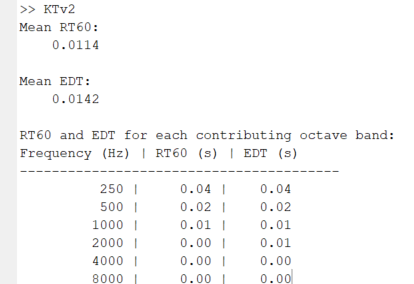
# Monday, 19 August 2024

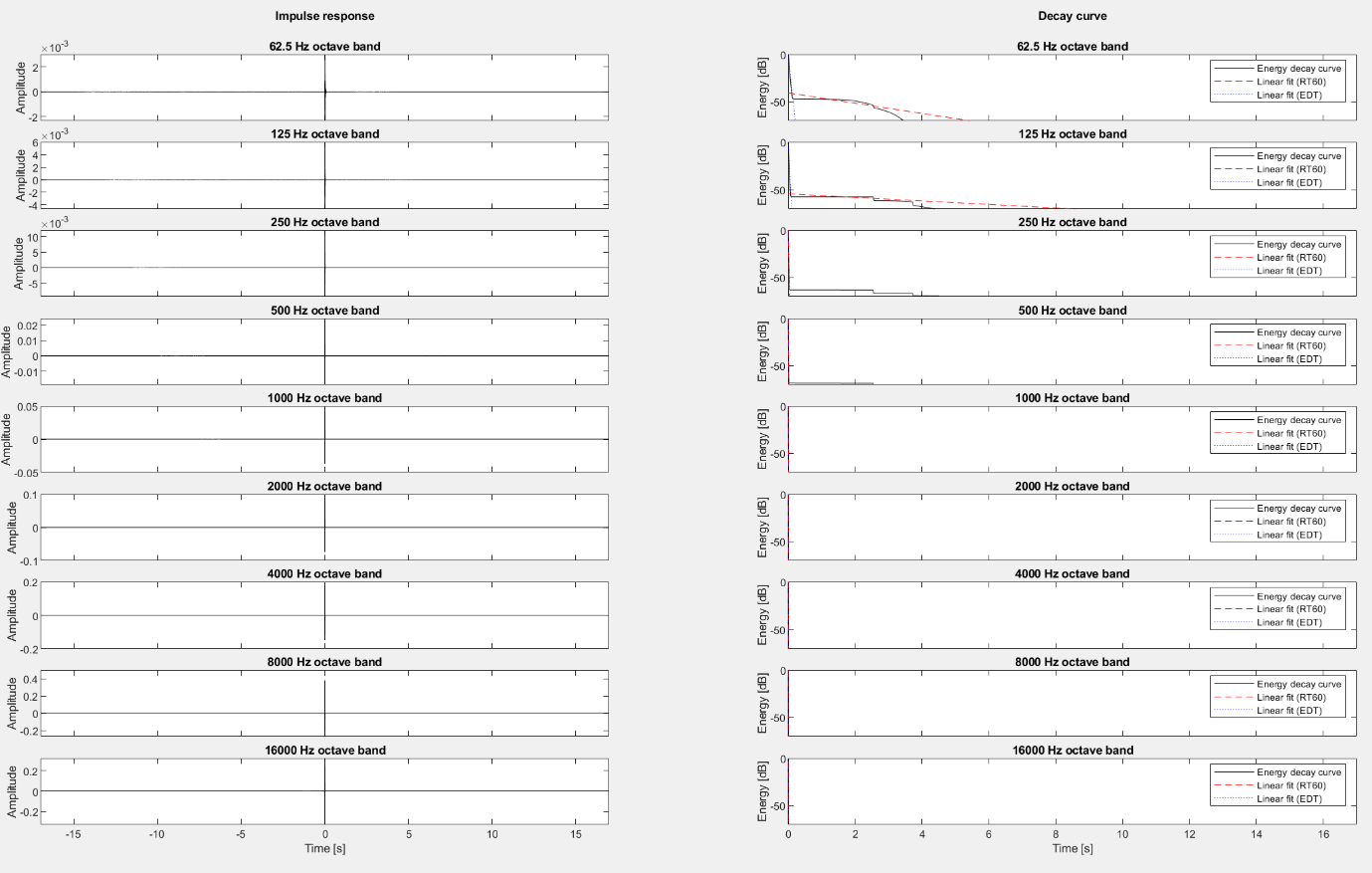
WFH, restudy RIR and deconvolution. Also looked into existing Unity project that have RIR/realistic simulation (not found).

