FMCW Radar for Range Measurement, Velocity Measurement, and Localization

12312722 Wu Tangzi 12312720 Liu Haichuan

[Outline]

I. Background and Understanding of FMCW

1.Basic Background

Frequency Modulated Continuous Wave (FMCW) radar is a radar system that detects targets by continuously transmitting a frequency-modulated signal, that is, a signal whose frequency varies over time. Unlike conventional pulse radar, it does not transmit short pulse signals; instead, it transmits a continuous waveform, usually a linear frequency-modulated (LFM) signal whose frequency changes with time.

2.Advanteges of FMCW Radar

(1) High-precision range and velocity measurement:

FMCW radar accurately measures target range through frequency modulation and simultaneously obtains target velocity by using the Doppler effect. This gives FMCW radar high precision in both range and velocity measurement, making it suitable for applications that require both target range and velocity.

(2) Low power consumption:

FMCW radar generally uses low-power continuous-wave transmission, resulting in relatively low power consumption. It is therefore suitable for applications with strict power requirements, such as unmanned driving and autonomous driving.

(3) Simple hardware architecture:

Compared with phased-array radar, FMCW radar usually has a simpler structure and lower cost, especially in low-power and low-cost systems.

(4) Strong anti-interference capability:

FMCW radar is relatively resistant to external interference. Because its operating frequency is continuously modulated, it can better suppress interference signals in complex environments.

(5) Suitable for short-range detection:

FMCW radar can effectively detect short-range targets with high accuracy, and it is widely used in vehicle radar, weather radar, and other applications.

3.the Disadvantages of FMCW Radar to Some Extent

(1) Relatively limited range-measurement distance:

FMCW radar is suitable for short- and medium-range detection, and its measurement range is shorter than that of pulse radar. Its ability to detect long-range targets is relatively weak.

(2) Difficulty in processing high-speed targets:

Its response to high-speed moving targets is relatively weak, and velocity-measurement errors or signal aliasing may occur, affecting performance.

(3) Resolution and bandwidth limitations:

The resolution of FMCW radar is limited by frequency bandwidth. A narrower frequency band may lead to lower resolution.

(4) Significant environmental influence:

FMCW radar is relatively sensitive to weather conditions. Rain, snow, fog, and similar conditions may cause signal attenuation and affect detection performance.

4.Some Good Research Directions

(1) Efforts can be made to improve its velocity- and range-measurement coverage. Technologies such as active phased arrays can be used to further improve the velocity-measurement range and measurement accuracy of FMCW radar.

(2) Its anti-interference capability can be further investigated to improve performance under adverse weather and environmental conditions, which is very important for automotive radar applications.

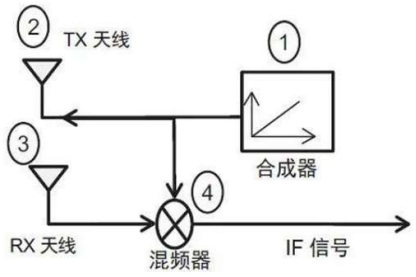
(3) The cost of the radar can be further reduced through additional optimization of the manufacturing process.

II. Theoretical Analysis (Only the Important Parts Are Discussed in

Detail Here: Range Resolution and Velocity Resolution)

1. Basic Procedure

Tx chirp -> receive Rx chirp -> mixing -> IF signal -> LP filter -> ADC data acquisition -> FFT on ADC data. The target range is determined from the peak in the spectrum.



Signal: $\mathrm{x}(t) = \mathrm{A}\cos\left(2\pi fct + \theta(t)\right)[u(t) - u(t-\tau)]$

2. Range Resolution

Definition: the minimum distance at which two objects can be resolved as two peaks in the IF spectrum.

Method 1:

$$\mathrm{R} = \frac{\tau c}{2}$$

$$\mathrm{B} = \mathrm{K} \times T \longrightarrow f_m = \mathrm{K} \times$$

$$\mathrm{R} = \frac{f_m c}{2K}$$

$$\Delta R = \frac{\Delta f_m c}{2S}$$

$$\Delta f_m = \frac{f_s}{N_r} - - f_s = \frac{N_r}{T} \longrightarrow \Delta f_m = \frac{1}{T}$$

$$\Delta R = \frac{\Delta f_m c}{2K} = \frac{c}{2KT} = \frac{c}{2B} = \frac{3 \times 10^8}{2 \times 150 \times 10^6} m = 1m$$

Method 2 (more suitable for explaining the MATLAB code):

$f_s = \frac{N_r}{T}$The bandwidth occupies 1024 bins, and the beat frequency occupies $f_m = \frac{n}{T}$1 bin. Therefore, $f_m$it is $f_s \frac{n}{N_r}$times

$$f_m = \frac{f_s}{N} \times n$$

$$\mathrm{R} = \frac{cTf_m}{2B} = \frac{cT}{2B} \times \frac{f_s}{N} \times n = \frac{c}{2B} \times \frac{Tf_s}{N_r} \times n = 1 \times 1 \times n = 1 \times n$$

$$\Delta R = 1m$$

3. Velocity Resolution

Velocity-measurement method: transmit 2 linear chirps separated by T_c, where the corresponding range FFT of each chirp has the same peak at the same position but a different phase. The phase difference between peaks $\Delta\phi$corresponds directly to object motion.

$$\Delta x = \mathrm{v}T_c$$

$$\Delta\phi = \frac{4\pi\Delta x}{\lambda} = \frac{4\backslash pi\mathrm{v}\mathrm{T_c}}{\lambda}$$

$$\mathrm{v} = \frac{\lambda\Delta\phi}{4\pi T_c}$$

As long as the separation between two discrete frequencies $\Delta\phi$is greater than one sample$\frac{2\pi}{N}\mathrm{radius/}$ $\mathrm{sample} = \frac{1}{N}\mathrm{cycles/smaple}$, the two frequencies can be resolved.

Method 1:

$$\phi_0 = \frac{4\pi d}{\lambda}$$

Phase difference between two adjacent periods:$\Delta\phi = \frac{4\pi vT}{\lambda}$

$$\mathrm{v} = \frac{\lambda\Delta\phi}{4\pi T}$$

$$\Delta v = \frac{\lambda}{2N_dT} = \frac{\frac{c}{f_c}}{2N_dT} = \frac{c}{2N_dTf_c} = \frac{3\times10^8}{2\times128\times8\times10^{-6}\times77\times10^9} = 1.9024\mathrm{m/s}$$

Method 2:

$$\omega = \frac{4\pi f_c vT}{c}$$

$$\mathrm{v} = \frac{\omega c}{4\pi f_c T}$$

Within $N_d \times T$ time, the sampled-wave frequency (number of sampled waves divided by time) is F_s, and the number of sampled waves is N_d, so $F_s = \frac{N_d}{N_d\times T} = \frac{1}{T} \longrightarrow \Delta f = \frac{F_s}{N_d}$

$$\Delta\omega = 2\pi\frac{\Delta f}{F_s} = \frac{2\pi}{N_d}$$

$$\Delta v = \frac{\Delta\omega\times c}{4\pi f_c T} = \frac{c}{2f_cN_dT} = 1.9024m/s$$

III. Answers to Problems (a)-(e) and Result Presentation

1. Problem (a)

Given the radar parameters:

- Carrier frequency fc = 77GHz
- Bandwidth B =150MHz
- Chirp period Tchirp = 8us
- Chirp slope K = B / Tchirp = 18.75 MHz/us

Using MATLAB to simulate and sample the mixed signal with following sampling parameters:

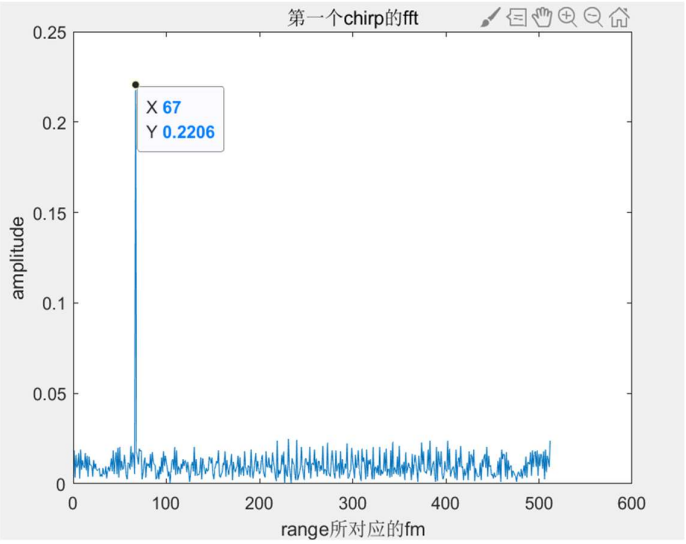
- Number of samples per chirp period: Nr = 1024
- Number of chirps per sample: Nd = 128

Problem a): For a single stationary target (see Blackboard, vector x1), solve for the target distance from radar.

