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

LẬP TRÌNH ÂM THANH

CHƯƠNG 6

Lập trình cơ bản phần 2

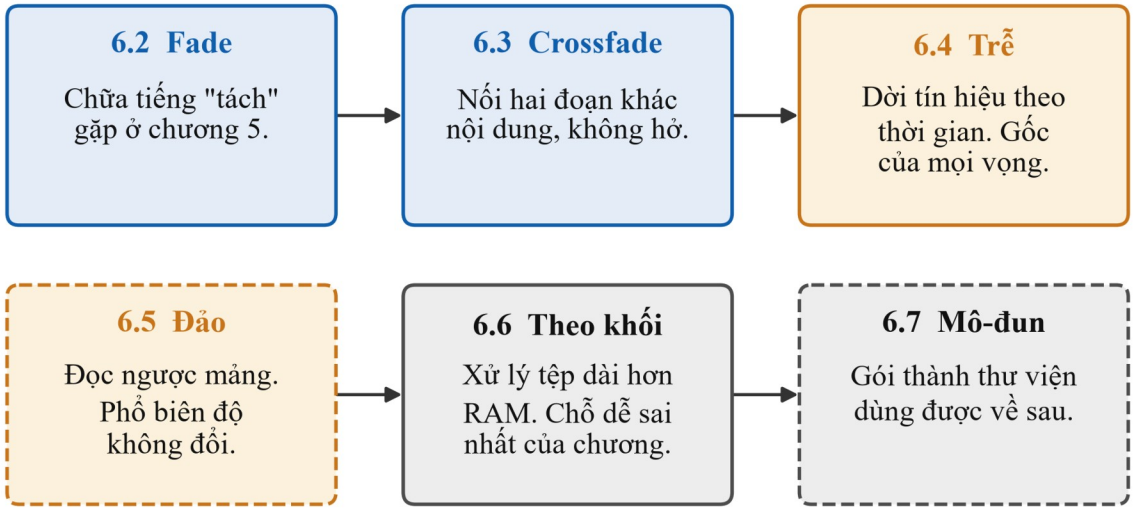
Sau chương này người học phải làm được

- Giải thích bằng số vì sao chỗ nối hai đoạn phát ra tiếng "tách", và chữa dứt điểm bằng fade.
- Chọn dạng đường bao và độ dài fade phù hợp từng tình huống, nêu cái giá của mỗi lựa chọn.
- Viết crossfade, chọn đúng đường bao theo hai nguồn có tương quan hay không, tính độ dài kết quả.
- Viết phép trễ nguyên mẫu và lẻ mẫu; đổi giữa mili giây, số mẫu, khoảng cách.
- Đảo trục thời gian đúng cách; giải thích vì sao phổ biên độ không đổi mà tai nghe khác hẳn.
- Xử lý tệp dài hơn bộ nhớ bằng cách chia khối; phân biệt phép có trạng thái với phép không nhớ.
- Mang trạng thái bộ lọc qua các khối và chứng minh kết quả bằng đúng khi lọc cả mảng.
- Gói các hàm đã viết thành mô-đun có bố cục rõ ràng, kèm bốn loại phép kiểm.

MỤC 6.1

Vị trí chương học trong học phần

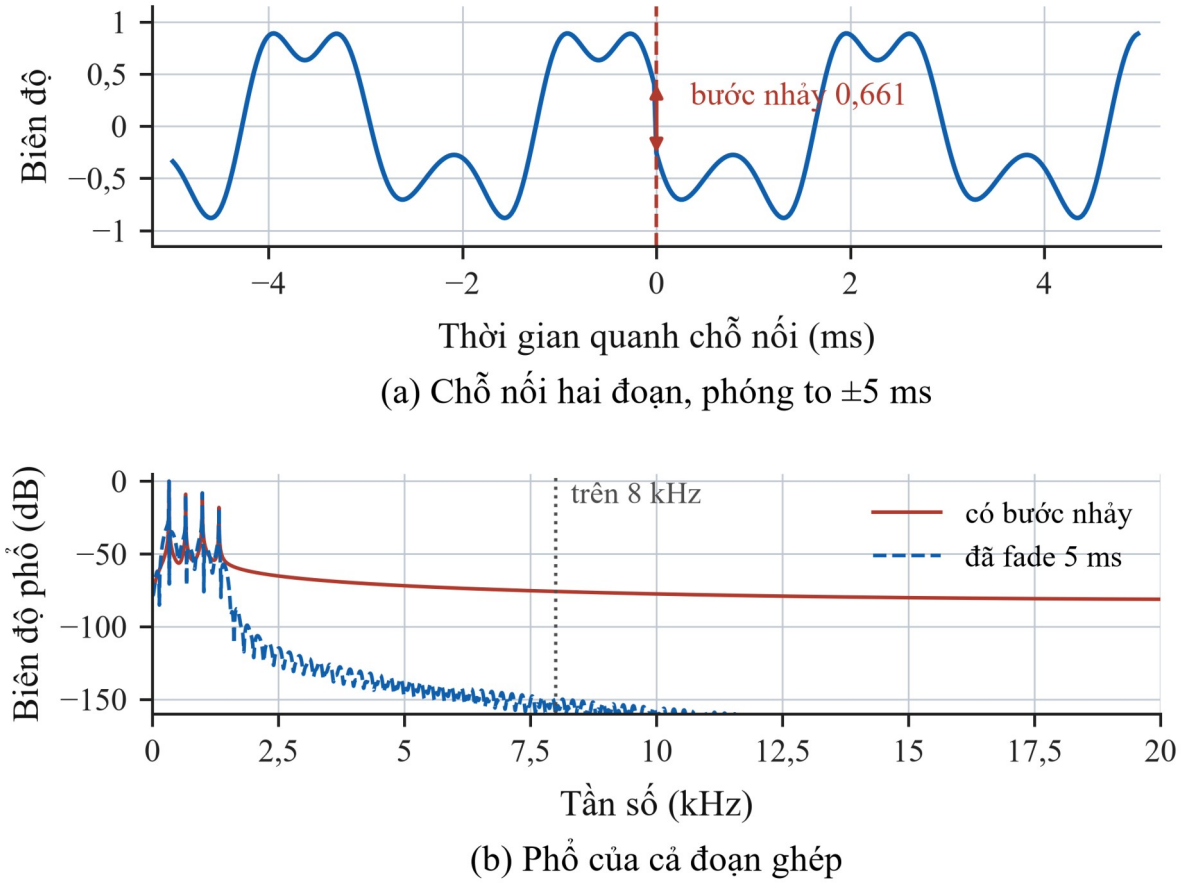
- Chương 6 khép lại mô-đun M4; chương 5 làm việc với biên độ, chương 6 làm việc với trục thời gian.
- Mỗi mục bắt đầu từ một hiện tượng nghe được rồi mới đi tới công thức.
- Mục 6.7 dạy gói mã thành mô-đun; bốn chương tới đều xây trên mô-đun ấy.



Chương 5 làm việc với BIÊN ĐỘ: nhân, cộng, chuẩn hoá. Chương 6 làm việc với TRỤC THỜI GIAN: dời, đảo, chồng lấn, cắt thành khối.

Chỗ nối phát ra tiếng "tách"

- Mục 5.6 để lại câu hỏi: ghép hai đoạn cắt thường nghe một tiếng "tách" rất ngắn.
- Nguyên nhân trực tiếp là bước nhảy giữa mẫu cuối đoạn trước và mẫu đầu đoạn sau.
- Nguồn dựng hình chỉ có bốn hài, cao nhất 1,32 kHz; năng lượng ở 10, 15, 20 kHz do chỗ nối sinh ra.
- Sau fade 5 ms, đường xanh tụt xuống dưới −140 dB ở cùng dải ấy.



Hình 6.2 — Tiếng "tách" ở chỗ nối, nhìn theo thời gian và theo tần số

Năng lượng cao tần trên 8 kHz

Tiếng "tách" là năng lượng cao tần do một bước nhảy sinh ra. Phần năng lượng trên 8 kHz so với tổng đo được nó mà không cần nghe. Ghép thẳng cho −44,9 dB; ghép sau fade 5 ms cho −128,6 dB, chênh gần 84 dB. Trong sigtools là hàm nang_luong_cao_tan(x, tu_hz=8000).

Ba nguyên tắc học chương 6

- Nghe trước, đo sau: tạo hiện tượng trên máy của người học trước khi đọc phần giải thích.
- Bộ lọc ở chương 8, hiệu ứng ở chương 10, thời gian thực ở chương 11 đều dựa trên phép trễ và xử lý theo khối.
- Mục 6.7 quan trọng ngang các mục kỹ thuật; thiếu nó, bài tập lớn không hoàn thành được.

6.1.4 · BẢNG 6.1

Bốn nội dung gần chương 6 nhưng thuộc chương sau

Dễ gộp nhầm	Vì sao không thuộc chương 6	Thuộc chương
Vọng lặp nhiều lần, hồi âm phòng	Cần vòng phản hồi, tức bộ lọc có phản hồi	Chương 8, 10
Đổi độ dài, giữ cao độ	Phải phân tích rồi tổng hợp lại theo khung	Chương 9
Xử lý khi đang phát	Cần vòng gọi lại và ràng buộc thời hạn	Chương 11
Cân độ to giữa nhiều bản thu	Đỉnh và RMS không đủ; cần thước LUFS	Chương 7

Cả bốn đều xây trên phép trễ và xử lý theo khối của chương này.

MỤC 6.2

Fade và đường bao — chữa tiếng "tách"

- Fade là một phép nhân từng phần tử với mảng đường bao đi từ 0 lên 1.
- Dạng đường bao quyết định còn nghe thấy tiếng tách hay không.
- Câu hỏi thực tế: chọn bao nhiêu mili giây, và đặt fade ở đâu cho đúng.

Đường bao tuyến tính và nửa cosin

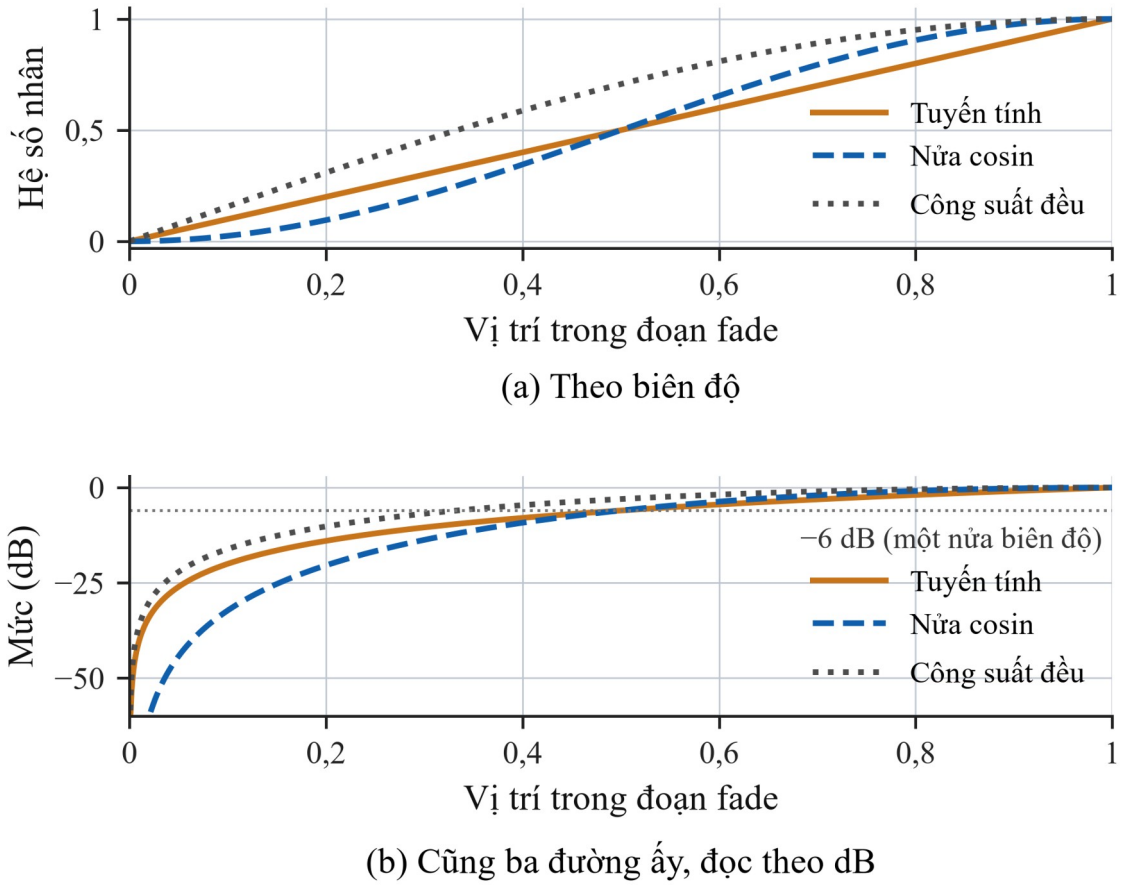
$$y[n] = w[n] \cdot x[n]\,; \qquad w_{\mathrm{tt}}[n] = \frac{n}{N-1}, \quad w_{\cos}[n] = \frac{1}{2}\left[1 - \cos\frac{\pi n}{N-1}\right]$$

(6.1)

- w là đường bao dài N mẫu; w_tt là đường bao tuyến tính; w_cos là nửa cửa sổ Hann.
- Cả hai đi từ 0 tới 1 và đều dùng được.
- Khác nhau ở đạo hàm tại hai đầu; chính đạo hàm quyết định còn tiếng tách hay không.

Ba dạng đường bao theo biên độ và theo decibel

- Theo thang decibel, đường tuyến tính lao xuống nhanh ở cuối rồi chững lại: nghe như fade "hụt".
- Đường nửa cosin có đạo hàm bằng không ở cả hai đầu, không tạo góc nhọn trong tín hiệu.
- Vì vậy nửa cosin là lựa chọn mặc định để chống tiếng tách.



