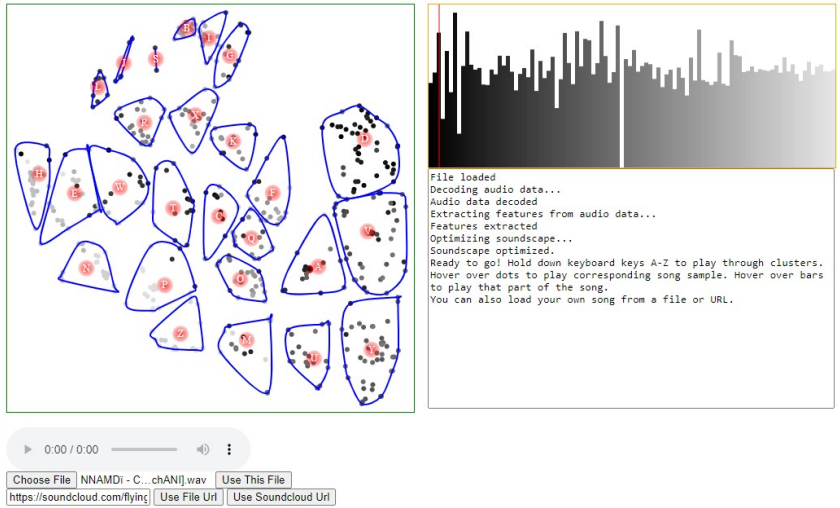
Durham University

Introduction to Music Computing

Audio representations of music data

Dr Eamonn Bell
eamonn.bell@durham.ac.uk

Motivating example



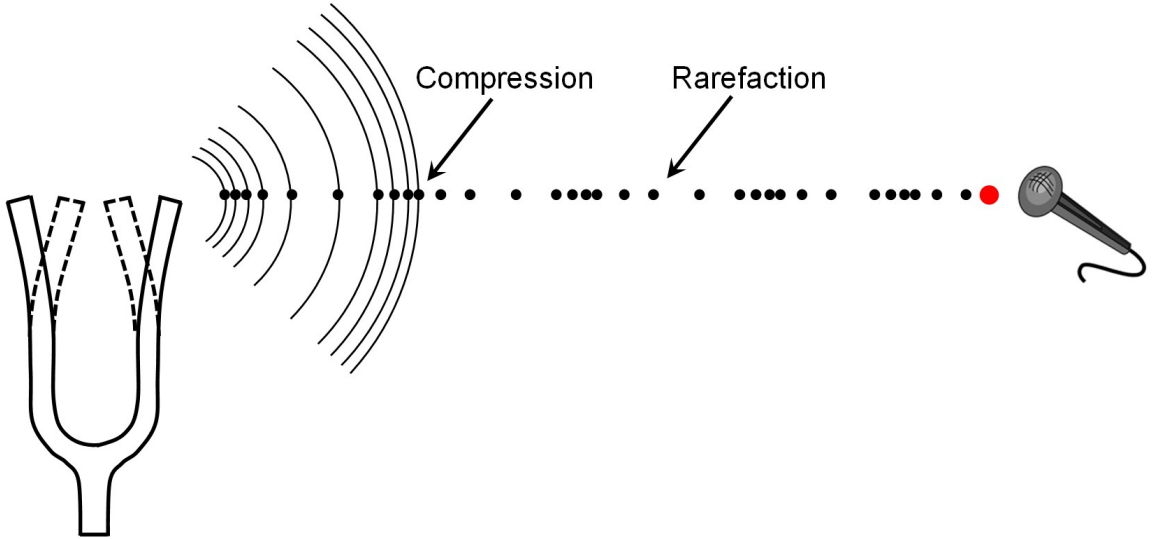
MusicMapper

https://fatsmcgee.github.io/MusicMappr/

Benjamin, Ethan, and Jaan Altosaar. “MusicMapper: Interactive 2D Representations of Music Samples for in-Browser Remixing and Exploration.” In Proceedings of the International Conference on New Interfaces for Musical Expression, edited by Edgar Berdahl and Jesse Allison, 325–26. Baton Rouge, Louisiana, USA: Louisiana State University, 2015. https://doi.org/10.5281/zenodo.1179018.

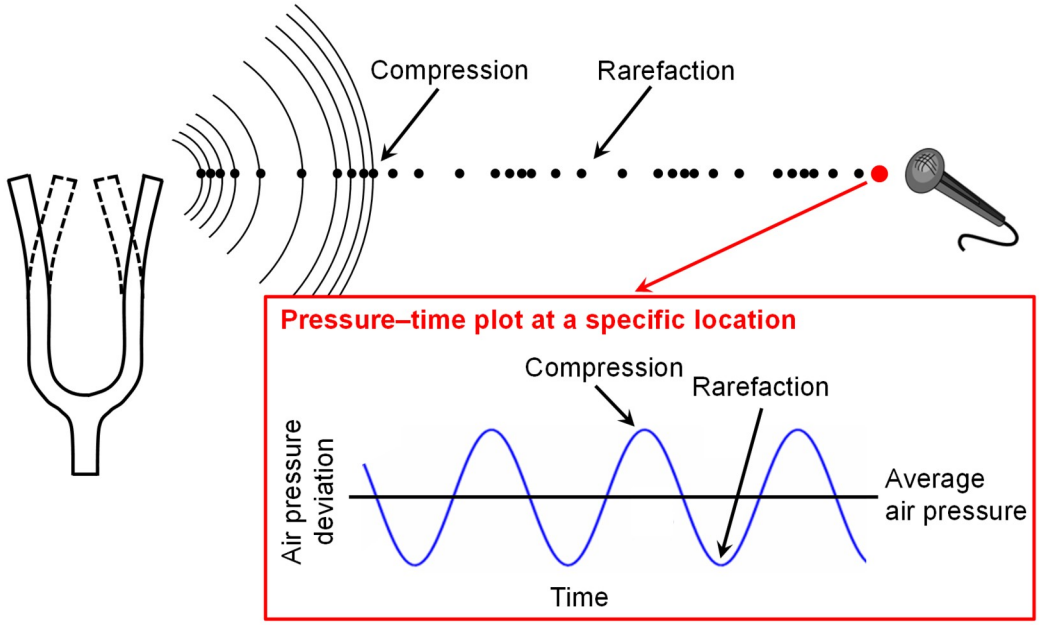
Durham University

Sound is moving air



[FMP21] Fig 1.17

Sound is moving air

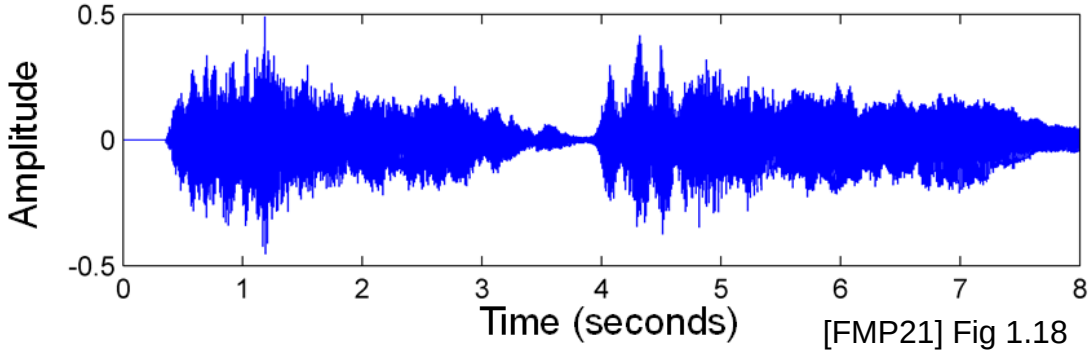


[FMP21] Fig 1.17

Waveforms

We can plot the intensity of the sound pressure (**amplitude**) as a function of **time**

This is called a **waveform** plot.

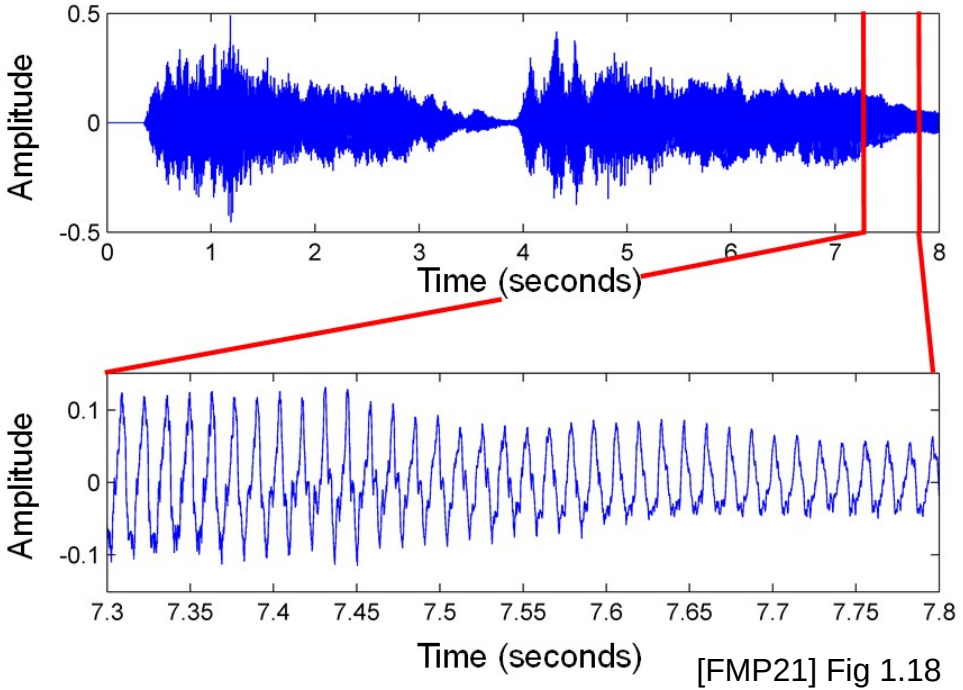


[FMP21] Fig 1.18

Music can be thought of as sound with structure

At the right scale, a repeating or **periodic** structure emerges

High and low pressure values repeat in an alternating and regular way

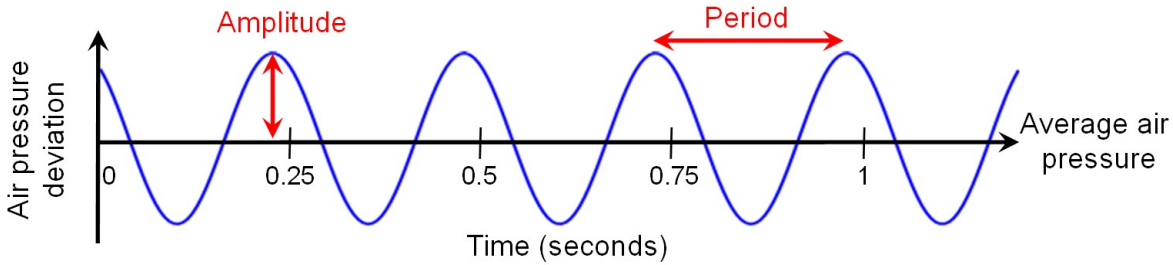


[FMP21] Fig 1.18

Simple sounds

Some simple sounds can be specified parametrically as a combination of

- **amplitude**
- **period**



[FMP21] Fig 1.19

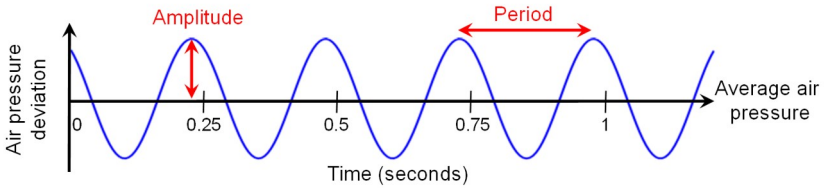
which, plugged into the appropriate mathematical function, can specify a signal $f(t)$ (where t is time)

Sinusoids

We call such a simple sound a **pure** or harmonic **tone**. It has a **sinusoidal** waveform, meaning it is well-modeled by the $\sin(\cdot)$ function from trigonometry

$$f : \mathbb{R} \to \mathbb{R}$$

$$f(t) := A\sin(2\pi(\omega t - \varphi))$$



- p : **period**, or time between two successive peaks
  - $\omega = 1/p$: **frequency** (measured in Hz – i.e. cycles per second or c.p.s.)
- A : **amplitude,** air pressure at peaks
- ϕ : **phase** (to be defined)

Pitch

A perceptual attribute, linked to how “high” or “low” we hear a sound to be

For **pure tones**:

- high ω means high pitch
- low ω means low pitch

The audible range for humans is between c. 20Hz and 20,000Hz (20 kHz)

For complex tones that have a prominent periodic component ([FMP21] Fig 1.18), this also holds

$\omega_A = 261.6$ (Hz)
$\omega_B = 440$

Pitch differences

More generally, two pairs of **simple sounds** seem equally “far apart” in **pitch** if the ratio of the **frequencies** of the **sinusoids** that generate them is identical

$\omega_A = 246.94$ (Hz)
$\omega_B = 311.13$

ω_{A} then ω_{B}

$\omega_{A'} = 392$
$\omega_{B'} = 493.88$

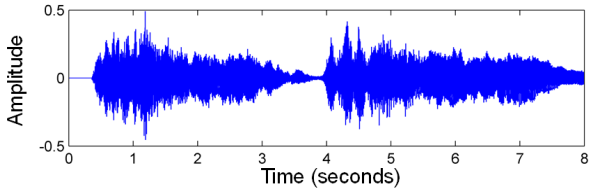
$\omega_{A'}$ then $\omega_{B'}$

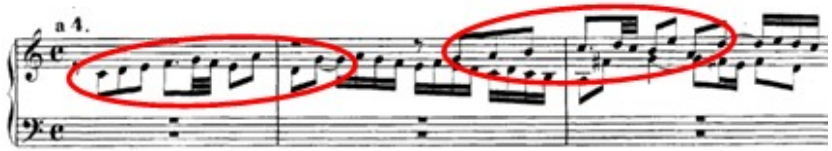
$\omega_A/\omega_B = \omega_A/\omega_B \approx 0.79$

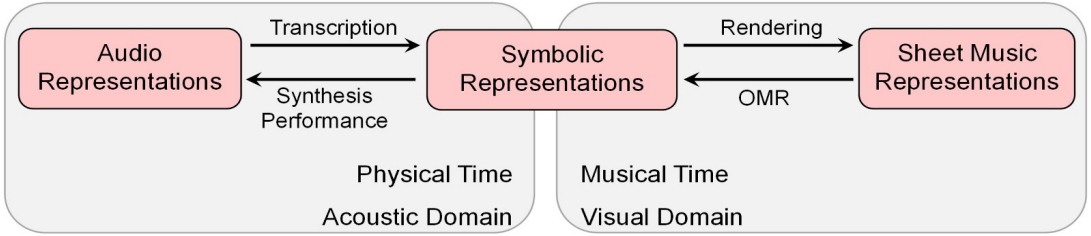
Many ways to represent music

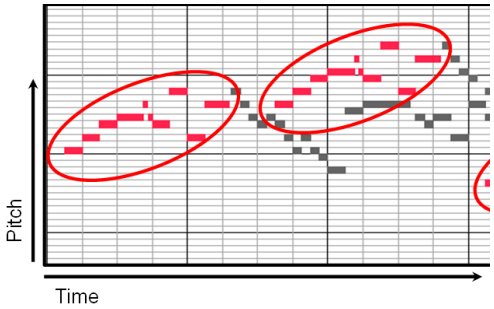
- **Audio**
  - the vibrations in the air that we (or the recording device) register
  - e.g. .wav, MP3, FLAC, vinyl LP
- **Symbolic**
  - lower-bandwidth computational representation
  - e.g. MIDI, MusicXML
- **Sheet Music**
  - a visual format used by musicians (a set of instructions)
  - e.g. printed sheet music (maybe as PDF), handwritten score, guitar tabs

Many ways to represent music



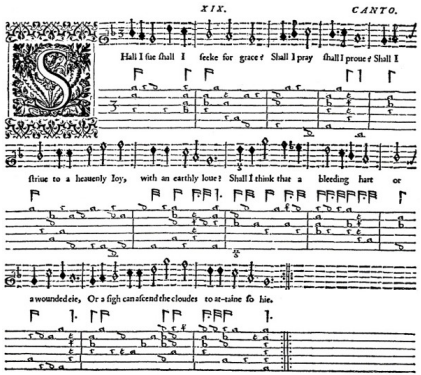


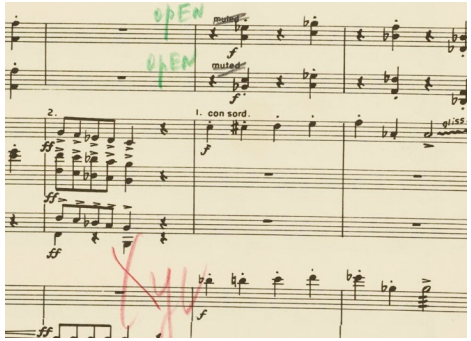




[FMP21] Figs 1.12, 1.18, 1.24

Durham University





(Clayton et al. 2020)

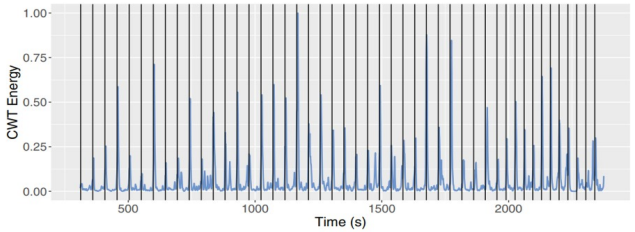


FIG. 10. Pairwise CWT Energy for the vocalist and tabla player in VK_Multani, khyal in vilambit ektal, from the NIR corpus, plotted over time (onsets of metrical cycles labeled via vertical lines).


```
X: 1
T: The Tarbolton
R: reel
M: 4/4
L: 1/8
K: Edor
D|Eeed e2 BA|GBAF GFEF|Dddc d2 AF|GBAG FDDF|
Eeed efge|fedf edBA|GABG FGA=c|BGAF GE E:|
gfef (3gfe be|gebe gfef|d2 fd adfd|ABAG FDDF|
(3GGG BG (3FFF AF|Eeef gfed|B2 dB AGFA|BGAF GE E:|
```

Activity at time of memory (open response)

Please describe the **memory** associated with this music. Please give as much detail as possible (such as **who** you were with, **where** you were, and **what** you were doing in the remembered event) (open response)

(Jakubowski and Ghosh 2021)

Durham University

Octaves

We say that two **simple sounds** are **octave-related** if the ratio of frequencies of the sinusoid that generates them is a power of 2

$\omega_A = 200$ (Hz)
$\omega_B = 300$
$\omega_C = 400$
$\omega_D = 3200$

Sounds that are octave-related sound very similar. When played simultaneously, listeners find them difficult to tell apart.

This suggests that **pitch perception is logarithmic in frequency**

+ means sum the signals

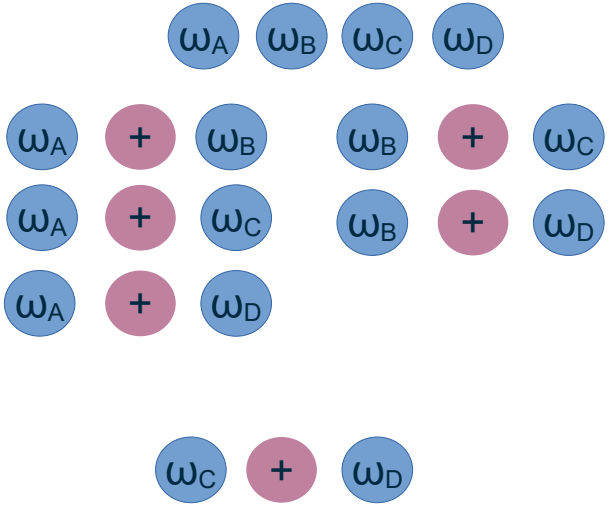
Octaves

We say that two **simple sounds** are **octave-related** if the ratio of frequencies of the sinusoid that generates them is a power of 2

$\omega_A = 200$ (Hz)
$\omega_B = 300$
$\omega_C = 400$
$\omega_D = 3200$

Sounds that are octave-related sound very similar. When played simultaneously, listeners find them difficult to tell apart.

