FINAL REPORT

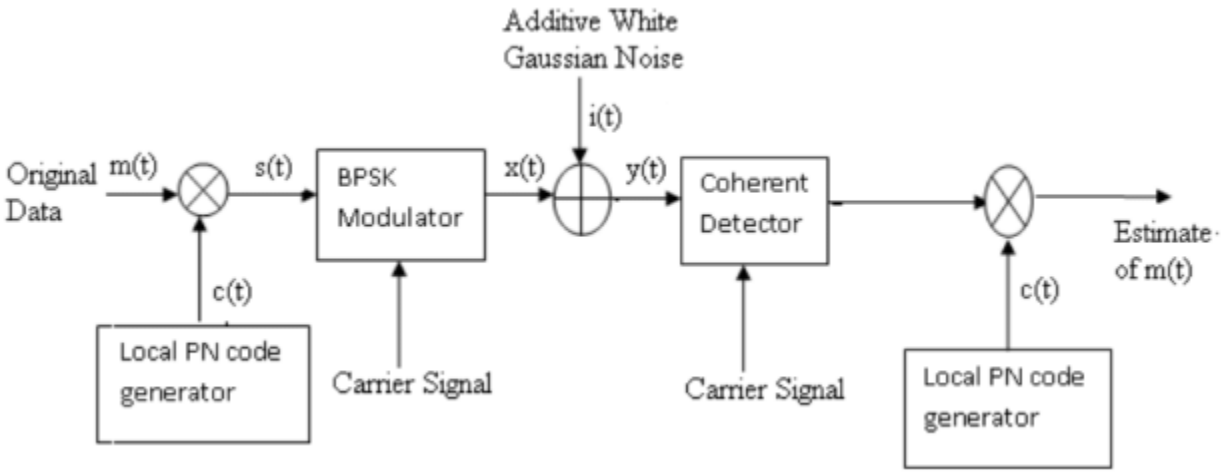
End to end simulation of DSSS-based radar system

:- Vidit Prakash

A. INTRODUCTION

The objective of this project is to develop an end-to-end simulation of a radar system based on Direct Sequence Spread Spectrum (DSSS). DSSS is a technique where the message signal is multiplied by a pseudo-random sequence, spreading its bandwidth so that the signal's energy is distributed across a broader spectrum, making it resemble noise. The motivation for using DSSS stems from the channel capacity theorem, which states that communication performance can be improved by increasing bandwidth, even when the signal-to-noise ratio is low.

