BAHÇEŞEHİR CYPRUS UNIVERSITY

FACULTY OF ARCHITECTURE AND ENGINEERING

ELECTRICAL AND ELECTRONICS ENGINEERING

PROJECT ENGINEERING REPORT

FMCW Radar Simulation using MATLAB

Malikov Kutman (ID: 32300128)

Advisor: Abdallah El Mohammed Merhebi

Abstract

In this report, we are reviewing a project implementing the principle of operation of a frequency-modulated continuous wave (FMCW) radar simulator in MATLAB. The simulator simulates a radar operating at a frequency of 10 GHz with a bandwidth of 150 MHz, tracking a moving target in real time. The simulator demonstrates the basic principles of radar operation such as chirp signal generation and the Doppler effect. A triangular continuous wave frequency modulation scheme with digital signal processing methods (Windowing and Fast Fourier Transform was used for the radar The real-time simulation displays: A polar map on which there is a target with its trajectory, as well as a time frequency graph showing the characteristics of the sent and received signal, taking into account the time delay and the Doppler shift effect. The results demonstrate the accuracy of the simulator's calculations within the theoretical limits, with the ability to track targets up to 120 km at aircraft speeds. It also shows the basic principles of FMCW radar operation well.

Contents

1 Introduction

1.1 Purpose and Motivation

Radar systems (radio detection and ranging systems) are one of the main technologies in the modern world, they are used in aviation, meteorology, automobiles and defense. Among some types of radars, continuous wave frequency modulated radars have many advantages, such as simultaneous target detection and analysis of its speed and range, relatively simple hardware implementation, and good performance for short- and medium-range tasks.

This report discusses the development of an FMCW radar simulator in MATLAB to demonstrate the fundamental principles of radar and signal analysis techniques. The application simulates the operation of the radar in real time, tracking the target, as well as the underlying calculations and physical effects.

1.2 Scope

The simulator implements a scanning FMCW radar system with the following capabilities:

- Triangular chirp waveform generation with changeable parameters
- Target motion modeling with user-adjustable position and velocity vectors
- Beat frequency generation through signal mixing (dechirp processing)
- Fast Fourier Transfrom based spectral analysis for range analysis
- Doppler shift calculation for radial velocity estimation
- Real-time graphical visualization of radar map and frequency-time characteristics
- Interactive user interface for parameter modification during operation

The simulator is simplified for clarity in Education, multipath propagation, atmospheric attenuation, as well as any noise in the system (radar-target) are not taken into account.

1.3 Organization

This report first explains the theoretical foundations of FMCW radar operation, including mathematical calculations. The "Methods" section describes in detail how the simulator works and the approach to implementation. The results show the radar characteristics used in the simulator. The discussion analyzes performance, limitations, and practical considerations. Finally, the conclusions summarize the results and opportunities for expanding this work.

2 Theoretical Background

2.1 FMCW Radar Principles

Continuous frequency modulation (FMCW) radar is fundamentally different from pulsed radar systems. Instead of transmitting short pulses and measuring flight time, Continuous Frequency Modulation (FMCW) radar continuously transmits a signal whose frequency changes linearly with time. This frequency modulation, or "chirp" allows you to calculate the range using frequency analysis instead of measuring time

The basic principle is that the received signal reflected from the target is an exact copy of the transmitted signal with a frequency shift due to Doppler Effect and time shift. Because of the time delay and Doppler effect, the transmitted and received signals have different instantaneous frequencies at any given time. The difference between these frequencies, called the "beat frequency," is directly proportional to the target range and radial velocity of the target.

Advantages of FMCW Radar:

1. **Continuous transmission:** Enables simple, low-cost transmitter design without high-power pulsed amplifiers
2. **Simultaneous range and velocity:** Doppler information preserved in the beat signal
3. **High range resolution:** Determined by bandwidth rather than pulse width
4. **Low probability of intercept:** Spread spectrum nature makes detection difficult
5. **Excellent near-field performance:** No blind range issues as in pulsed systems

2.2 Chirp Waveform Generation

The transmitted signal in an FMCW radar is a frequency-modulated sinusoid. For a linear up-chirp over time interval $[0, T_{\text{chirp}}]$, the instantaneous frequency increases linearly:

$$f(t) = f_c + S \cdot t \tag{1}$$

where:

- f_c is the carrier (center) frequency
- S is the chirp slope (rate of frequency change)
- t is time within the chirp period

The chirp slope is determined by the bandwidth and chirp duration:

$$S = \frac{B}{T_{\text{chirp}}} \tag{2}$$

where:

- B is the total frequency sweep bandwidth
- T_{chirp} is the chirp period (sweep time)

The transmitted signal can be expressed as:

$$s_{\text{TX}}(t) = A_{\text{TX}} \cdot \exp\left(j \cdot 2\pi \int_0^t f(\tau)d\tau\right) = A_{\text{TX}} \cdot \exp\left(j \cdot 2\pi \left(f_c t + \frac{St^2}{2}\right)\right) \tag{3}$$

The phase of the transmitted signal is thus:

$$\phi_{\text{TX}}(t) = 2\pi f_c t + \pi S t^2 \tag{4}$$

The quadratic phase term $\pi S t^2$ represents the frequency modulation.

