Time domain representation

- given frequency of $F_s = 44.1$ kHz & 441,000 samples, what is the duration? #samples/F_s = 10,000 ns = 10 s
- audio typically normalized to $[-1;1]$ range.

Discrete time linear filters

$x(n) \rightarrow \overset{h(n)}{\underset{H(z)}{\rightarrow}} \rightarrow y(n) = h(n) * x(n)$

input; filter specified by IR and transfer function; ↑ output obtained by convolution

Energy & power signals

- energy of a signal $x(n)$ is $E = \sum_{-\infty}^{\infty} |x(n)|^2$
  if E is finite $(0 < E < \infty)$, then $x(n)$ is an energy signal.
- many signals with infinite energy have a finite average power.
  Power of a discrete-time signal $x(n)$ is $P = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} |x(n)|^2$.
  (i.e. power is the average energy). if E is finite, then $P = 0$.
  if E is infinite, then power may or may not be infinite.
  if P is finite & non-zero, the signal is called a power signal.
- for example the unit step sequence $u(n) = \begin{cases} 1 & \text{if } n \geq 0 \\ 0 & \text{if } n < 0 \end{cases}$ has infinite energy and power $P = \frac{1}{2N+1} \sum_{n=0}^{N} u(n)^2 = \frac{N+1}{2N+1} = \frac{1+1/N}{2+1/N} = 1/2$ so $u(n)$ is a power signal.
- note: if a signal is periodic, its power is finite & equals the avg. power over a single period.

Block diagrams

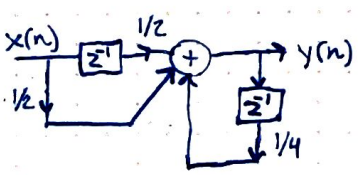
adder: $x_1(n)$, $x_2(n)$ → $y(n) = x_1(n) + x_2(n)$

multiplier: $x_1(n)$, $x_2(n)$ → $y(n) = x_1(n) \cdot x_2(n)$

unit delay element: $x(n) \rightarrow z^{-1} \rightarrow y(n) = x(n-1)$

constant multiplier: $x(n) \xrightarrow{a} y(n) = a\,x(n)$

- (example)
  sketch the diagram for:
  $y(n) = \frac{1}{4} y(n-1) + \frac{1}{2} x(n) + \frac{1}{2} x(n-1)$

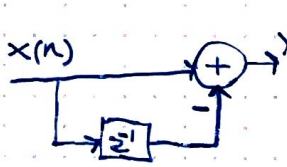


In reality, one might use two adders instead of one.

Dynamic systems

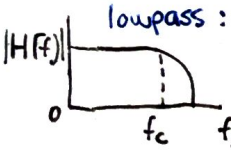
A discrete-time system is dynamic if its outputs depend on the past (has memory). If an output is dependent only on the input at that given timestep, it is static.

(example) differentiator: $x(n)$ → $y(n) = x(n) - x(n-1)$

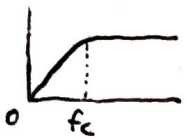


Basic Filters

examples of basic filters that attenuate select frequencies.



lowpass: attenuate (lower) frequencies higher than f_c. there may be resonance which amplifies frequencies around f_c.



highpass: attenuate frequencies lower than f_c. resonance still applies.

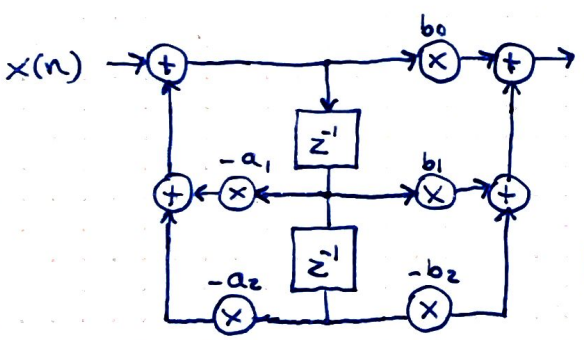
bandpass: a combination of lowpass/highpass. alternatively, a band reject filter can attenuate only the frequences that would pass in BP.

