TRƯỜNG ĐẠI HỌC XÂY DỰNG HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN

LẬP TRÌNH ÂM THANH

CHƯƠNG 5

Lập trình cơ bản phần 1

Chương 5 · 5 tiết · Mã học phần 531833 · 75 tiết · Học kỳ 5

Hà Nội – 2026

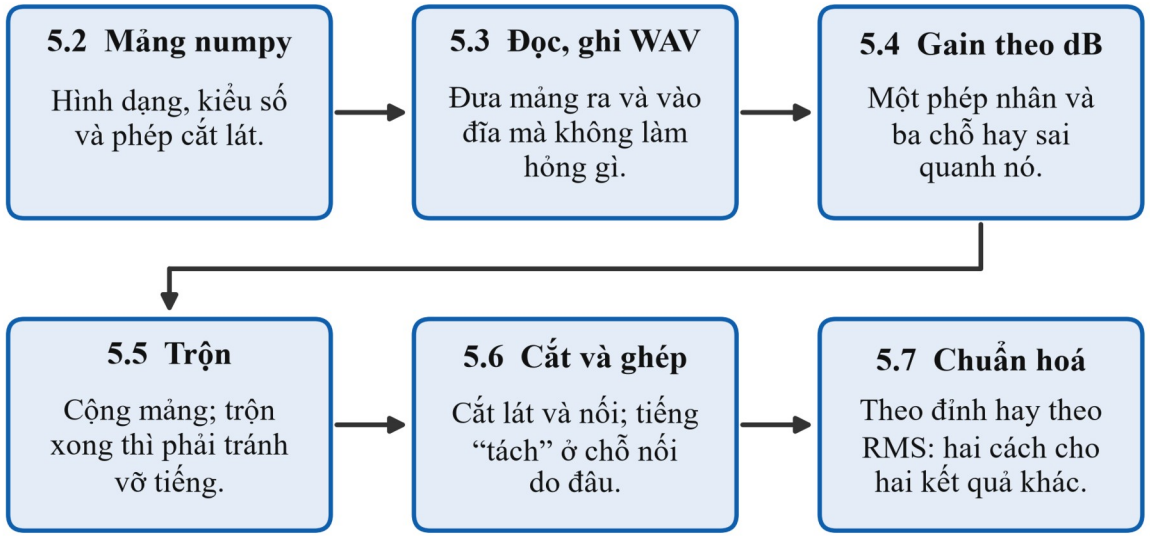
Sau chương này người học phải làm được

- Đọc hình dạng và kiểu số của mảng âm thanh; phân biệt mono với stereo; giải thích vì sao sr phải giữ riêng.
- Cắt lát mảng theo thời gian; biết khi nào phải gọi .copy().
- Viết mã theo lối vector hoá, không dùng vòng lặp trên từng mẫu.
- Đọc, ghi tệp WAV đúng quy ước; chọn kiểu số phù hợp; chặn ba lỗi đọc ghi hay gặp.
- Chỉnh gain theo decibel; đếm mẫu bị cắt đỉnh; giải thích vì sao cắt đỉnh nghe rè.
- Trộn nhiều nguồn có kiểm mức; tính headroom cần chừa theo số nguồn.
- Cắt, ghép nhiều đoạn; nhận ra nguyên nhân tiếng "tách" ở chỗ nối.
- Chuẩn hoá theo đỉnh và theo RMS (Root Mean Square – giá trị hiệu dụng); chọn cách phù hợp với từng mục đích.

MỤC 5.1

Vị trí chương học trong học phần

- Chương đầu tiên của mô-đun M4 "Lập trình cơ bản".
- Mỗi mục kết thúc bằng một đoạn mã chạy được, không phải một công thức.
- Không có khái niệm toán mới; cái mới là cách viết chúng thành mã.



Từ chương này, mỗi mục kết thúc bằng một đoạn mã chạy được, không phải bằng một công thức.

Sáu thao tác trên đủ làm phần lớn công việc biên tập âm thanh hằng ngày; không có khái niệm toán mới nào.

Chương này đòi hỏi phải gõ

Bốn chương đầu hỏi "chuyện gì đang xảy ra"; chương này hỏi "gõ cái gì vào máy". Đọc mà không gõ thì không học được mục nào của chương.

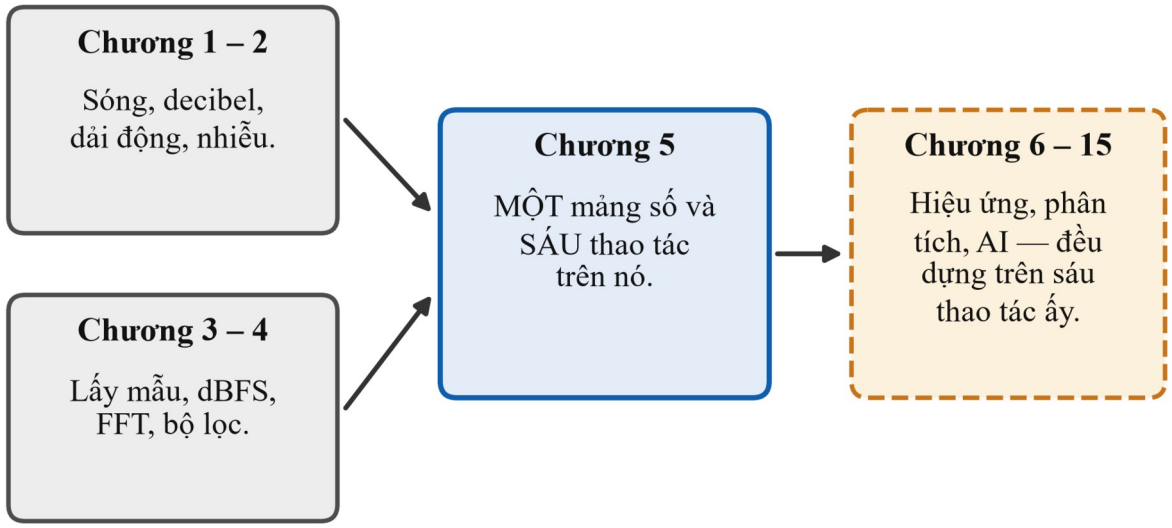
Sáu thao tác của chương 5 và việc chúng làm

Thao tác	Việc thực tế nó làm	Mục
Đọc và ghi tệp	Đưa âm thanh vào chương trình và trả ra đĩa	5.3
Chỉnh gain	To lên hay nhỏ đi theo decibel	5.4
Trộn	Ghép nhạc nền với lời thoại, chồng nhiều lớp	5.5
Cắt	Lấy một đoạn, bỏ phần thừa ở đầu và cuối	5.6
Ghép	Nối nhiều đoạn theo thứ tự mới	5.6
Chuẩn hoá	Đưa mức về đích thống nhất trước khi giao bài	5.7

Cả sáu thao tác đều gói gọn trong một, hai dòng numpy.

Bốn chương lý thuyết đọng lại thành một mảng số

- Ghép sáu thao tác là dựng được một bản thu phỏng vấn có nhạc nền.
- Đó là công việc của người dựng âm thanh cơ bản.
- Từ chương này, người học tự viết chương trình làm việc ấy hàng loạt.



Điều khác biệt của chương này

Bốn chương trước hỏi "chuyện gì đang xảy ra".
Chương này hỏi "gõ cái gì vào máy".

Nếu một mục không kết thúc bằng mã chạy được thì mục ấy chưa xong.

Hình 5.2 — Bốn chương lý thuyết đọng lại thành một mảng số và sáu thao tác trên nó

Ba thói quen học

- Gõ lại, đừng sao chép: mã đủ ngắn để gõ trong một, hai phút; gõ tay mới lộ chỗ chưa hiểu.
- Chạy trên tệp của chính người học: hai tệp WAV dặn ở cuối chương 4 dùng cho cả chương.
- Sai thì soi, đừng đoán: lỗi thường không báo khi chạy; in shape, in đỉnh, vẽ phổ đồ.

MỤC 5.2

Mảng numpy cho âm thanh

- Âm thanh là một mảng ndarray, cộng một số tần số lấy mẫu nằm riêng.
- Cần nắm: hình dạng mảng, cắt lát, vector hoá, kiểu số.
- Cuối mục có hàm mô tả mảng để dùng làm thói quen đầu tiên.

Số mẫu và thời lượng

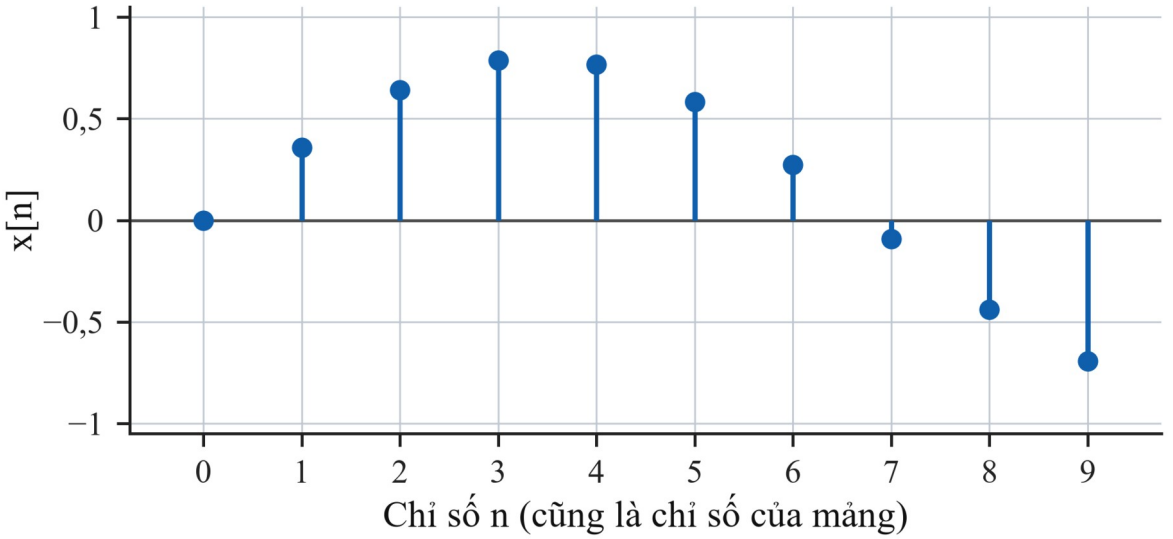
$$x:\ x[0], x[1], \ldots, x[N-1]; \qquad N = \lceil t \cdot f_s \rceil$$

(5.1)

- N là số mẫu, t là thời lượng (giây), fs là tần số lấy mẫu.
- Dấu ngoặc đặc biệt nghĩa là làm tròn về số nguyên gần nhất.
- Lệch một mẫu là lệch cả phép cắt về sau.

Dãy mẫu và mảng trong bộ nhớ là một

- Dãy mẫu vẽ ra và mảng numpy là cùng một thứ.
- Tần số lấy mẫu nằm ngoài mảng, trong một biến riêng.
- Để lạc sr thì mảng trở nên vô nghĩa.



(a) Dãy mẫu vẽ ra

0,00	0,36	0,64	0,79	0,77	0,58	0,27	−0,09	−0,44	−0,69
0	1	2	3	4	5	6	7	8	9

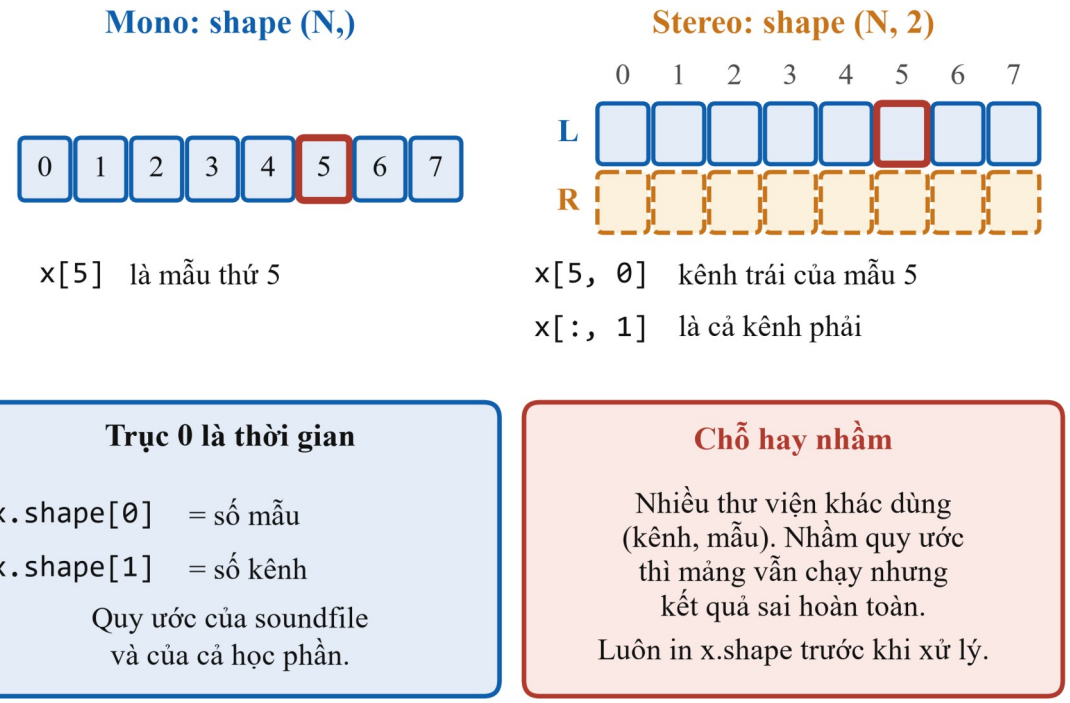
`x.shape = (10,)` `x.dtype = float64`

(b) Cùng một thứ: mảng numpy trong bộ nhớ

Hình 5.3 — Dãy mẫu vẽ ra và mảng numpy trong bộ nhớ là cùng một thứ

Hình dạng mảng mono và stereo

- Mono cho mảng một chiều, shape (N,).
- Stereo cho mảng hai chiều: hàng là mẫu, cột là kênh, shape (N, 2).
- Trục thời gian ở trục 0 theo soundfile; librosa mặc định dùng ngược lại.
- Nhầm quy ước là lỗi âm thầm: vẫn chạy nhưng kết quả vô nghĩa.



Hình 5.4 — Mảng mono có shape (N,), mảng stereo có shape (N, 2)

Quy ước hình dạng mảng

$$\text{mono}: (N,) \qquad \text{stereo}: (N, 2), \qquad x_{\text{mono}}[n] = \frac{x_L[n] + x_R[n]}{2} \tag{5.2}$$

- Hàng là mẫu theo thời gian, cột là kênh.
- Đây là quy ước của soundfile và của cả học phần.
- Thư viện khác có thể đặt ngược; luôn kiểm shape khi nhận mảng từ nơi khác.

