Creative Computing II

Christophe Rhodes
c.rhodes@gold.ac.uk

Autumn 2010, Wednesdays:
10:00–12:00: RHB307 & 14:00–16:00: WB316
Winter 2011, Wednesdays:
10:00–12:00: RHB307 & 14:00–16:00: WB316

Linear systems have the property that *superposition* and *scaling* of their input signals yield the corresponding scaled superposition of their outputs.

Systems

Review: Linear Systems

Linear systems have the property that *superposition* and *scaling* of their input signals yield the corresponding scaled superposition of their outputs.

For any input signals x_1 and x_2, if

$$y_1 = H\{x_1\}$$

and

$$y_2 = H\{x_2\},$$

a system H is linear if

$$H\{\alpha x_1 + \beta x_2\} = \alpha y_1 + \beta y_2$$

Systems

Review: Linear Systems

Linear systems have the property that *superposition* and *scaling* of their input signals yield the corresponding scaled superposition of their outputs.

For any input signals x_1 and x_2, if

$$y_1 = H\{x_1\}$$

and

$$y_2 = H\{x_2\},$$

a system H is linear if

$$H\{\alpha x_1 + \beta x_2\} = \alpha y_1 + \beta y_2$$

(Almost) all systems in the real world are linear systems *for small enough signals*.

Time-invariant systems have the property that the output signal of the system for a given input signal does not depend explicitly on absolute time.

For any input signal x with $y = H\{x\}$, the system H is time-invariant if

$$H\{T_\delta\{x\}\} = T_\delta\{y\}$$

Where T_δ is a delay system for arbitrary delay.

Time-invariant systems have the property that the output signal of the system for a given input signal does not depend explicitly on absolute time.

For any input signal x with $y = H\{x\}$, the system H is time-invariant if

$$H\{T_\delta\{x\}\} = T_\delta\{y\}$$

Where T_δ is a delay system for arbitrary delay.
Many systems of interest in the real world are time-invariant systems.

Systems

Review: Convolution

The convolution operation is notated

$$y(t) = (h * x)(t)$$

In discrete time we define the operation as

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

The convolution operation is notated

$$y(t) = (h * x)(t)$$

In discrete time we define the operation as

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

Why convolution?

- implementation of LTI systems!

The output y of a system H for an input signal x is the convolution of the input and the impulse response of the system.

Systems

Review: Fourier Transforms and Convolution

Previously:

- system H response to signal x is $h * x$;
- direct convolution computation has complexity $O(N^2)$.

Previously:

- system H response to signal x is $h * x$;
- direct convolution computation has complexity $O(N^2)$.

Fourier Transform of a convolution is the product of the Fourier Transforms:

$$\mathcal{F}(h * x) = \mathcal{F}(h) \times \mathcal{F}(x)$$

so system output for signal x is

$$\mathcal{F}^{-1}(\mathcal{F}(h) \times \mathcal{F}(x))$$

Systems

Review: Fourier Transforms and Convolution

Previously:

- system H response to signal x is $h * x$;
- direct convolution computation has complexity $O(N^2)$.

Fourier Transform of a convolution is the product of the Fourier Transforms:

$$\mathcal{F}(h * x) = \mathcal{F}(h) \times \mathcal{F}(x)$$

so system output for signal x is

$$\mathcal{F}^{-1}(\mathcal{F}(h) \times \mathcal{F}(x))$$

Octave:

- `ifft(fft(h,length([h x])-1).*fft(x,length([h x])-1))`

$$y = \mathcal{F}^{-1}(\mathcal{F}(h) \times \mathcal{F}(x))$$

so

$$\mathcal{F}(y) = \mathcal{F}(h) \times \mathcal{F}(x)$$

- $\mathcal{F}(h)$ is the **frequency response** of the system.
- the frequency spectrum of the output signal is the product of the spectrum of the input and the frequency response of the system.

Filtering

Application of systems to multimedia.

- audio:
  - mixing and EQ;
  - acoustics;
  - sound effects;
  - subtractive synthesis.
- image:
  - various effects
    - blurring;
    - edge detection;
    - sharpening;
    - ...

Audio Filtering

Mixing desks

Wikimedia Commons (user Binksternet)
Public Domain

Audio Filtering

Mixing desks

Mixing ‘console’ or ‘desk’:

- gain controls;
- channel *equalizers*;
- (and other functionality).

Gain control:

- controls proportion of channel in the entire *mix*;
- usually a slider (‘fader’) controlling a variable resistor (‘pot’).

Audio Filtering

Mixing desks

Mixing ‘console’ or ‘desk’:

- gain controls;
- channel *equalizers*;
- (and other functionality).