Canonical filters

- a system is canonical when the number of delays in a block diagram representation equals the order of that system (it is minimized).
- the canonical second-order filter is described by the difference equations:

$$x_h(n) = x(n) - a_1 x_h(n-1) - a_2 x_h(n-2)$$
$$y(n) = b_0 x_h(n) + b_1 x_h(n-1) + b_2 x_h(n-2)$$

and the block diagram



which leads to the following transfer function:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

- if we simply set $a_2 = b_2 = 0$, the filter reduces to a first-order filter. depepding on the coeficients a_1 & b_1 the first-order canonical filter can be used to implement a lowpass or highpass filter.
- implementing lowpass/highpass filters using second-order filters is also possible, but it requires another parameter Q that controls the level of resonance. For $Q = 1/\sqrt{2}$ the filter is maximally flat.
- a first-order allpass filter is given by the transfer function:

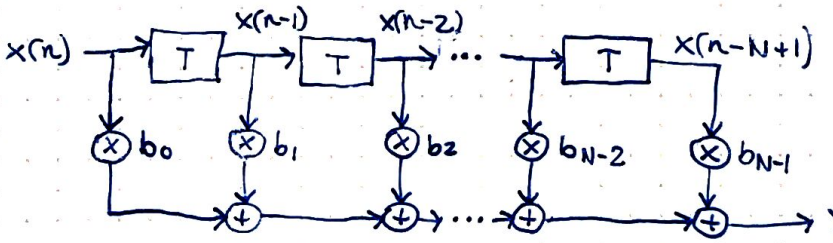
$$A(z) = \frac{z^{-1} + c}{1 + cz^{-1}} \quad \text{where} \quad c = \frac{\tan(\pi \cdot f_c/f_s) - 1}{\tan(\pi \cdot f_c/f_s) + 1}$$

$$H(z) = \tfrac{1}{2}(1 \pm A(z)) \quad \text{where } + \Rightarrow \text{LP} \ \& \ - \Rightarrow \text{HP}$$

a LP/HP filter can be designed using an allpass filter.

FIR filters

- filters with feedback loops can cause an input sample to excite an output signal for a fairly long time depending on the parameters. such filters are called infinite impulse response (IIR) filters.
- filters without feedback loops are called finite impulse response (FIR) filters, and they guarantee for a unit impulse to only last for a fixed period of time. they typically require higher filter orders and thus more computing power as compared to IIR structures. when FIR filters are implemented by fast convolutions (by FFT) they can however achieve comparable performance (but requires a fixed filter, e.g. fixed frequency bands).
- FIR filters are used in cases where the impulse response of a real system has been measured and needs to be simulated.
- an example of a FIR filter is shown below:



FIR filter

which has the difference equation:

$$y(n) = \sum_{i=0}^{N-1} b_i \cdot x(n-i) = b_0 x(n) + b_1 x(n-1) + \cdots + b_{N-1} x(n-N+1)$$

an is a weighted sum of the input samples.

if the input sample is the unit impulse $\delta(n) = \begin{cases} 1 & \text{if } n=0 \\ 0 & \text{if } n \neq 0 \end{cases}$ we get the impulse response of the system: $h(n) = \sum_{i=0}^{N-1} b_i \cdot \delta(n-i) = b_n$

which has the Z-transform (transfer function):

$$H(z) = \sum_{i=0}^{N-1} b_i \cdot z^{-i}$$

(but more about the Z-transform later)

- an IIR filter typically has both a numerator & denominator in the transfer function (like on the previous page)

Z-transform

- the z-transform of a discrete-time signal $x(n)$ is a power signal:

$$X(z) \equiv \sum_{n=-\infty}^{\infty} x(n) \cdot z^{-n} \equiv Z\{x(n)\}$$

where z is a complex variable. since it is an infinite power series, it exists only for those values of z for which the series converges.

- (examples)

$x_1(n) = \{1, 2, \underset{\downarrow}{5}, 7, 0, 1\} \Rightarrow X_1(z) = 1 + 2z^{-1} + 5z^{-2} + 7z^{-3} + z^{-5}$

$x_2(n) = \{1, 2, \underset{\downarrow}{5}, 7, 0, 1\} \Rightarrow x_2(z) = z^2 + 2z + 5 + 7z^{-1} + z^{-3}$

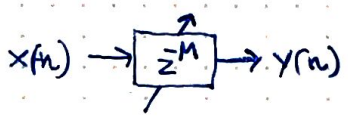
- from above example, it is easy to see that the series converges for all points in the z-plane (except from $z=0$ or $z=\infty$) as long as the input signal has a finite duration.

- while the Fourier Transform is the frequency representation of a continous function (of which Laplace can be applied to solve differential equations), the Discrete-Time Fourier Transform is the frequency representation of a discrete function (of which z-transform can be used to solve differenceequations).