This suggests that **pitch perception is logarithmic in frequency**



12-tone equal temperament

A conventional way to divide an octave into “steps” that are equally spaced perceptually. These steps are called **semitones**

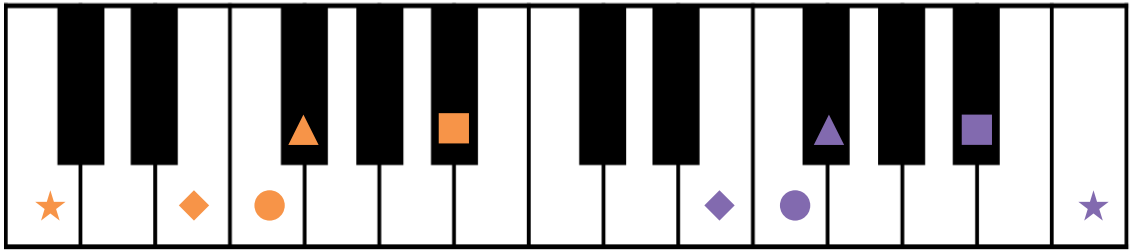
Two **semitones** make a **tone;** twelve **semitones** make an **octave**

Octave-related sounds sound similar

12-tone equal temperament

A conventional way to divide an octave into “steps” that are equally spaced perceptually. These steps are called **semitones**

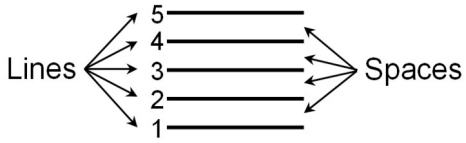
:= the steps modelled by adjacent keys on a piano



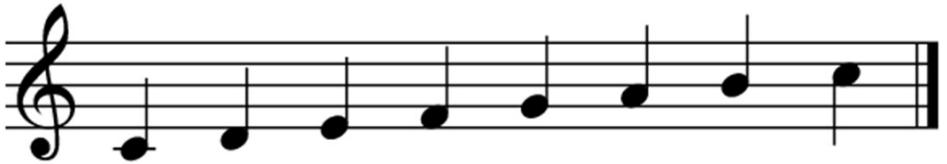
Two **semitones** make a **tone;** twelve **semitones** make an **octave**

Octave-related sounds sound similar

Pitch conventions



“staff notation”



[FMP21] Figs 1.4, 1.5 (adapted)

Scientific Pitch Notation (SPN)

A4 := **440Hz**

A represents the **pitch class**
4 represents the **octave**

A4

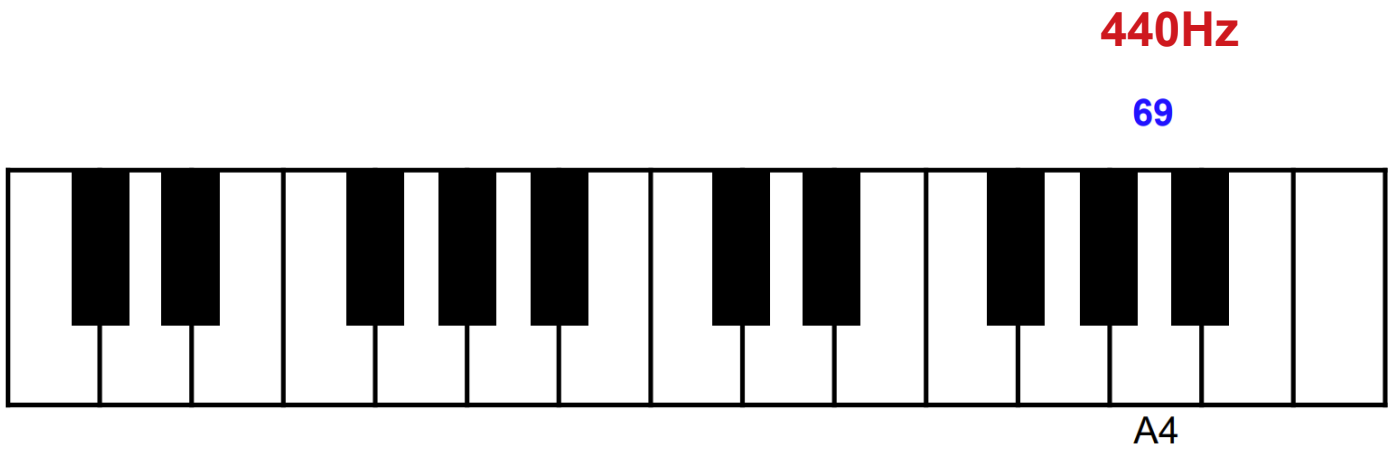
69

440Hz

MIDI note numbers (MNN)

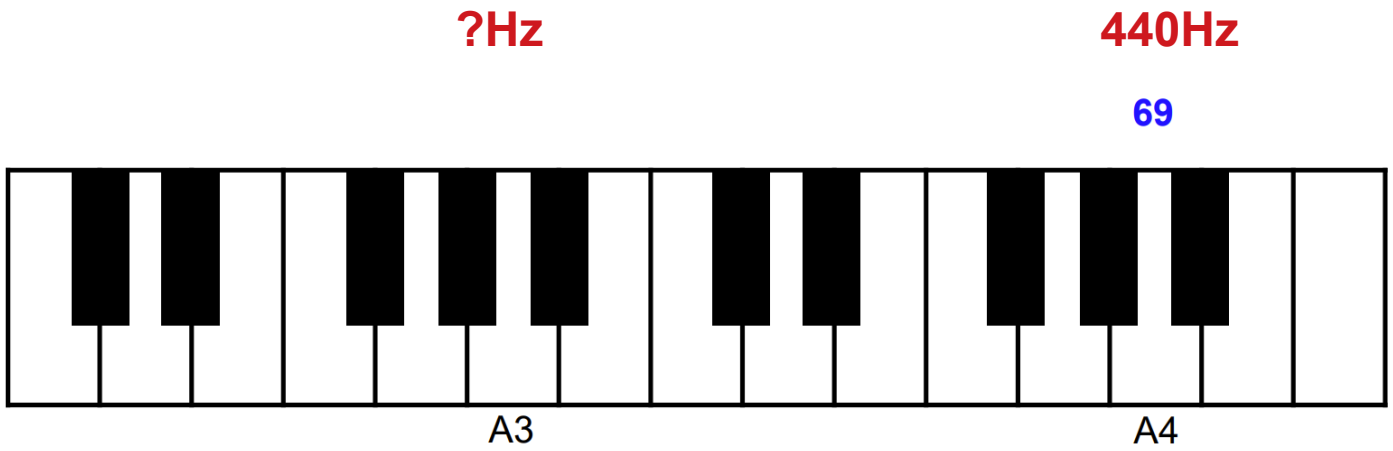
In the **MIDI** standard, we assign A4 to note number **69**

SPN and MNN mapped to piano

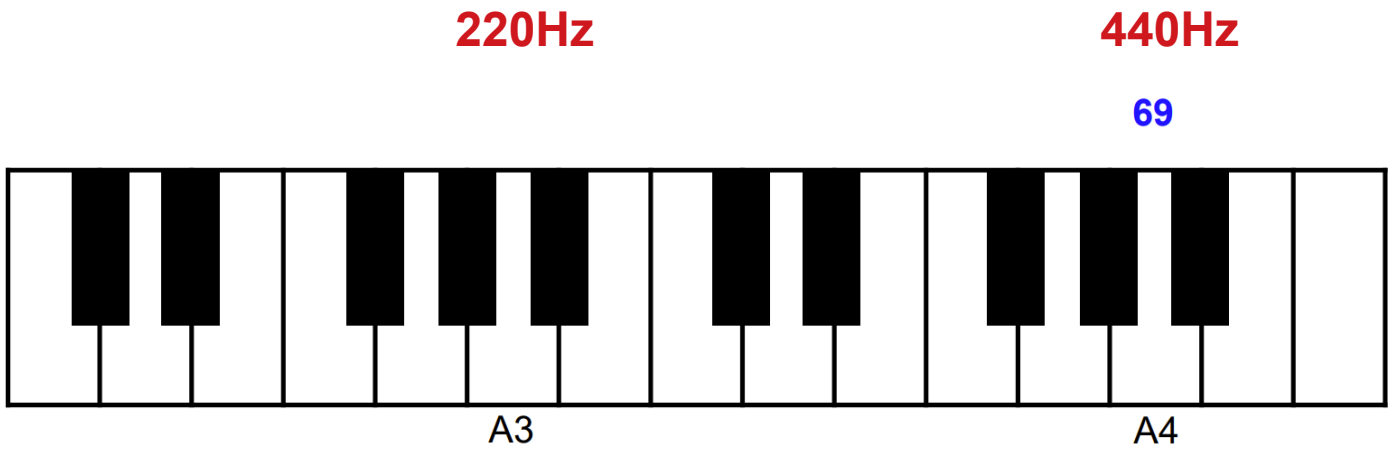


[FMP21] Fig 1.2 (adapted)

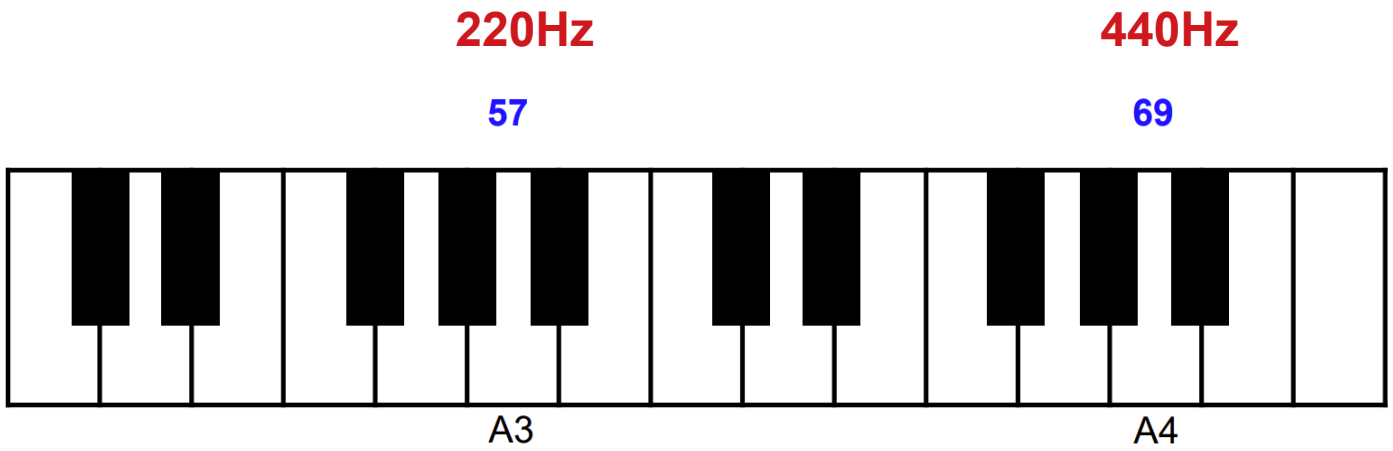
SPN and MNN mapped to piano



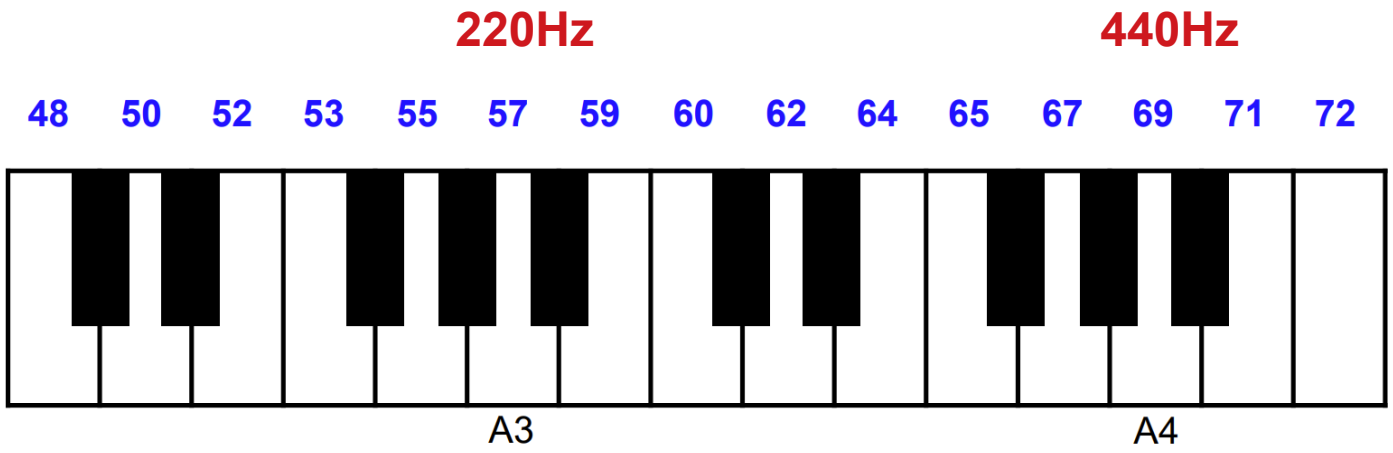
SPN and MNN mapped to piano



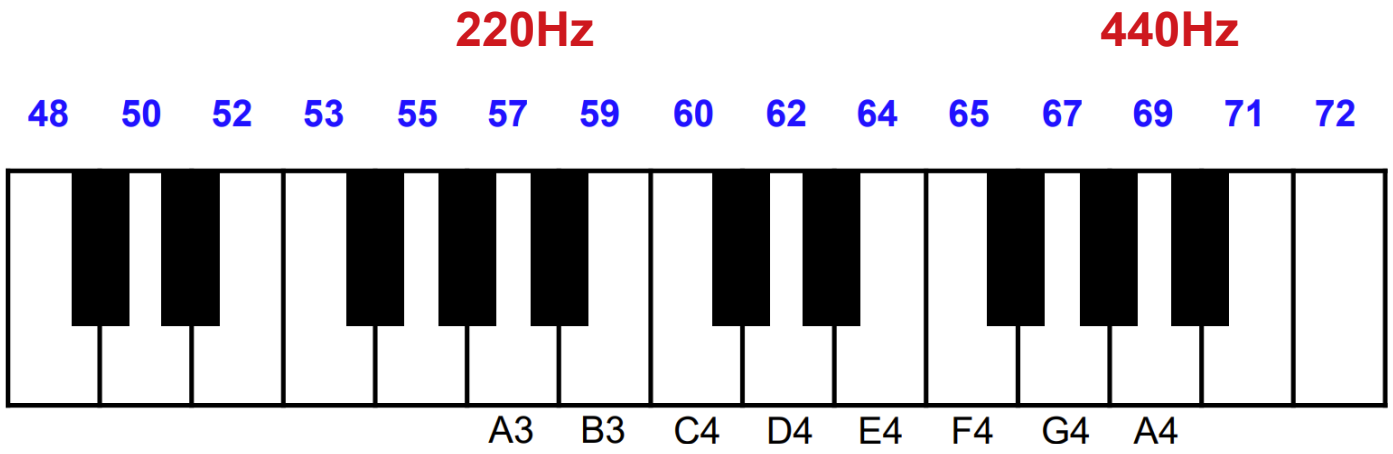
SPN and MNN mapped to piano

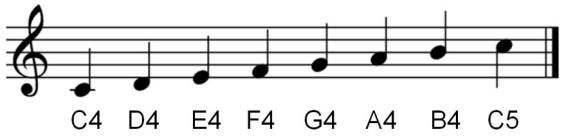


SPN and MNN mapped to piano



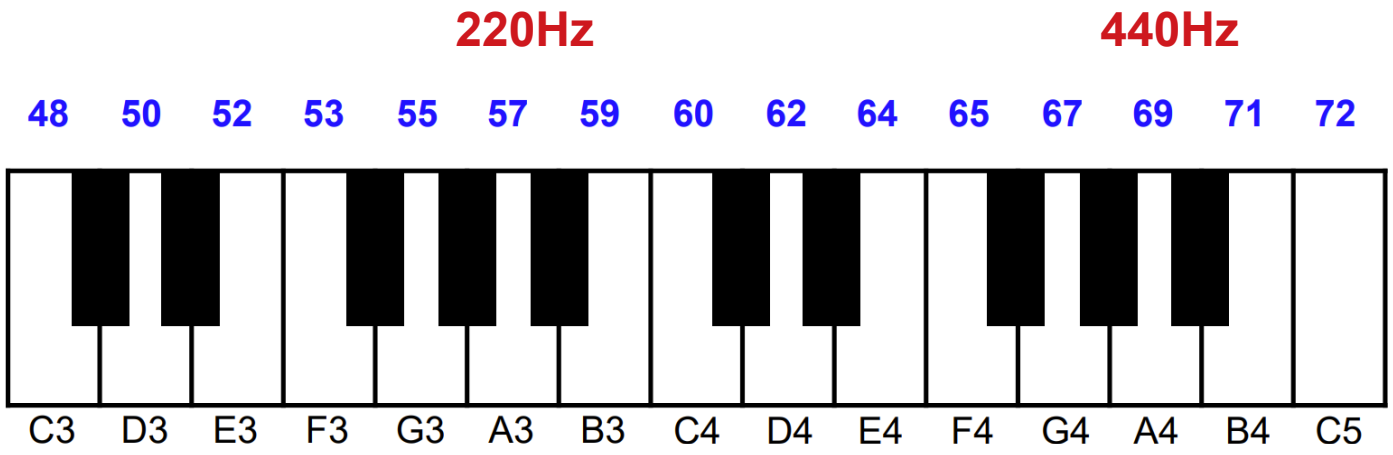
SPN and MNN mapped to piano



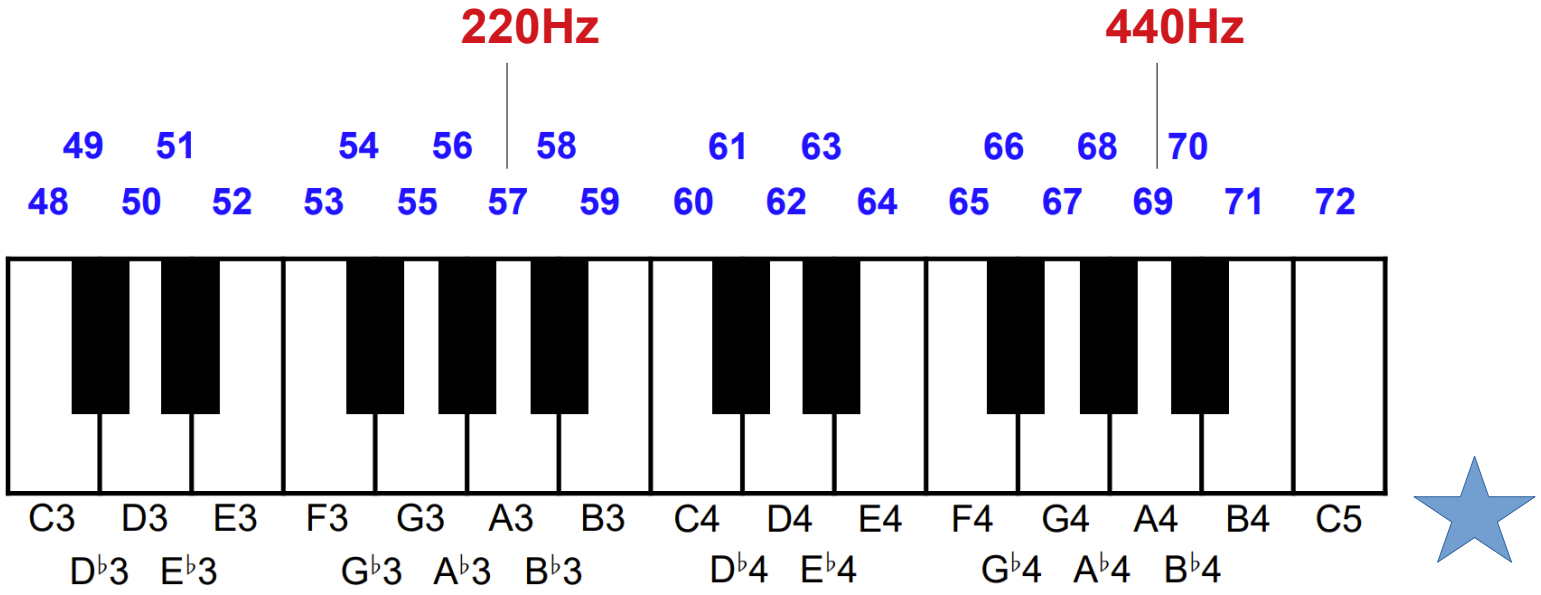


Durham University

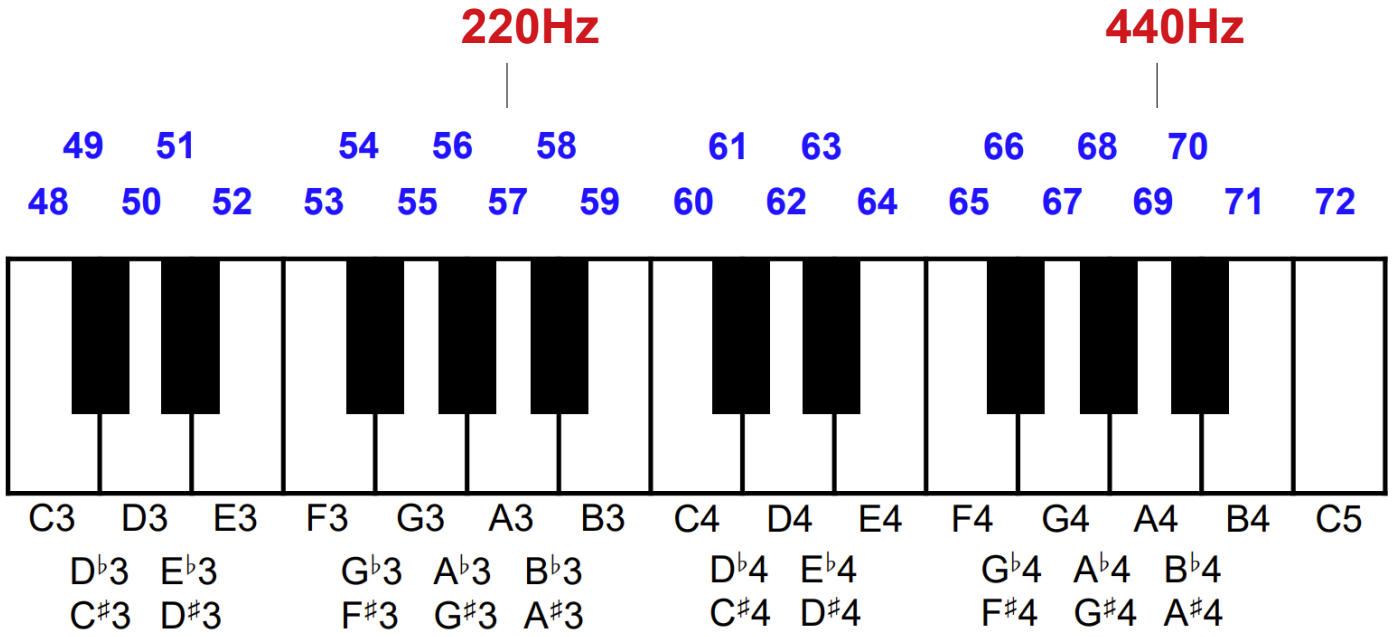
SPN and MNN mapped to piano



SPN and MNN mapped to piano



SPN and MNN mapped to piano



enharmonic equivalence

Durham University

Computing central frequency given a MIDI note number

Given p a MIDI note number (MNN), F_{pitch} is the frequency that generates a sinusoid with the required pitch:

$$F_{\text{pitch}}(p) = 2^{(p-69)/12} \cdot 440$$

Computing central frequency given a MIDI note number

Given p a MIDI note number (MNN), F_{pitch} is the frequency that generates a sinusoid with the required pitch:

$$F_{\text{pitch}}(p) = 2^{(p-69)/12} \cdot 440$$

- **MNN:** corresponds to equal-tempered piano keys (**A4** = **69**)
- **equal-tempered:** every pair of adjacent notes on the piano is equally spaced, perceptually speaking (octaves evenly divided)
- **pitch spacing:** equally spaced pairs of pitches have the same frequency ratio between each member of the pair

Other attributes of complex sounds

Harmonics: the mixture and balance of the components of simple sounds that make up a complex sound

Dynamics: the amplitude of a sound from a physical perspective, and how it changes over time

Timbre (or, **tone colour**): the quality of complex sounds that allows us, e.g., to tell different voices and instruments apart, AOTBE

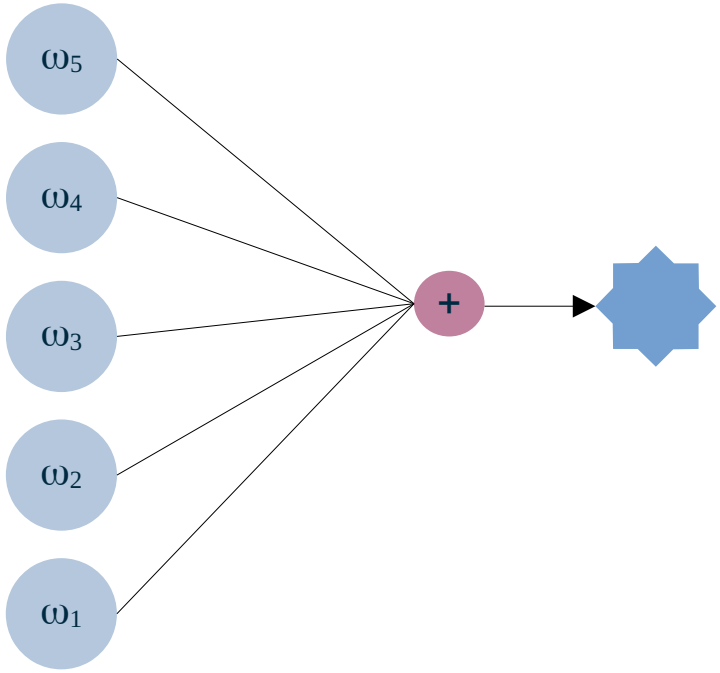
Loudness: the chief perceptual consequences of a sound's **dynamics**

Partials

Complex sounds (e.g. musical notes) be viewed as a mixture of simple sounds, represented here by their frequencies alone (ω_1, ω_2, ...)

