به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

آزمایشگاه پردازش بی درنگ سیگنال های دیجیتال

تمرین شماره ۵

نام و نام خانوادگی : سهند خوشدل

شماره دانشجویی : ۸۱۰۱۹۶۶۰۷

پاییز ۹۹

بخش اول: ۶-۴-۳-۱: پیاده سازی spectrum analyzer (تخمین گر طیف) در c

مرحله ۲و۳) تعریف آرایه **spectrum** با طول **N=513** از نوع **float** و دو آرایه **ping, pong** از نوع **complex** - تشکیل تابع **ISR**:

```
#include <stdio.h>
#include <math.h>
#include "fft.c"
#include "Complex.c"


# define N 1024


// function declarations:
void ISR(float* a, float* b, complex* ping, complex* pong);


float spectrum[N/2+1] = {0};

complex ping[N];
complex ping[N];


void ISR (complex* ping, complex* pong, float* a, float* b, ) {
    for (int i = 0; i<N; i++) {

        // the real part of each of ping and pong signals is going to be the real signal a
        ping[i].x = a[i];
        // the imaginary part of each of ping and pong signals is going to be the real signal b
        ping[i].y = b[i];

        pong[i].x = a[i];
        pong[i].y = b[i];
    }
}

```

مرحله ۴)

a) گرفتن تبدیل فوریه ۱۰۲۴ نقطه ای از آرایه **ping/pong** به کمک توابع داده شده **(fft)**

```
15
16  complex ping[N];
17  complex ping[N];
18
19
20  fft(ping, 1024, 1);
21  fft(pong, 1024, 1);
22
```

b) اضافه کردن مجموع مربعات ضرایب فوریه به آرایه spectrum:

به کمک تابع ccabs که در کد complex.c آمده ست، اندازه یک تابع complex در یک نقطه بخصوص برگردانده می شود.:

```
19
20  fft(ping, 1024, 1);
21  fft(pong, 1024, 1);
22
23  for (int i = 0; i < N / 2 + 1; i++) {
24      spectrum[i] += (ccabs(pong[i])*ccabs(pong[i]) + ccabs(pong[N - 1 - i])*ccabs(pong[N - 1 - i])) / 2;
25  }
26
```

c) اثبات رابطه ضرایب DFT:

d) نرمالایز کردن آرایه spectrum برای حفظ توان کل: (تقسیم بر N*8) :

```
for (int i = 0; i < N / 2 + 1; i++) {
    spectrum[i] += (ccabs(pong[i])*ccabs(pong[i]) + ccabs(pong[N - 1 - i])*ccabs(pong[N - 1 - i])) / 2;
    spectrum[i] += spectrum[i] / (L * N);
}
```

f) پاک کردن آرایه spectrum و تکمیل و اصلاح ساختار spectrum analyzer:

تمام کار هایی که تا کنون کردیم باید برای k=0,…,512 انجام شود. بنابراین یک لوپ اصلی برای عملیات فوق قرار می دهیم. هر بار اجرای این لوپ به منظور گرفتن fft در یک بازه ۱۰۲۴ نقطه ای و تخمین طیف با متوسط گیری از ۱۰۲۴ FFT نقطه ای است. بدیهی هر چه تعداد نقاط در بازه fft گرفتن بیشتر باشدو دقت تخمین طیف بالاتر خواهد بود.

همچنین نرمالایز کردن بر L تنها هر L فریم انجام شود. اما در هر فریم نرمالیزیشن با فاکتور N باید صورت بگیرد بنابراین:

برای آنکه بتوانیم بافر spectrum را در انتهای هر L فریم خالی کنیم (آرایه را پاک کنیم)، باید تمام عملیات انجام شده را روی یک آرایه buffer (به صورت موقتی) انجام دهیم.آن را داخل آرایه اصلی ریخته و خالی کرده و و تخمین های اصلی را در آرایه اصلی نگه داریم تا بعدا استفاده کنیم.

از آنجایی که خروجی خط مجاسبه ضرایب به صورت double است، نوع آرایه های buffer , spectrum را به double تغییر می دهیم.

در نهایت کد به صورت زیر در می آید:

```
#include <stdio.h>
#include <math.h>
#include "fft.c"
#include "Complex.c"


# define N 1024
# define L 8

// function declarations:
void ISR(float* a, float* b, complex* ping, complex* pong);

len_signal  = 80000

double spectrum[N/2+1] = {0};
double buffer[N / 2 + 1] = { 0 };

complex ping[N];
complex ping[N];


fft(ping, 1024, 1);
fft(pong, 1024, 1);

max_iteration = floor(len_signal/(2*N));

for (int j = 0; j<max_iteration; j++) {

    for (int i = 0; i < N / 2 + 1; i++) {
        buffer[i] += (ccabs(pong[i])*ccabs(pong[i]) + ccabs(pong[N -1-i])*ccabs(pong[N-1-i])) / 2;
        buffer[i] = buffer[i]

        if ((j - 1) % L == 0){
            spectrum[i] += buffer[i] / (L* N);
            buffer[i] = 0;
        }

    }
```