Channel equalizer:

- per-channel controls for gain in particular frequency ranges:
  - bass: low-frequency;
  - mid-range;
  - treble: high-frequency;
- digital mixing consoles use discrete LTI systems

Audio Filtering

Mixing desks

Mixing 'console' or 'desk':

- gain controls;
- channel *equalizers*;
- (and other functionality).

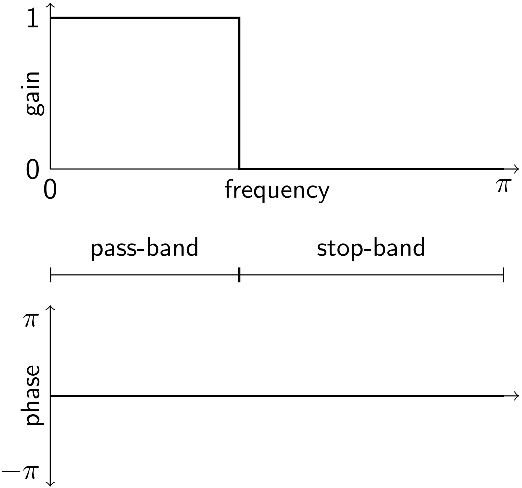
Other functionality:

- pan and balance;
- submix routing;
- talkback;
- external effects.

Audio Filtering

Finite Impulse Response Filters

Ideal **low-pass** filter frequency response:



Ideal filters are not possible.

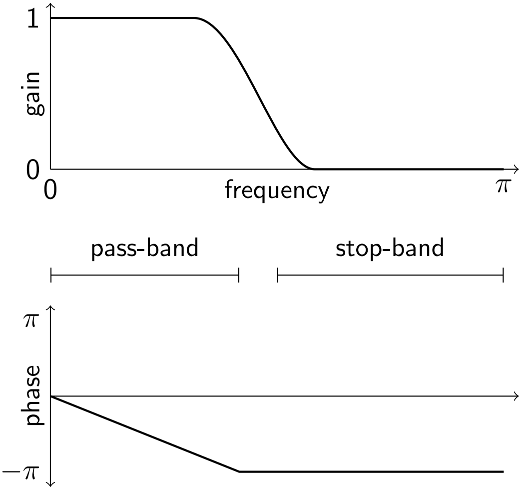
- finiteness of impulse-response;
- Heisenberg uncertainty;

Should we just give up?

Audio Filtering

Finite Impulse Response Filters

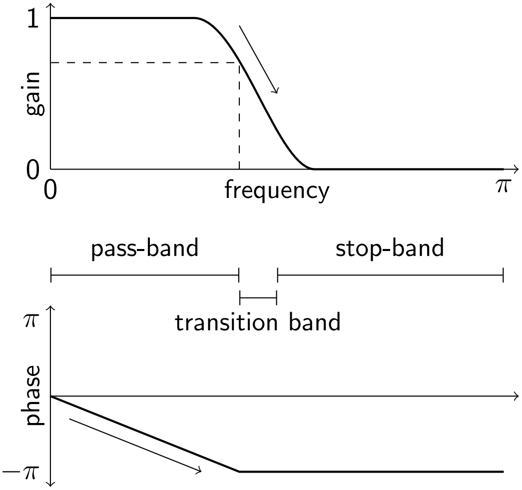
Practical **low-pass** filter frequency response:



Audio Filtering

Finite Impulse Response Filters

Practical **low-pass** filter frequency response:



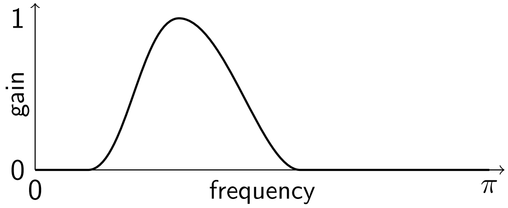
Practical **high-pass** filter frequency response:

- gain of close to 1 for high frequencies;
- gain of $\frac{1}{\sqrt{2}}$ at cutoff frequency;
- rapid decline in gain at frequencies lower than cutoff;
- linear phase delay in pass-band region.

Audio Filtering

Finite Impulse Response Filters

Band-pass filter: allow through only frequencies within a certain range.



Practical band-pass filter frequency response:

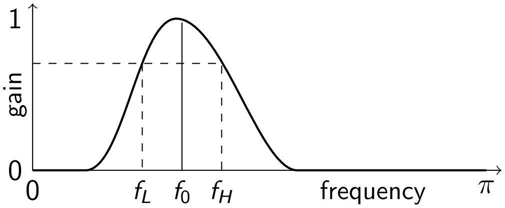
- gain of close to 1 in pass-band region;
- gain of $\frac{1}{\sqrt{2}}$ at cutoff frequencies (lower and upper);
- rapid decline in gain at frequencies outside pass-band;
- linear phase delay in pass-band region.

