به نام خدا

گزارش تکلیف عملی درس سیگنال

رسول کامکار

۹۸۲۶۶۵۳

فهرست

بخش اول

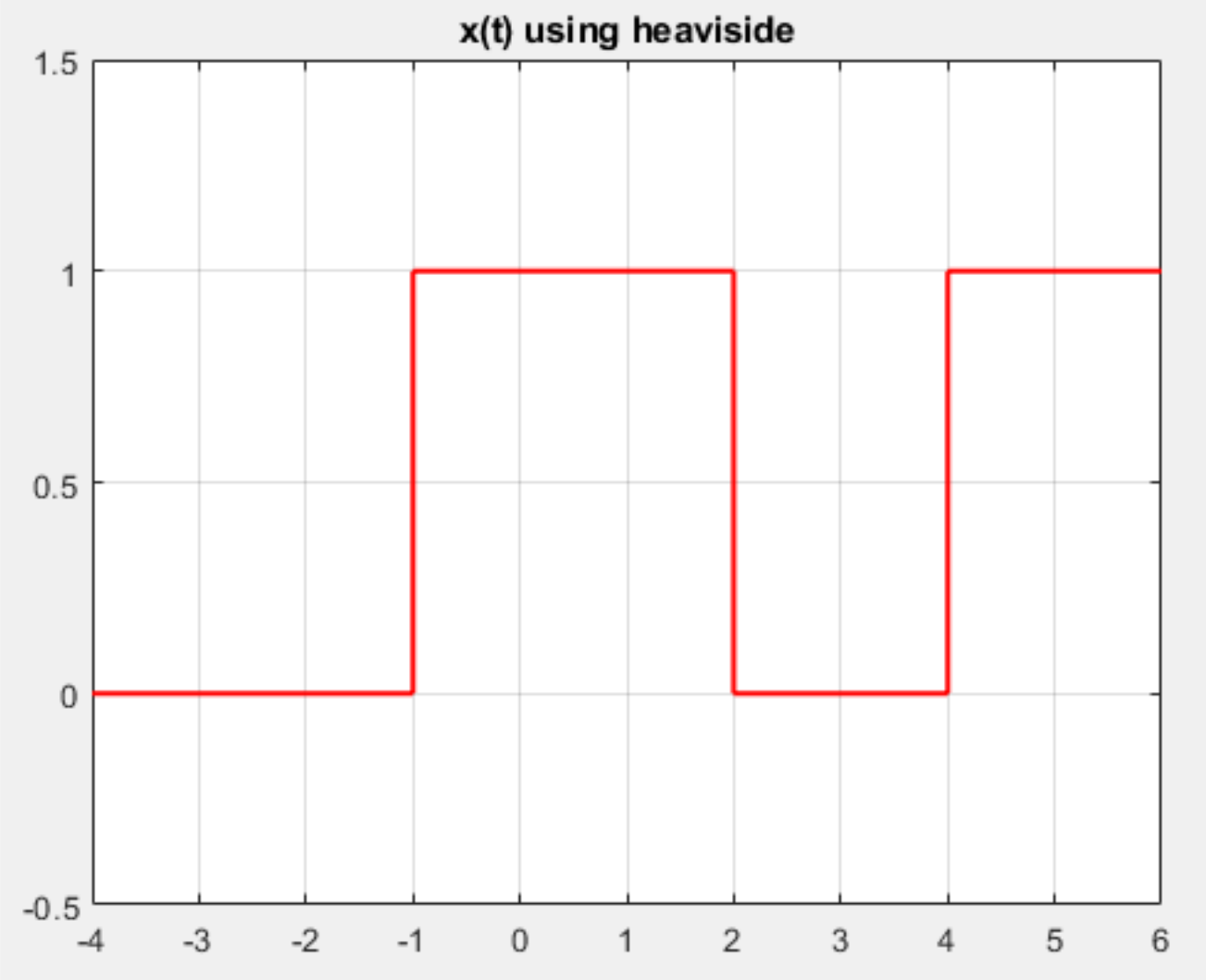
کد این بخش در فایل Q1.m قرار دارد.

سکشن یک

برای ساخت تابع مورد نظر، از تابع Heaviside استفاده شده است. به این صورت که مانند رابطه سیگنال داخل سوال، ۳ تابع Heaviside با شیفت‌های +۱ -۲ -۴ با هم ترکیب شده‌اند.

تابع ساخته شده در فایل x_t_with_heaviside.m قرار دارد.

خروجی این سکشن:



سکشن دو

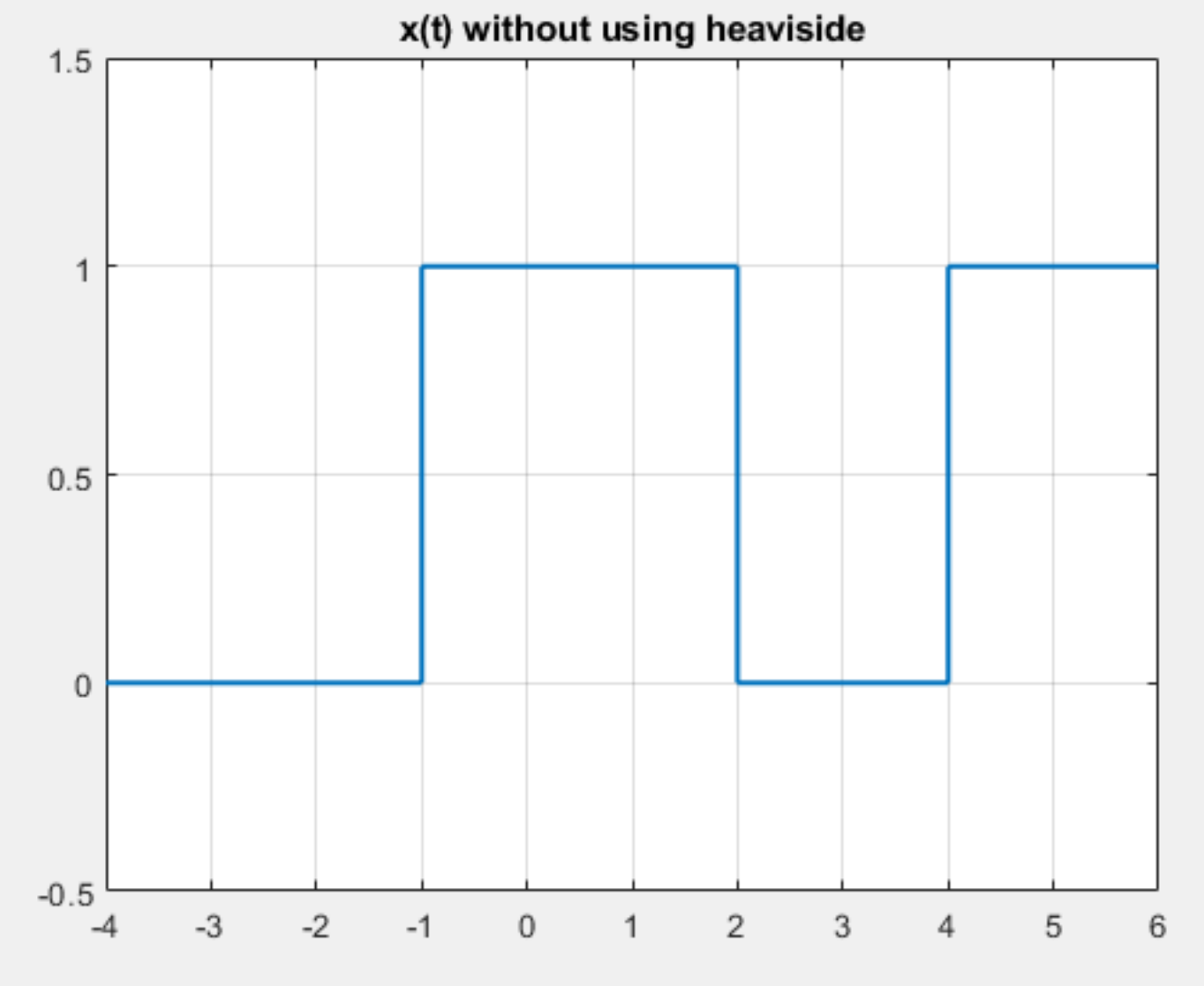
برای ساخت تابع بدون Heaviside، از ایندکس دهی شرطی استفاده شده است

به این صورت که همه مقادیر خروجی برابر ۱ قرار گرفته، سپس مقادیر محور زمان خارج از بازه‌های مورد نظر، مقدار ۰ گرفته‌اند (t محور زمان است)

```
y = ones(size(t));
y((t<-1) | (t>2 & t<4)) = 0;
```

تابع ساخته شده در فایل x_t_without_heaviside.m قرار دارد.