Các phép hay dùng trên mảng nhiều kênh

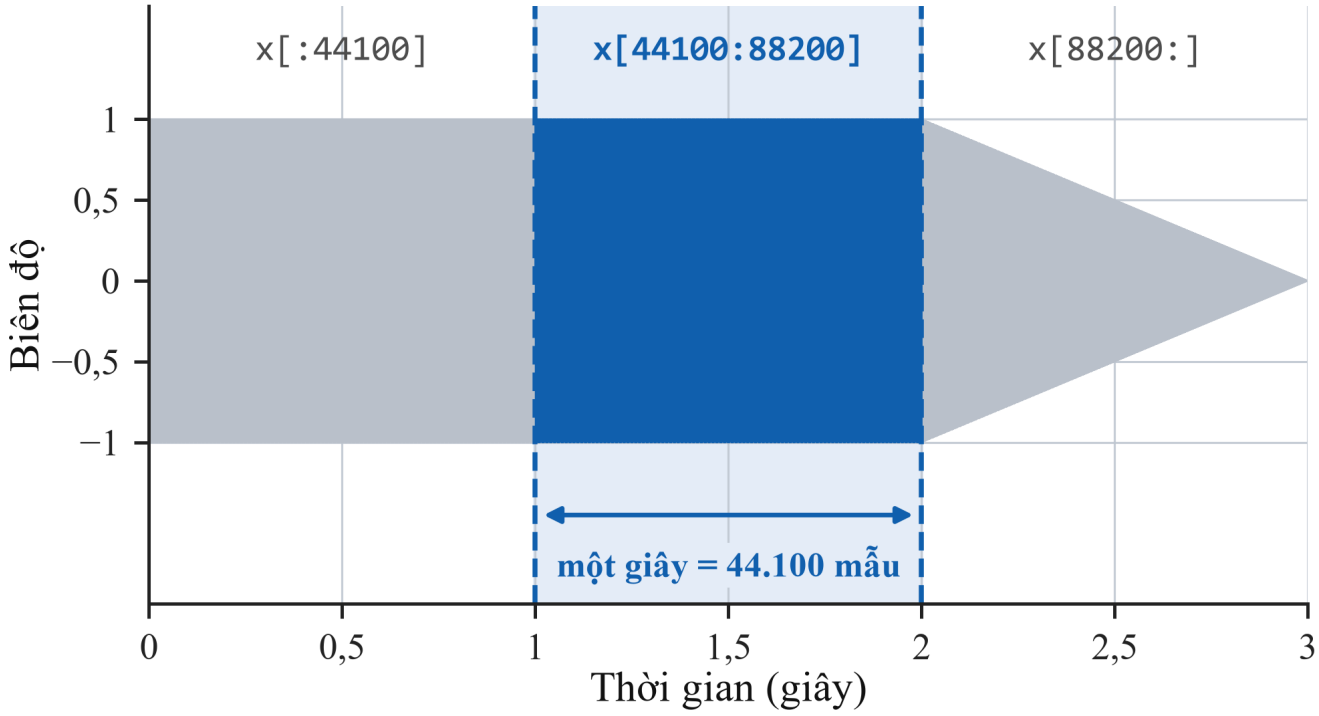
Việc cần làm	Cách viết	Chú ý
Đếm số kênh	x.shape[1] nếu x.ndim == 2, ngược lại là 1	Mono không có shape[1]
Lấy kênh trái	x[:, 0]	Cho ra mảng một chiều
Gộp về mono	x.mean(axis=1)	Lấy trung bình, không lấy tổng
Nhân đôi mono thành stereo	np.column_stack([x, x])	Hai kênh giống hệt nhau
Số mẫu	len(x) hoặc x.shape[0]	Là số mẫu mỗi kênh

Gộp mono phải lấy trung bình

Lấy tổng hai kênh gần giống nhau thì biên độ tăng gần gấp đôi và vượt toàn thang. Lấy trung bình luôn an toàn. Mọi phép cộng tín hiệu đều có nguy cơ vượt thang.

Cắt lát bằng chỉ số mẫu, nghĩ bằng giây

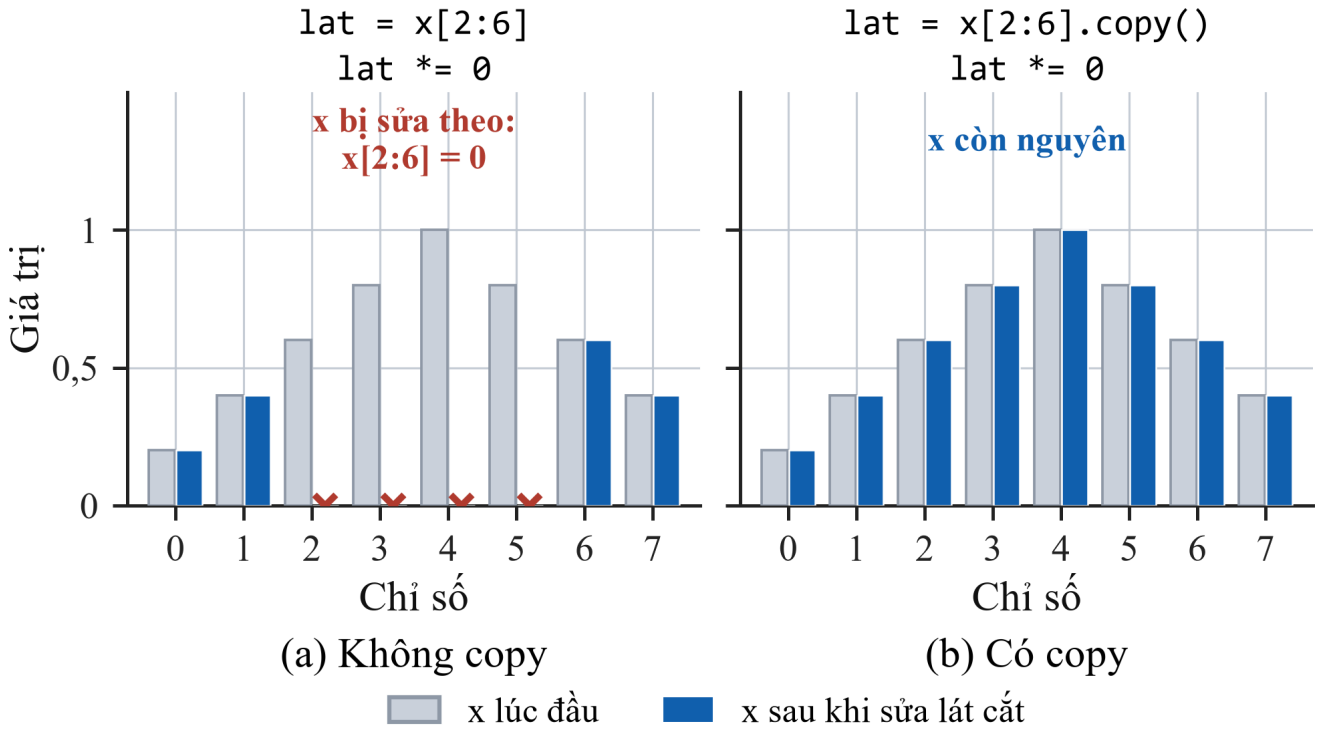
- Cắt lát chính là phép chọn một khoảng thời gian.
- Nghĩ bằng giây, viết bằng chỉ số, đổi qua lại bằng sr.
- x[1*sr : 2*sr] đọc ra ngay là từ giây thứ nhất tới giây thứ hai.



Hình 5.5 — Cắt lát bằng chỉ số mẫu, nhưng nghĩ bằng giây

Cắt lát không sao chép dữ liệu

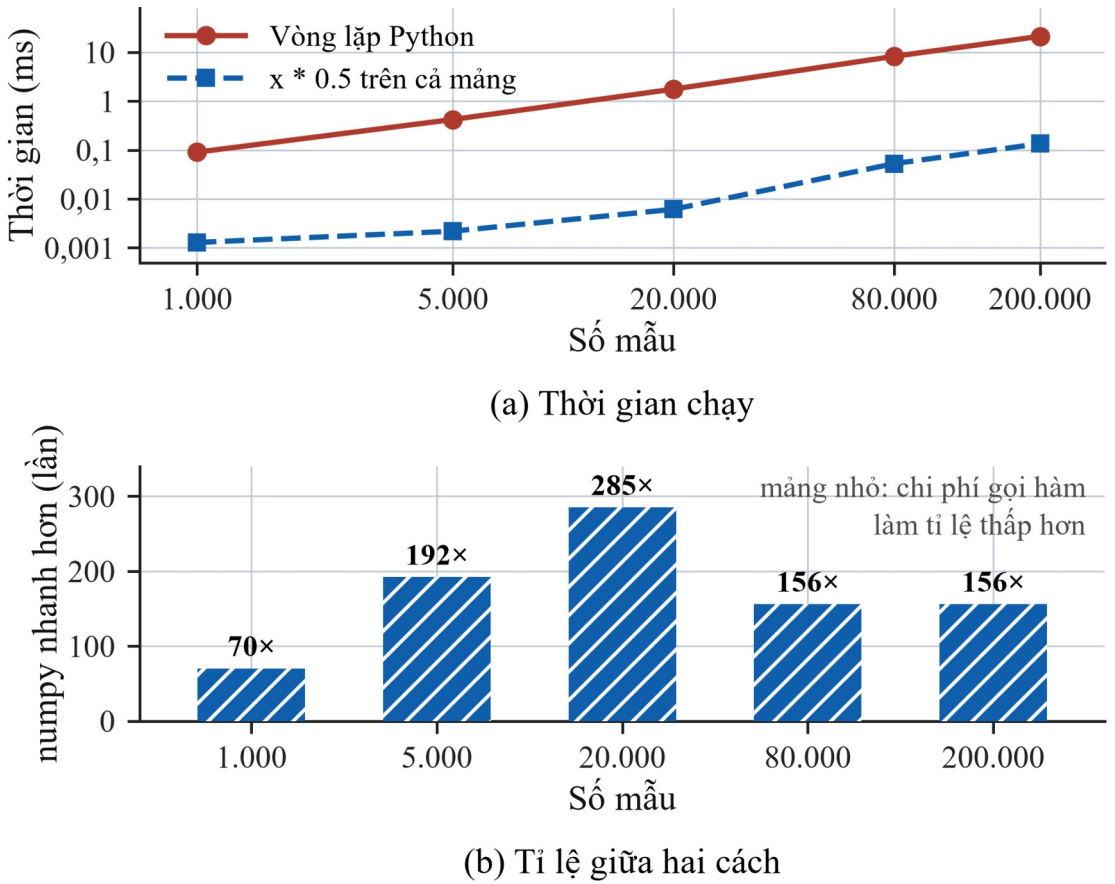
- Lát cắt là một "cửa sổ" nhìn vào cùng vùng nhớ với mảng gốc.
- Sửa lát cắt là sửa luôn mảng gốc; lỗi rất khó lần ra.
- Quy tắc: định sửa lát cắt thì gọi .copy(); chỉ đọc thì không cần.



Hình 5.6 — Cùng một phép sửa lát cắt: không copy thì mảng gốc bị sửa theo

Vòng lặp và phép trên cả mảng

- Cùng một phép nhân viết hai cách: vòng lặp Python và phép trên cả mảng.
- Chênh lệch tốc độ cỡ hàng trăm lần; con số cụ thể tuỳ máy.
- numpy đẩy phép tính xuống mã máy tối ưu sẵn.



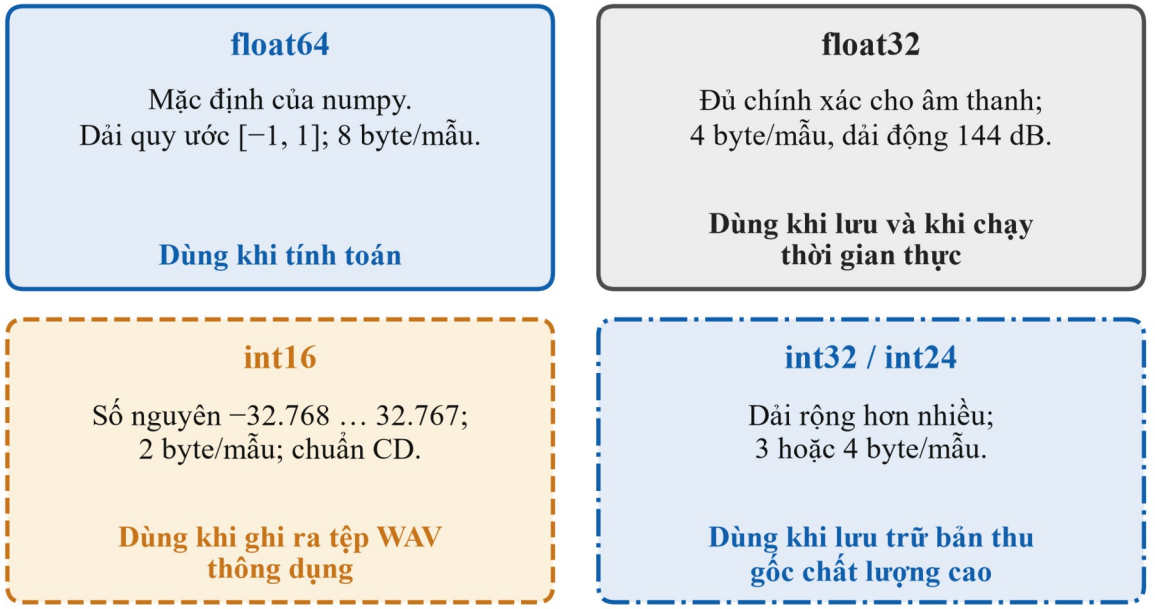
Hình 5.7 — Cùng một phép nhân viết hai cách: vòng lặp Python và phép trên cả mảng

Không viết vòng lặp trên từng mẫu

Bài hát bốn phút stereo có hơn hai mươi triệu mẫu. Ngoại lệ duy nhất là thuật toán cần trạng thái từ mẫu trước, như bộ lọc IIR (Infinite Impulse Response – đáp ứng xung vô hạn); ngay khi ấy vẫn nên gọi hàm thư viện.

Bốn kiểu số trong xử lý âm thanh

- Chương 3 đã nêu quy tắc vàng về số nguyên và số thực.
- Giờ viết nó thành thao tác cụ thể trên mảng.
- Mỗi kiểu có chỗ dùng riêng.



Quy tắc vàng (chương 3), nay viết thành mã: đọc ra float, xử lý bằng float, chỉ đổi sang int ở bước ghi cuối cùng.

