

به نام خدا

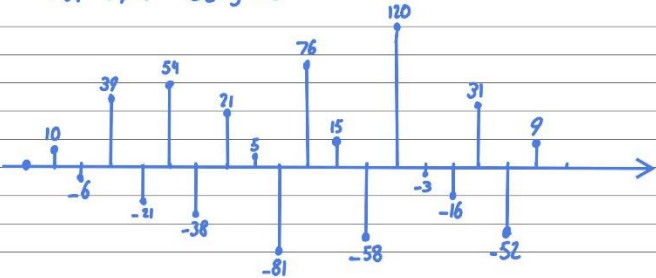
Signals & Systems

CA1

ایمان رسولی پرتو 810199425 , پارسا ستاری 810199436 , فردین عباسی 810199456

بخش 1

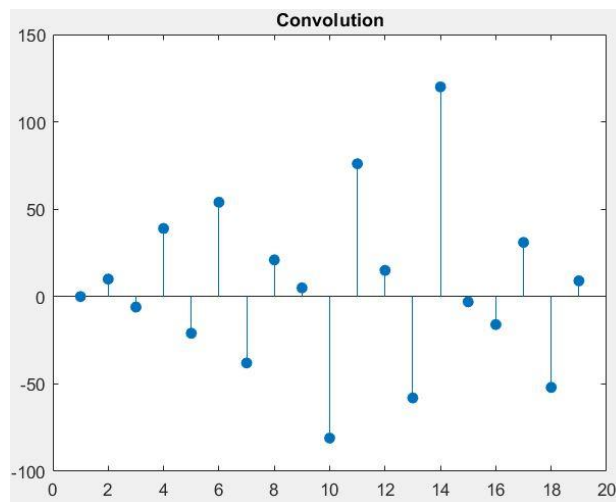
✚ برای محاسبه کانولوشن دو بردار در وهله اول نیاز به سائز دو بردار داریم ؛ دیدیم که حاصل کانولوشن به اندازه مجموع طول بردار طول داره ، در نتیجه در محاسبه کانولوشن n رو تا طول کانولوشن (conv_len) پیش میبریم (فرض کردیم بردارها از $n = 1$ شروع میشن و از conv_len به بعد هم صفر خواهد بود) ؛ یک متغیر کمکی برای محاسبه ترم $x(i) \times y(n - i)$ ها و جمع اونها در نظر میگیریم تا به عنوان حاصل کانولوشن هر نقطه لحاظ کنیم .
✚ محاسبه کانولوشن دو بردار به شکل دستی :

$$\begin{aligned}x[n] * y[n] &= s[n] = \sum_{i=1}^{n-1} x[i] \cdot y[n-i] \\s[1] &= 0 \\s[2] &= x[1] \cdot y[1] = 2 \times 5 = 10 \\s[3] &= x[1] \cdot y[2] + x[2] \cdot y[1] = 2 \times 2 + -2 \times 5 = -6 \\s[4] &= x[1] \cdot y[3] + x[2] \cdot y[2] + x[3] \cdot y[1] = 2 \times 4 + -2 \times 2 + 7 \times 5 = 39 \\s[5] &= x[1] \cdot y[4] + \dots + x[4] \cdot y[1] = 2 \times -6 + -2 \times 4 + 7 \times 2 + -3 \times 5 = -21 \\&\vdots \\s[6] &= 2 \times 5 + -2 \times -6 + 7 \times 4 + -3 \times 2 + 2 \times 5 = 54 \\&\vdots \\s[7:] &= [-38 \ 21 \ 5 \ -81 \ 76 \ 15 \ -58 \ 120 \ -3 \ -16 \ 31 \ -52 \ 9] \\i.e. : s[19] &= x[8] \cdot y[11] = 1 \times 9 = 9 \\for i > 19 \quad s[i] &= 0\end{aligned}$$