Hình 6.3 — Ba dạng đường bao, nhìn theo biên độ và theo decibel

Căn bậc hai của nửa cosin

$$w_{cs}[n] = \sqrt{w_{cos}[n]}\ ; \qquad w_{cs}^2[n] + w_{cs}^2[N-1-n] = 1$$

(6.2)

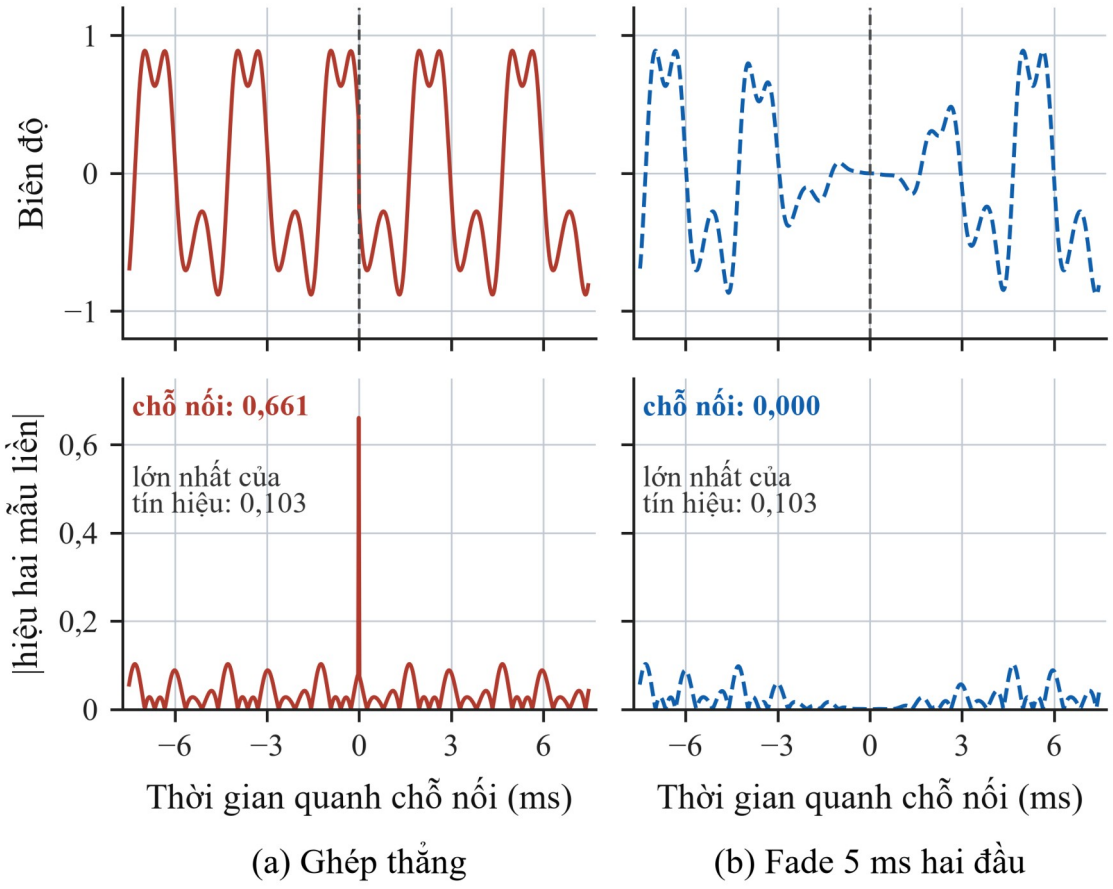
- Dòng thứ hai: bình phương đường bao tại n cộng bình phương tại vị trí đối xứng bằng đúng 1.
- Bình phương biên độ là công suất, nên tổng công suất giữ nguyên khi hai nguồn chồng lên nhau.
- Chỉ dùng cho crossfade giữa hai nguồn độc lập, không dùng cho fade vào và fade ra thông thường.

Ba đường bao và chỗ dùng

Đường bao	Công thức	Dùng khi nào
Tuyến tính	w = n / (N − 1)	Hầu như không dùng cho fade ngắn; hợp lý cho fade dài vài giây
Nửa cosin	w = ½·[1 − cos(πn/(N−1))]	Mặc định: fade vào, fade ra, cắt ghép, crossfade cùng một bản thu
Công suất đều	w = căn của nửa cosin	Chỉ cho crossfade giữa hai nguồn độc lập

Chỗ nối trước và sau khi fade

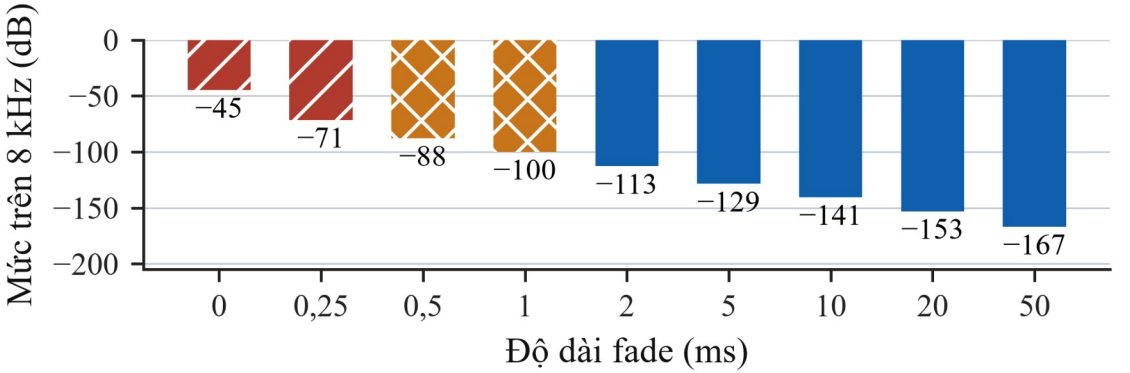
- Mẫu cuối đoạn A về 0 và mẫu đầu đoạn B ở 0 thì chỗ nối không còn bước nhảy.
- Bản ghép thẳng để lại gai 0,660, gấp sáu lần biến thiên lớn nhất của tín hiệu (0,110).
- Sau fade, bước nhảy tại chỗ nối đúng bằng không: nhân với 0 thì không có sai số làm tròn.
- Đây là một trong số ít chỗ kiểm thử cho một dấu bằng tuyệt đối, không phải ngưỡng dung sai.



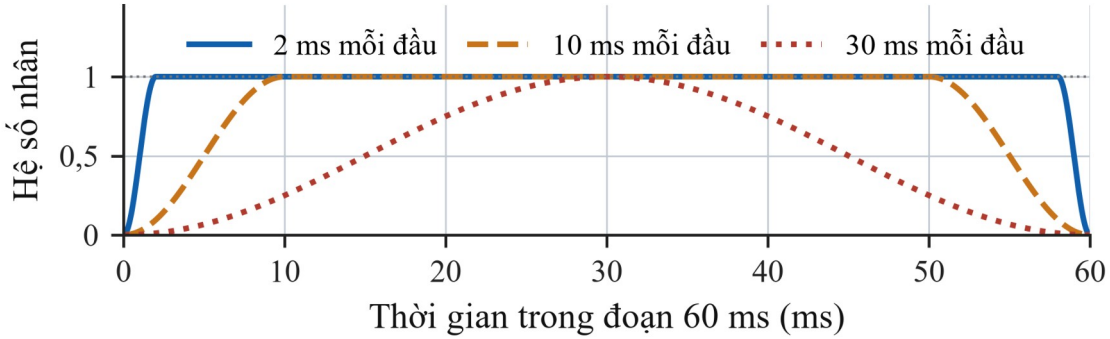
Hình 6.4 — Chỗ nối trước và sau khi fade — dạng sóng và hiệu hai mẫu liền

Hai áp lực ngược chiều

- Áp lực về phía dài: bước nhảy trải trên nhiều mẫu hơn; từ không fade tới 5 ms, từ −45 dB xuống −129 dB.
- Áp lực về phía ngắn: fade lấn vào nội dung.
- Trên đoạn 60 ms, fade 30 ms mỗi đầu chiếm trọn đoạn và làm nhỏ phần đầu tiếng.
- Quy ước học phần: chưa có lý do khác thì fade 5 ms, tức 220 mẫu ở 44,1 kHz.



(a) Fade càng dài, tiếng tách càng nhỏ



(b) Đường bao trên một đoạn chỉ dài 60 ms

Hình 6.5 — Hai áp lực ngược chiều khi chọn độ dài fade

Độ dài fade theo tình huống

Độ dài	Dùng cho	Người nghe cảm nhận
0,5 – 2 ms	Cắt ghép lời nói; ghép khối	Không nghe có fade, chỉ hết tiếng tách
5 – 10 ms	Cắt ghép nhạc; mở, đóng đoạn mẫu	Vẫn không nghe thấy; mặc định an toàn
20 – 50 ms	Chuyển giữa hai câu nhạc; che chỗ vá	Bắt đầu nghe ra là "mờ dần"
0,5 – 3 s	Kết bài, mở bài, chuyển cảnh phim	Hiệu ứng nghệ thuật có chủ ý

Bốn khoảng độ dài fade và tình huống tương ứng

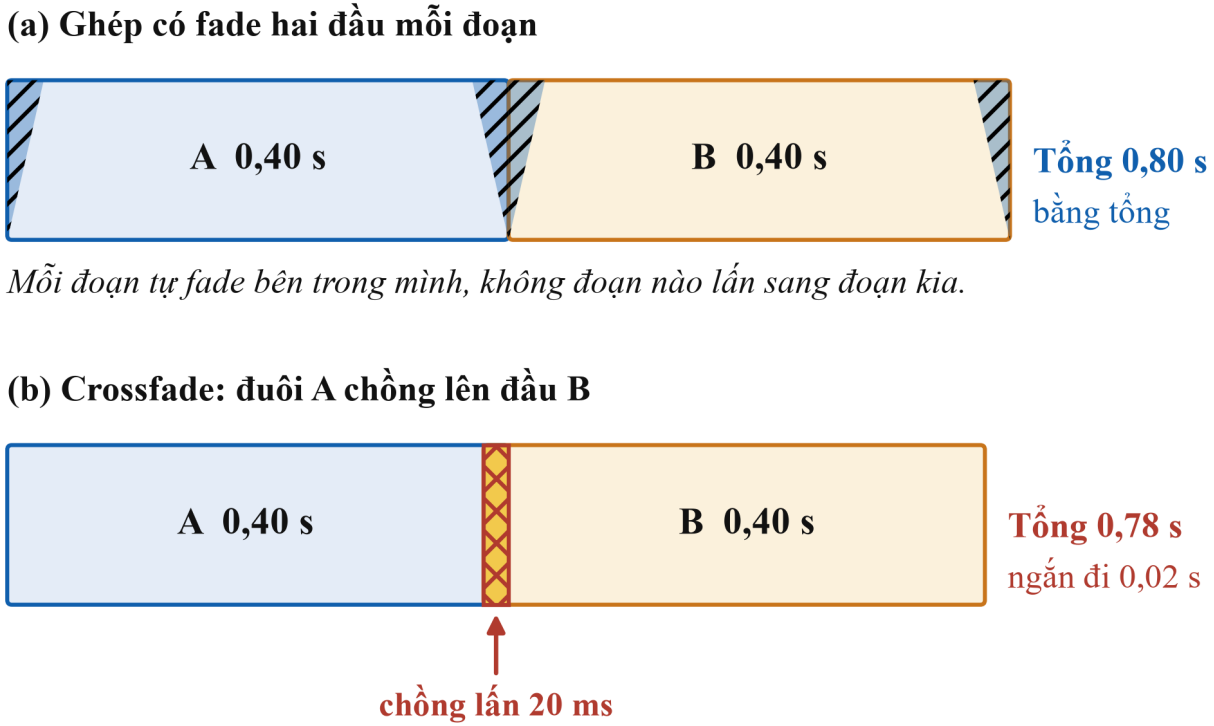
Độ dài	Dùng cho	Cảm nhận
0,5 – 2 ms	Cắt ghép lời nói; ghép khối trong xử lý theo khối	Ngắn đến mức không nghe thấy là có fade
5 – 10 ms	Cắt ghép nhạc; mở và đóng một đoạn mẫu	Mặc định an toàn, dùng khi chưa biết chọn gì
20 – 50 ms	Chuyển tiếp giữa hai câu nhạc; che chỗ vá	Đã nghe ra là "mờ dần", không còn vô hình
0,5 – 3 s	Kết bài, mở bài, chuyển cảnh phim	Là hiệu ứng nghệ thuật, không còn là chống tách

Không có con số đúng duy nhất: chọn theo việc đang làm, rồi nghe lại chỗ nối để xác nhận.

Hình 6.6 — Bốn khoảng độ dài fade và tình huống tương ứng

Fade không làm đổi độ dài

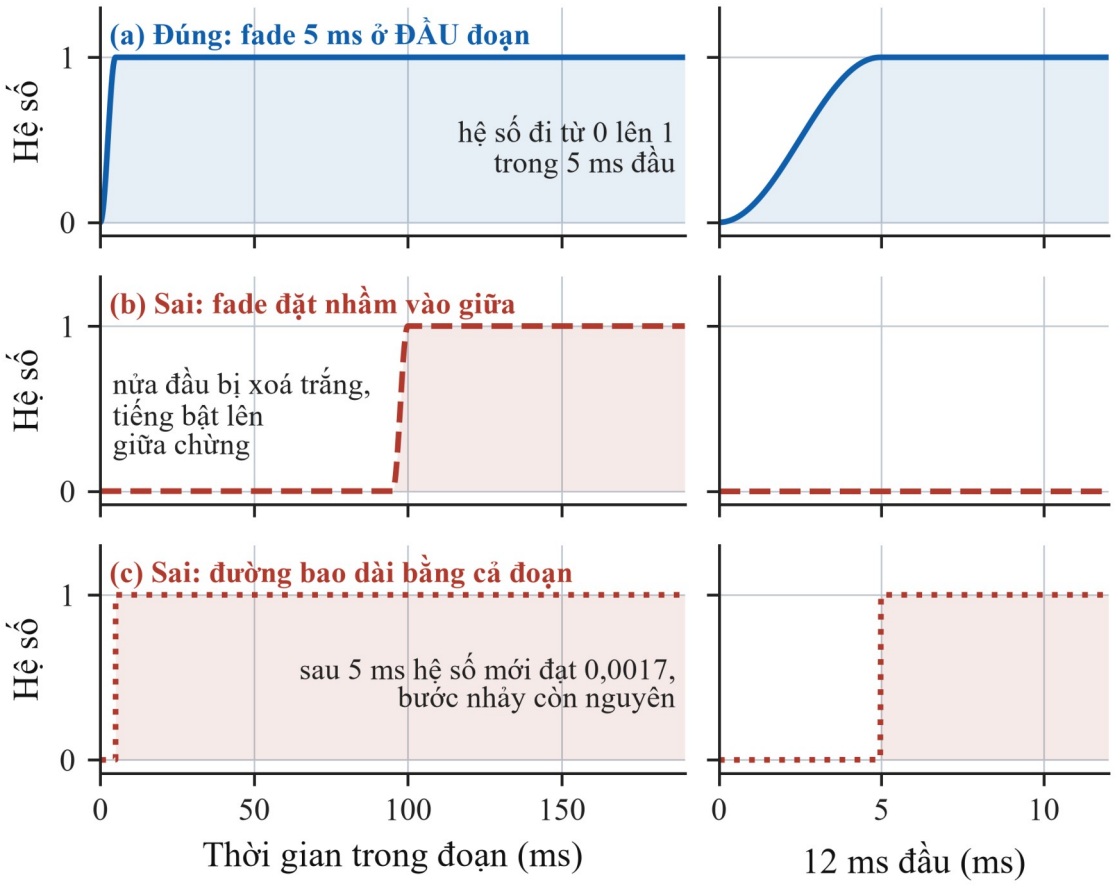
- Fade nhân đường bao vào các mẫu có sẵn: không mẫu nào thêm vào, không mẫu nào bỏ đi.
- Ghép ba đoạn 0,40 s đã fade được đúng 1,20 s.
- Crossfade cho hai đoạn chồng lên nhau nên ngắn đi đúng bằng phần chồng lấn ở mỗi chỗ nối.



Hình 6.7 — Ghép có fade giữ nguyên độ dài; crossfade thì không

Sai mà máy không báo lỗi

- Đặt nhầm vị trí: nhân vào giữa mảng, nửa đầu bị xoá trắng.
- Đường bao dài bằng cả đoạn, chỉ lấy 220 phần tử đầu: hệ số mới đạt 0,0017, vẫn còn tiếng tách.
- Fade một đầu, bỏ sót đầu kia: bước nhảy giảm khoảng một nửa, vẫn nghe được trên tai nghe tốt.
- Tự kiểm: in mẫu đầu, mẫu cuối và hiệu lớn nhất giữa hai mẫu liền ở chỗ nối; cả ba phải bằng 0.