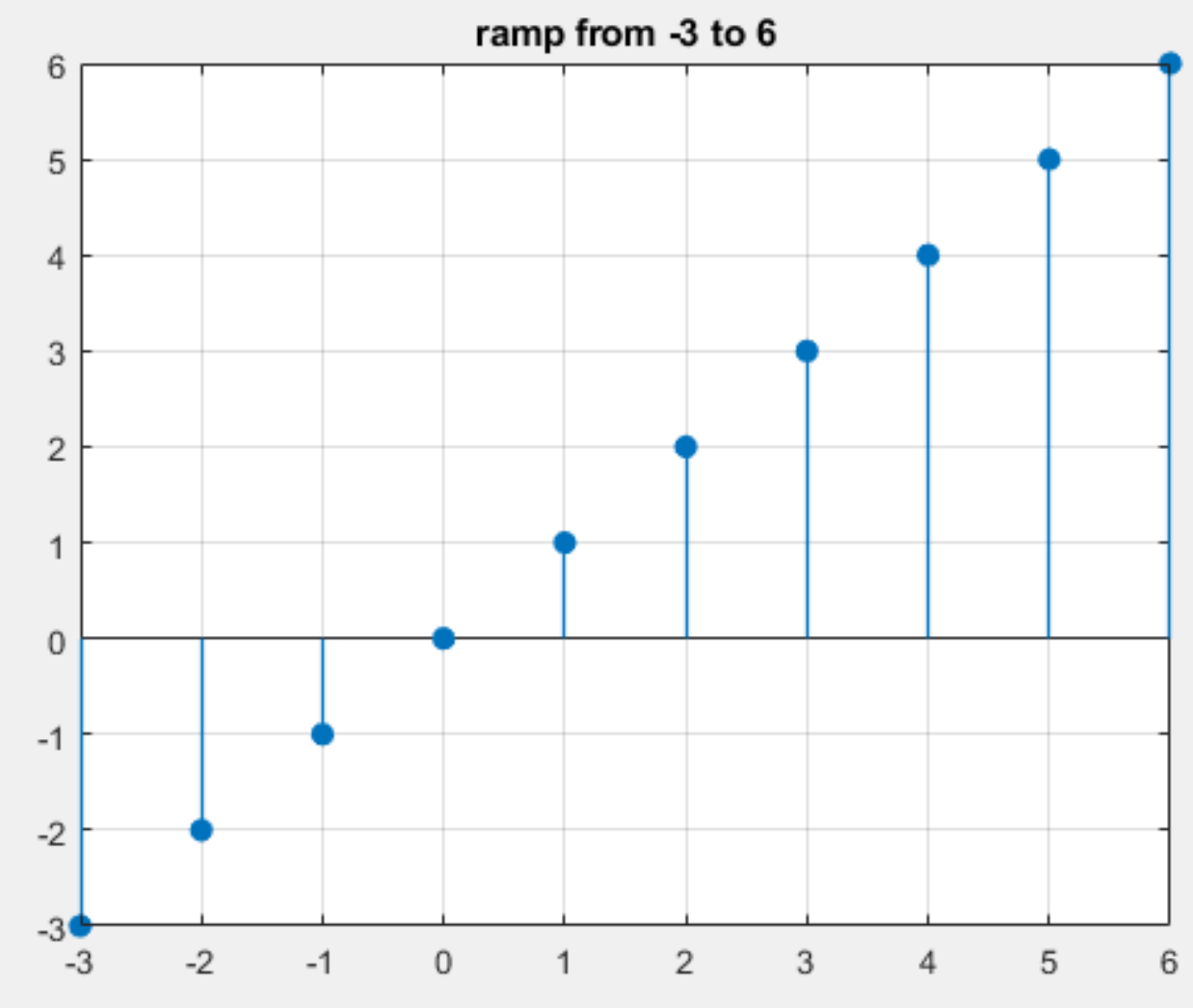
خروجی این سکشن:



سکشن سه

برای رسم این سیگنال، یک آرایه از 3- تا 6 ساخته (با استفاده از دستور ’ : ‘) سپس مقادیر هر دو محور نمودار را برابر این آرایه قرار می‌دهیم تا شیب ثابت ساخته شود.

خروجی این سکشن:



سکشن چهار

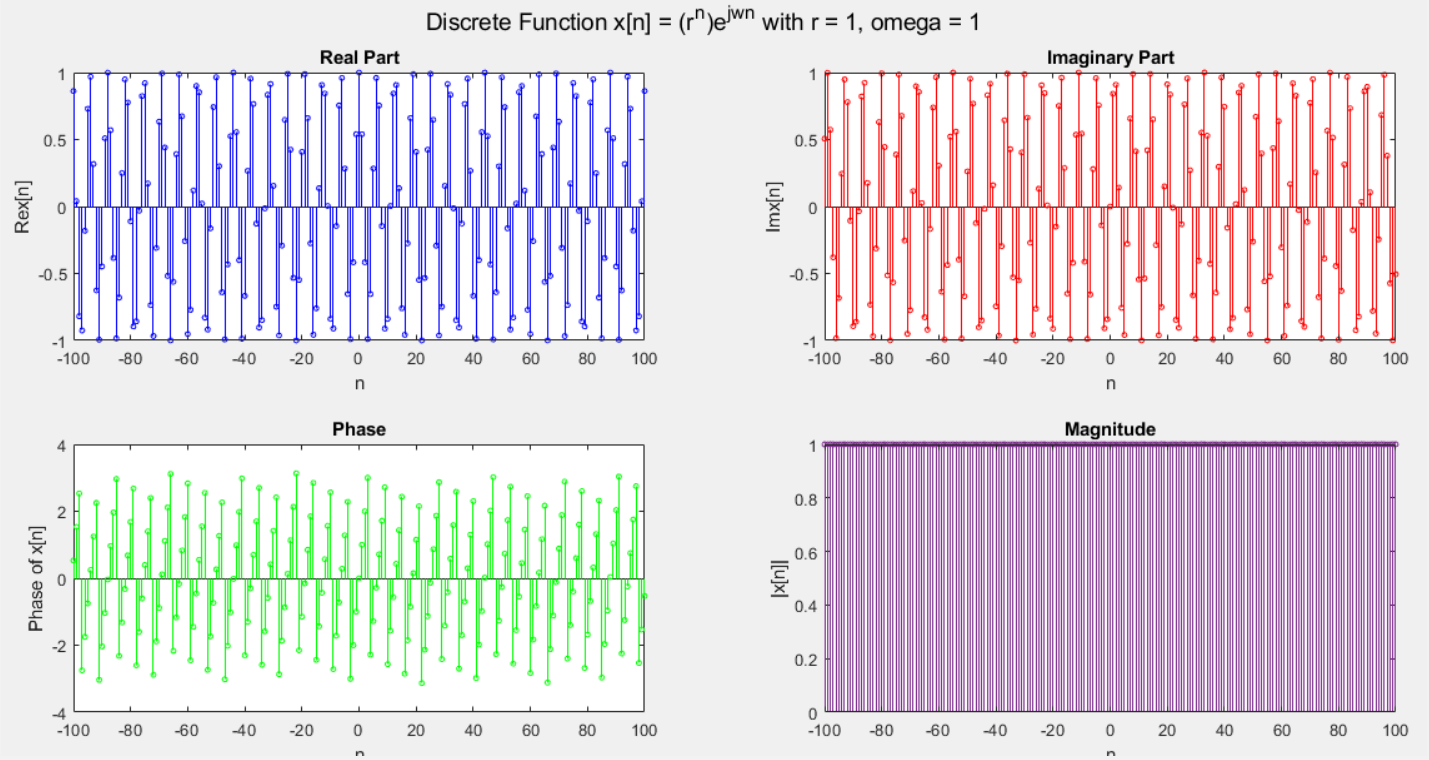
در ابتدا، برای تنظیم محور زمان، n را از 100- تا 100+ مقدار می‌دهیم.

