FMCW Chirp Doppler Radar System

#	Name	ID	Sec	B.N.
1	Kareem Ashraf Mostafa	91240568	3	3
2	Kareem Amr Mahmoud	91240574	3	5

Table of Contents

Introduction

Frequency Modulated Continuous Wave (FMCW) radar represents a fundamental approach to range and velocity measurement in radar systems. This technique continuously transmits frequency-modulated signals and analyzes the received echoes to determine both the distance and relative velocity of targets. The principle relies on measuring the frequency difference between transmitted and received signals, which encodes information about target parameters.

This report documents the analysis and simulation of an FMCW Chirp Doppler Radar system. The primary objective is to demonstrate the signal processing chain required to extract range and velocity information from radar returns. The system uses linear frequency modulation, commonly known as chirp signals, where the transmitted frequency increases linearly over time within each pulse repetition interval.

The processing methodology employs a two-dimensional approach: Fast Fourier Transform (FFT) operations are applied first along the fast-time axis to resolve range, and subsequently along the slow-time axis to resolve velocity. This dual-domain processing results in a range-velocity map that reveals target locations in both dimensions simultaneously.

The simulation examines system performance across three signal-to-noise ratio conditions: 5dB, 10dB, and 15dB. These scenarios illustrate how varying noise levels affect target detection and parameter estimation accuracy. Each processing stage—from signal generation through FFT analysis to final range-velocity mapping—is documented with corresponding visualizations to clarify the underlying principles and transformations occurring at each step.

Design Flow

The FMCW radar system implementation follows a sequential signal processing architecture consisting of five primary stages. The process begins with the TX Generator, which creates a series of linear frequency-modulated chirp signals according to specified radar parameters including bandwidth, chirp duration, and pulse repetition frequency. The RX Generator then models the received signal by introducing time delays and Doppler frequency shifts corresponding to target range and velocity, while adding white Gaussian noise to simulate realistic operating conditions at the desired SNR level.

The received signal matrix undergoes Fast-Axis FFT processing, where FFT is applied along each chirp (fast-time dimension) to extract range information and convert the time-domain signal into the frequency domain. This is followed by Slow-Axis FFT processing, which operates across multiple chirps (slow-time dimension) to extract velocity information through Doppler analysis. The output of this two-dimensional FFT processing produces a Range-Velocity Map, a two-dimensional representation where one axis corresponds to target range and the other to target velocity, with peak magnitudes indicating target locations. An optional target detection algorithm analyzes this map to automatically identify and extract the range and velocity parameters of detected targets. The entire processing chain transforms raw time-domain radar signals into interpretable target information, enabling both visualization and quantitative analysis of radar returns.

The design is formed from 9 source code files:

1. radar_param.m # defining design paramerers
2. tx_gen.m # generating the transmitted signal
3. rx_gen.m # generating the received signal (from the channel)
4. fast_fft.m # fast-axis FFT processing
5. slow_fft.m # slow-axis FFT processing
6. range_velocity.m # range-velocity diagram preparing
7. detection.m # BONUS part: auto detect target parameters
8. plt.m # plot all needed figures
9. main.m # running all of the previous in order

Design Parameters

1. Core Parameters:
   - {fmin, fmax} # Frequency threshold for signal BW (specs)
   - {fc, c, lambda} # Carrier frequency, speed of light, carrier wavelength
   - {Bw} # Signal bandwidth

```
%% Core Parameters
fmin = 76e9;                % Lower frequency limit (Hz)
fmax = 77e9;                % Upper frequency limit (Hz)
fc = (fmin + fmax) / 2;     % Carrier frequency (Hz)
Bw = fmax - fmin;           % Bandwidth (Hz)
c = 3e8;                    % Speed of light (m/s)
lambda = c / fc;            % Carrier wavelength (m)
```

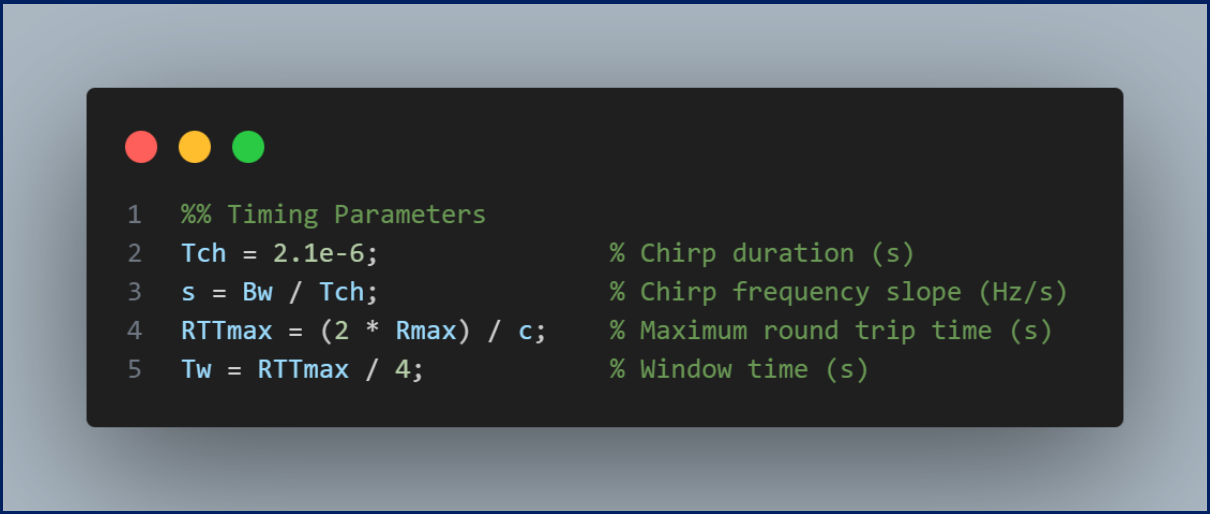
Figure 1: core-parameters

2. Resolution Parameters (specs)
   - {Rmax, Rmin} # Maximum unambiguous range, range resolution
   - {Vmax, Vmin} # Maximum unambiguous velocity, velocity resolution

```
%% Specs Parameters
Rmax = 250;                 % Maximum range (m)
Rmin = 0.75;                % Minimum range (m)
Vmax = 100;                 % Maximum speed (m/s)
Vmin = 0.5;                 % Minimum speed (m/s)
```

Figure 2: resolution-parameters

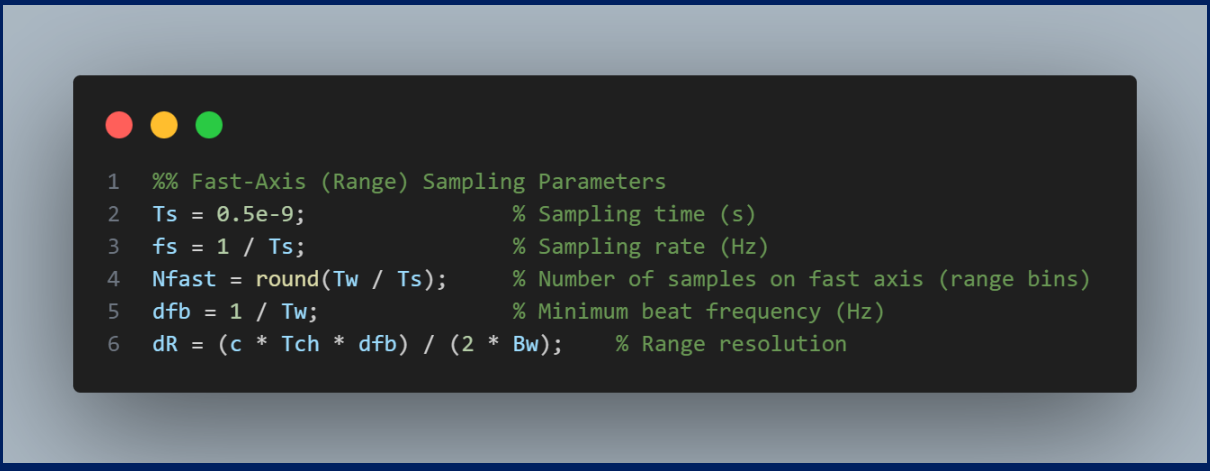
3. Timing Parameters
   - {RTTmax}
     # Maximum round trip corresponding to maximum target range = 2*Rmax/c = 1.67u
   - {Tw}
     # Processing window. As we know, dfbeat = (Bw*2*Rmin)/(Tch*c) then, dfbeat = 5/Tch.
     # Consider, dfbeat = 1/Tw = 5/Tch = 5/(RTTmax+Tw). Hence, we can get that,
     # Tw = RTTmax/4
   - {Tch}
     # Chirp time; it should be larger than {RTTmax+Tw}; larget than {1.25*1.67u}; larger than
     # {2.08u}; {2.1u}
   - {s} # Slope = Bw/Tch



```
%% Timing Parameters
Tch = 2.1e-6;               % Chirp duration (s)
s = Bw / Tch;               % Chirp frequency slope (Hz/s)
RTTmax = (2 * Rmax) / c;    % Maximum round trip time (s)
Tw = RTTmax / 4;            % Window time (s)
```