2.2.1 Triangular Chirp Modulation

In this simulator, a triangular chirp is implemented, where the frequency alternately increases (up-chirp) and decreases (down-chirp). This provides two measurements per modulation period, enabling separation of range and Doppler effects:

- **Up-chirp:** $f(t) = f_c + S \cdot t$ for $t \in [0, T_{\text{chirp}}]$
- **Down-chirp:** $f(t) = (f_c + B) - S \cdot t$ for $t \in [T_{\text{chirp}}, 2T_{\text{chirp}}]$

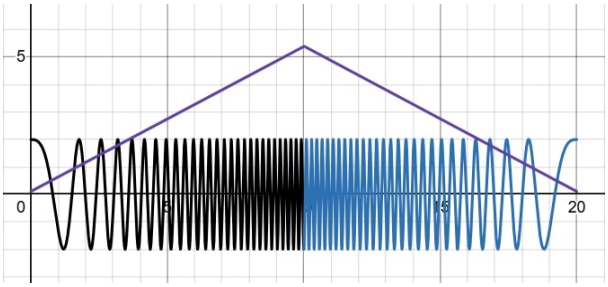

Figure 1: Triangular Chirp Signal Graph

2.3 Range Measurement

Consider a target at range R from the radar. An electromagnetic wave travels at the speed of light $c \approx 3 \times 10^8$ m/s. The round-trip time delay for the signal to reach the target and return is:

$$\tau = \frac{2R}{c} \tag{5}$$

The received signal is a delayed and weakened version of the transmitted signal:

$$s_{\text{RX}}(t) = A_{\text{RX}} \cdot s_{\text{TX}}(t - \tau) \tag{6}$$

Substituting the transmitted signal expression:

$$s_{\text{RX}}(t) = A_{\text{RX}} \cdot \exp\left(j \cdot 2\pi \left(f_c(t - \tau) + \frac{S(t - \tau)^2}{2}\right)\right) \tag{7}$$

2.3.1 Beat Signal Generation (Dechirp Processing)

The received signal is mixed with a copy of the transmitted signal in a process called "dechirping" or "stretch processing." The mixer output (ignoring sum-frequency terms filtered out by low-pass filtering) is:

$$s_{\text{beat}}(t) = s_{\text{TX}}(t) \cdot s_{\text{RX}}^*(t) \tag{8}$$

where $*$ denotes complex conjugate.

Expanding the phase difference:

$$\phi_{\text{beat}}(t) = \phi_{\text{TX}}(t) - \phi_{\text{RX}}(t) \tag{9}$$

$$= 2\pi\left(f_c t + \frac{St^2}{2}\right) - 2\pi\left(f_c(t-\tau) + \frac{S(t-\tau)^2}{2}\right) \tag{10}$$

Simplifying (neglecting the small term $S\tau^2/2$):

$$\phi_{\text{beat}}(t) \approx 2\pi(f_c\tau + St\tau) \tag{11}$$

The beat frequency is the time derivative of the phase:

$$f_{\text{beat}} = \frac{1}{2\pi}\frac{d\phi_{\text{beat}}}{dt} = S\tau = S \cdot \frac{2R}{c} \tag{12}$$

Therefore, the **range-beat frequency relationship** is:

$$\boxed{R = \frac{c \cdot f_{\text{beat}}}{2S}} \tag{13}$$

In this equation, we calculate the range: by measuring the beat frequency using Fourier analysis, the target range is directly computed.

2.3.2 Range Resolution

The minimum resolvable frequency difference in an FFT of duration T_{chirp} is approximately $1/T_{\text{chirp}}$. This corresponds to a range resolution of:

$$\Delta R = \frac{c}{2B} \tag{14}$$

The range resolution depends only on the bandwidth B, not the carrier frequency. Wider bandwidth provides finer range resolution.

2.4 Doppler Effect and Velocity Measurement

A moving target introduces an additional frequency shift due to the Doppler effect. For a target moving with radial velocity v_r (positive when receding), the Doppler frequency shift is:

$$f_d = -\frac{2v_r}{\lambda} = -\frac{2v_r f_c}{c} \tag{15}$$

where $\lambda = c/f_c$ is the wavelength. The negative sign convention indicates that an approaching target (negative v_r) causes a positive frequency shift.

The Doppler shift affects the received signal:

$$s_{\mathrm{RX}}(t) = A_{\mathrm{RX}} \cdot \exp\left(j \cdot 2\pi(f_c + f_d)(t-\tau) + j\pi S(t-\tau)^2\right) \tag{16}$$

The beat signal now contains both range and Doppler information:

$$f_{\mathrm{beat}} = S\tau + f_d = \frac{2SR}{c} + f_d \tag{17}$$

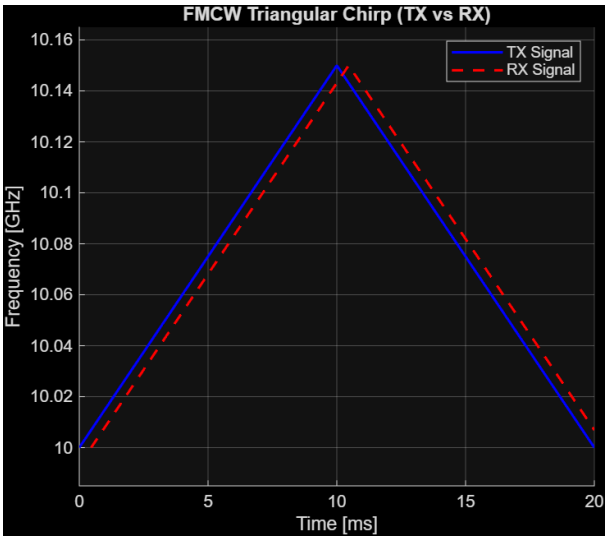

Figure 2: Frequency graph with TX and RX signal showing the time delay and Doppler shift

