SIGNAL AND SYSTEM

DR . MOHAMED ALI

- *team member*

~ MAHMOUD ATEF
~ AHMED HATEM
~ HANY BAHNSY
~ ASEEL
~ NANCY

Introduction

1. SAW TOOTH WAVE:

A SAW TOOTH WAVE IS A TYPE OF PERIODIC WAVEFORM THAT INCREASES LINEARLY OVER TIME AND THEN SHARPLY DROPS BACK TO THE STARTING POINT, RESEMBLING THE TEETH OF A SAW. THIS WAVEFORM IS WIDELY USED IN SIGNAL PROCESSING, ELECTRONICS, AND CONTROL SYSTEMS.

CHARACTERISTICS:

PERIODIC AND REPETITIVE: THE SAW TOOTH WAVE REPEATS AT REGULAR INTERVALS, MAKING IT USEFUL FOR GENERATING FIXED FREQUENCIES IN VARIOUS APPLICATIONS.

LINEAR RISE: THE WAVE RISES LINEARLY FROM A MINIMUM VALUE TO A MAXIMUM VALUE BEFORE ABRUPTLY DROPPING BACK TO THE STARTING POINT.

APPLICATIONS: IT'S COMMONLY USED IN GENERATING CLOCK SIGNALS, IN OSCILLATORS, AND IN SIGNAL PROCESSING.

MATHEMATICAL REPRESENTATION: IF TTT REPRESENTS TIME, A BASIC SAW TOOTH FUNCTION CAN BE REPRESENTED AS:

- $X(T)=A \cdot (T \bmod T)$

- WHERE AAA IS THE AMPLITUDE, AND TTT IS THE PERIOD.

2. UNIT STEP FUNCTION:

THE UNIT STEP FUNCTION, OFTEN REFERRED TO AS THE HEAVISIDE FUNCTION, IS A MATHEMATICAL FUNCTION THAT STARTS AT ZERO AND THEN JUMPS TO ONE AT A CERTAIN POINT IN TIME. THIS FUNCTION IS USED TO MODEL SUDDEN CHANGES OR EVENTS IN TIME.

- **CHARACTERISTICS:**
  - SUDDEN CHANGE: THE FUNCTION IS ZERO FOR ALL TIME BEFORE A SPECIFIC POINT, AND THEN IT JUMPS TO ONE AT THAT POINT.
  - APPLICATIONS: IT IS USED TO REPRESENT EVENTS THAT START AT A SPECIFIC TIME, SUCH AS TURNING ON A DEVICE OR STARTING A PROCESS IN A CONTROL SYSTEM.

- **MATHEMATICAL REPRESENTATION:**

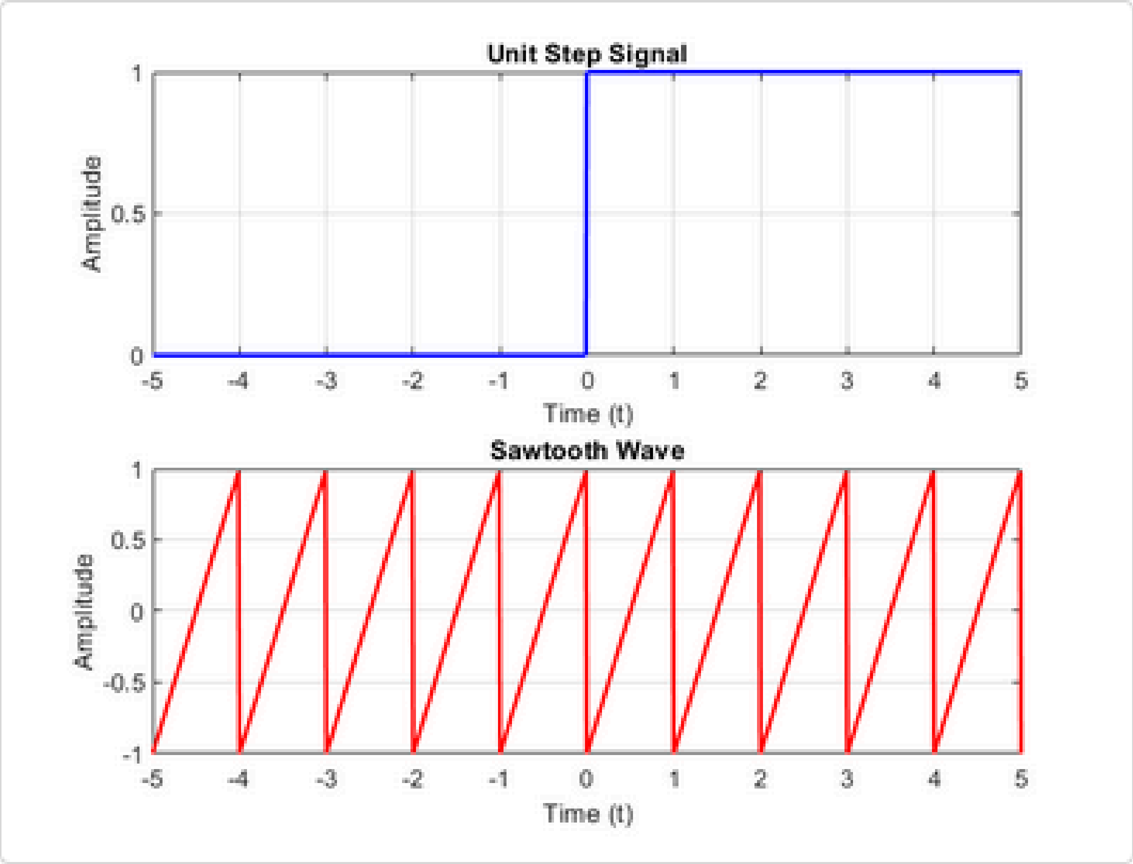
  - $U(T)=\{\ 0 \text{ FOR } T < 0$
  - $U(T)=\{\ 1 \text{ FOR } T \geqslant 0$

WHERE U(T)U(T)U(T) IS THE UNIT STEP FUNCTION.

