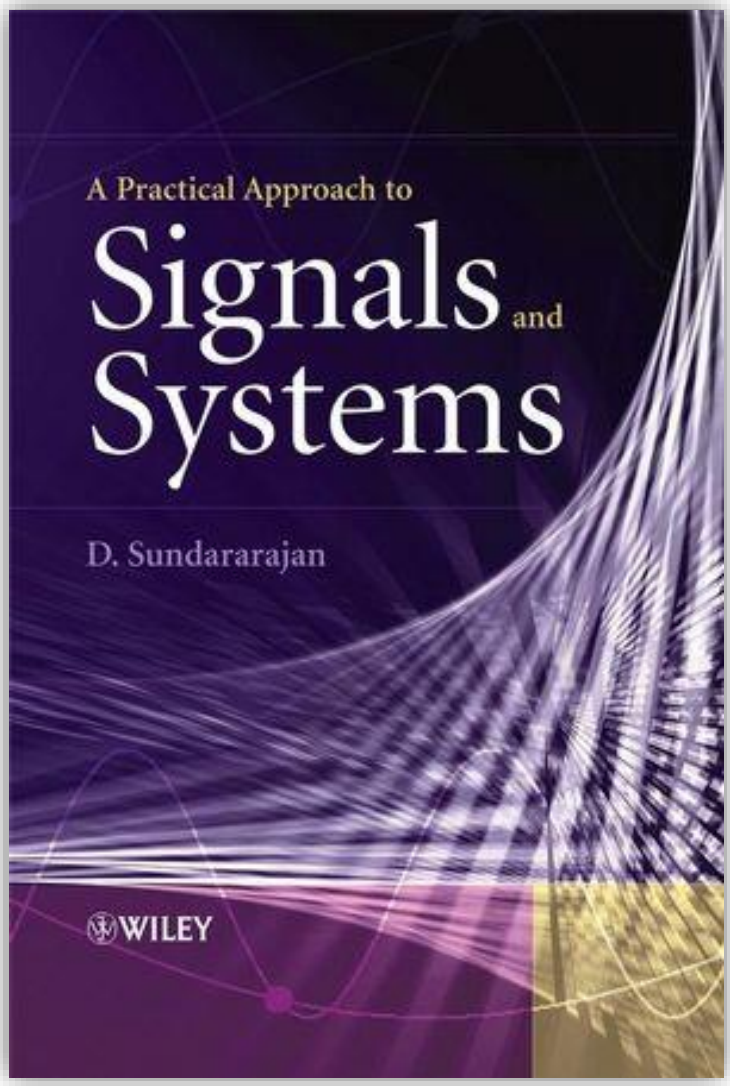


پروژه نهایی درس سیگنال‌ها و سیستم‌ها

پردازش صوت با استفاده از متلب

چکیده

با توجه به درسی که در طول ترم آموختیم, حال نوبت استفاده عملی از آن رسیده و در این پروژه سعی شده که پردازش صوت مورد بررسی قرار بگیرد و بتوانیم عملکرد فیلتر کردن سیگنال‌ها را نیز بهتر دریابیم.

