e^{-j\alpha s}}{-j\alpha}\Big|_0^2 - \int_0^2 \frac{e^{-j\alpha s}}{-j\alpha}\, ds = -\frac{2e^{-j2\alpha}}{j\alpha} + \frac{1-e^{-j2\alpha}}{(j\alpha)^2}$$

KMS

CamScanner

Topic : Date : / /

$$I_1 = e^{j2\alpha}\beta = e^{j2\alpha}\left(-\frac{2e^{-j2\alpha}}{j\alpha} + \frac{1-e^{-j2\alpha}}{(j\alpha)^2}\right)$$

$$I_1 = -\frac{2}{j\alpha} + \frac{e^{j2\alpha}-1}{\alpha^2}$$

$$I_1 + I_2 = -\frac{2}{j\alpha} - \frac{e^{j2\alpha}-1}{\alpha^2} + \frac{2(1-e^{-j2\alpha})}{j\alpha}$$

$$\frac{-2+2-2e^{-j2\alpha}}{j\alpha} = -\frac{2e^{-j2\alpha}}{j\alpha}$$

$$I_1 + I_2 = -\frac{2e^{-j2\alpha}}{j\alpha} - \frac{e^{j2\alpha}-1}{\alpha^2}$$

$$D_n = \frac{1}{4}(I_1 + I_2) = -\frac{e^{-j2\alpha}}{2j\alpha} - \frac{e^{j2\alpha}-1}{4\alpha^2}$$

$$\alpha = \frac{n\pi}{2} \longrightarrow e^{j2\alpha} = e^{jn\pi} = (-1)^n, \quad e^{-j2\alpha} = (-1)^n$$

$$D_n = \frac{(-1)^{n+1}}{jn\pi} - \frac{(-1)^n - 1}{n^2\pi^2} \qquad n \neq 0$$

$$D_n = \begin{cases} j/n\pi & n \text{ even} \quad (n \neq 0) \\ 1/jn\pi + 2/n^2\pi^2 & n \text{ odd} \end{cases}$$

$$D_0 = \frac{3}{4}$$

Topic : Date : / /

b) 1- Magnitude plot:

- The plot shows the values of $|D_n|$ for different harmonic indices n
- The largest value is at $n=0$ and the magnitudes get smaller as $|n|$ increases

2- Phase plots

- The plot shows the angle of each coefficient D_n.
- Because the signal is real, the phase is symmetric: $\angle D_{-n} = -\angle D_n$