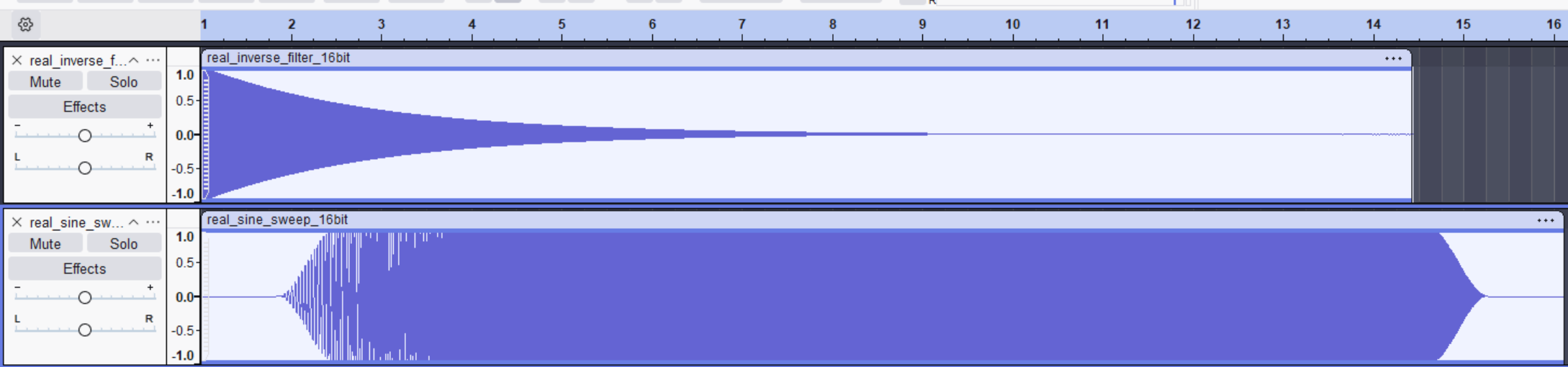
Tested RIR using generated sweep as recorded sweep:



Im assuming this anomaly (floor) is due to inaccurate inverse filter thus causing problem after deconvolve.

Instead of generating the sine sweep and inverse filter ourselves, let’s try using audacity plugin instead.



Old sine sweep and inverse filter: 

# Tuesday, 20 August 2024

## Meeting w/ Dr Hansung Kim:

### What I did last 2 weeks:

Implemented direct Unity audio recording  
Created test scenes to isolate Steam Audio issues  
Experimented with various audio parameters

### Problems encountered:

Deconvolution/inverse filter inaccuracy issue.  
Audio clipping limiting further tuning  
High EDT value (compared to RT60)  
Peak in open air test environment

### Plans:

Regenerate sine sweep and inverse filter using audacity plugin instead of python manual coding to eliminate errors etc  
Retest on test scene and generated as recorded to sanity check  
Have a meeting with Mona for evaluation progress.  
Update progress with Dr, Mona and Atiyeh.

### Take time off on 29 and 30, but work on 2 September instead. Meeting w/ Dr on 2nd September, and meet Mona for system handoff (laptop, VR HMDs) on different date.

Make a demo video or try to make it easy for VR to work (whichever most convenient to demo)

Send poster for print before 28th.

Prepare report (around 20 pages), no deadline but don’t overwork outside intern time if possible.

Make it well structured for handoff! (Well documented and structured)

Think about writing papers as dual First author w/ Mona or secondary author (prob depends on how well I got result evaluation going).

Let’s rewrite all this into proper todo list.

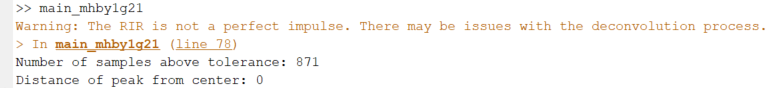
## Next 2 week TODO:

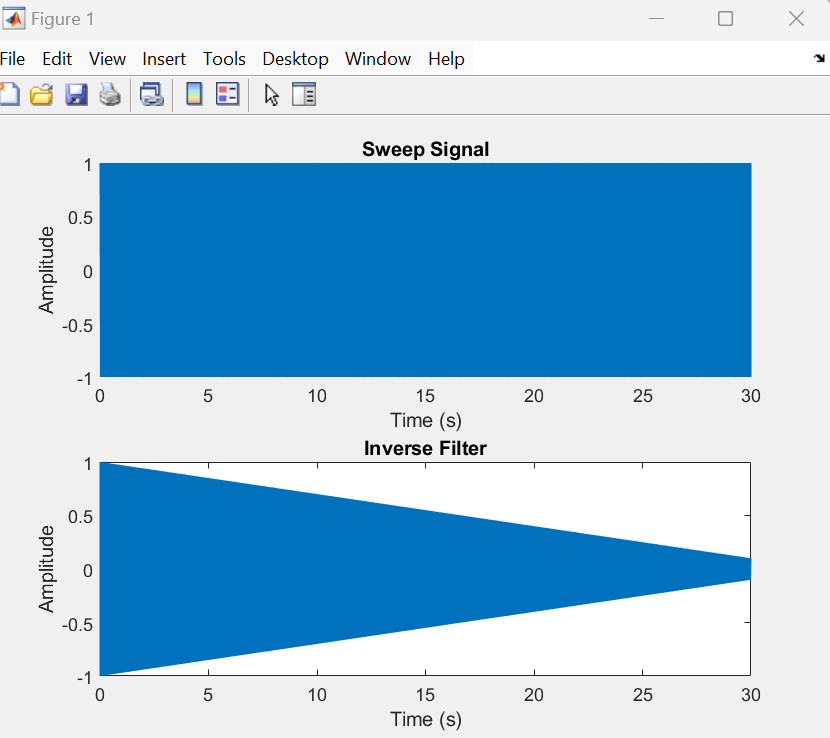
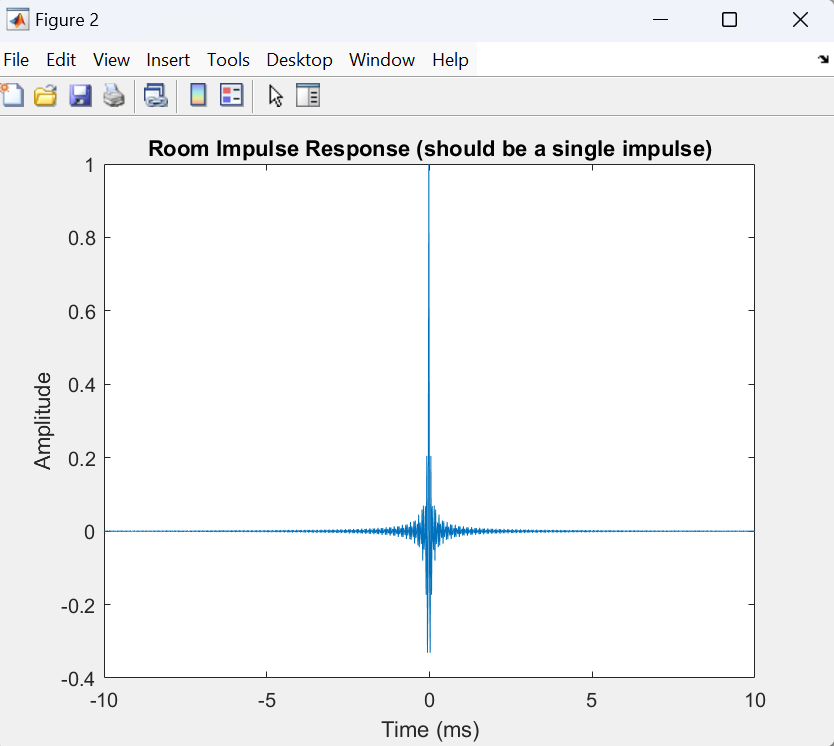
* Send poster for print before 28th.
* Make a demo video or try to make it easy for VR to work (whichever most convenient to demo)
* Prepare report (around 20 pages), no deadline but don’t overwork outside intern time if possible.
* Meeting w/ Dr Hansung on 2nd Septembe, Mona on any day after 3rd for handoff.
* Make it well structured for handoff! (Well documented and structured)
* Think about writing papers as dual First author w/ Mona or secondary author (prob depends on how well I got result evaluation going).

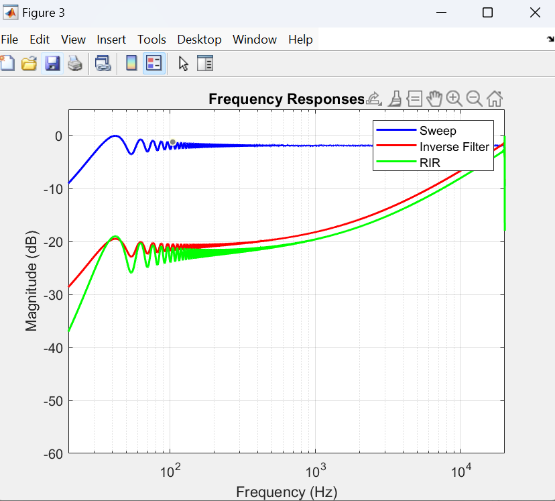
## Immediate TODO:

1. Regenerate sine sweep and inverse filter using audacity plugin instead of python manual coding to eliminate errors etc
2. Retest on test scene and generated as recorded to sanity check
3. Have a meeting with Mona for evaluation progress.
4. Update progress with Dr, Mona and Atiyeh.

Gave up on no. 1 because I think its impossible to get better approximate of sine sweep from input only, and generating sine sweep together with inverse filter (where parameter is known and defined) is a better way. Although because this would be exactly the same as Mona’s previous one, so instead I used matlab to generate the sine sweep and inverse filter instead of Python.

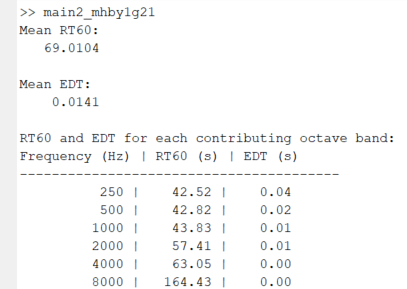
I also ran the deconvolve on the on the generated sweep straight away (without any simulation/noise) to get as perfect impulse response as I can, it still not perfect but good enough imo which prove the deconvolve is working. 

🡨 It looks a lot different than previously because its on linear instead of exponential I think.