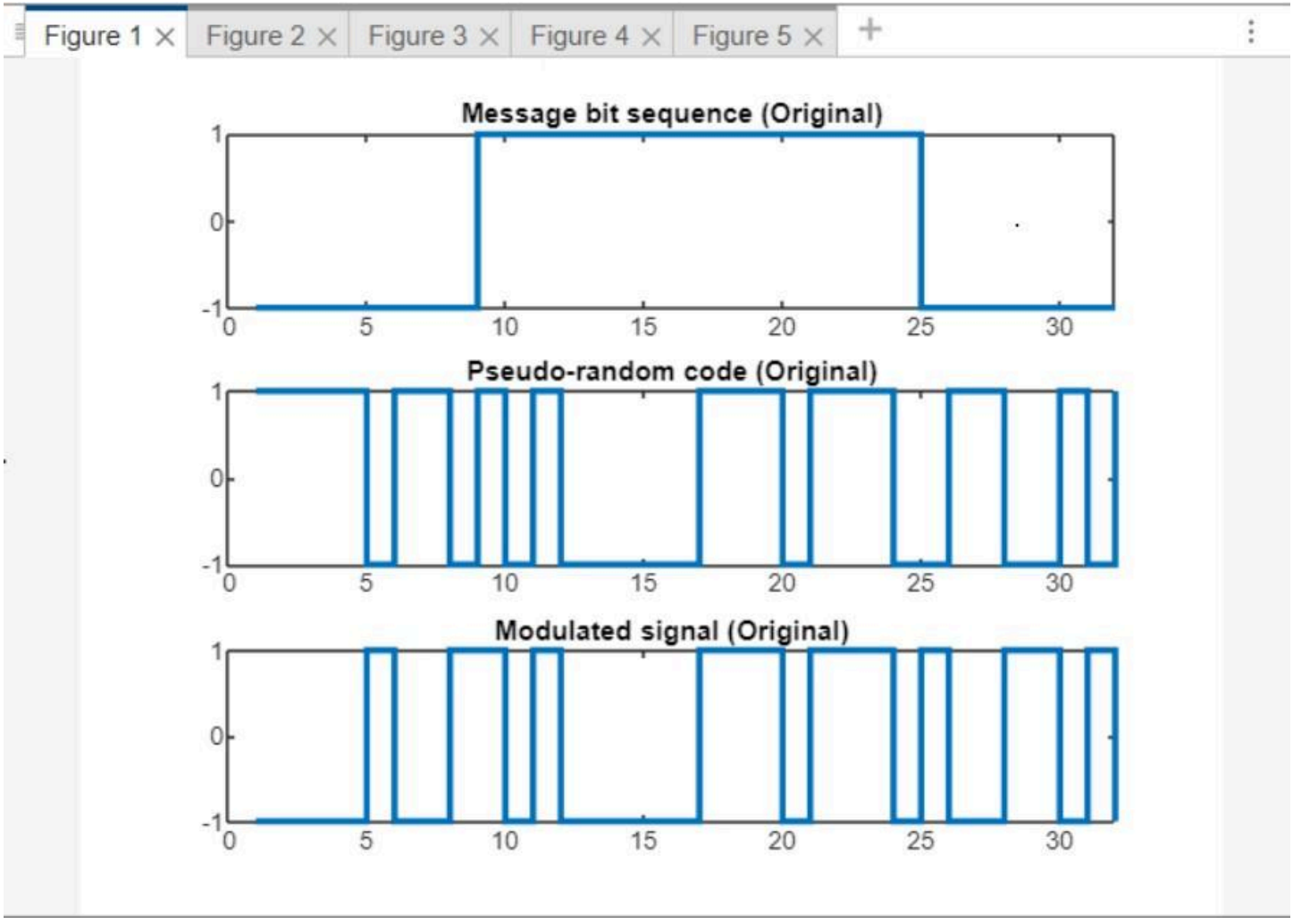
By expanding the transmit signal's bandwidth, DSSS helps mitigate intersymbol interference (ISI) and reduce the impact of narrowband interference. Another key advantage is the enhanced security of the transmission, as only receivers with the correct pseudo-random sequence can retrieve the original message signal. Without this sequence, the transmitted signal will appear as noise, making it impossible for unauthorized receivers to detect the message. These features make DSSS a suitable technique for radar systems.