پس از دریافت r و ω از کاربر، با استفاده از عملیات‌های element-wise در متلب (که با . نمایش داده می‌شوند) رابطه مورد نظر را محاسبه می‌کنیم.

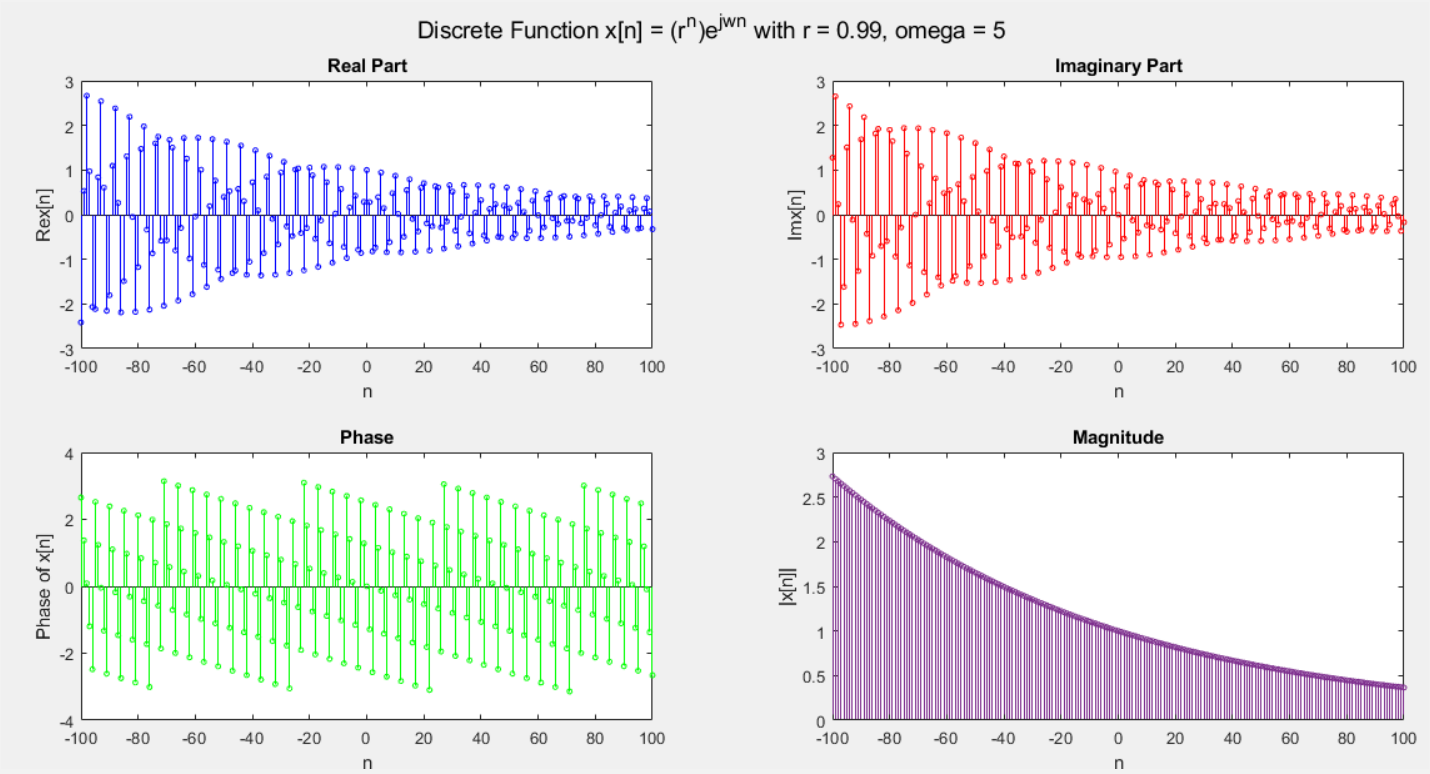
```
x_n = (r.^n) .* exp(1j .* w .* n);
```

در نهایت با استفاده از ۴ تابع abs – angle – imag – real ، قسمت‌های مورد نیاز را به دست آورده و نمایش می‌دهیم

خروجی این سکشن برای r = 1 , ω = 1



خروجی این سکشن برای $r = 0.99$, $\omega = 5$



بخش دوم

کد این بخش در فایل Q2.m قرار دارد.

سکشن یک

در ابتدا محور زمان را از 50- تا 50+ مقدار می‌دهیم.

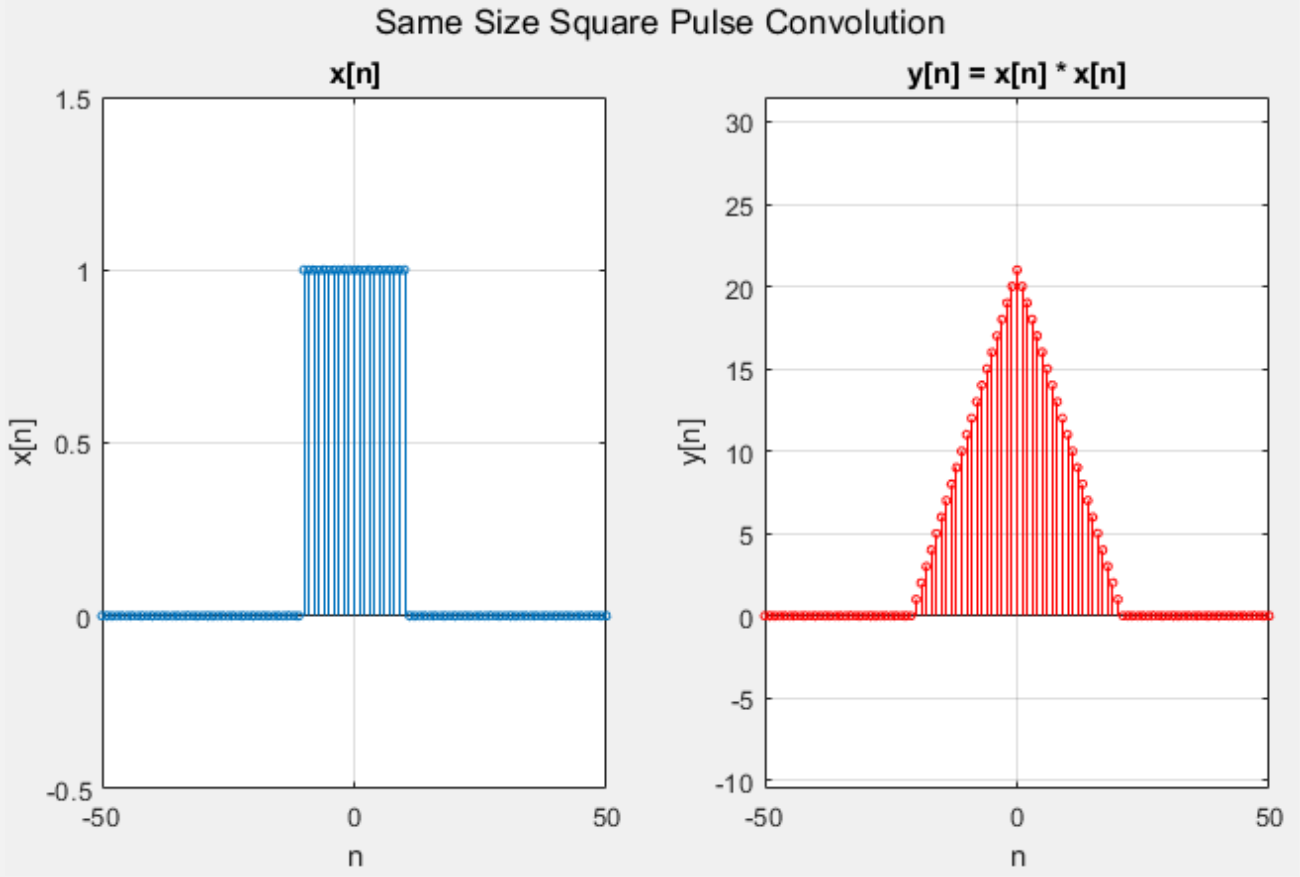
سپس موج مربعی گسسته را می‌سازیم، برای این منظور من از تابع repelem که یک بردار با یک مقدار تکرار شده می‌سازد استفاده کرده‌ام، به این صورت که ۳ بخش مختلف یک موج مربعی گسسته را ساخته و concat کرده‌ام.

```
x = cat(2,repelem(0,40),repelem(1,21),repelem(0,40));
```

