MỤC LỤC

Chương 3. Số hoá âm thanh.....5
3.1 Vị trí của chương trong cuốn sách.....5
3.2 Lấy mẫu và định lý Nyquist.....7
3.3 Chồng phổ và bộ lọc chặn nó.....13
3.4 Lượng tử hoá, độ sâu bit và dither.....17
3.5 DBFS và cách máy tính lưu mẫu.....20
3.6 Dung lượng, định dạng và nén có tổn hao.....24
3.7 Codec nơ-ron.....29
3.8 Hướng dẫn thực hành.....33
3.9 Bài tập, thảo luận và tổng kết.....36

DANH MỤC HÌNH VẼ

Hình 3.1 Hai phép rời rạc hoá độc lập: lấy mẫu cắt trục thời gian, lượng tử hoá cắt trục biên độ.....6
Hình 3.2 Đường đi của chương 3: từ phép lấy mẫu tới định dạng tệp và codec nơ-ron. .6
Hình 3.3 Lấy mẫu: chỉ dãy chấm được lưu lại, đường liền nét thì không.....8
Hình 3.4 Trực giác hình học của định lý: mỗi chu kỳ cần tối thiểu hai điểm đo.....9
Hình 3.5 Cùng một sóng 5 Hz, ba mật độ lấy mẫu khác nhau.....9
Hình 3.6 Nối chấm bằng đoạn thẳng thì sai; dựng lại bằng sinc thì trùng khít.....10
Hình 3.7 Hàm sinc: bằng 1 tại mẫu của chính nó, bằng 0 tại mọi mẫu khác.....10
Hình 3.8 Tần số cao nhất biểu diễn được của từng chuẩn, đối chiếu với giới hạn nghe 12
Hình 3.9 Tăng tần số lấy mẫu được gì và không được gì.....12
Hình 3.10 Bản đồ gập: mọi tần số trên Nyquist đều rơi về một tần số trong vùng an toàn.....13
Hình 3.11 Kiểm thật trên tín hiệu: cùng một sóng 5.000 Hz, hai tần số lấy mẫu, hai kết quả.....14
Hình 3.12 Bánh xe quay ngược, vân moiré, cánh quạt dưới đèn nháy: cùng một hiện tượng.....15
Hình 3.13 Bộ lọc chống chồng phổ cắt phần trên Nyquist trước khi lấy mẫu.....15
Hình 3.14 Cùng một tín hiệu có thành phần siêu âm: lấy mẫu thẳng và lấy mẫu sau khi lọc.....16
Hình 3.15 Cùng một tín hiệu ở ba độ sâu bit: số mức càng nhiều, bậc thang càng mịn 17
Hình 3.16 Sai số lượng tử nhìn riêng ra: bị chặn trong nửa bước, nhưng có quy luật…18
Hình 3.17 Dải động theo độ sâu bit, và vùng dải động của phần lớn âm nhạc.....18
Hình 3.18 Phổ của riêng sai số: không dither thì có vạch nhọn, có dither thì phẳng.....19
Hình 3.19 Dither đúng một lần, ở đúng bước hạ bit cuối cùng.....20
Hình 3.20 Thang dBFS và quan hệ của nó với hai thang đã học.....21
Hình 3.21 Đổi số thực sang int16: dải không đối xứng và hiện tượng kẹp biên.....22
Hình 3.22 Hai cách lưu mẫu, và nguyên tắc cốt lõi khi chọn giữa chúng.....22
Hình 3.23 Hình dạng mảng của mono, stereo và đa kênh.....23
Hình 3.24 Dung lượng một phút ở các cấu hình khác nhau, và số giờ chứa được trong một ổ 64 GB.....24
Hình 3.25 Số bit còn lại cho mỗi mẫu, từ WAV không nén tới codec nơ-ron.....25
Hình 3.26 Ba nhóm định dạng và câu hỏi phân biệt chúng.....25
Hình 3.27 Ba điểm mù của tai và mức tiết kiệm mà mỗi điểm mang lại.....26
Hình 3.28 Năm bước bên trong một bộ mã hoá MP3 hoặc AAC.....27
Hình 3.29 Vị trí vạch cắt theo bitrate, và hình dạng của nó trên phổ.....27
Hình 3.30 Đổi đuôi tệp không phục hồi chất lượng.....28
Hình 3.31 Chất lượng cảm nhận theo bitrate — dạng định tính của ba codec.....29
Hình 3.32 Ba khối của một codec nơ-ron: mã hoá, lượng tử hoá vector, giải mã.....30
Hình 3.33 Ranh giới giữa codec cổ điển và codec nơ-ron đang mờ đi.....31
Hình 3.34 Chuỗi token: vừa để nén, vừa là ngôn ngữ cho AI tạo sinh.....32
Hình 3.35 Kết quả phải thấy: hai sóng khác nhau cho ra phổ giống hệt nhau.....34
Hình 3.36 Phổ của riêng sai số ở ba trường hợp, trên cùng một thang tuyệt đối.....35
Hình 3.37 Bốn nhánh khái niệm của chương 3 và chương nào sẽ dùng lại từng nhánh 40

DANH MỤC BẢNG BIỂU

Bảng 3.1 Ba tham số của một tệp âm thanh số....7
Bảng 3.2 Tần số lấy mẫu chuẩn và lý do chọn....11
Bảng 3.3 Vài ví dụ gập ở tần số lấy mẫu 8.000 Hz....14
Bảng 3.4 Độ sâu bit, số mức và bước lượng tử....17
Bảng 3.5 Vài mốc dBFS nên thuộc....21
Bảng 3.6 Bốn quy ước dùng ở mọi chương sau....23
Bảng 3.7 Ba nhóm định dạng — so sánh....25
Bảng 3.8 Vạch cắt tiêu biểu theo bitrate MP3....28
Bảng 3.9 Ba codec có tổn hao phổ biến nhất....29
Bảng 3.10 Vài mốc đáng chú ý của Opus....31
Bảng 3.11 Chẩn đoán nhanh khi kết quả không như hình....36
Bảng 3.12 Thuật ngữ chương 3....40

CHƯƠNG 3
SỐ HOÁ ÂM THANH

Mục tiêu của chương

Sau chương này, người đọc làm được những việc sau.

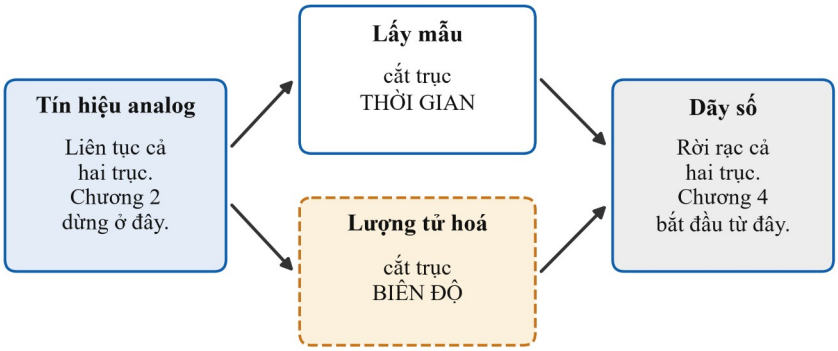
- Phân biệt được hai phép rời rạc hoá của số hoá, nói được mỗi phép cắt trục nào và do tham số nào điều khiển.
- Phát biểu và vận dụng được định lý Nyquist–Shannon, và giải thích được vì sao dãy mẫu rời rạc lại chứa đủ thông tin của đường liên tục.
- Tính được tần số thấy được của một tần số vượt Nyquist, và giải thích được vì sao phải chặn chồng phổ trước khi lấy mẫu.
- Nêu được quan hệ giữa độ sâu bit, số mức và dải động; giải thích được dither làm gì và dùng ở bước nào.
- Đổi được giữa biên độ mẫu và dBFS; phát biểu và giải thích được nguyên tắc cốt lõi về số nguyên và số thực.
- Phân biệt được ba nhóm định dạng, nêu được ba điểm mù của tai làm nền cho nén có tổn hao, và nhận diện được dấu vết của nó trên phổ.
- Mô tả được nguyên lý của codec nơ-ron và nêu được nó nối với phần AI tạo sinh ở cuối cuốn sách thế nào.

> **Chương này khác hai chương trước ở chỗ nào**
>
> Chương 1 nói về tự nhiên. Chương 2 nói về thiết bị. Chương này nói về một *phép biến đổi* — và đó là phép biến đổi mà toàn bộ phần còn lại của cuốn sách đứng trên.
>
> Vì thế nó cũng là chương có nhiều công thức nhất trong bốn chương đầu. Nhưng mỗi công thức đều đi kèm một hình dựng từ tín hiệu thật, và mỗi con số đều kiểm được bằng mã.
>
> Nếu chỉ có thời gian đọc kỹ *một* mục, hãy đọc mục 3.5 — nó ngắn nhất nhưng được dùng ở mọi chương sau.

3.1 Vị trí của chương trong cuốn sách

3.1.1 Mở nốt cái ô vuông của chương 2

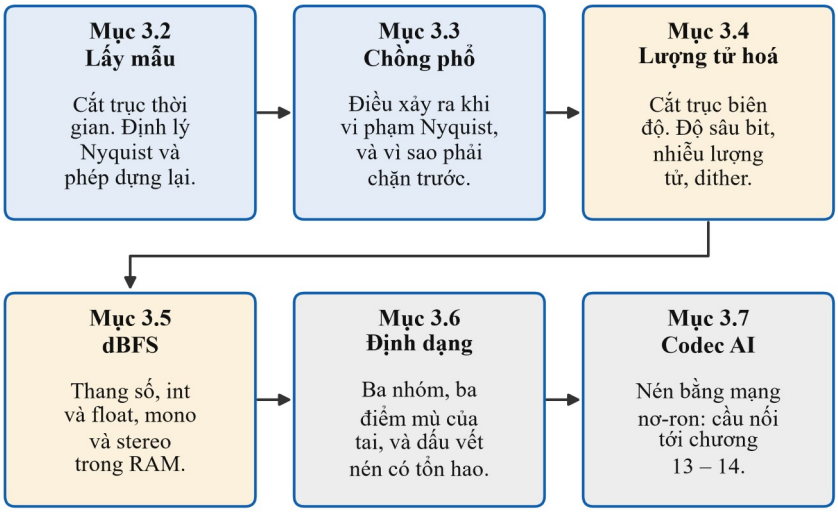
Chương 2 vẽ chuỗi tín hiệu thành bảy khối, và một trong bảy khối ấy chỉ là một ô vuông ghi chữ ADC. Chương này mở ô vuông đó ra. Bên trong nó chỉ có hai phép, và mọi thứ còn lại của chương đều là hệ quả của hai phép ấy.



Hình 3.1 Hai phép rời rạc hoá độc lập: lấy mẫu cắt trục thời gian, lượng tử hoá cắt trục biên độ

Chương 2 đã nhắc trước một lần, và đây là chỗ nhắc lại lần cuối vì nó chi phối cả chương: hai phép này *độc lập* với nhau. Chúng do hai bộ phận khác nhau thực hiện, gây hai loại sai số khác nhau, và được điều khiển bằng hai tham số khác nhau — tần số lấy mẫu và độ sâu bit. Nếu gộp chúng làm một thì nội dung chương khó nắm bắt; nếu tách được thì dễ theo dõi.

3.1.2 Sáu mục và đường đi của chương



Hình 3.2 Đường đi của chương 3: từ phép lấy mẫu tới định dạng tệp và codec nơ-ron

Chương này là chương bản lề của cả cuốn sách

Từ chương 4 trở đi, *mọi* dòng mã của cuốn sách đều làm việc trên dãy số mà chương này tạo ra.

Vì thế những quy ước tưởng nhỏ nhặt ở đây — mẫu nằm trong khoảng nào, mono khác stereo ở chỗ nào, thang dBFS lấy mốc ở đâu — sẽ xuất hiện trong từng bài tập của mười hai chương còn lại.

Nếu chỉ có thời gian đọc kỹ *một* mục của chương này, hãy đọc kỹ mục 3.5.

3.1.3 Ba tham số sẽ theo suốt cuốn sách

Chương này sinh ra ba con số, và từ chương 4 trở đi gần như mọi hàm trong cuốn sách đều mang theo ít nhất một trong ba con số ấy. Ba con số này quyết định dung lượng tệp, dải tần biểu diễn được, dải động, và cả hình dạng mảng trong bộ nhớ — biết ba con số là biết gần hết những gì cần biết về một tệp âm thanh.

Bảng 3.1 Ba tham số của một tệp âm thanh số

Tham số	Ký hiệu quen	Do phép nào sinh ra	Học ở mục nào
Tần số lấy mẫu	sr, fs	Phép cắt trục *thời gian*	3.2
Độ sâu bit	bits, b	Phép cắt trục *biên độ*	3.4
Số kênh	ch, channels	Quy ước lưu trữ	3.5.3

3.1.4 Gợi ý cách đọc chương này

Chương này có nhiều công thức hơn ba chương trước cộng lại. Nhưng nhìn kỹ thì chúng chỉ trả lời bốn câu hỏi, xếp thành hai cặp đối xứng nhau: mỗi trục cần cắt bao nhiêu thì đủ, và cắt thiếu thì hỏng thế nào. Trục thời gian trả lời bằng định lý Nyquist ở mục 3.2.2 và hiện tượng chồng phổ ở mục 3.3. Trục biên độ trả lời bằng độ sâu bit ở mục 3.4.2 và nhiễu lượng tử ở mục 3.4.3.

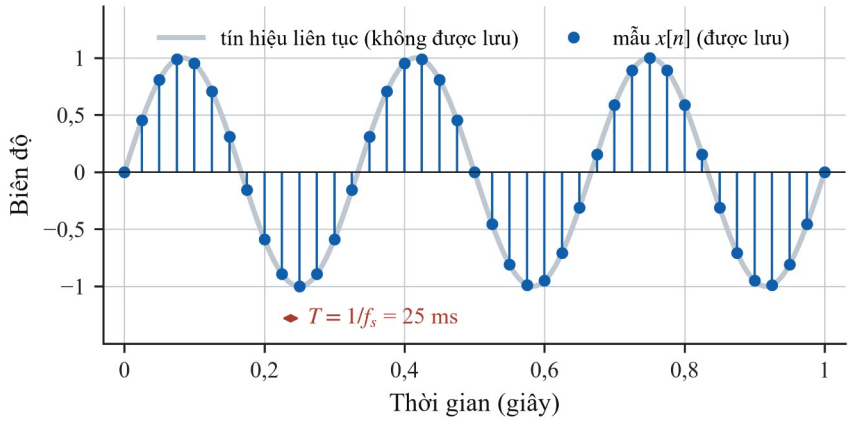
Đọc theo khung ấy thì bốn mục dài nhất của chương trở thành bốn ô của một bảng hai chiều. Hai mục còn lại — 3.5 về cách lưu mẫu và 3.6 về định dạng — không phải lý thuyết mới mà là hệ quả thực hành của bốn ô ấy. Đến cuối mỗi mục, thử tự trả lời: mục vừa rồi thuộc ô nào?

3.2 Lấy mẫu và định lý Nyquist

3.2.1 Lấy mẫu là gì

Lấy mẫu là phép đo giá trị tín hiệu tại những thời điểm cách đều nhau. Số lần đo trong một giây gọi là *tần số lấy mẫu*, ký hiệu fs, đơn vị héc. Khoảng thời gian giữa hai mẫu liên tiếp là chu kỳ lấy mẫu.

$$x[n] = x(nT), \qquad T = \frac{1}{f_s} \tag{3.1}$$



Hình 3.3 Lấy mẫu: chỉ dãy chấm được lưu lại, đường liền nét thì không

Một con số để cảm nhận: ở chuẩn CD 44.100 Hz, hai mẫu liên tiếp cách nhau 22,7 micro giây. Trong một giây có 44.100 con số cho mỗi kênh; một bài hát bốn phút stereo có khoảng 21,2 triệu con số. Đó là toàn bộ những gì máy tính biết về bài hát ấy.