Figure 3: timing-parameters

4. Fast-Axis Parameters
   - {Ts, fs} # Sampling Time, sampling rate
   - {Nfast} # No. of samples = round(Tw/Ts)
   - {dfb, dR} # Min. beat freq., range resolution



```
%% Fast-Axis (Range) Sampling Parameters
Ts = 0.5e-9;                % Sampling time (s)
fs = 1 / Ts;                % Sampling rate (Hz)
Nfast = round(Tw / Ts);     % Number of samples on fast axis (range bins)
dfb = 1 / Tw;               % Minimum beat frequency (Hz)
dR = (c * Tch * dfb) / (2 * Bw);    % Range resolution
```

Figure 4: fast-axis-parameters

5. Slow-Axis Parameters
   - {PRF, PRI}
     # Pulse repetition freq., Pulse repetition interval; considering velocity resolution,
     # It should be less than 9.8u; let's take it {4*Tch} = {4*2.1u > 9.8u}
   - {N}
     # dfD = 1/Tw = 1/(N*PRI) = (2*Vmin*fc)/c -> N = 2^nexpow2(round(467)) = 512
   - {dfD, dV} # Min. Doppler freq., velocity resolution

```
%% Slow-Axis (Doppler) Parameters
PRI = 4 * Tch;              % Pulse repetition interval (s)
PRF = 1 / PRI;              % Pulse repetition frequency
N = 512;                    % Number of chirps (slow-time samples)
dfd = 1 / (N * PRI);        % Minimum Doppler frequency (Hz)
dV = (lambda * dfd) / 2;    % Velocity resolution
```

Figure 5: slow-axis-parameters

6. Noise Parameters
   - {SNR_dB} # Signal to noise ration in dB (5, 10, or 15)
   - {SNR_lin} # Linear SNR = 10^(SNR_dB/10)

```
%% Noise Parameters
SNR_dB = 15;                % SNR in dB
SNR_lin = 10^(SNR_dB/10);   % Linear SNR
```

Figure 6: noise-parameters

7. Targets Parameters
   - {A} # Signal amplitude (considering channel attenuation)
   - {R, V} # Chosen values
   - {RTT, fd} # RTT = 2*R/c, fd = 2*v/lambda

```
%% Target Parameters
target(1).A = 1e-3;          % Amplitude of target-1
target(1).R = 80;
target(1).V = 30;            % Velocity of target-1 (m/s) [approaching]
target(1).RTT = (2 * target(1).R) / c;      % RTT for target-1 (s)
target(1).fd = (2 * target(1).V) / lambda;  % Doppler for target-1 (Hz)

target(2).A = 1e-3;          % Amplitude of target-2
target(2).R = 40;
target(2).V = -50;           % Velocity of target-2 (m/s) [receding]
target(2).RTT = (2 * target(2).R) / c;      % RTT for target-2 (s)
target(2).fd = (2 * target(2).V) / lambda;  % Doppler for target-2 (Hz)
```

Figure 7: targets-parameters

Code Snippets

TX Generator

```
%% ========================================================================
%% 2. TX SIGNAL GENERATION
%% ========================================================================
fprintf('\n========== TX SIGNAL GENERATION ==========\n');
t_fast_plt = 0 : Ts : Tch;
t_fast = (0 : Nfast - 1).' * Ts;          % [Nfast x 1] fast time vector
tx_chirp_plt = exp(1j * 2*pi * (fmin * t_fast_plt + 0.5 * s * t_fast_plt.^2));
tx_chirp = exp(1j * 2*pi * (fmin * t_fast_plt + 0.5 * s * t_fast_plt.^2));
tx_sig = repmat(tx_chirp, 1, N);          % [Nfast x N] data matrix
```

Figure 8: tx_gen

This section generates the transmitted FMCW chirp signal used by the radar. A fast-time vector is defined based on the sampling period, and a complex baseband linear frequency modulated (LFM) chirp is generated using the starting frequency and chirp slope. The chirp is then replicated across multiple pulses to form a 2D transmit signal matrix, which represents consecutive FMCW chirps and serves as the input for subsequent range and Doppler processing.

RX Generator

```
%% ========================================================================
%% 3. RX SIGNAL GENERATION
%% ========================================================================

%% Signal Generation
num_targets = length(target);
rx_sig = zeros(Nfast, N);         % Beat signal [Nfast x N]
rx_chirp = zeros(Nfast, 1);       % First beat chirp

for m = 1:N
  for k = 1:num_targets
    % Target parameters
    R = target(k).R;          % Range in meters
    V = target(k).V;          % Velocity in m/s
    A = target(k).A;          % Amplitude
    RTT = target(k).RTT;      % Round Trip Time
    f_beat = s * RTT;         % Beat frequency
    f_d = 2 * V / lambda;
    beat_signal = A * exp(1j * 2*pi * (f_beat * t_fast + f_d * (m-1) * PRI));
    rx_sig(:, m) = rx_sig(:, m) + beat_signal;
  end
end
```

Figure 9: rx_gen-signal-generation

This part models the received beat signal by superimposing the contributions of all targets across multiple chirps. For each target, the beat frequency is determined by the round-trip delay, while the Doppler shift accounts for target velocity across slow time. The resulting complex beat signals are accumulated to form a 2D received signal matrix suitable for range–Doppler processing.

```
%% Adding Gaussian Noise
signal_power = mean(abs(rx_sig(:)).^2);
noise_power = signal_power / SNR_lin;
noise = sqrt(noise_power/2) * (randn(size(rx_sig)) + 1j*randn(size(rx_sig)));
rx_sig = rx_sig + noise;
```

Figure 10: rx_gen-adding-noise

In this section, complex additive white Gaussian noise (AWGN) is introduced to the received signal to simulate realistic radar channel conditions. The noise power is computed from the signal power and a predefined SNR, ensuring controlled and physically meaningful noise levels in the simulation.

```
%% Generate DELAYED TX SIGNAL
rx_chirp_plot = zeros(Nfast, 1);
for k = 1:num_targets
  A = target(k).A;
  R = target(k).R;
  V = target(k).V;
  RTT = target(k).RTT;
  f_d = 2 * V / lambda;
  t_delayed = t_fast - RTT;
  % Generate delayed TX with Doppler shift
  for n = 1:Nfast
    if t_delayed(n) >= 0 && t_delayed(n) <= Tch
      rx_chirp_plot(n) = rx_chirp_plot(n) + A * exp(1j * 2*pi * ...
        (fmin * t_delayed(n) + 0.5 * s * t_delayed(n)^2 + ...
        f_d * t_fast(n)));
    end
  end
end

% Add Gaussian Noise to it
rx_chirp_plot = rx_chirp_plot + noise(:, 1);
```

Figure 11: rx_gen-delayed-tx-generation

This part reconstructs the time-delayed transmitted chirp as received from each target. The signal is delayed by the round-trip time and modulated with a Doppler frequency shift to reflect target motion. This delayed chirp is mainly used for visualization and verification of the time-domain received waveform.

Fast-Axis FFT

```
%% ========================================================================
%% 4. Fast-Axis FFT (Fbeat detection)
%% ========================================================================
first_chirp = rx_sig(:, 1);     % Taking only the firts column
window_range = hamming(Nfast);
windowed_chirp = first_chirp .* window_range;

% Apply FFT
range_fft = fft(windowed_chirp, Nfast);
range_fft_mag = abs(range_fft);

fbeat = (0 : Nfast-1) * (fs / Nfast);     % Beat Frequency Axis
R_fft = (c * Tch * fbeat) / (2 * Bw);     % Range Axis

% taking first half positive frequencies
Nhalf = floor(Nfast/2);
fbeat_half = fbeat(1 : Nhalf);
R_fft_half = R_fft(1 : Nhalf);
range_fft_mag_half = range_fft_mag(1 : Nhalf);
```

Figure 12: fast-fft

In this section, the beat signal of the first received chirp is processed along the fast-time axis to extract range information. A Hamming window is applied to reduce spectral leakage before performing the FFT. The resulting beat frequency spectrum is then mapped to physical range using the FMCW radar range equation, and only the positive frequency components are retained since they contain the valid target range information.

Slow-Axis FFT

```
%% ========================================================================
%% 5. Slow-Axis FFT (FDoppler detection)
%% ========================================================================
[~, max_range_bin] = max(range_fft_mag_half);
range_bin_idx = max_range_bin;
slow_time_signal = rx_sig(range_bin_idx, :);

window_doppler = hamming(N);
windowed_slow = slow_time_signal(:) .* window_doppler;

% Apply FFT (to get fD)
doppler_fft = fft(windowed_slow, N);
doppler_fft_mag = abs(doppler_fft);

% Centerlized around zero
doppler_fft_mag_shifted = fftshift(doppler_fft_mag);

doppler_freq = (-N/2 : N/2-1) * (PRF / N);      % Doppler Frequency Axis
velocity = (lambda * doppler_freq) / 2;         % Velocity axis
```

Figure 13: slow-fft

This section extracts target velocity information by performing FFT processing along the slow-time axis. The range bin corresponding to the strongest target is first selected, and a Hamming window is applied to reduce Doppler spectral leakage. The FFT output is then centered around zero frequency and mapped to physical velocity using the Doppler relationship, enabling accurate target speed estimation.

