Digital Filtering

Shape the spectrum of a signal
-Removing

[Graph]

-Enhance

[Graph]

[bass boost]
- Shape spectrum

(i)

Filter $\text{shape}$

(ii)

- Consider
  - Frequency domain
  - Time domain
If:

- $x(t)$ is the input signal
- $X(s)$ is its Fourier transform

$h(s)$ is the filter specification
$h(t)$ is the inverse Fourier transform of $H(s)$

1) $y(t) = x(t) * h(t)$  % convolution
2) $Y(s) = X(s) H(s)$  % multiplication

I. Frequency domain

- Point-by-point multiplication

- $X(s)$
- Input
- $H(s)$
- Filter
- $Y(s)$
- Output
II. Time Domain

\[ y(n) = \sum_{k=-\infty}^{\infty} x(k) h(n-k) \]

- \(N\) is length of \(h\) - filter,
- Requires \(N\) multiplies, \(N-1\) adds \(\} \) each output \(y(n)\)

\[ x(n) \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \]
\[ h(n) \quad x \quad x \quad x \quad x \quad x \]
\[ y(n) \quad \Delta \]

**MATLAB**

\[ \text{Out} = \text{filter} \left( \text{coeffs, 1, in} \right) \]

Complex \(h(t)\) - more complex calculations

Real \(h(t)\) - preferred

\[ \text{Imag} \left( \text{Even}(s) \right) = 0 \]
\[ \text{Real} \left( \text{Odd}(s) \right) = 0 \]
\[ F(z) = \text{Real} \left( \text{Even} \right) + \text{Imag} \left( \text{Odd} \right) \]
In the time domain, choose a real and even filter

\[ H(s) \] then real and even

\[ H(s) = H(-s) \]

→ linear phase

Filter \( x(t) \) in time domain \( \rightarrow y(t) \)

**Approach 1**

convolve \( x(t) + h(t) = y(t) \)

**Approach 2**

1) Transform \( h(t) \) \( \rightarrow H(s) \)
2) Transform \( x(t) \) \( \rightarrow X(s) \)
3) Multiply \( X(s) \) \( \times H(s) = Y(s) \)
4) Inverse transform \( Y(s) \) \( \rightarrow y(t) \)

**Debugging a filter** whatever is appropriate

dist min\( \leq [000,1000] \) power of 2
Digital Filter Coefficient design

- Butterworth
- Chebyshew

Parks-McClellan Method

- Published in early 70's
- Iterative algorithm
- Computationally efficient
- Works by inputting
  1) length of filter - $h(t)$

  2) frequency/magnitude pairs
- See Oppen & Sommer
  - Given in dB
  - Min/max attenuation/ripple

Over frequency regions

Ex: Low-pass filter
- Max ripple ± 4 dB in passband
- Sampling frequency 100 MHz
- Pass band is DC - 12.5 MHz
- Min attenuation 22 dB 19 - 50 MHz
\[
\begin{align*}
100 \text{ MHz} & \rightarrow 2\pi \\
50 \text{ MHz} & \rightarrow \pi \\
12.5 \text{ MHz} & \rightarrow \frac{\pi}{4} = 0.25\pi \\
14 \text{ MHz} & \rightarrow 0.38\pi \\
\end{align*}
\]

\[
\text{freq} = [0, 0.25, 0.38, 1]
\]

\[
\text{mag} = [1, 1, 0, 0]
\]

don't care

\[
\text{remez}( )
\]

help remez

first argument = # taps (coeffs) - 1

\[
\text{coeffs} = \text{remez} (20, [0, 0.25, 0.38, 1], [1, 1, 0, 0]); \quad \% 21 \text{tap}
\]

To see coeffs,

\[
\text{stem}([-10:10, \text{coeffs}]);
\]
Solving the frequency response of filters

I. Method I - freqz() in MATLAB

- exact calc.
- very fast

\[ |A(f)| \]

\[ h(t) \]

\[ \text{freqz()} \]

Ex: using FFT
II. Method II

1) Make a "flat" "white" spectrum input signal

2) Send signal into the filter + look at output spectrum

   - requires many samples for accurate output

   - slower

   + sometimes the only way

   • HW rounding

   • signal saturation

   • matlab code

     \[ \text{in} = \text{rand}(1, 100000) - 0.5 \; \]

     \[ \text{out} = \text{conv} \left( \text{wells}, \text{in} \right) + 0.25 \; \] 90 LSB bias

     \[ \text{abs} \left( \text{fft} \left( \text{out} \right) \right) \]

     \[ \text{psd} \left( \text{out} \right) \]

     \[ \text{spectrum} \left( \text{out} \right) \]