भारतीय सूचना प्रौद्योगिकी संस्थान, नागपुर
Indian Institute of Information Technology, Nagpur

Applied Signals & System (ECE – 205)
Lab-2 Report
Batch- ECE- IoT-1

Submitted by – Jjateen Gundesha (BT22ECI002) **Submitted to - Dr. Nikhil Agrawal**

Aim: The aim of this lab is to explore signal processing and analysis in MATLAB by generating various discrete sequences, calculating their energy, and computing Fourier series coefficients.

Theory:

Introduction
In this lab, we delve into the world of signal processing and analysis, building on the fundamental concepts introduced in Lab-1. We have expanded our capabilities to generate and analyze discrete sequences, calculate the energy of signals, and compute Fourier series coefficients.

Sequence Generation
We have developed a custom function called `seq_generator` for generating different types of discrete sequences. This function accepts several parameters, including the type of sequence, range, and amplitude, and returns the desired sequence. The following types of sequences are supported:

1. Delta Function
2. Step Function
3. Signum Function
4. Exponential Decay ($0 < a < 1$)
5. Double-Sided Exponential Decay ($0 < a < 1$)
6. Gate Function
7. Sine Wave
8. Sinc Function

The type of sequence can be chosen by passing an integer value as a parameter to the `seq_generator` function.

Energy Calculation

The `Energy_Fx` function takes the data sequence and the step size as input and returns the energy of the signal. It is a valuable tool for signal characterization and analysis.

Fourier Series Computation

To explore the representation of signals in the frequency domain, we have implemented the computation of Fourier series coefficients. In this exercise, a square wave signal is used for demonstration. The following steps are involved:

1. Define the time vector `t` and the signal `f_t`.
2. Plot the square wave signal.
3. Compute the Fourier series coefficients using the discrete Fourier transform.

Code:

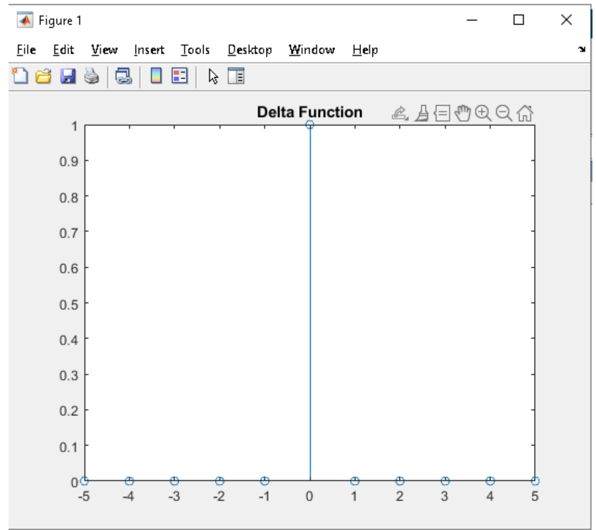
```
function [x_n] = seq_generator(n, type_seq, amplitude)
    len_n = length(n);
    x_n = zeros(1, len_n);  % Preallocate x_n

    if type_seq == 1
        % Type 1: Delta Function
        for ii = 1:len_n
            if n(ii) == 0
                x_n(ii) = amplitude;
            else
                x_n(ii) = 0;
            end
        end
    elseif type_seq == 2
        % Type 2: Step Function
        for ii = 1:len_n
            if n(ii) >= 0
                x_n(ii) = amplitude;
            else
                x_n(ii) = 0;
            end
        end
    elseif type_seq == 3
        % Type 3: Signum Function
        for ii = 1:len_n
            if n(ii) < 0
                x_n(ii) = -amplitude;
            elseif n(ii) > 0
                x_n(ii) = amplitude;
            else
                x_n(ii) = 0;
            end
        end
    elseif type_seq == 4
```

```
        % Type 4: Exponential Decay (0 < a < 1)
        for ii = 1:len_n
            if n(ii) >= 0
                x_n(ii) = amplitude ^ n(ii);
            else
                x_n(ii) = 0;
            end
        end
    elseif type_seq == 5
        % Type 5: Double-Sided Exponential Decay (0 < a < 1)
        for ii = 1:len_n
            if n(ii) >= 0
                x_n(ii) = amplitude ^ n(ii);
            else
                x_n(ii) = amplitude ^ (-n(ii));
            end
        end
    elseif type_seq == 6
        % Type 6: Gate Function
        for ii = 1:len_n
            if abs(n(ii)) < 5
                x_n(ii) = amplitude;
            else
                x_n(ii) = 0;
            end
        end
    elseif type_seq == 7
        % Type 7: Sine Wave
        M = 1;
        N = 50;
        for ii = 1:len_n
            x_n(ii) = amplitude * sin((2 * pi * M / N) * n(ii));
        end
    elseif type_seq == 8
        M = 1;
        N = 15;
        j = M/N;
        for ii = 1:len_n
            if n(ii) == 0
                x_n(ii) = amplitude;
            else
  x_n(ii) = amplitude * sin((pi * j * n(ii))) / (pi * n(ii)* j);
            end
        end
end
```

