به نام خدا

پروژه اول سیگنال ها و سیستم ها

استاد درس : دکتر سعید اخوان

نرگس غلامی

شماره دانشجویی : ۸۱۰۱۹۸۴۴۷

سوال اول:

در بخش اول از ما خواسته شده است که تابع کانوولوشن را پیاده سازی بکنیم:

```
function y = myConv(x, h)

    l1 = length(x);
    l2 = length(h);
    N = l1 + l2 -1;

    x = [x, zeros(1,N-l1)];
    h = [h, zeros(1,N-l2)];


    if nargout
        y = zeros(1, N);

        for n = 1 : N
            for m = 1 : n
                y(n) = y(n) + x(m)*h(n - m + 1);
            end
        end
    else
        t = zeros(1, N);

        for n = 1 : N
            for m = 1 : n
                t(n) = t(n) + x(m)*h(n - m + 1);
            end
        end
        plot(t)
    end
end
```

توضیحات در مورد تابع پیاده سازی شده:

ابتدا طول سیگنال حاصل کانوولوشن را مشخص می‌کنیم و هر دو سیگنال ورودی را به اندازه سیگنال خروجی اکستند می‌کنیم و جای آن صفر قرار می‌دهیم.

با استفاده از nargout می توانیم متوجه شویم تابع ما خروجی دارد یا خیر. اگر خروجی داشت، سیگنال حاصل کانوولوشن را باز می‌گردانیم و اگر نداشت همانجا پلات آن را رسم می‌نماییم.

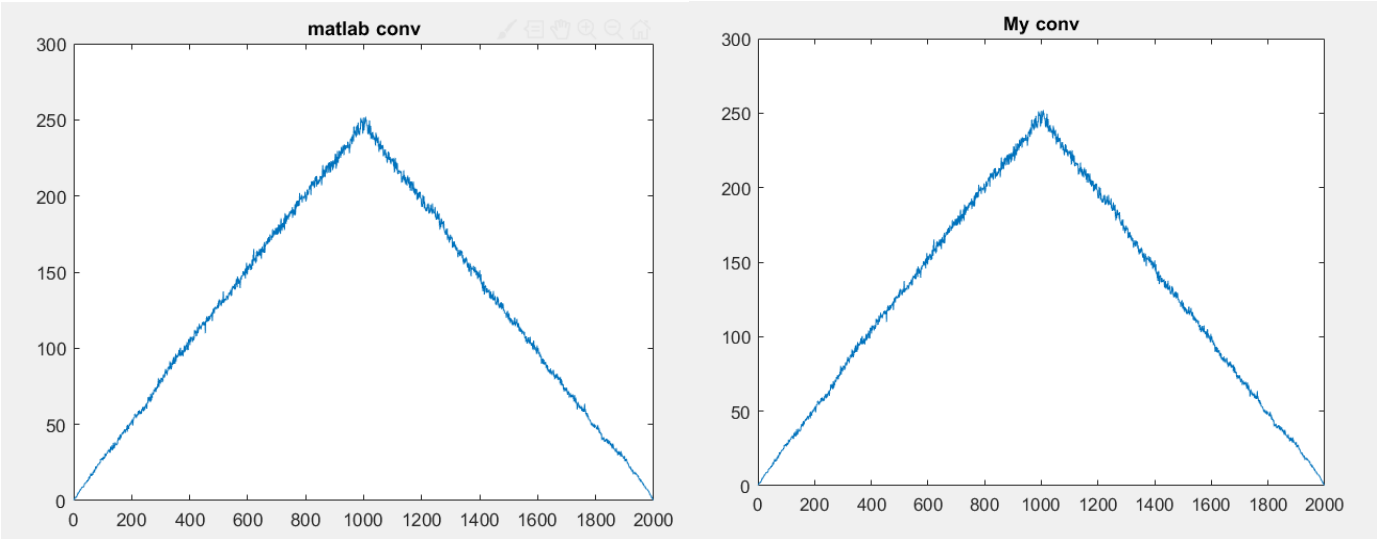
منطق تابع کانوولوشن نیز مانند عبارت نوشته شده در صورت سوال است.