Hình 6.8 — Một cách đặt fade đúng và hai cách sai không báo lỗi

Đường bao fade và hàm fade vào

```
import numpy as np

def duong_bao_fade(n, kieu="cosin", len_dan=True):
    """Đường bao dài n mẫu, đi lên nếu len_dan=True."""
    if n <= 0:  return np.zeros(0)
    if n == 1:  return np.ones(1)
    u = np.linspace(0.0, 1.0, n)
    if kieu == "tuyen_tinh":
        w = u
    elif kieu == "cong_suat":
        w = np.sqrt(0.5 * (1.0 - np.cos(np.pi * u)))
    else:                       # "cosin" – mặc định
        w = 0.5 * (1.0 - np.cos(np.pi * u))
    return w if len_dan else w[::-1]

def fade_vao(x, ms=5.0, sr=44100, kieu="cosin"):
    a = np.array(x, dtype=float, copy=True)
    n = min(int(round(ms / 1000 * sr)), a.shape[0])
    if n <= 0:  return a
    w = duong_bao_fade(n, kieu, True)
    a[:n] *= w if a.ndim == 1 else w[:, None]
    return a
```

(rút gọn) copy=True không sửa mảng người gọi; min(...) chặn fade dài hơn đoạn; w[:, None] nhân đúng cho stereo.

Fade hai đầu và ghép mượt

```
# fade_ra: giống fade_vao, nhưng dùng đường bao đi xuống
#   w = duong_bao_fade(n, kieu, False);  a[-n:] *= ...

def fade_hai_dau(x, ms=5.0, sr=44100, kieu="cosin"):
    return fade_ra(fade_vao(x, ms, sr, kieu), ms, sr, kieu)

def ghep_muot(doan, ms=5.0, sr=44100, kieu="cosin"):
    """Fade hai đầu MỖI đoạn rồi mới nối. Không đổi độ dài."""
    return np.concatenate(
        [fade_hai_dau(x, ms, sr, kieu) for x in doan], axis=0)

# kiểm ngay tại chỗ:
a, b = doan[0], doan[1]
truoc = abs(b[0] - a[-1])
fa, fb = fade_hai_dau(a), fade_hai_dau(b)
sau = abs(fb[0] - fa[-1])
print(f"bước nhảy: {truoc:.3f} -> {sau:.3f}")
# bước nhảy: 0.660 -> 0.000
```

MỤC 6.3

Crossfade — nối hai đoạn không cùng nội dung

- Fade hai đầu để lại cái hõm ở chỗ nối; crossfade cho hai đoạn chồng lên nhau.
- Đường bao đúng phụ thuộc hai nguồn có tương quan hay không.
- Crossfade làm kết quả ngắn đi; phải tính trước độ dài.

Crossfade và độ dài kết quả

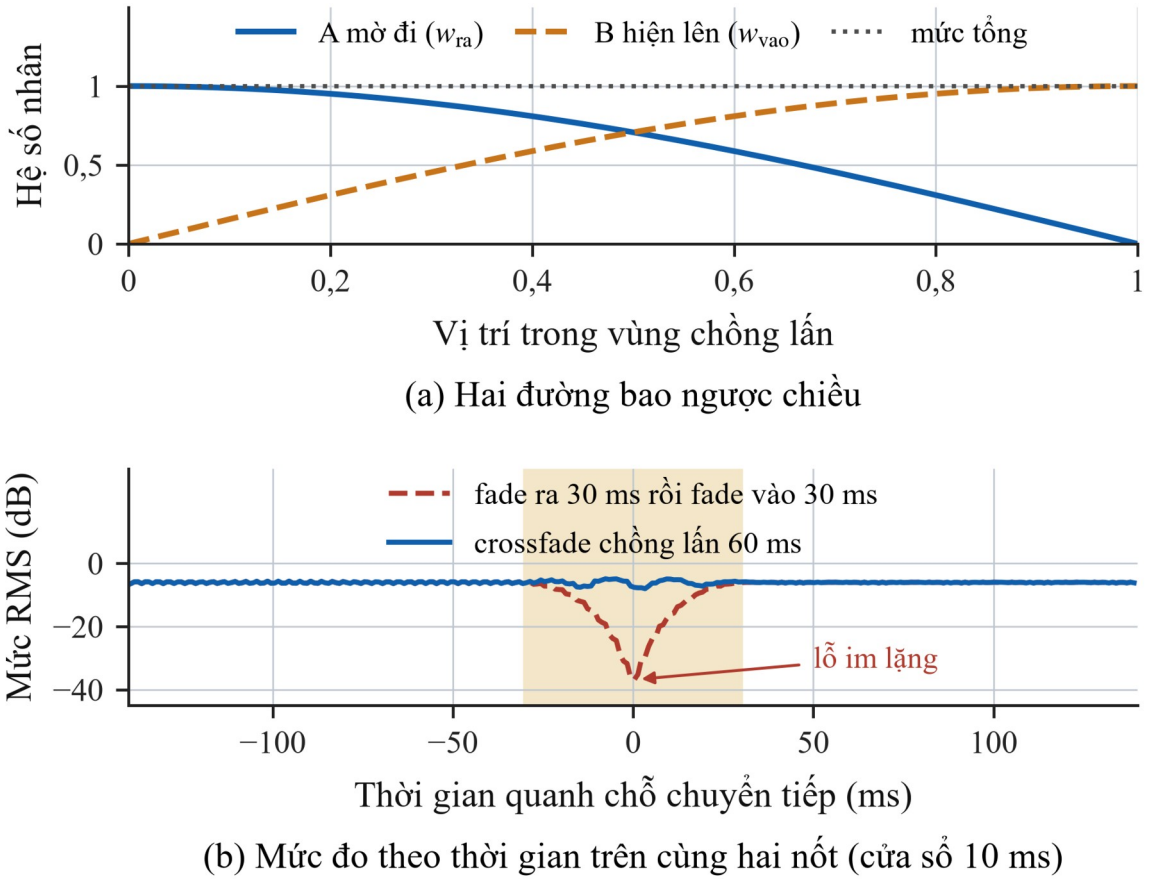
$$y[n] = a[n]\,w_{\text{ra}}[n] + b[n]\,w_{\text{vào}}[n]\,; \qquad N_{\text{ra}} = N_a + N_b - N_{\text{chồng}}$$

(6.3)

- Fade hai đầu làm âm thanh tụt về không rồi lên lại; với fade 30 ms thành một lỗ im lặng.
- Trong vùng chồng lấn cả hai đoạn cùng phát nên mức tổng không tụt về không.
- N_ra = N_a + N_b − N_chồng: kết quả ngắn hơn tổng đúng bằng phần chồng lấn.

Mức đo được ở chỗ chuyển tiếp

- Đường đỏ: fade ra rồi fade vào, 30 ms mỗi bên; mức RMS (Root Mean Square – giá trị hiệu dụng) tụt xuống dưới −35 dB ở giữa.
- Đường xanh: crossfade chồng lấn 60 ms trên cùng hai nốt; mức gần như không đổi.



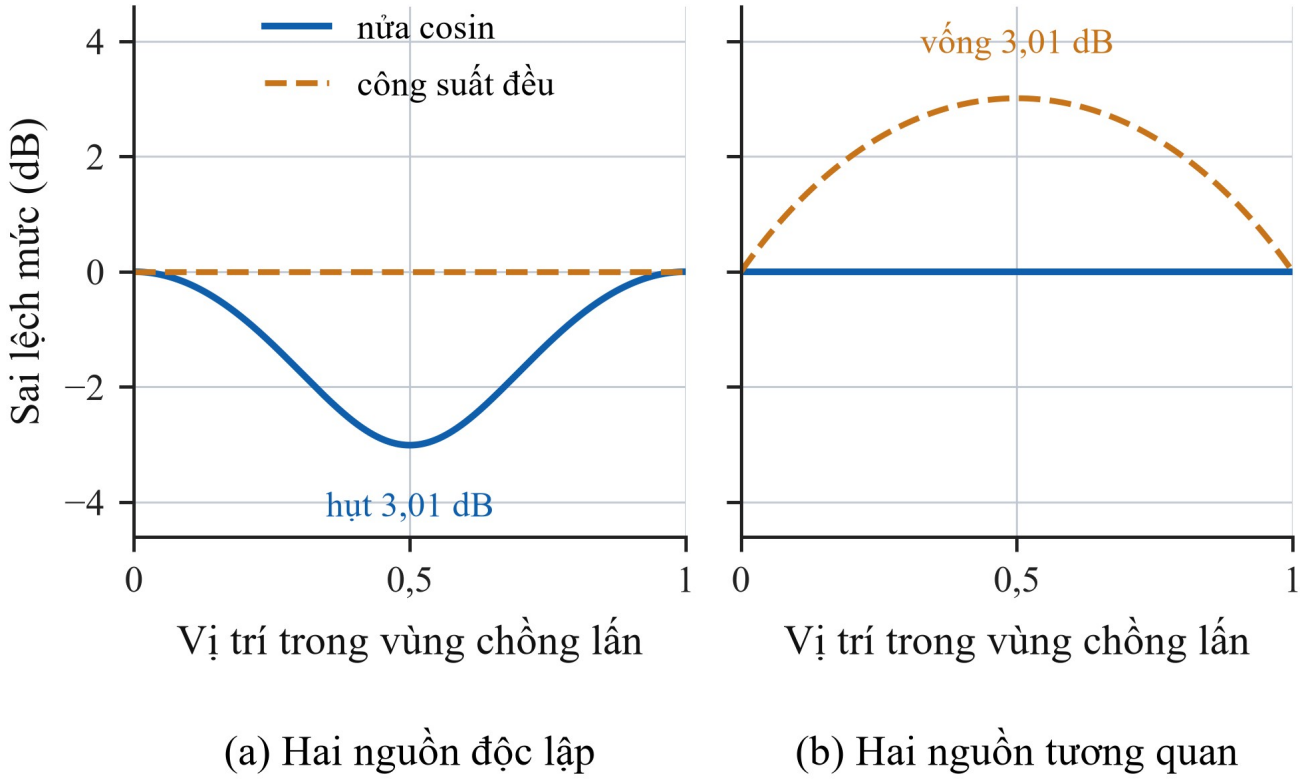
Hình 6.9 — Hai đường bao ngược chiều, và mức đo được ở chỗ chuyển tiếp

Hai nguồn, hai điều kiện

- Hai nguồn độc lập (hai bản thu, hai bài hát): biên độ cộng theo căn của tổng bình phương.
- Muốn mức tổng không đổi phải có $w^2 + v^2 = 1$: đường bao công suất đều.
- Hai nguồn tương quan (hai lát của cùng bản thu): biên độ cộng thẳng.
- Muốn mức tổng không đổi phải có $w + v = 1$: đường bao nửa cosin.

Chọn nhầm đường bao lệch 3,01 dB

- Chọn nhầm không báo lỗi; mức giữa chỗ chuyển tiếp lệch đúng 3,01 dB.
- Hụt nếu dùng nửa cosin cho hai nguồn độc lập; vống nếu dùng công suất đều cho hai nguồn tương quan.



Hình 6.10 — Sai lệch mức ở chỗ chuyển tiếp khi chọn nhầm đường bao

Bốn tổ hợp đường bao và nguồn

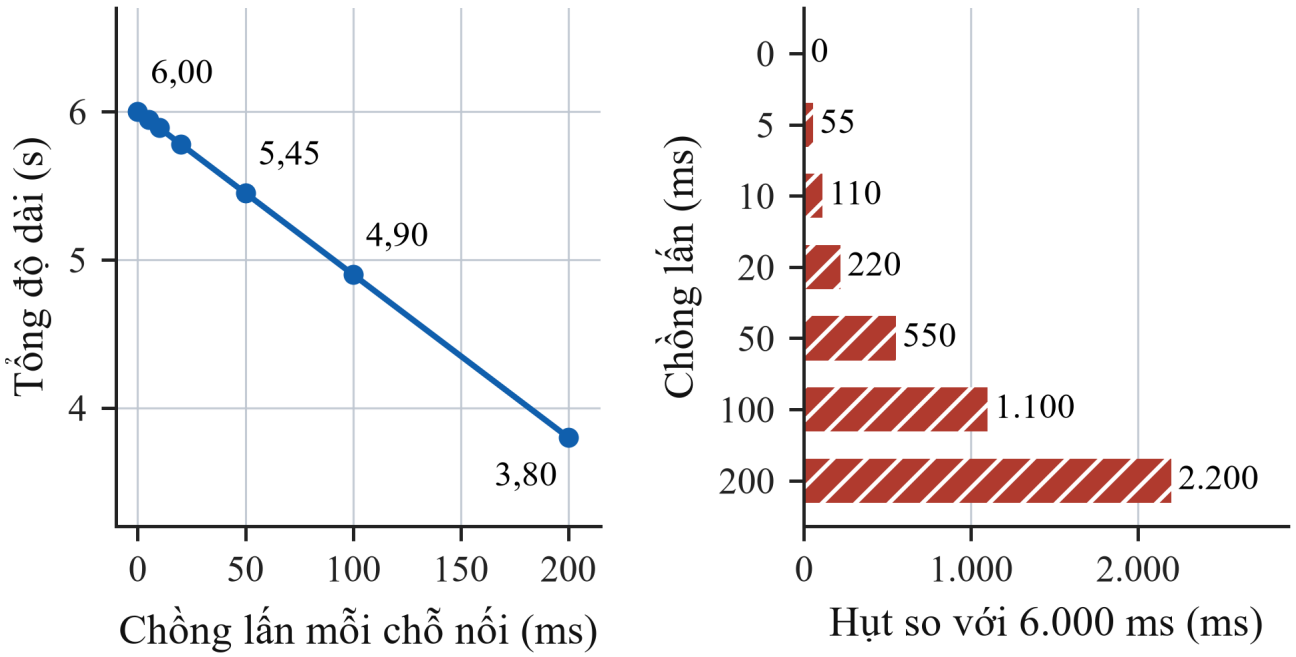
Nguồn	Nửa cosin	Công suất đều
Độc lập (hai bản thu khác nhau)	Hụt 3,01 dB ở giữa	Đúng, mức không đổi
Tương quan (cùng một bản thu)	Đúng, mức không đổi	Vống 3,01 dB ở giữa

Ba decibel có đáng bận tâm không

Ba decibel là ngưỡng nhiều người nghe bắt đầu nhận ra thay đổi độ to. Một bản dựng có hàng chục chỗ chuyển tiếp hụt 3 dB sẽ nghe "không chắc tay". Hai đoạn là hai tệp khác nhau thì dùng công suất đều; hai lát cùng một tệp thì dùng nửa cosin. Trong sigtools, kieu mặc định của crossfade là "cong_suat".

Sai lệch độ dài tích luỹ

- Mười hai đoạn nửa giây, nối bằng crossfade 200 ms, cho 3,80 s chứ không phải 6,00 s.
- Ở nhịp 120 nhịp một phút, 2,2 s hụt là hơn bốn phách.
- Cách xử lý: tính ngược từ đầu; K đoạn có K − 1 chỗ nối, tổng phần mất là (K − 1) nhân độ dài chồng lấn.