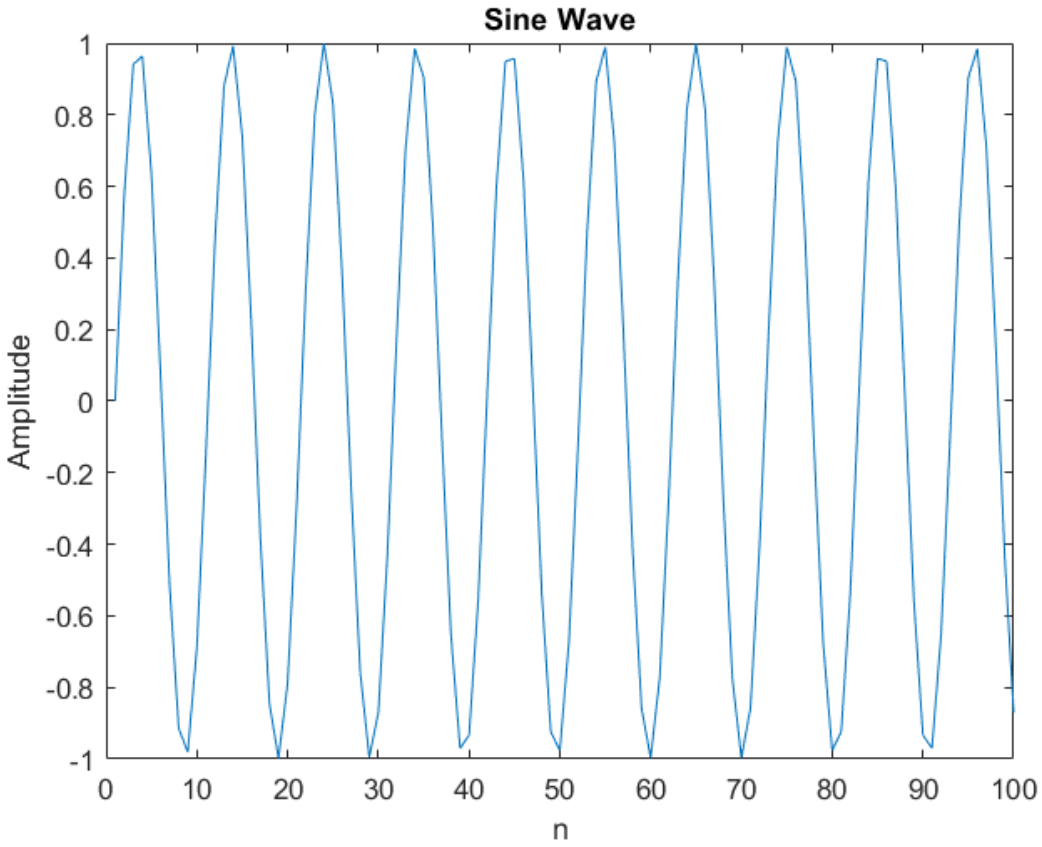
بخش دوم: ۶-۴-۳-۲: تست تخمین گر طیف به کمک سیگنال های ورودی تست

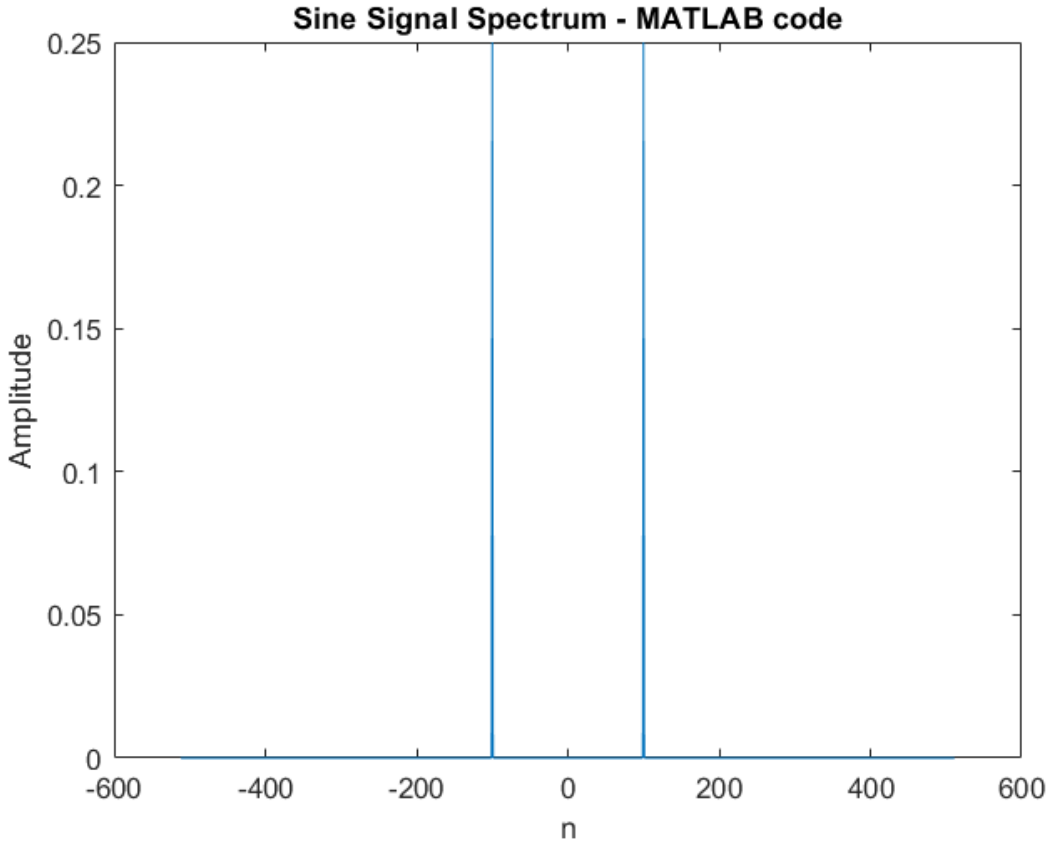
ساخت تابع مولد cosine در c:

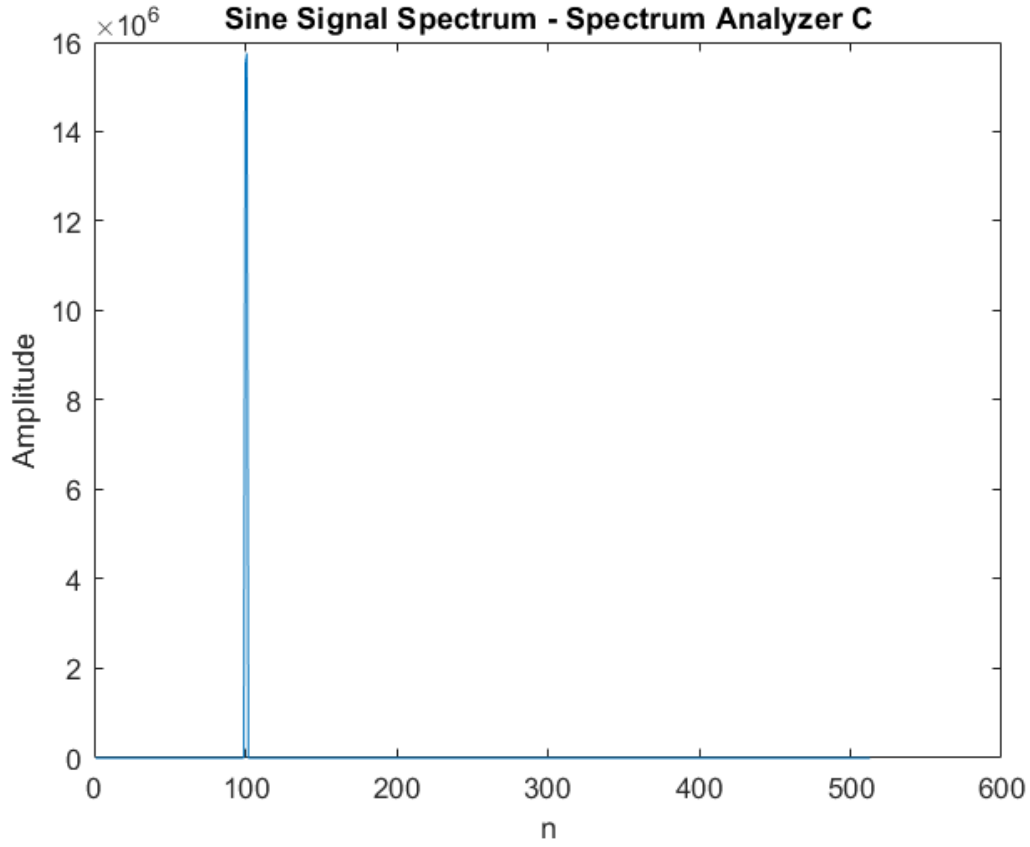
```
82  void cos(float* s) {
83      for (int i = 0; i < LENGTH; i++)
84          s[i] = 10000 * cos(i * 100 * 2 * PI / 1024);
85  }
```

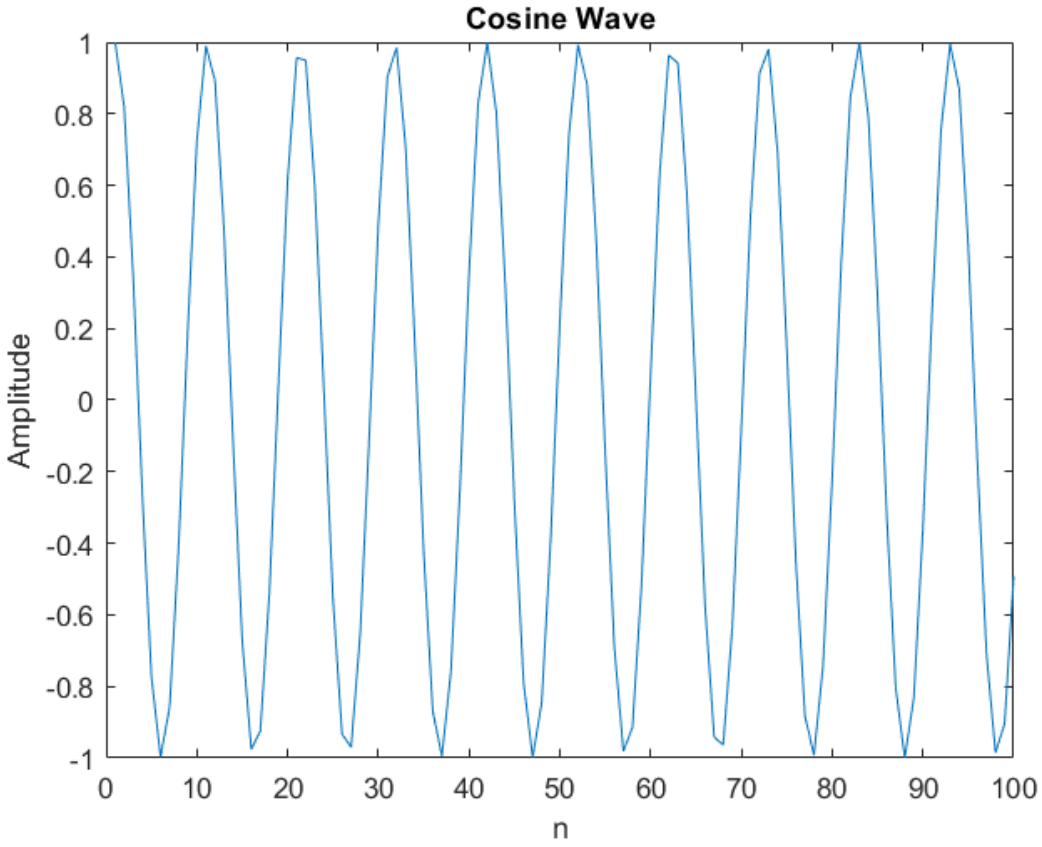
خواندن خروجی fft تولید شده در c با مد متلب و رسم طیف:

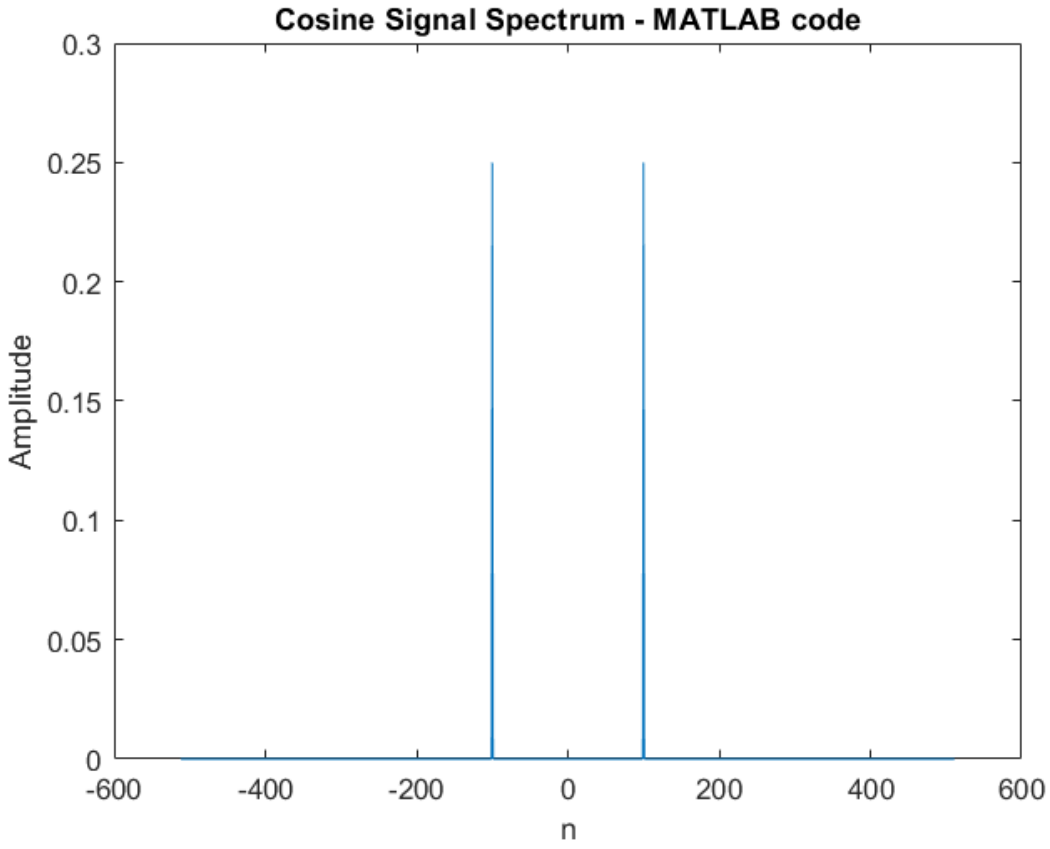
تولید سیگنال های تست در متلب:

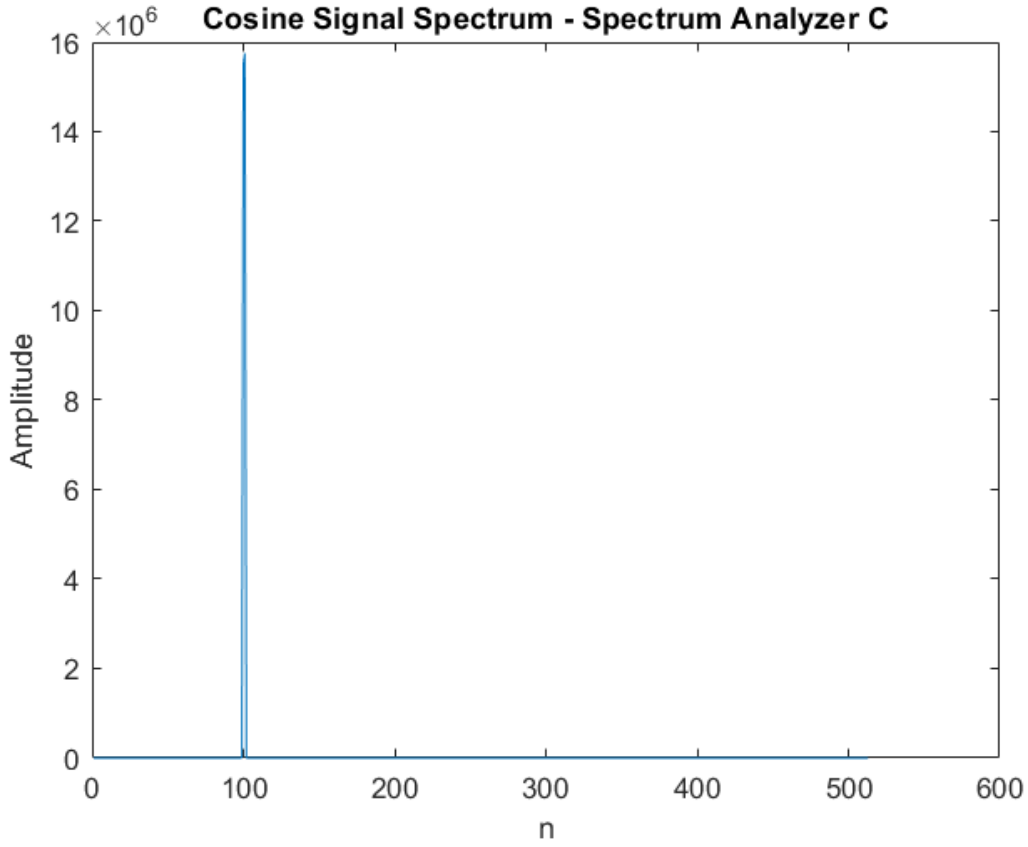


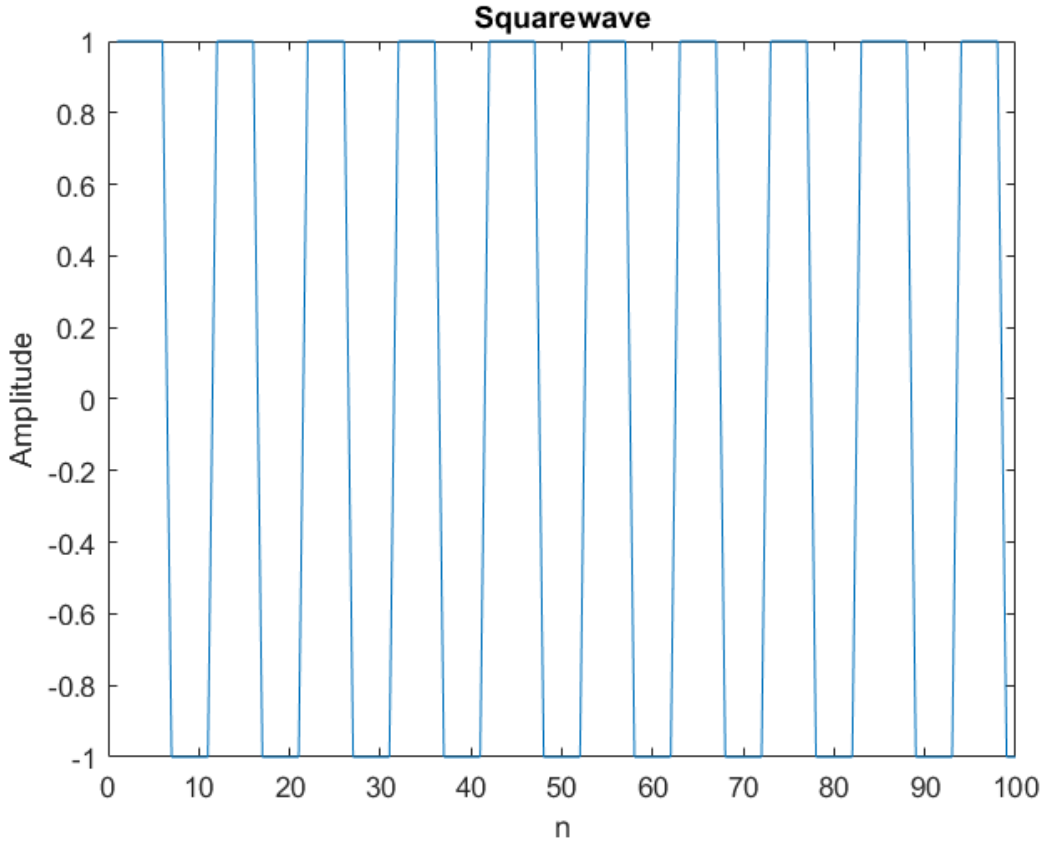

Sine Signal Spectrum - MATLAB code
0.25
0.2
0.15
0.1
0.05
0
Amplitude
-600
-400
-200
0
200
400
600
n