Live Editor - Untitled.mlx *

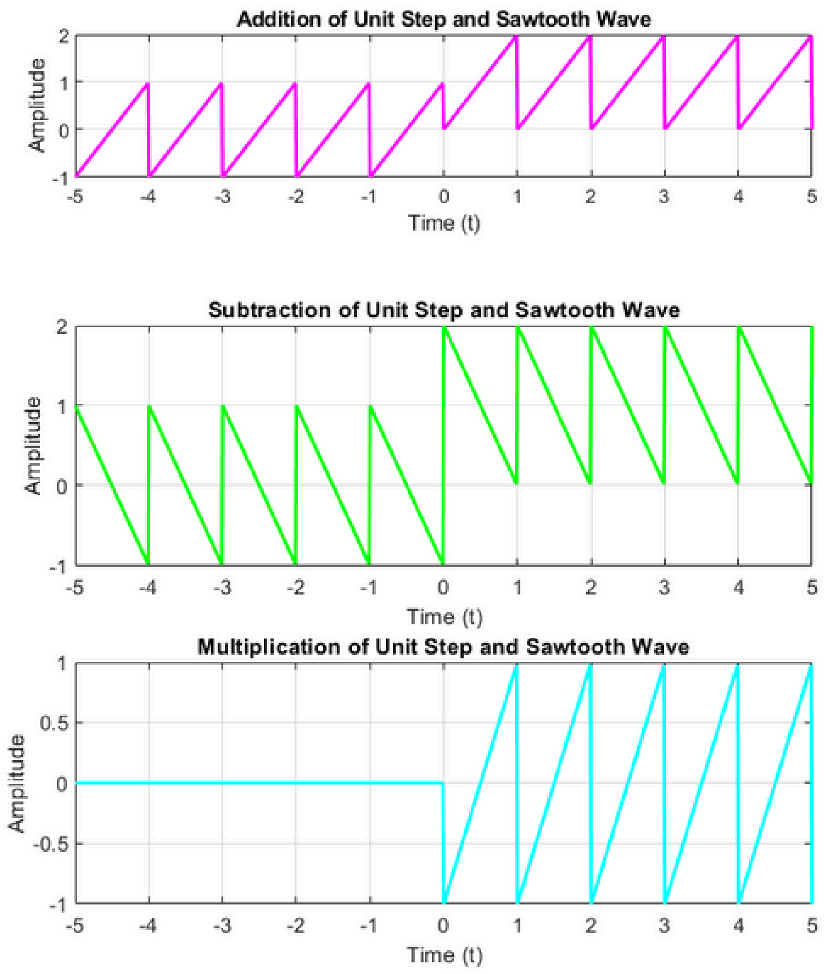
Untitled.mlx *

```matlab
% Unit Step Signal
t = -5:0.01:5; % Time vector
unit_step = t >= 0; % Unit step function
figure;
subplot(2, 1, 1);
plot(t, unit_step, 'b', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Unit Step Signal');
grid on;

% Sawtooth Wave
f = 1; % Frequency in Hz
saw_wave = sawtooth(2 * pi * f * t); % Sawtooth wave
subplot(2, 1, 2);
plot(t, saw_wave, 'r', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Sawtooth Wave');
grid on
```



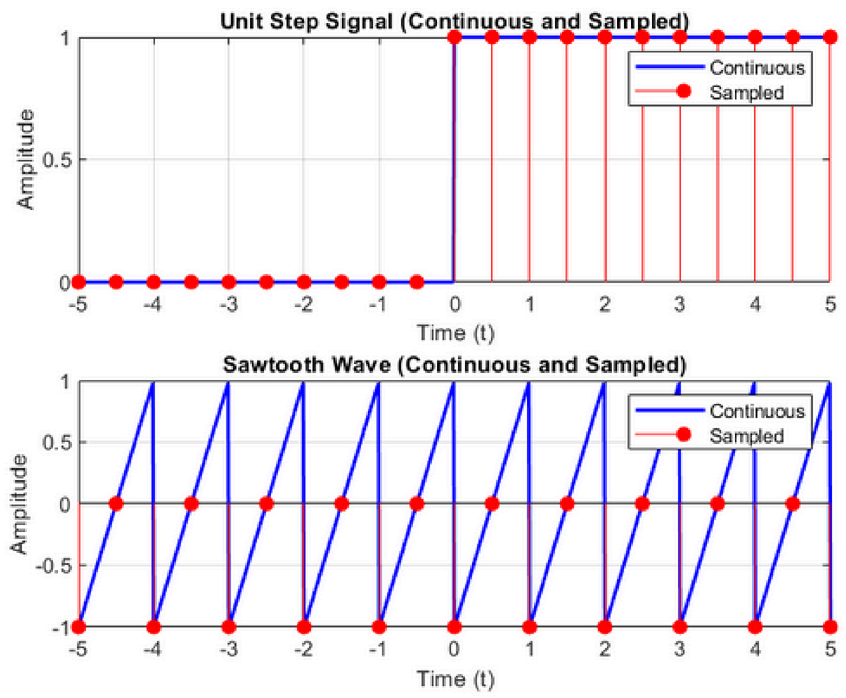
Untitled.mlx * Untitled2.mlx *

```
% Addition of Signals
signal_add = unit_step + saw_wave;
subplot(3, 1, 3);
plot(t, signal_add, 'm', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Addition of Unit Step and Sawtooth Wave');
grid on;

figure;

% Subtraction of Signals
signal_sub = unit_step - saw_wave;
subplot(2, 1, 1);
plot(t, signal_sub, 'g', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Subtraction of Unit Step and Sawtooth Wave');
grid on;

% Multiplication of Signals
signal_mult = unit_step .* saw_wave;
subplot(2, 1, 2);
plot(t, signal_mult, 'c', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Multiplication of Unit Step and Sawtooth Wave');
grid on;
```

