TRƯỜNG ĐẠI HỌC SÂN KHẤU – ĐIỆN ẢNH HÀ NỘI
KHOA CÔNG NGHỆ ĐIỆN ẢNH – TRUYỀN HÌNH

KỸ THUẬT AUDIO – VIDEO

CHƯƠNG 3

Số hóa âm thanh
Lấy mẫu · Lượng tử hóa · Nyquist

5 tiết · Dành cho sinh viên Đạo diễn âm thanh, Quay phim, Đạo diễn điện ảnh – truyền hình, Dựng phim

Hà Nội – 2026

Số hóa = hai phép cắt độc lập, nối tiếp nhau

Ở Chương 2 ta đã chốt: tín hiệu analog **liên tục kép** — liên tục theo thời gian VÀ liên tục theo biên độ. Máy tính hữu hạn không lưu nổi. Vậy phải cắt. Cắt hai lần, trên hai trục khác nhau.

- **Cắt trục ngang — trục thời gian:** chỉ giữ lại giá trị tại những mốc cách đều nhau. Gọi là **lấy mẫu (sampling)**.
- **Cắt trục dọc — trục biên độ:** làm tròn mỗi giá trị về một trong số hữu hạn mức cho trước. Gọi là **lượng tử hóa (quantization)**.
- Hai phép này **hoàn toàn độc lập**: tần số lấy mẫu và độ sâu bit là hai tham số riêng, chỉnh cái này không ảnh hưởng cái kia.
- Cả Chương 3 chỉ là khai triển hai câu trên. Nắm được ranh giới giữa hai phép cắt là nắm được cả chương.

BẢN ĐỒ CHƯƠNG

Sáu mảng kiến thức của Chương 3



Hai phép rời rạc hóa ở 3.1 và 3.2 là gốc; bốn mảng còn lại đều là hệ quả trực tiếp của chúng.

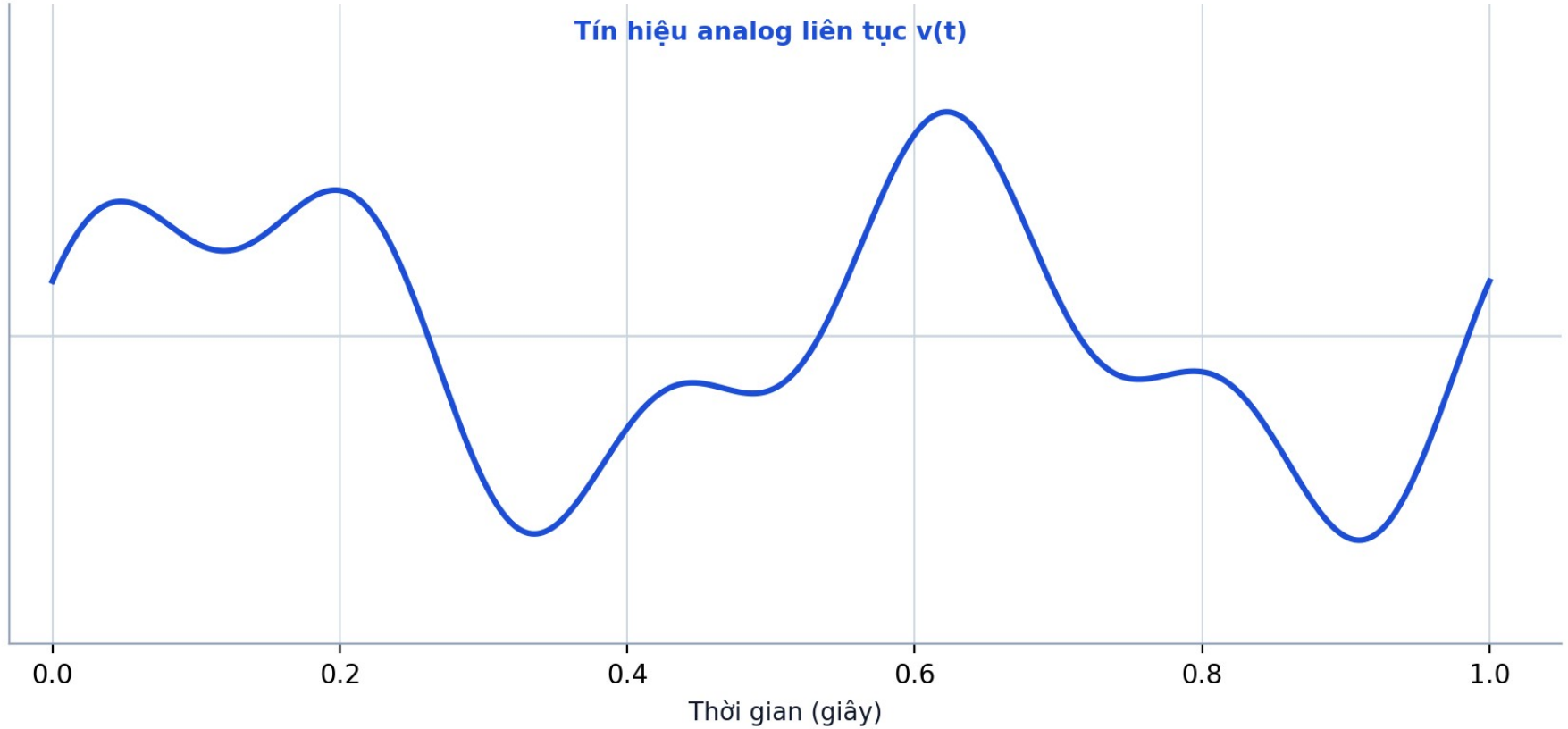
TIẾT 1

Lấy mẫu và định lý Nyquist – Shannon

- Rời rạc hóa trục thời gian: fs, chu kỳ T = 1/fs
- Dãy p(nT) chính là mảng NumPy trong máy tính
- Định lý Nyquist–Shannon: fs > 2·f_max
- Trực giác hình học: vì sao cần tối thiểu hai điểm mẫu

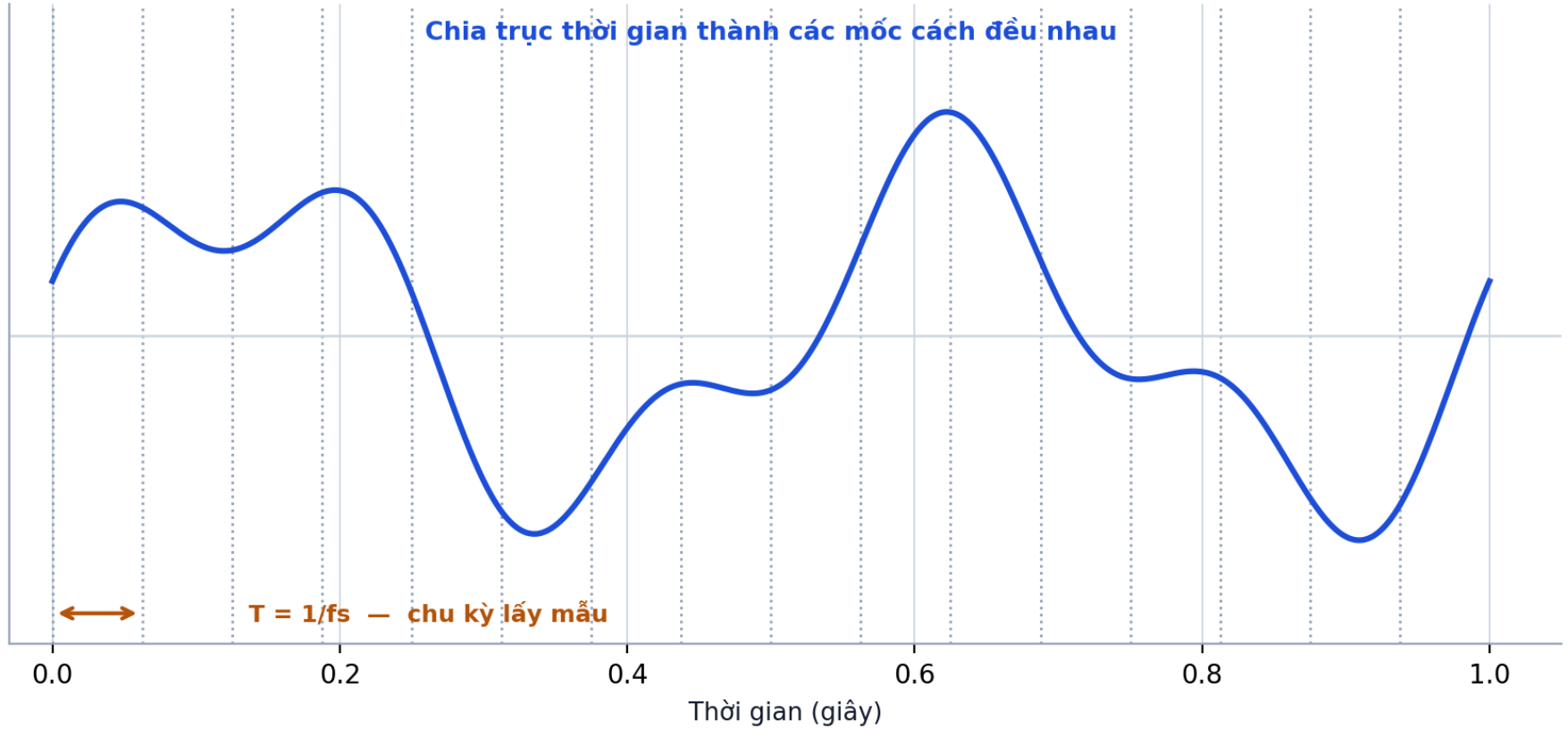


Bước 1 — tín hiệu analog liên tục



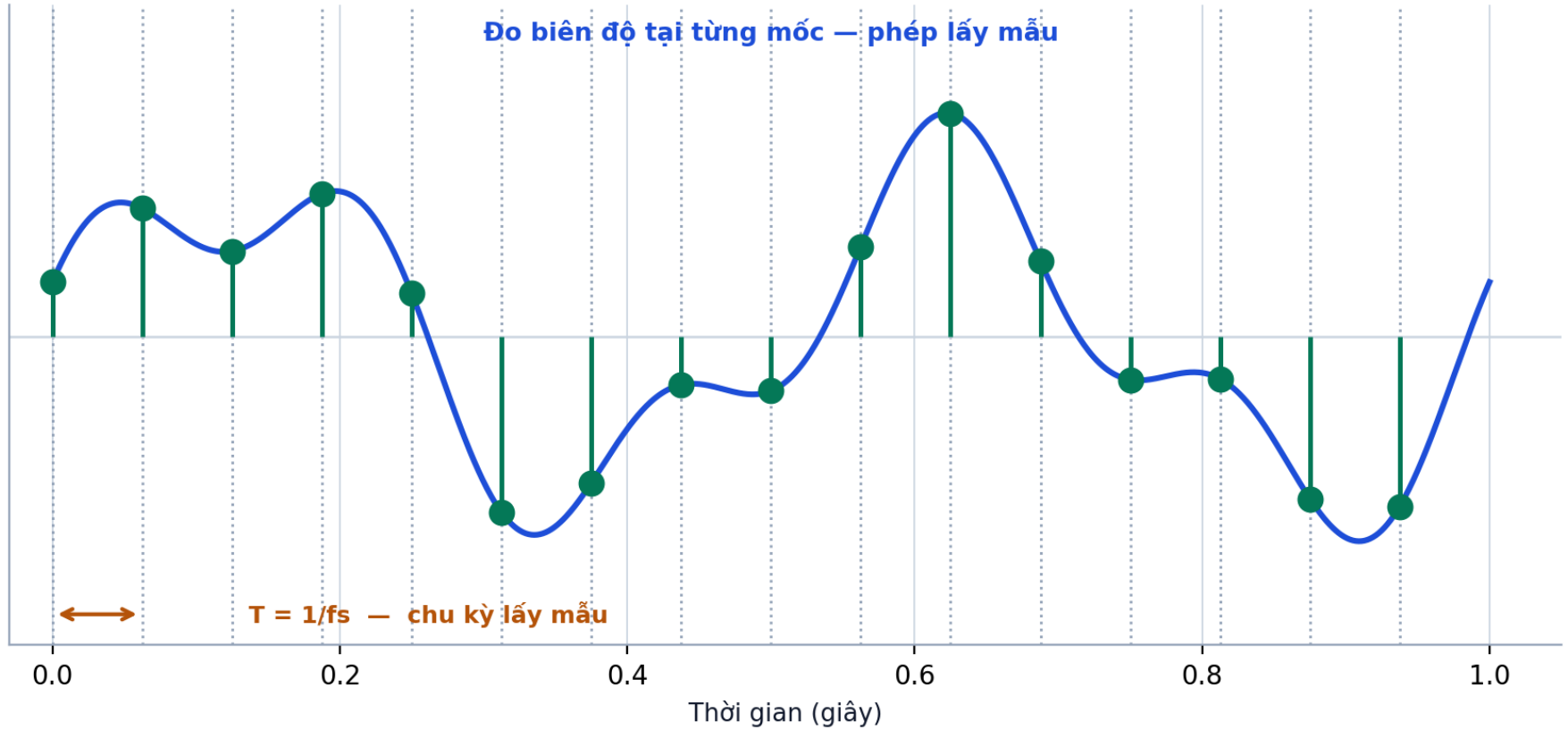
Sóng điện áp v(t) từ preamp: liên tục, trơn tru, xác định tại mọi thời điểm.

Bước 2 — chia trục thời gian thành mốc đều



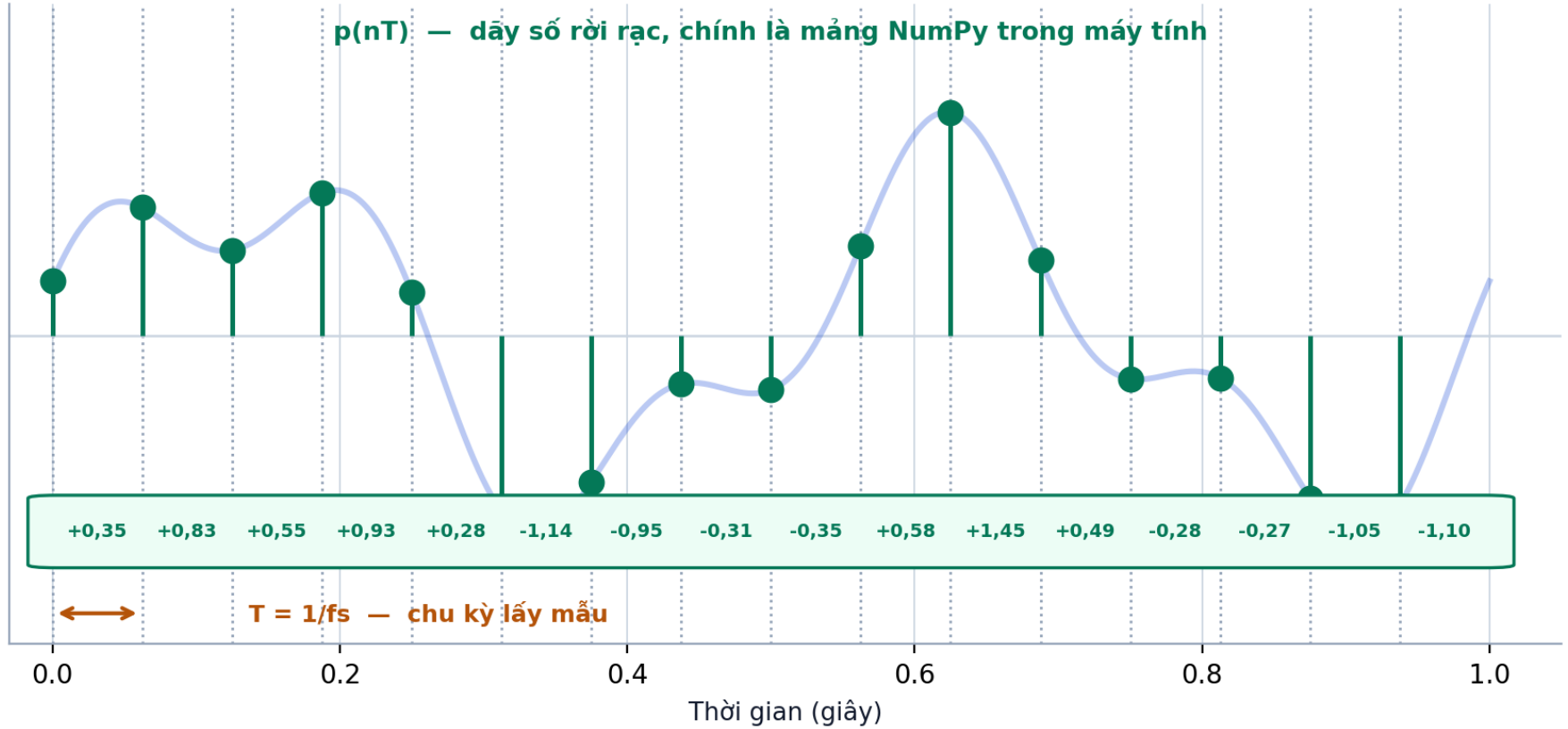
Khoảng cách không đổi giữa hai mốc là chu kỳ lấy mẫu T = 1/fs. Số mốc trong một giây là tần số lấy mẫu fs.