Xử lý trực tiếp trên int là nguồn của gần hết các lỗi tràn số và méo tiếng mà người mới gặp.

Hình 5.8 — Bốn kiểu số gặp trong xử lý âm thanh và chỗ dùng của mỗi kiểu

Bốn kiểu số và đặc tính

Kiểu	Byte mỗi mẫu	Dải giá trị	Dùng ở đâu
float64	8	quy ước [−1, 1]	Mặc định của numpy; dùng khi tính toán
float32	4	quy ước [−1, 1]	Đủ chính xác; dùng khi lưu và chạy thời gian thực
int16	2	−32.768 … 32.767	Chuẩn CD; định dạng WAV phổ biến nhất
int24 / int32	3 / 4	rộng hơn nhiều	Bản thu gốc chất lượng cao

Quy tắc vàng viết thành ba bước

- Đọc ra float: soundfile mặc định trả float64 trong [−1, 1], dù tệp gốc là kiểu gì.
- Xử lý bằng float: không tràn số, sai số làm tròn không nghe được.
- Chỉ đổi sang int ở bước ghi cuối cùng, và để thư viện tự làm.
- Không tự nhân 32.767 rồi ép kiểu bằng tay: đó là nguồn của gần hết lỗi méo tiếng.

Xem một mảng âm thanh có gì trước khi xử lý

```
import numpy as np
import soundfile as sf

def mo_ta(x, sr, ten="mang"):
    """In ba con so phai biet ve moi mang am thanh."""
    kenh = 1 if x.ndim == 1 else x.shape[1]
    print(f"{ten}: shape={x.shape}  dtype={x.dtype}")
    print(f"  {kenh} kenh, {sr} Hz, {len(x) / sr:.2f} giay")
    print(f"  dinh  = {np.abs(x).max():.4f}"
          f"  ({20 * np.log10(max(np.abs(x).max(), 1e-9)):.1f} dBFS)")
    print(f"  RMS   = {np.sqrt((x ** 2).mean()):.4f}")

x, sr = sf.read("giong_doc.wav")
mo_ta(x, sr, "giong doc")
```

Thói quen đầu tiên của mọi chương trình âm thanh: lộ ngay shape lạ, dtype lạ hay đỉnh bằng 0.

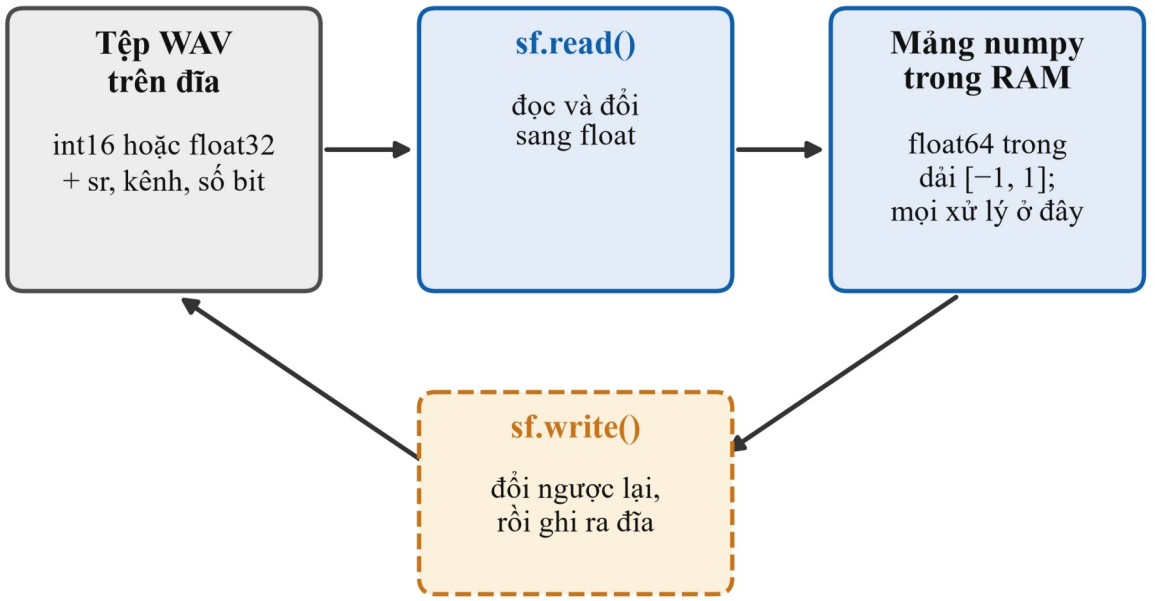
MỤC 5.3

Đọc và ghi tệp WAV

- Đường đi: tệp trên đĩa, đọc lên thành float, xử lý, đổi ngược rồi ghi.
- Đọc: ba dòng và bốn điều cần kiểm.
- Ghi: một dòng và ba quyết định; kèm ba lỗi hay gặp.

Từ đĩa tới RAM và ngược lại

- Thư viện đọc tệp và đổi sang float; mọi xử lý diễn ra trên float.
- Ở bước cuối, thư viện đổi ngược và ghi ra đĩa.
- Mảng không tự nhớ tần số lấy mẫu; chương trình phải tự giữ và truyền đi.
- Để lạc sr thì không có cách nào đoán lại.



Ba thứ đi kèm mảng, do chương trình tự giữ:

tần số lấy mẫu sr · số kênh (đọc từ shape) · kiểu số khi ghi ra

Mảng không tự nhớ tần số lấy mẫu; mất sr là mất tất cả.

Hình 5.9 — Từ tệp trên đĩa tới mảng trong RAM và ngược lại

Đọc một tệp và kiểm ngay bốn thứ

```
import numpy as np
import soundfile as sf

x, sr = sf.read("nguon.wav")     # luon tra ve float64 [-1, 1]

assert x.ndim in (1, 2),      "hinh dang la"
assert sr in (44100, 48000),  f"tan so la: {sr}"
assert np.abs(x).max() <= 1.0, "vuot toan thang ngay tu tep"
assert len(x) > 0,            "tep rong"

print(f"{len(x) / sr:.2f} giay, {sr} Hz,",
      f"{1 if x.ndim == 1 else x.shape[1]} kenh")
```

Mỗi dòng assert chặn một lỗi đã từng làm sai cả một mẻ xử lý khi chạy trên hàng trăm tệp của người khác.

Tham số của sf.read và khi nào cần đổi

Tham số	Mặc định	Khi nào cần đổi
dtype	'float64'	Đặt 'float32' khi tệp rất dài, tiết kiệm nửa bộ nhớ
always_2d	False	Đặt True để mọi tệp ra mảng hai chiều, kể cả mono
start, stop	cả tệp	Đọc một đoạn giữa tệp mà không nạp cả tệp
frames	cả tệp	Đọc theo từng khối; cách xử lý tệp rất dài

Ghi ra tệp, có kiểm trước

```
dinh = np.abs(y).max()
if dinh > 1.0:
    raise ValueError(f"vuot toan thang: {dinh:.3f} — chua ghi")

sf.write("ket_qua.wav", y, sr, subtype="PCM_16")
```

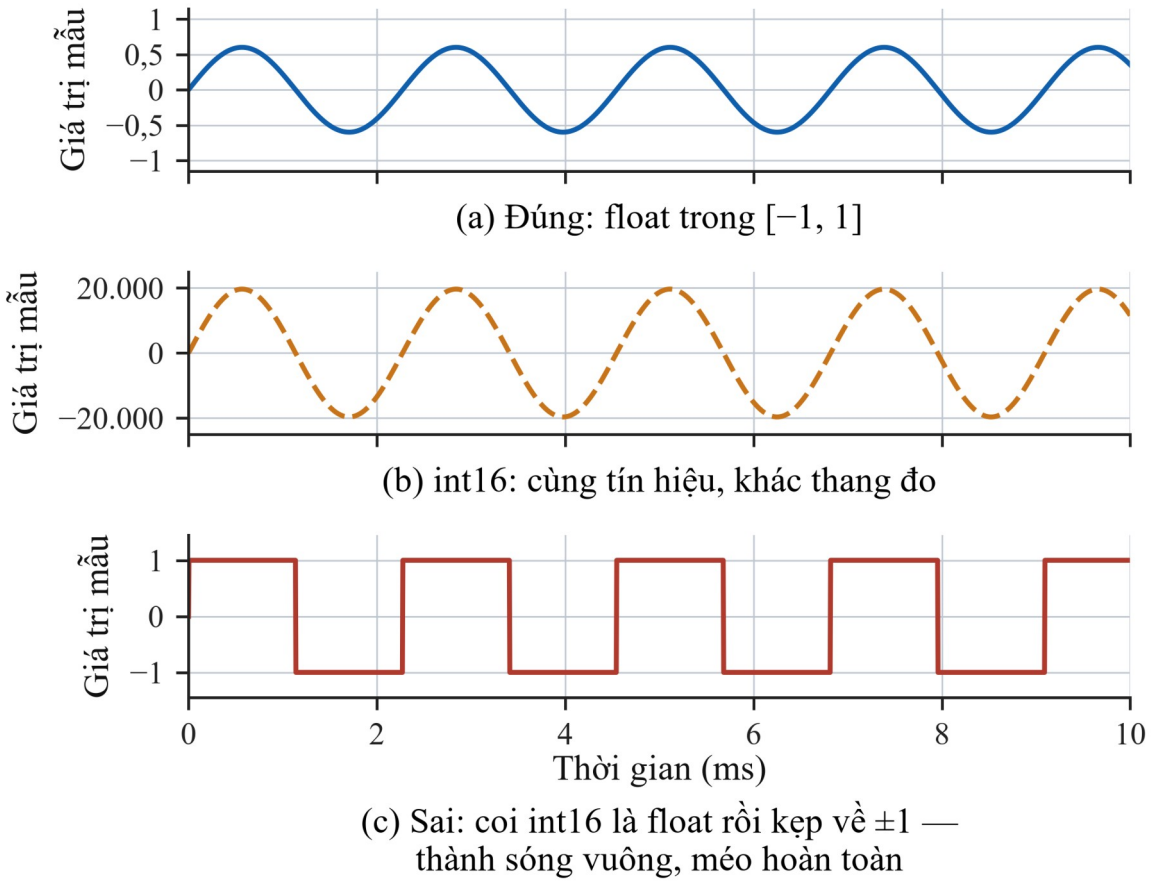
Một dòng gọi chứa ba quyết định: tần số lấy mẫu, kiểu số, dải giá trị.

Một dòng gọi, ba quyết định

- Tần số lấy mẫu: phải là con số đã đọc từ tệp gốc; ghi sai thì tiếng nhanh hoặc chậm, cao độ lệch.
- Kiểu số: PCM_16 để giao đi; PCM_24 cho bản gốc cần xử lý tiếp; float cho tệp trung gian.
- Dải giá trị: mảng phải nằm trong [−1, 1]; vượt thì thư viện lặng lẽ kẹp và tiếng bị rè.

Xử lý mảng int16 như thể là float

- Nhân tay với 32.767 rồi ép kiểu: cắt cụt thay vì làm tròn.
- Mẫu vượt 1 thì tràn số, giá trị đảo dấu; tiếng ra thành nhiễu.
- Cách ép tay cũng không áp dither.
- Truyền subtype cho sf.write để thư viện lo cả ba việc.



Hình 5.10 — Điều xảy ra khi lấy mảng int16 mà xử lý như thể nó là float

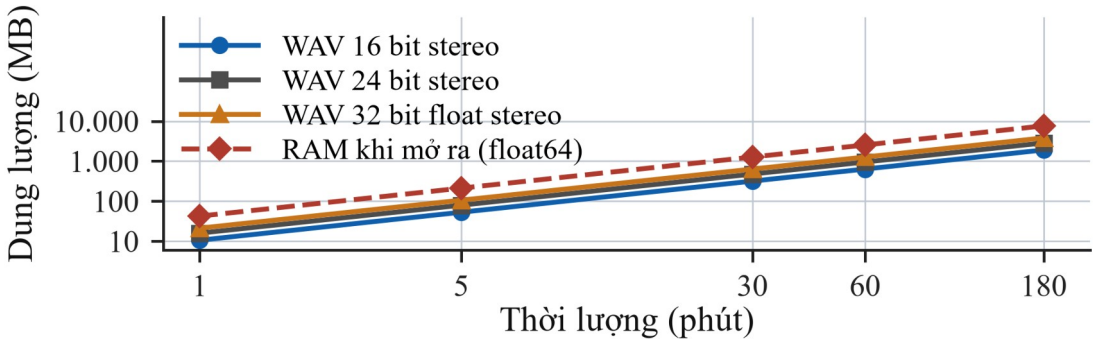
Dung lượng tệp và bộ nhớ khi mở ra

$$S = f_s \cdot \frac{b}{8} \cdot N_{\text{kênh}} \cdot t\ ; \qquad R_{\text{RAM}} = f_s \cdot 8 \cdot N_{\text{kênh}} \cdot t \quad [\text{byte}]$$

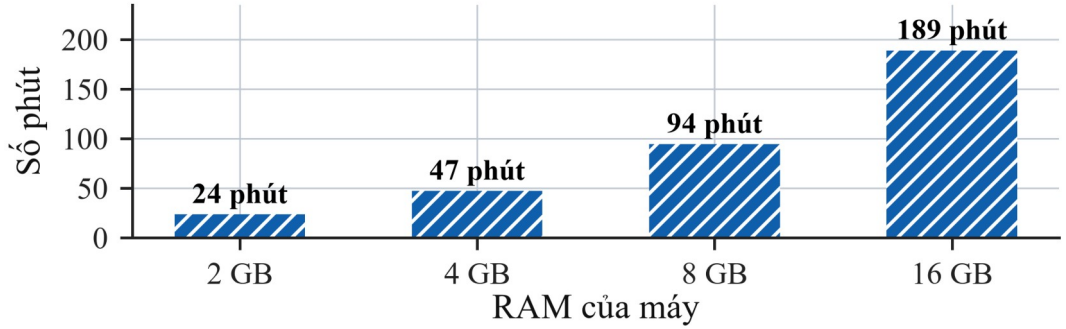
(5.3)

- Vế trái: dung lượng tệp trên đĩa, đã gặp ở chương 3.
- Vế phải: bộ nhớ của mảng khi mở ra; 8 là số byte của một giá trị float64.
- RAM gấp bốn lần tệp WAV 16 bit, luôn như vậy.
- Tệp dài: đọc đoạn (start, stop), đọc theo khối (sf.blocks), hoặc dtype float32.