Band-pass filter:

- Combination (convolution) of low-pass and high-pass filter (with overlapping pass-band regions):
- difference between upper and lower cutoffs: **bandwidth**;
- ratio between centre frequency and bandwidth: **quality factor**;

Audio Filtering

Finite Impulse Response Filters



$$B = f_H - f_L = \Delta f$$

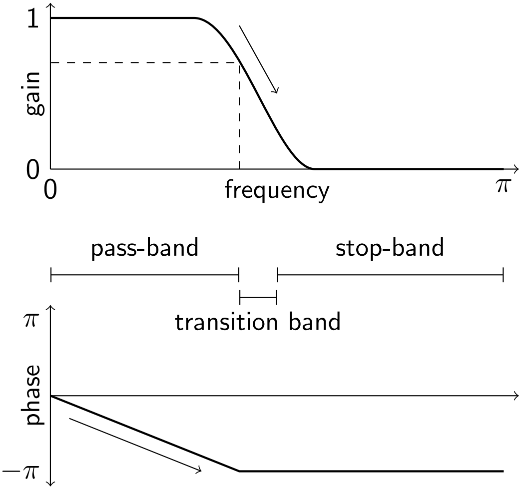
and

$$Q = \frac{f_0}{B}$$

Audio Filtering

Finite Impulse Response Filters

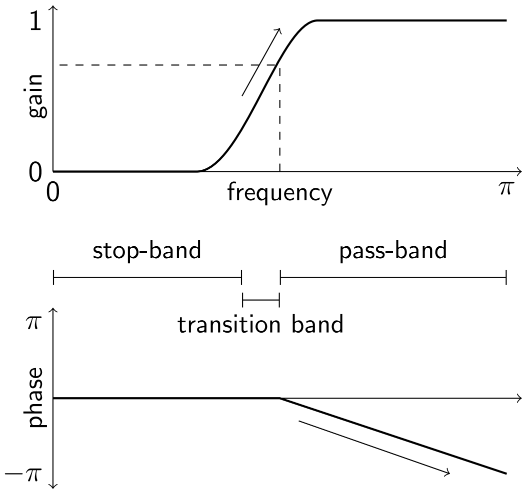
Practical **low-pass** filter frequency response:



Audio Filtering

Finite Impulse Response Filters

Practical **high-pass** filter frequency response:



Audio Filtering

Finite Impulse Response Filters

Practical **low-pass** filter frequency response:

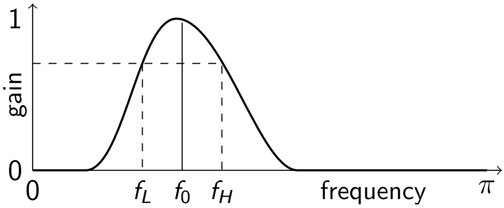
- gain of close to 1 for low frequencies;
- gain of $\frac{1}{\sqrt{2}}$ at cutoff frequency;
- rapid decline in gain at frequencies higher than cutoff;
- linear phase delay in pass-band region.

Practical **high-pass** filter frequency response:

- gain of close to 1 for high frequencies;
- gain of $\frac{1}{\sqrt{2}}$ at cutoff frequency;
- rapid decline in gain at frequencies lower than cutoff;
- linear phase delay in pass-band region.

Audio Filtering

Finite Impulse Response Filters



$$B = f_H - f_L = \Delta f$$

and

$$Q = \frac{f_0}{B}$$

Audio Filtering

Finite Impulse Response Filters

Octave:

- filter construction with `fir1` function:
    - `n`: **order** parameter;
    - `w`: band edges;
    - `type`, `window`, `scale` parameters;
- visualization with `freqz` function;
- application with `filter` function.
    - `y = filter(fir1(...), 1, x)`
    - or use `conv`

Audio Filtering

Subtractive Synthesis

e.g. Moog synthesizers

- Start with a rich waveform:

```
f = zeros(1,44100);
f(56:55:22050) = 1;
f(44100:-1:22051) = f(1:22050);
x = real(ifft(f));
```

- Apply a filter to the waveform:

```
y = conv(h,x);
```

- Play the filtered waveform:

```
sound(y, 44100)
```

(cf. **additive synthesis**: constructing waveform from explicit addition of partials)

