Multimedia Systems, Winter Semester 2024
Electrical & Computer Engineering Department, AUTH

Voice encoder-decoder using ETSI GSM 06.10 speech coding standard

Afroditi Avramidou

Athina Apostolidou

1st Deliverable:

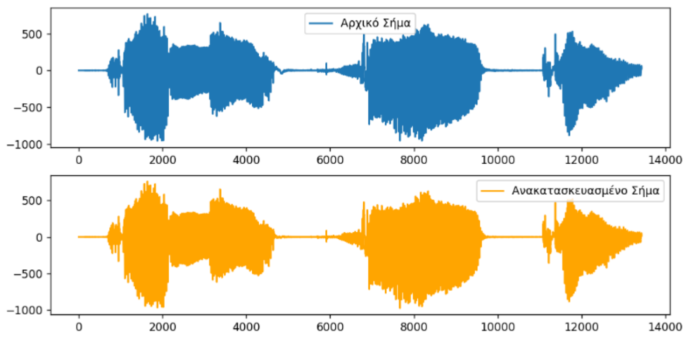
This stage involves the preprocessing of the speech signal, specifically the Offset Compensation and Pre-emphasis stages, as well as the Short Term Analysis, which is divided into the following steps: estimation of the optimal short-term predictor and the short-term prediction.
Initially, in the **preprocess(frame)** function, a single frame is processed by applying the Offset Compensation and Pre-emphasis procedures, as described in section 3.1.1 of the standard. When these are completed, we obtain the processed signal s(n).
Regarding the Short Term Analysis, since it is performed at frame level, the signal is first split into frames of 160 samples each using the function **split_into_frames(signal, frame_size=160).** The function **autocorrelation(frame, lag)** with frame and lag (lag = 0,…,8) arguments, is used based on section 3.1.4 of the standard, to calculate the first 9 values of the autocorrelation function using the estimator ACF(k) ≈ $r_s(k)$.
With **compute_R_and_r(signal, order=9)** function, 8x8 matrix R and 8×1 column vector r with the autocorrelation coefficients of the signal s(n) are calculated. Then using **solve_predictor_coefficients(signal, order=9)**, the Normal Equations R·w = r are being solved in order to find the vector w = [a1, . . . , a8]^T, which gives the coefficients a_k (k = 1 . . . 8) of the predictor: ŝ(n) = $\sum_{k=1}^{8} a_k s(n-k)$. These coefficients are used to eventually generate the residual: d(n) = s(n) - ŝ(n). The conversion: a_k, k = 1 . . . 8 < –– > r(i), i = 1 . . . 8 from coefficients to Reflection Coefficients is already implemented. The function **reflection_coef_to_LAR(r)** (section 3.1.6) implements the second conversion:
r(i), i = 1 . . . 8 < –– > LAR(i), i = 1 . . . 8. Correspondingly, **LAR_to_reflection_coef(LAR)** performs the inverse conversion, which is needed during decoding. The quantization and coding of the LAR(i) values is performed with function **quantization_coding_LAR(LAR)** (section 3.1.7), which returns the quantized integer values of LAR (LARq). Finally, the decoding of the LARq values is performed using the function **decode_of_LARc(LARq)** (section 3.1.8). All of the above functions are helper functions used in the required encoder and decoder: **RPE_frame_st_coder** and **RPE_frame_st_decoder**, respectively.
The last helper function used during decoding is: **iir_deemphasis(s_r)**, which performs post-processing on the predicted signal ŝ(n), reversing the first pre-processing stage described in section 3.2.4 of the standard (there the symbol sr is being used instead of ŝ). This function takes as input a list of samples and returns a list of filtered samples (s_{ro}).

Next, we have the implementation of the required encoder: **RPE_frame_st_coder(s0: np.ndarray, prev_frame_st_resd: np.ndarray)**, which computes the short-term residual signal d'(n). First, preprocess(s0) is used for pre-processing of the speech signal as described above. Then, Short Term Analysis is performed through a sequence of function calls: solve_predictor_coefficients(processed, order=9) to compute the order-9 predictor coefficients (we use 9 because the first coefficient is 1 and the remaining 8 are the polynomial coefficients), polynomial_coeff_to_reflection_coeff, reflection_coef_to_LAR, quantization_coding_LAR(LAR), decode_of_LARc(LARq), LAR_to_reflection_coef(LARc), reflection_coeff_to_polynomial_coeff(dec_r).
Through this process, we obtain the decoded polynomial coefficients, which are needed for the filtering process. Based on these, the short-term prediction stage is executed, which

includes: using an FIR filter to compute filter coefficients based on the previous 8 samples, estimating the predicted signal ŝ(n), and computing the short-term residual signal d(n) (curr_frame_st_resd) by subtracting the predicted signal from the processed signal.