In the context of a given complex sound, the most prominent constituent simple sounds are called **partials**

The number, frequency, and amplitude of a sound's partials has strong bearing on the **timbre** of a sound



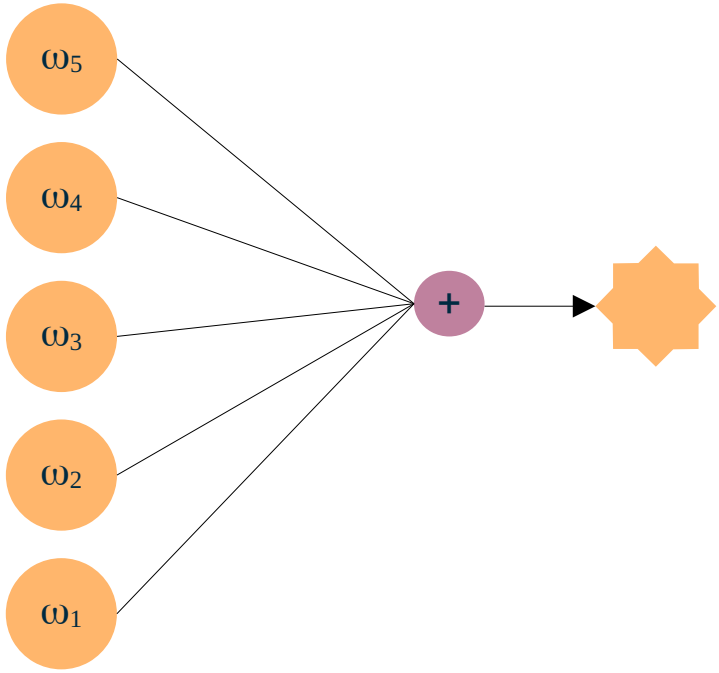
+ means sum the signals

Partials

Complex sounds (e.g. musical notes) be viewed as a mixture of simple sounds, represented here by their frequencies alone (ω_1, ω_2, ...)

In the context of a given complex sound, the most prominent constituent simple sounds are called **partials**

The number, frequency, and amplitude of a sound's partials has strong bearing on the **timbre** of a sound

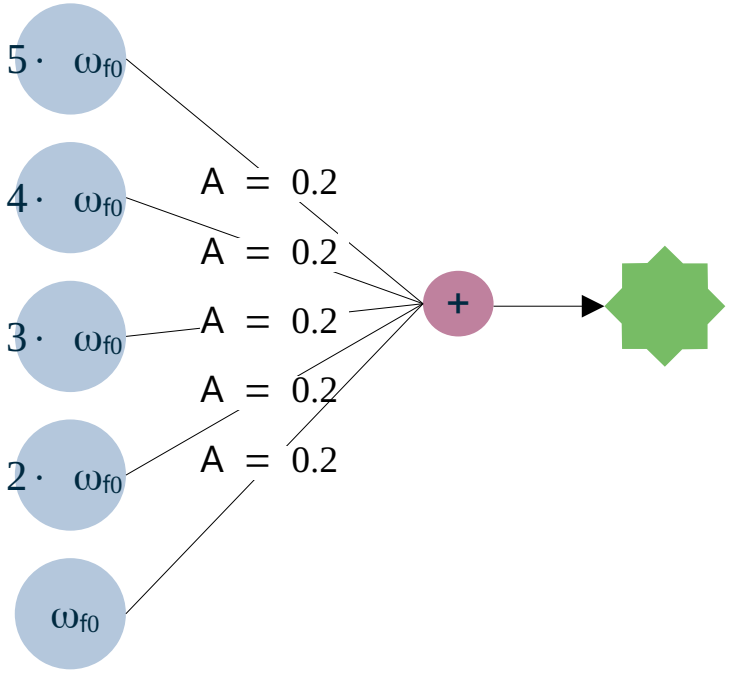


+ means sum the signals

Partials, fundamental, and harmonics

The **first partial** has the lowest frequency and is called the **fundamental**

Harmonic partials (or, simply, **harmonics**) are partials that are integer multiples of the fundamental frequency

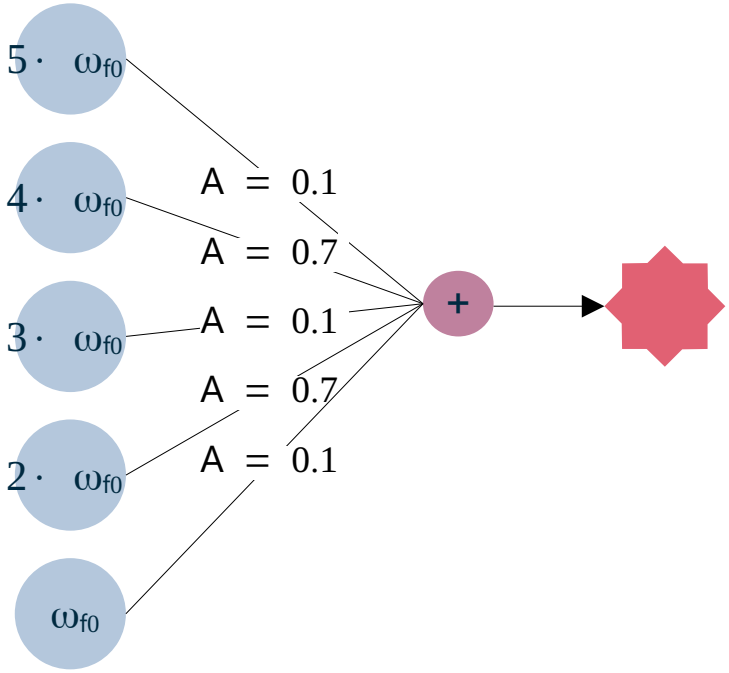


+ means sum the signals

Partials, fundamental, and harmonics

The **first partial** has the lowest frequency and is called the **fundamental**

Harmonic partials (or, simply, **harmonics**) are partials that are integer multiples of the fundamental frequency

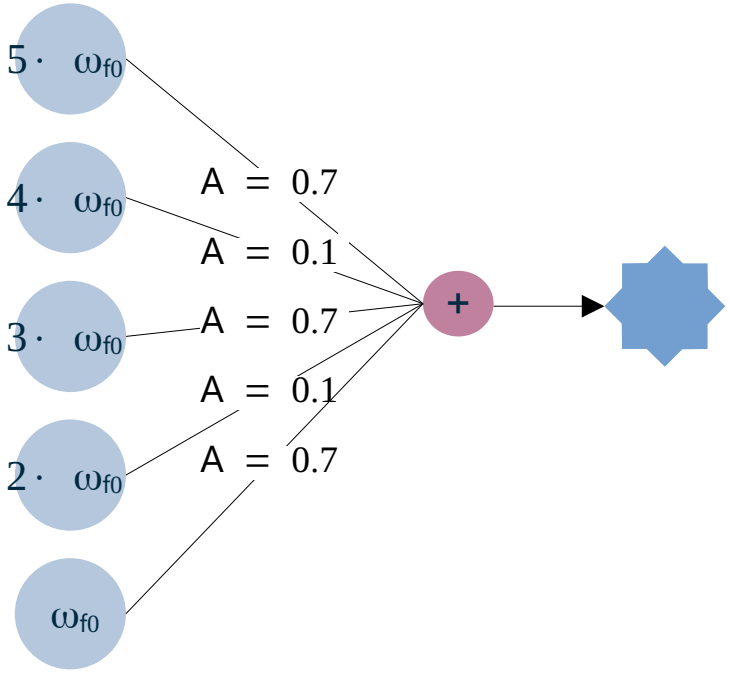


+ means sum the signals

Partials, fundamental, and harmonics

The **first partial** has the lowest frequency and is called the **fundamental**

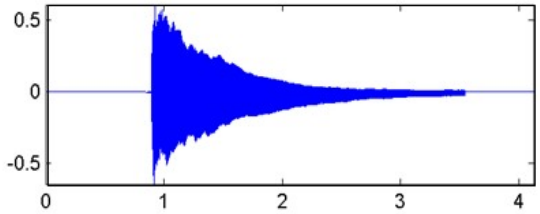
Harmonic partials (or, simply, **harmonics**) are partials that are integer multiples of the fundamental frequency

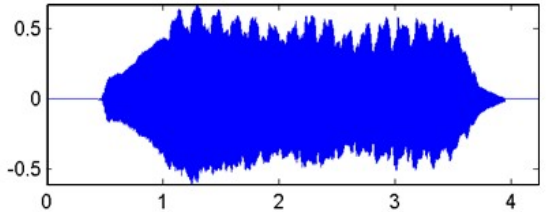


+ means sum the signals

Limits of analysis in the pressure-time domain

Claim: these **waveforms** represent sounds that come from performances of the same **pitch**

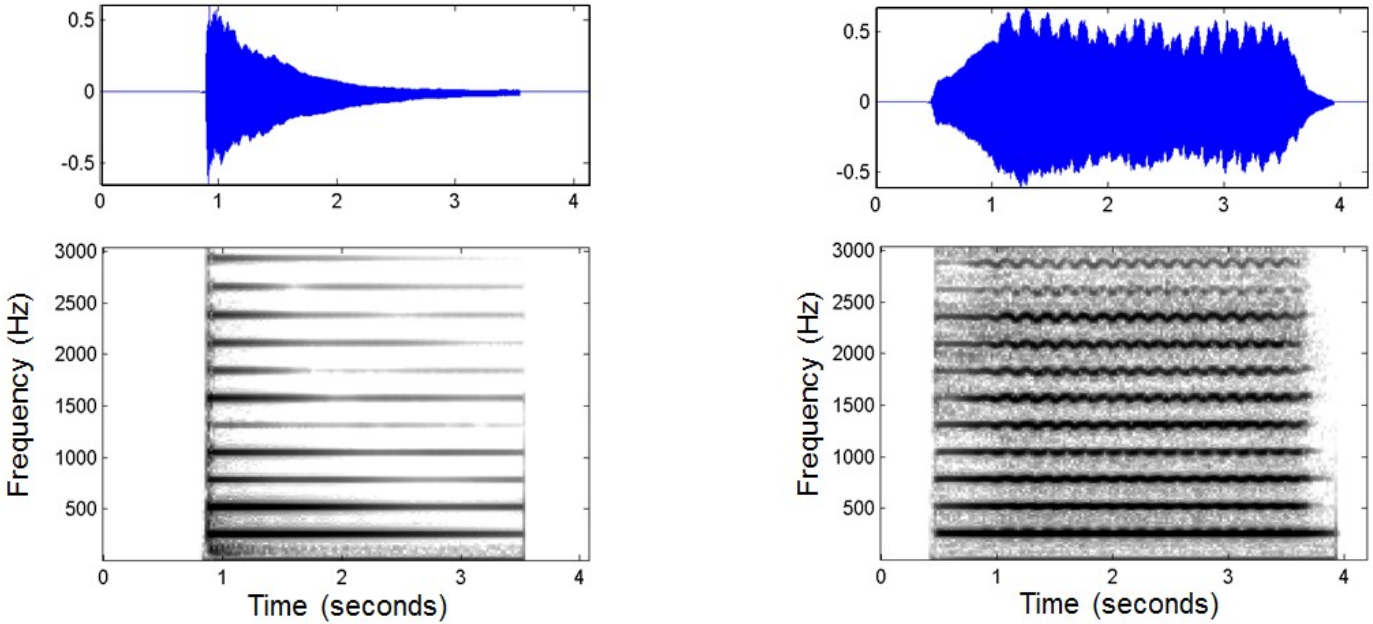




[FMP21] Fig 1.23 (adapted)

Can we validate this claim? What aspects of these sounds can we determine from the waveform alone?

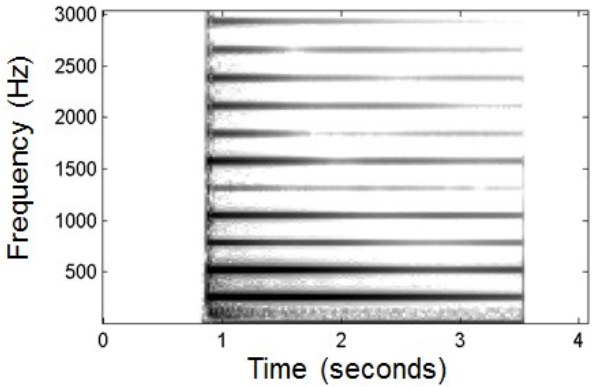
More complex representations?

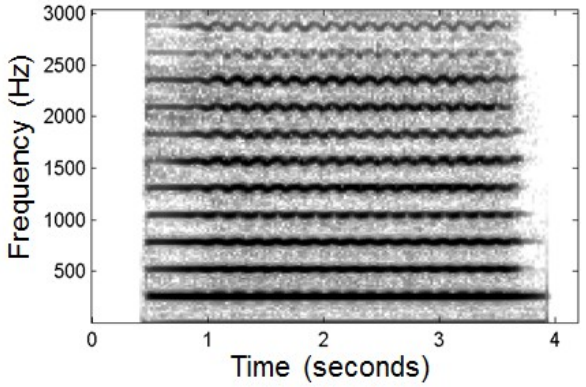


[FMP21] Fig 1.23 (adapted)

Spectrograms

(These) **spectrograms** are visualisations of the results of the Short-Time **Fourier Transform** of the **DT-signal** represented in the waveform directly above them in the plot





[FMP21] Fig 1.23 (adapted)

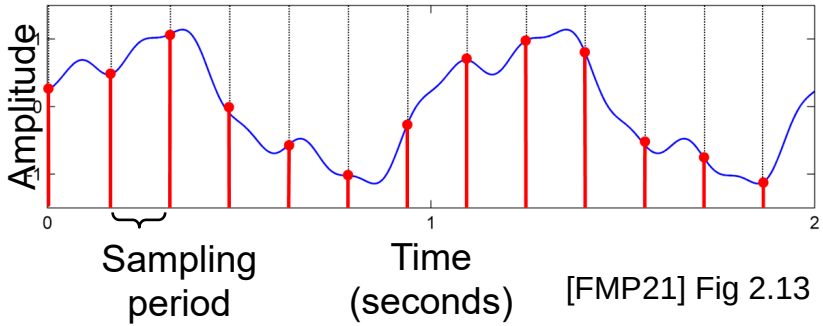
Discretization (analog-to-digital conversion)

Let $f(t)$ represent some signal that varies as a function of time. Depending on the domain and range of f, it could be:

- **analog**: $f(t)$ produces real numbers (R)
- **continuous-time** (CT): $t \in R$
- **discrete-time** (DT): t is some discrete subset of R (usually some $I \subset Z$)

Discretization

sampling: reducing a **continuous-time** (CT) signal to a **discrete-time** (DT) signal

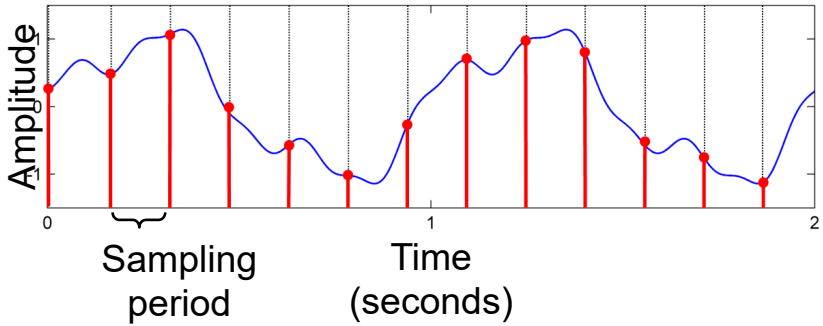


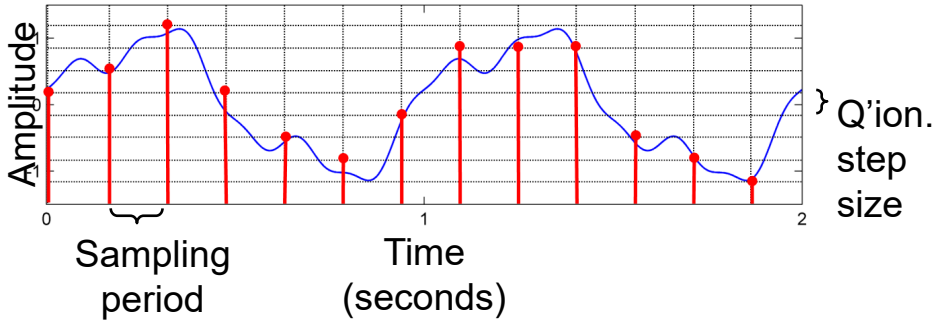
[FMP21] Fig 2.13

Discretization

sampling: reducing a **continuous-time** (CT) signal to a **discrete-time** (DT) signal

quantization: replacing the continuous range of possible (amplitude) values with a discrete range of possible values





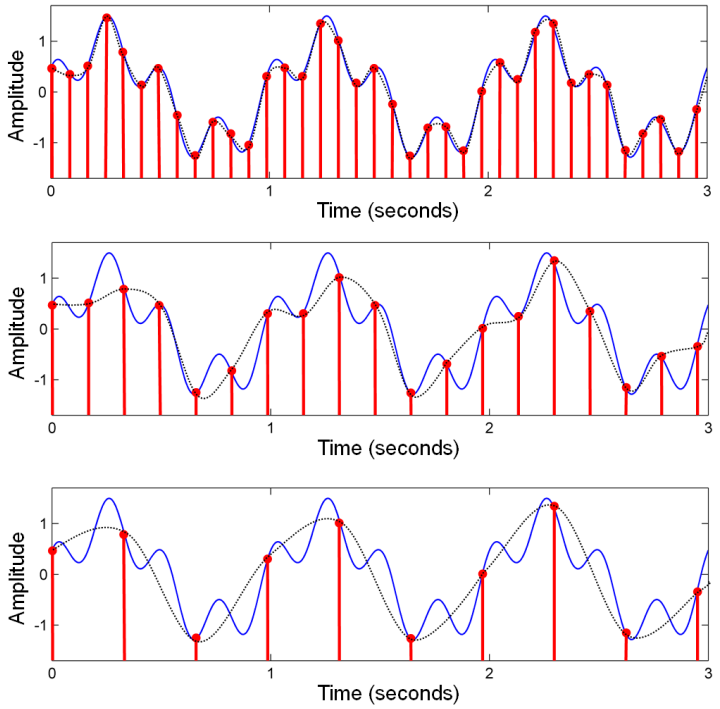
[FMP21] Fig 2.13

Sample rate - Informally

sample rate: how “often” we make observations of the CT signal

This will determine the fidelity of our discrete representation.

There are also practical consequences for recording, analysis, and playback



[FMP21] Fig 2.14

Sampling Theorem (Nyquist-Shannon) - Informally

How often do we need to sample an analog signal to ensure perfect reconstruction?

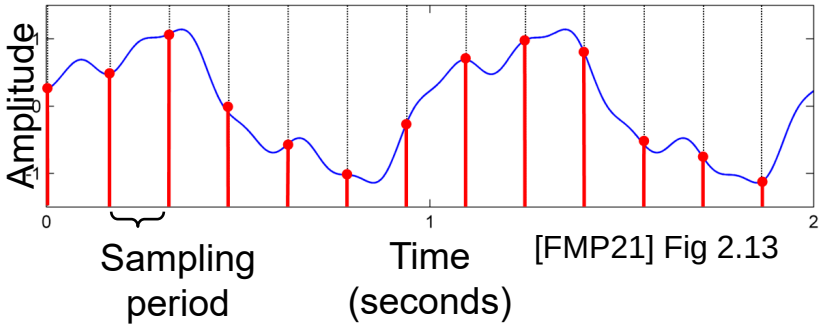
Twice as often as the highest-frequency component we wish to represent

Equidistant sampling (*T*-sampling)

A CT-signal can be transformed into a DT-signal by fixing $T > 0$ and setting

$$x(n) := f(n \cdot T)$$

for $n \in Z$.



[FMP21] Fig 2.13

We call $x(n)$ the sample of f at time $t = n \cdot T$ where T is the **sampling period** and $F_s = 1/T$ is the **sampling rate** (or sample rate) measured in Hertz (Hz)

Sampling – working with time

We represent sampled audio data computationally as 1-D arrays of numeric data (e.g. `np.ndarray`)

Question: If `A[]` is an array representing 30 seconds of audio as a T-sampled DT-signal, how many entries are in the array?

Sampling – working with time

Consequence 1: If we use `A[ ]` to locate something about a signal using its index, we need to know the sample rate to locate it in “real” time.

Consequence 2: Playback systems (digital-analog converters) need to know the rate at which the data was sampled to correctly reproduce the sound.

[FMP Notebook “STFT: Conventions and Implementations”]

Sampling Theorem (Nyquist-Shannon)

An Ω -bandlimited signal $f \in L^2(R)$ can be reconstructed from the T -sampling of f with $T := (1/2\Omega)$

(Ω -bandlimited means that components with frequencies above Ω Hz do not substantively contribute to the signal)

Fourier analysis: the idea – informally

Compare a signal with sinusoids of various frequencies (ω). For each frequency parameter $\omega \in \mathbb{R}$, we will get

- a magnitude coefficient $d_\omega \in \mathbb{R}_{\geq 0}$
- a phase coefficient $\phi_\omega \in \mathbb{R}$

with the property that when d_ω large:

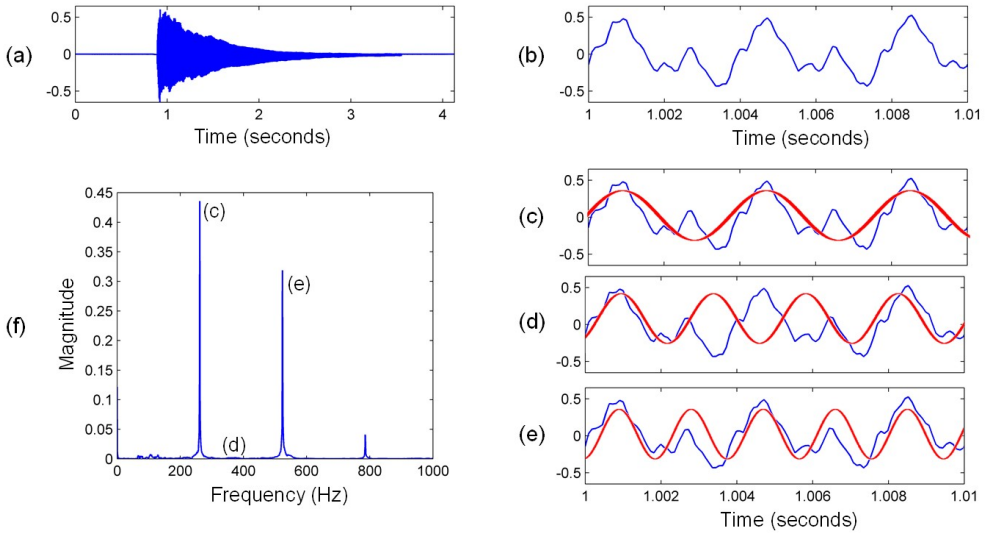
- high similarity between signal and a sinusoid of frequency ω
- the signal contains some periodic oscillation with that frequency

Fourier analysis – illustration

Consider the recording of a piano playing the note **C4** (261.6Hz).

