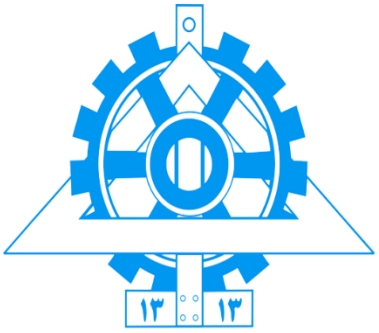
به نام خدا

دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر



تمرین کامپیوتری 4

اصول سیستم های مخابراتی

دکتر صباغیان

نام: فردین

نام خانوادگی: عباسی

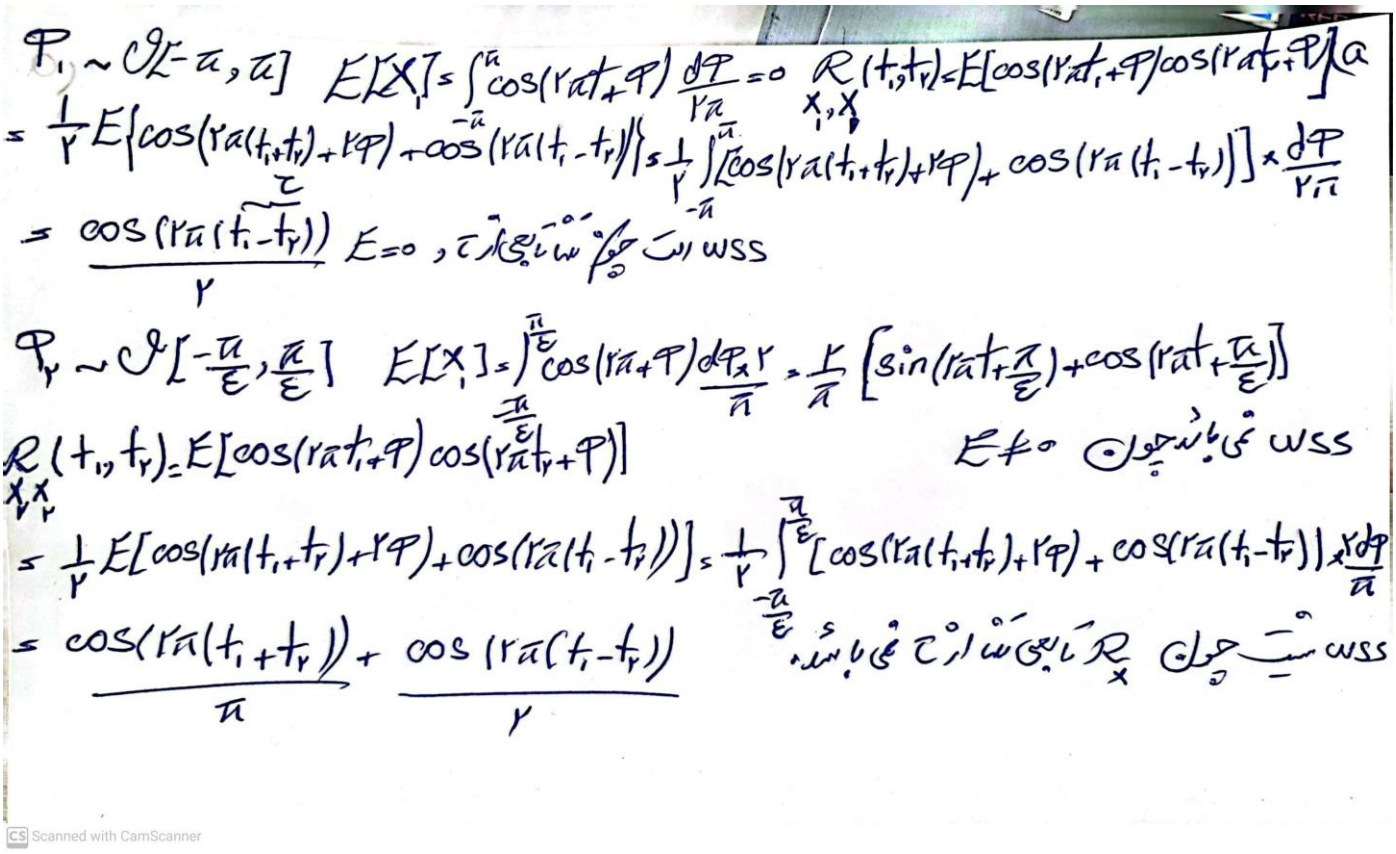
شماره دانشجویی : 810199456

نیمسال اول 02-1401

خواسته ها

بخش 1

A)



B)

Check the code!

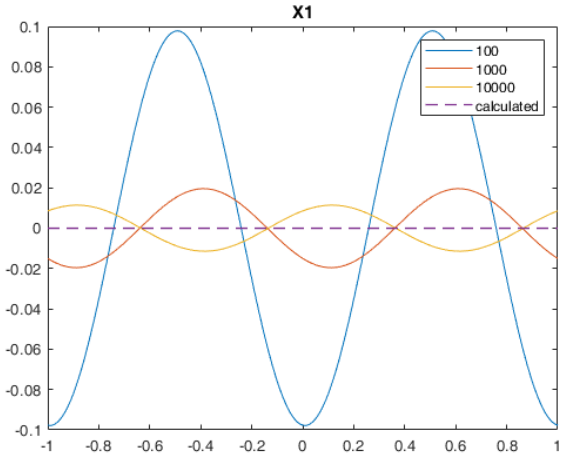
C)

```
m1_1 = 1×201
    -0.0977   -0.0978   -0.0976   -0.0970 ...

m1_2 = 1×201
    -0.0150   -0.0158   -0.0165   -0.0171 ...

m1_3 = 1×201
     0.0086    0.0090    0.0094    0.0098 ...
```

میانگین هر فرآیند در x1:



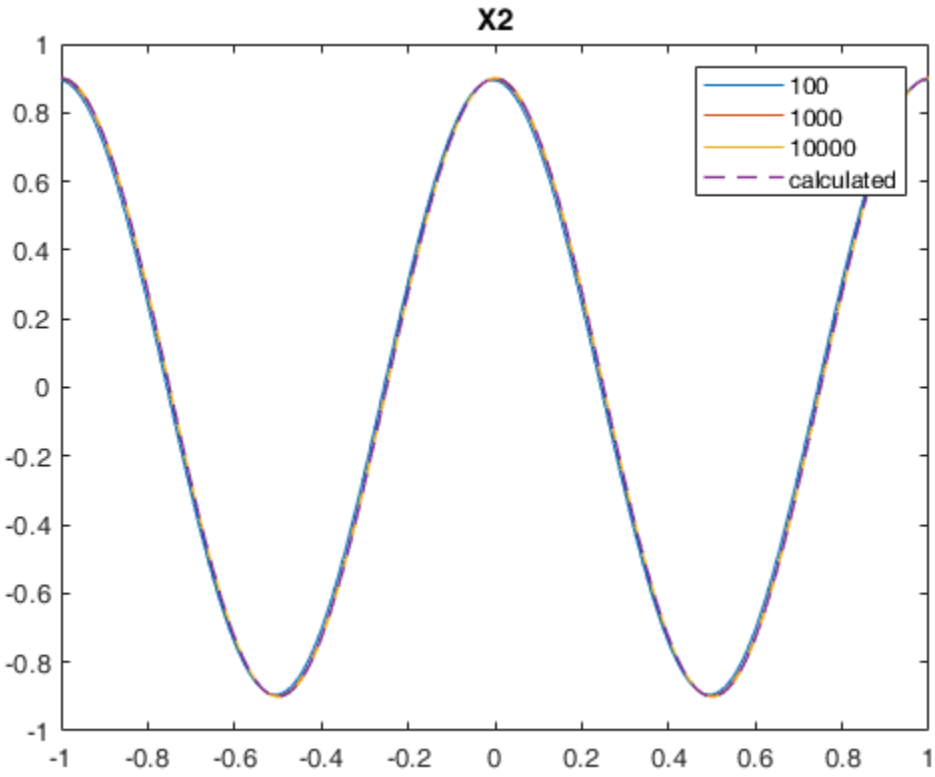
در اولی، چون با افزایش نمونه ها میانگین متغیر های تصادفی تشکیل دهنده فرآیند در لحظه های مختلف به صفر میل می کند، در نتیجه شرط استقلال زمانی میانگین فرآیند برقرار است.

```
m2_1 = 1×201
    0.8940    0.8900    0.8826    0.8716 ...

m2_2 = 1×201
    0.8978    0.8958    0.8903    0.8813 ...

m2_3 = 1×201
    0.9015    0.9000    0.8949    0.8862 ...
```