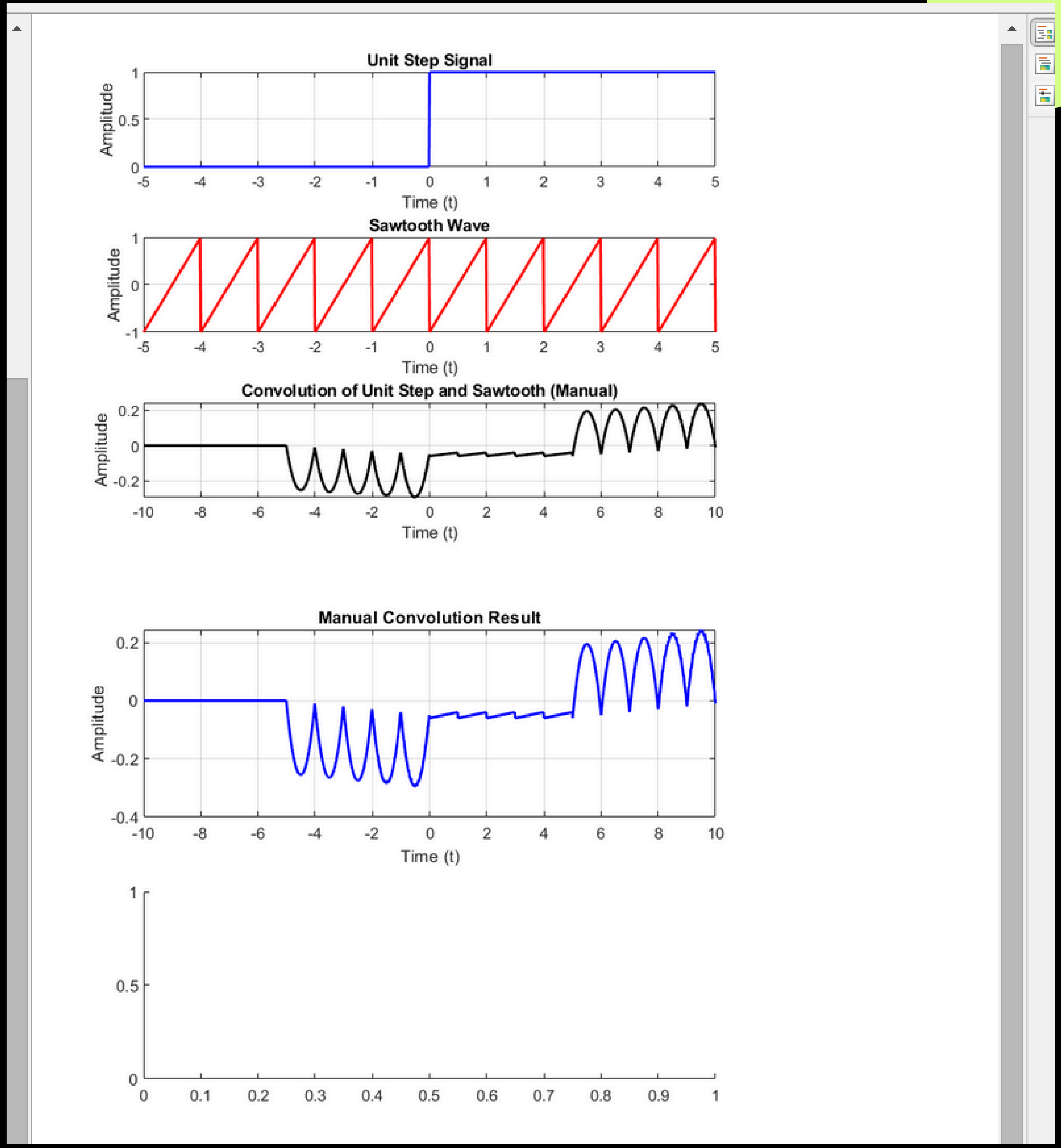


Untitled.mlx * | Untitled2.mlx * | Untitled3.mlx *

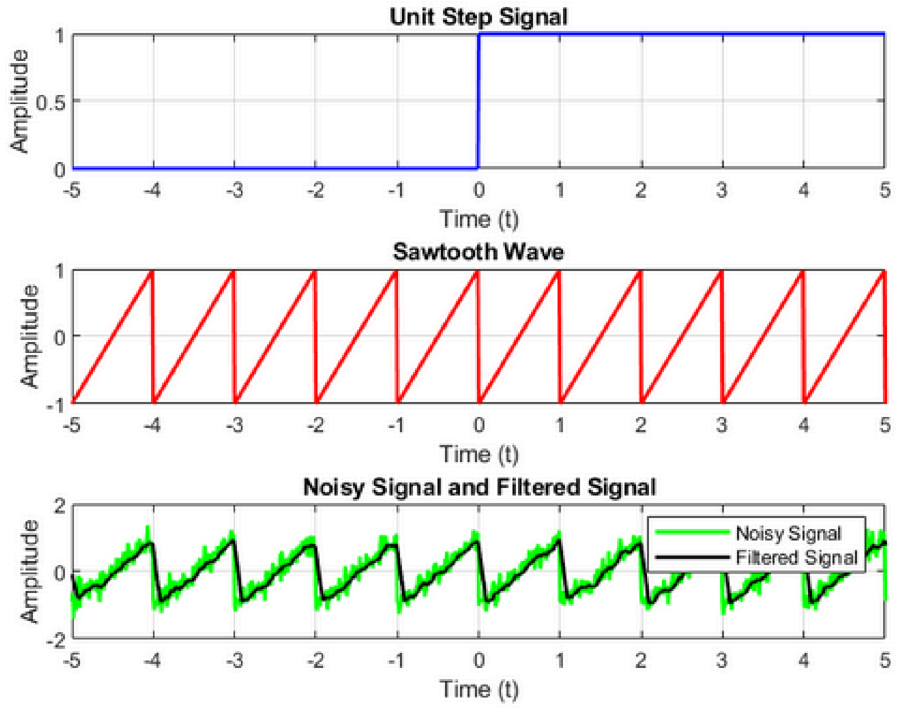
```matlab
% Sampling of Signals

% Unit Step Signal
t = -5:0.01:5; % Continuous time vector
unit_step = t >= 0; % Unit step function

% Sampling parameters
Ts = 0.5; % Sampling interval
t_sampled = -5:Ts:5; % Sampled time vector
unit_step_sampled = t_sampled >= 0; % Sampled unit step function

figure;

% Plot continuous unit step signal
subplot(2, 1, 1);
plot(t, unit_step, 'b', 'LineWidth', 1.5); hold on;
stem(t_sampled, unit_step_sampled, 'r', 'filled'); % Sampled signal
xlabel('Time (t)');
ylabel('Amplitude');
title('Unit Step Signal (Continuous and Sampled)');
legend('Continuous', 'Sampled');
grid on;

% Sawtooth Wave
f = 1; % Frequency in Hz
saw_wave = sawtooth(2 * pi * f * t); % Continuous sawtooth wave
saw_wave_sampled = sawtooth(2 * pi * f * t_sampled); % Sampled sawtooth wave

% Plot continuous and sampled sawtooth wave
subplot(2, 1, 2);
plot(t, saw_wave, 'b', 'LineWidth', 1.5); hold on;
stem(t_sampled, saw_wave_sampled, 'r', 'filled'); % Sampled signal
xlabel('Time (t)');
ylabel('Amplitude');
title('Sawtooth Wave (Continuous and Sampled)');
legend('Continuous', 'Sampled');
grid on;
```



```matlab
% Unit Step Signal
t = -5:0.01:5; % Time vector
unit_step = t >= 0; % Unit step function

% Sawtooth Wave
f = 1; % Frequency in Hz
saw_wave = sawtooth(2 * pi * f * t); % Sawtooth wave

% Plot Unit Step Signal
figure;
subplot(3, 1, 1);
plot(t, unit_step, 'b', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Unit Step Signal');