این کد یک سیگنال مربعی با طول ۲۱ تولید می‌کند.

در نهایت این سیگنال را با خودش با استفاده از تابع conv، کانوالو می‌کنیم.

خروجی این سکشن:



سکشن دو

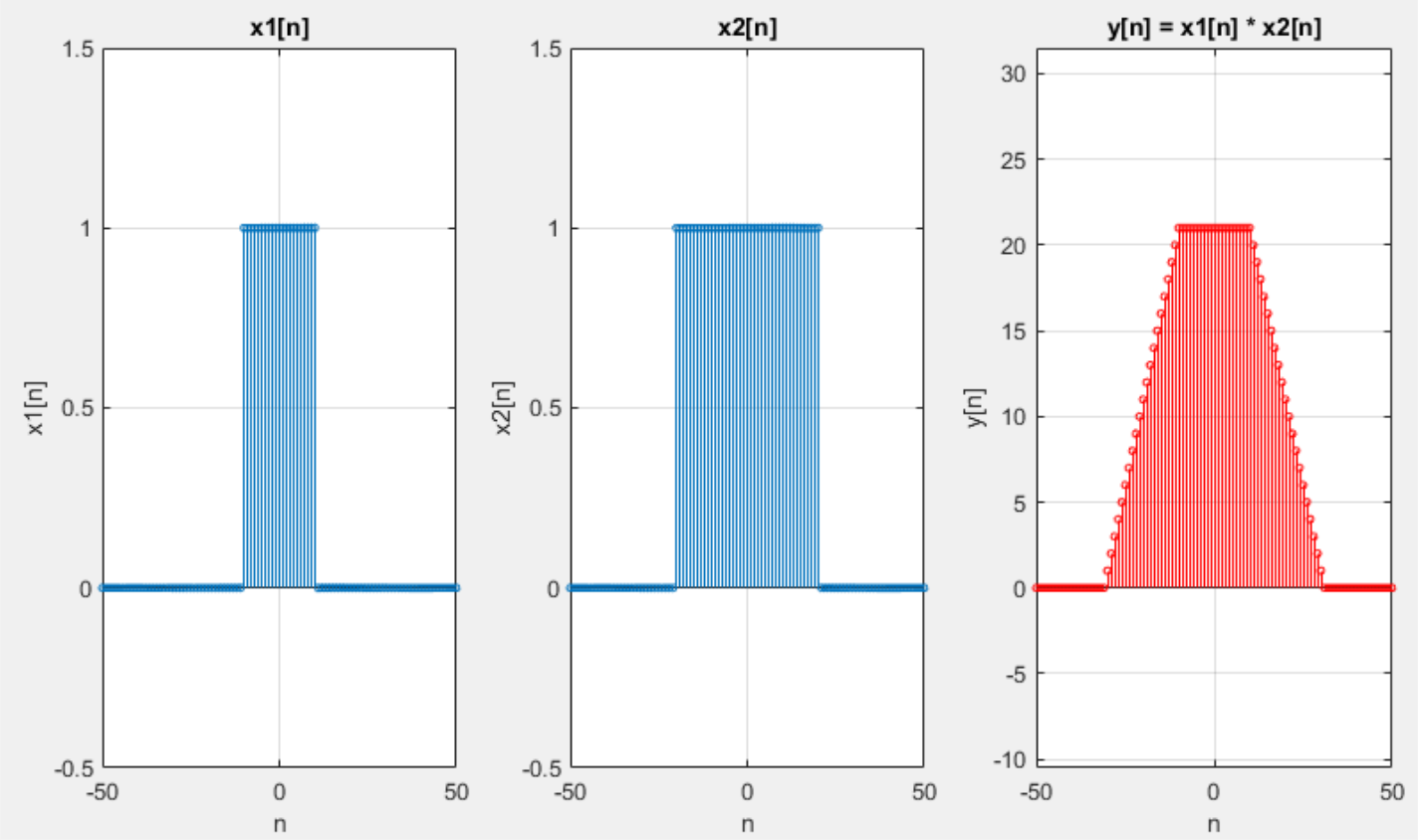
با استفاده از کدی مشابه قسمت قبل، دو سیگنال با طول‌های 21 و 41 تولید کرده و کانوالو می‌کنیم.

طول ساق پایین ذوزنقه برابر با جمع طول دو سیگنال یعن ۶۲ است

طول ساق بالای ذوزنقه برابر با تفاضل دو سیگنال یعنی ۲۰ است

ارتفاع ذوزنقه برابر با مساحت سیگنال کوتاه‌تر، یعنی ۲۱ است

خروجی این سکشن:



سکشن سه

در ابتدا دو سیگنال مورد نظر را پیاده‌سازی می‌کنیم

برای محاسبه $x(t)$، در ابتدا سیگنال $x_1(t) = e^{-t}$ را با استفاده از بردار محور زمان t مقداردهی کرده، سپس سیگنال $x(t)$ را برای نقاط خارج بازه برابر 0 و برای نقاط داخل بازه برابر $x_1(t)$ قرار می‌دهیم، برای اینکار از ایندکس‌دهی شرطی برروی محور t استفاده می‌کنیم. کد این تابع در فایل x_t_q2.m قرار دارد.