Dung lượng tệp, bộ nhớ khi mở ra và giới hạn theo RAM



(a) Dung lượng trên đĩa và trong RAM

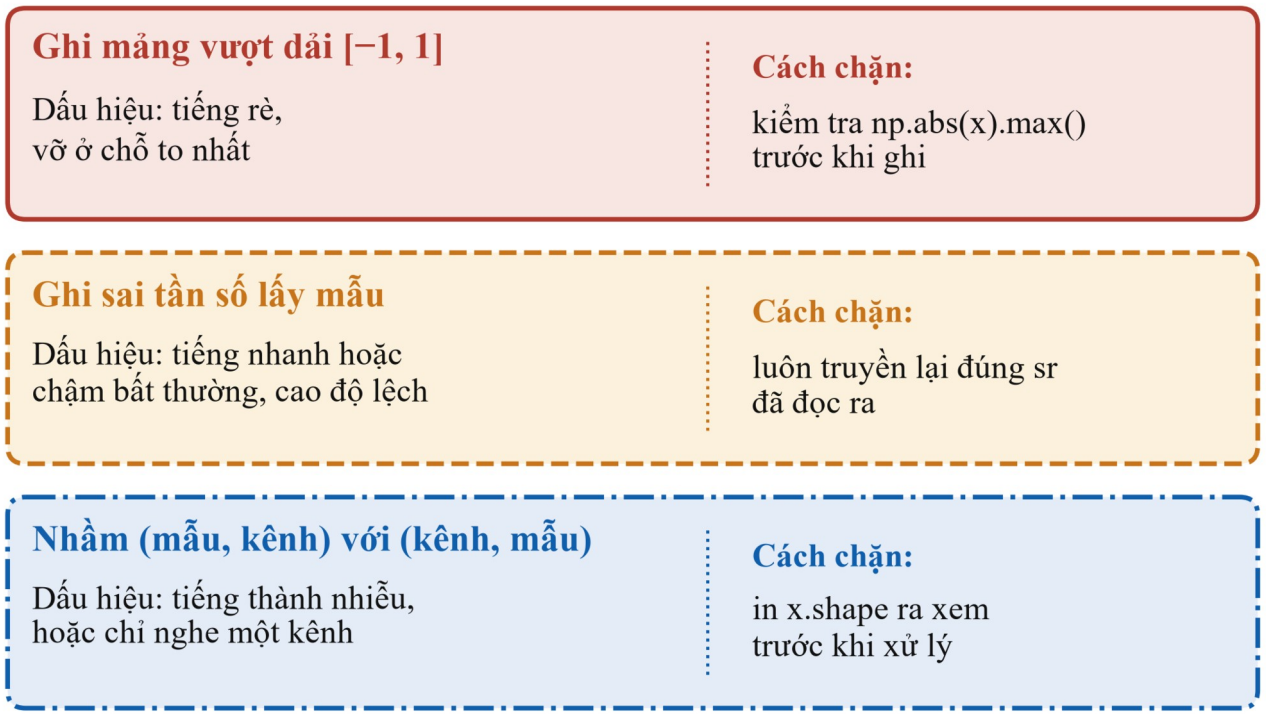


(b) Thời lượng tối đa mở cùng lúc (dùng một nửa RAM)

Hình 5.11 — Dung lượng tệp, bộ nhớ khi mở ra, và giới hạn thực tế theo RAM của máy

Dấu hiệu nhận ra và cách chặn

- Cả ba lỗi đều chạy xong mà không báo; chỉ âm thanh bị sai.
- Đây là đặc điểm của gần hết lỗi trong lập trình âm thanh.
- Phải nghe và nhìn phổ đồ sau mỗi bước, không chỉ tin chương trình chạy trót lọt.



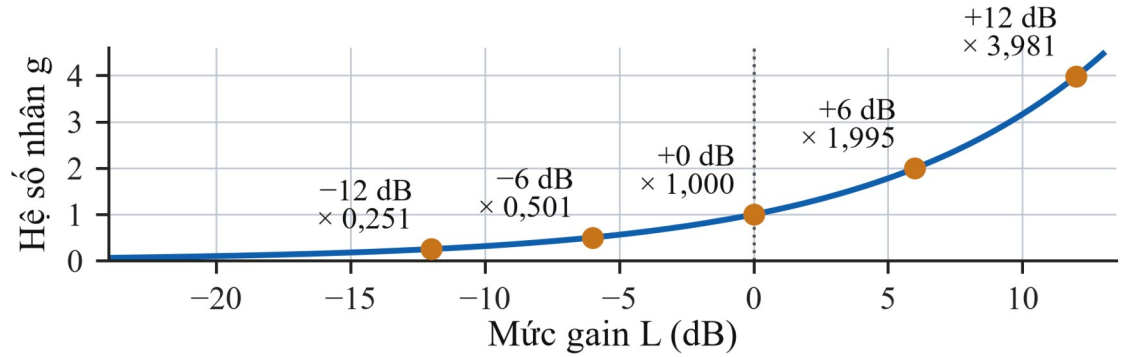
Cả ba lỗi đều không báo lỗi khi chạy; chỉ khi nghe hoặc nhìn phổ đồ mới phát hiện được.

Hình 5.12 — Ba lỗi đọc ghi, dấu hiệu nhận ra và cách chặn

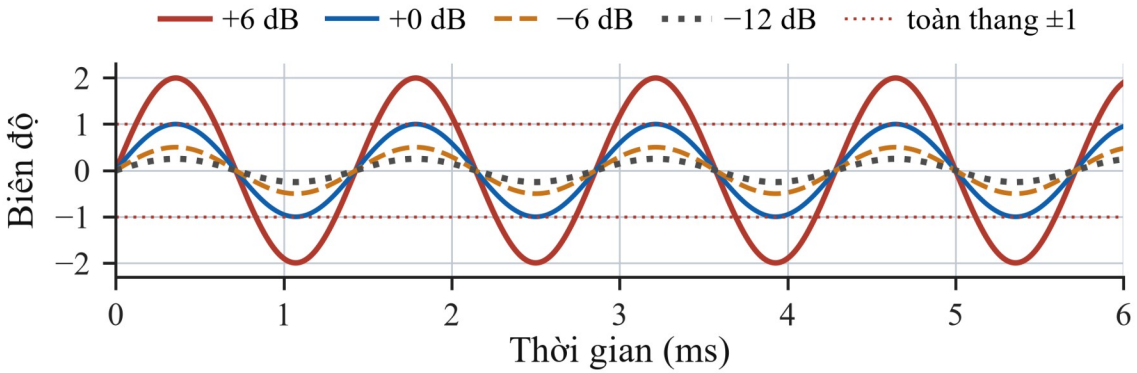
MỤC 5.4

Chỉnh gain theo decibel

- Chỉnh gain là nhân cả mảng với một hệ số.
- Cái khó là chọn con số và biết điều gì xảy ra khi vượt thang.
- Gồm: công thức dB, cắt đỉnh, thứ tự các phép, đỉnh giữa hai mẫu.



(a) Hệ số nhân theo mức gain: $g = 10^{L/20}$



(b) Cùng một tín hiệu ở bốn mức gain

Từ decibel ra hệ số nhân

$$y[n] = g \cdot x[n], \qquad g = 10^{L/20} \tag{5.4}$$

- L là mức gain tính bằng decibel; g là hệ số nhân.
- Đây là công thức decibel của chương 1, dùng theo chiều ngược.
- Hạ 6 dB thì nhân khoảng 0,501; hạ 20 dB thì nhân 0,100.

Biết đỉnh đích, tính gain cần chỉnh

$$L_{\text{cần}} = L_{\text{đích}} - 20\log_{10} x_{\text{đỉnh}}\ ; \qquad x_{\text{đỉnh}} = \max_{n} |x[n]|$$

(5.5)

- Ldich là mức đỉnh mong muốn (dBFS (decibel so với mức toàn thang, Full Scale)); xdinh là đỉnh hiện tại.
- Đây là phép tính bên trong mọi nút "normalize" của phần mềm biên tập.
- Mục 5.7 sẽ dùng lại công thức này.

Vài mức gain nên thuộc lòng

Mức dB	Hệ số nhân	Nghe ra sao
−20 dB	0,100	nhỏ hẳn, như lùi ra xa
−12 dB	0,251	nhỏ rõ rệt
−6 dB	0,501	còn một nửa biên độ
−3 dB	0,708	còn một nửa công suất; thay đổi nhẹ
+0 dB	1,000	không đổi
+3 dB	1,413	to lên một chút
+6 dB	1,995	gấp đôi biên độ
+10 dB	3,162	nghe như to gấp đôi

Gấp đôi biên độ là +6 dB, nhưng nghe như to gấp đôi cần khoảng +10 dB: tai không nghe theo biên độ.

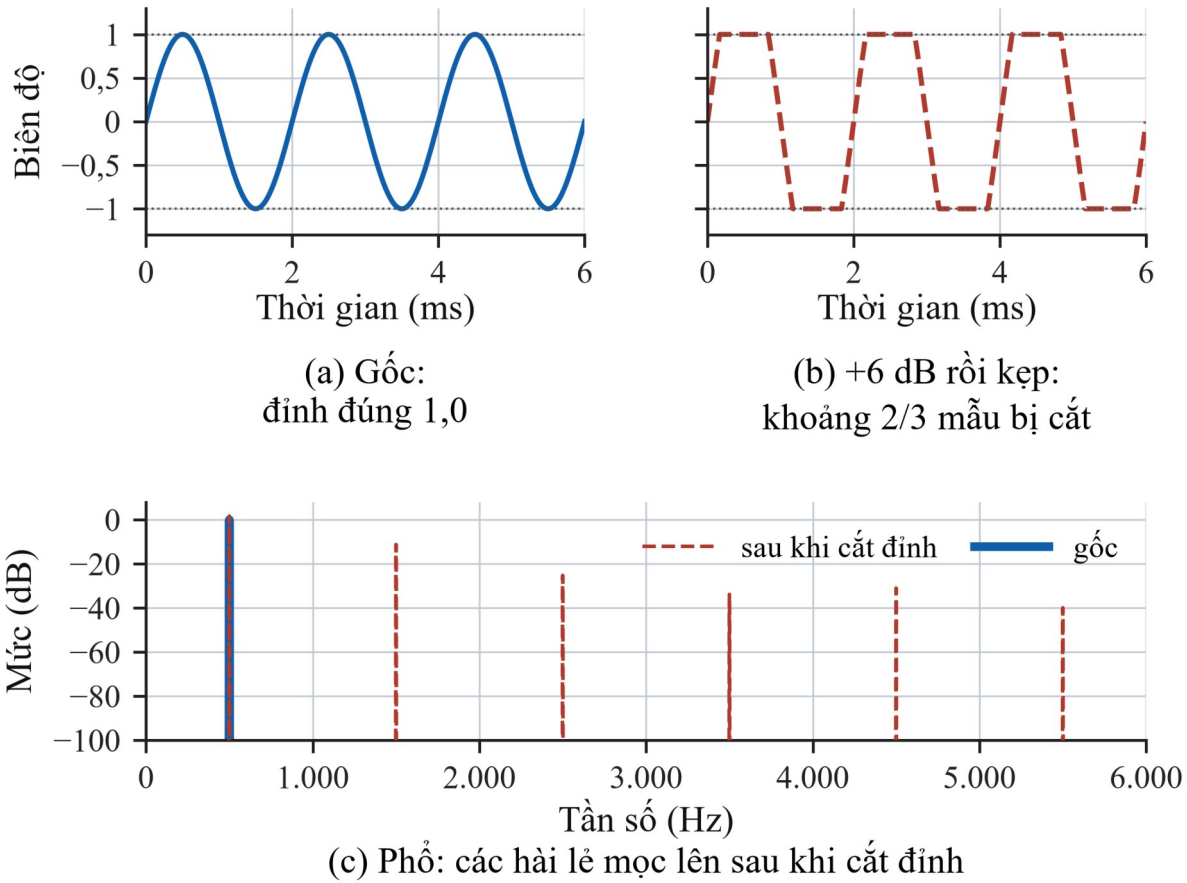
Hàm chỉnh gain dùng lại được

```
def gain_db(x, db):
    """Nhan tin hieu voi he so ung voi muc dB cho truoc."""
    return x * (10.0 ** (db / 20.0))

# vi du: ha nhac nen 12 dB de nhuong cho loi thoai
nhac_nen = gain_db(nhac_nen, -12.0)
```

Cắt đỉnh trên dạng sóng và trên phổ

- Trong mảng float, giá trị lớn hơn 1 không gây hại gì.
- Chỉ khi ghi tệp hoặc phát ra loa, giá trị mới bị kẹp về ±1.
- Kẹp biến sóng sin thành gần sóng vuông, sinh ra chuỗi hài lẻ.
- Các hài ấy không có trong tín hiệu gốc và không gỡ bỏ được.



Hình 5.14 — Cắt đỉnh nhìn trên dạng sóng và nhìn trên phổ

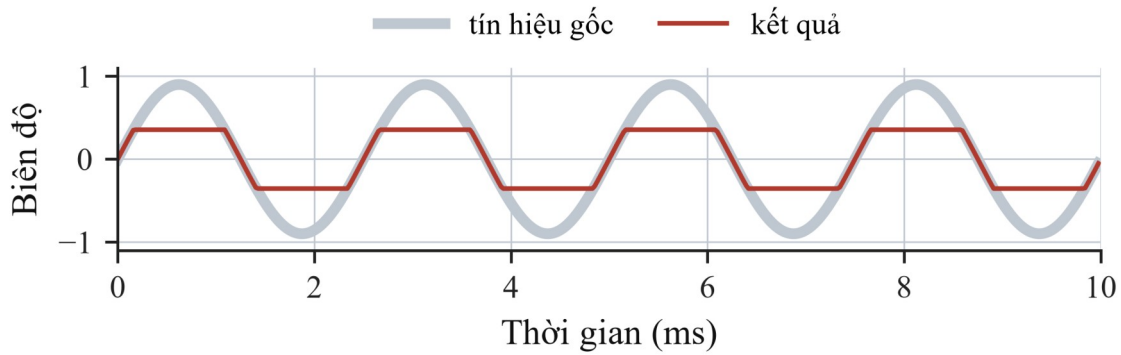
Đếm số mẫu bị cắt đỉnh trước khi ghi

```
def kiem_cat_dinh(x, ten="tin hieu"):
    """Bao cao so mau vuot toan thang. Tra ve True neu sach."""
    d = int(np.sum(np.abs(x) > 1.0))
    if d == 0:
        print(f"{ten}: sach, dinh {np.abs(x).max():.4f}")
        return True
    print(f"{ten}: {d} mau vuot ({100 * d / x.size:.3f} %),",
          f"dinh {np.abs(x).max():.3f}")
    return False
```

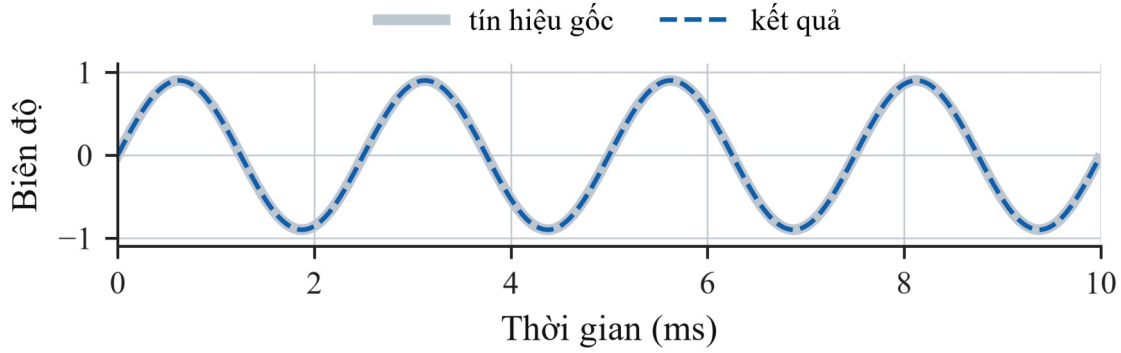
Gọi sau mọi phép có thể làm tín hiệu to lên: chỉnh gain, trộn, lọc.