grid on;

% Plot Sawtooth Wave
subplot(3, 1, 2);
plot(t, saw_wave, 'r', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Sawtooth Wave');
grid on;

% Convolution of Unit Step and Sawtooth Wave (Manual Calculation)
dt = 0.01; % Time step (sampling interval)
t_conv = -10:dt:10; % Time vector for convolution result
conv_result_manual = zeros(size(t_conv)); % Initialize convolution result

% Perform convolution using the formula (discrete convolution)
for i = 1:length(t_conv)
    tau = t_conv(i) - t; % Shifted time vector
    % Perform the integral-like sum (discrete convolution)
    conv_result_manual(i) = sum(unit_step .* interp1(t, saw_wave, tau, 'linear', 0)) * dt;
end

% Plot Manual Convolution Result
subplot(3, 1, 3);
plot(t_conv, conv_result_manual, 'k', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Convolution of Unit Step and Sawtooth (Manual)');
grid on;

% Convolution using MATLAB's conv function
conv_result = conv(unit_step, saw_wave, 'same') * dt; % Normalize by sampling interval

% Compare the results
% Add a figure for comparison
figure;
subplot(2, 1, 1);
plot(t_conv, conv_result_manual, 'b', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Manual Convolution Result');
grid on;

subplot(2, 1, 2);
plot(t_conv, conv_result, 'r', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('MATLAB conv() Function Result');
grid on;

% Display Comparison Result
disp('Comparison of manual and MATLAB conv() results:')
disp('Manual Convolution vs MATLAB conv()')
disp('Difference:');
disp(conv_result_manual - conv_result);
```

