گزارش کار پروژه سیگنال ها و سیستم ها
دکتر امینی

محمد صدرا حسینی
۴۰۱۱۱۰۴۴۸

۱ لود کردن فیلتر ها و پخش فایل صوتی رمزنگاری شده

ابتدا با استفاده از کامند $load$ فایل فیلتر ها را روی $workspace$ لود می کنیم. سپس ابتدا با کامند $audioread$ فایل صوتی $encode$ شده را می خوانیم که خروجی این تابع دو متغیر پهنای فرکانسی سیگنال ظبط شده و نیز کانال های سیگنال صوتی است(در اینجا یک کانال فقط داریم). برای پخش صدا نیز باید از دستور $play$ استفاده کنیم که ورودی آن، خروجی تابع $audioplayer$ است که این تابع همان دو خروجی تابع $audioread$ است. کد در تصویر زیر آمده است

```
%% in this part, we load the sound file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[y , fs] = audioread('testmusic.wav');
load("filters.mat")

player = audioplayer(y , fs);

%play the sound
play(player)

%%%%%%%%%%%%
%define matrix A
A            = [ 0   0   1   0   0 ; ...
                 0   0   0   0   1 ; ...
                 0   0   0   1   0 ; ...
                .05  0   0   0   0 ; ...
                 0   1   0   0   0];

%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% You have to fill in this part     |   |          %
%                                   |   |          %
%                                  \     /         %
%                                   \   /          %
%                                    \ /           %
%                                     .            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

شکل ۱: لود کردن فیلتر ها و پخش فایل صوتی رمزنگاری شده

۲ معکوس ماتریس تبدیل

ماتریس تبدیلی که از آن برای جابجایی بازه های فرکانسی استفاده کردیم در ماتریس A ذخیره کردیم. برای محاسبه ماتریس معکوس می بایست از دستور $inv(A)$ استفاده کنیم که در صورت معکوس پذیر بودن، خروجی معکوس

ماتریس ورودی خواهد بود. کد و خروجی آن در تصویر زیر مشخص است

```
%% computing the inverse matrix
% you should descramble band acording to inverse matrix
%%%%%%%%%%%%%%%%%%%%%%%%
Ainv = zeros(size(A));
Ainv = inv(A);
disp(Ainv)
```

شکل ۲: محاسبه ماتریس معکوس با استفاده از دستور inv

0	0	0	20	0
0	0	0	0	1
1	0	0	0	0
0	0	1	0	0
0	1	0	0	0

شکل ۳: معکوس ماتریس تبدیل

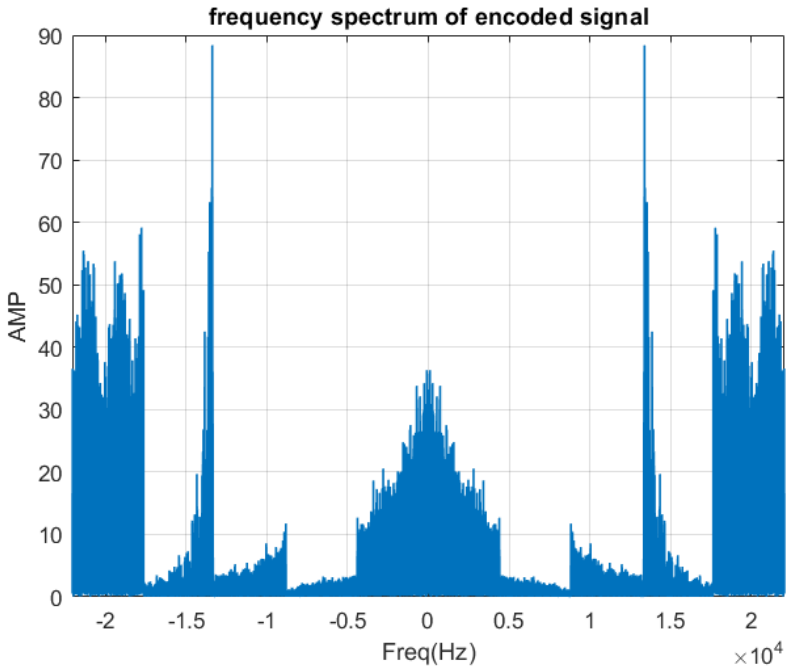
۳ محاسبه و رسم طیف فرکانسی سیگنال رمزنگاری شده

ابتدا با استفاده توامان از دستور های fft و $fftshift$ تبدیل فوریه سیگنال ورودی را محاسبه می کنیم. برای رسم تبدیل فوریه می بایست بازه فرکانسی را درست بدهیم. کل پهنای فرکانس f_s است. کل تعداد نقاطی هم که از خروجی تبدیل فوریه داریم برابر N است. پس کافیست این N داده را به طور متقارن روی بازه f_s پخش کنیم.در نهایت با استفاده از دستور $plot$ شکل را می کشیم. با استفاده از دستور $gridon$ شکل را مش بندی می کنیم. با استفاده از دستور $title$ برای نمودار یک عنوان مناسب انتخاب کرده و نیز با

دستور های $xlabel$ و $ylabel$ محور ها را نشان گذاری می کنیم. استفاده از دستور $xlim$ نیز می توان تعیین کرد وقتی پنجره نمودار باز می شود، محور افقی نمودار چه بازه ای از اعداد را نشان بدهد که چون پهنای باند فرکانسی سیگنال را داریم، به طور متقارن قرار می دهیم. در تصویر زیر کد و خروجی آن آمده است

