Ti mas zhtaei h ekfwnhsh tou 1ou paradoteou; Wrapper to read/write

160 samples from a .wav file. Check out what those weird functions in the pictures mean. Also this journal will either be in english or greeklish because groff can't handle greek characters well. Exw ton akoloythw kwdika: LARc: np.ndarray, curr_frame_st_resd: np.ndarray = RPE_frame_st_coder(s0: np.ndarray) Afto shmainei oti h synarthsh pairnei san argument to to opoio einai numpy array, kai exei return values dyo numpy arrays, poy ginontai assign sta kai Apo,ti katalavainw sto prwto paradoteo prepei apo to s0 na vroume ta LARc kai curr_frame_st_resd (current frame short term residual) kai epeita na kanoume thn antistrofh diadikasia ths apokwdikopoihshs, apo ta LARc kai to curr_frame_st_resd na vroume to s0. Prwto vhma eksakolouthei na einai to na melethsw th domh enos .wav arxeiou. I know we have a wav file of 8kHz sampling rate, but how many bits in a sample? 8 or 16? Do we need to support both? The provided sample wav file has 16 bits per sample. Using python's module, interfacing with .wav files seems quite easy and straightforward. I have all I need to be able to read 160 frames from a .wav file. Implement a function named to do exactly that. Gia twra paw na kanw ena dialleimma plenontas piata kai trwgontas meshmeriano. E, na mhn pesw kai gia mia meshmerianh siesta!!! I've returned, and I am storing 160 samples in a numpy array called s0, but it seems it is passed as a string? So s0 only has one element, that of all the samples of the audio data. I need to find some way to make the function return an array of bytes instead of a string, or a singular element, whichever it is. I have a suspicion that the samples are in buffer-like objects and I can laod them in via the function. I felt really tired and gave up for today... There's always tomorrow! Only started working in the evening. I asked ChatGPT how to convert from readframes() to numpy array. The results are in! Using numpy.frombuffer() and changing the dtype depending on sample width (8 or 16-bit), creates a two-dimensional 1x160 numpy array. If I then ravel() it, I get my precious one-dimensional numpy byte array. audio_wrapper_read_data() is finished! It also quits if the .wav file doesn't have 8 or 16 bits per sample. I talked with Giwrgos about the structure of this assignment. I'm to implement the read aspect of the audio wrapper (already done) and the encoder part. I should make my own branch for the encoder. The encoder should be on a separate file (separate even to the decoder) and another -like file should call everything from a central point. Giwrgos made a good point, it doesn't matter how many times I call audio_wrapper_read_data(), it will only get the first 160 samples of the wav file. When I complete the encoder, I should test it with a modified audio_wrapper_read_data() function. It will make use of a counter of how many times

I called it and audio_data.setpos() to set the required offset. I'm feeling quite sleepy at 22:20 p.m., so I should make me some light food and then go to bed early, Thursday is a day of work but Friday will be dedicated to accompanying my dad on a work-related trip, so that day will be wasted. Started really late, 13:30 p.m. Pretty frustrated and angry that I spent most of my night and my morning played brain-dead video games on my phone. I have now deleted all of them and I vow to not install any more in the near future. But for now let's calm down and focus on this. At some point I should check out how to and from a function in python. For now I'll start with the function. Pws vriskw ta LARc kai curr_frame_st_resd; H mallon akoma kalytera, poio apo ta dyo prepei na vrw prwto kai pws; Kseroume oti vriskomaste sto short term analysis. Ok prwta ypologizoume ta LARc kai meta to short term residual. Omws pws ypologizoume ta ak? Pws xrhsimopoioume thn Rw = r? Ok, so this is easier than it seems. rs(k) = Sum(s(i)s(i-k)), where s = [k, 159] and k = 0,1,...,8. Since we know all rs(k)s, we can calculate a1,a2,...,a8 from the normal matrix equations. Let's do it! The autocorrelations are done, seems like that was pretty easy. So are the prediction coefficients ak. Next step is to find the LARc. I am a bit troubled, do I calculate the residual with ak or with LARc? I need to use the function from hw_utils.py. What is the meaning of the second argument, ? It has a default value, so I'm going to leave it as it is right now. Reflection coefficient generation has failed... It seems like the algorithm fails when any of a1, a2,... are equal to 1 or -1. This is probably a skill issue on my part, maybe I did something wrong on the way? Or do I have to provide a final_error other than the default? I'll try to get another set of 160 samples from the audio wav file for now. Trying different positions on different files shows that my code mostly fails rather works even sooner than the reflection coefficient calculation. The matrix multiplication part seems to either underflow or overflow in some mystical way. In certain positions there seems to be an overflow when multiplying two scalars s0[i]*s0[i-k], part of calculating the autocorrelation's approximation. Let's try solving this first, then worrying about aks being 1. Setting up a try...except block doesn't catch the exception, but using the numpy.multiply function made it stop complaining about the multiplication overflow. The reflection coefficients can be calculated for 16-bit wav audio if I set