Bước 3 — đo biên độ tại từng mốc



Mỗi mốc cho một giá trị biên độ tức thời. Đây chính là phép lấy mẫu (sampling).

Bước 4 — kết quả là một dãy số



Dãy p(nT) với n = 0, 1, 2, 3… — mảng một chiều trong bộ nhớ máy tính mà ta sẽ lập trình xử lý trực tiếp.

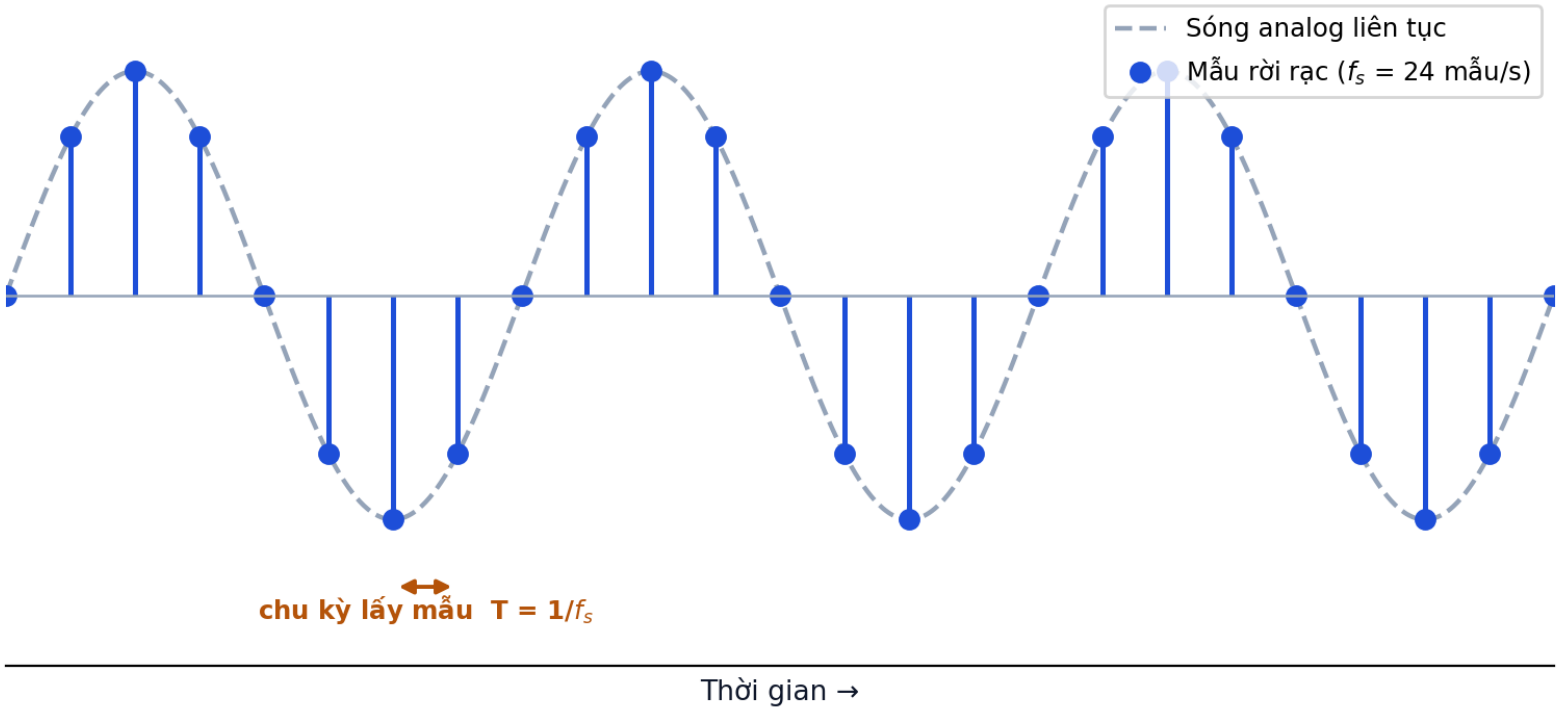
fs và mảng mẫu là cặp đôi không thể tách rời

Một mảng số `p` thuần túy trong Python **hoàn toàn vô nghĩa** nếu ta không biết khoảng thời gian T giữa hai phần tử kề nhau là bao lâu.

- Mọi thao tác đọc/ghi tệp hay xử lý tín hiệu số đều yêu cầu **hoặc trả về** tham số fs đi kèm mảng mẫu.
- Cú pháp chuẩn: `x, fs = sf.read('giong_noi.wav')` — `x` là mảng NumPy, `fs` là tần số lấy mẫu.
- Nếu tùy tiện đổi fs khi lưu tệp — thu ở 48 kHz nhưng ghi ra với fs = 24 kHz — tốc độ phát **chậm đi một nửa** và cao độ **tụt xuống đúng một quãng tám**.
- Đây cũng là nguyên lý của hiệu ứng đổi giọng thô sơ: chỉ cần khai báo sai fs là giọng người thành giọng quái vật.

HÌNH 3.1

Rời rạc hóa trục thời gian



Máy tính không lưu "đường cong", chỉ lưu dãy số đo được tại các mốc thời gian cách đều nhau.

Hình 3.1 — Sóng analog liên tục được đo tại các mốc thời gian cách đều nhau.

CÂU HỎI TẠI LỚP

Muốn ghi lại được một sóng sin, tối thiểu cần bao nhiêu điểm mẫu trong mỗi chu kỳ?

A 1 điểm là đủ

B Tối thiểu 2 điểm

C Tối thiểu 10 điểm

D Càng nhiều càng tốt, không có ngưỡng tối thiểu

Suy nghĩ 30 giây — rồi trả lời

Định lý lấy mẫu Nyquist – Shannon

Để **tái lập hoàn hảo** một tín hiệu liên tục có tần số giới hạn từ phiên bản mẫu rời rạc của nó, tần số lấy mẫu fs bắt buộc phải lớn hơn ít nhất **gấp hai lần** tần số cao nhất f_max có trong tín hiệu gốc.

$$f_s > 2f_{\max} \; ; \qquad f_{\text{Nyquist}} = \frac{f_s}{2}$$

- **Tần số Nyquist f_N = fs/2** — giới hạn tần số cao nhất mà hệ thống số biểu diễn trung thực được.
- **Trực giác hình học:** một chu kỳ sóng sin là một lần lên và một lần xuống. Muốn biết có dao động, cần ít nhất **một điểm ở nửa chu kỳ dương và một điểm ở nửa chu kỳ âm** — tức 2 điểm.
- Dưới ngưỡng đó, các điểm mẫu rơi vào vị trí sai lệch và mô tả **sai** sang một sóng khác có tần số thấp hơn nhiều.

Ứng dụng: tần số lấy mẫu của đĩa CD

Tình huống: Tai người nghe được tối đa **f_max = 20.000 Hz**. Theo Nyquist, fs chỉ cần lớn hơn 40.000 Hz. Vậy vì sao chuẩn Compact Disc lại chọn con số kỳ lạ **44.100 Hz** chứ không phải 40.000 hay 42.000?

1. Áp dụng định lý: fs > 2 × 20.000 = **40.000 Hz**. Đó là ngưỡng sàn tuyệt đối.

2. Nhưng bộ lọc chống răng cưa trong thiết bị **không thể cắt dựng đứng** tại đúng 20 kHz — mọi bộ lọc analog đều có một vùng chuyển tiếp có độ dốc hữu hạn.

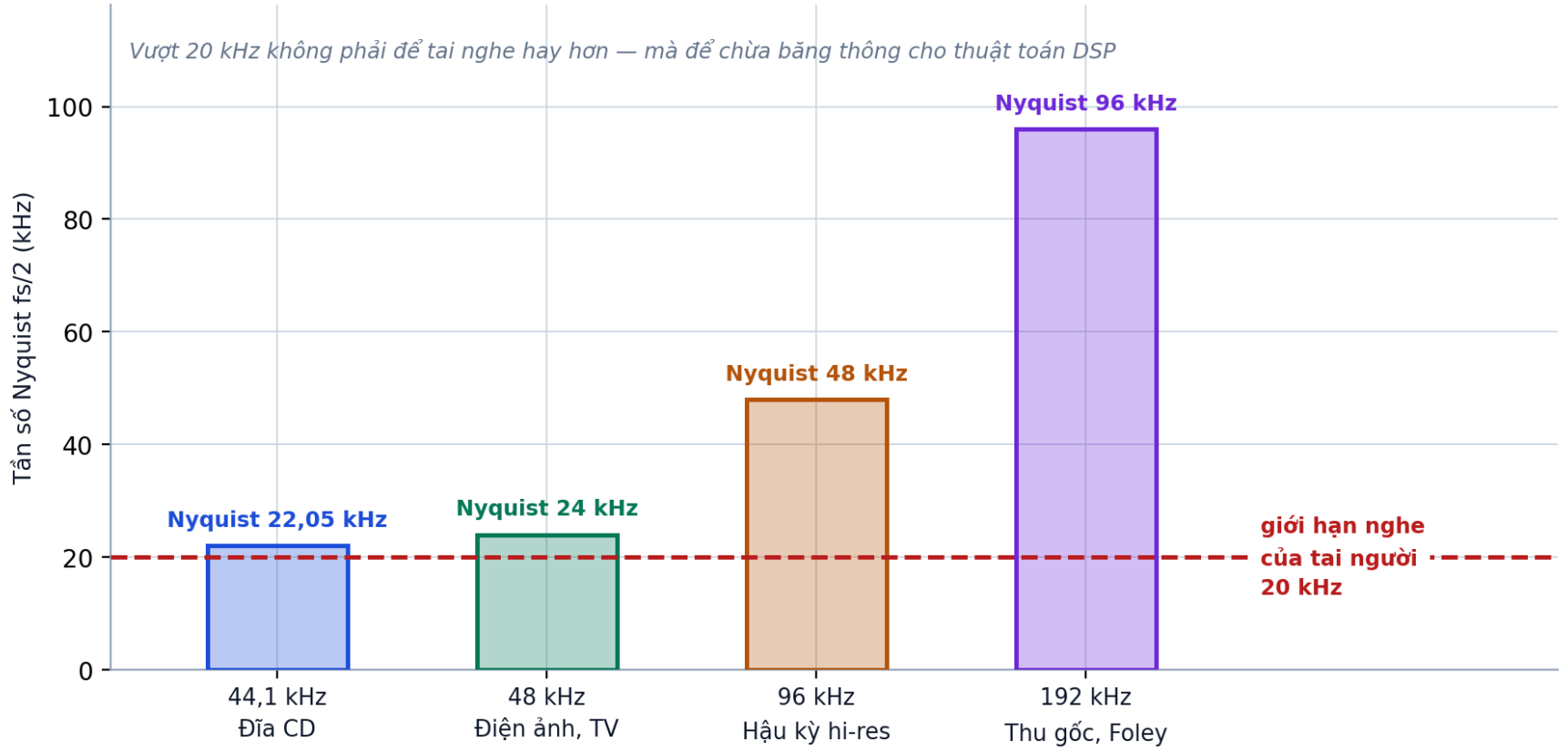
3. Phải chừa ra một **khoảng đệm** cho vùng chuyển tiếp đó: 44.100 − 40.000 = **4.100 Hz**.

4. Trong khoảng đệm này, bộ lọc có chỗ để suy giảm dần từ mức đầy đủ ở 20 kHz xuống mức không đáng kể ở 22,05 kHz.

Kết luận nghề: Con số 44,1 kHz không phải ngẫu nhiên: nó **phủ kín dải nghe của người** đồng thời **chừa đủ chỗ cho bộ lọc analog thực tế**. Đây là ví dụ điển hình của việc tiêu chuẩn kỹ thuật luôn phải cộng thêm biên an toàn so với lý thuyết thuần túy.

BẢNG 3.2 → BIỂU ĐỒ

Bốn tần số lấy mẫu tiêu chuẩn



Mọi chuẩn đều có tần số Nyquist vượt trên 20 kHz — nhưng vượt bao nhiêu thì tùy mục đích.

BẢNG 3.2

Tần số lấy mẫu chuẩn và phạm vi ứng dụng

Tần số lấy mẫu fs	Tần số Nyquist fs/2	Phạm vi ứng dụng thực tế
44.100 Hz (44,1 kHz)	22.050 Hz	Đĩa CD, nhạc số phổ thông, phân phối trực tuyến
48.000 Hz (48 kHz)	24.000 Hz	Video, truyền hình kỹ thuật số, âm thanh điện ảnh
96.000 Hz (96 kHz)	48.000 Hz	Sản xuất phòng thu chuyên nghiệp, hậu kỳ độ phân giải cao
192.000 Hz (192 kHz)	96.000 Hz	Thu âm gốc, thiết kế tiếng động Foley, lưu trữ bản gốc

*Sinh viên khoa Công nghệ Điện ảnh – Truyền hình sẽ làm việc với **48 kHz** trong hầu hết mọi dự án. Nhận file 44,1 kHz từ nhạc sĩ thì phải chuyển đổi trước khi đưa vào timeline.*

CHỐT LẠI MỤC 3.1

Chốt lại: lấy mẫu và Nyquist

Lấy mẫu = đo biên độ tại các mốc thời gian **cách đều**, cho ra dãy p(nT).

Trực giác: một chu kỳ sin cần **tối thiểu 2 điểm** — một dương, một âm.

fs là số mẫu mỗi giây; **T = 1/fs** là chu kỳ lấy mẫu.

44,1 kHz phủ dải nghe 20 kHz + chừa 4,1 kHz cho bộ lọc analog.

Nyquist: fs > 2·f_max, và **f_N = fs/2** là trần tần số biểu diễn được.

fs và mảng mẫu là cặp đôi — tách rời là sai tốc độ và sai cao độ.

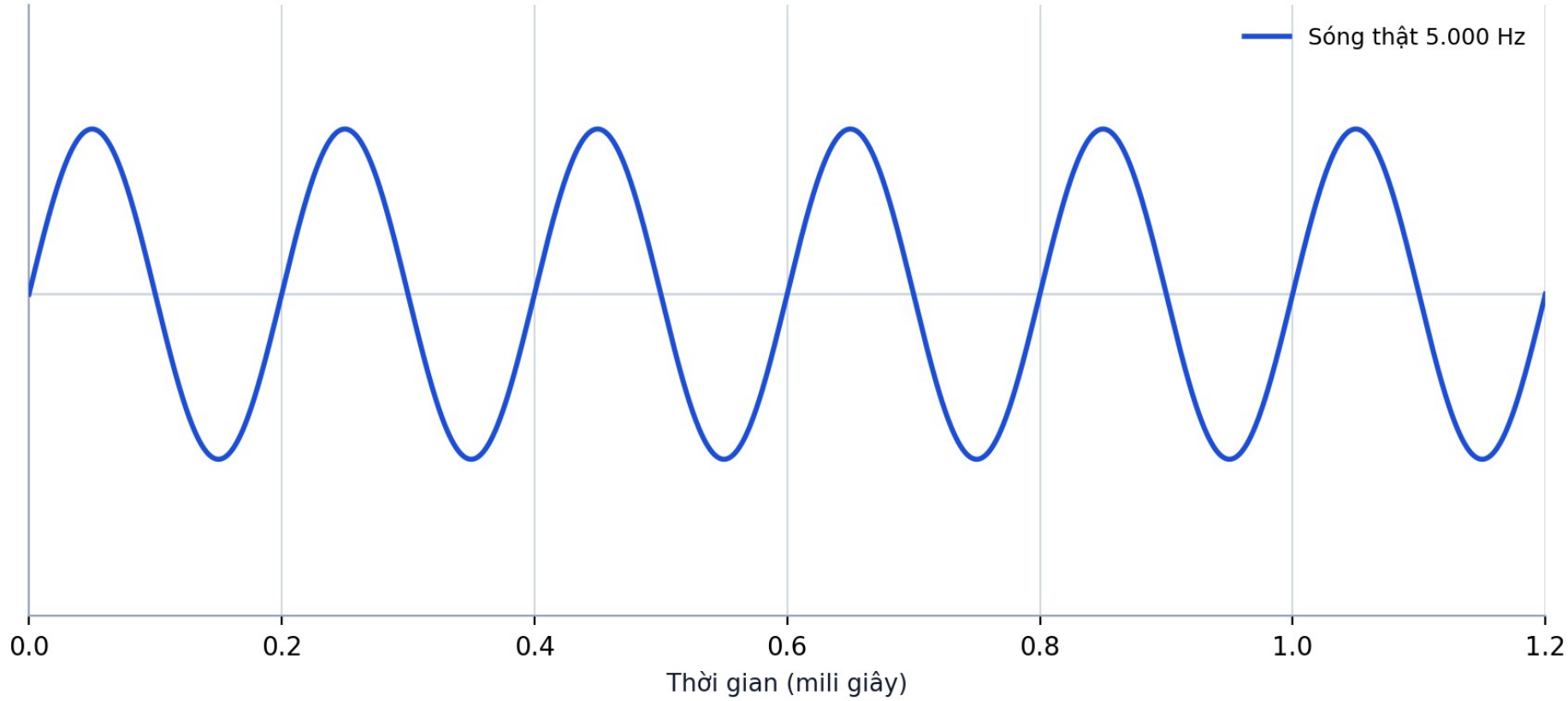
TIẾT 2

Chồng phổ và bộ lọc chống răng cưa

- Hiện tượng aliasing: tần số cao bị "gấp" thành tần số thấp giả mạo
- Công thức tính tần số giả mạo f_alias = |fs − f|
- Ví dụ điện ảnh: bánh xe quay ngược trong phim
- Vì sao bộ lọc bắt buộc phải nằm trước ADC