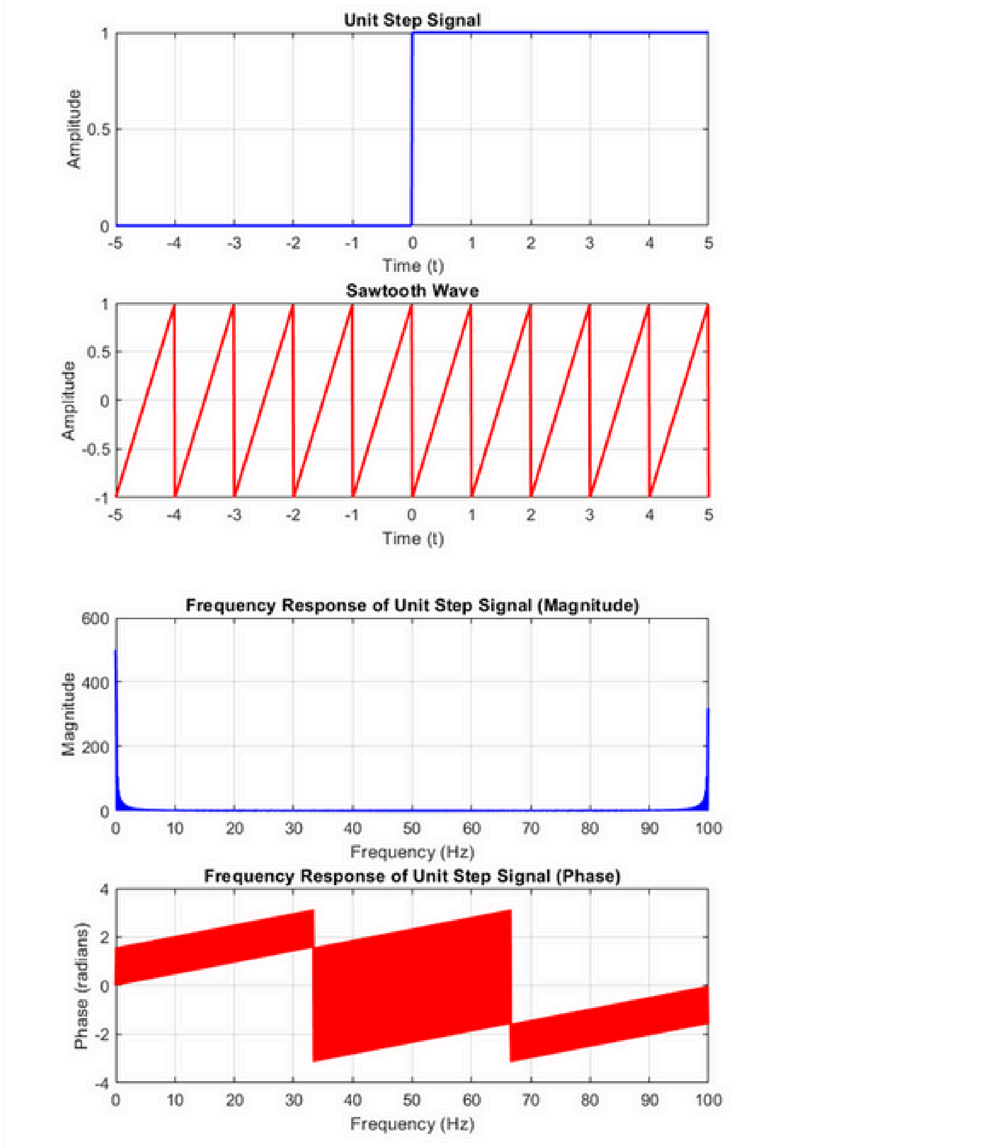


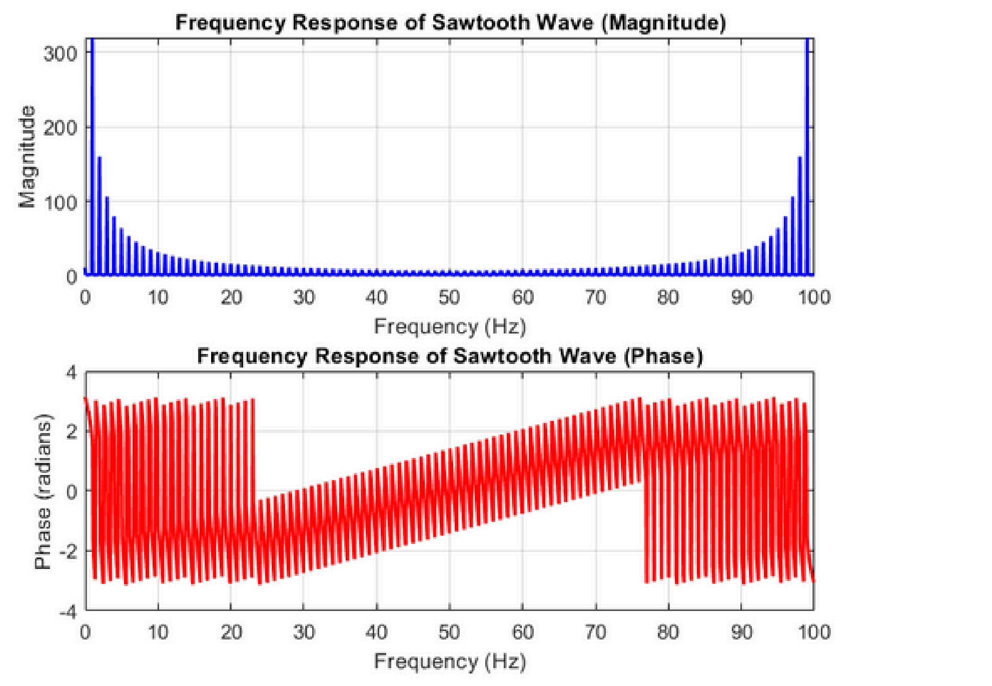
```
% Unit Step Signal
t = -5:0.01:5; % Time vector
unit_step = t >= 0; % Unit step function
figure;
subplot(3, 1, 1);
plot(t, unit_step, 'b', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Unit Step Signal');
grid on;

% Sawtooth Wave
f = 1; % Frequency in Hz
saw_wave = sawtooth(2 * pi * f * t); % Sawtooth wave
subplot(3, 1, 2);
plot(t, saw_wave, 'r', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Sawtooth Wave');
grid on;

% Add Noise to Sawtooth Wave
noise = 0.2 * randn(size(t)); % Gaussian noise
noisy_signal = saw_wave + noise;

% Apply Averaging Filter
window_size = 10; % Number of points for averaging
avg_filter = ones(1, window_size) / window_size; % Averaging filter coefficients
filtered_signal = filter(avg_filter, 1, noisy_signal);

% Plot Noisy and Filtered Signal
subplot(3, 1, 3);
plot(t, noisy_signal, 'g', 'LineWidth', 1.5); hold on;
plot(t, filtered_signal, 'k', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Noisy Signal and Filtered Signal');
legend('Noisy Signal', 'Filtered Signal');
grid on;
```



```
% Unit Step Signal
t = -5:0.01:5; % Time vector
unit_step = t >= 0; % Unit step function
figure;
subplot(2, 1, 1);
plot(t, unit_step, 'b', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Unit Step Signal');
grid on;

% Sawtooth Wave
f = 1; % Frequency in Hz
saw_wave = sawtooth(2 * pi * f * t); % Sawtooth wave
subplot(2, 1, 2);
plot(t, saw_wave, 'r', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Sawtooth Wave');
grid on;

% Frequency Response using FFT
Fs = 100; % Sampling Frequency in Hz
N = length(t); % Number of points in the signal
t_step = t(2) - t(1); % Time step (sampling period)

% Compute FFT of the signals (Unit Step and Sawtooth)
unit_step_fft = fft(unit_step);
saw_wave_fft = fft(saw_wave);

% Frequency axis for plotting (from 0 to Nyquist frequency)
f_axis = (0:N-1)*(Fs/N);

% Compute magnitude and phase
unit_step_magnitude = abs(unit_step_fft);
unit_step_phase = angle(unit_step_fft);

saw_wave_magnitude = abs(saw_wave_fft);
saw_wave_phase = angle(saw_wave_fft);

% Plot Frequency Response using FFT (Magnitude and Phase)
figure;
subplot(2, 1, 1);
plot(f_axis, unit_step_magnitude, 'b', 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Frequency Response of Unit Step Signal (Magnitude)');
grid on;
```

```
subplot(2, 1, 2);
plot(f_axis, unit_step_phase, 'r', 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Phase (radians)');
title('Frequency Response of Unit Step Signal (Phase)');
grid on;

% Plot Frequency Response using Sawtooth Wave (Magnitude and Phase)
figure;
subplot(2, 1, 1);
plot(f_axis, saw_wave_magnitude, 'b', 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Frequency Response of Sawtooth Wave (Magnitude)');
grid on;

subplot(2, 1, 2);
plot(f_axis, saw_wave_phase, 'r', 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Phase (radians)');
title('Frequency Response of Sawtooth Wave (Phase)');
grid on;

% Frequency Response Using Formula
% For continuous signals, this would normally require analytical calculations.
% Here we use the FFT as an approximation.
```