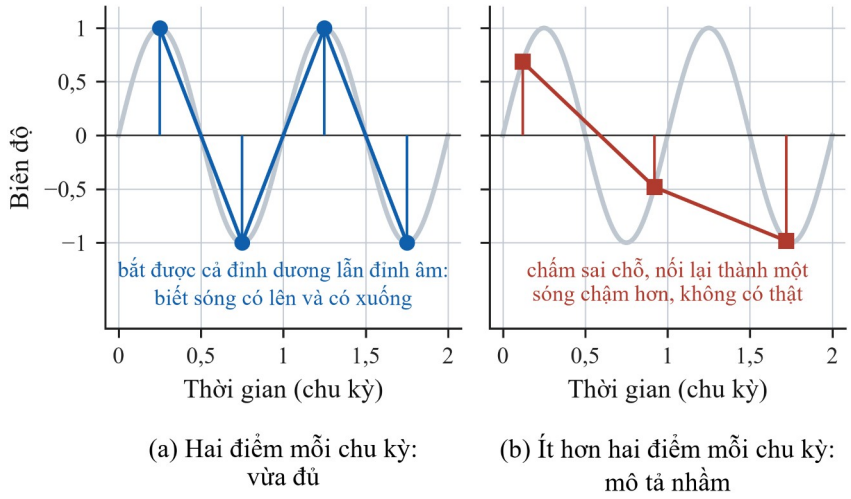
> **Câu hỏi của cả mục này**
>
> Đường liền nét không được lưu ở đâu cả. Chỉ dãy chấm được lưu.
>
> Vậy từ dãy chấm ấy có dựng lại được đường liền nét không? Và nếu được thì với điều kiện nào?
>
> Câu trả lời — định lý Nyquist–Shannon — là một trong những kết quả đẹp nhất của kỹ thuật tín hiệu, và nó phản trực giác hơn nhiều người tưởng.

3.2.2 Định lý Nyquist–Shannon

Định lý phát biểu rằng: để khôi phục *trọn vẹn* một tín hiệu, tần số lấy mẫu phải lớn hơn hai lần tần số cao nhất có trong tín hiệu ấy.

$$f_s > 2f_{\max}\,; \qquad f_{\text{Nyq}} = \frac{f_s}{2} \tag{3.2}$$

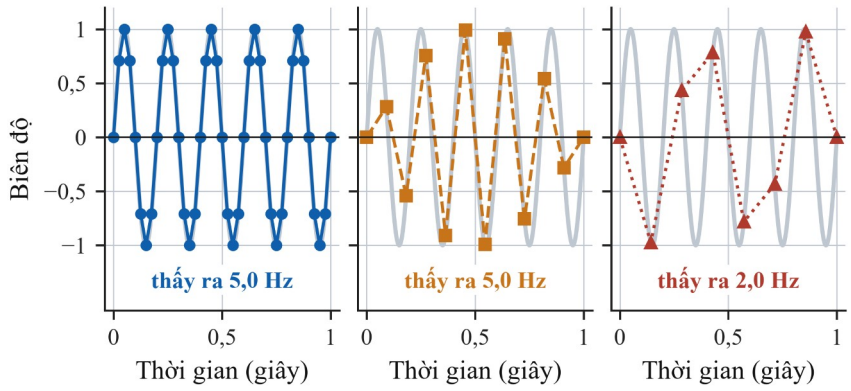
Giá trị bằng nửa tần số lấy mẫu gọi là *tần số Nyquist*. Đó là tần số cao nhất mà một hệ thống số biểu diễn trung thực được. Mọi thứ ở trên nó không phải là "biểu diễn kém đi" mà là biểu diễn *sai* hẳn — mục 3.3 sẽ nói kỹ.



(a) Hai điểm mỗi chu kỳ: vừa đủ (b) Ít hơn hai điểm mỗi chu kỳ: mô tả nhầm

Hình 3.4 Trực giác hình học của định lý: mỗi chu kỳ cần tối thiểu hai điểm đo

Trực giác này không cần một dòng toán nào. Muốn ghi lại được rằng sóng có *lên* và có *xuống* thì mỗi chu kỳ phải có ít nhất hai điểm đo — một cho phần trên, một cho phần dưới. Lấy mẫu thưa hơn thì các điểm rơi sai chỗ và nối lại thành một sóng chậm hơn, không có thật.



(a) $f_s = 8f$: thoải mái (b) $f_s = 2{,}2f$: vừa đủ (c) $f_s = 1{,}4f$: vi phạm

Hình 3.5 Cùng một sóng 5 Hz, ba mật độ lấy mẫu khác nhau

Một điểm tinh tế về dấu lớn hơn

Định lý yêu cầu fs > 2·fmax — lớn hơn hẳn chứ không chỉ bằng — và thực tế luôn lấy dư khá nhiều.

Lý do: ở đúng ngưỡng $2 \cdot f_{max}$, nếu các mẫu rơi trúng chỗ tín hiệu cắt không thì mọi mẫu đều bằng không — mất sạch. Hơn nữa bộ lọc chống chồng phổ cần một khoảng để suy giảm dần, không thể cắt thẳng đứng.

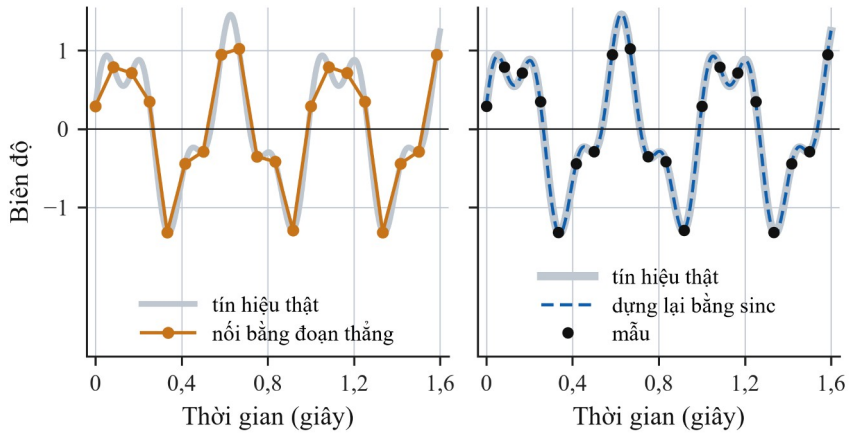
Chuẩn CD chọn 44.100 Hz cho dải nghe 20 kHz — dư 4.100 Hz so với ngưỡng lý thuyết 40.000 Hz. Khoảng dư ấy chính là chỗ cho bộ lọc làm việc.

3.2.3 Phần phản trực giác: dựng lại bằng sinc

Định lý không chỉ nói "lấy mẫu đủ dày thì không mất gì". Nó còn chỉ ra *cách* dựng lại, và cách ấy không phải là nối các chấm bằng đoạn thẳng.

$$x(t) = \sum_{n=-\infty}^{+\infty} x[n]\,\mathrm{sinc}\left(\frac{t - nT}{T}\right) \tag{3.3}$$

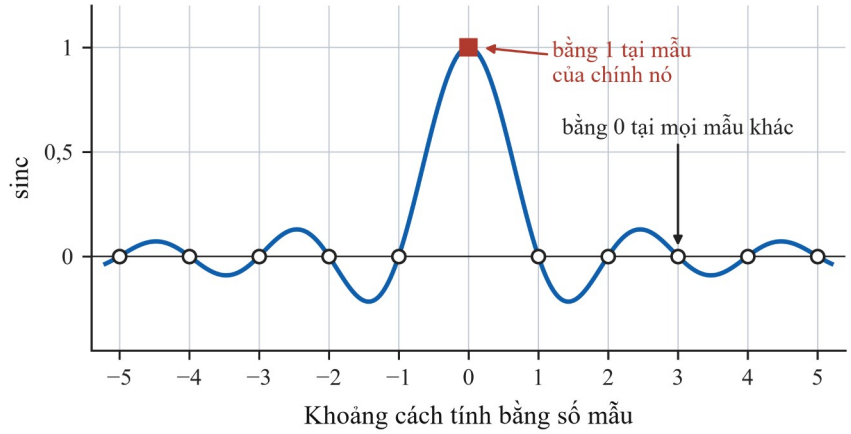
Mỗi mẫu toả ra một hàm sinc; cộng tất cả các hàm ấy lại thì được đúng tín hiệu gốc. Đây là chỗ phản trực giác nhất của cả chương: giữa hai mẫu không có thông tin nào được lưu, vậy mà tín hiệu ở giữa hai mẫu vẫn xác định được chính xác.



(a) Nối chấm bằng đoạn thẳng: sai (b) Dựng lại bằng sinc: trùng khít

Hình 3.6 Nối chấm bằng đoạn thẳng thì sai; dựng lại bằng sinc thì trùng khít

Lý do nằm ở một tính chất rất gọn của hàm sinc: tại mốc lấy mẫu của chính nó thì nó bằng 1, còn tại *mọi* mốc lấy mẫu khác thì nó bằng đúng 0.



Hình 3.7 Hàm sinc: bằng 1 tại mẫu của chính nó, bằng 0 tại mọi mẫu khác

Nhờ tính chất ấy, tại mỗi mốc lấy mẫu chỉ có đúng một hàm đóng góp, nên tổng nhất định đi qua đúng giá trị của mẫu. Còn giữa hai mẫu thì mọi hàm cùng đóng góp — và đó chính là chỗ thông tin được khôi phục.

Điều này có ý nghĩa thực tế gì

Nó là câu trả lời dứt điểm cho một hiểu lầm rất phổ biến: rằng âm thanh số "có bậc thang" nên kém âm thanh analog.

Dạng sóng bậc thang mà nhiều phần mềm vẽ ra chỉ là cách *vẽ* cho tiện. Bộ chuyển đổi ngược thật sự không phát ra bậc thang — nó dựng lại đường cong trơn theo đúng công thức trên.

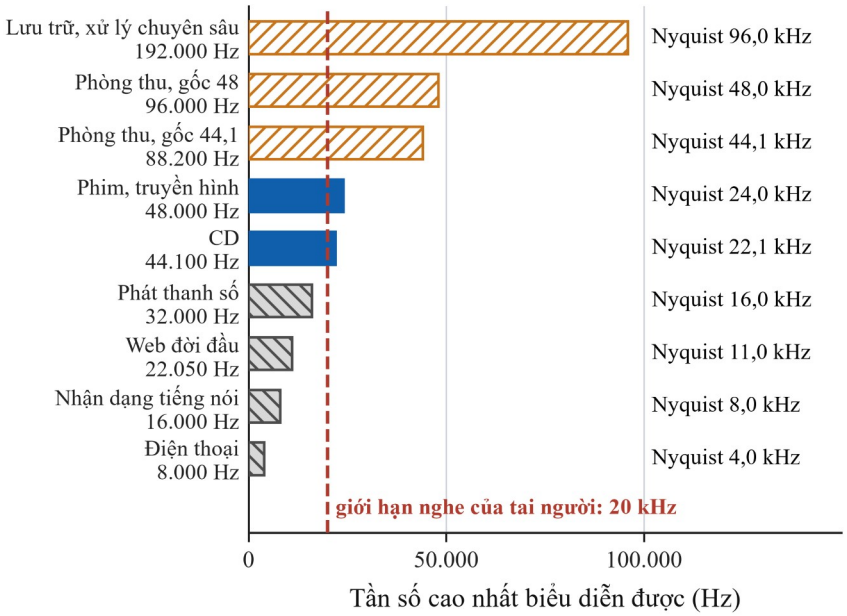
Bậc thang là chuyện của phép *lượng tử hoá* ở mục 3.4, và đó là một phép hoàn toàn khác.

3.2.4 Các tần số lấy mẫu chuẩn

Những con số quen thuộc trong ngành không phải ngẫu nhiên. Mỗi con số có một lý do, và biết lý do thì không phải học thuộc.

Bảng 3.2 Tần số lấy mẫu chuẩn và lý do chọn

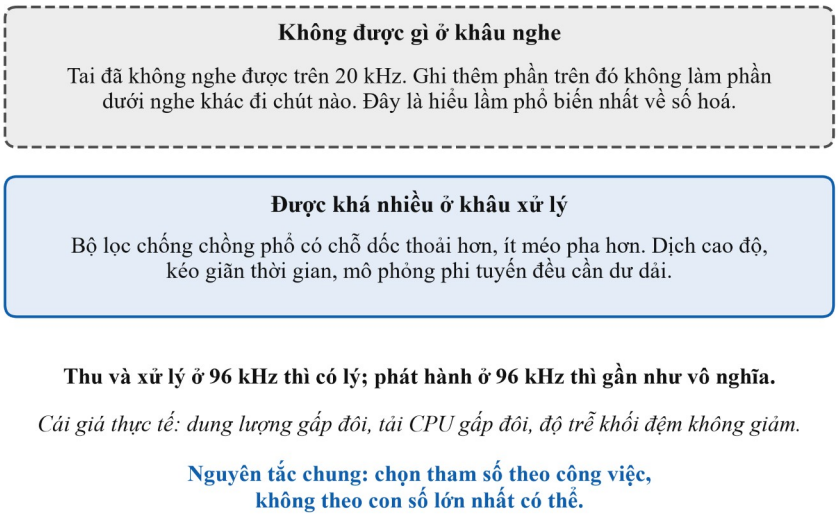
Tần số	Nyquist	Dùng ở đâu	Vì sao chọn con số ấy
8.000 Hz	4,0 kHz	Điện thoại	đủ cho dải 300–3.400 Hz của tiếng nói
16.000 Hz	8,0 kHz	Nhận dạng tiếng nói	gấp đôi, giữ được phụ âm gió
22.050 Hz	11,0 kHz	Web đời đầu	đúng một nửa chuẩn CD, đổi qua lại dễ
32.000 Hz	16,0 kHz	Phát thanh số	Nyquist 16 kHz, đủ với người nghe phổ thông
44.100 Hz	22,1 kHz	CD	chọn để vừa khung hình máy ghi video thời đó
48.000 Hz	24,0 kHz	Phim, truyền hình	chia chẵn cho 24, 25, 30 hình mỗi giây
88.200 Hz	44,1 kHz	Phòng thu, gốc 44,1	gấp đôi chuẩn CD, hạ về không cần nội suy
96.000 Hz	48,0 kHz	Phòng thu, gốc 48	gấp đôi chuẩn phim
192.000 Hz	96,0 kHz	Lưu trữ, xử lý chuyên sâu	để lại chỗ cho dịch cao độ và hiệu ứng



Hình 3.8 Tần số cao nhất biểu diễn được của từng chuẩn, đối chiếu với giới hạn nghe

3.2.5 Một quan niệm sai phổ biến: tần số lấy mẫu càng cao càng hay

Đây là hiểu lầm phổ biến nhất về số hoá, và nó gây tốn kém thực tế cho nhiều người. Lập luận nghe có vẻ hợp lý: lấy mẫu dày hơn thì gần với đường cong gốc hơn, nên nghe hay hơn. Nhưng định lý ở mục 3.2.2 đã bác bỏ chính lập luận đó: một khi đã vượt ngưỡng Nyquist, tín hiệu được khôi phục *trọn vẹn*, không có chuyện "gần hơn nữa".



Hình 3.9 Tăng tần số lấy mẫu được gì và không được gì

Điều này không có nghĩa 96 kHz là vô dụng. Lợi ích của nó có thật, nhưng nằm ở khâu *xử lý* chứ không ở khâu nghe: bộ lọc chống chồng phổ có chỗ dốc thoải hơn nên ít méo pha; các phép dịch cao độ, kéo giãn thời gian và mô phỏng phi tuyến đều sinh ra thành phần tần số cao và cần chỗ chứa chúng.

Phát biểu gọn để mang theo

Thu và *xử lý* ở 96 kHz thì có lý. *Phát hành* ở 96 kHz thì gần như vô nghĩa.