Driver Codes:

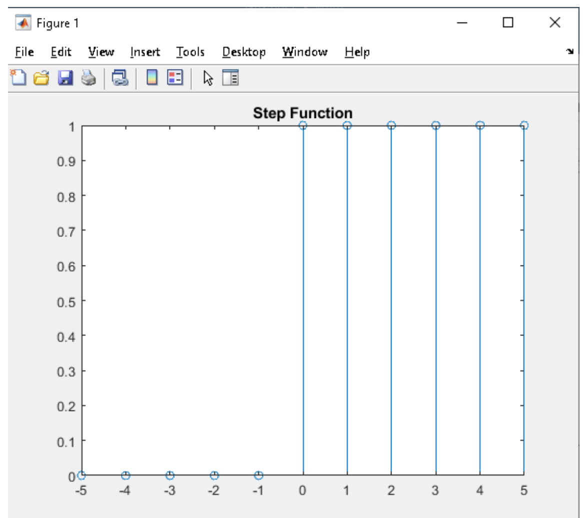
1)Delta Function (type_seq == 1):

```
n = -5:5; % Example range of n
amplitude = 1; % Example amplitude
x_n = seq_generator(n, 1, amplitude);
stem(n, x_n);
title('Delta Function');
```



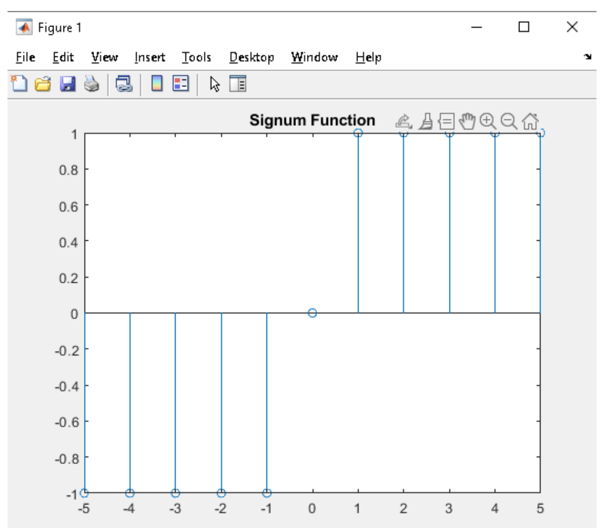
2) Step Function (type_seq == 2):

```
n = -5:5; % Example range of n
amplitude = 1; % Example amplitude
x_n = seq_generator(n, 2, amplitude);
stem(n, x_n);
title('Step Function');
```



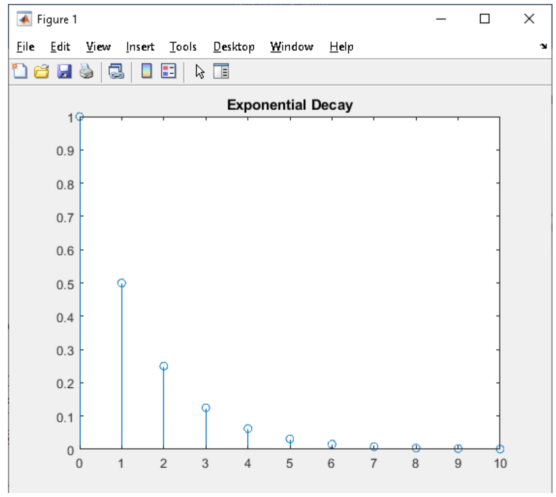
3) Signum Function (type_seq == 3):

```
n = -5:5; % Example range of n
amplitude = 1; % Example amplitude
x_n = seq_generator(n, 3, amplitude);
stem(n, x_n);
title('Signum Function');
```