به نام خدا

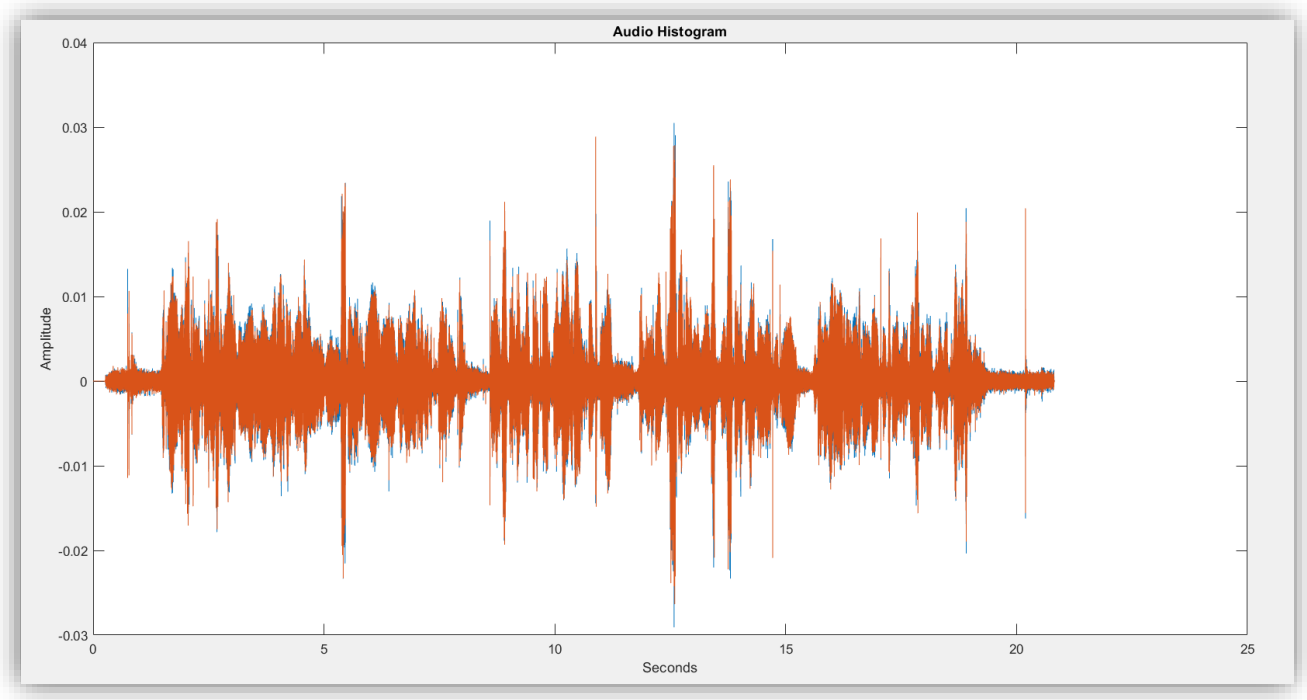
ملیکا احمدی رنجبر با شماره داشنجویی: ۹۷۵۲۱۰۳۶

فهرست

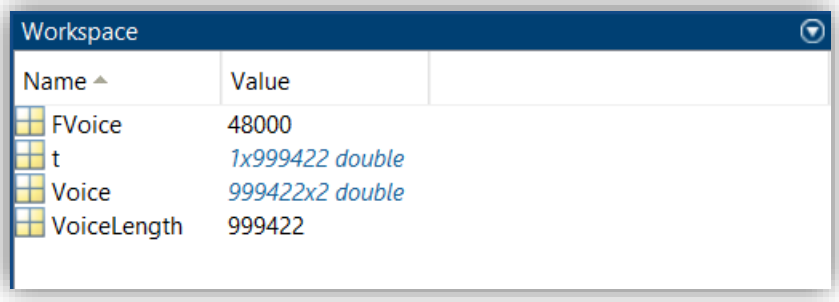
ضمن تشکر از تمامی زحمات کشیده شده توسط استاد و دستیار آموزشی, جناب آقای جلیلوند, همچنین آرزوی سلامتی برای ایشان توجه شما را به مستندی که از مراحل انجام پروژه و همچنین توضیحات لازم تهیه نموده, جلب میکنم.

این پروژه را شامل 3 بخش دانسته که به ترتیب در صورت سوال نیز قابل مشاهده است.

- در ابتدا تنها لازم بود که فایل صوتی مورد نظر را در پوشه پروژه قرار داده (همان پوشه که فایل اسکریپت در آن قرار دارد) و با استفاده از دستوراتی که زده شده, آن را بخوانیم و بعد آن را اجرا کنیم. سپس توسط دستوارتی ساده و همیشگی که در طول ترم بسیار با آن آشنا شدیم, هیستوگرام فایل صوتی خود را نشان دهیم(Plot) . این سوال در اسکریپتی با نام Question1 انجام شده است و هیستوگرام فایل صوتی همانطور که در برنامه اجرا شده نیز قالب مشاهده است به صورت زیر است:



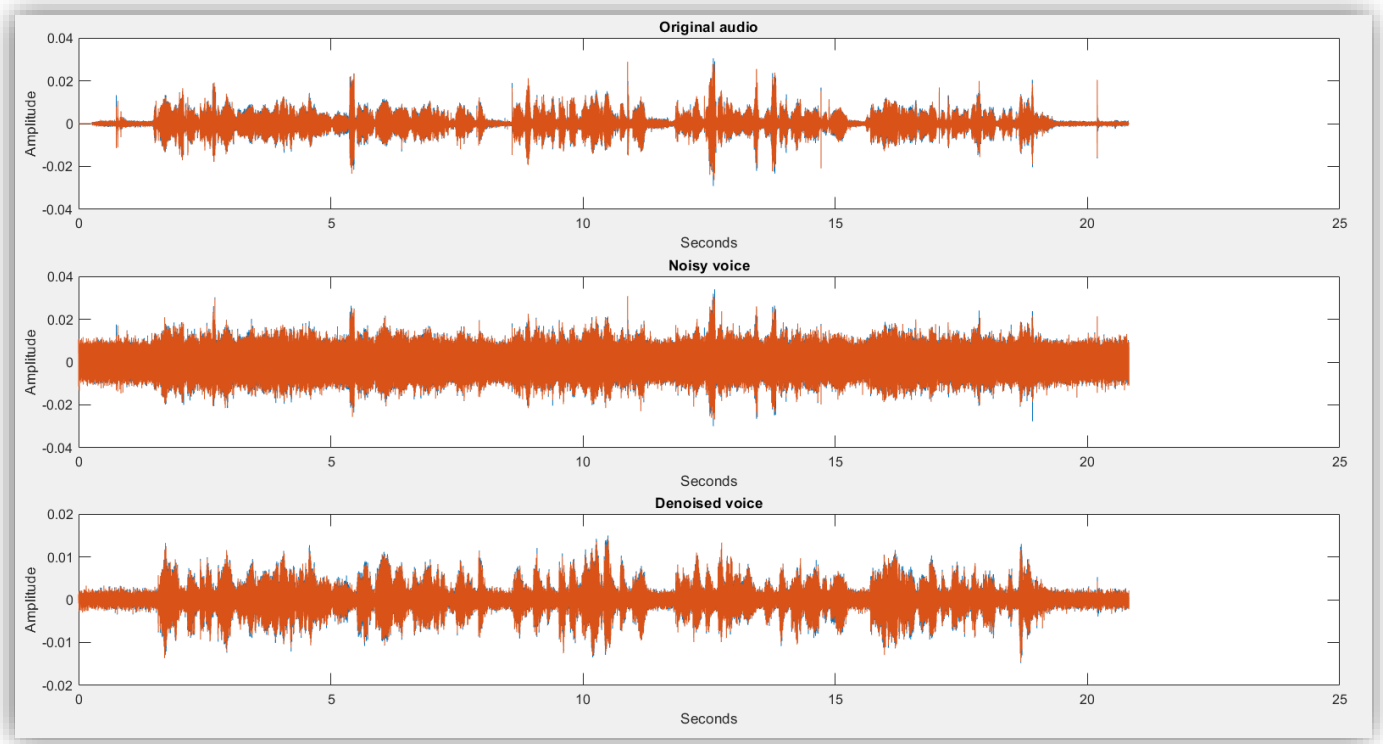
در رابطه با چگونگی ذخیره شدن سیگنال صوتی با توجه به برداشتی که از قسمت های مختلف کار داشته‌ام و تحقیقات لازم, متوجه شدم که به شکل آرایه‌ای دو بعدی ذخیره می‌شوند و در پایین نیز با توجه به اطلاعات داده شده در برنامه متوجه میشویم:



در این قسمت Voice همان سیگنال صوتی ما است و آرایه دو بعدی است که طول آن نیز در بالا مشخص شده.

FVoice نیز Sample rate می‌باشد. برای اطلاعات گرفتن درباره‌ی خود فایل نیز می‌توان از دستور audioinfo('Sample.mp4a') استفاده کرد.