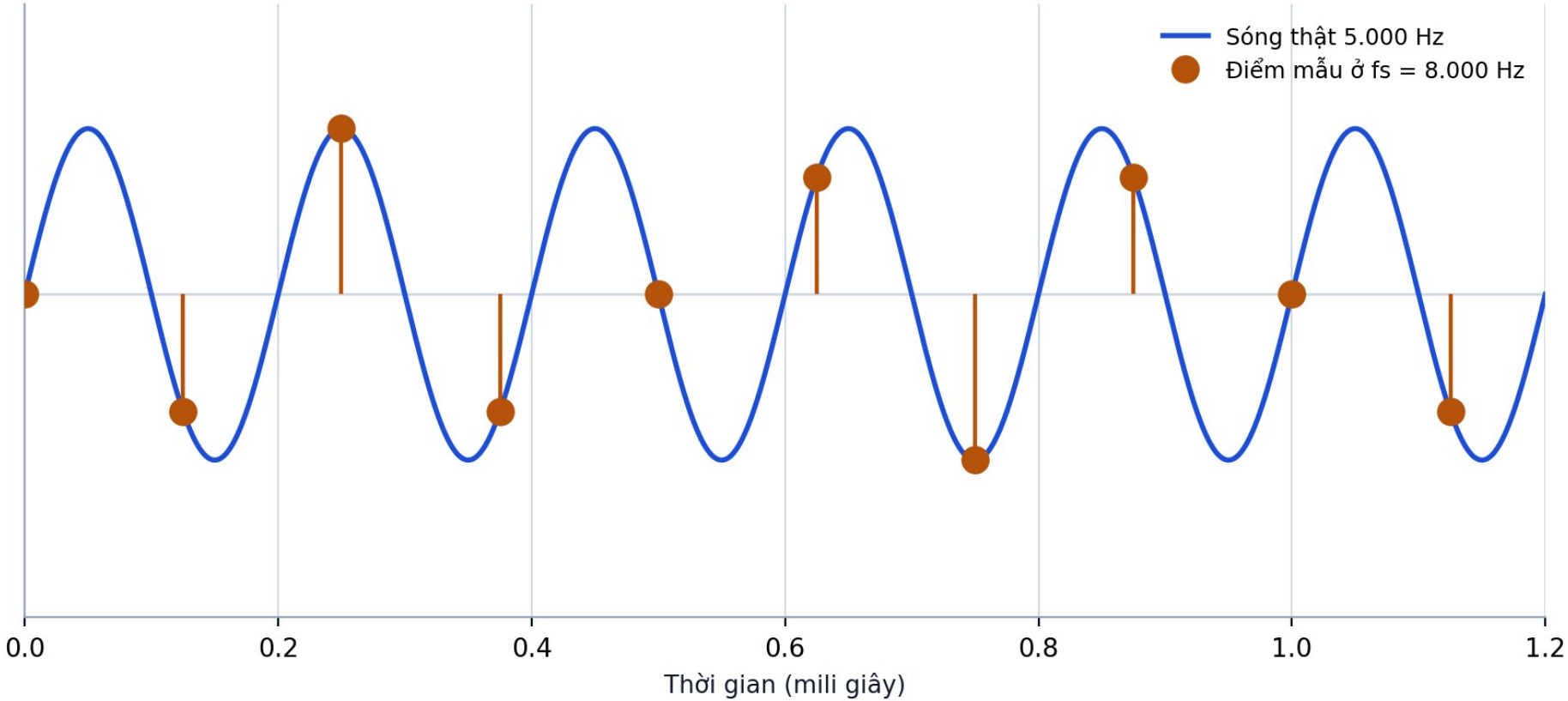


Bước 1 — một sóng 5.000 Hz đi vào hệ thống



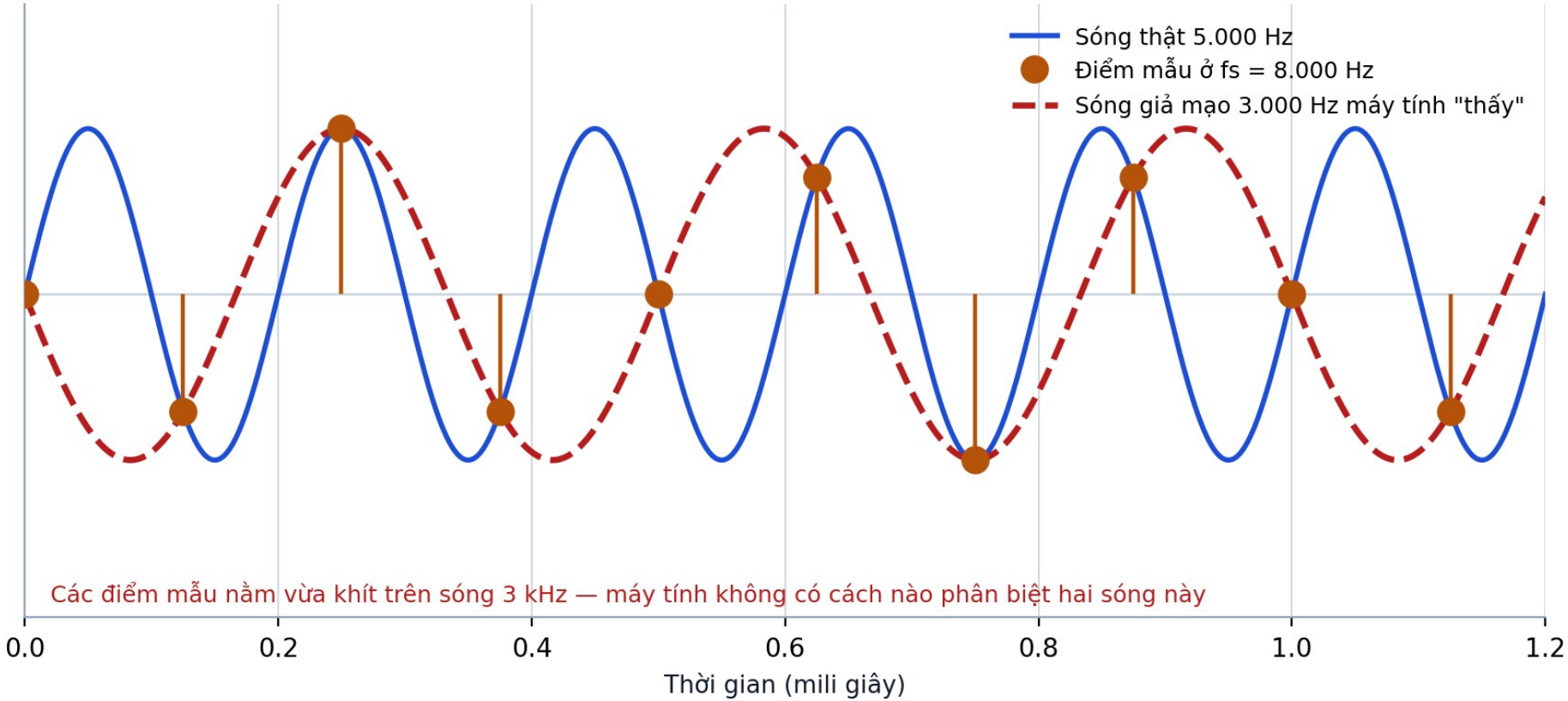
Hệ thống lấy mẫu ở fs = 8.000 Hz, tức tần số Nyquist chỉ 4.000 Hz. Sóng 5.000 Hz đã vượt ngưỡng.

Bước 2 — các điểm mẫu rơi vào đâu



Mỗi chu kỳ sóng 5 kHz chỉ nhận được chưa tới 2 điểm mẫu — không đủ để mô tả đúng hình dáng.

Bước 3 — máy tính "thấy" một sóng khác hẳn



Các điểm mẫu nằm vừa khít trên một sóng 3.000 Hz. Trong mảng số, sóng 5 kHz đã biến thành 3 kHz — vĩnh viễn.

CÔNG THỨC (3.2)

Tính tần số giả mạo sau chồng phổ

Tần số cao vượt ngưỡng Nyquist bị **gấp đối xứng** quanh dải Nyquist và hiện ra dưới dạng một tần số thấp giả mạo.

$$f_{alias} = |f_s - f| \quad (\text{Hz})$$

- Với ví dụ vừa xem: f_alias = |8.000 − 5.000| = **3.000 Hz** — đúng bằng sóng đỏ trên hình.
- Công thức áp dụng cho trường hợp f nằm giữa f_N và fs. Với tần số cao hơn nữa, phổ tiếp tục gấp qua lại nhiều lần.
- Điểm chết người: tần số giả mạo **trùng khít** lên các tần số thật trong cùng mảng số. Không thuật toán nào bóc tách hay sửa chữa được nữa.

VÍ DỤ ĐIỆN ẢNH

Bánh xe quay ngược trong phim — cũng là aliasing

Aliasing không chỉ có trong âm thanh. Nó xuất hiện phổ biến trong kỹ thuật hình ảnh — và sinh viên khoa mình đã thấy nó hàng trăm lần mà không biết tên.

- Trong phim, khi ô tô tăng tốc, có lúc **nan hoa bánh xe như quay chậm dần rồi giật lùi ngược** hướng chuyển động thật.

- Nguyên nhân: máy quay đang **"lấy mẫu" hình ảnh** với tần số khung hình 24 hình/giây.

- Khi bánh xe quay quá nhanh — vượt tần số Nyquist của tốc độ khung hình — nan hoa đi qua các góc lớn hơn góc phân giải cho phép.

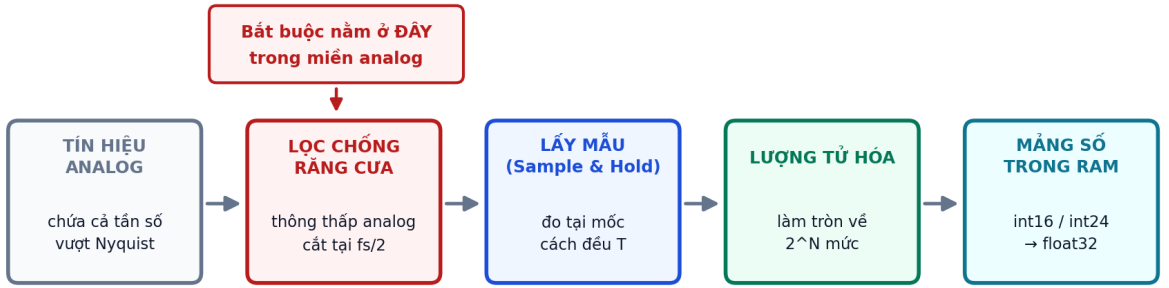
- Các khung hình thu được bị **"gấp" ngược hướng**, tạo ảo giác quay chậm hoặc quay lùi. Đây chính là aliasing trong miền không gian – thời gian.

GIẢI PHÁP

Bộ lọc chống răng cưa phải nằm trong miền analog

Vì aliasing không sửa được sau khi đã xảy ra, kỹ thuật bắt buộc phải **chặn đứng nó ngay trong môi trường analog, trước khi lấy mẫu**.

- Một **bộ lọc thông thấp analog** có tần số cắt đặt tại đúng ngưỡng Nyquist (fs/2) được tích hợp **ngay trước** khối ADC.
- Nhiệm vụ: cắt bỏ hoàn toàn mọi thành phần tần số vượt ngưỡng, đảm bảo tín hiệu vào ADC sạch bóng các tần số nguy hiểm.
- **Không thể thay bằng thuật toán phần mềm.** Sau khi lấy mẫu, tần số giả mạo và tần số thật đã nằm chồng lên nhau trong cùng những con số — không có thông tin nào để tách chúng ra.
- Đây là lý do câu hỏi ôn tập số 3 của chương hỏi đúng điểm này. Sinh viên phải trả lời được bằng lập luận, không phải bằng trí nhớ.

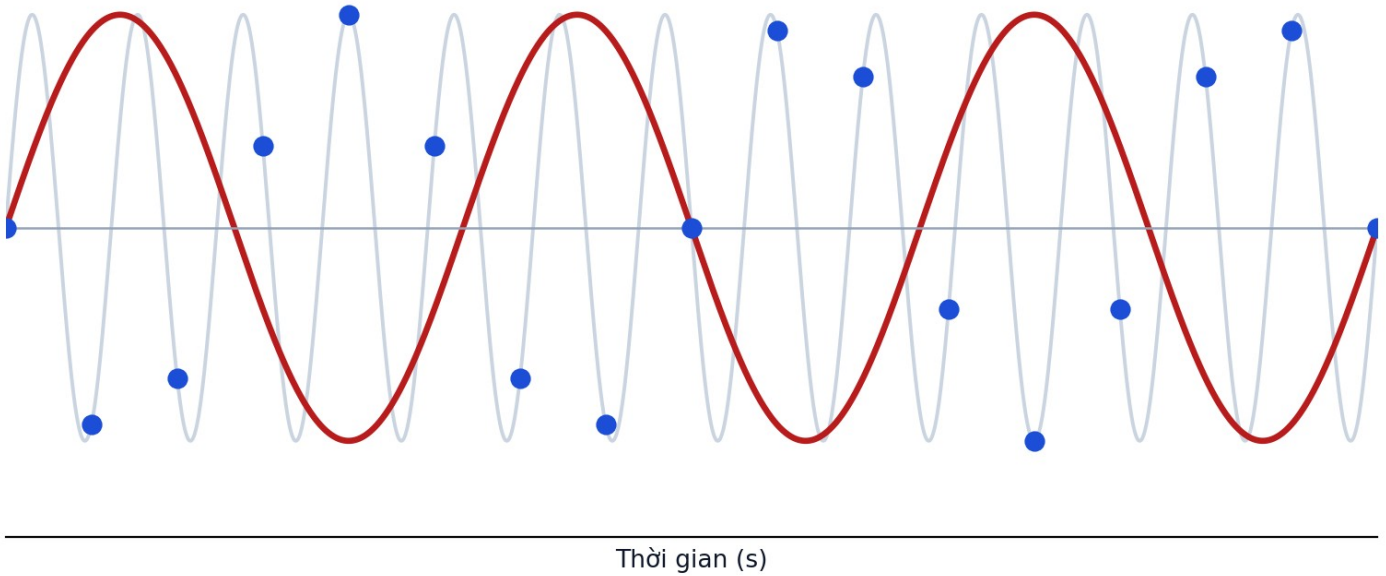


Một khi đã lấy mẫu, tần số giả mạo trùng khít lên tần số thật — không thuật toán nào bóc tách lại được

Vị trí bắt buộc của bộ lọc chống răng cưa trong khối ADC.

HÌNH 3.2

Hiện tượng chồng phổ trong giáo trình



Hình 3.2 — Sóng tần số cao vượt ngưỡng Nyquist bị lấy mẫu thưa và biến thành sóng giả mạo tần số thấp.

Bài tập 3 — xác định aliasing của ba thành phần tần số

Tình huống: Một tín hiệu chứa đồng thời ba đơn âm: **f_1 = 1.000 Hz, f_2 = 9.000 Hz, f_3 = 15.000 Hz**. Đưa vào bộ lấy mẫu với **fs = 16.000 Hz**. Xác định tần số Nyquist, thành phần nào an toàn, thành phần nào bị dội phổ và giá trị thu được.

1. Tần số Nyquist: f_N = fs/2 = 16.000/2 = **8.000 Hz**.

2. **f_1 = 1.000 Hz** < 8.000 Hz ⇒ nằm dưới ngưỡng, **an toàn**, khôi phục đúng 1.000 Hz.

3. **f_2 = 9.000 Hz** > 8.000 Hz ⇒ aliasing: f_out2 = |16.000 − 9.000| = **7.000 Hz**.

4. **f_3 = 15.000 Hz** > 8.000 Hz ⇒ aliasing: f_out3 = |16.000 − 15.000| = **1.000 Hz**.