B. Experiment

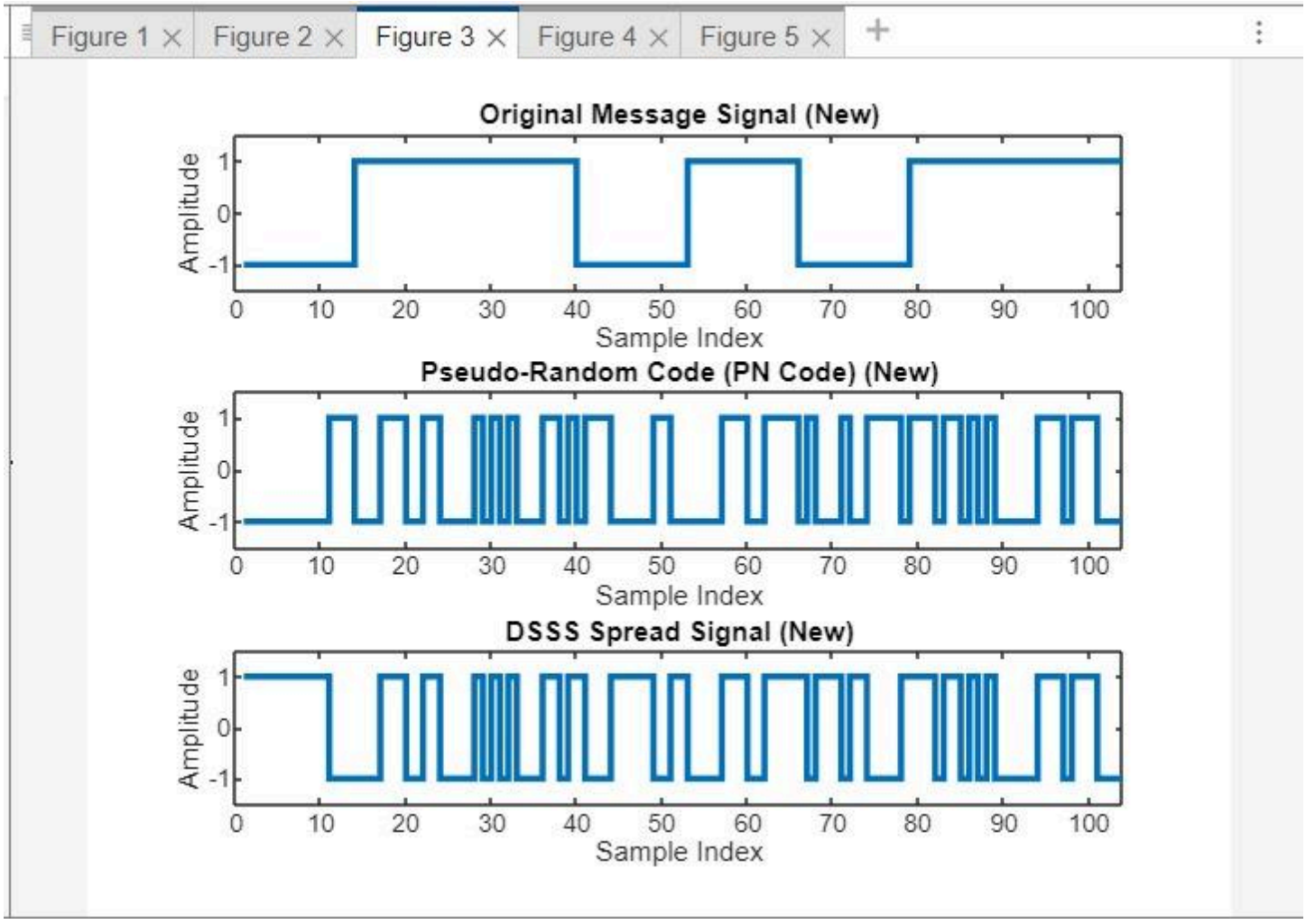
Signal Generation: Pseudo-random Code Generation: The first step is to create a pseudo-random binary sequence (PRBS), which serves as the spreading code. This sequence is typically a long series of ones and zeros, generated using techniques like Linear Feedback Shift Registers (LFSRs). The properties of the code, such as its