Audio Filtering

Echo and Reverb

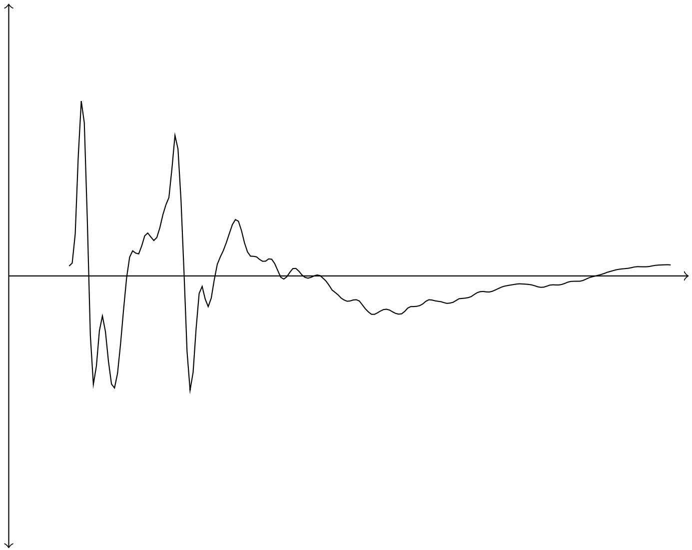
Rooms are systems too. Their impulse response can be categorized into two parts:

- echo:
  - few, discrete impulses at particular times;
  - caused by first reflections of sound off one or two surfaces;
  - typical timescale: $\sim 0.1s$.
- reverb:
  - noisy, decaying waveform;
  - caused by superimposed echoes of echoes of echoes (of echos...);
  - typical timescale: $\sim$ 1s–10s.

Audio Filtering

Echo and Reverb

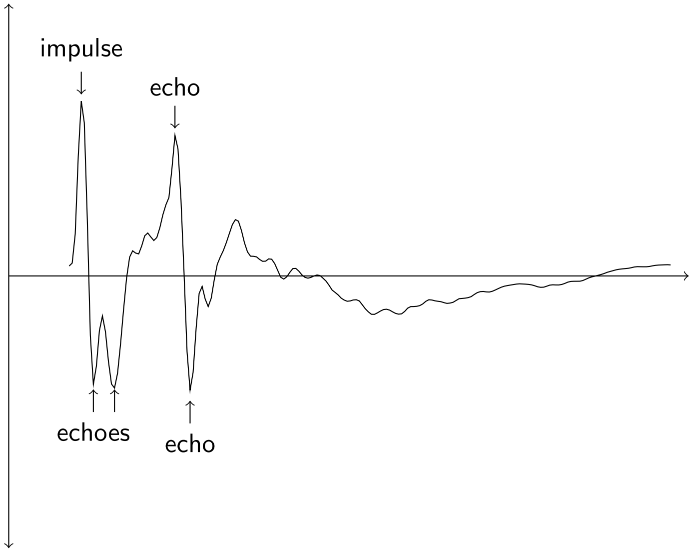
Echo:



Audio Filtering

Echo and Reverb

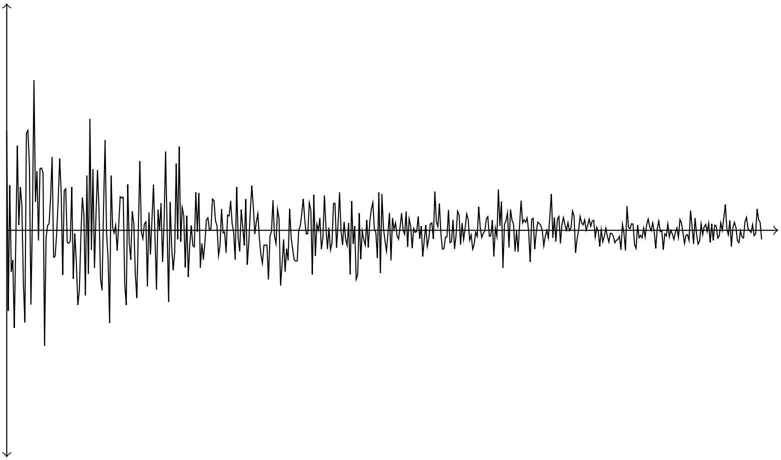
Echo:



Audio Filtering

Echo and Reverb

Reverb:



Resampling: changing the sample rate of a discrete-time signal (while preserving its meaning).

Applications:

- Applying a filter to a signal with a different sample rate;
- Resampling synthesis: pitch shifting.

Octave: `resample` operator.

- `x`: signal to resample;
- `p`: interpolation factor;
- `q`: decimation factor.

Net effect is to transorm signal sampled at frequency f into the same signal sampled at frequency $\frac{p}{q}f$.