```
%% spectrum of sound
%%%%%%%%%%%%%%%%%%%%%%%%
% use FFT function defined in example
figure;
Y = fftshift(fft(y));
N = numel(y);
freq = (-N / 2 : N / 2 - 1) / N * fs;
plot(freq, (abs(Y)) , 'LineWidth' , 1) ;
grid on
title('frequency spectrum of encoded signal');
ylabel('AMP');
xlabel('Freq(Hz)');
xlim ([-fs / 2, fs / 2]);
```

شکل ۴: طیف فرکانسی سیگنال رمزنگاری شده



شکل ۵: خروجی متلب برای طیف فرکانسی سیگنال رمزنگاری شده

۴ جدا کردن بازه های فرکانسی سیگنال رمزنگاری با استفاده از فیلتر ها

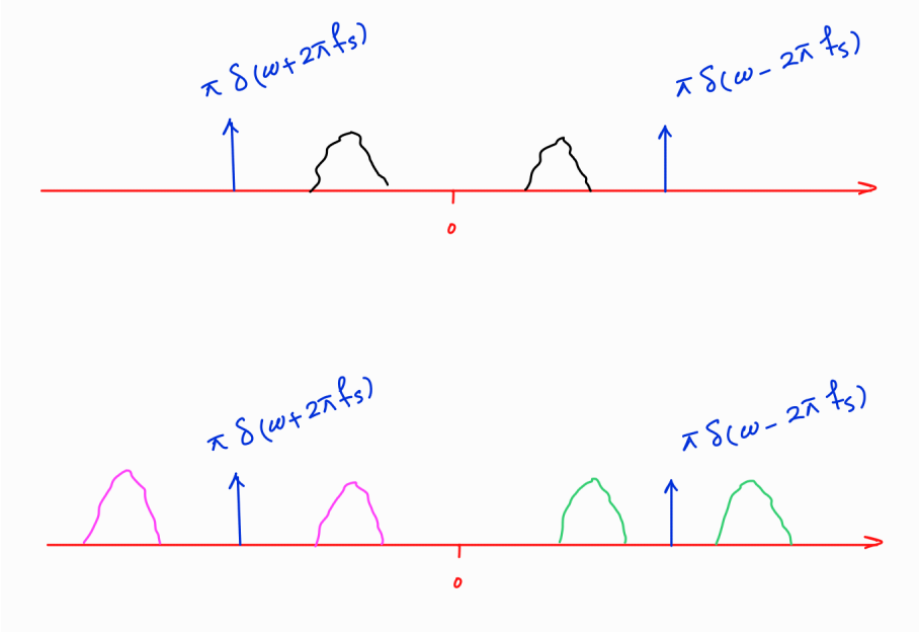
با اثر دادن فیلتر ها روی سیگنال ورودی می توان بازه های فرکانسی را از یکدیگر جدا کرد. همچنین برای استفاده از تابع کسینوس برای شیفت باید یک بازه برای زمان سیگنال کسینوسی بیابیم. از آنجایی که ۲۸ ثانیه کل تایم محتوای صوتی است و نیز پهنای باند فرکانسی f_s است، پس باید با استفاده از تابع $linspace$ در بازه ۰ تا ۲۸ ثانیه، 28×44100 داده تولید کنیم. تصویر کد فوق، به شرح زیر است