میانگین هر فرآیند در x2:

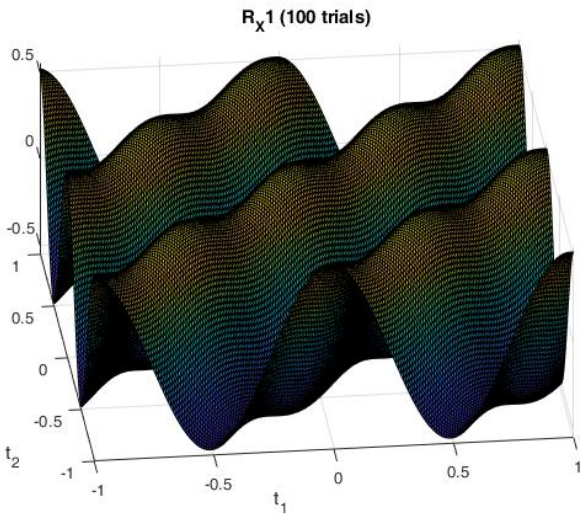


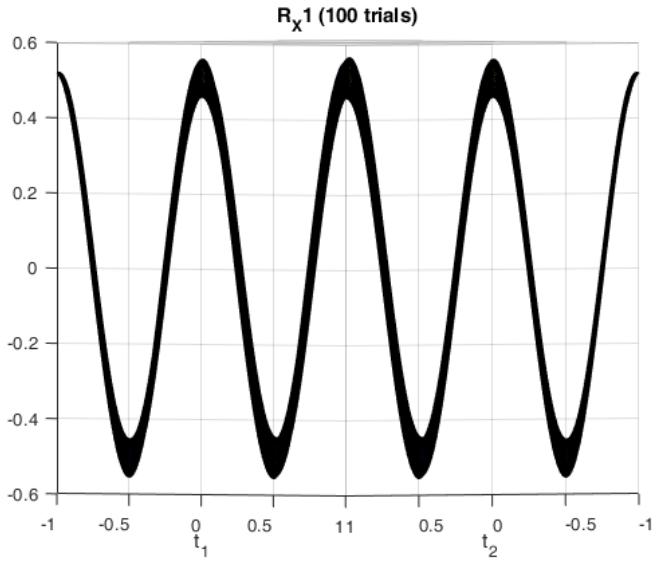
در دومی، چون با افزایش نمونه ها میانگین متغیر های تصادفی تشکیل دهنده فرآیند در لحظه های مختلف به یک تابع سینوسی میل میکند، در نتیجه نهایتا شرط استقلال زمانی میانگین فرآیند را برآورده نمی کند.

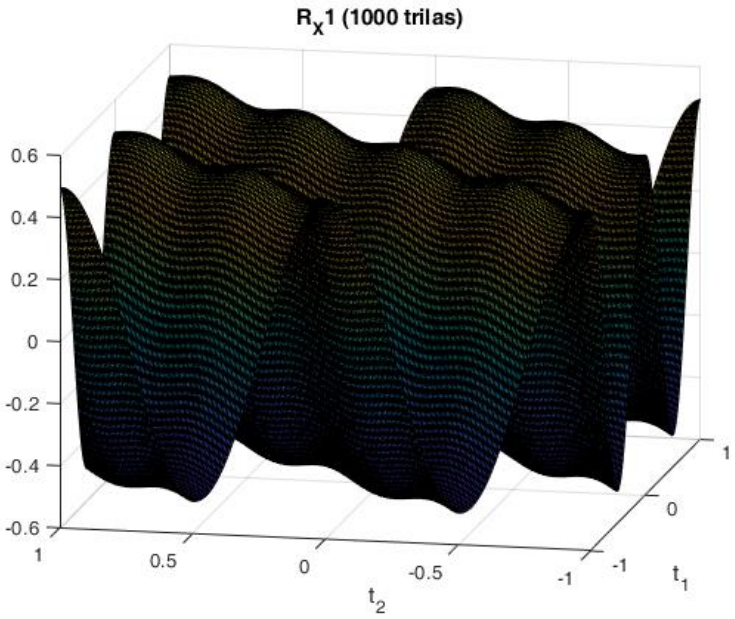
D)

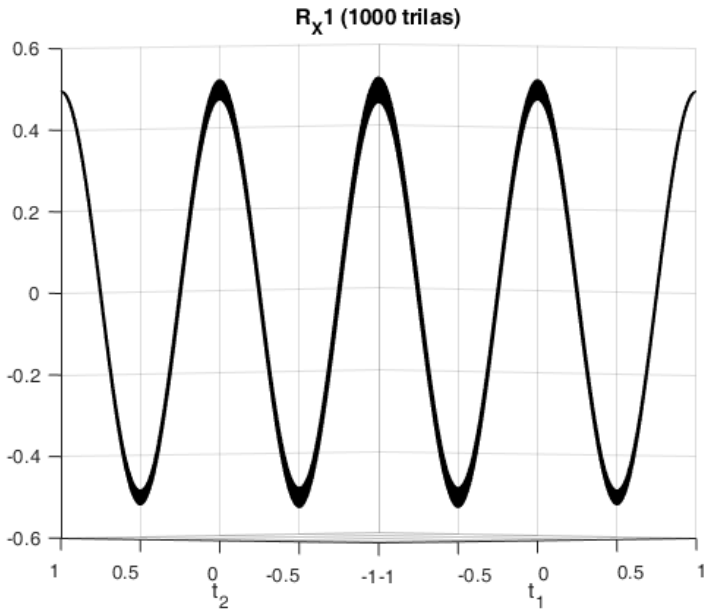
میانگین های محاسبه شده بر روی شکل های قبل به صورت خط چین مشخص است؛ میانگین هر دو فرآیند با افزایش تعداد داده ها به میانگین محاسبه شده همگرا می شود.

E)