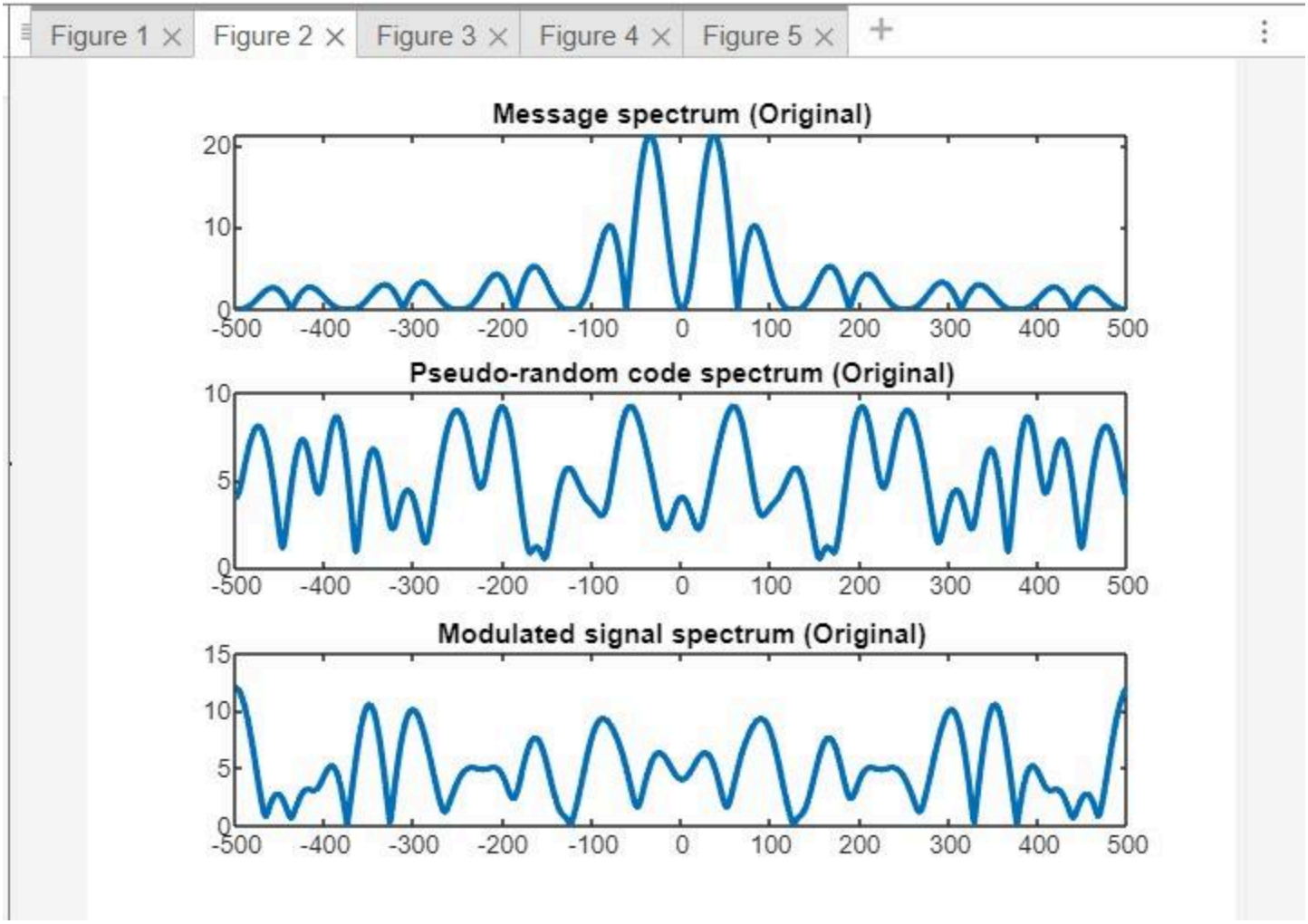
length and correlation characteristics, are important because they directly impact the radar's performance in terms of security and signal-to-noise ratio.
Modulation: The next step is to modulate the radar's message signal using this pseudo-random code. In DSSS, the message signal, which could be a simple carrier wave (e.g., a sine wave), is multiplied by the pseudo-random code. The modulation process spreads the signal's bandwidth over a wide range of frequencies, making it harder for narrowband interference or noise to affect the transmitted signal. The resulting modulated signal appears as noise to any receiver that doesn't know the pseudo-random sequence, ensuring transmission security.



Transmit Signal: Simulating Signal Transmission: Once the signal is modulated, it is ready for transmission. In this simulation, the transmitted radar signal propagates through the environment, typically assumed to be free space. The transmitted signal will travel at the speed of light, and its propagation can be modeled using the radar range equation. This equation considers factors such as the transmitted power,