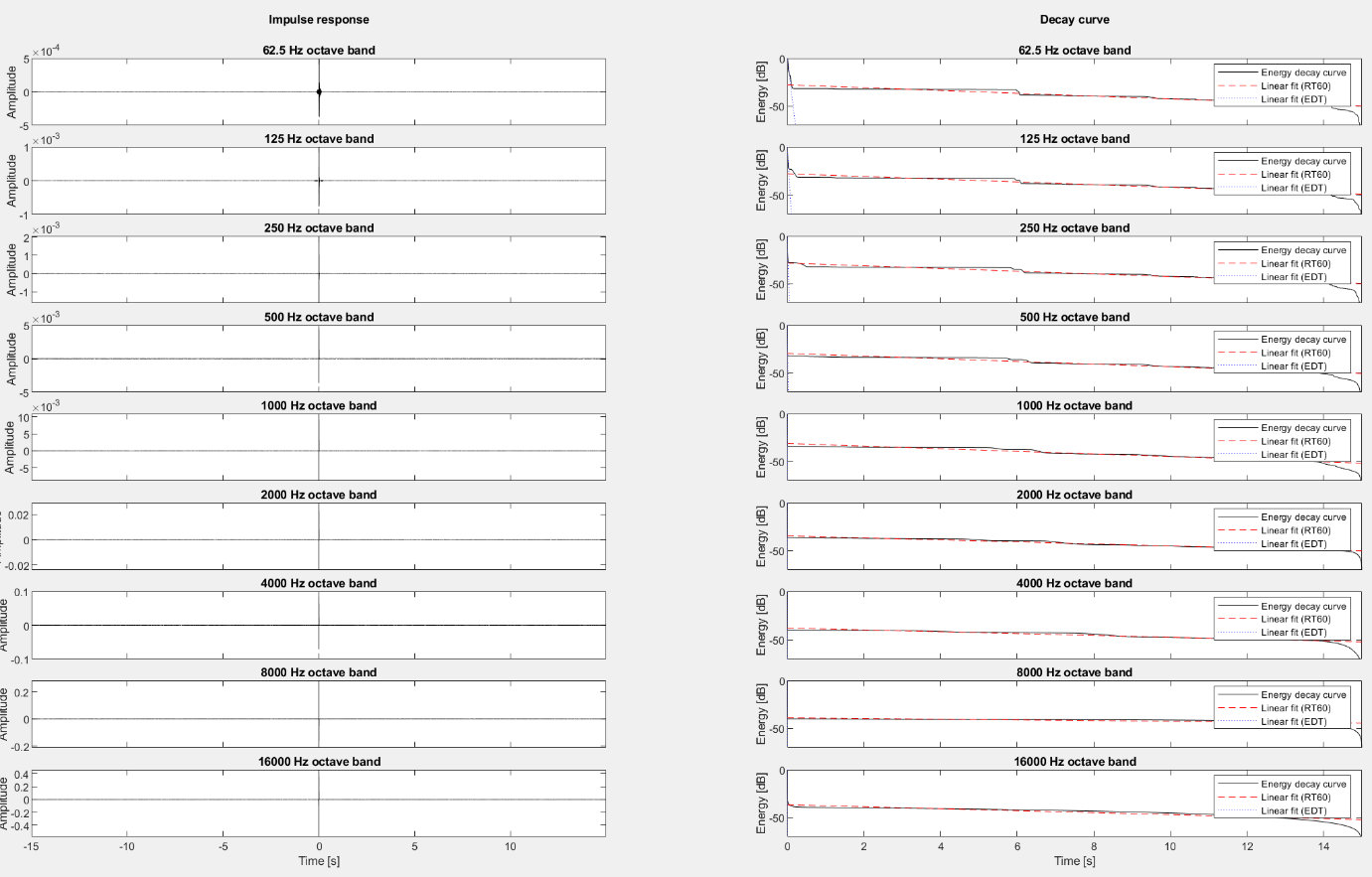
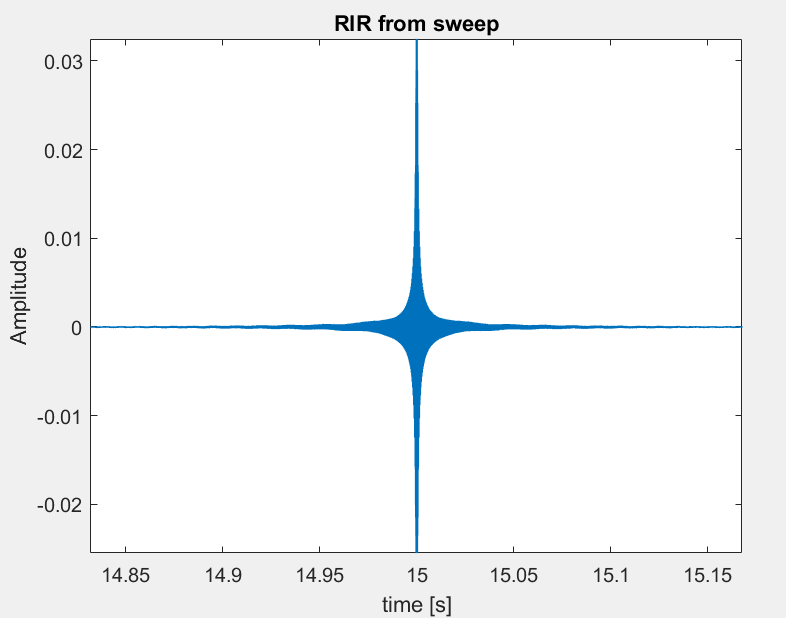
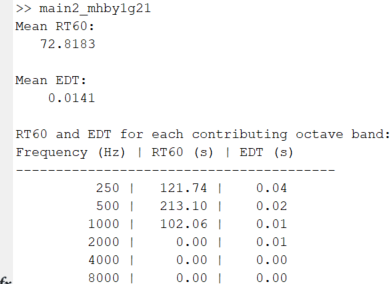
Notice the RIR is very short which is good sign (ms unit)