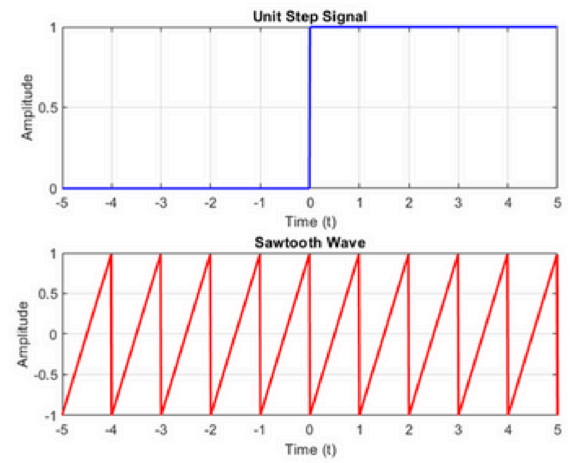


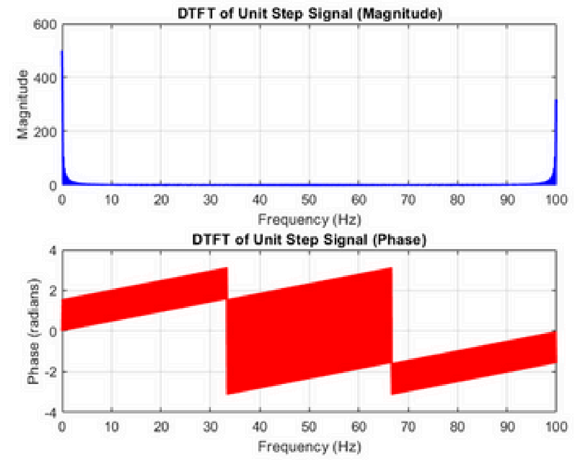


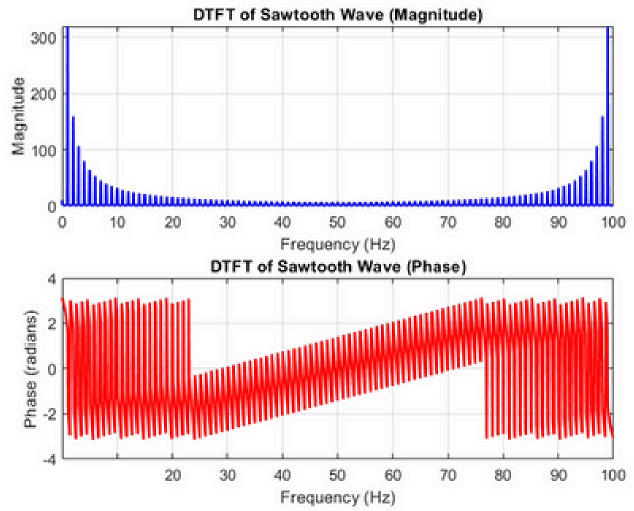
```matlab
% Unit Step Signal
t = -5:0.01:5; % Time vector
unit_step = t >= 0; % Unit step function

% Sawtooth Wave
f = 1; % Frequency in Hz
saw_wave = sawtooth(2 * pi * f * t); % Sawtooth wave

% Plot Unit Step Signal and Sawtooth Wave
figure;
subplot(2, 1, 1);
plot(t, unit_step, 'b', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Unit Step Signal');
grid on;

subplot(2, 1, 2);
plot(t, saw_wave, 'r', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Sawtooth Wave');
grid on;

% Discrete Time Fourier Transform (DTFT) Approximation using FFT
Fs = 100; % Sampling frequency in Hz
N = length(t); % Number of samples in the signal
f_axis = (0:N-1)*(Fs/N); % Frequency axis for plotting

% Compute FFT of the signals (Unit Step and Sawtooth)
unit_step_fft = fft(unit_step);
saw_wave_fft = fft(saw_wave);

% Magnitude and Phase of the Fourier Transforms
unit_step_magnitude = abs(unit_step_fft);
unit_step_phase = angle(unit_step_fft);

saw_wave_magnitude = abs(saw_wave_fft);
saw_wave_phase = angle(saw_wave_fft);

% Plot Frequency Response using FFT (Magnitude and Phase)
figure;
subplot(2, 1, 1);
plot(f_axis, unit_step_magnitude, 'b', 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('DTFT of Unit Step Signal (Magnitude)');
grid on;
```

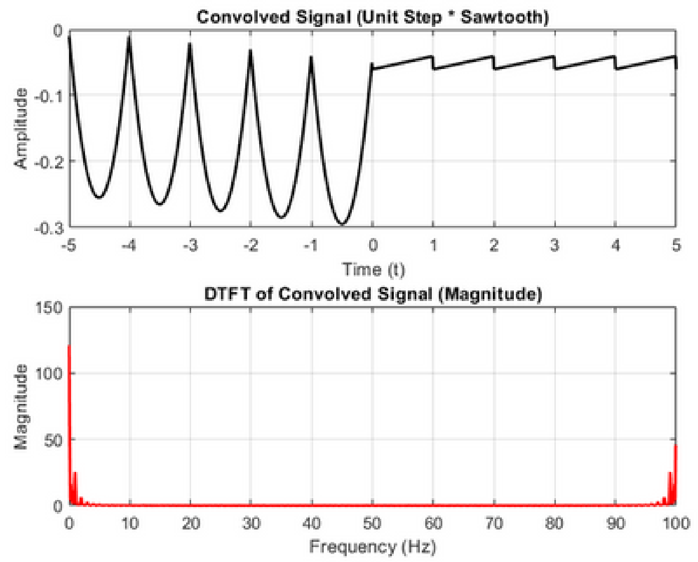
```matlab
subplot(2, 1, 2);
plot(f_axis, unit_step_phase, 'r', 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Phase (radians)');
title('DTFT of Unit Step Signal (Phase)');
grid on;

% Plot Frequency Response of Sawtooth Wave
figure;
subplot(2, 1, 1);
plot(f_axis, saw_wave_magnitude, 'b', 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('DTFT of Sawtooth Wave (Magnitude)');
grid on;

subplot(2, 1, 2);
plot(f_axis, saw_wave_phase, 'r', 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Phase (radians)');
title('DTFT of Sawtooth Wave (Phase)');
grid on;

% Time Shifting Property: Shift the Unit Step Signal by 2 units
shifted_unit_step = circshift(unit_step, 200); % Time shift of 2 units (in samples)

% Compute FFT of the shifted signal
shifted_unit_step_fft = fft(shifted_unit_step);
shifted_unit_step_magnitude = abs(shifted_unit_step_fft);
shifted_unit_step_phase = angle(shifted_unit_step_fft);

% Plot Time Shifting Property
figure;
subplot(2, 1, 1);
plot(t, shifted_unit_step, 'g', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Shifted Unit Step Signal');
grid on;

subplot(2, 1, 2);
plot(f_axis, shifted_unit_step_magnitude, 'b', 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('DTFT of Shifted Unit Step Signal (Magnitude)');
grid on;

% Convolution Property: Convolve the Unit Step and Sawtooth Wave
convolved_signal = conv(unit_step, saw_wave, 'same') * (t(2) - t(1)); % Convolution result
```

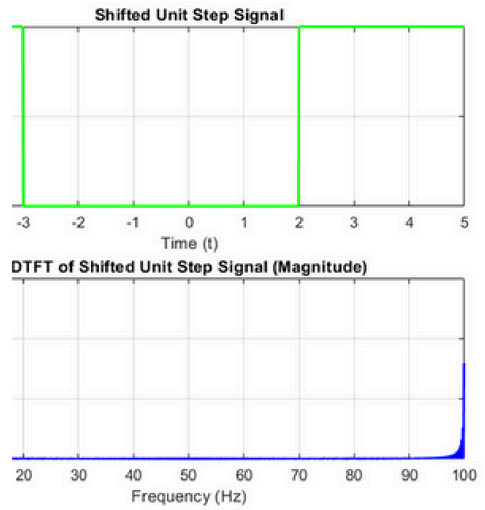
```matlab
% Plot Convolution Property
figure;
subplot(2, 1, 1);
plot(t, convolved_signal, 'k', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Convolved Signal (Unit Step * Sawtooth)');
grid on;

subplot(2, 1, 2);
plot(f_axis, convolved_signal_magnitude, 'r', 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('DTFT of Convolved Signal (Magnitude)');
grid on;
```