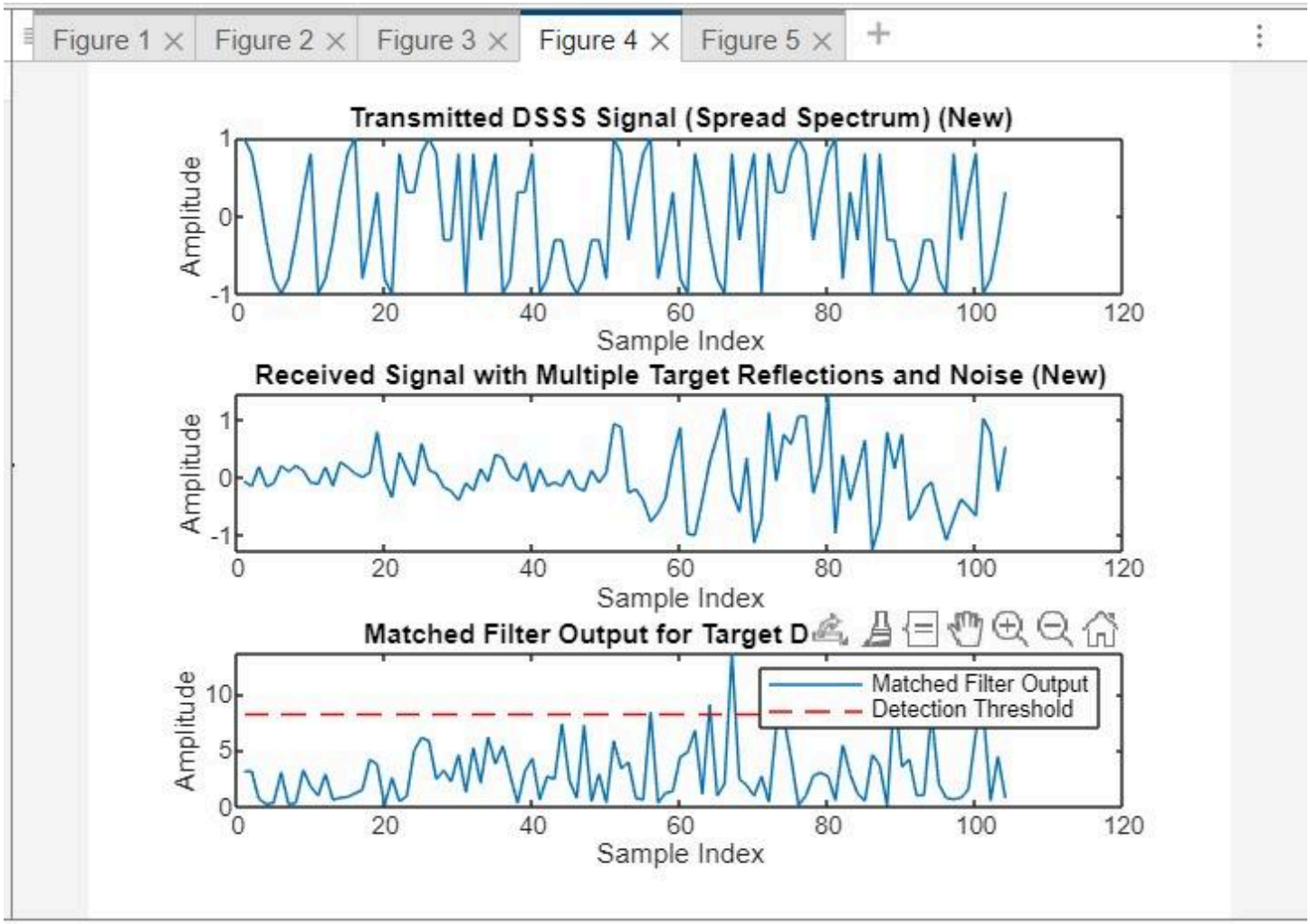
antenna gain, and the distance between the radar and the target. In this step, signal power loss due to free-space path loss can be simulated.
Transmit Antenna Characteristics: The radar's antenna characteristics, such as its directivity, beam width, and efficiency, should also be considered during transmission simulation. A more directional antenna will focus energy in a specific direction, enhancing target detection.





Target Reflection:

- **Modeling Target Reflection**: The transmitted signal is reflected off any object (target) present in the radar's detection range. The reflection depends on the target's size, shape, material, and radar cross-section (RCS). A larger or more reflective target will return a stronger signal to the radar receiver.
- **Doppler Effect**: If the target is moving, the Doppler effect should also be considered. This causes a shift in the frequency of the reflected signal based on the target's velocity relative to the radar, which can help estimate the target's speed.
- **Noise Addition**: To simulate real-world radar operation, noise is added to the reflected signal. This could be thermal noise, atmospheric noise, or interference from other systems. The noise is typically modeled as Gaussian noise and is combined with the reflected signal to simulate the noisy conditions radar systems operate in. The signal-to-noise ratio (SNR) plays a crucial role in determining how well the radar can detect the target in the presence of this noise.