5. So sánh kết quả: f_3 bị dội thành **đúng 1.000 Hz** — trùng khít lên f_1 vốn là tần số thật.

Kết luận nghề: Sóng 15 kHz giả dạng thành 1 kHz và **chồng lên tín hiệu thật**, gây méo tiếng và nhiễu loạn thông tin nghiêm trọng. Kết luận: bắt buộc phải đặt bộ lọc chống răng cưa trước ADC để cắt bỏ cả 9 kHz và 15 kHz trước khi lấy mẫu.

CHỐT LẠI MỤC 3.1

Chốt lại: chồng phổ và bộ lọc

Aliasing = tần số vượt f_N bị gấp đối xứng thành tần số thấp giả mạo.

Tần số giả mạo **trùng khít** lên tần số thật ⇒ không phần mềm nào tách lại được.

Công thức: **f_alias = |fs − f|** cho vùng giữa f_N và fs.

Bộ lọc chống răng cưa là bộ lọc thông thấp **analog**, cắt tại fs/2, đặt **trước ADC**.

Ví dụ trực quan: **bánh xe quay ngược** trong phim — aliasing của 24 hình/giây.

Sinh viên phải giải thích được **vì sao không thể lọc sau** — đây là câu hỏi thi.

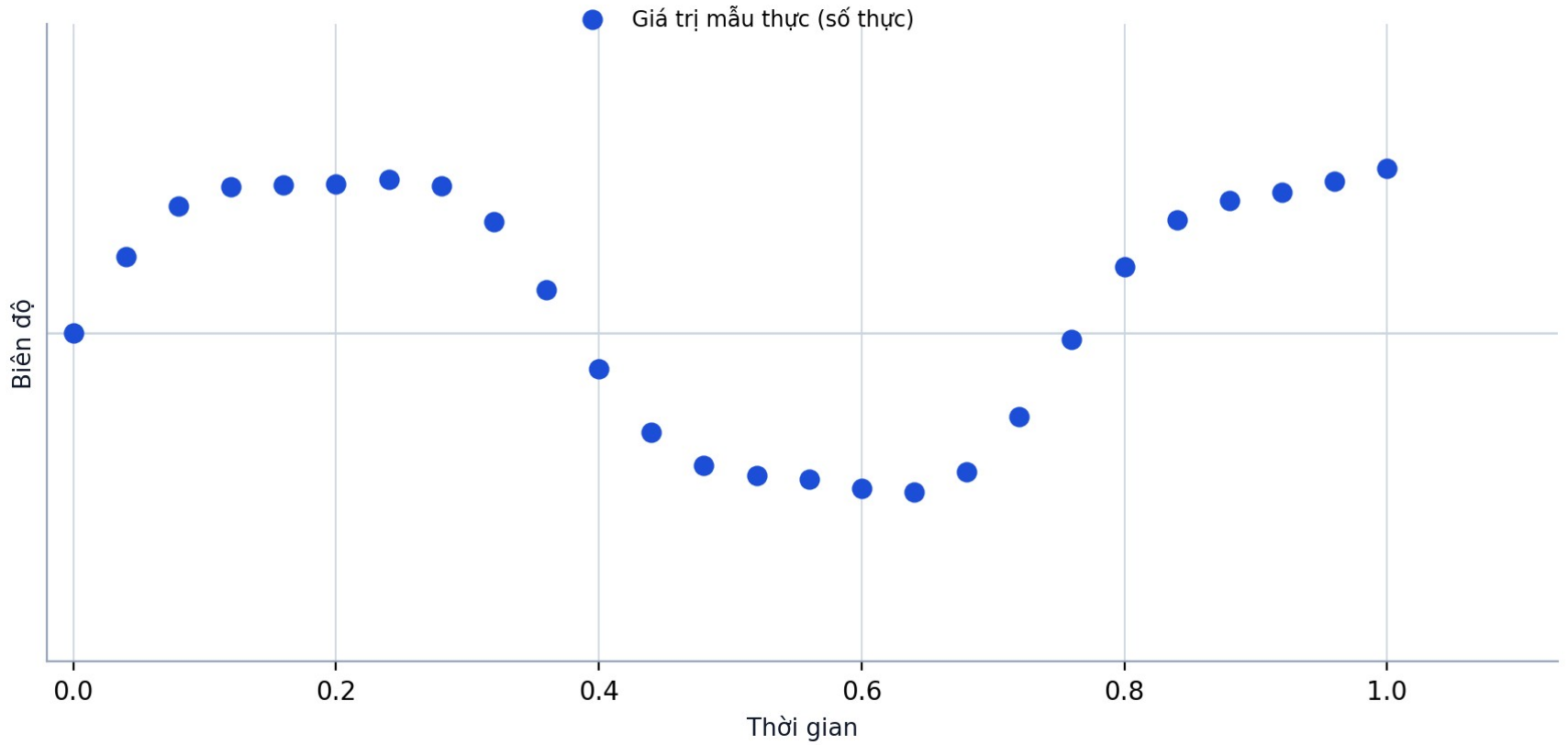
TIẾT 3

Lượng tử hóa, độ sâu bit và kỹ thuật dither

- Rời rạc hóa trục biên độ: M = 2^N mức lượng tử
- Nhiễu lượng tử và quan hệ tuyến tính với dải động
- Vì sao mỗi bit thêm vào cho đúng +6,02 dB
- Dither: biến méo hài có hại thành nhiễu nền dễ chịu

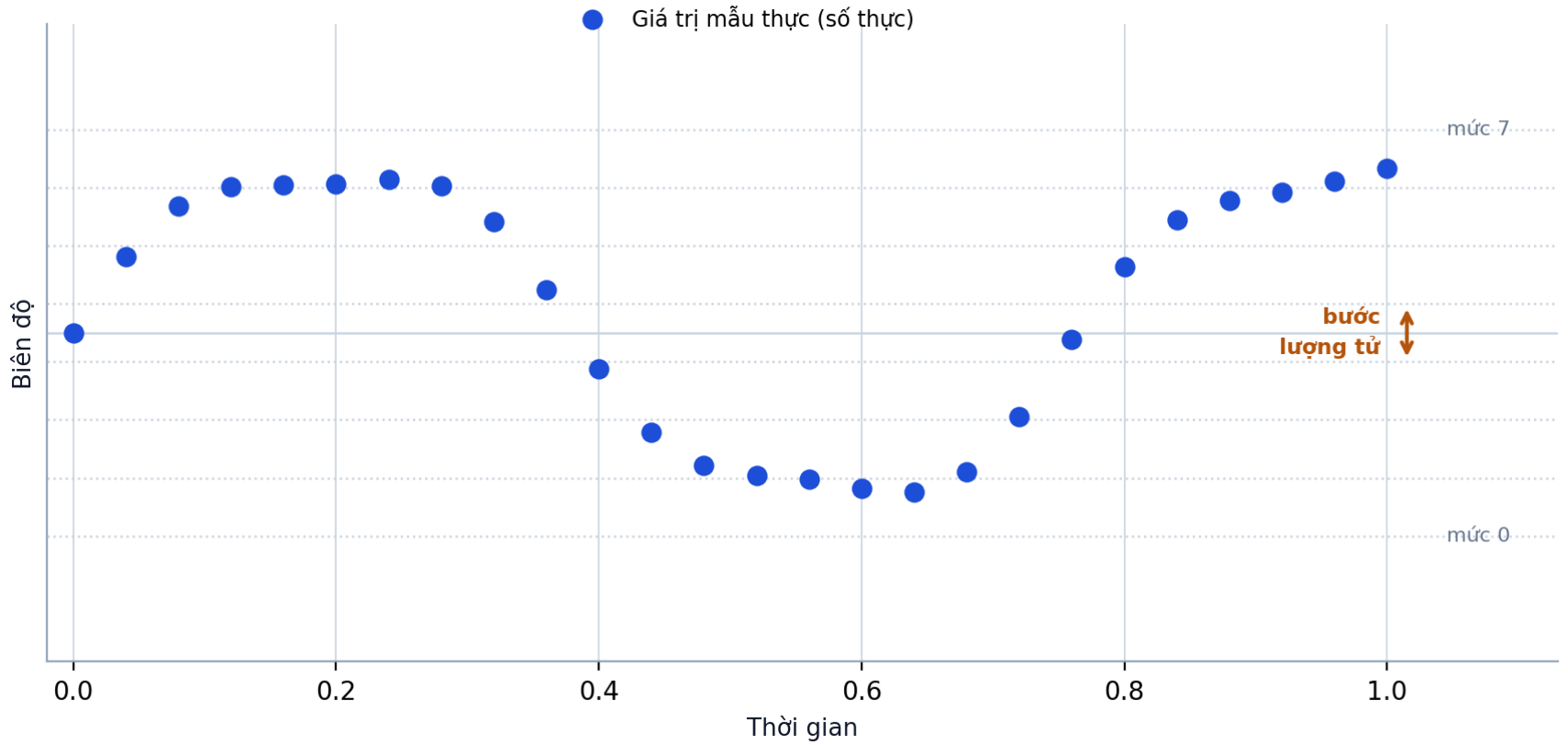


Bước 1 — các giá trị mẫu vẫn là số thực vô hạn chữ số



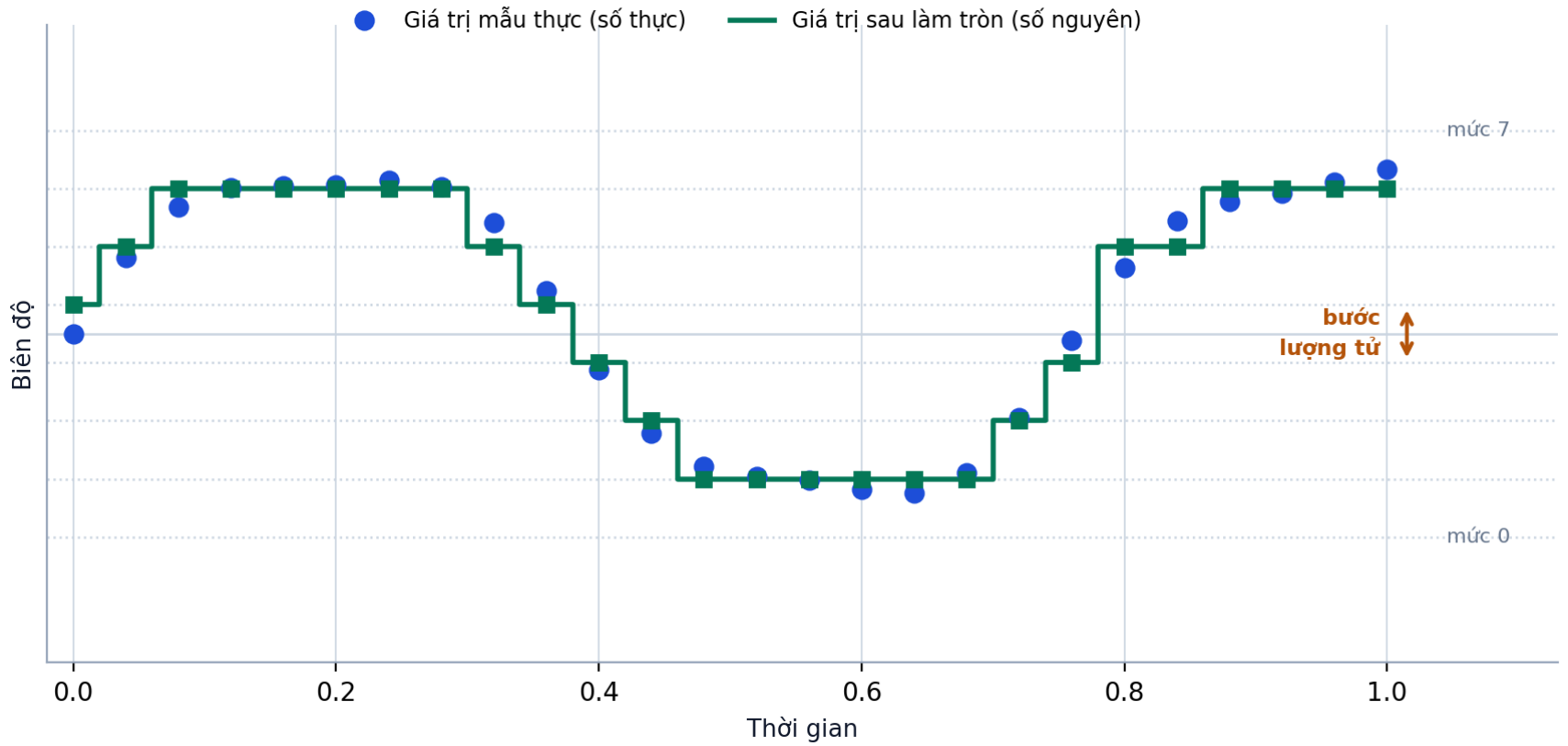
Sau khi lấy mẫu, biên độ tại mỗi mốc nT vẫn là số thực kiểu 0,548172938491… — máy tính vẫn chưa lưu được.

Bước 2 — dựng lưới các mức biên độ cho phép



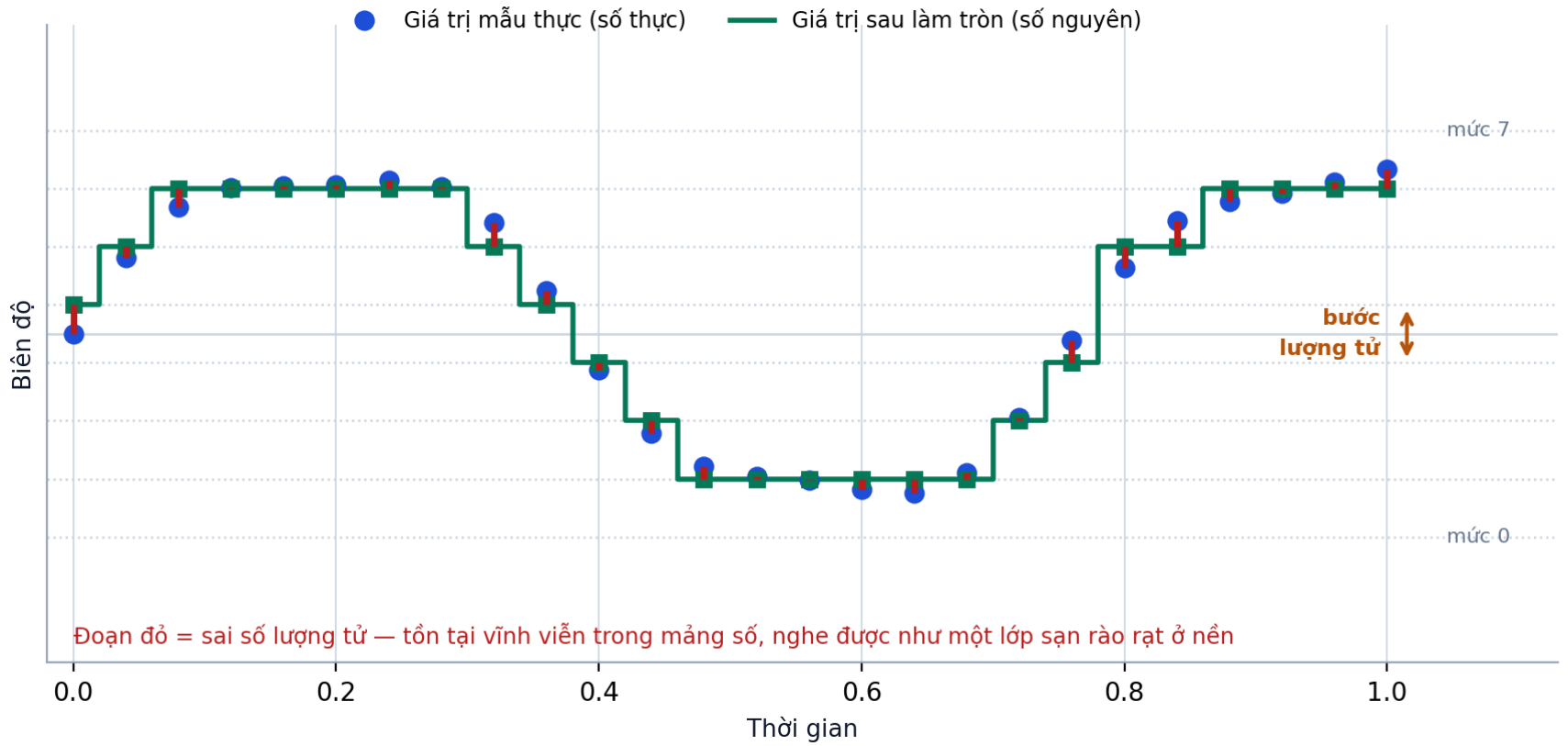
Hệ thống chỉ chấp nhận một số hữu hạn mức. Khoảng cách giữa hai mức kề nhau gọi là bước lượng tử.