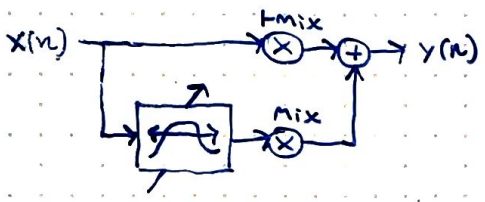
- important property: convolution in time domain equals multiplication in the transformed domain (whether it is z-transform or DTFT).

Lecture 01: sound effects

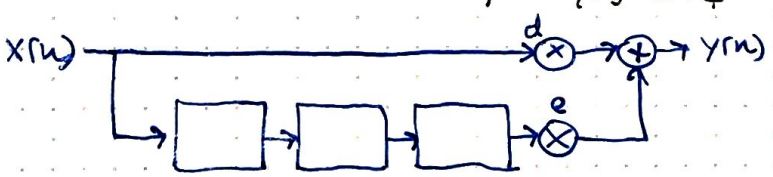
- **Vibrato** is a periodical pitch variation created by periodically varying delay:



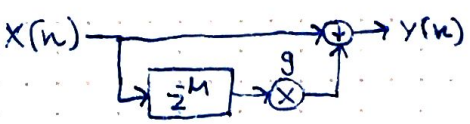
- **wah-wah** adds variable boost for certain frequencies by changing the center band-pass filter:



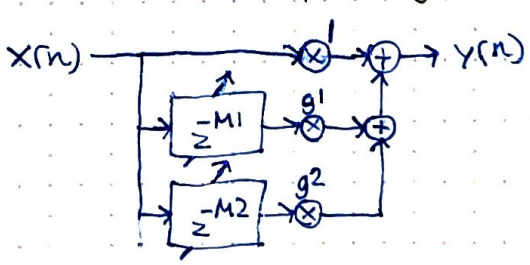
- **phaser** adds varying phase effect by mixing the signal with a series of notch filters with slowly varying frequencies:



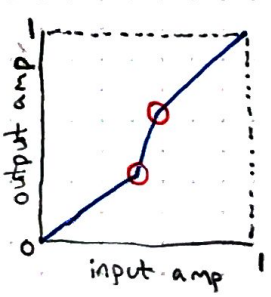
- **echo** is added by mixing with a time delayed (>50 ms) version:



- **chorous** (many instruments playing the same but with slight time/pitch variation). we obtain this by adding randomly varied and delayed versions (10-25ms):



- **dynamic range compression** changes loudness:



Lecture 02: noise removal (phase vocoder + STFT)

- **vocoder** = voice encoder is an analysis/synthesis system to reproduce speech.
- **phase vocoder** is a vocoder with added computation of time-varying frequency
- **Discrete Fourier Transform**

  analysis: $X(k) = \sum_{n=0}^{N-1} x_p(n)\, e^{-j\frac{2\pi}{N}k\cdot n}$

  synthesis: $x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\, e^{j\frac{2\pi}{N}k\cdot n}$

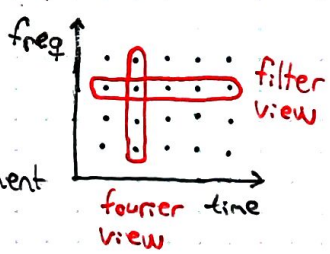
  k corresponds to frequency kF_s/N.
- **Short-time Fourier Transform** $X(n,k) = \sum_{m=-\infty}^{\infty} x(m)\cdot h(n-m)\cdot e^{-j\frac{2\pi}{N}k\cdot m}$

  where $h(n-m)$ is a time window.
- spectral shape of windows:

rect. bartlett hann hamming

↑ we typically use this one

- **phase vocoder** is an algorithm that seperates temporal & spectral information of a signal. this can be acheived either by filterbanks (heterodyning) or Fourier.
- the Fourier interpretation is by **discrete STFT**: computing the Fourier series of small windows at a time. if windows are too small, however, it will cause smearing in the spectrum because each component will include some energy from bins nearby.



- the **FFT** algorithm runs in $O(n \lg n)$ time as opposed to filterbanks in $O(n^2)$ time.
- **time scaling** by phase vocoder is to space windows further apart when computing the inverse. to avoid changes in frequency, phase is rescaled by the same factor as the time change, such that e.g. a phase change of 45° over an interval N is now a 90° change over $2N$ time.
- **pitch change**: do time scaling by the desired change factor and simply play back sound using the original sample rate. 1.5x slower.