Fig 2.1 (a-b) represent views on the signal in the time domain. Fig 2.1 (f) represents a view on the signal in the frequency domain.

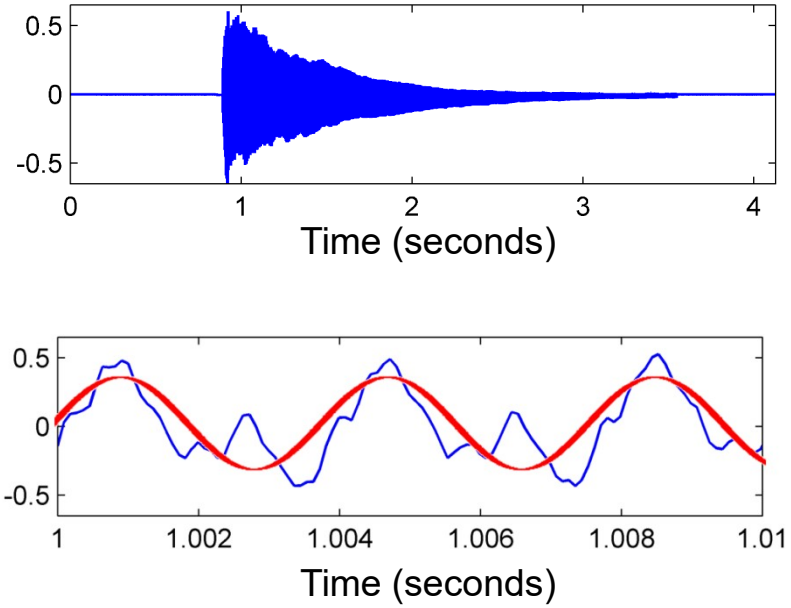
We obtain frequency-domain representations by performing the **Fourier transform** on a time-domain representation.

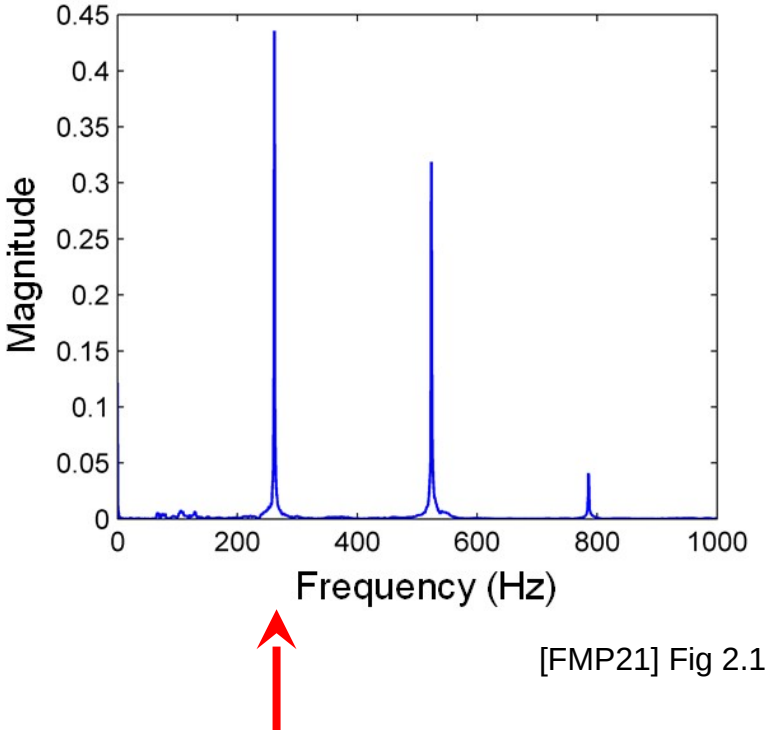


[FMP21] Fig 2.1

Durham University

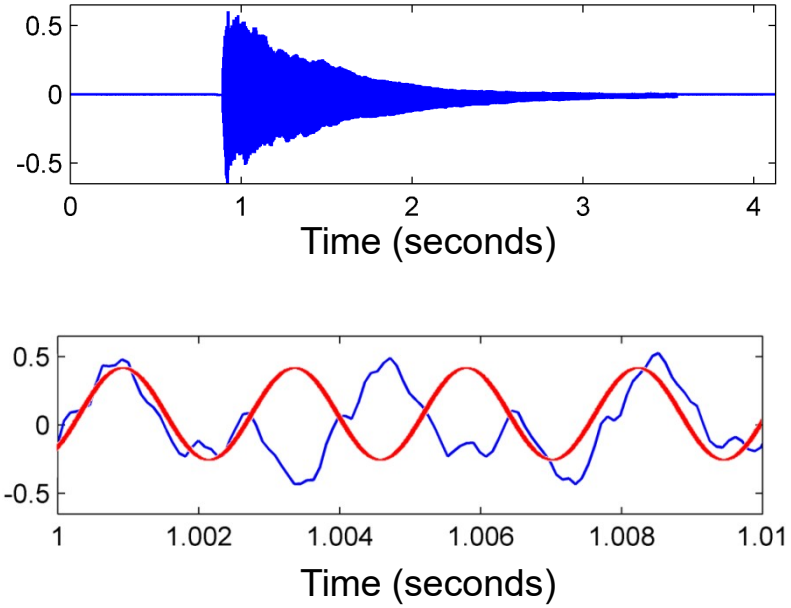
Fourier analysis – illustration

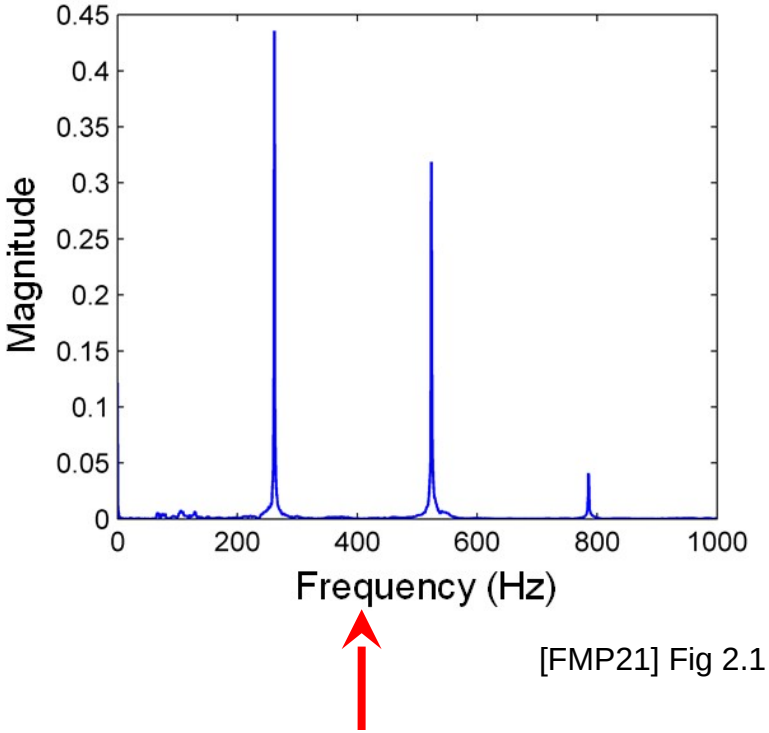




[FMP21] Fig 2.1

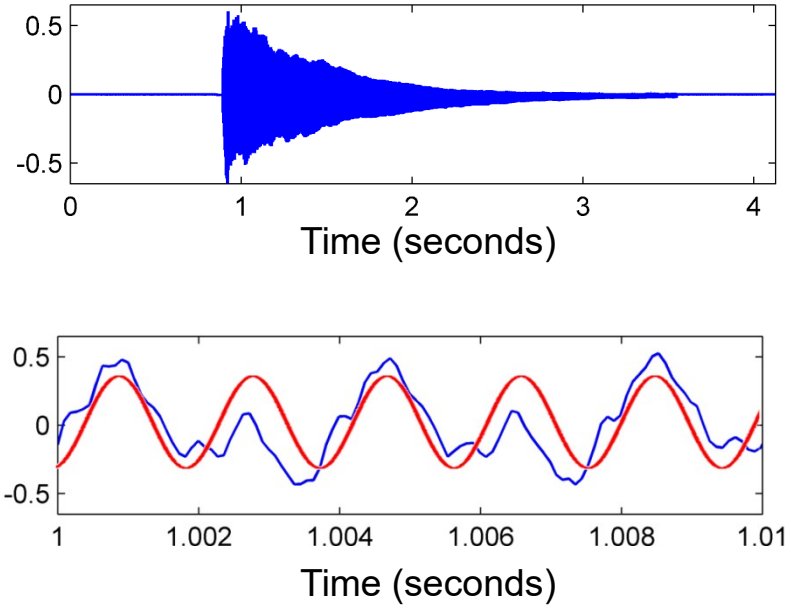
Fourier analysis – illustration

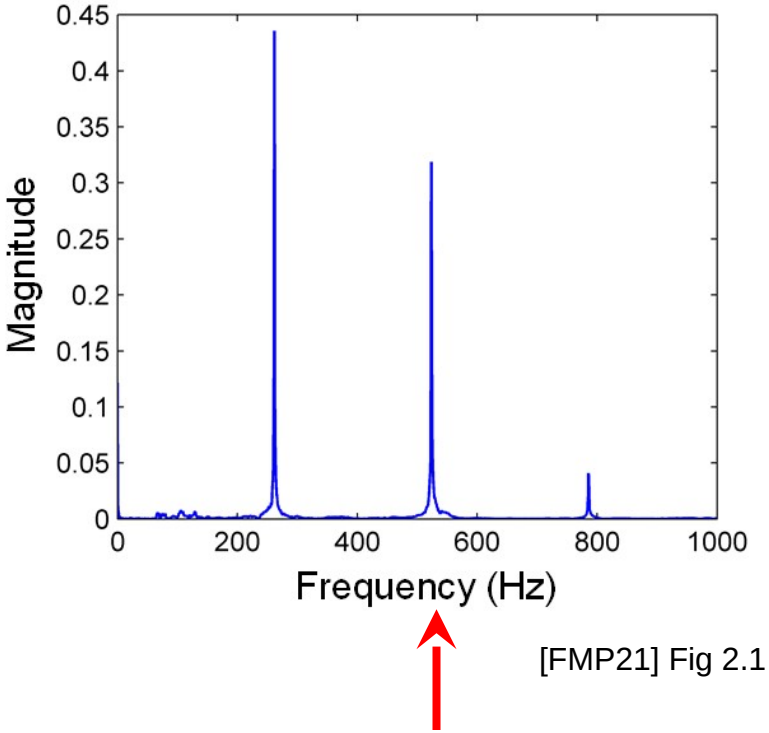




[FMP21] Fig 2.1

Fourier analysis – illustration





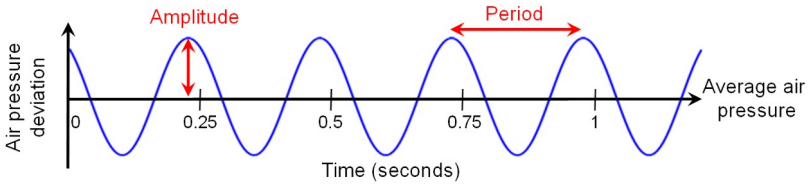
[FMP21] Fig 2.1

Sinusoids as “prototypes”

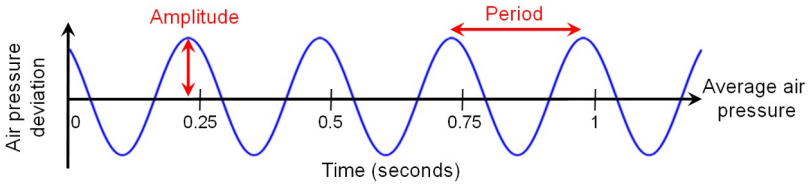
Recall

$$g\colon \mathbb{R} \to \mathbb{R}$$

$$g(t) := A\sin(2\pi(\omega t - \varphi))$$



Sinusoids as “prototypes”



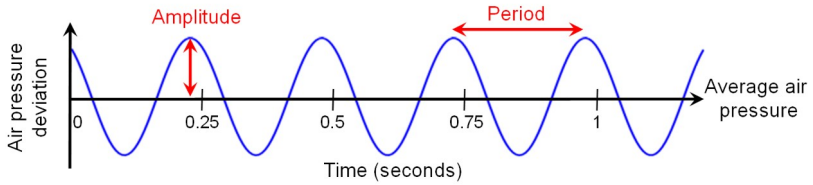
Recall

$$g : \mathbb{R} \to \mathbb{R}$$
$$g(t) := A \sin(2\pi(\omega t - \varphi))$$

Noting the similarity between sin and cos, consider a family of sinusoidal functions parameterised by ω and ϕ:

$$\cos_{\omega,\varphi} : \mathbb{R} \to \mathbb{R}$$
$$\cos_{\omega,\varphi}(t) := \sqrt{2} \cos(2\pi(\omega t - \varphi))$$

Sinusoids as “prototypes”



Recall

$$g : \mathbb{R} \to \mathbb{R}$$
$$g(t) := A \sin(2\pi(\omega t - \varphi))$$

Noting the similarity between sin and cos, consider a family of sinusoidal functions parameterised by ω and ϕ:

$$\cos_{\omega,\varphi} : \mathbb{R} \to \mathbb{R}$$
$$\cos_{\omega,\varphi}(t) := \sqrt{2} \cos(2\pi(\omega t - \varphi))$$

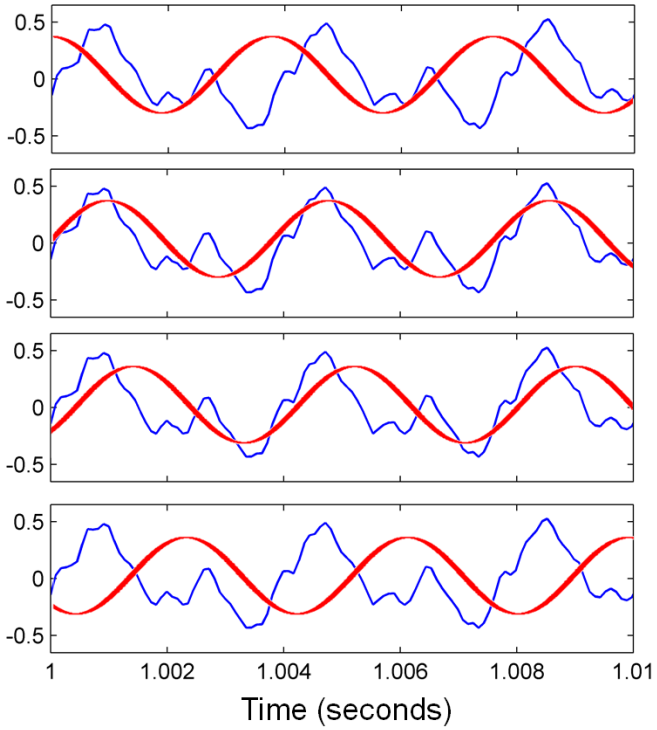
ϕ represents **phase**, which can be thought of as giving an “offset” of the sinusoid

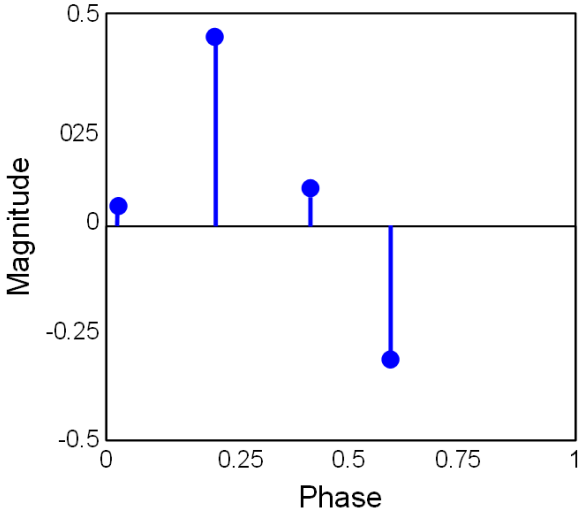
This family of functions can be used to specify “prototype” sounds that get summed together to produce complex sounds

Interpretation of phase (ϕ)

$$\cos_{\omega,\varphi} : \mathbb{R} \to \mathbb{R}$$

$$\cos_{\omega,\varphi}(t) := \sqrt{2}\cos(2\pi(\omega t - \varphi))$$



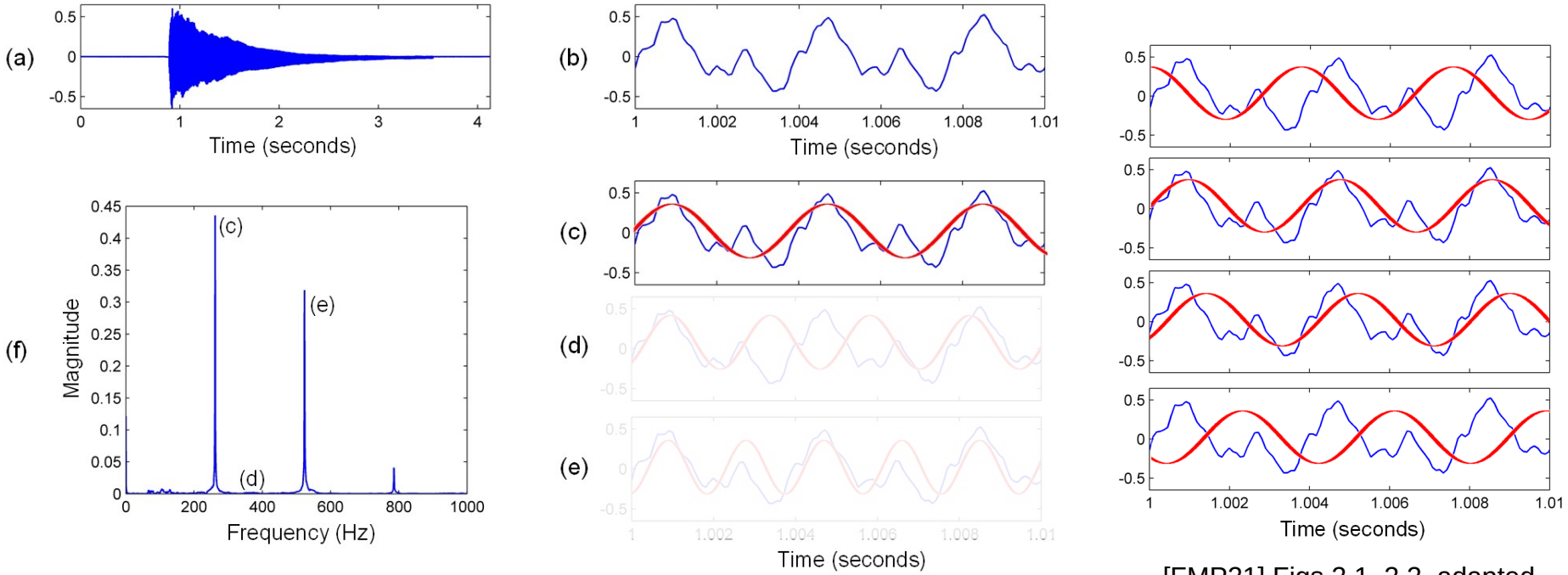


[FMP21] Fig 2.2

Durham University

Picking the phase (ϕ) that maximizes similarity

$$\cos_{\omega,\varphi} : \mathbb{R} \to \mathbb{R}$$
$$\cos_{\omega,\varphi}(t) := \sqrt{2}\cos(2\pi(\omega t - \varphi))$$



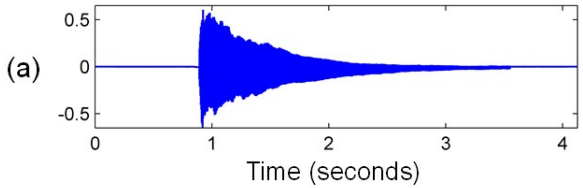
[FMP21] Figs 2.1, 2.2, adapted

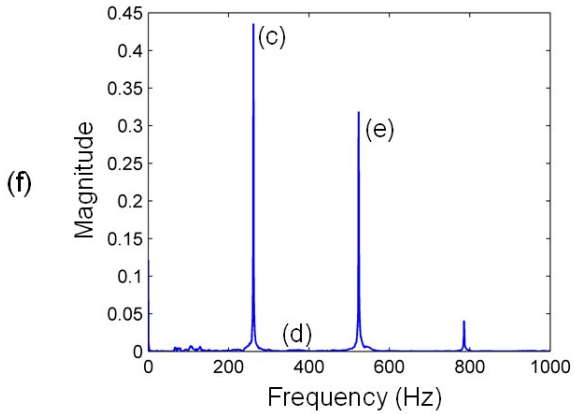
Durham University

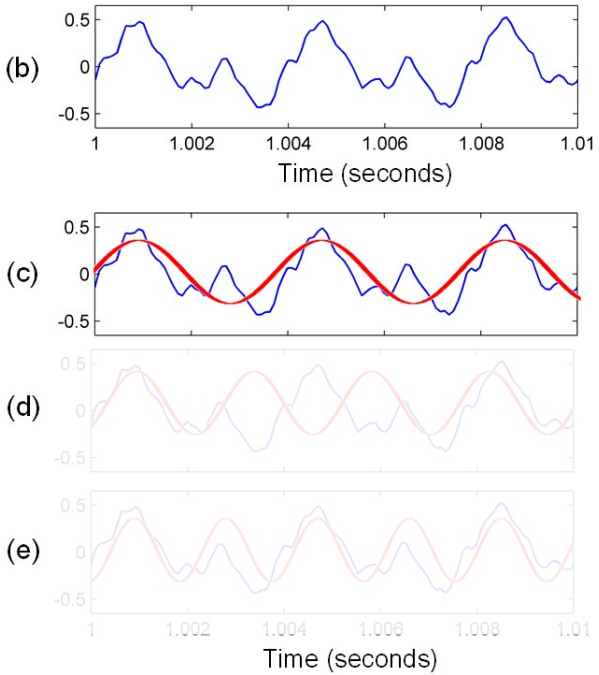
Picking the phase (ϕ) that maximizes similarity

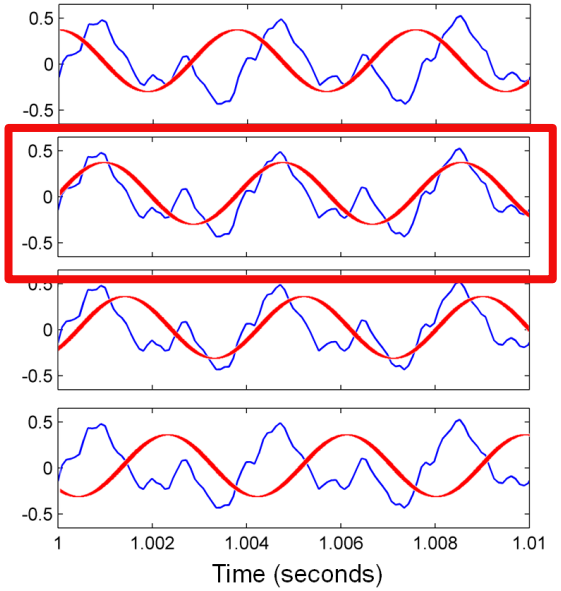
$$\cos_{\omega,\varphi} : \mathbb{R} \to \mathbb{R}$$

$$\cos_{\omega,\varphi}(t) := \sqrt{2}\cos(2\pi(\omega t - \varphi))$$









[FMP21] Figs 2.1, 2.2, adapted

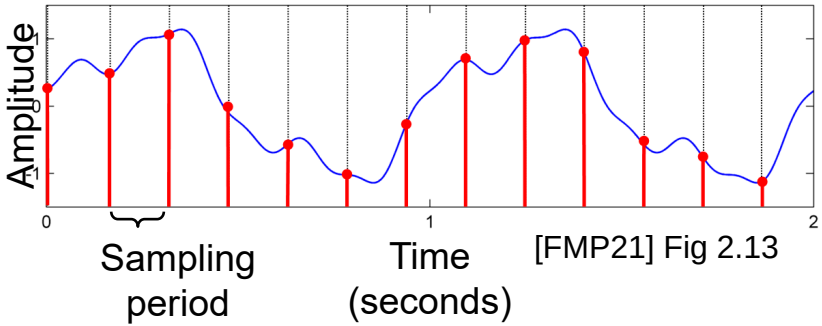
Durham University

Equidistant sampling (*T*-sampling)

A CT-signal can be transformed into a DT-signal by fixing $T > 0$ and setting

$$x(n) := f(n \cdot T)$$

for $n \in Z$.