Hình 6.11 — Sai lệch độ dài tích luỹ khi nối nhiều đoạn bằng crossfade

Crossfade hai đoạn

```
def crossfade(a, b, ms=20.0, sr=44100, kieu="cong_suat"):
    """Đuôi a mờ đi trong khi đầu b hiện lên.
    Kết quả NGẮN hơn tổng hai độ dài đúng bằng phần chồng lấn."""
    a = np.asarray(a, dtype=float)
    b = np.asarray(b, dtype=float)
    n = min(int(round(ms / 1000 * sr)), a.shape[0], b.shape[0])
    if n <= 0:
        return np.concatenate([a, b], axis=0)
    xuong = duong_bao_fade(n, kieu, False)
    len_  = duong_bao_fade(n, kieu, True)
    if a.ndim == 2:
        xuong, len_ = xuong[:, None], len_[:, None]
    giua = a[-n:] * xuong + b[:n] * len_
    return np.concatenate([a[:-n], giua, b[n:]], axis=0)

# phép kiểm độ dài bắt buộc:
y = crossfade(a, b, ms=20.0)
assert len(y) == len(a) + len(b) - int(round(0.020 * 44100))
```

Dòng assert bắt lỗi hay gặp nhất: quên trừ phần chồng lấn khi tính vị trí các đoạn sau.

Ba tình huống chọn crossfade là sai

- Độ dài phải đúng tuyệt đối (dựng theo nhịp, khớp hình, ghép khối): dùng ghép có fade.
- Hai đoạn cùng nội dung ở vùng chồng lấn: lệch pha, sinh hiện tượng lược tần số, âm nghe rỗng đi.
- Vùng chồng lấn rơi vào tiếng gõ: đầu tiếng bị mờ dần và thành tiếng "thụt".
- Đặt chuyển tiếp vào chỗ biên độ nhỏ nhất quanh điểm nối: khoảng lặng giữa hai từ, hoặc ngay trước một phách.

MỤC 6.4

Phép trễ

- Trễ là dịch mọi mẫu sang phải và đệm số không vào đầu mảng.
- Ba đại lượng quy đổi qua lại: mili giây, số mẫu, khoảng cách.
- Trễ lẻ mẫu phải nội suy và phải trả giá.

Số mẫu trễ

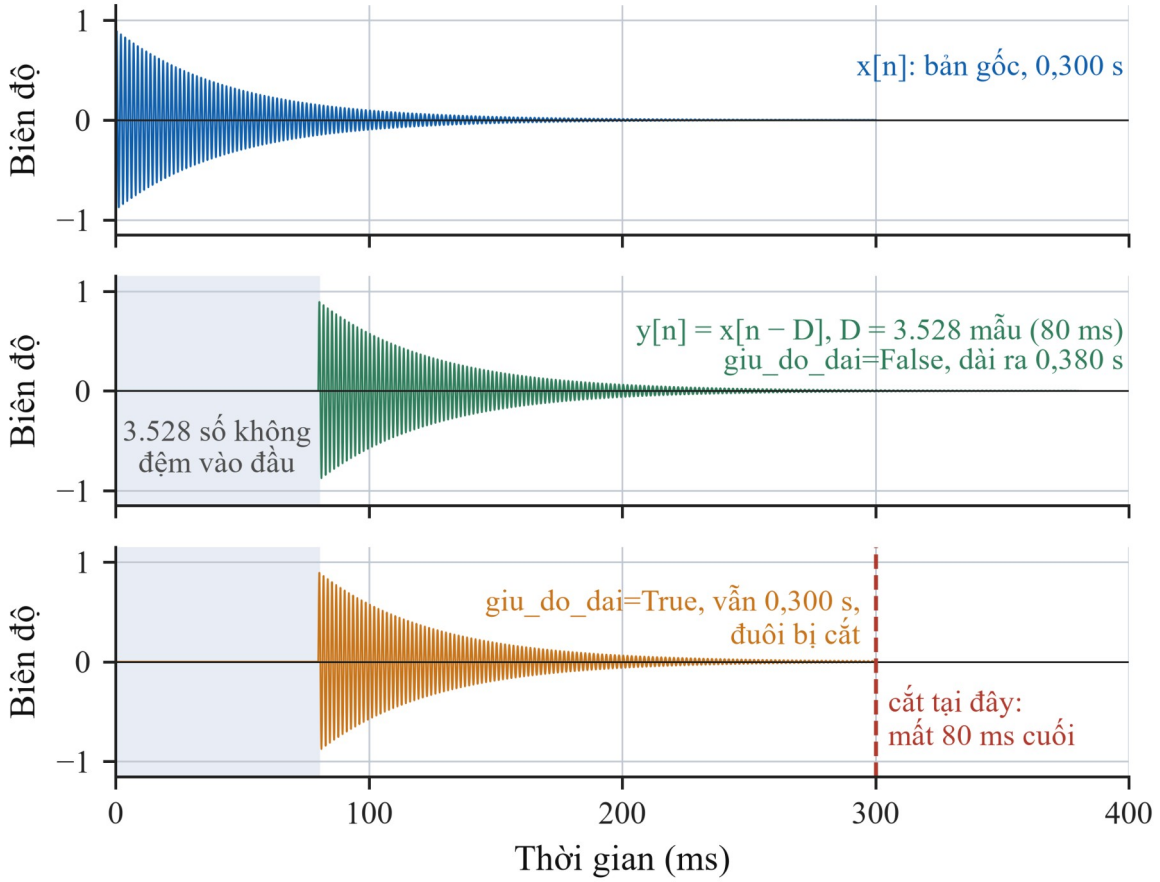
$$y[n] = x[n - D], \qquad D = \left\lfloor \frac{t_{\text{ms}}}{1000} \cdot f_{\text{s}} \right\rfloor$$

(6.4)

- D là số mẫu trễ, t_ms là độ trễ tính bằng mili giây, fs là tần số lấy mẫu.
- Trên mảng số, D chỉ nhận giá trị nguyên mẫu; phần lẻ ở mục 6.4.4.
- Ở 44,1 kHz: 20 ms là 882 mẫu; 80 ms là 3 528 mẫu; 250 ms là 11 025 mẫu.

Giữ nguyên độ dài hoặc để mảng dài ra

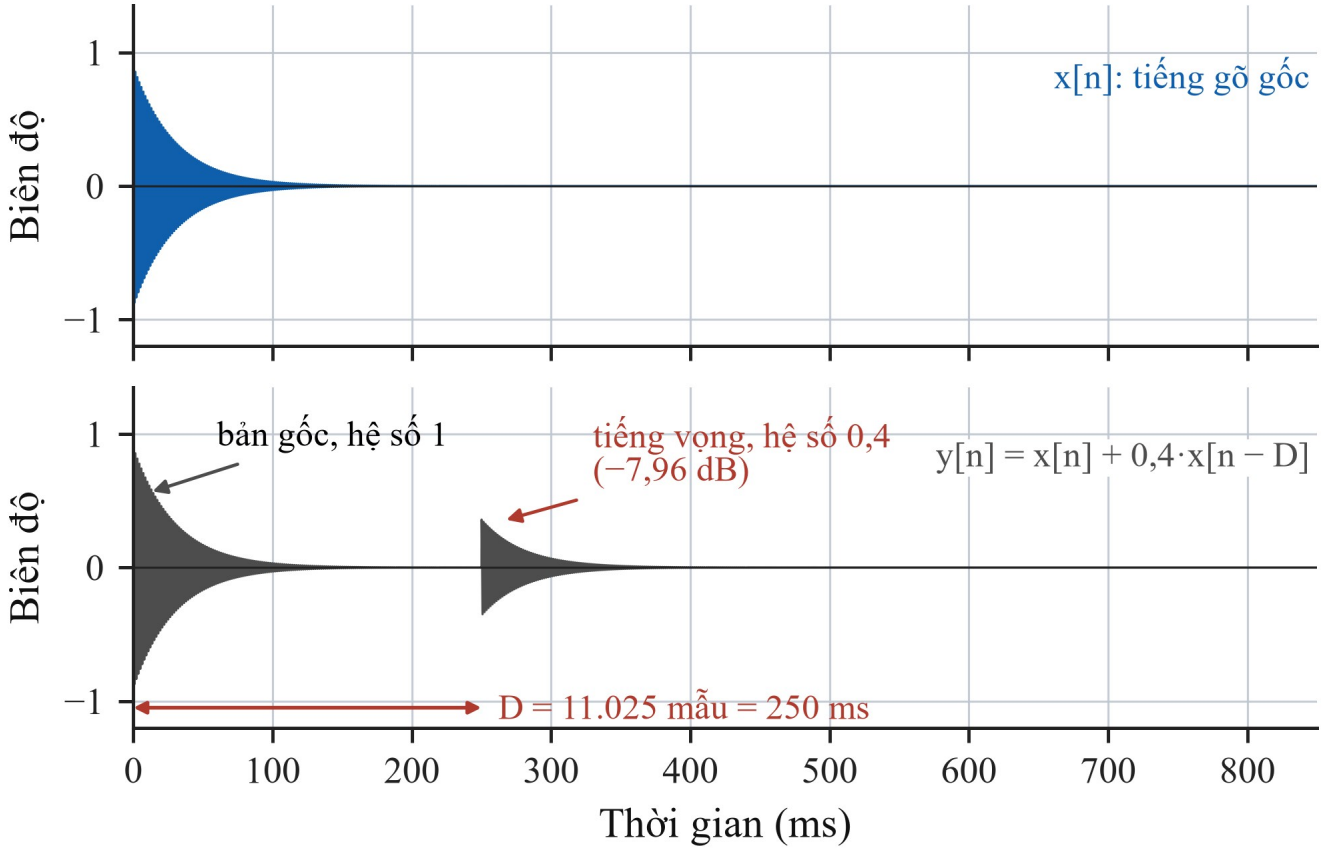
- Mỗi lần gọi hàm trễ phải quyết định có giữ nguyên độ dài mảng hay không.
- Giữ độ dài: mảng khớp các mảng khác, nhưng mất D mẫu ở cuối.
- Để dài ra: không mất mẫu nào, nhưng phép cộng sau đó phải đệm mảng ngắn cho bằng.



Hình 6.12 — Trễ 80 ms, giữ nguyên độ dài và không giữ nguyên độ dài

Trễ, giảm biên độ, cộng lại

- Hệ số 0,4 tương ứng −7,96 dB nên tiếng vọng nhỏ hơn tiếng gốc.
- Không có khái niệm mới so với chương 5: một phép trễ và một phép trộn.
- Đưa đầu ra vòng ngược về đầu vào thì thành bộ vọng lặp: hạt nhân của chương 8.



Hình 6.13 — Một tiếng gõ và bản có thêm tiếng vọng trễ 250 ms, hệ số 0,4

Tiếng vọng đơn

```
def echo_don(x, ms=250.0, g=0.4, sr=44100):
    """Một tiếng vọng duy nhất."""
    tre = tre_ms(x, ms, sr)          # dài hơn x đúng D mẫu
    return tron(x, g * tre)          # tron() tự đệm cho bằng
```

Phép trễ là viên gạch của bộ lọc (chương 8) và hiệu ứng (chương 10): cần viết gọn và kiểm thử ngay.

Từ quãng đường sang độ trễ

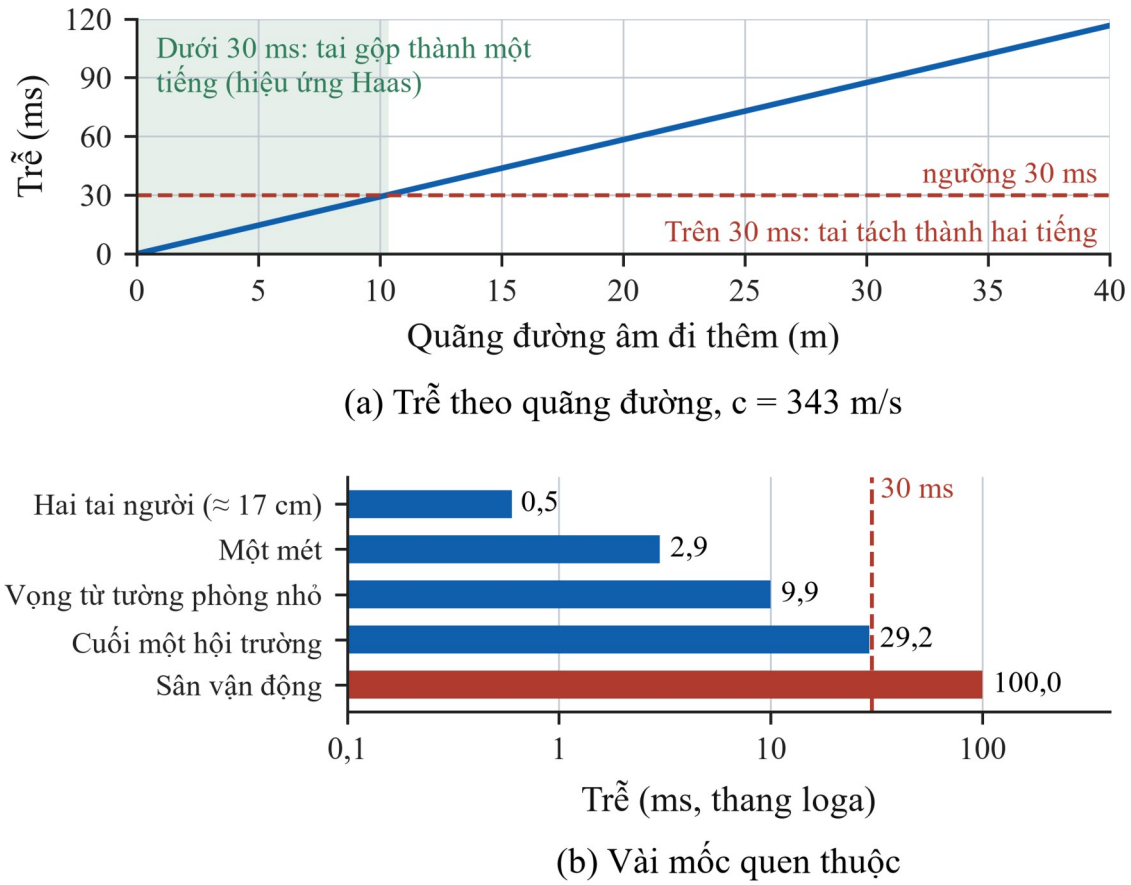
$$t_{\text{ms}} = \frac{d}{c} \cdot 1000, \qquad c = 343 \text{ m/s}$$

(6.5)

- d là quãng đường (m), c là tốc độ âm thanh (khoảng 343 m/s), t_ms là độ trễ (ms).
- Một mét tương ứng 2,9 ms; mười mét tương ứng 29,2 ms.

Ngưỡng 30 ms của hiệu ứng Haas

- Dưới 30 ms, tai gộp tiếng trực tiếp và tiếng trễ thành một tiếng, chỉ thấy "rộng hơn" hoặc "lệch".
- Trên 30 ms, tai tách thành hai tiếng và gọi là tiếng vọng.
- Hai tai cách nhau khoảng 17 cm, tương ứng 0,5 ms: độ trễ lớn nhất để định hướng ngang.
- Trễ 20 ms ứng 6,9 m (một tiếng); trễ 100 ms ứng 34,3 m (tiếng riêng).