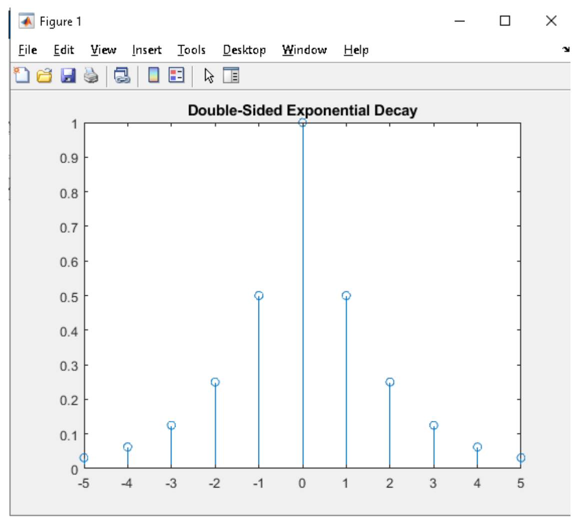
4) Exponential Decay (type_seq == 4):

n = 0:10;
amplitude = 0.5; % Example amplitude
x_n = seq_generator(n, 4, amplitude);
stem(n, x_n);
title('Exponential Decay');



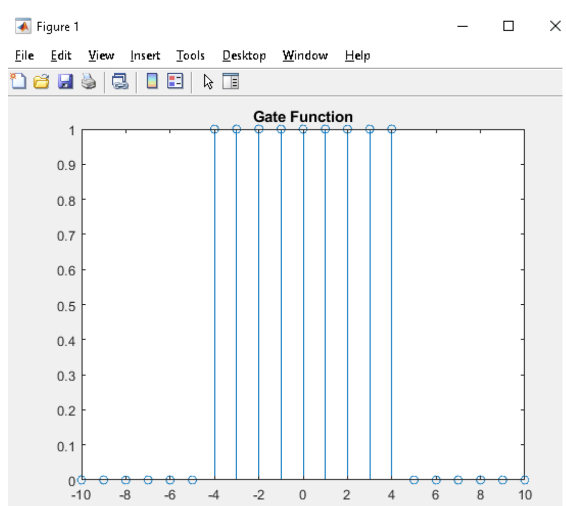
5) Double-Sided Exponential Decay (type_seq == 5):

n = -5:5; % Example range of n
amplitude = 0.5; % Example amplitude
x_n = seq_generator(n, 5, amplitude);
stem(n, x_n);
title('Double-Sided Exponential Decay');



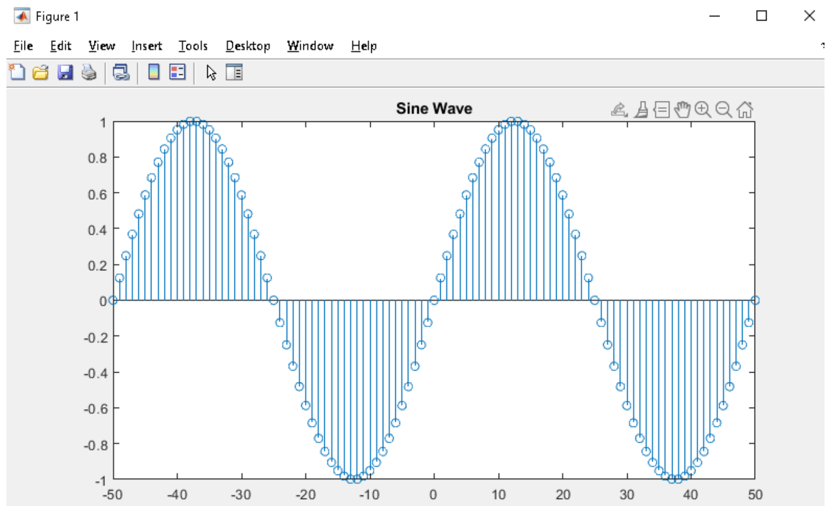
6) Gate Function (type_seq == 6):

n = -10:10; % Example range of n
amplitude = 1; % Example amplitude
x_n = seq_generator(n, 6, amplitude);
stem(n, x_n);
title('Gate Function');