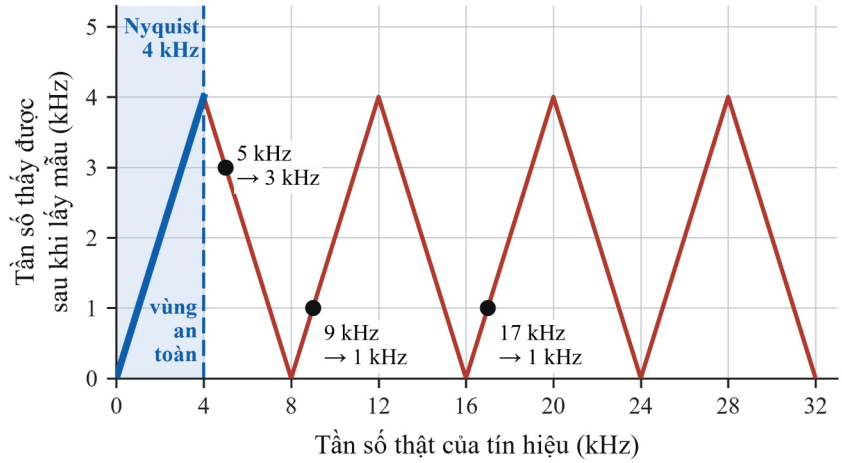
Cái giá phải trả là thực tế: dung lượng gấp đôi, tải CPU gấp đôi. Và độ trễ khối đệm *không* giảm, vì cùng số mẫu thì cùng số phép tính — chỉ có thời gian gom đủ một khối là ngắn lại.

Nguyên tắc chung của cả cuốn sách: chọn tham số theo *công việc*, không theo con số lớn nhất có thể.

3.3 Chồng phổ và bộ lọc chặn nó

3.3.1 Điều gì xảy ra khi vi phạm Nyquist

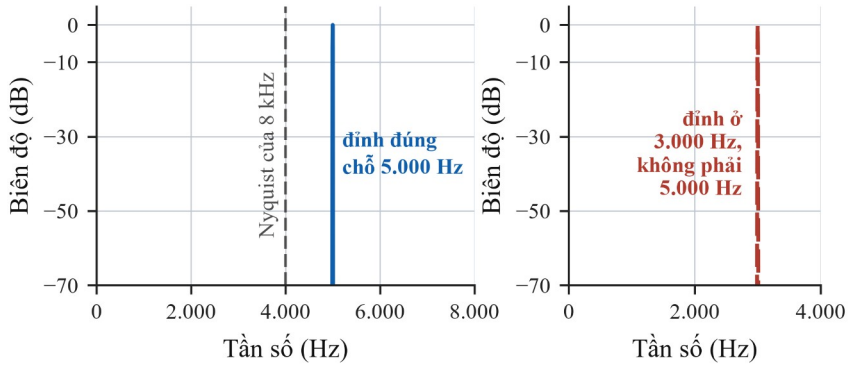
Khi tín hiệu chứa tần số vượt quá tần số Nyquist, phép lấy mẫu không còn đủ điểm để mô tả đúng. Tần số cao ấy không biến mất, và cũng không bị làm mờ — nó bị *gập* xuống thành một tần số thấp hơn, và tần số giả ấy trộn lẫn vào phần tín hiệu thật.



Hình 3.10 Bản đồ gập: mọi tần số trên Nyquist đều rơi về một tần số trong vùng an toàn

$$f_{\text{thay}} = |f - k f_s|, \qquad k = \text{round}\left(\frac{f}{f_s}\right) \tag{3.4}$$

Đường gấp khúc trong hình trên là nguồn gốc của tên gọi "gập": nó trông giống một tờ giấy được gấp nhiều lần. Chú ý một điều quan trọng: phép gập *không* thuận nghịch. Sau khi 5.000 Hz đã thành 3.000 Hz, không có thông tin nào còn lại để biết nó vốn là 5.000 hay vốn là 3.000.



(a) Lấy mẫu ở 96 kHz (b) Lấy mẫu ở 8 kHz

Hình 3.11 Kiểm thật trên tín hiệu: cùng một sóng 5.000 Hz, hai tần số lấy mẫu, hai kết quả

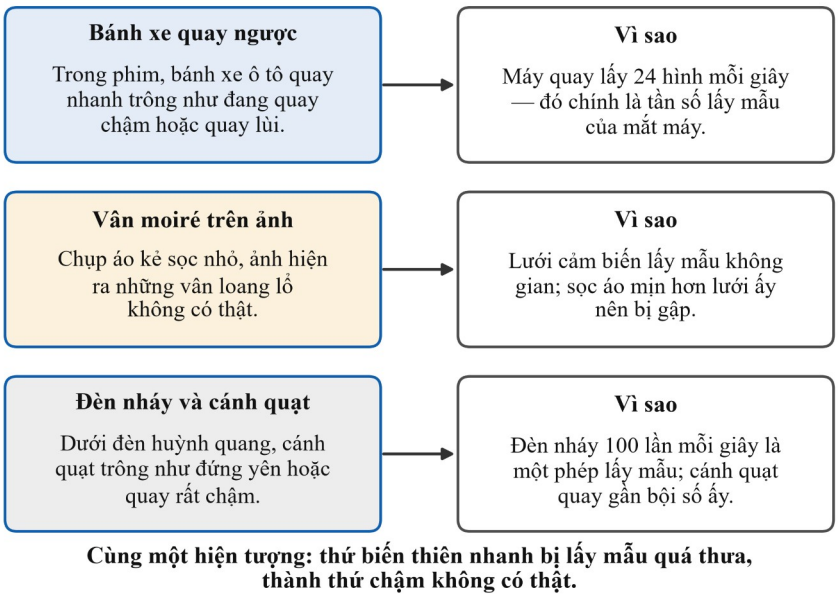
Bảng 3.3 Vài ví dụ gập ở tần số lấy mẫu 8.000 Hz

Tần số thật	Có vượt Nyquist 4.000 Hz?	Tần số thấy được
1.000 Hz	không	1.000 Hz
3.000 Hz	không	3.000 Hz
5.000 Hz	có	3.000 Hz ← giả
7.000 Hz	có	1.000 Hz ← giả
9.000 Hz	có	1.000 Hz ← giả
17.000 Hz	có	1.000 Hz ← giả

Hai dòng cuối bảng đáng chú ý: 9.000 Hz và 17.000 Hz đều gập về cùng 1.000 Hz. Nghĩa là ba tần số khác nhau — 1.000, 9.000 và 17.000 — sau khi lấy mẫu ở 8.000 Hz cho ra những dãy số giống hệt nhau. Đó không phải "khó phân biệt" mà là *không có* gì để phân biệt.

3.3.2 Chồng phổ không chỉ có trong âm thanh

Trước khi bàn cách chặn, nên nhận ra rằng hiện tượng này quen thuộc hơn chúng ta tưởng. Ba ví dụ dưới đây đều là cùng một chuyện, chỉ khác cái đang bị lấy mẫu.

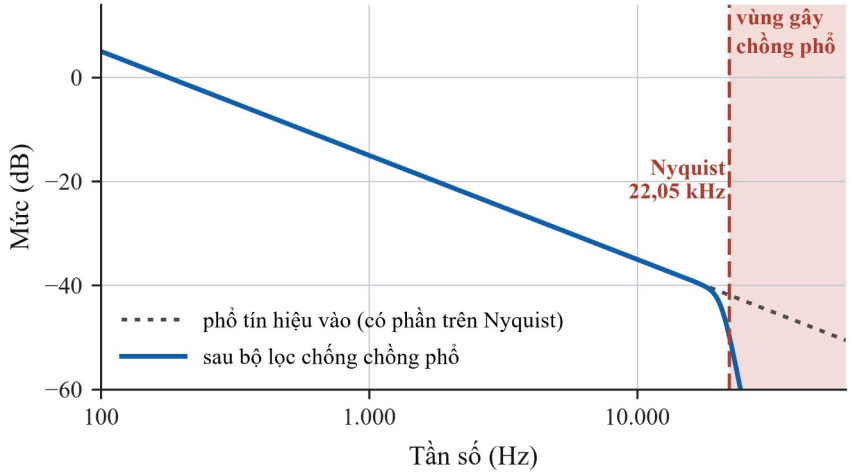


Hình 3.12 Bánh xe quay ngược, vân moiré, cánh quạt dưới đèn nháy: cùng một hiện tượng

Nếu hiểu vì sao bánh xe ô tô trong phim trông như quay ngược thì đã hiểu chồng phổ. Máy quay lấy 24 hình mỗi giây — đó chính là một phép lấy mẫu, và bánh xe quay nhanh hơn ngưỡng Nyquist của nó thì bị mô tả nhầm thành quay chậm hoặc quay lùi.

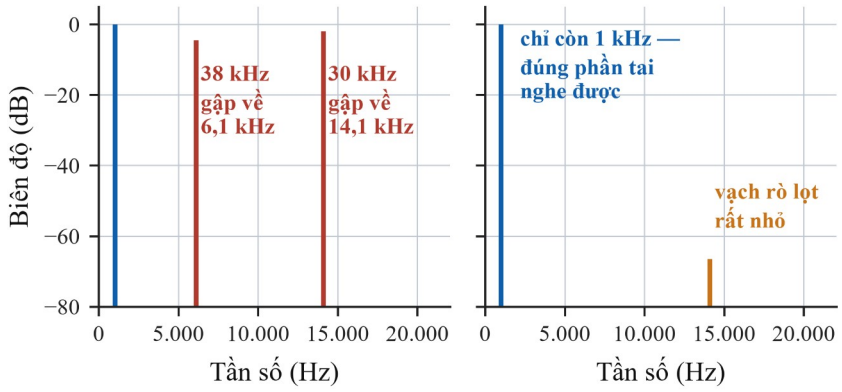
3.3.3 Bộ lọc chống chồng phổ — và vì sao vị trí của nó là bắt buộc

Không có cách nào sửa chồng phổ sau khi đã lấy mẫu. Vì thế giải pháp duy nhất là chặn từ trước: đặt một bộ lọc thông thấp ngay *trước* bộ chuyển đổi, cắt bỏ mọi tần số vượt Nyquist trong khi tín hiệu còn ở dạng analog.



Hình 3.13 Bộ lọc chống chồng phổ cắt phần trên Nyquist trước khi lấy mẫu

Đây chính là khối "lọc chống răng cưa" đã xuất hiện trong sơ đồ giao diện âm thanh ở chương 2. Lúc ấy nó chỉ là một ô vuông chưa giải thích; giờ thì đã rõ vì sao nó phải có mặt và vì sao nó phải nằm đúng ở vị trí ấy.



(a) Không lọc trước khi lấy mẫu (b) Có lọc trước khi lấy mẫu

Hình 3.14 Cùng một tín hiệu có thành phần siêu âm: lấy mẫu thẳng và lấy mẫu sau khi lọc

Điều đáng lưu ý ở hình trên

Hai thành phần siêu âm 30 kHz và 38 kHz vốn *hoàn toàn vô hại* — tai không nghe được chúng.

Nhưng sau khi lấy mẫu ở 44,1 kHz mà không lọc, chúng biến thành hai vạch ở 14,1 kHz và 6,1 kHz — nằm giữa dải nghe, và giờ thì nghe rất rõ.

Chồng phổ không làm mất tín hiệu; nó *biến* thứ vô hại thành thứ có hại. Đó là lý do bộ lọc chống chồng phổ không phải một tuỳ chọn nâng cấp chất lượng, mà là một khối bắt buộc của mọi bộ chuyển đổi.

Chú ý ở khung phải của hình: vẫn còn một vạch rất nhỏ quanh 14 kHz, thấp hơn tín hiệu chính khoảng sáu mươi decibel. Đó không phải lỗi vẽ. Bộ lọc thật không cắt thẳng đứng được, nên luôn còn một phần năng lượng lọt qua. Điều này cho thấy: mục tiêu không phải "loại bỏ hoàn toàn" mà là "đẩy xuống dưới ngưỡng nghe được" — một cách nghĩ sẽ gặp lại ở chương 5 khi thiết kế bộ lọc số.

3.3.4 Chồng phổ trong mã chương trình

Đến đây dễ có cảm giác chồng phổ là chuyện của phần cứng, đã có bộ lọc xử lý nên người lập trình không cần quan tâm. Nhận định này sai, và đây là chỗ quan trọng nhất của mục 3.3 với một cuốn sách về lập trình.

Bộ lọc chống chồng phổ chỉ bảo vệ được đúng một chỗ: lối vào của bộ chuyển đổi. Nhưng mã của chúng ta còn tự sinh và tự biến đổi tín hiệu, và mỗi lần như thế là một cơ hội tạo ra chồng phổ mới mà không có bộ lọc nào chặn.

- **Khi sinh tín hiệu bằng công thức.** Cộng các hài của một sóng vuông theo đúng công thức ở chương 1 thì mọi hài vượt Nyquist đều gập xuống. Sóng vuông 5.000 Hz sinh kiểu ấy ở 44.100 Hz sẽ có những vạch lạ không phải hài của nó. Cách đúng: dừng cộng ở hài cuối cùng còn dưới Nyquist.
- **Khi hạ tần số lấy mẫu.** Lấy một mẫu trong mỗi hai mẫu để hạ từ 44,1 kHz xuống 22,05 kHz mà không lọc trước thì mọi thứ trên 11 kHz gập xuống. Đây đúng là hình 3.14, chỉ khác là lần này lỗi nằm trong mã chứ không ở phần cứng.

– **Khi làm méo phi tuyến.** Mọi phép phi tuyến — cắt ngọn, bão hoà, mô phỏng đèn điện tử — đều sinh ra hài mới ở tần số cao. Nếu tín hiệu vào đã gần Nyquist thì các hài ấy gập xuống thành những tần số không hài hoà, gây méo tiếng rõ rệt. Chương 8 sẽ gặp lại vấn đề này.

Hai cách xử lý, đều sẽ gặp lại

Lọc trước khi hạ. Trước mỗi phép hạ tần số lấy mẫu, chạy một bộ lọc thông thấp cắt ở Nyquist *mới*. Thư viện chuẩn mực đã làm sẵn bước này — nhưng cần biết bước ấy tồn tại thì mới kiểm tra được.

Làm việc ở tần số cao hơn. Nhân tần số lấy mẫu lên vài lần, làm phép phi tuyến ở đó, lọc, rồi hạ về. Kỹ thuật này gọi là oversampling, và nó chính là lý do thứ hai để dùng 96 kHz đã nói ở mục 3.2.5.

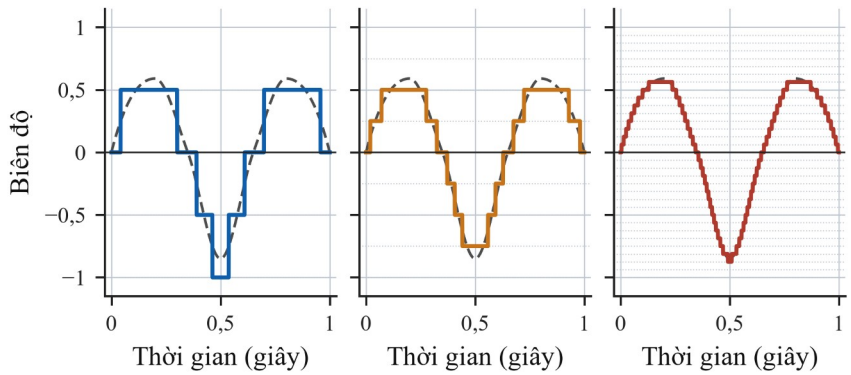
Cả hai đều dựa trên đúng một nguyên tắc: giữ mọi thành phần nằm dưới Nyquist tại *mọi* bước, không chỉ ở bước đầu.