Hình 6.14 — Từ khoảng cách sang độ trễ, và vài mốc quen thuộc

Nội suy tuyến tính

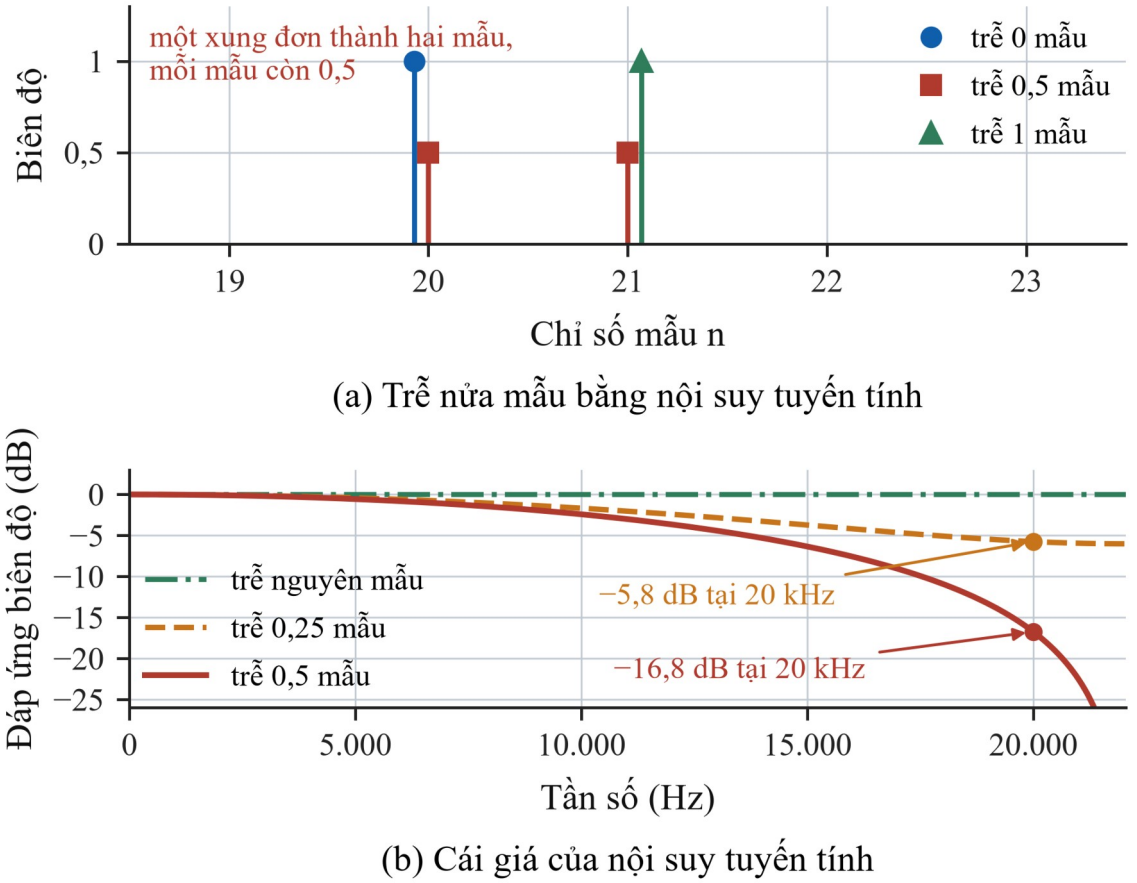
$$y[n] = (1-\alpha)x[n-D] + \alpha x[n-D-1], \qquad 0 \le \alpha < 1$$

(6.6)

- Trễ nguyên mẫu chỉ dời chỗ nên không mất gì.
- Trễ lẻ mẫu phải đoán giá trị giữa hai mẫu; cách rẻ nhất là nội suy tuyến tính.
- Alpha là phần lẻ của độ trễ, từ 0 đến 1; alpha bằng 0 thì trở về trễ nguyên mẫu.

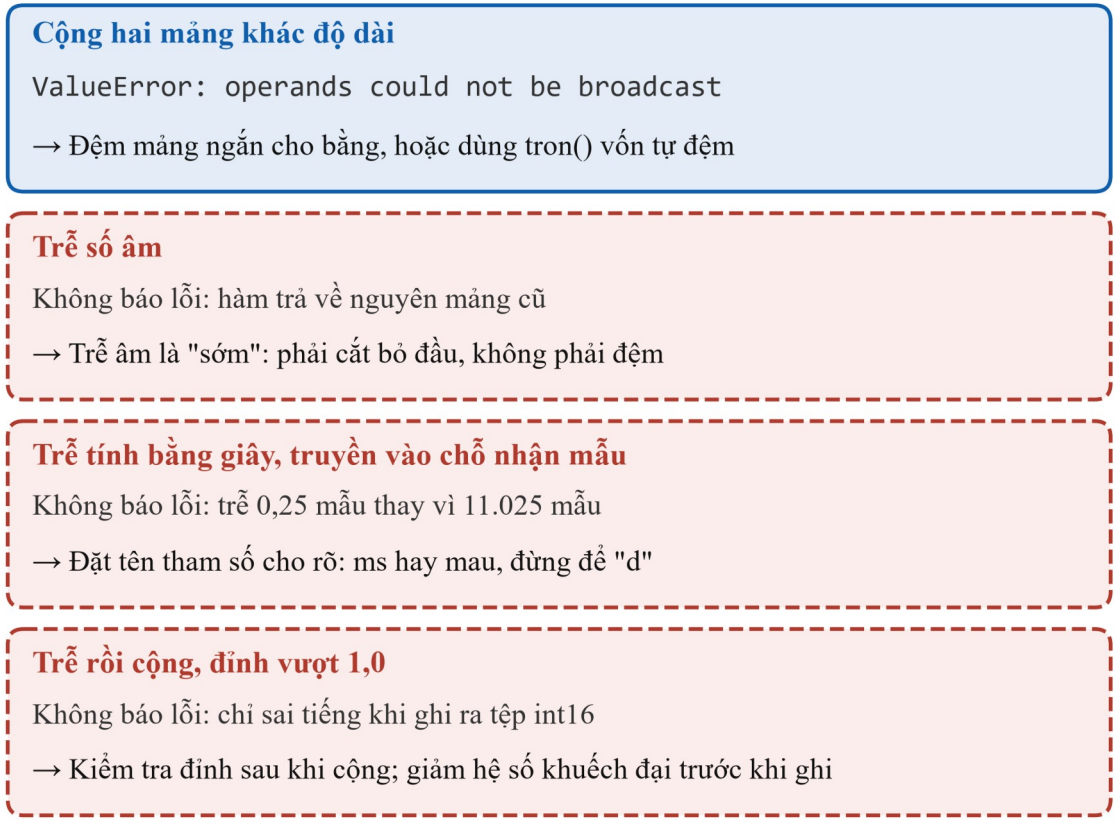
Cái giá: một bộ lọc thông thấp

- Xung đơn bị trải thành hai mẫu, mỗi mẫu còn một nửa.
- Trên trục tần số đó là bộ lọc thông thấp: với trễ nửa mẫu, đáp ứng ở 20 kHz tụt 16,8 dB.
- Bộ trễ chất lượng cao dùng nội suy bậc cao hơn; học phần này chỉ cần tránh trễ lẻ mẫu khi không cần.



Hình 6.15 — Trễ nửa mẫu bằng nội suy tuyến tính và đáp ứng biên độ của nó

Bốn lỗi thường gặp quanh phép trễ và cách chặn



Hình 6.16 — Bốn lỗi thường gặp quanh phép trễ và cách chặn

Trễ, có chặn các lỗi

```
def tre_mau_so(x, n, giu_do_dai=False):
    """Đẩy tín hiệu muộn đi n mẫu."""
    a = np.asarray(x, dtype=float)
    n = int(n)
    if n < 0:
        raise ValueError("trễ âm — dùng hàm som() nếu cần cắt đầu")
    if n == 0:
        return a
    pad = (n, 0) if a.ndim == 1 else ((n, 0), (0, 0))
    ra = np.pad(a, pad)
    return ra[:a.shape[0]] if giu_do_dai else ra

def tre_ms(x, ms, sr=44100, giu_do_dai=False):
    """Tên tham số là ms — không nhận giây, không nhận mẫu."""
    return tre_mau_so(x, int(round(ms / 1000 * sr)), giu_do_dai)
```

Ném ngoại lệ khi trễ âm thay vì im lặng trả mảng cũ; tên tham số "ms" nói rõ đơn vị.

Quy đổi giữa mili giây, số mẫu và khoảng cách

Trễ	Ở 44,1 kHz	Ở 96 kHz	Quãng đường	Tai nghe thấy gì
0,5 ms	22 mẫu	48 mẫu	17 cm	Định hướng trái phải
1 ms	44 mẫu	96 mẫu	34 cm	Vẫn là một tiếng
5 ms	220 mẫu	480 mẫu	1,7 m	Độ dài fade mặc định
20 ms	882 mẫu	1 920 mẫu	6,9 m	Một tiếng, nghe rộng ra
30 ms	1 323 mẫu	2 880 mẫu	10,3 m	Ngưỡng tách thành hai tiếng
100 ms	4 410 mẫu	9 600 mẫu	34,3 m	Rõ là tiếng vọng riêng
250 ms	11 025 mẫu	24 000 mẫu	85,8 m	Tiếng vọng của mục 6.4.2

Không viết cứng số mẫu: 5 ms là 220 mẫu ở 44,1 kHz nhưng 480 mẫu ở 96 kHz.

MỤC 6.5

Đảo thời gian

- Đảo thời gian là phép đơn giản nhất của chương: đọc mảng theo thứ tự ngược.
- Phổ biên độ không đổi, chỉ pha thay đổi, nhưng tai nghe khác hẳn.
- Chú ý điểm dễ sai: x[::-1] chỉ là một view.

Đảo thời gian và phổ

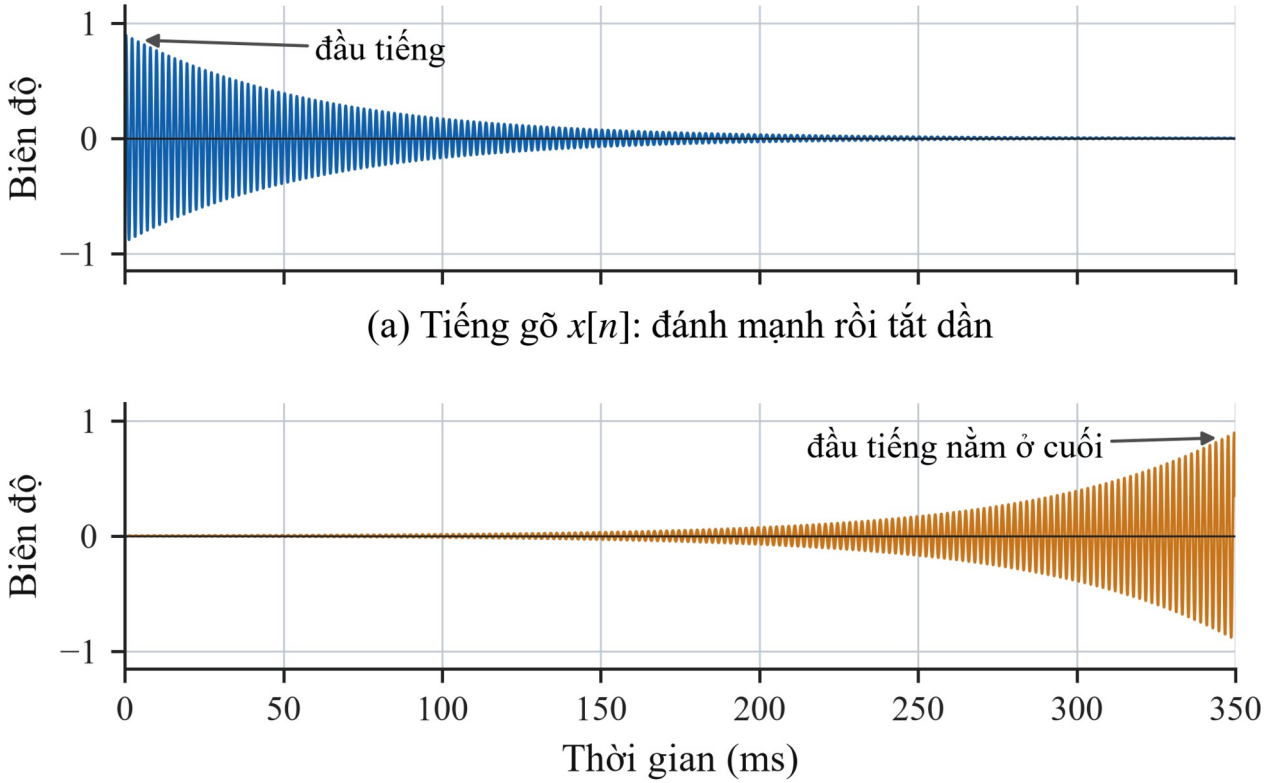
$$y[n] = x[N-1-n]; \qquad |Y[k]| = |X[k]|$$

(6.7)

- Trong numpy, đảo thời gian là x[::-1].
- Dòng thứ hai: phổ biên độ của tín hiệu đảo bằng đúng phổ biên độ của tín hiệu gốc.
- Chỉ có pha thay đổi.

Một tiếng gõ và bản đảo của nó

- Tiếng gõ gốc đánh mạnh rồi tắt dần: tai nhận ra vật bị gõ.
- Bản đảo lớn dần rồi dứt đột ngột: cảm giác "âm thanh chạy ngược", không có trong tự nhiên.



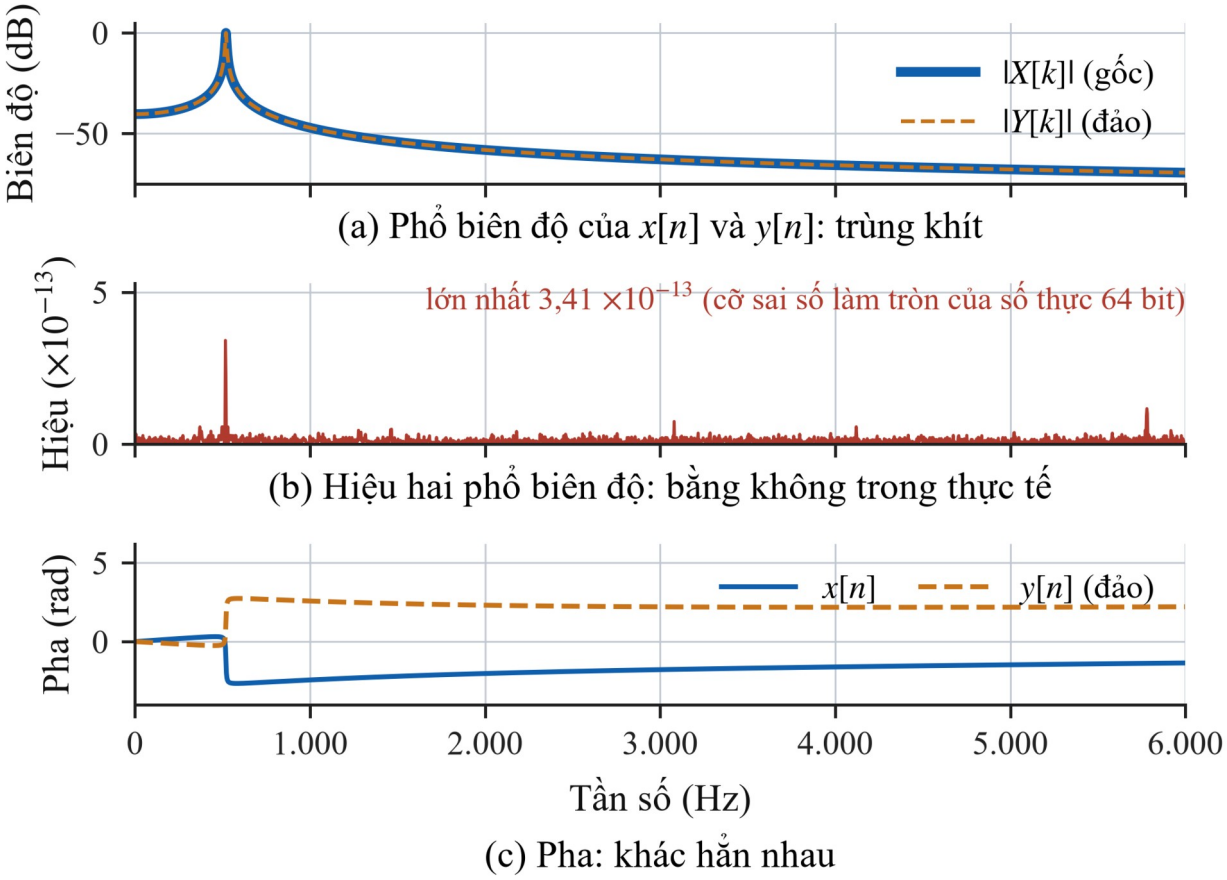
(a) Tiếng gõ $x[n]$: đánh mạnh rồi tắt dần

