



دانشکده مهندسی کامپیوتر

انتقال داده ها

۹۸۵۲۲۰۹۴ نیکی نزاکتی

پروژه نهایی

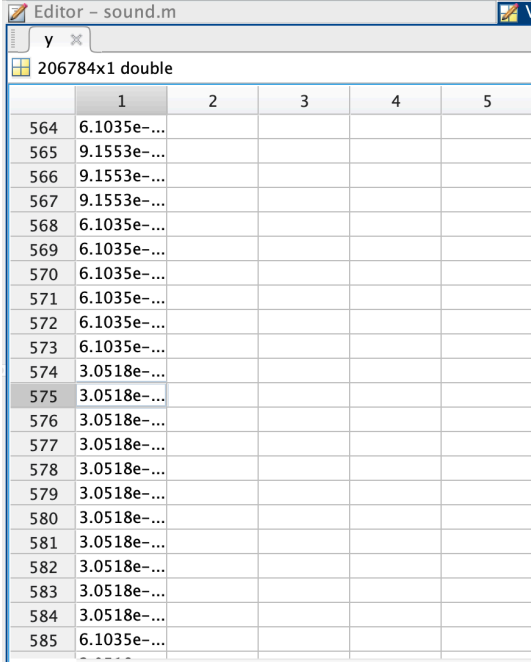
۱ پروژه صوت

۱.۱.

• خواندن صوت و به دست آوردن اندازه آن توسط کد زیر انجام شد که خروجی اندازه آن [206784,1] بود.

```
sound.m  x  +
1
2      %3.1
3      [y, Fs] = audioread('name.wav');
4
5      N = size(y); % sound size
6      disp(N)
7
```

• الفبای منبع در حالت ذکر شده چیست؟ همان طور که در تصویر مشخص است، y از اعداد اعشاری تا ۱۵ رقم اعشار تشکیل شده است.



	1	2	3	4	5
564	6.1035e-...				
565	9.1553e-...				
566	9.1553e-...				
567	9.1553e-...				
568	6.1035e-...				
569	6.1035e-...				
570	6.1035e-...				
571	6.1035e-...				
572	6.1035e-...				
573	6.1035e-...				
574	3.0518e-...				
575	3.0518e-...				
576	3.0518e-...				
577	3.0518e-...				
578	3.0518e-...				
579	3.0518e-...				
580	3.0518e-...				
581	3.0518e-...				
582	3.0518e-...				
583	3.0518e-...				
584	3.0518e-...				
585	6.1035e-...				

• سرعت تولید سمبل ۲۰۶۸۷۴ و زمان شنیدن صدا ۳.۴ است که به صورت زیر به دست می آید:

```
14      l=length(y) %speed
15
16      t=l/Fs      %time
17
```

Workspace	
Name ▲	Value
Fs	48000
I	206784
N	[206784,1]
player	1x1 audioplayer
t	4.3080
y	206784x1 double

۲.۱

- نحوه محاسبه سیگنال به نویز در متلب به این صورت است: $r = \text{snr}(x_i, y)$ که این نسبت را برای سیگنال x_i و نویز y محاسبه می کند.
- به صورت زیر به فایل صوتی نویز سفید اضافه می کنیم. زمانی که نسبت سیگنال به نویز 0 dB است یعنی نسبت نویز با خود سیگنال برابر است و به اندازه صدا نویز می شنویم. در حالت 10 dB سیگنال اصلی بیشتر قابل تشخیص است.

```

19
20 zeronois = awgn(y,0)
21 audiowrite('zero_noise.wav', zeronois, Fs);
22
23 tennois = awgn(y,10)
24 audiowrite('ten_noise.wav', tennois, Fs);

```

- برای بدست آوردن صدای اولیه از فیلتر دهی استفاده می کنیم. سنجیدن معیار عملکرد از مقایسه باند و همچنین اندازه دو سیگنال و محاسبات بین آن ها به صورت زیر به دست می آید:

```

Editor - /Users/nikinezakati/Desktop/DC/NikiNezakati-98522094-Project/sound.m
sound.m
26
27 b = fir1(800, .2);
28 zero_denoise = filter(b, 1, zeronois);
29 audiowrite('zero_denoise.wav', zero_denoise, Fs);
30
31 b = fir1(800, .2);
32 ten_denoise = filter(b, 1, tennois);
33 audiowrite('ten_denoise.wav', ten_denoise, Fs);
34
35 [x, fsx] = audioread('zero.wav'); % load an audio file
36 [y, fsy] = audioread('zero_denoise.wav'); % load an audio file
37
38 x = x(:, 1); % get the first channel
39 y = y(:, 1); % get the first channel
40 nx = length(x); % signal length
41 ny = length(y); % signal length
42 yPad = y;
43 yCrop = y;
44 xPad = x;
45 xCrop = x;
46 if nx > ny
47     yPad(nx) = 0;
48     xCrop = xCrop(1:ny);
49 elseif nx < ny
50     xPad(ny) = 0;
51     yCrop = yCrop(1:nx);
52 end
53 c1 = corrcoef(xPad, yPad);
54 c2 = corrcoef(xCrop, yCrop);
55
56
57 [z, fsz] = audioread('ten_denoise.wav'); % load an audio file
58
59 x = x(:, 1); % get the first channel
60 z = z(:, 1); % get the first channel
61 nx = length(x); % signal length
62 nz = length(z); % signal length