(ex) if analysis hop factor is $R_a = 256$ & inverse is $R_b = 384$, new signal is

Lecture 03: Wiener filters

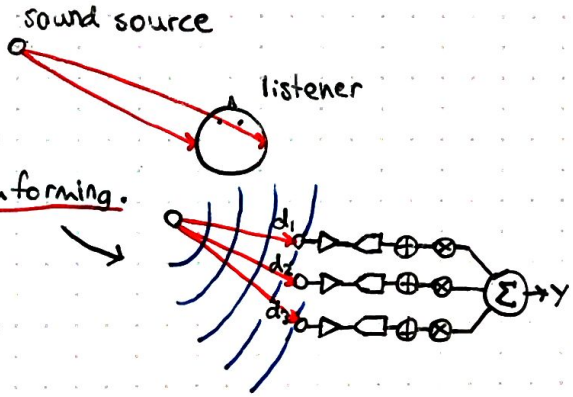
- can be implemented as an IIR or FIR filter, but FIR is simpler so we focus on that in this lecture.
- the filter takes as input a signal $y(n)$ and produces and output $\hat{x}(n)$, which is the least mean square error estimate of a desired target signal $x(n)$. filter input-output relation is given by:

$$\hat{x}(n) = \sum_{k=0}^{P-1} w_k y(n-k) = w^T y$$

where y is the input and w is the Wiener filter coefficient vector that we try to optimize. the error is then $e(n) = x(n) - \hat{x}(n) = x - yw$.

- if we have as many coefficients as input samples, we can solve it for $e=0$. in reality, we typically have way more samples than coefficients, so we instead try to minimize the average error (e.g. L2-norm), i.e. $E[e^2(n)]$.
- Wiener filters assume that the spectra of the signal & noise are linearly seperable, but that is usually not the case. therefore, the practical use of Wiener filters for audio is limited. furthermore, it is also assumed that the problem is stationary (e.g. white noise).

Lecture 08: source seperation

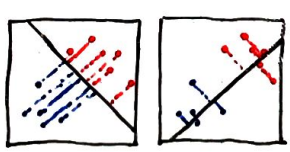


- spatial information can be extracted by beamforming.
- Linear mixing ICA model:
  if we assume that sources are uncorrelated,
  i.e. $p(\underline{s}) = p(s_1, s_2, \dots, s_N) = p(s_1)\,p(s_2)\dots p(s_N)$,
  and that sources are not gaussian distributed we can do ICA.

$$\underline{x}(n) = A[\underline{s}(n)] + \underline{\epsilon}(n), \quad \underline{u}(n) = W[\underline{x}(n)] = W[A[\underline{s}(n)] + \epsilon(n)] \approx \underline{s}(n)$$

  where $\underline{x}(n)$ is the signal received by the microphone and $A[\cdot]$ is the mixing procedure of sources. we seek to determine an operator $W[\cdot]$ that can invert the mixing operator $A[\cdot]$ (resulting in $u(n)$ as shown above.

- principal component analysis
  recall that a basis of a subspace V is a set of vectors $v_1, v_2, \dots, v_n$ s.t. $\text{span}(v_1, v_2, \dots, v_n) = V$, i.e. the vectors $v_1, v_2, \dots, v_n$ are sufficient to generate all of V by linear combination. In PCA we want to find the orthogonal basis that best separates data distributions, i.e. accounts for the most variance:



(ex) principal components of a 2D space onto a 1D space. the right-hand example clearly separates the best & thus also preserves information the best.

- independent component analysis tries to perfom a linear transformation that best seperates a number of observable variables into independent hidden variables, i.e. for observables $[x_1, x_2, \dots, x_N]^T$ we want to estimate a matrix W that generates hidden variables $[y_1, y_2, \dots, y_N]$ s.t. they are entirely independent (orthogonal) $I(y_i, y_j) = 0$ and $I(\underline{y}, \underline{x})$ is maximized by maximum likelihood. (I: mutual information)

  $\Rightarrow I(\underline{y}, \underline{x})$ maximized means that mutual information is shared maximally s.t. we can reconstruct x from y with minimal error. the algorithm is as follows:

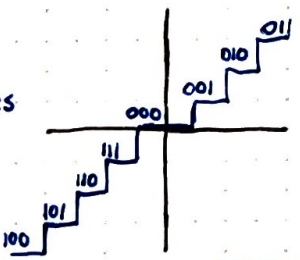
1) calculate $\underline{u}(n) = W\underline{x}(n)$ where $\underline{x}$ is our training vector. (we have N_s such vectors)
2) calculate $\Delta W = \left(I + \frac{1}{N_s}\sum_{n=1}^{N_s} \phi(\underline{u}(n)) \cdot \underline{u}^T(n)\right) \cdot W$ (our gradient)
3) update W, i.e. $W \leftarrow W + \eta \cdot \Delta W$ where η is the learning rate
4) repeat untill convergence.