Also added Frequency response comparison for all 3 signals for sanity check according to this [stack exchange](https://dsp.stackexchange.com/questions/41696/calculating-the-inverse-filter-for-the-exponential-sine-sweep-method), which is different but im assuming this is because im using linear (need to revise later if this still gives problem).

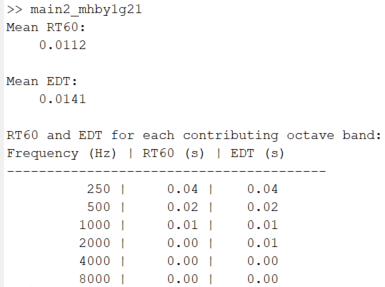
Let’s also add in the RT60 and EDT calculation to make sure.

Separated the 🡪

main\_mhby1g21.m to main1 and main2 respectively in order of to run them (will require three next for generation, rir deconvolve, and rt60/edt analysis). As seen on the right, its not working on default y\_fit due to noise floor again interestingly even though the RIR is really short and looks correct.

This means either my inverse filter or deconvolve is still wrong or maybe because im using linear, or noise floor is just inevitable, lets use y\_fit [-5 -35] first just to be sure.

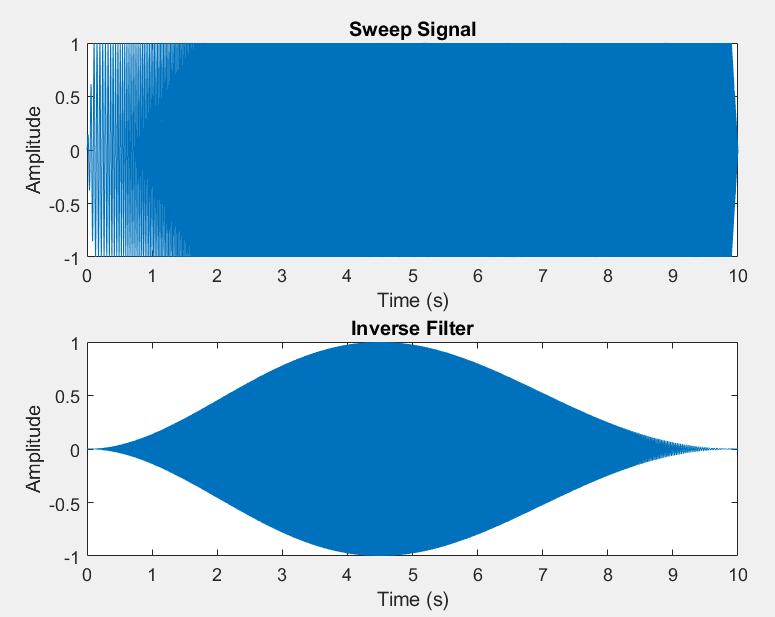
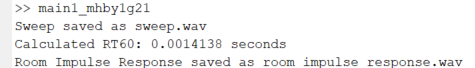
As seen on right, even -5 -35 is not enough due to noise floor anomaly.