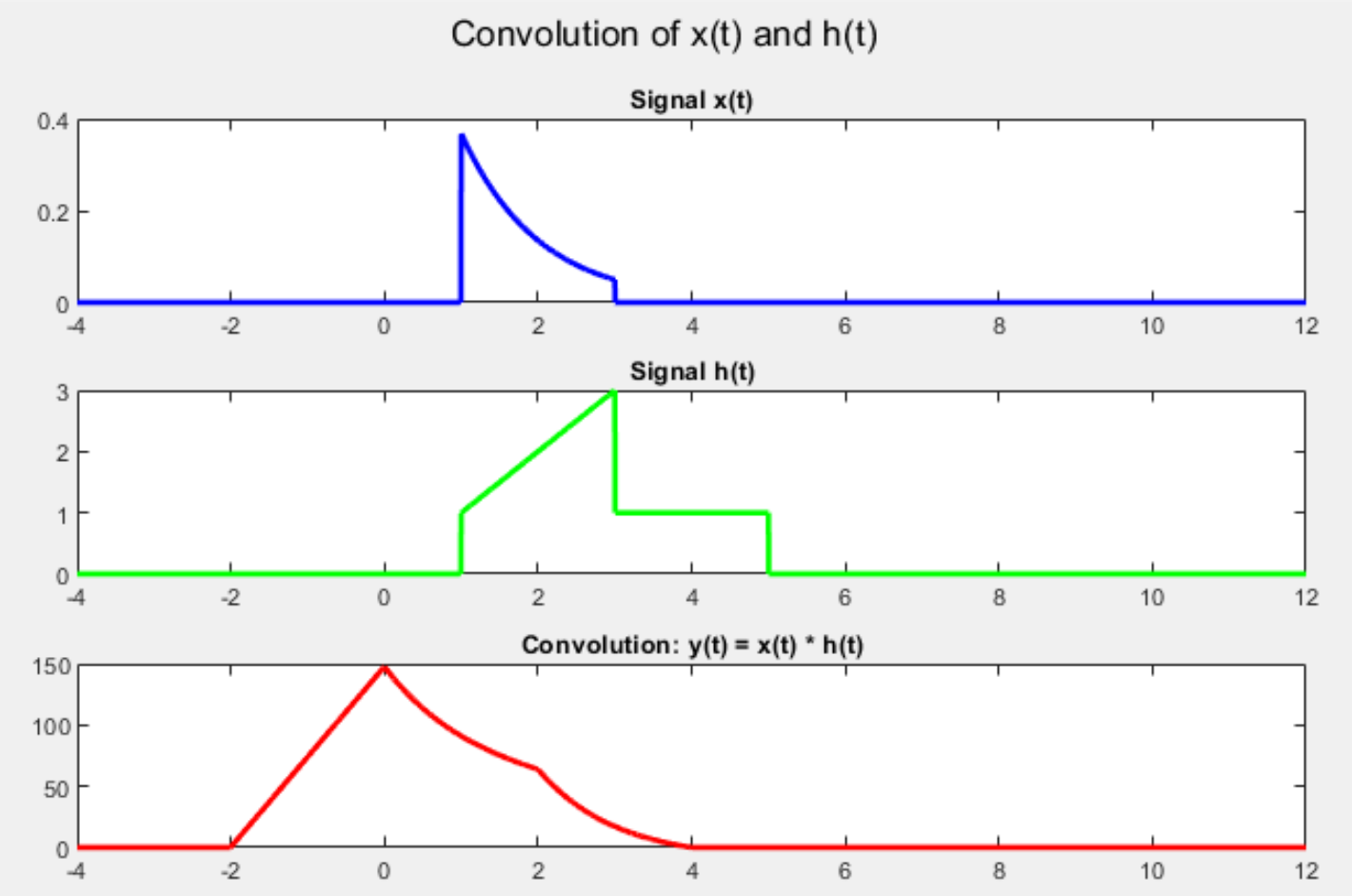
```
y = zeros(size(t));
y_exp = (exp(-t));
y((t>=1 & t<=3)) = y_exp(t>=1 & t<=3);
```

برای محاسبه $h(t)$، مشابه قسمت قبل از ایندکس‌دهی شرطی استفاده می‌کنیم، کد این تابع در فایل h_t_q2.m قرار دارد.

```
y = zeros(size(t));
y((1<=t & t<=3)) = t((1<=t & t<=3));
y((3<t & t<=5)) = ones(size(t(3<t & t<=5)));
```

در نهایت برای محاسبه پاسخ سیستم، کانولوشن این دو سیگنال را محاسبه می‌کنیم. به این صورت که محور زمان را به صورت پیوسته مقداردهی میکنیم تا محاسبه دقیق تر انجام شود (-4:0.005:12) ، سپس خروجی دو سیگنال را برای این زمان محاسبه کرده و کانولوشن را محاسبه می‌کنیم.

خروجی این سکشن:



بخش سوم

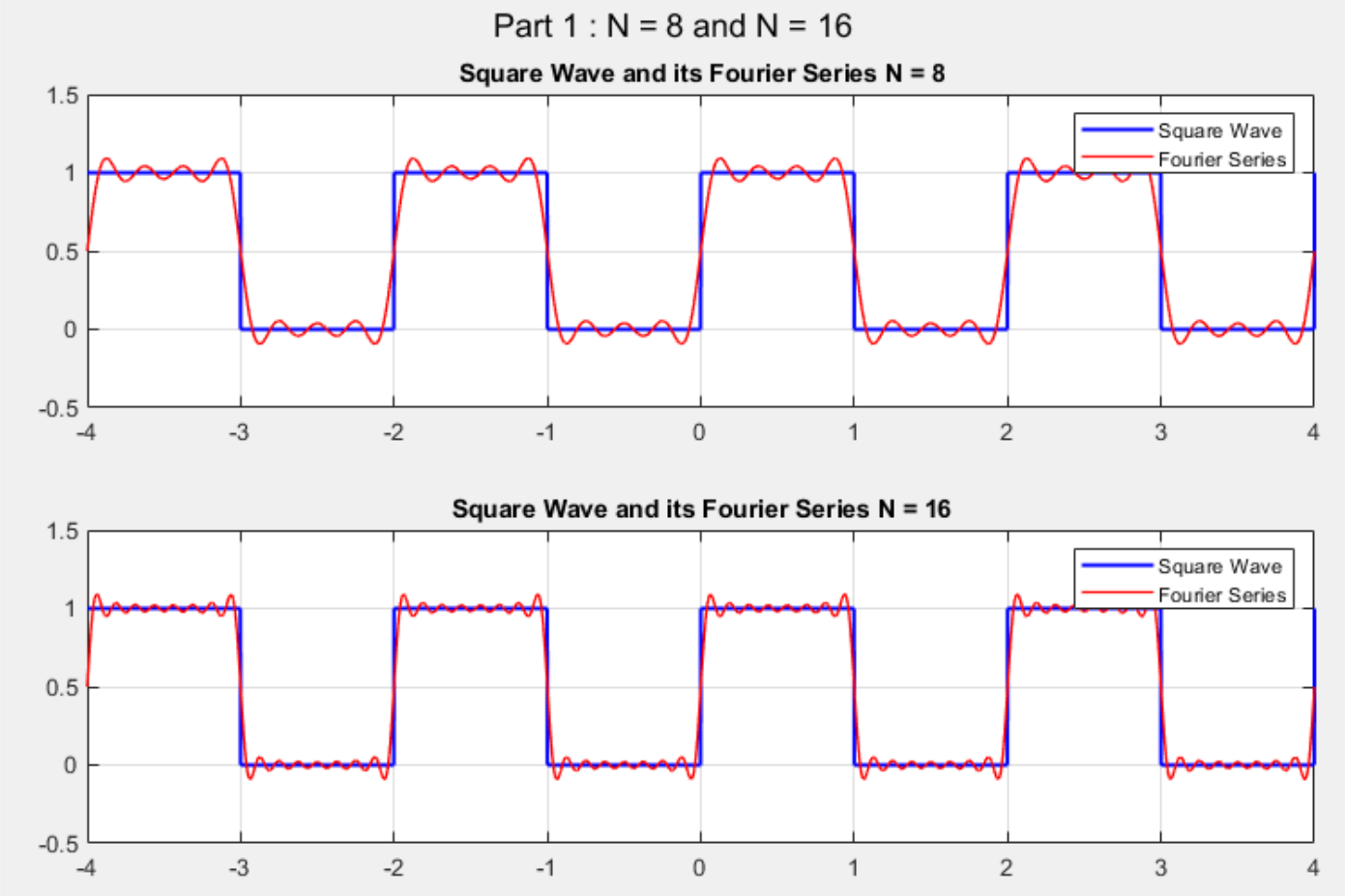
کد این بخش در فایل Q3.m قرار دارد.