[FMP21] Fig 2.13

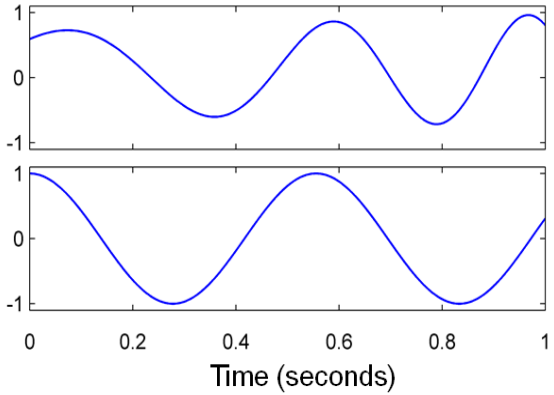
We call $x(n)$ the sample of f at time $t = n \cdot T$ where T is the **sampling period** and $F_s = 1/T$ is the **sampling rate** (or sample rate) measured in Hertz (Hz)

Measuring the similarity between functions

Given two signals, we would like some way to compute their similarity

Let's say two signals f and g are similar if:

- f takes positive values when g does
- f takes negative values when g does

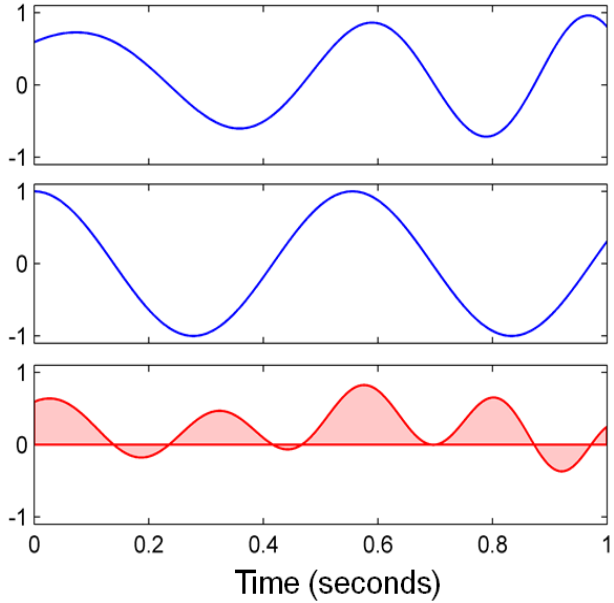


[FMP21] Fig 2.3, adapted

Measuring the similarity between functions

Given two signals f: R → R and g: R → R, the similarity between these can be computed by taking the integral (over time) of their product

$$\int_{t \in \mathbb{R}} f(t) \cdot g(t)\, dt$$

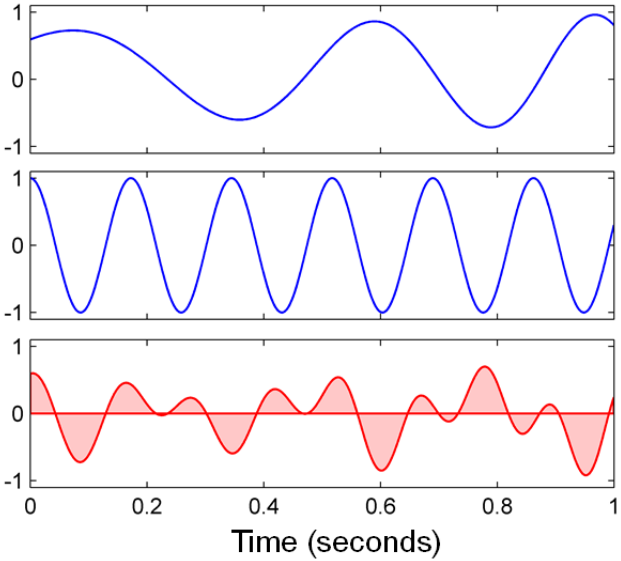


[FMP21] Fig 2.3

Measuring the similarity between functions

Given two signals f: R → R and g: R → R, the similarity between these can be computed by taking the integral (over time) of their product

$$\int_{t \in \mathbb{R}} f(t) \cdot g(t)\, dt$$



[FMP21] Fig 2.3

Fourier analysis – first formal statement

We can think of Fourier analysis as an optimization problem, where the goal is to find – for some fixed $\omega \in \mathrm{R}$ – coefficients with the following properties:

$$d_\omega := \max_{\varphi \in [0,1)} \left(\int_{t \in \mathbb{R}} f(t) \cos_{\omega,\varphi}(t)\, dt \right)$$

$$\varphi_\omega := \arg\max_{\varphi \in [0,1)} \left(\int_{t \in \mathbb{R}} f(t) \cos_{\omega,\varphi}(t)\, dt \right)$$

Fourier analysis – first formal statement

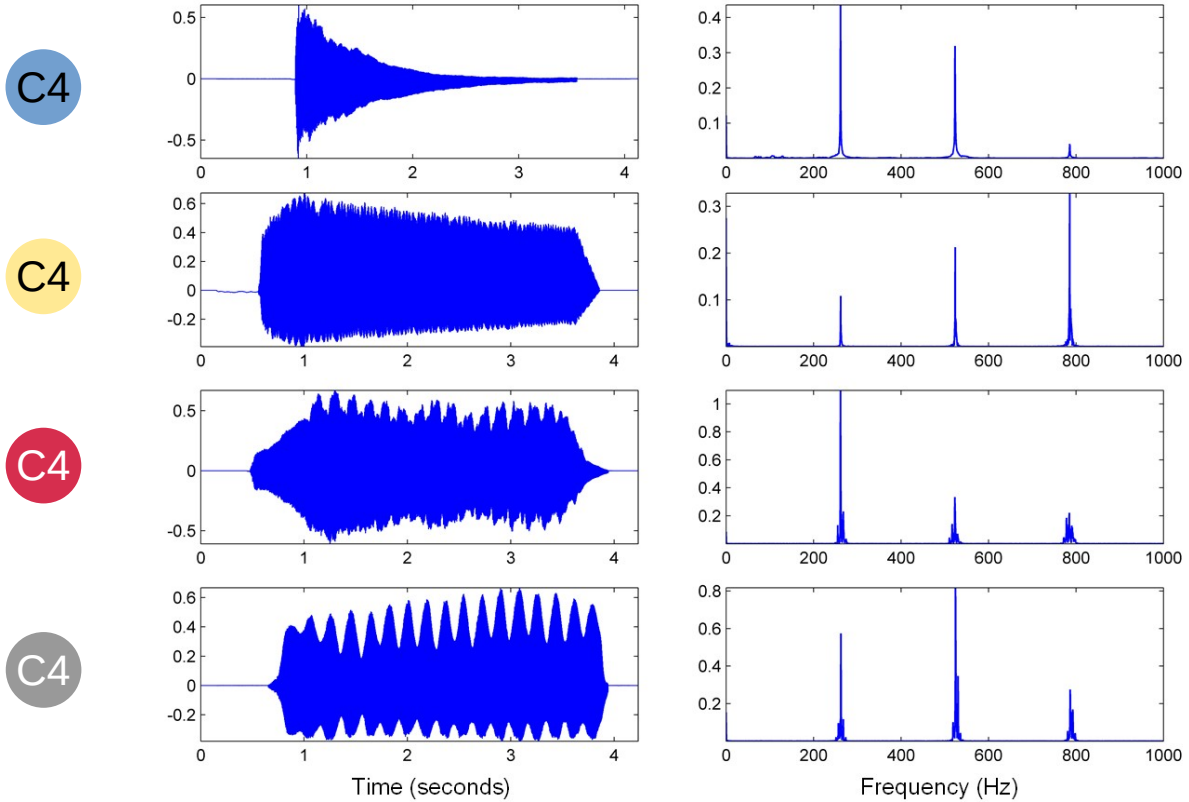
We can think of Fourier analysis as an optimization problem, where the goal is to find – for some fixed $\omega \in \mathrm{R}$ – coefficients with the following properties:

$$d_\omega := \max_{\varphi \in [0,1)} \left(\int_{t \in \mathbb{R}} f(t) \cos_{\omega,\varphi}(t)\, dt \right)$$

$$\varphi_\omega := \arg\max_{\varphi \in [0,1)} \left(\int_{t \in \mathbb{R}} f(t) \cos_{\omega,\varphi}(t)\, dt \right)$$

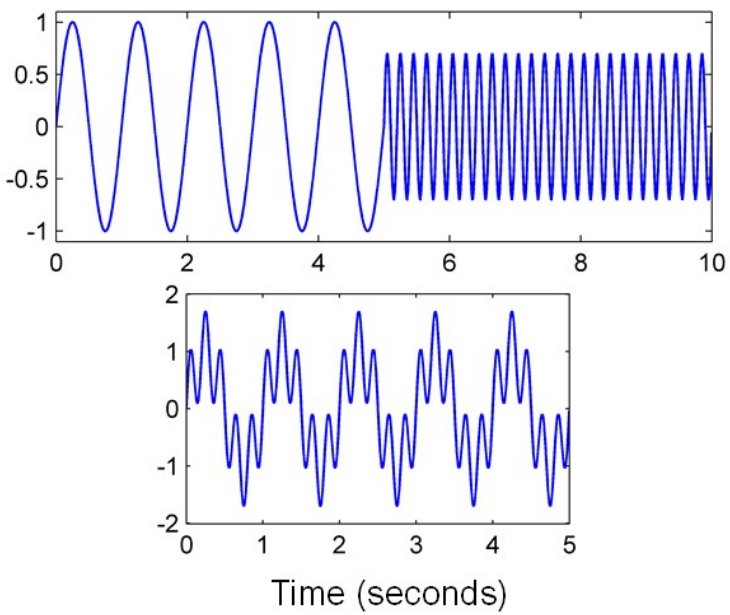
Claim*: We can write down closed-form expressions for these optimal coefficients, and compute a "collection" of arbitrarily many of them (i.e. covering $\omega \in \mathrm{R}$). This "collection" is the **Fourier transform** of f.

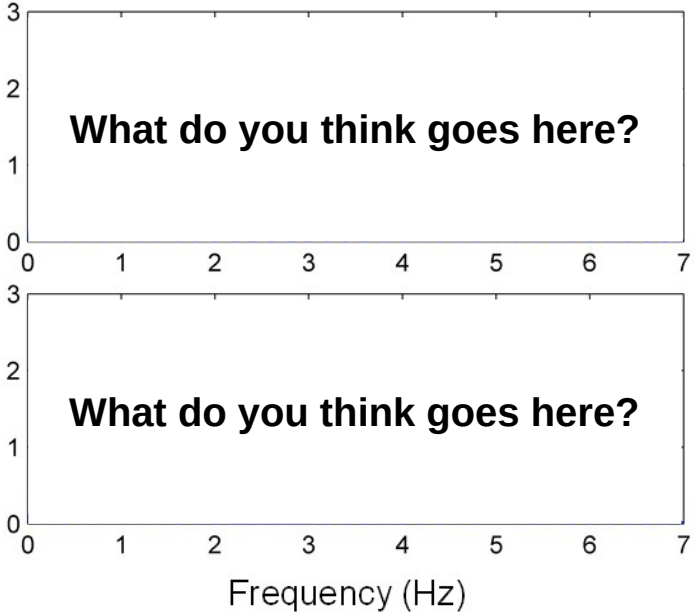
Fourier analysis – comparing sounds



[FMP21] Fig 2.5

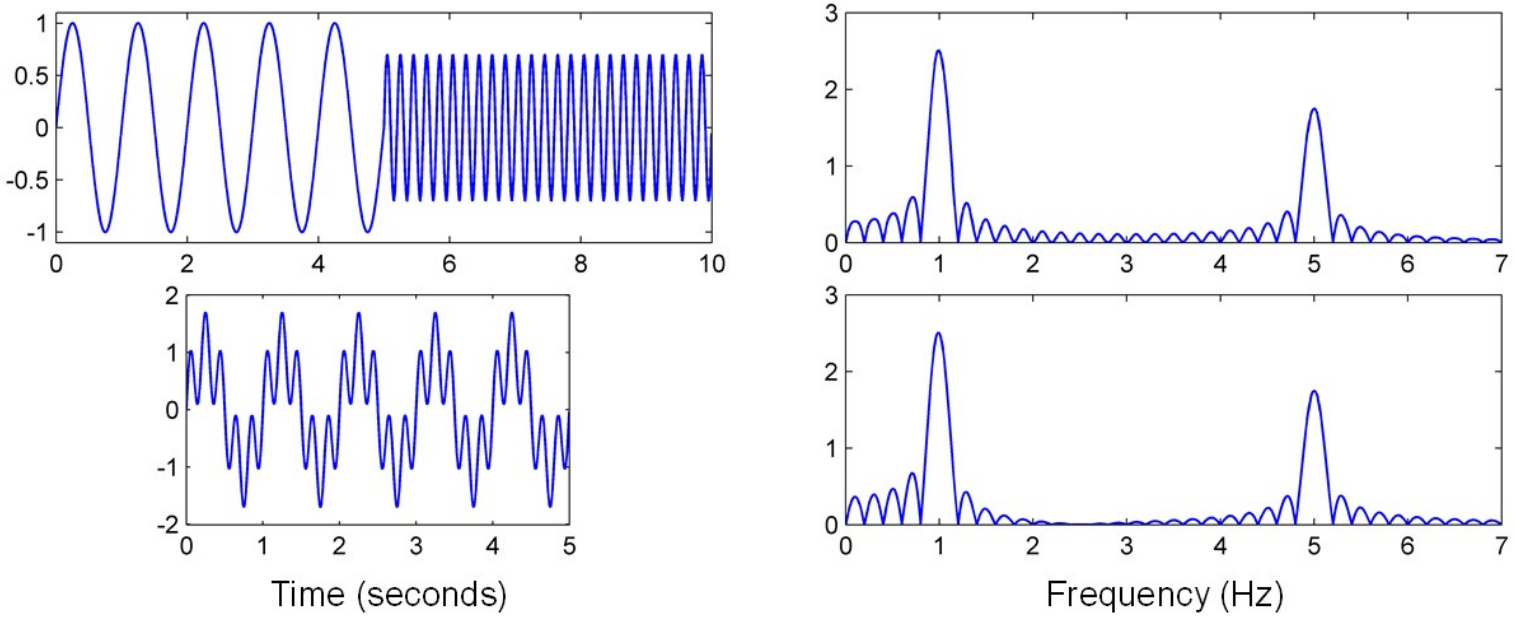
Fourier analysis – illustration





[FMP21] Fig 2.6, adapted

Fourier analysis – losing temporal information



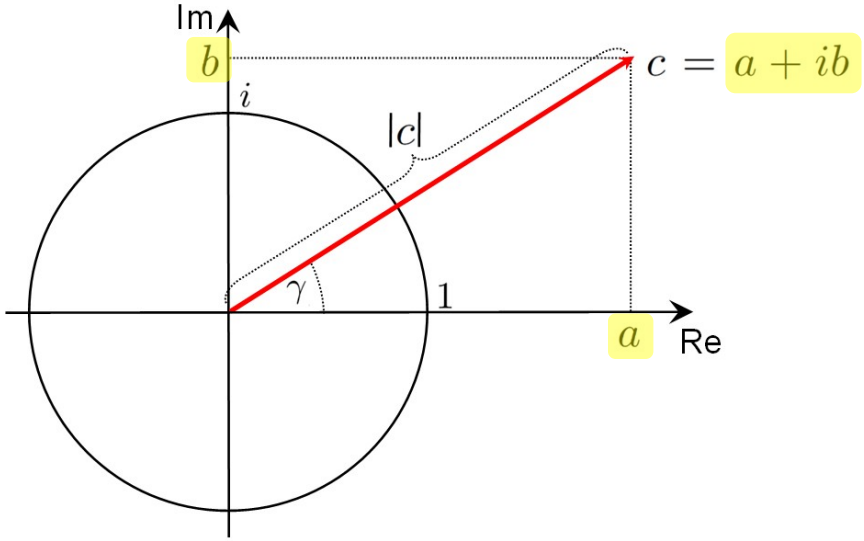
[FMP21] Fig 2.6

Complex numbers

Any complex number $c \in C$ can be written in the form $c = a + ib$, where $i^2 = -1$

We call:

- $a \in R$ the **real part** of c
- $b \in R$ the **imaginary part** of c

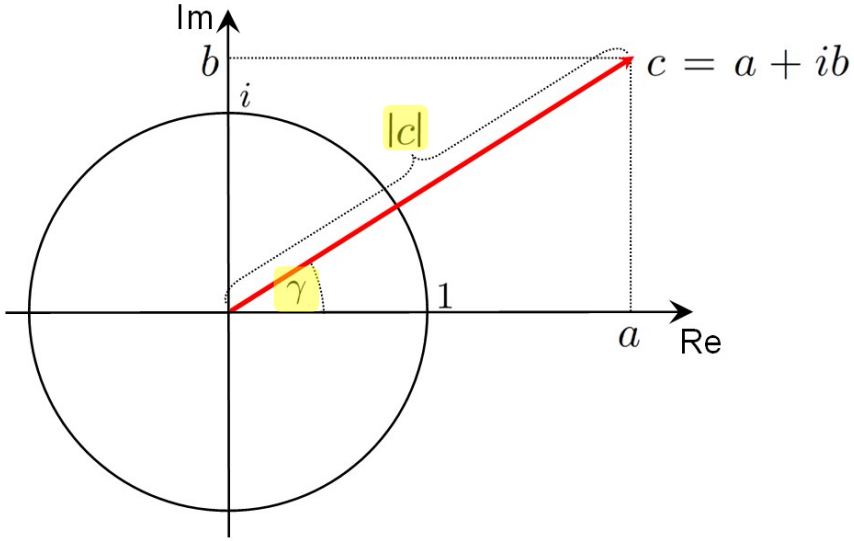


[FMP21] Fig 2.4, adapted

Complex numbers – polar coordinate form

Any complex number $c \in \mathbb{C}$ also be written in **polar coordinate** form, as a combination of $|c| \in \mathbb{R}_{\geq 0}$ and $\gamma \in [0, 2\pi)$ where:

- $|c|$ is the distance from the origin
- γ is the angle in radians between the (positive) horizontal axis and a line between the origin and c



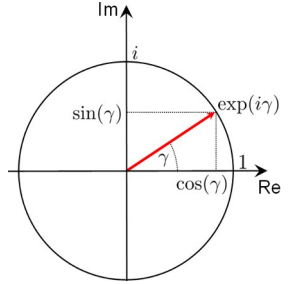
[FMP21] Fig 2.4, adapted

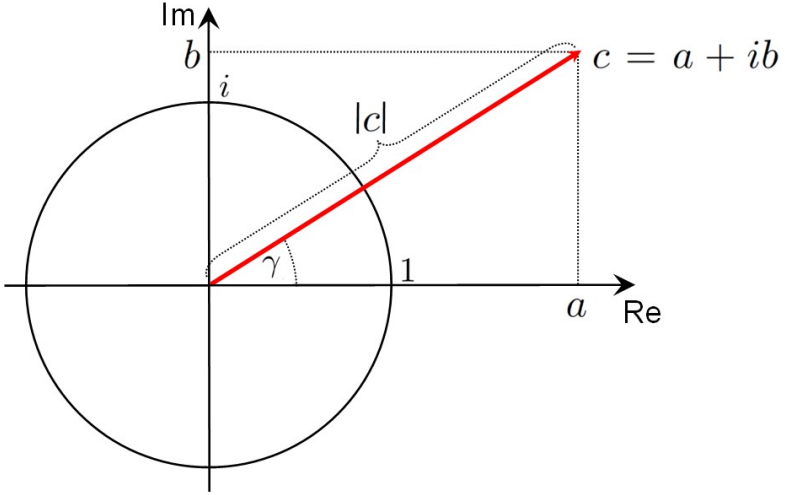
Complex numbers – polar coordinate form

For some $c \in C$ specified as $c = a + ib$

$$|c| := \sqrt{a^2 + b^2}$$

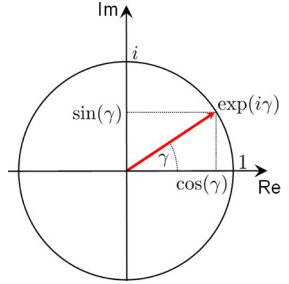
$$\gamma := \operatorname{atan2}(b, a)$$