Thus, the residual represents what remains after removing the short-term dependencies of the speech signal and is returned by the encoder along with the quantized LARc values required for synthesis.

Finally, the decoder: **RPE_frame_st_decoder(LARc: np.ndarray, curr_frame_st_resd: np.ndarray)** performs the inverse process, the reconstructing the original speech signal using the short-term residual and the LAR coefficients extracted from the encoder. By calling the functions: LAR_to_reflection_coef(LARc) and reflection_coeff_to_polynomial_coeff(dec_r), the LAR coefficients are converted back into reflection coefficients, and then into polynomial coefficients, as defined by the standard. These coefficients are then used to build the denominator of the LPC synthesis filter, which is applied to the short-term residual signal to reconstruct the audio waveform. Finally, the reconstructed signal is passed through an IIR de-emphasis filter using the helper function iir_deemphasis(s_0), which compensates for the high-frequency emphasis applied during preprocessing. The decoder thus returns the final reconstructed voice frame.



The initial and the reconstructed voice signal

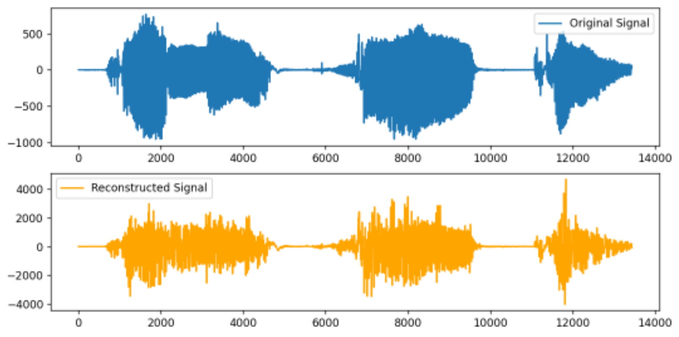
From the above plot, we observe that we have an almost perfect reconstruction of the initial signal.

2nd Deliverable:

At this stage, part of the Long Term Analysis is added and specifically, the estimation of the Long Term prediction parameters N (pitch period) and b (gain factor), that are used during the prediction $\hat{d}$ (n) = b*d(n − N) of the final prediction error, and the synthesis of the short-term residuals from previous subframes so that they can be used in the estimation process. Regarding the estimation process of the Long Term prediction parameters, the helper functions that have been implemented and are described in section 3.1.13 of the standard are**: compute_cross_correlation(subframe, prev_residual, lag_range=(40, 120))**, that computes the cross-correlation $R_j(\lambda)$ between the current subframe of the short-term residual signal and a previously reconstructed residual signal for a range of values λ in [40, 120], **find_optimal_lag(cross_correlation, lag_range=(40, 120))**, that finds the lag value N_j that maximizes the cross-correlation, **compute_gain_factor(subframe, prev_residual, optimal_lag),** that computes the gain factor b_j, which scales the previous residual to approximate the current subframe. The above three functions are used within the requested function **RPE_subframe_slt_lte(d: np.ndarray, prev_d: np.ndarray)**, which calculates the LTP parameters N_j and b_j for a given subframe.

As part of the prediction process, the following encoding/decoding functions of N_j and b_j are implemented**: encode_ltp_lag(Nj: int)**, **decode_ltp_lag(Ncj: int)**, **encode_gain(bj)**, and **decode_gain(bcj)**. Following that, the function **long_term_analysis_filtering(d, prev_d_reconstructed, ltp_params)** is implemented according to section 3.1.16 of the standard. It uses the LTP parameters to compute the long term residual signal e(n) and the reconstructed signal d'(n). This function returns e(n) as well as the updated previous residual signal, which will be used as input in future frames. Finally, the function: **long_term_synthesis_filtering(e: np.ndarray, prev_d_reconstructed: np.ndarray, ltp_params: list)** is implemented, that synthesizes the sequences d'(n) and returns the reconstructed residual signal of the current frame and the updated residual to be used in the next frame.

In the final phase, the required encoder and decoder are implemented:
RPE_frame_slt_coder(s0: np.ndarray, prev_frame_st_resd: np.ndarray), and
RPE_frame_slt_decoder(LARc: np.ndarray, Nc: List[int], bc: List[int], curr_frame_ex_full: np.ndarray, prev_frame_st_resd: np.ndarray), that use all the above helper functions.
The encoder performs RPE and LTP analysis, processes the signal at subframe level, calculates the LTP parameters and applies long-term analysis and synthesis. It encodes the residual signal and returns it along with the excitation sequence for the current frame and the encoded parameters.
The decoder, on its part, performs the reverse process: it takes the encoded parameters and the excitation sequence as input, applies long-term prediction synthesis (computation of d''(n)), and decodes the audio signal, thus it gives as a result the reconstructed signal and the reconstructed residual.



The initial and the reconstructed voice signal

From the above plot, we observe that the waveform of the reconstructed signal approaches that of the original quite well.