3.4 Lượng tử hoá, độ sâu bit và dither

3.4.1 Rời rạc hoá trục còn lại

Sau khi lấy mẫu, mỗi giá trị đo được vẫn là một số thực với vô hạn chữ số — máy tính không lưu được. Lượng tử hoá là phép làm tròn mỗi mẫu về một trong số hữu hạn mức cho phép. Số mức ấy do *độ sâu bit* quyết định.

$$M = 2^b\,; \qquad \Delta = \frac{2A_{\max}}{2^b} \tag{3.5}$$



(a) 2 bit — 4 mức (b) 3 bit — 8 mức (c) 5 bit — 32 mức

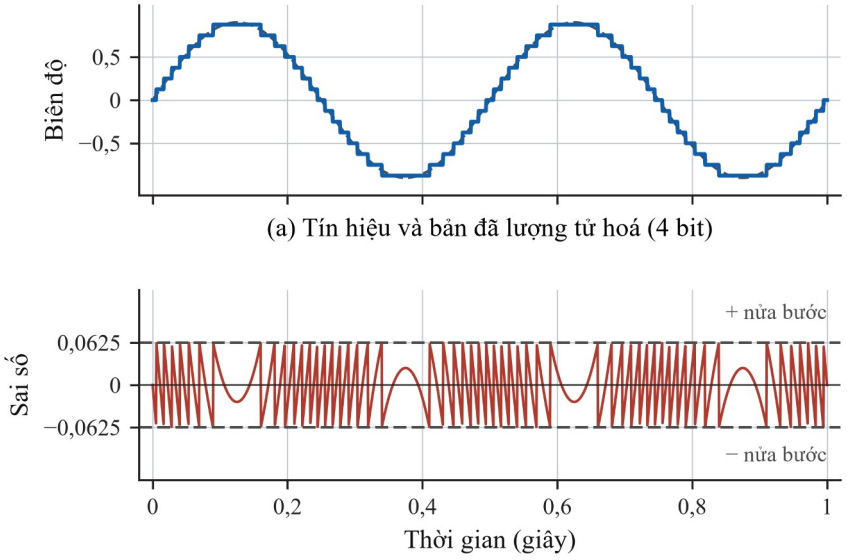
Hình 3.15 Cùng một tín hiệu ở ba độ sâu bit: số mức càng nhiều, bậc thang càng mịn

Bảng 3.4 Độ sâu bit, số mức và bước lượng tử

Độ sâu bit	Số mức	Bước lượng tử (phần của toàn thang)	Dải động
8 bit	256	0,781 %	49,9 dB
12 bit	4.096	0,049 %	74,0 dB
16 bit	65.536	30,52 phần triệu	98,1 dB
20 bit	1.048.576	1,91 phần triệu	122,2 dB
24 bit	16.777.216	0,12 phần triệu	146,2 dB

3.4.2 Nhiễu lượng tử — và vì sao nó bất lợi hơn nhiễu thông thường

Chênh lệch giữa giá trị thật và giá trị đã làm tròn gọi là nhiễu lượng tử. Nó bị chặn cứng trong nửa bước lượng tử, nên về mặt năng lượng thì rất nhỏ. Tuy nhiên, năng lượng chưa phản ánh đầy đủ ảnh hưởng của nhiễu.



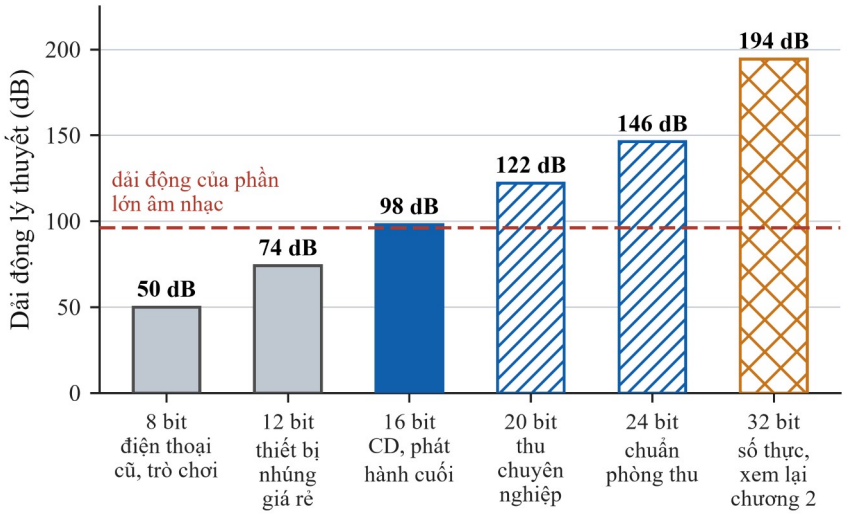
(a) Tín hiệu và bản đã lượng tử hoá (4 bit)

(b) Sai số lượng tử — luôn nằm trong nửa bước

Hình 3.16 Sai số lượng tử nhìn riêng ra: bị chặn trong nửa bước, nhưng có quy luật

$$e_{\text{rms}} = \frac{\Delta}{\sqrt{12}}; \qquad \text{SQNR} \approx 6{,}02\,b + 1{,}76 \quad [\text{dB}] \tag{3.6}$$

Công thức trên cho quan hệ quen thuộc: mỗi bit thêm vào cho thêm 6,02 dB dải động. Con số 1,76 dB đi kèm đến từ hằng số căn 12 trong công thức nhiễu hiệu dụng — sai số phân bố đều trong một bước thì giá trị hiệu dụng của nó bằng bước chia căn 12.



Hình 3.17 Dải động theo độ sâu bit, và vùng dải động của phần lớn âm nhạc

Vì sao thu ở 24 bit khi phát hành chỉ cần 16

Không phải vì tai nghe được 146 dB — không ai nghe được thế.

Mà vì lúc *thu* chưa biết đỉnh lớn nhất sẽ ở đâu, nên phải để lại headroom rộng (chương 2, mục 2.3.5). Headroom rộng thì tín hiệu nằm thấp, và cần dải động dư để nó không chạm sàn nhiễu.

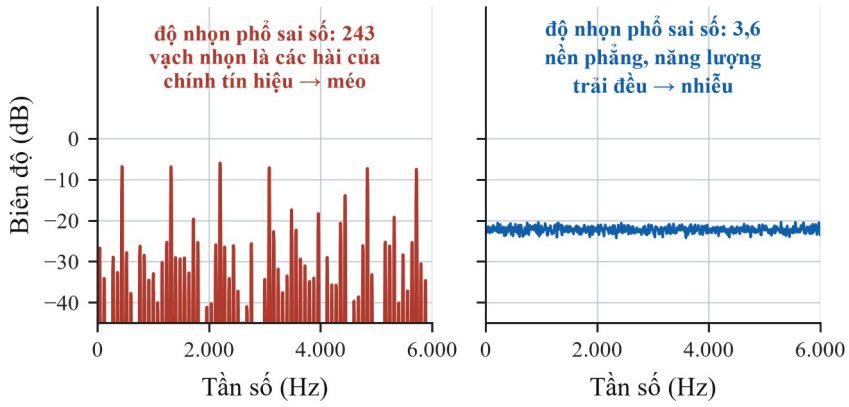
Và vì mọi phép xử lý trung gian đều làm tròn một lần. Bắt đầu từ 24 bit thì sau mười phép vẫn còn dư; bắt đầu từ 16 bit thì không.

Đây đúng là lập luận của quy tắc gain staging ở chương 2, chỉ đổi sang ngôn ngữ số.

3.4.3 Dither — đổi méo lấy nhiễu

Quay lại hình ở mục 3.4.2 và nhìn kỹ khung dưới: sai số lượng tử *không* ngẫu nhiên. Nó lặp lại theo nhịp của tín hiệu, vì cùng một giá trị vào thì luôn cho cùng một sai số ra. Sai số có quy luật thì phổ của nó không phẳng mà có những vạch nhọn ở các bội tần số tín hiệu — và tai nghe ra thành *méo*.

Đây là điểm mấu chốt mà nhiều tài liệu chỉ nêu sơ lược: vấn đề của lượng tử hoá không phải là mức năng lượng của sai số, mà là *hình dạng* của nó. Cùng một mức, sai số dạng tone gây khó chịu hơn nhiều so với sai số dạng nhiễu.



(a) Không dither — sai số là méo (b) Có dither — sai số là nhiễu

Hình 3.18 Phổ của riêng sai số: không dither thì có vạch nhọn, có dither thì phẳng

Dither là kỹ thuật cố tình cộng một lượng nhiễu ngẫu nhiên rất nhỏ *trước* khi làm tròn. Nhiễu ấy phá vỡ quan hệ giữa tín hiệu và sai số, biến sai số từ tone thành nhiễu nền phẳng.

Dither có cái giá của nó

Đo trên một đoạn nhỏ: độ nhọn phổ của sai số giảm từ 243 xuống 3,6 — tức từ rất tone thành gần như phẳng.

Nhưng năng lượng sai số *tăng* lên vài decibel, vì chúng ta vừa cộng thêm nhiễu vào. Đó là cái giá.

Đánh đổi này có lợi vì tai người chịu được nhiễu nền đều dễ hơn nhiều so với méo bám theo tín hiệu. Ở đoạn nhạc nhỏ, chênh lệch nghe được rất rõ.

Dạng nhiễu chuẩn của ngành là TPDF — phân bố tam giác, biên độ một bước lượng tử. Nó khử được cả tương quan bậc nhất lẫn bậc hai, thứ mà nhiễu phân bố đều không khử hết.

3.4.4 Khi nào dùng dither



Hình 3.19 Dither đúng một lần, ở đúng bước hạ bit cuối cùng

Quy tắc này giống hệt quy tắc gain staging ở chương 2 và sẽ giống hệt quy tắc nén có tổn hao ở mục 3.6: mọi phép biến đổi *không* hoàn tác được đều nên dồn về đúng một lần, ở bước cuối cùng. Đây là một nguyên tắc chung của lĩnh vực, không riêng gì dither.

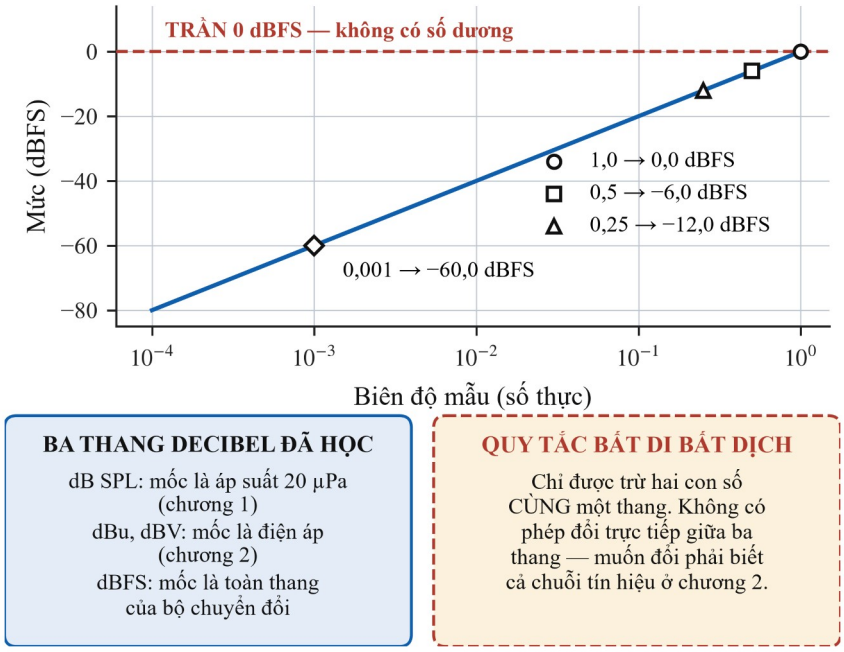
3.5 DBFS và cách máy tính lưu mẫu

3.5.1 Thang decibel thứ ba

Đây là mục ngắn nhất chương nhưng được dùng nhiều nhất: mọi dòng mã của mười hai chương còn lại đều dựa trên những quy ước ở đây.

Chương 1 học dB SPL, mốc là áp suất 20 micro pascal. Chương 2 học dBu và dBV, mốc là điện áp. Chương này thêm thang thứ ba: dBFS, mốc là *toàn thang* của bộ chuyển đổi.

$$L_{\mathrm{FS}} = 20\log_{10}|a| \iff a = 10^{L_{\mathrm{FS}}/20} \tag{3.7}$$



Hình 3.20 Thang dBFS và quan hệ của nó với hai thang đã học

Điểm khác biệt quan trọng nhất

dB SPL và dBu đo một đại lượng *vật lý*, nên chúng có cả số dương lẫn số âm — 94 dB SPL là một mức có thật.

dBFS đo so với *trần*, nên mọi tín hiệu hợp lệ đều mang giá trị *âm*. Thấy một con số dBFS dương là biết ngay đã tràn.

Và như chương 2 đã nhấn: chỉ được trừ hai con số *cùng* một thang. Không có phép đổi trực tiếp giữa ba thang — muốn đổi phải biết cả chuỗi tín hiệu.

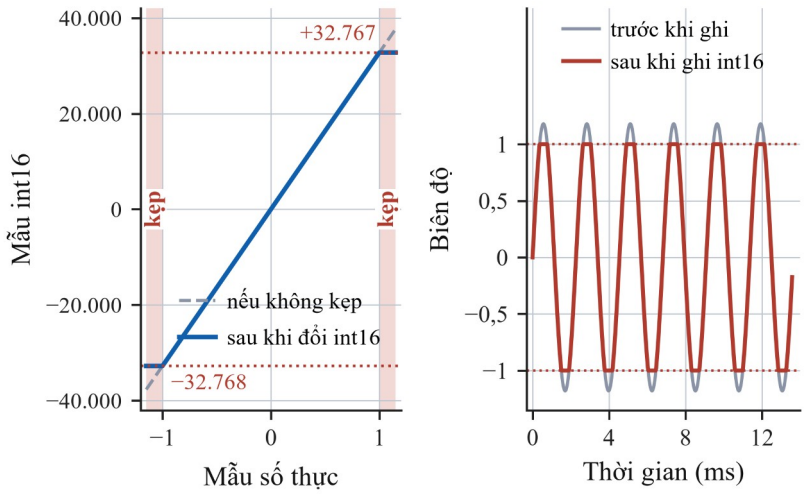
Bảng 3.5 Vài mốc dBFS nên thuộc

Biên độ mẫu	Mức dBFS	Ý nghĩa
1,0	0,00 dBFS	Trần — chạm mức này là sát ngưỡng méo cắt
0,5	−6,02 dBFS	Nửa toàn thang — chia đôi biên độ là −6 dB (chương 1)
0,126	−18,00 dBFS	Mức làm việc chuẩn của phòng thu (chương 2)
0,0010	−60,00 dBFS	Đoạn rất nhỏ; vẫn còn xa sàn nhiễu của 16 bit

3.5.2 Số nguyên hay số thực

Dữ liệu âm thanh trong máy tính có hai cách biểu diễn, và chọn nhầm là nguồn gốc của một loại lỗi rất khó tìm.

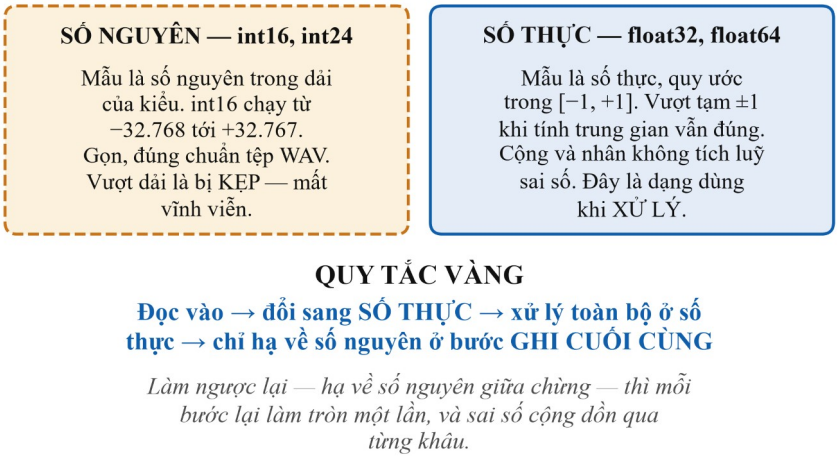
$$i = \text{round}(a \cdot 2^{b-1}), \qquad -2^{b-1} \le i \le 2^{b-1} - 1 \tag{3.8}$$



(a) Đổi số thực sang int16 (b) Đỉnh bị cắt phẳng

Hình 3.21 Đổi số thực sang int16: dải không đối xứng và hiện tượng kẹp biên

Chú ý dải int16 *không* đối xứng: từ −32.768 tới 32.767. Mẫu +1,0 chẵn bị kẹp về 32.767 chứ không thành 32.768. Chênh lệch một đơn vị này vô hại về mặt nghe, nhưng nó là dấu hiệu cho thấy phép đổi float sang int có chi tiết cần đọc tài liệu thư viện thay vì tự viết mà không đối chiếu.