محاسبه کانولوشن دو بردار با تابع Convolution

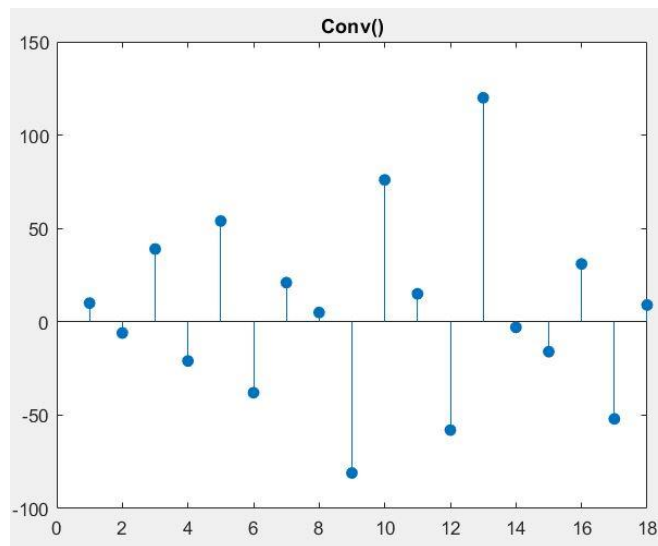
```
ans =  
  
Columns 1 through 17  
    0    10   -6   39  -21   54  -38   21    5  -81   76   15  -58  120   -3  -16   31  
  
Columns 18 through 19  
  -52     9
```

و رسم در نمودار : (رسم گسسته با استفاده از دستور stem)



محاسبه کانولوشن دو بردار با استفاده از دستور conv() :

```
ans =  
  
Columns 1 through 17  
    0    10   -6   39  -21   54  -38   21    5  -81   76   15  -58  120   -3  -16   31  
  
Columns 18 through 19  
  -52     9
```

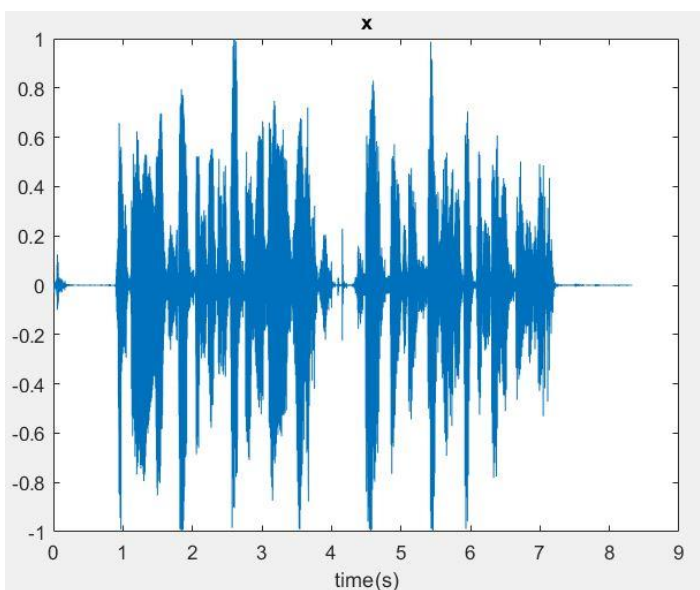


میبینیم که نمودار در حالت سوم بگونه ای هست که انگار نسبت به نمودار قبلی یک واحد به چپ منتقل شده ؛ دلیل این هست که نرم افزار متلب برای محاسبه کانولوشن بردارها رو از صفر شروع میکنه اما در نوشتن تابع Convolution فرض کردیم بردارها از یک شروع میشن (همونطور که متلب بردارها رو از اندیس 1 شروع میکنه) ؛ در نتیجه حاصل هر سه قسمت یکسان خواهد بود.

بخش 2

➤ به کمک دستور audioread فایل 'my_sound' رو در بردار x ذخیره میکنیم ، همچنین فرکانس نمونه برداری (F_s) که خروجی دیگه audioread هست رو در ماتریس F_s ذخیره میکنیم . برخی از ویس ها دو کاناله هستن که یک کانال اونها رو استخراج میکنیم و استفاده میکنیم (به کمک خواص ماتریس ها در متلب)

➤ برای رسم سیگنال ها برحسب زمان ها باید محور افق رو بر حسب زمان اسکیل کرد ؛ چون برحسب فرکانس نمونه برداری هستن . برای اینکار بردار t رو تعریف میکنیم که با استپ $1/F_s$ تا طول سیگنالی که قراره رسم بشه جلو میره . سپس این بردار رو از درایه دوم تا اخر در نظر میگیریم تا با ساینز بردارهایی که قراره برحسبش رسم بشن هماهنگ باشه ؛ این موارد رو در یک تابع (plt) پیاده سازی میکنیم ؛ رسم سیگنال x :