(b) Bản đảo $y[n] = x[N-1-n]$: lớn dần rồi dứt đột ngột

Hình 6.17 — Một tiếng gõ và bản đảo của nó

Hai tín hiệu khác nhau, phổ biên độ trùng khít

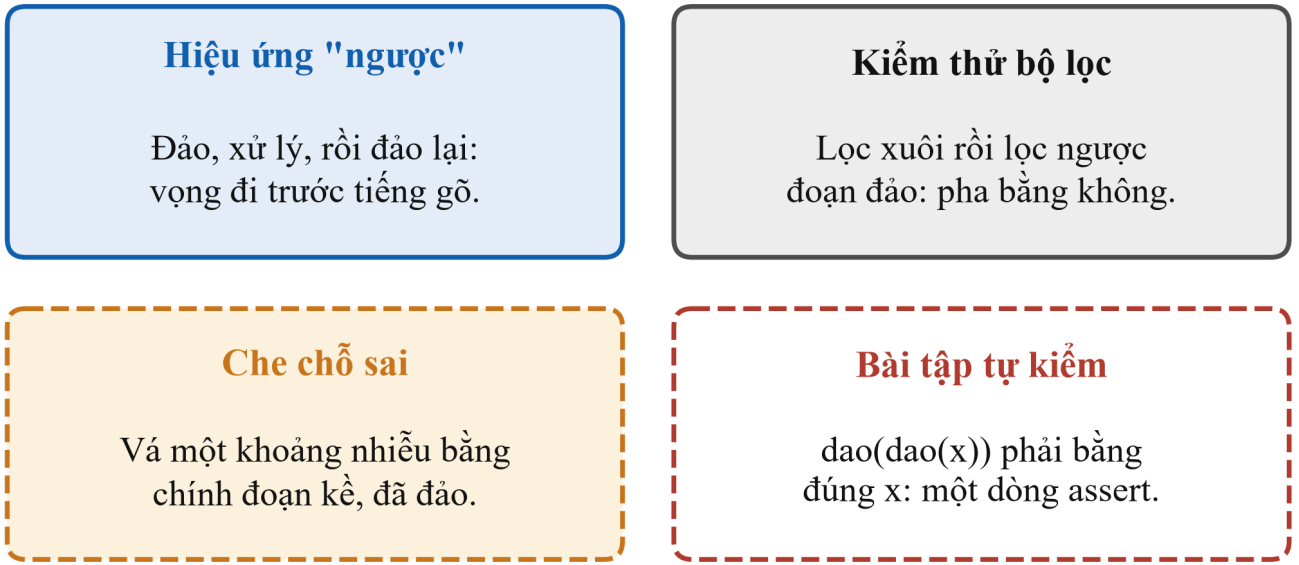
- Mục 4.5 khẳng định phổ biên độ không mô tả đủ một tín hiệu; đảo thời gian là ví dụ thứ ba.
- Hiệu lớn nhất giữa hai phổ biên độ cỡ 10 mũ trừ 13, bằng sai số làm tròn của số thực 64 bit.
- Pha thì khác hẳn; phổ đồ mất pha trong từng khung nên không thay được việc nghe lại.



Hình 6.18 — Phổ biên độ, hiệu hai phổ, và pha của tín hiệu gốc và tín hiệu đảo

Bốn chỗ dùng thật của phép đảo

- Lọc, đảo, lọc lần nữa bằng cùng bộ lọc rồi đảo lại cho bộ lọc pha bằng không.
- Không làm lệch thời điểm của thành phần tần số nào; trong scipy là filtfilt, dùng ở chương 8.



Hình 6.19 — Bốn chỗ dùng thật của phép đảo thời gian

Đảo thời gian và điểm dễ sai .copy()

```
def dao_thoi_gian(x):
    """Đảo trục thời gian. Mảng nhiều kênh chỉ đảo trục 0."""
    a = np.asarray(x, dtype=float)
    return a[::-1].copy() if a.ndim == 1 else a[::-1, :].copy()

# phép kiểm bắt buộc — đảo hai lần phải về đúng bản gốc:
assert np.array_equal(dao_thoi_gian(dao_thoi_gian(x)), x)
```

x[::-1] là view với bước nhảy âm; một số thư viện ghi tệp đòi mảng liên tục nên báo lỗi hoặc ghi dữ liệu sai.

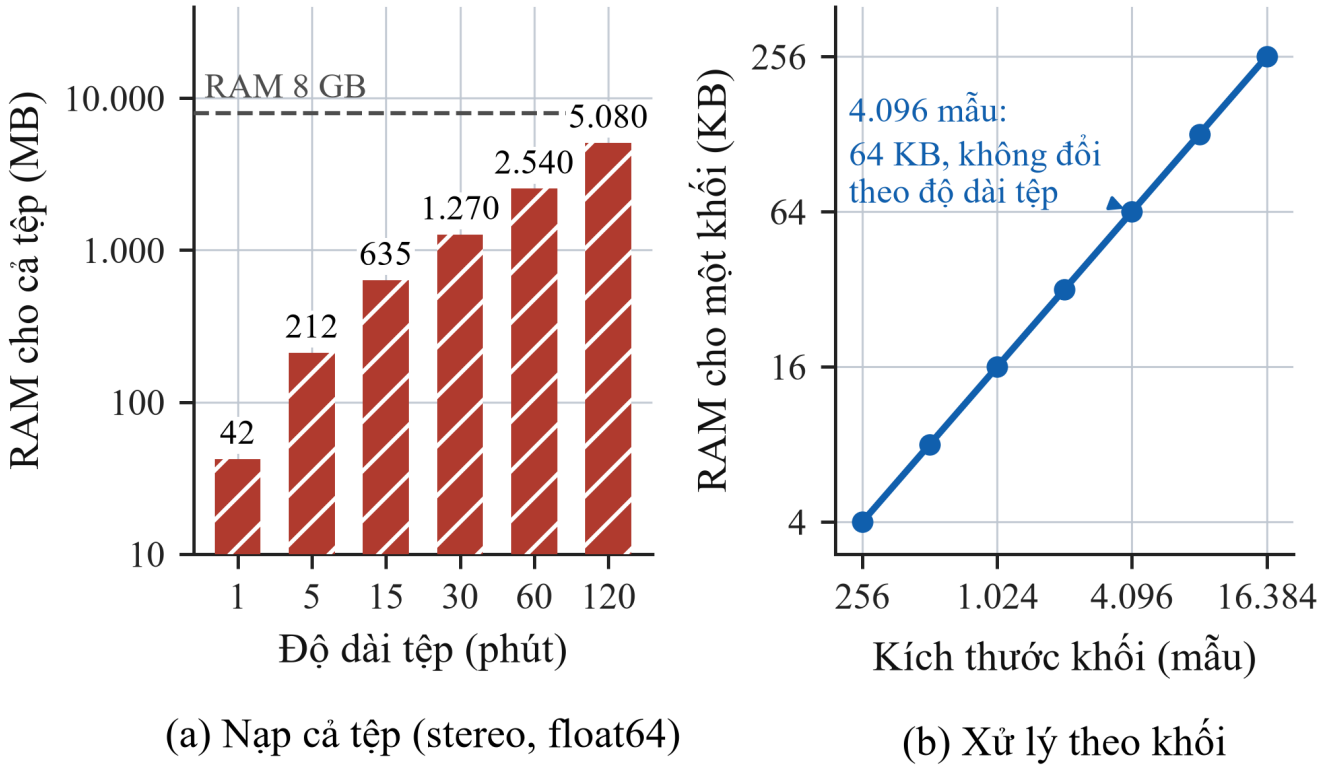
MỤC 6.6

Xử lý theo khối cho tệp dài

- Tệp dài không nạp hết vào bộ nhớ: đọc một khối, xử lý, ghi, rồi thả khối đi.
- Phép không nhớ chia khối thoải mái; phép có trạng thái phải mang trạng thái qua khối.
- Kích thước khối là một sự đánh đổi giữa tốc độ và độ trễ.

RAM cho cả tệp so với một khối

- Bản ghi hai tiếng, stereo, số thực 64 bit chiếm 5,1 GB; máy 8 GB có thể treo ngay khi đọc.
- Chia khối thì bộ nhớ không phụ thuộc độ dài tệp.
- Khối 4 096 mẫu stereo chiếm 64 KB, dù tệp dài ba phút hay ba tiếng.



Hình 6.20 — RAM cần cho cả tệp so với RAM cần cho một khối

Số khối và bộ nhớ một khối

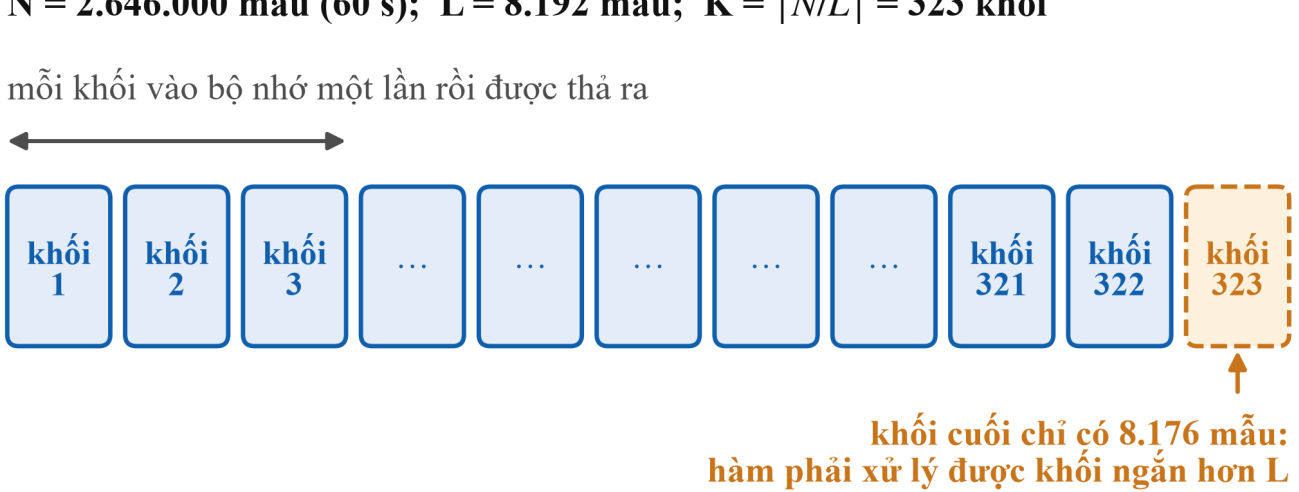
$$K = \left\lceil \frac{N}{L} \right\rceil ; \qquad R_{\text{khối}} = L \cdot N_{\text{kênh}} \cdot 8 \quad [\text{byte}]$$

(6.8)

- K là số khối, N là tổng số mẫu, L là kích thước khối; làm tròn lên vì khối cuối thường không đầy.
- R_khối là bộ nhớ một khối, N_ch là số kênh, 8 byte cho mỗi mẫu số thực 64 bit.

Khối cuối cùng ngắn hơn L

- Với N = 2 646 000 và L = 8 192, khối thứ 323 chỉ có 8 176 mẫu.
- Mọi hàm xử lý khối phải chạy đúng với khối ngắn hơn L.
- Kiểm nhanh: chạy thử với tệp có độ dài không chia hết cho L.



Vòng lặp: đọc khối → xử lý khối → ghi khối → quên khối.

Hình 6.21 — Một tệp 60 giây chia thành 323 khối 8 192 mẫu

Chia khối và xử lý từng khối

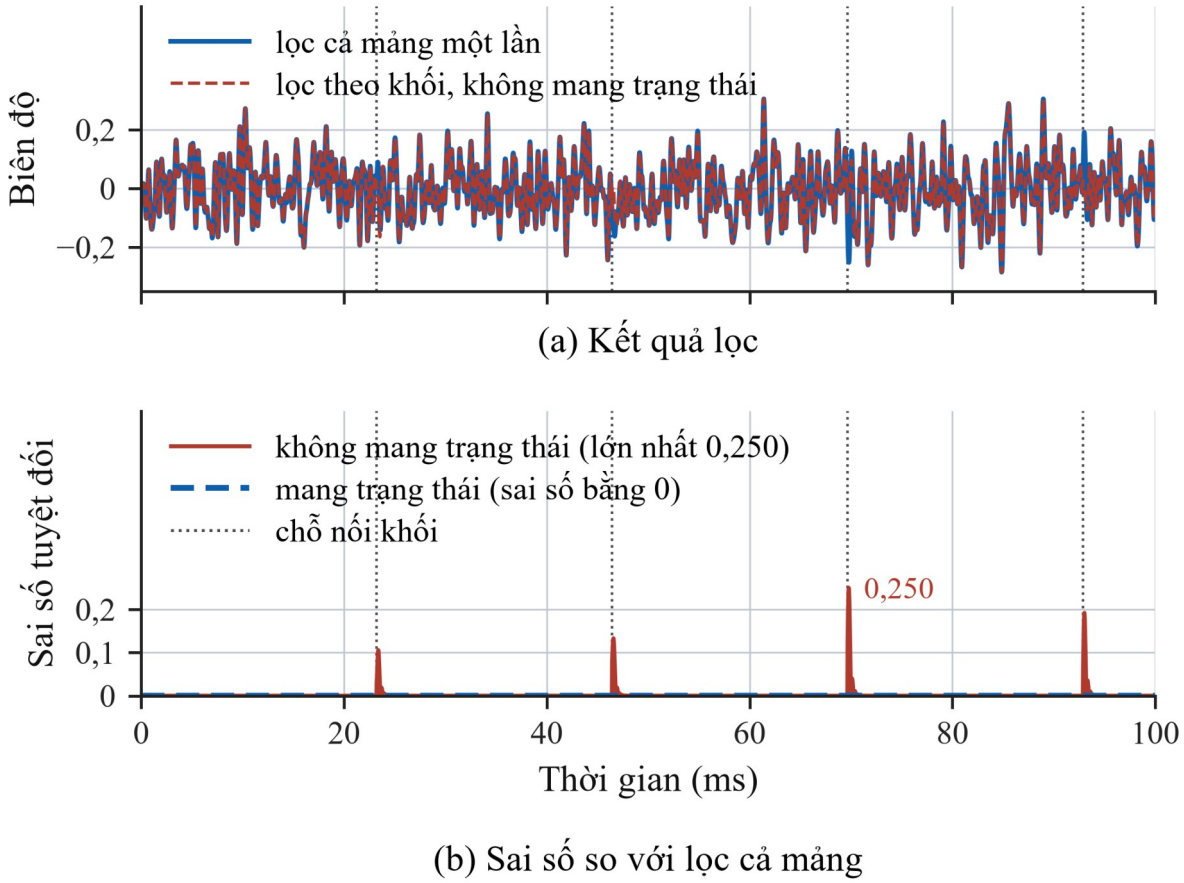
```
def chia_khoi(x, khoi):
    """Cắt mảng thành danh sách khối; khối cuối có thể ngắn hơn."""
    a = np.asarray(x, dtype=float)
    k = int(khoi)
    return [a[i:i + k] for i in range(0, a.shape[0], k)]

def xu_ly_theo_khoi(x, khoi, ham):
    """ĐÚNG cho phép KHÔNG NHỚ. SAI cho phép có trạng thái."""
    return np.concatenate([ham(k) for k in chia_khoi(x, khoi)])

# phép kiểm: chia rồi ghép lại phải bằng đúng bản gốc
assert np.array_equal(
    np.concatenate(chia_khoi(x, 8192)), x)
```

Lọc theo khối có và không mang trạng thái

- Phép không nhớ: kết quả của một mẫu chỉ phụ thuộc chính mẫu ấy, như nhân hệ số.
- Bộ lọc có trạng thái; gọi lại từ đầu ở mỗi khối là khởi động lại từ số không.
- Nhiễu trắng qua Butterworth bậc bốn, khối 1 024 mẫu: sai số lớn nhất 0,250 khi không mang trạng thái.
- Nghe ra tiếng lách tách đều đặn cứ 23 ms; mang trạng thái đúng thì sai số bằng không.