CamScanner

Part II

```
%% User Input
disp("Welcome to Final Project 9229_9634");
fs = input("Enter sampling frequency (default 1000): ");
if isempty(fs)
    fs = 1000;
end
t1 = input("Enter start of time axis (default -10): ");
if isempty(t1)
    t1 = -10;
end
t2 = input("Enter end of time axis (default 10): ");
if isempty(t2)
    t2 = 10;
end

nbp = input("Enter number of breakpoints (default 3): ");
if isempty(nbp)
    nbp = 3;
end

% default breakpoints
bp_default = [-2, 0, 3];

bp = zeros(1, nbp);
for i = 1:nbp
    if i <= length(bp_default)
        def = bp_default(i);
    else
        def = 0;    % for extra breakpoints
    end

    in = input(sprintf("Enter breakpoint %d (default %g): ", i, def));
    if isempty(in)
        bp(i) = def;
    else
        bp(i) = in;
    end
end


%% Region Splitting
bp = sort(bp);
t = t1 : 1/fs : t2;

all_points = [t1, bp, t2];
regions = {};

for k = 1:length(all_points)-1
    if k < length(all_points)-1
        t_reg = t(t >= all_points(k) & t < all_points(k+1));
    else
        t_reg = t(t >= all_points(k) & t <= all_points(k+1));
    end
    regions{k} = t_reg;
end

fprintf("\nRegions created successfully: %d regions\n\n", numel(regions));
input("Press Enter to continue\n");
```

```
%% Main Menu

menu = sprintf('%s', ...
"----------- MENU -----------" + newline + ...
"1) DC" + newline + ...
"2) Ramp" + newline + ...
"3) Polynomial" + newline + ...
"4) Exponential" + newline + ...
"5) Sinusoidal" + newline + ...
"6) Gaussian pulse" + newline + ...
"7) Sawtooth wave" + newline + ...
"----------------------------" + newline);

clc;

%% Generate the signal based on regions

x = [];
t_full = [];

for k = 1:length(regions)

    fprintf("\nChoose signal type for Region %d:\n", k);
    disp(menu);

    choice = input("Enter choice: ");
    while isempty(choice) || choice < 1 || choice > 7
        choice = input("\nError! Choose valid menu option: ");
    end

    fprintf("\nEnter parameters for region %d:\n", k);
    yk = generate_signal(choice, regions{k});

    x = [x yk];
    t_full = [t_full regions{k}];
end

fprintf("\nSignal generated successfully.\n\n");
input("Press Enter to continue\n");

%% Operation Menu

opMenu = sprintf('%s', ...
"----------- MENU -----------" + newline + ...
"1) Amplitude scaling" + newline + ...
"2) Time reversal" + newline + ...
"3) Time shift" + newline + ...
"4) Expansion"+ newline + ...
"5) Compression"+ newline + ...
"6) Addition of random noise"+ newline + ...
"7) Smoothing using moving average" + newline + ...
"8) None" + newline + ...
"----------------------------" + newline);

clc; disp(opMenu);
op_choice = input("Enter choice: ");
i = 0; MAX_ATTEMPTS = 3;
while (isempty(op_choice) || op_choice < 1 || op_choice > 8) && i < MAX_ATTEMPTS
    op_choice = input("Error! Choose valid menu option: ");
    i = i+1;
end
if i == MAX_ATTEMPTS
disp("Too many attempts, try again later.");
end

% Apply operation

[t_new, x_new] = apply_operation(op_choice, t_full, x);

fprintf("\nOperation applied successfully.\n\n");
```

```
%% Plot and Save
% Make sure vectors have the same length
min_len = min(length(t_new), length(x_new));
t_new = t_new(1:min_len);
x_new = x_new(1:min_len);

figure;
plot(t_new, x_new, 'LineWidth', 1);
xlabel("Time");
ylabel("x(t)");
title("Generated Signal");
grid on;

save_figure();

fprintf("Figure saved successfully!\n");

function save_figure()
    fig = gcf;

    num  = fig.Number;
    fname = sprintf('Figure%d.png', num);

    % Force light-mode
    set(fig, 'Color', 'white');
    ax = findall(fig, 'Type', 'axes');
    set(ax, 'Color','white', 'XColor','black','YColor','black');

    % Save PNG
    exportgraphics(fig, fname, 'Resolution', 300);
end

%% Generate Signal Function
function y = generate_signal(choice, t)

    switch choice
        case 1
            A = input("Enter DC amplitude (default 1): "); if isempty(A), A=1; end
            y = A * ones(size(t));

        case 2
            m = input("Enter slope (default 1): "); if isempty(m), m=1; end
            c = input("Enter intercept (default 0): "); if isempty(c), c=0; end
            y = m*t + c;

        case 3
            coeffs = input("Enter polynomial coefficients [a b c] (default [1 0 0]): ");
            if isempty(coeffs), coeffs = [1 0 0]; end
            y = polyval(coeffs, t);

        case 4
            A = input("Enter amplitude (default 1): "); if isempty(A), A=1; end
            a = input("Enter exponent (default -1): "); if isempty(a), a=-1; end
            y = A * exp(a*t);

        case 5
            A = input("Enter amplitude (default 1): "); if isempty(A), A=1; end
            f = input("Enter frequency (default 1): "); if isempty(f), f=1; end
            y = A * sin(f*t);

        case 6
            A = input("Enter amplitude (default 1): "); if isempty(A), A=1; end
            mu = input("Enter mean (default 0): "); if isempty(mu), mu=0; end
            s = input("Enter sigma (default 1): "); if isempty(s), s=1; end
            y = A * exp(-((t-mu).^2)/(2*s^2));

        case 7
            A = input("Enter amplitude (default 1): "); if isempty(A), A=1; end
            f = input("Enter frequency (default 1): "); if isempty(f), f=1; end
            y = A * (2*((t*f) - floor(0.5 + t*f)));
    end
end
```

```m
%% Apply Operation Function
function [t_out, y_out] = apply_operation(op_choice, t, y)

    switch op_choice

        case 1   % Amplitude scaling
            s = input("Enter scaling factor (default 1): ");
            if isempty(s), s = 1; end
            y_out = s * y;
            t_out = t;

        case 2   % Time reversal
            % no parameter needed
            t_out = -fliplr(t);
            y_out = fliplr(y);

        case 3   % Time shift
            sh = input("Enter shift amount (default 0): ");
            if isempty(sh), sh = 0; end
            t_out = t + sh;
            y_out = y;

        case 4   % Expansion
            a = input("Enter expansion factor (default 1): ");
            if isempty(a), a = 1; end
            t_out = t * a;
            y_out = y;

        case 5   % Compression
            a = input("Enter compression factor (default 1): ");
            if isempty(a), a = 1; end
            t_out = t / a;
            y_out = y;

        case 6   % Add random noise
            snr = input("Enter SNR (dB, default 30): ");
            if isempty(snr), snr = 30; end
            y_out = awgn(y, snr, 'measured');
            t_out = t;

        case 7   % Moving-average smoothing
            w = input("Enter window length (default 5): ");
            if isempty(w), w = 5; end
            y_out = movmean(y, w);
            t_out = t;

        case 8   % None
            y_out = y;
            t_out = t;

        otherwise
            warning("Invalid operation code. No operation applied.");
            y_out = y;
            t_out = t;
    end
end
```