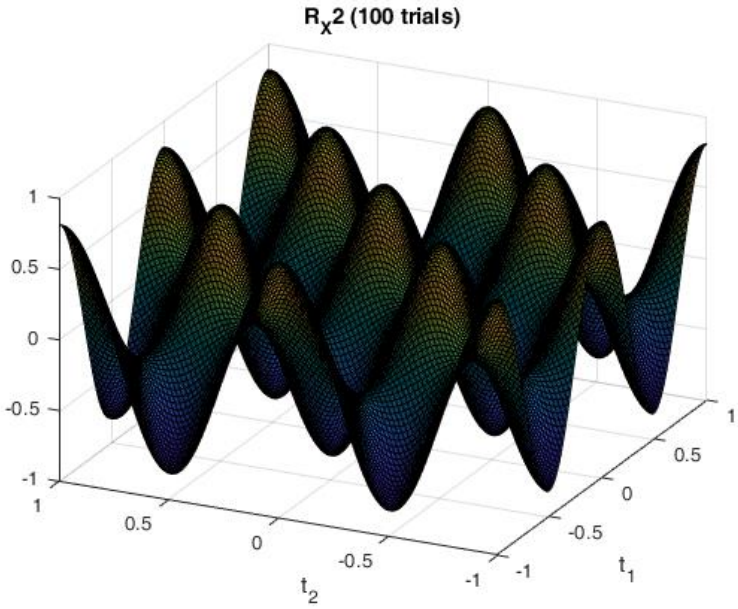


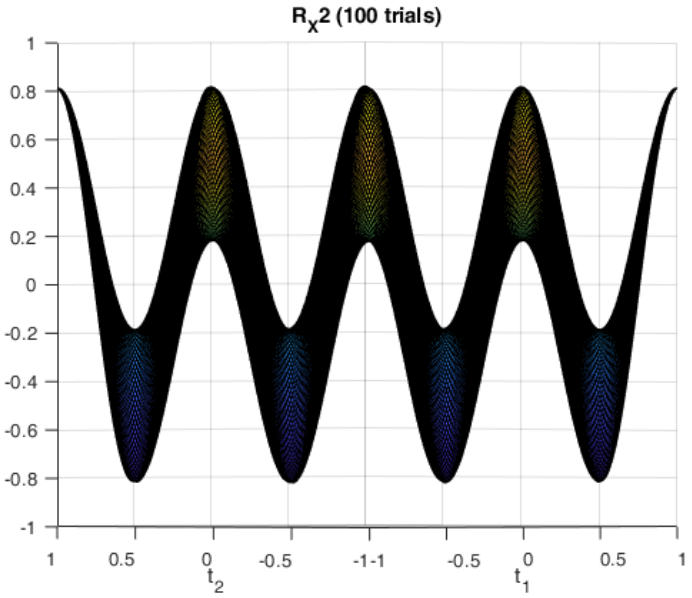


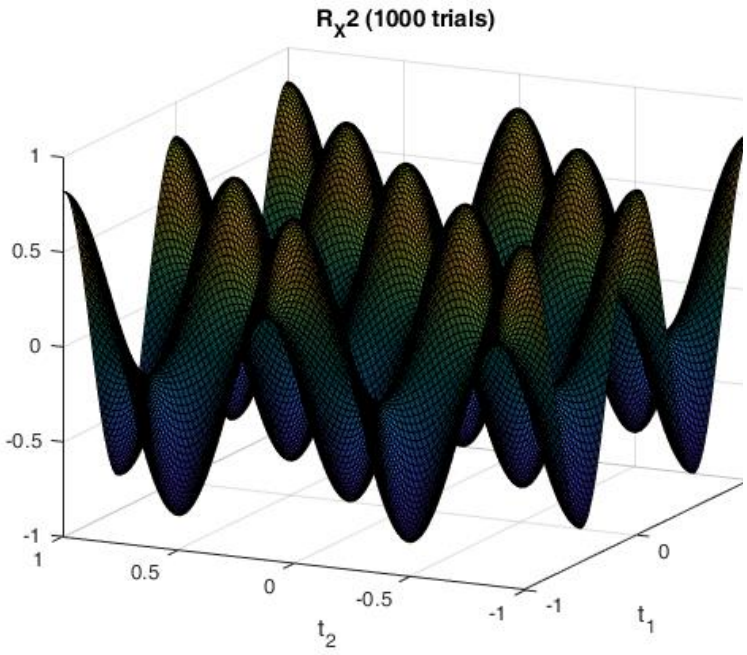


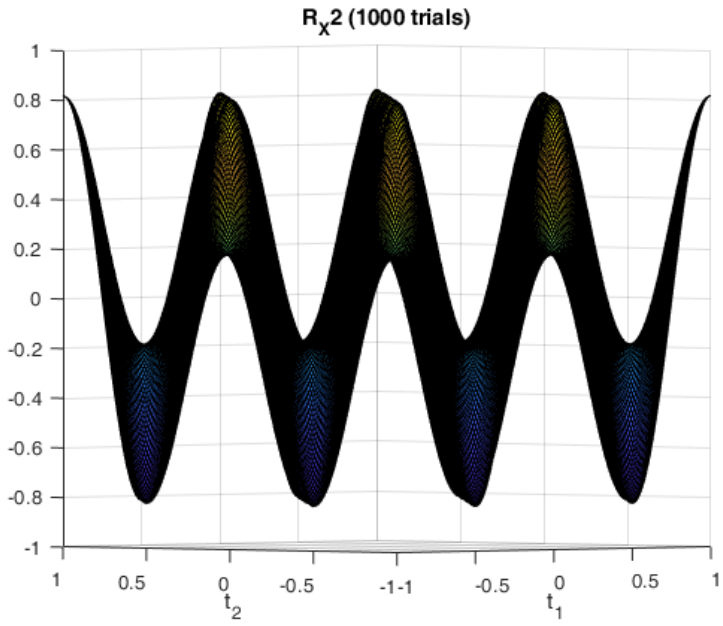


در نمودار های X1 اگر از زاویه ای به آن نگاه کنیم که در آن اختلاف زمانی t1-t2 ثابت باشد، به یک نمودار تقریبا دو بعدی می رسیم که با بیشتر شدن تعداد نمونه ها مسطح تر است که یعنی با افزایش نمونه ها، تابع خود همبستگی در ازای اختلاف زمانی معین، مقادیر یکسانی اختیار میکند و تنها تابعی از t1-t2. که شرط وابستگی تابع خودهمبستگی به تنها اختلاف زمانی برقرار است.





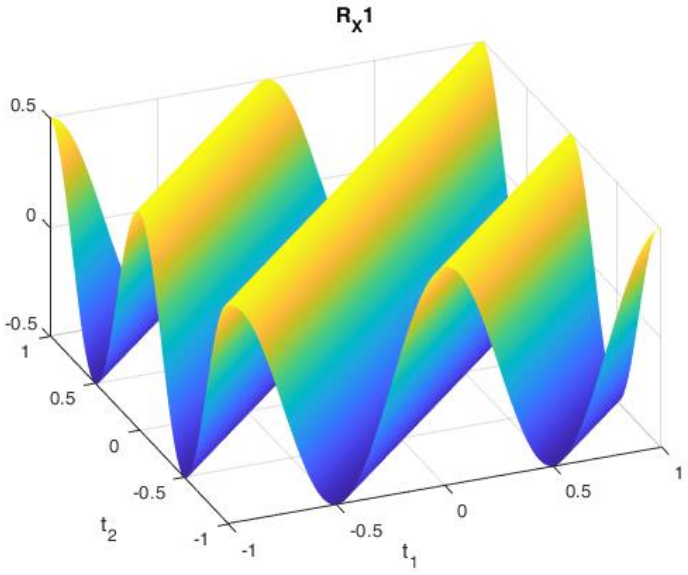


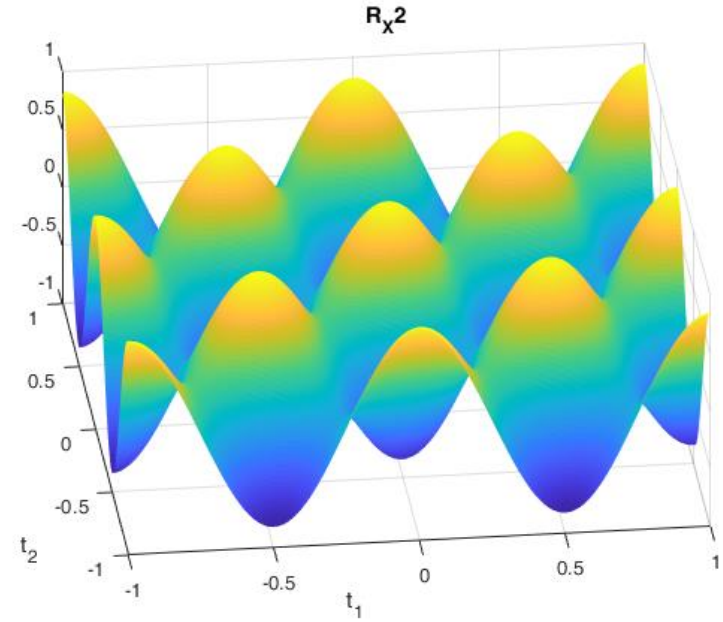


در فرآیند دوم، با نگاه از زاویه ای که در آن اختلاف t1-t2 ثابت باشد، تابع دارای مقادیر متفاوتی می باشد و مستقل از t1 و t2 است پس شرط وابستگی صرف به t1-t2 برقرار نیست.