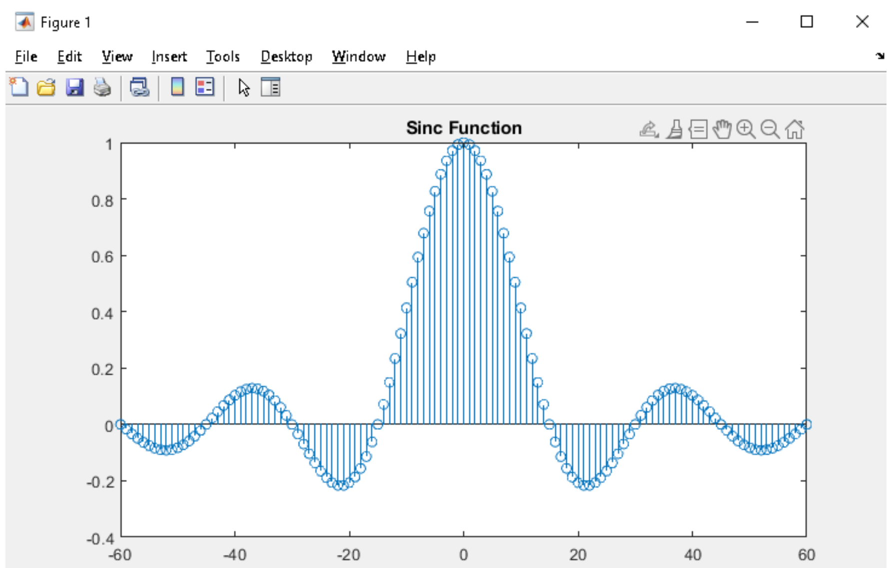
7) Sine Function (type_seq == 7):

n = -50:50; % Example range of n
amplitude = 1; % Example amplitude
x_n = seq_generator(n, 7, amplitude);
stem(n, x_n);
title('Sine Wave');



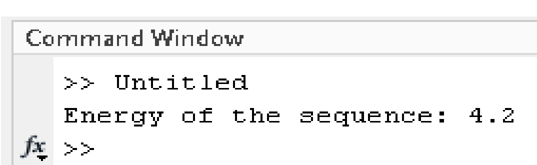
8) Sinc Function (type_seq == 8):

n = -60:60;
amplitude = 1;
x_n = seq_generator(n, 8, amplitude);
stem(n, x_n);
title('Sinc Function');



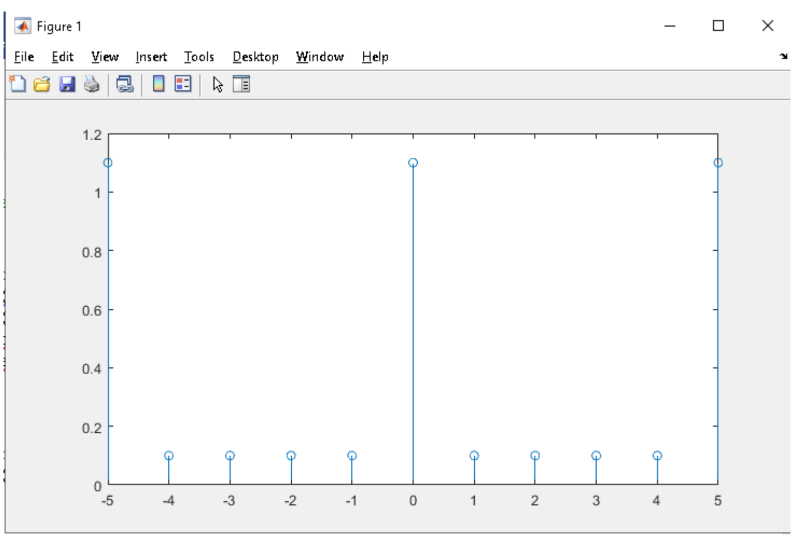
Example: Energy Calculation

```
function energy = Energy_Fx(data, step)
    data_squared = data.^2;
    energy = (step / 2) * (data_squared(1) + data_squared(end) + 2 *
sum(data_squared(2:end - 1)));
end

data = [1, 2, 3, 4, 5];
step = 0.1; % Step size
energy = Energy_Fx(data, step);
disp(['Energy of the sequence: ', num2str(energy)]);
```



Example: Fourier Series Coefficients Calculation

```
no_fourier_coff = 5; % Number of Fourier coefficients
k = -no_fourier_coff:1:no_fourier_coff; % Coefficient indices
tp = 1; % Period of the signal
ss = 0.1; % Sampling step
t = 0:ss:tp; % Time vector
len_t = length(t);
omega_not = (2 * pi) / tp;

% Create a square wave signal
f_t = zeros(1, len_t);
f_t(t <= 0.5) = 1;

% Plot the square wave
plot(t, f_t);

% Compute Fourier series coefficients
a_k = zeros(1, length(k));
for ii = 1:(no_fourier_coff * 2 + 1)
    t1 = f_t .* exp(-1j * omega_not * k(ii) * t);
    t2 = Energy_Fx(t1, ss) * 2;
    a_k(ii) = t2 / tp;
end

% Plot the magnitude of the Fourier coefficients
stem(k, abs(a_k));
```



Conclusion:

Thus, we successfully studied about plotting Discrete Time Signals, Energy Calculation and calculating Fourier coefficients in MATLAB.