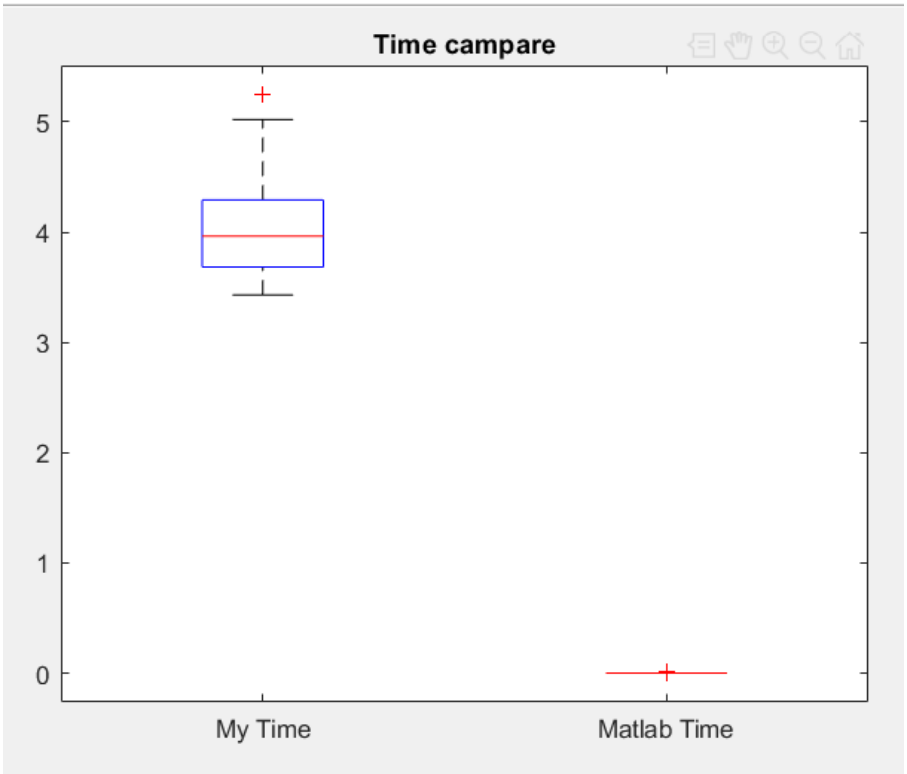
در قسمت بعد از ما خواسته شده که زمان این تابع را برای ورودی نمونه به طول ۱۰۰۰۰ اندازه بگیریم و صد بار این عمل را تکرار کنیم. همین کار را با تابع کانوولوشن متلب تکرار می‌کنیم و هر دو را با تابع باکس پلات کشیده و مقایسه شان را انجام می‌دهیم. من برای این که محاسبه ورودی به طول ۱۰۰۰۰ زمان زیادی می‌برد خروجی‌ها را با ورودی به طول ۱۰۰۰ نشان دادم.

توضیحات کد بعد از تکه کد داده شده است:

```
time1 = zeros(1, 100);
time2 = zeros(1, 100);
x = rand(1,1000)
h = rand(1,1000);
for i = 1:100
    tic
    z1 = myConv(x, h);
    time1(i) = toc;
    tic
    z2 = conv(x,h);
    time2(i) = toc;
end
plot(z1)
plot(z2)
g = [zeros(1, 100); ones(1, 100)];
d = [time1; time2]
b = boxplot(d(:), g(:),'Labels',{'Time one','Time two'})
title('Time campare')
```

ابتدا دو آرایه برای تایم های تابع کانوولوشن خود و یکی برای تابع کانوولوشن متلب تعریف می‌نماییم. سپس ورودی های نمونه را دریافت می‌کنیم و ۱۰۰ بار عملیات کانوولوشن را هم با تابع خود و هم با تابع متلب انجام می‌دهیم و زمان های آن ها را جداگانه در آرایه های مخصوص خود ذخیره می‌کنیم. برای اندازه گرفتن زمان از تابع تیک تاک استفاده می‌کنیم. حال پلات هر دو را رسم کرده و بعد باکس پلات این دو را کنار همدیگر مشاهده می نماییم.



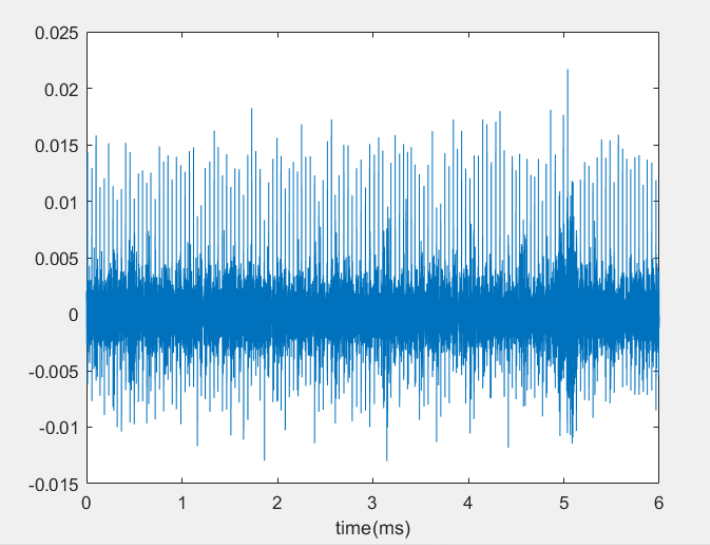


همان‌طور که مشاهده می‌شود زمان صرف شده با تابع myConv گستردگی زیادی دارد و مقدار زمان بیشتری طول می‌کشد ولی از آن طرف تابع کانوولوشن متلب زمان کمتری طول می‌کشد و گستردگی کمتری نیز دارد.