Hai phép gain, hai thứ tự, hai kết quả

- Hạ 9 dB rồi tăng lại 9 dB có về như cũ không?
- Có, nếu giữa hai phép không có bước nào kẹp giá trị.
- Nếu có bước kẹp ở giữa thì phần bị kẹp mất vĩnh viễn.



(a) Tăng 9 dB → kẹp → hạ 9 dB (lệch tối đa 0,545)

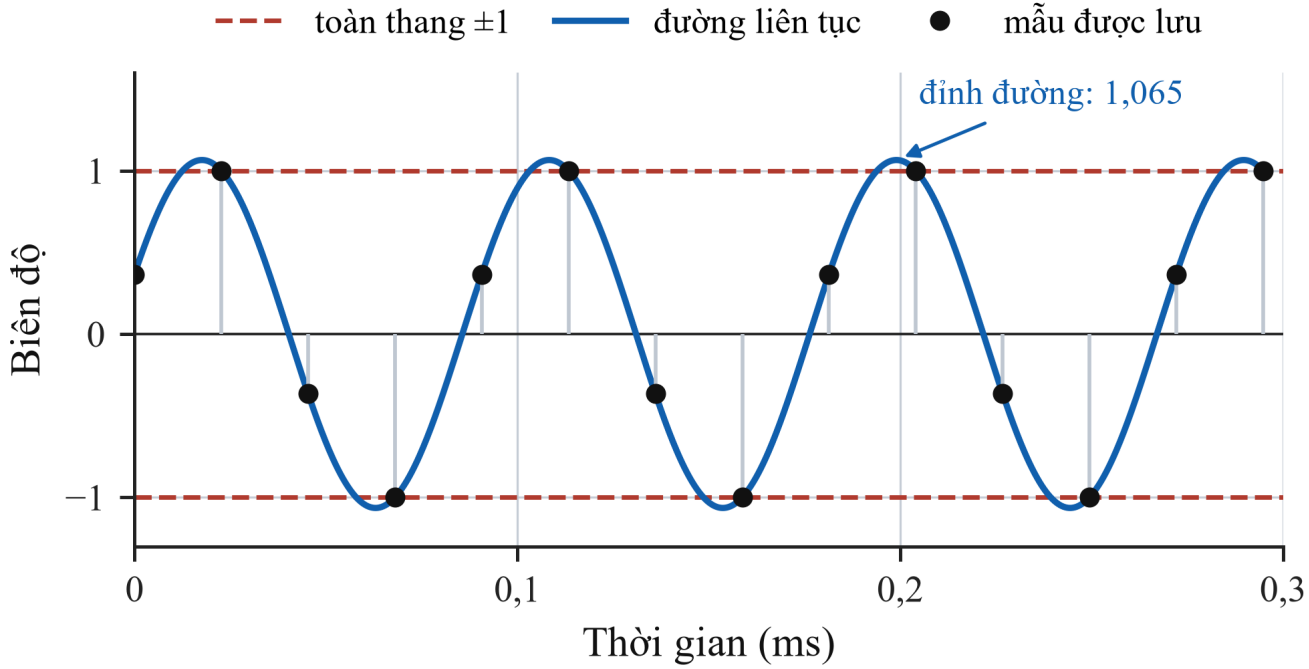


(b) Hạ 9 dB → tăng 9 dB (lệch tối đa $\approx 10^{-16}$)

Hình 5.15 — Cùng hai phép gain, hai thứ tự, hai kết quả hoàn toàn khác nhau

Chuẩn hoá về 0 dBFS vẫn bị méo

- Bộ chuyển đổi ngược phát ra đường liên tục đi qua các mẫu.
- Mọi mẫu đúng toàn thang, nhưng đường liên tục lên tới 1,065.
- Cao hơn toàn thang 0,54 dB; phần cứng sẽ cắt đúng chỗ ấy.
- Nên nhắm −1 dBFS; tín hiệu MP3, AAC có thể vượt vài dB nên chuẩn phát thanh dùng −2 dBFS hoặc thấp hơn (true peak).



Hình 5.16 — Mọi mẫu đều đúng toàn thang, nhưng đường liên tục thì vượt lên trên

MỤC 5.5

Trộn nhiều nguồn

- Trộn là phép cộng các mảng, và đó là chỗ sinh ra vấn đề.
- Cần biết hạ bao nhiêu trước khi trộn, và hai nguồn cùng mức cộng lại to lên bao nhiêu.
- Trộn hai đoạn khác độ dài: đệm số không hoặc cắt cụt.

Phép trộn và mức hạ trước khi trộn

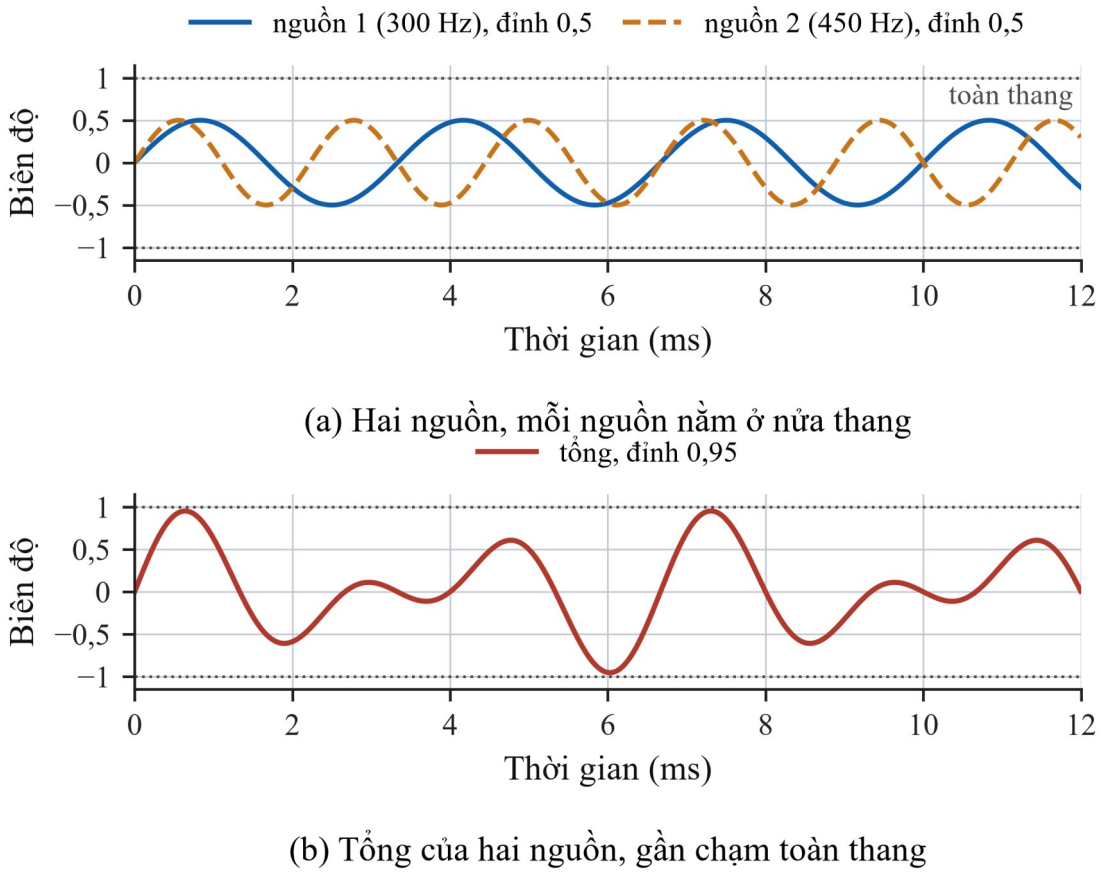
$$y[n] = \sum_{i=1}^{K} x_i[n]\,; \qquad L_{\text{hạ}} = L_{\text{đích}} - 20\log_{10}\left(\sum_{i=1}^{K} p_i\right)$$

(5.6)

- Vế trái là phép trộn: cộng các nguồn.
- Vế phải: hạ mỗi nguồn để tổng không vượt thang.
- pi là đỉnh của nguồn thứ i; Ldich là mức đích của bản trộn.

Hai nguồn ở nửa thang, tổng gần chạm toàn thang

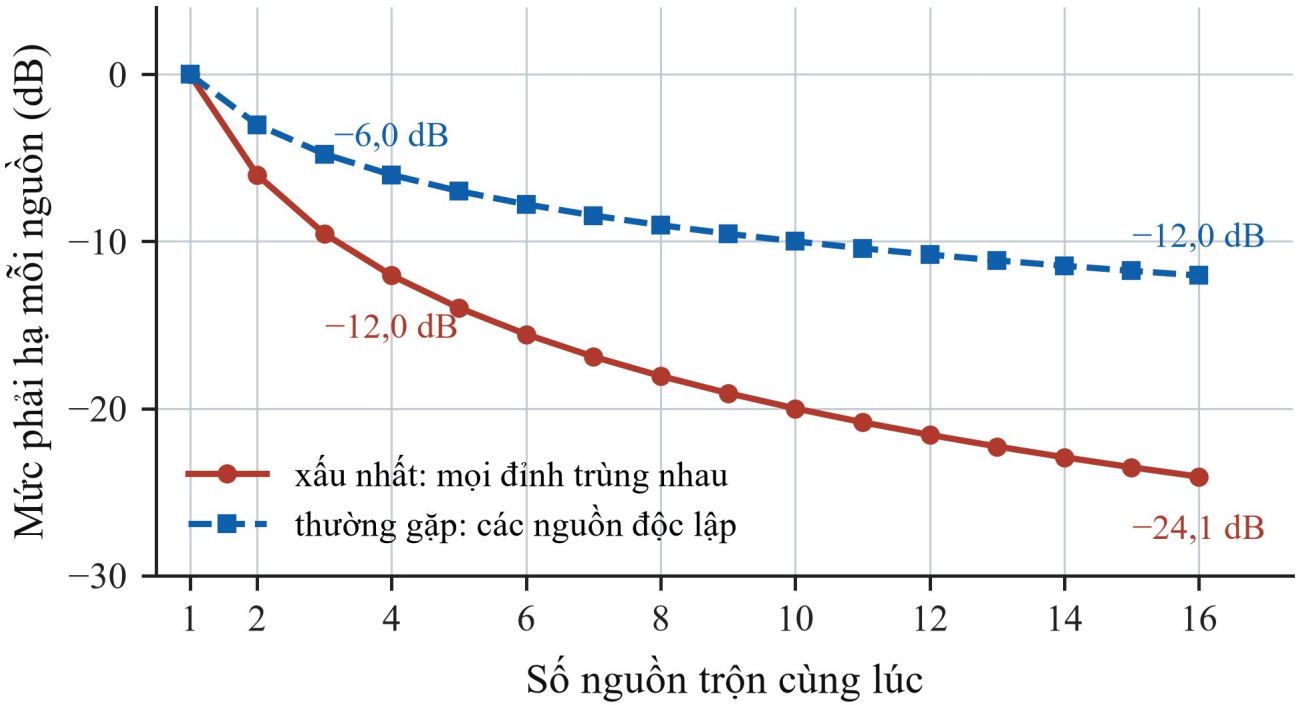
- Hai sóng gặp nhau trong không khí thì áp suất cộng vào nhau.
- Trộn không lấy trung bình mà cộng.
- Bốn nguồn cùng đỉnh 0,5 có thể cho đỉnh tới 2,0.
- Bảng trộn của phần mềm tự hạ mức; khi tự viết mã thì không ai hạ hộ.



Hình 5.17 — Hai nguồn đều nằm ở nửa thang, nhưng tổng thì gần chạm toàn thang

Mức hạ mỗi nguồn theo số nguồn

- Đồ thị cho hai giả định: cộng biên độ và cộng công suất.
- Số nguồn càng nhiều thì mức phải hạ càng lớn.
- Hai đường chênh nhau, và khoảng chênh ấy chính là headroom có thể đánh đổi.



Hình 5.18 — Mức phải hạ mỗi nguồn theo số nguồn, ở hai giả định khác nhau

Hai cách ước lượng và khi nào dùng

Cách ước lượng	Công thức	Khi nào dùng
Xấu nhất: cộng biên độ	hạ $20 \cdot \log_{10}(K)$ dB	Nguồn có thể trùng đỉnh: nhiều bản sao của một tiếng
Thường gặp: cộng công suất	hạ $10 \cdot \log_{10}(K)$ dB	Nguồn độc lập: lời thoại, nhạc nền, tiếng động môi trường

Với bốn nguồn: xấu nhất hạ −12,0 dB, thường gặp hạ −6,0 dB. Nên hạ theo ước lượng thường gặp, rồi đo đỉnh thật và chỉnh lại.

Trộn nhiều nguồn có kiểm mức

```
def tron_an_toan(nguon, dinh_dich_db=-1.0):
    """Tron nhieu mang, tu ha muc de tong khong vuot thang."""
    n = max(len(x) for x in nguon)
    tong = np.zeros(n)
    for x in nguon:
        tong[:len(x)] += x              # tu dem zero cho ngan hon

    dinh = np.abs(tong).max()
    if dinh <= 0:
        return tong
    can = dinh_dich_db - 20 * np.log10(dinh)
    print(f"dinh sau tron {20 * np.log10(dinh):.1f} dBFS",
          f"-> chinh {can:+.1f} dB")
    return tong * (10 ** (can / 20))
```

Trộn trước, đo và chỉnh sau; làm được vì đang ở trong float, nơi giá trị vượt 1 không gây hại.