❖ در این قسمت فایل خود را با اضافه کردن سیگنال هایی رندوم به صورت نویزی درآوردیم. همانطور که در اجرای پروژه قابل مشاهده است, هر بخش از کار به طور کامل نشان داده شده است و هیستوگرام تمامی مراحل قابل رویت است. این سوال در اسکریپتی با نام Question2 قرار دارد و شامل ۳ هیستوگرام می‌باشد در ابتدا هیستوگرام فایل اصلی را نشان داده‌ایم , به منظور مقایسه سیگنال فیلتر شده با آن و همچنین سیگنال نویزی را نیز نمایش می‌دهم. در نهایت همچین نمایشی را پس از اجرا شدن مرحله به مرحله فایل‌های صوتی خواهیم داشت:



طبق شکل نشان داده‌شده در عکس بالا می‌بینیم که پس از افزودن نویز دامنه فایل ما تغییر می‌کند و شنیدن صدای اصلی دشوار می‌شود.

حال نوبت به آن می‌رسد که نویزهای ایجاد شده را حذف کنیم و با سیگنال صوتی اصلی که در ابتدا نشان داده شده مقایسه کنیم. که همانطور که می‌بینیم تلاش شده‌است تا حد امکان نویزها حذف گردد. در مقایسه شکل سیگنال اول و آخر می‌توان دریافت که کمی شباهت بیشتری به فایل اصلی دارد.

حذف کردن نویز را با اعمال فیلتر و مشخص کردن فرکانس قطع و پارامترهای دیگری مانند انتخاب نوع سیگنال (برای مثال پایین گذر یا بالا گذار بودن آن یا انواع دیگر) انجام می‌دهیم.

❖ در سوال ۳ باید عمل اکو گذاری بر روی فایل صوتی مورد نظرمان انجام دهیم. برای این کار همانطور که در اسکریپت Question3 نیز قابل رویت است از یک فانکشنی به اسم EchoGenerator که دارای 4 ورودی است که هر کدام در این گزارش به طور کامل توضیح داده شده استفاده می‌کنیم.

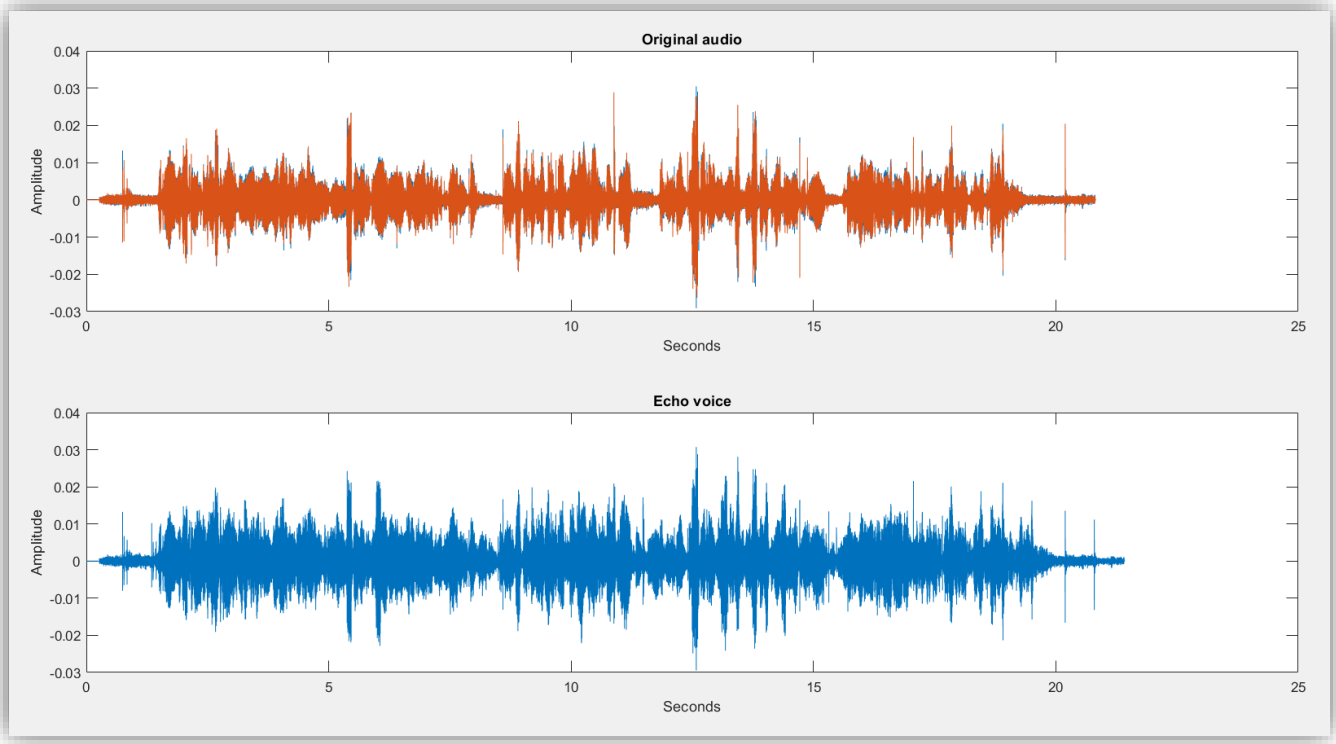
ورودی‌های این فانکشن در ابتدا سیگنال صوت اصلی‌مان می‌باشد. ورودی دوم آن Sample rate است که هر دوی این ورودی‌ها از روش سوال یک است. بعد از این دو ورودی پارامترهای Delay و Amplitude را داریم که طبق بررسی‌هایی که انجام دادم و تعداد زیادی تست با اعداد مختلف به این نتیجه رسیدم که Delay زمانی است که پس از اجرا شدن صوت اصلی اکوها اجرا می‌شود و به طول می‌انجامد. برای مثال اگر Delay را برابر ۰,۱ قرار دهیم مقدار اکویی که با برمی‌گردد کمتر است (از لحاظ بازگشت صدا به نظر می‌رسد مدت بسیار کمی اکو می‌شود) ولی اگر همانطور که در فایل اجرایی است ۰,۴ باشد اکوها را چون مدت زمان بیشتری تاخیر دارد واضح‌تر می‌شنویم.

