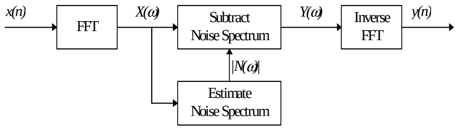
Speech Enhancement

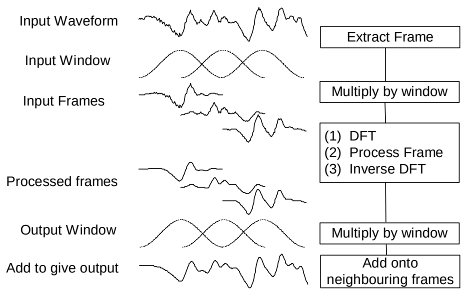
- Spectral subtraction principles
- Overlap-add processing
  - Windowing
  - Oversampling
- Noise subtraction
- Noise estimation
  - Minimisation buffers
- Output Buffers

Spectral Subtraction



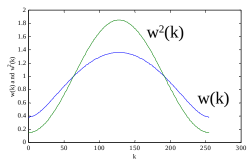
- Processing is entirely in the frequency domain
  - Estimate noise spectrum when the speaker is silent
  - Assume the noise spectrum doesn't change rapidly
  - Subtract the noise spectrum from the input signal
  - Convert back into the time domain to generate the output

Overlap-Add Processing



Input and Output Windows

- Each frame is multiplied by the input window and then by the output window.
  - Need the windows to avoid spectral artifacts from discontinuities at the frame boundaries.
  - Choose the windows so that the overlapped windows sum to a constant.
  - Square root of Hamming window:



$$w(k) = \sqrt{1 - 0.85185\cos\left((2k+1)\pi / N\right)} \quad \text{for} \quad k = 0, \cdots, N-1$$

Frame Length and Oversampling

- Frame length is a compromise
  - Long frames give good frequency resolution but poor time resolution (and normally require more processing)
  - Short frames give good time but poor frequency resolution
  - FFT is more efficient if length is a power of 2
  - We choose a length of 256 = 32 ms @ 8 kHz
- Each frequency bin is sampled once per frame
  - Need to sample fast enough to avoid aliassing if the magnitude of a frequency component changes rapidly.
  - Frequency component amplitude changes are smoothed by the input/output windows
  - For Hamming window 4× over-sampling (a 75% overlap) is best but we can get away with 2× over-sampling (a 50% overlap) if processing power is in short supply.

Subtracting Noise Spectrum

- If we knew the phase of the noise: $Y(\omega) = X(\omega) - N(\omega)$
- Since we don't, we subtract magnitudes (or else powers):

$$Y(\omega) = X(\omega) \times \frac{|X(\omega)| - |N(\omega)|}{|X(\omega)|} = X(\omega) \times \left(1 - \frac{|N(\omega)|}{|X(\omega)|}\right) = X(\omega) \times g(\omega)$$

- $g(\omega)$ can go negative if our estimated noise exceeds the input signal. Hence we limit $g(\omega)$ to some minimum:

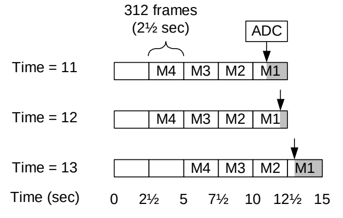
$$g(\omega) = \max\left(\lambda, 1 - \frac{|N(\omega)|}{|X(\omega)|}\right)$$

Estimating the Noise

- Very hard to detect reliably when speaker is silent
- Instead, assume that he stops for a bit at least every 10 sec
  - At each frequency, take the minimum power over the past 10 sec
  - Multiply by a compensation factor ($\alpha \approx 2$) to estimate the **average** noise amplitude as opposed to the minimum.
- Method 1: Store all speech spectra over the past 10 sec
  - Too much storage: 625 or 1250 frames depending on overlap
  - Too much calculating to find the minimum
- Method 2: Calculate the minimum of each 2.5 sec chunk
  - Take the minimum of the current and three previous chunks at each frequency separately.
  - Not so accurate but much less storage & computation

Minimum Buffers

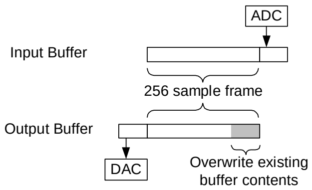
- M2, M3 and M4 each hold a complete spectrum that is the minimum over a 2½ sec interval.
- M1 contains the minimum spectrum of the frames since the last chunk boundary.



- We always take the minimum of M1, M2, M3 and M4.
- This corresponds to an interval of between 7½ and 10 sec.
- Every 2½ sec we transfer M3→M4, M2 →M3, M1 →M2 and then reinitialise M1 to the current input frame.

Input/Output Buffers

- 4× oversampling ⇒ we must process a 256-sample frame while ADC reads in the next 64 samples.
- Since frames overlap, we must add results onto those from previous frames.



- The last 64 samples of our result frame doesn't have any previous data to add on to, so we just overwrite the previous buffer contents.
- Use 1¼ frame circular buffers for input and output
- We have a 1¼ frame **algorithmic** delay (independent of processor speed) from input→output.