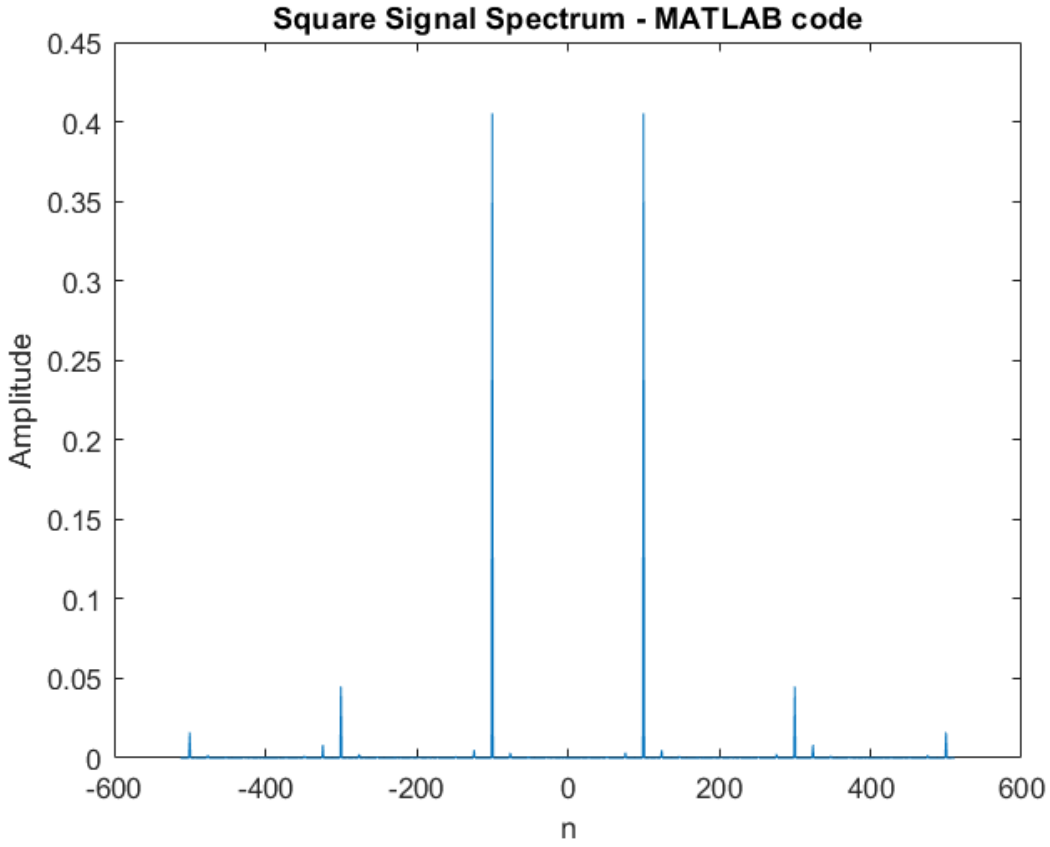

Sine Signal Spectrum - Spectrum Analyzer C
×10^6
16
14
12
10
8
6
4
2
0
Amplitude
0
100
200
300
400
500
600
n

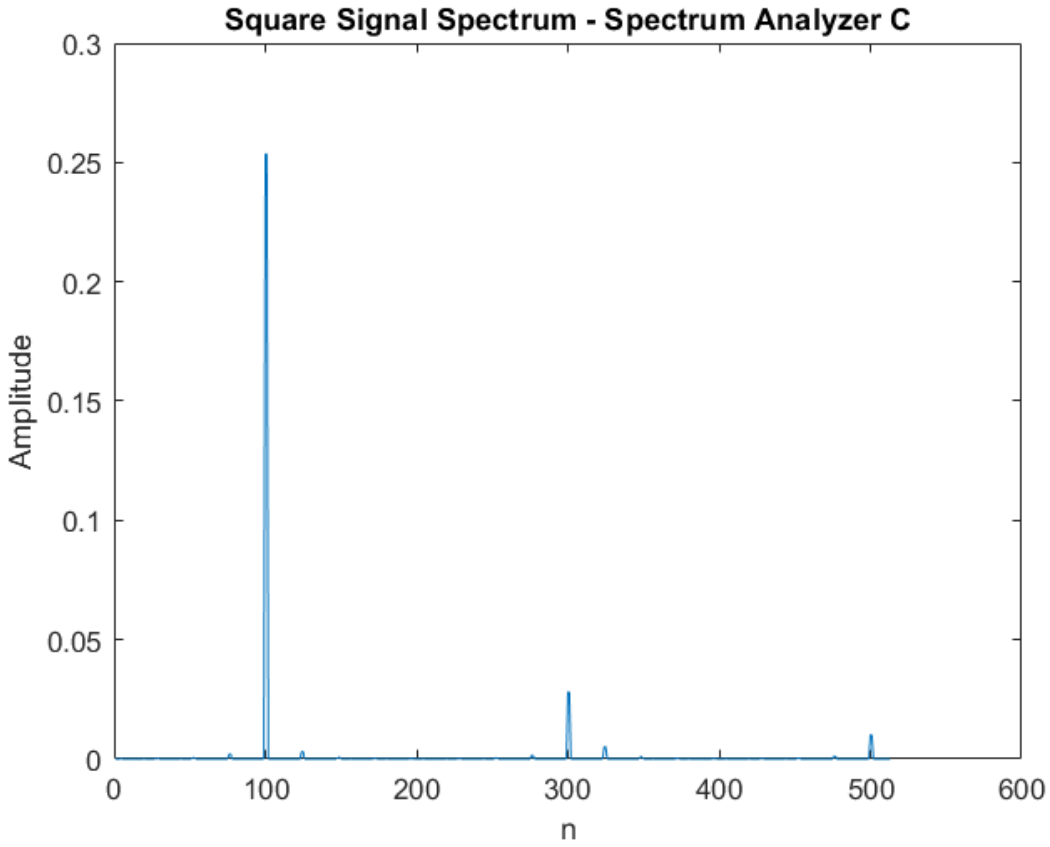

Cosine Wave
Amplitude
n
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
10
20
30
40
50
60
70
80
90
100