➤ بررسی LTI بودن سیستم :

$$y[n] = x[n] + \alpha x[n - n_0] \rightarrow y[n - n_1] = x[n - n_1] + \alpha x[n - n_1 - n_0]$$

$$z_1[n] = x[n - n_1] \rightarrow w[n] = x[n - n_1] + \alpha x[n - n_1 - n_0] \rightarrow w[n] = y[n]$$

$$\rightarrow \text{TI} \quad \checkmark$$

$$y_1[n] = x_1[n] + \alpha x_1[n - n_0], y_2[n] = x_2[n] + \alpha x_2[n - n_0]$$

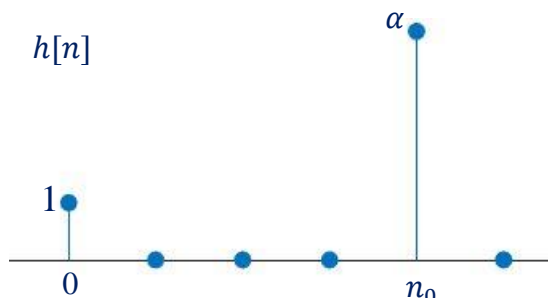
$$z_2[n] = \beta x_1[n] + \gamma x_2[n]$$

$$\rightarrow w[n] = \beta x_1[n] + \gamma x_2[n] + \alpha(\beta x_1[n - n_0] + \gamma x_2[n - n_0])$$

$$= \beta(x_1[n] + \alpha x_1[n - n_0]) + \gamma(x_2[n] + \alpha x_2[n - n_0]) = \beta y_1[n] + \gamma y_2[n]$$

→ LTI ✓

$$h[n] = \delta[n] + \alpha \delta[n - n_0]$$



✚ فرکانس نمونه برداری در بخش های قبل 44100 هرتز بدست اومد ؛ یعنی در هر ثانیه 44100 بار

نمونه برداری انجام میشه در نتیجه برای داشتن حدودا یک ثانیه تاخیر کافیه $x[n]$ رو به اندازه Fs

شیفت بدیم و در نتیجه : $n_0 = Fs$

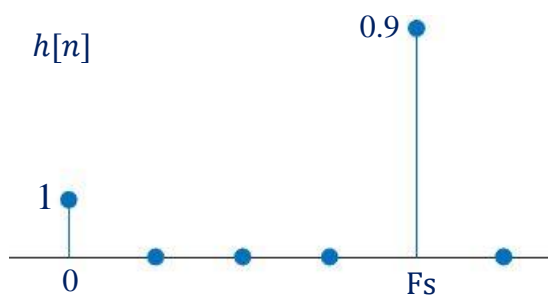
برای پیدا کردن α با توجه به رابطه انرژی سیگنال داریم :

$$E_{ecco} = \sum \alpha^2 x[n - n_0]^2, \quad E = \sum x[n]^2$$

$$\rightarrow \frac{E_{ecco}}{E} = 0.81 \rightarrow \alpha = 0.9$$

در نتیجه پاسخ ضربه برابر خواهد بود با :

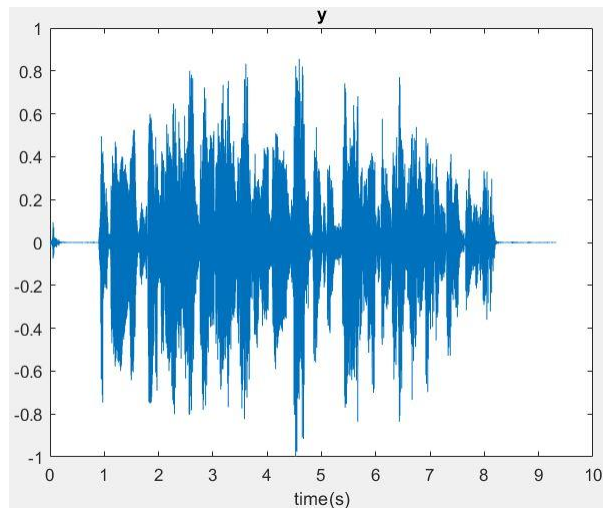
$$h[n] = \delta[n] + 0.9\delta[n - Fs]$$