```
%% decomposing the input into 5 bands
%%%%%%%%%%%%%%%%%%%%%%%%%
y1 = filter(Band1, y);
y2 = filter(Band2, y);
y3 = filter(Band3, y);
y4 = filter(Band4, y);
y5 = filter(Band5, y);
t = linspace(0, 28, fs * 28);
```

شکل ۶: جدا کردن بازه های فرکانسی با استفاده از فیلتر های تعریف شده

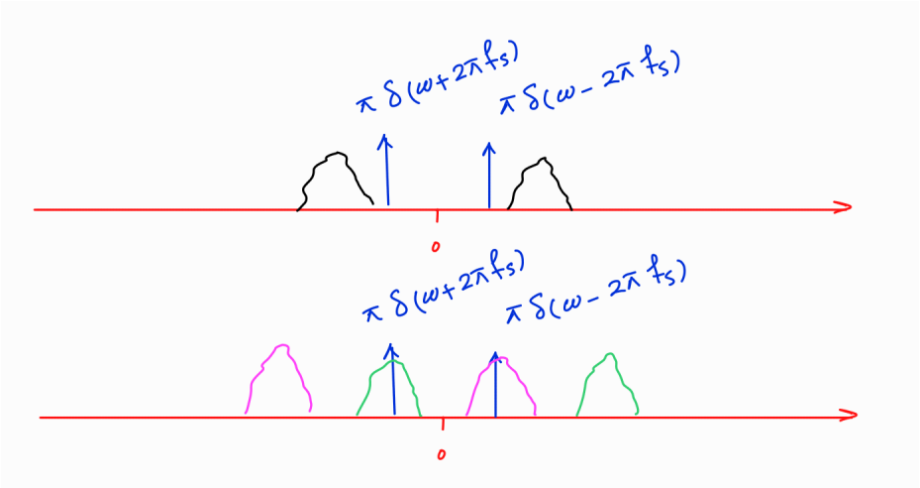
۵ جابجا کردن باند های فرکانسی

برای جابجا کردن بازه های فرکانسی از ایده ضرب کردن در کسینوس استفاده می کنیم. فرض کنیم می خواهیم یک باند با فرکانس کمتر را به فرکانس بالاتر انتقال دهیم. ایده از شکل زیر نشات می گیرد.



شکل ۷: انتقال از فرکانس های پایین به فرکانس ها بالا

طبق شکل فوق، دلتای سمت راست، پس از کانوالو شدن در تبدیل فوریه سیگنال، تپه های سبز و به همین ترتیب دلتای سمت چپ، تپه های صورتی رنگ تولید می شود. حال اگر یک فیلتر میانگذر با فرکانس بالاتر از فرکانس مرکزی تپه های سیاه بگذاریم، صورتی سمت چپ و سبز سمت راست از فیلتر عبور می کند.
به همین برای آوردن از فرکانس بالاتر به فرکانس پایین تر، مطابق شکل زیر عمل می کنیم.



شکل ۸: انتقال از فرکانس های بالا به فرکانس های پایین

در شکل فوق، اگر یک فیلتر با فرکانس قطع و عبور پایین تر فرکانس مرکزی تپه های سیاه باشد عملاً از فرکانس های بالاتر به فرکانس های پایین تر انتقال داریم. با این کار صورتی راست و سبز چپ انتخاب می شوند.
حال هر کدام از باند ها را با روش فوق جابجا می کنیم. تنها نکته این است که فرکانس را باید مثبت فرض کنیم، پس باید قدر مطلق اختلاف فرکانس های عبور را در 4410 ضرب کنیم که پهنای فرکانسی هر کدام از بازه ها برابر $\frac{f_s}{10} = 4410$ است. نحوه جابجایی هر بازه و نیز رسم طیف فرکانسی آن و نیز انتقال یافته آن در کد های زیر آمده است. تنها نکته اضافه در رسم نمودارها استفاده از $hold\ on$ برای رسم همزمان دو نمودار در یک پنجره است. همچنین استفاده از $legend$ برای لیبل زدن به هر کدام از نمودار ها برای تمایز دادن آنهاست. خروجی متلب نیز در ادامه آمده است

```
%% Band1
% calculate frequency of shifting
f14 = abs(4 - 1) * fs / 10;
% shift using multipling by cosine function
shift1 = cos(2 * pi * f14 * t);
x1 = 40 * y4 .* shift1';
x1 = filter(Band1,x1);