Following the second approach for deriving range resolution, we perform an FFT on the mixed signal x1. The peak position corresponds to the number of bins n occupied by fm among the Nr = 1024 samples.

Thus, R = c/(2B) x n = n = 67 m.

The MATLAB result is as follows:



Code:

```
1. fc=77*10^9;
2. B=150*10^6;
3. Tchirp=8*10^(-6);
```

```
K=B/Tchirp;
Nr=1024;
Nd=128;
signal_d=fft(x1(1:1024))./Nr;
signal_d=abs(signal_d);
signal_d=signal_d(1:(Nr/2));
figure;
plot(signal_d);
xlabel("range");
ylabel("amplitude");
title("FFT of the first chirp");
```

2. Problem (b)

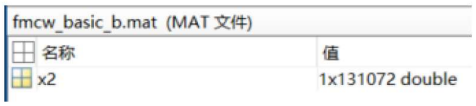
Given the radar parameters:

- Carrier frequency fc = 77GHz
- Bandwidth B =150MHz
- Chirp period Tchirp = 8us
- Chirp slope K = B / Tchirp = 18.75 MHz/us

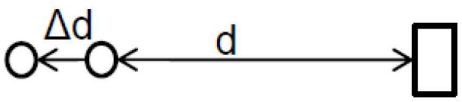
Using MATLAB to simulate and sample the mixed signal with following sampling parameters:

- Number of samples per chirp period: Nr = 1024
- Number of chirps per sample: Nd = 128

Problem b): For a single moving target, given the mixed signal (see Blackboard, vector x2), solve for the target distance and velocity.

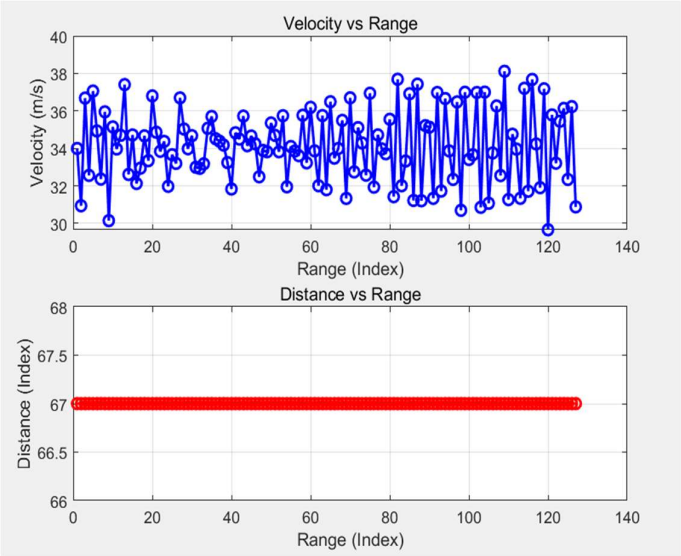


Note: Starting from problem b, a deeper investigation into FMCW radar is required to find the solution. Hint for velocity calculation: due to the Doppler effect, the phase of the IF signal from adjacent chirps will show significant changes, but the frequency change will be less noticeable.



(1) Method 1:

According to the formula$v = \frac{\lambda w}{4\pi T}$, the phase difference of the signal is calculated directly, and the velocity is then calculated directly.



Code:

```
% Parameter settings
fc = 77e9; % carrier frequency
B = 150e6; % bandwidth
Tchirp = 8e-6; % chirp duration
K = B / Tchirp; % slope
c = 3e8; % speed of light
lambda = c / fc; % wavelength
Nr = 1024; % number of samples
Nd = 128; % number of groups

% initialize arrays
w = zeros(1, Nd - 1); % angle
v = zeros(1, Nd - 1); % velocity
dis = zeros(1, Nd - 1); % range

% process the signal
for i = 1:Nd
% compute the FFT of the signal
    signal = fft(x2(1024*(i-1)+1:1024*i)) / Nr;
signal_val = abs(signal); % take absolute value
signal_val = signal_val(1:(Nr / 2)); % keep only the first half

% find the peak position of the signal
    loc = 1;
    for j = 1:Nr/2
        if (signal_val(j) > signal_val(loc))
```

```
            loc = j;
        end
    end

    if i >= 2
% compute the current signal phase
        cur = signal(loc) / abs(signal(loc));
w(i - 1) = angle(cur / pre); % compute angle

% ensure the angle is between 0 and 2π
        while w(i - 1) < 0
            w(i - 1) = w(i - 1) + 2 * pi;
        end

% compute velocity and range
        v(i - 1) = (lambda * w(i - 1)) / (4 * pi * Tchirp);
dis(i - 1) = loc; % range is the peak position
    end

pre = signal(loc) / abs(signal(loc)); % update the previous phase
end

% plot the results
figure;
subplot(2, 1, 1);
plot(v, 'b-o', 'LineWidth', 1.5);
xlabel("Range (Index)");
ylabel("Velocity (m/s)");
title("Velocity vs Range");
grid on;

subplot(2, 1, 2);
plot(dis, 'r-o', 'LineWidth', 1.5);
xlabel("Range (Index)");
ylabel("Distance (Index)");
title("Distance vs Range");
grid on;

disp(['Lambda/4/Tchirp: ', num2str(lambda / (4 * Tchirp))]);

```

(2) Method 2:

The velocity-measurement method is based on 2D-FFT. First, an FFT is performed on the mixed signal along the range dimension. The mixed signal must be converted into matrix form, e.g., 512 x 128. Then, a second FFT is performed on the resulting target column to display the phase information of the object.

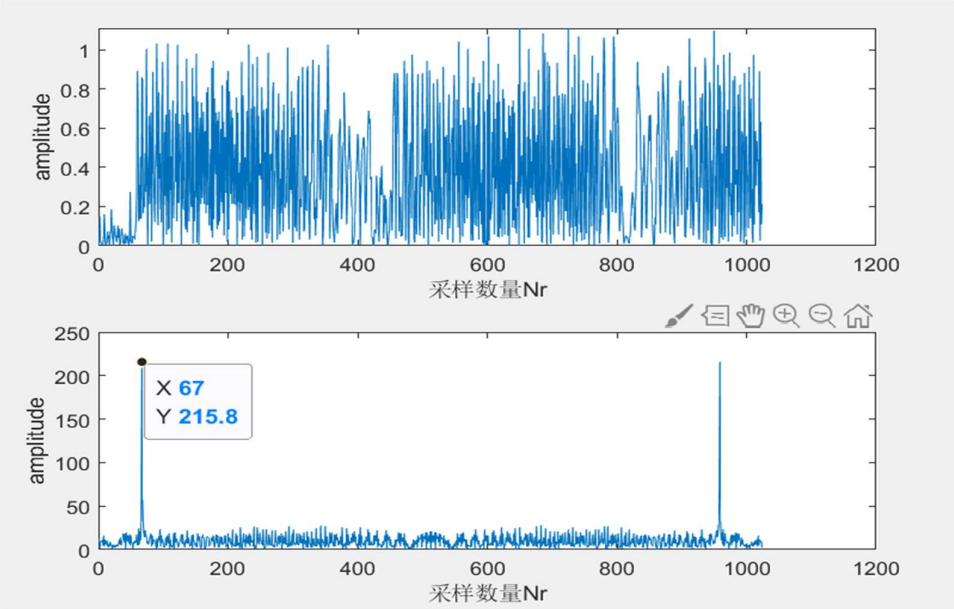
Change the array dimensions using reshape(x2, Nr, Nd), where Nr is the number of rows after reshaping and Nd is the number of columns after reshaping.