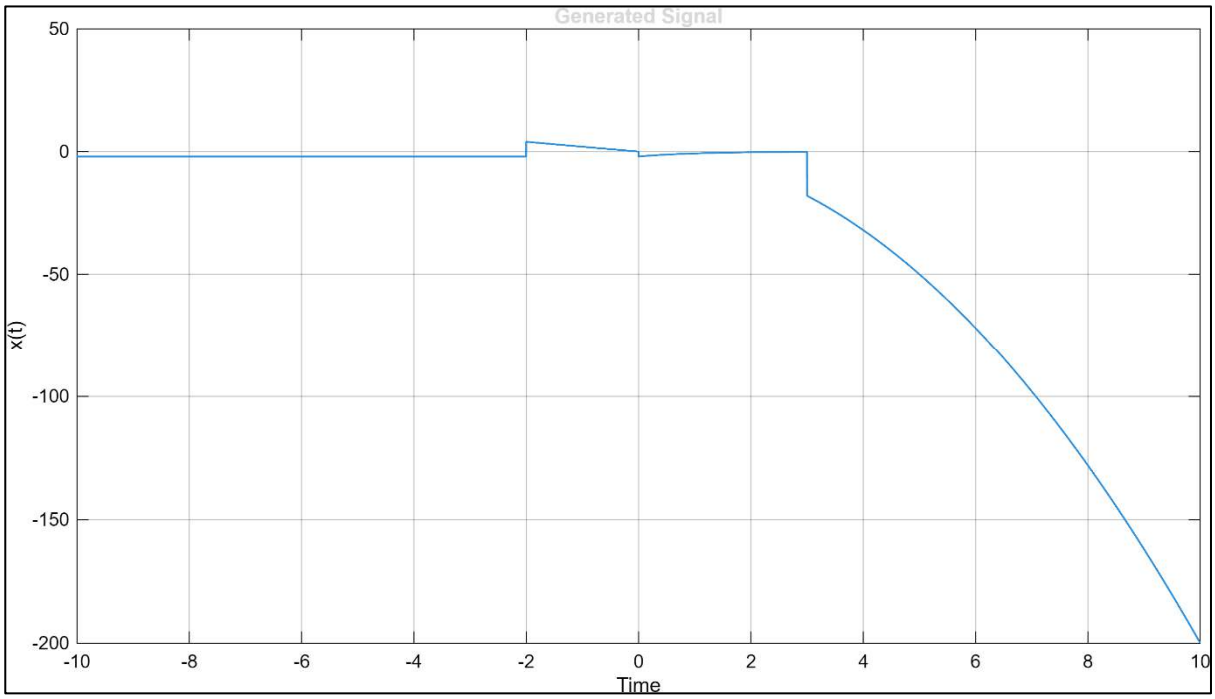
```
%% AUTO MODE (Generate 10 Random Signals)
fprintf("\nStarting AUTO MODE: Generating 10 random signals...\n");

for k = 1:10

    % Random FS and time axis
    fs_auto = randi([500 2000]);
    t1_auto = -5;
    t2_auto = 5;
    t_auto = t1_auto : 1/fs_auto : t2_auto;

    % Random number of breakpoints (0 to 4)
    nbp_auto = randi([0 4]);

    % Generate random breakpoints
    if nbp_auto > 0
        bp_auto = sort( (t1_auto + (t2_auto - t1_auto) .* rand(1, nbp_auto)) );
    else
        bp_auto = [];
    end

    % Build regions
    all_points_auto = [t1_auto, bp_auto, t2_auto];
    regions_auto = {};

    for j = 1:length(all_points_auto)-1
        t_reg = t_auto(t_auto >= all_points_auto(j) & t_auto <= all_points_auto(j+1));
        regions_auto{j} = t_reg;
    end

    % Generate random type (1 to 7)
    choice_auto = randi([1 7]);

    % Build the signal
    x_auto = [];
    for j = 1:length(regions_auto)
        x_auto = [x_auto generate_signal_auto(choice_auto, regions_auto{j})];
    end

    % Save the figure
    figure;
    plot(t_auto, x_auto, 'LineWidth', 1);
    title(sprintf("AUTO SIGNAL %d", k));
    xlabel("t");
    ylabel("x(t)");
    grid on;

    save_figure();

    fprintf("Saved auto signal %d\n", k);
end

fprintf("\nAUTO MODE completed successfully!\n\n");

%% AUTO Signal Generator Function
function y = generate_signal_auto(choice, t)

    switch choice
        case 1   % DC
            A = (rand()*4) - 2;   % random amplitude between -2 and 2
            y = A * ones(size(t));

        case 2   % Ramp
            m = (rand()*4) - 2;   % random slope
            c = (rand()*4) - 2;   % random intercept
            y = m*t + c;

        case 3   % Polynomial
            % Random polynomial ax^2 + bx + c
            coeffs = [(rand()*2-1), (rand()*4-2), (rand()*4-2)];
            y = polyval(coeffs, t);

        case 4   % Exponential
            A = rand()*2;        % 0 to 2
            a = (rand()*4) - 2;  % -2 to 2
            y = A * exp(a*t);

        case 5   % Sinusoidal
            A = rand()*2;        % Amplitude 0–2
            f = rand()*5 + 0.5;  % Frequency 0.5–5.5 Hz
            y = A * sin(2*pi*f*t);

        case 6   % Gaussian Pulse
            A = rand()*2;            % amplitude 0–2
            mu = rand()*2 - 1;       % mean between -1 and 1
            s = rand()*0.9 + 0.1;    % sigma between 0.1 and 1
            y = A * exp(-((t-mu).^2)/(2*s^2));

        case 7   % Sawtooth
            A = rand()*2;
            f = rand()*5 + 0.5;
            y = A * (2*((t*f) - floor(0.5 + t*f)));
    end
end
```

Trial Run:

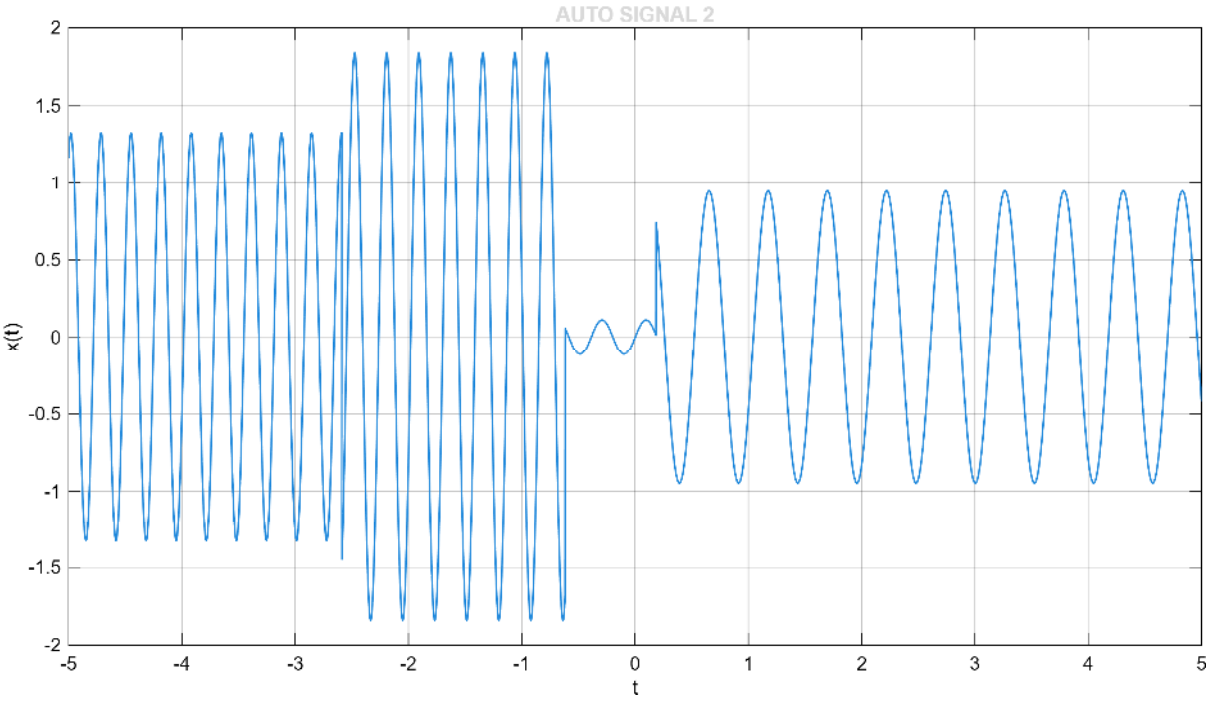
All default values, 4 regions (DC, ramp, Exponential, Polynomial)