Hình 3.22 Hai cách lưu mẫu, và nguyên tắc cốt lõi khi chọn giữa chúng

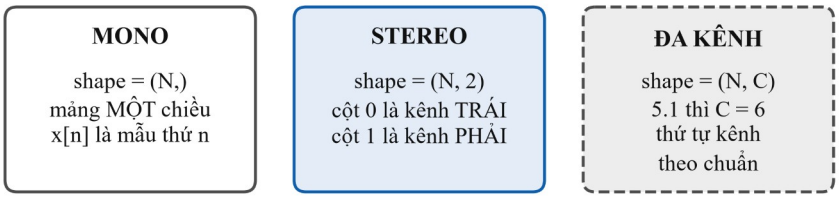
Nguyên tắc cốt lõi — ghi vào chỗ dễ nhìn

Đọc vào → đổi ngay sang *số thực* → xử lý toàn bộ ở số thực → chỉ hạ về số nguyên ở bước *ghi cuối cùng*.

Lý do thứ nhất: số thực cho phép vượt tạm ±1,0 khi tính trung gian. Cộng hai tín hiệu rồi chia đôi thì kết quả vẫn đúng, dù giữa chừng có vượt trần.

Lý do thứ hai: mỗi lần hạ về số nguyên là một lần làm tròn. Hạ giữa chừng nhiều lần thì sai số cộng dồn qua từng khâu — đúng như nhiễu cộng dồn dọc chuỗi analog ở chương 2.

3.5.3 Mono, stereo và hình dạng mảng



Cách tránh viết hai nhánh mã: đọc bằng always_2d=True thì mảng LUÔN hai chiều, kể cả với tệp mono.

Ghép về một kênh để đo: x.mean(axis=1) — nhưng nhớ rằng ghép hai kênh lệch pha có thể triệt tiêu nhau (chương 2).

Hình 3.23 Hình dạng mảng của mono, stereo và đa kênh

Quy ước này quyết định cách viết vòng lặp và cách gọi mọi hàm xử lý ở các chương sau. Cách tránh phải viết hai nhánh mã: đọc bằng always_2d=True thì mảng luôn hai chiều, kể cả với tệp mono, và x.shape[1] luôn dùng được.

Khi cần ghép về một kênh để đo, phép thường dùng là lấy trung bình các cột. Nhưng nhớ lại tình huống thảo luận ở chương 2: ghép hai kênh lệch pha có thể triệt tiêu nhau. Nếu ghép xong thấy mất trầm thì không phải mã sai — đó là hiện tượng vật lý đã học.

3.5.4 Bảng tra nhanh — nên chép ra một trang riêng

Toàn bộ mục 3.5 gom lại vừa một bảng. Từ chương 4 trở đi, mỗi khi viết một hàm xử lý âm thanh, bốn dòng dưới đây là bốn câu hỏi cần trả lời trước khi viết dòng mã đầu tiên.

Bảng 3.6 Bốn quy ước dùng ở mọi chương sau

Câu hỏi	Quy ước của sách	Vì sao
Mẫu nằm trong khoảng nào?	Số thực trong [−1,0 ; +1,0]	Vượt tạm khi tính trung gian vẫn đúng; chỉ kẹp ở bước ghi cuối
Đo mức bằng thang nào?	dBFS, mốc là 1,0 nên luôn âm	Thấy số dương là biết đã tràn
Mảng có hình dạng nào?	Luôn (N,) hoặc (N, số kênh)	Đọc bằng always_2d=True thì không phải viết hai nhánh mã
Hạ về số nguyên khi nào?	*Chỉ* ở bước ghi tệp cuối cùng	Mỗi lần hạ là một lần làm tròn; hạ giữa chừng thì sai số cộng dồn

Bốn dòng này giải quyết phần lớn lỗi của người mới

"Vì sao chuẩn hoá lên 1,0 mà vẫn nghe nhỏ?" — vì đang lẫn giữa biên độ mẫu và độ to cảm nhận. Xem lại dòng thứ nhất và mục 2.6.1.

"Vì sao cộng hai tệp lại nghe bị méo?" — vì tổng vượt 1,0 và bị kẹp ở bước ghi. Cộng xong phải chia hoặc chuẩn hoá lại.

"Vì sao mã chạy được với tệp này mà lỗi với tệp kia?" — gần như luôn là chuyện mono và stereo. Xem lại dòng thứ ba.

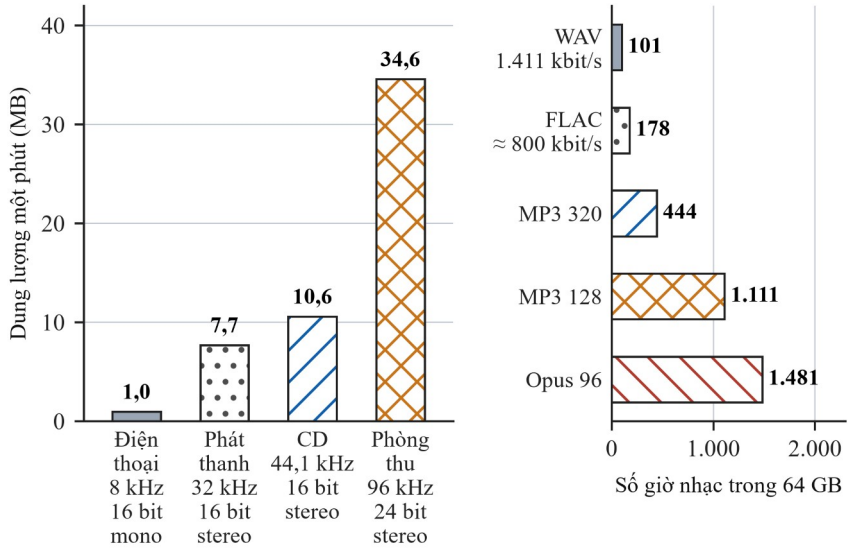
"Vì sao qua nhiều bước xử lý thì nghe nhiễu nền tăng dần?" — vì đang hạ về số nguyên ở mỗi bước. Xem lại dòng thứ tư.

3.6 Dung lượng, định dạng và nén có tổn hao

3.6.1 Dung lượng của âm thanh không nén

Đã có đủ tham số thì tính dung lượng chỉ là một phép nhân.

$$S = f_s \cdot \frac{b}{8} \cdot N_{\text{ch}} \cdot t \quad [\text{byte}] \tag{3.9}$$



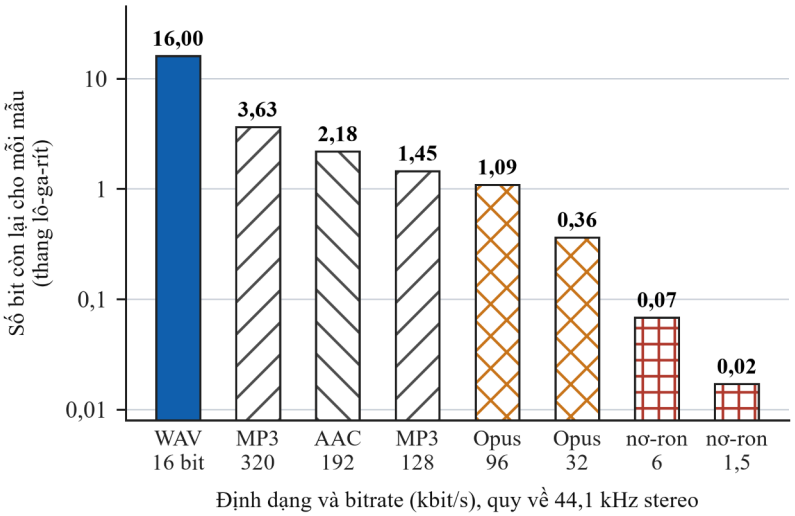
(a) Dung lượng không nén (b) Cùng một ổ 64 GB

Hình 3.24 Dung lượng một phút ở các cấu hình khác nhau, và số giờ chứa được trong một ổ 64 GB

Một phút âm thanh chất lượng CD có dung lượng 10,58 MB — tưởng chưa lớn. Tuy nhiên, một ổ 64 GB chỉ chứa được 101 giờ ở dạng ấy, so với 1.111 giờ nếu nén ở MP3 128 kbit/s. Chính khoảng cách hơn mười một lần này là lý do tồn tại của mọi kỹ thuật nén.

$$R = f_s \cdot b \cdot N_{\text{ch}} \quad [\text{bit/s}]\,; \qquad b_{\text{mẫu}} = \frac{R_{\text{coded}}}{f_s \cdot N_{\text{ch}}} \tag{3.10}$$

Công thức thứ hai đảo ngược câu hỏi: cho trước bitrate thì mỗi mẫu còn được bao nhiêu bit? Con số này cho thấy tầm quan trọng của nén rõ hơn mọi tỉ lệ phần trăm.



Hình 3.25 Số bit còn lại cho mỗi mẫu, từ WAV không nén tới codec nơ-ron

MP3 128 kbit/s chỉ còn 1,45 bit cho mỗi mẫu, so với 16 bit của bản gốc — nhỏ hơn mười một lần. Tuy vậy, phần lớn người nghe không phân biệt được. Điều này chỉ khả thi nếu chúng ta biết chắc phần bị bỏ đi là phần tai không dùng tới.

3.6.2 Ba nhóm định dạng, phân biệt bằng một câu hỏi



Hình 3.26 Ba nhóm định dạng và câu hỏi phân biệt chúng

Bảng 3.7 Ba nhóm định dạng — so sánh

Nhóm	Ví dụ	Giảm bao nhiêu	Khôi phục được không	Dùng khi nào
Không nén	WAV, *aiff*	0 %	Không cần — chưa mất gì	Thu và xử lý
Nén không tổn hao	FLAC, *alac*	40 – 60 %	*Có* — chính xác từng mẫu	Lưu trữ lâu dài
Nén có tổn hao	MP3, AAC, Opus	khoảng 90 %	*Không* — mất vĩnh viễn	Phát hành, phát trực tuyến

Hai ẩn dụ để nhớ

Nén không tổn hao là gói vali: cuộn quần áo lại, xếp khít các khoảng trống. Vali nhẹ và gọn hơn, nhưng mở ra vẫn đủ từng món.

Nén có tổn hao là tóm tắt sách: bản tóm tắt ngắn hơn nhiều lần và giữ trọn ý chính, nhưng không thể dựng lại nguyên văn từng câu đã lược bỏ.

Quy tắc nghề rút ra: *Thu* không nén → *lưu trữ* không tổn hao → *phát hành* có tổn hao, một lần duy nhất, ở bước cuối.

3.6.3 Vì sao nén có tổn hao lại giảm được tới chín mươi phần trăm

Câu trả lời không nằm ở một thuật toán khéo léo hơn, mà nằm ở chính giới hạn của tai người. Nén có tổn hao là kỹ thuật loại bỏ đúng những thành phần mà tai vốn không nghe thấy — và ba điểm mù cho phép làm điều đó đều đã học ở chương 1.

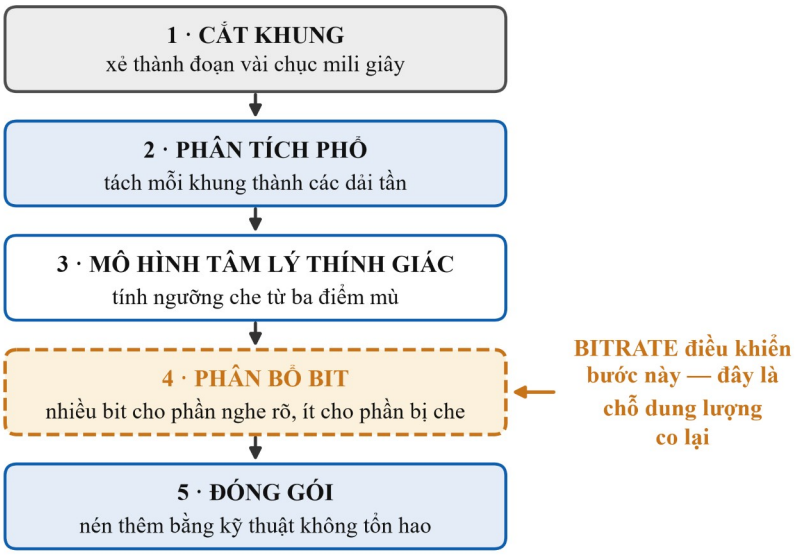


Hình 3.27 Ba điểm mù của tai và mức tiết kiệm mà mỗi điểm mang lại

Điểm mù thứ hai — che lấp theo tần số — là cốt lõi của toàn bộ kỹ thuật. Trong một bản nhạc thật, tại mỗi thời điểm luôn có vô số âm nhỏ bị các âm to che lấp. Tai vốn không nghe thấy chúng, nên bộ mã hoá loại bỏ chúng mà người nghe không nhận ra khác biệt nào. Đây là chỗ dung lượng co lại mạnh nhất, nhiều hơn hẳn so với chỉ dựa vào đường cong ngưỡng nghe tĩnh.

Điểm mù thứ ba — che lấp theo thời gian — có một chi tiết phản trực giác đáng nhắc lại: tai bị che không chỉ *sau* một tiếng lớn (khoảng 100 – 200 mili giây) mà còn *trước* nó khoảng 2 – 5 mili giây. Hiệu ứng che trước nghe có vẻ phi lý nhưng có thật, và nguyên nhân nằm ở cách hệ thính giác, tức tai cùng não, xử lý chứ không chỉ ở tai.

3.6.4 Bên trong một bộ mã hoá

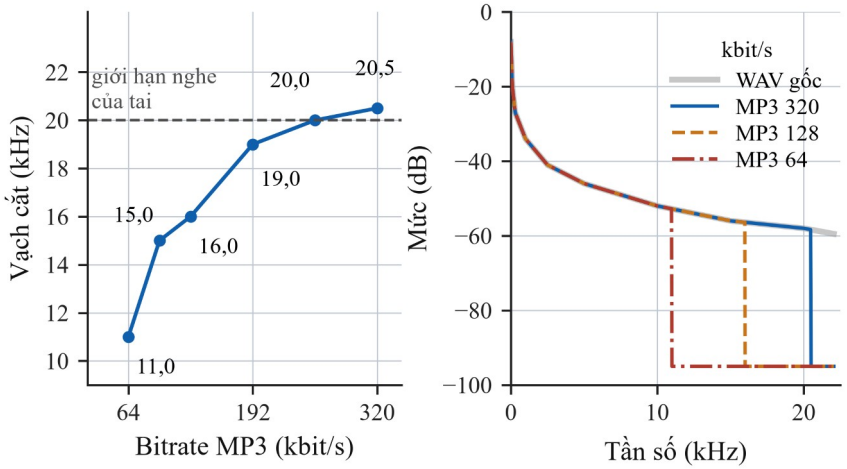


Hình 3.28 Năm bước bên trong một bộ mã hoá MP3 hoặc AAC

Bước thứ hai — phân tích ra phổ tần số — chính là nội dung của chương 4. Bước thứ ba dùng ba điểm mù vừa nói. Và bước thứ tư là chỗ tham số *bitrate* tác động: bitrate chính là ngân sách bit mà bộ mã hoá được phép tiêu cho mỗi giây.