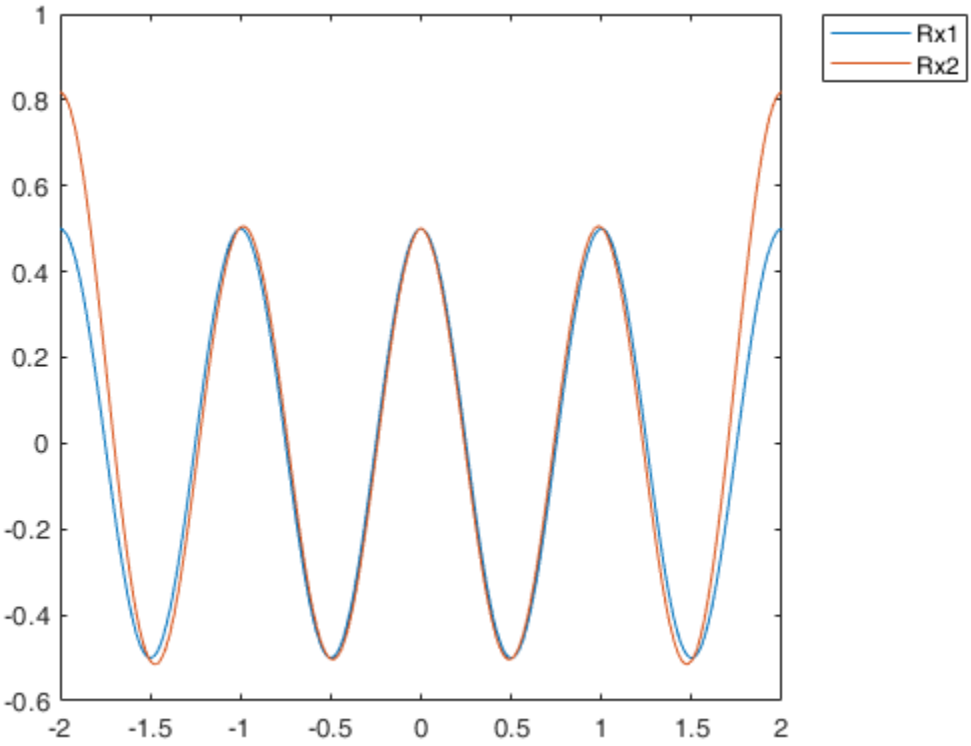
F)

توابع محاسبه شده برای تابع خودهمبستگی فرآیندهای اول و دوم، شباهت زیادی به توابع بدست آمده از نمونه های تصادفی وجود دارد و در اصل با افزایش نمونه ها و هموار تر شدن نمودارهای نمونه برداری شده به نمودارهای توابع محاسبه شده می رسیم.





G)

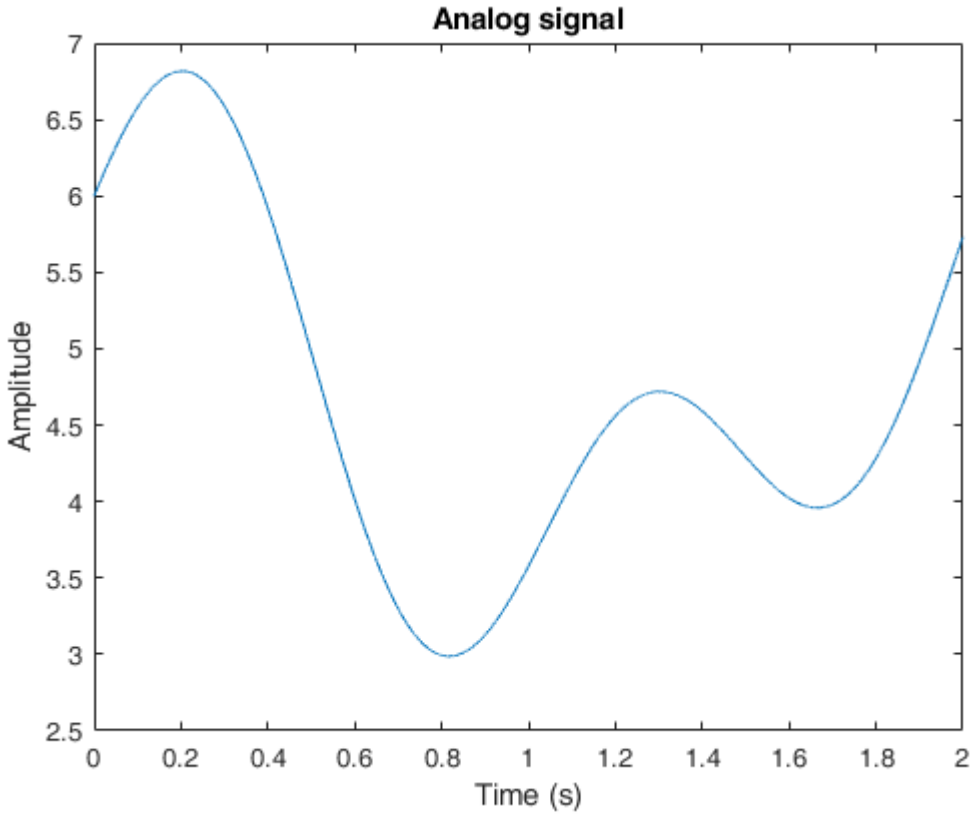


با جایگزین کردن میانگین مقادیر تابع خود همبستگی برای یک اختلاف زمانی مشخص در ازای زمان های مختلف، تابع همبستگی این دو فرآیند تطابق پیدا می کنند.

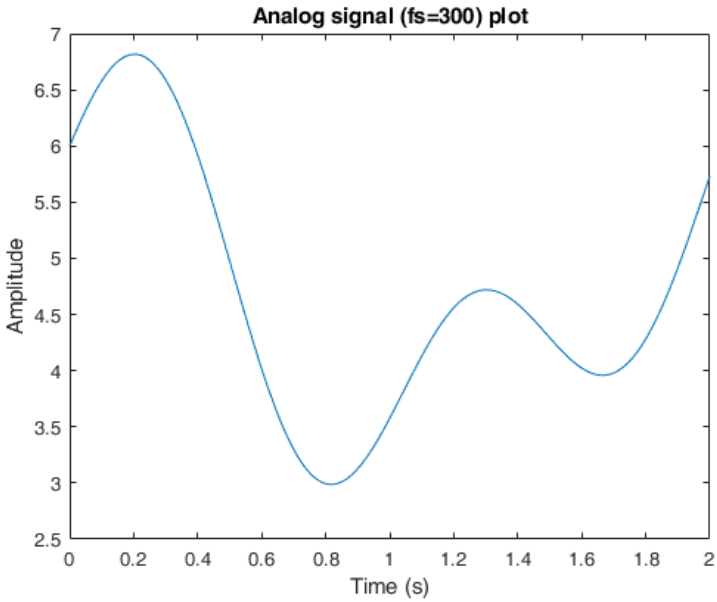
بخش 2

نمونه برداری

برای رسم این سیگنال, از 15000 نقطه استفاده شده است, که به شکل زیر در آمده است.



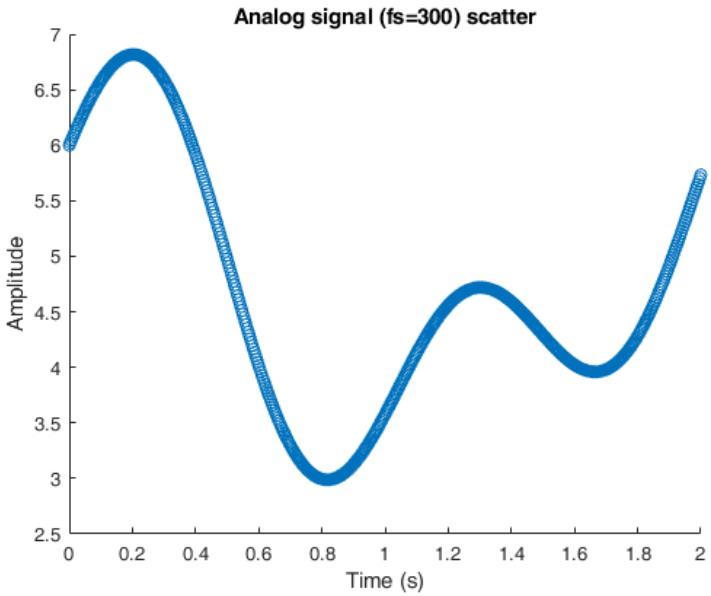
بعد از نمونه برداری با فرکانسی گفته شده شکل زیر را خواهیم داشت:

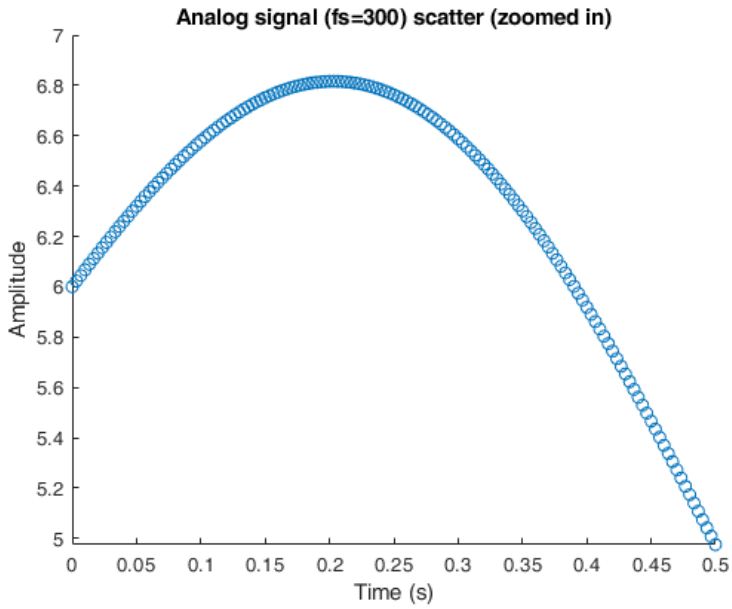


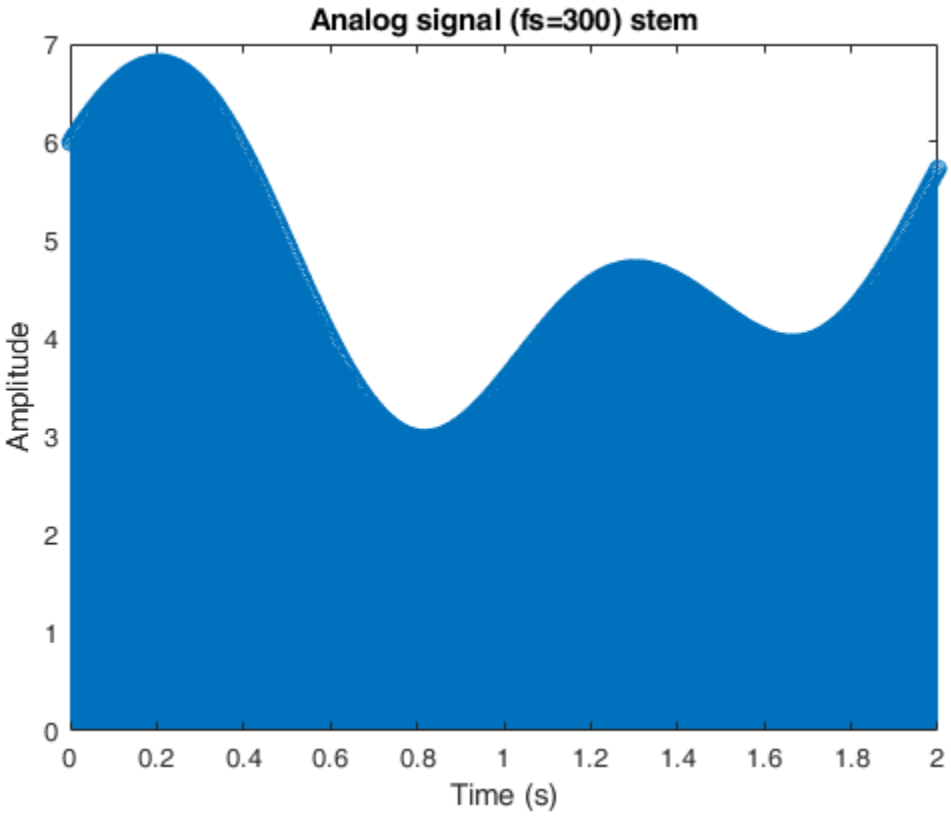
چون فرکانس 300 هرتز است و متلب حین رسم پیوسته سازی انجام میدهد, برای همین مشخص نیست که نمونه برداری رخ داده است.

برای همین با استفاده از دستور scatter و stem نیز این نمودارها را رسم میکنیم تا گسسته آن را ببینیم.

از آنجایی که 600 دیتا وجود دارد برای همین بازهم گسسته شدن دیتاها واضح نیست برای همین کمی بر روی نمودار زوم میکنیم تا مشخص شود.







همانطور که مشاهده میشود از سیگنال اولیه به صورت گسسته نمونه برداری شده است.

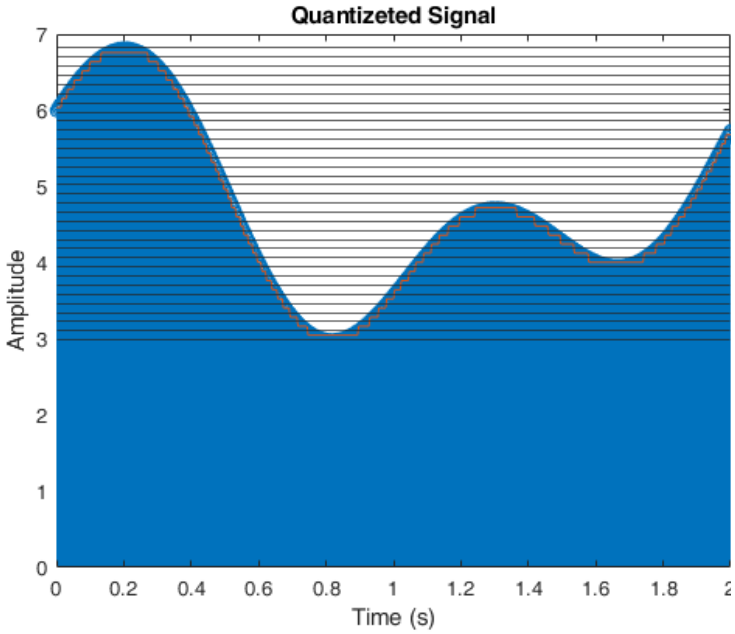
اعمال سطوح کوآنتیزاسیون

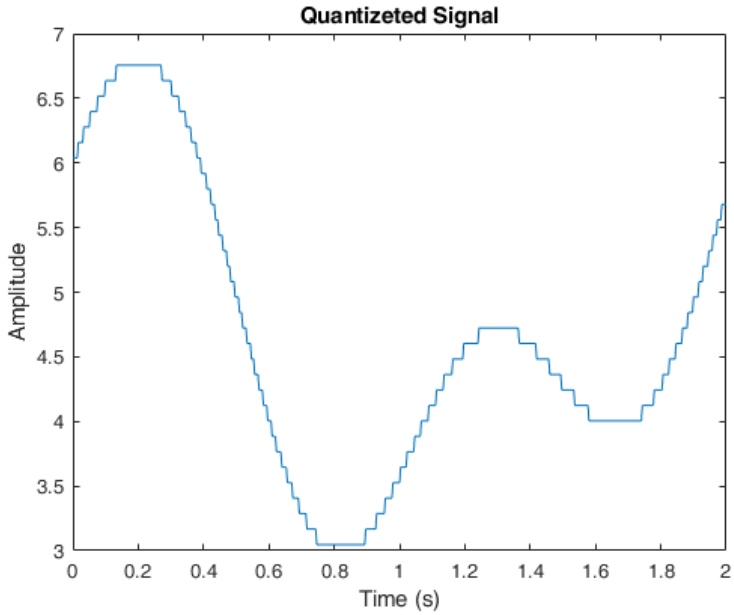
ابتدا باید 32 سطح را درست کنیم. برای اینکار نیاز به 33 خط داریم به طوریکه ازmin سیگنال تا max آن قرار گیرد.

```
N = 33;
lines = linspace(min(g_sample),max(g_sample),N);
Quantization = zeros(1,length(g_sample));
```

سپس با یک حلقه , for نقاطی که بین دو خط متوالی قرار میگیرند, بر روی مرکز آن دو خط تصویر میشوند.