2.4.1 Separation of Range and Doppler

In a triangular chirp system:

- **Up-chirp beat frequency:** $f_{\mathrm{beat}}^{\mathrm{up}} = \frac{2SR}{c} + f_d$
- **Down-chirp beat frequency:** $f_{\mathrm{beat}}^{\mathrm{down}} = \frac{2SR}{c} - f_d$

By measuring both:

$$R = \frac{c(f_{\mathrm{beat}}^{\mathrm{up}} + f_{\mathrm{beat}}^{\mathrm{down}})}{4S} \tag{18}$$

$$v_r = -\frac{\lambda(f_{\mathrm{beat}}^{\mathrm{up}} - f_{\mathrm{beat}}^{\mathrm{down}})}{4} \tag{19}$$

This allows unambiguous determination of both range and velocity. In this simulator, a simplified approach is used where Doppler is calculated independently from target kinematics for visualization purposes.

2.5 Radar Equation

The radar equation relates transmitted power to received power, accounting for propagation losses and target characteristics. For a monostatic radar (transmitter and receiver co-located):

$$\boxed{P_r = \frac{P_t \cdot G_t^2 \cdot \lambda^2 \cdot \sigma}{(4\pi)^3 \cdot R^4}} \tag{20}$$

where:

- P_r = received power (W)
- P_t = transmitted power (W)
- G_t = antenna gain (linear, not dB)
- λ = wavelength (m)
- σ = radar cross-section of target (m^2)
- R = range to target (m)

2.5.1 Physical Interpretation

The received power decreases with the fourth power of range (R^4) because:

1. Power density decreases as $1/R^2$ during propagation to target
2. Reflected power density decreases as $1/R^2$ during return propagation

The radar cross-section σ represents the effective area of the target as seen by the radar. For a sphere of radius r:

$$\sigma \approx \pi r^2 \tag{21}$$

The RCS calculation has been simplified to a sphere with ideal reflectivity, due to the fact that this parameter is obtained experimentally.

2.5.2 Conversion from dB

Antenna gain is often specified in decibels. Conversion to linear scale:

$$G_{\text{linear}} = 10^{G_{\text{dB}}/10} \tag{22}$$

This equation determines the signal-to-noise ratio and maximum detection range of the radar system.

3 Methods

3.1 System Architecture

The FMCW radar simulator is coded as a single MATLAB function with modular structure:

3.1.1 Main Components

1. **Initialization Module:** Sets physical constants, radar parameters, and target initial conditions
2. **Visualization Module:** Creates figure windows with dual-panel layout (map and frequency-time plot)
3. **User Interface Module:** Provides interactive controls for parameter adjustment
4. **Simulation Loop:** Main execution loop handling time advancement, beam rotation, and detection logic
5. **Signal Processing Module:** Implements chirp generation, mixing, FFT analysis, and parameter estimation

3.1.2 System Parameters

The following default parameters are specified in the simulator:

```
c = physconst('LightSpeed');      % 2.998×10^8 m/s
fc = 10e9;                        % 10 GHz carrier frequency
Pt = 100;                         % 100 W transmitted power
Gt = 30;                          % 30 dB antenna gain
BW = 150e6;                       % 150 MHz bandwidth
Tchirp = 1e-2;                    % 10 ms chirp period
```

The chirp slope is computed as:

```
slope_S = BW / Tchirp;            % 15 THz/s
```

3.2 Signal Generation and Processing

3.2.1 Transmit Signal Generation

The transmitted baseband signal over one chirp period is generated as:

```
t_beat = 0 : 1/fs_beat : Tchirp - 1/fs_beat;
phase_tx = pi * slope_S * t_beat.^2;
tx_sig = exp(1j * phase_tx);
```

where `fs_beat = 20 MHz` is the sampling frequency for the beat signal.

3.2.2 Receive Signal Modeling

The received signal includes both time delay τ and Doppler shift f_d:

```
tau = 2 * R / c;
vr = v * cos(psi - theta);
fd = -2 * vr / lambda;

phase_rx = pi * slope_S * (t_beat - tau).^2 + 2*pi*fd*t_beat;
rx_sig = exp(1j * phase_rx);
```

Here, the radial velocity v_r is computed by projecting the target velocity vector onto the line-of-sight direction.