3.6.5 Dấu vết của nén có tổn hao

Vì tai ít nhạy nhất ở dải cao, khi thiếu bit thì bộ mã hoá cắt bỏ phần cao nhất của phổ trước tiên. Trên phổ đồ, điều này hiện thành một đường ranh giới ngang rất sắc — cắt dứt khoát như một bức tường.



(a) Bitrate càng thấp, vạch cắt càng tụt

(b) Vạch cắt nhìn trên phổ

Hình 3.29 Vị trí vạch cắt theo bitrate, và hình dạng của nó trên phổ

Bảng 3.8 Vạch cắt tiêu biểu theo bitrate MP3

Bitrate	Vạch cắt ước lượng	Nhận xét
64 kbit/s	khoảng 11,0 kHz	nghe rõ mất dải cao, tiếng đục
128 kbit/s	khoảng 16,0 kHz	nhận ra nếu nghe kỹ bằng tai nghe tốt
192 kbit/s	khoảng 19,0 kHz	khó nhận ra với phần lớn người nghe
320 kbit/s	khoảng 20,5 kHz	gần như trong suốt

Ứng dụng: Phát hiện tệp "lossless giả"

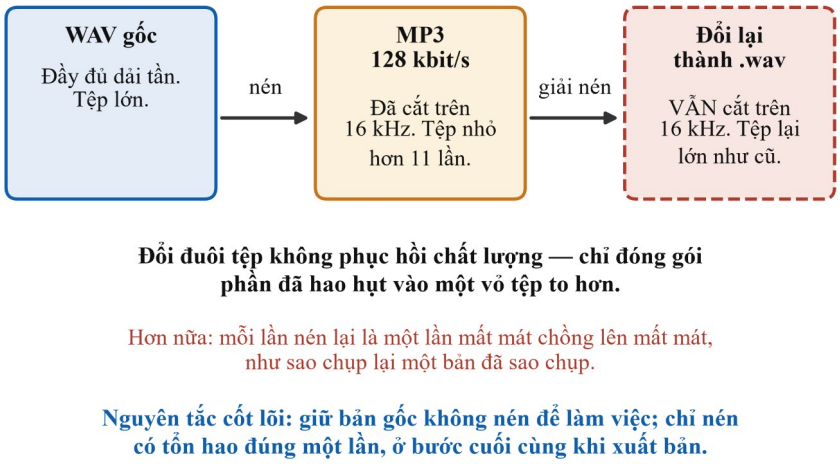
Một tệp đặt đuôi .flac nhưng có vạch cắt sắc ở 16 kHz thì chắc chắn được dựng lại từ nguồn MP3 128 kbit/s.

Dữ liệu tần cao đã mất từ lần nén trước; đóng gói lại vào vỏ FLAC chỉ làm tệp lớn hơn, không phục hồi được gì.

Đây là bài thực hành 2 của chương này, và là một kỹ năng dùng được ngay khi mua hoặc nhận tệp âm thanh từ người khác.

Lưu ý một trường hợp ngoại lệ: bản thu gốc từ nguồn cũ — băng từ, đĩa than số hoá lâu năm — cũng có thể thiếu dải cao. Vạch cắt dứt khoát như bức tường là dấu hiệu của nén; dải cao thoải dần thì thường là do nguồn.

3.6.6 Nén có tổn hao không hoàn tác được



Hình 3.30 Đổi đuôi tệp không phục hồi chất lượng

Đổi một tệp MP3 thành .wav làm dung lượng tăng từ 0,96 MB lên 10,58 MB — gấp 11 lần — mà chất lượng vẫn đúng bằng bản MP3. Tệ hơn, mỗi lần nén lại là một lần mất mát chồng lên mất mát, tương tự sao chụp lại một bản đã sao chụp.

Ba quy tắc cùng một dạng

Chương 2 — gain staging: đặt mức đúng *ngay từ đầu* chuỗi, vì khuếch đại về sau không sửa được.

Mục 3.4 — dither: dither *đúng một lần*, ở bước hạ bit cuối cùng.

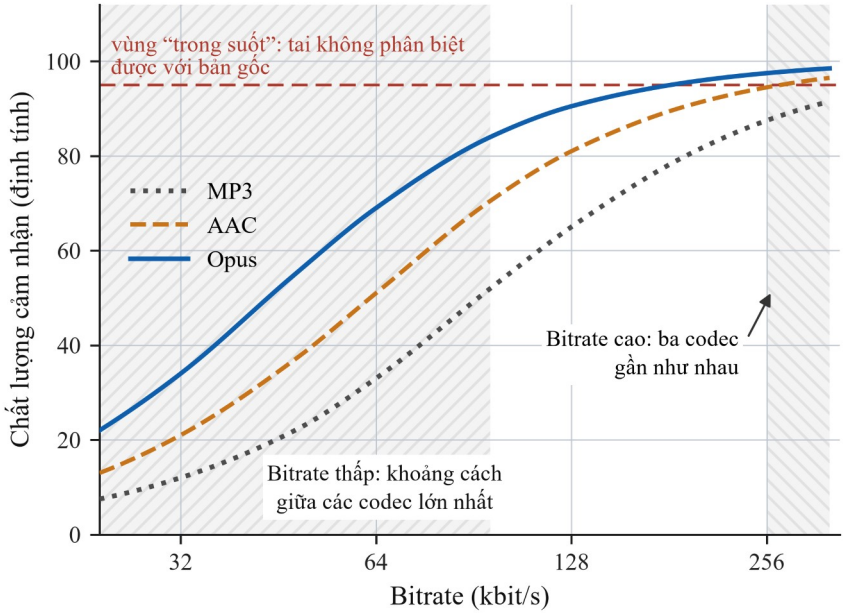
Mục này — nén có tổn hao: nén *đúng một lần*, ở bước xuất bản cuối cùng.

Ba quy tắc, cùng một nguyên tắc: mọi phép biến đổi *không* hoàn tác được đều nên dồn về đúng một lần, càng muộn càng tốt. Nguyên tắc này sẽ còn gặp lại ở chương 7 và chương 10.

3.7 Codec nơ-ron

3.7.1 So sánh ba codec cổ điển trước đã

Trước khi bàn tới hướng mới, cần đặt đúng chỗ ba codec đang dùng nhiều nhất. Với cùng một ngân sách bit, codec mới hơn sử dụng ngân sách bit hiệu quả hơn — nhưng khoảng cách giữa chúng không đều theo bitrate.



Hình 3.31 Chất lượng cảm nhận theo bitrate — dạng định tính của ba codec

Bảng 3.9 Ba codec có tổn hao phổ biến nhất

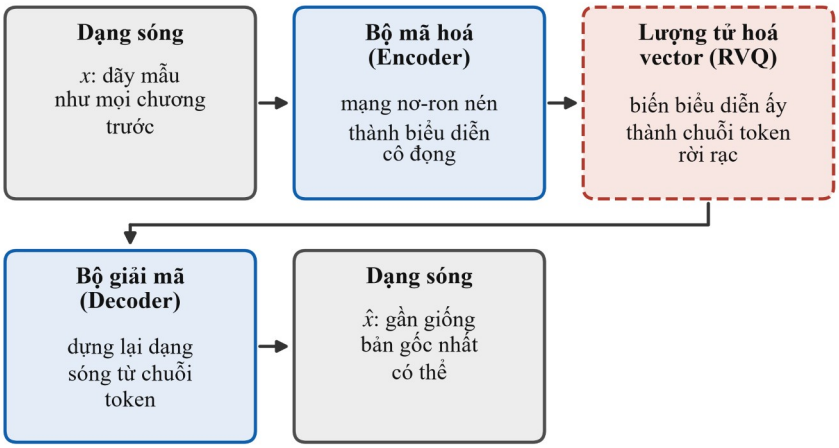
Codec	Điểm mạnh	Điểm yếu	Dùng ở đâu
MP3	Tương thích rộng nhất, không có đối thủ về mặt này	Kém hẳn ở bitrate thấp	Kho nhạc cũ, thiết bị đời trước
AAC	Hiệu quả hơn MP3 ở cùng bitrate	Cần bản quyền trong một số ngữ cảnh	Video, phát trực tuyến, thiết bị Apple
Opus	Mạnh nhất ở bitrate thấp; độ trễ rất thấp; mở hoàn toàn	Ít thiết bị phần cứng cũ hỗ trợ	Gọi thoại thời gian thực, nền web

Điều đáng nhớ không phải thứ hạng mà là *hình dạng* của đường cong: chênh lệch giữa các codec lớn nhất ở bitrate thấp và gần như biến mất từ khoảng 256 kbit/s trở lên. Đó là lý do MP3 dù đã cũ vẫn còn dùng được — ở bitrate cao nó vẫn đủ tốt.

3.7.2 Codec nơ-ron: đổi từ điển cố định lấy từ điển tự học

Các codec vừa nói đều dựa trên những quy tắc do con người thiết kế sẵn: một mô hình tâm lý thính giác cố định, viết ra từ các thí nghiệm nghe. Từ khoảng năm 2021 xuất hiện một hướng đảo ngược cách tiếp cận: để một mạng nơ-ron *tự học* cách biểu diễn âm thanh thật gọn, từ dữ liệu, thay cho quy tắc thủ công. Loại codec này gọi là codec nơ-ron (tiếng Anh: neural audio codec): codec dùng mạng nơ-ron học sâu để mã hoá và giải mã.

$$x \longrightarrow \text{Enc} \longrightarrow z \longrightarrow \text{RVQ} \longrightarrow [q_1, \ldots, q_K] \longrightarrow \text{Dec} \longrightarrow \hat{x} \qquad (3.11)$$

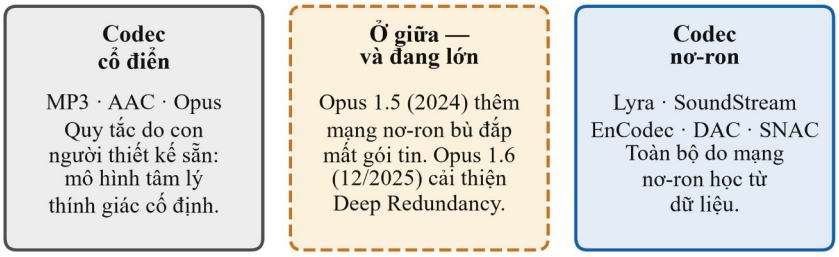


Hình 3.32 Ba khối của một codec nơ-ron: mã hoá, lượng tử hoá vector, giải mã

Điểm mấu chốt để nối với phần đã học: đây *cùng* một ý tưởng với lượng tử hoá ở mục 3.4 — biểu diễn một thứ liên tục bằng một tập hữu hạn ký hiệu. Khác biệt duy nhất là tập ký hiệu ấy. Lượng tử hoá cổ điển dùng các mức biên độ *cố định*, cách đều nhau. Codec nơ-ron dùng một từ điển token (ký hiệu rời rạc) do mạng tự học, và các token ấy không cách đều nhau mà tập trung vào những dạng âm hay gặp.

3.7.3 Một cập nhật: ranh giới đang mờ đi

Cách trình bày "codec cổ điển đối lập codec nơ-ron" tiện để bắt đầu, nhưng đến thời điểm viết sách này nó đã không còn mô tả đúng thực tế.



Hình 3.33 Ranh giới giữa codec cổ điển và codec nơ-ron đang mờ đi

Bảng 3.10 Vài mốc đáng chú ý của Opus

Phiên bản	Thời điểm	Điều đáng chú ý
Opus 1.5	tháng 3 năm 2024	Thêm mạng nơ-ron bù đắp mất gói tin, và Deep Redundancy — một bộ tự mã hoá biến phân nhúng dữ liệu khôi phục vào từng gói
Opus 1.6	tháng 12 năm 2025	Cải thiện Deep Redundancy, thêm mô-đun mở rộng băng thông, hỗ trợ thử nghiệm 96 kHz
Bản thảo Opus HD	tháng 10 năm 2025	Đề xuất tại IETF, mở rộng tới 96 kHz — nhưng đây là kỹ thuật xử lý tín hiệu cổ điển, *không* dùng học máy

Điều đáng rút ra — và đây là một kỹ năng, không phải một danh sách

Opus — một codec vẫn được xếp vào nhóm "cổ điển" — nay đã có thành phần mạng nơ-ron bên trong. Deep Redundancy của nó là một trong những nỗ lực đầu tiên chuẩn hoá thuật toán học sâu tại IETF.

Vì vậy không nên học thuộc "hai phe". Điều đáng nhớ là một *trục*: trong một codec, bao nhiêu phần do con người thiết kế và bao nhiêu phần do máy học từ dữ liệu. Trục ấy vẫn đúng khi những cái tên trong bài này đã cũ.

Và một câu hỏi dùng được với mọi codec chưa ra đời: nó đánh đổi cái gì lấy cái gì, và ở bitrate nào thì đánh đổi ấy có lợi?

Mọi mốc thời gian trong mục này tính tới tháng 9 năm 2026. Người đọc ở thời điểm muộn hơn nên kiểm lại — nhưng cách đặt câu hỏi thì không cũ.

3.7.4 Vì sao phần này lại nằm trong chương về số hoá



Hiểu nguyên lý nén hôm nay chính là bước chuẩn bị cho phần AI tạo sinh ở cuối học phần.

Không phải trùng hợp: cả hai đều trả lời cùng một câu hỏi — biểu diễn âm thanh thật gọn thì biểu diễn thế nào.

Hình 3.34 Chuỗi token: vừa để nén, vừa là ngôn ngữ cho AI tạo sinh

Chuỗi token mà codec nơ-ron sinh ra không chỉ dùng để nén. Nó còn là thứ mà các mô hình AI tạo sinh nhạc và giọng làm việc trên đó: mô hình sinh ra một chuỗi token, rồi bộ giải mã dựng nó thành âm thanh. Đó chính là nội dung chương 13 và 14.

Không phải trùng hợp. Nén và sinh đều trả lời cùng một câu hỏi: biểu diễn âm thanh thật gọn thì biểu diễn thế nào. Hiểu nguyên lý nén ở chương này chính là bước chuẩn bị cho phần AI ở cuối cuốn sách.

3.7.5 Cách đọc một thông báo về codec mới

Mục 3.7.3 đã nói rằng danh sách tên trong bài này sẽ cũ. Vậy khi gặp một codec chưa từng nghe tên — điều chắc chắn xảy ra trong vài năm tới — thì đánh giá nó thế nào? Năm câu hỏi dưới đây dùng được với mọi codec, kể cả những codec chưa ra đời.

- **Nó chạy ở bitrate nào, và so ở đúng bitrate ấy chưa?** Đây là câu hỏi bị bỏ qua nhiều nhất. Một codec "tốt hơn MP3" ở 32 kbit/s có thể không hơn gì ở 256 kbit/s — xem lại hình dạng đường cong ở mục 3.7.1.
- **Nó đo chất lượng bằng cách nào?** Bằng phép thử nghe có người thật, hay bằng một chỉ số tính máy? Chỉ số tính máy tiện nhưng có thể tối ưu ngược — mô hình học cách cải thiện chỉ số nhưng không cải thiện chất lượng nghe thực tế.
- **Nó tốn bao nhiêu tính toán, và ở đâu?** Codec nơ-ron thường đòi hỏi nhiều tính toán hơn codec cổ điển nhiều lần. Chạy trên máy chủ thì không sao; chạy trên điện thoại hoặc thiết bị nhúng thì đó là yếu tố quyết định.
- **Độ trễ của nó là bao nhiêu?** Nén hiệu quả thường cần nhìn trước một khoảng tín hiệu. Với nghe nhạc thì độ trễ vài trăm mili giây không sao; với gọi thoại thì không dùng được — đây đúng là đánh đổi đã học ở mục 2.6.2.
- **Ai dùng được nó?** Một codec chỉ giải mã được bằng phần mềm mới, trên phần cứng mới, có ràng buộc bản quyền, thì dù tốt tới đâu cũng khó thay thế MP3 trong thực tế. Đây là lý do MP3 vẫn được dùng sau ba mươi năm.