Range-Velocity Map

```
%% ========================================================================
%% 6. Range Velocity Diagram
%% ========================================================================
window_2d_range = hamming(Nfast);
window_2d_doppler = hamming(N);
window_2d = window_2d_range .* window_2d_doppler';
% Apply windowing
rx_windowed = rx_sig .* window_2d;
% Range FFT Fast-Axis
range_fft_2d = fft(rx_windowed, Nfast, 1);

% Doppler FFT Slow-Axis
doppler_fft_2d = fft(range_fft_2d, N, 2);
rd_matrix = fftshift(doppler_fft_2d, 2);
rd_mag = abs(rd_matrix);

% Range axis
range_res = c / (2 * Bw);  % Theoretical range resolution
range_axis = (0:Nfast-1) * (c/(2*Bw)) * (fs/Nfast) * Tch;

% Velocity axis (after fftshift)
velocity_bins = -N/2:N/2-1;
velocity_axis = velocity_bins * (lambda * PRF) / (2 * N);

% Limit to reasonable ranges for display
range_max_display = 250;  % meters
velocity_max_display = 100;  % m/s

% Find indices for display limits
range_display_idx = find(range_axis <= range_max_display, 1, 'last');
vel_display_idx_pos = find(velocity_axis <= velocity_max_display, 1, 'last');
vel_display_idx_neg = find(velocity_axis >= -velocity_max_display, 1, 'first');

% Extract subset for display
range_axis_display = range_axis(1:range_display_idx);
velocity_axis_display = velocity_axis(vel_display_idx_neg:vel_display_idx_pos);
rd_mag_display = rd_mag(1:range_display_idx, vel_display_idx_neg:vel_display_idx_pos);
```

Figure 14: range-velocity

This section constructs the range–velocity (Range–Doppler) diagram by applying 2D windowing followed by FFT processing along both fast-time and slow-time axes. The fast-axis FFT extracts range information, while the slow-axis FFT provides Doppler (velocity) separation, with the spectrum centered using FFT shift. The resulting magnitude matrix is mapped to physical range and velocity axes and is limited to practical display bounds for clear visualization of detected targets.

Bonus: Target Parameters Detection

```
%% ========================================================================
%% 7. Bonus-Section (Detection of Ranges & Velocites)
%% ========================================================================
% Find peaks for fast-axis
[peaks, peak_indices] = findpeaks(range_fft_mag_half, ...
  'SortStr', 'descend', 'NPeaks', 2, ...
  'MinPeakHeight', max(range_fft_mag_half)*0.1);

% Find peaks for slow-axis
[doppler_peaks, doppler_indices] = findpeaks(doppler_fft_mag_shifted, ...
  'SortStr', 'descend', 'NPeaks', 2, ...
  'MinPeakHeight', max(doppler_fft_mag_shifted)*0.1);

detected_targets = struct([]);

for k = 1:length(target)
  % Find nearest bins to expected target location
  [~, range_idx] = min(abs(range_axis_display - target(k).R));
  [~, vel_idx]   = min(abs(velocity_axis_display - target(k).V));
  % Local search window (bins)
  search_range = 8;
  search_vel   = 8;
  range_start = max(1, range_idx - search_range);
  range_end   = min(length(range_axis_display), range_idx + search_range);
  vel_start   = max(1, vel_idx   - search_vel);
  vel_end     = min(length(velocity_axis_display), vel_idx + search_vel);
  % Extract local Range-Doppler region
  search_region = rd_mag_display(range_start:range_end, ...
    vel_start:vel_end);
  % Find strongest peak in local region
  [max_val, max_idx] = max(search_region(:));
  [local_r, local_v] = ind2sub(size(search_region), max_idx);
  % Convert to global indices
  global_r = range_start + local_r - 1;
  global_v = vel_start   + local_v - 1;
  % Store detected parameters
  detected_targets(k).R = range_axis_display(global_r);
  detected_targets(k).V = velocity_axis_display(global_v);
  detected_targets(k).magnitude = max_val;
end

```

Figure 15: detection

This section performs target detection by identifying dominant peaks in the range and Doppler spectra. Peak detection is first used to locate strong candidates along each axis, after which a localized search is carried out in the range–Doppler map around the expected target positions. The strongest response within each local region is selected, and the corresponding range and velocity estimates are extracted and stored as detected target parameters.

Plot Results

```
%% 1. TX Signal
t_fast_plt = 0 : Ts : Tch;
figure("Name", "TX");
plot(t_fast_plt*1e6, real(tx_chirp_plt), 'b', 'LineWidth', 1.5);
xlabel('Time (µs)');
ylabel('Amplitude');
title('TX FMCW Chirp (Real Part)');
grid on;
```

Figure 16: plt-tx

This plot shows the real part of the transmitted FMCW chirp over one chirp duration, verifying the linear frequency modulation in the time domain.

```
%% 2. RX Signal (Delayed TX)
zoom_samples = 1:min(20000, length(t_fast));
figure("Name", "RX");
plot(t_fast(zoom_samples)*1e6, real(rx_chirp_plot(zoom_samples)), 'b', 'LineWidth', 1.5);
xlabel('Time (µs)');
ylabel('Amplitude');
title('RX FMCW Chirp (Real Part - Delayed TX)');
grid on;

```

Figure 17: plt-rx

This plot illustrates the received time-delayed chirp, confirming the effect of target range and Doppler on the transmitted signal.

```
%% 3. Fast-Axis FFT (Range Profile)
figure("Name", "FastAxis_FFT");
plot(R_fft_half, range_fft_mag_half, 'b', 'LineWidth', 1.5);
xlabel("Range (m)");
ylabel("Magnitude");
title("Fast-Axis FFT (Range Domain)");
grid on;
xlim([0, 250]);

% Mark peaks
hold on;
for i = 1:min(2, length(peak_indices))
  R_detected = R_fft_half(peak_indices(i));
  f_beat_detected = fbeat_half(peak_indices(i));
  plot(R_detected, peaks(i), 'ro', 'MarkerSize', 10, 'LineWidth', 2);
  text(R_detected, peaks(i)*1.05, ...
    sprintf('R=%.1f m\nf_b=%.1f MHz', R_detected, f_beat_detected/1e6), ...
    'FontSize', 8, 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');
end

% Mark expected ranges
for k = 1:length(target)
  xline(target(k).R, 'g--', 'LineWidth', 1);
end
hold off;
legend('Range Profile', 'Detected Peaks', 'Expected Ranges', 'Location', 'northeast');

```

Figure 18: plt-fast

This figure shows the magnitude of the fast-time FFT applied to the received beat signal, producing the target range profile. Dominant peaks correspond to detected targets, while annotations highlight the estimated beat frequencies and ranges. Expected target ranges are overlaid to validate the accuracy of range detection.

```
%% 4. Slow-Axis FFT (Velocity Profile)
figure("Name", "SlowAxis_FFT");
plot(velocity, doppler_fft_mag_shifted, 'b', 'LineWidth', 1.5);
xlabel("Velocity (m/s)");
ylabel("Magnitude");
title("Slow-Axis FFT (Velocity Domain)");
grid on;
xlim([-100, 100]);

% Mark peaks
hold on;
for i = 1:min(2, length(doppler_indices))
  v_detected = velocity(doppler_indices(i));
  f_doppler_detected = doppler_freq(doppler_indices(i));
  plot(v_detected, doppler_peaks(i), 'ro', 'MarkerSize', 10, 'LineWidth', 2);
  text(v_detected, doppler_peaks(i)*1.05, ...
    sprintf('v=%.1f m/s\nf_d=%.1f kHz', v_detected, f_doppler_detected/1e3), ...
    'FontSize', 8, 'HorizontalAlignment', 'center', 'BackgroundColor', 'white');
end

% Mark expected velocities
for k = 1:length(target)
  xline(target(k).V, 'g--', 'LineWidth', 1);
end
hold off;
legend('Doppler Spectrum', 'Detected Peaks', 'Expected Velocities', 'Location', 'northeast');
```

Figure 19: plt-slow

This plot presents the Doppler spectrum obtained by applying FFT along the slow-time axis at the selected range bin. Peak locations indicate target velocities after Doppler frequency-to-velocity mapping. Expected velocities are marked to assess detection performance.

```
%% 5. Range-Velocity Diagram (2D FFT)
figure("Name", "2D");
imagesc(velocity_axis_display, range_axis_display, 20*log10(rd_mag_display + eps));
xlabel('Velocity (m/s)');
ylabel('Range (m)');
title('Range-Velocity Diagram (Range-Doppler Map)');
ylim([0, max(range_axis_display)]);
colorbar;
axis xy;
colormap('jet');
clim([-20, max(20*log10(rd_mag_display(:) + eps))]);

% Mark targets
hold on;
for k = 1:length(target)
  % Expected (red x)
  plot(target(k).V, target(k).R, 'rx', 'MarkerSize', 12, 'LineWidth', 2);

  % Detected (yellow circle)
  if k <= length(detected_targets)
    plot(detected_targets(k).V, detected_targets(k).R, 'yo', ...
      'MarkerSize', 10, 'LineWidth', 2, 'MarkerFaceColor', 'yellow');
    text(detected_targets(k).V, detected_targets(k).R + 2, ...
      sprintf('T%d\nR=%.1f m\nv=%.1f m/s', k, detected_targets(k).R, detected_targets(k).V), ...
      'FontSize', 8, 'FontWeight', 'bold', 'HorizontalAlignment', 'center', ...
      'BackgroundColor', 'white', 'EdgeColor', 'black');
  end
end
hold off;
grid on;