سکشن یک

در ابتدا موج مربعی را می‌سازیم، برای انجام این کار اول مشخصات این موج (فرکانس، دوره تناوب و محور زمان) را مشخص کرده سپس با استفاده از دستور square موج را تولید و ذخیره می‌کنیم، پس از آن سری فوریه را برای مقادیر $N_1 = 8$, $N_2 = 16$ محاسبه می‌کنیم، برای اینکار ابتدا هر دو خروجی را برابر a0 که duty cycle موج است قرار می‌دهیم، سپس با استفاده از رابطه سینک، سری فوریه را برای مقادیر مختلف n محاسبه کرده و جمع می‌زنیم

```
x_fs1 = zeros(size(t))+(sum(x)/length(t));
for n = 1:2:N1
   x_fs1 = x_fs1 + 2/(n*pi) * sin(2 * pi * n * f * t);
end
```

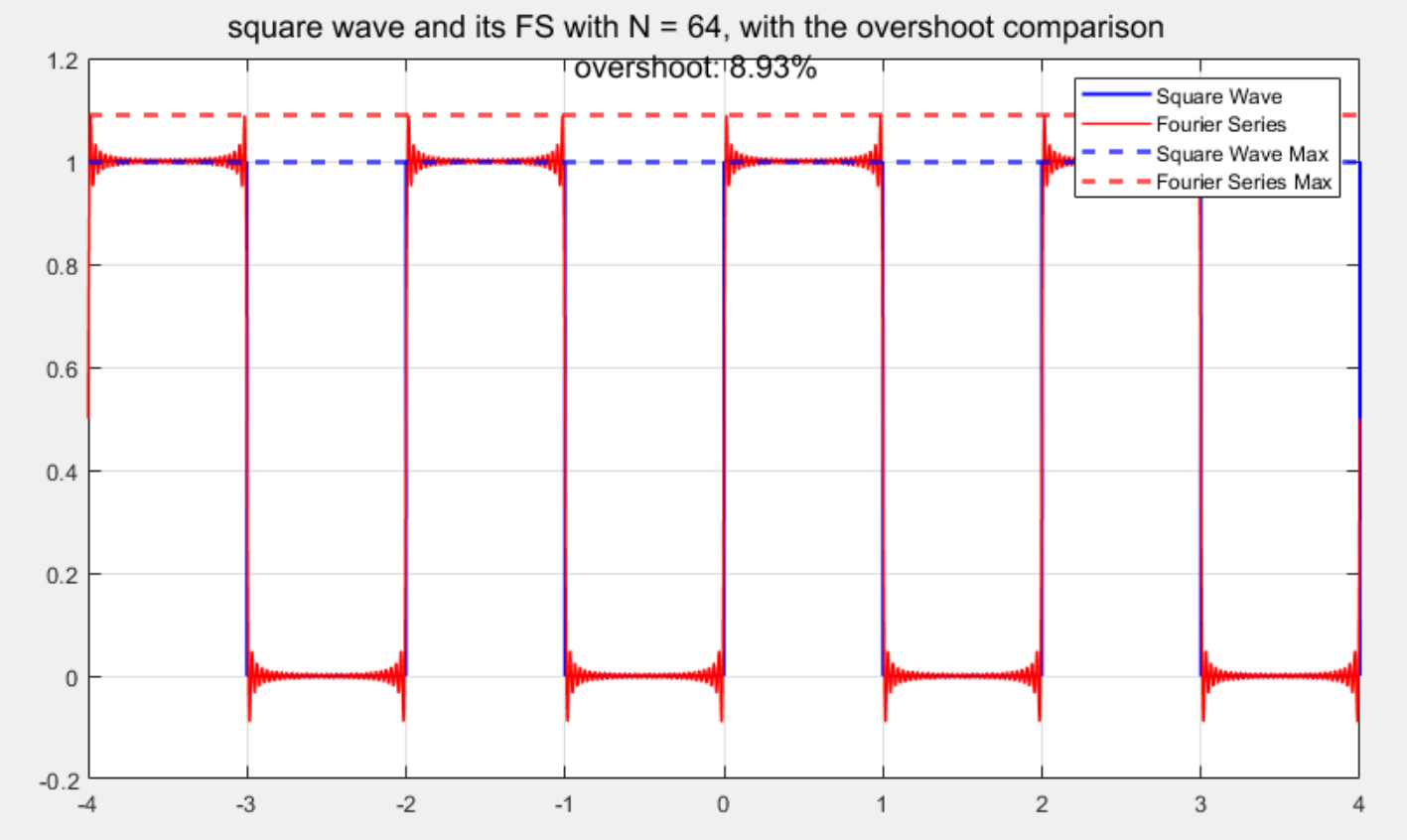
خروجی این سکشن:



سکشن دو

برای این بخش، سری فوریه را مشابه قسمت قبل برای $N = 64$ محاسبه می‌کنیم و برای محاسبه مقدار overshoot در پدیده گیبس، مقادیر max را برای موج و سری فوریه آن محاسبه کرده و مقایسه می‌کنیم، این مقدار برابر با 8.93% است.

خروجی این سکشن:

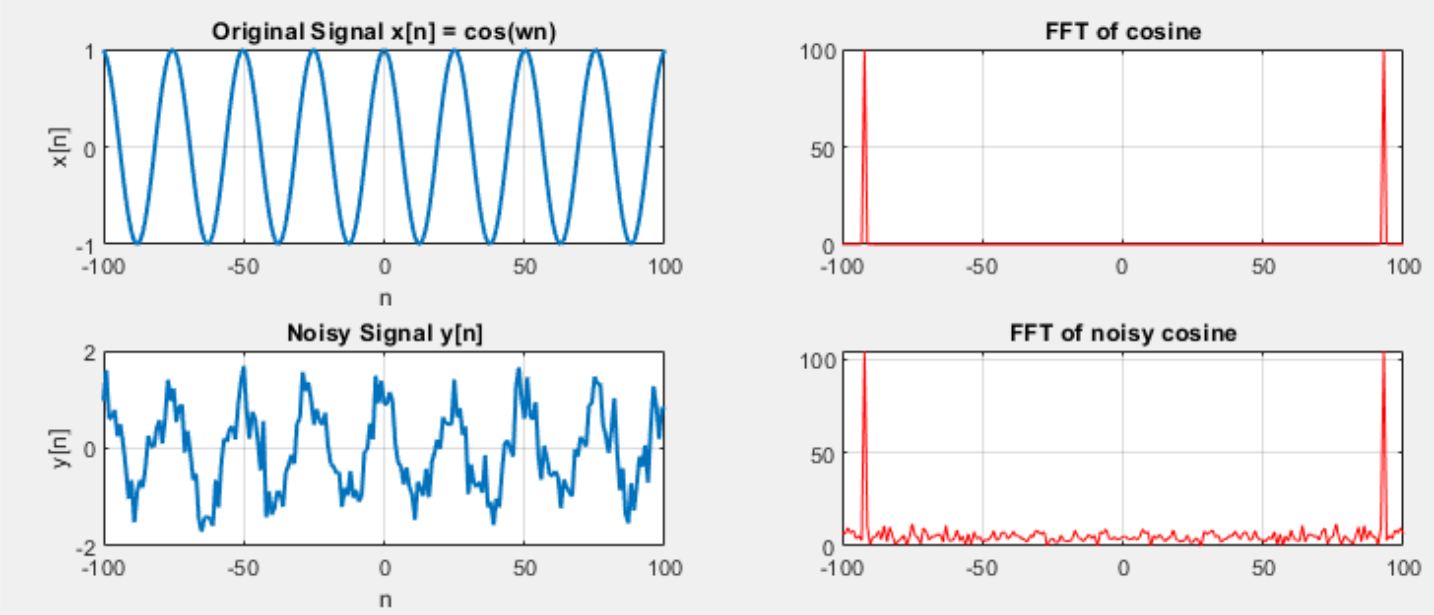


بخش ۴ - ۱

کد این بخش در فایل Q4_1.m قرار دارد.