سوال دوم:

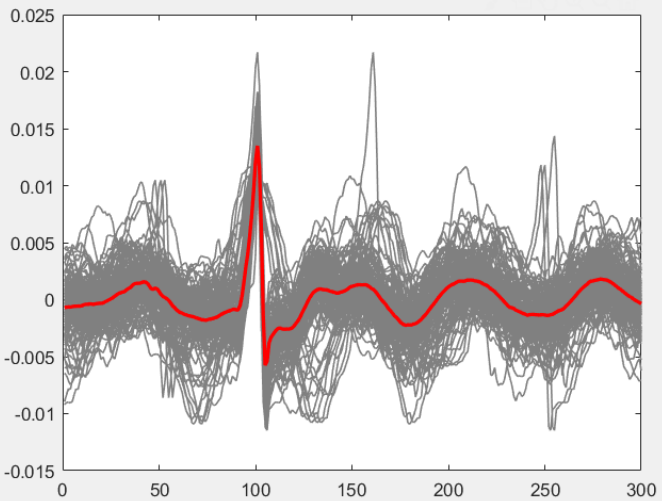
```
f = load('ecg.mat');
v = f.ecg;
b = reshape(v, [1,60001]);
t = linspace(0, 6, 60001);
plot(t,b)
xlabel("time(ms)");
b(b < 0.01) = 0;
[val, idx] = findpeaks(b);
z = zeros(length(idx), 300);
for i = 1:length(idx)
    z(i, :) = v(idx(i)-100: idx(i)+199);
end
avg = mean(z);

plot(z', 'Color', [0.5, 0.5,0.5,0.5], 'linewidth', 1)
hold on
plot(avg, 'Color', 'Red')
hold off
```

ابتدا فایل ecg در متلب لود می‌شود و سپس پلات این داده را برحسب زمان رسم می‌نماییم. خروجی مانند شکل زیر است.



سپس برای تشکیل آرایه‌ای از پیک ها، آن هایی را که کوچکتر از یک صدم هستند را صفر میکنیم و ۱۰۰ نمونه از قبل آن و ۲۰۰ نمونه از بعد آن را می‌گیریم و در یک آرایه ثبت می‌نماییم. حال پلات آن را در کنار پلات میانگین آن رسم می کنیم.

خروجی به صورت زیر روبرو خواهد بود:



بخش سوم:

```
[x, Fs] = audioread('matlab.mp3')
sound(x, Fs);

y = zeros(length(x), 1);
a = 0.5;
ts = 1/Fs;
n0 = 0.5/ts
for n = 1:length(x)
    if(n - n0 > 0)
        y(n) = x(n) + a*x(n-n0);
    else
        y(n) = x(n);
    end
end

sound(y, Fs);
h = [1 zeros(1,n0-1) a]
y2 = conv(x, h)
sound(y2, Fs)
plot(y)
plot(y2)
y3 = conv(flip(x), y)
plot(y3)

y4 = y3;
y4 (y4 < 140) = 0;
[val, idx] = findpeaks(y4);
[x,y] = max(val);
[x1, y1] = min(val);
n0 =  idx(y1)-idx(y)
```

ابتدا صوت مورد نظر در متلب لود می شود و بعد تابع گفته شده روی تابع ایکس اعمال می شود و صدا اکو می‌شود.

N0 برابر با نصف فرکانس می‌باشد.

اثبات این که این سیسیتم LTI می باشد:

① $y[n] = x[n] + \alpha x[n-n0]$

$z[n] = x[n-n1]$ $\qquad w[n] = z[n] + \alpha z[n-n0]$

$y[n-n1] = x[n-n1] + \alpha x[n-n1-n0]$

$w[n] = x[n-n1] + \alpha x[n-n0-n1]$ ⟹ تغییر ناپذیر با زمان

② $y_1[n] = x_1[n] + \alpha x_1[n-n0]$

$y_2[n] = x_2[n] + \alpha x_2[n-n0]$

$z[n] = \alpha_1 x_1[n] + \beta x_2[n]$

$w[n] = z[n] + \alpha z[n-n0] = \alpha_1 x_1[n] + \beta x_2[n] + \alpha\alpha_1 x_1[n-n0] + \alpha\beta x_2[n-n0]$