```

Figure 20: plt-2D

This 2D range–Doppler map jointly represents target range and velocity by combining fast-time and slow-time FFT processing. The magnitude response reveals target separability in both dimensions, with detected and expected target positions overlaid. This visualization confirms correct range–velocity coupling and target resolution.

```
%% 6. Print Results
fprintf('\n========== DETECTION RESULTS ==========\n');
fprintf('\nRange-Velocity Map Detection:\n');

for k = 1:length(detected_targets)
  fprintf('  Target %d:\n', k);
  fprintf('    Detected  R = %.2f m,  V = %.2f m/s\n', ...
    detected_targets(k).R, detected_targets(k).V);
  fprintf('    Expected  R = %.2f m,  V = %.2f m/s\n', ...
    target(k).R, target(k).V);
  fprintf('    Errors:   ΔR = %.3f m, ΔV = %.3f m/s\n', ...
    abs(detected_targets(k).R - target(k).R), ...
    abs(detected_targets(k).V - target(k).V));
end

fprintf('\n========== SUMMARY ==========\n');
fprintf('Range Resolution    : %.3f m\n', dR);
fprintf('Velocity Resolution : %.3f m/s\n', dV);
fprintf('Max Range           : %.1f m\n', Rmax);
fprintf('Max Velocity        : %.1f m/s\n', Vmax);
```

Figure 21: print results

This section numerically reports the detected range and velocity of each target and compares them with the ground-truth values. Detection errors are computed to quantify estimation accuracy. System performance metrics such as resolution and maximum detectable range and velocity are also summarized.

Main Program

```
clear; close all; clc;

run radar_param;
run tx_gen;
run rx_gen;
run fast_fft;
run slow_fft;
run range_velocity;
run detection;
run plt;
```

Figure 22: main

This script serves as the top-level controller for the FMCW radar simulation. It sequentially executes parameter initialization, transmit and receive signal generation, range and Doppler processing, target detection, and result visualization, ensuring a structured and modular radar signal processing workflow.

Results

In this part of the report, we will go through the results and plotting the desired signals when SNR value in dB = (5, 10, or 15)

SNR = 5dB

TX Signal

- As shown the frequency of the chirping signal increases over time.
- Note: TX signal will not chang over different SNR values.

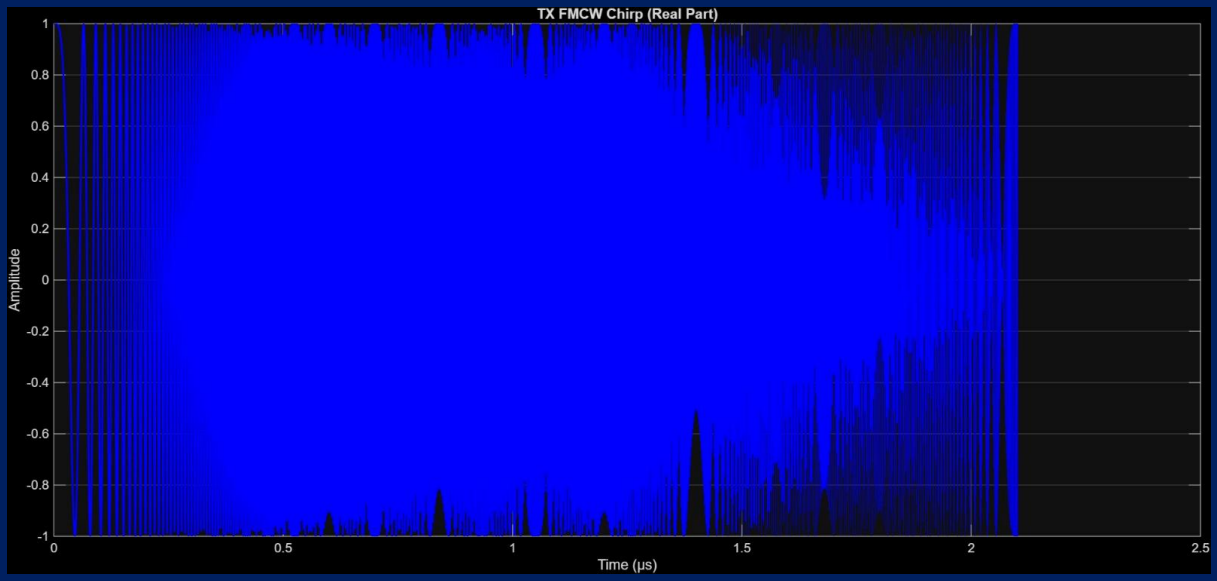


Figure 23: tx-5dB

RX Signal

- RX signal before the window time, is only Gaussian noise as we add noise to 0
- Once hitting window interval, the frequency of the signal also increases over time.

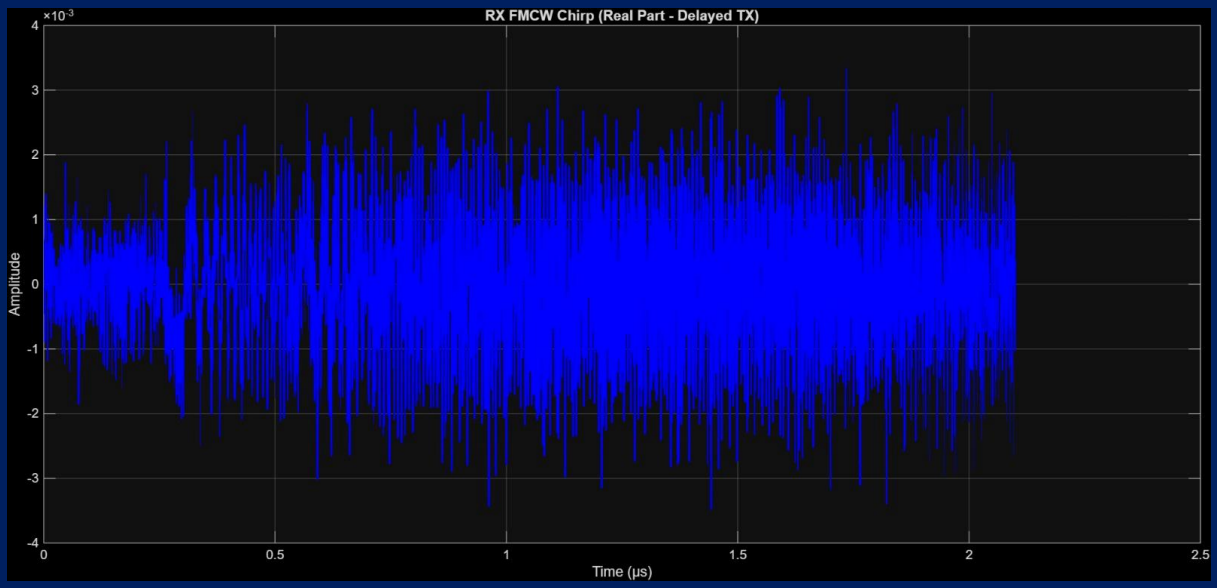


Figure 24: rx-5dB

Fast-Axis FFT

- FFT pins show the beat frequency and range of each target correctly (as defined).
- Noise is clear but not critical as the amplitudes are big enough.

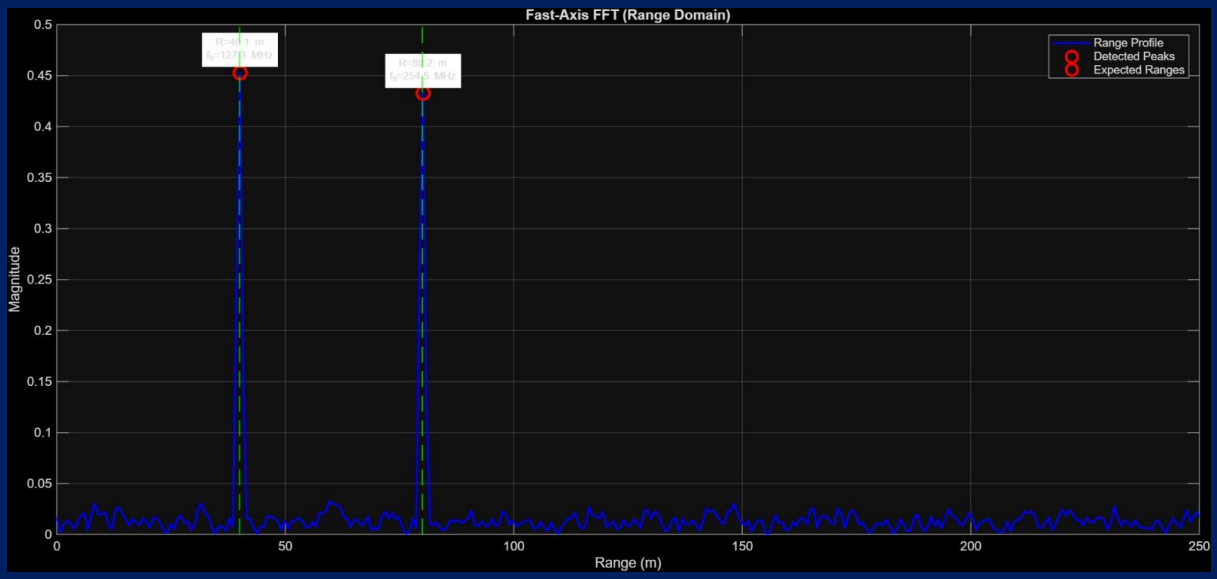


Figure 25: fast-5dB

Slow-Axis FFT

- FFT pins show Doppler frequency and velocity of each target correctly (as defined).
- Noise is clear but not critical as the amplitudes are big enough.