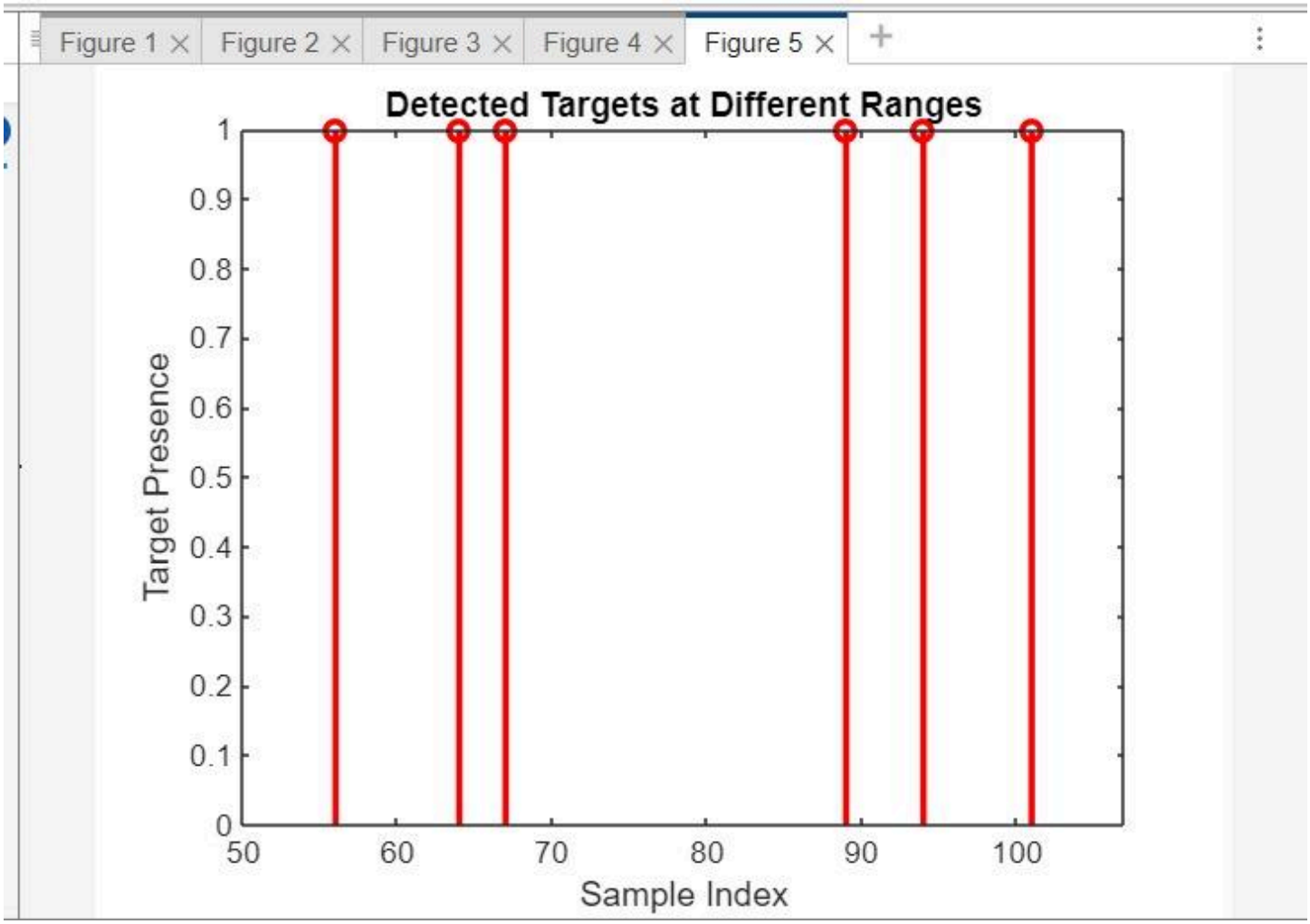


Despreading and Demodulation:

- **Despreading**: When the reflected signal, combined with noise, is received by the radar, the first step is to despread it. Despreading involves multiplying the received signal by the same pseudo-random code used during the signal generation phase. This operation compresses the spread spectrum back to its original bandwidth, allowing the radar to recover the message signal.
- **Demodulation**: After despreading, the demodulation process retrieves the underlying message signal from the received signal. Common modulation schemes used in DSSS radar systems include Binary Phase Shift Keying (BPSK) or Quadrature Phase Shift Keying (QPSK). Demodulation helps recover the phase and frequency information of the transmitted signal, which is crucial for identifying and processing the radar's target.

Target Detection:

- **Signal Processing for Target Detection**: The despread and demodulated signal is processed to detect the presence of a target. This involves performing various signal processing techniques such as matched filtering, pulse compression, and threshold detection.
    - **Matched Filtering**: The reflected signal is passed through a matched filter to maximize the signal-to-noise ratio and increase the likelihood of detecting weak reflections from distant or small targets.
    - **Pulse Compression**: If the radar uses pulse modulation, pulse compression techniques can be employed to improve range resolution. This allows the radar to distinguish between closely spaced targets by compressing the received pulse's duration while maintaining high energy.
    - **Thresholding and Detection**: A threshold is set to distinguish between noise and the actual target signal. If the received signal exceeds this threshold, the system determines that a target is present. False alarms (when noise exceeds the threshold) and missed detections (when the target signal does not exceed the threshold) must be carefully managed using techniques like constant false alarm rate (CFAR) detection.
- **Range and Velocity Estimation**: After target detection, the system estimates the range to the target by measuring the time delay between the transmitted and received signals. Additionally, if there is a Doppler shift, the target’s velocity can be estimated by analyzing the frequency shift in the reflected signal.



