From Beats to Diffraction
A Journey Through Waves

June 18, 2025

Prologue

We will travel through sound, light, and water the way early scientists did—asking *why?* at every turn, inventing just **one new piece of maths** at a time, and checking our ideas with code you can run in Google Colab.

1 The First Mystery: Why Two Notes Throb (Audio Beats)

A Little Scene from 1787

Picture *Ernst Chladni* striking two tuning-forks that disagree by the tiniest smidge. The air grows loud–soft–loud–soft like a giant breathing. Chladni wants to explain that pulse. Radio pioneer *Reginald Fessenden* will later harness the same trick (~1900) to ride news and music on invisible waves.

What We Are About to Do

We will **multiply** a fast wiggle (the *carrier*) by a slow wiggle (the *envelope*). When the envelope is high, the carrier grows; when it is low, the carrier shrinks. That arithmetic trick is called *amplitude modulation*.

The One New Formula

$$y(t) = A\,\sin(2\pi f_c t)\,\cos(2\pi f_e t) \tag{1}$$

How it works:

- A: a constant that multiplies the combined wave to control overall loudness.
- t: the running clock on your stopwatch.
- f_c: carrier frequency—how many tiny wiggles per second (440 Hz is the musical A).
- f_e: envelope frequency—how many loud–soft cycles per second.
- π: lower-case Greek pi (first printed by *William Jones*, 1706).

Workable Examples

1. **Finding loudness at two instants.** With $A = 1$, $f_c = 440\,\text{Hz}$ and $f_e = 5\,\text{Hz}$, compute $y(0.002\,\text{s})$ and $y(0.004\,\text{s})$ using the formula.
2. **Open question.** If we double the envelope frequency to 10 Hz, how does the time between loud peaks change?

Python Check-up (Colab)

Listing 1: Beats in Python

```python
import numpy as np, matplotlib.pyplot as plt

A, fc, fe = 1, 440, 5
t = np.linspace(0, 0.5, 20_000)
y = A*np.sin(2*np.pi*fc*t) * np.cos(2*np.pi*fe*t)

plt.plot(t*1e3, y)
plt.xlabel('Time (ms)')
plt.ylabel('Amplitude')
plt.title('Audio Beat Wave')
plt.show()
```

2 The Bell-Shaped Secret Window (Gaussian)

Gauss and the Star That Wouldn't Sit Still (1809)

Astronomer *Carl Friedrich Gauss* tried to guess where the dwarf planet Ceres had gone after it slipped behind the Sun. He needed a maths tool that gives *full credit to the middle* and *little credit to the edges*: the famous bell curve.

What We Are About to Do

We will swap the cosine envelope for a **smooth bell** that starts small, swells gently, and falls away without a bump, producing a *single burst* instead of endless throbs.

The One New Formula

$$G(t) = \exp\bigl(-t^2/(2\sigma^2)\bigr) \tag{2}$$

How it works:

- Inside the exponent, t^2 grows fast as you walk away from zero; the minus sign makes $G(t)$ shrink.
- σ (sigma, lower-case; adopted for standard deviation by *Karl Pearson*, 1894) sets how wide the hill is—larger σ makes a wider bell.

Workable Examples

1. **Half-second hill.** With $\sigma = 0.1\,\mathrm{s}$ find $G(0)$ and $G(0.1)$.
2. **Try widening the hill.** Double σ to $0.2\,\mathrm{s}$. How tall is $G(0.1)$ now?

Python Sketch

Listing 2: Gaussian envelope

```python
import numpy as np, matplotlib.pyplot as plt

sigma = 0.1
t = np.linspace(-0.5, 0.5, 1_000)
G = np.exp(-(t**2) / (2*sigma**2))

plt.plot(t, G)
plt.title('Gaussian Envelope')
plt.xlabel('t (s)')
plt.ylabel('G(t)')
plt.show()
```

3 A Single Audible Ping (Gaussian-Windowed Sine)

Gabor's 1946 Thought Experiment

Engineer *Dennis Gabor* imagined cutting a sine wave out of time with a bell window so he could paint pictures of sound in both time *and* frequency. That idea won him a Nobel Prize and later gave birth to MP3s.

What We Are About to Do

Glue the hill from Section 2 on top of the sine from Section 1. When the hill is high, the tone is loud; when low, silence.

The One New Formula

$$y(t) = A\,e^{-t^2/(2\sigma^2)}\,\sin\bigl(2\pi f_c t\bigr) \tag{3}$$

Workable Examples

1. **Quick ping.** With $f_c = 440\,\text{Hz}$ and $\sigma = 0.01\,\text{s}$, count the number of crest-to-crest oscillations under the bell. (Hint: audible span $\approx 6\sigma$.)
2. **Your turn.** Make the bell five times wider. How many oscillations now?