شکل نهایی به صورت زیر خواهد شد.



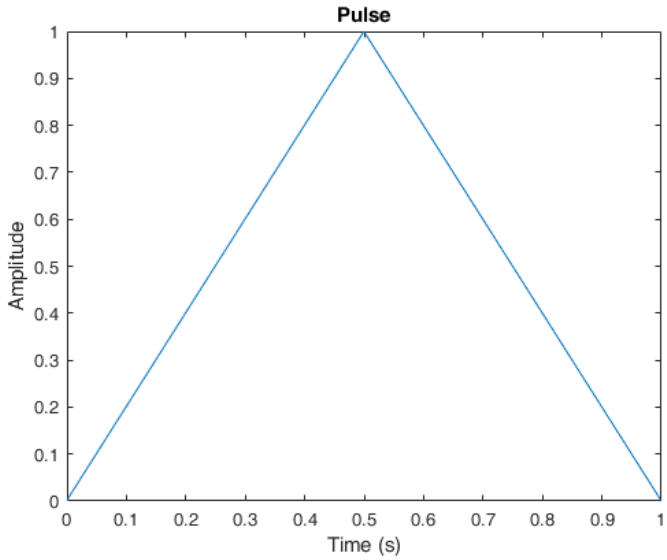


همانطور که در شکل چپ مشاهده میشود, سیگنال اصلی رسم شده, به صورت نمودار راست کوانتیزه شده است.

همچنین 33 خطی که تشکیل 32 سطح میدهند نیز مشخص شده اند و واضح است که مقادیر بین هر دوخط متوالی, روی مرکز سطح تصویر شده است.

دیجیتال سازی سیگنال کوآنتیزه شده

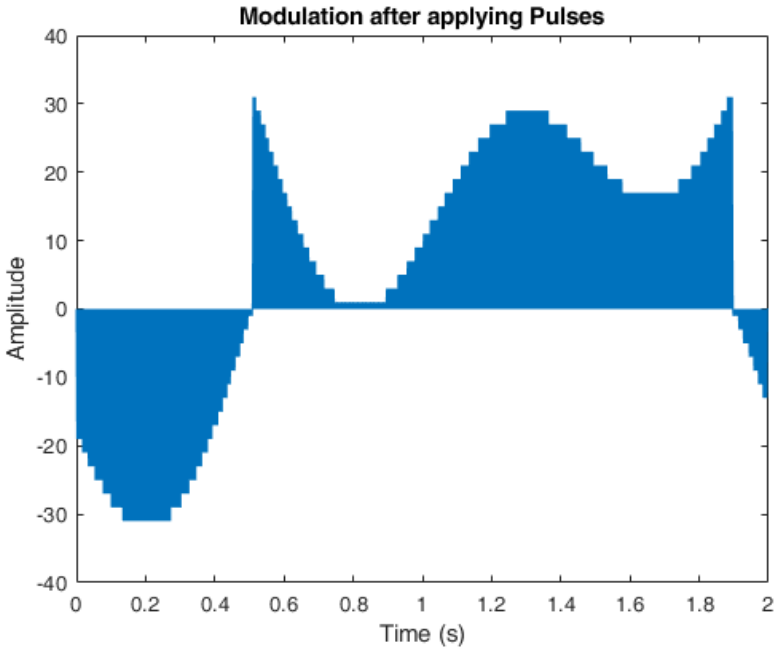
ابتدا پالس را با فرکانس خواسته شده میسازیم و آن را رسم می کنیم.



حال برای پیدا کردن انرژی این پالس, کافیست تا در یک حلقه for تمامی مقادیر آن را در خودش ضرب کرده و با یکدیگر جمع کنیم.

```
energy_pulse = 0;
for i=1:length(pulse)
    energy_pulse = energy_pulse + (pulse(i) * pulse(i));
end
energy_pulse
```

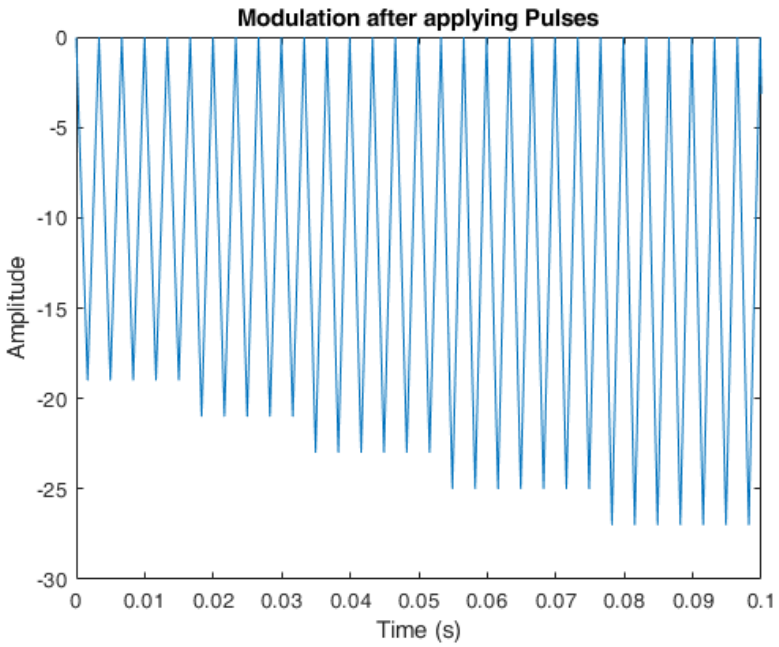
مشاهده میشود که انرژی پالس برابر با مقدار تقریبی 333 در آمده است.



به ازای هر کدام از 32 سطح کوانتیزاسیون، یک عدد از 0 تا 31 اختصاص دهید که همانطور که پی شتر ذکر شد، به عنوان دیجیت شناخته می شود.
میتوان دید که تعداد زیادی پالس های مثلثی از کنار هم قرار گرفتن, شکلی شبیه شکل رو به رو ایجاد کردند.

برای دیدن این پالس ها در شکل پایین کمی در نمودار

زوم شده است.



همچنین باید توجه کرد که هر پالس, به اندازه 1000 خانه ماتریکس فضا اشغال کرده زیرا فرکانس آن برابر با 1000 بوده است و تعداد پالس ها نیز برابر با 600 میباشد.
دلیل اینکه شکل نسب به محور x قرینه است و نسبت به سیگنال اصلی نیز قرینه است, همان موضوعی است که در ابتدای این سکشن بحث شد.
اگر عدد گذاری دیجیت ها به طوری میبود که سطح وسط برابر با 0 میبود و دیگر سطح ها متناسب با آن تغییر میکردند, آنگاه شکل مدوله شده نیز برابر با

شکل سیگنال اصلی میشد.