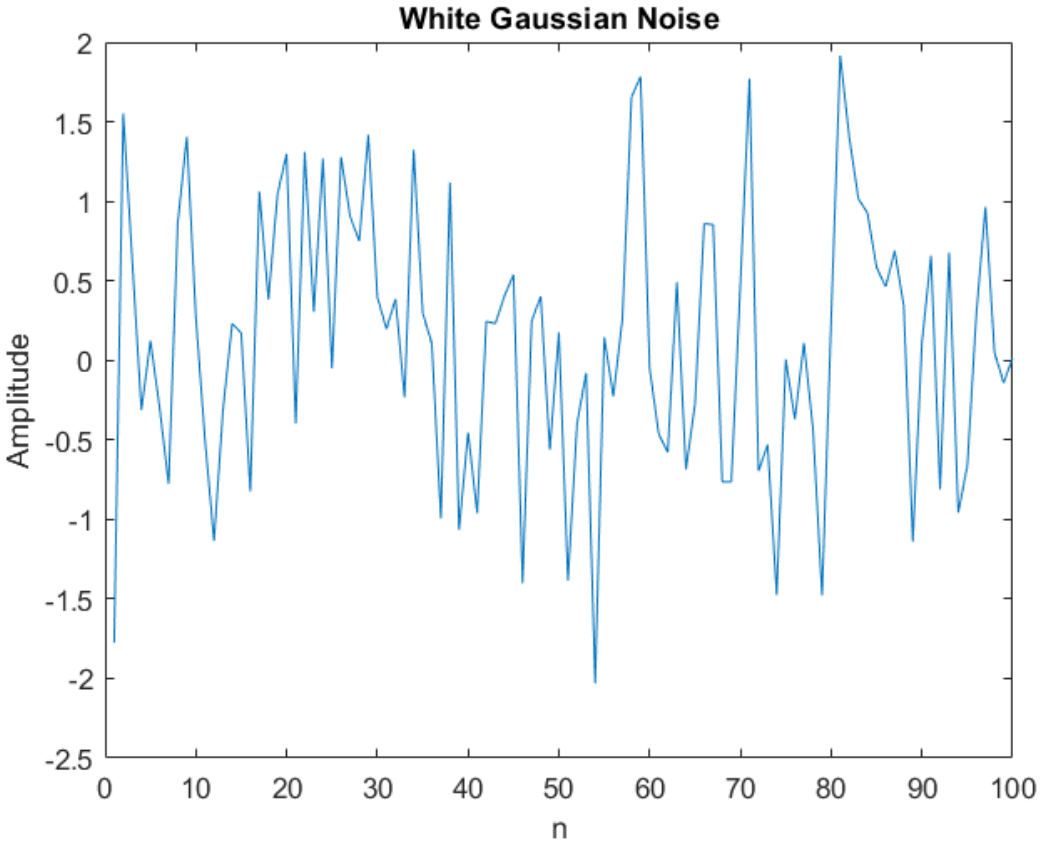

Cosine Signal Spectrum - MATLAB code
Amplitude
n
0.3
0.25
0.2
0.15
0.1
0.05
0
-600
-400
-200
0
200
400
600