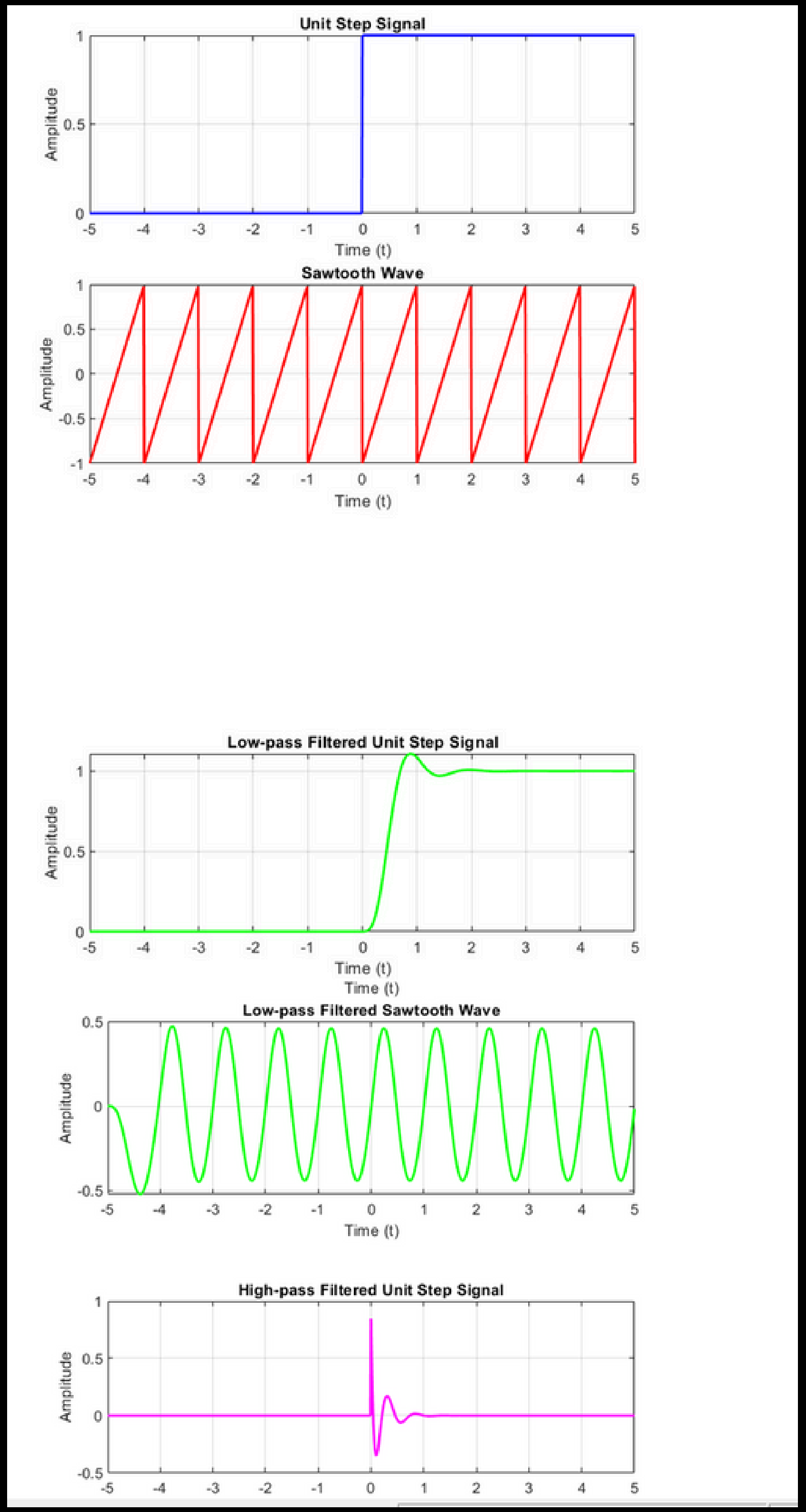




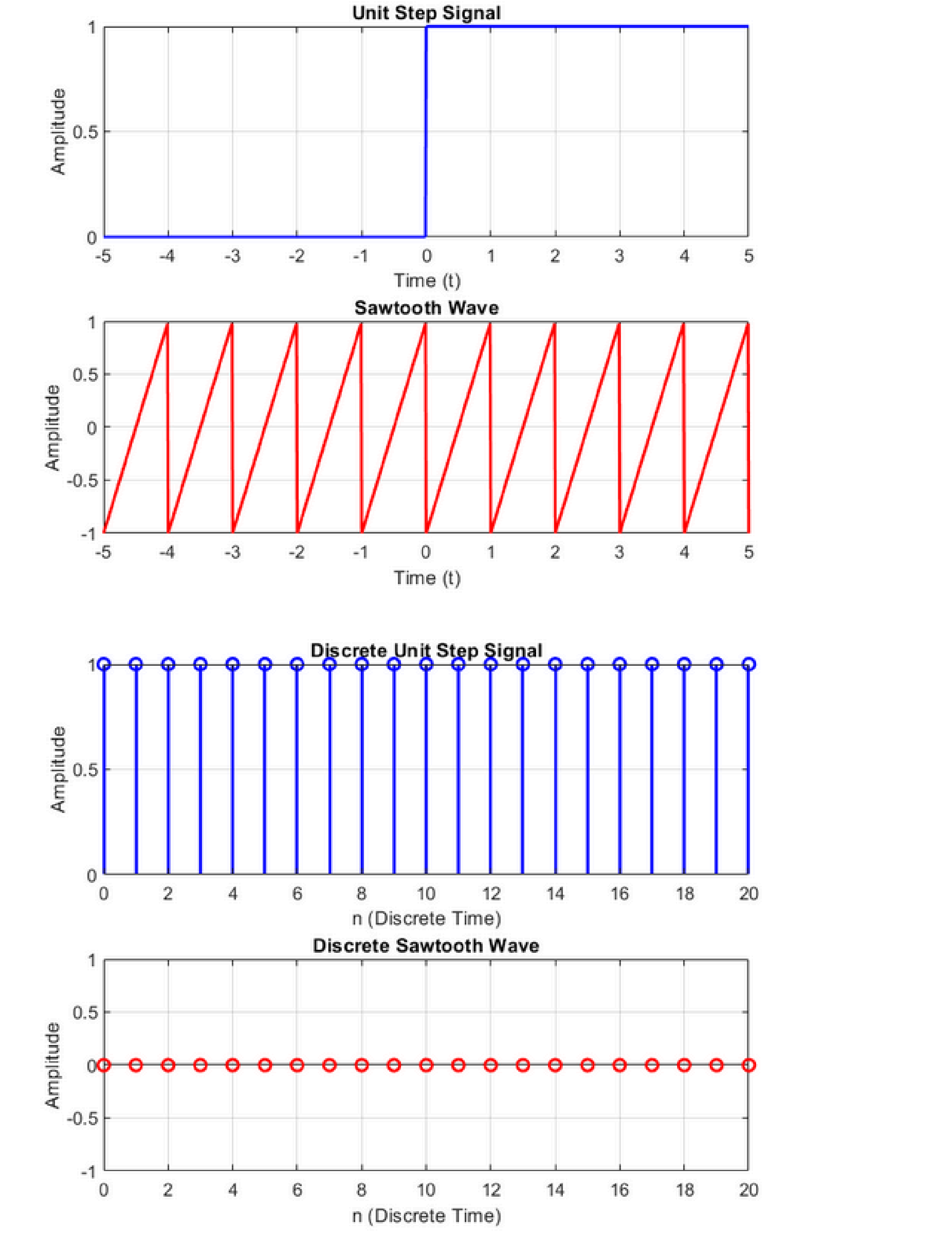


```
% Unit Step Signal
t = -5:0.01:5; % Time vector
unit_step = t >= 0; % Unit step function

% Sawtooth Wave
f = 1; % Frequency in Hz
saw_wave = sawtooth(2 * pi * f * t); % Sawtooth wave

% Plot Unit Step Signal and Sawtooth Wave
figure;
subplot(2, 1, 1);
plot(t, unit_step, 'b', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Unit Step Signal');
grid on;

subplot(2, 1, 2);
plot(t, saw_wave, 'r', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Sawtooth Wave');
grid on;

% Filter Design: Low-pass, High-pass, and Band-pass Filters

% Low-pass filter (e.g., cutoff frequency = 1 Hz)
lp_cutoff = 1; % Low-pass filter cutoff frequency
Fs = 100; % Sampling frequency
nyquist = Fs / 2; % Nyquist frequency
lp_norm_cutoff = lp_cutoff / nyquist; % Normalized cutoff frequency
[b_lp, a_lp] = butter(4, lp_norm_cutoff, 'low'); % 4th order low-pass filter

% High-pass filter (e.g., cutoff frequency = 2 Hz)
hp_cutoff = 2; % High-pass filter cutoff frequency
hp_norm_cutoff = hp_cutoff / nyquist; % Normalized cutoff frequency
[b_hp, a_hp] = butter(4, hp_norm_cutoff, 'high'); % 4th order high-pass filter

% Band-pass filter (e.g., pass frequencies between 1 and 3 Hz)
bp_low = 1; % Low frequency for band-pass
bp_high = 3; % High frequency for band-pass
bp_norm_low = bp_low / nyquist; % Normalized low cutoff
bp_norm_high = bp_high / nyquist; % Normalized high cutoff
[b_bp, a_bp] = butter(4, [bp_norm_low, bp_norm_high], 'bandpass'); % 4th order band-pass filter

% Filter the Signals
filtered_unit_step_lp = filter(b_lp, a_lp, unit_step); % Apply low-pass filter to unit step signal
filtered_saw_wave_lp = filter(b_lp, a_lp, saw_wave); % Apply low-pass filter to sawtooth wave
```

```
filtered_unit_step_hp = filter(b_hp, a_hp, unit_step); % Apply high-pass filter to unit step signal
filtered_saw_wave_hp = filter(b_hp, a_hp, saw_wave); % Apply high-pass filter to sawtooth wave

filtered_unit_step_bp = filter(b_bp, a_bp, unit_step); % Apply band-pass filter to unit step signal
filtered_saw_wave_bp = filter(b_bp, a_bp, saw_wave); % Apply band-pass filter to sawtooth wave
Run to Here
Run up to this line and pause
d Signals

% Low-pass filtered signals
figure;
subplot(2, 1, 1);
plot(t, filtered_unit_step_lp, 'g', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Low-pass Filtered Unit Step Signal');
grid on;

subplot(2, 1, 2);
plot(t, filtered_saw_wave_lp, 'g', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Low-pass Filtered Sawtooth Wave');
grid on;

% High-pass filtered signals
figure;
subplot(2, 1, 1);
plot(t, filtered_unit_step_hp, 'm', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('High-pass Filtered Unit Step Signal');