[FMP21] Fig 2.4, adapted

Complex numbers – polar coordinate form



For some $c \in \mathbb{C}$ specified as $c = a + ib$

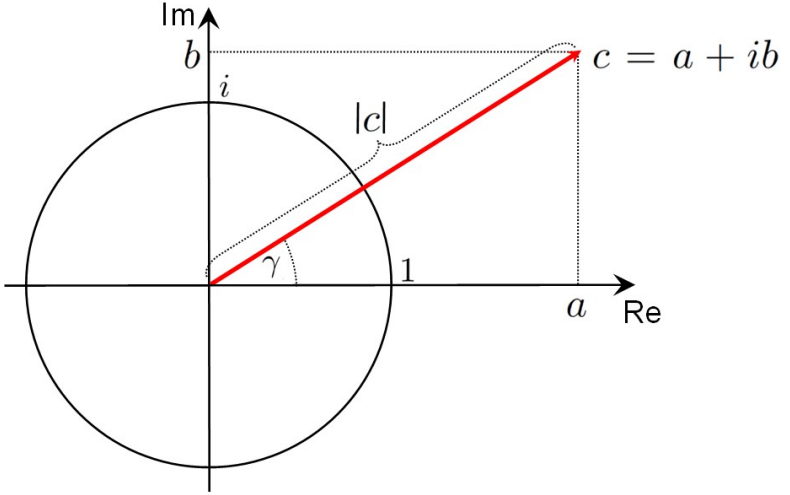
$$|c| := \sqrt{a^2 + b^2}$$
$$\gamma := \operatorname{atan2}(b, a)$$

Define/note the relationship

$$\exp(i\gamma) := \cos(\gamma) + i\sin(\gamma)$$

And we get

$$c = |c| \cdot \exp(i\gamma)$$



[FMP21] Fig 2.4, adapted

Note: $\exp(i\gamma)$ is 2π-periodic

Link to animation

Fourier transform (“analysis”)

$$\hat{f} : \mathbb{R} \to \mathbb{C}$$

$$\hat{f}(\omega) := c_\omega$$

$$c_\omega := \frac{d_\omega}{\sqrt{2}} \cdot \exp(2\pi i(-\varphi\omega))$$

$$\begin{aligned}\hat{f}(\omega) &= \int_{t\in\mathbb{R}} f(t)\exp(-2\pi i\omega t)\,dt \\ &= \int_{t\in\mathbb{R}} f(t)\cos(-2\pi\omega t)\,dt + i\int_{t\in\mathbb{R}} f(t)\sin(-2\pi\omega t)\,dt\end{aligned}$$

Fourier transform (“analysis”)

[previous slide]

$$\hat{f}(\omega) = \int_{t\in\mathbb{R}} f(t)\exp(-2\pi i\omega t)\,dt$$
$$= \int_{t\in\mathbb{R}} f(t)\cos(-2\pi\omega t)\,dt + i\int_{t\in\mathbb{R}} f(t)\sin(-2\pi\omega t)\,dt$$

We state (without proof) that, for a given frequency ω,

$$d_\omega = \sqrt{2}|\hat{f}(\omega)|$$

gives the **magnitude coefficient** that is the maximum of the desired expression.

$$d_\omega := \max_{\varphi\in[0,1)} \left(\int_{t\in\mathbb{R}} f(t)\cos_{\omega,\varphi}(t)\,dt \right)$$

Fourier transform ("analysis")

[previous slide]

$$\hat{f}(\omega) = \int_{t\in\mathbb{R}} f(t)\exp(-2\pi i\omega t)\,dt$$
$$= \int_{t\in\mathbb{R}} f(t)\cos(-2\pi\omega t)\,dt + i\int_{t\in\mathbb{R}} f(t)\sin(-2\pi\omega t)\,dt$$

We state (without proof) that, for a given frequency ω,

$$d_\omega = \sqrt{2}|\hat{f}(\omega)|$$

gives the **magnitude coefficient** that is the maximum of the desired expression.

$$d_\omega := \max_{\varphi\in[0,1)} \left(\int_{t\in\mathbb{R}} f(t)\cos_{\omega,\varphi}(t)\,dt \right)$$

And that

$$\varphi_\omega = -\frac{1}{2\pi}\gamma_\omega$$

gives the **phase** that maximises that expression, where $|\hat{f}(\omega)|$ and γ_ω give the FT in polar form

$$\varphi_\omega := \arg\max_{\varphi\in[0,1)} \left(\int_{t\in\mathbb{R}} f(t)\cos_{\omega,\varphi}(t)\,dt \right)$$

Fourier representation (“synthesis”)

$$f(t) = \int_{\omega \in \mathbb{R}_{\geq 0}} d_\omega \sqrt{2} \cos(2\pi(\omega t - \varphi_\omega))\, d\omega$$
$$= \int_{\omega \in \mathbb{R}} c_\omega \exp(2\pi i \omega t)\, d\omega$$

Discrete case – plan

Our definition of the Fourier transform so far involves:

- integrals
- analog, CT-signals for **both** the input signal f(t) and the “prototype” sinusoids

Given what we know computers are better at, we:

1) replace integrals with sums
2) discretize any CT-signals using (equidistant) T-sampling

Discrete Fourier Transform – informally

signal

sampled signal

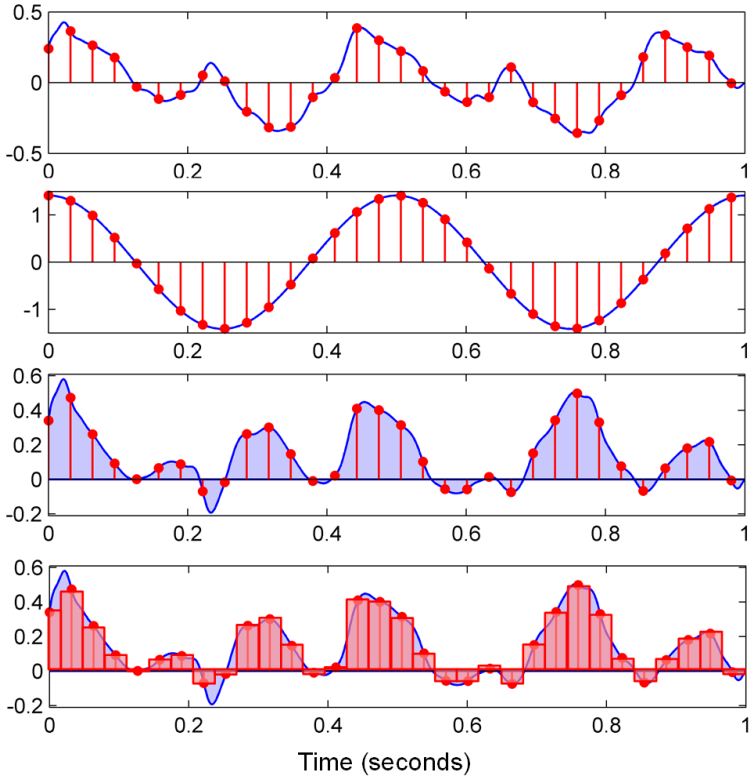
prototype

sampled prototype

pointwise product

area under pointwise product

approximation of area under pointwise product



[FMP21] Fig 2.7

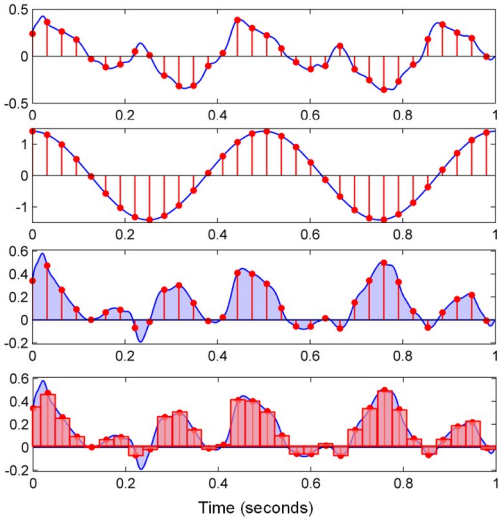
Discrete Fourier Transform – (1) approximating $\int$

Replace integral

$$\hat{f}(\omega) = \int_{t \in \mathbb{R}} f(t) \exp(-2\pi i \omega t)\, dt$$

with sum (“Riemann approximation”)

$$\hat{f}(\omega) \approx \sum_{n \in \mathbb{Z}} T\, f(nT) \exp(-2\pi i \omega n T)$$



[FMP21] Fig 2.7

Note: This is a useful approximation for signals that are “well-behaved” in a certain sense, and in these cases loses accuracy for “large” ω

Discrete Fourier Transform – (2) discretization

We use equidistant T-sampling to produce a discretization of the CT-signal, setting $T = 1$.

$$\hat{f}(\omega) \approx \sum_{n\in\mathbb{Z}} T\, f(nT) \exp(-2\pi i\omega nT) \qquad \hat{x}(\omega) := \sum_{n\in\mathbb{Z}} x(n) \exp(-2\pi i\omega n)$$

We relate the sampled FT back to the original function as follows:

$$\hat{x}(\omega) \approx \frac{1}{T}\hat{f}\left(\frac{\omega}{T}\right)$$

Corollary: when sampling at the Nyquist frequency, we do not need to consider all values of ω to fully capture the signal

Discrete Fourier Transform – further issues

We still have some problems:

- summation over an infinite number of samples
- ω is continuous, but we cannot consider an infinite amount of “prototypes”

Compromise:

- sum over a finite number of samples (N)
- sum over a finite number of frequencies (M)

Convention: Set $N = M$

Discrete Fourier Transform – final form

If x(n) is a DT-signal, and we have N samples bearing information to analyse, the **Discrete Fourier Transform** giving the k-th component is:

$$X(k) = \hat{x}(k/N) = \sum_{n=0}^{N-1} x(n) \exp(-2\pi i k n/N)$$

Discrete Fourier Transform – final form

If x(n) is a DT-signal, and we have N samples bearing information to analyse, the **Discrete Fourier Transform** giving the k-th component is:

$$X(k) = \hat{x}(k/N) = \sum_{n=0}^{N-1} x(n) \exp(-2\pi i k n/N)$$

This returns a complex number that we can take the magnitude of, which we use to determine the amount that the simple sound with frequency $F_{coef}(k)$ contributes to the signal

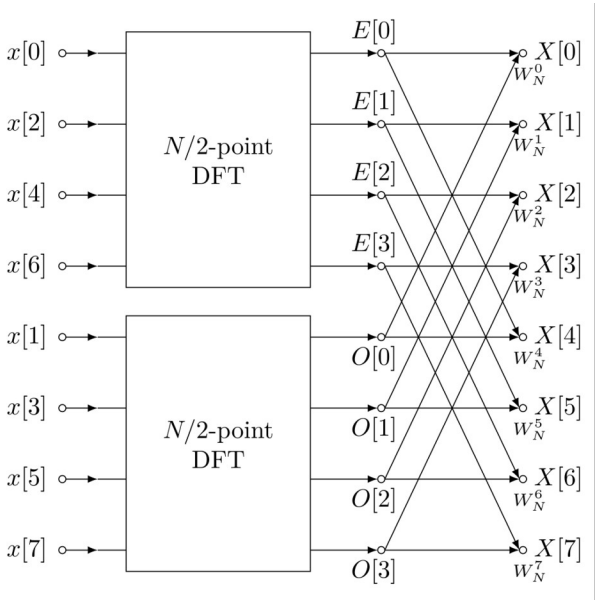
$$F_{\text{coef}}(k) := \frac{k}{N \cdot T} = \frac{k \cdot F_s}{N}$$

Fast Fourier Transform

Computing the coefficients for one frequency is linear in N , so computing all coefficients (that make sense) is quadratic.

But we can get O(N log N) for “nice” N, which explains why N is often set to some power of 2.

Fast Fourier transform algorithms (e.g. Cooley-Tukey) work by recursively splitting the DFT into smaller chunks and exploiting the periodicity of the complex exponentials



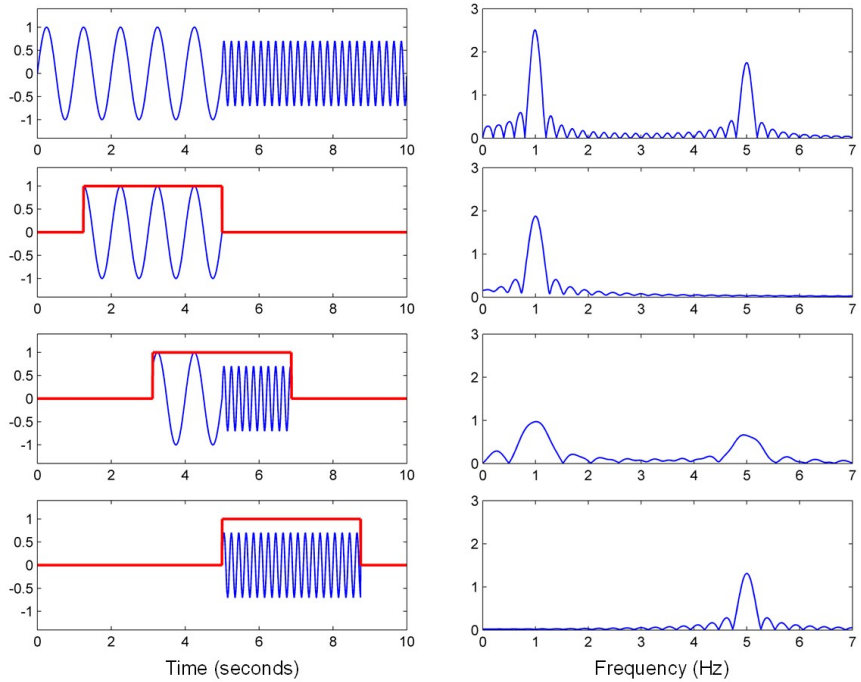
CC BY-SA 4.0/User:Yangwenbo99

Cooley, James W.; Tukey, John W. (1965). “An algorithm for the machine calculation of complex Fourier series”. *Math. Comput.* 19 (90): 297–301. doi:10.2307/2003354.

Windowing – idea

To preserve space (and to give interpretable frequency-domain representations) don't consider the whole waveform, but rather some sliding view on a subset of it. This means:

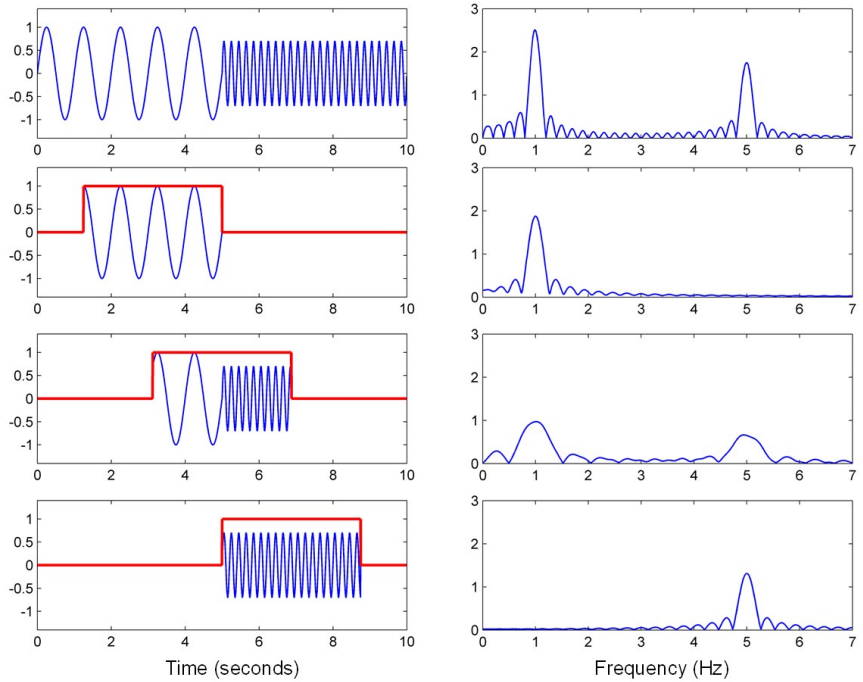
- a different **spectrum** depending on where (in time) we look;
- artifacts in the frequency-domain representation.



[FMP21] Fig 2.8

Windowing – implementation

We can implement a window as the product of the sampled signal and a window function – $x(n) \cdot w(n)$ – and apply it before the DFT is taken



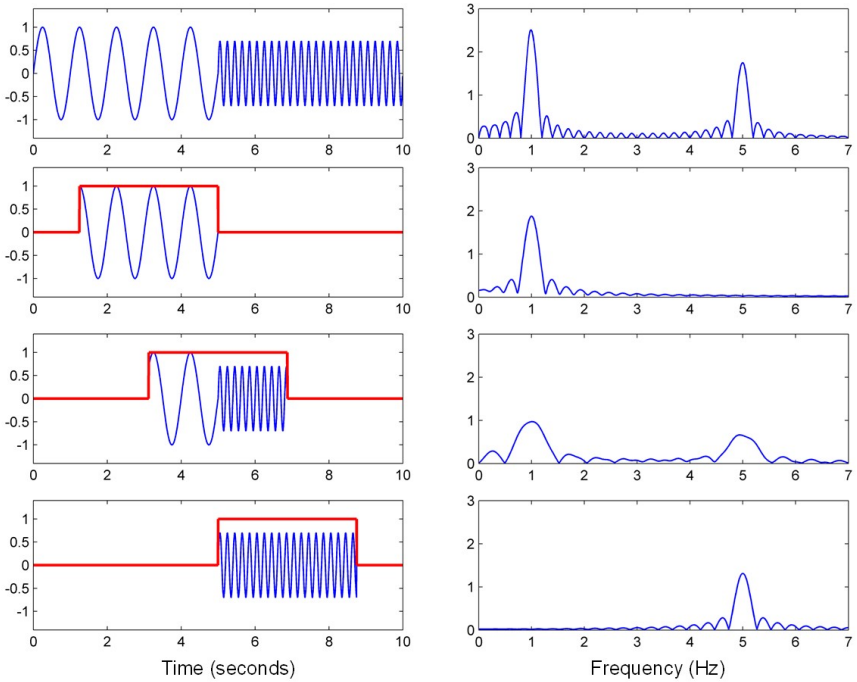
[FMP21] Fig 2.8

Windowing – implementation

We can implement a window as the product of the sampled signal and a window function – $x(n) \cdot w(n)$ – and apply it before the DFT is taken

Here, a **square window** is defined to be 1 on the range of indexes to be considered, and 0 elsewhere.

Smoother windows are used in practice to mitigate artifacts (e.g. **Hann** window)



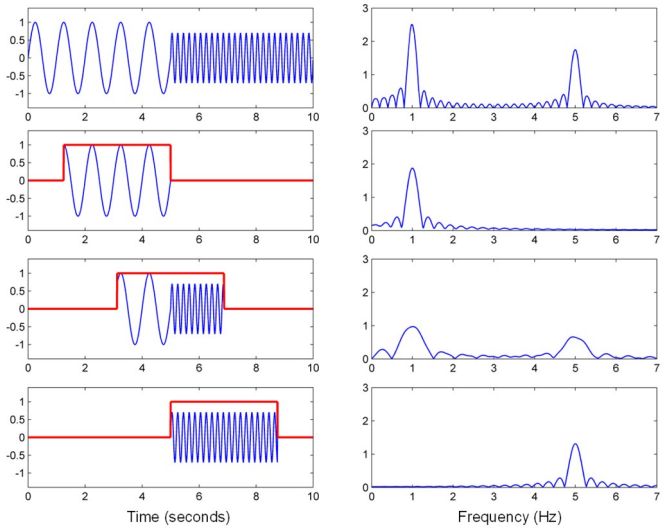
[FMP21] Fig 2.8

Relationship between STFT and spectrogram

Discrete short-time Fourier transform (STFT):

$$\mathcal{X}(m,k) = \sum_{n=0}^{N-1} x(n+mH)w(n)\exp(-2\pi ikn/N)$$

k : index into frequencies
m : index into STFT frames
n : index into DT-signal x
w(n) : window function
N : length of window (in samples)
H : hop size



[FMP21] Fig 2.8

Relationship between STFT and spectrogram

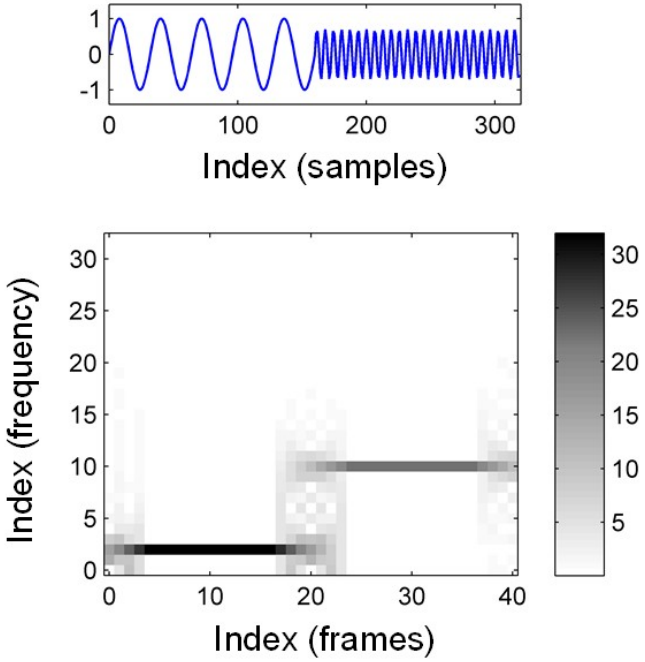
Recalling that the **STFT** X is a function of:

- time (here expressed in frames m)
- frequency (here expressed as index into frequencies, k)

that returns a complex number, which has a magnitude.