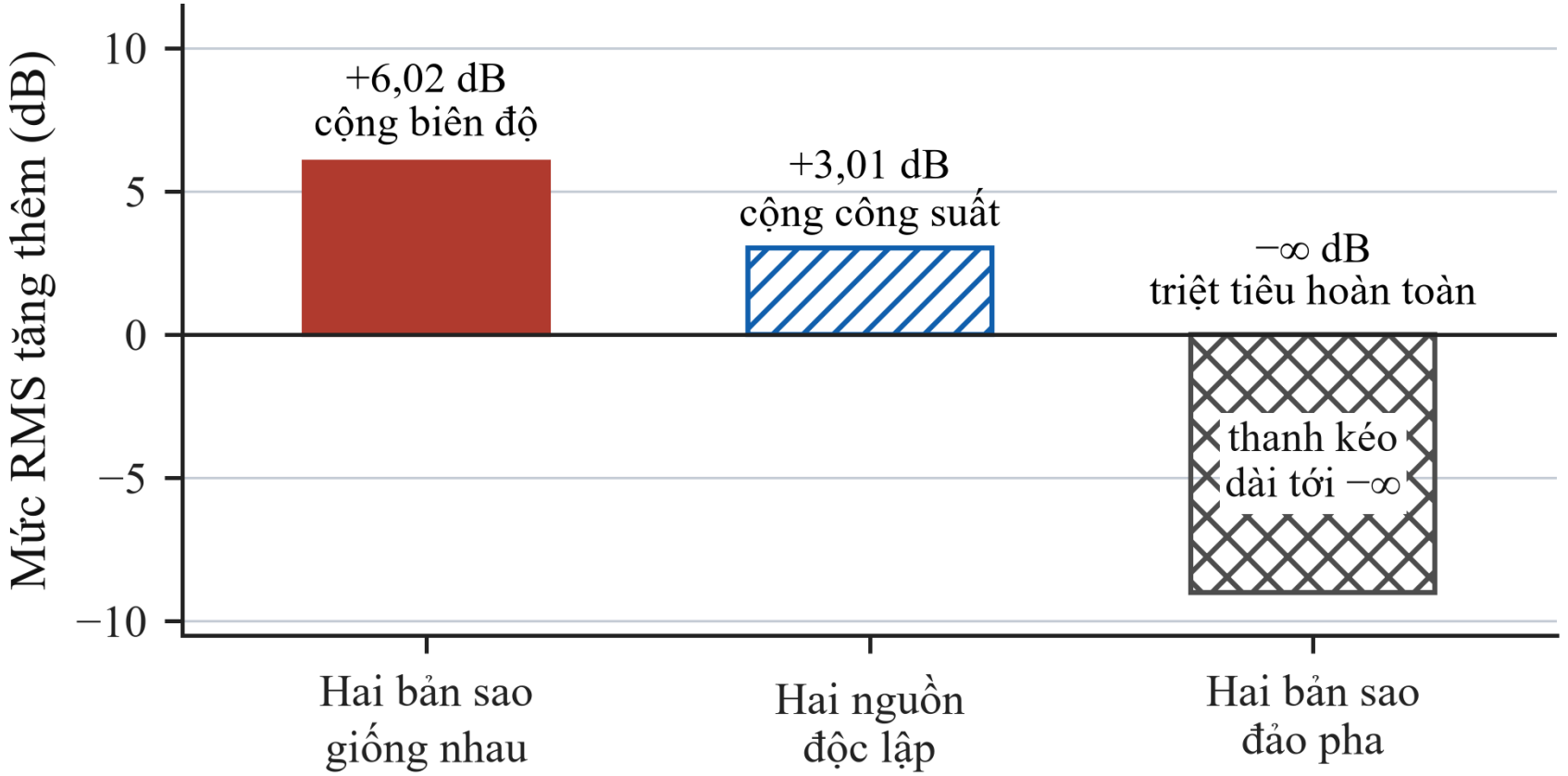
Hai nguồn cùng mức trộn lại to lên bao nhiêu

$$L_{\text{tươngquan}} = 20\log_{10}(a_1 + a_2)\,; \qquad L_{\text{độclập}} = 20\log_{10}\sqrt{a_1^2 + a_2^2}$$

(5.7)

- Không có đáp số duy nhất; tuỳ quan hệ giữa hai nguồn.
- Hai bản sao y nhau: biên độ cộng thẳng, +6,02 dB.
- Hai nguồn độc lập: công suất cộng, chỉ +3,01 dB.
- Hai bản sao đảo pha: triệt tiêu sạch, còn đúng số không.

Ba trường hợp trộn hai nguồn cùng mức



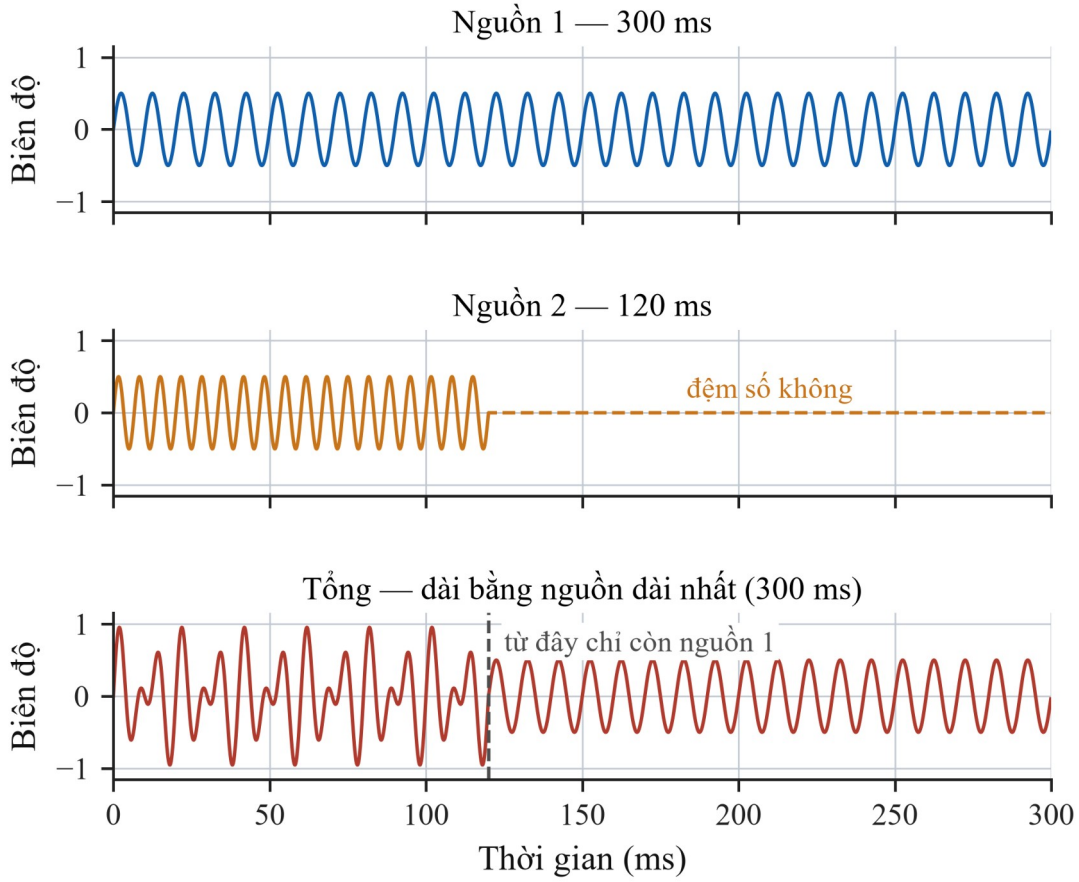
Hình 5.19 — Ba trường hợp trộn hai nguồn cùng mức, ba kết quả hoàn toàn khác nhau

Đảo pha giữa hai kênh không phải chuyện nhân tạo

Hai kênh stereo đảo pha nghe hay trên tai nghe, nhưng gộp về mono thì triệt tiêu, tiếng mỏng hoặc biến mất. Cách kiểm gọn: gộp về mono rồi nghe. Chương 8 sẽ quay lại.

Đệm số không cho đoạn ngắn

- numpy báo lỗi ngay khi cộng hai mảng khác độ dài.
- Cách 1: đệm số không cho đoạn ngắn; bản trộn dài bằng đoạn dài nhất. Gần như luôn chọn cách này.
- Cách 2: cắt cụt đoạn dài; chỉ chọn khi cố ý.
- Muốn một nguồn bắt đầu muộn thì đệm số không vào đầu; đó là phép trễ của chương 6.

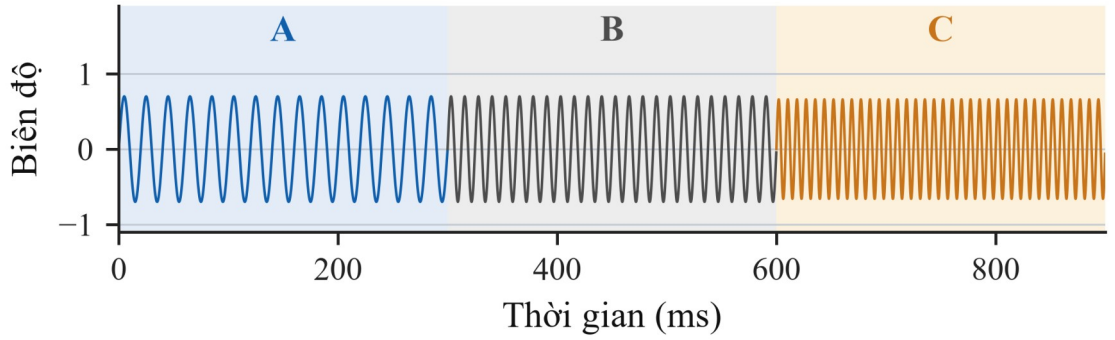


Hình 5.20 — Trộn hai đoạn khác độ dài: đệm số không cho cái ngắn

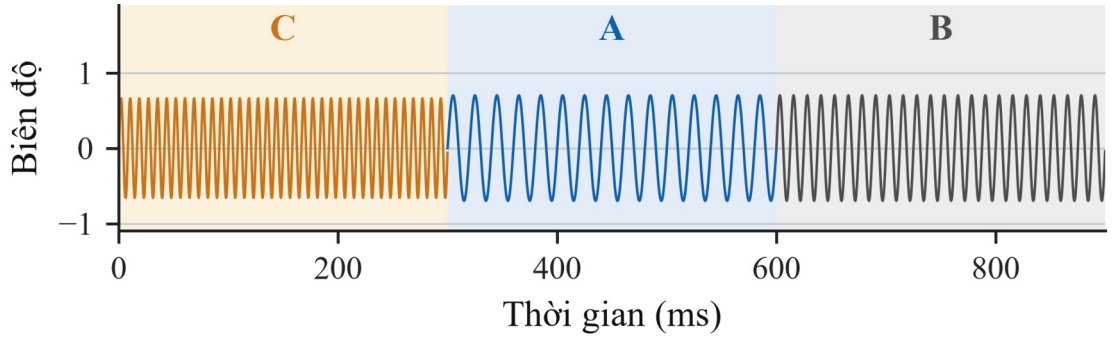
MỤC 5.6

Cắt và ghép

- Dựng phi tuyến quy về hai phép: cắt lát và nối mảng.
- Chỗ nối sinh tiếng "tách"; chương 6 chữa dứt điểm bằng fade.
- Ghép nhiều mảng đòi ba điều phải khớp.



(a) Bản gốc, cắt thành ba đoạn A, B, C



(b) Sau khi ghép lại theo thứ tự C – A – B

Cắt ra và nối lại

$$i = \lceil t_0 \cdot f_s \rceil , \quad j = \lceil t_1 \cdot f_s \rceil ; \qquad N_{\text{ghép}} = \sum_{i=1}^{K} N_i$$

(5.8)

- Cắt cảnh, đổi thứ tự, bỏ đoạn thừa, lặp đoạn đều quy về hai phép này.
- Phép cắt lát lấy một đoạn của mảng.
- Phép nối ghép nhiều mảng theo trục thời gian.

Cắt theo giây và ghép theo thứ tự mới

```
def cat(x, t0, t1, sr):
    """Lay doan [t0, t1) tinh bang giay."""
    i = max(0, round(t0 * sr))
    j = min(len(x), round(t1 * sr))
    return x[i:j].copy() if j > i else x[:0]

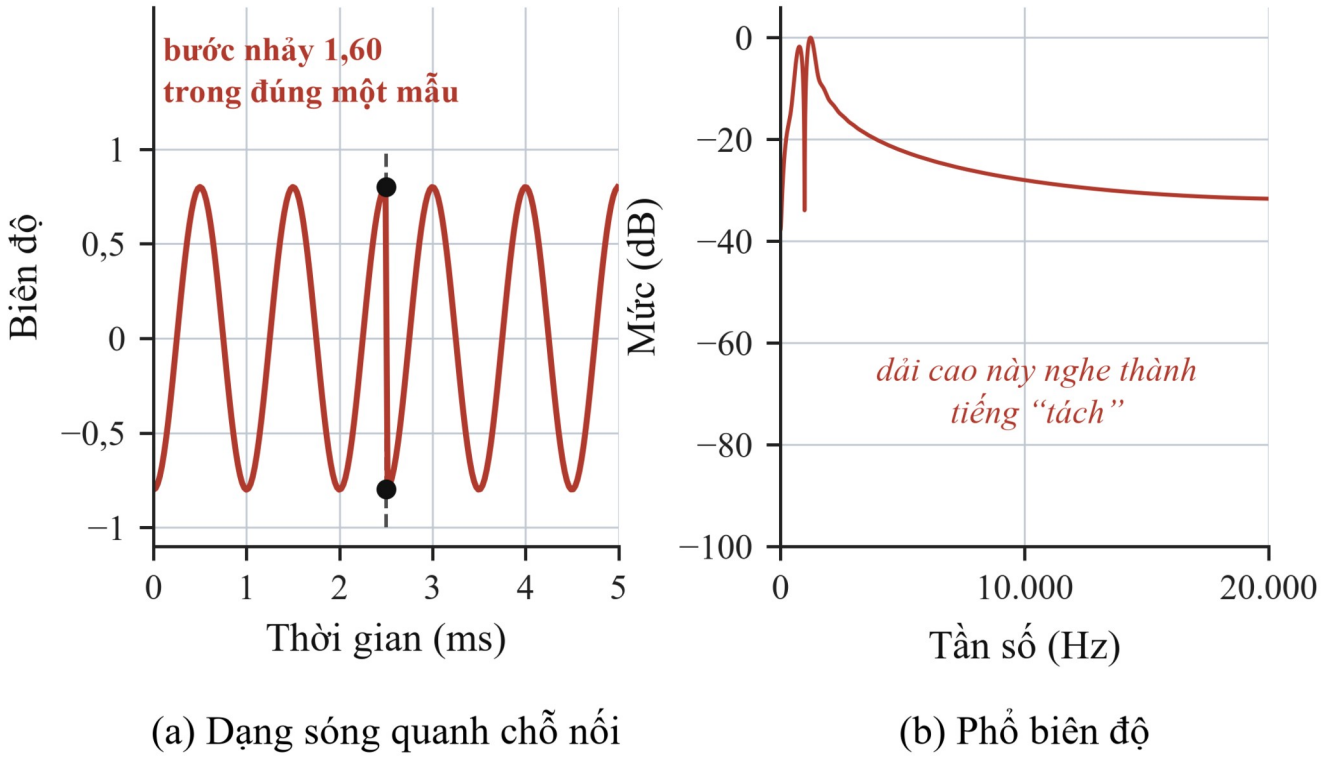
A = cat(x, 0.0, 0.3, sr)
B = cat(x, 0.3, 0.6, sr)
C = cat(x, 0.6, 0.9, sr)

moi = np.concatenate([C, A, B])       # doi thu tu
print(f"{len(moi) / sr:.2f} giay")
```

max và min chặn t0 âm hoặc t1 vượt cuối tệp; .copy() tách đoạn khỏi mảng gốc.