grid on;

subplot(2, 1, 2);
plot(t, filtered_saw_wave_hp, 'm', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('High-pass Filtered Sawtooth Wave');
grid on;

% Band-pass filtered signals
figure;
subplot(2, 1, 1);
plot(t, filtered_unit_step_bp, 'c', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Band-pass Filtered Unit Step Signal');
grid on;

subplot(2, 1, 2);
plot(t, filtered_saw_wave_bp, 'c', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Band-pass Filtered Sawtooth Wave');
grid on;
```



```matlab
% Unit Step Signal
t = -5:0.01:5; % Time vector
unit_step = t >= 0; % Unit step function

% Sawtooth Wave
f = 1; % Frequency in Hz
saw_wave = sawtooth(2 * pi * f * t); % Sawtooth wave

% Plot Unit Step Signal and Sawtooth Wave
figure;
subplot(2, 1, 1);
plot(t, unit_step, 'b', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Unit Step Signal');
grid on;

subplot(2, 1, 2);
plot(t, saw_wave, 'r', 'LineWidth', 1.5);
xlabel('Time (t)');
ylabel('Amplitude');
title('Sawtooth Wave');
grid on;

% Now calculate the Z-Transform of the unit step and sawtooth signals

% Discretize the unit step and sawtooth signals
n = 0:20; % Discrete time index (can adjust for longer duration)
unit_step_discrete = n >= 0; % Unit step signal in discrete time
saw_wave_discrete = mod(n, 1); % A simple discretization of sawtooth wave

% Plot the Discretized Signals
figure;
subplot(2, 1, 1);
stem(n, unit_step_discrete, 'b', 'LineWidth', 1.5);
xlabel('n (Discrete Time)');
ylabel('Amplitude');
title('Discrete Unit Step Signal');
grid on;

subplot(2, 1, 2);
stem(n, saw_wave_discrete, 'r', 'LineWidth', 1.5);
xlabel('n (Discrete Time)');
ylabel('Amplitude');
title('Discrete Sawtooth Wave');
grid on;

% Z-Transform using MATLAB's Symbolic Math Toolbox

% Define the symbolic variable 'z'
syms z n;

% Unit step signal in discrete time
unit_step_z_transform = ztrans([unit_step_discrete], n, z);
disp('Z-Transform of Discrete Unit Step Signal:');
disp(unit_step_z_transform);

% Sawtooth wave signal in discrete time (simplified version)
saw_wave_z_transform = ztrans([saw_wave_discrete], n, z);
disp('Z-Transform of Discrete Sawtooth Signal:');
disp(saw_wave_z_transform);
```




```
Z-Transform of Discrete Unit Step Signal:
```

$$\left(\frac{z}{z-1} \quad \frac{z}{z-1} \quad \frac{z}{z-1} \quad \frac{z}{z-1} \quad \frac{z}{z-1} \quad \frac{z}{z-1} \quad \frac{z}{z-1} \quad \frac{z}{z-1} \quad \frac{z}{z-1} \quad \frac{z}{z-1} \quad \frac{z}{z-1} \quad \frac{z}{z-1} \quad \frac{z}{z-1} \quad \frac{z}{z-1} \quad \frac{z}{z-1} \quad \frac{z}{z-1} \quad \frac{z}{z-}\right.$$

```
Z-Transform of Discrete Sawtooth Signal:
```

$$(0 \quad 0)$$