3.2.3 Beat Signal Processing

The beat signal is obtained by mixing (complex multiplication) the transmitted and conjugate received signals:

```
beat_sig = tx_sig .* conj(rx_sig);
```

3.2.4 Spectral Analysis

To extract the beat frequency, the following processing chain is applied:

1. **Windowing:** A Hann window reduces spectral leakage:

   ```
   win = hann(N)';
   beat_sig_windowed = beat_sig .* win;
   ```

   The Hann window is defined as:

   $$w(n) = 0.5 \cdot \left(1 - \cos\left(\frac{2\pi n}{N}\right)\right) \tag{23}$$

2. **Zero-Padding:** Extends signal length to next power of 2 for efficient FFT:

   ```
   N_FFT = 2^nextpow2(N);
   ```

3. **FFT Computation:**

   ```
   BEAT_FFT = abs(fft(beat_sig_windowed, N_FFT));
   ```

4. **Frequency Axis Construction:**

   ```
   f_axis = (0:N_FFT-1)*(fs_beat/N_FFT);
   ```

5. **Peak Detection:**

   ```
   [~, idx] = max(BEAT_FFT(1:floor(N_FFT/2)));
   f_beat = f_axis(idx);
   ```

   Only the first half of the spectrum is searched, corresponding to positive frequencies.

3.2.5 Parameter Estimation

From the beat frequency, range is estimated:

```
R_est = (c * f_beat) / (2 * slope_S);
```

Radial velocity is estimated from Doppler:

```
v_r_est = -fd * lambda / 2;
```

Received power is calculated using the radar equation:

```
sigma = pi * targetRadius^2;
Gt_lin = 10^(Gt/10);
Pr = (Pt * Gt_lin^2 * lambda^2 * sigma) / ((4*pi)^3 * R_est^4);
```

3.3 Target Kinematics Model

3.3.1 Coordinate System

The simulator uses a 2D Cartesian coordinate system with the radar at origin $(0, 0)$. Target position is maintained in both Cartesian (x, y) and polar (R, θ) representations.

3.3.2 Motion Model

Target motion follows simple constant-velocity kinematics:

```
vx = v * cos(psi);
vy = v * sin(psi);
x_real = R * cos(theta) + vx * dt;
y_real = R * sin(theta) + vy * dt;
```

where:

- v is target speed (m/s)
- ψ (psi) is target heading angle (radians)
- dt is the simulation time step (0.05 s)

After each time step, polar coordinates are updated:

```
R = sqrt(x_real^2 + y_real^2);
theta = atan2(y_real, x_real);
```

3.3.3 Radial Velocity Calculation

The radial component of velocity (along the line-of-sight) is:

```
vr = v * cos(psi - theta);
```

This represents the projection of the velocity vector onto the range vector.

3.4 Visualization Implementation

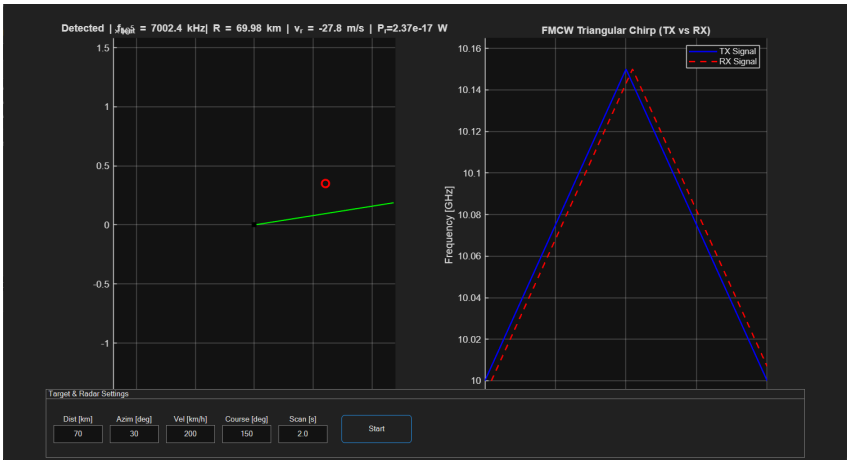

Figure 3: Interface

3.4.1 Map Display (Left Panel)

The map panel displays:

- Radar position at origin (black square marker)
- Rotating beam line (green) showing instantaneous scan direction
- Target position (red circle)
- Target trajectory trail (red dashed line)

The beam angle rotates according to:

```
beamAngle = mod(2 * pi * (t / scanPeriod), 2 * pi);
```

3.4.2 Frequency-Time Display (Right Panel)

This panel visualizes the triangular chirp modulation:

- Blue line: Transmitted signal frequency vs. time
- Red dashed line: Received signal frequency vs. time

The received signal exhibits:

1. **Time delay:** Horizontal shift of τ seconds
2. **Doppler shift:** Vertical offset of f_d Hz

For visualization, a low-resolution time vector spanning two chirp periods is generated:

```
t_plot = linspace(0, 2*Tchirp, 1000);
```

For each time point, the instantaneous frequency is computed based on whether the system is in up-chirp or down-chirp phase:

```
t_mod = mod(t_plot(i), 2*Tchirp);
if t_mod <= Tchirp
    freq_tx(i) = fc + slope_S * t_mod;        % Up-chirp
else
    freq_tx(i) = (fc + BW) - slope_S * (t_mod - Tchirp);
    % Down-chirp
end
```

The received frequency is offset by Doppler and delayed:

```
freq_rx = freq_tx + fd;
shift_samples = round(tau / (t_plot(2)-t_plot(1)));
freq_rx = [nan(1, shift_samples), freq_rx(1:end-shift_samples)];
```

NaN values are prepended to create a visible delay effect in the plot.