% filter result for deleting redundant bands generated by delta function
figure;
X1 = fftshift(fft(x1));
N = numel(x1);
freq = (-N / 2 : N / 2 - 1) / N * fs;
plot(freq, (abs(X1)) , 'LineWidth' , 1);
hold on
plot(freq, abs(fftshift(fft(y4))), 'LineWidth', 1);
grid on
title('absolute value of fourier transform of x1 and y4');
ylabel('AMP');
xlabel('Freq(Hz)');
xlim ([-fs / 2, fs / 2]);
legend('fourier transform of x1', 'fourier transform of y4');
```

(آ) باند اول سیگنال نهایی

```
%% Band2
% calculate frequency of shifting
f25 = abs(5 - 2) * fs / 10;
% shift using multipling by cosine function
shift2 = cos(2 * pi * f25 * t);
x2 = 2 * y5 .* shift2';
% filter result for deleting redundant bands generated by delta function
x2 = filter(Band2,x2);

% plot fourier transform of band and shifted band
figure;
X2 = fftshift(fft(x2));
N = numel(x1);
freq = (-N / 2 : N / 2 - 1) / N * fs;
plot(freq, (abs(X2)) , 'LineWidth' , 1);
hold on
plot(freq, abs(fftshift(fft(y5))), 'LineWidth', 1);
grid on
title('absolute value of fourier transform of x2 and y5');
ylabel('AMP');
xlabel('Freq(Hz)');
xlim ([-fs / 2, fs / 2]);
legend('fourier transform of x2', 'fourier transform of y5');
```

(ب) باند دوم سیگنال نهایی

```
%% Band3
% calculate frequency of shifting
f31 = abs(1 - 3) * fs / 10;
% shift using multipling by cosine function
shift3 = cos(2 * pi * f31 * t);
x3 = 2 * y1 .* shift3';
% filter the result for deleting redundant bands generated by delta function
x3 = filter(Band3,x3);

% plot fourier transform of band and shifted band
figure;
X3 = fftshift(fft(x3));
N = numel(x3);
freq = (-N / 2 : N / 2 - 1) / N * fs;
plot(freq, (abs(X3)) , 'LineWidth' , 1);
hold on
plot(freq, abs(fftshift(fft(y1))), 'LineWidth', 1);
grid on
title('absolute value of fourier transform of x3 and y1');
ylabel('AMP');
xlabel('Freq(Hz)');
xlim ([-fs / 2, fs / 2]);
legend('fourier transform of x3', 'fourier transform of y1');
```

(ج) باند سوم سیگنال نهایی

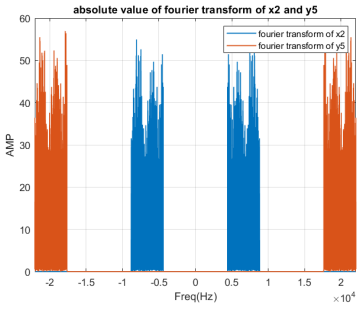
```
%% Band4
% calculate frequency of shifting
f43 = abs(2 - 4) * fs / 10;
% shift using multipling by cosine function
shift4 = cos(2 * pi * f43 * t);
x4 = 2 * y3 .* shift4';
% filter the result for deleting redundant bands generated by delta function
x4 = filter(Band4,x4);

% plot fourier transform of band and shifted band
figure;
X4 = fftshift(fft(x4));
N = numel(x4);
freq = (-N / 2 : N / 2 - 1) / N * fs;
plot(freq, (abs(X4)) , 'LineWidth' , 1);
hold on
plot(freq, abs(fftshift(fft(y3))), 'LineWidth', 1);
grid on
title('absolute value of fourier transform of x1 and y4');
ylabel('AMP');
xlabel('Freq(Hz)');
xlim ([-fs / 2, fs / 2]);
legend('fourier transform of x4', 'fourier transform of y3');
```

(د) باند چهارم سیگنال نهایی

```
%% Band5
% calculate frequency of shifting
f52 = abs(2 - 5) * fs / 10;
% shift using multipling by cosine function
shift5 = cos(2 * pi * f52 * t);
x5 = 2 * y2 .* shift5';
% filter the result for deleting redundant bands generated by delta function
x5 = filter(Band5,x5);