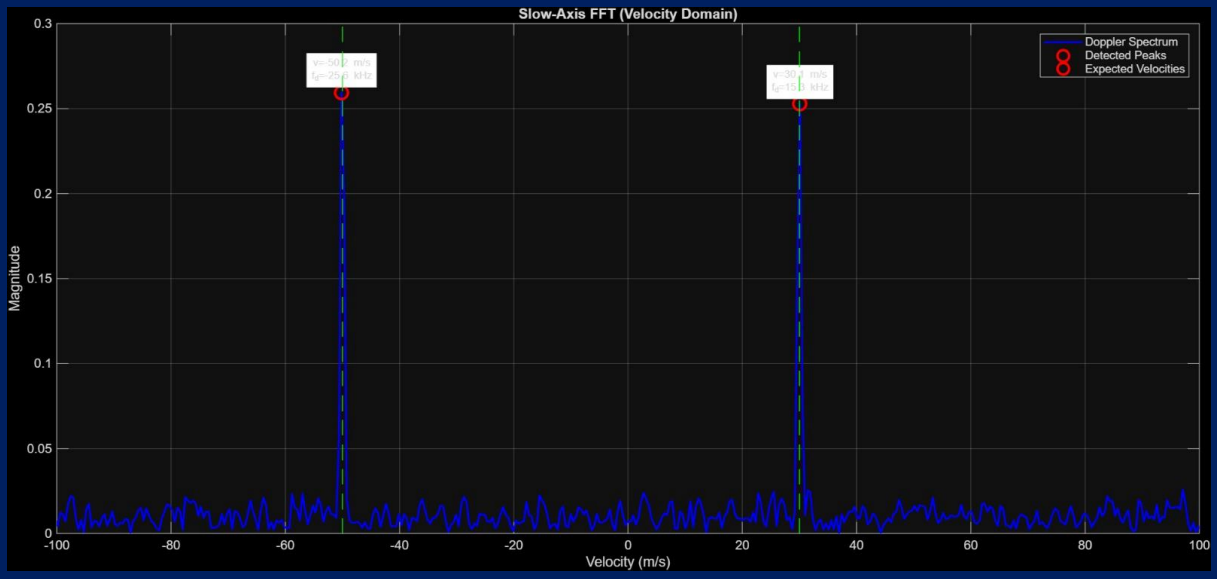


Figure 26: slow-5dB

Range-Velocity Map

- Targets parameters are shown correctly.

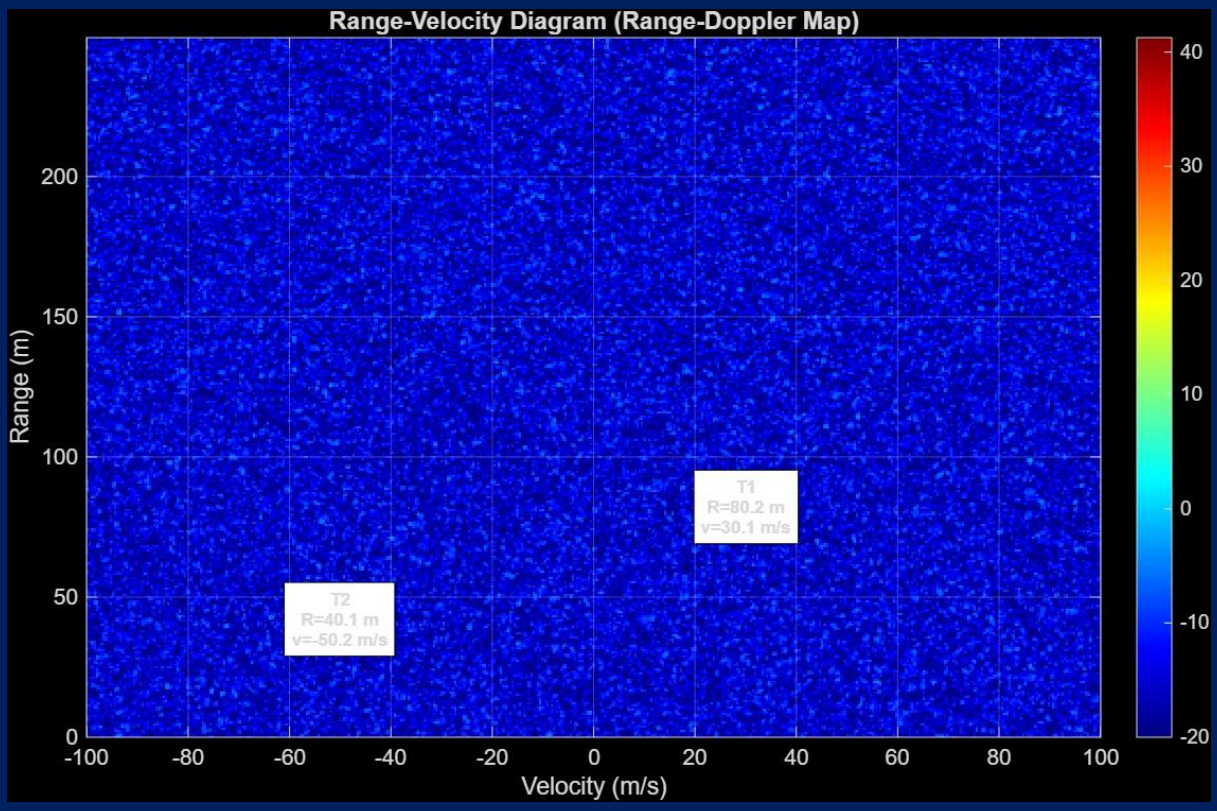


Figure 27: 2D-5dB

Bonus: Terminal Summary

- Terminal shows Range-Velocity Map Detection results for each target.
- Terminal shows a summary of range & velocity minimum and maximum unambiguous values

```
SNR: 5.0 dB

========== DETECTION RESULTS ==========

Range-Velocity Map Detection:
  Target 1:
    Detected  R = 80.17 m,  V = 30.09 m/s
    Expected  R = 80.00 m,  V = 30.00 m/s
    Errors:   ΔR = 0.168 m, ΔV = 0.090 m/s
  Target 2:
    Detected  R = 40.08 m,  V = -50.15 m/s
    Expected  R = 40.00 m,  V = -50.00 m/s
    Errors:   ΔR = 0.084 m, ΔV = 0.150 m/s

========== SUMMARY ==========
Range Resolution    : 0.756 m
Velocity Resolution : 0.456 m/s
Max Range           : 250.0 m
Max Velocity        : 100.0 m/s
```

Figure 28: summary-5dB

SNR = 10dB

TX Signal

- As shown the frequency of the chirping signal increases over time.
- Note: TX signal will not chang over different SNR values.