Python — Listen with Your Eyes

Listing 3: Gaussian-windowed tone

```python
import numpy as np, matplotlib.pyplot as plt

A, fc, sigma = 1, 440, 0.01
t = np.linspace(-0.05, 0.05, 4_000)
pulse = A*np.exp(-(t**2)/(2*sigma**2)) * np.sin(2*np.pi*fc*t)

plt.plot(t*1e3, pulse)
plt.xlabel('Time (ms)')
plt.title('Gaussian-Windowed Audio Pulse')
plt.show()
```

4 Slowing the Music for Water (Scaling Frequency)

George Airy Watches Ocean Swells (1845)

On a windy day the sea looks nothing like a violin string. *Sir George Airy* noticed that water waves unfold in slow, heavy packets—perfect cousins to our audio ping but with far fewer wiggles.

What We Are About to Do

Keep the same bell but **stretch time** so the sine hardly has time to wiggle. The trick: divide the frequency by a big number N.

The One New Formula

$$f_w = \frac{f_c}{N}, \qquad N \gg 1 \tag{4}$$

Workable Examples

1. **Make 440 Hz crawl.** With $N = 100$, compute f_w and sketch a water pulse under $\sigma = 0.5$ s.
2. **Challenge.** What if $N = 500$?

Python Ripple

Listing 4: Slow ripple

```python
import numpy as np, matplotlib.pyplot as plt

A, fc, N, sigma = 1, 440, 100, 0.5
fw = fc / N
t = np.linspace(-4, 4, 8_000)
water = A*np.exp(-(t**2)/(2*sigma**2)) * np.sin(2*np.pi*fw*t)

plt.plot(t, water)
plt.title('Gaussian-Windowed Water Pulse')
plt.xlabel('t (s)')
plt.show()
```

5 From Line to Circle — The Pebble Splash (2-D Ripple)

Poisson & Fresnel Draw Ripples (1818)

Working for Napoleon's map-makers, *Siméon Poisson* solved how a single splash spreads; *Augustin-Jean Fresnel* soon borrowed the idea for light.

What We Are About to Do

Drop our Section 3 pulse at the origin of a pond. The crest now forms a **circle**. We introduce the word *wavefront*: the thin line marking everywhere the crest has reached.

The One New Formula

$$u(r,t) = A\, e^{-(r-vt)^2/(2\sigma^2)} \cos\bigl(k(r-vt)\bigr) \tag{5}$$

How it works:

- $r = \sqrt{x^2+y^2}$ is distance from the splash.
- v is wave speed.
- $k = 2\pi/\lambda$ is the wave-number (*Scott Russell*, 1844).
- λ is wavelength (lower-case lambda; *William Thomson*, 1850).

Workable Examples

1. **Pebble numbers.** With $v = 0.3$ m/s, $\lambda = 0.1$ m, $\sigma = 0.05$ m, find the crest radius at $t = 1$ s and compute k.
2. **Double λ.** How does ring spacing change?

Python Pond Picture

Listing 5: Circular ripple

```
import numpy as np, matplotlib.pyplot as plt

A, v, lam, sigma, t = 1, 0.3, 0.1, 0.05, 1.0
k = 2*np.pi / lam
```

```
x = y = np.linspace(-0.5, 0.5, 400)
X, Y = np.meshgrid(x, y)
r = np.sqrt(X**2 + Y**2)
U = A*np.exp(-((r - v*t)**2)/(2*sigma**2)) * np.cos(k*(r - v*t))

plt.imshow(U, extent=[-0.5, 0.5, -0.5, 0.5], origin='lower', cmap='RdBu')
plt.colorbar(label='Amplitude')
plt.title('Circular Ripple at t = 1 s')
plt.xlabel('x (m)')
plt.ylabel('y (m)')
plt.show()
```

6 From 1-D to 2-D – Circular Ripple Packet

What We Are About to Do

We let the Gaussian-windowed tone from Section 4 explode outward in every direction. Each crest is now a **circle** whose radius grows at the wave-speed v. The bell envelope still keeps the packet finite, so it travels as a neat, moving hill on the pond.

The One New Formula

$$U(r,t) = A\,\exp\!\Big[-\frac{(r-vt)^2}{2\sigma^2}\Big]\,\cos\!\big(k(r-vt)\big) \tag{6}$$

How it Works

- $r = \sqrt{x^2+y^2}$ is how far you are from the splash point.
- The bell centre sits at $r = vt$; that term makes the packet *move* without changing shape.
- The exponential provides the smooth bell. Width σ decides how fat the packet is.
- $k = 2\pi/\lambda$ sets ripple spacing; bigger k means more ripples per metre.
- A is a master volume knob for the whole field.