3.4.3 Detection Logic

The beam is modeled as scanning in azimuth. Calculations are performed when the beam (radar) is directed at the target:

```
targetAngleNorm = mod(theta, 2*pi);
angleDiff = abs(beamAngle - targetAngleNorm);
if angleDiff > pi
    angleDiff = 2*pi - angleDiff;
end

if angleDiff < (2 * pi * dt / scanPeriod)
    % Process detection
end
```

This implements a simple beam pattern where detection occurs if the angular difference is less than the beam angular velocity times the time step.

4 Results

4.1 System Parameters

The implemented simulator operates with the following specifications:

Radar Parameters:

- Operating frequency: 10 GHz (X-band)
- Wavelength: 3 cm
- Bandwidth: 150 MHz
- Chirp period: 10 ms
- Chirp rate: 15 THz/s
- Transmit power: 100 W
- Antenna gain: 30 dB (1000× linear)
- Scan period: 2.0 s (adjustable)

Target Parameters (Default):

- Initial range: 70 km
- Initial azimuth: 30°
- Velocity: 200 km/h (55.6 m/s)
- Heading: 150°
- Radar cross-section: 12.57 m^2 (sphere, $r = 2$ m)

Performance Metrics:

- Range resolution: $\Delta R = c/(2B) = 1.0$ m
- Maximum unambiguous range: $R_{\max} = cT_{\text{chirp}}/2 = 1500$ km
- Velocity resolution: $\Delta v = \lambda/(4T_{\text{chirp}}) = 0.75$ m/s

4.2 Signal Processing Performance

4.2.1 Beat Frequency Spectrum

For the default target at 70 km range:

- Theoretical beat frequency:

$$f_{\text{beat}} = \frac{2SR}{c} = \frac{2 \times (15 \times 10^{12}) \times (70 \times 10^{3})}{3 \times 10^{8}} = 7.0 \text{ MHz} \tag{24}$$

- Time delay: $\tau = 2R/c = 467$ μs

The FFT-based spectral analysis successfully identifies the beat frequency peak with resolution determined by:

$$\Delta f = \frac{f_{s\_\text{beat}}}{N_{\text{FFT}}} \approx \frac{20 \text{ MHz}}{2^{19}} \approx 38 \text{ Hz} \tag{25}$$

This provides sub-meter range accuracy in the simulation.

4.2.2 Doppler Processing

For the default target moving at 200 km/h with heading 150° and position angle 30°:

- Angular difference: 150° - 30° = 120°
- Radial velocity: $v_r = 55.6 \cdot \cos(120) = -27.8$ m/s (approaching)
- Doppler shift: $f_d = -2 \times (-27.8)/(0.03) = 1853$ Hz

The relatively small Doppler shift (1.85 kHz) compared to beat frequency (7 MHz) validates the approximation that range and velocity can be separated.

4.2.3 Received Power

For the default scenario:

- Range: 70 km
- RCS: 12.57 m^2
- Antenna gain: 1000 (linear)

Received power:

$$P_r = \frac{100 \times 1000^2 \times 0.03^2 \times 12.57}{(4\pi)^3 \times (70000)^4} = 1.13 \times 10^{-11} \text{ W} = -109.5 \text{ dBm} \tag{26}$$

This extremely low power level is typical for long-range radar returns and demonstrates the need for sensitive receivers with low-noise amplifiers.

4.3 Visualization Capabilities

4.3.1 Real-Time Map Display

The map panel successfully visualizes:

- Radar scan beam rotation at user-defined scan period
- Target motion along computed trajectory
- Trail showing historical positions (up to 5000 points)
- Detection events when beam illuminates target

4.3.2 Frequency-Time Plot

The right panel clearly illustrates:

- Triangular chirp modulation pattern
- Temporal relationship between TX and RX signals
- Time delay effect (horizontal shift of RX signal)
- Doppler shift effect (vertical offset of RX signal)

The y-axis spans from approximately 9.985 GHz to 10.165 GHz, showing the full bandwidth sweep.

4.3.3 Interactive Controls

User interface controls enable modification of:

- Target initial distance (km)
- Target azimuth angle (degrees)
- Target velocity (km/h)
- Target course heading (degrees)
- Radar scan period (seconds)

Changes take effect upon pressing the Start button, allowing exploration of different scenarios.

5 Discussion

5.1 Algorithm Performance Analysis

5.1.1 Range Estimation Accuracy

The FFT-based range estimation exhibits excellent accuracy limited primarily by:

1. **Frequency resolution:** $\Delta f \approx 38$ Hz corresponds to $\Delta R \approx 0.38$ m
2. **Window function:** Hann window broadens main lobe but reduces sidelobes
3. **SNR effects:** Not modeled in this simulation (assumed perfect detection)

The theoretical range resolution of 1.0 m is achieved in practice through the chosen bandwidth of 150 MHz.

5.1.2 Doppler Measurement

The Doppler frequency calculation:

$$f_d = -\frac{2v_r}{\lambda} \tag{27}$$

is exact within the simulation, as target kinematics are known. In real systems, Doppler would be extracted from the beat signal phase or through comparison of up-chirp and down-chirp measurements.