Bước 3 — làm tròn mỗi mẫu về mức gần nhất



Đây là phép lượng tử hóa: mỗi số thực trở thành một số nguyên chỉ mức.

Bước 4 — phần chênh lệch là nhiễu lượng tử



Sai số giữa giá trị thật và giá trị đã làm tròn không biến mất — nó tồn tại vĩnh viễn trong mảng số.

CÔNG THỨC (3.3)

Số mức lượng tử theo độ sâu bit

Số lượng mức biên độ biểu diễn được do **độ sâu bit** (bit depth) quyết định, ký hiệu N.

$$M = 2^N$$

- **16-bit:** 2^{16} = **65.536 mức** — chuẩn đĩa CD và xuất bản nhạc số thương mại.
- **24-bit:** 2^{24} = **16.777.216 mức** — hơn 16 triệu mức, chuẩn thu âm và hậu kỳ chuyên nghiệp.
- **32-bit float:** dùng dấu phẩy động với 24-bit phần định trị — dải động vượt 1.500 dB, coi như vô hạn. Dùng cho xử lý nội bộ trong DAW.

- Càng nhiều mức, bước lượng tử càng nhỏ, mẫu được làm tròn càng sát giá trị analog thật.

Quan hệ tuyến tính giữa độ sâu bit và dải động

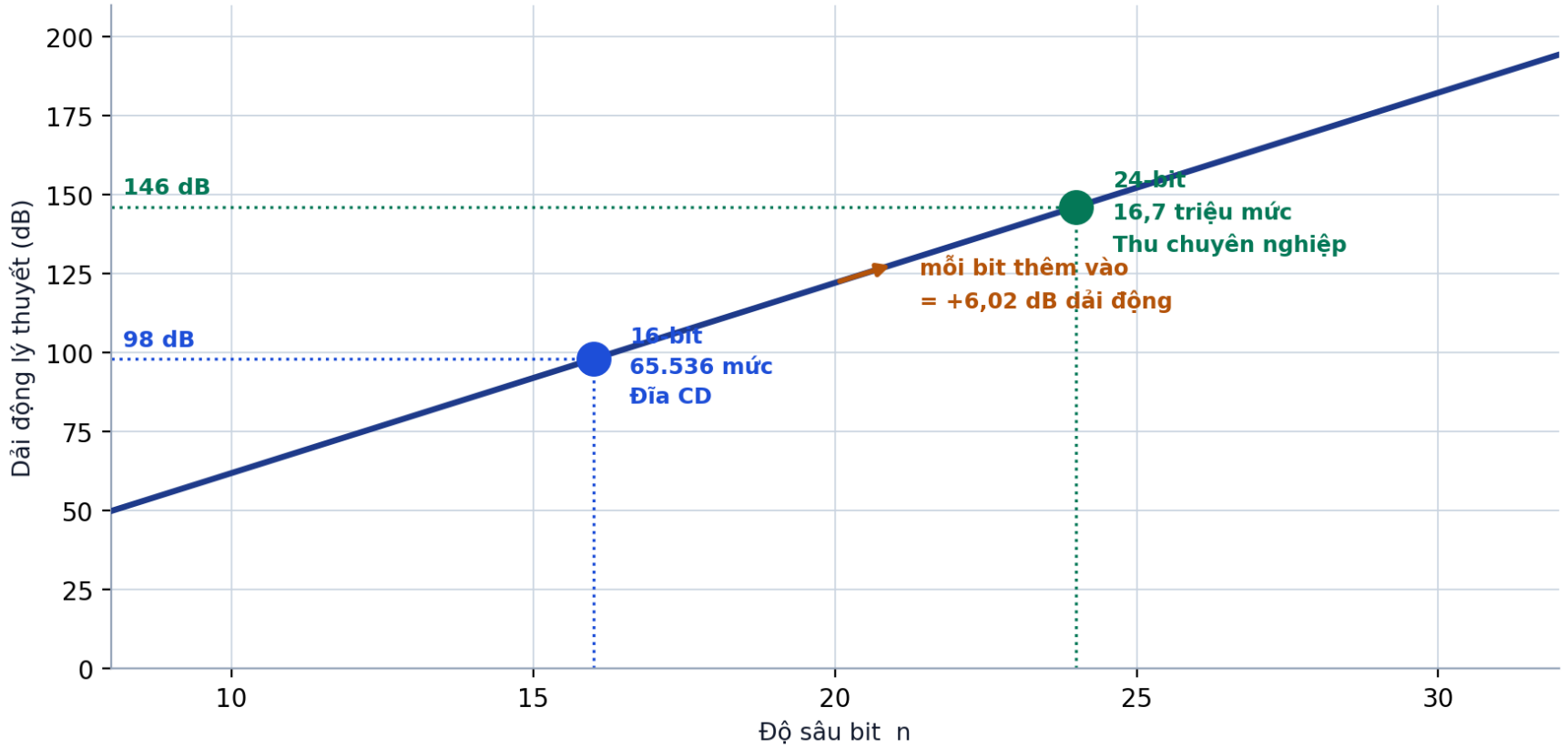
Mỗi khi tăng thêm **1 bit**, số mức tăng gấp đôi, bước lượng tử chia đôi, biên độ nhiễu lượng tử giảm một nửa — tức giảm đúng **6 dB**.

$$\mathrm{DR} \approx 6{,}02\,N + 1{,}76 \quad (\mathrm{dB})$$

- **16-bit:** DR ≈ 6,02 × 16 + 1,76 ≈ **98 dB** (thường làm tròn nói là ~96 dB).
- **24-bit:** DR ≈ 6,02 × 24 + 1,76 ≈ **146 dB** (thường nói ~144 dB).
- Quan hệ này là **tuyến tính**: cứ mỗi bit là một bậc thang +6,02 dB, không hơn không kém.
- Đây là công thức phải thuộc — bài tập 2 của chương hỏi trực tiếp, và nó cũng là căn cứ để chọn định dạng khi lập trình.

BẢNG 3.3 → BIỂU ĐỒ

Đường thẳng 6,02 dB mỗi bit



Quan hệ tuyến tính hoàn hảo. Mỗi bit thêm vào đẩy trần dải động lên đúng một bậc 6,02 dB.

BẢNG 3.3

Độ sâu bit và dải động kỹ thuật

Độ sâu bit	Số mức lượng tử	Dải động lý thuyết	Phạm vi ứng dụng
16-bit	65.536	~96 dB	Đĩa CD, xuất bản nhạc số thương mại phổ thông
24-bit	16.777.216	~144 dB	Thu âm chuyên nghiệp, hậu kỳ âm thanh phòng thu
32-bit float	Dấu phẩy động (24-bit phần định trị)	trên 1.500 dB (coi như vô cực)	Xử lý nội bộ trong DAW, thu âm dải rộng

*Dải động 96 dB của 16-bit đủ phủ hầu hết thể loại nhạc phổ thông. Nhưng trong **sản xuất**, 144 dB của 24-bit là bắt buộc — nó cho kỹ sư khoảng headroom rộng rãi để căn Gain mà không sợ clip.*

Bài tập 2 — chọn độ sâu bit theo dải động phần cứng

Tình huống: (a) Tính dải động lý thuyết của định dạng **20-bit**. (b) Một hệ thống phần cứng thu âm có dải động vật lý (sàn nhiễu nội tại mạch điện) là **100 dB**. Kỹ sư nên chọn **16-bit hay 24-bit** cho phần mềm thu âm? Giải thích nguyên lý.

1. (a) Áp dụng công thức: DR ≈ 6,02 × n = 6,02 × 20 = **120,4 dB**.
2. (b) Nếu chọn **16-bit**: dải động chỉ ~96 dB. Mà 96 dB **< 100 dB** của phần cứng.
3. Hệ quả: định dạng số trở thành **nút thắt cổ chai**, bóp hẹp dải động thực của thiết bị — mua phần cứng tốt mà không dùng hết.
4. Nếu chọn **24-bit**: dải động ~144 dB. Mà 144 dB **> 100 dB** của phần cứng.
5. Hệ quả: định dạng số **bao trùm trọn vẹn** sàn nhiễu vật lý, bảo toàn 100% chất lượng phần cứng mà không lãng phí vô ích.

Kết luận nghề: Chọn **24-bit**. Nguyên lý tổng quát: độ sâu bit của định dạng số phải **lớn hơn** dải động của phần cứng, để phần mềm không trở thành khâu yếu nhất trong chuỗi.

CÂU HỎI TẠI LỚP

Chuyển bản master 24-bit về 16-bit để ghi đĩa CD. Chuyện gì xảy ra với những đoạn âm thanh rất nhỏ — đuôi âm vang, tiếng thì thầm?

Gợi ý: những giá trị biên độ rất nhỏ chỉ dùng tới **vài mức lượng tử** cuối cùng. Sai số làm tròn lúc đó còn **ngẫu nhiên** nữa không?

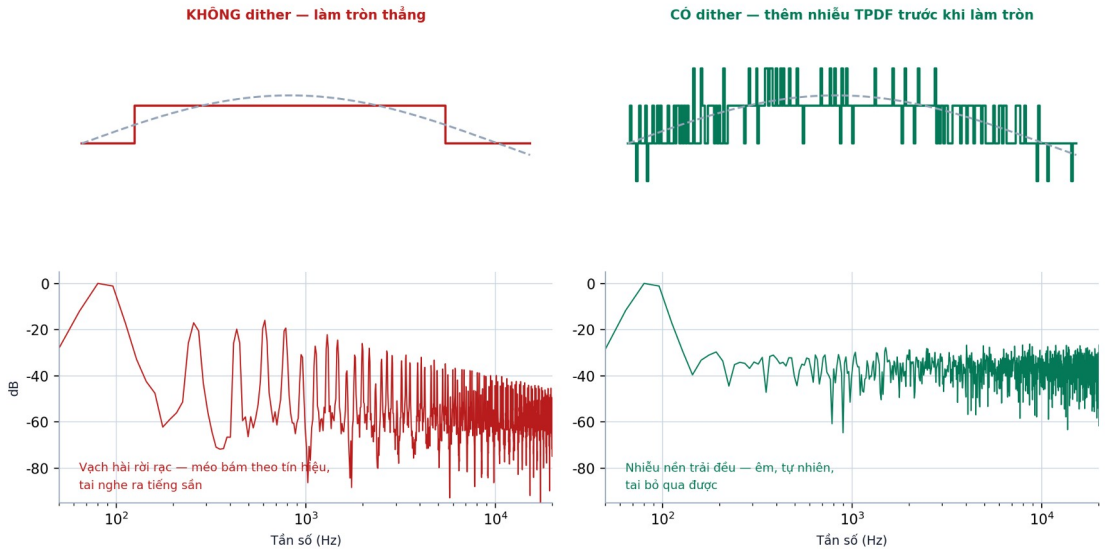
Suy nghĩ 30 giây — rồi trả lời

GIẢI THÍCH

Đáp án — sai số hết ngẫu nhiên, biến thành méo hài

Với tín hiệu rất nhỏ, sai số lượng tử không còn ngẫu nhiên mà mang tính chu kỳ bám theo tín hiệu gốc — sinh méo hài, tai nghe ra tiếng sần rách rất khó chịu.

- Sai số **tương quan** với tín hiệu ⇒ sinh ra các **vạch hài rời rạc** trong phổ (bên trái).
- Tai người rất nhạy với vạch hài — nghe ra ngay như tiếng sần, tiếng méo.
- Ngược lại, nhiễu **trải đều** trên phổ (bên phải) thì tai bỏ qua được dễ dàng.
- Nghịch lý sư phạm: cách chữa là **chủ động thêm nhiễu vào** — kỹ thuật dither.



MỤC 3.2.3

Kỹ thuật Dither

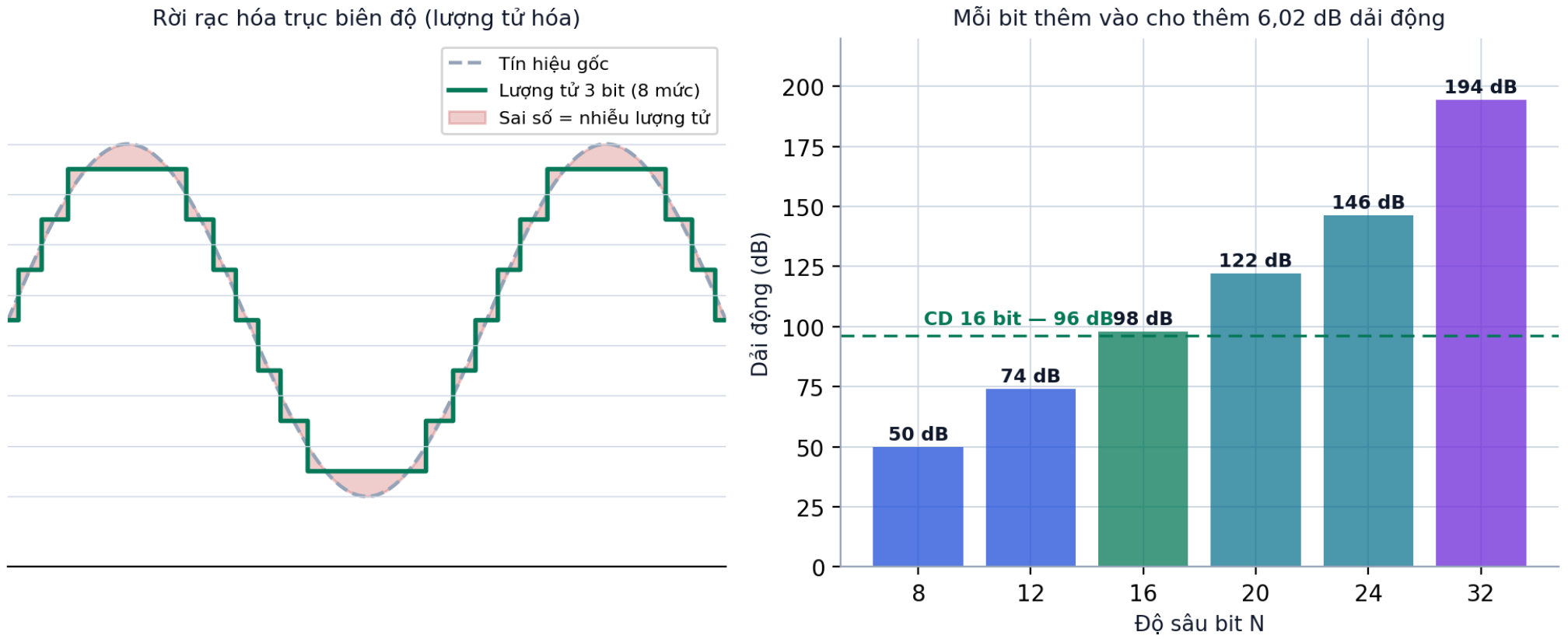
Dither là việc chủ động trộn thêm một lượng nhiễu ngẫu nhiên cực nhỏ vào tín hiệu **ngay trước** khi thực hiện phép làm tròn hạ bit.

- Loại nhiễu chuẩn: **nhiễu trắng dạng tam giác TPDF**, biên độ bằng đúng **1/2 bước lượng tử** — rất nhỏ.
- Tác dụng: **ngẫu nhiên hóa sai số lượng tử**, cắt đứt mối tương quan giữa sai số và tín hiệu gốc.
- Kết quả: méo hài có hại biến thành một lớp **nhiễu nền rào rạt đều đặn, tự nhiên, êm dịu** — tai chấp nhận được.

- Đây là **bước bắt buộc** trong khâu xuất bản (mastering) cuối cùng của mọi chuỗi sản xuất đa phương tiện. Bỏ qua dither khi hạ bit là lỗi nghiệp vụ.

HÌNH 3.3

Lượng tử hóa và quan hệ với dải động



Hình 3.3 — Lượng tử hóa trục biên độ, sai số lượng tử và quan hệ tuyến tính giữa độ sâu bit với dải động.