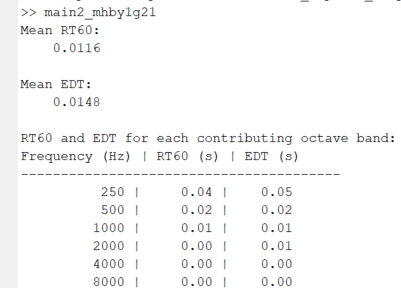
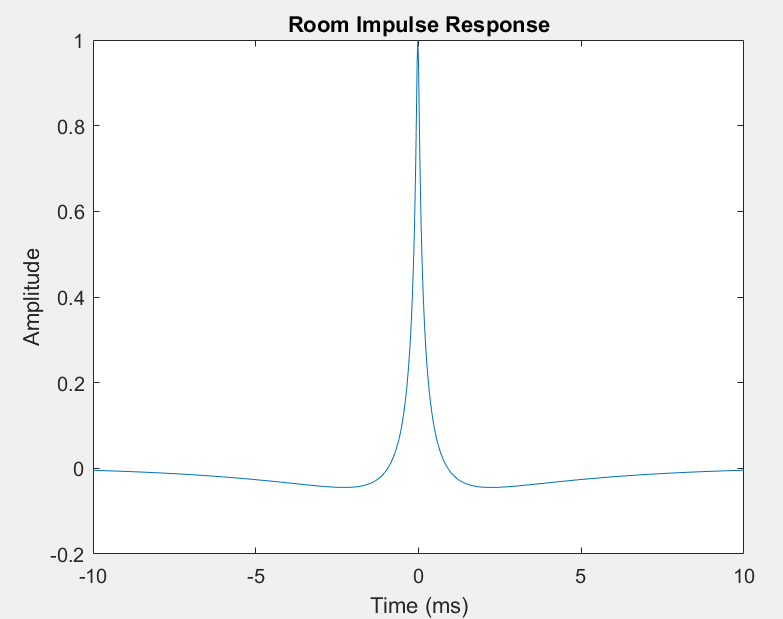
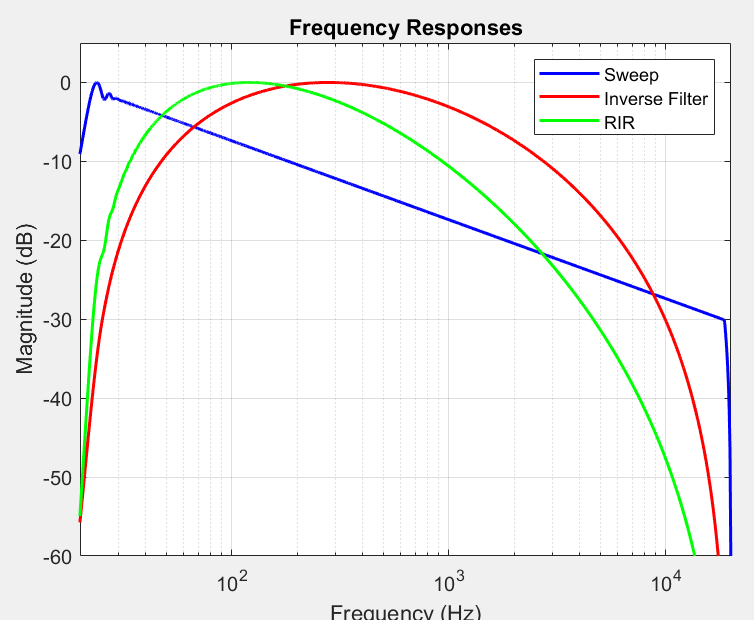
🡨-5 -25 give correct value though, however this is obviously not fit (pun intended) for use at all as this is the exact same sweep, not even simulated and it already have such high noise floor and anomaly/issues..

Ok interestingly enough, decreasing the duration of sine sweep lower the noise floor level but didn’t completely remove it, this means the problem might stem from floating point precision error cumulation or something like that.

Updated main1\_mhby1g21.m script with following improvements:

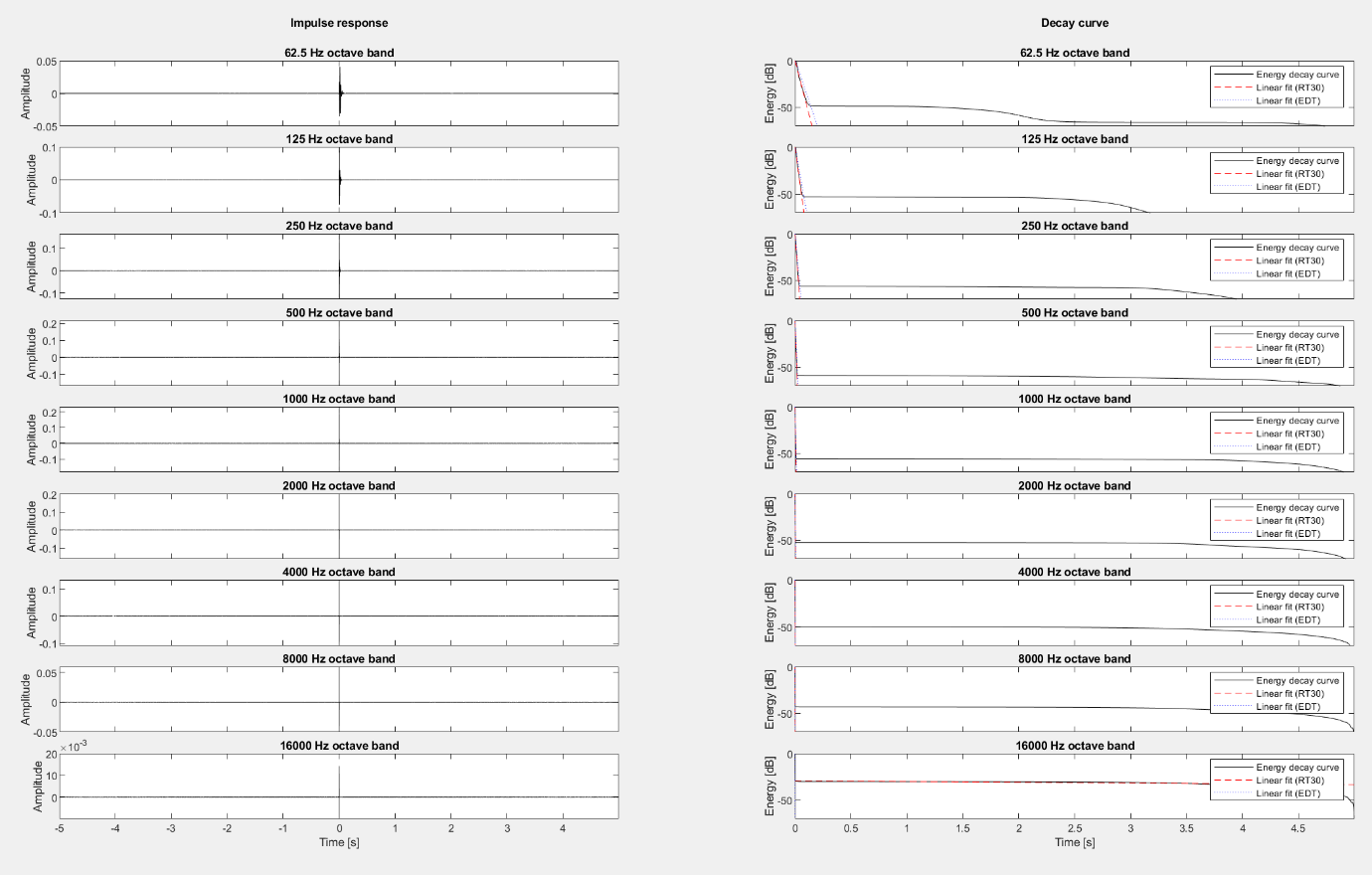
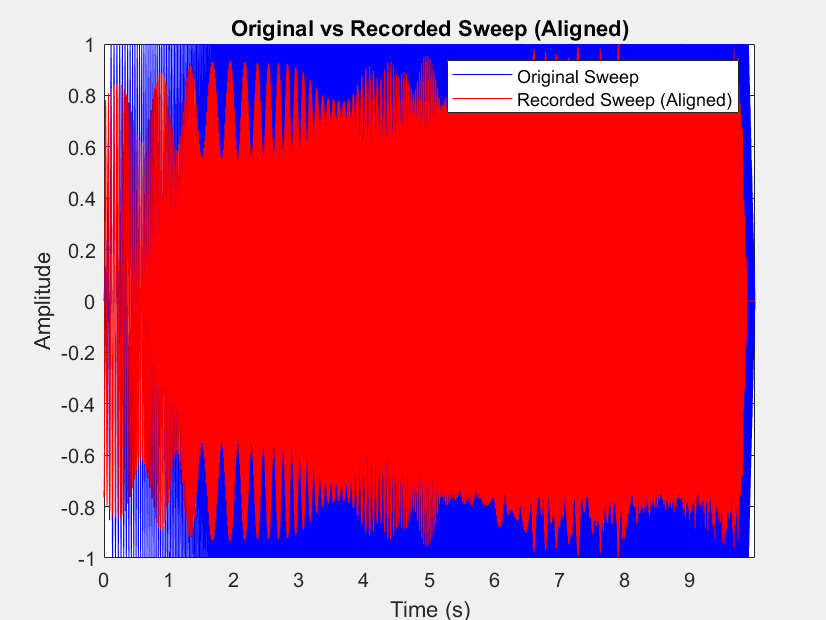
* Uses an exponential sine sweep instead of a linear chirp.
* Applies a fade-in and fade-out to the sweep.
* Uses a window function (Hann window) on the inverse filter.
* Ensures double precision for all calculations.
* Implements a custom RT60 calculation function with a noise floor cutoff.

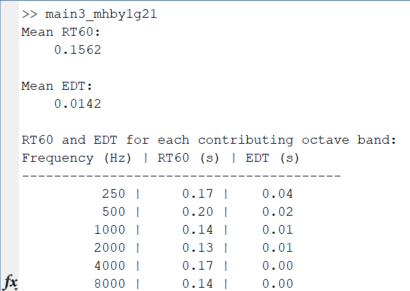
This results into best one so far! GOOD NEWS LETSGOO! 