First FFT:

plot(abs(Mix_1d(:,1)));

plot(abs(fft(Mix_1d(:,1))));

The result is as follows:



Second FFT:

doppler_axis=v_res*(-Nd/2:Nd/2-1);→speed

Nd is the number of columns of the frequency-domain signal and represents the number of samples in the Doppler direction. This range is usually selected to be symmetric around the frequency-domain center.

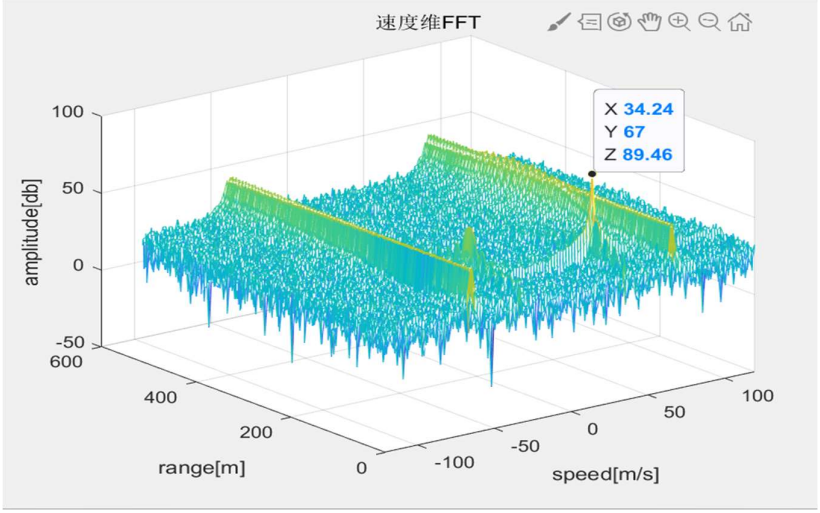
v_res is the velocity increment corresponding to each frequency step. The product of the two represents the velocity-measurement range.

range_axis=range_res*(1:Nr/2);→range

Nr is the number of samples in the range direction. Because of the symmetry of the Fourier transform, only Nr/2 is used as the range-measurement interval.

Use the mesh function to draw the mesh plot -> the position of the peak corresponds to the velocity and range of the object at the current moment.

The plotted result is as follows:



At this time, the velocity is 34.24 m/s.

At this time, the range is 67 m.

Code:

```
fc=77*10^9;
B=150*10^6;
Tchirp=8*10^(-6);
K=B/Tchirp;
c=3*10^8;
lamda=c/fc;
range_res=1;
Nr=1024;
Nd=128;
v_res=lamda/(2*Tchirp*Nd);
Mix_1d=reshape(x2,[Nr,Nd]);
figure(1);
subplot(2,1,1);
plot(abs(Mix_1d(:,1)));
xlabel("number of samples Nr");
ylabel("amplitude");
subplot(2,1,2);
plot(abs(fft(Mix_1d(:,1))));
xlabel("number of samples Nr");
ylabel("amplitude");

%sig_fft=fft(Mix_1d,Nr);
%figure(2);
%mesh(db(abs(sig_fft./max(sig_fft))));
%title("range obtained from the first FFT, i.e., range-domain FFT");
%xlabel("chirps[Nd]");
%ylabel("Range[m]");
%zlabel("amplitude[db]");
```

```
%for i=1:Nr;
    %sig_fft2(i,:)=fftshift(fft(sig_fft(i,:)));
%end
sig_fft2=fft2(Mix_1d,Nr,Nd);
sig_fft3=zeros(Nr/2,Nd);
for i=1:Nr/2
    sig_fft3(i,:)=fftshift(sig_fft2(i,:));
end
doppler_axis=v_res*(-Nd/2:Nd/2-1);
range_axis=range_res*(1:Nr/2);
figure(3);
mesh(doppler_axis,range_axis,db(abs(sig_fft3)));
xlabel("speed[m/s]");
ylabel("range[m]");
zlabel("amplitude[db]");
title("velocity-domain FFT");

```

3. Problem (c)

Given the radar parameters:

- Carrier frequency fc = 2.253 GHz
- Bandwidth B =178 MHz (i.e. working frequency range 2.253 ~ 2.431 GHz)
- Chirp period Tchirp = 1/65 s
- Sampling frequency fs = 50 KHz

Problem c): what are the **measurement ranges and accuracies** for the distance and velocity of a single target? Compare to the radar in 1) basic tasks, which radar in better in those metrics respectively? why?

Analyze this set of parameters: Nr = fs x Tchirp = 769.2, approximately 770.

From the previous analysis, we obtain:

Range-measurement interval: 1-385 m.

Velocity-measurement interval: approximately -2.164 m/s to 2.164 m/s.

Range accuracy: 0.843 m.

Velocity accuracy: assuming a single sampling duration of 0.1 s,

v_{res}=0.666m/s

Analyze the parameters in Problems (a) and (b):

Range-measurement interval: 1-512 m.

Velocity-measurement interval: -121.8 to 119.9.

Range accuracy: 1 m.

Velocity accuracy: 1.9 m/s.

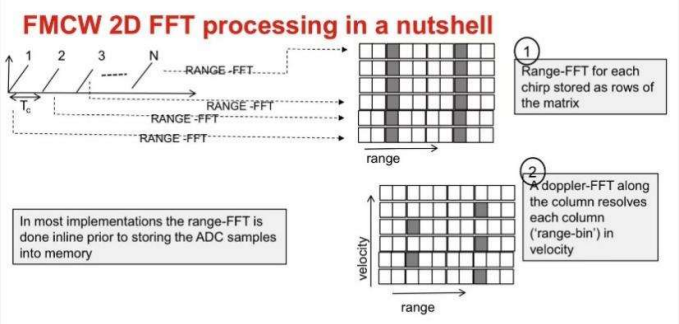
Conclusion: this set of parameters provides higher accuracy in range and time measurement. However, compared with the parameters in Problems (a) and (b), its velocity-measurement range is too small, and the corresponding peak cannot even be obtained in the MATLAB mesh plot. Using the same plotting method, because the two peaks are too close to each other, they overlap, making it impossible to distinguish their specific positions. Therefore, from the perspective of practical needs, the parameters in Problems (a) and (b) are more suitable for measuring the range and velocity of objects over a larger interval, while the parameters in Problem (c) are more suitable for measuring objects whose velocity varies over a very small interval, with stricter limitations.

4. Problem (d)

Given the radar parameters:

- Carrier frequency fc = 2.253 GHz
- Bandwidth B =178 MHz (i.e. working frequency range 2.253 ~ 2.431 GHz)
- Chirp period Tchirp = 1/65 s
- Sampling frequency fs = 50 KHz

Problem d): Select a set of radar parameters, and design a method to measure the distance and velocity of M (where M >=2) moving objects, illustrate the details with a MATLAB simulation case.



Source: TAXAS INSTRUMNETS

（1） Basic principle:

Map the discrete angular-frequency axis to the velocity axis and the frequency axis to the range axis.

（2） Implementation strategy:

① Generate transmitted and received signals.

② Mix the received signals.

③ Reshape the mixed signal to prepare for FFT (reshape).

④ Perform 2D-FFT to analyze the mixed signal.

⑤ Find peaks in the frequency domain.

⑥ Find peaks whose amplitude exceeds 90% of the maximum value.

⑦ Select the first Nr/2 samples and perform frequency shifting.

⑧ Create the coordinate axes for velocity and range.

⑨ Find the positions of the first and second maxima (row and column coordinates).

⑩ Convert the row and column coordinates into actual velocity and range.