% plot fourier transform of band and shifted band
figure;
X5 = fftshift(fft(x5));
N = numel(x5);
freq = (-N / 2 : N / 2 - 1) / N * fs;
plot(freq, (abs(X5)) , 'LineWidth' , 1);
hold on
plot(freq, abs(fftshift(fft(y2))), 'LineWidth', 1);
grid on
title('absolute value of fourier transform of x5 and y2');
ylabel('AMP');
xlabel('Freq(Hz)');
xlim ([-fs / 2, fs / 2]);
legend('fourier transform of x5', 'fourier transform of y2');
```

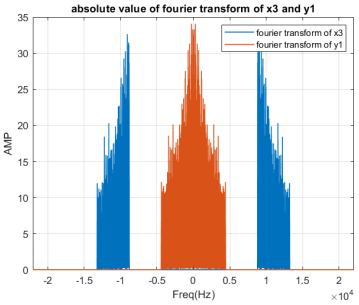
(ه) باند پنجم سیگنال نهایی

شکل ۹: جابجایی بازه های فرکانسی

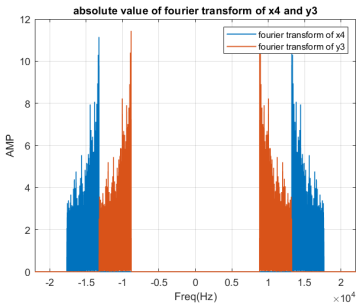


(آ) باند اول سیگنال نهایی

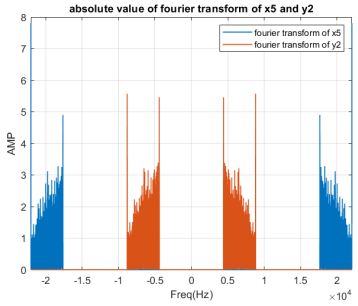
(ب) باند دوم سیگنال نهایی



(ج) باند سوم سیگنال نهایی



(د) باند چهارم سیگنال نهایی



(ه) باند پنجم سیگنال نهایی

شکل ۱۰: طیف بازه های فرکانسی، قبل و بعد از جابجایی

یک نکته ای که باید رعایت کنیم این است که پس از ضرب کردن در کسینوس ها می بایست سیگنال ها را با ضریب ۲ تقویت کنیم، زیرا

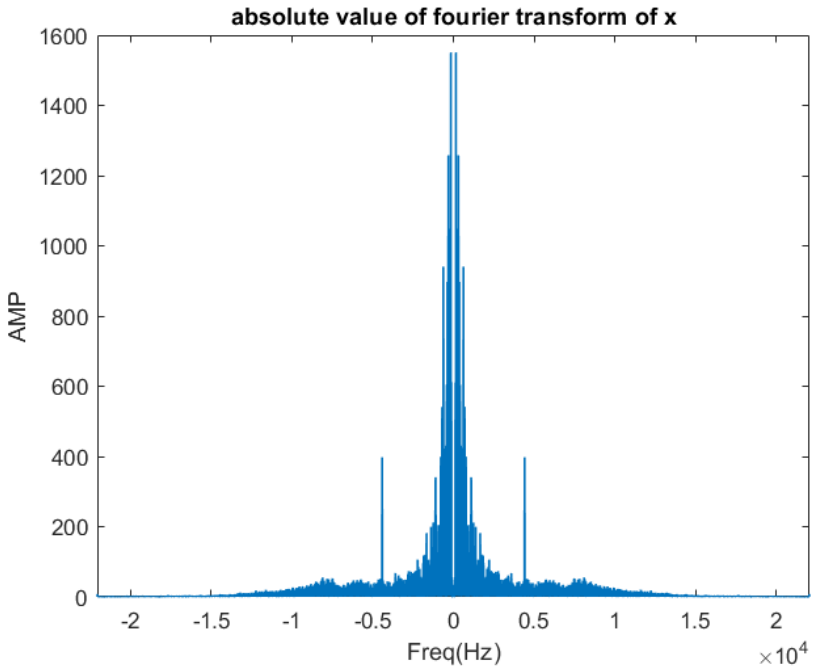
$$\mathcal{F}\{f(t) \times cos(2\pi f_{shift}t)\} = \frac{1}{2\pi}\mathcal{F}\{f(t)\} * (\pi\delta(\omega - 2\pi f_{shift}) + \pi\delta(\omega + 2\pi f_{shift}))$$
$$= \frac{1}{2}(F(\omega - 2\pi f_{shift}) + F(\omega + 2\pi f_{shift}))$$

۶ چک کردن نتیجه خروجی

ابتدا همه باند ها را با یکدیگر جمع می زنیم و طیف فرکانسی سیگنال نهایی را رسم می کنیم. کد و خروجی متلب در ادامه آمده است