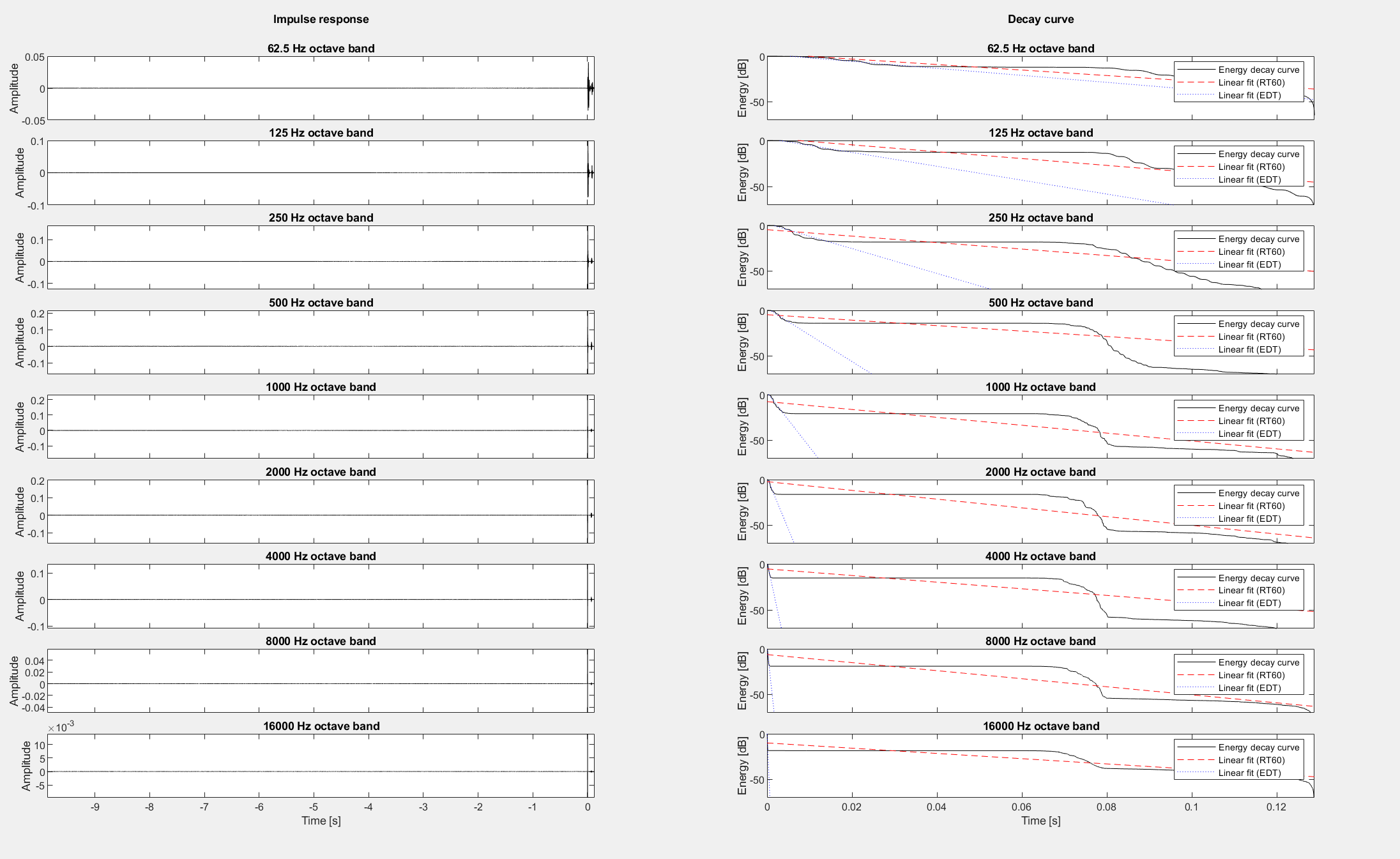


Look at how clean the RIR is (literally near dirac delta), idk what freq response really means here and the inverse filter/ sweep looks a bit strange due to modification but it seems working. This is obviously with recorded sweep being the generated sweep for sanity check.

As seen below, the noise floor still exist but atleast its way lower now that y\_fit [-5 -35] can be used reliably. Now lets run this on Unity and hope it works as well.

OK, it doesn’t work straight away obviously, theres lot variable in play especially in aligning the signals, worse part is that removing silence does not really work imo because certain frequency (esp lowest would probably not be captured in Unity, and thus the recorded sweep would seem shorter than it should) but this is the best I got for now (hack-ish cross correlation after silence removal. 

This looks correct but when u look at actual octave bands decay curve, it still have weird bump so idk.



We can change the silence threshold though so lets try that. Im gonna commit first just in case.

OK Nvm, changing the silence threshold from 0.001% to 0.0001% causes RT60 calc to gives lots of insufficient dynamic range warning and also give worse RT60 value due to higher floor somehow.

So the choice now imo is to modify our unity audio capture to automatically capture as soon as the audio is played so no need to realign on post.

Before doing that, lets try the working ish one (on top, 0.001% silence threshold) with KT scene just for sanity check. TBF, the result we got from open air is kinda similar ish to LR in Kim21 which should be the baseline imo.

Ok vol0.5 is too much for KT, lots of audio clipping and presumably also what causing not enough dynamic range warning.

Unfortunately going to vol0.1 also decrease RT60 as expected and didn’t resolve EDT issue (EDT higher than RT60)

Using attenuation combined with 0.5to5 ratio doesn’t fix the issue either tho the former does allow us to use higher vol with less clipping to get more dynamic range.  
I’m convinced the issue for EDT might either be intrinsic with my KT scene or with the alignment, so alignment would be my last try, lets do that tomorrow first thing!

### Update to TODO:

* Sync Unity audio recording with audio player so no need to align in post.
* Retest on generated as recorded to sanity check, open air test, and all scene after.
* Have a meeting with Mona for evaluation progress.
* Update progress with Dr Hansung, Mona and Atiyeh.

# Wednesday, 21 August 2024

As said yesterday, lets sync audio recording with audio player. This is fairly simple, just modified AudioRecorder.cs a bit and added few lines on DelayedAudioPlay.cs to invoke the recording before audio start playing at exact same second.