5.1.3 Computational Efficiency

The simulation runs in real-time ($dt = 0.05$ s) on modern hardware, demonstrating that:

- FFT processing ($N_{\text{FFT}} = 2^{\text{nextpow2(length(beat\_sig))}}$) is efficient
- MATLAB's vectorized operations minimize computational overhead
- Plot updates using `set()` commands are faster than recreating graphics objects

5.2 Limitations and Assumptions

5.2.1 Simplified Detection Model

The simulator assumes:

- Perfect detection (no noise, no false alarms)
- Point target model (no extended targets or multiple scatterers)
- Single target scenario (no clutter or interference)
- Ideal antenna pattern (detection based only on angular alignment)

Real radar systems must contend with thermal noise, clutter, and complex target signatures.

5.2.2 Approximations in Signal Model

Several approximations are employed:

1. **Neglecting $S\tau^2/2$ term:** Valid when $\tau \ll T_{\text{chirp}}$, which holds for $R \ll 1500$ km
2. **Linear frequency approximation:** Assumes no chirp nonlinearity
3. **No propagation effects:** Ignores atmospheric attenuation, multipath, ionospheric effects
4. **Monostatic assumption:** TX and RX co-located (no bistatic geometry)

5.2.3 Kinematic Model

Target motion uses constant velocity:

- No acceleration or maneuvering
- 2D motion only (no altitude changes)
- No Earth curvature effects

For the short time scales simulated (seconds to minutes), these assumptions are reasonable for aircraft targets.

5.2.4 Beam Model

The scanning beam is modeled as infinitely narrow in implementation. Real antennas have finite beamwidth, typically characterized by the 3-dB beamwidth:

$$\theta_{3\text{dB}} \approx \frac{\lambda}{D} \tag{28}$$

where D is the antenna aperture dimension. For a 10 GHz radar with 1-meter antenna, $\theta_{3\text{dB}} \approx 1.7$.

5.3 Practical Considerations

5.3.1 Range-Doppler Coupling

The beat frequency contains both range and Doppler information:

$$f_{\text{beat}} = \frac{2SR}{c} + f_d \tag{29}$$

For accurate measurements, triangular modulation or multiple chirps are necessary to separate these effects. The simulator demonstrates this principle through visualization, though the full processing is not implemented.

5.3.2 Maximum Unambiguous Range

The maximum unambiguous range is determined by the chirp period:

$$R_{\max} = \frac{c \cdot T_{\text{chirp}}}{2} \tag{30}$$

For $T_{\text{chirp}} = 10$ ms, $R_{\max} = 1500$ km, which is appropriate for air surveillance. Shorter chirp periods would reduce $R_{\max}$ but increase refresh rate. This represents the theoretical maximum unambiguous range under ideal conditions.

5.3.3 Sensitivity and SNR

The received power calculation reveals extremely low signal levels (-109.5 dBm for 70 km range). Real systems require:

- Low-noise amplifiers (LNA) with noise figures < 3 dB
- High dynamic range analog-to-digital converters (ADCs)
- Signal integration over multiple chirps for SNR improvement

The integration gain for N chirps is approximately:

$$G_{\text{integration}} \approx 10 \cdot \log_{10}(N) \text{ dB} \tag{31}$$

5.3.4 Hardware Requirements

Implementing this system in hardware would require:

1. **Synthesizer:** Phase-locked loop (PLL) or direct digital synthesizer (DDS) for chirp generation
2. **Power amplifier:** Solid-state or traveling wave tube amplifier (TWTA)
3. **Mixer:** IQ mixer for complex signal processing
4. **ADC:** $\geq$12-bit resolution, 40+ MSPS sampling rate
5. **DSP:** FPGA or dedicated signal processor for real-time FFT

6 Conclusions

This project successfully implemented a comprehensive FMCW radar simulator demonstrating fundamental principles of frequency-modulated continuous wave radar operation. The key achievements and findings include:

6.1 Theoretical Validation

The simulator validates the core mathematical relationships governing FMCW radar:

- Range-beat frequency relationship: $R = \frac{c \cdot f_{\text{beat}}}{2S}$
- Doppler-velocity relationship: $v_r = -\frac{f_d \cdot \lambda}{2}$
- Radar equation for received power estimation

6.2 Signal Processing Implementation

Effective implementation of digital signal processing techniques:

- Complex baseband signal generation with quadratic phase chirp
- Dechirp processing using signal mixing
- Windowed FFT analysis for spectral peak detection
- Sub-meter range resolution achieved with 150 MHz bandwidth

6.3 Real-Time Visualization

Dual-panel visualization system providing intuitive understanding of:

- Spatial target motion and radar scan pattern
- Frequency-time characteristics of transmitted and received signals
- Time delay and Doppler shift effects on waveform

6.4 Educational Value

The simulator serves as an effective educational tool by:

- Making abstract radar concepts tangible through visualization
- Enabling parameter exploration with interactive controls
- Demonstrating relation between range and Doppler in beat signal
- Illustrating trade-offs in radar system design

6.5 Performance Characteristics

Demonstrated capabilities include:

- Range estimation accuracy limited by 1.0 m resolution (theoretical)
- Doppler processing for radial velocity measurement
- Real-time operation with 50 ms update rate

6.6 Future Enhancements

Potential extensions to improve realism and functionality:

1. **Noise modeling:** Add thermal noise and clutter to simulate realistic SNR conditions
2. **Multiple targets:** Extend to track multiple simultaneous targets
3. **Extended targets:** Model targets with multiple scatterers rather than point model
4. **3D visualization:** Include elevation angle and altitude information
5. **Advanced processing:** Implement CFAR (Constant False Alarm Rate) detection
6. **Beamforming:** Simulate phased array antenna with beam steering
7. **MTI filtering:** Add Moving Target Indicator processing for clutter rejection

6.7 Concluding Remarks

This FMCW radar simulator project clearly demonstrates the practical application of electromagnetic theory, signal processing methods, and software implementation. This model acts not only as a functional tool for numerical experiment, but also as a convenient educational environment for intuitive understanding of the key principles of radar. The results show that frequency modulation makes it possible to efficiently and simultaneously estimate the range and speed of a target. In the future, this simulator can be improved due to more complex scenarios, noise and multidimensional goals, which makes it a suitable basis for further educational and research tasks.