```
x = x1 + x2 + x3 + x4 + x5;

% plot frequency spectrum of output signal
figure;
X = fftshift(fft(x));
N = numel(x);
freq = (-N / 2 : N / 2 - 1) / N * fs;
plot(freq, (abs(X)) , 'LineWidth' , 1);
title('absolute value of fourier transform of x');
ylabel('AMP');
xlabel('Freq(Hz)');
xlim ([-fs / 2, fs / 2]);


player = audioplayer(x , fs);
play(player)
```

شکل ۱۱: چک کردن سیگنال نهایی و طیف آن



شکل ۱۲: طیف فرکانسی سیگنال نهایی پس از کدگشایی

۷ ذخیره سازی فایل خروجی

برای ذخیره سازی فایل خروجی از تابع *audiowrite* استفاده کردم که آرگومان های اول تا سوم آن به ترتیب عبارتند از نام فایل خروجی، سیگنال خروجی و نیز پهنای باند فرکانسی سیگنال. در شکل زیر کد آن آمده است

```
audiowrite('output.wav' , x , fs)
```

شکل ۱۳: ذخیره سازی خروجی

۸ استفاده از تبدیل فوریه

در این بخش ابتدا ۵ سیگنال خالی با طول آرایه صوت تعریف می کنیم. حال به هر آرایه با توجه به بازه فرکانسی جابجا شده از ماتریس A ، مقادیر مناسب

از تبدیل فوریه شیفت یافته سیگنال ورودی را نسبت می دهیم و در نهایت آنها را با هم جمع می کنیم. حال اندازه حاصل را می کشیم و میبینیم کاملاً مشابه چیزی شده که قبل تر با استفاده از کسینوس ها بدست آوردیم. حال یکبار سیگنال را شیفت داده، سپس تبدیل فوریه معکوس گرفته و بلافاصله آن را شیفت می دهیم. این کار منطقی است زیرا خروجی fft شیفت ندارد و در نتیجه ورودی $ifft$ نیز باید بدون شیفت باشد. پس از پخش خواهیم دید که کاملاً نتایج سازگار با روش قبلی است. در نهایت هم فایل را ذخیره می کنیم. کد ها و نتایج در زیر آمده است

```
%% Using fft and ifft for descrambling
% initializing final bands fourier transform
X1 = zeros(length(y), 1);
X2 = zeros(length(y), 1);
X3 = zeros(length(y), 1);
X4 = zeros(length(y), 1);
X5 = zeros(length(y), 1);

% assigning proper values to X_is
len = length(y);

Y = fftshift(fft(y));
```

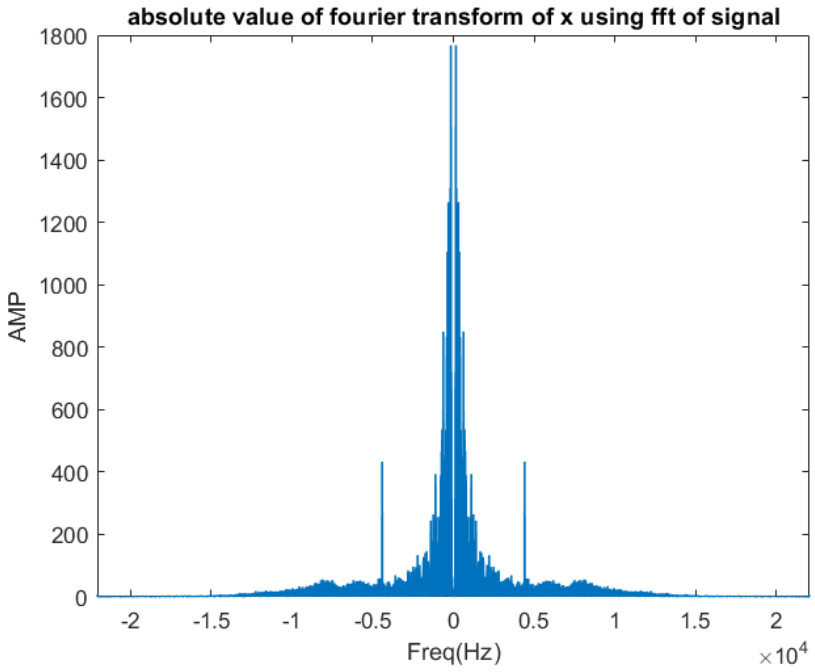
شکل ۱۴: ساختن آرایه ها

