Speech and Audio Signal Processing

TECHNISCHE UNIVERSITÄT DARMSTADT

Tutorial 1

Prof. Dr.-Ing. H. Puder
Adaptive Systems for Processing of Speech and Audio Signals
Graduate School Computational Engineering

Problem 1 Basics: Read, Write, Playback, Block Processing

A digital speech signal has been recorded with N samples at a sampling rate of $f_s = 20\,\text{kHz}$. It is stored in `speech_sample.wav`.

a) Load the speech signal with `[x,f_s] = audioread('speech_sample.wav');` where `f_s` is the sampling frequency and write a function

```
X_b = split_signal(x, M, OL, w)
```

that splits signal `x` into blocks of length `M` with a given overlap ratio `OL`, $0 \leq OL < 1$. The function should also apply a window `w` defined by a vector of the same length as each block. Store all B blocks in a matrix `X_b` such that each row of `X_b` corresponds to one windowed block of `x`.

Hint: For simplicity, truncate the signal to a multiple of the block length.

b) Apply a gain by multiplying each signal block `b` with an individual gain factor `g(b)`. Use a sinusoid (with offset) as gain function, i.e.,

$$g(b) = 0.5 \sin(\omega t(b)) + 0.5 \tag{1}$$

with P periods over the duration of the entire signal and

$$\omega = 2\pi P \frac{f_s}{N}, \tag{2}$$

$$t(b) = b\frac{M}{f_s}, \quad b = 0, 1, ..., B. \tag{3}$$

Use the `split_signal` function from a) with a block length of $M = 512$, an overlap ratio of $OL = 0.5$ and a Hann window. Then loop through the blocks to apply the gain.

MATLAB functions: `hann`

c) Now, implement a function

```
x = merge_blocks(X_b)
```

that returns one signal vector from a matrix X_b containing the signal blocks with an overlap of 50% as shown in Figure 1.

Create an audio signal from the blocks you applied the gain to and listen to the result. What would happen if there was no overlapping of the signal blocks (i.e., $OL = 0$) while using a rectangular window?

MATLAB functions: `soundsc`

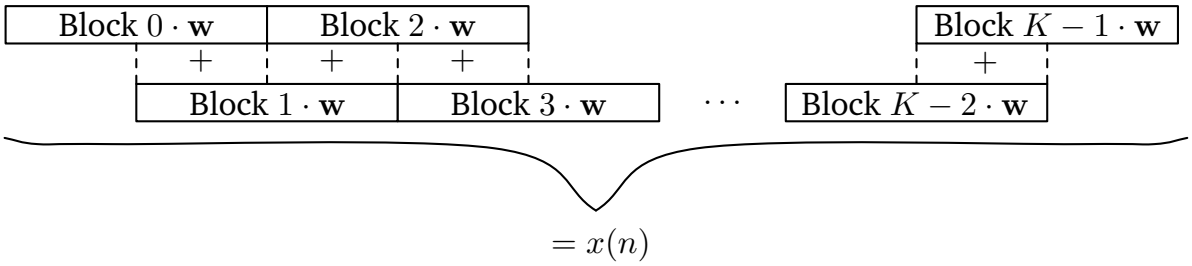

Figure 1: Illustration of the merging process. Block k denotes the k-th signal block with $k = 1,\dots,K-1$ (K = number of blocks) while $\mathbf{w} = [w(0),\dots,w(M-1)]$ is given by a Hann window of length M.

Problem 2 Linear Prediction

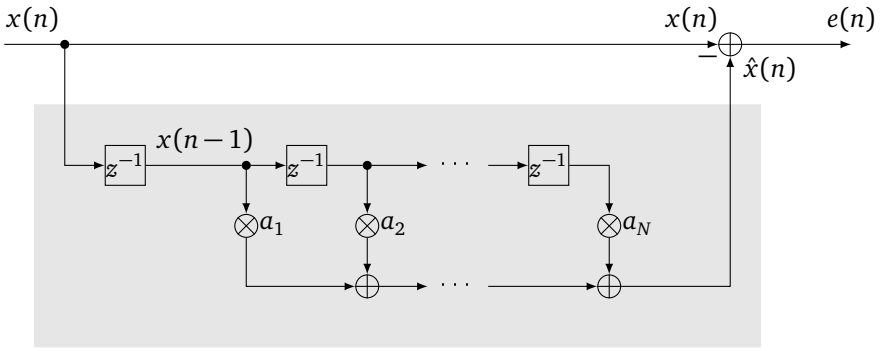

Figure 2: Block diagram of a linear prediction filter of order N.

