From Beat Notes to Wavelet Decomposition: A Self-Contained Walk Through Sines, Windows, and Classic Wavelets

Contents

1 Combined Sine Waves and Beat Notes

Definition. Let $f_1, f_2 \in \mathbb{R}_{>0}$ be two pitch frequencies in hertz (cycles per second). A *pure sinusoid* at frequency f is $\sin(2\pi f t)$. The constant π (Greek letter *pi*) was first used for the circle ratio by William Jones in 1706 and has since become universal for angular calculations.

Beat-note identity. Adding two pure sinusoids gives

$$\sin(2\pi f_1 t) + \sin(2\pi f_2 t) = 2\cos\Big(2\pi\frac{f_1 - f_2}{2}\,t\Big)\,\sin\Big(2\pi\frac{f_1 + f_2}{2}\,t\Big).$$

The factor $\cos(2\pi|f_1 - f_2|t/2)$ modulates amplitude at the *beat frequency* $|f_1 - f_2|$. When f_1 and f_2 differ only slightly, the amplitude rises and falls slowly; when they differ greatly, no slow modulation is audible or visible in the waveform.

Listing 1: One-second beat note at 110 Hz and 116 Hz

Concrete code example.

```
import numpy as np, matplotlib.pyplot as plt
from IPython.display import Audio

sr = 44100 # sample rate (Hz)
f1, f2 = 110, 116 # A2 and  B2
t = np.linspace(0, 1.0, sr, endpoint=False)
sig = np.sin(2*np.pi*f1*t) + np.sin(2*np.pi*f2*t)
sig = sig / np.max(np.abs(sig)) # normalise

plt.plot(t, sig)
plt.title("110 Hz + 116 Hz -beat  6 Hz")
plt.xlabel("time (s)"); plt.ylabel("amplitude")
plt.show()

Audio(sig, rate=sr)
```

Piano demonstration. Press A_2 (110 Hz) together with $B\flat_2$ (approximately 116 Hz). A volume undulation at ≈ 6 Hz is heard; this matches the cos-modulation predicted above.

2 Need for Temporal Localisation and the Gaussian Window

Problem statement. Beat analysis reveals *global* interference patterns, but real data (speech syllables, seismic reflections, image edges) require knowing *when* a feature occurs. A window function $w(t)$ multiplies the signal, confining analysis to a finite time span.

Gaussian window.

$$w(t) = \exp\Big(-\frac{t^2}{2\sigma^2}\Big).$$

The Greek letter σ (*sigma*) denotes *standard deviation*, introduced with that meaning by Karl Pearson in 1894. Carl Friedrich Gauss published this bell formula in 1809 while modelling astronomical error distributions.

Heisenberg–Gabor limit. Werner Heisenberg (1927) proved for any square-integrable $g(t)$

$$(\Delta t)(\Delta f) \;\geq\; \tfrac{1}{2},$$

where Δt and Δf are root-mean-square spreads in time and frequency. Dennis Gabor (1946) applied the same inequality to signal processing. The Gaussian attains equality, giving the best possible simultaneous concentration in both domains.

Listing 2: Plot of Gaussian window with = 0.25 s

Concrete code example.

```
sigma = 0.25
t = np.linspace(-1, 1, sr*2, endpoint=False)
g = np.exp(-(t**2)/(2*sigma**2))

plt.plot(t, g)
plt.title(r"Gaussian $e^{-t^{2}/2\sigma^{2}}$ ($\sigma = 0.25\,$s)")
```

```
plt.xlabel("time (s)"); plt.ylabel("amplitude")
plt.show()
```

3 Euler's Formula and the Argand Plane Rotation

Leonhard Euler (1748) established

$$e^{i\theta} = \cos\theta + i\sin\theta,$$

where θ (Greek *theta*, adopted by Euler for angles) measures rotation in radians. Jean-Robert Argand (1806) introduced the geometric representation of complex numbers: the point $(\cos\theta, \sin\theta)$ is a unit vector making angle θ with the positive x-axis.

Define

$$z(t) = w(t)\, e^{i\omega t}, \qquad \omega = 2\pi f,$$

with ω (Greek *omega*) chosen for angular frequency by William Thomson (Lord Kelvin) c. 1867. Here $|z(t)| = w(t)$ and the argument of $z(t)$ advances at ω.

Animated Argand arrow. Insert the following in a Colab cell; it shows the arrow growing and shrinking with $w(t)$ while rotating.