دریافت سیگنال دیجیتال در گیرنده

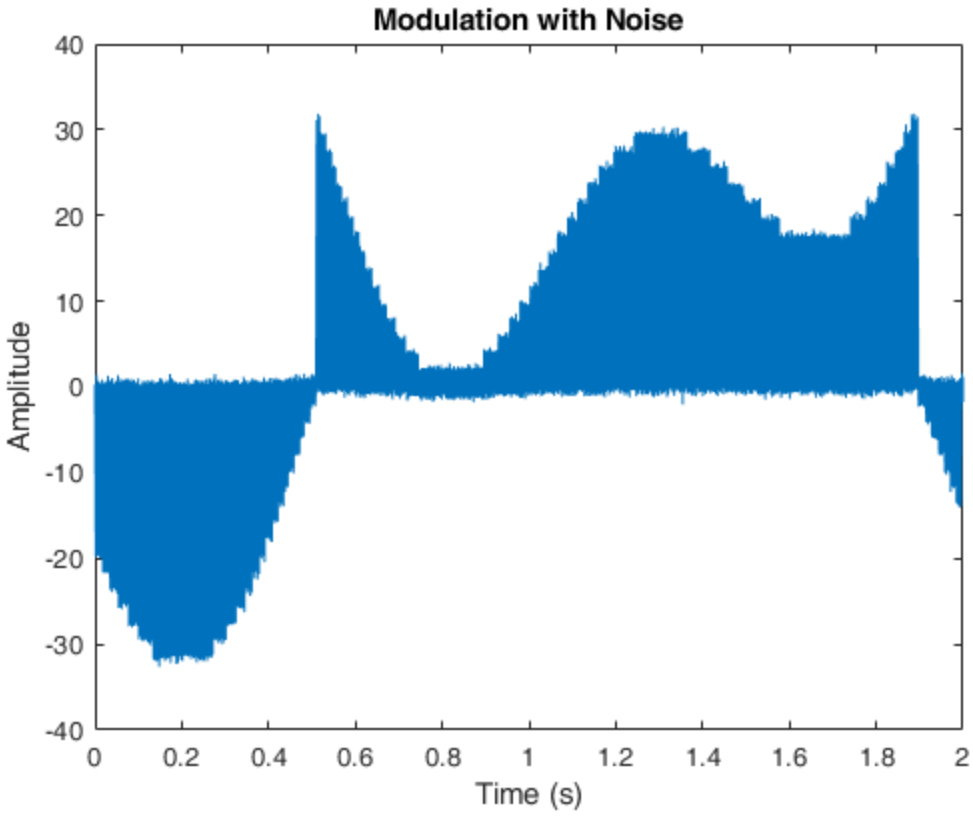
ابتدا با توجه به آنکه گفته شده مقدار SNR برابر با 2 دسی بل باشد, مقدار انرژی نویز و سیگنال را به دست می آوریم و سپس مقدار واریانس نویز را محاسبه میکنیم. ابتدا نویز را با تابع randn با مقادیری رندوم به دست می آوریم و پس از محاسبه واریانس, با تابع normrnd آن را دوباره میسازیم.

energy_signal = 4.1396e+05
energy_noise = 1.7939e+04

$$SNR = 20dB \rightarrow 10log_{10}^{p} = 20 \rightarrow \frac{p_x}{p_n} = 100, \frac{413960}{A^2 17939} = 100 \rightarrow A = 0.48$$

نویز را با پارامترهای گفته شده به صورت زیر تعریف میکنیم:

```
noise = normrnd(0,3.81,[1,length(modulate_signal)]);
```



و در نهایت با اضافه کردن آن به سیگنال خواهیم داشت

دیکود کردن سیگنال دیجیتال

ابتدا به ترتیب, به ازای هر سمپل از مدولاسیون که به آن نویز اضافه کردیم, پالس پایه را در آن ضرب کرده و سپس با استفاده از sum انرژی متقابل آن را محاسبه میکنیم. سپس این انرژی را به انرژی پالس پایه تقسیم میکنیم.

با این کار آرایه ای به اندازه سیگنال کوانتیزه شده خواهیم داشت که هر درایه آن برابر با مقدار سطح سیگنال کوانتیزه شده میباشد.

تمامی موارد بالا طبق کد زیر انجام شده است.

```
noise_pulse = zeros(1,length(pulse));
energy_each_pulse = zeros(1,length(bits));
for i=1:length(bits)
   noise_pulse = signal_with_noise(((i-1)*1000)+1:i*1000).*pulse;
   energy_each_pulse(i) = sum(noise_pulse) / energy_pulse;
end
```

که bits همان دیجیت 0 تا 31 میباشد که به هر سطح اساین شده است و pulse نیز همان پالس پایه است.

حال مقادیری که در قسمت قبل به دست آمده را باید با آرایه دو بعدی که در بخش دیجیتال سازی سیگنال کوانتایز شده به دست آورده بودیم مقایسه کنیم تا بتوانیم سطوح کوانتیزاسیون را بیابیم.

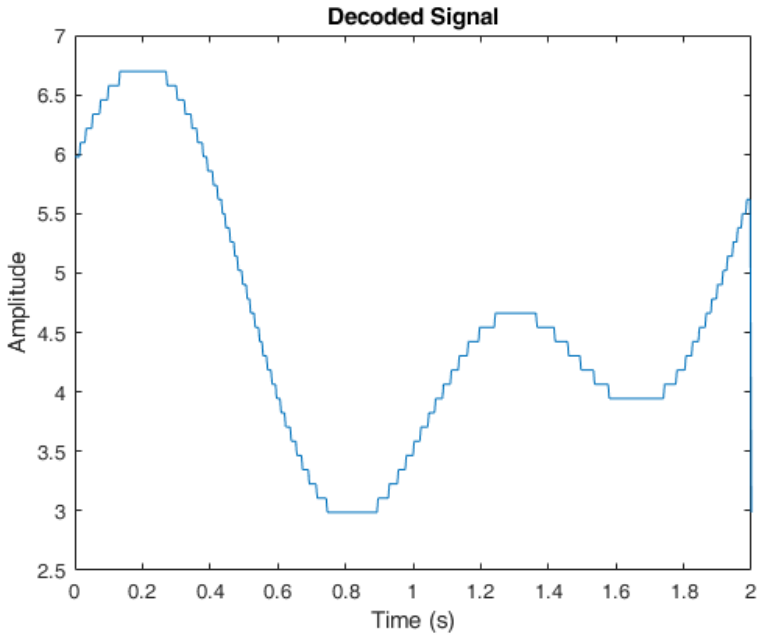
توجه شود که مقادیری که در energy_each_pulse ذخیره شده ممکن است به صورت اعشاری باشد برای همین باید نزدیک ترین عدد فرد به آن را پیدا کنیم.

در حلقه for ای که زده شده است طبق الگوریتم زیر نزدیک ترین عدد فرد پیدا شده و جایگذاری مقادیر اصلی میشود.

```
nearest_odd_number = energy_each_pulse(i);
is_odd = mod(nearest_odd_number,2) < 1;
nearest_odd_number = floor(nearest_odd_number);
nearest_odd_number(is_odd) = nearest_odd_number(is_odd)+1;
energy_each_pulse(i) = nearest_odd_number;
```

حال کافیست ببینیم هرکدام از مقادیر energy_each_pulse برابر با کدام از ستون اول آرایه دو بعدی است تا مقدار متناظر آن در ستون دوم را در ماتریسی ذخیره کنیم و آن را رسم کنیم.

بعد از مراحل بالا به شکل زیر میرسیم.



همانطور که میبینیم سیگنال decode شده در گیرنده, برابر با همان سیگنال کوانتیزه شده است که در بخش های قبل به دست آورده بودیم.

مدلاسیون دلتا

A)

Delta modulation (DM or Δ-modulation) is an analog-to-digital and digital-to-analog signal conversion technique used for transmission of voice information where quality is not of primary importance. DM is the simplest form of differential pulse-code modulation (DPCM) where the difference between successive samples is encoded into n-bit data streams.

Reference

PCM	DM
In PCM, feedback does not exist in transmitter or receiver.	While in DM, feedback exists in transmitter.
Per sample 4, 8, or 16 bits are used.	Here, only one bit is used per sample.
PCM requires highest transmitter bandwidth.	DM requires lowest transmitter bandwidth.
PCM is complex in terms of complexity of implementation.	While DM is simple in terms of complexity of implementation.
PCM has good signal to noise ratio.	While DM has poor signal to noise ratio.
PCM is costly.	DM is cheap.
PCM may be a technique wont to digitally represent sampled analog signals.	Digital to analog and analog to digital converter.
In PCM, signal requires encoder and decoder both sides.	In DM, signal can modulate and demodulate.
PM is mostly used in video telephony and audio telephony.	DM is mostly used in speeches as well as images.
Reference	