In this problem a prediction filter as depicted in Figure 2 is considered. The filter coefficients a_i, $i = 1,\dots,N$ are obtained by the Wiener solution. The relationship between the output $\hat{x}(n)$ of the predictor and the input signal $x(n)$ is given by:

$$\hat{x}(n) = \sum_{i=1}^{N} a_i x(n-i) = \mathbf{a}^T \mathbf{x}(n-1),$$

where $\mathbf{x}(n-1) = [x(n-1),\dots,x(n-N)]^T$ and $\mathbf{a} = [a_1,\dots,a_N]^T$. Consequently, the filter is of order N. The filter coefficients are calculated according to the Yule-Walker equation:

$$\mathbf{a} = \hat{\mathbf{R}}_{xx}(0)^{-1} \hat{\mathbf{r}}_{xx}(1) \tag{4}$$

with estimates of the autocorrelation matrix $\mathbf{R}_{xx}(0)$ and autocorrelation vector $\mathbf{r}_{xx}(1)$, which are given by

$$\mathbf{R}_{xx}(0) = \text{Toeplitz}([r_{xx}(0),\dots,r_{xx}(N-1)]^T) = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \dots & r_{xx}(N-1) \\ r_{xx}(1) & r_{xx}(0) & \dots & r_{xx}(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}(N-1) & r_{xx}(N-2) & \dots & r_{xx}(0) \end{bmatrix}$$

and $\mathbf{r}_{xx}(1) = [r_{xx}(1),\dots,r_{xx}(N)]^T$.

The autocorrelation function $r_{xx}(l)$ can be estimated using the autocorrelation method:

$$\hat{r}_{xx}(l) = \frac{1}{M} \sum_{n=0}^{M-1-l} x(n)x(n+l) \tag{5}$$

with $l = 0,\dots,N$ and M being the amount of samples in x.

a) Write a function `r_xx=autocorr(x, N)` that implements the autocorrelation method as described in Eq. (5), where `x` is the signal vector and `N` the filter order.

b) Write a function `a=predictor_YuleWalker(x, N)` that calculates the filter coefficients **a** accordingly to Eq. (4) with the same inputs as in a). There are many possibilities in MATLAB to create an inverse of a matrix. Note that they all differ in their numerical method of inverting a matrix. For our purpose, you can choose any of the methods (e.g. simple matrix division operator \ or / for left- or right-sided division).

   MATLAB functions: `toeplitz`

In a last step, we want to apply the prediction filter onto a signal and analyze the result. For this, use the `noise_generator` function provided in moodle which realizes a signal from an almost pink noise process.

c) Create a signal containing 20,000 samples. Estimate its autocorrelation function and generate signal $\hat{x}(n)$ by applying a prediction filter of order $N = 12$. Finally, determine the error by $e(n) = x(n) - \hat{x}(n)$ and compare the PSD estimates of $e(n)$ and $x(n)$. For this, use Welch's power spectral density estimate with a Hann window of length 256 and an overlap of 50% to achieve a sufficiently smooth estimate.

   MATLAB functions: `pwelch`

d) For the time-invariant prediction filter use the speech signal `(speech_sample.wav)` instead of the generated noise signal and apply the same processing steps as described in c). This is equivalent to applying a mean prediction to the non-stationary speech signal.

Since speech is a non-stationary signal its statistical properties are time-variant. This results in time-variant prediction filter for achieving a higher prediction gain than for time-invariant filters.

e) Use the `split_signal` function (Problem 1a) to split the signal into weighted blocks, calculate the prediction filter for each block, apply these filters to the input signal blocks, and calculate the prediction error blocks. Finally merge the error signal blocks to an error signal as result of blockwise prediction error filtering. Compare the power spectral density of this signal with the one obtained for the error signal of d). Also compare the spectrograms of the two error signals from d) and e) by calling: `spectrogram(err,1024,512,1024,f_s)` where err is the respective error signal.