Command Window

New to MATLAB? See resources for Getting Started.

```
Detected Targets at Sample Indices:
    56    64    67    89    94   101

>>
```

C. CONCLUSION

1. **Signal Generation**: Create a pseudo-random code and use it to modulate the radar's message signal, spreading the signal across a wide frequency spectrum.
2. **Transmit Signal**: Simulate the radar transmitting this modulated signal into the environment.
3. **Target Reflection**: Model how this transmitted signal reflects off targets in the environment, taking into account the radar cross-section of the target and adding noise to simulate real-world conditions.

4. **Despreading and Demodulation**: At the receiver, despread the received signal using the same pseudo-random code and demodulate it to recover the original message signal.
5. **Target Detection**: Apply signal processing techniques, such as matched filtering and pulse compression, to identify the presence of a target and estimate its range and velocity.

By following these steps, a comprehensive simulation of a DSSS-based radar system can be developed, enabling the study of radar performance in various operational scenarios, including the presence of noise, interference, and moving targets.

D. Appendices

```
clear all; close all; clc;


%% Parameters
Fs = 1e6;              % Sampling frequency (1 MHz)
fc = 1e5;              % Carrier frequency (100 kHz)
bit_t = 1e-3;          % Bit duration (1 ms)
num_bits = 100;        % Number of bits in the pseudo-random code
SNR = 10;              % Signal-to-Noise Ratio in dB
num_targets = 3;       % Number of targets


%% Generate the Pseudo-Random Code (PN code) for DSSS
pn_code = randi([0, 1], 1, num_bits);  % Generate random bits
pn_code(pn_code == 0) = -1;            % Map 0s to -1 for BPSK modulation


%% Message Signal (DSSS) - Example message generation
```

```
message = [0 1 1 0 1 0 1 1];          % Example binary message
message(message == 0) = -1;           % Map 0s to -1 for BPSK modulation

% Adjust num_bits to ensure it's a multiple of message length
message_length = length(message);
if mod(num_bits, message_length) ~= 0
    num_bits = message_length * ceil(num_bits / message_length); % Adjust num_bits to the nearest multiple
end

% Expand the message to match the length required for spreading
message_signal = kron(message, ones(1, num_bits / message_length));

% Ensure pn_code matches the length of message_signal
if length(pn_code) < length(message_signal)
    pn_code = repmat(pn_code, 1, ceil(length(message_signal) / length(pn_code)));
end
pn_code = pn_code(1:length(message_signal));  % Truncate if necessary

spread_signal = message_signal .* pn_code;  % Spread the message using the PN code

%% Carrier Signal Generation
t = (0:length(spread_signal)-1) / Fs;  % Time vector
```

```
carrier = cos(2 * pi * fc * t);        % Carrier signal

%% Transmit the Spread Spectrum Signal
tx_signal = spread_signal .* carrier;  % Modulated transmit signal

%% Simulate the Received Signal with Multiple Target Reflections and Noise
delays = [50, 150, 300];  % Delays in samples representing the targets' distances
rx_signal = zeros(1, length(tx_signal));

% Simulate reflections from multiple targets
for i = 1:num_targets
    % Create the delayed signal by padding zeros at the beginning
    delayed_signal = [zeros(1, delays(i)), tx_signal];

    % Ensure the delayed signal has the same length as rx_signal
    delayed_signal = delayed_signal(1:length(rx_signal));

    % Add the delayed signal to the received signal
    rx_signal = rx_signal + delayed_signal;  % Accumulate reflections from each target
end

%% Add Noise to the Received Signal
% Calculate the power of the transmitted signal