> **Vì sao dạy cách hỏi thay vì dạy danh sách**
>
> Danh sách tên codec trong bất kỳ cuốn sách nào cũng cũ sau vài năm. Năm câu hỏi trên thì không.
>
> Chúng cũng là năm câu hỏi mà một kỹ sư thực hành phải trả lời khi chọn codec cho một sản phẩm — không phải "cái nào tốt nhất" mà "cái nào phù hợp với ràng buộc của mình".
>
> Cách đặt câu hỏi này sẽ được dùng lại ở chương 13 và 14 khi đánh giá các mô hình AI tạo sinh — cũng là một lĩnh vực mà danh sách tên cũ đi rất nhanh.

3.8 Hướng dẫn thực hành

3.8.1 Tái hiện chồng phổ

Bài này cho thấy tận mắt điều mục 3.3 mô tả: một tần số cao bị lấy mẫu thưa sẽ biến thành tần số thấp giả, và không có cách nào phân biệt với tần số thấp thật.

Liệt kê 3.1 chong_pho.py — sinh hai sóng, lấy mẫu thưa, so phổ

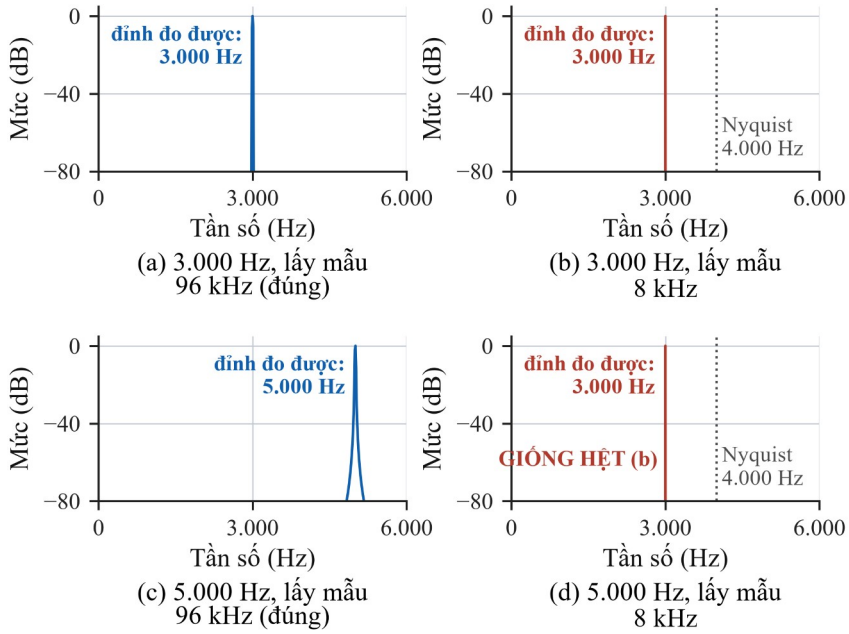
```
import numpy as np
import soundfile as sf

SR_CAO, SR_THAP = 96000, 8000
DUR = 1.0

def sin_o(f, sr):
    t = np.arange(int(DUR * sr)) / sr
    return 0.5 * np.sin(2 * np.pi * f * t)

def dinh_pho(x, sr):
    w = np.hanning(len(x))
    X = np.abs(np.fft.rfft(x * w))
    f = np.fft.rfftfreq(len(x), 1 / sr)
    return f[np.argmax(X)]

for f0 in (3000.0, 5000.0):
    cao  = sin_o(f0, SR_CAO)     # lấy mẫu dày – đúng
    thap = sin_o(f0, SR_THAP)    # lấy mẫu thưa – bị gập
    print(f"{f0:6.0f} Hz | ở 96 kHz đo ra {dinh_pho(cao, SR_CAO):6.0f} Hz"
          f" | ở 8 kHz đo ra {dinh_pho(thap, SR_THAP):6.0f} Hz")
    sf.write(f"thap_{int(f0)}.wav", thap, SR_THAP)
```



Hình 3.35 Kết quả phải thấy: hai sóng khác nhau cho ra phổ giống hệt nhau

Việc phải làm sau khi chạy

Nghe hai tệp thap_3000.wav và thap_5000.wav. Chúng nghe *giống nhau*, dù một cái đáng lẽ phải cao hơn hẳn.

Mở cả hai bằng Audacity ở chế độ Spectrogram — cùng một vạch ở 3.000 Hz.

Đây không phải hạn chế của phần mềm. Hai tệp chứa những dãy số giống hệt nhau; không có thông tin nào để phân biệt.

3.8.2 Hạ độ sâu bit, có và không có dither

Liệt kê 3.2 ha_bit.py — lượng tử hoá một đoạn nhỏ, hai cách

```
import numpy as np
import soundfile as sf

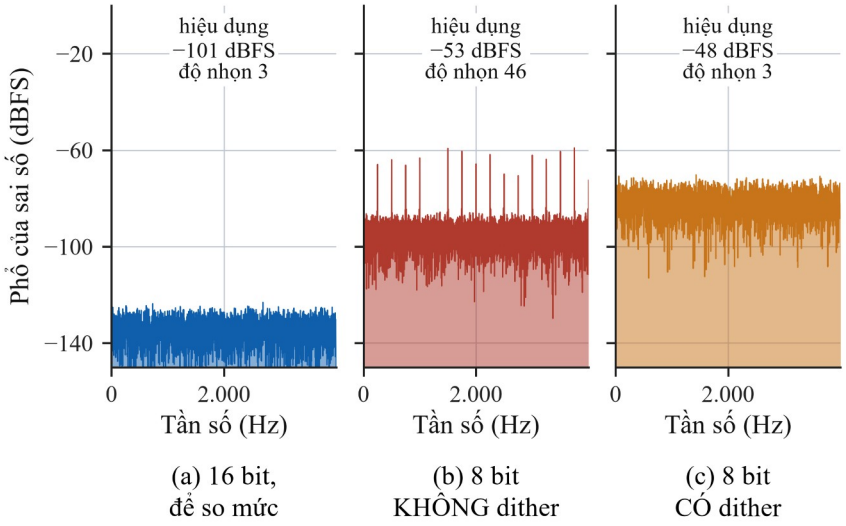
SR, DUR, BIT = 44100, 3.0, 8
t = np.arange(int(DUR * SR)) / SR

# đoạn NHỎ – khoảng -34 dBFS, đúng chỗ dither quan trọng
x = 0.02 * (np.sin(2 * np.pi * 220 * t)
            + 0.5 * np.sin(2 * np.pi * 660 * t))

buoc = 2.0 / (2 ** BIT)          # bước lượng tử
rng  = np.random.default_rng(0)
tpdf = buoc * (rng.random(len(x)) - rng.random(len(x)))

khong = np.round(x / buoc) * buoc            # không dither
co    = np.round((x + tpdf) / buoc) * buoc   # có dither

for ten, y in (("khong_dither", khong), ("co_dither", co)):
    e = y - x
    print(ten, "| sai số hiệu dụng =",
          round(20 * np.log10(max(np.sqrt(np.mean(e ** 2)), 1e-12)), 1),
          "dBFS")
    sf.write(f"{ten}.wav", y, SR)
```



Hình 3.36 Phổ của riêng sai số ở ba trường hợp, trên cùng một thang tuyệt đối

Nghe hai tệp bằng tai nghe, tăng âm lượng. Bản không dither có một thứ nghe như tiếng lạo xạo bám theo giai điệu; bản có dither có tiếng nhiễu nền đều, to hơn một chút nhưng dễ chịu hơn nhiều. Đó chính là sự khác nhau giữa *méo* và *nhiễu*.

3.8.3 Nhận diện tệp "lossless giả"

Bài này không cần viết mã, nhưng là kỹ năng dùng được ngay.

- Lấy một tệp WAV gốc. Dùng Audacity hoặc ffmpeg xuất ra MP3 128 kbit/s và MP3 320 kbit/s.
- Mở lần lượt ba tệp bằng Audacity ở chế độ Spectrogram, hoặc bằng Spek.
- Quan sát: bản WAV có năng lượng trải đều tới khoảng 20 – 22 kHz; bản 320 cắt nhẹ quanh 20 kHz; bản 128 có vạch cắt sắc quanh 16 kHz.
- Bây giờ đổi bản MP3 128 trở lại thành WAV rồi mở bằng Spek: vạch cắt 16 kHz *vẫn còn*. Đó chính là "lossless giả".

Một lưu ý để không kết luận vội

Thiếu dải cao không phải lúc nào cũng do nén. Bản thu từ nguồn cũ — băng từ, đĩa than, thiết bị đời trước — cũng thiếu dải cao một cách tự nhiên.

Dấu hiệu phân biệt nằm ở *hình dạng* chỗ cắt: nén để lại một đường ranh giới ngang rất sắc, dựng đứng như bức tường. Nguồn cũ để lại một đường thoải dần.

Nguyên tắc chung khi đọc phổ: chú ý hình dạng trước, chú ý con số sau.

3.8.4 Ba lỗi hay gặp khi chạy hai đoạn mã trên

Bảng 3.11 Chẩn đoán nhanh khi kết quả không như hình

Triệu chứng	Nguyên nhân thường gặp	Cách sửa
Đỉnh phổ đo ra lệch vài chục héc so với con số trong bài	Độ phân giải tần số của phép đo có hạn: nó bằng tần số lấy mẫu chia số mẫu	Kéo dài thời lượng tín hiệu; một giây ở 8 kHz cho độ phân giải 1 Hz — đủ chính xác
Phổ có nhiều vạch phụ quanh đỉnh chính	Chưa nhân cửa sổ Hanning trước khi biến đổi	Thêm dòng nhân cửa sổ; chương 4 sẽ giải thích vì sao nó cần thiết
Tệp ghi ra méo tiếng, dạng sóng có đỉnh bị cắt phẳng	Biên độ vượt 1,0 nên bị kẹp ở bước ghi — đúng lỗi của mục 3.5.2	Kiểm np.max(np.abs(x)) trước khi ghi; chia bớt hoặc chuẩn hoá
Hai tệp dither và không dither nghe giống hệt nhau	Nghe qua loa máy tính trong phòng ồn; hoặc quên tăng âm lượng	Nghe bằng tai nghe, tăng âm lượng. Khác biệt nằm ở nền rất nhỏ

Dòng đầu bảng đáng chú ý vì nó không phải lỗi: độ phân giải tần số của một phép đo phổ bằng tần số lấy mẫu chia cho số mẫu. Đo một đoạn ngắn thì đỉnh phổ nhất định lệch đôi chút. Đây là ràng buộc thật của phép biến đổi, và chương 4 sẽ nói kỹ — cùng với cái giá phải trả khi muốn độ phân giải cao hơn.

3.9 Bài tập, thảo luận và tổng kết

3.9.1 Bài tập tính toán

Bảy bài dưới đây đều giải được bằng các công thức của chương. Yêu cầu trình bày phép tính, không chỉ ghi kết quả.

Bài 1 — Nyquist và chồng phổ

Một hệ thống lấy mẫu ở 16.000 Hz.

- a) Tần số Nyquist là bao nhiêu?
- b) Một sóng 6.000 Hz đi qua hệ ấy sẽ đo ra bao nhiêu?
- c) Một sóng 12.000 Hz thì sao?
- d) Có tần số nào khác cũng cho ra cùng kết quả với câu c không?
- *Đáp số: a) 8.000 Hz. b) 6.000 Hz — dưới Nyquist nên đúng. c) 4.000 Hz — đã bị gập. d) có, và có vô số: 4.000 Hz thật, rồi 20.000, 28.000 Hz… Mọi tần số cách nhau đúng một bội của 16.000 Hz đều cho ra cùng một dãy số.*

Bài 2 — chọn tần số lấy mẫu

Một dự án cần ghi tiếng dơi, có thành phần tới 45 kHz.

- a) Tần số lấy mẫu tối thiểu theo lý thuyết là bao nhiêu?
- b) Trong các chuẩn thông dụng, nên chọn con số nào và vì sao?

- c) Nếu vẫn thu ở 48 kHz thì thành phần 45 kHz sẽ hiện ra ở đâu?
- *Đáp số: a) 90.000 Hz. b) 192.000 Hz — 96.000 Hz tuy vượt ngưỡng lý thuyết nhưng Nyquist của nó chỉ 48 kHz, để lại quá ít khoảng cho bộ lọc suy giảm. c) gập về 3.000 Hz — nằm giữa dải nghe, và sẽ nghe rất rõ.*

Bài 3 — độ sâu bit và dải động

So sánh hai cấu hình thu: 16 bit và 24 bit.

- a) Mỗi cấu hình cho bao nhiêu mức và bao nhiêu dB dải động?
- b) Chênh lệch dải động giữa hai cấu hình là bao nhiêu?
- c) Nếu đặt mức danh định ở −18 dBFS thì mỗi cấu hình còn cách sàn nhiễu bao nhiêu dB?
- *Đáp số: a) 16 bit: 65.536 mức, 98,1 dB. 24 bit: 16.777.216 mức, 146,2 dB. b) 48,2 dB — đúng 8 bit nhân 6,02. c) 16 bit còn 80,1 dB; 24 bit còn 128,2 dB. Đây là lý do 24 bit cho phép đặt mức linh hoạt hơn nhiều.*

Bài 4 — dBFS

Một đoạn tín hiệu có mẫu đỉnh bằng 0,35.

- a) Mức đỉnh là bao nhiêu dBFS?
- b) Còn bao nhiêu headroom tới trần?
- c) Nhân mọi mẫu với 2 thì mức đỉnh thành bao nhiêu?
- d) Nhân với 3 thì sao — có tràn không?
- *Đáp số: a) −9,12 dBFS. b) 9,12 dB. c) −3,10 dBFS — tăng đúng 6,02 dB như quy tắc ở chương 1. d) đỉnh thành 1,05, tức 0,42 dBFS — số dương, nghĩa là đã tràn. Ghi ra tệp số nguyên sẽ bị kẹp và méo.*

Bài 5 — dung lượng

Một buổi thu podcast dài 90 phút, thu ở 48.000 Hz, 24 bit, hai kênh.

- a) Dung lượng tệp WAV là bao nhiêu megabyte?
- b) Bitrate của nó là bao nhiêu?
- c) Nếu xuất ra MP3 128 kbit/s thì có dung lượng bao nhiêu, giảm mấy lần?
- d) Một ổ 512 GB chứa được bao nhiêu buổi thu như thế ở dạng WAV?
- *Đáp số: a) 1.555 MB. b) 2.304 kbit/s. c) 86 MB, giảm 18 lần. d) khoảng 329 buổi thu — tưởng là nhiều, nhưng một dự án phim tài liệu dễ dàng vượt con số ấy.*

Bài 6 — bit cho mỗi mẫu

Một tệp Opus 64 kbit/s, stereo, tần số lấy mẫu 48.000 Hz.

- a) Mỗi mẫu còn được bao nhiêu bit?
- b) So với bản gốc 24 bit thì nhỏ hơn bao nhiêu lần?
- c) Con số ở câu a nhỏ hơn 1. Điều đó có mâu thuẫn không? Giải thích.
- *Đáp số: a) 0,667 bit. b) nhỏ hơn 36 lần. c) không mâu thuẫn. Bộ mã hoá không mã hoá từng mẫu riêng lẻ — nó mã hoá cả một khung vài chục mili*

giây một lúc, trong miền tần số. Con số "bit mỗi mẫu" chỉ là phép chia trung bình để dễ cảm nhận, không mô tả cách làm việc thật của codec.

Bài 7 — tổng hợp: chọn cấu hình cho một dự án

Một nhóm làm phim tài liệu: thu phỏng vấn ngoài hiện trường, dựng trên máy tính, phát hành trên nền tảng trực tuyến. Hãy chọn tham số cho ba khâu và giải thích từng lựa chọn.