the initial file cursor position to 493. Anything earlier than that makes it fail due to error. The .wav file header is only 44-bytes long, why do I have to go that far for the program to work? 8-bit wav audio still doesn't work at all. Correction, 8-bit wav also works if I go far enough into the file... What is this sorcery??? I'm taking a break to work out, take a shower, throw out the garbage and eat lunch. I have a hunch that the problem is the way I am getting samples from the .wav file in the audio_wrapper. I am currently reading a web article explaining how wav files work. If after reading I still can't make it work, I should try another method of getting samples from a wav file other than the wave module. Maybe will do the job? For now, web article. Following a conversation with my friend Alex I found out that I had forgot the sum part of the autocorrelation! I just multiplied! No wonder that was full of errors... Now that that is fixed, no singular matrices exceptions, no multiplication overflows and the reflection coefficients get calculated without any protests. However... When I try to calculate the Log Are Ratios, I find that, while the ak array is of size 8, the reflection coefficients array is ! How do I fix this? Is this even intentional? Shouldn't they all have size 8? What a mystery... I will continue to study this phenomenon in the coming days... Or just send an email asking if this behavior is expected. Probably won't do any work tomorrow due to previously mentioned work trip. Maybe Saturday again? Giwrgos was right, pre-processing is a completely different step from short-term analysis. I implemented this today without too many problems. Sent an email to daletras@ece.auth.gr to ask about two things: 1. Whether we need to support both 8-bit and 16-bit wav files and both big-endian and little-endian byte orders. 2. If having 7 elements in the reflection coefficient array is expected behavior. Response email hasn't arrived yet, so I will continue with the 7 reflection coefficients as if nothing has happened. Section 3.1.7 of the codec covers this. It was pretty easy, just some simple arbitrary calculations. The results don't look too quantized / encoded to me, but who am I to judge, right? I did notice that my reflection coefficients are also not in the desired range of (-1, 1]. That is further cause for concern, but for now I will wait for the answer from Dimitris Aletras. Email still hasn't come in. What is that guy up to? Marching on to FIR filtering and prediction error calculation. These are covered by Sections 3.1.8 until 3.1.11. I was happy to see the decoded LARd and krd followed LAR and kr closely. The residual is calculated by using an FIR filter with coefficients derived from LARd. I got quite perplexed at this point, is the output of the filter the residual itself or the prediciton of s0, s_hat? I'm going to ask Giwrgos, he said this part was easy so he might have a better understanding over it. Still no email. Giwrgos gave some feedback on my code. Here is what I need to fix: 1. The FIR filter's output is the residual, not s_hat 2. The FIR filter's coefficients are the decoded ak, not the decoded LARd. 3. e_final is autocorrelation's r[0] I managed to implement all of these. We are awaiting for the email from Dimitris Aletras about the 8th kr. It is also not certain that the decoded ak is in FIR filter coefficient-ready form. Besides that, the encoder can be regarded as finished. I feel a bit tired from all this work. I think I should take a break from this assignment for a few days. I'm back! Dimitris Aletras responded to my email and Giwrgos pointed out some stuff in my code that requires attention, so there's some work to be done. Dimitris's email pointed out two major points. First is that, at least for the scipy read function, it shouldn't matter if a wav file has 16-bit or 8-bit samples. Secondly, Aletras says that we should use 9 elements as an input to polynomial_coeff_to_reflection_coeff, probably the same elements that comprise the FIR filter, but do we use normal ak? We can't use ak' because those requre the reflection coefficients to be already calculated. Weird... To shmeio pou mou eipe na koitaksw sthn ekfwnhsh den voithaei katholou, kai den vriskw th nea ekdosh ths ekfwnhshs sto elearning... As far as Giwrgos' points are concerned, I should try a normal convolution rather than using the ambiguous lfilter function when calculating the residual... Or at least compare the results of both methods. My plan is to test the scipy wav read method and the convolution, then send an email asking for more information about the input to the function that generates the reflection coefficients, mention that I can't find the new version ths ekfwnhshs and add any more trouble I had along the way Testing the scipy wav read compared to the wave module read. Theoretically, we shouldn't need to care about whether the wav samples are 16- or 8-bit. Scipy wav read is a million times easier omg.... I'm using that one from now on. I don't trust the scipy.signal.lfilter function, so I'll try using the FIR filter using normal convolution. Convolution seems to work better, it's as if lfilter cropped out some values... I want to talk about reflection coefficients, clarifications on the output of the FIR filter (with input of s0 and akd, output is s_hat or the residual?) and ekfwnhsh not correctly uploaded. Sent it. That's enough for today, I'm ill... O Aletras mou leei oti "eimai ston swsto dromo" xwris na dwsei parapanw ek-shghseis... Ypothetw oti prepei apla na valw sthn eisodo ths polynomial_coeff_to_reflection_coeff() to array [1, -a1, -a2, ...]