(3) Results:

Set target parameters:

range1 = 7.5; % range of target 1 (m)

v1 = 20; % velocity of target 1 (m/s)

range2 = 3; % range of target 2 (m)

v2 = 5; % velocity of target 2 (m/s)

Measurement results:

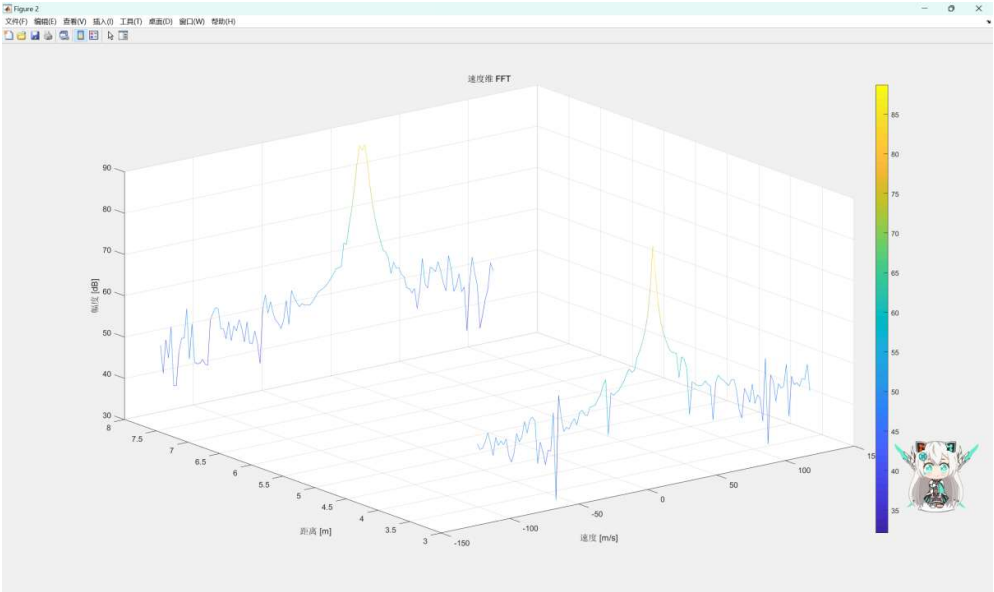
Range of object 1: 3 m.

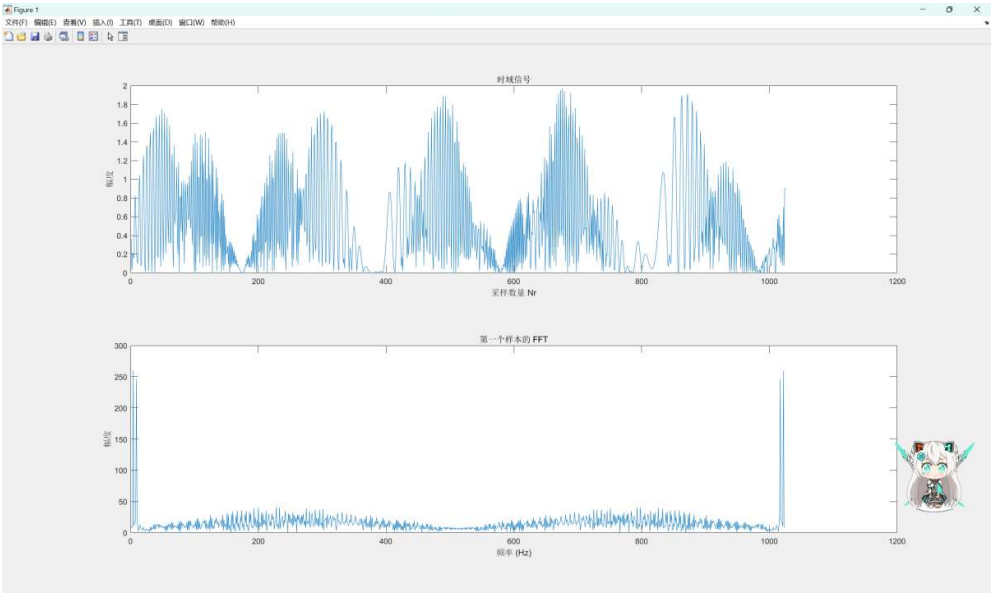
Velocity of object 1: 3.8048 m/s.

Range of object 2: 8 m.

Velocity of object 2: 20.9263 m/s.

It can therefore be seen that the measurement accuracy is very high.





(4) Code:

```
1. % Radar system parameters
2. fc = 77e9; % carrier frequency (Hz)
3. B = 150e6; % bandwidth (Hz)
4. Tchirp = 8e-6; % chirp duration (s)
5. K = B / Tchirp; % chirp slope
6. c = 3e8; % speed of light (m/s)
7. lambda = c / fc; % wavelength (m)
8. range_res = c / (2 * B); % range resolution (m)
9. Nr = 1024; % number of range samples
10. Nd = 128; % number of Doppler samples
11. v_res = lambda / (2 * Tchirp * Nd); % velocity resolution (m/s)
12.
13. % target parameters
```

```
range1 = 7.5; % range of target 1 (m)
v1 = 20; % velocity of target 1 (m/s)
range2 = 3; % range of target 2 (m)
v2 = 50; % velocity of target 2 (m/s)
range1=floor(range1+0.5);
range2=floor(range2+0.5);

% time vector
t = linspace(0, Nd * Tchirp, Nr * Nd);

% compute time delay
tao1 = 2 * (range1 + v1 * t) / c; % time delay of target 1
tao2 = 2 * (range2 + v2 * t) / c; % time delay of target 2

% generate transmitted and received signals
Tx = cos(2 * pi * (fc * t + (K * t.^2) / 2));
Rx1 = cos(2 * pi * (fc * (t - tao1) + (K * (t - tao1).^2) / 2));
Rx2 = cos(2 * pi * (fc * (t - tao2) + (K * (t - tao2).^2) / 2));

% mixed signal
Mix1 = Tx .* Rx1; % mixed signal of target 1
Mix2 = Tx .* Rx2; % mixed signal of target 2
Mix = Mix1 + Mix2; % total mixed signal

% reshape the mixed signal
Mix_1d = reshape(Mix, [Nr, Nd]);

% plot the original signal and its FFT
figure(1);
subplot(2, 1, 1);
plot(abs(Mix_1d(:, 1)));
xlabel("number of samples Nr");
ylabel("amplitude");
title('time-domain signal');

subplot(2, 1, 2);
plot(abs(fft(Mix_1d(:, 1))));
xlabel("frequency (Hz)");
ylabel("amplitude");
title('FFT of the first sample');

% find peaks in the frequency domain
[pks1, locs1] = findpeaks(abs(fft(Mix_1d(:, 1))));
locs2 = []; % initialize the second position array
cnt = 1;

% find peaks whose amplitude exceeds 90% of the maximum value
```

```
for i = 1:length(locs1)
    if(pks1(i) > max(pks1) * 0.9 && locs1(i) <= Nr/2)
        locs2(cnt) = locs1(i);
        cnt = cnt + 1;
    end
end

% perform 2D FFT
sig_fft = fft2(Mix_1d, Nr, Nd);
sig_fft2 = zeros(Nr/2, Nd);

% apply frequency shifting using peak positions
for i = 1:length(locs2)
    sig_fft2(locs2(i), :) = fftshift(sig_fft(locs2(i), :));
end

% create coordinate axes for velocity and range
doppler_axis = v_res * ((-Nd/2:(Nd-1)/2)-1);
range_axis = range_res * ((1:Nr/2)-1);

% find the first and second maxima of the FFT
% find the positions of the first and second maxima
loc_x=1;
loc_y=1;
for i=1:Nr/2
    for j=1:Nd
        if(abs(sig_fft2(i,j))>abs(sig_fft2(loc_x,loc_y)))
            loc_x=i;
            loc_y=j;
        end
    end
end

lc_x=1;
lc_y=1;
for i=1:Nr/2
    for j=1:Nd
        if (abs(sig_fft2(i,j))>abs(sig_fft2(lc_x,lc_y)))&&((i==loc_x)==0)
            lc_x=i;
            lc_y=j;
        end
    end
end

% display maximum positions
disp(['first maximum position (row, column): (', num2str(loc_x), ', ', num2str(loc_y), ')']);
disp(['second maximum position (row, column): (', num2str(lc_x), ', ', num2str(lc_y), ')']);
```

```
108.
109. % convert positions to actual range and velocity
110. max_distance = range_axis(loc_x); % range corresponding to the first maximum
111. max_velocity = doppler_axis(loc_y); % velocity corresponding to the first maximum
112.
113. second_distance = range_axis(lc_x); % range corresponding to the second maximum
114. second_velocity = doppler_axis(lc_y); % velocity corresponding to the second maximum
115.
116. % display results
117. disp(['range corresponding to the first maximum: ', num2str(max_distance), ' m']);
118. disp(['velocity corresponding to the first maximum: ', num2str(max_velocity), ' m/s']);
119. disp(['range corresponding to the second maximum: ', num2str(second_distance), ' m']);
120. disp(['velocity corresponding to the second maximum: ', num2str(second_velocity), ' m/s']);
121.
122. % plot results
123. figure(2);
124. mesh(doppler_axis, range_axis, db(abs(sig_fft2)));
125. xlabel("velocity [m/s]");
126. ylabel("range [m]");
127. zlabel("amplitude [dB]");
128. title("velocity-domain FFT");
129. colorbar; % add a colorbar to help interpret amplitude
130.
```