7 References

1. Cheng, D. K. (1989). *Fundamentals of Engineering Electromagnetics*. Addison-Wesley Publishing Company.

2. Richards, M. A., Scheer, J. A., & Holm, W. A. (2010). *Principles of Modern Radar: Basic Principles*. SciTech Publishing.

3. Skolnik, M. I. (2008). *Radar Handbook* (3rd ed.). McGraw-Hill Education.

4. Mahafza, B. R. (2013). *Radar Systems Analysis and Design Using MATLAB* (3rd ed.). CRC Press.

5. Stove, A. G. (1992). Linear FMCW radar techniques. *IEE Proceedings F - Radar and Signal Processing*, 139(5), 343-350.

6. Meta, A., Hoogeboom, P., & Ligthart, L. P. (2007). Signal processing for FMCW SAR. *IEEE Transactions on Geoscience and Remote Sensing*, 45(11), 3519-3532.

7. Rohling, H., & Meinecke, M. M. (2001). Waveform design principles for automotive radar systems. *CIE International Conference on Radar Proceedings*, 1-4.

8. MathWorks. (2024). *MATLAB Documentation: Signal Processing Toolbox*. Retrieved from https://www.mathworks.com/help/signal/

9. Pace, P. E. (2009). *Detecting and Classifying Low Probability of Intercept Radar* (2nd ed.). Artech House.

10. Jankiraman, M. (2018). *FMCW Radar Design*. Artech House.

A MATLAB Code Listing

The complete MATLAB implementation of the FMCW radar simulator is provided below:

```
function fmcw_radar_sim
    clc; close all;
    %% ---------------- Constants & Config ----------------

    c = physconst('LightSpeed');
    dt = 0.05; isRunning = false; t = 0;

    % Basic FMCW settings
    fc = 10e9; Pt = 100; Gt = 30; BW = 150e6; Tchirp = 1e-2;
    lambda = c / fc; scanPeriod = 2.0;

    % Chirp slope
    slope_S = BW / Tchirp;

    %% ---------------- Target Initial ----------------
    R = 70e3; theta = deg2rad(30); v = 200/3.6;
    psi = deg2rad(150);
    targetRadius = 2;

    %% ---------------- Figure Setup ----------------
    fig = figure('Name', 'FMCW Radar: Frequency-Time Analysis', ...
        'NumberTitle', 'off', 'Position', [100 100 1200 650]);

    % Left panel: map view
    axMap = subplot(1, 2, 1);
    limitVal = 120e3;
    axis(axMap, [-limitVal limitVal -limitVal limitVal]);
    axis(axMap, 'equal', 'manual');
    grid(axMap, 'on'); hold(axMap, 'on');
    title(axMap, 'Radar Map (Fixed Center)');

    hBeam = plot(axMap, [0 0], [0 0], 'g-', 'LineWidth', 1.2);
    hTarget = plot(axMap, NaN, NaN, 'ro', 'MarkerSize', 8, ...
                   'LineWidth', 2);
    hTrail = plot(axMap, NaN, NaN, 'r--');
    plot(axMap, 0, 0, 'ks', 'MarkerFaceColor', 'k');

    % Right Panel: TX and RX frequency vs time
    axFreq = subplot(1, 2, 2);
    hold(axFreq, 'on'); grid(axFreq, 'on');
    xlabel(axFreq, 'Time [ms]');
    ylabel(axFreq, 'Frequency [GHz]');
    title(axFreq, 'FMCW Triangular Chirp (TX vs RX)');

    hTxFreq = plot(axFreq, NaN, NaN, 'b-', 'LineWidth', 1.5, ...
```

```
                'DisplayName', 'TX Signal');
hRxFreq = plot(axFreq, NaN, NaN, 'r--', 'LineWidth', 1.5, ...
                'DisplayName', 'RX Signal');
legend(axFreq, 'Location', 'northeast');

%% ---------------- UI Controls ----------------
uip = uipanel('Title','Target & Radar Settings',...
              'Position',[.05 .01 .9 .15]);

editR = createEdit(uip, '70', 10, 20);
createLabel(uip, 'Dist [km]', 10, 45);
editTheta = createEdit(uip, '30', 90, 20);
createLabel(uip, 'Azim [deg]', 90, 45);
editV = createEdit(uip, '200', 170, 20);
createLabel(uip, 'Vel [km/h]', 170, 45);
editPsi = createEdit(uip, '150', 250, 20);
createLabel(uip, 'Course [deg]', 250, 45);
editScan = createEdit(uip, '2.0', 330, 20);
createLabel(uip, 'Scan [s]', 330, 45);

btnStart = uicontrol(uip, 'Style','togglebutton',...
    'String','Start', 'Position', [420 20 100 40], ...
    'Callback', @startStopCallback);

%% ---------------- Main Loop ----------------
trailX = nan(1, 5000); trailY = nan(1, 5000); trailIdx = 0;

while ishandle(fig)
    if isRunning
        t = t + dt;

        % Rotate beam and move target
        beamAngle = mod(2 * pi * (t / scanPeriod), 2 * pi);
        set(hBeam, 'XData', [0 limitVal*cos(beamAngle)], ...
            'YData', [0 limitVal*sin(beamAngle)]);