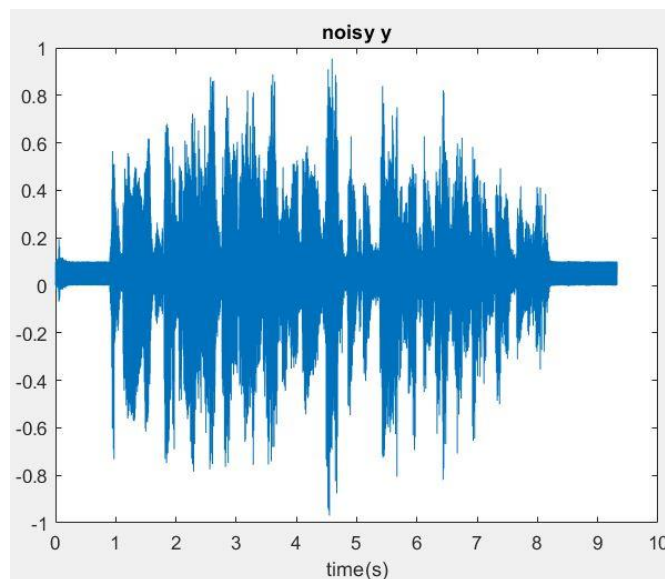
اما چون نرم افزار متلب اندیس بردارها رو از یک شروع میکنه پاسخ ضربه رو به این شکل نظر میگیریم :

$$h[1] = 1, h[Fs + 1] = 0.9$$

مقادیر بردار x بین $-1, 1$ بود ؛ اما مقادیر بردار y بیشتر از $-1, 1$ میشه ؛ برای جلوگیری از بروز خطا هنگام `audiowrite` مقادیر y رو اسکیل میکنیم تا در بازه $-1, 1$ قرار بگیره ؛ به این منظور تمامی درایه های y رو به ماکسیمم تقسیم میکنیم تا اسکیل انجام شه ؛ رسم سیگنال خروجی y :



✚ با توجه به اینکه سیگنال هامون رو در -1 order تا 1 اسکیل کردیم ، نویز مون هم باید در این محدوده باشه ؛ اما تابع `randn()` اینکار رو برای ما انجام نمیده و نیاز به اسکیل کردن داره (مشابه کاری که در بخش قبل برای y کردیم) چون این تابع با وجود تمرکز روی محدوده $(-1, 1)$ میتونه داده های خارج از این محدوده هم تولید بکنه ؛ برای خلاصه تر شدن و راحتی کار تابع `rand()` رو داریم که مقادیری بین $-1, 1$ ایجاد میکنه ؛ در نتیجه به کمک تابع `rand()` یک نویز ایجاد میکنیم و سیگنال y رو با 0.1 سیگنال نویز جمع میکنیم و حاصل به شکل زیر خواهد شد :



وقتی ورودی و خروجی رو داریم ، y تاخیر یافته x هست در نتیجه n_0 برابر تفاوت طول این دو بردار تقسیم به فرکانس نمونه برداری F_s هست (تا به ثانیه تبدیل بشه) ؛ در واقع برخی نقاطی که در $x[n]$ صفره در $x[n-n_0]$ مقدار داره و در y ظاهر میشه ؛ همچنین برای α خواهیم داشت :

$$y[n] = x[n] + \alpha x[n - n_0] \rightarrow \underbrace{y[n] - x[n]}_{z[n]} = \alpha x[n - n_0]$$

$$\rightarrow \alpha = \frac{\max(z)}{\max(x[n - n_0])} = \frac{\max(z)}{\max(x)}$$

*Note that: $\max(x) = \max(\text{shifted } x)$

بنابراین تابعی مینویسیم که دو سیگنال x , y رو بگیره و در خروجی α, n_0 مورد نظر رو برگردونه .

سیگنال x رو به عنوان ورودی و سیگنال noisy_y رو به عنوان خروجی به تابعی که در بخش قبل ایجاد کردیم میدیم و دو خروجی α, n_0 رو میگیریم ؛ میبینم که مقادیر بدست اومده با مقادیری که از تئوری حساب کردیم همخوانی مطلوبی داره :

```
a = 0.796774 & n0 = 1.000000
```