5. Problem (e)

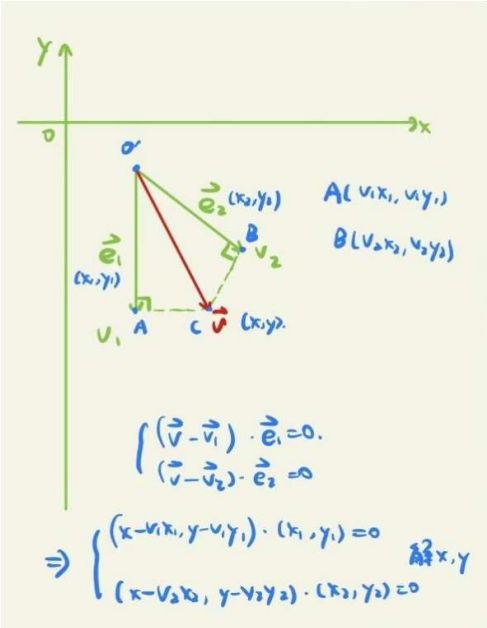
(1) Implementation strategy (using two targets and three receivers as an example):

1) Apply the method in (d) three times to find the projected velocities and ranges of the two targets relative to the three receivers.

2) Find the points: using the three receivers as circle centers and the measured ranges as radii, draw two circles for each receiver and find their intersections.

3) Compute the resultant velocity, including magnitude and direction. The method is shown in the figure on the right and solves a system of two linear equations.

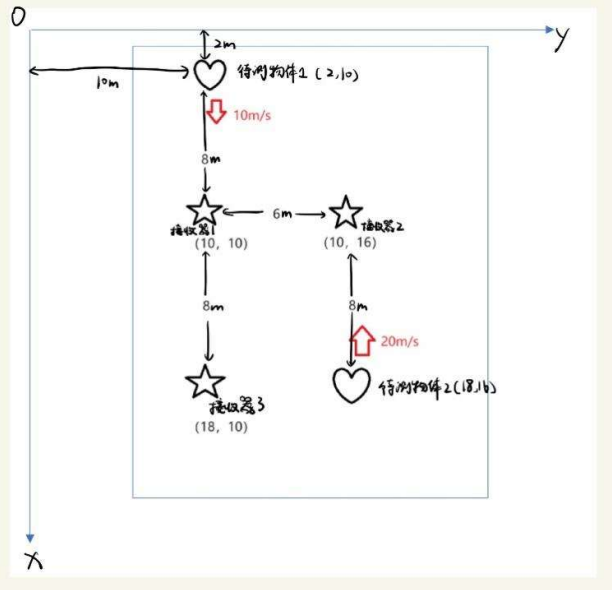
In practical situations, receivers that are close to one another and lie on the same straight line are usually used. However, if a receiver-array design is required, this method can also be implemented. Therefore, we use receivers that are not collinear and not too close to one another as an example, only to demonstrate the principle and verify its feasibility and correctness. Users of the code can design the receiver arrangement themselves.



Principle of Velocity Composition

(2) Results:

```
1. The coordinates of target 1 are as follows (row, column):
2.      2
3.
4.     10
5.
6. The coordinates of target 2 are as follows (row, column):
7.     18
8.
9.     16
10.
11. Magnitude of the velocity of target 1 and its vertical and horizontal velocity components
(downward and rightward directions are positive):
12.    11.6928
13.
14.    11.4144
15.    -2.5365
16.
17. Magnitude of the velocity of target 2 and its vertical and horizontal velocity components
(downward and rightward directions are positive):
18.    21.2694
19.
20.   -20.9263
21.    -3.8048
22.
```



Test Design

It can therefore be seen that the accuracy of position and velocity measurement is very high.