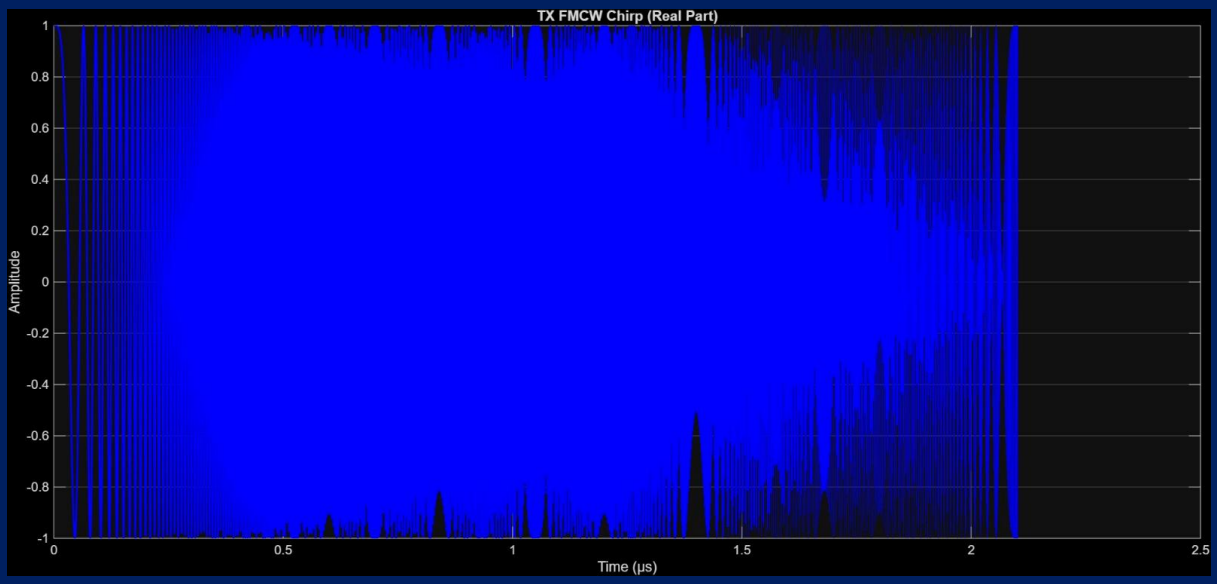


Figure 29: tx-5dB

RX Signal

- RX signal before the window time, is only Gaussian noise as we add noise to 0
- Once hitting window interval, the frequency of the signal also increases over time.
- It's clear that this RX signal is neat more than in 5dB results

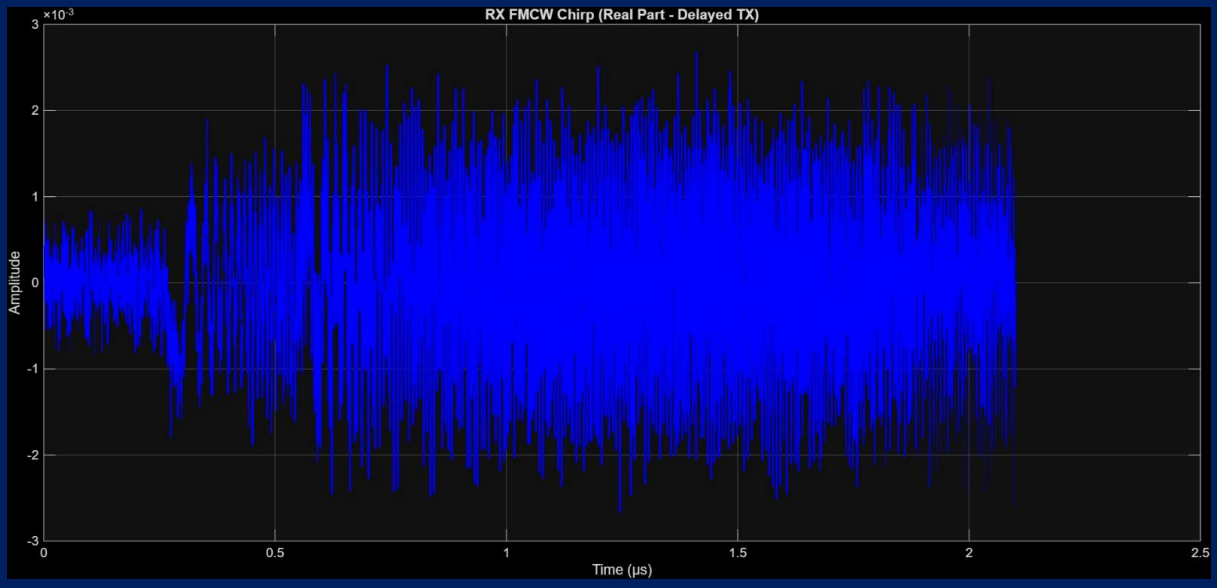


Figure 30: rx-10dB

Fast-Axis FFT

- FFT pins show the beat frequency and range of each target correctly (as defined).
- Noise is less than in 5dB results but it's also not critical

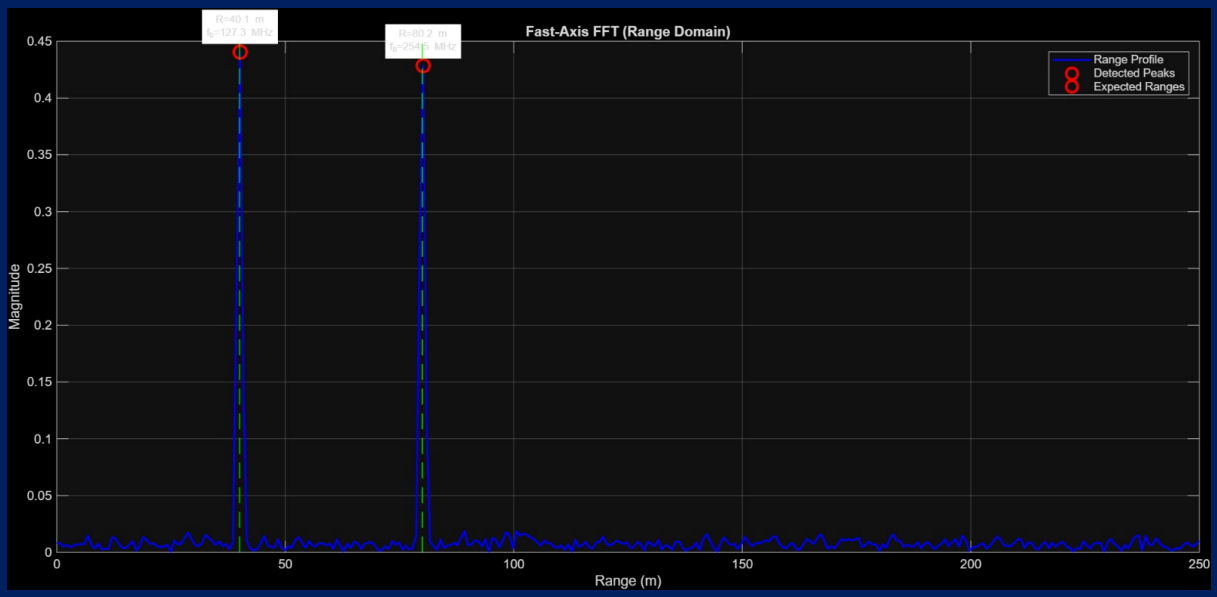


Figure 31: fast-10dB

Slow-Axis FFT

- FFT pins show Doppler frequency and velocity of each target correctly (as defined).
- Noise is less than in 5dB results but it's also not critical