سکشن یک

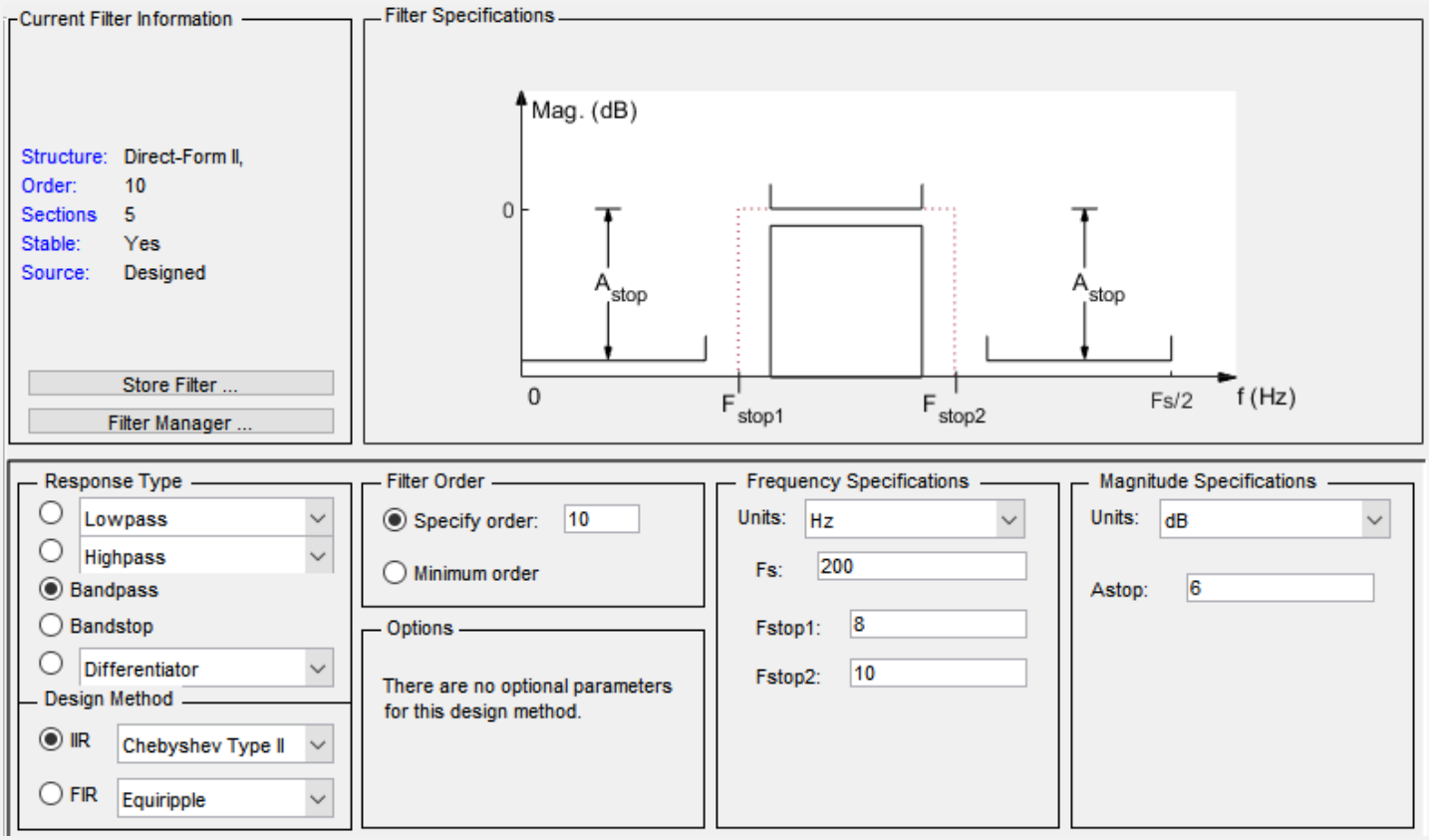
با انتخاب مقدار $\omega 0 = 0.25$ و محور زمان گسسته n از 100- تا 100+، سیگنال کسینوس مورد نظر را تولید کرده و ذخیره می‌کنیم. سپس با اعمال تابع AGWN با پارامتر $SNR = 5$، به سیگنال نویز اضافه کرده و آن‌را ذخیره می‌کنیم، سپس حوزه فرکانسی هر دو سیگنال را با استفاده از FFT محاسبه کرده و در نهایت همه این موارد را نمایش می‌دهیم



همانطور که در شکل قابل مشاهده است، سیگنال بدون نویز در یک بازه فرکانسی بسیار محدود دارای ضربه است، از این ویژگی می‌توان برای ساخت فیلتر در بخش بعد استفاده کرد.

سکشن دو

برای ساخت فیلتر، از مشخصات زیر در صفحه دیزاین فیلتر استفاده کردم.