Listing 3: Argand rotation with Gaussian magnitude

```
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import numpy as np
from IPython.display import HTML

sigma, f = 0.25, 1.0
duration, fps = 2.0, 30
frames = int(duration*fps)
ts = np.linspace(-duration/2, duration/2, frames)
radii = np.exp(-(ts**2)/(2*sigma**2))
angles = 2*np.pi*f*ts

fig, ax = plt.subplots(figsize=(4,4))
ax.set_aspect("equal"), ax.axis("off")
ax.set_xlim(-1.1,1.1), ax.set_ylim(-1.1,1.1)
circle = plt.Circle((0,0), 1, color="lightgray", fill=False, ls="--")
ax.add_patch(circle)
line, = ax.plot([], [], lw=3, color="darkorange")

def update(i):
    r, th = radii[i], angles[i]
    x, y = r*np.cos(th), r*np.sin(th)
    line.set_data([0,x],[0,y]); line.set_alpha(max(r,0.05))
    return line,
ani = FuncAnimation(fig, update, frames=frames, blit=True, interval=1000/fps)
HTML(ani.to_jshtml())
```

4 Gaussian-Windowed Cosine–Sine Pair (Morlet Packet)

Definition.

$$\psi_{\cos}(t) = w(t)\cos\omega t, \qquad \psi_{\sin}(t) = w(t)\sin\omega t.$$

Both phases are required: a local signal might begin at a maximum (cosine-aligned) or at a zero-crossing (sine-aligned).

Listing 4: Plot & play real part of a Morlet packet

Concrete code example.

```
def morlet_real(t, f=440, sigma=0.05):
    w = np.exp(-(t**2)/(2*sigma**2))
    return w*np.cos(2*np.pi*f*t)

dur = 0.4
t = np.linspace(-dur/2, dur/2, int(sr*dur), endpoint=False)
sig = morlet_real(t)

plt.plot(t, sig)
plt.title("Morlet (real) 440 Hz  =0.05 s")
plt.xlabel("time (s)"); plt.ylabel("amplitude"); plt.show()
Audio(sig/np.max(np.abs(sig)), rate=sr)
```

5 Classic Wavelet Forms and Their Origins

5.1 Mexican-Hat (Ricker) Wavelet

$$\psi_{\mathrm{MH}}(t) = \Big(1 - \frac{t^2}{\sigma^2}\Big)\exp\Big(-\frac{t^2}{2\sigma^2}\Big).$$

History. Norman Ricker (1953) devised this even-symmetric pulse while investigating seismic reflection shapes.

Current use. Detecting circular blobs in images; spotting P- and S-wave arrivals in modern seismology.

5.2 Morlet Wavelet

Complex Gaussian packet (1978–1984) by petroleum geophysicist Jean Morlet, formalised with Alex Grossmann. Enables the continuous wavelet transform (CWT) now standard in EEG microstate analysis.

5.3 Haar Wavelet

Alfréd Haar (1909) introduced the first orthonormal step function for his thesis on orthogonal series. The mother wavelet is $\psi(t) = 1$ on $[0, \frac{1}{2})$, -1 on $[\frac{1}{2}, 1)$, 0 otherwise. Widely used in lossless image codecs.

5.4 Daubechies Wavelets

Ingrid Daubechies (1987) created compact-support, smooth, orthogonal wavelets; db4 uses filter $h = [0.4829629, 0.8365163, 0.2241439, -0.1294095]$. Employed by the FBI for fingerprint compression and in numerical PDE schemes.

6 First-Level Haar Discrete Wavelet Transform

For four samples $x = [x_0, x_1, x_2, x_3]$,

$$\text{averages} = \frac{x_0 + x_1}{2},\ \frac{x_2 + x_3}{2}, \qquad \text{details} = \frac{x_0 - x_1}{2},\ \frac{x_2 - x_3}{2}.$$

Listing 5: Manual 4-sample Haar DWT

```
x = np.array([7., 5., 1., 1.])
avg = (x[0::2] + x[1::2]) / 2 # [6, 1]
detail = (x[0::2] - x[1::2]) / 2 # [1, 0]
coeffs = np.concatenate([avg, detail])
print(coeffs) # -> [6. 1. 1. 0.]
```

7 Wavelet Decomposition Beyond Level 1

Apply the same average/detail split recursively to the coarse block, doubling time resolution at each level. For practical work use a vetted library (e.g. PyWavelets), but every algorithm obeys the step illustrated above.

Summary

Starting from two audible sine waves we derived beat modulation, then introduced Gaussian time-windows to control temporal locality, respecting the Heisenberg–Gabor uncertainty limit. Euler's formula and the Argand plane provided a geometric view of complex oscillations, motivating the Gaussian-windowed cosine–sine pair (Morlet). We reviewed historically significant wavelets—Mexican-Hat, Morlet, Haar, Daubechies—and executed a hand calculation of the first-level Haar transform.

All code samples can be typed directly into a Google Colab notebook and will produce the stated plots and audio widgets.