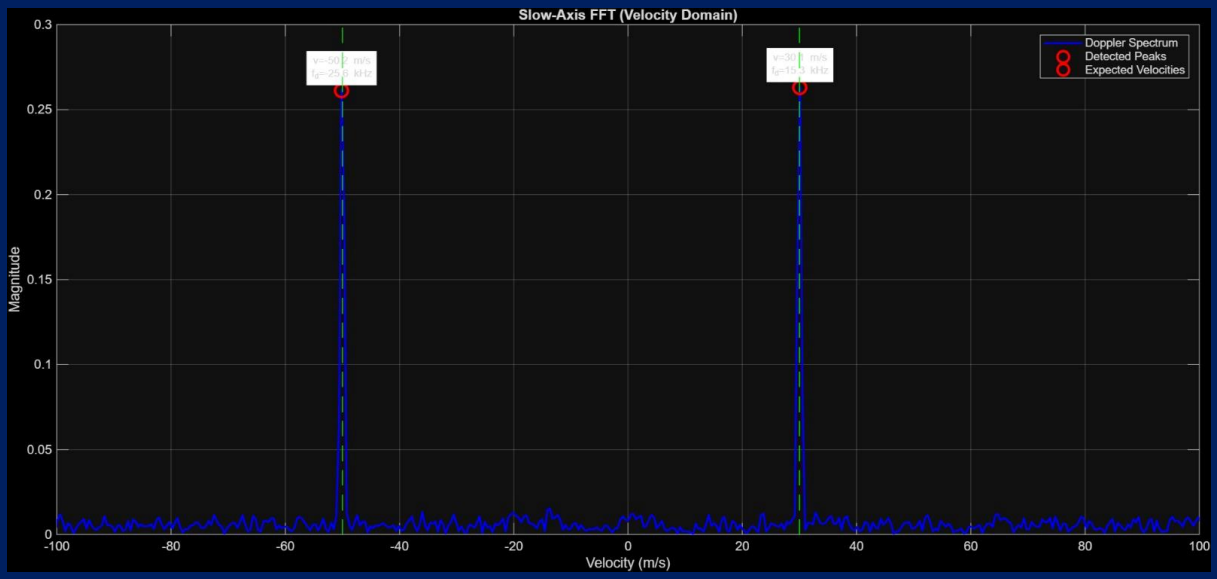


Figure 32: slow-10dB

Range-Velocity Map

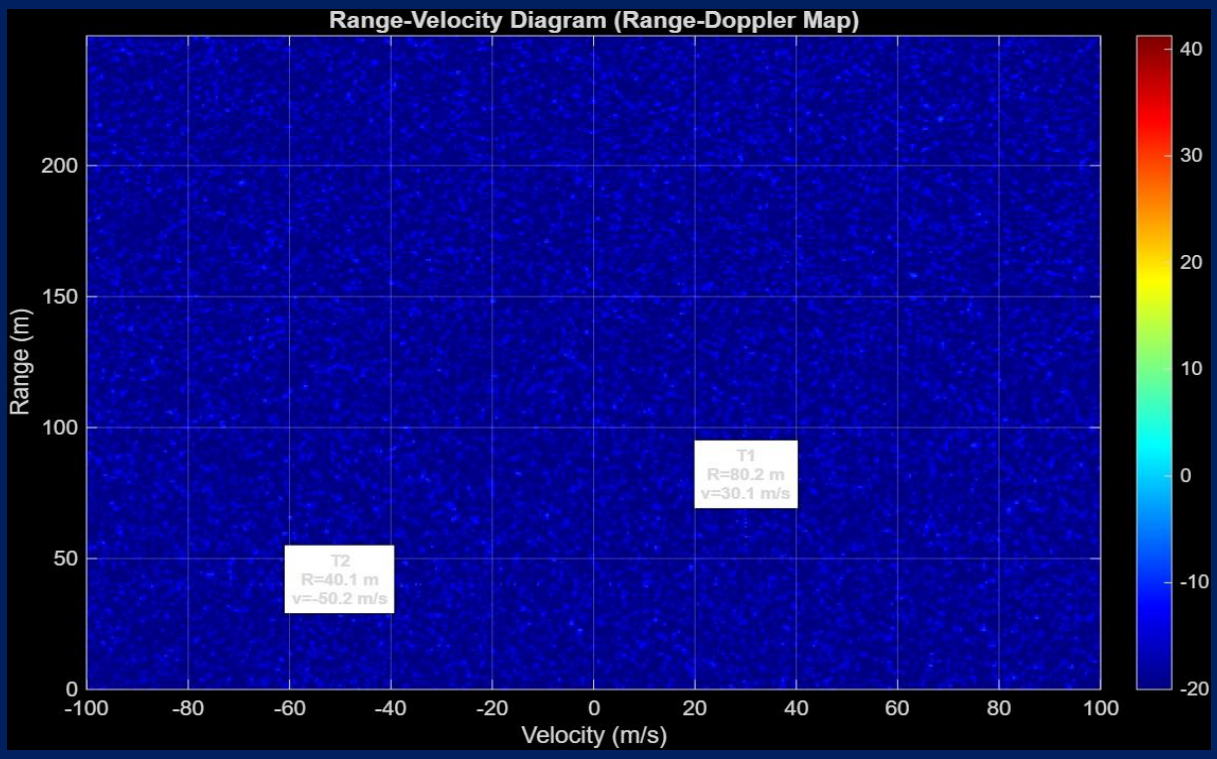


Figure 33: 2D-10dB

Bonus: Terminal Summary

```
SNR: 10.0 dB

========== DETECTION RESULTS ==========

Range-Velocity Map Detection:
  Target 1:
    Detected  R = 80.17 m,  V = 30.09 m/s
    Expected  R = 80.00 m,  V = 30.00 m/s
    Errors:   ΔR = 0.168 m, ΔV = 0.090 m/s
  Target 2:
    Detected  R = 40.08 m,  V = -50.15 m/s
    Expected  R = 40.00 m,  V = -50.00 m/s
    Errors:   ΔR = 0.084 m, ΔV = 0.150 m/s

========== SUMMARY ==========
Range Resolution     : 0.756 m
Velocity Resolution  : 0.456 m/s
Max Range            : 250.0 m
Max Velocity         : 100.0 m/s
```

Figure 34: summary-10dB

SNR = 15dB

TX Signal

- As shown the frequency of the chirping signal increases over time.
- Note: TX signal will not chang over different SNR values.

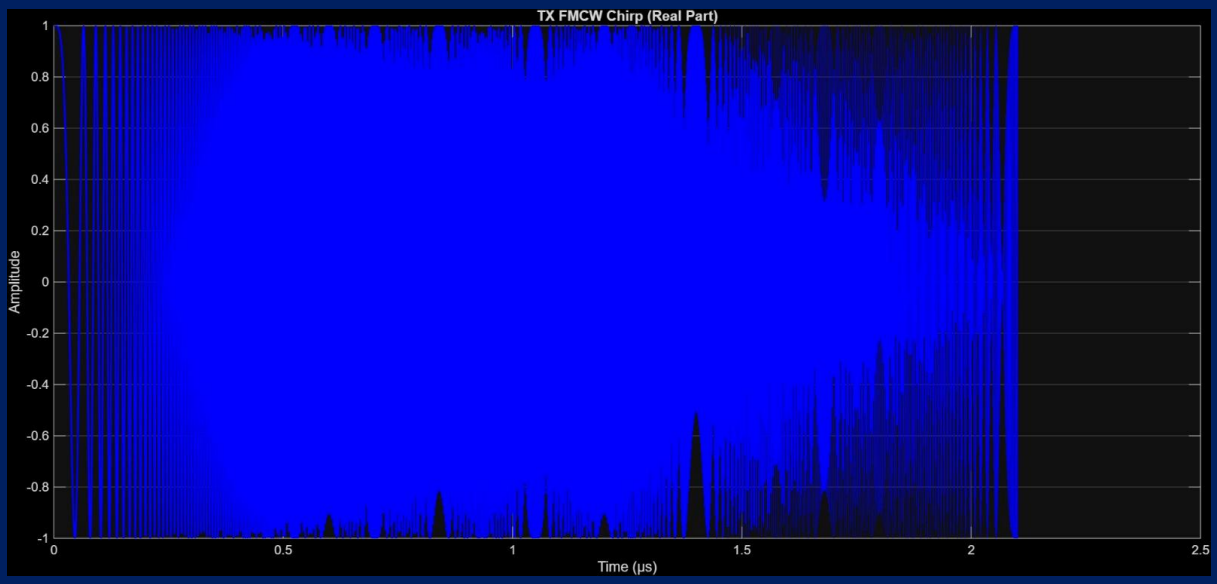


Figure 35: tx-15dB

RX Signal

- RX signal before the window time, is only Gaussian noise as we add noise to 0
- Once hitting window interval, the frequency of the signal also increases over time.
- It's clear that this RX signal is neat more than in 10dB results

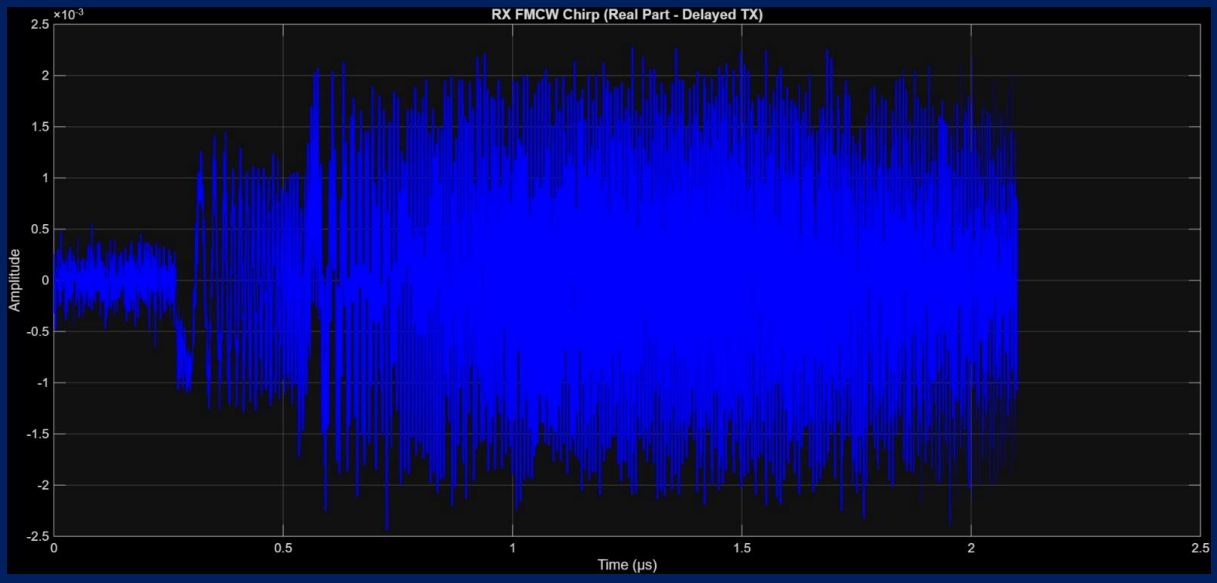


Figure 36: rx-15dB

Fast-Axis FFT

- FFT pins show the beat frequency and range of each target correctly (as defined).
- Noise is less than in 10dB results but it's also not critical