```
% first band
X1(4 * len / 10 + 1 : 5 * len / 10, 1) = 20 * Y(1 * len / 10 + 1 : 2 * len / 10, 1);
X1(5 * len / 10 + 1 : 6 * len / 10, 1) = 20 * Y(8 * len / 10 + 1 : 9 * len / 10, 1);

% second band
X2(3 * len / 10 + 1 : 4 * len / 10, 1) = Y(0 * len / 10 + 1 : 1 * len / 10, 1);
X2(6 * len / 10 + 1 : 7 * len / 10, 1) = Y(9 * len / 10 + 1 : 10 * len / 10, 1);

% third band
X3(2 * len / 10 + 1 : 3 * len / 10, 1) = Y(4 * len / 10 + 1 : 5 * len / 10, 1);
X3(7 * len / 10 + 1 : 8 * len / 10, 1) = Y(5 * len / 10 + 1 : 6 * len / 10, 1);

% fourth band
X4(1 * len / 10 + 1 : 2 * len / 10, 1) = Y(2 * len / 10 + 1 : 3 * len / 10, 1);
X4(8 * len / 10 + 1 : 9 * len / 10, 1) = Y(7 * len / 10 + 1 : 8 * len / 10, 1);

% fifth band
X5(0 * len / 10 + 1 : 1 * len / 10, 1) = Y(3 * len / 10 + 1 : 4 * len / 10, 1);
X5(9 * len / 10 + 1 : 10 * len / 10, 1) = Y(6 * len / 10 + 1 : 7 * len / 10, 1);
```

شکل ۱۵: مقدار دهی به آرایه ها

```
% plot absolute value of fourier transform of signal
X = X1 + X2 + X3 + X4 + X5;
figure;
N = numel(X);
freq = (-N / 2 : N / 2 - 1) / N * fs;
plot(freq, abs((X)) , 'LineWidth' , 1);
title('absolute value of fourier transform of x using fft of signal');
ylabel('AMP');
xlabel('Freq(Hz)');
xlim ([-fs / 2, fs / 2]);
```

شکل ۱۶: رسم اندازه تبدیل فوریه خروجی



شکل ۱۷: طیف سیگنال خروجی

```
% fft inverse and correction on symmetry of signal
x = ifft(fftshift(X));
player = audioplayer(real(x) , fs);
play(player)

%save the output
audiowrite('decodingusingfft.wav' , real(x), fs);
```

شکل ۱۸: تبدیل فوریه معکوس و ذخیره سازی

۹ ضبط صدا با استفاده از متلب

ابتدا یک آبجکت برای رکورد صوت می سازیم که خروجی تابع $audiorecord$ است. آرگومان اول تا سوم به ترتیب عبارتند از پهنای فرکانسی، حافظه ذخیره

سازی و تعداد کانال هاست(یک می گذاریم). سپس مدت زمان رکورد را تعیین می کنیم. با استفاده از تابع $recordblocking$ صوت را ضبط کرده و با استفاده از تابع $getaudiodata$ آن را به سیگنال صوتی تبدیل کرده و با تابع $audiowrite$ آن را ذخیره می کنیم. پس از ذخیره سازی کد را کامنت می کنیم تا دوباره ران نشود.

```
%% Record my audio and save it
% recObj = audiorecorder(44100, 16, 1);
% recDuration = 30;
% recordblocking(recObj,recDuration);
% myrec= getaudiodata(recObj);
% audiowrite('myVoiceOrg.wav', myrec, 44100);
```

شکل ۱۹: نحوه ضبط صدا در متلب

۱۰ پخش صدای ضبط شده

الگوریتم قبلاً توضیح داده شده است. اینجا فقط کد را می گذاریم

```
%% Play my recorded voice
[g , fs] = audioread('myVoiceOrg.wav');

player = audioplayer(g , fs);
disp(size(g))
%play the sound
play(player)
```

شکل ۲۰: پخش صدایی که با متلب ضبط شد

۱۱ نحوه رمزگذاری سیگنال با استفاده از الگوریتم صورت مساله(امتیازی)

کافیست دقیقا همان کاری که برای کدگشایی سیگنال در مسئله اصلی کردیم را انجام دهیم با این تفاوت که باز هم باید سیگنال ها را در ۲ ضرب کنیم، زیرا باز هم جابجایی با استفاده از ضرب در کسینوس بوده است. کد آن به صورت زیر است