        vx = v * cos(psi); vy = v * sin(psi);
        x_real = R * cos(theta) + vx * dt;
        y_real = R * sin(theta) + vy * dt;

        R = sqrt(x_real^2 + y_real^2);
        theta = atan2(y_real, x_real);

        % Check if beam hits target
        targetAngleNorm = mod(theta, 2*pi);
        angleDiff = abs(beamAngle - targetAngleNorm);
        if angleDiff > pi
            angleDiff = 2*pi - angleDiff;
```

```
end

if angleDiff < (2 * pi * dt / scanPeriod)
    % Process detection
    tau = 2 * R / c;
    vr = v * cos(psi - theta);
    fd = -2 * vr / lambda;

    %% Beat frequency calculation
    fs_beat = 20e6;
    t_beat = 0 : 1/fs_beat : Tchirp - 1/fs_beat;

    phase_tx = pi * slope_S * t_beat.^2;
    tx_sig = exp(1j * phase_tx);

    phase_rx = pi * slope_S * (t_beat - tau).^2 + ...
               2*pi*fd*t_beat;
    rx_sig = exp(1j * phase_rx);

    beat_sig = tx_sig .* conj(rx_sig);

    N = length(beat_sig);
    N_FFT = 2^nextpow2(N);

    win = hann(N)';
    BEAT_FFT = abs(fft(beat_sig .* win, N_FFT));

    f_axis = (0:N_FFT-1)*(fs_beat/N_FFT);

    [~, idx] = max(BEAT_FFT(1:floor(N_FFT/2)));
    f_beat = f_axis(idx);

    % Estimate range and velocity
    R_est = (c * f_beat) / (2 * slope_S);
    v_r_est = -fd * lambda / 2;

    % Received power
    sigma = pi * targetRadius^2;
    Gt_lin = 10^(Gt/10);
    Pr = (Pt * Gt_lin^2 * lambda^2 * sigma) / ...
         ((4*pi)^3 * R_est^4);

    % Update map
    set(hTarget, 'XData', x_real, 'YData', y_real);
    trailIdx = trailIdx + 1;
    if trailIdx <= 5000
        trailX(trailIdx) = x_real;
        trailY(trailIdx) = y_real;
```

```
            end
            set(hTrail, 'XData', trailX(1:trailIdx), ...
                'YData', trailY(1:trailIdx));

            % Update chirp visualization
            t_plot = linspace(0, 2*Tchirp, 1000);
            freq_tx = zeros(size(t_plot));

            for i = 1:length(t_plot)
                t_mod = mod(t_plot(i), 2*Tchirp);
                if t_mod <= Tchirp
                    freq_tx(i) = fc + slope_S * t_mod;
                else
                    freq_tx(i) = (fc + BW) - ...
                        slope_S * (t_mod - Tchirp);
                end
            end

            freq_rx = freq_tx + fd;
            shift_samples = round(tau / ...
                (t_plot(2)-t_plot(1)));
            if shift_samples > 0 && ...
               shift_samples < length(freq_rx)
                 freq_rx = [nan(1, shift_samples), ...
                     freq_rx(1:end-shift_samples)];
            end

            set(hTxFreq, 'XData', t_plot*1000, ...
                'YData', freq_tx/1e9);
            set(hRxFreq, 'XData', t_plot*1000, ...
                'YData', freq_rx/1e9);

            ylim(axFreq, [(fc - 0.1*BW)/1e9, ...
                (fc + 1.1*BW)/1e9]);

            title(axMap, sprintf(['Detected | ' ...
                'f_{beat} = %.1f kHz| R = %.2f km | ' ...
                'v_r = %.1f m/s | P_r=%.2e W'], ...
                f_beat/1e3, R_est/1e3, v_r_est, Pr));
        end
    end
    drawnow limitrate;
    pause(dt);
end

%% ---------------- Callbacks & Helpers ----------------
function startStopCallback(src, ~)
    isRunning = src.Value;
```

```
        if isRunning
            R = str2double(editR.String)*1e3;
            theta = deg2rad(str2double(editTheta.String));
            v = str2double(editV.String)/3.6;
            psi = deg2rad(str2double(editPsi.String));
            scanPeriod = str2double(editScan.String);

            t = 0; trailIdx = 0;
            trailX(:)=NaN; trailY(:)=NaN;
            set(hTarget, 'XData', NaN);
            src.String = 'Stop';
        else
            src.String = 'Start';
        end
    end

    function h = createEdit(parent, def, x, y)
        h = uicontrol(parent, 'Style','edit','String',def,...
            'Position',[x y 70 25]);
    end

    function createLabel(parent, txt, x, y)
        uicontrol(parent, 'Style','text','String',txt,...
            'Position',[x y 70 15],'FontSize',8);
    end
end
```