. Anarwtiemai pws eprepe na to mantepsoume monoi mas afto... Gia shmera tha kanw afto, analoga pws ni-wthw isws prospathisw na kanw thn epeksergasia se ola ta frames enos audio file. Aaaa ksexasa oti prepei na koitaksw kai an to preprocessing m einai legit k na kanw rename ola ta s0 gia na einai symfwno me tis onomasies toy protypou. Poly entharryntika apotelesmata! Oi kvantismenes ekdoxes LARc moiazoun poly me ta LAR, paro-moia kai oi apokwdikopoihmenes ekdoxes twn LAR, reflection coeff kai ak moiazoun poly me ta arxika... Nomizw to vrhkame! I also renamed s0 arrays to be specification compliant both in preprocessing and the encoder. Iterating over the whole file seems to work like a charm! Encoder works on different audio wav files too. We omit some bits at the end of each wav file if number of samples is not divisible by 160. It's no big deal if we lose some bits from the end of the file, right? 160 / 8000 = 20ms, that's not too bad, but it definitely will be audible. The 1st assignment is basically finished from my part. I need to wait on Giwrgos to finish up his decoder and compare the output with the input signal. That will be the final test. I could try implementing Log Area Ratios Interpolation, it shouldn't be too hard. Then I should probably get started on the report before we get too deep into assignment 2, as Giwrgos doesn't know any LaTeX at all... For now I'll take a break, eat lunch, have a noon siesta, and chill out mexri to ergasthrio hlektronikhs 3 gia thn ergasia telestikou enisxyth. I tried later in the day to implement interpolation, but I didn't quite get where this interpolation should happen, since we are only seeing the 160 samples at the beginning of the encoder, not ever again. Maybe this is a decoder thing? Later in the same day I started working on the report. Will we manage to complete the 2nd assignment before the deadline? To programma ths eksetastikhs mou einai poly fortwmeno, amfivallw... Re-started the report because I didn't like how it was going at all. Now it has a much better structure I think! We officially have our first reconstructed wav file! This is big news! Unfortunately, it sounds like shit. Not total noise, but I think all we introduced with our work was just noise. I did notice that if we skip prepro-cessing, the audio file actually sounds clearer between the voice sounds. Are we introducing noise both in the pre-processing and the short term analysis part? More importantly, are we making mistakes in both st analysis and pre-processing, or is some of this natural? Tha kanoume klhsh mallon me ton Giwrgo shmera na doume ti mellei gen-esthai... Mallon giati paizei na mhn mporei giati allakse h wra sthn prova Angus to vrady... Axxx... Kataramenh mpanta... Elpizw na vroume to provlhma ston kwdika syntoma. I am trying to isolate the problem and optimize the code. Moved the preprocessign section out of the main loop, now it happens only once and is applied to the whole array of audio data, not just 160 samples at a time. It sounds exactly the same, so at least we know our logic and loops are working correctly. I put some parentheses at the preprocessing calculations to enforce the operation priority I wanted explicitly, this actually did make the reconstructed wav file as clear as the one that didn't use preprocessing at all! Damn implicitness! It did bite us in the ass this time. Some ranges were wrong when calculating the decoded LARds. I found it! The biggest problem was due to miscommunication. I had understood that I should return the decoded LARds while Giwrgos had understood that the output of the encoder were the encoded LARcs. So we ended up decoding the Log Area Ratios twice which just made no sense and introduced all the noise. Now the reconstructed audio is quite clear! I think this is actually the 1st level done! I should quickly get back to finishing the report. Maybe we should attempt level 2 as well? We have another problem! Our program doesn't handle wav files that aren't 16-bit and have their peak amplitude below -25dB... Also I have moved the preprocessing outside of the main loop and it processes a wav file all at once, problem is, if the wav file is big enough, a cryptic integer overflow warning is thrown, even though the number of samples isn't big enough to overflow an int32 and neither are the calculations. If I move it inside the loop like I did it before, I get singular matrix errors on some files! Tough times! I am not sure why I didn't bother to write after the discord call we had with Giwrgos. Maybe it went so well that I completely forgot about this journal. Dimitris Aletras says we don't need to to worry too muxh about input files so that's our input file problem done... Giwrgos made some further tweaks to the code and it seems to be working quite fine now, maybe even better than right after our call. Overflow problem?? Not sure if it's worth addressing considering the code is running correctly. Τ*ι θα γνει αν γρψω ελληνικ* Ξ*χασα.* Α*νυπο*μ*ον για το* 2*ο επιπεδο αυτς της εργασας.* Ε*λπζω να προλβου*μ*ε και αυτ και την αναφορ, γιατ ο χρνος πιζει...* Π*ρτεινα κλση στον* Γ*ιργο για αριο βρδυ να συζητσου*μ*ε ποι θα εναι το πλνο* μ*ας.* Σμ*ερα δνω αξιοπιστι και τη βαρι*μ*αι τσο που αντ για αυτ κνω αυτ και κοιτζω το επ*μ*ενο* μ*ρος της εκφνησης* Π*ολυ*μ*σα.*