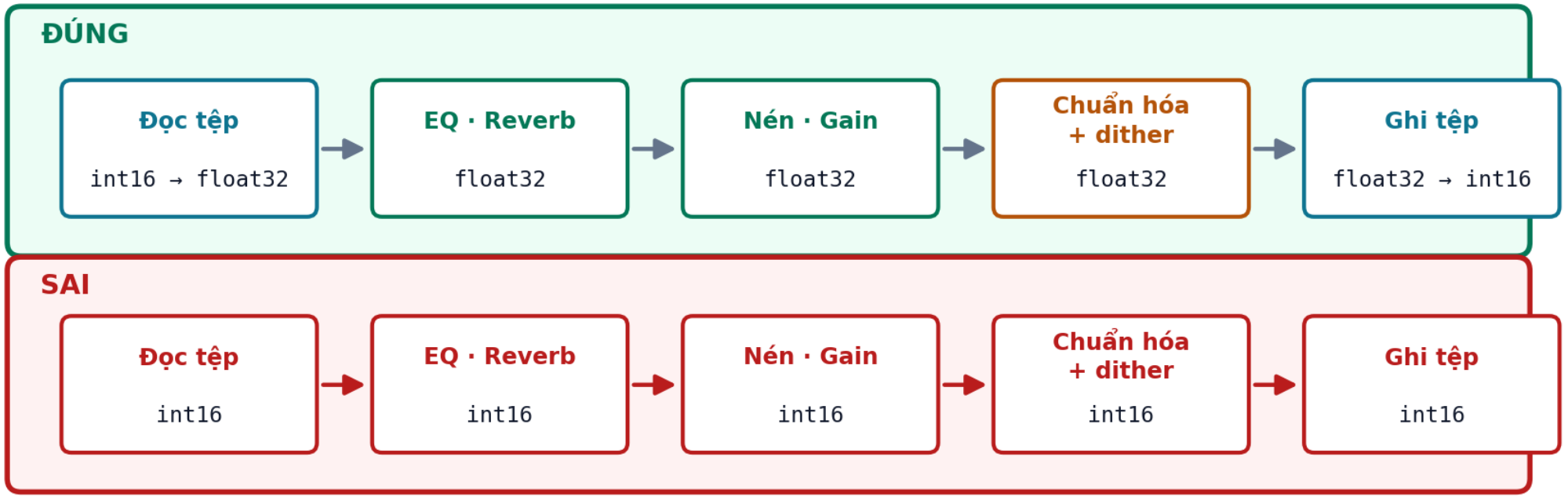
Lượng tử hóa chính là phép chuyển đổi float ↔ int

Trong máy tính, mọi thuật toán xử lý âm thanh đều chạy trên số thực **float32** trong dải [−1,0 … +1,0] để bảo toàn độ chính xác và tránh tràn số giữa chừng.

- Lượng tử hóa thực chất là **phép nhân tỷ lệ rồi ép kiểu** từ float32 sang số nguyên int16 khi xuất tệp.
- `xi = (np.clip(xf, -1.0, 1.0) * 32767.0).astype(np.int16)` — float32 → int16.
- `xf = xi.astype(np.float32) / 32767.0` — đọc int16 về float32 để xử lý.
- Quy tắc bất di bất dịch: **chỉ lượng tử hóa hạ bit ở dòng lệnh ghi tệp cuối cùng**. Toàn bộ tính toán trung gian bắt buộc chạy trên float32.

QUY TẮC VÀNG

float32 xuyên suốt, chỉ hạ int ở bước cuối



Tính trực tiếp trên int16 xuyên suốt: sai số làm tròn tích lũy ở MỌI bước

Tính trực tiếp trên int16 xuyên suốt là sai lầm nghiêm trọng — sai số làm tròn tích lũy ở mọi bước.

CHỐT LẠI MỤC 3.2

Chốt lại: lượng tử hóa và dither

Lượng tử hóa = làm tròn biên độ về một trong **M = 2^N** mức cho trước.

16-bit ≈ 96 dB (nghe); **24-bit ≈ 144 dB** (làm); **32-bit float** ≈ vô hạn (xử lý nội bộ).

Nhiễu lượng tử = chênh lệch giữa giá trị thật và giá trị đã làm tròn — tồn tại vĩnh viễn.

Dither = thêm nhiễu TPDF nửa bước lượng tử trước khi hạ bit, biến méo hài thành nhiễu nền êm.

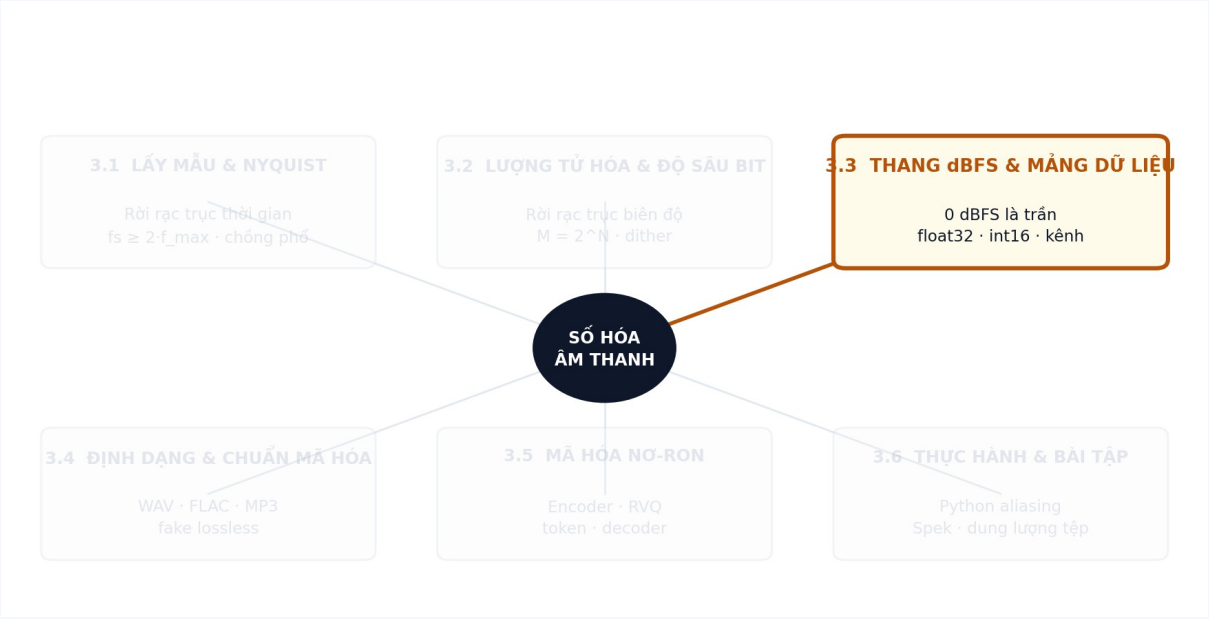
DR ≈ 6,02·N + 1,76 dB — mỗi bit thêm vào cho đúng một bậc 6 dB.

Quy tắc vàng: **float32 xuyên suốt, chỉ ép int ở dòng ghi tệp cuối cùng**.

TIẾT 4

Thang dBFS, mảng dữ liệu và định dạng tệp

- dBFS: 0 dBFS là trần, mọi mức hợp lệ đều âm
- Kiểu dữ liệu int / float và cấu trúc số kênh
- Ba nhóm định dạng: không nén · lossless · lossy
- Nhận diện tệp giả danh Fake Lossless

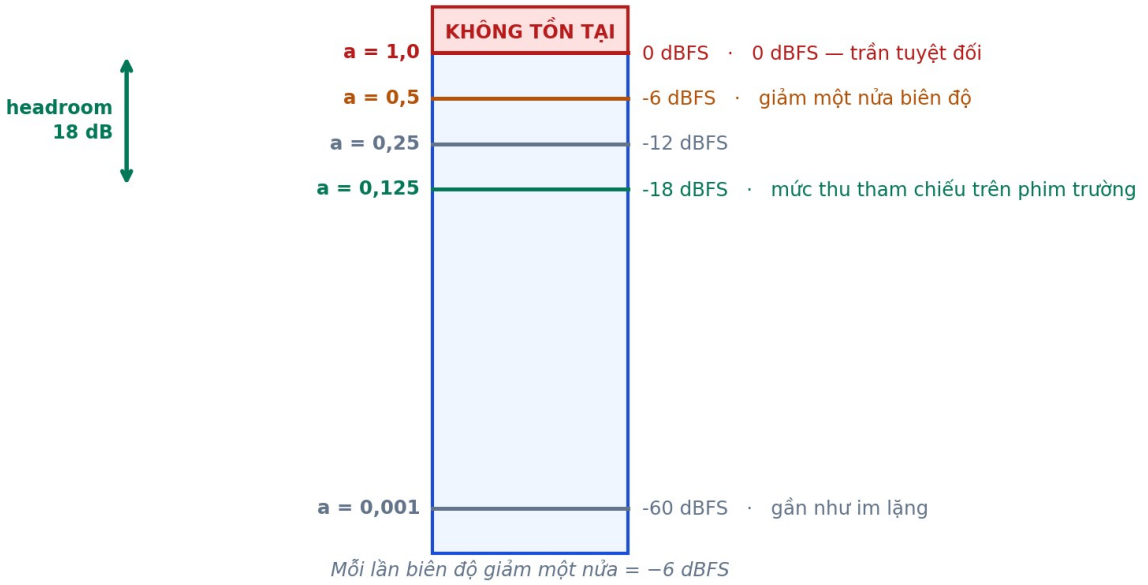


MỤC 3.3.1

Thang decibel toàn dải — dBFS

Khác với **dB SPL** đo áp suất âm ngoài đời (Chương 1), thang **dBFS** đo mức tín hiệu trong máy tính và lấy mốc theo cách ngược lại.

- Mức cực đại hệ thống số biểu diễn được — biên độ tuyệt đối **|1,0|** — quy ước là **0 dBFS**.
- Do đó **mọi mức hợp lệ đều mang giá trị âm**. Không có gì lớn hơn 0 dBFS: vượt là clip.
- Mỗi lần biên độ **giảm một nửa** thì mức giảm đúng **−6 dBFS**.
- Mức **−18 dBFS** (hoặc −20 dBFS) là mức thu tham chiếu an toàn trên phim trường: chừa 18 dB headroom chống méo cắt đỉnh khi diễn viên đột ngột hét.



a = 1,0 → 0 dBFS · a = 0,5 → −6 dBFS · a = 0,125 → −18 dBFS.

CÔNG THỨC (3.5)

Quy đổi biên độ số thực sang dBFS

Công thức hai chiều giữa biên độ số thực |x| và giá trị decibel toàn dải.

$$L_{\text{dBFS}} = 20\log_{10}\frac{|x|}{x_{\max}} \quad (\text{dBFS})$$

- a = 1,0 → **0 dBFS** (trần tuyệt đối) · a = 0,5 → **−6 dBFS**
- a = 0,25 → **−12 dBFS** · a = 0,125 → **−18 dBFS**
- Vẫn là hệ số **20** vì đây là tỷ số biên độ, không phải tỷ số công suất — giống hệt SNR ở Chương 2.
- Trong Python: `dbfs = 20 * np.log10(np.max(np.abs(x)))` — chính là dòng lệnh trong `audio_info.py` đã viết ở Chương 2.

MỤC 3.3.2

Mảng dữ liệu trong bộ nhớ — hai đặc tính cốt lõi

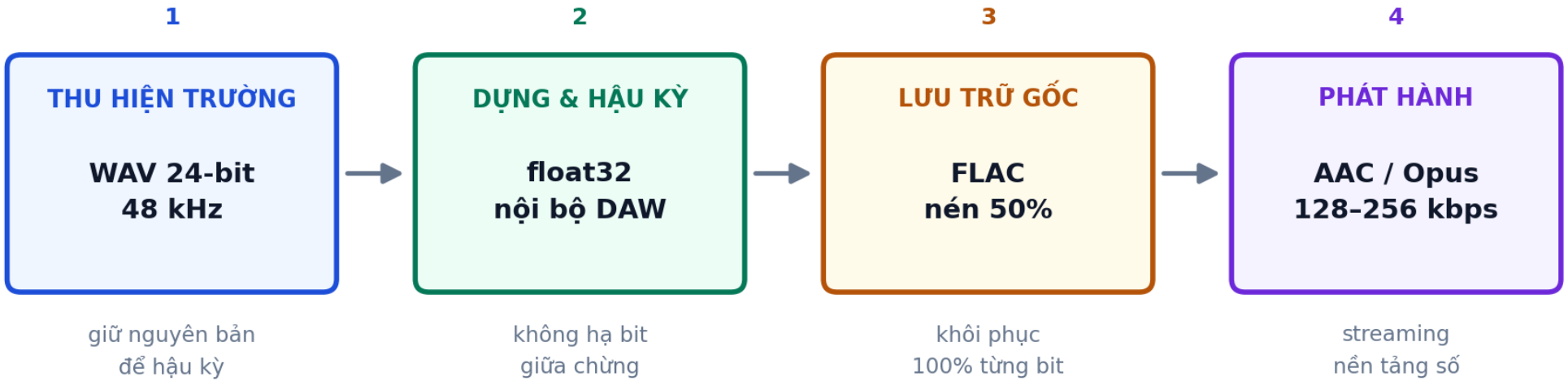
Dữ liệu âm thanh số khi nạp vào bộ nhớ được tổ chức theo **kiểu dữ liệu phần tử** và **cấu trúc số kênh**.

- **Số nguyên (int16, int24):** mỗi mẫu là số nguyên trong dải của kiểu (int16: −32.768 … +32.767). Gọn nhẹ, tối ưu khi **ghi tệp lưu trữ hoặc truyền dẫn**.
- **Số thực (float32, float64):** mỗi mẫu là số thực trong dải [−1,0 … +1,0]. **Bắt buộc khi viết thuật toán DSP** — loại bỏ méo do sai số tích lũy và cho phép vượt ±1,0 tạm thời ở tính toán trung gian mà chưa clip ngay.
- **Mono:** mảng NumPy **một chiều** gồm N phần tử.
- **Stereo:** mảng **hai chiều** kích thước [N mẫu × 2 kênh]. Quy ước quốc tế: **cột 0 là kênh Trái, cột 1 là kênh Phải**.
- **Đa kênh (5.1, 7.1, Dolby Atmos):** mảng [N × K] với các cột cho Center, Subwoofer, Surround L/R…

QUY TẮC VÀNG

Ba bước bất biến khi lập trình xử lý âm thanh

Quy tắc vàng: chỉ hạ bit và nén có tổn hao ở BƯỚC CUỐI CÙNG



Nén lossy ở giữa dây chuyền = mất dữ liệu vĩnh viễn, mọi khâu sau đều làm việc trên bản đã hỏng

Chỉ hạ bit và nén có tổn hao ở bước cuối cùng — nén lossy ở giữa dây chuyền là mất dữ liệu vĩnh viễn.

MỤC 3.4

Ba nhóm định dạng tệp âm thanh

Mảng mẫu sau khi thu nhận được đóng gói thành tệp theo ba nhóm, phân chia dựa trên phương thức nén.

- **1. Không nén — PCM: WAV, AIFF.** Lưu trực tiếp nguyên bản toàn bộ điểm mẫu, không qua thuật toán nén nào. Chất lượng nguyên bản tuyệt đối, đọc ghi đơn giản, không tốn CPU giải nén. Nhược điểm: **dung lượng khổng lồ**.

- **2. Nén không tổn hao — FLAC, ALAC.** Dùng dự đoán tuyến tính LPC tìm quy luật giữa các mẫu, chỉ lưu sai số thặng dư rồi nén entropy. Giảm **40–60%** dung lượng, giải nén khôi phục **chính xác từng bit** — chất lượng bảo toàn 100%.

- **3. Nén có tổn hao — MP3, AAC, Opus.** Giảm tới hơn **90%** dung lượng bằng cách **loại bỏ vĩnh viễn** những thành phần tai người không cảm nhận được.

MỤC 3.4.3

Nén có tổn hao dựa trên mô hình tâm lý thính giác

Codec lossy không cắt bừa. Nó phân tích phổ Fourier rồi áp dụng đúng những quy luật sinh lý thính giác mà ta đã học ở **Chương 1**.

- Áp dụng **đường cong đẳng âm Fletcher–Munson** (ISO 226) và hiện tượng **che lấp âm (Auditory Masking)**.
- Thành phần nào bị **che khuất bởi âm to lân cận** thì bị loại bỏ hoàn toàn — không phân bổ bit để mã hóa.
- **MP3** (1993, MPEG-1 Layer 3): phổ biến lâu đời nhất. Nhược điểm: ở bitrate thấp méo rõ ở dải cao.
- **AAC**: hiệu quả vượt trội MP3, hỗ trợ tới 48 kênh. Mặc định trên iTunes và các nền tảng streaming hiện đại.
- **Opus** (IETF, 2012): hiện đại nhất, kết hợp SILK (nén giọng nói) và CELT (nén nhạc), độ trễ cực thấp, hiệu quả tốt ở mọi dải bitrate.

BẢNG 3.4

Tổng hợp đặc tính ba nhóm định dạng

Nhóm định dạng	Định dạng tiêu biểu	Tổn hao?	Cơ chế hoạt động cốt lõi
Không nén	WAV, AIFF	Không	Lưu trực tiếp mảng số PCM thô, dung lượng cực lớn
Nén không tổn hao	FLAC, ALAC	Không	Nén entropy toán học, giải nén khôi phục 100% bit gốc
Nén có tổn hao	MP3, AAC, Opus	Có	Loại bỏ dữ liệu theo mô hình che lấp thính giác

*Trong sản xuất phim: **thu và hậu kỳ dùng WAV**, **lưu trữ dùng FLAC**, **phát hành dùng AAC hoặc Opus**. Không bao giờ đưa file lossy vào giữa dây chuyền.*

VÍ DỤ GIẢI TỪNG BƯỚC

Bài tập 1 — tính dung lượng tệp WAV và so sánh các định dạng

Tình huống: Bản thu **stereo (2 kênh)**, fs = **48.000 Hz**, độ sâu **24-bit**, thời lượng **3 phút**. Tính dung lượng tệp WAV theo MB. So sánh nếu nén FLAC (tỷ lệ 50%) và MP3 320 kbps.