```
%% Encoding with the algorithm that defines in problem
% seperating each band of original voice
g1 = filter(Band1, g);
g2 = filter(Band2, g);
g3 = filter(Band3, g);
g4 = filter(Band4, g);
g5 = filter(Band5, g);
t = linspace(0, 30, fs * 30);
```

(آ) فیلتر کردن و جدا کردن بازه های فرکانسی

```
% encoding the voice
shift1 = cos(2 * pi * f31 * t);
encoded1 = 2 * g3 .* shift1';
encoded1 = filter(Band1, encoded1);

shift2 = cos(2 * pi * f52 * t);
encoded2 = 2 * g5 .* shift2';
encoded2 = filter(Band2, encoded2);

shift3 = cos(2 * pi * f43 * t);
encoded3 = 2 * g4 .* shift3';
encoded3 = filter(Band3, encoded3);

shift4 = cos(2 * pi * f14 * t);
encoded4 = 2 * (1 / 20) * g1 .* shift4';
encoded4 = filter(Band4, encoded4);

shift5 = cos(2 * pi * f25 * t);
encoded5 = 2 * g2 .* shift5';
encoded5 = filter(Band5, encoded5);

encoded = encoded1 + encoded2 + encoded3 + encoded4 + encoded5;
```

(ب) جابجا کردن بازه ها بر اساس ماتریس A

```
% saving as encoded voice
audiowrite('encodedVoice.wav' , encoded , fs);
```

(ج) ذخیره سازی فایل کدگذاری شده

شکل ۲۱: کدگذاری فایل ضبط شده با الگوریتم ماتریسA

۱۲ کدگشایی(امتیازی)

این بخش کاملاً شبیه کاری ست که در صورت مساله اصلی کردیم. پس فقط کد ها را می آوریم

```
%% decoding like previous part
[a , fs] = audioread('encodedVoice.wav');

player = audioplayer(a , fs);

%play my encoded voice
play(player)
```

(آ) پخش کردن فایل کدگذاری شده

```
a1 = filter(Band1, a);
a2 = filter(Band2, a);
a3 = filter(Band3, a);
a4 = filter(Band4, a);
a5 = filter(Band5, a);
t = linspace(0, 30, fs * 30);
```

(ب) فیلتر کردن باند های ورودی برای جابجایی

```
% Band1
% calculate frequency of shifting
f14 = abs(4 - 1) * fs / 10;
% shift using multipling by cosine function
shift1 = cos(2 * pi * f14 * t);
b1 = 40 * a4 .* shift1';
b1 = filter(Band1,b1);
```

(ج) جابجایی باند اول

```
% Band2
% calculate frequency of shifting
f25 = abs(5 - 2) * fs / 10;
% shift using multipling by cosine function
shift2 = cos(2 * pi * f25 * t);
b2 = 2 * a5 .* shift2';
% filter result for deleting redundant bands generated by delta function
b2 = filter(Band2,b2);
```

(د) جابجایی باند دوم

```
% Band3
% calculate frequency of shifting
f31 = abs(1 - 3) * fs / 10;
% shift using multipling by cosine function
shift3 = cos(2 * pi * f31 * t);
b3 = 2 * a1 .* shift3';
% filter the result for deleting redundant bands generated by delta function
b3 = filter(Band3,b3);
```

(ه) جابجایی باند سوم

```
% Band4
% calculate frequency of shifting
f43 = abs(2 - 4) * fs / 10;
% shift using multipling by cosine function
shift4 = cos(2 * pi * f43 * t);
b4 = 2 * a3 .* shift4';
% filter the result for deleting redundant bands generated by delta function
b4 = filter(Band4,b4);
```

(و) جابجایی باند چهارم

```
% Band5
% calculate frequency of shifting
f52 = abs(2 - 5) * fs / 10;
% shift using multipling by cosine function
shift5 = cos(2 * pi * f52 * t);
b5 = 2 * a2 .* shift5';
% filter the result for deleting redundant bands generated by delta function
b5 = filter(Band5,b5);
```

(ز) جابجایی باند پنجم

```
b = b1 + b2 + b3 + b4 + b5;


player = audioplayer(b , fs);
play(player)
```

(ح) چک کردن خروجی و پخش کردن آن

```
audiowrite('myDecodedVoice.wav' , b , fs);
```

(ط) ذخیره سازی خروجی نهایی

شکل ۲۲: کدگشایی با استفاده از الگوریتم صورت مساله