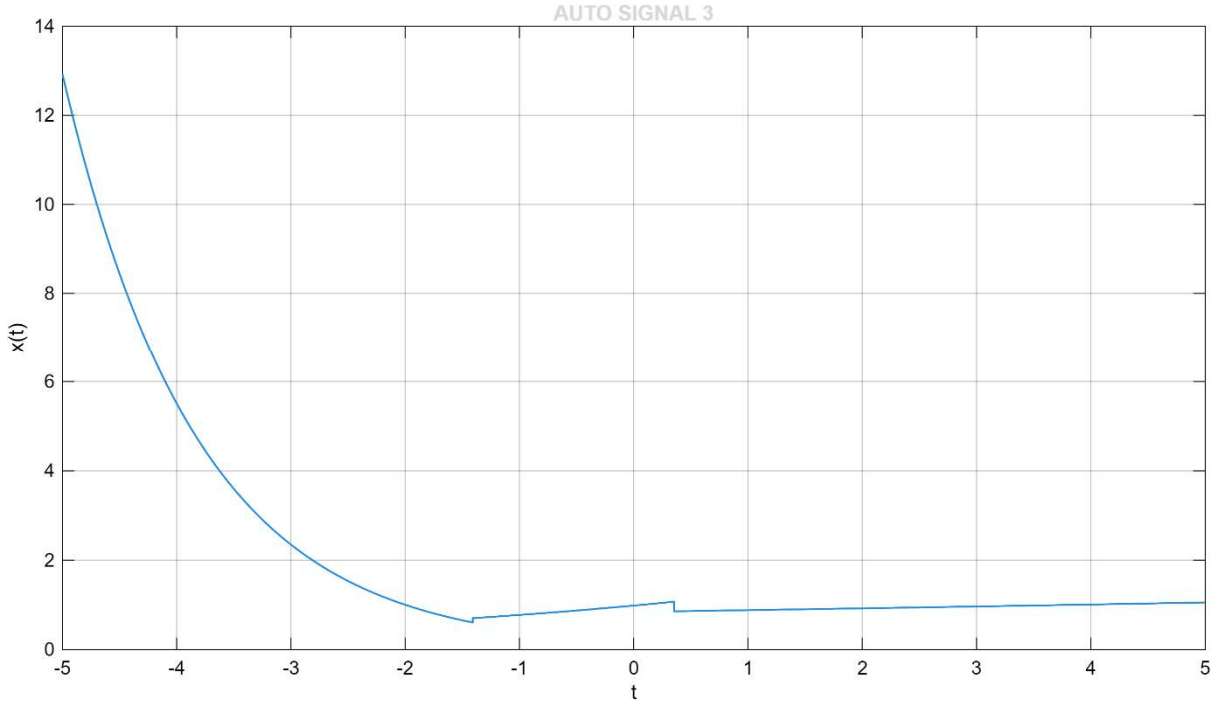
Operation: Amplitude Scale (-2)

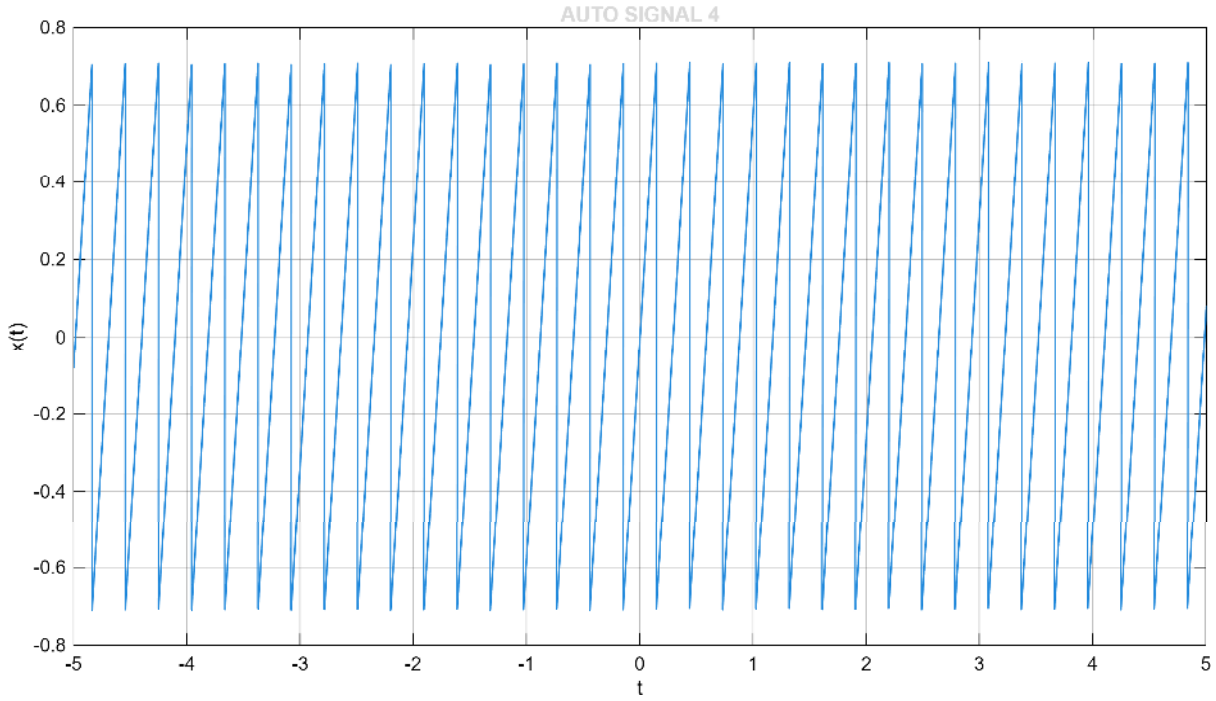
Generated Signal:

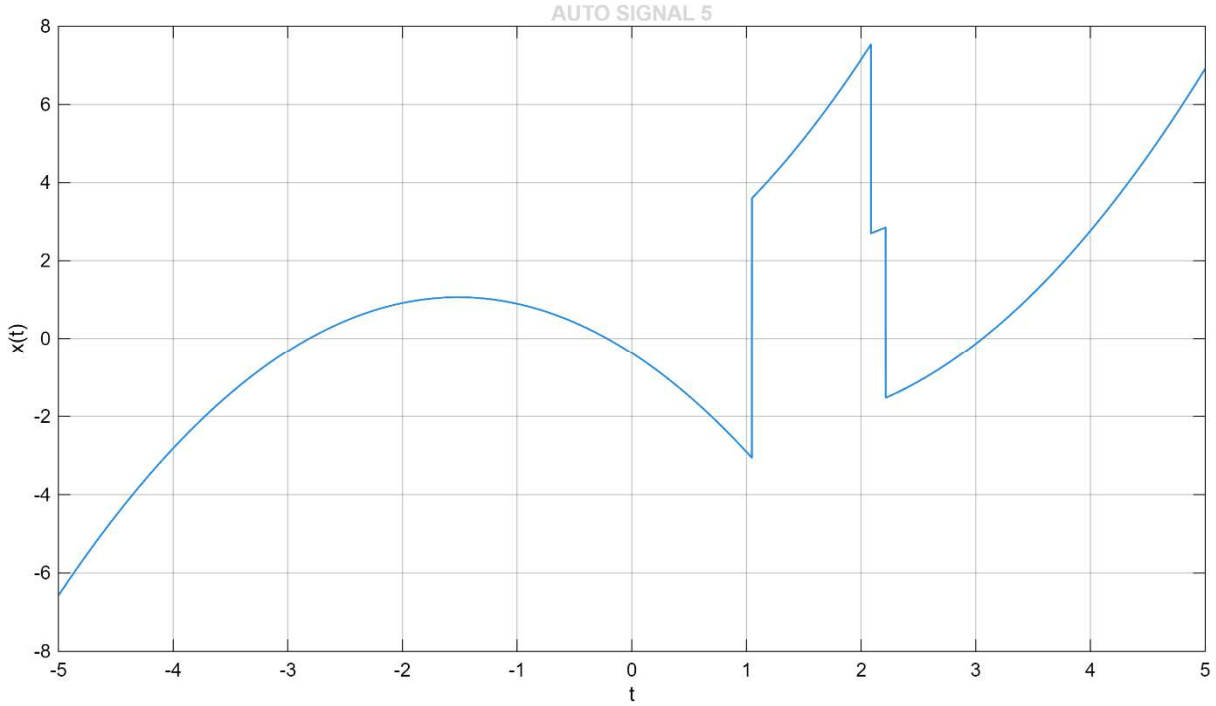


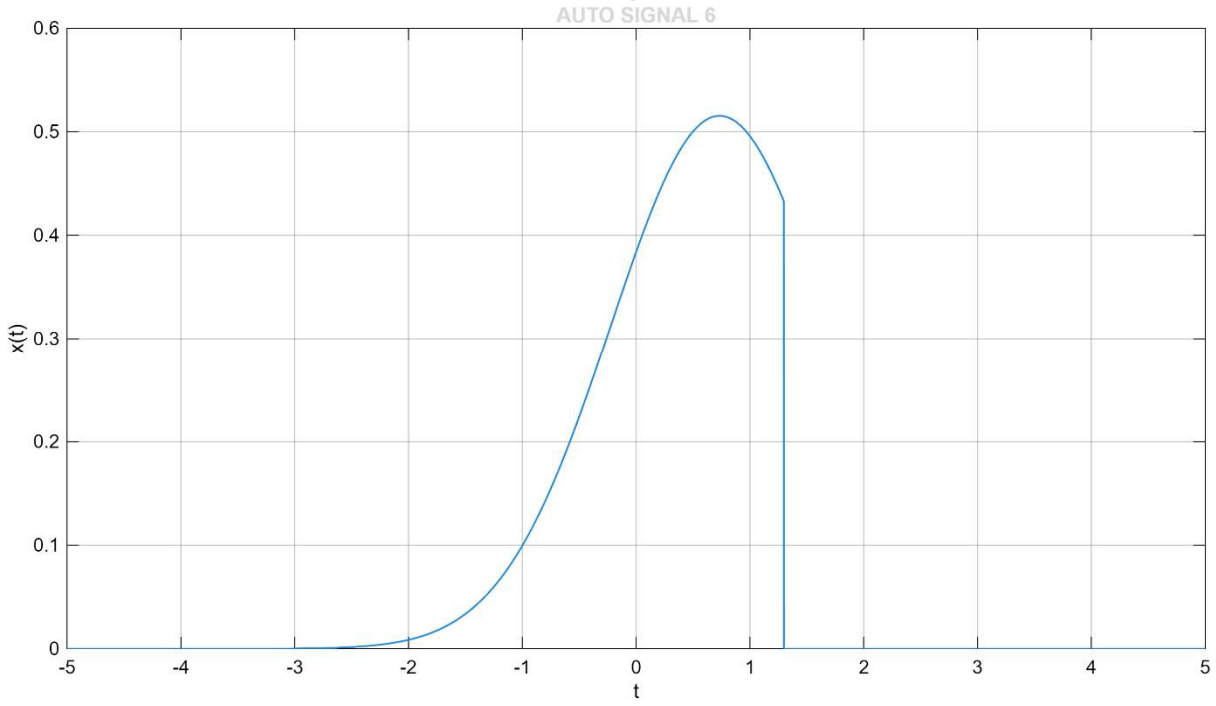
AUTO-Generated Signals (10):