```

```

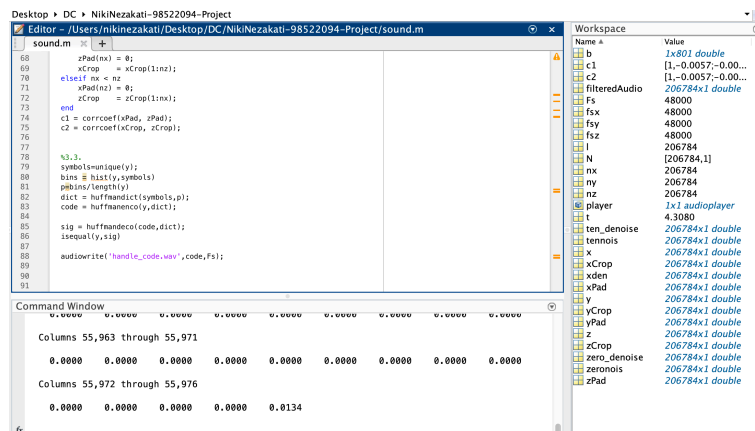
Editor - /Users/nikinezakati/Desktop/BC/NikiNezakati-98522094-Project/
sound.m
41 ny = length(y); % signal length
42 yPad = y;
43 yCrop = y;
44 xPad = x;
45 xCrop = x;
46 if nx > ny
47     yPad(nx) = 0;
48     xCrop = xCrop(1:ny);
49 elseif nx < ny
50     xPad(ny) = 0;
51     yCrop = yCrop(1:nx);
52 end
53 c1 = corrcoeff(xPad, yPad);
54 c2 = corrcoeff(xCrop, yCrop);
55
56 [z, fsz] = audioread('ten_denoise.wav'); % load an audio file
57
58
59 x = x(:, 1); % get the first channel
60 z = z(:, 1); % get the first channel
61 nx = length(x); % signal length
62 nz = length(z); % signal length
63 zPad = z;
64 zCrop = z;
65 xPad = x;
66 xCrop = x;
67 if nx > nz
68     zPad(nx) = 0;
69     xCrop = xCrop(1:nz);
70 elseif nx < nz
71     xPad(nz) = 0;
72     zCrop = zCrop(1:nx);
73 end
74 c1 = corrcoeff(xPad, zPad);
75 c2 = corrcoeff(xCrop, zCrop);
76
77

```

Workspace	
Name	Value
b	1x801 double
c1	[1,-0.0057;-0.00...
c2	[1,-0.0057;-0.00...
filteredAudio	206784x1 double
Fs	48000
fsx	48000
fsy	48000
fsz	48000
l	206784
N	[206784,1]
nx	206784
ny	206784
nz	206784
player	1x1 audioplayer
t	4.3080
ten_denoise	206784x1 double
tennois	206784x1 double
x	206784x1 double
xCrop	206784x1 double
xden	206784x1 double
xPad	206784x1 double
y	206784x1 double
yCrop	206784x1 double
yPad	206784x1 double
z	206784x1 double
zCrop	206784x1 double
zero_denoise	206784x1 double
zeronois	206784x1 double
zPad	206784x1 double

۳.۱

- ابتدا مقادیر unique موجود در y را به دست می آوریم، سپس با استفاده از هیستوگرام تعداد هر کدام را به دست می آوریم و در نهایت احتمال هر کدام را محاسبه می کنیم. در نهایت کد هافمن را تست می کنیم و با توجه به کد زیر به درستی آن پی می بریم.
- محاسبه کنید که برای انتقال این فایل ر یک لینک مخابراتی با سرعت 64kbits ه مقدار زمان لازم است؟ ابتدا کد هافمن به دست آمده را به فایل تبدیل می کنیم. حجم این فایل 634kB شد و اگر این مقدار را تقسیم بر 64kbits کنیم، عدد s 80 به دست می آید.



```

68 zPad(nz) = 0;
69 xCrop = xCrop(1:nz);
70 elseif nx < nz
71 xPad(nz) = 0;
72 xCrop = xCrop(1:nz);
73 end
74 c1 = corrcoeff(xPad, xPad);
75 c2 = corrcoeff(xCrop, xCrop);
76
77
78 %3.3.
79 symbols=unique(y);
80 bins = hist(y,symbols)
81 pkins=length(y)
82 dict = huffmandict(symbols,p);
83 code = huffmanenco(y,dict);
84
85 sig = huffmandeco(code,dict);
86 isequal(y,sig)
87
88 audiowrite('handle_code.wav',code,Fs);
89
90
91

```

- ما برای انتقال اطلاعات معمولاً نیازمند فشرده سازی بیشتری می باشیم. در نتیجه مجبور هستیم به سراغ کد کننده های با اتلاف برویم.