1. Quy đổi đơn vị: 24-bit = 24/8 = **3 byte mỗi mẫu**; 3 phút = 3 × 60 = **180 giây**.

2. Công thức (3.6): Dung lượng = fs × (số bit / 8) × số kênh × thời gian.

3. Thay số: 48.000 × 3 × 2 × 180 = **51.840.000 byte**.

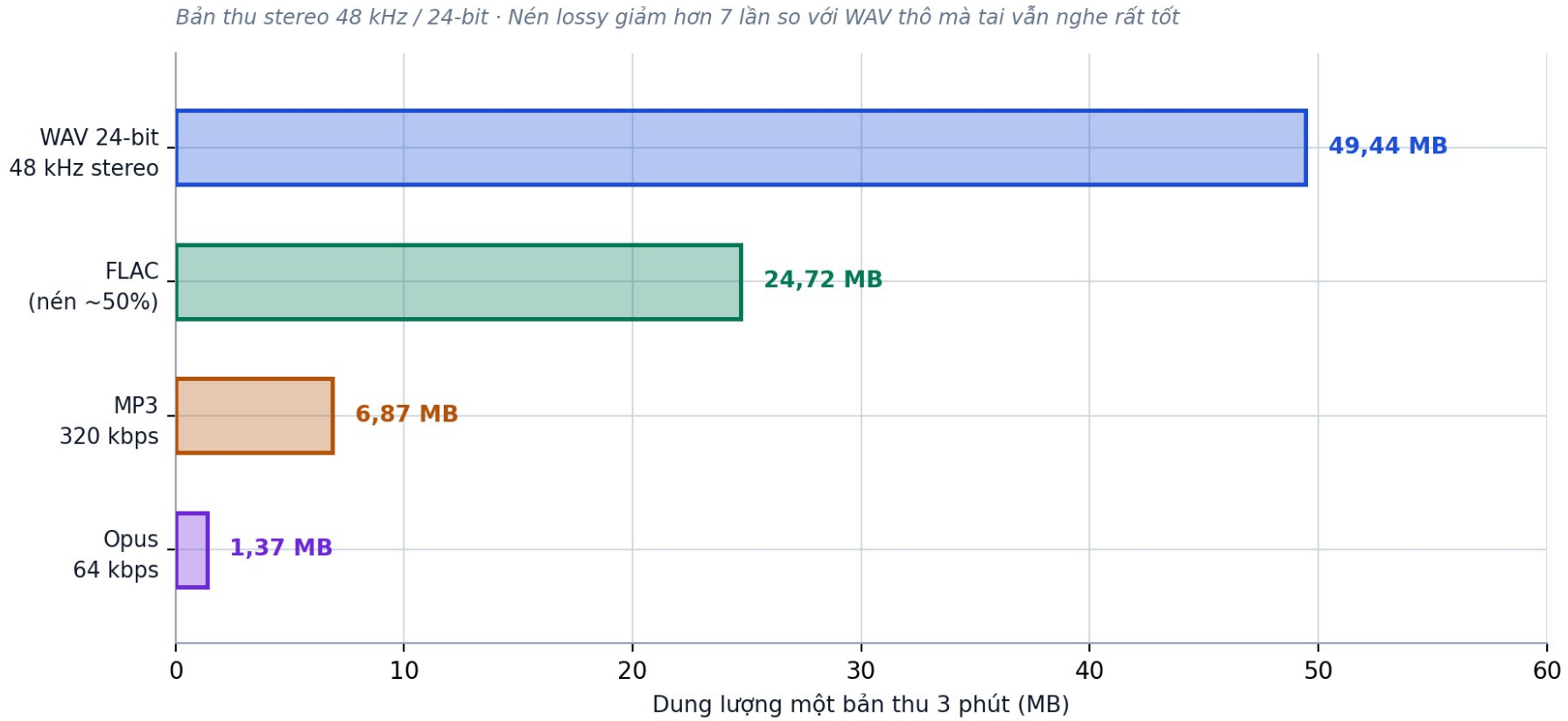
4. Quy đổi sang MB (chia 1.048.576): 51.840.000 / 1.048.576 ≈ **49,44 MB**.

5. FLAC giảm 50% ⇒ còn ≈ **24,7 MB**. MP3 320 kbps: 320.000/8 = 40.000 byte/giây × 180 = 7.200.000 byte ≈ **6,87 MB**.

Kết luận nghề: Nén lossy giảm dung lượng **hơn 7 lần** so với WAV thô mà tai vẫn nghe rất tốt. Nhưng con số đó đánh đổi bằng dữ liệu bị xóa vĩnh viễn — nên chỉ dùng ở khâu phát hành cuối cùng.

TRỰC QUAN HÓA

Dung lượng bốn định dạng cho cùng một bản thu



Cùng một bản thu 3 phút: từ 49,44 MB xuống 1,37 MB — chênh nhau 36 lần.

HÌNH 3.4

Ba nhóm định dạng trong quy trình sản xuất



Hình 3.4 — Ba nhóm định dạng tệp âm thanh và vị trí sử dụng trong quy trình sản xuất.

Một tệp có đuôi .wav hay .flac
hoàn toàn chưa chắc
mang chất lượng nguyên bản.

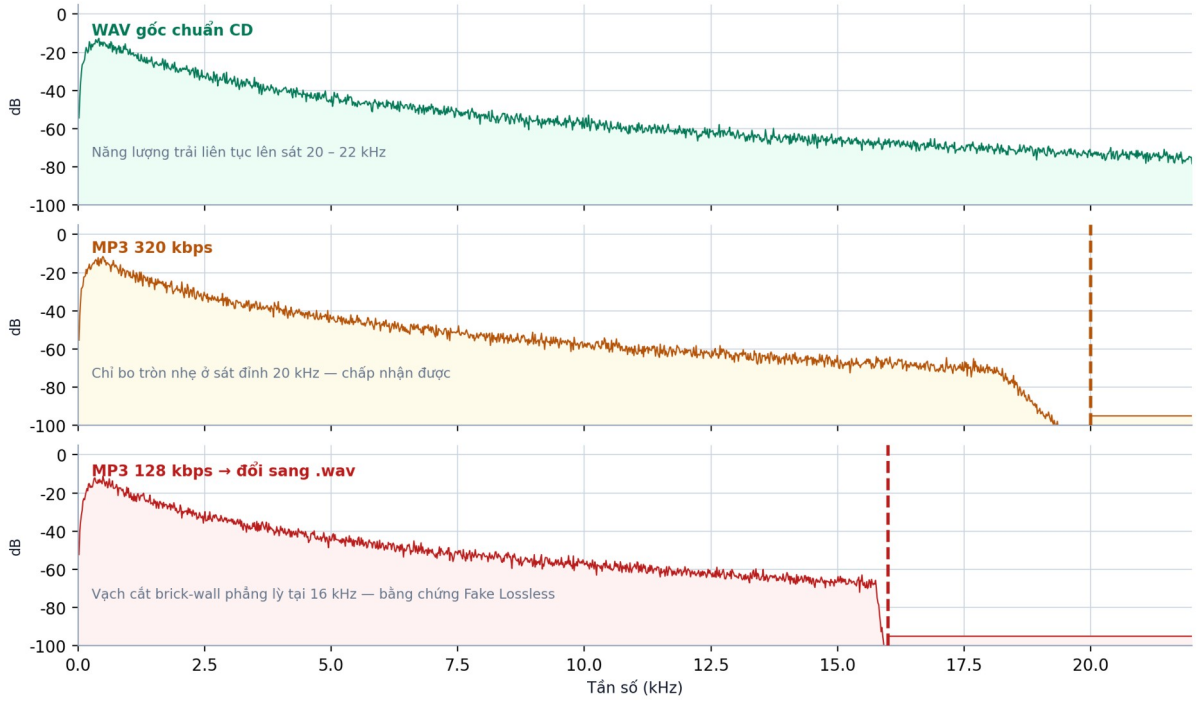
Kỹ năng giám định nguồn âm thanh — dùng được ngay khi nhận file từ đối tác.

MỤC 3.4.4

Fake Lossless — và dấu vết không thể xóa

Nếu một tệp lossy chất lượng thấp (MP3 128 kbps) bị đổi đuôi hoặc **transcode ngược** lên WAV/FLAC, dữ liệu dải cao đã bị codec xóa vĩnh viễn **không thể khôi phục**.

- Sản phẩm thu được là tệp **"Fake Lossless"**: dung lượng phình to gấp 10 lần nhưng chất lượng bên trong vẫn là chất lượng lossy ban đầu.
- Dấu vết nằm ở **vạch cắt phổ** trên đồ thị spectrogram.
- MP3 ở 128 kbps luôn cắt bỏ hoàn toàn dải trên **16 kHz**, tạo một đường ranh giới ngang **phẳng lỳ, sắc nét** — gọi là **brick-wall cutoff**.
- Mở tệp bằng **Spek** hoặc chế độ Spectrogram của **Audacity** là phát hiện ngay. Vạch cắt 16 kHz là bằng chứng không thể chối cãi.



Trên: WAV gốc. Giữa: MP3 320k. Dưới: MP3 128k đổi sang .wav — vạch cắt 16 kHz vẫn nguyên.

BÀI TẬP 4

Quy trình giám định một tệp master_piece.flac

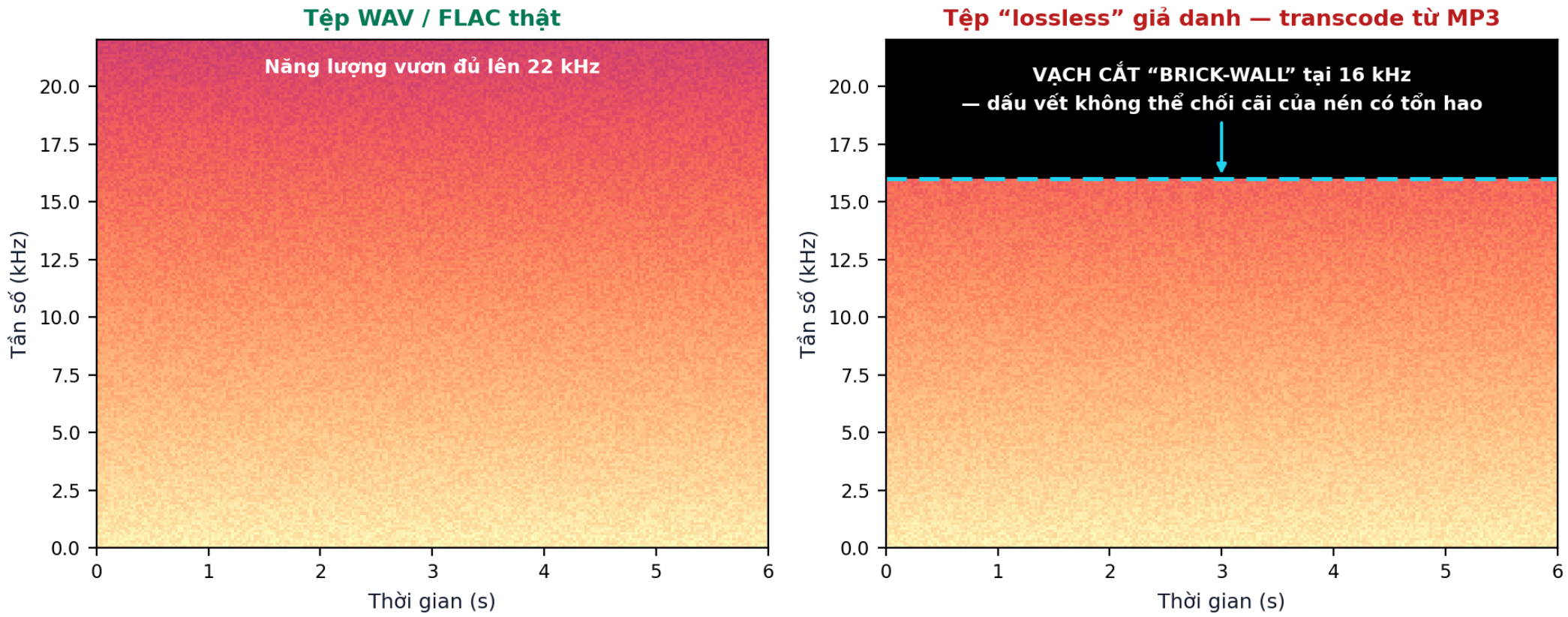
Tình huống: Đối tác gửi tệp `master_piece.flac` làm nhạc nền cho phim, khẳng định là bản chất lượng cao gốc CD. Trình bày quy trình khoa học xác định đây là bản gốc thật hay tệp giả danh transcode từ MP3 128 kbps.

1. Nạp tệp vào phần mềm phân tích phổ **Spek**, hoặc chế độ **Spectrogram** của Audacity.
2. Quan sát dải năng lượng phổ theo trục tần số, từ 0 lên tới 22 kHz.
3. Nếu là **lossless thật**: các vệt năng lượng trải đều, **nhấp nhô tự nhiên**, vươn cao lên tận sát 20 – 22 kHz.
4. Nếu là **Fake Lossless**: xuất hiện đường ranh giới ngang **phẳng lỳ sắc nét tại đúng 16 kHz**, vùng từ 16 kHz đến 22 kHz **tối đen hoàn toàn**.
5. Kết luận: brick-wall cutoff tại 16 kHz là dấu vết không xóa được của codec MP3 128 kbps. Dữ liệu dải cao đã mất và không bao giờ quay lại.

Kết luận nghề: Từ chối đưa tệp vào chuỗi sản xuất phim và yêu cầu đối tác gửi lại bản gốc thật. Đây là kỹ năng nghiệm thu nguồn mà kỹ sư âm thanh phải làm **trước** khi bắt đầu hậu kỳ, không phải sau khi đã dựng xong.

HÌNH 3.5

Nhận diện Fake Lossless bằng phổ đồ



Hình 3.5 — Vạch cắt brick-wall là dấu vết không thể chối cãi của nén có tổn hao.

CHỐT LẠI MỤC 3.4

Chốt lại: dBFS, mảng dữ liệu và định dạng

dBFS: 0 dBFS là trần tuyệt đối, mọi mức hợp lệ đều âm; giảm nửa biên độ = −6 dBFS.

Mono = mảng 1D; **Stereo** = mảng [N × 2], cột 0 là kênh Trái.

−18 dBFS là mức thu tham chiếu an toàn trên phim trường.

Ba nhóm định dạng: **không nén** (WAV) · **lossless** (FLAC) · **lossy** (MP3, AAC, Opus).

int16/int24 để lưu trữ và truyền dẫn; **float32** bắt buộc khi tính toán DSP.

Fake Lossless lộ ở vạch brick-wall 16 kHz trên spectrogram — kiểm tra nguồn trước khi làm.

TIẾT 5

Mã hóa nơ-ron, thực hành và tổng kết chương

- Neural Audio Codec: encoder – RVQ – token – decoder
- Bốn cột mốc: SoundStream · EnCodec · DAC · Mimi
- Ba bài thực hành: aliasing Python · Spek · format_tool
- Tổng kết tám điểm và chuẩn bị cho Chương 4



MỤC 3.5

Vì sao codec truyền thống đã chạm trần

MP3, AAC, Opus đều dựa trên **quy tắc do con người thiết lập sẵn** — một mô hình tâm lý thính giác tĩnh, viết cứng trong thuật toán. Cách tiếp cận đó đã tới giới hạn.

- Từ năm **2021**, hướng đi mới: **Neural Audio Codec** — codec âm thanh mạng nơ-ron.
- Thay vì tuân theo quy tắc người viết, nó **tự học** cách nén và giải nén trực tiếp từ kho dữ liệu khổng lồ bằng mạng nơ-ron sâu.
- Kết quả: chất lượng tự nhiên ở bitrate **dưới 3 kbps** — giảm **10 lần so với Opus** và **100 lần so với WAV thô**.
- Cơ chế: thay các mức lượng tử hóa biên độ cố định bằng một **từ điển token thông minh tự học**.

MỤC 3.5.1

Kiến trúc ba khối của một Neural Audio Codec



Encoder nén thành biểu diễn ẩn → RVQ làm tròn về chỉ số trong từ điển mã → chuỗi token rời rạc → Decoder tái tạo bằng GAN.