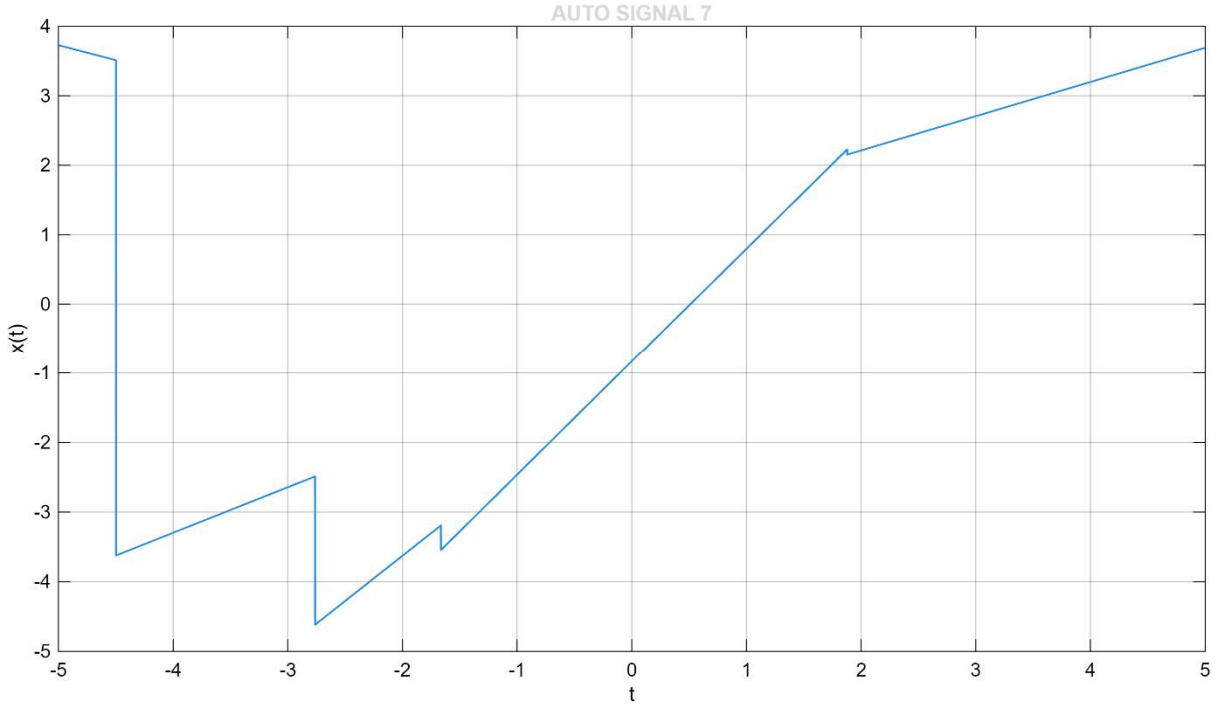


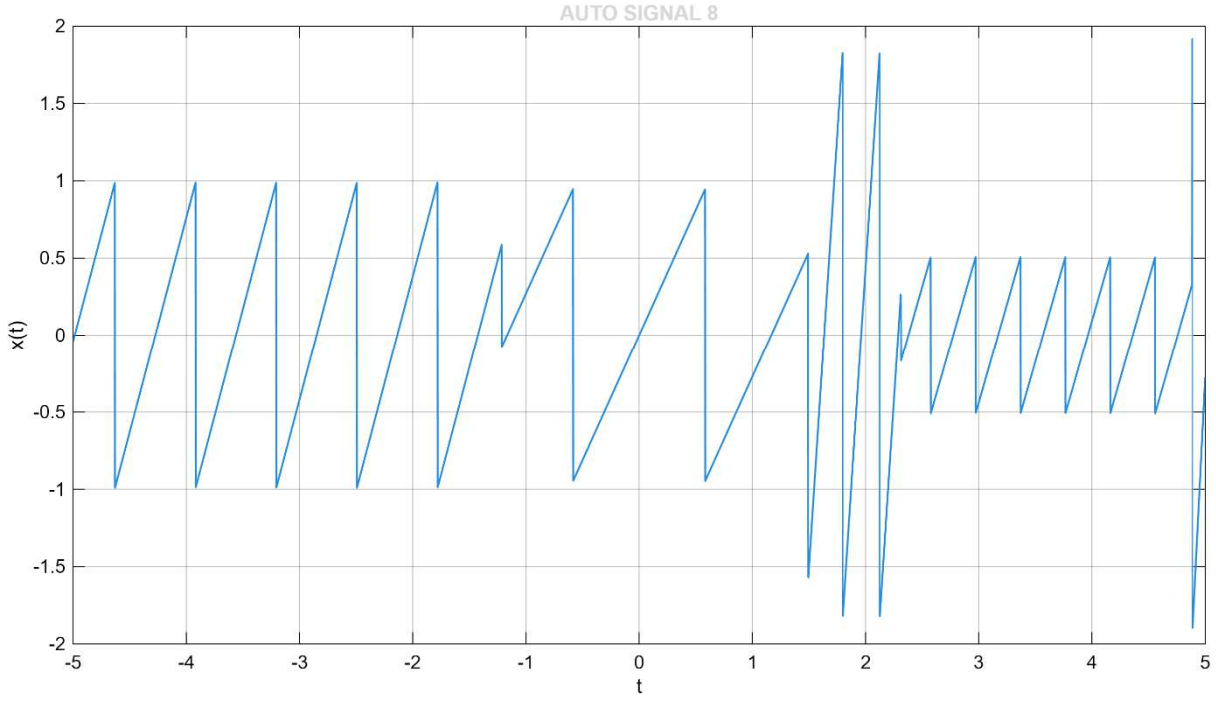

AUTO SIGNAL 3
x(t)
t
14
12
10
8
6
4
2
0
-5
-4
-3
-2
-1
0
1
2
3