B)

x1: fs=20 &delta=0.4

Editor - C:\Users\asus\Desktop\CA4\Delta_Modulation.m

Delta_Modulation.m | DM.m

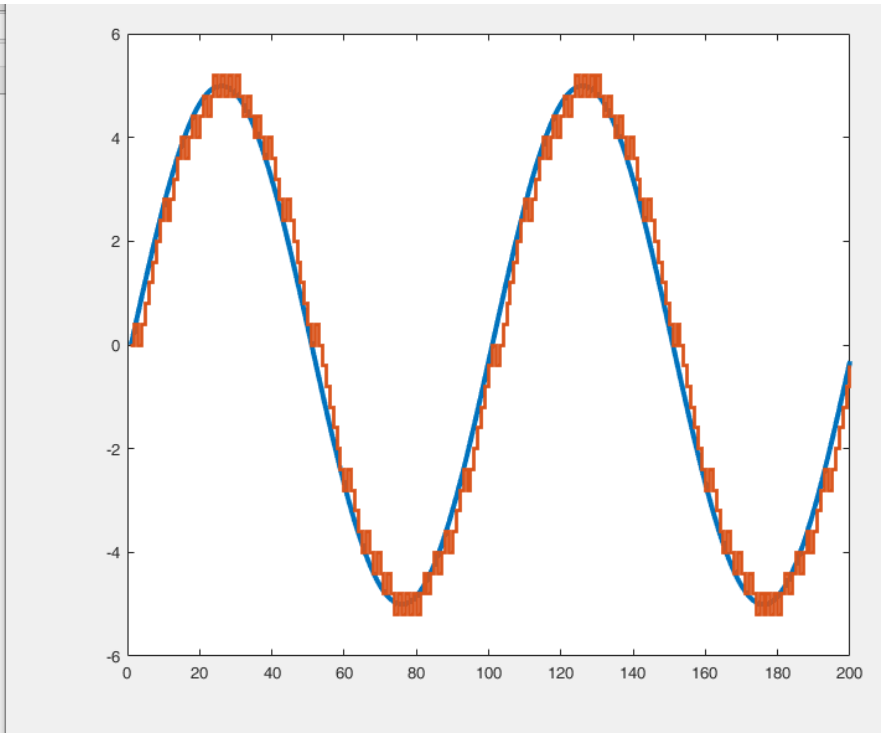
```
clear
clc
close all

fs = 20;
t = 0:1/fs:10-1/fs;

x1 = 5*sin(2*pi*t/5);
x2 = sinc(t);
x3 = exp(-t);
input = x1;

N = length(t);
delta = 0.4;
[code, x_dm] = DM(input, delta);

plot(input, "LineWidth", 3)
hold on
stairs(x_dm, "LineWidth", 2)
```



x2: fs=100 & delta=0.01

litor - C:\Users\asus\Desktop\CA4\Delta_Modulation.m

)elta_Modulation.m | DM.m

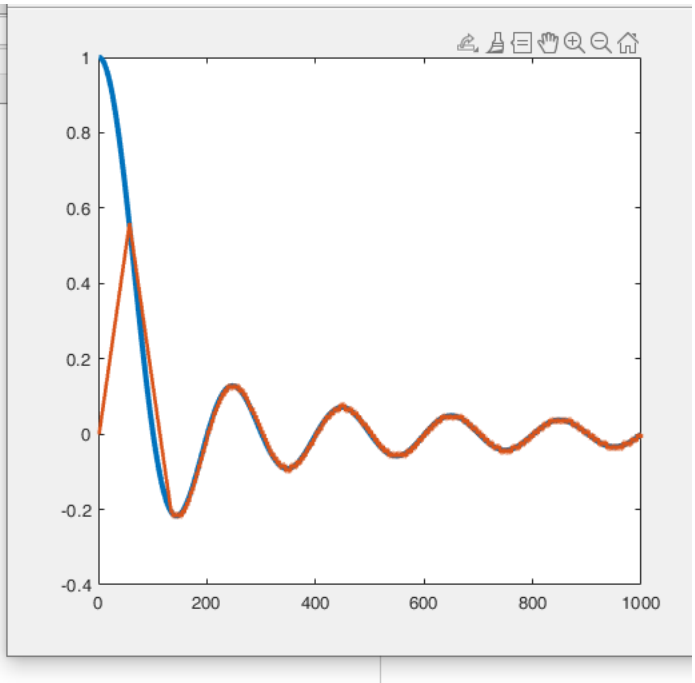
```
clear
clc
close all

fs = 100;
t = 0:1/fs:10-1/fs;

x1 = 5*sin(2*pi*t/5);
x2 = sinc(t);
x3 = exp(-t);
input = x2;

N = length(t);
delta = 0.01;
[code, x_dm] = DM(input, delta);

plot(input, "LineWidth", 3)
hold on
stairs(x_dm, "LineWidth", 2)
```



x3: fs=200 & delta=0.01

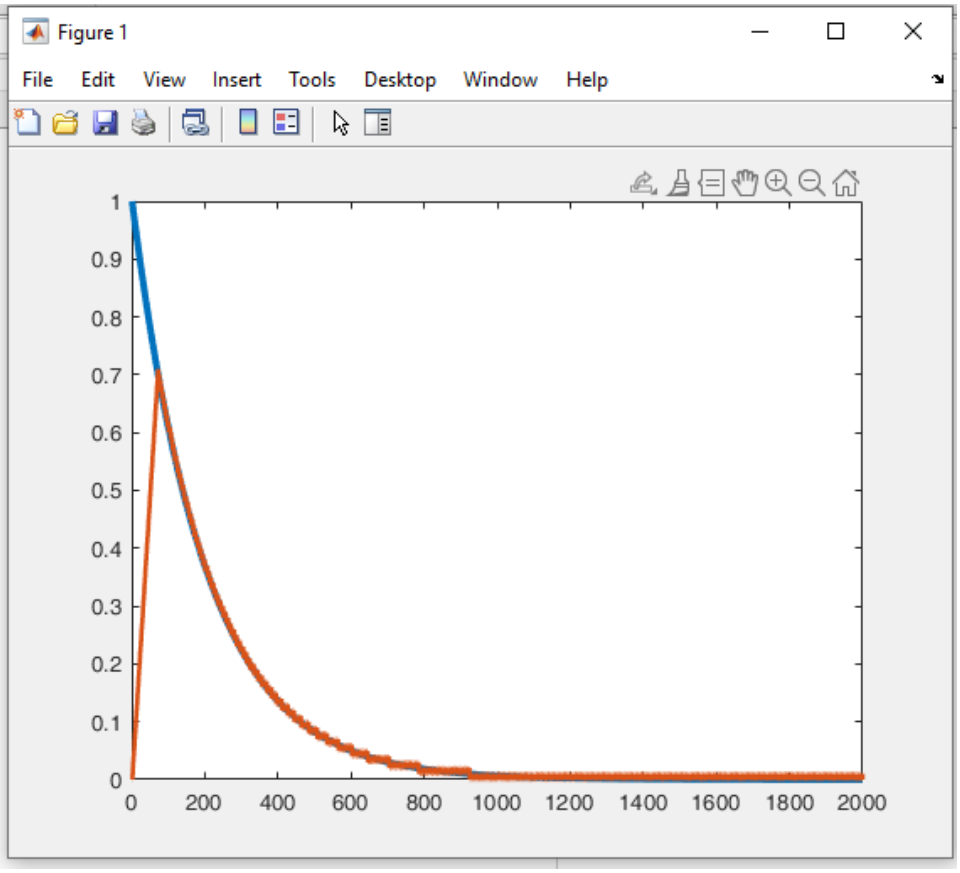
```
clear
clc
close all

fs = 200;
t = 0:1/fs:10-1/fs;

x1 = 5*sin(2*pi*t/5);
x2 = sinc(t);
x3 = exp(-t);
input = x3;

N = length(t);
delta = 0.01;
[code, x_dm] = DM(input, delta);

plot(input, "LineWidth", 3)
hold on
stairs(x_dm, "LineWidth", 2)
```



C)

مقدار ایده آلی برای fsو delta وجود ندارد بلکه بسته به سیگنال ورودی انتخاب میشود تا کمترین خطا را داشته باشد

E)

An advanced form of delta modulation is adaptive delta modulation. As we know, granular noise and slope overload distortion are the types of noise seen in delta modulation. To eliminate these kinds of noise, an adaptive delta modulation technique was developed. Slope error seen in the delta modulation technique is reduced in this process. Slope overload error and granular error are completely removed using this process. Quantized noise is removed in the demodulation process using an LPF (low pass filter).