（3） Code:

```
% Radar system parameters
fc = 77e9; % carrier frequency (Hz)
B = 150e6; % bandwidth (Hz)
Tchirp = 8e-6; % chirp duration (s)
K = B / Tchirp; % chirp slope
c = 3e8; % speed of light (m/s)
lambda = c / fc; % wavelength (m)
range_res = c / (2 * B); % range resolution (m)
Nr = 1024; % number of range samples
Nd = 128; % number of Doppler samples
v_res = lambda / (2 * Tchirp * Nd); % velocity resolution (m/s)
d1=6;
d2=8;

%% process the first group of targets

% target parameters
range1 = 8; % range of target 1 (m)
v1 = 10; % velocity of target 1 (m/s)
range2 = 10; % range of target 2 (m)
v2 = 16; % velocity of target 2 (m/s)
range1=floor(range1);
range2=floor(range2);

% time vector
t = linspace(0, Nd * Tchirp, Nr * Nd);
```

```
27.
28. % compute time delay
29. tao1 = 2 * (range1 + v1 * t) / c; % time delay of target 1
30. tao2 = 2 * (range2 + v2 * t) / c; % time delay of target 2
31.
32. % generate transmitted and received signals
33. Tx = cos(2 * pi * (fc * t + (K * t.^2) / 2));
34. Rx1 = cos(2 * pi * (fc * (t - tao1) + (K * (t - tao1).^2) / 2));
35. Rx2 = cos(2 * pi * (fc * (t - tao2) + (K * (t - tao2).^2) / 2));
36.
37. % mixed signal
38. Mix1 = Tx .* Rx1; % mixed signal of target 1
39. Mix2 = Tx .* Rx2; % mixed signal of target 2
40. Mix = Mix1 + Mix2; % total mixed signal
41.
42. % reshape the mixed signal
43. Mix_1d = reshape(Mix, [Nr, Nd]);
44.
45. % plot the original signal and its FFT
46. figure(1);
47. subplot(2, 1, 1);
48. plot(abs(Mix_1d(:, 1)));
49. xlabel("number of samples Nr");
50. ylabel("amplitude");
51. title('time-domain signal of the first group of targets');
52.
53. subplot(2, 1, 2);
54. plot(abs(fft(Mix_1d(:, 1))));
55. xlabel("frequency (Hz)");
56. ylabel("amplitude");
57. title('FFT of the first sample of the first group of targets');
58.
59. % find peaks in the frequency domain
60. [pks1, locs1] = findpeaks(abs(fft(Mix_1d(:, 1))));
61. locs2 = []; % initialize the second position array
62. cnt = 1;
63.
64. % find peaks whose amplitude exceeds 90% of the maximum value
65. for i = 1:length(locs1)
66.     if(pks1(i) > max(pks1) * 0.9 && locs1(i) <= Nr/2)
67.         locs2(cnt) = locs1(i);
68.         cnt = cnt + 1;
69.     end
70. end
71.
72. % perform 2D FFT
73. sig_fft = fft2(Mix_1d, Nr, Nd);
```

```
sig_fft2 = zeros(Nr/2, Nd);

% apply frequency shifting using peak positions
for i = 1:length(locs2)
    sig_fft2(locs2(i), :) = fftshift(sig_fft(locs2(i), :));
end

% create coordinate axes for velocity and range
doppler_axis = v_res * ((-Nd/2:(Nd-1)/2)-1);
range_axis = range_res * ((1:Nr/2)-1);

% find the first and second maxima of the FFT
loc_x=1;
loc_y=1;
for i=1:Nr/2
    for j=1:Nd
        if(abs(sig_fft2(i,j))>abs(sig_fft2(loc_x,loc_y)))
            loc_x=i;
            loc_y=j;
        end
    end
end

%display(loc_x);
%display(loc_y);

lc_x=1;
lc_y=1;
for i=1:Nr/2
    for j=1:Nd
        if (abs(sig_fft2(i,j))>abs(sig_fft2(lc_x,lc_y)))&&((i==loc_x)==0)
            lc_x=i;
            lc_y=j;
        end
    end
end

% display maximum positions
disp(['first maximum position (row, column): (', num2str(loc_x), ', ', num2str(loc_y), ')']);
disp(['second maximum position (row, column): (', num2str(lc_x), ', ', num2str(lc_y), ')']);

% convert positions to actual range and velocity
max_distance = range_axis(loc_x); % range corresponding to the first maximum
max_velocity = doppler_axis(loc_y); % velocity corresponding to the first maximum

second_distance = range_axis(lc_x); % range corresponding to the second maximum
second_velocity = doppler_axis(lc_y); % velocity corresponding to the second maximum
```

```
121.
122. % display results
123. disp(['Rx1:range corresponding to the first-group maximum: ', num2str(max_distance), ' m']);
124. disp(['Rx1:velocity corresponding to the first-group maximum: ', num2str(max_velocity), '
m/s']);
125. disp(['Rx1:range corresponding to the second-group maximum: ', num2str(second_distance), '
m']);
126. disp(['Rx1:velocity corresponding to the second-group maximum: ', num2str(second_velocity), '
m/s']);
127.
128. % plot results
129. figure(2);
130. mesh(doppler_axis, range_axis, db(abs(sig_fft2)));
131. xlabel("velocity [m/s]");
132. ylabel("range [m]");
133. zlabel("amplitude [dB]");
134. title("velocity-domain FFT of the first group of targets");
135. colorbar; % add a colorbar to help interpret amplitude
136.
137.
138. %% process the second group of targets
139. % target parameters
140. arange1 = 10; % range of target 1 (m)
141. av1 = 8; % velocity of target 1 (m/s)
142. arange2 = 8; % range of target 2 (m)
143. av2 = 20; % velocity of target 2 (m/s)
144. arange1=floor(arange1);
145. arange2=floor(arange2);
146.
147. % time vector
148. at = linspace(0, Nd * Tchirp, Nr * Nd);
149.
150. % compute time delay
151. atao1 = 2 * (arange1 + av1 * at) / c; % time delay of target 1
152. atao2 = 2 * (arange2 + av2 * at) / c; % time delay of target 2
153.
154. % generate transmitted and received signals
155. aTx = cos(2 * pi * (fc * at + (K * at.^2) / 2));
156. aRx1 = cos(2 * pi * (fc * (at - atao1) + (K * (at - atao1).^2) / 2));
157. aRx2 = cos(2 * pi * (fc * (at - atao2) + (K * (at - atao2).^2) / 2));
158.
159. % mixed signal
160. aMix1 = aTx .* aRx1; % mixed signal of target 1
161. aMix2 = aTx .* aRx2; % mixed signal of target 2
162. aMix = aMix1 + aMix2; % total mixed signal
163.
164. % reshape the mixed signal
```

```
aMix_1d = reshape(aMix, [Nr, Nd]);

% plot the original signal and its FFT
figure(3);
subplot(2, 1, 1);
plot(abs(aMix_1d(:, 1)));
xlabel("number of samples Nr");
ylabel("amplitude");
title('time-domain signal of the first group of targets');

subplot(2, 1, 2);
plot(abs(fft(aMix_1d(:, 1))));
xlabel("frequency (Hz)");
ylabel("amplitude");
title('FFT of the first sample of the first group of targets');

% find peaks in the frequency domain
[pks1, locs1] = findpeaks(abs(fft(aMix_1d(:, 1))));
locs2 = []; % initialize the second position array
cnt = 1;

% find peaks whose amplitude exceeds 90% of the maximum value
for i = 1:length(locs1)
    if(pks1(i) > max(pks1) * 0.9 && locs1(i) <= Nr/2)
        locs2(cnt) = locs1(i);
        cnt = cnt + 1;
    end
end

% perform 2D FFT
asig_fft = fft2(aMix_1d, Nr, Nd);
asig_fft2 = zeros(Nr/2, Nd);

% apply frequency shifting using peak positions
for i = 1:length(locs2)
    asig_fft2(locs2(i), :) = fftshift(asig_fft(locs2(i), :));
end

% create coordinate axes for velocity and range
adoppler_axis = v_res * ((-Nd/2:(Nd-1)/2)-1);
arange_axis = range_res * ((1:Nr/2)-1);

% find the first and second maxima of the FFT
aloc_x=1;
aloc_y=1;
for i=1:Nr/2
    for j=1:Nd
```

```
        if(abs(asig_fft2(i,j))>abs(asig_fft2(aloc_x,aloc_y)))
            aloc_x=i;
            aloc_y=j;
        end
    end
end

%display(aloc_x);
%display(aloc_y);

alc_x=1;
alc_y=1;
for i=1:Nr/2
    for j=1:Nd
        if (abs(asig_fft2(i,j))>abs(asig_fft2(alc_x,alc_y)))&&((i==aloc_x)==0)
            alc_x=i;
            alc_y=j;
        end
    end
end

% display maximum positions
disp(['first maximum position (row, column): (', num2str(aloc_x), ', ', num2str(aloc_y),
')']);
disp(['second maximum position (row, column): (', num2str(alc_x), ', ', num2str(alc_y),
')']);

% convert positions to actual range and velocity
amax_distance = arange_axis(aloc_x); % range corresponding to the first maximum
amax_velocity = adoppler_axis(aloc_y); % velocity corresponding to the first maximum

asecond_distance = arange_axis(alc_x); % range corresponding to the second maximum
asecond_velocity = adoppler_axis(alc_y); % velocity corresponding to the second maximum

% display results
disp(['Rx2:range corresponding to the first-group maximum: ', num2str(amax_distance), ' m']);
disp(['Rx2:velocity corresponding to the first-group maximum: ', num2str(amax_velocity), '
m/s']);
disp(['Rx2:range corresponding to the second-group maximum: ', num2str(asecond_distance), '
m']);
disp(['Rx2:velocity corresponding to the second-group maximum: ', num2str(asecond_velocity),
' m/s']);

% plot results
figure(4);
mesh(adoppler_axis, arange_axis, db(abs(asig_fft2)));
xlabel("velocity [m/s]");
```

```
ylabel("range [m]");
zlabel("amplitude [dB]");
title("velocity-domain FFT of the first group of targets");
colorbar; % add a colorbar to help interpret amplitude

%% process the third group of targets
% target parameters
brange1 = 16; % range of target 1 (m)
bv1 = 10; % velocity of target 1 (m/s)
brange2 = 6; % range of target 2 (m)
bv2 = 0; % velocity of target 2 (m/s)
brange1=floor(brange1);
brange2=floor(brange2);

% time vector
bt = linspace(0, Nd * Tchirp, Nr * Nd);

% compute time delay
btao1 = 2 * (brange1 + bv1 * bt) / c; % time delay of target 1
btao2 = 2 * (brange2 + bv2 * bt) / c; % time delay of target 2

% generate transmitted and received signals
bTx = cos(2 * pi * (fc * bt + (K * bt.^2) / 2));