CẦU NỐI AI

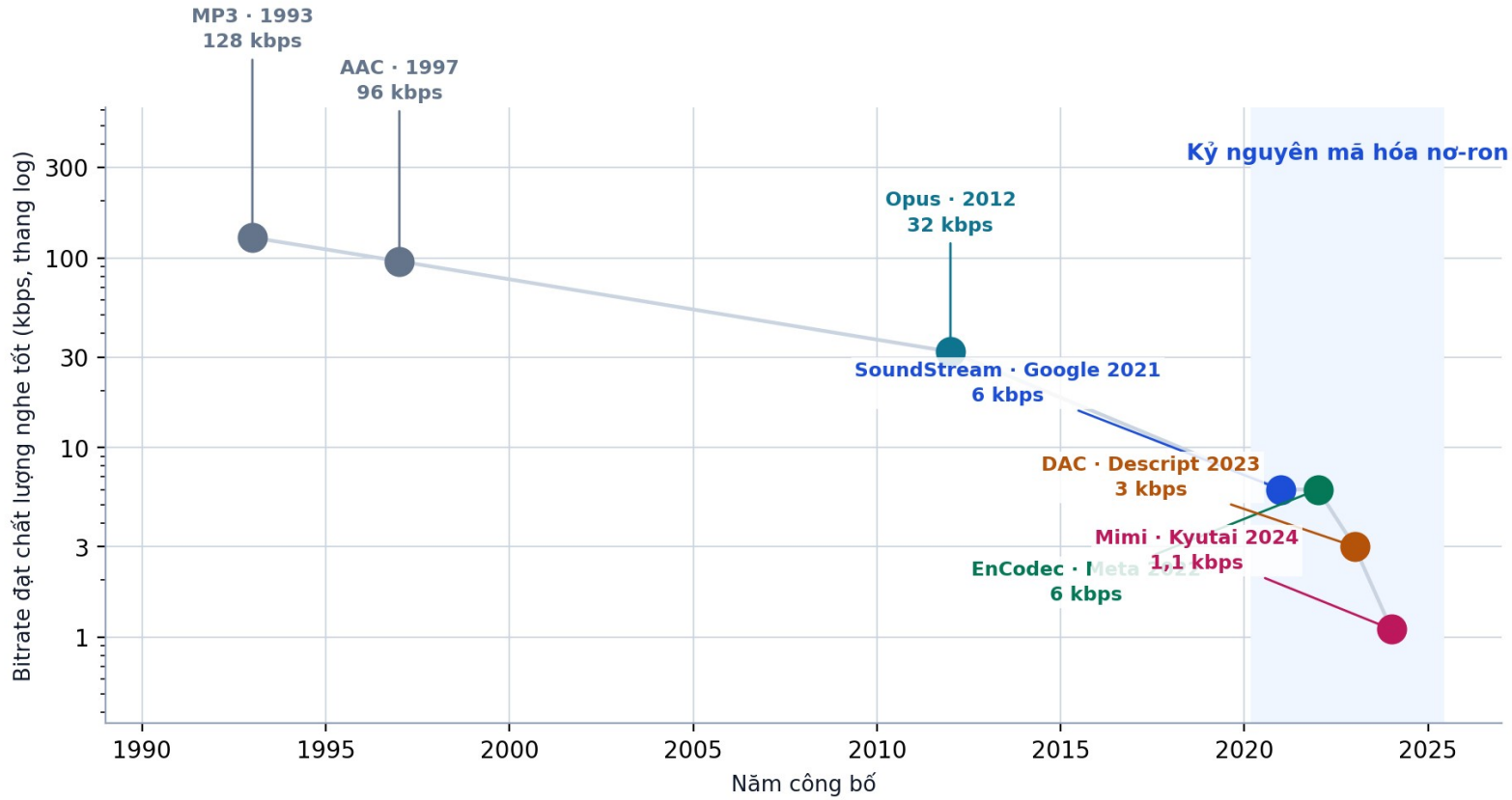
Chuỗi token chính là ngôn ngữ của AI tạo sinh nhạc

Chuỗi token do RVQ sinh ra không chỉ để nén. Điểm kỳ diệu là nó đã biến **sóng âm liên tục** thành các **"từ vựng" số nguyên rời rạc**.

- Nhờ đó, chuỗi token đưa thẳng vào **mô hình ngôn ngữ lớn (Transformer LLM)** được — y như đưa văn bản.
- Khi AI "sáng tác" nhạc (Suno, Udio) hoặc sinh giọng nói tự nhiên (ElevenLabs), thực chất mô hình đang thực hiện **thuật toán dự đoán token kế tiếp** trong từ điển mã.
- Neural Audio Codec là **chiếc cầu nối kỹ thuật duy nhất** liên kết thế giới số hóa cổ điển của chương này với kỷ nguyên âm thanh AI tạo sinh.
- Đây cũng là lý do phải học kỹ lượng tử hóa: không hiểu lượng tử hóa thì không hiểu RVQ, không hiểu RVQ thì không hiểu AI sinh nhạc hoạt động thế nào.

BẢNG 3.5 → BIỂU ĐỒ

Bốn cột mốc và cuộc đua hạ bitrate



Từ 128 kbps của MP3 năm 1993 xuống 1,1 kbps của Mimi năm 2024 — hơn 100 lần trong ba thập kỷ.

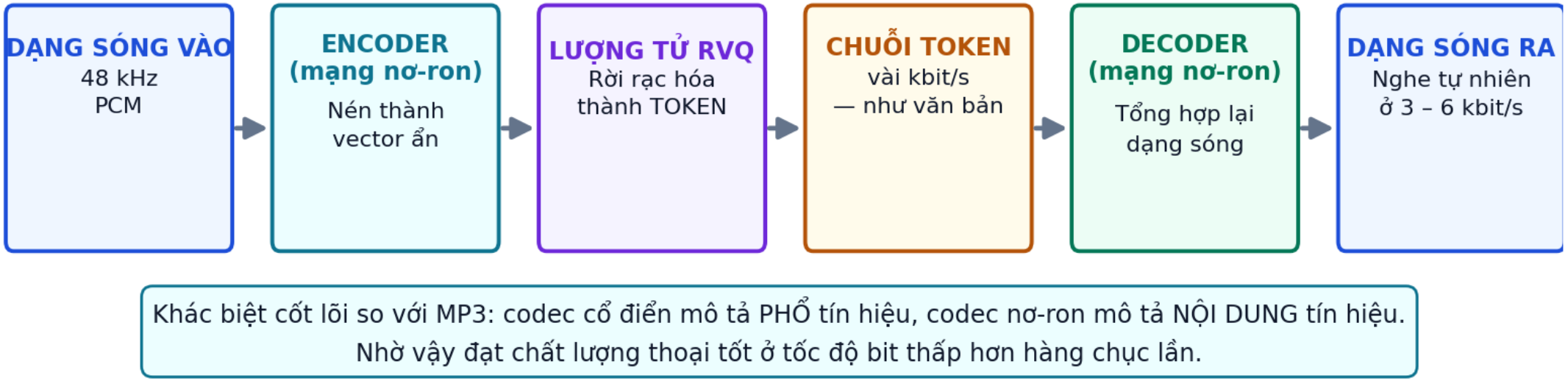
BẢNG 3.5

Các cột mốc Neural Audio Codec

Mô hình · Hãng	Năm	Đóng góp cốt lõi
SoundStream · Google	2021	Đặt nền móng kiến trúc RVQ đầu tiên; nén và giải nén thời gian thực
EnCodec · Meta AI	2022	Mở rộng lên stereo 48 kHz dải rộng, đạt chất lượng âm nhạc gốc
DAC · Descript	2023	Cải thiện tỷ lệ nén vượt bậc, triệt tiêu hoàn toàn nhiễu tạo tác
Mimi · Kyutai	2024	Tối ưu sâu cho mô hình ngôn ngữ âm thanh thời gian thực, độ trễ siêu thấp

HÌNH 3.6

Kiến trúc bộ mã hóa âm thanh nơ-ron



Hình 3.6 — Encoder – lượng tử RVQ – chuỗi token – decoder.

Mục 3.6 — Thực hành trên lớp

Ba bài: tái hiện aliasing bằng Python · nhận diện dấu vết nén bằng Spek · khảo sát và chuyển đổi định dạng

BÀI THỰC HÀNH 3.6.1

Tận tai nghe hiện tượng chồng phổ

Sinh viên tự lập trình để **tận mắt và tận tai** kiểm chứng aliasing. Script `aliasing_demo.py` sinh ra hai tệp WAV ở cùng fs = 8.000 Hz (Nyquist 4.000 Hz).

- **Tệp 1 — `ok_3000.wav`:** sóng sin **3.000 Hz**, nằm dưới ngưỡng Nyquist ⇒ ghi đúng bản chất.
- **Tệp 2 — `alias_5000.wav`:** sóng sin **5.000 Hz**, vượt ngưỡng ⇒ bị dội ngược.
- Nghe thử hai tệp trên tai nghe: **cao độ giống hệt nhau**, dù đề bài nói một cái phải cao hơn hẳn.
- Mở cả hai bằng Audacity ở chế độ **Spectrogram**: cả hai đều chỉ có một vạch, nằm tại **3.000 Hz**.
- Đây là bằng chứng trực tiếp: tệp 5 kHz đã **vĩnh viễn** trở thành 3 kHz. Không có gì để phục hồi.

BÀI THỰC HÀNH 3.6.2

Nhận diện dấu vết nén bằng Spek

Thực hành kỹ năng phân tích phổ để kiểm chứng chất lượng nguồn âm thanh thực tế — bốn bước.

- **Bước 1:** chuẩn bị một file nhạc WAV gốc chuẩn CD (44,1 kHz / 16-bit). Dùng Audacity hoặc FFmpeg xuất ra hai bản: **MP3 320 kbps** và **MP3 128 kbps**.
- **Bước 2:** mở lần lượt ba tệp bằng phần mềm phân tích phổ **Spek**.
- **Bước 3:** so sánh đặc tuyến — WAV gốc trải đều lên 20–22 kHz; MP3 320k chỉ bo tròn nhẹ ở đỉnh; MP3 128k có **vạch cắt phẳng lỳ tại 16 kHz**, phía trên tối đen.
- **Bước 4 (quyết định):** transcode ngược file MP3 128k trở lại WAV. Mở lại bằng Spek — **vạch cắt 16 kHz vẫn còn nguyên** dù đuôi đã là .wav và dung lượng phình gấp 10 lần.

BÀI THỰC HÀNH 3.6.3

Đọc hiểu format_tool.py — khảo sát và chuyển đổi định dạng

Tình huống: Script `format_tool.py` đọc một tệp WAV, in ra toàn bộ tham số số hóa, ước lượng dung lượng lý thuyết, rồi xuất ra ba định dạng khác nhau để so sánh. Nó làm việc đó thế nào?

1. Đọc tệp và trích siêu dữ liệu: `x, fs = sf.read(filename)` và `info = sf.info(filename)` — cho fs, subtype (PCM_16 / PCM_24 / FLOAT), số kênh.
2. Suy ra các đại lượng dẫn xuất: `duration = len(x) / fs` và **tần số Nyquist = fs // 2**.
3. Ước lượng dung lượng theo công thức (3.6): `size = fs *(bits // 8)* num_channels * duration`.
4. Chuẩn hóa về float32 trước khi ghi: `x_float = x.astype(np.float32)` — đúng quy tắc vàng.
5. Xuất ba tệp để so sánh: `subtype='PCM_16'`, `subtype='PCM_24'`, và `.flac` (nén không tổn hao).

Kết luận nghề: So sánh dung lượng thực tế ba tệp trên ổ cứng: PCM_24 lớn hơn PCM_16 đúng **1,5 lần** (24/16), còn FLAC nhỏ hơn khoảng **một nửa** mà chất lượng bảo toàn 100%. Con số thực tế khớp với công thức lý thuyết — đó là mục đích của bài thực hành.

CÂU HỎI TẠI LỚP

Thu ở 24-bit rồi xuất ra CD 16-bit. Trong quy trình, dither phải đặt ở bước nào?

A Ngay sau khi thu, trước mọi xử lý

B Sau mỗi lần chạy plugin để làm mịn dần

C Ngay trước phép làm tròn hạ bit, ở lần xuất cuối cùng

D Sau khi đã ghi ra file 16-bit

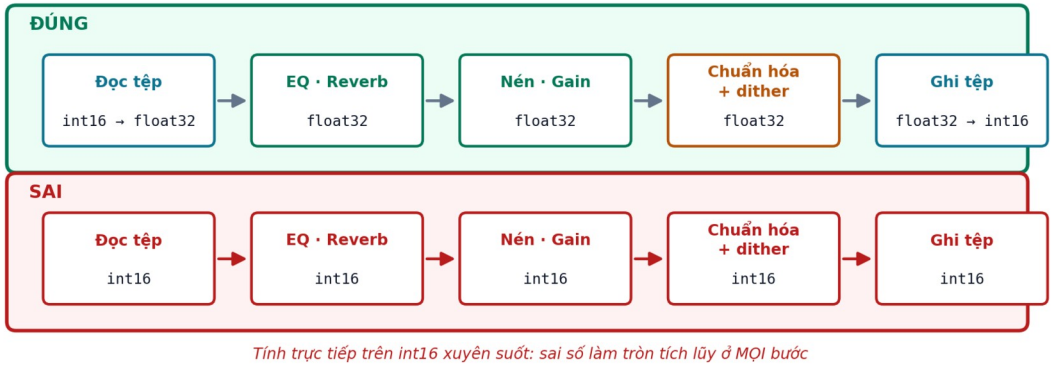
Suy nghĩ 30 giây — rồi trả lời

GIẢI THÍCH

Đáp án C — ngay trước phép làm tròn cuối cùng

Dither chỉ có tác dụng khi nằm ngay trước phép làm tròn hạ bit, và chỉ được áp dụng đúng một lần ở lần xuất cuối cùng.

- **A sai:** thu xong vẫn còn 24-bit, chưa có phép hạ bit nào để mà chống méo hài.
- **B sai nghiêm trọng:** dither nhiều lần thì các lớp nhiễu **cộng dồn** lên nhau, sàn nhiễu bị đẩy cao vô ích.
- **D vô nghĩa:** làm tròn đã xảy ra rồi, méo hài đã sinh ra rồi — thêm nhiễu lúc này chỉ tổ bẩn tín hiệu.
- **C đúng:** đây chính là bước 3 trong quy tắc vàng — chuẩn hóa, dither, rồi mới ép kiểu về int ở dòng ghi tệp.



Tính trực tiếp trên int16 xuyên suốt: sai số làm tròn tích lũy ở MỌI bước

CHỐT LẠI MỤC 3.6

Tóm tắt Chương 3 — tám điểm cốt lõi

1. Số hóa = hai phép rời rạc hóa độc lập: **lấy mẫu** (trục thời gian) và **lượng tử hóa** (trục biên độ).

2. Nyquist–Shannon: fs ≥ 2·f_max. Vi phạm thì aliasing dội tần số cao về tần số thấp giả mạo.

3. Bộ lọc chống răng cưa là lọc thông thấp **analog** đặt **trước ADC** — lấy mẫu rồi thì không sửa được nữa.

4. Độ sâu bit N quyết định **2^N** mức và dải động ≈ **6,02·N dB**. 16-bit ≈ 96 dB; 24-bit ≈ 144 dB.

5. Dither: thêm nhiễu ngẫu nhiên cực nhỏ trước khi hạ bit, biến méo hài có hại thành nhiễu nền dễ chịu.

6. dBFS: 0 dBFS là trần. Quy tắc vàng — xử lý trên **float32** [−1, 1], chỉ hạ **int** ở khâu ghi tệp cuối.

7. Ba nhóm định dạng: không nén PCM · lossless FLAC/ALAC · lossy MP3/AAC/Opus theo mô hình che lấp. **Fake lossless** lộ ở vạch brick-wall.

8. Neural Audio Codec: nén sóng thành **chuỗi token** qua từ điển tự học — cầu nối bước vào kỷ nguyên âm thanh AI tạo sinh.

CHUẨN BỊ

Chương 4 — Cơ sở xử lý tín hiệu số

Chương 3 cho ta **dãy số**. Chương 4 dạy cách **nhìn dãy số đó theo một chiều khác** — miền tần số — và các công cụ toán học để thao tác trên nó.

- **Biến đổi Fourier và thuật toán FFT** — chuyển dãy số từ miền thời gian sang miền tần số.
- **Phổ đồ STFT** — nhìn tần số biến thiên theo thời gian; chính là công cụ Spectrogram vừa dùng để bắt Fake Lossless.
- **Bốn loại bộ lọc số** và **phép tích chập** — nền tảng của công nghệ vang tích chập trong hậu kỳ điện ảnh.
- Nhiệm vụ chuẩn bị: chạy thành công cả `aliasing_demo.py` và `format_tool.py`; cài **Spek**; ôn lại khái niệm sóng sin, tần số và pha ở **Chương 1**.

Hết Chương 3.

Chương 4 — Cơ sở xử lý tín hiệu số: miền thời gian và miền tần số, Fourier, FFT, STFT, bộ lọc số và tích chập.
Tài liệu tham khảo đầy đủ ở cuối Chương 3, trang 53–54 giáo trình.