Hình 6.22 — Lọc theo khối có mang và không mang trạng thái

Mang trạng thái qua các khối

$$(y_k, z_k) = \text{lọc}(b, a, x_k, z_{k-1})$$

(6.9)

- Ghi nhớ trạng thái sau mỗi khối và truyền vào lần gọi sau.
- Trong scipy, lfilter nhận và trả trạng thái qua tham số zi.
- Trong công thức, z_k là trạng thái sau khối thứ k.

Lọc theo khối, có mang trạng thái

```
from scipy import signal as sg

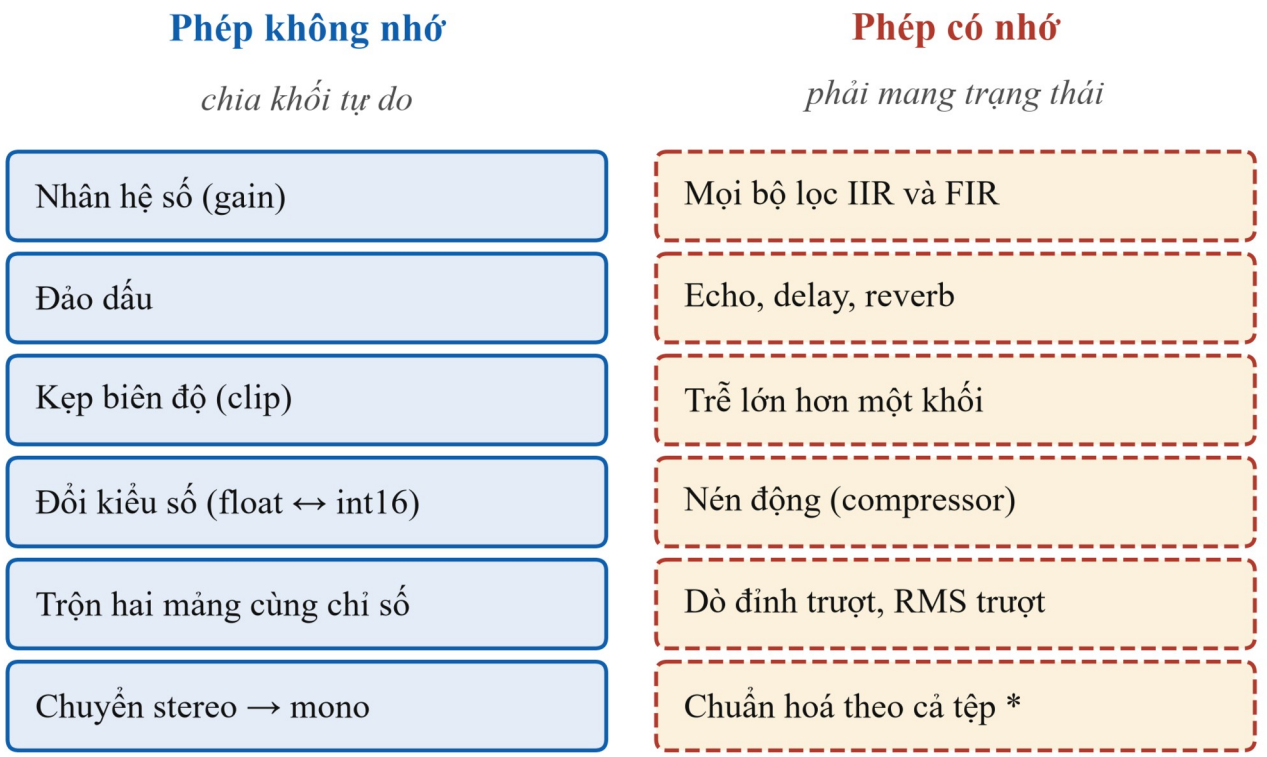
def xu_ly_theo_khoi_co_trang_thai(x, khoi, b, a):
    """Lọc theo khối, MANG trạng thái từ khối trước sang khối sau."""
    zi = sg.lfilter_zi(b, a) * 0.0     # bắt đầu từ trạng thái nghỉ
    ra = []
    for k in chia_khoi(x, khoi):
        y, zi = sg.lfilter(b, a, k, zi=zi)   # <- nhận và trả zi
        ra.append(y)
    return np.concatenate(ra)

# phép kiểm bắt buộc: phải bằng ĐÚNG kết quả lọc cả mảng
dung = sg.lfilter(b, a, x)
mang = xu_ly_theo_khoi_co_trang_thai(x, 1024, b, a)
assert np.max(np.abs(dung - mang)) == 0.0
```

Phép kiểm "so với cách chậm" bắt lỗi trạng thái bằng một dấu bằng tuyệt đối; viết ba dòng này trước khi viết hàm.

Phân loại phép có trạng thái hay không

- Tự hỏi: kết quả của mẫu này có phụ thuộc mẫu nào khác không? Không thì chia khối thoải mái.
- Khó nhất là chuẩn hoá theo cả tệp: đỉnh chỉ biết sau khi đọc hết.
- Cách duy nhất là đọc hai lượt: lượt một đo đỉnh, lượt hai nhân hệ số; chương 7 gặp lại với LUFS (Loudness Units Full Scale – đơn vị độ to so với toàn thang).

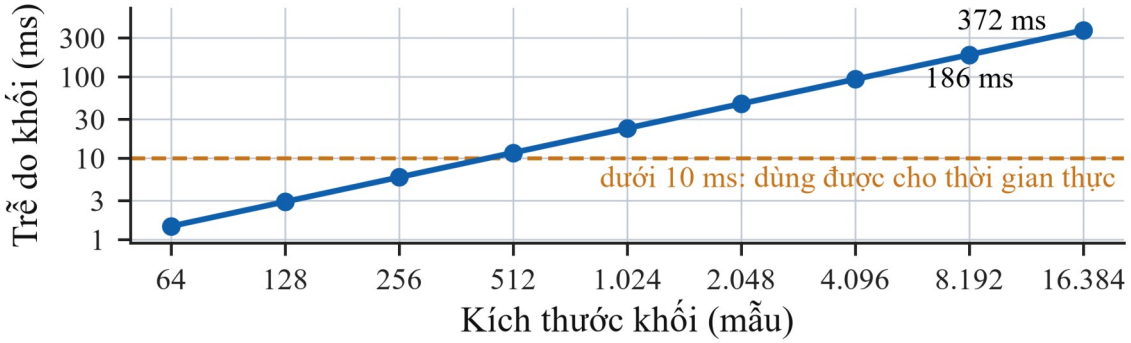


** Chuẩn hoá cần biết đỉnh của cả tệp, mà đỉnh ấy chỉ biết sau khi đã đọc hết; vì vậy phải đọc hai lượt: lượt một đo đỉnh, lượt hai nhân hệ số.*

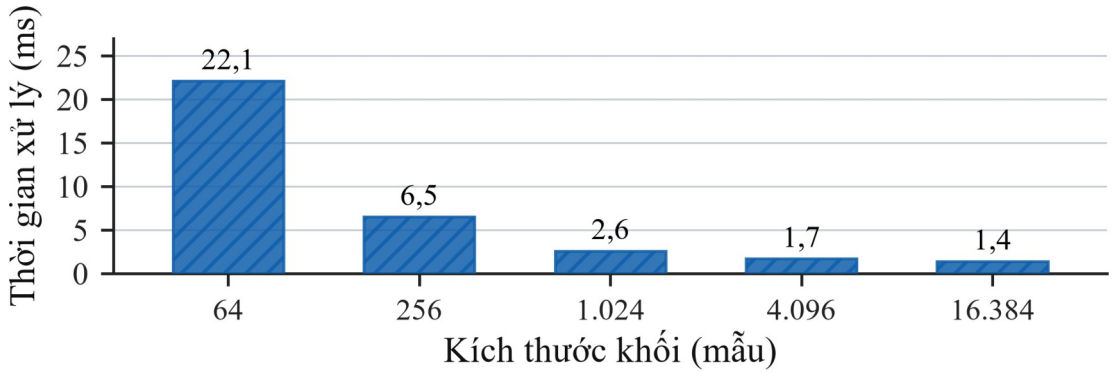
Hình 6.23 — Phân loại các phép theo có trạng thái hay không

Độ trễ và chi phí vòng lặp

- Khối càng lớn thì càng ít vòng lặp; từ khối 64 xuống 4 096, thời gian xử lý giảm khoảng mười lần.
- Khối càng lớn thì độ trễ càng nhiều: khối 8 192 mẫu đã là 186 ms.
- Số đo thời gian phụ thuộc máy; người học tự chạy lại, bậc độ lớn thì không đổi.



(a) Khối càng lớn, trễ càng nhiều



(b) Khối càng nhỏ, chi phí vòng lặp càng lớn

Hình 6.24 — Trễ và chi phí vòng lặp theo kích thước khối

Chọn kích thước khối theo mục đích

Mục đích	Kích thước nên dùng	Vì sao
Xử lý tệp ngoại tuyến	4 096 – 16 384	Không ai đợi trực tiếp, tối ưu tốc độ
Phát lại có xử lý	512 – 2 048	Trễ 12 – 46 ms, không nhận ra khi chỉ nghe
Thời gian thực có nhạc cụ	64 – 256	Trễ dưới 6 ms, người chơi không thấy "chậm tay"
Phân tích phổ theo khung	Bằng kích thước FFT	Khối trùng khung phân tích của chương 4

Xử lý tệp dài mà không nạp cả tệp

```
import soundfile as sf

L = 8192
with sf.SoundFile("vao.wav") as f_vao:
    with sf.SoundFile("ra.wav", "w",
                      samplerate=f_vao.samplerate,
                      channels=f_vao.channels,
                      subtype="PCM_24") as f_ra:
        zi = None
        for khoi in f_vao.blocks(blocksize=L, dtype="float64"):
            if zi is None:
                zi = np.zeros((max(len(a), len(b)) - 1,
                               f_vao.channels))
            y, zi = sg.lfilter(b, a, khoi, axis=0, zi=zi)
            f_ra.write(y)

# Bộ nhớ dùng: một khối, bất kể tệp dài bao nhiêu.
```

Khối cuối ngắn hơn L; zi có một cột mỗi kênh; tệp trung gian ghi PCM_24, không phải PCM_16.

MỤC 6.7

Đóng gói thành mô-đun dùng lại được

- Hai chương đã sinh khoảng mười tám hàm; bốn chương tới sinh thêm chừng ấy.
- Bốn quy tắc cho một hàm dùng lại được, và bốn loại phép kiểm.
- Kết quả: mô-đun am gồm bốn tệp, dùng cho cả bài tập lớn.

Bốn bản sao so với một mô-đun

- Mã sao chép: sửa một lỗi phải sửa ở bốn chỗ; bỏ sót một chỗ thì hai tệp kết quả khác nhau mà không biết vì sao.
- Đến chương 12 sẽ không ai còn biết bản nào là bản đúng.
- Mô-đun được nhiều kịch bản gọi: sửa một nơi, mọi nơi đều đúng.

Trước: bốn bản sao của cùng một đoạn mã

```
lam_bai/
    thu1.py           ← sao chép 40 dòng
    thu2.py           ← sao chép, sửa 1 số
    thu2_final.py     ← sao chép lần nữa
    thu2_final_v2.py  ← khác nhau ở đâu?

Sửa một lỗi → phải sửa ở BỐN chỗ. Bỏ sót một chỗ
→ hai kết quả khác nhau mà không ai biết vì sao.
```

Sau: một mô-đun, nhiều kịch bản gọi nó

```
am/
    __init__.py
    io.py          doc_wav, ghi_wav
    bien_do.py     gain_db, chuan_hoa_dinh
    thoi_gian.py   fade, crossfade, tre
    khoi.py        chia_khoi, xu_ly_theo_khoi
    test_am.py     các phép kiểm
lam_bai/thu1.py: import am  ← 6 dòng
Sửa một lỗi → sửa ĐÚNG MỘT chỗ.
```

Hình 6.25 — Bốn bản sao của một đoạn mã, và một mô-đun được nhiều kịch bản gọi

Hàm khó dùng lại và hàm dùng lại được

- Nhận mảng, trả mảng: hàm tính toán không đọc và không ghi đĩa.
- Mọi con số là tham số có giá trị mặc định; số 220 và số 44100 phải đi cùng nhau.
- Không sửa mảng đầu vào tại chỗ: dùng np.array(x, copy=True) ở đầu hàm.
- Có một dòng docstring nói hàm làm gì.

Hàm khó dùng lại

```
def xu_ly():
    x, sr = doc_wav("D:/bai/giong.wav")
    x = x * 0.5
    x[:220] *= np.linspace(0, 1, 220)
    ghi_wav("D:/bai/ra.wav", x, 44100)
    print("xong")
```

· Đường dẫn cứng → máy khác chạy sai
· 220 và 44100 rời nhau → đổi sr là sai
· Vừa đọc, tính, ghi, in → khó kiểm thử

Hàm dùng lại được

```
def fade_vao(x, ms=5.0, sr=44100):
    """Nhân đường bao đi lên vào đầu đoạn."""
    a = np.array(x, dtype=float, copy=True)
    n = min(int(ms / 1000 * sr), len(a))
    a[:n] *= duong_bao_fade(n)
    return a
```

· Nhận mảng, trả mảng, không sửa tại chỗ
· ms và sr là tham số, không phải hằng
· Kiểm thử được bằng một dòng assert

Hình 6.26 — Một hàm khó dùng lại và một hàm dùng lại được

Bốn loại phép kiểm dùng cho mô-đun của lớp

Tính chất bất biến

```
dao(dao(x)) == x
len(fade(x)) == len(x)
chia rồi ghép == x
```

Trường hợp biên

```
mảng rỗng
mảng một mẫu
fade dài hơn cả đoạn
```

Giá trị đã biết

```
gain_db(x, 0) == x
gain_db(x, -6.02)
→ đỉnh còn một nửa
```

So với cách chậm

```
lọc theo khối
== lọc cả mảng
(sai số bằng 0)
```

Bốn loại phép kiểm viết được chỉ bằng assert, không cần thư viện nào.

Quy ước của lớp: mỗi hàm mới thêm vào am/ phải kèm ít nhất một phép kiểm trước khi được dùng trong bài tập lớn.