signal_power = mean(abs(tx_signal).^2);
```

```
% Calculate the noise power based on the desired SNR
noise_power = signal_power / (10^(SNR/10));

% Generate Gaussian noise with the calculated noise power
noise = sqrt(noise_power) * randn(size(tx_signal));

% Add the noise to the received signal
noisy_signal = rx_signal + noise;

%% Despreading and Demodulation using a Matched Filter
matched_filter = fliplr(pn_code);  % Matched filter is the time-reversed PN code
demod_signal = noisy_signal .* carrier;  % Demodulate the received signal

% Perform matched filtering on the demodulated signal
matched_output = conv(demod_signal, matched_filter, 'same');

%% Original Code: DSSS Modulation (Retain Original Plots)
Fs_old = 1000; fc_old = 100; fp = 4; bit_t_old = 0.1;
m = [0 0 1 1 1 1 0 0];  % Original message
m(m == 0) = -1;         % BPSK mapping
message_old = repmat(m, fp, 1);
message_old = reshape(message_old, 1, []);
```

```
pn_code_old = randi([0, 1], 1, length(m) * fp);
pn_code_old(pn_code_old == 0) = -1;
DSSS_old = message_old .* pn_code_old;

%% Plot Time-Domain Signal, PN Code, and DSSS Signal from Original Code
figure;
subplot(3,1,1); stairs(message_old, 'linewidth', 2);
title('Message bit sequence (Original)');
axis([0 length(message_old) -1 1]);

subplot(3,1,2); stairs(pn_code_old, 'linewidth', 2);
title('Pseudo-random code (Original)');
axis([0 length(pn_code_old) -1 1]);

subplot(3,1,3); stairs(DSSS_old, 'linewidth', 2);
title('Modulated signal (Original)');
axis([0 length(DSSS_old) -1 1]);

%% Plot Frequency-Domain Signals from Original Code
f_old = linspace(-Fs_old/2, Fs_old/2, 1024);
figure;
subplot(3,1,1); plot(f_old, abs(fftshift(fft(message_old, 1024))), 'linewidth', 2);
title('Message spectrum (Original)');
```

```
subplot(3,1,2); plot(f_old, abs(fftshift(fft(pn_code_old, 1024))), 'linewidth', 2);
title('Pseudo-random code spectrum (Original)');

subplot(3,1,3); plot(f_old, abs(fftshift(fft(DSSS_old, 1024))), 'linewidth', 2);
title('Modulated signal spectrum (Original)');

%% DSSS Modulation Figures (New Code)
figure;
subplot(3, 1, 1);
stairs(message_signal, 'linewidth', 2);
title('Original Message Signal (New)');
xlabel('Sample Index'); ylabel('Amplitude');
axis([0 length(message_signal) -1.5 1.5]);

subplot(3, 1, 2);
stairs(pn_code, 'linewidth', 2);
title('Pseudo-Random Code (PN Code) (New)');
xlabel('Sample Index'); ylabel('Amplitude');
axis([0 length(pn_code) -1.5 1.5]);

subplot(3, 1, 3);
stairs(spread_signal, 'linewidth', 2);
title('DSSS Spread Signal (New)');
xlabel('Sample Index'); ylabel('Amplitude');
```

```
axis([0 length(spread_signal) -1.5 1.5]);

%% Matched Filter and Radar Target Detection Figures (New Code)
figure;
subplot(3, 1, 1);
plot(tx_signal);
title('Transmitted DSSS Signal (Spread Spectrum) (New)');
xlabel('Sample Index'); ylabel('Amplitude');

subplot(3, 1, 2);
plot(noisy_signal);
title('Received Signal with Multiple Target Reflections and Noise (New)');
xlabel('Sample Index'); ylabel('Amplitude');

subplot(3, 1, 3);
plot(abs(matched_output));
title('Matched Filter Output for Target Detection (New)');
xlabel('Sample Index'); ylabel('Amplitude');
hold on;
threshold = 0.6 * max(abs(matched_output));  % Threshold for target detection
plot([1 length(matched_output)], [threshold threshold], 'r--');
legend('Matched Filter Output', 'Detection Threshold');

%% Identify Targets Based on Threshold (New Code)
```

```
detected_targets = find(abs(matched_output) > threshold);
disp('Detected Targets at Sample Indices:');
disp(detected_targets);

% Visualize target positions
figure;
stem(detected_targets, ones(size(detected_targets)), 'r', 'LineWidth', 2);
title('Detected Targets at Different Ranges');
xlabel('Sample Index'); ylabel('Target Presence');
```