We define the **spectrogram** (or, “power spectrum”) as

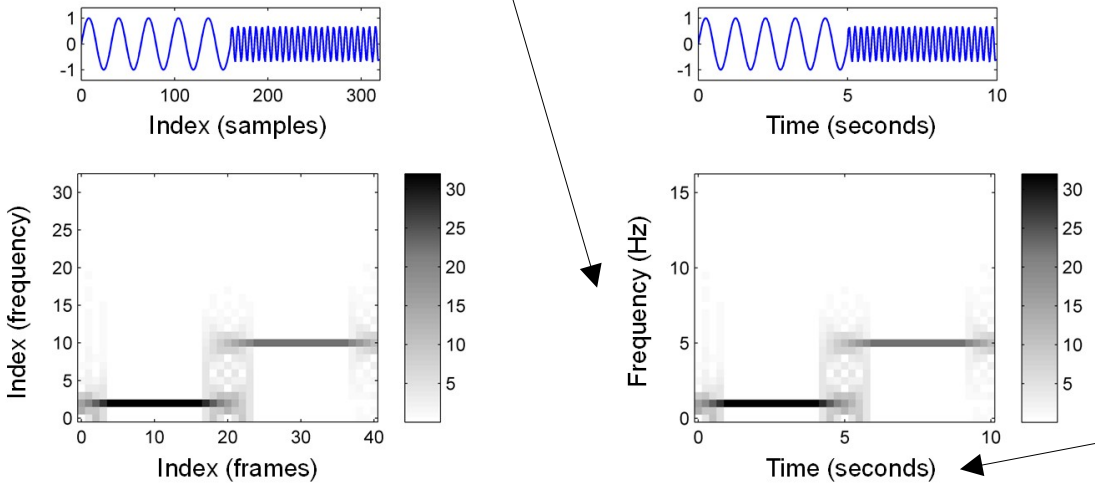
$$\mathcal{Y}(m,k) := |\mathcal{X}(m,k)|^2$$



[FMP21] Fig 2.9, adapted

STFT/spectrogram – interpretation of indices

$$F_{\text{coef}}(k) := \frac{k \cdot F_s}{N}$$



[FMP21] Fig 2.9

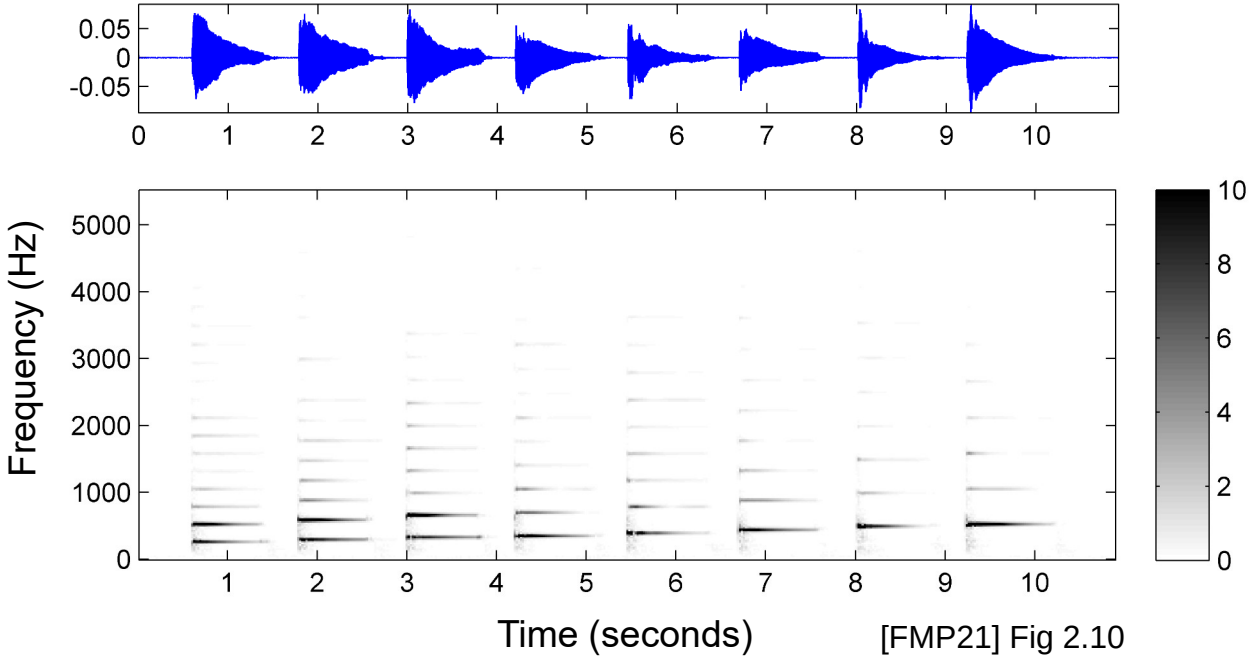
[FMP21] Fig 2.9

k : index into frequencies
m : index into STFT frames
N : length of window in samples
H : hop size
F_s : sample **rate**

$$T_{\text{coef}}(m) := \frac{m \cdot H}{F_s}$$

We often see H = N/2 as a common choice of hop length

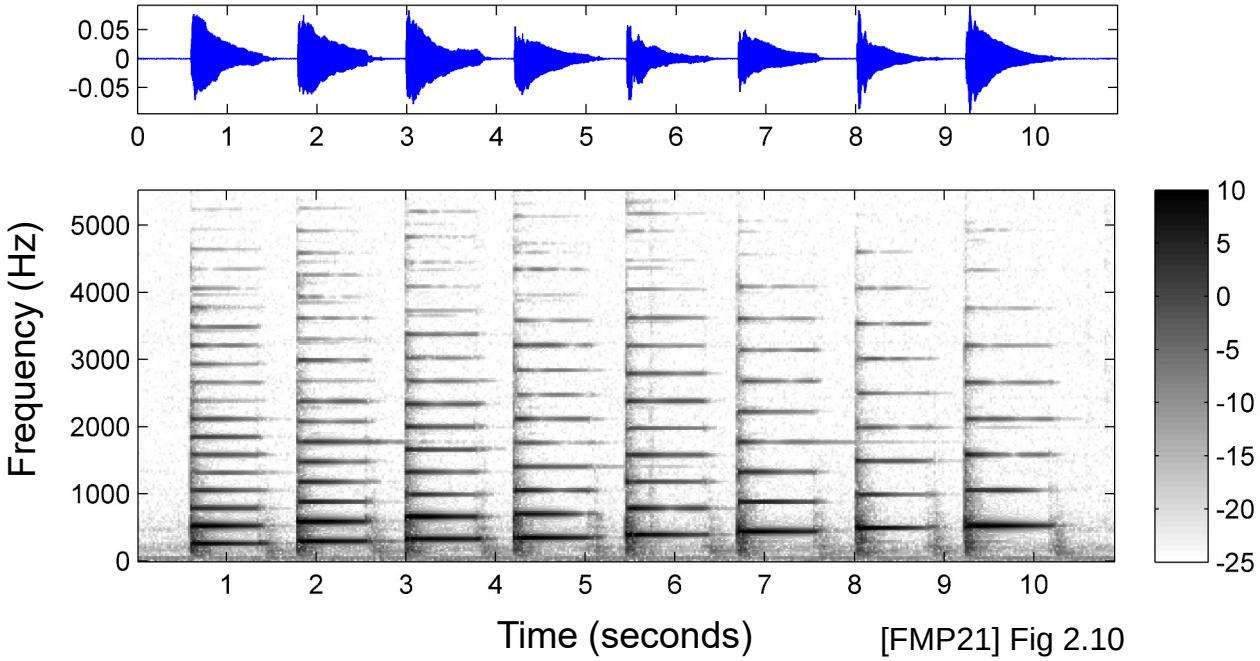
STFT/spectrogram – example



[FMP21] Fig 2.10

STFT/spectrogram – example

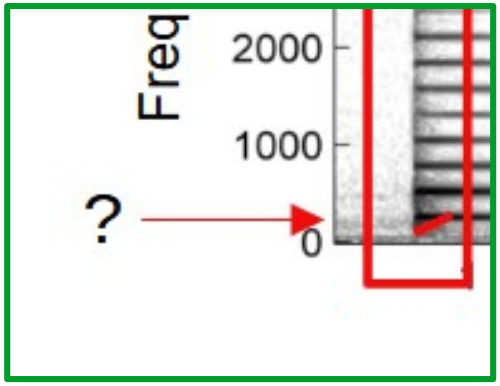
(We've rescaled the magnitude values, since the perception of loudness is not linear - and, in fact, depends on pitch)

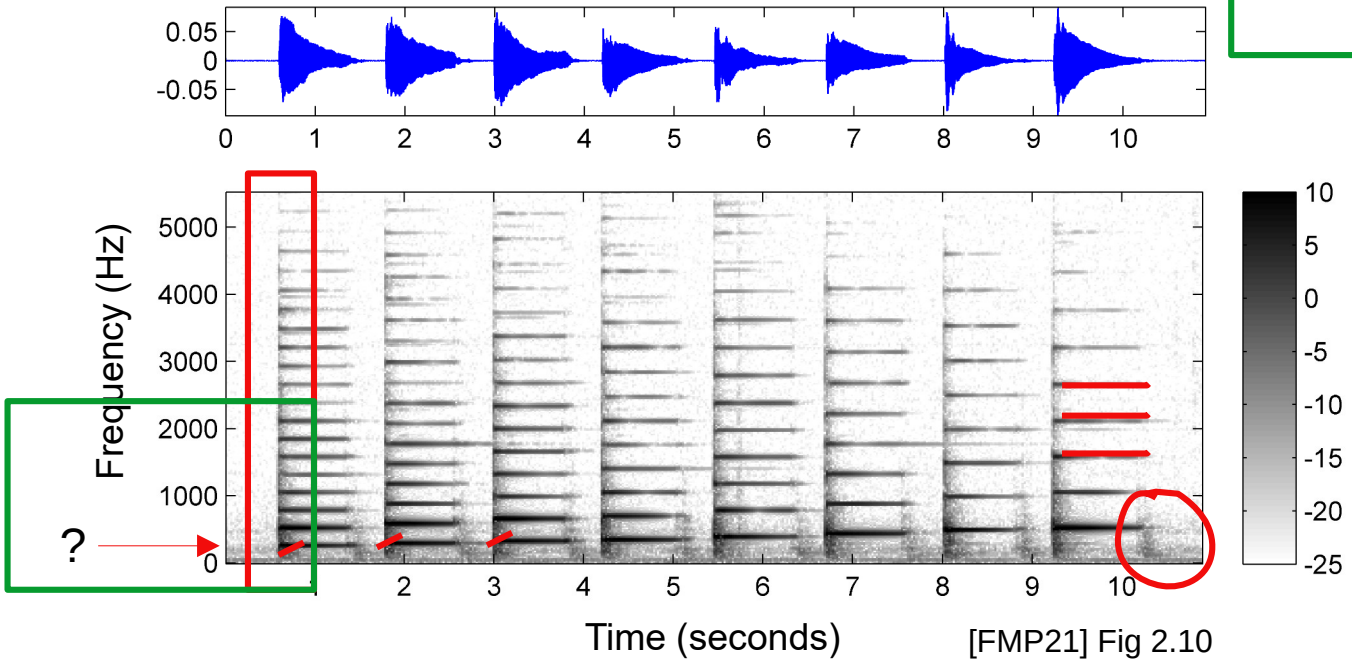


[FMP21] Fig 2.10

Durham University

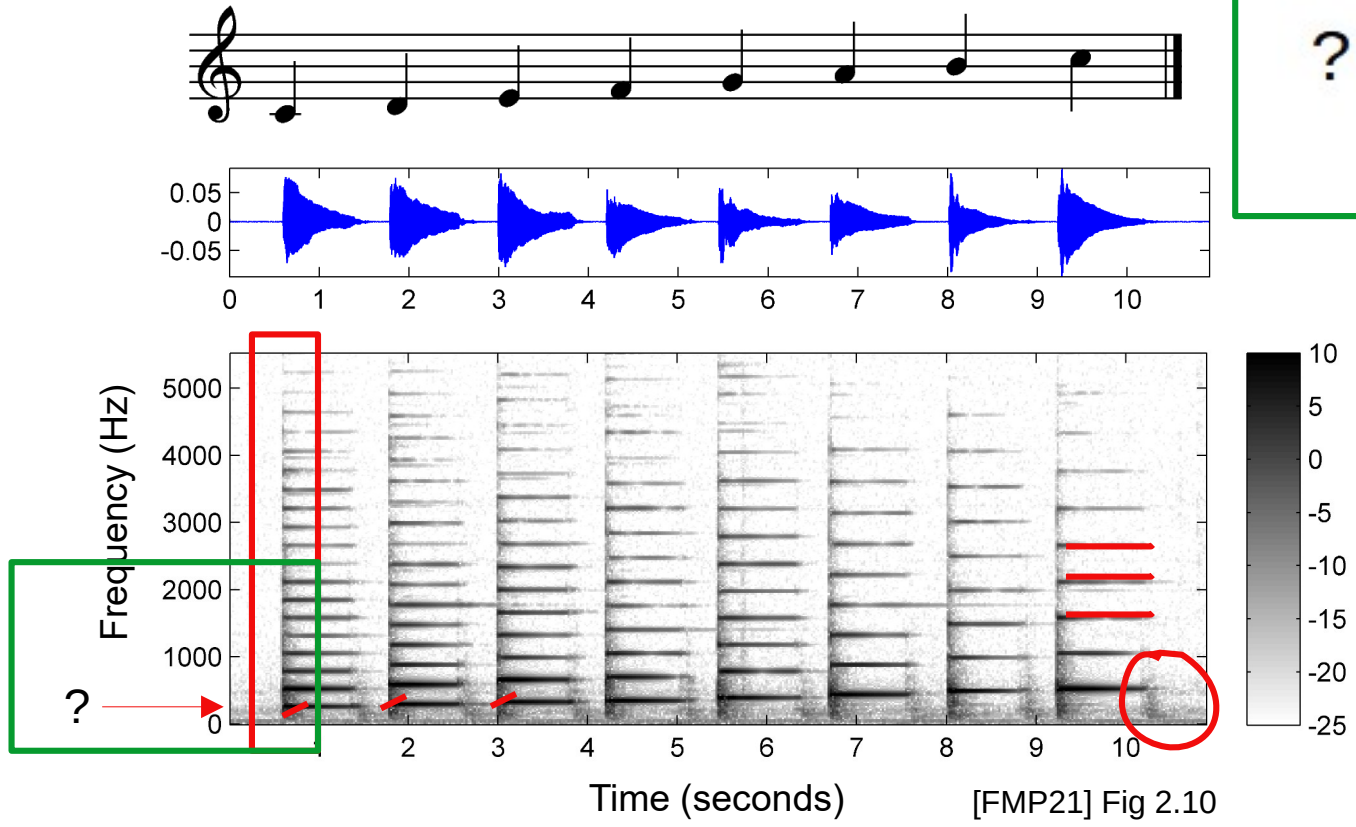
STFT/spectrogram – example

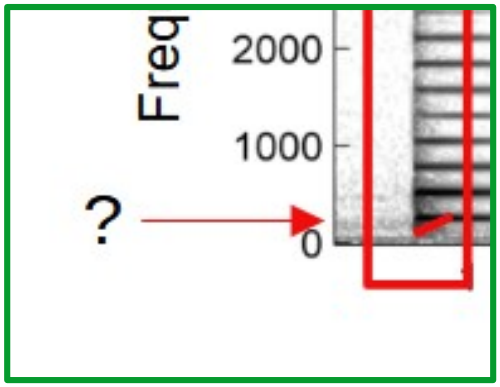




[FMP21] Fig 2.10

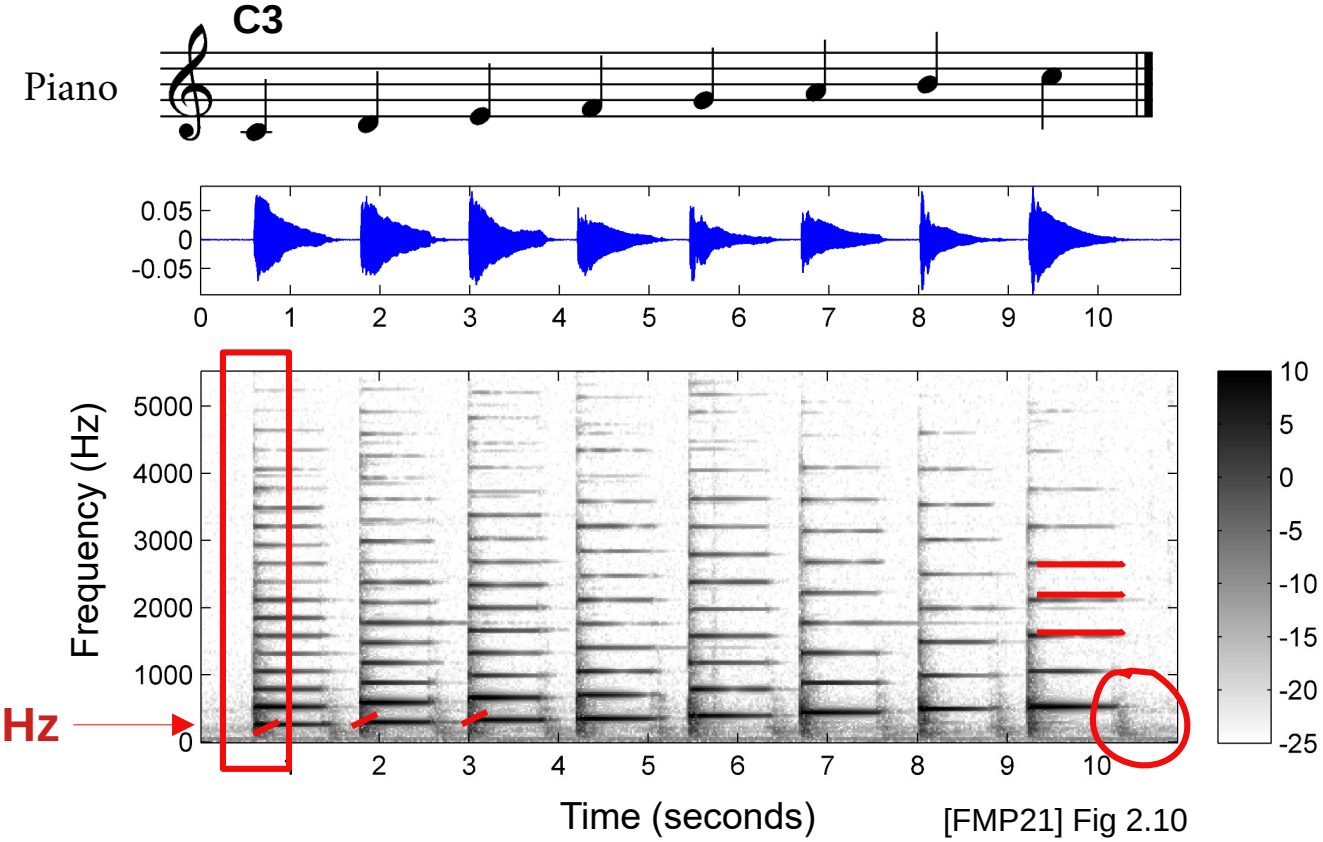
STFT/spectrogram – example





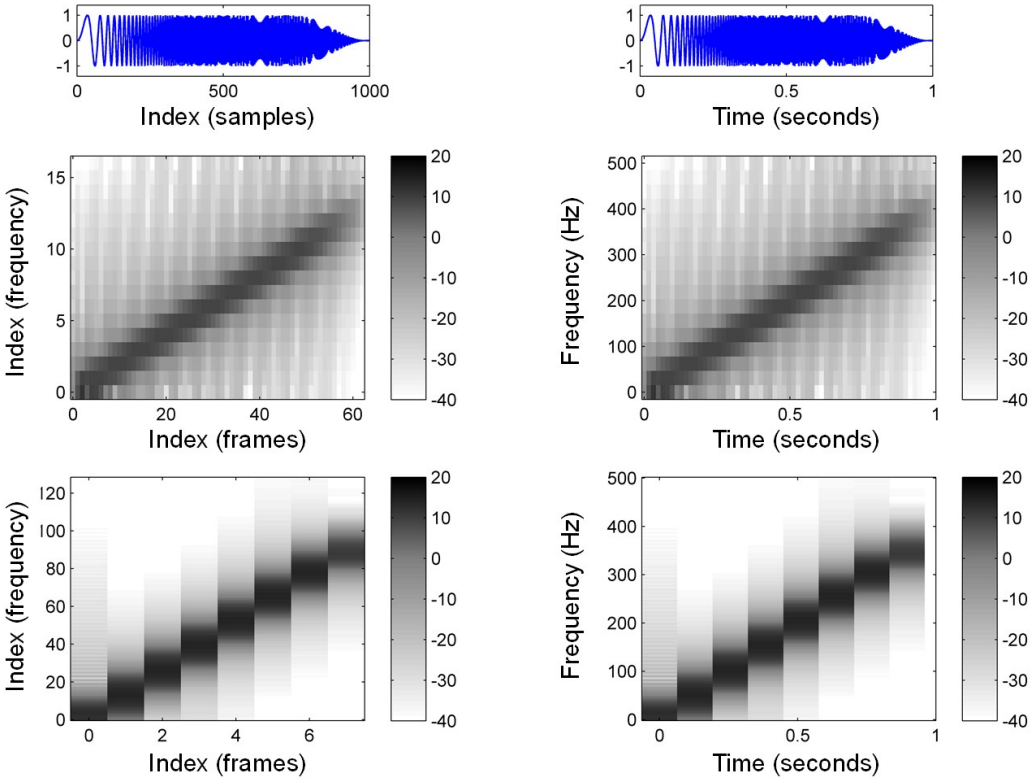
[FMP21] Fig 2.10

STFT/spectrogram – example



[FMP21] Fig 2.10

STFT/spectrogram – tradeoffs



N : length of window in samples

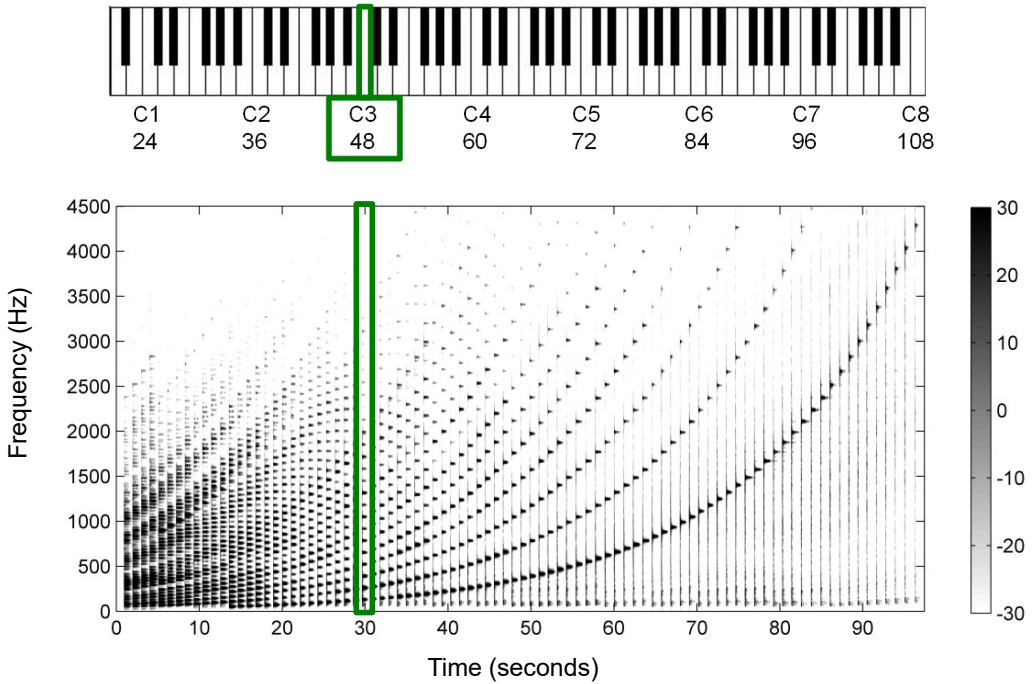
H : hop size

F_s : sample **rate**

[FMP21] Fig 2.33

Durham University

STFT of chromatic scale on piano



[FMP21] Fig 3.3

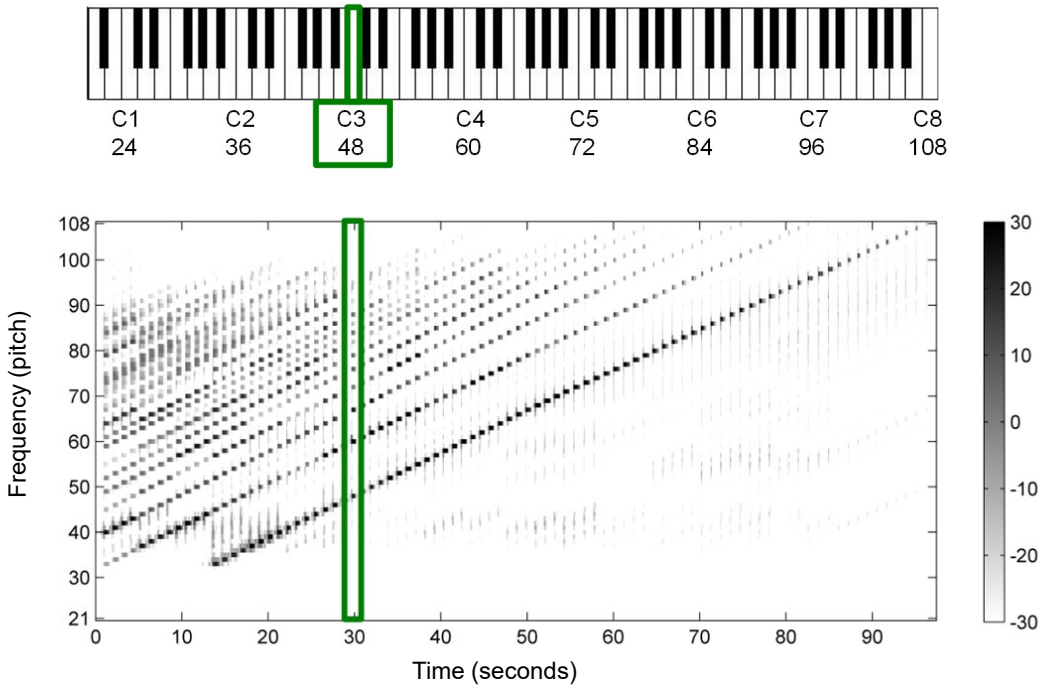
The piano produces a “complex” **sound** with **harmonic partials**. This magnitude **spectrogram** is the result of applying the **STFT** to the time-domain (**waveform**) representation of a digitally **sampled** recording of each note on a **12TET** piano played in succession.