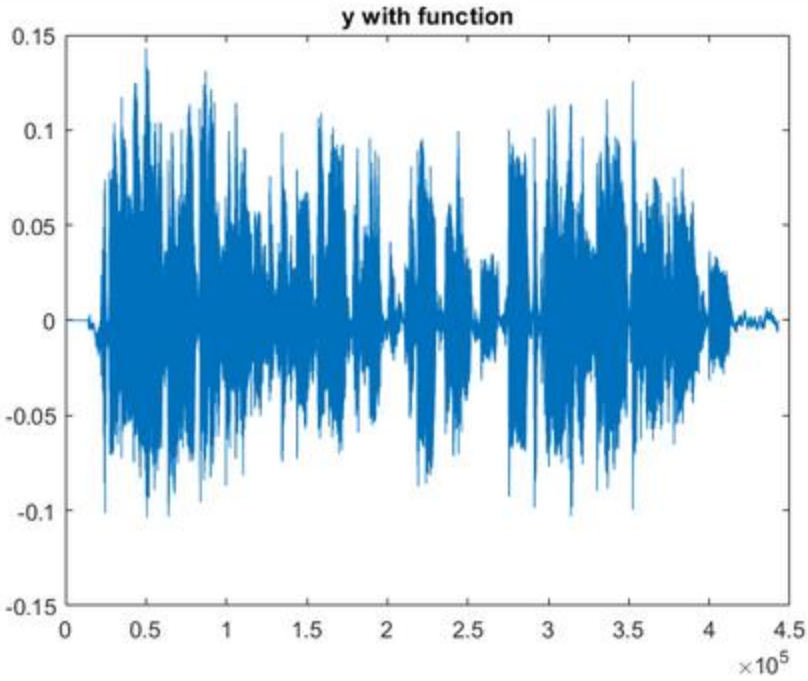
$\alpha_1 y_1[n] + \beta y_2[n] = \alpha_1 x_1[n] + \alpha_1\alpha x_1[n-n0] + \beta x_2[n] + \alpha\beta x_2[n-n0]$

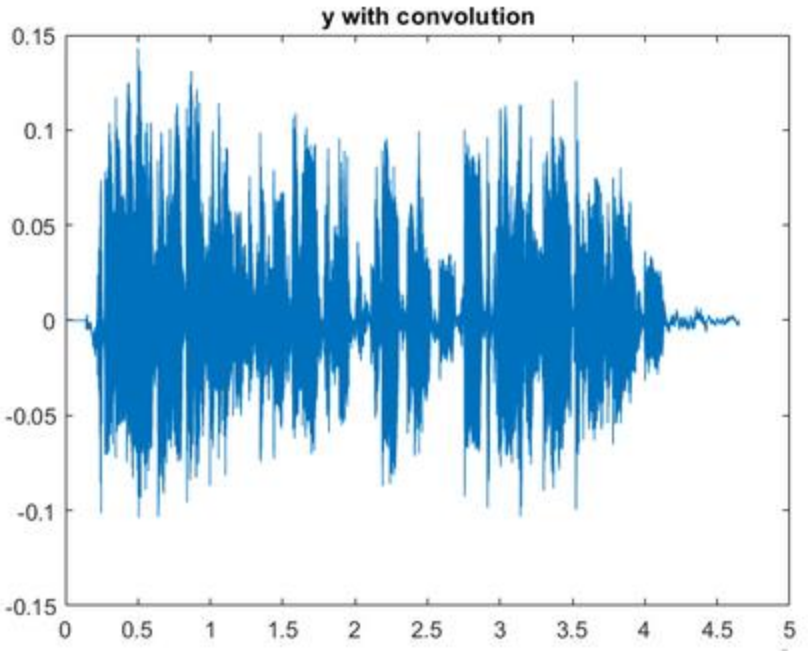
خطی می باشد

پاسخ ضربه این سیستم را به این صورت محاسبه می‌کنیم که فرض می‌کنیم تابع ضربه ورودی تابع است و پاسخ ضربه را بدست می‌آوریم،

سپس کانوالت آن را با سیگنال اولیه محاسبه می‌کنیم و صدا را پخش می‌کنیم. مشاهده می‌شود که صدا اکو شده است.

و شکل سیگنال ساخته شده اول با کانوالت آن یکی می‌باشد.


y with function
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
×10^5


y with convolution
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

برای پاسخ به قسمت آخر از تکه کد زیر استفاده می‌کنیم.

```
y3 = conv(flip(x), y)
plot(y3)
```

یعنی کانوولوشن سیگنال معکوس ایکس را با ایگرگ بدست می‌آوریم و پلات آن را رسم می‌نماییم. اختلاف محل پیک‌ها مقدار n0 را به ما خواهد داد.

در حقیقت با این دستور ما دنبال نمودار x در y می‌گردیم و جاهایی که مانند x را پیدا می‌کند، پیک می‌زند. مشاهده می‌شود که مقدار n0 را بدست آورده است.

```
y4 = y3;
y4 (y4 < 140) = 0;
[val, idx] = findpeaks(y4);
[x,y] = max(val);
[x1, y1] = min(val);
n0 =  idx(y1)-idx(y)
```