Lecture 09: audio encoding

NB: wave-files typically use 16 bits.

- bitrate is defined as bitrate = sampling rate · # quantization bits · #channels.
  for digital audio of 44.1 kHz, 16 bits & 2 channels that is 1.4 Mbit/second. (0.175 MB/s)
- we can do either lossless encoding such as FLAC or lossy (but perceptually transparent) encoding such as MPEG (we focus on this)
- example of quantization using 3 bits:

  $e_q(n) = x(n) - x_q(n)$ $\quad 2^3 = 8$ values



- sound pressure level is defined as

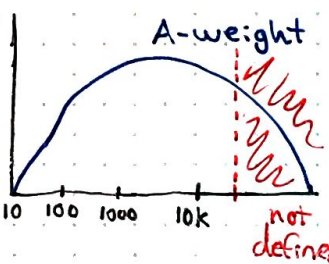
  $L_p = 20 \cdot \log_{10}\left(\frac{p}{p_{ref}}\right)$ dB

  where p_{ref} is a reference sound pressure, typically 20 μPa.
  we perceive loudness relatively to stimuli strength.

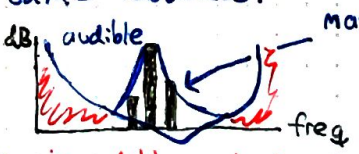
Sound	dB
whisper	30
office	45
traffic	80
concert	120
jet motor	140

pain threshold

- one way to measure loudness as perceived by humans is by A-weighting, typically used in environmental measurements to evaluate potential noise health effects, but is also used in e.g. speech recognition applications.

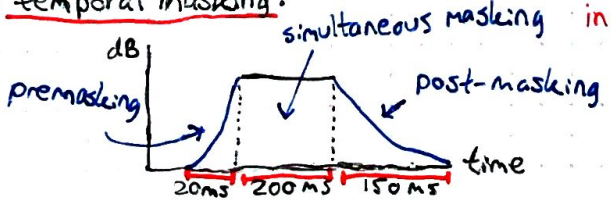


- it makes sense to have better sound quality (less quantization) in the loud frequencies and more quantization elsewhere... cue μ-law compr.
  Speech has a wide dynamic range, so it makes sense to reduce the dynamic range of the signal, then perform quantization & compute the inverse transform, thereby preserving speech better.
- critical band is the band of audio frequencies within which a second tone interferes with the perception of the first tone by auditory masking.
  (ex) the critical band of a 1 kHz noise is 160Hz → all noises within this band is perceived with the same loudness.
- simultaneous frequency masking:



inaudible: outside human hearing

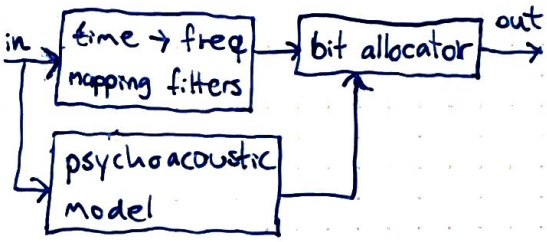
- temporal masking:



Lecture 09: audio encoding – continued

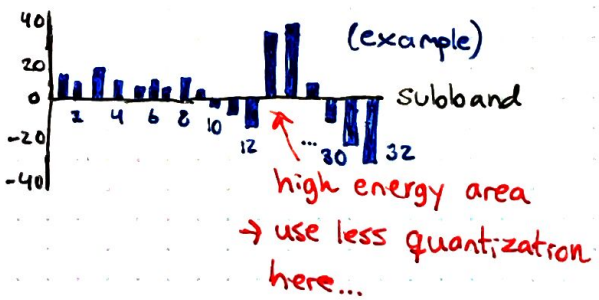
- **MPEG-I audio features:** three modes are available, each reducing the bitrate further at the expense of larger computational complexity.
  Layer 1) bitrate > 128 kbps, 2) ≈ 128 kbps, 3) ≈ 64 kbps