bRx1 = cos(2 * pi * (fc * (bt - btao1) + (K * (bt - btao1).^2) / 2));
bRx2 = cos(2 * pi * (fc * (bt - btao2) + (K * (bt - btao2).^2) / 2));

% mixed signal
bMix1 = bTx .* bRx1; % mixed signal of target 1
bMix2 = bTx .* bRx2; % mixed signal of target 2
bMix = bMix1 + bMix2; % total mixed signal

% reshape the mixed signal
bMix_1d = reshape(bMix, [Nr, Nd]);

% plot the original signal and its FFT
figure(5);
subplot(2, 1, 1);
plot(abs(bMix_1d(:, 1)));
xlabel("number of samples Nr");
ylabel("amplitude");
title('time-domain signal of the first group of targets');

subplot(2, 1, 2);
plot(abs(fft(bMix_1d(:, 1))));
xlabel("frequency (Hz)");
ylabel("amplitude");
title('FFT of the first sample of the first group of targets');
```

```
301.
302. % find peaks in the frequency domain
303. [pks1, locs1] = findpeaks(abs(fft(bMix_1d(:, 1))));
304. locs2 = []; % initialize the second position array
305. cnt = 1;
306.
307. % find peaks whose amplitude exceeds 90% of the maximum value
308. for i = 1:length(locs1)
309.     if(pks1(i) > max(pks1) * 0.9 && locs1(i) <= Nr/2)
310.         locs2(cnt) = locs1(i);
311.         cnt = cnt + 1;
312.     end
313. end
314.
315. % perform 2D FFT
316. bsig_fft = fft2(bMix_1d, Nr, Nd);
317. bsig_fft2 = zeros(Nr/2, Nd);
318.
319. % apply frequency shifting using peak positions
320. for i = 1:length(locs2)
321.     bsig_fft2(locs2(i), :) = fftshift(bsig_fft(locs2(i), :));
322. end
323.
324. % create coordinate axes for velocity and range
325. bdoppler_axis = v_res * ((-Nd/2:(Nd-1)/2)-1);
326. brange_axis = range_res * ((1:Nr/2)-1);
327.
328. % find the first and second maxima of the FFT
329. bloc_x=1;
330. bloc_y=1;
331. for i=1:Nr/2
332.     for j=1:Nd
333.         if(abs(bsig_fft2(i,j))>abs(bsig_fft2(bloc_x,bloc_y)))
334.             bloc_x=i;
335.             bloc_y=j;
336.         end
337.     end
338. end
339.
340. %display(aloc_x);
341. %display(aloc_y);
342.
343. blc_x=1;
344. blc_y=1;
345. for i=1:Nr/2
346.     for j=1:Nd
347.         if (abs(bsig_fft2(i,j))>abs(bsig_fft2(blc_x,blc_y)))&&((i==bloc_x)==0)
```

```
            blc_x=i;
            blc_y=j;
        end
    end
end

% display maximum positions
disp(['first maximum position (row, column): (', num2str(bloc_x), ', ', num2str(bloc_y), ')']);
disp(['second maximum position (row, column): (', num2str(blc_x), ', ', num2str(blc_y), ')']);

% convert positions to actual range and velocity
bmax_distance = brange_axis(bloc_x); % range corresponding to the first maximum
bmax_velocity = bdoppler_axis(bloc_y); % velocity corresponding to the first maximum

bsecond_distance = brange_axis(blc_x); % range corresponding to the second maximum
bsecond_velocity = bdoppler_axis(blc_y); % velocity corresponding to the second maximum

% display results
disp(['Rx3:range corresponding to the first-group maximum: ', num2str(bmax_distance), ' m']);
disp(['Rx3:velocity corresponding to the first-group maximum: ', num2str(bmax_velocity), ' m/s']);
disp(['Rx3:range corresponding to the second-group maximum: ', num2str(bsecond_distance), ' m']);
disp(['Rx3:velocity corresponding to the second-group maximum: ', num2str(bsecond_velocity), ' m/s']);

% plot results
figure(6);
mesh(bdoppler_axis, brange_axis, db(abs(bsig_fft2)));
xlabel("velocity [m/s]");
ylabel("range [m]");
zlabel("amplitude [dB]");
title("velocity-domain FFT of the first group of targets");
colorbar; % add a colorbar to help interpret amplitude

%% find points

if max_distance<second_distance
    cur=max_distance;
    max_distance=second_distance;
    second_distance=cur;
    cur=max_velocity;
    max_velocity=second_velocity;
    second_velocity=cur;
end
```

```
390.
391. if amax_distance<asecond_distance
392.     cur=amax_distance;
393.     amax_distance=asecond_distance;
394.     asecond_distance=cur;
395.     cur=amax_velocity;
396.     amax_velocity=asecond_velocity;
397.     asecond_velocity=cur;
398. end
399.
400. if bmax_distance<bsecond_distance
401.     cur=bmax_distance;
402.     bmax_distance=bsecond_distance;
403.     bsecond_distance=cur;
404.     cur=bmax_velocity;
405.     bmax_velocity=bsecond_velocity;
406.     bsecond_velocity=cur;
407. end
408.
409. y=max_distance;
410. x=max_distance;
411. ay=max_distance;
412. ax=max_distance+d1;
413. by=max_distance+d2;
414. bx=max_distance;
415. res=1e5;
416. cy=0;
417. cx=0;
418. for i=1:max_distance*2
419.     for j=1:max_distance+d1+amax_distance
420.         dis1=min(abs(sqrt((y-i)*(y-i)+(x-j)*(x-j))-max_distance),abs(sqrt((y-i)*(y-i)+(x-
j)*(x-j))-second_distance));
421.         dis2=min(abs(sqrt((ay-i)*(ay-i)+(ax-j)*(ax-j))-amax_distance),abs(sqrt((ay-i)*(ay-
i)+(ax-j)*(ax-j))-asecond_distance));
422.         dis3=min(abs(sqrt((by-i)*(by-i)+(bx-j)*(bx-j))-bmax_distance),abs(sqrt((by-i)*(by-
i)+(bx-j)*(bx-j))-bsecond_distance));
423.         if(dis1+dis2+dis3<res)
424.             res=dis1+dis2+dis3;
425.             cy=i;
426.             cx=j;
427.         end
428.     end
429. end
430. disp('The coordinates of target 1 are as follows (row, column)');
431. disp(cy);
432. disp(cx);
433. dy=cy;
```

```
dx=cx;

res=1e5;
cy=0;
cx=0;
for i=1:max_distance*2
    for j=1:max_distance+d1+amax_distance
        if i==dy&&j==dx
            continue;
        end

        dis1=min(abs(sqrt((y-i)*(y-i)+(x-j)*(x-j))-max_distance),abs(sqrt((y-i)*(y-i)+(x-
j)*(x-j))-second_distance));
        dis2=min(abs(sqrt((ay-i)*(ay-i)+(ax-j)*(ax-j))-amax_distance),abs(sqrt((ay-i)*(ay-
i)+(ax-j)*(ax-j))-asecond_distance));
        dis3=min(abs(sqrt((by-i)*(by-i)+(bx-j)*(bx-j))-bmax_distance),abs(sqrt((by-i)*(by-
i)+(bx-j)*(bx-j))-bsecond_distance));
        if(dis1+dis2+dis3<res)
            res=dis1+dis2+dis3;
            cy=i;
            cx=j;
        end
    end
end
disp('The coordinates of target 2 are as follows (row, column)');
disp(cy);
disp(cx);
%% compute resultant velocity
% define receiver coordinates
R1 = [y; x]; % coordinates of receiver 1
R2 = [ay; ax]; % coordinates of receiver 2
R3 = [by; bx]; % coordinates of receiver 3

% compute the original velocity of the first target coordinate
T1 = [dy; dx]; % first target coordinate
v_rel1 = calculate_relative_velocity(max_distance, second_distance, dy, dx, y, x,
max_velocity, second_velocity);
v_rel2 = calculate_relative_velocity(amax_distance, asecond_distance, dy, dx, ay, ax,
amax_velocity, asecond_velocity);
v_rel3 = calculate_relative_velocity(bmax_distance, bsecond_distance, dy, dx, by, bx,
bmax_velocity, bsecond_velocity);
dR1 = R1 - T1; dR1 = dR1 / norm(dR1);
dR2 = R2 - T1; dR2 = dR2 / norm(dR2);
dR3 = R3 - T1; dR3 = dR3 / norm(dR3);

x1=dR1(1);y1=dR1(2);
x2=dR2(1);y2=dR2(2);
```

```
c = (v_rel1*x1*x1+v_rel1*y1*y1);
f = (v_rel2*x2*x2+v_rel2*y2*y2);

% define the coefficient matrix and constant vector
A = [x1, y1; x2, y2]; % coefficient matrix
B = [c; f]; % constant vector

% use the backslash operator to solve the equation
solution = A \ B;

v_val=sqrt(solution(1)^2+solution(2)^2);
v_ang=atan(solution(2)/solution(1));
disp('Magnitude of the velocity of target 1 and its vertical and horizontal velocity components (downward and rightward directions are positive)');
disp(v_val);
disp(solution);

% compute the second target coordinate (same logic, but with different parameters)
T2 = [cy; cx]; % second target coordinate
v_rel1 = calculate_relative_velocity(max_distance, second_distance, cy, cx, y, x, max_velocity, second_velocity);
v_rel2 = calculate_relative_velocity(amax_distance, asecond_distance, cy, cx, ay, ax, amax_velocity, asecond_velocity);
v_rel3 = calculate_relative_velocity(bmax_distance, bsecond_distance, cy, cx, by, bx, bmax_velocity, bsecond_velocity);
dR1 = R1 - T2; dR1 = dR1 / norm(dR1);
dR2 = R2 - T2; dR2 = dR2 / norm(dR2);
dR3 = R3 - T2; dR3 = dR3 / norm(dR3);

x1=dR1(1);y1=dR1(2);
x2=dR2(1);y2=dR2(2);
c = (v_rel1*x1*x1+v_rel1*y1*y1);
f = (v_rel2*x2*x2+v_rel2*y2*y2);

% define the coefficient matrix and constant vector
A = [x1, y1; x2, y2]; % coefficient matrix
B = [c; f]; % constant vector

% use the backslash operator to solve the equation
solution = A \ B;

v_val=sqrt(solution(1)^2+solution(2)^2);
v_ang=atan(solution(2)/solution(1));
disp('Magnitude of the velocity of target 2 and its vertical and horizontal velocity components (downward and rightward directions are positive)');
disp(v_val);
disp(solution);
```

```
517.
518. function v_rel = calculate_relative_velocity(max_distance, second_distance, dy, dx,
receiver_y, receiver_x, max_velocity, second_velocity)
519.     distance = sqrt((receiver_y - dy)^2 + (receiver_x - dx)^2);
520.     if abs(max_distance - distance) < abs(second_distance - distance)
521. v_rel = max_velocity; % maximum velocity of the target relative to the receiver
522.     else
523. v_rel = second_velocity; % second velocity of the target relative to the receiver
524.     end
525. end
526.
```