Workable Examples

1. **Surface ripple.** With $v = 0.30$ m s^{-1}, $\lambda = 0.15$ m, $\sigma = 0.20$ m, find the crest radius at $t = 2$ s.
2. **Slower packet.** Keep λ, σ but set $v = 0.10$ m s^{-1}. Where is the crest at $t = 2$ s?

Python/Colab Verification

Listing 6: Circular packet

```python
import numpy as np, matplotlib.pyplot as plt

x = y = np.linspace(-1, 1, 400)
X, Y = np.meshgrid(x, y)
r = np.sqrt(X**2 + Y**2)

v, lam, sigma, t = 0.30, 0.15, 0.20, 2.0
k = 2*np.pi / lam
U = np.exp(-(r - v*t)**2 / (2*sigma**2)) * np.cos(k * (r - v*t))

plt.imshow(U, cmap='RdBu', extent=[-1, 1, -1, 1])
plt.colorbar()
plt.title('Circular Ripple Packet')
plt.show()
```

7 Straight-Edge Impulse & Convolution of Many Point Sources

What We Are About to Do

Instead of a single splash, we strike an entire straight board on the water. We model the board by adding *infinitely many* little circular packets—one for each point along the edge—and summing them up with an integral.

The One New Formula

$$U_{\text{line}}(x, y, t) = \int_{-L/2}^{L/2} U\big(r', t\big)\, \mathrm{d}x', \qquad r' = \sqrt{(x - x')^2 + y^2} \tag{7}$$

How it Works

- Each x' along the board launches a packet $U(r', t)$.
- The distance r' from that source to the observation point (x, y) sets the packet's delay.
- The integral $\int \mathrm{d}x'$ adds all these contributions, building the full wavefront.
- Board length L controls how wide the summed pulse is: long boards make flatter, broader fronts.

Workable Examples

1. **Long board.** $L = 1.0$ m, $v = 0.30$ m s^{-1}, $\sigma = 0.20$ m, $t = 1$ s — show the nearly flat crest at $y = 0.5$ m.
2. **Short paddle.** $L = 0.20$ m — observe the much stronger curvature.

Python/Colab Verification

Listing 7: Line-source superposition

```python
import numpy as np, matplotlib.pyplot as plt

v, lam, sigma = 0.30, 0.15, 0.20
k = 2*np.pi / lam
L, t, y_obs = 1.0, 1.0, 0.5

x_obs = np.linspace(-2, 2, 400)

def point_packet(r):
    return np.exp(-(r - v*t)**2/(2*sigma**2)) * np.cos(k*(r - v*t))

U_line = []
for x in x_obs:
    xs = np.linspace(-L/2, L/2, 400)
    r = np.sqrt((x - xs)**2 + y_obs**2)
    U_line.append(np.trapz(point_packet(r), xs))

plt.plot(x_obs, U_line)
plt.title('Wavefront from Straight Edge')
plt.xlabel('x (m)')
plt.show()
```

8 Obstacle & Re-Emission – Diffraction at a Slit

What We Are About to Do

We shine our wave through a narrow opening and ask how bright the far-field pattern is at each angle. Mathematically, that means taking the Fourier transform of a rectangular aperture—classical single-slit diffraction.

The One New Formula

$$I(\theta) = I_0 \left[\tfrac{\sin\beta}{\beta}\right]^2, \qquad \beta = \frac{\pi a}{\lambda}\sin\theta \tag{8}$$

How it Works

- a is slit width; narrower a spreads the pattern wider.
- θ measures the observation angle from the central axis.
- β is a phase term: each path through the slit gains a phase proportional to $\sin\theta$.
- The squared $\sin\beta/\beta$ factor makes a bright central maximum and evenly spaced dark minima where $\beta = n\pi$.
- I_0 simply scales the whole intensity curve.

Workable Examples

1. **Red laser.** $\lambda = 650\,\text{nm}$, $a = 0.10\,\text{mm}$. First dark at $\sin\theta \approx \lambda/a \approx 6.5 \times 10^{-3}$. Calculate θ.
2. **Narrower slit.** $a = 0.05\,\text{mm}$ — the first-dark angle changes to what θ?

Python/Colab Verification

Listing 8: Single-slit diffraction

```python
import numpy as np, matplotlib.pyplot as plt

lam, a = 532e-9, 0.10e-3
theta = np.linspace(-5e-3, 5e-3, 2_000)
beta = np.pi * a * np.sin(theta) / lam
I = (np.sinc(beta / np.pi))**2

plt.plot(np.degrees(theta), I)
plt.xlabel(r'$\theta$ (deg)')
plt.ylabel('Relative Intensity')
plt.title('Single-Slit Fraunhofer Diffraction')
plt.show()
```