Bước nhảy đột ngột rải năng lượng khắp dải cao

- Đoạn trước kết thúc ở một giá trị, đoạn sau bắt đầu ở giá trị khác hẳn.
- Bước nhảy đột ngột là một xung, và xung chứa mọi tần số.
- Tai nghe chuỗi tần số cao ấy thành tiếng "tách".



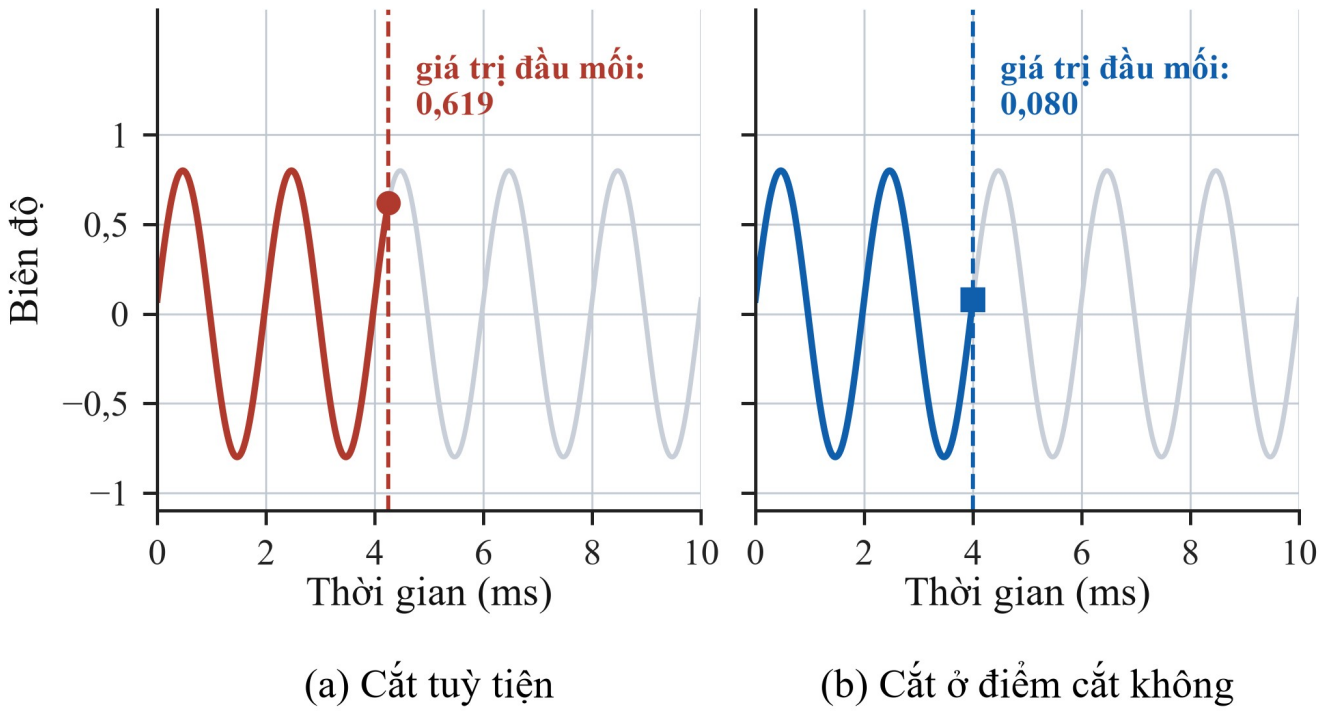
Hình 5.22 — Chỗ nối có bước nhảy đột ngột, và bước nhảy ấy rải năng lượng khắp dải cao

Nêu vấn đề ở đây, chữa ở chương 6

Cách chữa dứt điểm là fade vài mili giây ở hai đầu mỗi đoạn, thuộc nhóm thao tác của chương 6.
Người học cần nghe tiếng tách trên bản dựng của mình trước.

Cắt tuỳ tiện và cắt ở điểm cắt không

- Chọn chỗ cắt rơi đúng lúc sóng đi qua giá trị không.
- Giá trị ở đầu mối khi đó gần bằng nhau, bước nhảy nhỏ.
- Giảm nhẹ được tiếng tách mà không cần fade.



Hình 5.23 — Cắt tuỳ tiện và cắt ở điểm cắt không: giá trị ở đầu mối khác hẳn nhau

Tìm điểm cắt không gần nhất

```
def cat_khong_gan_nhat(x, i, huong=1):
    """Tim chi so gan i nhat ma song di qua khong (chieu len)."""
    x = x if x.ndim == 1 else x.mean(axis=1)
    len_qua = np.where((x[:-1] <= 0) & (x[1:] > 0))[0] + 1
    if len(len_qua) == 0:
        return i
    return int(len_qua[np.argmin(np.abs(len_qua - i))])

i = cat_khong_gan_nhat(x, round(2.5 * sr))
print(f"cat o mau {i} thay vi {round(2.5 * sr)},",
      f"lech {abs(i - round(2.5 * sr)) / sr * 1000:.2f} ms")
```

5.6.4 · BẢNG 5.7

Ba điều phải khớp khi ghép

Điều phải khớp	Nếu không khớp	Cách chặn
Số kênh	numpy báo lỗi ngay; trường hợp may mắn nhất	Đưa mọi đoạn về cùng số kênh
Tần số lấy mẫu	numpy không biết sr nên không báo; đoạn phát nhanh hoặc chậm	Kiểm sr của mọi nguồn lúc đọc
Kiểu số	numpy tự nâng kiểu; hại khi trộn float với int	Đọc mọi thứ ra float từ đầu

48.000 Hz ghép vào 44.100 Hz: phát nhanh khoảng chín phần trăm. Muốn ghép thì phải lấy mẫu lại, việc của chương 6.

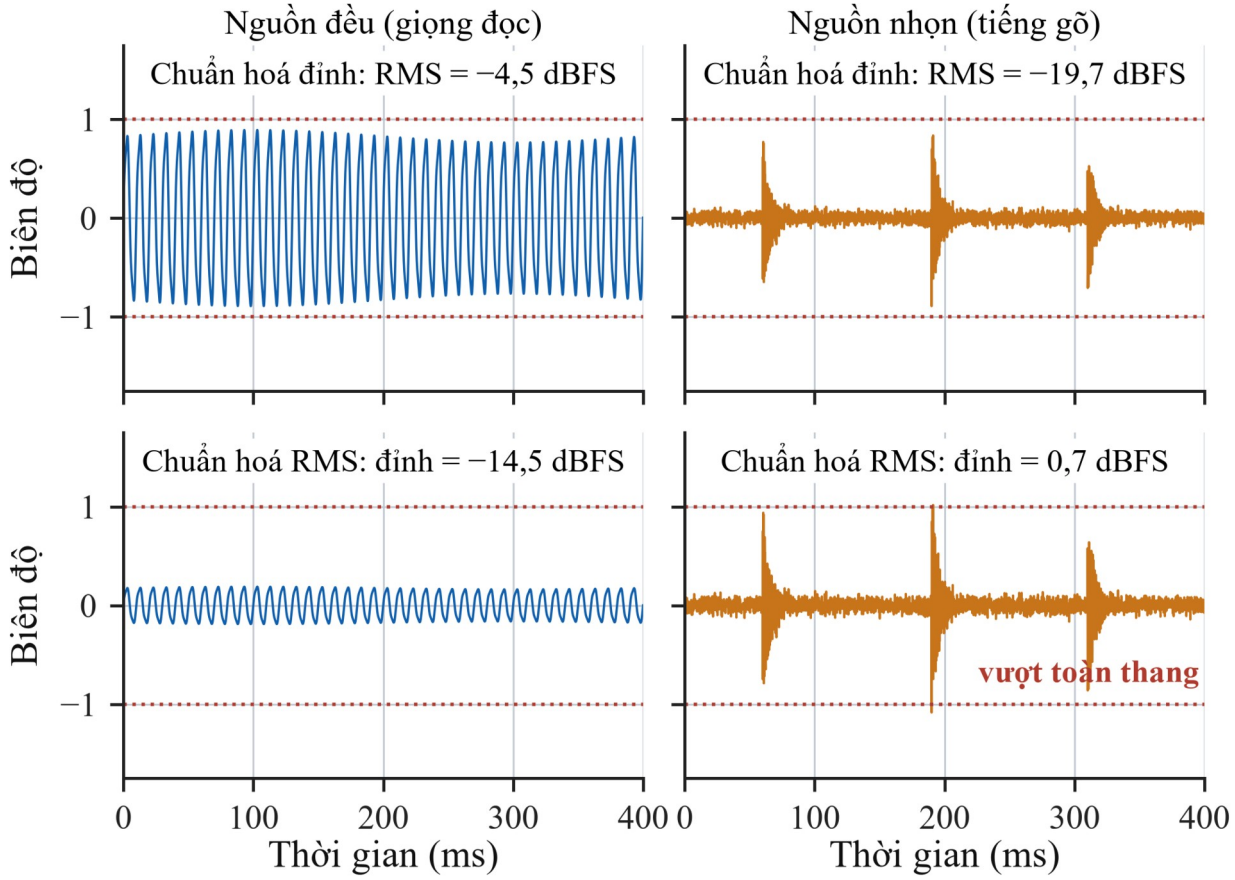
MỤC 5.7

Chuẩn hoá

- Chuẩn hoá là đưa mức về một đích thống nhất.
- "Mức" có hai nghĩa: đỉnh và RMS; chọn nghĩa nào quyết định kết quả.
- Chuẩn hoá là bước cuối, không phải bước đầu.

Hai nguồn, hai cách, bốn kết quả

- Đỉnh: dùng hết thang mà không vượt; mẫu lớn nhất rơi đúng mức đích. RMS: nghe to bằng nhau; mức trung bình bình phương rơi đúng mức đích.
- Hàng trên: cùng đỉnh −1 dBFS nhưng RMS lệch nhau tới 15 dB; nghe to nhỏ khác hẳn.
- Hàng dưới: nghe cân nhau, nhưng nguồn nhiều đỉnh nhọn thì đỉnh vượt toàn thang.



Hình 5.24 — Cùng hai nguồn, hai cách chuẩn hoá, bốn kết quả khác hẳn nhau

Hệ số đỉnh tính bằng decibel

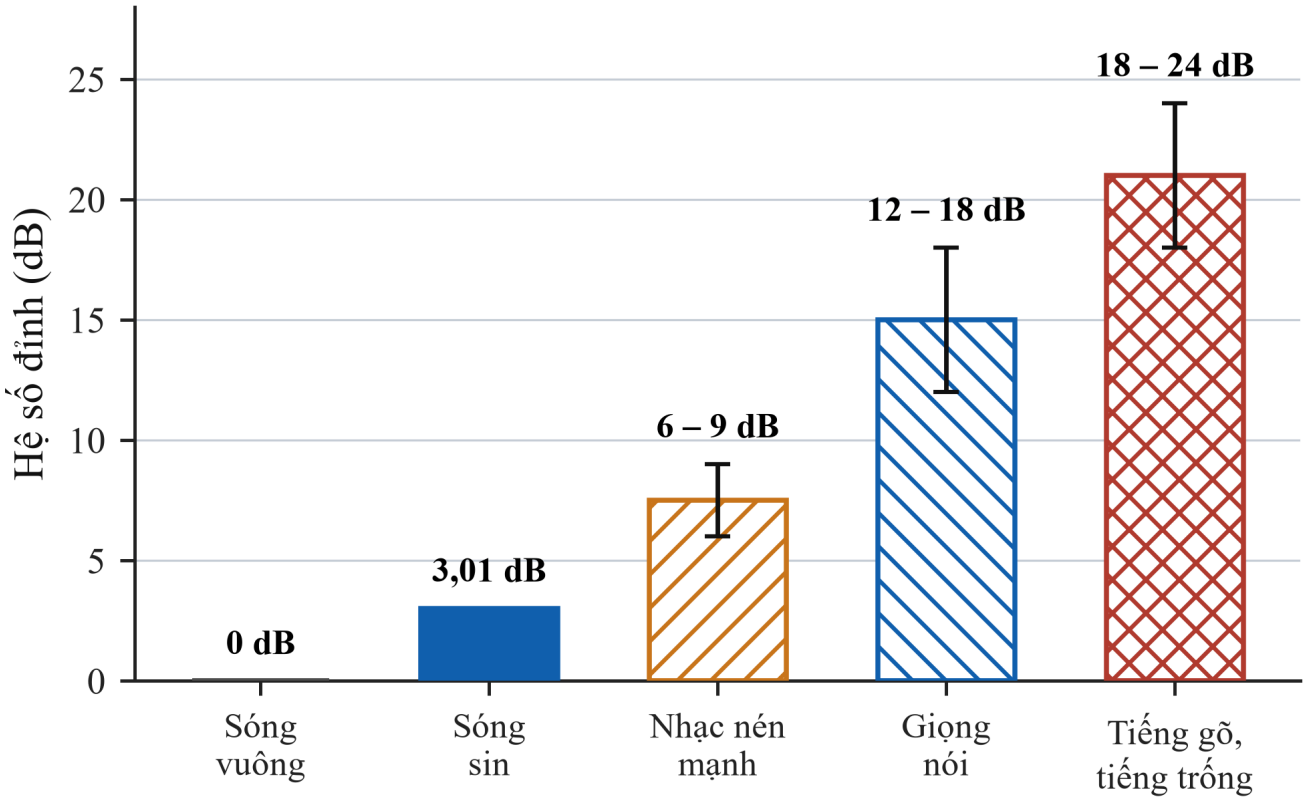
$$C = 20\log_{10}\frac{x_{\text{đỉnh}}}{x_{\text{rms}}}; \qquad x_{\text{rms}} = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1} x[n]^2}$$

(5.9)

- C cho biết đỉnh cao hơn mức trung bình bao nhiêu.
- Tức là tín hiệu "nhọn" đến mức nào.
- Con số này giải thích vì sao hai cách chuẩn hoá cho kết quả khác nhau.