The **harmonic** structure of the **partials** is discernable.

The **fundamental** is also evident, as well as the **logarithmic** relation between **center frequency** and **pitch**.

Durham University

Rebinning frequency of spectrogram



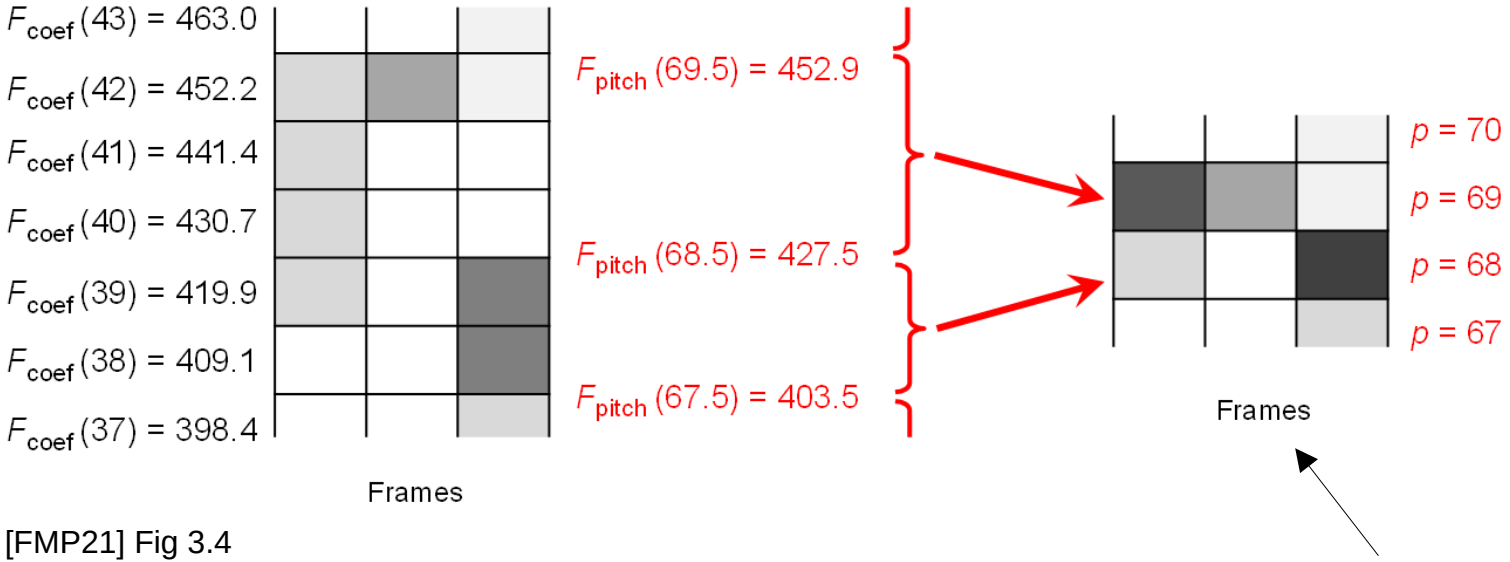
[FMP21] Fig 3.3

Given the same **STFT** (we do not change the dimensions of the frames), we can rebin the frequency axis so that magnitude values for various F_{coef} corresponding to MIDI note numbers are pooled

In effect, we take $\log_2$ of the frequency axis ticks, to produce a **log-frequency spectrogram** that better represents the perceptual story

Durham University

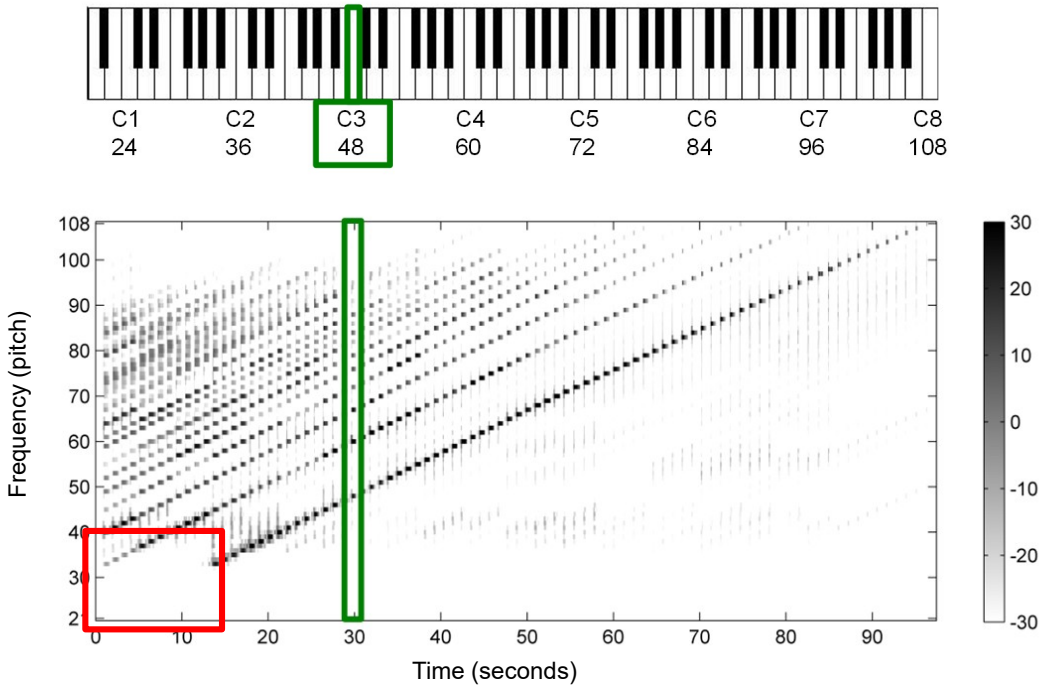
Rebinning frequency of spectrogram - details



[FMP21] Fig 3.4

"partitioned logarithmically, labelled linearly"

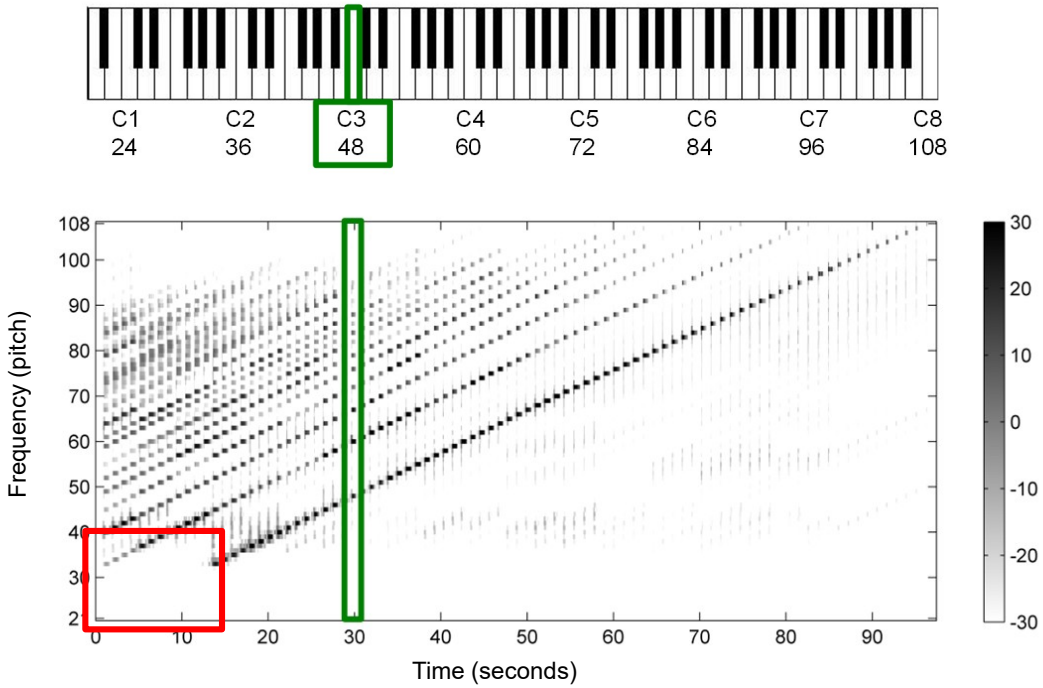
Rebinning frequency of spectrogram – issues?



[FMP21] Fig 3.3

What's going on at the lower end?

Rebinning frequency of spectrogram – issues?

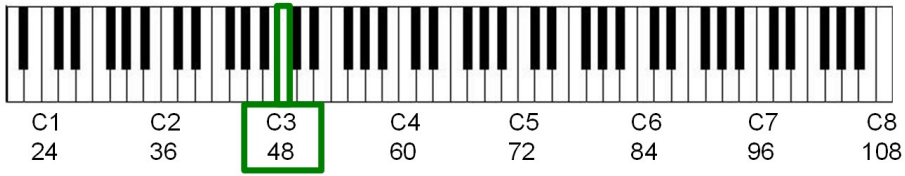


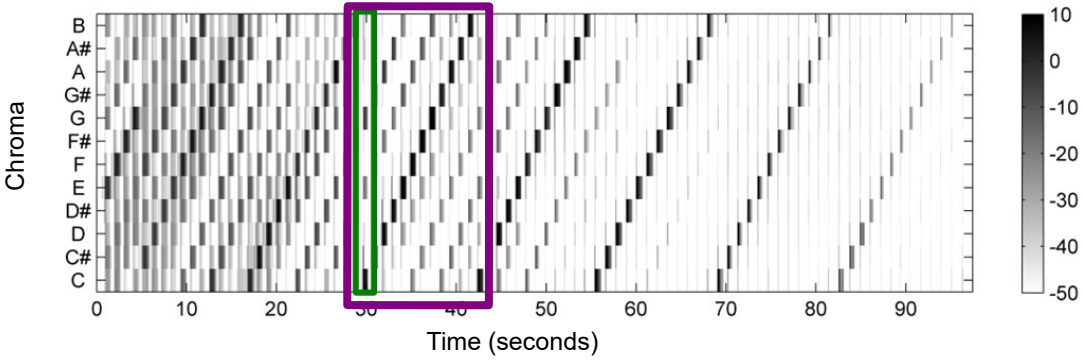
[FMP21] Fig 3.3

What's going on at the lower end?

- frequency response of microphone: not sensitive to low frequencies?
- energy concentrated in upper partials for low notes (cf. "missing fundamental")
- result of binning (potential solution: multiresolution STFT)

Summing energy over bins that are octave-related





[FMP21] Fig 3.3

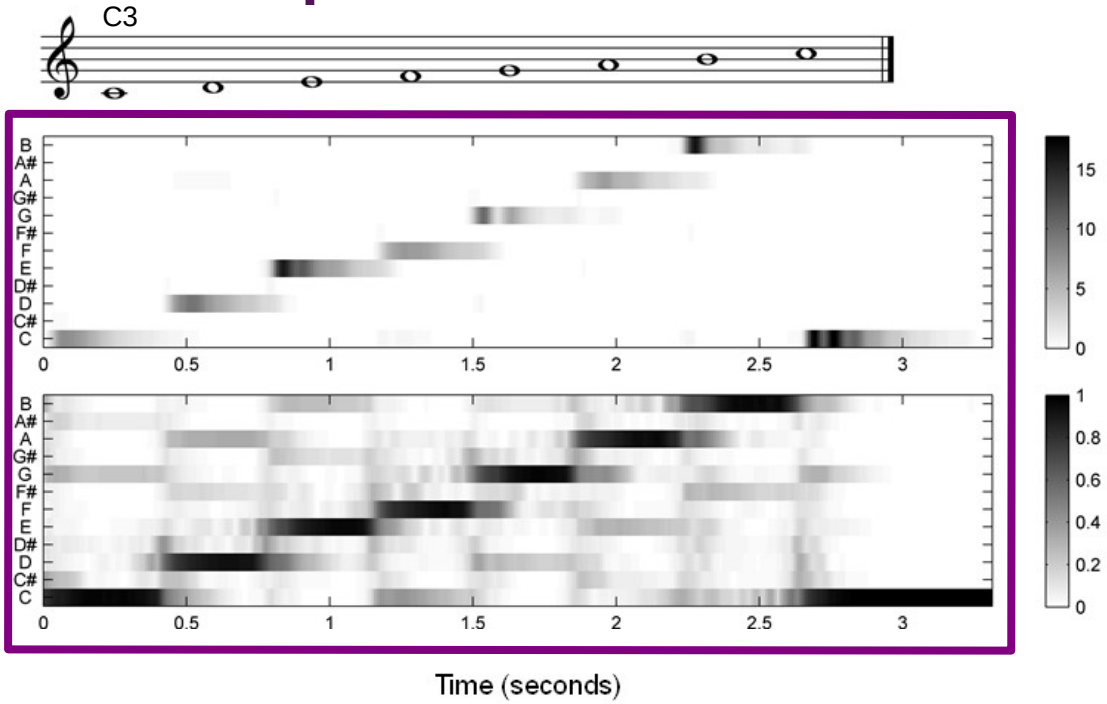
Sum up the magnitudes of all bins that correspond to the same **pitch class** (~ **chroma**), which is the letter part of the SPN name for each pitch.

Now each frame is a represented by 12-dimensional vector populated with a magnitude value for each chroma

$$C(t = 30) \approx [8,\ 1,\ 0.5,\ 0.2,\ 4,\ 0.1,\ 0.05,\ 6.5,\ 0.,1,\ 0.5,\ 0.6,\ 0.2]$$

Durham University

Applications of chroma features – (1) – monophonic transcription



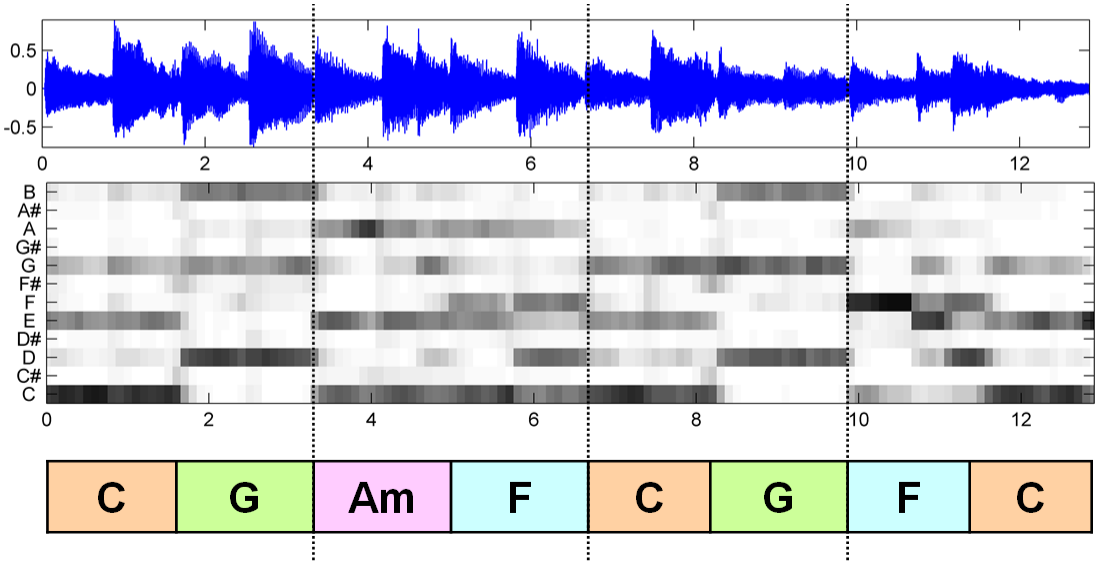
[FMP21] Fig 3.8 (adapted)

Chroma features can clearly be used to detect the **pitch class** of single notes (i.e. ignoring the octave in which that note appears).

These two plots show the results of two different normalisation schemes.

Though the diatonic scale (C, D, E, etc.) is clear in both, the second plot depicts artifacts – why?

Applications of chroma features – (2) – chord recognition



$$\mathbf{C}_{\text{template}} := \{C, E, G\} := [1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0] \in \{0, 1\}^{12}$$
$$\mathbf{G}_{\text{template}} := \{G, B, D\} := [0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1] \in \{0, 1\}^{12}$$
$$\mathbf{Am}_{\text{template}} := \{A, C, E\} := [1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0] \in \{0, 1\}^{12}$$
$$\vdots$$

[FMP21] Fig 5.1 (adapted)

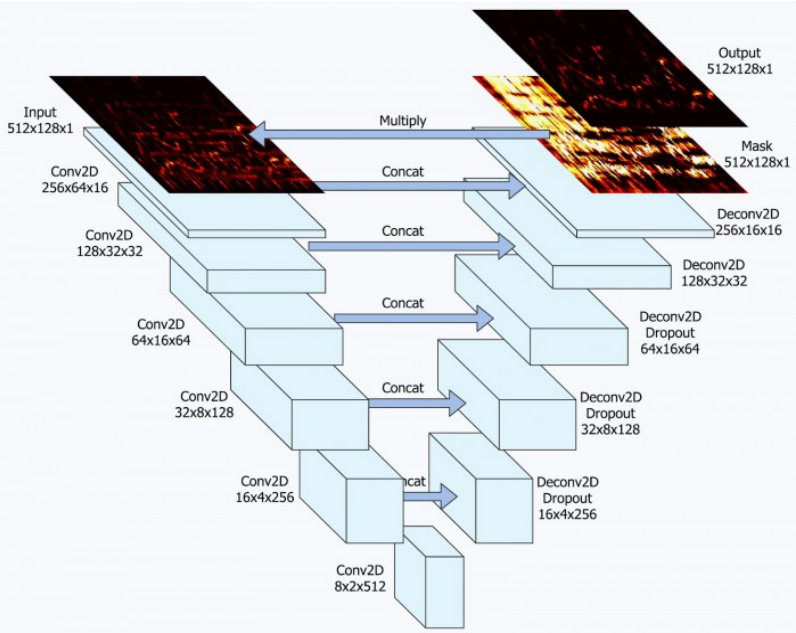
Chords are combinations of pitch classes (roughly), so **chroma** features can be used to design algorithms that do automatic chord recognition (ACR)

At each time point, we can measure the observed chroma vector and compare it to a vocabulary of chord "templates" in clever ways

But there is a logic to the sequence too, meaning some chords are more likely to follow others. Idea: use/learn this information!

[FMP21] Fig 5.1 (adapted)

Audio processing as image processing?



Jansson et al. (2017) “Singing Voice Separation with Deep U-Net Convolutional Networks”, ISMIR 2017.

Sound source separation is “the process of isolating individual sounds in an auditory mixture of multiple sounds” [Manilow et al.] (cf. “cocktail party problem”, auditory scene analysis – Bregman)

(Magnitude) spectrograms can evidently be thought of as images, with classic computer vision approaches reused (e.g. semantic segmentation) to solve music processing tasks

But this requires a way to get from the spectrogram to the waveform (time-domain), such as the **Inverse Fourier Transform**

Recap

- Many musical sounds can be modeled well as a combination of simpler sounds (**sinusoids**), because they often have a principal periodic component
- The **frequency** of these **sinusoids** largely determines the perceived **pitch** of these sounds, while the number and relative **amplitude** of the most prominent of these sinusoids determine the **timbre** of these sounds
- **12-tone equal temperament** is the model for **pitch** relations embodied by the **piano keyboard**
- **Time-domain representations** of sound can be produced and manipulated using **sampling** (perhaps at the **Nyquist rate**), and visualised as **waveforms**
- Frequency-domain representations are produced by the **Fourier Transform**, which has **continuous** and **discrete** forms
- **Magnitude spectrograms** reveal the harmonic structure of digitized musical sounds, and are produced by the **Short-Time (Discrete) Fourier Transform** in practice