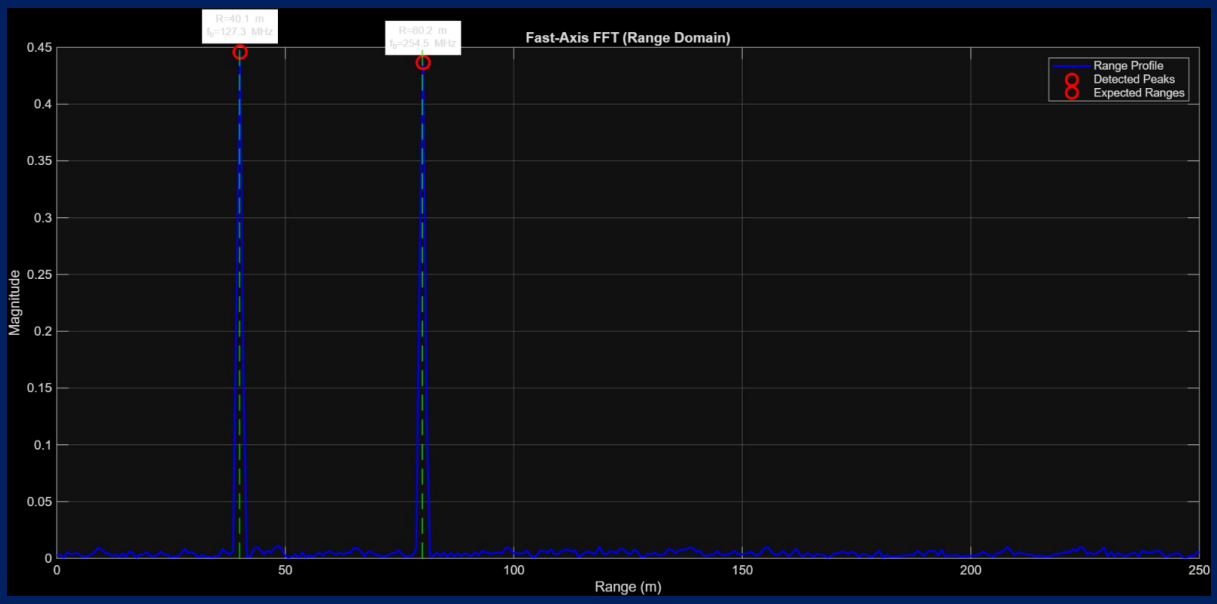


Figure 37: fast-15dB

Slow-Axis FFT

- FFT pins show Doppler frequency and velocity of each target correctly (as defined).
- Noise is less than in 10dB results but it's also not critical

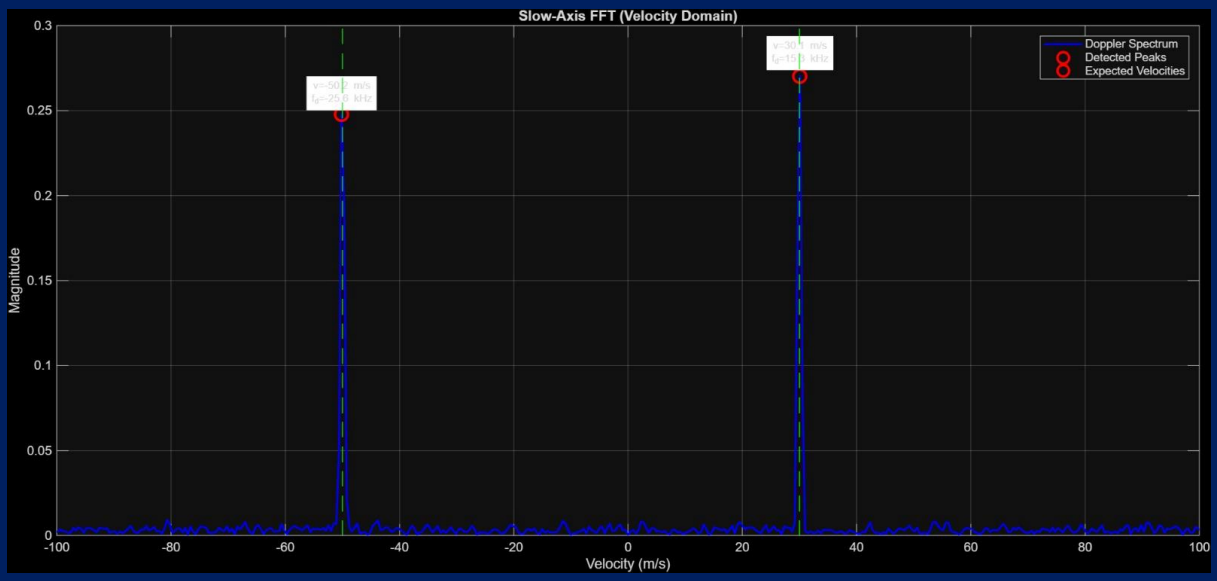


Figure 38: slow-15dB

Range-Velocity Map

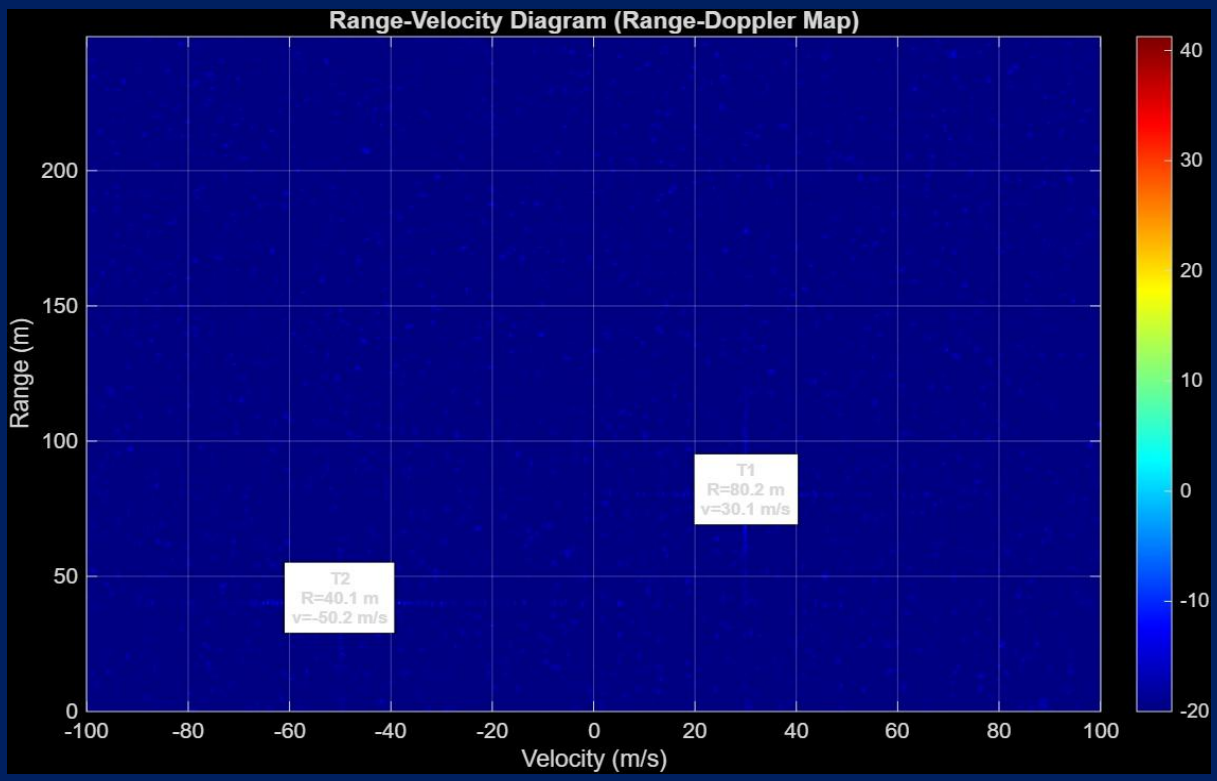


Figure 39: 2D-15dB

Bonus: Terminal Summary

```
SNR: 15.0 dB

========== DETECTION RESULTS ==========

Range-Velocity Map Detection:
  Target 1:
    Detected  R = 80.17 m,  V = 30.09 m/s
    Expected  R = 80.00 m,  V = 30.00 m/s
    Errors:   ΔR = 0.168 m, ΔV = 0.090 m/s
  Target 2:
    Detected  R = 40.08 m,  V = -50.15 m/s
    Expected  R = 40.00 m,  V = -50.00 m/s
    Errors:   ΔR = 0.084 m, ΔV = 0.150 m/s

========== SUMMARY ==========
Range Resolution     : 0.756 m
Velocity Resolution : 0.456 m/s
Max Range            : 250.0 m
Max Velocity         : 100.0 m/s
```

Figure 40: summary-15dB

References

- **M. A. Richards, *Fundamentals of Radar Signal Processing*, 2nd ed. New York, NY, USA: McGraw-Hill, 2022.**

- **M. I. Skolnik, *Introduction to Radar Systems*, 3rd ed. New York, NY, USA: McGraw-Hill, 2001.**

- **M. Jankiraman, *FMCW Radar Design*. Norwood, MA, USA: Artech House, 2018.**

- **V. Winkler, "Range Doppler Detection for Automotive FMCW Radars," in *European Radar* Conference (EuRAD), Munich, Germany, 2007, pp. 63–66.**

Appendix

- GitHub Repo: https://github.com/kareem05-ash/RadarSystem-MATLAB