Hệ số đỉnh của năm loại nguồn điển hình

- Sóng vuông thấp nhất; tiếng gõ, tiếng trống cao nhất.
- Đây là giá trị tiêu biểu để so sánh, không phải quy định.
- Mỗi bản thu một khác.



Hình 5.25 — Hệ số đỉnh của năm loại nguồn âm điển hình

Hệ số đỉnh tiêu biểu và ý nghĩa

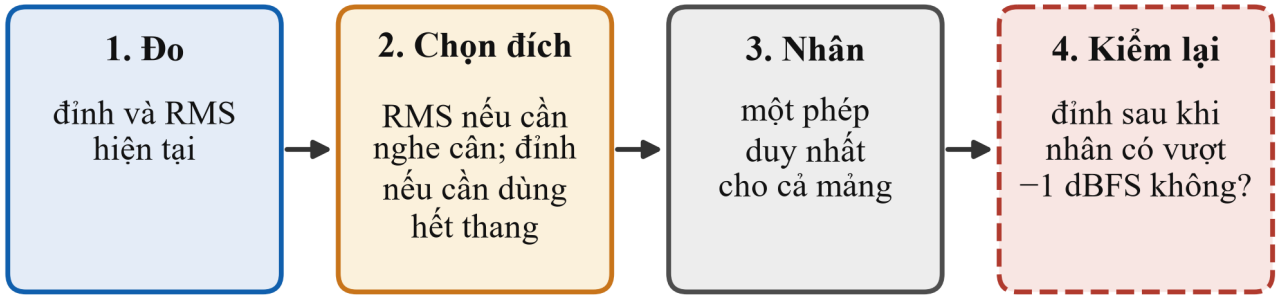
Loại nguồn	Hệ số đỉnh	Nghĩa là
Sóng vuông	0 dB	Đỉnh bằng RMS; chuẩn hoá đỉnh là to hết cỡ
Sóng sin	3,01 dB	Mốc lý thuyết, tính ra được
Nhạc nén mạnh	6 – 9 dB	Nghe to nhưng mất động lực
Giọng nói	12 – 18 dB	Chuẩn hoá đỉnh thì phần lớn thời gian vẫn nhỏ
Tiếng gõ, trống	18 – 24 dB	Rất nhọn; chuẩn hoá đỉnh gần như không làm to lên

Cách chọn giữa hai kiểu chuẩn hoá

- Cần dùng hết thang (xử lý tiếp, hệ thống có dải động hạn chế): chuẩn hoá đỉnh, nhắm −1 dBFS.
- Cần nghe cân nhau (cân mức giữa các đoạn, các tệp): chuẩn hoá RMS, rồi kiểm lại đỉnh.
- Lên nền tảng phát trực tuyến: dùng LUFS (Loudness Units Full Scale – đơn vị độ to so với toàn thang), phép đo độ to cảm nhận; chương 9 sẽ nói tới.

Bốn bước của một phép chuẩn hoá an toàn

- Đo mức hiện tại, tính gain cần chỉnh.
- Nhân cả mảng với hệ số tương ứng.
- Kiểm lại đỉnh sau chuẩn hoá.
- Bước kiểm lại hay bị bỏ nhất và là bước duy nhất bắt được lỗi.



Bước 4 là bước hay bị bỏ nhất — và là bước duy nhất bắt được lỗi.

Nếu chuẩn hoá RMS mà đỉnh vượt: hoặc hạ đích RMS xuống, hoặc dùng bộ nén (chương 9).

Không bao giờ chữa bằng cách kẹp: kẹp là méo, và méo thì không gỡ lại được.

Hình 5.26 — Bốn bước của một phép chuẩn hoá an toàn

Chuẩn hoá theo đỉnh hoặc theo RMS, có kiểm lại

```
def chuan_hoa(x, dich_db=-1.0, theo="dinh"):
    """Chuan hoa ve dich_db. theo = "dinh" hoac "rms"."""
    if theo == "dinh":
        muc = np.abs(x).max()
    else:
        muc = np.sqrt((x ** 2).mean())
    if muc <= 0:
        return x

    can = dich_db - 20 * np.log10(muc)
    y = x * (10 ** (can / 20))

    dinh_moi = 20 * np.log10(max(np.abs(y).max(), 1e-9))
    if dinh_moi > -0.1:
        print(f"CANH BAO: dinh sau chuan hoa {dinh_moi:.2f} dBFS")
    return y
```

Ba cách xử lý, một đúng hai sai

- Sai: kẹp bằng np.clip. Kẹp là méo, và méo không gỡ được.
- Sai: hạ cả bản xuống cho đỉnh vừa. Làm thế là quay lại chuẩn hoá đỉnh, mất mục đích nghe cân nhau.
- Đúng: hạ đích RMS một chút rồi thử lại; hoặc dùng bộ nén kéo đỉnh nhọn xuống trước (chương 9).

Không chuẩn hoá từng nguồn ngay sau khi đọc

- Chuẩn hoá từng nguồn về cùng đỉnh xoá quan hệ mức giữa chúng; tiếng thì thầm và tiếng hét to bằng nhau.
- Sau khi trộn, mức thay đổi hết; công chuẩn hoá lúc đầu thành vô ích.
- Trình tự đúng: đọc, cân mức bằng gain, trộn, cắt, ghép, rồi chuẩn hoá một lần ở cuối.

Ba bài thực hành và tiêu chí hoàn thành

Bài	Việc phải làm	Coi là xong khi
TH1 · Đọc và mô tả	In shape, dtype, thời lượng, số kênh, đỉnh, RMS của hai tệp WAV	Nói được tệp nào to hơn, tệp nào nhọn hơn
TH2 · Ghép có nhạc nền	Cắt ba đoạn giọng đọc, ghép thứ tự mới, trộn nhạc nền hạ 12 dB	Nghe được kết quả; chỉ ra chỗ có tiếng tách
TH3 · Chuẩn hoá và ghi	Chuẩn hoá bản dựng về −1 dBFS theo đỉnh, rồi RMS −20 dBFS; ghi hai tệp	So được hai tệp, giải thích vì sao nghe khác

TH2 là bài quan trọng nhất

Bài ghép năm trong sáu thao tác của chương và kết thúc bằng một vấn đề chưa giải được. Người học phải nghe thấy tiếng tách trên bản dựng của mình thì chương 6 mới có ý nghĩa.

Bài 1 đến bài 4

- Bài 1: tệp stereo 44.100 Hz, 3 phút 20 giây, 24 bit; tính dung lượng và bộ nhớ float64; viết mã kiểm lại.
- Bài 2: viết hàm mô tả mảng (kênh, thời lượng, đỉnh, RMS, hệ số đỉnh); giải thích chênh lệch giữa hai tệp.
- Bài 3: đỉnh 0,32; tính gain đưa về −1 dBFS; trộn thêm ba nguồn cùng đỉnh thì mỗi nguồn hạ bao nhiêu dB, theo hai cách.
- Bài 4: cắt ba đoạn, ghép ngược thứ tự, đo bước nhảy tại mỗi chỗ nối và sắp xếp theo độ lớn.

Bài 5 đến bài 7

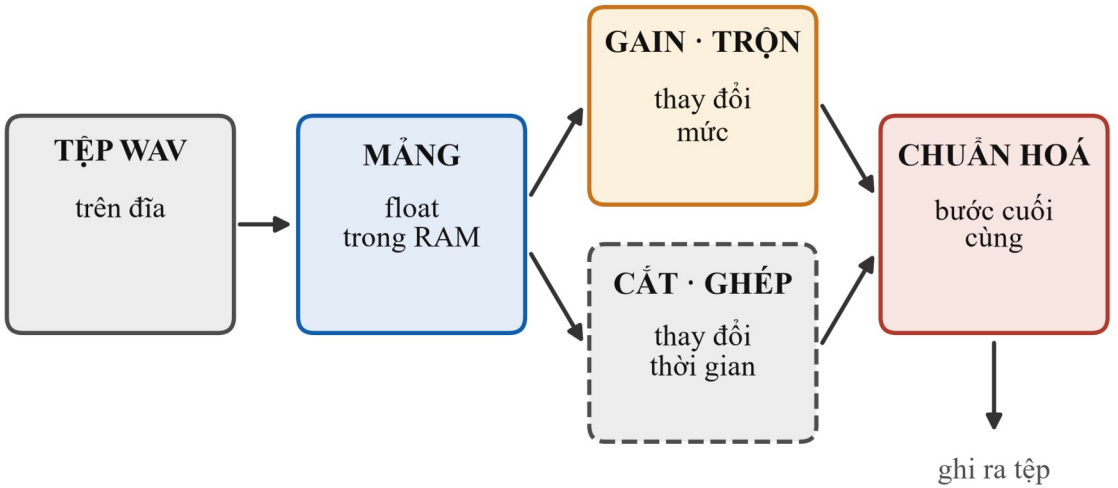
- Bài 5: chuẩn hoá đỉnh −1 dBFS và RMS −20 dBFS; so đỉnh, RMS; thử RMS −12 dBFS và liên hệ với hệ số đỉnh.
- Bài 6: phân tích quy trình sai (nhân 32.767, ép int16, nhân 2, ghi); chỉ ba lỗi và viết lại cho đúng.
- Bài 7: dựng đoạn giới thiệu 30 giây; nêu trình tự thao tác, phép đo và một rủi ro ở mỗi bước.

Bài 1 đến 5 nộp tệp .py chạy được và ảnh chụp kết quả; bài 6, 7 nộp bài viết.

Bốn lỗi hay gặp và cách tự phát hiện

Lỗi	Dấu hiệu	Cách chữa
Sửa lát cắt mà bỏ .copy()	Đoạn giữa bản thu đổi mức hoặc mất tiếng	Gọi .copy() khi định sửa lát cắt
Trộn xong không kiểm đỉnh	Tiếng rè ở chỗ to nhất	Gọi hàm đếm cắt đỉnh sau mọi phép làm to lên
Chuẩn hoá từng nguồn trước khi trộn	Mất quan hệ mức; thì thầm to bằng hét	Chuẩn hoá một lần ở cuối
Ghép hai tệp khác tần số lấy mẫu	Một đoạn phát nhanh hoặc chậm	Kiểm sr của mọi nguồn lúc đọc

Sáu thao tác nối thành một dây chuyền, bốn quy tắc đi cùng



Bốn quy tắc đi cùng dây chuyền này

1. Đọc ra float, xử lý bằng float, chỉ đổi sang int ở bước ghi.
2. Không viết vòng lặp trên từng mẫu; làm trên cả mảng.
3. Sau mỗi phép có thể làm to lên, phải đo lại đỉnh.
4. Chuẩn hoá là bước cuối, không phải bước đầu.

Hình 5.27 — Sáu thao tác nối với nhau thành một dây chuyền, và bốn quy tắc đi cùng

Muốn làm gì thì gõ gì

Đọc một tệp	`x, sr = sf.read("a.wav")`
Xem mảng có gì	`x.shape, x.dtype, len(x) / sr`
Gộp stereo về mono	`x.mean(axis=1)`
Hạ 6 dB	`x * 10 ** (-6 / 20)`
Trộn hai nguồn	`a + b  # phải cùng độ dài`
Lấy đoạn 1 – 2 giây	`x[1 * sr : 2 * sr]`
Nối hai đoạn	`np.concatenate([a, b])`
Chuẩn hoá đỉnh −1 dBFS	`x * 10 ** (-1 / 20) / np.abs(x).max()`
Ghi ra tệp 16 bit	`sf.write("b.wav", x, sr, subtype="PCM_16")`

Hình 5.28 — Muốn làm gì thì gõ gì — chín dòng đủ cho phần lớn công việc

Việc tự làm: ghép thư mục WAV thành một tệp

Bước	Dùng mục nào
Duyệt thư mục, đọc từng tệp, kiểm sr và số kênh	5.3.2
Đo RMS từng tệp để biết tệp to, tệp nhỏ	5.7.2
Chỉnh gain từng tệp về cùng mức RMS	5.4.1 và 5.7.1
Ghép tất cả theo thứ tự tên tệp	5.6.1
Chuẩn hoá bản ghép về −1 dBFS theo đỉnh	5.7.3
Ghi tệp PCM_16, kiểm đỉnh trước khi ghi	5.3.3

Khoảng ba mươi dòng; còn khiếm khuyết tiếng tách ở chỗ nối, chương 6 chữa bằng ba dòng nữa.

Tám câu nên trả lời không cần mở tài liệu

- Tệp stereo 44.100 Hz dài 10 giây cho shape bằng bao nhiêu? (5.2.2)
- Vì sao gộp về mono phải lấy trung bình? (5.2.2) · Khi nào phải gọi .copy()? (5.2.3)
- Hạ 6 dB thì nhân với bao nhiêu? Hạ 20 dB thì sao? (5.4.1)
- Vì sao cắt đỉnh nghe rè, nói bằng ngôn ngữ của chương 4? (5.4.2)
- Trộn bốn nguồn cùng mức thì hạ mỗi nguồn bao nhiêu dB? (5.5.2)
- Vì sao chỗ nối kêu "tách"? (5.6.2)
- Hai bản cùng đỉnh −1 dBFS có nghe to bằng nhau không, vì sao? (5.7.2)

Chuẩn bị cho chương 6

- Chương 6 hoàn tất mô-đun M4: trễ, đảo thời gian, fade, xử lý theo khối, đóng gói mô-đun.
- Chương 6 là mốc đầu tiên của bài tập lớn.
- Làm xong TH2 và giữ tệp có tiếng tách; chương 6 sẽ chữa chính tệp ấy.
- Đọc lại mục 5.2.4 về vector hoá; chuẩn bị một tệp WAV dài trên năm phút.

Những điều cần nhớ

1. Âm thanh là một mảng cộng một số sr nằm riêng; in shape, dtype, sr trước khi làm bất cứ điều gì.
2. Đọc ra float, xử lý bằng float, chỉ đổi sang int ở bước ghi; để thư viện tự đổi.
3. Không viết vòng lặp trên từng mẫu; làm trên cả mảng.
4. Trộn là cộng, không phải lấy trung bình; sau mỗi phép có thể làm to lên, đo lại đỉnh.
5. Cắt ghép sinh tiếng tách ở chỗ nối; chương 6 chữa bằng fade.
6. Chuẩn hoá là bước cuối: đỉnh để dùng hết thang, RMS để nghe cân; kiểm lại đỉnh sau chuẩn hoá RMS.