- encoder has the following architecture:



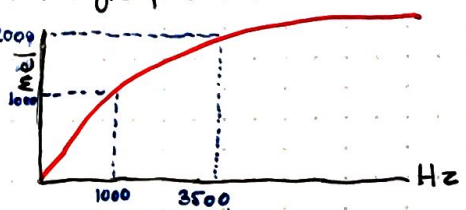
  - **filterbank** transforms samples to freq. domain in 32 subbands. not lossless!
  - **psychoacoustic model** computes the acoustically irrelevant parts of the signal.

  we do this by 1) calculating signal-to-mask ratio (SMR) for each subband
  SMR = signal energy / masking threshold
  idea is to use less bits in subbands with a low SMR (because we cannot hear the difference due to masking)



Lecture 10: sound search

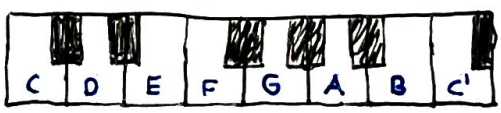
- there are two aspects of sound search: search engines & similarity metrics
- indexing: fingerprints. a song is characterized (indexing) (recommendation) by its spectrum. by storing points in spectrum above some quantile, a song can be indexed and searched for with high precision.
- Mel Frequency Cepstrum Coefficients:



we usually do the following to obtain MFCCs:

1) take the Fourier transform of a signal
2) map powers of the spectrum onto the non-linear mel-scale
3) take the logs of the powers at each mel frequency
4) take the discrete cosine transform (DCT) as if it was a signal
5) the MFCCs are the amplitudes of the resulting spectrum

where the DCT is similar to discrete Fourier transform but only produces real values. as we transform twice, the MFCCs can be considered a "spectrum of a spectrum".

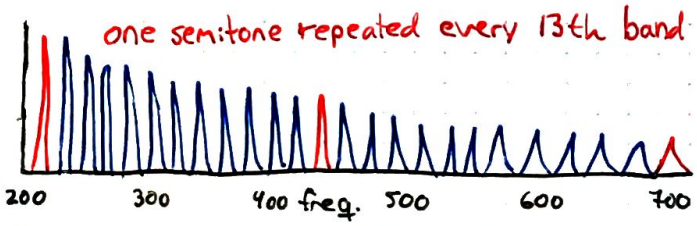


- Chroma Features:

we divide the frequency spectrum into 12 semitones:

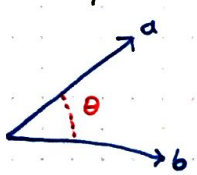
1) take the discrete Fourier transform
2) apply a log-freq filterbank
3) apply folding . . .

due to the log, resulting amplitudes are relative.



- cosine distance is a measure of dissimilarity between vectors:

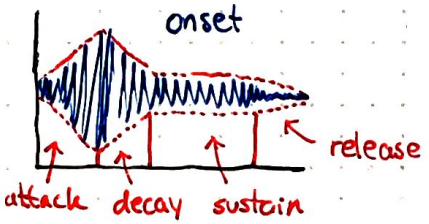
$$C_{cos}(a,b) = \frac{a^T b}{\|a\| \cdot \|b\|} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2} \cdot \sqrt{\sum_i b_i^2}}$$



$$\cos(\theta) = C_{cos}(a,b)$$

if the angle between two vectors is small in some n-dimensional space, they must be similar.

Lecture 11: beat detection



- what is an onset? →
- how can onsets be detected?

1) Lowpass filter to remove noise (preprocessing)
2) square signal to compute energy (opt. take log of energy for loudness)
3) find peaks in energy using some algorithm

- there are many more advanced techniques. one can analyse a signal across frequency bands, look for sudden changes in energy or build models for signal prediction (onset if actual signal differs a lot from pred.)
- an important observation: energy is concentrated in low frequencies but we are more interested in high-frequency changes = apply weights. (HF content)
- for peak detection, one wants to use adaptive thresholds.
- depending on the instrument, one feature may be better than the other.
- the tempo extraction technique applied in our study does the following:

1) compute STFT of signal
2) take the logarithm of the magnitude (power)
3) compute the discrete derivative to get a novelty curve representing sudden changes in energy.
4) we then do template matching by autocorrelation
5) each freq. bin is weighted by a log-normal dist. and maximum value of the weighted tempogram is then the tempo