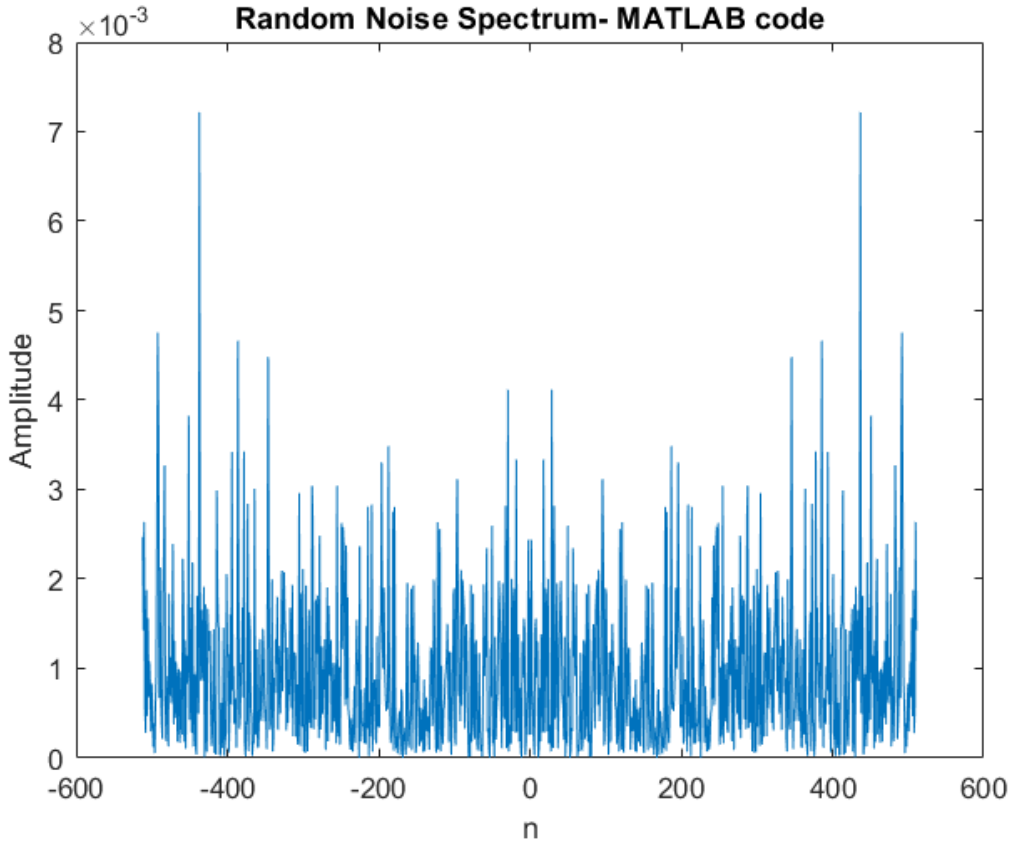

Cosine Signal Spectrum - Spectrum Analyzer C
×10^6
16
14
12
10
8
6
4
2
0
Amplitude
0
100
200
300
400
500
600
n

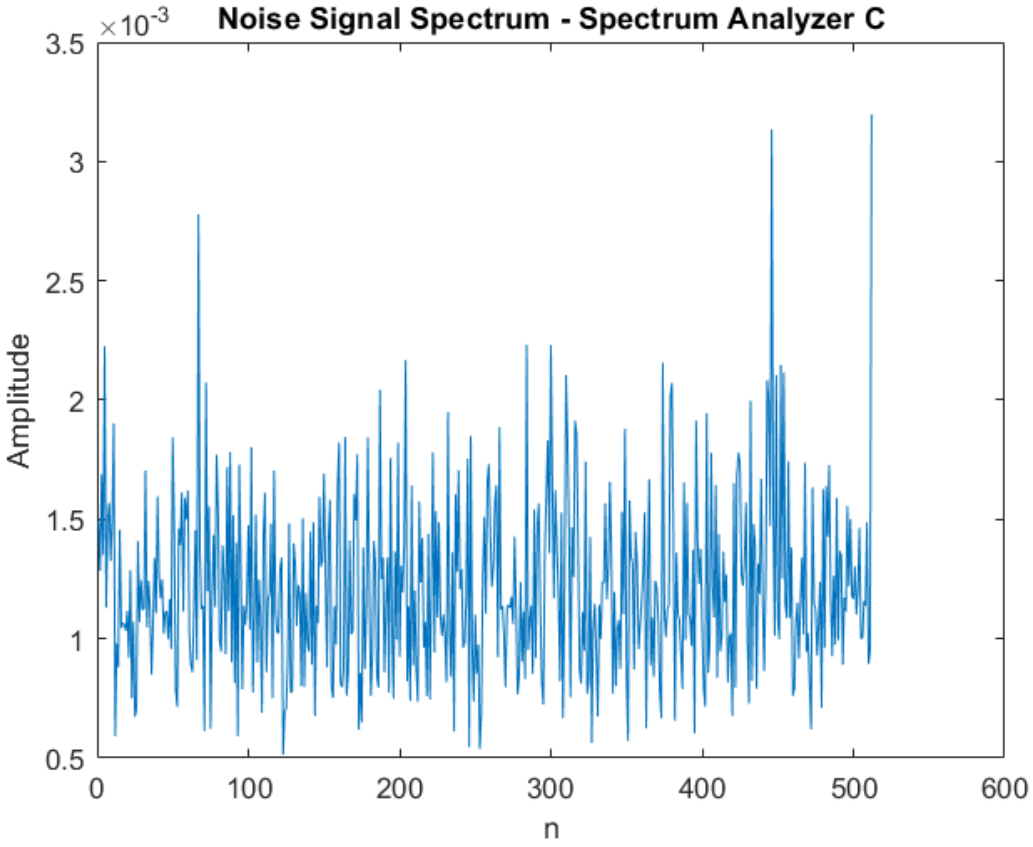

Squarewave
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
Amplitude
0
10
20
30
40
50
60
70
80
90
100
n


Square Signal Spectrum - MATLAB code
Amplitude
n
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-600
-400
-200
0
200
400
600


Square Signal Spectrum - Spectrum Analyzer C
Amplitude
n
0.3
0.25
0.2
0.15
0.1
0.05
0
0
100
200
300
400
500
600


White Gaussian Noise
Amplitude
n
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
10
20
30
40
50
60
70
80
90
100


Random Noise Spectrum- MATLAB code
×10^-3
Amplitude
n
8
7
6
5
4
3
2
1
0
-600
-400
-200
0
200
400
600


Noise Signal Spectrum - Spectrum Analyzer C
×10^-3
3.5
3
2.5
2
1.5
1
0.5
Amplitude
0
100
200
300
400
500
600
n