- a) Khâu *thu*: tần số lấy mẫu, độ sâu bit, định dạng. Vì sao?
- b) Khâu *xử lý*: dữ liệu nên ở dạng gì trong bộ nhớ? Có dither ở khâu này không?
- c) Khâu *phát hành*: định dạng, bitrate, có dither không, dither ở bước nào?
- d) Nếu sau khi phát hành nhóm muốn dựng lại một cảnh, họ lấy tệp nào ra làm? Vì sao không lấy tệp đã phát hành?
- *Đáp số gợi ý: a) 48.000 Hz vì chia chẵn cho tốc độ khung hình; 24 bit vì ngoài hiện trường không kiểm soát được đỉnh nên cần dải động dư; WAV không nén. b) số thực trong bộ nhớ, không dither ở khâu này — dither giữa chừng chỉ làm nhiễu cộng dồn. c) AAC hoặc Opus, khoảng 128 – 192 kbit/s cho tiếng nói; có dither, và đúng một lần tại bước hạ từ số thực xuống 16 bit ngay trước khi mã hoá. d) lấy bản WAV gốc. Tệp đã phát hành đã mất phần dải cao và đã qua dither; dựng lại từ nó là chồng mất mát lên mất mát.*

3.9.2 Câu hỏi ôn tập

- 1. Số hoá gồm hai phép rời rạc hoá nào? Mỗi phép cắt trục nào?
- 2. Hai phép ấy do tham số nào điều khiển?
- 3. Phát biểu định lý Nyquist–Shannon.
- 4. Vì sao chuẩn CD chọn 44.100 Hz mà không phải đúng 40.000 Hz?
- 5. Nêu trực giác hình học của định lý — vì sao cần hai điểm mỗi chu kỳ?
- 6. Định lý nói dựng lại tín hiệu bằng cách nào? Nối chấm bằng đoạn thẳng có đúng không?
- 7. Hàm sinc có tính chất gì khiến tổng của chúng đi đúng qua mọi mẫu?
- 8. Chồng phổ là gì? Vì sao phải chặn nó *trước* khi lấy mẫu?
- 9. Tính tần số thấy được của 12.000 Hz khi lấy mẫu ở 16.000 Hz.
- 10. Vì sao ba tần số khác nhau có thể cho ra cùng một dãy số?
- 11. Kể ba hiện tượng đời thường cũng là chồng phổ.
- 12. Bộ lọc chống chồng phổ nằm ở đâu trong chuỗi tín hiệu của chương 2?
- 13. Vì sao tăng tần số lấy mẫu không làm âm nghe hay hơn? Nó có ích ở đâu?
- 14. Độ sâu bit quyết định gì? Nêu công thức số mức và công thức dải động.
- 15. Vì sao thu ở 24 bit khi phát hành chỉ cần 16 bit?
- 16. Nhiễu lượng tử khác nhiễu nhiệt của chương 2 ở điểm nào?
- 17. Dither làm gì? Nó có làm sai số nhỏ đi không?
- 18. Dither nên dùng ở bước nào và bao nhiêu lần?
- 19. dBFS lấy mốc ở đâu? Vì sao mọi con số dBFS hợp lệ đều âm?

- 20. Nêu nguyên tắc cốt lõi về int và float. Giải thích hai lý do của nó.
- 21. Dải int16 chạy từ đâu tới đâu? Vì sao nó không đối xứng?
- 22. Mảng stereo có hình dạng thế nào? always_2d=True giải quyết vấn đề gì?
- 23. Viết công thức dung lượng tệp WAV và công thức bitrate.
- 24. Phân biệt ba nhóm định dạng bằng một câu hỏi duy nhất.
- 25. Nêu ba điểm mù của tai. Điểm mù nào cho tiết kiệm lớn nhất?
- 26. Mô tả năm bước của một bộ mã hoá MP3. Bitrate tác động ở bước nào?
- 27. "Lossless giả" là gì? Nhận diện bằng dấu hiệu nào, và cần thận trọng với trường hợp nào?
- 28. Vì sao đổi MP3 thành .wav không làm chất lượng tốt hơn?
- 29. Codec nơ-ron khác codec cổ điển ở chỗ nào về nguyên lý?
- 30. Vì sao nói ranh giới giữa hai loại codec ấy đang mờ đi?

3.9.3 Bài thảo luận

Tình huống 1 — bộ sưu tập nhạc "chất lượng cao"

Một người mua được một bộ sưu tập nhạc quảng cáo là "lossless 24 bit / 96 kHz", tệp đuôi .flac, dung lượng rất lớn. Nghe thử thì thấy không khác gì bản MP3 đang có. Người ấy nhờ bạn kiểm tra.

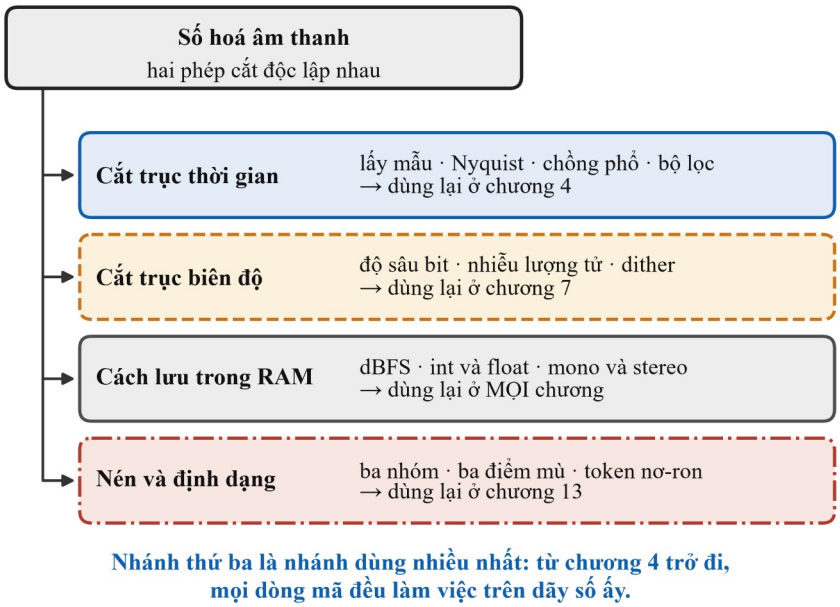
- Nêu quy trình kiểm tra, theo thứ tự từ nhanh tới kỹ.
- Dấu hiệu nào chứng minh được tệp có nguồn gốc từ nén có tổn hao?
- Giả sử tệp thật sự là 24 bit / 96 kHz và không có dấu vết nén. Người ấy có nên kỳ vọng nghe hay hơn bản CD 16 bit / 44,1 kHz không? Trả lời dựa vào mục 3.2.5 và 3.4.2.
- Trong trường hợp nào thì 24 bit / 96 kHz *thật sự* có giá trị, và với ai?

Tình huống 2 — thu tiếng động vật ngoài thiên nhiên

Một nhóm người yêu thiên nhiên muốn thu tiếng côn trùng và chim trong rừng để phân tích. Họ có một máy ghi hiện trường đặt ở 48 kHz, 24 bit. Sau khi phân tích, họ thấy phổ có những vạch lạ ở khoảng 3 – 5 kHz mà khi nghe lại bản thu thì không nghe thấy tiếng nào ở đó.

- Nêu giả thuyết có khả năng nhất, và giải thích bằng khái niệm của chương này.
- Đưa ra một phép thử phân biệt được giả thuyết ấy với các nguyên nhân khác.
- Nếu giả thuyết đúng thì bản thu đã có còn cứu được không? Vì sao?
- Đề xuất cấu hình thu cho lần sau, kèm lý do cho từng tham số.

3.9.4 Bản đồ khái niệm



Hình 3.37 Bốn nhánh khái niệm của chương 3 và chương nào sẽ dùng lại từng nhánh

Sáu điều phải nhớ sau chương này

Số hoá gồm *hai* phép *độc lập*: lấy mẫu cắt trục thời gian, lượng tử hoá cắt trục biên độ.

Định lý Nyquist: fs > 2·f_max. Vượt ngưỡng rồi thì tín hiệu khôi phục *trọn vẹn* — dựng lại bằng sinc, không phải nối chấm.

Chồng phổ *biến* thứ vô hại thành thứ có hại, và không sửa được sau khi lấy mẫu. Phải chặn bằng bộ lọc từ trước.

Mỗi bit cho thêm 6,02 dB dải động. Dither đổi *méo* lấy *nhiễu*, dùng đúng một lần ở bước hạ bit cuối.

Nguyên tắc cốt lõi: đọc vào đổi ngay sang số thực, xử lý ở số thực, chỉ hạ về số nguyên ở bước ghi cuối cùng.

Ba nhóm định dạng; nén có tổn hao dựa trên ba điểm mù của tai và *không* hoàn tác được — nén đúng một lần, ở bước cuối.

3.9.5 Bảng thuật ngữ Anh – Việt

Bảng 3.12 Thuật ngữ chương 3

Tiếng Anh	Tiếng Việt	Ghi chú
Digitization	Số hoá	gồm hai phép rời rạc hoá độc lập
Sampling	Lấy mẫu	cắt trục thời gian
Sample rate	Tần số lấy mẫu	ký hiệu fs, đơn vị héc
Nyquist frequency	Tần số Nyquist	bằng nửa tần số lấy mẫu
Nyquist–Shannon theorem	Định lý lấy mẫu	$fs > 2 \times f_{max}$

Tiếng Anh	Tiếng Việt	Ghi chú
Aliasing	Chồng phổ	tần số cao bị gập thành tần số thấp giả
Anti-aliasing filter	Bộ lọc chống chồng phổ	đặt *trước* bộ chuyển đổi
Sinc interpolation	Nội suy sinc	phép dựng lại mà định lý bảo đảm đúng
Quantization	Lượng tử hoá	cắt trục biên độ
Bit depth	Độ sâu bit	quyết định số mức và dải động
Quantization noise	Nhiễu lượng tử	sai số làm tròn; có quy luật nên nghe thành méo
Dither	Dither	cộng nhiễu nhỏ trước khi làm tròn
TPDF	Phân bố tam giác	dạng dither chuẩn của ngành
dBFS	Decibel toàn thang	mốc là trần bộ chuyển đổi; luôn âm
Clipping	Méo cắt đỉnh	vượt trần thì bị kẹp, mất vĩnh viễn
PCM	Điều chế mã xung	cách lưu mẫu thô của tệp WAV
Lossless	Nén không tổn hao	khôi phục chính xác từng mẫu
Lossy	Nén có tổn hao	phần bỏ đi mất vĩnh viễn
Bitrate	Tốc độ bit	ngân sách bit cho mỗi giây
Masking	Che lấp	điểm mù cho tiết kiệm lớn nhất
Brick-wall cutoff	Vạch cắt dựng đứng	dấu vết của nén lossy
Transcoding	Chuyển mã (nén lại)	mất mát chồng lên mất mát
Neural audio codec	Codec nơ-ron	từ điển token (ký hiệu rời rạc) do mạng tự học
RVQ	Lượng tử hoá vector nhiều tầng	biến biểu diễn liên tục thành chuỗi token

3.9.6 Bảng tự kiểm trước khi sang chương 4

Tự trả lời tám câu này không mở tài liệu.

– Nói được hai phép cắt của số hoá và tham số điều khiển từng phép (3.1.1).
– Phát biểu định lý Nyquist và tính được tần số Nyquist của một fs bất kỳ (3.2.2).
– Tính được tần số thấy được của một tần số vượt Nyquist (3.3.1).

- Giải thích vì sao phải chặn chồng phổ *trước* khi lấy mẫu (3.3.3).
- Cho độ sâu bit, tính được số mức và dải động (3.4.1, 3.4.2).
- Nói được dither làm gì, và vì sao chỉ dùng một lần (3.4.3, 3.4.4).
- Đổi được giữa biên độ mẫu và dBFS, và phát biểu nguyên tắc cốt lõi về int/float (3.5).
- Phân biệt ba nhóm định dạng và giải thích vì sao lossy không hoàn tác được (3.6).

3.9.7 Chuẩn bị cho chương 4

Chương 4 đi vào *miền tần số*: biến đổi Fourier, phổ, phổ đồ và bộ lọc số. Chương 1 đã cho biết *đọc* một cái phổ; chương 4 sẽ cho biết nó được *tính* ra thế nào — và bước "phân tích ra phổ" của bộ mã hoá ở mục 3.6.4 chính là chỗ ấy.

- **Việc phải làm:** chạy được hai đoạn mã ở mục 3.8 và nghe được các tệp chúng sinh ra.
- **Bài tập:** bảy bài ở mục 3.9.1, có đáp số in sẵn để tự đối chiếu.
- **Ôn lại:** quan hệ tần số – chu kỳ ở chương 1, và khái niệm phổ hài ở mục 1.4. Chương 4 xây trực tiếp trên hai chỗ đó.
- **Nên làm thêm:** mở phổ đồ của vài đoạn nhạc khác thể loại và tập đoán: đoạn nào có nhiều năng lượng ở đâu, và vì sao.

> **Một câu hỏi để mang sang chương 4**
>
> Chương này đã nói: mọi tần số trên Nyquist đều bị gập xuống, và ba tần số khác nhau có thể cho ra cùng một dãy số.
>
> Vậy khi nhìn một dãy số, làm sao biết trong nó có những tần số nào? Ai đã tính ra cái phổ mà chúng ta vẫn nhìn từ chương 1 tới giờ, và tính bằng cách nào?
>
> Đó là câu hỏi của chương 4 — và câu trả lời của nó là công cụ được dùng nhiều nhất trong toàn bộ phần còn lại của cuốn sách.

Tài liệu tham khảo của chương

Sách và giáo trình

- Watkinson, J. — The Art of Digital Audio. Nguồn nền của phần lấy mẫu, lượng tử hoá và dither.
- Pohlmann, K. C. — Principles of Digital Audio. Nguồn của phần định dạng, dải động và các chuẩn tần số lấy mẫu.
- Smith, J. O. — Mathematics of the Discrete Fourier Transform. Nguồn của phần khôi phục bằng nội suy sinc; sẽ dùng nhiều hơn ở chương 4.
- Bosi, M., Goldberg, R. E. — Introduction to Digital Audio Coding and Standards. Nguồn của phần mô hình tâm lý thính giác và dây chuyền mã hoá.

Tiêu chuẩn và tài liệu kỹ thuật

- ISO/IEC 11172-3 và ISO/IEC 13818-7 — chuẩn của MP3 và AAC.
- RFC 6716 — Definition of the Opus Audio Codec, IETF.

– draft-ietf-mlcodec-opus-scalable-quality-extension — bản thảo mở rộng Opus HD tới 96 kHz, IETF, tháng 10 năm 2025.
– Tài liệu phát hành Opus 1.5 (tháng 3 năm 2024) và Opus 1.6 (tháng 12 năm 2025) — nguồn của phần Deep Redundancy ở mục 3.7.3.

Một lưu ý về các con số trong bài

> **Cách đọc các con số**
>
> Các con số có nguồn tiêu chuẩn — tần số lấy mẫu chuẩn, quan hệ giữa độ sâu bit và dải động, các mốc bitrate — được nêu đúng như tài liệu gốc quy định.
>
> Các con số mô tả vạch cắt của codec và chất lượng cảm nhận là giá trị *tiêu biểu* dùng để nhận dạng và so sánh, không phải quy định của chuẩn. Mỗi bộ mã hoá đặt vạch cắt hơi khác nhau.
>
> Các mốc thời gian của phần codec nơ-ron tính tới tháng 9 năm 2026. Người đọc ở thời điểm muộn hơn nên kiểm lại; cách đặt câu hỏi ở mục 3.7.3 thì không cũ theo thời gian.
>
> Mọi con số tính toán và mọi đáp số bài tập đều sinh ra từ mô-đun sigtools.py, đã qua 122 phép tự kiểm.