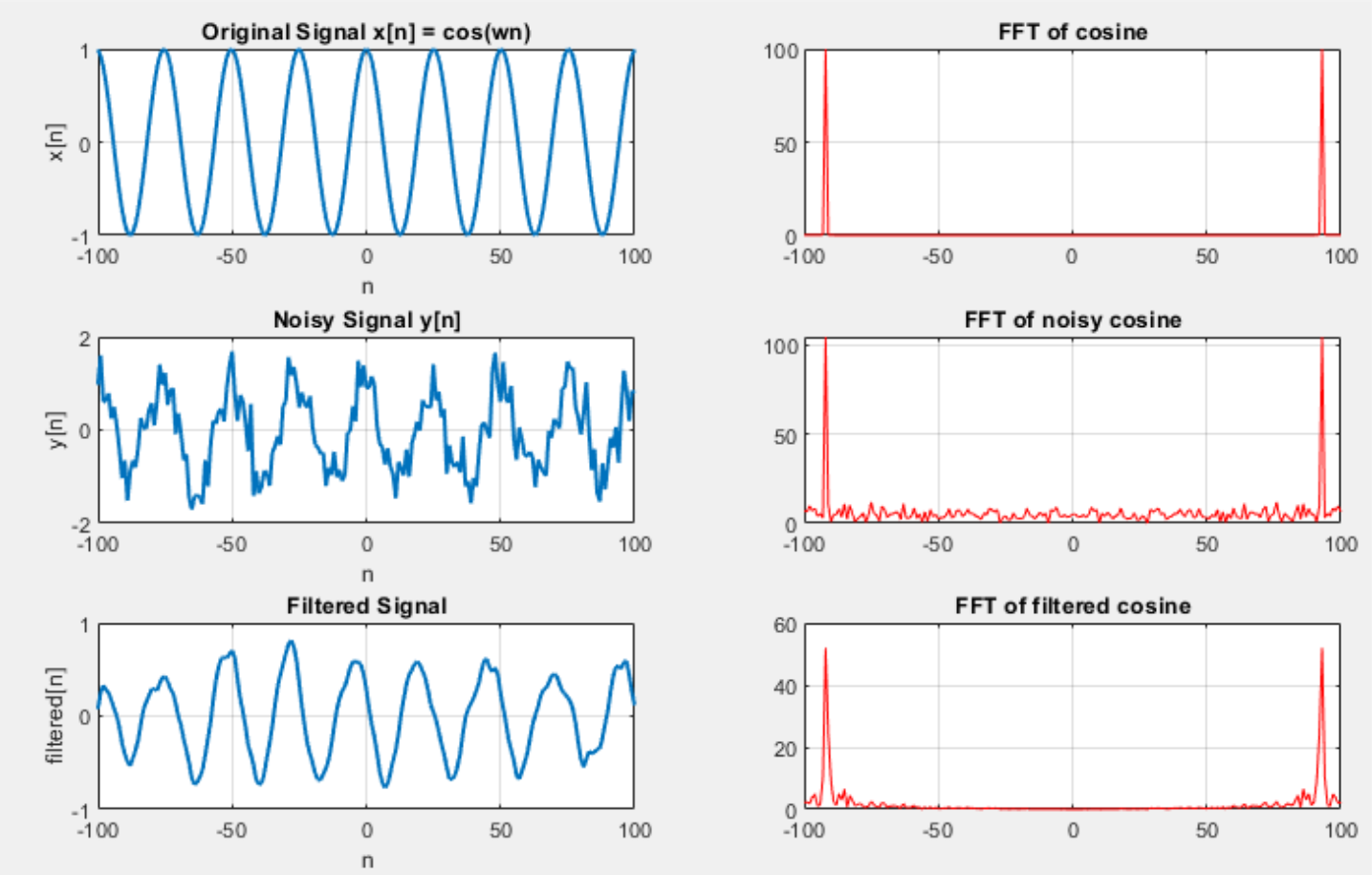


مقدار Fs برابر 200 است زیرا سیگنال در حوزه فرکانس تا فرکانس‌های -100 و 100 غیر صفر است، همچنین Fstop ها بر روی 8 و 10 تنظیم شده اند که بازه ضربه فرکانسی سیگنال کسینوسی ما است.

این فیلتر در فایل Q4_1_filter.fda ذخیره شده است.

پس از خروجی فیلتر به صورت آبجکت با نام Hd، فیلتر را بر روی سیگنال نویزی اعمال کرده، حوزه فرکانس سیگنال فیلتر شده را با FFT محاسبه کرده و در کنار سیگنال‌های قبلی نمایش می‌دهیم.

خروجی نهایی این بخش:

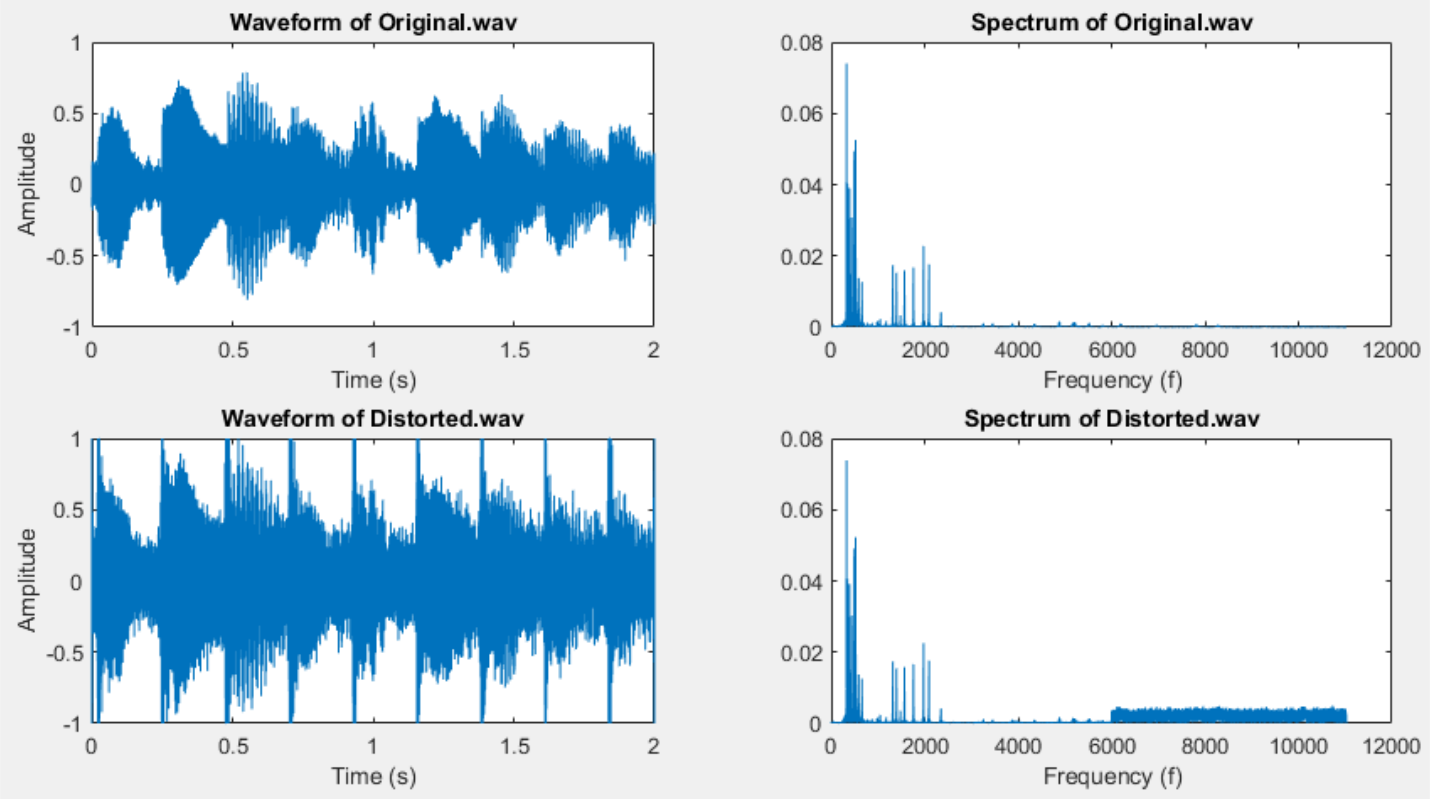


بخش ۴ - ۲

کد این بخش در فایل Q4_2.m قرار دارد.

سکشن یک

در ابتدا هر دو فایل صوتی را با تابع audioread لود کرده، سپس حوزه فرکانسی آن‌هارا با FFT محاسبه کرده و هر ۴ نمودار را نمایش می‌دهیم.



سکشن دو

با استفاده از تابع mean، مقدار MSE را محاسبه کرده و چاپ می‌کنیم

```
mse = mean((file1 - file2).^2)
```

مقدار محاسبه شده برابر 0.024987 است.

سکشن سه

همانطور که در شکل سکشن یک قابل مشاهده است، تشخیص نویز در حوزه زمانی بسیار مشکل است، اما در حوزه فرکانسی به راحتی می‌توان نویز را تشخیص داد که از بازه 6000 شروع می‌شود. در نتیجه برای حذف این نویز یک فیلتر lowpass با فرکانس 5750 بر فایل نویز دار اعمال کرده، و نتیجه را با استفاده از تابع audiowrite در فایل Recovered.wav ذخیره می‌کنیم.

سکشن چهار

مانند سکشن دو، MSE را بین صدای ریکاوری شده و صدای اصلی محاسبه می‌کنیم که برابر با مقدار 0.00070286 است که کاهش چشمگیری را در نویز نشان می‌دهد.

سکشن پنج

بله، اعمال فیلتر lowpass با حذف نویز در فرکانس‌های بالای 6000، باعث کاهش نویز و بهبود کیفیت صدا شده است.