IV. Project Summary

1. Division of Work and Contributions

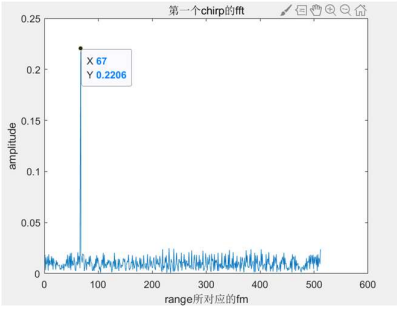
Liu Haichuan was responsible for completing Problems (b), (d), and (e).

Wu Tangzi was responsible for completing Problems (a), (b), (c), and (d).

The contribution ratio of the two members was 50% each.

2. Problems Encountered and Lessons Learned

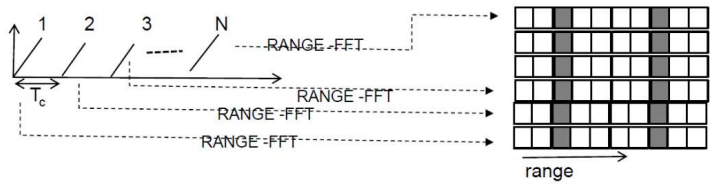
(1)At first, we were not clear about the significance of performing the Fourier transform. We were not able to understand why the waveform took that shape even after producing it through the relevant operations, especially how a certain peak was generated and why the x-coordinate of that peak should be selected as fm. This series of questions was indeed confusing. In fact, it involves the most basic knowledge from the theoretical course. Since the signal involved in mixing is a cosine signal, its Fourier transform pair is $\pi[\delta(w-w_0)+\delta(w+w_0)]$. Based on this theoretical foundation, the two peaks shown correspond respectively to the positive and negative half-axes of w_0, which reasonably explains why this x-coordinate is selected to calculate the object range.



(2) When using the second set of parameters to implement the content related to Problems (d) and (e), we initially assumed that, except for the mixing process being largely similar to that in Problems (a) and (b), the remaining steps would not differ significantly. However, the actual results were not as expected: in the 2D-FFT results, no peak position could be found at all. We spent a great deal of time troubleshooting and experimenting with this issue, even switching to methods such as the findpeaks function, but none produced satisfactory results, and the problem remained unresolved. After deeper analysis and

investigation, we realized that the velocity-measurement range had already been limited by the parameters. Regardless of the value assigned to Nd, under such a narrow velocity-measurement range, the object velocity could not be effectively resolved. This experience gave us an important lesson: in circuit design, in addition to ensuring that the circuit diagram is drawn accurately, the selection of reasonable parameters is also crucial. It has a significant impact on whether the entire circuit can achieve the expected function and effect.

(3) Doppler FFT was relatively difficult to understand, including why Doppler FFTcan be used to measure velocity.



多普勒FFT就是对矩阵竖着做一维FFT，然后将每一列叠加起来。

由多普勒效应可知，物体运动会改变雷达回波的频率，其反映出来的是相位的变化。

从上面的公式可得 $v = \frac{\lambda\phi}{4\pi T_c}$ ，其中 $\phi = 2\pi f$ 也就是回波的频率与物体的速度有关。

当物体向雷达靠近时，频率变大，当物体向雷达远离时频率变小，产生的频率差叫做多普勒频率，多普勒频率可以反映在相位 ϕ 的变化上，满足 $f_d = \frac{d\phi}{2\pi dt} = \frac{2v}{\lambda}$ 。

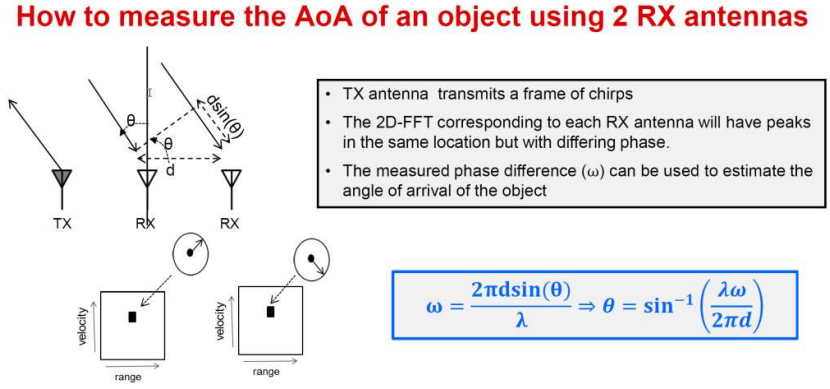
因速度产生了频率变换，对多个chirp信号做多普勒FFT，可以分辨出物体的不同速度。

*Source: **Fatih** from Zhihu.com*

(4) When solving for velocity, we kept trying to rely on code implementation, but eventually realized that manual mathematical derivation was still necessary. The lessons were: 1) do not rely excessively on computers; the human brain is very useful; and 2) brute force can sometimes produce surprising results.

3. Futher Discussion and Thoughts

In Problem (e), we originally wanted to use the strategy shown in the figure to solve for velocity, but the results were never satisfactory. Subsequent theoretical analysis showed that because the ωrange is limited, the θvalue range is also very small. Therefore, we suspected that there was an issue with this method, unless the ωrange is unrestricted. However, this is difficult to implement in code because the complex argument is limited).



Source:TAXAS INSTRUMENTS

V. Acknowledgements

We sincerely thank Teacher Li Jiamin for her careful guidance throughout the semester, which gave us a deep understanding and awareness of signal-and-system-related knowledge. Teacher Li attaches great importance to students' feedback and listens carefully to both questions raised in class and communication after class. In order to provide students with a better classroom experience, Teacher Li even purchased equipment at her own expense. In class, she used this equipment for demonstrations and explanations according to the course content and students' actual needs. This not only helped students understand the knowledge more intuitively, but also greatly enhanced the interest and interactivity of the class. We are deeply grateful for all of this.

The project component provided us with a good platform for combining theoretical knowledge with practice. In this process, by completing each project task, we explored professional knowledge in depth. We not only learned a great deal of professional knowledge, but also exercised our practical ability and teamwork skills. Everyone communicated with one another and made progress together during the project, gained a great deal, and developed a deeper understanding and mastery of professional knowledge.

We sincerely hope that the knowledge learned in this course can be internalized into our potential in microelectronics-related fields, so that at some future moment when we need it, we can recall what we learned in the Signal and Systems course in the fall semester of our sophomore year. May this knowledge help us and push us one step further toward the truth.

Though we are not especially gifted, after passing through this autumn, we have finally gained some understanding.