Hình 6.27 — Bốn loại phép kiểm dùng cho mô-đun của lớp

6.7.3 · BẢNG 6.8

Bốn loại phép kiểm và ví dụ

Loại	Ý tưởng	Ví dụ trong chương 6
Tính chất bất biến	Quan hệ đúng với mọi đầu vào	dao(dao(x)) bằng x; fade không đổi độ dài
Trường hợp biên	Đầu vào ở rìa miền xác định	Mảng rỗng; một mẫu; fade dài hơn đoạn
Giá trị đã biết	Đáp số tính được bằng tay	Nửa cosin: đầu bằng 0, cuối bằng 1
So với cách chậm	Hai cách cùng việc, so kết quả	Lọc khối có trạng thái bằng lọc cả mảng

Quy ước của lớp: mỗi hàm mới kèm ít nhất một phép kiểm; test_am.py phải chạy sạch trước mỗi lần nộp.

test_am.py tối giản cho mô-đun am

```
# am/test_am.py — chạy bằng: python -m am.test_am
import numpy as np
from . import thoi_gian as tg

def kiem():
    n = 0
    x = np.random.default_rng(0).normal(size=4410) * 0.3

    # 1. tính chất bất biến
    assert np.array_equal(tg.dao(tg.dao(x)), x);          n += 1
    assert len(tg.fade_hai_dau(x)) == len(x);             n += 1

    # 2. trường hợp biên
    assert len(tg.fade_vao(np.zeros(0))) == 0;            n += 1
    assert len(tg.fade_vao(x[:3], ms=50.0)) == 3;         n += 1

    # 3. giá trị đã biết
    w = tg.duong_bao_fade(220)
    assert w[0] == 0.0 and w[-1] == 1.0;                  n += 1

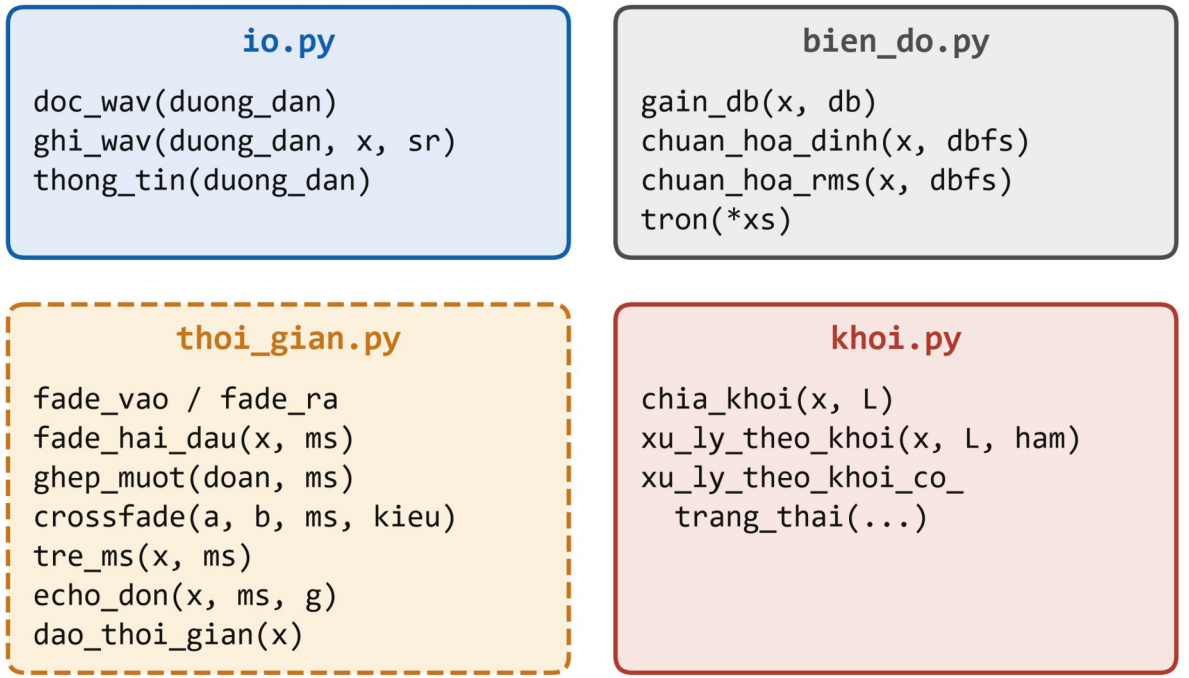
    # 4. so với cách chậm (mục 6.6.3)
    print(f"{n}/{n} phép kiểm đạt")

if __name__ == "__main__":
    kiem()
```

(rút gọn)

Bốn tệp và mười tám hàm của mô-đun am

- Bốn tệp: io.py, bien_do.py, thoi_gian.py, khoi.py, cùng __init__.py và test_am.py.
- __init__.py gom các tên hay dùng lên một tầng để kịch bản chỉ cần import am.



Mười tám hàm, đủ để làm giai đoạn 1 của bài tập lớn mà không phải viết thêm dòng xử lý tín hiệu nào.

Các chương sau chỉ thêm tệp mới; cấu trúc không đổi.

Hình 6.28 — Bốn tệp và mười tám hàm của mô-đun am

Bố cục thư mục và kịch bản hoàn chỉnh

```
am/
    __init__.py   io.py   bien_do.py   thoi_gian.py
    khoi.py       test_am.py

# am/__init__.py – gom các tên hay dùng lên một tầng
from .io        import doc_wav, ghi_wav, thong_tin
from .bien_do   import gain_db, chuan_hoa_dinh, chuan_hoa_rms, tron
from .thoi_gian import (duong_bao_fade, fade_vao, fade_ra,
                        fade_hai_dau, ghep_muot, crossfade,
                        tre_ms, echo_don, dao_thoi_gian)
from .khoi      import chia_khoi, xu_ly_theo_khoi

# kịch bản chỉ còn vài dòng:
import am

doan = [am.doc_wav(f"tu_lieu/{t}.wav")[0] for t in ("a", "b", "c")]
y = am.ghep_muot(doan, ms=5.0)
y = am.chuan_hoa_dinh(y, dinh_dbfs=-1.0)
am.ghi_wav("ket_qua.wav", y, 44100)
```

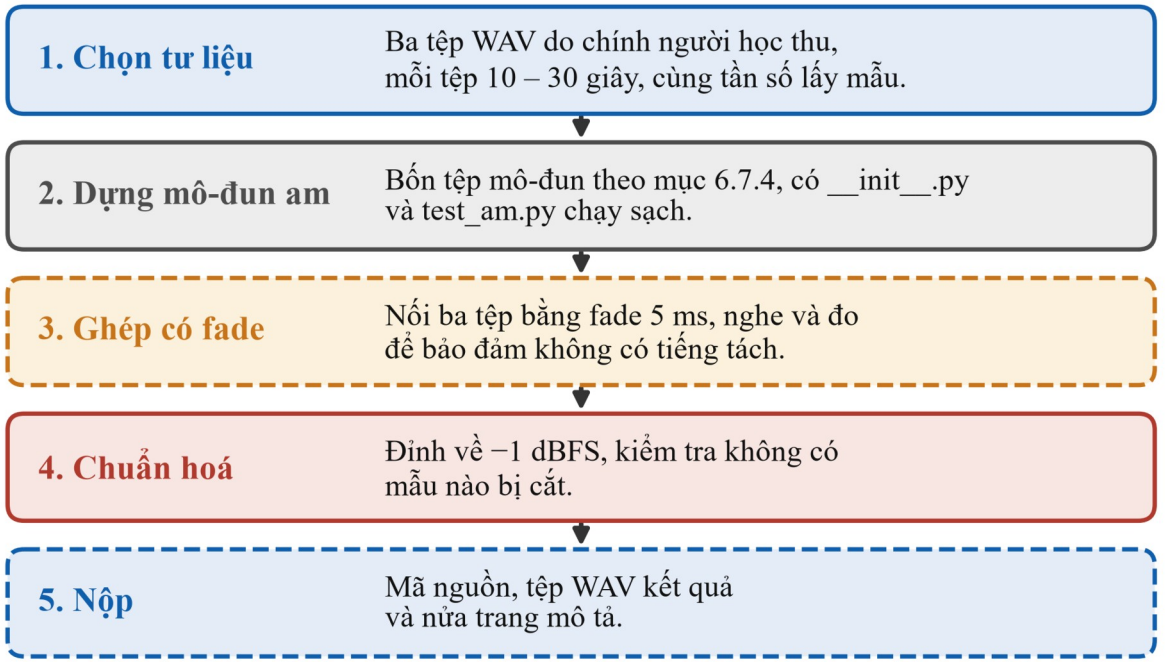
(rút gọn)

MỤC 6.8

Mốc bài tập lớn — giai đoạn 1

- Giai đoạn 1 nộp trước chương 8.
- Dùng lại đúng các hàm đã viết ở chương 5 và chương 6; không cần kiến thức mới.
- Chấm bốn tiêu chí, tổng 10 điểm.

Năm bước của giai đoạn 1



Chấm theo bốn tiêu chí: chạy được trên máy khác · không có tiếng tách ở chỗ nối · đỉnh đúng −1 dBFS · mô-đun tách khỏi kịch bản.

Toàn bộ giai đoạn 1 dùng lại các hàm đã viết ở chương 5 và chương 6.

Hình 6.29 — Năm bước của giai đoạn 1

Nội dung phải nộp

Hạng mục	Yêu cầu cụ thể
Tư liệu	Ba tệp WAV tự thu, 10 – 30 giây, cùng tần số lấy mẫu và số kênh
Mô-đun am	Bốn tệp io, bien_do, thoi_gian, khoi cùng __init__.py; kèm test_am.py chạy sạch
Kịch bản	lam_bai.py không quá 30 dòng, chỉ gọi mô-đun, không có dòng xử lý tín hiệu
Kết quả	Một tệp WAV: ba đoạn nối bằng ghep_muot, fade 5 ms, chuẩn hoá đỉnh về −1 dBFS
Mô tả	Nửa trang: nguồn tư liệu, tham số đã chọn, cách đã kiểm chỗ nối

Bốn tiêu chí chấm giai đoạn 1

Tiêu chí	Điểm	Cách kiểm
Chạy được trên máy khác	3	Giải nén vào thư mục mới, chạy lam_bai.py; không có đường dẫn tuyệt đối
Không có tiếng tách ở chỗ nối	3	Nghe; đo năng lượng trên 8 kHz quanh hai chỗ nối
Đỉnh đúng −1 dBFS	2	Sai lệch cho phép 0,1 dB; không mẫu nào chạm hoặc vượt 1,0
Mô-đun tách khỏi kịch bản	2	lam_bai.py không chứa numpy hay scipy; test_am.py chạy sạch

Lỗi hay mất điểm: đường dẫn tuyệt đối; chuẩn hoá trước khi ghép; ghi PCM_16 rồi đọc lại để xử lý tiếp.

MỤC 6.9

Thực hành, bài tập và tổng kết

- Ba bài thực hành, mỗi bài khoảng mười phút, trên tư liệu của chính người học.
- Bảy bài tập có đáp số bằng số.
- Sáu câu tổng kết; chuẩn bị ba tệp cho chương 7.

Nghe, đo và chứng minh

- Bài 1: ghép ba mảnh của giọng đọc, thẳng rồi bằng ghep_muot; in bước nhảy trước và sau tại hai chỗ nối.
- Bài 2: đo năng lượng trên 8 kHz của hai bản ghép; thử fade 0,25 · 1 · 5 · 20 ms và vẽ đồ thị cột.
- Bài 3: lọc nhiễu trắng cả mảng và theo khối 1 024 mẫu không mang trạng thái; in sai số lớn nhất rồi sửa lại.

Bài tập của chương 6 (đáp số)

Bài	Nội dung	Đáp số hoặc tiêu chí
1	Fade 5 ms ở 48 kHz và 96 kHz là bao nhiêu mẫu?	240 mẫu; 480 mẫu
2	Nối 20 đoạn 0,25 s bằng crossfade 50 ms; muốn đúng 5,000 s thì cắt mỗi đoạn bao nhiêu?	4,050 s; mỗi đoạn 0,2975 s
3	Hai loa cách 12 m và 4 m: chênh lệch trễ, nghe một hay hai tiếng?	23,3 ms; một tiếng, dưới 30 ms
4	Viết hàm som(x, ms) ngược với phép trễ, kèm phép kiểm	Hàm chạy, phép kiểm đạt
5	Tệp 44,1 kHz stereo dài 47 phút, chia khối 4 096: số khối và khối cuối?	30 362 khối; khối cuối 3 344 mẫu
6	Crossfade một đoạn với bản đảo của nó; vì sao nghe êm hơn ghép thẳng?	Hai bản cùng giá trị mẫu tại chỗ nối
7	Gói hàm của chương 5 và 6 thành mô-đun am, test_am.py ít nhất tám phép kiểm	Bước 2 của giai đoạn 1

Sáu câu tổng kết chương 6

Vì sao chỗ nối nghe "tách"?
Bước nhảy giữa hai mẫu sinh ra năng lượng cao tần.
Đo được: −44,9 dB → −128,6 dB sau fade 5 ms.

Fade và crossfade khác nhau ở đâu?
Fade giữ nguyên độ dài; crossfade rút ngắn đúng bằng phần chồng lấn.

Khi nào dùng đường bao công suất đều?
Khi hai đoạn là hai nguồn độc lập. Dùng nhầm thì mức ở giữa tụt hoặc vọt 3,01 dB.

Trễ 20 ms là bao nhiêu mẫu ở 44,1 kHz?
882 mẫu, tương ứng quãng đường 6,9 m. Dưới ngưỡng 30 ms nên vẫn nghe là một tiếng.

Đảo thời gian có đổi phổ biên độ không?
Không. Chỉ đổi pha — bằng chứng cho điều đã nói ở chương 4.

Vì sao lọc theo khối lại sai?
Vì bộ lọc có trạng thái. Quên mang thì lệch 0,250; mang đúng thì lệch bằng không.

Hình 6.30 — Sáu câu tổng kết chương 6

Từ −1 dBFS đến độ to cảm nhận

- Chương 7 mở mô-đun M5, bắt đầu từ chỗ giai đoạn 1 dừng: đỉnh đã về −1 dBFS (decibel so với mức toàn thang, Full Scale), vậy đã đủ chưa.
- Chuẩn bị ba tệp: nhạc pop hiện đại, nhạc cổ điển, giọng đọc; và một tệp có tiếng ồn nền rõ.
- Chuẩn hoá cả ba về −1 dBFS, nghe cùng mức loa và ghi lại ước lượng chênh lệch độ to bằng decibel.
- Đọc lại mục 1.5 về đường đẳng âm và mục 5.7 về đỉnh so với RMS.

Những điều cần nhớ

1. Fade là phép nhân với đường bao; nửa cosin 5 ms (220 mẫu ở 44,1 kHz) đưa bước nhảy ở chỗ nối về đúng bằng không.
2. Crossfade chồng lấn hai đoạn nên làm kết quả ngắn đi; nguồn độc lập dùng công suất đều, nguồn tương quan dùng nửa cosin.
3. Phép trễ đệm số không; đổi qua lại giữa mili giây, số mẫu và khoảng cách; trễ lẻ mẫu trả giá bằng một bộ lọc thông thấp.
4. Đảo thời gian giữ nguyên phổ biên độ nhưng đổi pha; tai nghe khác hẳn nên không thể chỉ nhìn phổ.
5. Xử lý theo khối giữ bộ nhớ không đổi; phép có trạng thái phải mang trạng thái, kiểm bằng so với lọc cả mảng.
6. Mô-đun am với bốn quy tắc và bốn loại phép kiểm là nền cho bài tập lớn và bốn chương sau.