4
5

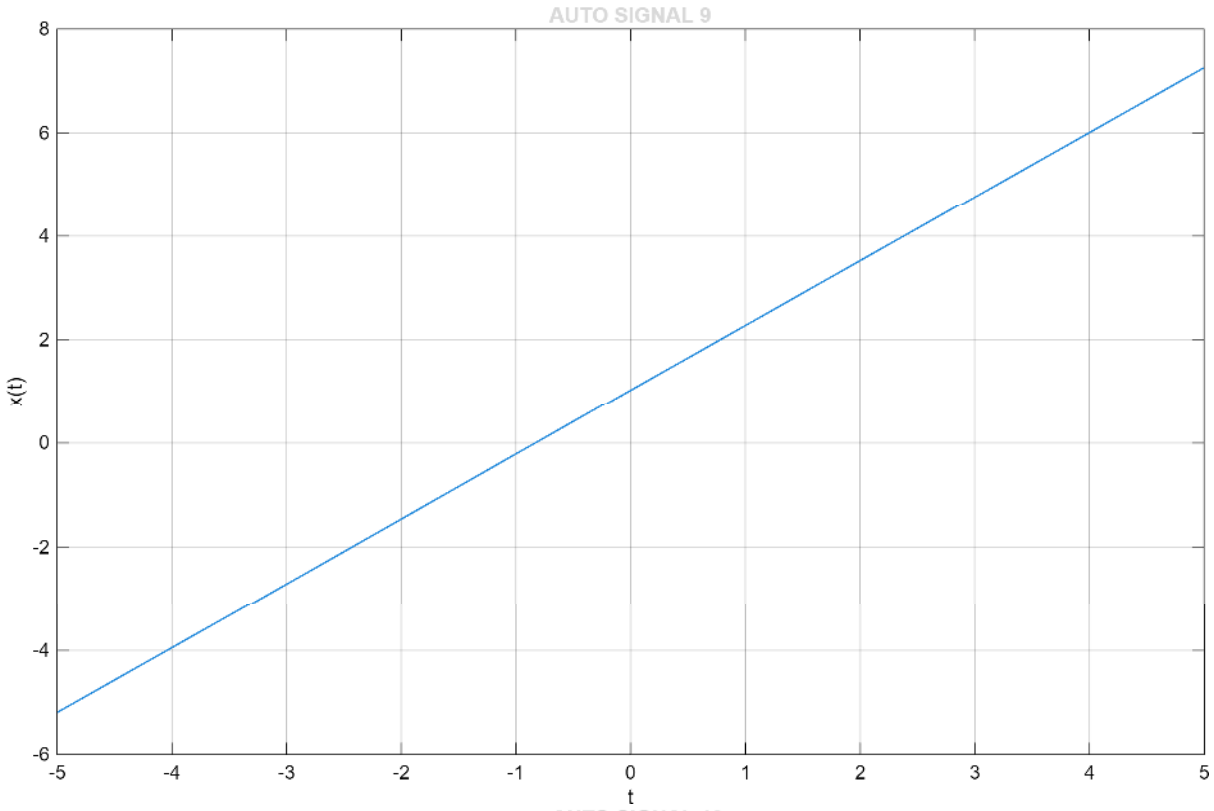

AUTO SIGNAL 4
x(t)
t
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-5
-4
-3
-2
-1
0
1
2
3
4
5