ورودی بعدی نیز Amplitude می‌باشد که دامنه‌ی اکوها را اضافه می‌کند که عملا حکم شدت صدای حاصل از اکو می‌باشد. برای مثال اگر این مقدار را برابر ۰,۸ بگذاریم صدای حاصل از اکو را قوی‌تر و **بلندتر** می‌شنویم در مقایسه با ورودی ۰,۲ و این نکات بسیار جالبی است.

می‌توان تمام این مراحل اکو گذاری را به طور واقعی در طبیعت نیز مشاهده کرد مثلا هنگامی که در کوه (دور تا دور در محاصره کوه باشد) صحبت کنیم متوجه مفاهیم این ورودی‌ها می‌شویم (از نظر من اگر تعداد کوه‌ها زیاد باشد و محیط بیشتر در محاصره کوه باشد گویی Delay زیاد شده است و حتی Amplitude).

نحوه ایجاد سیگنال اکو دار نیز به این شکل است که یک آرایه خالی دوبعدی در ابتدا تشکیل می‌دهیم (از لحاظ اندازه باید به Delay توجه داشت چراکه هرچه Delay بیشتر شود اندازه آرایه موردنظر افزایش می‌یابد) و بعد از آن آرایه جدید را با تکرار خانه‌های قبلی آرایه با توجه به مقدار ورودی‌های Delay و Amplitude مقداردهی می‌کنیم.

خروجی این اسکریپت شامل فایل صوتی اکو دار است به همراه هیستوگرام فایل صوتی اصلی و تغییر یافته (برای مقایسه راحت تر تغییرات) که به صورت زیر است:



منابع:

- ✓ https://www.mathworks.com/help/matlab/ref/audioread.html

برای آشنایی اولیه با دستورات مربوط به فایل صوتی از این منبع استفاده شده‌است.

- ✓ https://www.mathworks.com/matlabcentral/answers/68771-adding-noise-to-a-wav-file

برای آشنایی با اضافه کردن نویز

- ✓ https://www.mathworks.com/matlabcentral/answers/357022-can-you-help-remove-the-noise-from-this-audio-file

برای آشنایی با فیلتر کردن (پس از افزودن نویز به سیگنال)

- ✓ https://www.mathworks.com/matlabcentral/answers/479020-i-meet-some-problem-in-my-coursera-homework-echo-generator

برای آشنایی با اضافه کردن اکو به فایل صوتی

با تشکر فراوان از توجه شما و همکاری شما با ما در طول ترم