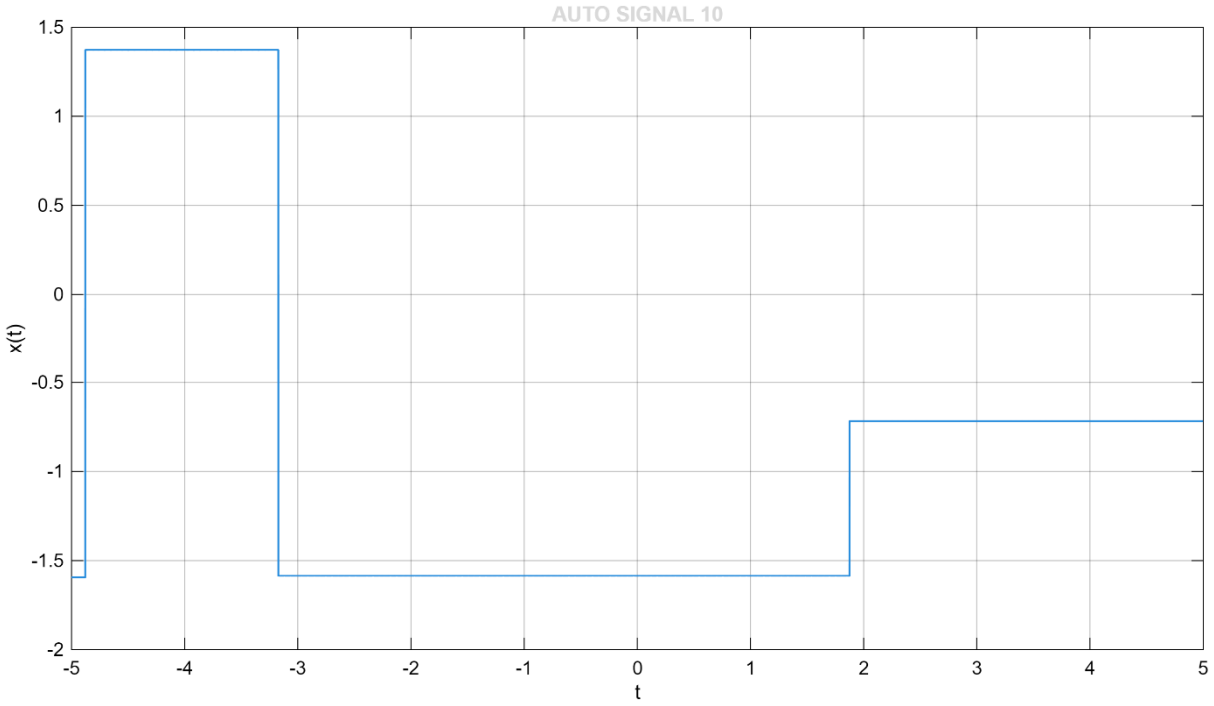

AUTO SIGNAL 5
x(t)
t
8
6
4
2
0
-2
-4
-6
-8
-5
-4
-3
-2
-1
0
1
2
3
4
5

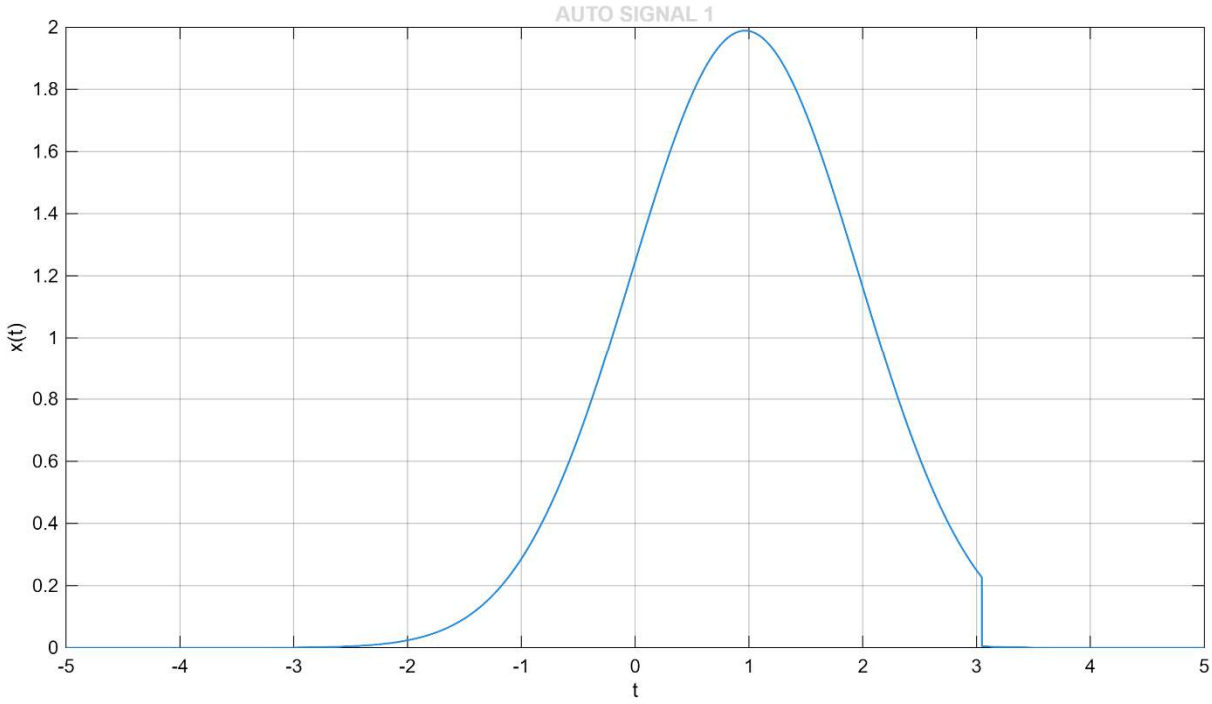

AUTO SIGNAL 6
x(t)
t
0.6
0.5
0.4
0.3
0.2
0.1
0
-5
-4
-3
-2
-1
0
1
2
3
4
5


AUTO SIGNAL 7
x(t)
t


AUTO SIGNAL 8
x(t)
t


AUTO SIGNAL 9
x(t)
t
8
6
4
2
0
-2
-4
-6
-5
-4
-3
-2
-1
0
1
2
3
4
5


AUTO SIGNAL 10
x(t)
t
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-5
-4
-3
-2
-1
0
1
2
3
4
5


AUTO SIGNAL 1
x(t)
t
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-5
-4
-3
-2
-1
0
1
2
3
4
5